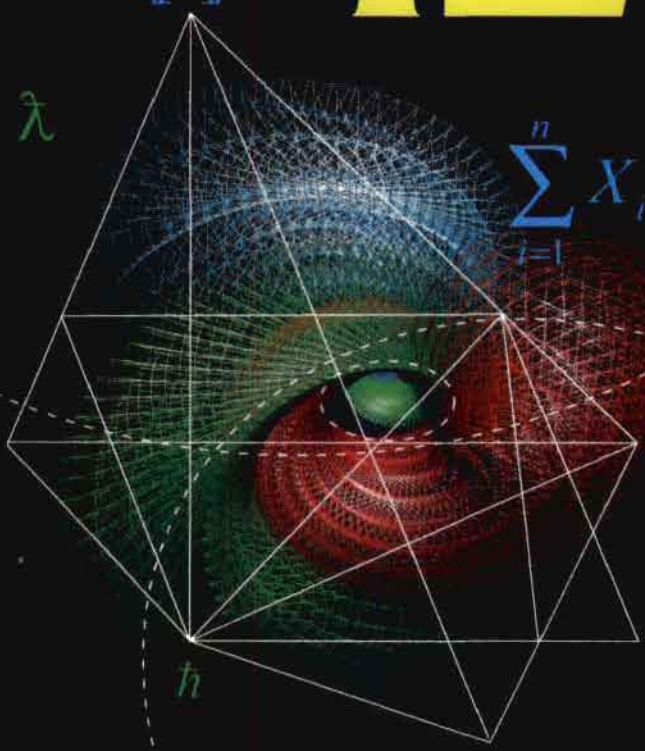
Д. Гурский, Е. Турбина

Вычисления в MATHCAD 12



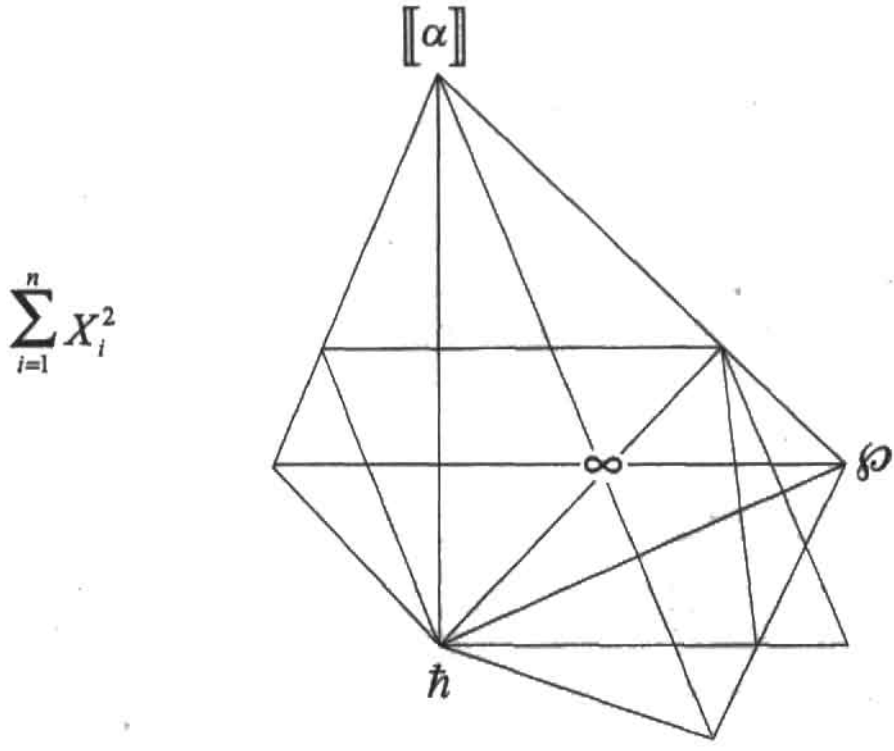
Из этой книги вы узнаете:

- Как эффективно использовать наиболее простой и популярный математический пакет
- Как быстро и легко решить задачи из любого раздела математики
- Как качественно обработать экспериментальные данные или визуализировать их

ПИТЕР

Д. Гурский, Е. Турбина

Вычисления в MATHCAD 12



ПИТЕР®
Москва · Санкт-Петербург · Нижний Новгород · Воронеж
Новосибирск · Ростов-на-Дону · Екатеринбург · Самара
Киев · Харьков · Минск
2006

Дмитрий Гурский, Екатерина Турбина

Вычисления в Mathcad 12

Заведующий редакцией (Минск)	*Ю. Гурский*
Руководитель проекта	*М. Моисеева*
Художник	*А. Татарко*
Корректоры	*Е. Павлович, О. Савицкая*
Верстка	*Г. Блинов*

ББК 32.973.23-018.2
УДК 004.4

Гурский Д. А., Турбина Е. С.

Г95 Вычисления в Mathcad 12. — СПб.: Питер, 2006. — 544 с.: ил.

ISBN 5-469-00639-5

Эта книга поможет вам разобраться в популярнейшем математическом пакете Mathcad 12. Прочитав ее, вы научитесь выбирать оптимальный метод для решения каждой конкретной задачи, а также в автоматическом режиме выполнять разнообразные расчеты — от простейшего вычисления алгебраических функций до создания программ и сложнейших операций с интегралами, рядами и матрицами.

ISBN 5-469-00639-5

ООО «Питер Принт». 194044, Санкт-Петербург, пр. Б. Сампсониевский, 29а.

Лицензия ИД № 05784 от 07.09.01.

Налоговая льгота — общероссийский классификатор продукции ОК 005-93, том 2; 953005 — литература учебная.

Подписано в печать 01.08.05. Формат 70×100$^{1}/_{16}$. Усл. п. л. 43,86. Тираж 3000 экз. Заказ № 2623.

Отпечатано с готовых диапозитивов в ФГУП «Печатный двор» им. А. М. Горького
Федерального агентства по печати и массовым коммуникациям.
197110, Санкт-Петербург, Чкаловский пр., 15.

Краткое содержание

Оглавление

Главы, не вошедшие в книгу

Данные главы в формате PDF можно загрузить с сайта издательства по адресу http://www.piter.com/download/.

Введение

Если попытаться выразить основную концепцию книги, которую вы держите в руках, то наиболее справедливым будет следующее утверждение: «Это издание о том, как сделать жизнь проще».

Действительно, уже давно подмечено, что лень (а, увы, не разум или мифическое стремление к совершенству) двигает прогресс вперед. Нежелание ходить пешком породило велосипед, нелюбовь к стирке дала машины-автоматы, рутина деления в столбик привела в конце концов к появлению калькулятора. А как много этой черной, скучной и неинтересной работы в математике! Думается, немало физиков подалось в лирики, столкнувшись с необходимостью, например, произвести численное решение на бумаге системы дифференциальных уравнений, полученной путем внезапного откровения! Поэтому возникновение чего-то такого, что могло бы, оставив нам радость быть архитекторами своих задач, взять на себя функции замешивания математического бетона и кладки вычислительных кирпичей, было просто неизбежно. Таким чудом, появившимся в 1988 году и действительно сделавшим жизнь уже нескольких миллионов пользователей проще, стала программа Mathcad от компании Mathsoft.

Mathcad может многое. Очень многое. Чтобы понять это, достаточно просто пролистать оглавление этой книги. Больше вам не придется потеть, решая уравнения и неравенства, ломать голову над сложным интегралом или ОДУ, вычислять множество значений функции, строя график, мучаться отсутствием художественных способностей, оформляя отчет. Все это Mathcad сделает за вас, причем (как правило) быстрее, точнее и красивее. Вам останется лишь правильно сформулировать задачу и проверить корректность ее решения.

Главной целью написания этой книги было показать, как, используя Mathcad, можно уничтожить рутину математических расчетов при решении задач и обработке экспериментальных данных. Издание рассчитано прежде всего на студентов естественного и технического профилей (однако она будет полезна и преподавателям, которые решатся на смелый эксперимент — соединить традиционный курс математики с изучением Mathcad). Содержимое книги опирается на усредненную вузовскую программу, поэтому авторы старались не выходить за ее пределы, дабы неоправданно не усложнить материал. Исключение представляют численные методы, которые обычно очень сжато излагаются в курсе высшей математики.

Вообще же, программирование занимает в данной книге ключевое место. И это абсолютно оправданно. В Mathcad сотни встроенных функций. Однако они покрывают лишь наиболее важные задачи математики. Алгоритмы же для решения более специальных математических проблем необходимо создавать самостоятельно.

Большинство решенных в этой книге задач взято из популярных сборников (их список приведен в конце книги), поэтому вы сможете получить практические навыки в реальных «полевых» условиях. Отдельные главы крайне полезны и старшеклассникам — на примерах из популярного задачника М. Сканави они изучат, как в Mathcad можно решать уравнения и системы уравнений. Отдельные главы посвящены неравенствам и алгебраическим преобразованиям.

В чем главное достоинство данной книги, почему ее стоит купить, а не ограничиться чтением одной лишь фирменной справки? Авторы старались написать книгу так, чтобы аналитический подход занимал в ней не меньшее место, чем описательный. Увы, но для эффективной работы в Mathcad мало прочитать справку. Нужен опыт, причем весьма основательный. К примеру, нужно научиться представлять задачу в такой форме, чтобы программе было проще с нею справиться. Нужно научиться действовать правильно тогда, когда Mathcad не сможет решить задачу. Нужно уметь обнаруживать ошибки, не доверяя слепо результату. Нужно уметь выбирать из множества способов решения оптимальный. Обо всем этом нельзя прочитать в фирменной справке, но можно в данной книге. В ней аккумулирован значительный опыт, полученный авторами за довольно продолжительный срок работы с Mathcad. Надеюсь, вам придется провести гораздо меньше вечеров в поисках путей решения очередной «неподдающейся» задачи, чем в свое время понадобилось нам.

Данная книга написана на основе 12 версии Mathcad. Однако ее можно использовать, даже если у вас имеется более ранняя версия программы. Дело в том, что практически все ключевые возможности появились достаточно давно (в Mathcad 8 и ранее). В последних же версиях Mathcad развивался в области «высшего пилотажа», представляющего интерес для очень немногих пользователей.

Если вы найдете ошибку или неточность либо у вас возникнут идеи относительно того, как сделать книгу лучше, то обязательно напишите авторам на адрес diis_ignotis@tut.by. Ваше письмо будет с благодарностью прочитано, а замечания учтены при подготовке следующего издания книги.

Главы книги на сайте издательства

При создании книги обсуждался вопрос о необходимости вкладывания в нее сопроводительного компакт-диска с главами, содержащими дополнительную информацию о работе в Mathcad. В итоге, чтобы не удорожать издание, было решено выпустить книгу без компакт-диска, а соответствующие главы (15–19) выложить на сайт издательского дома «Питер».

Чтобы их скачать, достаточно на сайте http://www.piter.com найти ссылку на книгу «Вычисления в Mathcad 12» и щелкнуть на расположенном рядом с ней значке с изображением дискеты, всплывающая подсказка которого указывает Скачать файлы к книге. С этой же целью можно открыть страницу с аннотацией книги и щелкнуть на ссылке Файлы к книге. После этого вы получите доступ к набору архивов, которые соответствуют главам книги.

От издательства

Ваши замечания, предложения и вопросы отправляйте по адресу электронной почты gurski@minsk.piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

Главы 15–19 в формате PDF можно загрузить с сайта издательства по адресу http://www.piter.com/download/.

На сайте издательства http://www.piter.com вы также найдете подробную информацию о других наших книгах.

Глава 1. Начальные сведения

В данной главе содержатся наиболее общие сведения о Mathcad, такие как настройка эффективного интерфейса пользователя, создание и использование математических формул, формат документов и использование справочной системы, а также некоторые другие. Если вы уже работали в системе Mathcad, то начинайте изучение данной книги с гл. 2 или, что тоже вполне приемлемо, прочитайте лишь интересующие вас разделы. Скорее всего, этого будет вполне достаточно, так как при создании книги преследовалась цель сделать все тематические главы максимально автономными. Да и изучить все возможности Mathcad очень и очень сложно, и, в принципе, делать этого не нужно. Можно смело утверждать, что тот их минимум, который может действительно понадобиться на практике 99 % инженеров, студентов или ученых, вполне возможно изучить и за один вечер. А все остальное будет затем сполна обеспечено вашей интуицией и опытом!

1.1. Что может Mathcad

Исходя из официальной статистики продаж, в нашей стране Mathcad занимает лидирующее положение среди всех остальных математических систем. И для этого есть вполне объективные причины. Перечислим основные, наиболее очевидные достоинства Mathcad, которые позволили ей завоевать такую огромную популярность.

- Высокая универсальность. Обычно математические программы рассчитаны на решение довольно узкого круга задач. Так, главный конкурент Mathcad — система MATLAB — предназначена для проведения численных расчетов, и прежде всего для непосредственного создания пользователем собственных численных алгоритмов. Система Maple, третья по популярности математическая программа, была создана для осуществления всевозможных аналитических преобразований. Помимо перечисленных, существует еще довольно значительное количество более скромных программ, предназначенных, как правило, либо для построения графиков, либо для статистической обработки информации. Кроме того, создано очень много всевозможных формульных редакторов, счет которых уже идет на сотни. В общем, если вы профессиональный математик, то вам пришлось бы иметь дело с добрым десятком разнообразных программ, каждая из которых выполняла бы, более или менее эффективно, свои узкие задачи. Пришлось бы — если бы не было Mathcad. Программа, которой посвящен наш долгий и, надеюсь, интересный разговор, совмещает в себе буквально все, что может понадобиться для упрощения расчетов или красочного оформления результата любому техническому специалисту. Если же вам не верится в это, то просто просмотрите названия глав данной книги. Убежден,

вы найдете здесь все, что может помочь если не сразу получить решение любой возникшей перед вами проблемы, то, по крайней мере, предельно упростить путь к его нахождению!

- Вторым огромным достоинством системы Mathcad является полное соответствие используемых в ней функций и операторов традициям оформления в математике. Если вы когда-нибудь сталкивались с другими математическими системами, то вы знаете, что это совершенно нехарактерное явление. В том же MATLAB интеграл подсчитывается специальной функцией, в скобках которой вы задаете пределы и точность. В Mathcad же интеграл — это привычный математический символ в виде вытянутой буквы *S*. Все дело в том, что создатель Mathcad фирма Mathsoft весьма последовательно, начиная с первой версии программы придерживается известного принципа What You See Is What You Get (Что вы видите, то и получите — WYSIWYG). А это означает, что в том случае, если вам нужно вычислить, например, производную, то для этого совершенно не обязательно читать учебники или справочники. Просто найдите на одной из рабочих панелей знакомый со школы оператор, точно так же, как вы бы это сделали на бумаге, введите в него функцию и переменную. Затем, правда, нужно поставить не «=» (что означает, что расчет будет проведен численно), а «→» (этот оператор показывает системе, что выражение должно быть преобразовано аналитическими методами). Однако, немного поэкспериментировав, вы без проблем нашли бы и этот не совсем очевидный ход.
- К несомненным преимуществам программы Mathcad относится то, что она может быть одинаково успешно использована и профессором математики, и простым школьником. В Mathcad можно как производить такие элементарные с точки зрения техники исполнения операции, как символьное интегрирование или подсчет значения функции, так и создавать свои вычислительные алгоритмы и математические модели с помощью специального, весьма простого и изящного языка программирования. Более того, Mathcad является, по сути, системой программирования без программирования. Что это значит, вы поймете тогда, когда мы приступим к непосредственному проведению вычислений.
- Широкие возможности открывает высокая степень интеграции Mathcad с другими Windows-приложениями. Особенно большое практическое значение имеет возможность сохранения документов в виде HTML-файлов (что позволяет вам выставлять свои решения в Интернете) и в качестве Word-документов. Значительно облегчить обработку данных позволяет отличная совместимость Mathcad и Excel.
- Самой высокой оценки заслуживает тот факт, что все расчеты в Mathcad проводятся в режиме реального времени и не требуют от пользователя никаких дополнительных команд.
- Замечательны возможности визуализации в Mathcad. Особенно поражают элементы оформления трехмерных объектов и создание собственных анимаций. Чтобы в этом убедиться, пролистайте гл. 6.
- Очень помогают процессу обучения работе в Mathcad ее обширные Ресурсы (Resources). Среди них вы можете найти сотни примеров использования Mathcad в различных областях науки и техники, десятки шпаргалок и справочных таблиц. Кроме того, в предлагаемом перечне находится весьма неплохой электронный учебник по Mathcad, требующий, правда, довольно высокого уровня знания английского языка.
- Помимо выполнения своих прямых математических функций, система Mathcad является очень неплохим текстовым и графическим редактором, по многим параметрам не уступающим специализированным программам.

- По сравнению с другими математическими пакетами, Mathcad стоит очень недорого (особенно если покупать одну из предыдущих версий). Поэтому его лицензировать могут даже наши небогатые учебные и научные заведения.

Перечисленные возможности далеко не полностью характеризуют достоинства Mathcad. Можно смело утверждать, что, купив программу и освоив элементы работы в ней, вы никогда об этом не пожалеете. Более того, то чувство восхищения, которое испытывает человек, решавший всю жизнь задачи на бумаге, увидевший, насколько быстро, точно и красиво это можно сделать с помощью Mathcad, очень долго не будет покидать его!

1.2. Новое в Mathcad 12

Если попытаться проанализировать недостатки предыдущих версий Mathcad, то одним из существенных можно назвать отсутствие универсальных средств взаимодействия с другими приложениями. В Mathcad 12 эта проблема решена благодаря тому, что новый собственный формат программы XMCD является открытым и основывается на XML.

XML (Extensible Markup Language — Расширяемый язык разметки) является потомком чрезвычайно обобщенного и сложного языка SGML, который был разработан в начале 1970-х годов. Предназначен SGML был для тех же целей, для которых сейчас применяется XML, — для семантической (проще говоря, смысловой) и структурной разметки текстовых документов. Однако широкого распространения SGML не получил из-за своей сложности. В 1996 году началась работа над упрощенной версией SGML, в которой были сохранены основные идеи языка, однако были урезаны редко используемые и излишне запутанные возможности. В результате в 1998 году появился XML 1.0, сразу получивший широкое признание. В настоящий момент все системы, так или иначе связанные с обменом данными, поддерживают XML.

Созданные в Mathcad файлы сохраняются в основанном на XML текстовом формате XMCD по умолчанию. Это облегчает дальнейшую работу с результатами вычислений с помощью различных приложений, поддерживающих XMCD (таких пока немного, но все еще только начинается). Кроме того, с внедрением архитектуры XML появилась возможность конвертировать Mathcad-файлы в HTML-страницы, а также преобразовывать их в формат PDF, благодаря чему вы с легкостью сможете опубликовать свои работы в Интернете, что сделает их доступными широкому кругу пользователей, или подготовить документы к печатному изданию.

В отличие от предыдущих версий, Mathcad 12 довольно требовательна к машинным ресурсам. Так, для установки Mathcad 12 вам потребуется не менее 150 Мбайт дискового пространства, процессор Pentium/Celeron 300 МГц, минимум 128 Мбайт оперативной памяти (для приемлемой производительности рекомендуется 256 Мбайт), операционная система не ниже Windows 2000.

Рассмотрим некоторые особенности установки Mathcad 12. Для полноценной работы программы необходимы Interner Explorer 6, Microsoft .NET Framework 1.1 и MSXML 4 (их можно найти на установочном диске Mathcad 12 или скачать с сайта www.microsoft.com). IE 6 требуется для доступа к справочной системе и HTML-документам Ресурсов Mathcad (Mathcad Resources), а также для открытия и сохранения HTML-файлов. Среда исполнения прикладных программ .NET Framework 1.1, в расчете на которую написан Mathcad 12, обеспечивает безопасное, в отношении всех системных ресурсов, выполнение кода и стабильную работу приложения. Также перевод Mathcad на платформу .NET гарантирует совместимость со следующими версиями Microsoft Windows.

Не рекомендуется устанавливать сжатую версию программы, так как при этом у вас не будет возможности доступа к полезнейшим приложениям справочной системы и Ресурсам Mathcad.

Итак, если вы уже купили программу — устанавливайте ее, запускайте и начинайте читать следующий раздел.

1.3. Начинаем работу

В данном разделе, несколько заглядывая вперед, рассмотрим некоторые наиболее общие принципы работы в Mathcad. Так, мы разберем, как проводить расчет значений выражения и функции, осуществлять аналитические преобразования и строить графики. Если вам что-то покажется непонятным, не огорчайтесь — в последующих главах мы вернемся ко всем этим вопросам и рассмотрим их обстоятельно. Кроме того, всегда помните, что Mathcad (равно как и вся математика) построена не столько на знаниях, сколько на интуиции. Так что если вам что-то неясно или вы не знаете, как провести ту или иную операцию, — не спешите листать учебник, а просмотрите для начала все рабочие панели и меню, понабирайте с клавиатуры, подумайте, как оно ДОЛЖНО быть. И, поверьте опыту, почти наверняка вы самостоятельно найдете решение возникших проблем. Запомните найденный ход при этом вы гораздо лучше, чем прочитав о нем мельком в книге.

Установив программу, запускайте ее одним из стандартных методов.

Внимательно рассмотрев окно программы (рис. 1.1), вы обнаружите в нем множество как знакомых, так и незнакомых элементов. Например, всегда по умолчанию открыты две панели (Toolbars), характерные для многих Windows-приложений: Formatting (Форматирование), отвечающая за редактирование текста, и Standard (Стандартные), содержащая ссылки на наиболее общие команды. Здесь вы найдете и знакомые по другим приложениям команды (таковых будет большинство) — копирования, вставки, проверки орфографии и прочие, а также кнопки нового для вас вида.

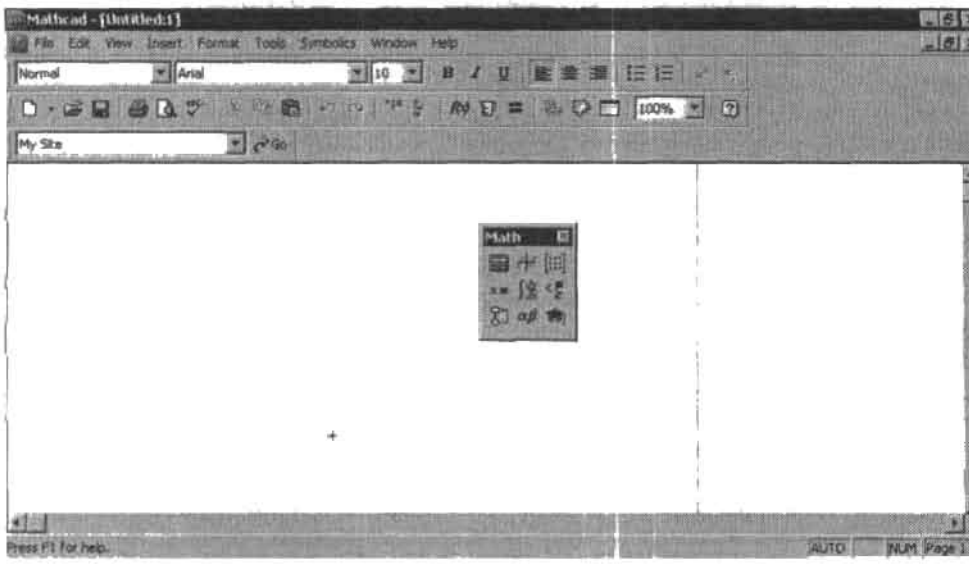


Рис. 1.1. Окно Mathcad при первом запуске

Почти стандартный вид имеет главное меню, назначение практически всех подменю которого для вас должно быть знакомым. Специфическими для Mathcad окажутся только два из них: Tools (Инструменты), содержащее все основные параметры проведения расчетов, и Symbolics (Символьные), содержащее команды быстрых аналитических преобразований. Совершенно стандартными в Mathcad являются и остальные элементы окон Windows-приложений: полосы прокрутки и строка состояния. Единственным же принципиально новым, по сравнению с другими приложениями, элементом окна Mathcad будет маленькая яркая панель. Называется она Math (Математические), и на ней расположены девять ссылок на различные панели Mathcad. Попробуем для интереса открыть все эти панели (рис. 1.2).

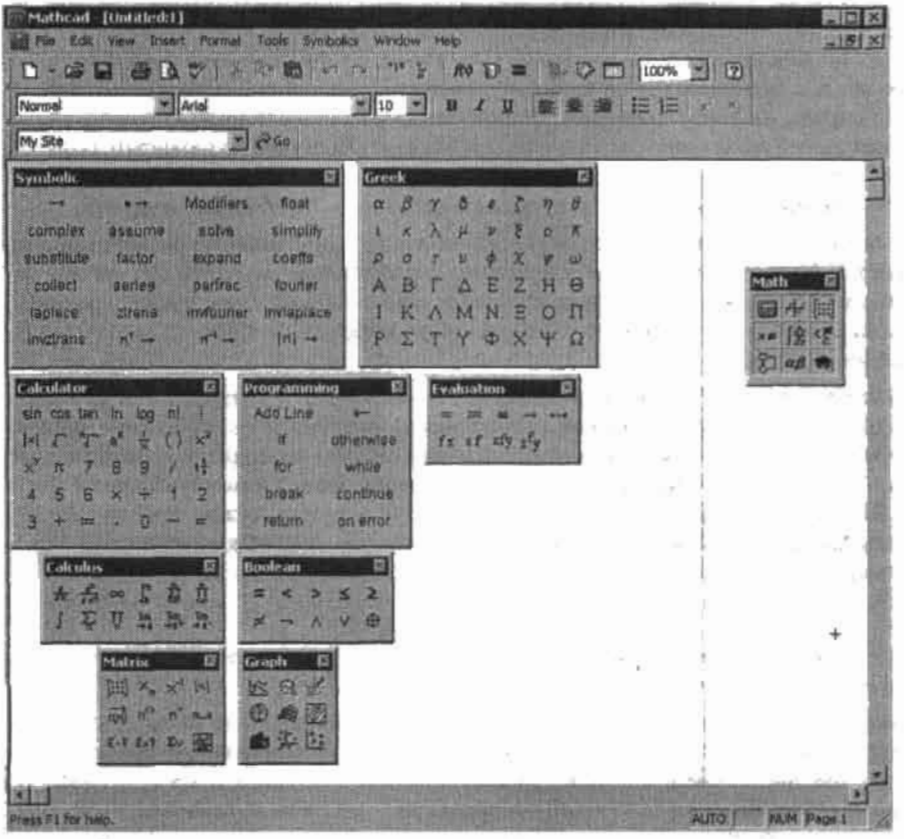


Рис. 1.2. Рабочие панели семейства Math (Математические)

Раскрытые панели семейства Math (Математические) демонстрируют такое разнообразие всевозможнейших операторов и команд, что могут испугать любого новичка. Однако, внимательнее присмотревшись к их содержимому и прочитав название, совсем нетрудно догадаться об их назначении. Так, например, панель Calculator (Калькулятор) содержит элементы, которые действительно можно встретить на любом научном калькуляторе: деление, умножение, синус, корень и прочие простейшие математические действия. На панели Graph (Графические) расположены кнопки со стилизованным изображением зависимостей нескольких типов. И действительно, данная панель служит,

как будет показано далее, для построения графиков. Чтобы понять назначение панели Calculus (Вычислительные), не обязательно читать ее название: все операторы (предназначенные для выполнения основных операций математического анализа), расположенные на ней, полностью соответствуют принятым в математике правилам оформления. А вот для того чтобы разобраться с назначением операторов панели Symbolic (Символьные), потребуется некоторое знание английского языка. Воспользовавшись словарем, можно догадаться, что оператор solve (Решить) решает уравнения и неравенства, а, например, substitute (Подставить) должен производить какие-то замены.

Еще раз внимательно рассмотрите все панели и для экономии места оставьте на экране четыре описанные, закрыв все остальные, которые нам в данном разделе не понадобятся.

Освободив достаточное пространство рабочего листа, внимательно изучите его. На первый взгляд он вам покажется простым белым полем, однако немного передвинув его видимую часть с помощью одной из полос прокрутки, вы обнаружите, что не ограниченное ни справа, ни снизу пространство документа разбито с помощью тонких серых штриховых линий на прямоугольные участки, соответствующие по размерам формату A4. Существование таких границ оправдывается практикой, так как они помогают правильно форматировать решенные и оформленные задачи перед их распечаткой.

Помимо линий границ, на рабочем листе можно увидеть маленький красный крестик (Crosshair). В Mathcad он выполняет функции курсора ввода. В общем случае он указывает на участок документа, в который будет вставлен любой объект: формула, текст, график или даже картинка. Переместить его можно простым щелчком мышью на нужном фрагменте листа.

Попробуйте теперь, переместив курсор ввода на наиболее подходящий для наблюдения участок видимой части рабочего листа, набрать какую-нибудь функцию, численное значение которой вам было бы интересно узнать. Чтобы это сделать, вспомните, что Mathcad — это система, максимально приближенная к традициям «бумажной» математики. Значит, чтобы вычислить, например, синус четырех, нужно просто набрать sin(4) и поставить «=». Сразу же после того, как вы зададите необходимый текст, справа от функции появится ее значение, вычисленное с точностью до третьего знака:

$$\sin(4) = -0.757$$

Попробуйте теперь затереть старое значение переменной и ввести новое — величина функции будет мгновенно пересчитана.

На примере вычисления значения синуса вы познакомились с одной из важнейших функций системы Mathcad — проведением расчетов в режиме реального времени: любое выражение, введенное в документ, будет сразу же обработано, и, по возможности, будет выдан ответ. Если вы когда-нибудь работали с другими математическими программами, то вы должны понимать, насколько это необычно. В подавляющем большинстве программ и языков программирования для запуска подсчета необходимо задействовать специальную команду. Расчеты же в режиме реального времени имеют преимущества в том, что позволяют находить ошибки и описки непосредственно в процессе создания алгоритма, что много проще, чем определить их тогда, когда алгоритм уже будет написан. Кроме того, усилий на вычисления при этом нужно потратить меньше.

Кстати, задать синус можно по-иному (проще). Для этого следует, поставив курсор ввода в нужную точку листа, нажать соответствующую кнопку панели Calculator (Калькулятор). При этом будет введена следующая заготовка:

$$\sin(■)$$

Черный прямоугольник в скобках функции — это маркер (placeholder), обозначающий какой-то незаполненный фрагмент (это может быть переменная или операнд). Сталкиваться с маркерами мы будем очень часто, поскольку они возникают на одной из стадий при задании практически любого объекта в Mathcad. При необходимости маркер, равно как и всю формулу, можно стереть, используя, например, клавишу Backspace.

Подведя курсор к маркеру, расположенному в скобках переменной, определите ее значение и затем введите (с клавиатуры или панели Calculator (Калькулятор)) оператор вывода «=». И опять моментально будет получен результат.

Все функции, используемые в Mathcad, можно разделить на встроенные и пользовательские. Встроенные функции — это функции, закон изменения которых задан в Mathcad изначально. Например, синус — это встроенная функция. Для получения значения встроенной функции достаточно просто корректно набрать ее имя с клавиатуры (другие способы задания рассмотрены в гл. 2). Встроенных функций в Mathcad сотни, и о большинстве из них мы поговорим в рамках данной книги.

Пользовательская функция — это функция, заданная произвольным образом с помощью сочетания переменных и встроенных функций. Например:

$$f(x) := \frac{\sin(x)}{x}$$

Правила определения функций в Mathcad абсолютно те же, что и в обычной математике. То есть для того, чтобы задать какую-либо новую функцию, нужно прописать ее имя (сочетание любого количества практически любых символов), переменные, от которых она зависит (если их несколько, то вводятся они в скобки через запятую), и определяющее ее математическое выражение. Единственное же отличие заключается в том, как выглядит оператор определения. В «бумажной» математике это обычное «=». В Mathcad же используется принятый в некоторых языках программирования (Pascal) оператор «:=» (Definition). Ввести данный оператор можно с панели Calculator (Калькулятор). Применять для присвоения какого-то значения простое равенство ни в коем случае нельзя, поскольку в Mathcad «=» выполняет совершенно конкретную роль оператора численного вывода.

Чтобы получить теперь значение заданной функции при определенном значении переменной, нужно ниже или правее ее выражения ввести ее имя с соответствующим числом в скобках и поставить «=». Например:

$$f(5) = -0.192 \qquad f(\pi) = 0$$

Если же вы введете данные выражения выше или левее определения функции, то результат получен не будет, а сами они окрасятся в красный цвет. Около первого из выражений появится яркая желтая панель сообщения об ошибке: This variable is undefined (Переменная не определена) (рис. 1.3).

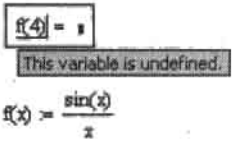


Рис. 1.3. Неверное взаимное расположение формул

Причина возникшей трудности связана с тем, что документ Mathcad — это не просто белый лист, а очень сложная логически активная программная среда. И даже такой

простой пример, как задание функции и получение ее численного значения — это тоже программа. Читает же программы Mathcad точно так же, как компилятор любого из языков программирования: слева направо и сверху вниз. Если же вы пытаетесь получить значение функции, введя ее имя с нужной величиной переменной раньше определяющего ее выражения, то вы заставляете подсчитывать систему неизвестную ей функцию. А это, естественно, невозможно.

Кстати, в Mathcad существует несколько сотен видов сообщений об ошибках.

Вычислить значение функции или выражения можно также, присвоив переменной (или переменным), от которых она зависит, конкретные числовые значения. Например:

$$x := 1 \qquad y := 2$$

$$\frac{1}{x^2} + \sqrt[12]{x - y} = 1.966 + 0.259i$$

Обратите внимание, что Mathcad может оперировать комплексными числами с той же эффективностью, что и обычными. Вообще, точность вычислений в Mathcad очень высока и составляет 15-16 знаков после запятой (при использовании символьных методов она может быть гораздо выше). Однако при принятых по умолчанию настройках ответ округляется до третьего знака. Каким же образом можно получить более точный результат, мы поговорим в гл. 2.

Вы уже, наверное, заметили, что вид курсора ввода сразу же меняется с красного крестика на голубой уголок, как только вы начинаете набирать какую-либо формулу. При редактировании выражений такой вид курсора имеет принципиальное значение, поскольку в Mathcad выделенной считается та часть формулы, которая охватывается его взаимно перпендикулярными линиями.

По умолчанию, как только вы начинаете набирать какую-то информацию, включается формульный режим ввода. Этот режим характеризуется прежде всего тем, что любые введенные в нем выражения анализируются исходя из их математического смысла. Однако, помимо формульного, в Mathcad существует и текстовый режим ввода. Чтобы перейти в него, набрав какой-нибудь текст, нажмите пробел. При этом поменяется шрифт введенного слова, а курсор ввода приобретет вид красной линии. При работе в текстовом режиме проводить расчеты невозможно, однако он все же очень полезен при создании всевозможных комментариев.

При работе в Mathcad важно понимать, что в нем реализованы идеи двух принципиально различающихся подходов к вычислениям. Более традиционные численные методы используют разнообразные алгоритмы, позволяющие более или менее точно получать численный результат той или иной математической операции за счет всевозможных приближений. Более сложными по своей технической реализации и более универсальными по возможностям являются символьные, или аналитические, методы. Работа символьного процессора связана с анализом текста самой преобразуемой формулы, а это позволяет, например, при интегрировании получать не просто десятичную дробь ответа, как в случае применения численного метода, а саму функцию первообразной. Кроме того, символьные результаты абсолютно точны. В общем, смело можно утверждать, что по всем параметрам аналитические решения в Mathcad превосходят численные. Однако, к сожалению, применение возможностей символьного процессора весьма ограничено тем, что для очень немногих задач существует аналитическое решение, а также тем, что далеко не со всеми задачами программа может справиться. Более подробная сравнительная характеристика численных и символьных методов приведена в следующей главе книги.

Некоторые примеры могут быть просчитаны как символьно, так и численно. Чтобы система могла определить, по какому типу следует проводить расчет, существуют два различных оператора вывода. С одним из них, оператором численного вывода «=», мы уже знакомы. Второй же, оператор символьного вывода, имеет вид стрелки, и ввести его можно с панели Symbolic (Символьные). Никаких различий в технике использования этих операторов нет.

Рассмотрим пример выполнения одного и того же преобразования символьным и численным процессором. Попробуем подсчитать несобственный интеграл вероятности. Для этого найдем на панели Calculus (Вычислительные) оператор определенного интеграла и введем его в документ:

$$\int_{\blacksquare}^{\blacksquare} \blacksquare \, d\blacksquare$$

Данный оператор содержит четыре маркера, в которые вводим информацию в соответствии с принятыми в математике правилами:

$$\int_{-\infty}^{\infty} \frac{e^{-x^2}}{2} dx$$

Чтобы задать степень экспоненты и икса, используйте соответствующие операторы панели Calculator (Калькулятор). Символ бесконечности можно ввести с панели Calculus (Вычислительные).

Чтобы получить расчет с помощью приближенного численного метода, вводим «=»:

$$\int_{-\infty}^{\infty} \frac{e^{-x^2}}{2} dx = 0.886$$

Если же вам необходим более корректный аналитический результат, используйте оператор символьного вывода «→»:

$$\int_{-\infty}^{\infty} \frac{e^{-x^2}}{2} dx \rightarrow \frac{1}{2} \cdot \pi^{\frac{1}{2}}$$

Приведенные примеры показывают принципиальное различие численных и символьных расчетов: символьный процессор производит вычисления подобно человеку и стремится получить результат в виде какого-то выражения, а численный процессор всегда выдает только десятичную дробь.

Помимо определенного, символьный процессор может вычислять и неопределенный интеграл (чего, естественно, нельзя сделать с помощью численных методов):

$$\int \sin(x) \cdot \cos(x) \cdot x \, dx \rightarrow \frac{-1}{2} \cdot x \cdot \cos(x)^2 + \frac{1}{4} \cdot \sin(x) \cdot \cos(x) + \frac{1}{4} \cdot x$$

Вообще, возможности аналитических преобразований в Mathcad просто удивительны. Весьма подробно о них мы поговорим в последующих главах, а пока рассмотрим пример аналитического решения уравнения с помощью оператора solve (Решить) панели Symbolic (Символика):

$$e^{x} - b \cdot a^{x} \text{ solve}, x \rightarrow \frac{-\ln(b)}{-1 + \ln(a)}$$

Очень просто можно построить в Mathcad и график любой функции. Для этого нужно задействовать кнопку с изображением плоской кривой панели Graph (Графические). При этом на лист будет вставлена специальная заготовка, в центральных маркерах которой, расположенных снизу и слева ее внутренней рамки, следует задать вид функции (или ее имя) и переменную, от которой она зависит. Вот и все (рис. 1.4).

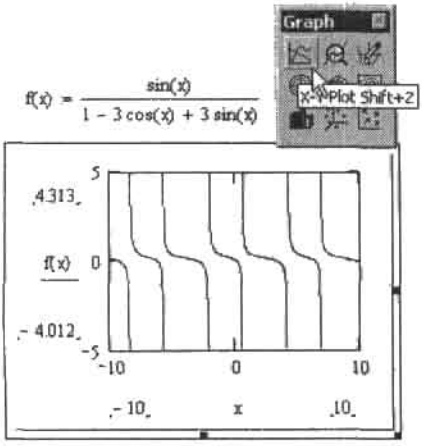


Рис. 1.4. Построение графика в Mathcad

Типов всевозможных графиков в Mathcad существует довольно много. Однако особенно поражают возможности форматирования их вида. Подробно этот вопрос будет рассмотрен в гл. 6.

В данном разделе авторы изложили тот минимум, владея которым, можно приступать к непосредственному изучению тематических глав. И если вам не терпится начать поскорее решать — пропустите остаток этой главы. К изложенному здесь материалу, касающемуся в основном всевозможных параметров оформления документа, вы сможете вернуться по мере необходимости.

1.4. Ввод информации

В этом разделе мы рассмотрим основные принципы задания формул и текстовых комментариев, а также особенности проведения некоторых стандартных операций с ними.

1.4.1. Создание формул

Научиться набирать формулы в Mathcad чрезвычайно просто, даже не читая учебники. Все дело в том, что работает формульный редактор в интуитивном режиме, и все

процедуры, которые в нем могут быть проделаны, технически совершенно очевидны. Поэтому подробно освещать этот вопрос мы не будем и остановимся лишь на некоторых узловых моментах.

Наиболее оптимальный способ изучить создание формул в Mathcad — это вместе набрать какое-нибудь непростое выражение, например:

$$\frac{\ln\left(\frac{x+y}{5+y}\right)+y^2}{\sqrt[3]{x+y}}\cdot e^{-x}$$

1. Приступить к созданию формулы можно с любого ее фрагмента. Мы поступим традиционно и начнем набор с левого верхнего угла выражения, в котором располагается натуральный логарифм.

   Наиболее просто и быстро набрать обозначение логарифма можно с клавиатуры. Если вы плохо владеете клавиатурой, то вам стоит воспользоваться специальной командой панели Calculator (Калькулятор). Независимо от выбранного метода, вы должны получить следующее выражение:

   $$\ln(■)$$

   Черный маркер в скобках функции, как вы помните, служит для обозначения фрагмента, в который следует ввести переменную или операнд. Кстати, если вы набираете функцию с клавиатуры, маркер появляется сразу же после того, как вы зададите левую скобку. Если вы вводите какой-либо оператор, то он сразу вставляется с количеством маркеров, соответствующим количеству его операндов. Например:

   $$■ + ■ \qquad \sqrt[■]{■}$$

   При редактировании в случае удаления маркера оператора (с помощью клавиши Backspace) удаляется и сам оператор. При этом на его месте могут появляться черные рамки, являющиеся в Mathcad маркерами операторов. Например:

   $$x \square ■$$

2. Задать отношение, от которого зависит натуральный логарифм, можно по-разному. Во-первых, можно сразу ввести заготовку дроби и затем по отдельности вносить в нее выражения для числителя или знаменателя. Вставляется же заготовка отношения либо нажатием клавиши «/», либо с помощью соответствующей команды меню Calculator (Калькулятор):

   $$\frac{■}{■}$$

   Однако мы поступим проще. Введем для начала числитель. Его можно просто набрать с клавиатуры:

   $$x + y$$

   Далее все это выражение требуется поделить на знаменатель. Однако просто нажать / вы не можете, так как при этом, в зависимости от положения курсора ввода, получится одно из следующих отношений:

   $$x + \frac{y}{■} \qquad \frac{x}{■} + y$$

Чтобы произвести деление правильно, вспомним, каким образом обозначается выделение в Mathcad. Для этого служат две взаимно перпендикулярные линии курсора формульного ввода (editing lines): горизонтальная (underline) и вертикальная (insertion line). Тот фрагмент формулы, который они охватывают, и считается выделенным. Обычно формулы образованы несколькими логическими уровнями. Так, в нашем случае это переменная, сочетание переменных, выполняющее функцию числителя, отношение сочетаний переменных, выступающее как аргумент логарифма, логарифм, сочетание функций как числитель, отношение сочетаний функций и, наконец, вся формула. Итого шесть уровней. Чтобы перейти от выделения более низкого уровня к более высокому, следует нажать пробел. Следовательно, чтобы, например, удалить всю формулу, пробел придется нажать шесть раз.

Чтобы выделить «x+y», нажимаем пробел один раз. При этом горизонтальная линия ввода сравняется по длине с выражением, и можно будет корректно ввести оператор деления:

$$\frac{x+y}{\blacksquare}$$

3. Чтобы переместить курсор ввода, просто щелкаем на нужном фрагменте левой кнопкой мыши. В нашем случае это маркер знаменателя. Никаких новых особенностей при задании самого знаменателя нет:

$$\ln\left(\frac{x+y}{5+y}\right)$$

4. Далее к заданному логарифму требуется прибавить y^2. Для этого прежде всего выделяем полностью его выражение тремя последовательными нажатиями пробела и вводим с клавиатуры (или панели Calculator (Калькулятор)) «+». В появившийся при этом справа от оператора маркер вводим нужную переменную. Чтобы возвести ее теперь во вторую степень, можно поступить по-разному. Лучшим вариантом для новичка является использование специальной команды все той же панели Calculator (Калькулятор). Нажав кнопку с изображением необходимой процедуры, вы сразу же зададите квадрат переменной. Однако набравшись опыта, вы никогда не будете при создании формул обращаться к описанному выше способу. Все дело в том, что практически любая важная процедура в Mathcad может быть проделана с помощью сочетаний клавиш (Hot Keys). Так, степень задается с помощью сочетания Shift+6. Нажав его, вы увидите, что на месте будущего показателя появился маркер, в который следует ввести нужный текст. Набор формул с помощью клавиатуры с использованием сочетаний клавиш позволяет значительно сэкономить время по сравнению с более простым технически, но куда менее эффективным заданием с применением панелей. Правда, при использовании обоих описанных методов нужно внимательно следить, чтобы выделенным был именно нужный фрагмент.

   В результате проделанных операций получим:

$$\ln\left(\frac{x+y}{5+y}\right)+y^2$$

5. Далее требуется поделить полученное выражение на кубический корень из суммы переменных. Для этого повторным нажатием необходимого количества пробелов выделяем его и нажимаем /. В появившийся маркер вставляем либо командой панели Calculator (Калькулятор), либо сочетанием Ctrl+\ оператор корня произвольной степени.

Введенный оператор содержит, исходя из его математического смысла, два маркера, в которые должны быть внесены соответственно порядок извлекаемого корня и выражение, над которым данная операция должна быть проведена. Последовательность, в которой производится заполнение маркеров оператора, не оказывает влияния на конечный результат, поэтому вы можете выполнить его так, как вам более удобно. Перемещаться в пределах одной формулы или оператора можно либо с помощью мыши, либо (лучше) используя клавиши управления курсором.

После задания кубического корня наша формула приобретет вид:

$$\frac{\ln\left(\frac{x+y}{5+y}\right)+y^2}{\sqrt[3]{x+y}}$$

6. Необходимо произвести умножение полученного выражения на e^{-x}. Для этого выделяем его и, обязательно поставив оператор умножения, вводим степень экспоненты либо с помощью панели Calculator (Калькулятор), либо с клавиатуры, используя сочетание Shift+6.

   Кстати, к заданию операторов умножения в Mathcad следует относиться очень осторожно, так как многие ошибки при решении задач бывают связаны именно с тем, что пользователи, набирая выражения, забывают про них. А так как оператор умножения по умолчанию не отображается, найти потом ошибку будет очень сложно.

   Ошибки, связанные с оператором умножения, бывают двух типов.

   - Если произведение должно стоять между двумя переменными (например, x и y), а вы запишете просто xy, то это сочетание будет воспринято системой как неопределенная переменная, о чем будет выдано сообщение This variable is undefined (Переменная не определена).
   - Если же нужно произвести умножение некоторой переменной на выражение в скобках, а вы забудете использовать соответствующий оператор, то полученное сочетание система воспримет как функцию. А поскольку эта переменная уже определена выше как число, система выдаст сообщение об ошибке: This name does not refer to a function (Данное имя не относится к функции).

   Если вам встретятся подобные сообщения, то первым делом проверьте наличие всех операторов умножения.

Вот и все. Наше выражение задано. Мы можем даже подсчитать его значение при определенных значениях переменной. Например:

$$x := \pi \qquad y := e$$

$$\frac{\ln\left(\frac{x+y}{5+y}\right)+y^2}{\sqrt[3]{x+y}}\cdot e^{-x} = 0.171$$

В качестве значений переменных были использованы две наиболее распространенные математические константы. Число π можно задать четырьмя основными способами:

- использовав соответствующую команду меню Calculator (Калькулятор);
- нажав сочетание клавиш Ctrl+Shift+P;

- введя соответствующий символ с панели Greek (Греческие), содержащей греческий алфавит;
- вставив с клавиатуры латинскую «р» и нажав сочетание Ctrl+G. Это общий способ задания греческих букв с клавиатуры.

На практике чаще всего используется второй метод, хотя для новичка, возможно, удобнее будет работать с панелью.

Основание натурального алгоритма вводится как соответствующая латинская буква с клавиатуры.

Редактируются формулы, как правило, по тем же принципам, что и создаются. Для удаления какого-то фрагмента требуется его выделить и нажать Backspace. Если удаляемый фрагмент образован более чем одним символом, то при однократном нажатии он лишь окрасится черной заливкой. Чтобы все-таки удалить его, нажмите соответствующую клавишу второй раз. Подобный режим удаления весьма оправдан, так как если бы формула вырезалась одним нажатием Backspace, то в режиме редактирования с помощью клавиатуры, случайно задействовав соответствующую кнопку, вы могли бы потерять какой-то трудновосстановимый объект. Кстати, выделить подобным образом нужный фрагмент формулы можно и осторожным протаскиванием курсора при нажатой левой кнопке мыши (рис. 1.5).

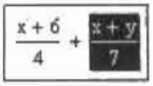


Рис. 1.5. Вид выделенного фрагмента при использовании мыши

Для удаления всей формулы или ее фрагмента можно также использовать соответствующие команды панели Standard (Стандартные) или контекстного меню.

Чтобы перемещаться по формуле при ее форматировании, нужно использовать либо клавиши управления курсором, либо мышь. Единственный неочевидный момент при этом — каким образом можно произвести добавление нужного фрагмента левее уже созданного участка? Делается это точно так же, как при стандартном форматировании. Единственное, что нужно дополнительно сделать, это развернуть с помощью клавиши Insert вертикальную линию курсора ввода влево (впрочем, это можно сделать и используя соответствующие клавиши управления курсором).

Заканчивая разговор о создании формул (рис. 1.6), приведем несколько полезных фактов.

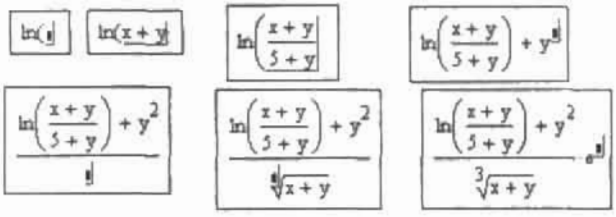


Рис. 1.6. Последовательность задания формулы

- Если создаваемая вами формула слишком длинная, например для распечатки на формате А4, вы можете разбить ее на две части. Для этого, выделив левую часть выражения (рис. 1.7), нажмете Ctrl+Enter.

$$f(x) := x - \frac{1}{6}\cdot x^3 + \frac{1}{120}\cdot x^5 - \frac{1}{5040}\cdot x^7 + \frac{1}{362880}\cdot x^9 - \frac{1}{39916800}\cdot x^{11}$$

Рис. 1.7. Разбиение длинной формулы

При этом выделенная часть будет взята в скобки, а на строку ниже появятся плюс и маркер, в который следует переместить оставшийся фрагмент. Маркер, в котором он находился ранее, и скобки нужно просто удалить.

$$f(x) := \left(x - \frac{1}{6}\cdot x^3 + \frac{1}{120}\cdot x^5 - \frac{1}{5040}\cdot x^7 \ldots \right) + \frac{1}{362880}\cdot x^9 - \frac{1}{39916800}\cdot x^{11}$$

Формулы, полученные с использованием описанной методики, работают ничуть не менее корректно, чем формулы без переносов.

$$f(x) := x - \frac{1}{6}\cdot x^3 + \frac{1}{120}\cdot x^5 - \frac{1}{5040}\cdot x^7 \ldots + \frac{1}{362880}\cdot x^9 - \frac{1}{39916800}\cdot x^{11} \qquad f\left(\frac{\pi}{6}\right) = 0.5$$

- При задании функций и выражений часто приходится использовать скобки. Конечно, можно для их задания применить и соответствующие кнопки клавиатуры, однако существует и более простой способ. Чтобы ввести пару круглых скобок, просто нажмите '. Кстати, аналогичная по функциям команда реализована и на панели Calculator (Калькулятор).
- Тип отображения некоторых операторов может быть различным. Каким образом можно его сменить, рассматривается в гл. 2.
- Иногда при задании похожих по виду выражений бывает полезным использовать стандартные для Windows команды копирования и вставки. Для этого обычно используются соответствующие им сочетания клавиш: Ctrl+C — для копирования в буфер, Ctrl+X — для вырезания формулы или части формулы в буфер, Ctrl+V — для вставки.

1.4.2. Создание текстовых областей

Mathcad — это математический редактор, и, естественно, формульный режим ввода имеет в нем основное значение. Однако иногда весьма полезным может быть и текстовый режим, который позволяет создавать всевозможные комментарии и качественно оформлять решенные задачи.

Необходимость введения в Mathcad режима, отличного от формульного, связана с тем, что сочетания, служащие для введения большинства служебных символов, употребляются в нем для задания всевозможных операторов. Так, например, при попытке ввести символ @ на лист будет вставлена графическая область кривой. В случае существования в Mathcad только одного режима ввода часто бы возникали проблемы со всевозможного рода неопределенностями.

Чтобы ввести текстовую область (Text Region), нажмите сочетание клавиш Shift+«"» (курсор ввода при этом должен располагаться на чистом участке документа — иначе будет вставлен оператор комплексного сопряжения). При этом появится специальная рамка, а курсор ввода приобретет вид красной вертикальной линии (рис. 1.8).

Рис. 1.8. Созданный текстовый регион

Набирается текст в Mathcad точно так же, как в любом текстовом редакторе. Если вы приблизительно знаете, сколько места на листе займет ваш комментарий, то можете сразу растянуть текстовую область до нужных размеров. Чаще же текст просто вводят в область, обрывая строки с помощью Enter, когда они достигают нужной длины (это приходится делать, так как Mathcad не выполняет автоматически переносов слов).

Перемещать текстовые области по документу можно точно так же, как формулы: протаскиванием при нажатой левой кнопке мыши.

Говорить о форматировании текста сейчас не будем, поскольку этот вопрос рассматривается в гл. 19. Отметим лишь, что оно имеет довольно широкие возможности и очень схоже с выполнением той же операции в Word.

Если вы хотите качественно оформить вашу задачу, то может понадобиться создать текст, содержащий математические формулы. В простом текстовом режиме сделать это невозможно, поэтому необходимо создавать в тексте островки формульного режима (рис. 1.9). Сделать это можно, выполнив следующую последовательность действий.

1. Переместите, щелкнув левой кнопкой мыши или используя клавиши управления курсором, курсор ввода в нужную точку текста.
2. Задействовав команду Insert ▸ Math Region (Вставить ▸ Математическая область) или соответствующее ей сочетание Ctrl+Shift+A, введите маркер формульного режима.
3. Введите в появившийся маркер необходимое математическое выражение.

Формулы, заданные описанным способом, вычисляются по тем же принципам, что и обычные.

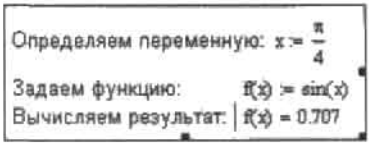


Рис. 1.9. Использование формул в текстовых областях

То, что формулы, заданные в тексте, являются вычисляемыми, не всегда может быть полезным. В некоторых случаях формулы комментария могут вносить некорректность или вовсе сбивать работу самого алгоритма. В таком случае можно поступить двояко. Во-первых, формулу можно сделать невычисляемой с помощью команды Disable Evaluation (Невычисляемая) контекстного меню. Однако при этом она будет помечена черным прямоугольником, что может быть неприемлемым при подготовке отчетов или докладов. Во-вторых, можно использовать возможность взаимодействия Mathcad с другими текстовыми редакторами, которые (прежде всего Word) лишены описанного недостатка. В последнем случае выполните следующую последовательность действий.

1. Создайте в Word необходимый текст. Для задания формул можно использовать как специализированное приложение Microsoft Equation, так и просто скопировать соответствующие выражения из Mathcad. При этом в самом Word они будут рассматриваться как OLE-объекты, однако при копировании в буфер будут переведены в формат точечных рисунков.

2. Скопируйте созданный текст в буфер.
3. Вставьте текст из буфера на нужный участок документа. Впрочем, последнее действие может быть реализовано по-разному.
   - Текст можно вставить в специально созданную текстовую область. При этом в дальнейшем будет существовать возможность его форматирования непосредственно в Mathcad.
   - Более важным практически является импорт текста в виде объекта OLE (реализуется при вставке текста непосредственно на документ). В этом случае текст перенесется в том же виде, в котором он был создан в Word. Чтобы изменить такой текст, нужно дважды щелкнуть на нем левой кнопкой мыши. При этом все меню и панели Mathcad будут заменены соответствующими элементами окна того приложения, в котором был создан объект.

Вставка текста в виде объекта OLE позволяет создавать текстовые комментарии с уровнем качества, недоступным текстовому редактору Mathcad. Таким образом, например, можно вводить комментарии с корректно расставленными переносами и специальными элементами, например буквицами.

1.4.3. Правка документа

Иногда те или иные операции редактирования требуется применить не к одному, а сразу к нескольким регионам. Конечно, можно отформатировать документ, применяя соответствующие операции к каждому из объектов по отдельности, однако если их достаточно много, то такой способ совершенно неэффективен. Существует несколько методов выделения группы регионов в зависимости от их взаимной организации.

- Если удалить, переместить или скопировать нужно все объекты, расположенные в документе, то используйте команду Select All (Выделить все) меню Edit (Правка). Также для выполнения этой операции можно задействовать традиционное для Windows-приложений сочетание Ctrl+A.
- Если должны быть отформатированы не все регионы, а только какая-то ограниченная группа, то ее нужно выделить, захватив специальным штрихованным прямоугольником (рис. 1.10). Операция эта стандартная для Windows и проводится протаскиванием при нажатой левой кнопке мыши.

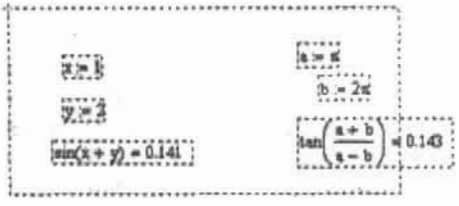


Рис. 1.10. Выделение группы регионов

- Если необходимо отредактировать группу из нескольких разрозненных регионов, их следует выделить последовательными щелчками левой кнопкой мыши при нажатой клавише Shift.

Снимается выделение, независимо от его типа, простым щелчком мышью на документе.

Рассмотрим теперь основные операции форматирования, которые могут быть использованы по отношению к выделенным регионам.

- Перемещение объектов по документу. Осуществляется точно так же, как, например, перемещение ярлыков на рабочем столе в Windows, протягиванием при нажатой левой кнопке мыши контуров регионов в нужную точку листа (указатель мыши при этом примет вид ладони).
- Если группу объектов требуется опустить на несколько строк вниз, то можно просто последовательно нажать несколько раз клавишу Enter.
- Чтобы удалить выделенные регионы, можно использовать клавиши Delete и Backspace, сочетание Ctrl+D, соответствующие команды контекстного меню и меню Edit (Правка).
- Копирование и вырезку в буфер, а также последующую вставку группы регионов можно осуществить стандартными командами и сочетаниями клавиш.
- Для выравнивания различных регионов друг относительно друга в Mathcad существует специальная команда Align Regions (Выровнять регионы) меню Format (Формат). Возможны два типа выравнивания: относительно вертикали и относительно горизонтали. Соответственно в подменю Align Regions (Выровнять регионы) имеются две команды: Across и Down. На практике обычно для выравнивания объектов используют не команды меню, а специальные кнопки панели Standard (Стандартные) (рис. 1.11).

Рис. 1.11. Кнопки выравнивания регионов

  Иногда при выравнивании регионы могут перекрываться. В этом случае система выдаст диалоговое окно с вопросом: Selected regions may overlap. Align selected regions? (Выбранные регионы могут перекрываться. Выровнять их?). Появление этого сообщения чаще всего бывает связано с тем, что пользователь пытается применить команду неподходящего для данной группы регионов выравнивания.
- Чтобы разделить перекрывающиеся регионы, можно использовать специальную команду Separate Regions (Разделить регионы) меню Format (Формат).

При работе с большими документами часто возникает необходимость вставить дополнительные формулы, комментарии или иные объекты между уже имеющимися. Чтобы освободить для них часть рабочей области, необходимо переместить нижележащие объекты вниз. Сделать это вручную бывает крайне проблематично. В этом случае очень полезной оказывается команда контекстного меню рабочей области Insert Lines (Вставить строки). Поставьте курсор перед группой объектов, которая должна быть смещена вниз, и задействуйте команду Insert Lines (Вставить строки). В открывшемся диалоговом окне укажите количество строк, необходимое для вставки (максимум 32 767).

Удалить лишние строки можно аналогичным образом, задействовав команду Delete Lines (Удалить строки) того же меню.

Важной возможностью при работе с документами Mathcad является так называемое обновление, предназначенное для очистки листа от всевозможных линий выделения, лишних символов и прочего «мусора», который неизбежно появляется, особенно при решении объемных и сложных задач. Чтобы обновить документ, задействуйте коман-

ду Refresh (Обновить) меню View (Вид) или нажмите сочетание Ctrl+R. В результате в документе должны остаться только значащие объекты.

1.5. Интерфейс пользователя

Интерфейс пользователя Mathcad, как уже было отмечено, весьма схож с любым Windows-приложением и содержит следующие структурные элементы.

- Главное, или верхнее, меню.
- Панели (Toolbars) Standard (Стандартная) и Formatting (Форматирование).
- Панели семейства Math (Математические).
- Контекстные меню.
- Строка состояния (Status Bar).
- Рабочая область (Worksheet).
- Диалоговые окна.

Практически все команды имеют несколько альтернативных способов выполнения с помощью меню, рабочих панелей или клавиатуры.

1.5.1. Главное меню

Главное меню Mathcad расположено стандартным для Windows-приложений образом, в верхней части окна программы.

Верхнее меню Mathcad содержит девять подменю команд, объединенных в соответствии с выполняемыми ими функциями.

- File (Файл). Подменю содержит наиболее общие команды работы с документом, такие как его создание, открытие, сохранение, печать, отправка по электронной почте, описание.
- Edit (Правка). В данном подменю расположены общие для Windows-программ команды правки: отмена неверного действия, вырезка, копирование, проверка орфографии, выделение, поиск и некоторые другие.
- View (Вид). Подменю содержит различные команды отображения рабочей области, регионов, панелей инструментов, установки для колонтитулов.
- Insert (Вставка). Меню предназначено для введения в документ различных объектов: графических, математических и текстовых областей, картинок, функций, матриц, единиц измерения, ссылок, компонентов и некоторых других.
- Format (Формат). Подменю содержит основные команды задания вида формул и результата вычислений, настройки стиля и шрифта и некоторые другие параметры.
- Tools (Инструменты). Данное подменю содержит все основные команды и параметры проведения расчетов в Mathcad, а также создания анимации.
- Symbolics (Символьные). В этом подменю расположены команды символьных расчетов, а также некоторые параметры, отвечающие за технику их проведения.
- Window (Окно). Стандартное меню, содержащее команды отображения окон документов.
- Help (Помощь). В данном подменю содержатся ссылки для вызова справочной системы, Ресурсов Mathcad, а также некоторых ресурсов в Интернете.

Подробно разбирать все команды, содержащиеся в главном меню, не имеет смысла. Мы это сделаем постепенно, подробно изучая соответствующие тематические главы. В данной главе упомянем лишь наиболее общие параметры и команды.

Помимо описанных отображаемых постоянно подменю, главное меню также содержит и подменю, отвечающее за отображение окна документа. В отличие от остальных, отображается оно не словом, а маленьким ярлыком XMCD-документа.

1.5.2. Контекстные меню

Контекстные, или всплывающие, меню выполняют функции, в общем схожие с главным меню. Однако у всплывающего меню есть огромное преимущество: его содержание зависит от того, при каких обстоятельствах (контексте) оно было вызвано. Система анализирует, какие параметры или команды могут понадобиться при форматировании именно этого объекта, и наиболее важные из них заносит в контекстное меню. Поэтому, осуществляя редактирование некоторой формулы или компонента, используйте лучше его: это позволит сэкономить время за счет того, что вам не придется производить поиск нужной настройки в обширном верхнем меню.

Вызывается всплывающее меню в Mathcad стандартным для Windows щелчком правой кнопкой мыши на соответствующем объекте.

1.5.3. Основные панели инструментов

Панели инструментов (Toolbars) — это специальные элементы окон Windows-приложений, позволяющие получать быстрый доступ к наиболее важным командам главного меню. Соответствующие ссылки на панелях разбиты на функциональные группы (с помощью специальных вертикальных линий-разделителей) и отображаются, как правило, в виде кнопок со стилизованными рисунками.

Панели Mathcad можно разделить на четыре группы: основные, математические, панель Resources (Ресурсы), содержащая список ссылок на огромное количество примеров, подсказок и справочных данных, и панель элементов управления (Controls). К основным относятся обычные для всех Windows-программ панели Formatting (Форматирование) и Standard (Стандартные), а также маленькая панель Math (Математические), содержащая ссылки на панели со специфическими для Mathcad элементами.

Присутствие или отсутствие панели в окне программы определяется специальным подменю Toolbars (Панели инструментов) меню View (Вид).

По умолчанию всегда открыты панели Formatting (Форматирование), Standard (Стандартные), Math (Математические) и Resources (Ресурсы). Математические же панели (отделены от основных в списке линией) вызываются по мере необходимости пользователем. Правда, для этого используют в основном команды панели Math (Математические) и к рассматриваемому меню прибегают лишь в том случае, если она будет случайно закрыта.

Всего панелей инструментов в Mathcad 15 (если учитывать панель модификации значений Modifiers (Модификаторы), которая является, в принципе, приложением панели аналитических расчетов Symbolic (Символьные)).

Изучать значение каждой кнопки рассматриваемых панелей мы не будем: содержание панели Formatting (Форматирование) должно быть вам хорошо известно по опыту работы с другими программами, специфические кнопки панели Standard (Стандартные)

будут рассмотрены в соответствующих тематических главах, а о панели Math (Математические) мы поговорим в следующем разделе. Сейчас же обсудим лишь наиболее общие моменты, связанные с работой с основными панелями инструментов.

По своим особенностям основные панели инструментов несколько отличаются от математических. Так, основные панели (кроме Math) и панель Resources отображаются по умолчанию стационарными, а математические — всегда плавающими (то есть их положение не зависит от перемещения окна программы). Впрочем, плавающей можно сделать и любую стационарную панель. Для этого нужно просто перетянуть ее контур при нажатой левой кнопке мыши в произвольную область экрана (чтобы ее снова сделать стационарной, достаточно дважды щелкнуть мышью на ее названии).

При подведении указателя мыши к определенной кнопке панели инструментов система выдаст всплывающую подсказку, кратко определяющую ее назначение. Более развернутый комментарий при этом отображается в строке состояния. Используя это свойство, можно изучать назначение неизвестных вам команд и не обращаясь к справочной системе. Кстати, при работе с математическими панелями комментарии в строке состояния не отображаются.

Закрыть основную панель инструментов можно одним из трех стандартных способов: либо сняв флажок из соответствующей строки подменю Toolbars (Панели инструментов) меню View (Вид), либо использовав команду Hide (Скрыть) контекстного меню, либо, если панель является плавающей, нажать кнопку с крестиком в ее верхнем правом углу.

Также, в случае основных панелей, пользователь самостоятельно может определить присутствие или отсутствие на них определенных ссылок. Чтобы это сделать, задействуйте команду Customize (Настроить) контекстного меню панели.

1.5.4. Математические панели

Всего математических панелей в Mathcad девять. Открываются они обычно с помощью соответствующих команд панели Math (Математические) (рис. 1.12), однако можно использовать и стандартный метод обращения к меню Toolbars (Рабочие панели) меню View (Вид).

Рис. 1.12. Панель Math (Математические)

Кратко охарактеризуем все панели семейства Math (Математические).

- Calculator (Калькулятор). На данной панели расположены арифметические операторы, цифры от 0 до 9, некоторые наиболее распространенные функции и математические константы, а также операторы вывода.
- Graph (Графические). С помощью этой панели можно вызвать заготовки для построения разнообразных графиков и поверхностей. Кроме того, здесь расположены ссылки на инструменты для анализа зависимостей.
- Matrix (Матричные). На этой панели расположены операторы создания, обращения, транспонирования матриц, а также операторы матричных индексов и столбцов. Кроме того, здесь вы можете найти и операторы для работы с векторами.

- Evaluation (Выражение). Здесь вы можете найти ссылки на все операторы ввода и вывода Mathcad, а также заготовки для создания пользовательских операторов.
- Calculus (Вычислительные). На этой панели содержатся используемые при решении задач математического анализа операторы: определенного и неопределенного интегралов, производных, пределов, сумм и произведений. Кроме того, отсюда вы можете задать символ бесконечности.
- Boolean (Булевы). Эта панель предназначена для задания логических операторов.
- Programming (Программирование). Панель содержит операторы языка программирования Mathcad.
- Greek (Греческие). На данной панели расположены буквы греческого алфавита.
- Symbolic (Символика). Панель предназначена для проведения всевозможных аналитических преобразований.

Обычно нет необходимости расположения на экране всех математических панелей, поэтому, переходя от решения задач одного типа к задачам другого, стоит закрывать ненужные. Сделать это можно как стандартными, описанными в предыдущем разделе способами, так и (лучше) простой деактивацией соответствующей кнопки панели Math (Математические).

Значительно сэкономить место окна программы позволяет изменение формы панелей. Чтобы это сделать, следует, подведя указатель мыши к нужному краю панели, выполнить протягивание в требуемом направлении. О степени изменения при этом формы панели можно судить по специальному контуру (рис. 1.13).

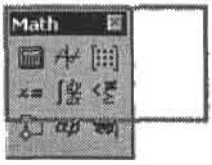


Рис. 1.13. Изменение формы панели

По умолчанию математические панели всегда отображаются плавающими. Однако наиболее часто используемые вами панели гораздо удобнее сделать стационарными. Для этого следует так же, как при простом перемещении, перетянуть контур панели к нужному краю окна программы. При этом панель, приняв форму строки или столбца, автоматически закрепится на свободном фрагменте. Также можно просто дважды щелкнуть левой кнопкой мыши на названии соответствующей панели, однако при этом ее размещение будет произведено автоматически, что не всегда делается удачно. Описанный подход позволяет не только сделать команды панелей инструментов более доступными, но и значительно сэкономить место на рабочем листе (начав работать в Mathcad, вы поймете, насколько это важно). Правда, наиболее крупные панели (Calculator, Programming, Greek, Symbolic) невозможно сделать стационарными, так как их операторы, отображенные в одну строку, заняли бы слишком много пространства.

Большинство операторов, расположенных на математических панелях инструментов, могут быть введены и с помощью соответствующего сочетания клавиш. Узнать его очень просто: для этого нужно подвести указатель мыши к кнопке интересующего вас оператора. Использование сочетаний клавиш позволяет значительно упрощать и ускорять набор информации в Mathcad, поэтому советую с самого начала стараться пользоваться именно ими.

1.5.5. Рабочая область

Рабочая область занимает большую часть экрана и имеет вид белого бесконечного листа, разбитого на прямоугольники формата A4 (при необходимости особенности разбиения можно изменить с помощью настроек окна Page Setup (Установки страницы) меню File (Файл)). Настроек, отвечающих за вид рабочей области, довольно много, поэтому рассмотрим их в отдельных подразделах.

Цвет рабочей области и регионов

По умолчанию и рабочая область, и регионы объектов отображаются одним цветом — белым. При этом документ выглядит так же, как при распечатке на принтере. Однако многие пользователи предпочитают работать в режиме, в котором используется серый фон. При этом регионы резко выделяются, и проводить форматирование и прослеживать логику решения намного проще. Перейти в данный режим можно, выполнив команду View ▸ Regions (Вид ▸ Регионы).

При желании рабочему листу можно придать совершенно произвольный оттенок. Чтобы это сделать, обратитесь к параметру Background (Заливка) подменю Color (Цвет) меню Format (Формат).

Произвольным образом можно задать и цвет любого региона. Чтобы это сделать, задействуйте команду Properties (Свойства) контекстного меню объекта (вызывается щелчком правой кнопкой мыши на нем). При этом откроется одноименное с командой окно, на вкладке Display (Отображение) установите флажок Highlight region (Цветной регион) (рис. 1.14). Затем на палитре Choose Color (Выбрать цвет) подберите наиболее подходящий для данного региона оттенок.

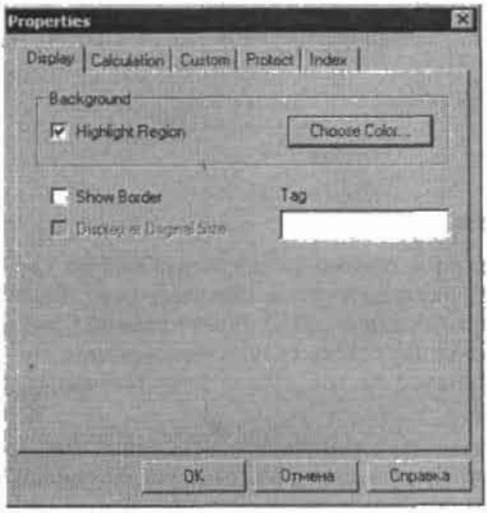


Рис. 1.14. Вкладка Display (Отображение) окна Properties (Свойства)

Здесь же вы можете окаймить выбранный объект рамкой, задействовав параметр Show Border (Показать границу).

Подробно вопрос о том, каким образом можно менять стиль, цвет и размеры формул и текста в Mathcad, мы поговорим в гл. 19.

Линейка

Если вам необходимо выполнить выравнивание регионов, форматирование текста или особенно построение чертежа, следует воспользоваться специальным инструментом Ruler (Линейка), вызываемым одноименной командой меню View (Вид).

В зависимости от типа редактируемых данных линейка может быть простой и текстовой.

С помощью простой линейки вы можете отмерять точные расстояния на листе. Чтобы пометить нужную точку, просто щелкните мышью на соответствующем ей делении на линейке. При этом на ней появится специальная метка, задействовав контекстное меню которой, можно визуализировать опорную линию (Guideline). С помощью того же меню можно выбрать единицы измерения градуировки линейки (рис. 1.15).

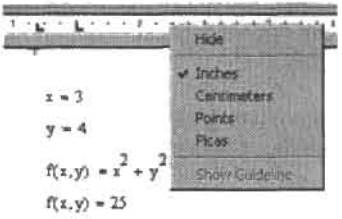


Рис. 1.15. Линейка (Ruler)

Текстовая линейка очень схожа с аналогичным инструментом Word и служит для установки отступов и абзацев (рис. 1.16).

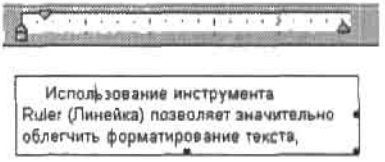


Рис. 1.16. Текстовая линейка

Перемещение по рабочей области

Наиболее стандартным и универсальным способом просмотра документа является использование вертикальной и горизонтальной полос прокрутки. Перемещать их бегунки можно либо протаскиванием мышью (при этом прокрутка будет плавной), либо с помощью соответствующих кнопок сверху и снизу строки (для скачкообразной прокрутки). Чтобы переместиться сразу на несколько листов, нужно щелкнуть мышью на нужном участке полосы прокрутки.

В случае больших документов удобно использовать клавиши пролистывания Page Up и Page Down. Кроме того, для перехода на нужную страницу в Mathcad имеется специальное окно Go to Page (Переместиться на страницу), вызываемое одноименной командой меню Edit (Правка). С помощью данного окна можно переместиться не только на произвольную страницу, задав ее номер в окошке Page Number, но и сразу на первую или последнюю страницу документа, задействовав пункты соответственно First Page или Last Page.

Помимо описанных способов, перемещаться по документу можно с помощью клавиш управления курсором или мыши.

Масштаб документа

При изменении масштаба не меняются ни шрифт текста, ни размеры картинок — увеличивается или уменьшается только их отображение на экране.

Необходимость изменения масштаба документа может возникнуть в том случае, если, например, вам нужно проследить логику решения, а при обычном отображении алгоритм задачи не помещается на один разворот. В такой ситуации масштаб следует уменьшить. И, наоборот, увеличьте его, если у вас плохое зрение и вам трудно изучать примеры.

Регулируется масштаб в Mathcad с помощью обычного для Windows списка Zoom (Масштаб) панели Standard (Стандартные). Здесь вы можете выбрать величину отображения от 25 до 200 % к обычному.

Если предложенные семь вариантов увеличения вам не подходят, то можете задать масштаб произвольным образом, открыв специальное окно Zoom (Масштаб) меню View (Вид).

Работа с несколькими документами

В Mathcad, как и в любой другой Windows-программе, существует возможность работы одновременно с несколькими документами. При этом расположены они могут быть по-разному. Взаимное расположение окон документов определяется тремя параметрами меню Window (Окно).

- Cascade (Каскад). Окна располагаются каскадом, и переходить от редактирования одного из них к другому можно, выполняя щелчок левой кнопкой мыши на видимом фрагменте соответствующего документа.
- Tile Horizontal (Горизонтально). Документы располагаются в виде горизонтальной мозаики, не перекрываясь.
- Tile Vertical (Вертикально). Документы располагаются вертикально, не перекрываясь.

Очень просто можно выполнять копирование в многооконном режиме: для этого нужный объект следует просто перетянуть с одного документа на другой, подобно тому, как перемещаются формулы по листу.

1.5.6. Строка состояния

В нижней части окна Mathcad располагается строка состояния (Status Bar), служащая для отображения самой общей служебной информации.

Чтобы отключить отображение строки состояния, снимите флажок Status Bar (Строка состояния) в меню View (Вид).

1.6. Работа с документами

В данном разделе мы поговорим о создании нового документа Mathcad, о возможных форматах его сохранения, о работе с шаблонами, а также о некоторых других стандартных операциях.

1.6.1. Создание чистого документа

При запуске Mathcad автоматически загружается чистый безымянный документ Untitled 1. Чтобы открыть еще один пустой лист уже в ходе работы, нужно воспользоваться одним из следующих способов.

- Задействуйте команду New (Новый) меню File (Файл).
- Нажмите сочетание клавиш Ctrl+N.
- Используйте кнопку New (Новый) панели Standard (Стандартные).

Если вы воспользуетесь первым способом (из приведенного списка) создания нового документа, то система предложит вам выбрать наиболее подходящий шаблон в меню Worksheet Templates (Шаблоны документов) открывшегося диалогового окна New (Новый) (рис. 1.17). Если вы собираетесь работать в обычном режиме, то либо оставьте помеченной выбранную по умолчанию строку Normal (Обычный), либо выделите строку Blank Worksheet (Чистый документ) (оба этих варианта абсолютно идентичны).

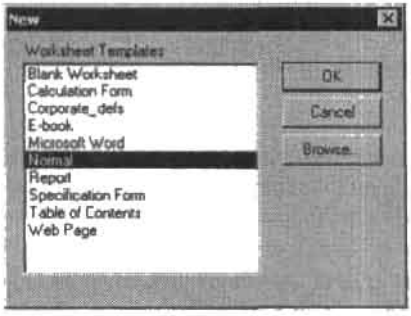


Рис. 1.17. Окно New (Новый)

Аналогичный список шаблонов можно открыть и при создании чистого документа с помощью кнопки New (Новый) панели Standard (Стандартные). Для этого нажмите кнопку с изображением стрелки правее белого листка кнопки New (Новый). Если нажать кнопку New (Новый) сразу, не обращаясь к соответствующему списку, то документ будет создан в стандартном шаблоне Normal (Обычный). То же самое относится и к использованию сочетаний клавиш.

1.6.2. Шаблоны

Мы уже столкнулись с таким важным понятием при работе в Mathcad, как шаблоны (templates). Попробуем разобраться, что это такое и чем они могут быть полезны.

Шаблон – это заготовка документа с сохраненными настройками, элементами интерфейса и даже формулами. Использование шаблонов позволяет значительно сэкономить время на создание и особенно на оформление алгоритма решения задач тех или иных типов. Так, например, если вы студент технического вуза, то, создав однажды шаблон оформления лабораторных работ, в дальнейшем будете тратить на их выполнение считанные минуты (и это с учетом того, что качество оформления будет высоким). Шаблоны незаменимы в том случае, если вам приходится постоянно выполнять аналогичные расчетные работы (а на практике, как правило, так и происходит).

Шаблоны можно разделить на два типа: встроенные и пользовательские.

Встроенные шаблоны отображаются с момента установки программы в уже знакомом нам списке Worksheets Templates (Шаблоны документов) команды создания пустого документа New (Новый). Всего их семь. К наиболее важным стоит отнести шаблон электронной книги (E-book) и веб-документа (Web page).

Впрочем, следует признать, что встроенные шаблоны имеют очень ограниченное применение. Гораздо более важны пользовательские шаблоны.

Чтобы создать собственный шаблон, выполните следующую последовательность действий.

1. Создайте документ, формулы, текст, графические области и настройки которого должны соответствовать будущему шаблону.
2. Зайдите в подменю Save As (Сохранить как) меню File (Файл).
3. Найдите специальную папку шаблонов Template (приблизительный адрес: `C:\Program Files\Mathsoft\Mathcad 12\Template`).
4. Назовите произвольным образом будущий шаблон. Например, USERTEMPLATE.
5. Сохраните созданный вами документ в одном из форматов: Mathcad XML Template (*.xmct) (Шаблон Mathcad с XML-разметкой) или Mathcad Template (*.mct) (Шаблон Mathcad битового формата) (рис. 1.18).

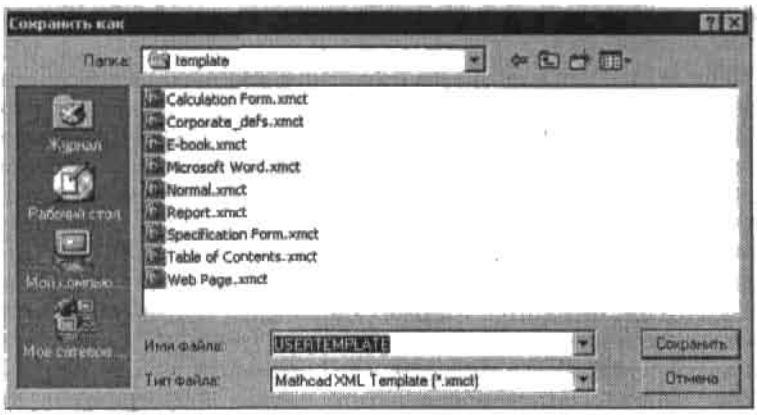


Рис. 1.18. Сохранение пользовательского шаблона

Если вы выполните описанную последовательность действий корректно, то ваш шаблон будет добавлен к списку встроенных, и в дальнейшем вы сможете иметь к нему доступ, например, с помощью все той же кнопки New (Новый) меню File (Файл).

Любой шаблон можно редактировать, подобно обычным документам Mathcad. Кстати, шаблон может быть сохранен не только в специальной папке, но и на любом месте диска. Правда, чтобы загрузить его, вам придется самостоятельно указать к нему путь с помощью меню Browse (Обзор) диалогового окна New (Новый). Кроме того, в качестве своеобразных шаблонов можно использовать и уже готовые XMCD- или MCD-документы, в том числе и из справочной системы программы.

1.6.3. Сохранение документа

Сохраняется документ в Mathcad стандартными для Windows способами.

- С помощью команды Save (Сохранить) меню File (Файл).
- Нажатием кнопки с изображением дискеты Save (Сохранить) панели Standard (Стандартные).
- Использованием сочетания Ctrl+S.

Если файл сохраняется впервые или сохранение его должно быть проведено под другим именем, в другом месте или в другом формате, то следует воспользоваться командой Save As (Сохранить как) меню File (Файл).

В Mathcad 12 имеются следующие форматы сохранения.

- Mathcad XML Document (*.xmcd). Документ сохраняется в созданном Mathsoft формате XML XMCD. Достоинство сохранения документа в этом формате заключается в том, что формат XMCD, в отличие от MCD, является открытым и, что немаловажно, простым. Открытость и простота формата должны способствовать развитию рынка ориентированных на Mathcad-пользователей «маленьких» приложений от сторонних разработчиков. Например, можно ожидать уже в скором времени появления плагина для Internet Explorer, с помощью которого можно будет просматривать документы Mathcad в сети без конвертирования их в HTML, которое не обеспечивает полной идентичности с оригиналом. Также вполне вероятно, что со временем документы Mathcad смогут распознавать, например, другие математические пакеты (подобно тому, как сейчас документы Macromedia Flash (формат SWF) могут быть созданы и отредактированы в таких приложениях, как Adobe Illustrator, CorelDRAW или 3ds max).

  То, что XMCD-формат ориентирован на развитие вспомогательного программного обеспечения, объясняет некоторые принципиальные изменения в нем по сравнению с форматом MCD. Так, графики, сгенерированные изображения и анимации в MCD не сохранялись как графические объекты. Сохранялась информация о том, что некоторый объект должен быть создан на основании некоторых данных. Когда же документ открывался Mathcad, то заново происходил обсчет и соответствующий объект рисовался с нуля. В принципе, точно так же происходит и в Mathcad 12. Однако XMCD-файл может содержать и не используемые явно Mathcad данные, которые описывают графические объекты как растровые картинки. Эти данные потенциально будут использоваться сторонним ПО для того, чтобы можно было визуализировать документ. Действительно, тот же плагин для Internet Explorer должен быть очень компактной и простой программой (иначе его никто не будет скачивать). Понятно, что при таких требованиях ввести в него возможность самостоятельного построения графика нереально. А вот визуализировать готовое растровое изображение в формате JPEG или PNG элементарно буквально одним вызовом стандартной функции операционной системы или браузера.

  Сохранять графические объекты как растровые картинки имеет смысл лишь в том случае, если вы рассчитываете, что ваш XMCD-документ будет открываться в какой-то сторонней программе. На данный момент поддерживающих XMCD-формат приложений почти нет, поэтому соответствующая возможность — это излишество. Чтобы растровое описание графических объектов не заносилось в XMCD-документ, необходимо активизировать настройку No Images (Без изображений) вкладки XML Options (Параметры XML) окна, открываемого командой File ▸ Properties (Файл ▸ Свойства). Если этого не сделать, то размер XMCD-документов будет больше (иногда — существенно).

  Если же вы все же решите сохранять растровое описание графических изображений, то можно выбрать, какой растровый формат для этого будет использован — PNG или JPEG (соответствующие настройки также расположены на вкладке XML Options (Параметры XML)). В последнем случае размер файла можно значительно уменьшить, правда, за счет потери качества изображений, которое регулируется в процентах от исходного настройкой Image Quality (Качество изображения).

Стороннее ПО, поддерживающее XMCD-формат, совсем не обязательно будет что-то вычислять. Скорее всего, вспомогательные приложения будут предназначены для визуализации документов Mathcad в необычных средах (например, браузере), а также для перевода документов в другие форматы (PDF, DOC или же форматы альтернативных математических пакетов). Поэтому в формате XMCD предусмотрена возможность сохранения результатов вычислений. Сам Mathcad, как и в случае, когда применяется формат MCD, всякий раз при открытии документа осуществляет вычисления заново. Сохраненные при создании XMCD-документа результаты вычислений Mathcad не учитывает. По этой причине заносить результаты вычислений в файл стоит лишь тогда, когда с ним будет работать какое-то стороннее приложение. Иначе это делать абсолютно бессмысленно и даже вредно, так как при этом увеличится размер файла.

Чтобы результаты вычислений заносились в XMCD-документ, установите флажок Save Numeric Results (Сохранить численные результаты) на вкладке XML Options (Параметры XML).

- Mathcad Compressed XML Document (*.xmcdz). Сжатый XMCD-формат. Удобен в том случае, если ваш документ достаточно объемный. При сохранении документа в этом формате происходит ZIP-сжатие генерируемого Mathcad XMCD-файла, что порой позволяет существенно уменьшить размер файла.
- Mathcad Worksheet (*.mcd). Документ сохраняется в битовом формате, который был единственным форматом сохранения документов Mathcad вплоть до 12 версии. В принципе, MCD-формат, будучи битовым, более компактен, чем текстовый XMCD-формат. Поэтому, если для вас не важна открытость по отношению к стороннему ПО, обеспечиваемая форматом XMCD, то документы лучше сохранять в формате MCD. Это даст заметную экономию в размере файла (в 2-3 и более раза — в зависимости от настроек).
- HTML File (*.htm). Формат веб-страницы. Сохранив документ в этом формате, вы сможете разместить его на собственной странице в Интернете, и ее можно будет просмотреть с помощью любого браузера. Особенность этого формата заключается в том, что все формулы будут сохранены в формате PNG как картинки, и для их сохранения будет создана отдельная папка.

  Сохранить документ в виде HTML-файла можно альтернативным способом, задействовав команду File ▸ Save As a Web Page (Файл ▸ Сохранить как веб-страницу). При этом откроется диалоговое окно Save as Web Page Options (Параметры сохраняемой веб-страницы), где вы можете указать некоторые параметры сохраняемой страницы, например изменить формат PNG, в котором по умолчанию сохраняются формулы и графические объекты, на JPEG.

  HTML-формат — это наиболее удобная форма распространения расчетов, сделанных в Mathcad. Достоинство этого подхода заключается в том, что структура документа сохраняется (в отличие от конвертирования в формат RTF). Правда, полное соответствие исходному документу достигается не всегда. Так, довольно посредственно сохраняются графики и текстовые области.
- Rich Text Format (*.rtf). Файл сохраняется в текстовом режиме, доступном прочтению любым текстовым редактором, и прежде всего Word. При этом формулы переводятся в векторные картинки, а текст остается доступным для редактирования. Большими недостатками этого формата является то, что структура документа в нем не сохраняется и все объекты отображаются одним столбцом. Обойти же такую трудность при создании текстовых отчетов можно, просто копируя фрагменты задачи из Mathcad в виде OLE-объектов или же используя HTML-формат.

- Mathcad Template (*.mct). Шаблоны Mathcad, сохраненные в битовом формате.
- Mathcad XML Template (*.xmct). Шаблоны Mathcad 12 формата XML.
- Mathcad 11, 2001i, 2001 Worksheet (*.mcd). Форматы предыдущих версий Mathcad. Необходимость их использования может быть связана, например, с тем, что вам придется обмениваться идеями и решенными задачами с коллегой, у которого нет новой версии программы.

По умолчанию Mathcad предлагает сохранять документы в формате XMCD, однако вы можете заменить его на битовый формат MCD либо на сжатый XMCDZ. Для этого выполните команду Tools ▸ Preferences (Инструменты ▸ Предпочтения). На вкладке Save (Сохранить) в списке Save Mathcad Files As (Сохранять Mathcad-файлы как) выберите интересующий вас формат.

1.6.4. Поиск в документе

Очень просто можно организовать в Mathcad поиск в документе определенного символа, слова или фразы. Чтобы найти интересующий вас фрагмент в большом документе, выполните следующую последовательность действий.

1. Выполнив команду Edit ▸ Find (Правка ▸ Найти) или нажав сочетание Ctrl+F, откройте специальное окно поиска Mathcad.
2. В строке Find What (Что найти) задайте искомый текст.
3. При необходимости произведите настройки особенностей поиска в списке параметров, расположенном ниже строки ввода.
   - Match whole word only (Выделять только целые слова). При поиске будет учтено только совпадение целых слов.
   - Match case (Учитывать регистр). При поиске два слова будут приняты за совпадающие, если одинаков регистр всех их символов.
   - Find in Text Regions (Искать в текстовых областях).
   - Find in Math Regions (Искать в математических областях).
4. Задать направление поиска можно с помощью переключателя Direction (Направление) (Up (Вверху) или Down (Внизу)).
5. Последовательно нажимая кнопку Find Next (Найти следующий), просмотрите все фрагменты, в которых встречается заданный текст.
6. Найдя нужный фрагмент, нажмите Cancel (Закончить). При этом курсор ввода будет перемещен к найденному объекту.

1.6.5. Автоматические замены

Если в документе должна быть заменена константа или переменная, то очень быстро и просто это можно сделать с помощью специальной возможности Mathcad. Чтобы ее реализовать, выполните следующую последовательность действий.

1. Задействуйте команду Edit ▸ Replace (Правка ▸ Заменить) или нажмите сочетание клавиш Ctrl+H для вызова специального диалогового окна Replace (Заменить).
2. В строке Find what (Что найти) появившегося диалогового окна задайте текст, подлежащий замене.

3. В строке Replace with (Заменить на) введите текст, на который требуется заменить определенный выше.
4. Настройте параметры поиска, если это необходимо (их описание приведено в предыдущем разделе).
5. С помощью последовательного нажатия нужного количества раз кнопки Find Next (Найти следующий) определите, какой именно фрагмент должен быть заменен.
6. Найдя объект, требующий замены, задействуйте команду Replace (Заменить).
7. Для замены всех фрагментов, удовлетворяющих заданным условиям, нажмите кнопку Replace All (Заменить все).
8. Для выхода из диалогового окна Replace (Заменить) нажмите Cancel (Закончить).

1.7. Справочная информация

В Mathcad можно получить справочную информацию из нескольких источников. Ссылки на них располагаются в меню Help (Помощь). В этом разделе мы кратко охарактеризуем назначение всех пунктов этого меню, а затем более подробно поговорим о работе со справочной системой и дадим некоторые рекомендации по использованию Ресурсов Mathcad (Mathcad Resources).

Всего в меню Help (Помощь) имеется 13 ссылок на различные объекты Mathcad (рис. 1.19). Рассмотрим их в том порядке, в котором они располагаются в меню.

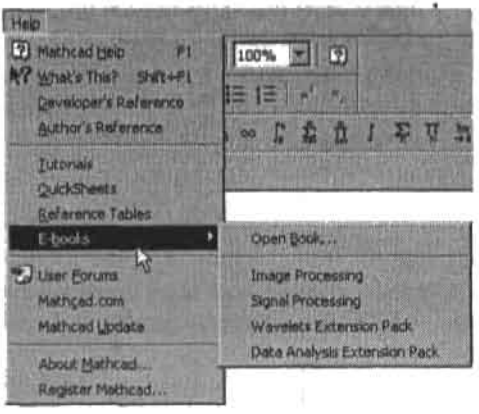


Рис. 1.19. Меню Help (Помощь)

- Mathcad Help (Справка Mathcad). Данная ссылка открывает окно справочной системы. Подробно о поиске необходимой информации с его помощью мы поговорим в следующем подразделе.
- What's This? (Что это такое?). Задействовав данный параметр, вы сможете получить контекстно-зависимую справочную информацию относительно указанного вами элемента.
- Developer's Reference (Справка для разработчиков). Эта ссылка открывает статьи справочной системы, предназначенной для опытных пользователей, создающих интегрированные приложения на основе Mathcad.

- Author's Reference (Справка для авторов). Очень небольшое справочное приложение, предназначенное для создателей собственных электронных книг.
- Tutorials (Учебники). Здесь вы можете получить наиболее общие сведения об особенностях работы с Mathcad. В данном разделе имеется несколько учебных статей (довольно неплохих, следует признать) по наиболее сложным и интересным возможностям программы.
- QuickSheets (Шпаргалки). Содержит сотни решенных задач из самых различных областей использования математики.
- Reference Tables (Справочные таблицы). Здесь вы можете получить общую справочную информацию, касающуюся характеристик материалов, узнать значения физических и математических констант, найти общий вид основных формул.
- E-books (Электронные книги). Пункт меню, позволяющий открыть сохраненные в виде файла электронные книги пользователя, а также встроенные электронные книги Mathcad, из которых вы можете почерпнуть полезные сведения о расширениях программы.
- User Forums (Форумы пользователя). Ссылка на официальный форум пользователей Mathcad, в котором проходит обмен идеями и выставляются варианты решения нестандартных задач.
- Mathcad.com. Сайт Mathcad, на котором можно найти массу полезной информации, касающейся возможностей программы, множество примеров решения задач из различных областей науки, построения интересных поверхностей. Также с сайта можно скачать встраиваемые в Mathcad электронные книги.
- Mathcad Update (Обновления Mathcad). С помощью этой команды можно подключиться к серверу компании Mathsoft и получить обновления программы.
- About Mathcad (О Mathcad). Небольшая статья о назначении и создателях программы.
- Register Mathcad (Зарегистрировать Mathcad). Ссылка на регистрационную страницу Mathcad в Интернете.

1.7.1. Справочная система

Окно справочной системы стандартного для Windows-приложений вида можно запустить либо нажав F1, либо задействовав кнопку Help (Помощь) панели Standard (Стандартные) (рис. 1.20).

Рис. 1.20. Кнопка Help (Помощь)

Справка в Mathcad является контекстно-зависимой, то есть содержание открывшегося окна зависит от обстоятельств, при которых оно было вызвано. Так, например, если вы используете оператор векторизации при вычислении какой-то функции от матрицы и вам интересно узнать, какие еще возможности открывает данный оператор, то просто выделите соответствующее выражение и нажмите F1. При этом будет открыта статья справочной системы Vectorize Operator (Оператор векторизации), в которой кратко, но содержательно излагаются основные моменты, связанные с заданием, использованием и возможностями рассматриваемого оператора (рис. 1.21).

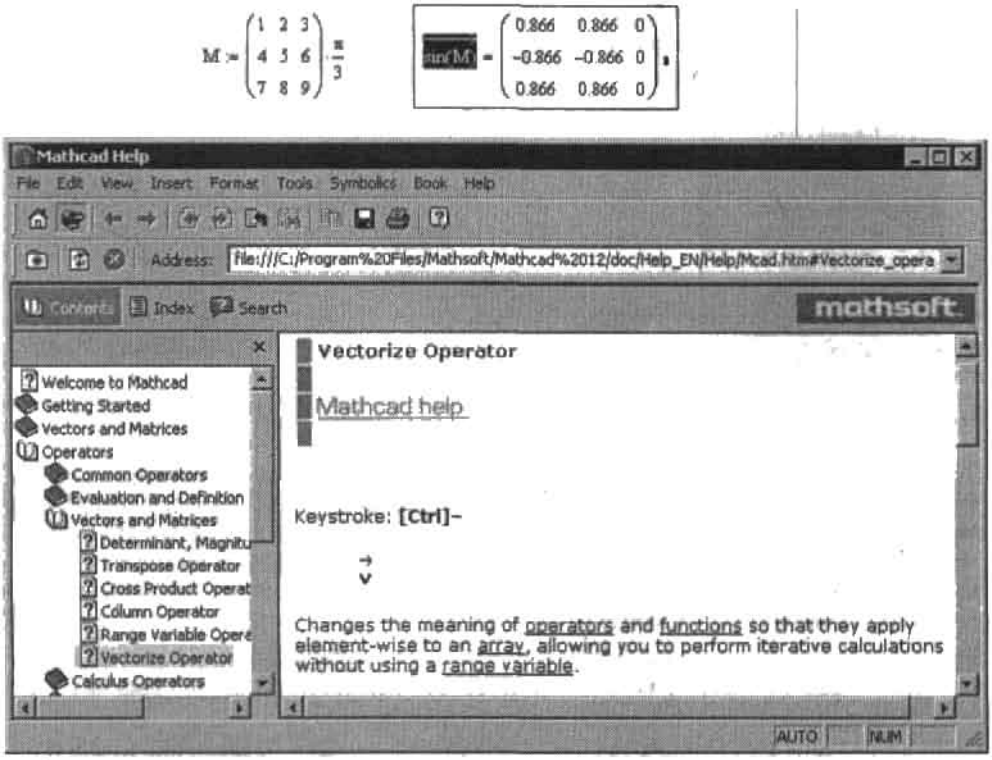


Рис. 1.21. Окно справочной системы Mathcad

Очень многие термины в статьях справочной системы представлены в виде ссылок на тематические разделы. Так, нажав ссылку functions текста статьи, представленной на рис. 1.21, мы откроем раздел Built-In functions (Встроенные функции), содержащий ссылки на описание всех встроенных функций Mathcad. Обязательно просматривайте ссылки, если термины, стоящие за ними, вам незнакомы или вы имеете о них лишь смутное представление. Это поможет вам быстро и качественно изучить материал.

В конце многих справочных статей располагается ссылка QuickSheet (Шпаргалка) (рис. 1.22). С помощью данной ссылки можно открыть документ с решенными задачами по теме справочной статьи. Изучать эти примеры намного полезней и проще, чем просто читать текст справки. Особенно это касается пользователей, не владеющих в достаточной степени английским языком.

Рис. 1.22. Ссылка на пример из шпаргалок

Другим часто встречающимся элементом справочных статей является ссылка на близкие по теме статьи Related Topics (Близкие статьи) (рис. 1.23).

Рис. 1.23. Ссылка Related Topics (Близкие статьи)

Эта ссылка приводит к отображению списка статей, в котором вам следует выбрать наиболее нужную или интересную тему (рис. 1.24). Щелкнув на ней левой кнопкой мыши, вы получите заинтересовавший вас раздел.

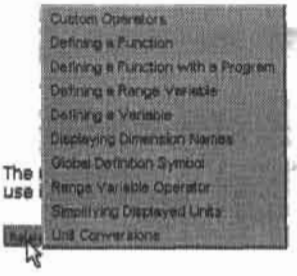


Рис. 1.24. Список близких по теме статей

Окно справочной системы Mathcad, как вы уже заметили, разделено, исходя из принятого в Windows стиля, на две части. Слева по умолчанию отображается содержание справочной системы (Contents), справа — текст выбранной статьи. Если левая часть окна мешает изучать справочный материал, то ее можно закрыть, нажав кнопку с крестиком, расположенную в верхней части окна. Чтобы затем снова получить возможность поиска материала, нажмите кнопку Contents (Содержание).

При необходимости вы можете переходить на другие страницы с помощью специальных кнопок Back (Назад), Forward (Вперед) и Home (Домой), расположенных на специальной панели инструментов вверху окна программы.

Контекстно-зависимый поиск далеко не всегда обеспечивает нахождение нужной информации. Чаще всего приходится самостоятельно искать необходимые сведения, используя кнопки Contents (Содержание), Index (Указатель) и Search (Поиск), расположенные в левой части окна программы.

Если вы не знаете точного названия нужной вам статьи, однако представляете, как приблизительно называется тематическая группа, к которой она относится, то вам нужно воспользоваться кнопкой Contents (Содержание). Все открывшиеся статьи разбиты на темы, которые отображаются в виде закрытых книг. Выбрав нужную вам тему, щелкните левой кнопкой мыши на соответствующей ей ссылке. При этом она раскроется, и вы должны будете найти среди множества статей данной тематики наиболее подходящую вам.

Так, например, если вы хотите найти справочный материал все о том же операторе векторизации, то, очевидно, прежде всего нужно открыть тему Operators (Операторы). Из предложенных тем вам нужно выбрать раздел Vectors and Matrices (Векторы и матрицы). В открывшемся списке матричных операторов вызовите уже знакомую нам статью Vectorize Operator (Оператор векторизации).

Вторая кнопка — Index (Указатель) открывает список справочных статей и групп близких по теме статей, расположенных в алфавитном порядке. Производить в нем поиск гораздо быстрее и удобнее, если вы точно знаете название необходимой вам справочной статьи. Так, с помощью данной кнопки сразу можно открыть статью Vectorize Operator (Оператор векторизации). Значительно облегчить поиск статьи можно, введя ее название в строку Type in the keyword to find (Введите ключевое слово для поиска). При этом система автоматически переместит отображение списка на нужный его фрагмент.

В некоторых случаях очень сложно определить даже тему нужного справочного материала. В этом случае следует обратиться к третьей кнопке окна справки Search (Поиск). Например, если вы хотите узнать о назначении панели Symbolic (Символьные), то введите в строку Type in the word(s) to search for (Задайте ключевое слово для поиска) следующий текст: Symbolic toolbar. Нажав Go, вы получите список статей, в которых встречалось это сочетание. Найти же среди них нужную уже совсем нетрудно (рис. 1.25).

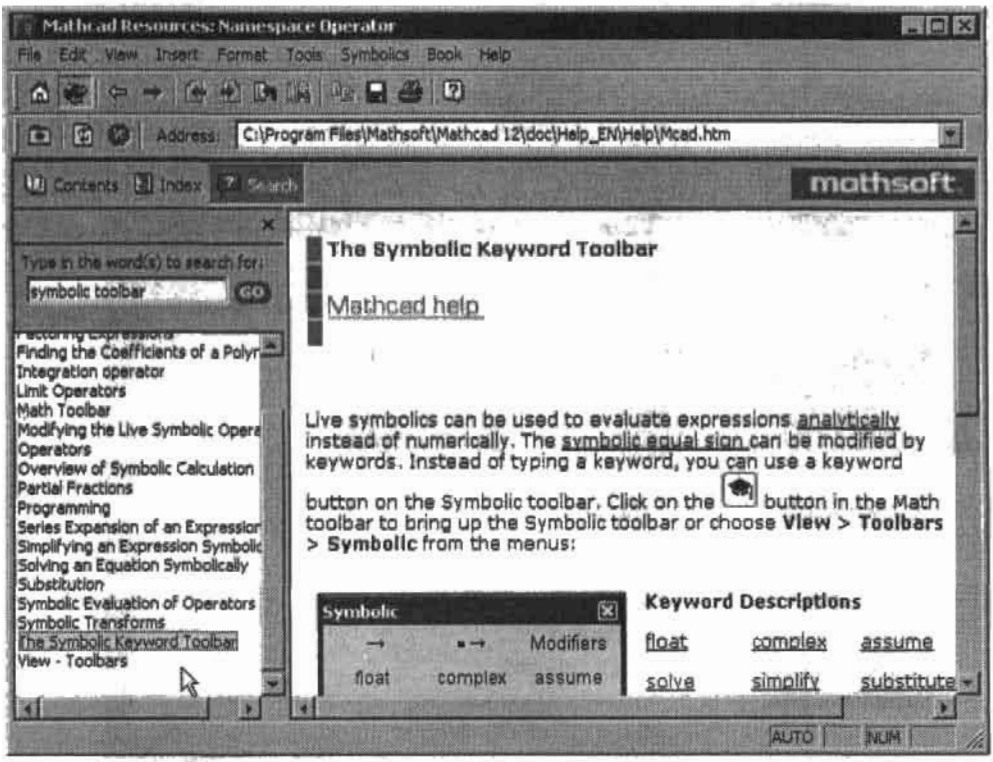


Рис. 1.25. Поиск справочных статей по ключевым словам

Справочная система Mathcad содержит не только описание различных элементов программы, но и довольно значительный объем чисто математической информации. При желании вы сможете найти подробное описание всех численных методов и математических функций, используемых в системе. В общем, если у вас возникли какие-либо трудности, независимо от того, относятся ли они к области программы или являются чисто математическими, смело обращайтесь к справочной системе Mathcad — и решение будет обязательно найдено!

1.7.2. Ресурсы Mathcad

Ресурсы (Resources) Mathcad являются важнейшим и полезнейшим приложением программы, позволяющим получать подробные сведения как о самой программе и особенностях проведения расчетов в ней, так и более общую информацию из различных разделов науки и техники. В их обширной базе данных можно найти примеры использования

большинства встроенных функций и операторов Mathcad, решенные задачи из самых различных областей приложения математики, многочисленные учебники и справочники.

Чтобы открыть окно Ресурсов Mathcad, воспользуйтесь кнопкой Go панели Resources (Ресурсы) либо выберите в предлагаемом списке интересующий раздел (рис. 1.26).

Рис. 1.26. Панель Ресурсов Mathcad

В зависимости от того, какой из тематических разделов вы выберете в списке панели Resources, перед вами откроется красочное окно приложения, содержащее несколько ссылок. Попробуем, например, обратиться к шпаргалкам Mathcad (рис. 1.27).

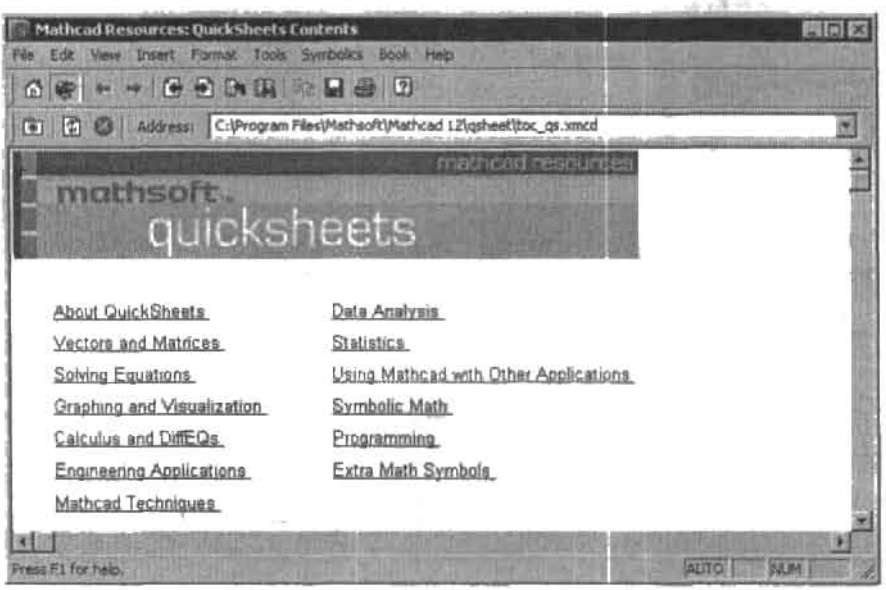


Рис. 1.27. Окно QuickSheets Ресурсов Mathcad

Шпаргалки (QuickSheets) — это распределенные по темам решенные создателями Mathcad разнообразные задачи. В Mathcad есть примеры по 13 темам. Вообще же Ресурсы Mathcad содержат сотни листов с решенными задачами. И в них отображены практически все важные возможности программы. Причем, как правило, сделано это очень интересно и в предельно доступной форме (правда, определенный минимум знания английского языка для чтения комментариев все же требуется). Все примеры приведены в виде обычных документов Mathcad, так что вы можете экспериментировать с ними так же, как с собственными задачами. Кстати, именно для этого в верхней части окна Mathcad Resources имеется отражение главного меню программы.

Изучая какую-нибудь тему, обязательно просматривайте соответствующие примеры из шпаргалок. Это необходимо в связи с тем, что в книге можно изложить лишь основополагающие принципы. Изучая же решенные задачи, вы сможете понять предмет гораздо глубже. Тем более это намного интереснее, чем просто читать справочник.

Открыв окно Reference Tables (Справочные таблицы), вы найдете список из более чем 30 справочных таблиц и схем по самым разным разделам математики, физики, химии и даже электроники. Если у вас возникнет необходимость найти определенную константу или формулу, смело обращайтесь к этому разделу ресурсов — и почти наверняка она там будет.

Весьма полезным для новичка может оказаться и раздел Tutorials (Учебники), содержащий несколько учебников по наиболее трудным, по мнению авторов программы, темам. И следует признать, что учебники эти написаны на весьма достойном уровне, правда, они требуют довольно неплохого знания английского языка.

Особенности навигации по Ресурсам Mathcad совершенно очевидны и, в принципе, ничем не отличаются от банального просмотра сайтов в Интернете. Так, переход между страницами осуществляется через гиперссылки, выделяемые в Mathcad подчеркиванием. Кроме того, аналогично тому же Internet Explorer, вверху окна Mathcad Resources имеется специальная панель, кнопки которой отвечают за различные команды, необходимые для работы с Ресурсами Mathcad. Вот эти кнопки.

- — Home (Начало). Нажав данную кнопку, вы перейдете на главную страницу Ресурсов Mathcad.
- — данная кнопка открывает Address Bar (Адресная панель) (рис. 1.28), предназначенную прежде всего для просмотра материалов, расположенных на интернет-сайтах.

Рис. 1.28. Address Bar (Адресная панель)

- — Back (Назад). Нажав эту кнопку, вы вернетесь на ранее открытую вами предыдущую страницу.
- — Forward (Вперед). Переход на уже открытую вами ранее страницу, расположенную логически позднее, чем данная.
- — Previous topic (Предыдущая статья). Нажав эту кнопку, вы перейдете к статье более общей, чем данная.
- — Next Topic (Следующая статья). Переход на расположенную в структуре Ресурсов Mathcad следующей статью. Соответствует нажатию ссылки. Если на странице имеется несколько ссылок, то откроется та статья, ссылка на которую расположена раньше других.
- — History list (Журнал). Данная кнопка открывает окно, содержащее список всех страниц Ресурсов Mathcad, на которых вы побывали при данной его загрузке. Используя его, вы сможете найти необходимый вам раздел, потратив минимум времени на его поиск.
- — Search (Поиск). Задействовав эту кнопку, вы откроете окно системы поиска (рис. 1.29). Очень полезная возможность, учитывая объемы информации, содержащиеся в Ресурсах Mathcad.
- — Copy (Копировать). Копирование выделенного фрагмента статьи.
- — Save As (Сохранить как). С помощью данной кнопки вы можете сохранить текущую статью в качестве самостоятельного документа.

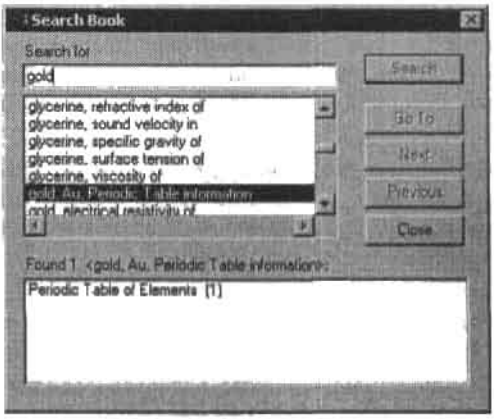


Рис. 1.29. Поиск информации в Ресурсах Mathcad

- — Print (Печать). Печать текущей статьи.
- — Help (Помощь). Переход к справочной системе Mathcad.

Так как статьи Ресурсов Mathcad являются, по сути, обычными XMCD-документами, то все расчеты и алгоритмы, приведенные в них, являются действующими. Поэтому при желании вы можете вносить в изучаемый вами пример любые изменения и дополнения (кстати, модифицированный фрагмент по умолчанию выделяется зеленым цветом), что позволит вам в нем разобраться гораздо лучше, чем при простом чтении текста. Влиять на особенности вычисления примера статьи, вставлять дополнительные функции и объекты, изменять многие параметры документа можно, используя команды главного меню окна Mathcad Resources (математические операторы вставляются в примеры командами соответствующих рабочих панелей самой программы) (рис. 1.30).

File Edit View Insert Format Tools Symbolics Book Help

Рис. 1.30. Главное меню окна Mathcad Resources

Главное меню окна Mathcad Resources практически полностью соответствует по своей структуре аналогичному объекту окна Mathcad. Из основных различий можно выделить наличие в нем меню Book (Книга), команды которого служат для форматирования статей Ресурсов Mathcad (рис. 1.31).

Команды меню Book (Книга) следующие.

- Annotate Book (Аннотировать книгу). При включении этого параметра станут доступными остальные команды рассматриваемого меню, позволяющие вносить изменения в уже существующие статьи.
- Highlight Changes (Окрашенные изменения). Если установлен данный флажок, то измененные формулы примеров будут выделяться другим оттенком.
- Save Section (Сохранить раздел). Задействовав эту команду, вы сможете сохранить внесенные в текущую статью изменения таким образом, что они будут отображаться и при последующих загрузках Ресурсов Mathcad.
- Save all Changes (Сохранить все изменения). Команда сохраняет изменения не только в текущем, но и во всех модифицированных ранее разделах.

- View Original Section (Видеть исходные разделы). Задействовав эту команду, вы сможете увидеть измененную статью в ее первоначальном виде, даже если модификация произошла не при данной загрузке. При этом все изменения сохраняются, и при последующих загрузках раздел будет отображаться с ними.
- View Edited Section (Видеть редактированный раздел). Команда, обратная предыдущей.
- Restore Section (Восстановить раздел). Задействовав эту команду, вы отмените все внесенные в текущий раздел изменения.
- Restore All (Восстановить все). Будут отменены все изменения, внесенные в любую из статей Mathcad Resources.

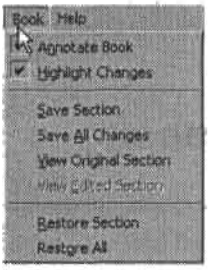


Рис. 1.31. Меню Book (Книга)

При необходимости вы можете копировать формулы и программы, приведенные в статьях Ресурсов Mathcad, в ваш документ. Зачастую такой подход позволяет не только сэкономить время, но и избежать ошибок. Найти же нужную вам задачу вы можете, используя систему поиска.

И, заканчивая наше краткое обозрение Ресурсов Mathcad, хотелось бы посоветовать: обращайтесь к ним всякий раз, когда будете изучать новую тему или вопрос. В сочетании с книгой это позволит вам получить исчерпывающие сведения об интересующей вас проблеме.

1.8. Электронные книги (E-Books) и пакеты расширений (Extension Packs) Mathcad

Электронные книги Mathcad (E-Books) представляют собой не что иное, как коллекцию Mathcad-документов, демонстрирующих возможности системы в решении задач по определенной тематике. В структуре электронной книги документы связаны между собой гиперссылками, что существенно облегчает навигацию по ней. В Mathcad имеются встроенные электронные книги, содержащие описание пакетов расширений Mathcad. Помимо них существует множество доступных в Интернете электронных книг, созданных самими пользователями. Большое собрание можно найти по адресу http://www.mathcad.com/resources/electronic_books/. Чтобы встроить электронную книгу в Mathcad, необходимо инсталлировать ее в папку Handbook (приблизительный адрес `C:\Program Files\Mathsoft\Mathcad 12\handbook`), после чего новый заголовок появится в списке панели Resources (Ресурсы) и в подменю E-Books (Электронные книги) меню Help (Помощь). При вызове электронной книги в окне Ресурсов Mathcad открывается ее содержание, в котором вы можете выбрать интересующую статью.

Принципы работы с электронными книгами в Mathcad абсолютно идентичны использованию статей Mathcad Resources. Чтобы вспомнить их, перечитайте подразд. 1.7.2.

Пакеты расширений (Extension Packs) дополняют Mathcad специализированным набором встроенных функций, использование которых ограничено лишь узкой областью. Так, например, пакет Image Processing Extension Packs содержит функции для обработки изображений, а пакет Signal Processing Extension Pack добавляет в систему функции создания цифровых фильтров и анализа сигналов. Дополнительные функции, инсталлированные с пакетом расширений, становятся доступными в окне Insert Function (Вставить функцию). Их список приводится в разделе, название которого совпадает с названием пакета расширений. На данный момент существует пять пакетов расширений, предоставляющих дополнительные возможности для обработки изображений, сигналов, анализа данных, проведения вейвлетных преобразований, комплексных расчетов и оптимизации. За дополнительную плату их можно приобрести на сайте http://www.mathcad.com/products/extension_packs/.

1.9. Mathcad Application Server (Сервер приложений Mathcad)

Следует признать, что для подавляющего большинства пользователей Mathcad является прежде всего инструментом для решения стандартных задач, а не средой разработки узкоспецифичных алгоритмов. Так, с помощью Mathcad рядовой студент может справиться с любыми поставленными перед ним проблемами, которые, разумеется, не выходят за рамки вузовской программы по высшей математике, например, решить систему уравнений, определить экстремум функции или провести статистическую обработку результатов лабораторной работы. То же самое касается инженерных, экономических и других задач — каждая отрасль науки базируется на давно отработанных расчетных схемах и моделях, которые оказываются эффективными в 90 % случаев и не требуют каких-либо дополнений или улучшений. Однако, чтобы реализовать их в Mathcad, необходимо, во-первых, знание самой программы, во-вторых, четкое понимание алгоритма, лежащего в основе работы той или иной модели. Для некоторых же пользователей стоимость самого пакета может и вовсе оказаться неоправданно высокой для целей, ради которых появилась необходимость его приобретения. Возникает вопрос, зачем покупать программу, самостоятельно программировать алгоритм и тратить время на его отладку, если для большинства тривиальных задач из различных областей такие программы давно созданы в Mathcad и протестированы? В данном случае для пользователя гораздо важнее доступ к ним, чем понимание того, каким способом можно решить поставленную задачу в Mathcad.

Именно для обеспечения доступа к интерактивным приложениям Mathcad в Интернете компания Mathsoft представила в 2003 году пакет Mathcad Application Server. Используя стандартный веб-браузер, пользователь может работать с Mathcad-документами, находящимися на сервере Mathcad Application Server, даже если Mathcad не установлен на его собственном компьютере. По сути, такие рабочие документы, опубликованные на сервере другими пользователями, являются одновременно примерами и шаблонами для решения разного рода задач. Огромное их количество можно найти по ссылкам http://twt.mpei.ac.ru/ochkov/VPU_Book_New/mas/index.html и http://mas.mathsoft.com/mas/. Взаимодействие с приложением осуществляется с помощью сетевых элементов интерфейса (WebControls) — текстовых полей, кнопок, списков, переключателей и флажков. Рассмотрим подробнее, как это происходит.

Предположим, перед вами стоит задача определить, принадлежат ли результаты двух серий экспериментов одной выборке. Для этого зайдите на страницу первого ресурса, указанного выше, в разделе «Аналитическая химия» перейдите по ссылке «Проверка принадлежности двух серий экспериментов к одной выборке». Перед вами откроется окно Mathcad-документа, в текстовых полях которого укажите вместо имеющихся собственные данные и нажмите кнопку Recalculate (Пересчитать). Запрос будет немедленно обработан сервером Mathcad Application Server аналогично тому, как это сделала бы и сама программа, и результаты вычислений сразу отобразятся в этом же окне. Mathcad Application Server поддерживает все встроенные функции Mathcad и позволяет сохранять исходный алгоритм в неизменном виде, поскольку сторонний пользователь имеет доступ только к тем полям, которые были определены автором документа.

Вы также можете поместить собственные Mathcad-разработки на Mathcad Application Server, обеспечив к ним широкий доступ в Интернете. Чтобы подготовить документ к публикации, прежде всего в нем необходимо заменить все операторы присваивания на сетевые элементы управления (WebControls). Для этого задействуйте команду Insert ▸ Control ▸ Web Control (Вставить ▸ Элементы управления ▸ Сетевые элементы управления). Перед вами откроется окно мастера установки сетевых элементов управления Web Control Setup Wizard, позволяющее задавать параметры вводимых в документ элементов. Малопонятные для рядового пользователя алгоритмы можно скрыть (о том, как это делается, рассказано в гл. 19). Не следует забывать и о комментариях, которые помогут другим пользователям разобраться в сути решения задачи. Связавшись с администратором сайта (например, одного из упомянутых выше), вы сможете поместить Mathcad-документ на сервер. Подробно о том, как создать интерактивное Mathcad-приложение, рассказывается на http://twt.mpei.ac.ru/ochkov/VPU_Book_New/mas/From_WorkSheet_to_WebSheet.html.

1.10. Сообщения об ошибках

Если по той или иной причине Mathcad не сможет произвести расчет некоторой формулы, то она будет окрашена в красный цвет и ниже нее появится характерное (желтого цвета) информационное сообщение о причинах возникшей трудности. Так, например, если вы попытаетесь произвести неопределенную операцию деления на нуль, Mathcad выдаст следующее сообщение: Divide by zero in function evaluation (При вычислении функции происходит деление на нуль) (рис. 1.32).

Рис. 1.32. Сообщение об ошибке

В большинстве случаев содержащейся в сообщении краткой характеристики причин ошибки бывает вполне достаточно, чтобы ее устранить. Однако если вы не сможете разобраться в смысле того или иного сообщения, то нажмите F1. При этом будет открыта статья справочной системы, посвященная причинам и способам разрешения возникшей ошибки.

Глава 2. Вычисления и типы данных

Эта глава посвящена основным принципам задания объектов и проведения вычислений в Mathcad. При ее написании авторы руководствовались принципом, что изложенной в ней информации должно быть достаточно для изучения большинства последующих специализированных глав независимо друг от друга. Приводимые в ней сведения являются базовыми, поэтому от того, как вы их усвоите, во многом зависит успешность изучения остального материала. По этой причине стоит прочитать данную главу особенно внимательно.

2.1. Типы данных в Mathcad

В Mathcad совсем немного типов данных по сравнению с универсальными языками программирования — всего три. Кратко охарактеризуем их (более детально они будут описаны позже).

- Числа (как действительные, так и комплексные). Все числа Mathcad хранит в одном формате (с плавающей точкой двойной точности), не разделяя их на целые и действительные. На одно число выделяется 64 бита. При этом десятичная часть (мантисса) не может превышать по длине 17 знаков, а порядок должен лежать между –307 и 307. Комплексные числа на уровне реализации представляют собой пару действительных чисел. При этом во многих видах расчетов число воспринимается как комплексное, даже если у него нет мнимой части. Описанные особенности чисел в Mathcad касаются только численных расчетов. При работе в символьном режиме совершенно другие уровни точности.
- Строки. В общем случае любой текст, заключенный в кавычки. На практике строки используются в основном для задания сообщений об ошибках, возникших при работе программ на языке Mathcad.
- Массивы. К ним относятся матрицы, векторы, тензоры, таблицы — любые упорядоченные последовательности элементов произвольного типа. К данным этого типа можно отнести и ранжированные переменные. Очень подробно принципы задания массивов и особенности работы с ними описаны в гл. 3.

В отдельную группу следует выделить так называемые размерные переменные, то есть единицы измерения, имеющие огромное значение в науке и технике.

В Mathcad нет логического типа данных. Для обозначения истины и лжи логическими операторами и функциями используются числа — 0 и 1.

2.2. Задание переменных и функций

В данном разделе мы рассмотрим основные вопросы, связанные с особенностями задания переменных и функций в Mathcad. Тому, как использовать данные элементы в расчетах, будет посвящен следующий раздел.

2.2.1. Задание переменных

Переменная — это именованный объект данных. Используя ее имя, можно обращаться к соответствующему объекту из любого участка документа, расположенного ниже или правее выражения задания переменной. Переменные помогают делать расчеты более простыми, понятными и компактными, поэтому без них не обходится решение ни одной неэлементарной задачи.

Чтобы определить некоторую переменную, выполните следующую последовательность действий.

1. Наберите имя переменной. В общем случае оно может состоять из произвольного количества практически любых символов. Правда, некоторые ограничения при задании имени все же существуют, и их мы обсудим в конце данного подраздела.
2. Введите оператор присваивания. Сделать это можно либо нажатием кнопки Definition (Присваивание) панели Calculator (Калькулятор) или Evaluation (Вычисления) семейства Math (Математические), либо с помощью сочетания клавиш Shift+«:». Второй вариант предпочтительнее с точки зрения быстроты и удобства, поэтому, так как к присвоению значений вам придется прибегать при решении практически любой задачи, с самого начала используйте лучше его. В том случае, если переменная определяется в документе впервые, вы можете нажать клавишу простого равенства («=»). При этом данный оператор, выполняющий в Mathcad функции оператора численного вывода, будет автоматически заменен оператором присваивания. Впрочем, в некоторых случаях этот вариант может быть неприемлем в связи с тем, что отдельные буквы и выражения зарезервированы в Mathcad. Так, например, не получится таким образом присвоить значение переменной, определенной с помощью греческой Ω, так как этой буквой обозначается размерность электрического сопротивления Ом. В случае таких переменных нужно использовать стандартный метод определения.
3. На место черного маркера, появившегося справа от оператора присваивания, введите значение вашей переменной.

В общем случае значение переменной может быть определено как число, матрица либо строка.

Чаще всего переменную приходится определять как число. Сделать, однако, это можно по-разному. Наиболее простой вариант — непосредственно присвоить переменной значение, равное некоторому действительному или мнимому числу.

Пример 2.1. Присваивание переменной численного значения

$$x := 12 \qquad y := \frac{3}{7} + \frac{2}{5}\cdot i$$

Можно определить переменную через число, заданное как некоторое конкретное значение функции или алгебраического выражения.

Пример 2.2. Определение переменной через значение функции

$$x := \pi$$

$$\text{VARIABLE} := \sin(x) + \frac{1}{x}$$

Чтобы присвоить переменной значение матрицы или вектора, последние должны быть просто введены в правый маркер оператора «:=». О том, как это сделать, мы поговорим в гл. 3. К определениям переменной данного типа можно также отнести и задание вектора значений с помощью оператора ранжированной переменной (range variable) (о нем также читайте в гл. 3).

Пример 2.3. Присваивание переменной матричного значения

$$M := \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

И, наконец, переменная может быть определена как строка. При этом ее значение обязательно должно быть взято в кавычки.

Пример 2.4. Определение переменной как строки

$$\text{String} := \text{"Mathcad"}$$

Значение переменной в Mathcad можно и переопределить: для этого операцию нового присваивания нужно провести правее или ниже старого присваивания.

Человека, впервые столкнувшегося с выполнением расчетов с помощью компьютера, вид оператора присваивания может несколько смутить. Это связано с тем, что в традиционном математическом синтаксисе функции, которые он выполняет, возложены на обычный знак «=». Ситуация эта, вообще говоря, не совсем характерна для системы Mathcad, разработчики которой стремились максимально близко следовать общепринятым традициям оформления «бумажной» математики, и связана она с попыткой найти компромисс между этими традициями и нормами программирования. Почему же нельзя использовать один оператор «=» и для присваивания, и для вывода значения, и для указания равенства двух объектов данных или выражений? Дело в том, что программа лишена логики — поэтому понять из контекста, какие функции в каждом конкретном случае выполнял бы оператор «=», она не смогла бы. По причине этого для каждой операции, которая на бумаге обозначается обычным «=», в Mathcad имеется индивидуальный оператор. Знак же «:=» довольно традиционен: он является оператором присваивания, например, в языке программирования Pascal. Однако в том случае, если вам нужно подготовить отчет или традиционный вид оператора присваивания для вас более удобен, тип его отображения можно изменить. Сделать это можно двумя способами.

Если вы хотите изменить вид оператора в конкретной формуле, то выполните по нему щелчок правой кнопкой мыши. При этом откроется контекстное меню формулы, в котором следует выбрать всплывающее меню View Definition As (Отображать присваивание как). В данном меню нужно переставить флажок из пункта Colon Equal (Равенство с двоеточием) в пункт Equal (Равенство).

В том случае, если все присваивания в документе должны быть отображены с помощью простого равенства, выполните команду Tools ▸ Worksheet Options ▸ Display (Инструменты ▸ Параметры документа ▸ Отображение). Здесь в списке Definition (Определение) выберите строку Equal (Равенство). При этом вы измените принятые по умолча-

нию установки документа, и все вводимые в дальнейшем определения будут отображаться как «:=».

Кстати, при выделении формулы, оператор присваивания в которой представлен в виде простого равенства, отобразится именно определенная по умолчанию его стандартная форма. Эта особенность необходима для того, чтобы существовала возможность отличить оператор присваивания от оператора глобального определения, логического равенства или численного вывода, которые могут принимать ту же форму.

Присвоить переменной значение в виде какого-то буквенного выражения можно только в том случае, если все символы или их сочетания, которые в нем используются, определены выше как конкретные числа, матрицы или строки (или же являются системными переменными). Иначе неопределенный символ будет выделен красным цветом и система выдаст сообщение об ошибке: This variable is undefined (Эта переменная не определена).

2.2.2. Функции

Функции в Mathcad делятся на две группы:

- функции пользователя;
- встроенные функции.

Техника использования функций обоих типов абсолютно идентична, а вот задание отличается принципиально.

Задание функций пользователя

Особенности определения функций пользователя (проще говоря, функций произвольного вида) в Mathcad полностью совпадают с принятыми в математике правилами. Для этого необходимо выполнить следующую последовательность действий.

1. Введите имя функции. В общем случае оно может быть совершенно произвольным, хотя определенные ограничения все-таки имеются. О них мы поговорим немного позже.
2. После имени функции следует ввести пару круглых скобок, в которых через запятую нужно прописать все переменные, от которых зависит функция. Задать функцию с параметром можно только в том случае, если ему выше присвоено конкретное числовое значение. Иначе система выдаст уже знакомое нам сообщение об ошибке: This variable is undefined.
3. Введите оператор присваивания «:=».
4. На месте черного маркера справа от введенного оператора присваивания задайте вид вашей функции. В выражение определяемой функции могут входить как непосредственно переменные, так и другие встроенные и пользовательские функции.

Пример 2.5. Задание функции пользователя

$$a := 1 \qquad b := 2$$

$$f(x,y) := \sin\left(\frac{a}{x} + \frac{b}{y}\right) \qquad FUNCTION(z) := \frac{z}{a+b}$$

$$userFunction(x,y,z) := \frac{x}{y \cdot z} + a \cdot f(x,y) + b \cdot \cos(FUNCTION(z))$$

Встроенные функции

Встроенные функции — это функции, заданные в Mathcad изначально. Поэтому, чтобы их использовать, достаточно просто корректно набрать имена функций с клавиатуры. Впрочем, существуют и другие способы вставки нужной встроенной функции. Наиболее распространенные из них можно ввести с панели Calculator (Калькулятор). К таким функциям относятся синус, косинус, тангенс, натуральный и десятичный логарифмы, экспонента. Для того же, чтобы задать все остальные встроенные функции Mathcad, нужно открыть специальное окно Insert Function (Вставить функцию). Проще всего это можно сделать нажатием одноименной кнопки панели Standard (Стандартные) с изображением стилизованного знака функции (рис. 2.1).

Рис. 2.1. Кнопка Insert Function (Вставить функцию) меню Standard (Стандартные)

Также, для того чтобы вызвать данное окно (рис. 2.2), можно использовать сочетание клавиш Ctrl+Shift+F или Ctrl+E. И, наконец, ссылка на него имеется в меню Insert (Вставка).

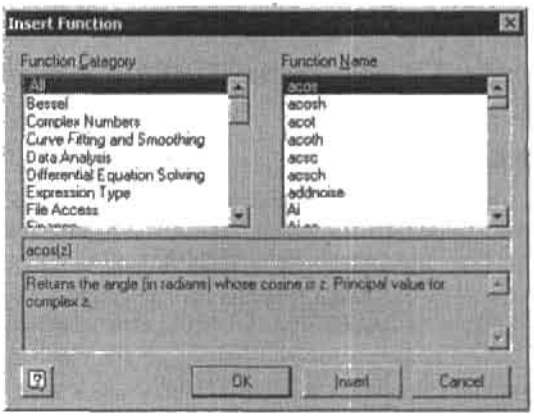


Рис. 2.2. *Окно* Insert Function (Вставить функцию)

Так как число встроенных функций Mathcad весьма значительно (несколько сотен), для удобства они распределены по тематическим группам. Их список, организованный в алфавитном порядке, расположен в окне Function Category (Категория функций). Всего в Mathcad 32 тематические группы функций.

При выборе определенной категории функций ее содержание отобразится в окне Function Name (Имя Функции). Чтобы ввести нужную функцию, выделите ее в списке с помощью мыши или клавиш управления курсором и нажмите OK (или лучше дважды щелкните на ней мышью).

По умолчанию в окне Function Name (Имя функции) отображается полный список всех встроенных функций, что соответствует категории All (Все). Производить поиск в полном списке несколько быстрее и удобнее, если вы приблизительно знаете написание имени нужной вам функции.

На окне Insert Function (Вставить функцию) имеется специальная зона, в которой отображается текст описания выбранной функции. Так, для первой функции списка

All acos (арккосинус) читаем: Returns the angle (in radians) whose cosine is z. Principal value for complex z (Возвращает угол (в радианах), косинус которого равен z. Действительная часть для комплексного z).

В том случае, если вам нужна более полная информация о некоторой функции, нежели дает сжатое сообщение окна Insert Function (Вставить функцию), вы можете обратиться к справочной системе Mathcad. Для этого вам нужно, выделив функцию, информацию о которой необходимо найти, нажать специальную кнопку Help (Помощь) в левом нижнем углу окна. При этом будет открыта статья справочной системы, в которой имеется упоминание о данной функции.

При вводе встроенных функций с клавиатуры следует помнить, что Mathcad различает регистр символов. Поэтому, если обычную функцию, образованную только строчными символами, вы введете с большой буквы, она распознана не будет. И, наоборот, функция, которая вводится с помощью окна Insert Function (Вставить Функцию) как последовательность прописных букв, аналогично должна быть набрана и вами.

2.2.3. Тонкости задания имен переменных и функций

Задавать имена переменных и функций в Mathcad можно практически произвольным образом. Однако некоторые ограничения, о которых нужно иметь представление, все же имеются. Изложению этого вопроса посвящен данный раздел.

Перечислим для начала, какие символы можно использовать при определении имени функции или переменной.

- Строчные и прописные буквы. Причем, следует помнить, что система Mathcad воспринимает маленькую и большую букву как различные символы (то есть имеется чувствительность к регистру). Кроме того, как различные будут прочитаны и буквы разных стилей.
- Числа от 0 до 9, если они стоят не в начале имени. Если цифра стоит в начале имени, то программа воспримет выражение как комплексное число (если за цифрой следует i или j), как бинарное, восьмеричное или шестнадцатеричное число (если за цифрой следует соответственно буква b, o или h), как умножение числа на переменную (во всех, кроме перечисленных ранее, случаях).
- Греческие буквы.
- Символы бесконечности (данный символ может быть только в начале имени), штриха (Ctrl+F7), подчеркивания, процента.
- Нижний индекс.

При задании последнего имеется некоторая тонкость. Дело в том, что для того чтобы задать индекс, не несущий математического смысла элемента некоторого массива, нельзя использовать клавишу «[» или соответствующую ей команду меню Matrix (Матричные). Для задания простого текстового индекса нажмите, поставив курсор в конец слова, клавишу «.». При этом курсор опустится на полстроки вниз, и вы сможете набрать текст индекса. Например:

$$X_{variable} := \sin(\pi)$$

Существует два варианта отображения текстового индекса. Чтобы выбрать один из вариантов, зайдите в контекстное меню формулы (щелкнув правой кнопкой мыши, предварительно поместив курсор на текст индекса). В открывшемся меню выберите команду

View Subscript as (Видеть нижний индекс как). В появившемся меню вам нужно определиться между двумя типами отображения.

- Large Subscript (Большой индекс). Параметр, определенный по умолчанию. Положение строчных букв индекса соответствует уровню нижней границы текста имени переменной или функции, к которому он относится.
- Small Subscript (Маленький индекс). Нижняя граница текста переменной или функции при таком типе отображения соответствует серединам прописных букв индекса.

Чтобы изменить особенности отображения текстового индекса глобально, нужно обратиться к меню Literal Index вкладки Display (Отображение) окна Worksheet Options (Параметры документа), которое открывается одноименной командой меню Tools (Инструменты).

Синтаксис при определении имени переменной или функции имеет следующие ограничения.

- Все буквы в имени должны иметь одинаковые стиль и шрифт.
- Имя не может содержать арифметических или любых других операторов.
- В идентификатор не могут входить пробельные символы (пробелы, знаки табуляции, переноса строк и пр.).
- Имя не может начинаться с цифры.
- Имена функций пользователя не должны совпадать с именами встроенных функций, поскольку это приведет к их переопределению. Например:

$$\sin(x) := 2\cdot\cos(x)$$

$$\sin(\pi) = -2$$

- Так как Mathcad не различает имен переменных и функций, то нельзя сначала задать функцию f(x), а потом переменную f (или наоборот), поскольку это приведет к неопределенности по причине переопределения одной из величин.
- Если вы переопределяете уже существующую функцию, то в новое выражение функции не должна входить ссылка на ее старую версию (например, f(x):=f(x)+2). Это связано с тем, что при этом система посчитает, что функция вызывает сама себя рекурсивно. Это приведет к тому, что при активации функции возникнет бесконечный цикл рекурсивных вызовов и, соответственно, произойдет сбой.

Очень часто приходится задавать имена переменных и функций на русском языке. Сделать это можно, перейдя на один из установленных у вас на компьютере кириллических шрифтов. Для этого нужно использовать специальный список панели Formatting (Форматирование) (точно так же, как в любом другом Windows-приложении).

Иногда качественное оформление документа требует использования в имени переменной или функции запрещенных символов. Обойти в таких случаях правила синтаксиса можно, используя два пути.

Во-первых, можно взять имя в квадратные скобки (вводятся сочетанием Ctrl+Shift+J). При этом, независимо от того, какие символы и в какой последовательности входят в него, оно будет восприниматься корректно. Например:

$$[\infty + \infty] := e^{500} \qquad [\infty + \infty] = 1.404\times 10^{217}$$

Во-вторых, если наличие квадратных скобок вас не устраивает, можно попробовать «обмануть» систему с помощью более сложного метода. Суть его заключается в пере-

ходе в текстовый режим ввода информации. Так, например, чтобы определить функцию, именем которой является символ возведения в степень ^, выполните следующую последовательность действий.

1. Сначала задайте какой-нибудь допустимый по правилам символ, например букву A.
2. Затем с помощью специального сочетания клавиш Ctrl+Shift+K перейдите в текстовый режим. Курсор ввода при этом окрасится в красный цвет.
3. Введите символ ^.
4. Сотрите введенную букву A.
5. Нажав повторно сочетание Ctrl+Shift+K, перейдите в обычный формульный режим.
6. Далее введите скобки, задайте переменные, от которых зависит функция, и сделайте присваивание соответствующему ей выражению.

В результате получим:

$$^{\wedge}(x, y) := x^{y} \qquad ^{\wedge}(3, 3) = 27$$

Ввести специальные символы в идентификаторы можно, скопировав их из документа, в котором они уже набраны. Среди шпаргалок Mathcad (Quicksheets) есть документ, в котором набран весь шрифт Symbol (это стандартный в Windows шрифт, содержащий специальные символы). Называется данный документ Extra Math Symbols (Дополнительные математические символы). Отсюда можно скопировать все важнейшие математические знаки.

2.2.4. Особенности использования идентификаторов существующих объектов

Представьте, что вы работаете с созданным не вами документом Mathcad. Это может быть электронная книга, пакет расширения или просто полезный алгоритм. Если документ большой или детали его реализации вас не интересуют, то вы вряд ли будете досконально его изучать. Кроме того, подобное изучение не всегда возможно, так как разработчики зачастую скрывают реализации алгоритмов, оставляя доступным лишь интерфейс созданного приложения. С учетом этого вполне возможно, что вы используете при задании собственной функции или переменной тот же идентификатор, какой был применен автором применяемого вами алгоритма. Это приведет к тому, что потенциально полезная функция или параметр окажутся недоступными из-за произошедшего переопределения.

Даже если вы не применяете чужих разработок, то проблема со случайным переопределением не исчезает. Дело в том, что в Mathcad около 400 встроенных функций, достаточно много встроенных переменных, констант и размерностей (которые являются разновидностью встроенных переменных). Поэтому вероятность того, что вы, создавая переменную или функцию, перекроете какой-то предопределенный объект, очень высока. Например, если дать переменной имя A, то станет недоступной размерность «ампер», если ее назвать N, то перекроется размерность «ньютон».

Трудно не согласиться, что проблема случайного переопределения функций и переменных довольно серьезна. В предыдущих версиях Mathcad был только один способ ее преодолеть — проявлять предельную внимательность. Однако в Mathcad 12 все стало гораздо проще. Теперь, если вы создадите переменную с именем, совпадающим с уже

известным системе идентификатором, то оно будет подчеркнуто волнистой зеленой линией и появится сообщение об ошибке (рис. 2.3).

Рис. 2.3. Переменная rank переопределяет встроенную функцию ранга матрицы rank

Важно отметить, что Mathcad лишь информирует о происходящем переопределении, но не блокирует его. Если вы не собираетесь использовать элемент, который перекрывает создаваемая вами переменная или функция, то соответствующее сообщение можно просто проигнорировать.

По умолчанию выводятся сообщения о переопределении встроенных функций, констант и размерностей. Если переопределяется встроенная переменная, то сообщение не появляется. Это связано с тем, что задание данных переменных можно осуществлять как из специального меню, так и (техничнее) простым их переопределением. Из создаваемых пользователем элементов сообщается о переопределении функций и переменных, хранящих объекты данных скалярных типов (то есть строки или числа). Если значением переменной является матрица или вектор, то сообщение об ее переопределении не появится (прочитав гл. 3, вы поймете почему).

То, как Mathcad осуществляет контроль над переопределением, весьма разумно. Однако вы можете изменить правила этого контроля «под себя». К примеру, если вы никогда не используете в расчетах размерности, то вам совершенно необязательно знать, что происходит переопределение какой-то из десятков встроенных в Mathcad размерностей. За особенности контроля над переопределением отвечают настройки вкладки Warnings (Предостережения) окна Preferences (Предпочтения), открываемого одноименной командой меню Tools (Инструменты). Главными элементами данной вкладки являются два меню: Built-In (Встроенные) и User Defined (Пользовательские). В меню Built-In можно указать, нужно ли вести контроль за переопределением встроенных функций (пункт Functions), размерностей (пункт Units), констант (пункт Constants), переменных (пункт Variables). В меню User Defined можно определить, необходимо ли отслеживать переопределение пользовательских функций (пункт Functions), переменных со скалярными значениями (пункт Scalar Variables), переменных, значениями которых являются матрицы или вектора (пункт Vectors and Matrices). Полностью отключить отслеживание переопределений можно, сняв флажок Show warnings on redefenitions of: (Отображать предостережения при переопределениях:).

Описанный выше контроль за переопределением предназначен исключительно для информирования пользователя. Однако можно ли действительно решить проблему перекрывания идентификаторов? К примеру, можно ли добиться того, чтобы в документе использовалась переменная N и при этом была доступна размерность N (ньютон)? В Mathcad 12 появился новый оператор, который делает это возможным. Называется он оператором пространства имен (Namespace operator).

Пространство имен — это классическое понятие программирования, которое появилось на заре развития объектно-ориентированных методов для снижения риска конфликта идентификаторов при написании объемных программ. Суть этого понятия заключается в том, что программа разделяется на автономные блоки — пространства имен. Переменные и функции напрямую «видят» только те элементы, которые от-

носятся к тому же пространству имен, что и они сами. Если же необходимо обратиться к переменной или функции другого пространства имен, то оно должно быть явно адресовано. Если отдельные блоки программы малы и каждый из них создается одним программистом, то использование пространств имен сводит вероятность конфликта идентификаторов практически к нулю.

Использование оператора Namespace дает возможность имитировать пространства имен в Mathcad. Чтобы задействовать этот оператор, нужно проделать следующие шаги.

1. Введите имя переменной или функции.
2. Поставив курсор ввода правее последней буквы имени, нажмите Ctrl+Shift+N. При этом появится оператор Namespace, который представляет собой нижний индекс с квадратными скобками. Например:

$$\sin_{[]}$$

3. В квадратных скобках укажите, к какому пространству имен относится данная переменная или функция. Всего в Mathcad четыре пространства имен:
   - mc — к этому пространству имен принадлежат все встроенные функции, переменные и константы;
   - unit — пространство имен единиц измерения;
   - user — к этому пространству относятся пользовательские функции, реализованные в виде встраиваемых в Mathcad компонентов (к примеру, пакеты расширений);
   - doc — функции и переменные, заданные непосредственно в документе. Если функция или переменная несколько раз переопределялась, то использовано будет последнее определение. Если пространство имен не указано, то первым по умолчанию на предмет наличия переменной или функции просматривается данное пространство.

   К примеру, чтобы вычислить значение синуса на основании встроенной функции, нужно набрать:

$$\sin_{[mc]}\left(\frac{\pi}{3}\right) = 0.866$$

Оператор пространства имен можно использовать только в численных расчетах. При проведении подсчета аналитически наличие данного оператора приведет к ошибке.

Пример 2.6. Использование оператора пространства имен

Зададим переменную с именем N:

$$N := 5$$

Переменная N перекрывает размерность N (ньютон). Поэтому задать переменную с такой размерностью без использования оператора Namespace не получится:

$$Force := 10N \qquad Force = 50$$

Чтобы задать переменную с размерностью силы, нужно посредством оператора Namespace указать, что N относится к пространству имен единиц измерения unit:

$$Force := 10N_{[unit]} \qquad Force = 10\ N$$

2.3. Проведение расчета численно

При работе в Mathcad нужно очень четко понимать, что в программе реализовано два принципиально разных подхода к вычислениям — численный и символьный. Более традиционный и простой тип расчета — численный. Он характеризуется тем, что подсчет значения функции или выражения производится приблизительно, для чего используются специальные численные алгоритмы. Символьный же подсчет происходит так же, как при решении задач на бумаге (то есть используются разного рода аналитические преобразования). В этом разделе мы подробно обсудим особенности численных расчетов. Символьной же математике посвящен следующий раздел.

2.3.1. Оператор численного вывода

Значение многих функций или выражений может быть подсчитано как символьно, так и численно. Чтобы система могла определить, каким именно способом должен быть произведен расчет, существуют два оператора вывода: численный и символьный.

В качестве численного оператора вывода выступает обычное «=». Ввести его можно как с клавиатуры, так и с двух панелей семейства Math (Математические): Calculator (Калькулятор) и Evaluation (Выражение), на которых он носит название Evaluate Numerically (Подсчитать численно).

Чтобы найти численное значение некоторого выражения, нужно просто ввести после него оператор «=».

Пример 2.7. Расчет численного значения выражения

$$y := 45 \qquad x := \frac{\pi}{3}$$

$$\frac{y^2 \cdot \sin(x)}{\tan\left(\frac{x}{5}\right)} = 8.251 \times 10^3 \qquad \frac{x^{-23}}{y^5} = 1.876 \times 10^{-9}$$

Аналогично можно найти численное значение и некоторой функции.

Пример 2.8. Численный расчет значения функции

$$f(x,y) := \frac{x^2 + y^2}{x - y}$$

$$x := 6 \qquad y := 5$$

$$f(x,y) = 61$$

Впрочем, для того чтобы подсчитать значение корректно заданной функции, совершенно не обязательно определять величины переменных выше ее выражения. Для этого можно просто записать в скобках ее имени вместо буквенных обозначений переменных их численные значения. Такой способ намного эффективнее описанного выше, поскольку позволяет получать одновременно значение функции при разных величинах переменных.

Пример 2.9. Расчет численного значения функции в нескольких точках

$$f(x,y) := \frac{x^2 + y^2}{x - y}$$

$f(5,6) = -61$ $f(6,5) = 61$ $f(3,7) = -14.5$ $f(67,98) = -454.613$

По умолчанию отображается лишь четыре знака мантиссы числа. Как увеличить количество отображаемых знаков, рассказывается в подразд. 2.3.4.

Точность численного расчета в Mathcad ограничена 17 знаками после запятой. Поэтому, если в ходе расчета будет получено число, меньшее 10^{-17}, оно будет округлено до нуля. Верхняя граница при численных расчетах равна машинной бесконечности: 10^{307}. Чтобы разобраться в природе данных ограничений, внимательно прочитайте следующий подраздел.

2.3.2. Особенности представления чисел и проведения арифметических расчетов

При проведении расчетов численно (в отличие от символьных расчетов) Mathcad использует аппаратно реализованные арифметические инструменты самого компьютера, оперируя 64-битными числами с плавающей точкой. Именно отсюда проистекают все особенности и недостатки численных расчетов, которые на первый взгляд могут показаться следствием несовершенства самой программы.

В математике считается, что множество чисел бесконечно. Это означает, что число может быть как сколь угодно большим, так и сколь угодно малым. Оно может быть образовано любым количеством цифр. Так, существуют бесконечные дроби и бесконечные трансцендентальные константы (например, число π или число e). Но в реальном мире все конечно. Даже вселенная имеет границы. Ограничена и память компьютера. А так как на каждую цифру числа нужно выделить несколько бит, то любая вычислительная машина считает с конечной точностью.

Точность вычислений определяется тем, сколько памяти выделяется на хранение каждого числа. На современных компьютерах используется 64-битный формат двойной точности (соответственно формат одинарной точности — 32-битный). Это позволяет напрямую закодировать 2^{64}=18 446 744 073 709 551 616 целых чисел. Совсем немного, не правда ли? На практике приходится работать куда с большим количеством числовых значений. Как же «запихнуть» в скромные 64 бита все необходимые числа?

Решением описанной сложности является формат представления чисел с использованием плавающей точки. Основная его идея заключается в следующем. Число разделяется на две части — значащую часть (мантиссу) и степень. Например, число 123 можно записать, как 1.23×10^2, число 123 000 — как 1.23×10^5, число 0.123 — как 1.23×10^{-1}. При этом возможное число вариантов для передаваемых 64 битами чисел резко возрастает. Правда, точность представления чисел от этого не повышается — она всегда равна номеру последнего знака мантиссы. Но так как нас практически всегда интересуют результаты ограниченной точности (чаще всего, всего лишь до третьего знака после запятой), то модель чисел с плавающей точкой вполне приемлема.

В общем случае число имеет форму $s\times m\times2^e$, где s — знак числа (1 бит), m — целое число, определяющее мантиссу (в процессорах Pentium под него отведено 52 бита),

e — показатель степени (обычно 10 бит). Проанализируем приведенные цифры. Если значимая часть числа может определяться 53 двоичными разрядами, то количество его десятичных разрядов не может превышать 53×lg(2)»15. Это означает, что точность вычислений ограничена 15 действительными знаками. Так и было во всех версиях Mathcad вплоть до 12. В 12 версии, вероятно, в связи с переходом на технологию .NET, длина мантиссы чисел увеличилась до 17 знаков. Это означает, что под мантиссу сейчас выделяется 55–56 бит. На показатель степени, как и раньше, выделяется 11 бит (это легко установить по значению максимального представимого числа). Итого получается 67–68 бит на число. Но как это возможно, если на аппаратном уровне под число с плавающей точкой выделяется всего 64 бита? Вероятно, среда Net Framework, в которой исполняется Mathcad, выделяет под мантиссу разное количество бит в зависимости от величины представляемого числа. Если число не очень большое и не очень маленькое (по абсолютному значению), то на то, чтобы описать его степень, требуется гораздо меньше, чем 11 бит. Поэтому часть бит можно отдать под мантиссу. Так как в 99 % случаев приходиться работать именно со средними числами, то не будет преступлением декларировать точность в 17 знаков мантиссы. Это не будет соблюдаться лишь, если показатель степени окажется больше (по абсолютному значению) 20. При этом точность снизится до 15–16 знаков.

Итак, точность представления чисел обычно составляет 17 знаков мантиссы. Если в мантиссе больше знаков, то произойдет округление:

$$1.99999999999999999=2$$

Если целое число образовано более чем 17 цифрами, то также производится округление до 17 значащих разрядов. Мантисса при этом будет записана в виде десятичной дроби, и у числа появится порядок:

$$6636637838383838838383883838383874646601018378337739383444 3=$$

$$=3.6366378383838391\times10^{62}$$

Обратите внимание на то, что мантисса соответствует исходному числу лишь до 15 знака. Это, вероятно, связано с тем, что в случае такого большого порядка на мантиссу выделяется лишь 53 бита.

Аналогичным образом преобразуются и дробные числа:

$$0.00000000000678994678765444434877 45=6.7899467876544444\times10^{-12}$$

То, что на хранение показателя степени числа с плавающей точкой выделяется ограниченное количество бит, означает, что невозможно работать с очень большими или очень малыми значениями. Максимальное число, которое может участвовать в вычислениях, называется машинной бесконечностью. Определить его, даже не прибегая к справочникам, очень просто. Исходя из описанных принципов представления чисел с плавающей точкой очевидно, что оно должно иметь максимально возможную мантиссу, умноженную на максимально возможную степень. В двоичном представлении это будет число, действительная часть которого образована 53 единицами, а порядок равен 2 в степени 971. Попробуем определить десятичное значение этого числа:

$$\text{mantissa} := \sum_{i=0}^{52} 2^i \qquad \text{stepen} := 2^{971}$$

$$\text{mantissa}\cdot\text{stepen} = 1.798\times10^{308}$$

Полученная величина и является приблизительным значением искомой машинной бесконечности. Если при расчете системой будет получено значение, большее машинной бесконечности, то она не сможет разместить его в отведенных на число 64 битах. Соответственно при этом возникнет сбой, и будет возвращено сообщение об ошибке (рис. 2.4).

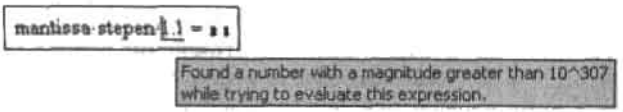


Рис. 2.4. Сообщение гласит: «При попытке вычислить это выражение найдено число со степенью, большей чем 10^307»

Ограничение имеется не только на положительное значение показателя степени числа с плавающей точкой, но и на отрицательное. Это означает, что существует число, все числа меньше которого воспринимаются как 0. Это число называется машинным нулем, и равно оно $1\times2^{-1074} \approx 4.94065645841247\times10^{-324}$.

В Mathcad по умолчанию до нуля округляются числа, по модулю меньшие 10^{-15}. Отчего так происходит, если лимит машинного нуля дает возможность оперировать куда меньшими значениями? Все дело в погрешности расчетов, предельная точность которых, ввиду ограничений на длину мантиссы числа, составляет, в зависимости от величины числа, 15–17 значимых цифр. Округление до 10^{-15} ее сглаживает, в результате чего неудобные и некрасивые ответы появляются реже. Например, если снизить порог нуля до 10^{-20}, то при вычислении синуса π мы получим следующий результат:

$$\sin(\pi) = 1.2246063538223773\times10^{-16}$$

В рамках погрешности численных расчетов ответ был получен верно. Однако для человека, далекого от мира компьютеров, но знающего, что $\sin(\pi)=0$, он неправилен. Чтобы отсечь погрешность, порог нуля должен составлять 10^{-15}:

$$\sin(\pi) = 0$$

Чаще всего снижать порог нуля нет никакого смысла. Ответы от этого точнее не станут, а погрешность проявится. Это стоит сделать лишь в том случае, если в рамках решаемой задачи вам придется проводить арифметические расчеты с очень малыми величинами (функции от таких значений обычно корректно не вычисляются). Для этого откройте вкладку Tolerance (Точность) окна Result Format (Формат результата) (открывается командой Result меню Format). На этой вкладке найдите окошко Zero threshold (Порог нуля). По умолчанию в него введено значение 15. Его вы можете заменить любым другим числом от 0 до 307 (при этом ни одна формула не должна быть выделена). Приведем несколько примеров расчетов при минимальном пороге нуля:

$$\frac{10^{-34}}{5^{90}} = 1.238\times10^{-97} \qquad \sin\left(10^{-100}\right) = 1\times10^{-100} \qquad e^{-167} \approx 2.97\times10^{-73}$$

Все числа в Mathcad рассматриваются как комплексные. И соответственно практически все математические операции рассчитаны на такой тип числа. Поэтому, помимо ошибки в действительной части, нужно учитывать и погрешность мнимой составляющей результата. Весьма характерна ситуация, когда, например, при численном определении корня нелинейного уравнения ответ, который, исходя из графика, должен быть

действительным числом, выдается с небольшой мнимой частью. Ее появление — результат погрешности вычислений, и, естественно, ее присутствие в ответе делает его математически некорректным. Чтобы избавиться от такой случайной мнимой части, следует уменьшить величину порога нуля для комплексных чисел Complex threshold рассматриваемой вкладки. По умолчанию данный параметр равен 10, что на пять порядков меньше порога нуля для действительных чисел. Это означает, что в том случае, если отношение действительной части к мнимой превысит 10^{10}, в качестве ответа будет отображена только действительная. И, наоборот, если это отношение окажется меньшим по модулю, чем 10^{-10}, на лист будет выведена только мнимая часть. Максимальное значение Complex threshold также меньше, чем Zero threshold, и составляет 63. Примеры округления комплексных чисел:

$$10 + 10^{-10} \cdot i = 10 \qquad 10^{-10} + 10 \cdot i = 10i$$

Если не знать особенностей представления чисел на компьютере, то возможны очень интересные «сюрпризы». Например, попробуем вычислить значения двух алгебраически идентичных выражений и посмотрим, совпадут ли результаты:

$$f(x) := \frac{x}{\sqrt{x+1} + \sqrt{x}} \qquad g(x) := x \cdot \left(\sqrt{x+1} - \sqrt{x}\right)$$

$$f(5000) - g(5000) = 4.121 \times 10^{-11} \qquad f(500000) - g(500000) = 3.726 \times 10^{-9}$$

Результаты расчетов по двум идентичным формулам различаются, пусть и незначительно. А это означает, что если бы мы пытались установить факт равенства этих выражений, то вывод был бы сделан неверно.

Причиной появления ошибок, подобных описанной, является то, что числа сохраняются с ограниченной точностью. Конечно, 15–17 знаков мантиссы — это более нежели достаточно для большинства расчетов. Но в некоторых случаях ошибки округления способны накапливаться, искажая результат весьма значительно. Например, в следующем примере показано, что путь, которым осуществляется вычисление, в случае численных расчетов влияет на результат (что связано с разной погрешностью разных операций):

$$\sum_{n=1}^{1000000} 0.00001234 = 12.3399999997527$$

$$0.00001234 \cdot 1000000 = 12.3400000000000$$

Даже более значительные ошибки, чем при суммировании и умножении, возникают при проведении таких арифметических операций, как деление и, особенно, извлечение корня. Кстати, чем ближе число, с которым проводится та или иная арифметическая операция, к крайним значениям численного промежутка, значениями в пределах которого может оперировать компьютер, тем относительная ошибка вычислений выше.

То, что число π хранится с точностью лишь до 17-го знака, является одной из причин погрешности, возникающей при вычислении значений тригонометрических функций. Обычно эта погрешность незаметна, однако она проявляется при определении значения функций вблизи нулей и разрывов (особенно, когда π предшествует большой множитель):

$$\sin(10000\pi) = -4.856\times10^{-13} \qquad \cos\left(\frac{\pi}{2}\cdot 100001\right) = 1.954\times10^{-11}$$

$$\tan\left(\frac{\pi}{2}\cdot 10001\right) = 1.013\times10^{12} \qquad \cot\left(\pi\cdot 10^{10}\right) = -4.466\times10^{5}$$

Из приведенных примеров можно сделать вывод, что в подавляющем большинстве случаев уровень доверия к численному результату не может включать все 15–17 цифр мантиссы. Однако и ниже 10-го знака точность опускается редко. Поэтому наиболее корректным будет использовать округленные ответы приблизительно таких порядков (впрочем, используемые на практике значения чаще всего округляются до 3-го знака — при этом все вычислительные ошибки Mathcad становятся практически незаметными). Если же вас интересует результат предельно высокой точности, для проведения вычислений нужно использовать не численный, а символьный расчет. Если же расчет можно осуществить только численно, то сгладить погрешности можно, уменьшив порог нуля (впрочем, данный подход применим далеко не всегда).

Иногда при расчетах возникает ситуация деления на нуль. При численных расчетах оперировать понятием «бесконечность» невозможно, поэтому система выдаст сообщение об ошибке. Чтобы оперировать выражениями, в которые входит бесконечность (или при вычислении которых бесконечность получается), расчет следует проводить символьно.

Самой большой странностью численной арифметики в Mathcad является то, что при делении 0 на 0 в результате получается 0:

$$\frac{0}{0} = 0 \qquad f(x) := \frac{\sin(x)}{x} \qquad f(0) = 0$$

То, что 0/0 не равно какому-то конечному числу — это аксиома. Если же признать, что 0 можно разделить на 0, то можно будет с легкостью доказать, например, что 1 равно 2 (попробуйте это сделать). Трудно сказать, что заставило разработчиков ввести в систему столь необычную особенность, но помнить о ней нужно, так как она может стать причиной трудноуловимой ошибки. Кстати, в универсальных языках программирования при делении 0 на 0 получается неопределенная числовая величина, обозначаемая обычно как NaN. В Mathcad же такой величины нет (вернее, есть, но она используется немного в другом контексте).

При проведении расчета символьно отношение 0/0 вызывает ошибку.

2.3.3. Особенности реализации математических функций

Используя в численных расчетах математические функции, важно иметь представление о том, как они реализуются на компьютере. Дело в том, что ввиду особенностей, лежащих в основе задания математических функций, возвращаемые ими значения далеко не всегда являются истинными. Если этого не учитывать, то можно допустить ошибку, найти которую будет очень и очень сложно.

Основные арифметические операции над числами с плавающей точкой, такие как сложение, умножение или деление, поддерживаются компьютером на аппаратном уровне. То есть их проделывает процессор напрямую. Поэтому скорость выполнения подобных операций может быть крайне высокой. Такие же операции, как вычисление синуса

или логарифма, аппаратно не поддерживаются. Причина этого не в том, что реализовать необходимый для этого алгоритм «в железе» невозможно. Нет, это вполне осуществимая задача (более того, основная аксиома микроэлектроники говорит, что аппаратно можно реализовать любой алгоритм). Просто вводить в процессор соответствующий элемент не имеет смысла, так как создать математические функции очень просто программно, используя только лишь аппаратно поддерживаемые арифметические операции.

В основе реализации таких невыразимых в общем случае через арифметические операции (они называются трансцендентными) математических функций, как экспонента, арктангенс или логарифм, лежит одна чрезвычайно важная идея, доказываемая в математическом анализе. Ее суть сводится к тому, что функция, при соблюдении некоторых ограничений, может быть заменена в окрестности данной точки на степенной ряд. Степенной ряд представляет собой сумму вида $a0 \cdot x^0 + a1 \cdot x^1 + a2 \cdot x^2 + \ldots + an \cdot x^n$, где a0, a1, a2 ... an — некоторые коэффициенты. Нахождение для функции приближающего ее ряда называется разложением в степенной ряд. Чем больше будет найдено членов разложения, тем точнее будет приближена функция. Также точность приближения зависит от того, насколько далеко располагается данная точка от той, для которой разложение было проведено.

Например, для синуса степенной ряд в точке разложения a=0 имеет вид (12 членов разложения, 6 из них равны 0):

$$\sin(x) = 1 \cdot x - \frac{1}{6} \cdot x^3 + \frac{1}{120} \cdot x^5 - \frac{1}{5040} \cdot x^7 + \frac{1}{362880} \cdot x^9 - \frac{1}{39916800} \cdot x^{11} + O\left(x^{13}\right)$$

Зная, в какой ряд разлагается функция, написать на основании него алгоритм для определения ее значений не составляет никакого труда. Единственное, нужно правильно оценить, сколько следует взять членов разложения, чтобы точность приближения была достаточной. Для этого существуют специальные формулы, приводить которые в этой главе будет излишним (соответствующий пример имеется в гл. 12).

Итак, математические функции вычисляются с помощью приближенного ряда. Точность таких вычислений равна в идеале точности представления чисел. Это означает, что в 14–15 знаке мантиссы ошибка будет почти наверняка. Конечно, это не важно в подавляющем большинстве случаев. Но иногда и эта ошибка может стать фатальной. Прежде всего, это касается точек разрывов и нулей функций. Например, для любого школьника очевидно, что $\sin(n \cdot \pi)$ при целых n равен 0. В Mathcad же при проведении расчета численно равенство $\sin(n \cdot \pi) = 0$ почти наверняка не будет соблюдаться. То же самое можно сказать о значении любой функции практически в любой точке: точным до 17 знака мантиссы оно не будет.

Существуют и другие техники для приближенного вычисления математических функций. Так, все тригонометрические функции рассчитываются на основании тангенса, значение которого находят посредством разложения в цепную дробь. Данная дробь сходится быстрее соответствующего ряда Тейлора.

2.3.4. Формат вывода численного результата

При стандартных установках численный ответ отображается только до третьего знака после запятой. Это связано как с тем, что на практике, как правило, более высокой точности и не требуется, так и с тем, что многие численные алгоритмы (например, решения уравнений) при обычных настройках работают именно с такой погрешностью.

Точность же расчета численных значений алгебраических выражений или встроенных функций (а также их сочетаний) постоянна при любых величинах системных переменных и составляет 17 знаков после запятой. Поэтому при желании вы можете получить и более точное, нежели при стандартных настройках, значение вашей переменной или функции. Для этого установите курсор на текст ответа и выполните двойной щелчок мышью. При этом откроется вкладка Number Format (Формат числа) окна Result Format (Формат результата). В окошке параметра Number of decimal places (Количество десятичных позиций) данной вкладки определите, с точностью до какого знака после запятой (от 0 до 17) должен быть отображен результат.

Кстати, открыть окно Result Format можно и с помощью соответствующей команды меню Format. Причем между изменениями вида результата этими двумя способами имеется существенное различие: в первом случае изменения коснутся одной лишь формулы, во втором — всего документа.

Пример 2.10. Отображение численного результата с различной точностью

$$\sin\left(\frac{\pi}{9}\right) = 0.342 \qquad \sin\left(\frac{\pi}{9}\right) = 0.3420201433$$

$$\sin\left(\frac{\pi}{9}\right) = 0.34202 \qquad \sin\left(\frac{\pi}{9}\right) = 0.3420201433256686$$

Обратите внимание на то, что менее точный результат получается в Mathcad не простым отбрасыванием лишних десятичных знаков, а исходя из строгих математических правил округления.

По умолчанию незначащие десятичные нули в численных результатах не отображаются. Однако, установив флажок Show trailing zeros (Показывать незначащие нули) на рассматриваемой вкладке, вы сделаете их видимыми.

Пример 2.11. Отображение незначащих нулей при различной точности

$$\frac{1}{4} = 0.2500000000 \qquad \frac{1}{\sin\left(\frac{\pi}{6}\right)} = 2.000000000000$$

Отображение больших чисел в десятичной форме не слишком удобно. Поэтому на практике обычно используется вид числа с порядком. При установках, принятых в Mathcad по умолчанию, в такой форме отображается любое число, если модуль порядка соответствующего ему числа с плавающей точкой превышает три. Например:

$$1500 = 1.5 \times 10^{3} \qquad 0.000015 = 1.5 \times 10^{-5}$$

При необходимости величину показателя степени, при превышении которой число будет отображаться с порядком, можно уменьшить или увеличить. Чтобы это сделать, следует ввести требуемую величину порога в окошко параметра Exponential Threshold (Порог экспоненты). Очевидно, что значение это должно лежать между 0 и 15.

Помимо обычной принятой в математике формы, числа с показателем степени могут быть отображены и в так называемом инженерном формате. Особенностью его является то, что показатель степени всегда отображается числом, кратным 3. Если такая

форма представления числа для вас более привычна, то вам следует задействовать параметр Show exponents in engineering format (Показывать экспоненту в инженерном формате).

Пример 2.12. Отображение числа в инженерном формате

$1000 = 1\times10^{3}$ $\qquad$ $100000 = 100\times10^{3}$

$10000 = 10\times10^{3}$ $\qquad$ $1000000 = 1\times10^{6}$

В Mathcad существует несколько типов представления численного результата. Выбрать наиболее подходящий из них вы можете в списке Format (Формат) рассматриваемой вкладки Number Format.

Всего имеется пять форматов численного результата.

- General (Основной). Формат, выбранный по умолчанию. Позволяет произвольным образом определять количество отображаемых десятичных знаков (Number of decimal places), порядковый порог (Exponential threshold), визуализацию незначащих нулей (Show trailing zeros). Все примеры, которые были приведены нами ранее, отображались именно в этом формате.
- Decimal (Десятичный). Результат отображается только в виде десятичной дроби. Десятичная часть числа контролируется параметром Number of decimal places (Количество десятичных позиций). Если полученный результат имеет целую часть, длиннее 15 знаков, все числа, лежащие в нем после 15 позиции, будут заменены нулями.

Пример 2.13. Представление результата в десятичном формате

General	Decimal
$\frac{1}{\sin\left(\pi+\frac{1}{\pi^{10}}\right)} = -9.365\times10^{4}$	$\frac{1.}{\sin\left(\pi+\frac{1}{\pi^{10}}\right)} = -93648.047$
$\frac{1}{1000} = 1\times10^{-3}$	$\frac{1}{1000} = 0.001$

- Scientific (Научный). Число отображается только со степенью таким образом, чтобы целая часть мантиссы состояла из одного символа. Количество десятичных знаков и отображение незначащих нулей результата определяется пользователем. Кроме того, существует возможность представления числа в техническом формате (параметр Show exponents as E±000 (Показывать показатель степени как E±000)).

Пример 2.14. Научный (Scientific) формат

$10 = 1\times10^{1}$ $\qquad$ $100000 = 100E+003$

$1.1 = 1.1000000000\times10^{0}$ $\qquad$ $1234567 = 1.23457E+006$

- Engineering (Инженерный). Формат очень близкий к научному. Единственным отличием является то, что порядок числа должен быть обязательно кратен трем.
- Fraction (Дробный). Очень необычный формат, резко отличающийся от всех остальных. Представляет результат вычислений в виде простой дроби. Естественно, такое представление является в большинстве случаев приближенным, однако его точ-

ность можно регулировать с помощью параметра Level of Accuracy (Уровень точности). Данный параметр определяет, с точностью до какого десятичного знака будет приближен результат. Очевидно, что область его изменения — от 0 до 15. По умолчанию числа, имеющие целую часть, приближаются неправильными дробями. Чтобы целая часть была выделена и в дробном формате, следует задействовать параметр Use Mixed Numbers (Использовать смешанные числа).

Пример 2.15. Дробный формат

$$0.7659 = \frac{566}{739} \qquad 0.7659 = \frac{10}{13} \qquad 0.7659 = \frac{7659}{10000}$$

$$67.987 = \frac{67987}{1000} \qquad 67.987 = 67\frac{987}{1000}$$

Численный результат может быть отображен не только в десятеричной, но и в двоичной, восьмеричной и шестнадцатеричной системах счисления. Чтобы сменить систему счисления, следует дважды щелкнуть мышью на результате вычисления. При этом появится окно Result Format (Формат результата), в котором следует перейти на вкладку Display Options (Параметры отображения). Здесь имеется специальный список Radix (Основание счисления), содержащий четыре пункта: Decimal (Десятичная), Binary (Двоичная), Octal (Восьмеричная), Hexadecimal (Шестнадцатеричная). Чтобы числа в разных системах счисления можно было отличить, в конец двоичного числа добавляется буква «b», восьмеричного — «o», шестнадцатеричного — «h».

Пример 2.16. Перевод числа в другие системы счисления

$$\text{infinity} := \sum_{n=0}^{52} 2^n \cdot 2^{971}$$

$$\text{infinity} = 0\text{b} \times 10\text{b}^{10000000000\text{b}} \qquad \text{infinity} = 2\text{o} \times 10\text{o}^{525\text{o}} \qquad \text{infinity} = 0\text{h} \times 10\text{h}^{100\text{h}}$$

Работать с числами альтернативных систем счисления в Mathcad можно точно так же, как с обычными десятичными числами.

2.4. Символьные расчеты

Прежде чем приступать непосредственно к разговору о технике проведения символьных расчетов в Mathcad, имеет смысл разобраться, что это такое и чем вычисления данного типа отличаются от численных.

2.4.1. Особенности символьных расчетов

Символьный расчет связан с получением результата аналитическими методами. Это означает, что, например, корни уравнения будут найдены благодаря выражению с помощью алгебраических преобразований одной переменной через все остальные или (при решении уравнения в общем виде) через параметры. Интеграл будет подсчитан с помощью известной формулы Ньютона–Лейбница: как разность значений первообразных в точках пределов интегрирования и т. д. Можно смело утверждать, что если вы никогда ранее не сталкивались с математическими расчетами с помощью компьютера,

то, скорее всего, вы в своей практике применяли только символьные методы. Это связано с тем, что непосредственно на бумаге производить расчет с помощью любого численного метода — это чрезвычайно тяжелая задача для человека (в связи с огромным объемом вычислительной работы). Для компьютера же просчитать сумму из миллионов элементов или обратить матрицу большой размерности — дело секунд и долей секунд. До самого последнего времени аналитические расчеты были уделом людей, а численные — компьютеров. Почему? Все дело в том, что решить даже не очень сложную задачу аналитически машина сможет далеко не всегда. Мышление человека гибко и пластично, оно способно к творчеству. Компьютер же, несмотря на все успехи технологий искусственного интеллекта, мыслить, естественно, не может. Это связано с тем, что он работает исходя из точных и четких алгоритмов, в которых все должно быть предусмотрено. А создать такой алгоритм далеко не всегда возможно. Однако те задачи, в основу решения которых могут быть положены строгие формулы и четкие алгоритмы (дифференцирование, интегрирование, поиск корней несложных уравнений) компьютером решены быть могут. Правда, для этого потребуется сохранить в виде базы данных значительное количество формул и алгоритмов аналитических преобразований. Поиск в этой базе, сопоставление переданной на обработку задачи с описанными в ней требуют большой мощности компьютера. Только к концу 1980-х массово стали выпускаться машины, памяти и производительности которых было достаточно для того, чтобы аналитические расчеты на компьютере стали реальностью. Численные же методы использовались ЭВМ с самого начала их истории.

История компьютерной символьной математики берет свое начало еще в 1960-х. Однако вплоть до конца 1980-х эта область была развита довольно слабо, что связано с дороговизной суперкомпьютеров, на которых системы символьной математики могли бы работать. Хотя значительные успехи имелись уже тогда, причем, как это ни удивительно, даже в нашей стране. Так, в конце 1970-х группе под руководством академика Глушкова удалось создать небольшие ЭВМ, которые поддерживали аналитические расчеты даже на аппаратном уровне. Был создан язык для проведения символьных расчетов «Аналитик». Однако общедоступная символьная компьютерная математика начинается с появлением мощных ПК типа IBM-486. В это время разрабатывается существующая и поныне система Derive, написанная на языке логического программирования Lisp. Вскоре появляется система Maple, основанная на наработанных за многие годы суперкомпьютерной эры символьной математики библиотеках старейшего языка математических расчетов Fortran (до настоящего времени Maple сохраняет лидерство в области аналитических расчетов). В то же время создается и наиболее мощный (но и дорогой) математический пакет Mathematica.

Система Mathcad была разработана в 1988 году как среда для проведения расчетов исключительно численно. Однако уже в четвертой версии программы появилась возможность решения задач аналитически. Причем, соответствующий символьный процессор не был создан Mathsoft самостоятельно. Взвесив все «за» и «против», Mathsoft предпочла купить ядро Maple. Так что символьный процессор является чужеродным телом в Mathcad, что можно почувствовать и сейчас (за прошедшие десять лет разработчики так и не смогли его полностью интегрировать со средой численных расчетов Mathcad).

Чем символьные расчеты лучше численных? Во-первых, они лишены погрешности. Численные же алгоритмы всегда дают результат приближенно. Конечно, точность такого приближения может быть высока — но всегда найдутся задачи, в которых погрешность проявится так, что ценность результата будет сведена к нулю. Не стоит забывать и про то, что погрешности свойственно накапливаться, что приводит к тому, что порой численные методы расходятся. Во-вторых, численное решение является частным.

То есть, например, при численном подсчете интеграла результат будет получен для каких-то конкретных значений пределов интегрирования и параметров подынтегральной функции. На основании него нельзя будет сказать, чему будет равен интеграл при других значениях пределов и параметров. Символьная математика дает возможность получить результат в общем виде, как формулу. Естественно, что информации из общей формулы можно почерпнуть куда больше, чем из частного значения или даже графика. В-третьих, даже при получении числового результата символьный подход имеет преимущество, так как при этом ответ представляется в виде арифметического выражения, более понятного и привычного для нас, чем десятичная дробь, которая выдается при проведении подсчета численно.

Есть ли у символьных расчетов недостатки? Естественно. Так, далеко не все задачи можно решить аналитически. Многие интегралы являются неберущимися, во многих уравнениях нельзя выразить неизвестную по причине того, что в их выражения входят разнородные функции, и т. д. Даже если задача имеет аналитическое решение, программа может его и не найти (все-таки, аналитические преобразования — это довольно нагруженная интеллектуально сфера). Иногда ответ выдается в виде громоздкого выражения, которое еще нужно суметь упростить. Нередки случаи, когда символьный процессор просто ошибается.

Увы, чтобы эффективно использовать системы символьной математики, этой самой математикой нужно владеть. Трудно решить неэлементарную задачу, просто нажимая кнопки. Наоборот, участвуя в процессе решения, направляя программу, можно справиться и с очень нетривиальными задачами. В данной книге имеется немало примеров, показывающих, как, сочетая вычислительные возможности Mathcad и собственную голову, можно справиться с теми проблемами, перед которыми сама программа пасует.

Сделаем выводы. Нельзя не согласиться, что символьные расчеты имеют целый ряд несомненных преимуществ перед численными. Поэтому ту или иную задачу сначала стоит попытаться решить символьно и лишь при неудаче (или если результат будет уж очень громоздок) использовать численные методы. При всех своих недостатках численные методы имеют одно достоинство — универсальность. Используя их, можно решить практически любую задачу, в то время как аналитические методы хороши лишь в случае отдельных, «удобных» примеров.

2.4.2. Принципы проведения расчета символьно

Чтобы задействовать для решения задачи символьный процессор, следует использовать специальный оператор вывода в виде стрелки «→» (Evaluate symbolically). Ввести его можно либо с помощью соответствующих кнопок панелей Symbolic (Символьные) или Evaluation (Вычисление), либо сочетанием клавиш Ctrl+«.».

В том случае, если аналитическому процессору не удастся получить результат, то справа от оператора символьного вывода будет выдано само же выражение:

$$\sin\left(\frac{\pi}{7}\right) \rightarrow \sin\left(\frac{1}{7}\cdot\pi\right)$$

Реже может использоваться сообщение об ошибке: No symbolic result was found (Символьный результат не был найден) или некоторые другие.

Очень часто подсчитать тот или иной пример можно как символьно, так и численно. Форма ответа при этом почти наверняка будет различной.

Пример 2.17. Символьный и численный подсчет значений выражений и функций

$$\sqrt{8}\ \text{simplify} \rightarrow 2\cdot 2^{\frac{1}{2}} \qquad \frac{1}{3}+\frac{7}{9} \rightarrow \frac{10}{9} \qquad \sin\left(\frac{\pi}{3}\right) \rightarrow \frac{1}{2}\cdot 3^{\frac{1}{2}}$$

$$\sqrt{8} = 2.828 \qquad \frac{1}{3}+\frac{7}{9} = 1.111 \qquad \sin\left(\frac{\pi}{3}\right) = 0.866$$

Анализируя приведенный пример, можно прийти к выводу, что символьный процессор «стремится» получить результат в такой же форме, как и человек, решая задачу на бумаге. То есть при сложении простых дробей и будет получена простая дробь, а не приближенное число с десятичной частью, как при численном подсчете. При извлечении корня из восьми символьный результат будет привычным: два корня из двух. А при вычислении синуса π/3 символьный процессор выдаст известное любому школьнику иррациональное выражение, а не десятичную дробь, которую мы бы получили при использовании в качестве оператора вывода «=». Согласитесь, что в приведенных примерах символьная форма результата куда более наглядная и привычная, чем числа с плавающей точкой, которые дает численный подсчет.

Эти рассуждения можно распространить и на другие математические операции, такие как решение уравнений, интегрирование и дифференцирование.

Пример 2.18. Форма результата при символьном и численном решении квадратного уравнения

Symbolic:

$$x^2 - 3x + 1\ \text{solve},x \rightarrow \begin{pmatrix} \frac{3}{2}+\frac{1}{2}\cdot\sqrt{5} \\ \frac{3}{2}-\frac{1}{2}\cdot\sqrt{5} \end{pmatrix}$$

Numerical:

$$\text{polyroots}\left(\begin{pmatrix} 1 \\ -3 \\ 1 \end{pmatrix}\right) = \begin{pmatrix} 0.382 \\ 2.618 \end{pmatrix}$$

Огромным достоинством символьных методов является то, что результат расчета может быть получен в общем виде. А это означает, что можно решать уравнения с параметрами и подсчитывать интегралы с буквенными коэффициентами, получать производную в виде функции, а не графика и т. д.

Пример 2.19. Алгебраические и аналитические символьные преобразования

$$a\cdot x^2 + b\,x + c\ \text{solve},x \rightarrow \begin{bmatrix} \frac{1}{2\cdot a}\cdot\left[-b+\left(b^2-4\cdot a\cdot c\right)^{\frac{1}{2}}\right] \\ \frac{1}{2\cdot a}\cdot\left[-b-\left(b^2-4\cdot a\cdot c\right)^{\frac{1}{2}}\right] \end{bmatrix}$$

$$\int_{a+b}^{a-b} \ln(x)\,dx \rightarrow \ln(a-b)\cdot a - \ln(a-b)\cdot b + 2\cdot b - \ln(a+b)\cdot a - \ln(a+b)\cdot b$$

$$\frac{d}{dx} a^{x^2+y^2+z^2} \rightarrow 2\cdot a^{\left(x^2+y^2+z^2\right)}\cdot x\cdot \ln(a)$$

Благодаря продемонстрированным возможностям, Mathcad может быть использован как очень неплохой справочник математических формул.

Среда разработки Mathcad по умолчанию работает исходя из правил проведения численных расчетов, а не символьных. Иногда это приводит к проблемам. Например, вы можете задать функцию с параметром, чтобы найти затем ее нули аналитически. Такая функция абсолютно приемлема с точки зрения символьного процессора, но не численного. Для него она некорректна, так как в нее входит неопределенная явно величина — параметр. Поэтому он будет выделен и появится сообщение This variable is undefined (Переменная не определена). Если расчет проводится аналитически, подобные сообщения нужно смело игнорировать. На результат они никак не повлияют.

Если в арифметическом выражении присутствует число с плавающей точкой или же оно не может быть рассчитано аналитически, то расчет будет проведен численно. Однако при этом будет использоваться программно реализованная арифметика длинных чисел, а не аппаратные средства для работы с 64-битными числами, как в случае применения оператора вывода «=». Это означает, что значение выражения или функции может быть найдено гораздо точнее тех 15–17 знаков, которые обеспечивает аппаратно поддерживаемая арифметика 64-битных чисел. Так, подсчитав количество десятичных знаков в ответе первого выражения в примере 2.20, вы обнаружите, что их 20. Это точность приблизительных расчетов аналитического процессора, принятая по умолчанию. При желании же любое выражение можно подсчитать с точностью до 4000 знаков (для этого используется оператор float, о котором мы поговорим чуть ниже).

Пример 2.20. Приблизительные расчеты, проводимые с помощью символьного процессора

$$\frac{78}{67} + 0.1^3 = 1.1651791044776119 \qquad \frac{78}{67} + 0.1^3 \rightarrow 1.1651791044776194$$

$$\sin\left(\frac{\pi}{45}\right) \text{float},30 \rightarrow .697564737441253007759588351941\text{e}{-1}$$

Арифметические расчеты с помощью символьного процессора могут быть особенно полезны, если величина результата меньше параметра Zero Threshold (Порог нуля) вкладки Tolerance (Точность) окна Result Format (Формат результата). В этом случае при использовании оператора численного вывода «=» в качестве ответа будет выдан нуль. Вычисление же символически даст правильный результат. Аналогично при проведении расчета аналитически можно оперировать величинами, превышающими машинную бесконечность.

При оперировании очень большими или очень малыми значениями результат может представлять собой число из десятков, сотен и даже тысяч цифр. Никаких ограничений на длину значимой части числа, подобных имеющим место в численных расчетах, при проведении подсчета аналитически нет. Это, с одной стороны, хорошо, так как

ответ можно получить без погрешности. С другой стороны, оперировать громоздкими числами сложно. Если результат аналитического вычисления получается слишком большим, его имеет смысл пересчитать в приблизительную десятичную дробь. Для этого можно использовать оператор численного вывода «=», но лучше задействовать оператор float.

Пример 2.21. Аналитический расчет в случае экстремально больших или малых величин

$$\frac{3^{1700}}{2^{1000}}\ \text{float},5 \rightarrow 1.1916\cdot 10^{510} \qquad e^{-1000}\ \text{float},5 \rightarrow 5.0760\cdot 10^{-435}$$

$$2^{300} \rightarrow 2037035976334486086268445688409378161051468393665936250636140449354381299763336706183397376$$

2.4.3. Способы проведения символьных преобразований

В системе Mathcad существует два альтернативных пути осуществления практически любого символьного преобразования: с помощью команд специального меню или операторов соответствующей панели.

Меню Symbolics (Символические) расположено в главном меню программы. Открыв его, вы увидите довольно длинный список различных параметров и команд, отвечающих за ту или иную операцию символьной математики. Чтобы осуществить необходимое символьное преобразование с помощью команд меню Symbolics, необходимо выполнить следующую последовательность действий.

1. Ввести выражение. Если это уравнение, то в качестве знака равенства следует использовать оператор логического равенства. В том случае, если уравнение задано в стандартном виде (то есть правая часть равна нулю), можно определить одну лишь левую часть: она будет приравнена к нулю по умолчанию.
2. В зависимости от того, какой тип символьных операций должен быть применен к введенному выражению, необходимо выделить либо переменную (если нужно использовать, например, команду решения уравнений или разложения в ряд), либо все выражение целиком (для команд символьной алгебры).
3. Выполнить необходимую команду.

В результате проделанных действий, в зависимости от действующих настроек, над, под или на месте исходного выражения появится ответ.

Пример 2.22. Символьные вычисления с помощью команд меню

Интегрирование (Symbolics ▸ Variable ▸ Integrate):

$$x^2\cdot\sin(2x)$$

$$\frac{-1}{2}\cdot x^2\cdot\cos(2\cdot x)+\frac{1}{4}\cdot\cos(2\cdot x)+\frac{1}{2}\cdot x\cdot\sin(2\cdot x)$$

Разложение в ряд Тейлора (Symbolics ▸ Variable ▸ Expand to Series):

$$x^2\cdot\sin(2x)$$

$$2\cdot x^3-\frac{4}{3}\cdot x^5+\frac{4}{15}\cdot x^7-\frac{8}{315}\cdot x^9+\frac{4}{2835}\cdot x^{11}+O\left(x^{13}\right)$$

Произвести настройку особенностей отображения результата при использовании для символьных расчетов меню Symbolics (Символические) можно с помощью специального меню Evaluation Style (Стиль вычислений). Параметры списка Show evaluations steps (Отображать шаги вычислений) данного меню определяют, каким образом по отношению к исходному выражению будет выведен результат символьных расчетов. Данный список содержит три пункта.

- Vertically, inserting lines (Вертикально, вставляя линии). Параметр, определенный по умолчанию. Результат вставляется в специальную, очищенную от других формул и выражений полосу рабочей области ниже исходного выражения. Остальные объекты листа при этом смещаются вниз на ширину данной полосы.
- Vertically, without inserting lines (Вертикально, не вставляя линий). Результат выводится ниже исходного выражения, однако это никак не сказывается на положении остальных формул и выражений вашего документа.
- Horizontally (Горизонтально). Результат отображается правее исходного выражения.

Если вы установите флажок Evaluate in Place (Вычислять на месте), то результат будет просто заменять исходное выражение.

При включении параметра Show Comments (Показать комментарии) ответ будет выводиться на лист вместе с текстовой строкой, содержащей информацию о проделанной символьной операции. Например, если была задействована команда решения уравнений Symbolics ▸ Variable ▸ Solve, то комментарием будет «msgMapleSolve». Несложно догадаться, что msg — это сокращение от message (сообщение). Maple — это название компании, разработавшей символьный процессор.

Панель Symbolic (Символьные) семейства Math (Математические) своим содержанием практически полностью повторяет соответствующее меню (рис. 2.5) (исключение составляют операторы интегрирования и дифференцирования, которые вынесены на панель Calculus (Вычислительные)). Однако между аналитическими вычислениями, осуществляемыми с помощью меню, и операциями, проводимыми с использованием панели, существует несколько принципиальных различий.

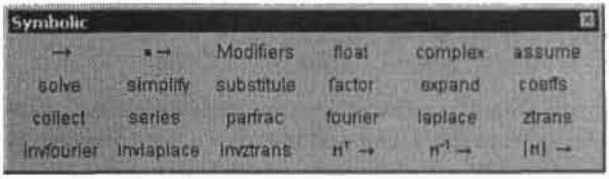


Рис. 2.5. Панели Symbolic (Символьные)

Во-первых, символьные операции, производимые с помощью панели Symbolic, осуществляются не командами, а специальными операторами. Ввести такой оператор можно очень просто, нажав на ней нужный пункт. Например, если необходимо решить уравнение, то следует выбрать пункт solve (решить):

$$\blacksquare \text{ solve}, \blacksquare \rightarrow$$

При этом появится оператор, образованный ключевым словом, двумя маркерами и оператором символьного вывода. На место левого маркера вводится уравнение, правого — переменная, по которой оно должно быть решено. Некоторые операторы содержат только один маркер (операторы символьной алгебры), а оператор разложения в ряд Тейлора — целых три:

$$\blacksquare\ \text{simplify} \rightarrow$$

$$\blacksquare\ \text{series}, \blacksquare, \blacksquare \rightarrow$$

Просто набрать с клавиатуры текст символьного оператора нельзя. Однако если вы часто прибегаете к символьным расчетам, то куда более эффективный путь ввода необходимых операторов дает использование специальной заготовки Symbolic Keyword Evaluation (Символьные вычисления с клавиатуры):

$$\blacksquare\ \blacksquare \rightarrow$$

Заготовка эта образована двумя маркерами и оператором символьного вывода. Введя в левый маркер ключевое слово, вы получите нужный вам оператор. Вставить же данную заготовку можно сочетанием клавиш Ctrl+Shift+«.».

Вторым принципиальным отличием между использованием панели и меню является то, что символьные преобразования с помощью команд меню осуществляются Mathcad без учета определений и присваиваний, сделанных в документе выше.

Приведем несколько примеров символьных преобразований с помощью операторов панели Symbolic.

Пример 2.23. Символьные преобразования операторами панели Symbolic

Решение уравнения:

$$x^2 + 2^x = 1\ \text{solve}, x \rightarrow 0$$

Упрощение выражения:

$$\frac{x^3 + y^3}{x + y}\ \text{simplify} \rightarrow x^2 - y \cdot x + y^2$$

Разложение в ряд Тейлора:

$$\sin(x)\ \text{series}, x, 10 \rightarrow 1 \cdot x - \frac{1}{6} \cdot x^3 + \frac{1}{120} \cdot x^5 - \frac{1}{5040} \cdot x^7 + \frac{1}{362880} \cdot x^9$$

Трудно не согласиться, что символьные преобразования, осуществляемые с помощью операторов панели Symbolic, куда более наглядны, чем при использовании команд соответствующего меню. Поэтому в дальнейшем особенности символьных операций мы будем рассматривать только на их примере. Вообще же, применение команд меню удобнее в том случае, если результат тех или иных символьных операций не нужно сохранять или использовать в последующих расчетах. В противном случае намного лучше использовать операторы панели Symbolic.

2.4.4. Совместное использование нескольких символьных операторов

Очень часто при преобразовании выражения требуется использовать сразу несколько команд. В системе Mathcad решение такого рода задач облегчено тем, что существует возможность совмещения нескольких символьных операторов в одном блоке. Например, если вам необходимо выполнить разложение в ряд и затем найти его сумму при

некотором значении переменной с какой-то определенной точностью, то выполните следующую последовательность действий.

1. Введите оператор разложения в ряд series (ряд):

$$\blacksquare \text{ series}, \blacksquare, \blacksquare \rightarrow$$

2. Установив курсор в крайний правый маркер, нажмите кнопку ввода оператора подстановки substitute (заместить).

$$\blacksquare \left| \begin{array}{l} \text{series}, \blacksquare, \blacksquare \\ \text{substitute}, \blacksquare = \blacksquare \end{array} \right. \rightarrow$$

3. Выделив полученный расчетный блок, введите оператор численного значения выражений float. Кстати, расположение нового оператора относительно блока зависит от того, в какую сторону повернут курсор: если его горизонтальная линия находится слева от блока, то оператор будет введен сверху, если она расположена справа, то добавление оператора произойдет снизу.

$$\blacksquare \left| \begin{array}{l} \text{series}, \blacksquare, \blacksquare \\ \text{substitute}, \blacksquare = \blacksquare \rightarrow \\ \text{float}, \blacksquare \end{array} \right.$$

4. В левый маркер, расположенный за появившейся линией блока, введите преобразуемое выражение. Правые маркеры символьных операторов заполняются стандартно.

$$\cos(x) \left| \begin{array}{l} \text{series}, x, 15 \\ \text{substitute}, x = 2 \rightarrow -.4161468396 \\ \text{float}, 10 \end{array} \right. \qquad \cos(2) = -0.4161468365$$

Совместно можно использовать не только операторы панели Symbolic. Также можно совмещать символьные операторы и операторы математического анализа панели Calculus.

2.4.5. Оператор float

Каждый из символьных операторов Mathcad будет подробно описан в соответствующем ему тематическом разделе. Сейчас же мы дадим описание лишь одного оператора — float.

Оператор численного расчета float (от floating point — «плавающая точка»), позволяет получить приблизительное значение некоторого выражения с нужной точностью в формате числа с плавающей точкой. Он содержит два маркера: в левый заносится выражение или функция, в правый — количество знаков после запятой, которые должен содержать ответ. Точность результата может быть задана числом от 1 до 250. А это означает, что, в принципе, вы можете получить ответ с точностью до 250 знаков. Согласитесь, такие цифры не могут не впечатлять!

Обычно оператор float используется, если выражение ответа, выданное аналитическим процессором, слишком громоздкое для того, чтобы его можно было успешно применять. Например, выражения корней алгебраических уравнений третьей и четвертой степеней могут занимать несколько листов. Единственный способ «выудить» полезную информацию из таких выражений — это пересчитать их в числа с плавающей точкой.

Пример 2.24. Использование оператора Float

Символьный ответ при решении следующего уравнения получается таким огромным, что система не может его отобразить. Это тот случай, когда без пересчета в приближенные десятичные дроби не обойтись.

$$x^4 + x^2 + x + 4 \;\Big|\; \begin{matrix}\text{solve}, x \\ \text{float}, 10\end{matrix} \rightarrow \begin{pmatrix} .8719084268 + 1.243765116\cdot i \\ .8719084260 - 1.243765116\cdot i \\ -.8719084261 + .9866594881\cdot i \\ -.8719084267 - .9866594881\cdot i \end{pmatrix}$$

Строго говоря, инструменты арифметики длинных чисел символьного процессора Mathcad позволяют оперировать с числами с мантиссой до 4000 знаков. Однако такие большие числа невозможно отобразить в среде разработки, поэтому оператор float работает с числами с мантиссой лишь до 250 знаков. Но если вы воспользуетесь командой Symbolics ▸ Evaluate ▸ Floating Point (Символьные ▸ Вычислить ▸ Плавающая точка), то значения можно будет получать и более точно. Правда, при этом они не будут отображаться в среде разработки, а будут помещаться в буфер обмена операционной системы. Но, сделав вставку в пустой текстовый документ, ими можно будет воспользоваться. Для примера рассчитаем число π с точностью до 500 знаков:

3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372458700660631558817488152092096282925409171536436789259036001133053054882046652138414695194151160943305727036575959195309218611738193261179310511854807446237996274956735188575272489122793818301194912

Оператор float работает с числами с плавающей точкой. Поэтому многое, что мы говорили в предыдущем разделе об особенностях численных расчетов в Mathcad, можно перенести на случай его использования.

2.5. Операторы

В наиболее общем смысле оператор — это символ или последовательность символов, обозначающий то или иное математическое действие. В Mathcad все математические операции осуществляются с помощью операторов либо встроенных функций. Причем к несомненным достоинствам системы следует отнести то, что все они полностью соответствуют принятым в математике правилам и традициям оформления.

Любой оператор Mathcad можно ввести двумя способами.

- С помощью специальной клавиши или сочетания клавиш.
- Используя соответствующую кнопку нужной панели семейства Math (Математические).

Исключений из этого правила существует только два: ни на одной из панелей не нашлось места для редко используемого оператора комплексного сопряжения и появившегося только в Mathcad 12 оператора пространств имен. Ввести оператор комплексного сопряжения можно с помощью сочетания клавиш Shift+«'». Обязательное условие при этом — курсор должен находиться на тексте преобразуемого выражения, иначе

будет вставлена текстовая область, вводимая тем же сочетанием. Оператор Namespace, как вы помните, вводится сочетанием Ctrl+Shift+N.

В соответствии с выполняемыми функциями все операторы Mathcad можно разделить на семь групп. Рассмотрим каждую группу операторов по отдельности.

2.5.1. Операторы выражения

Операторы данной группы располагаются на панели Evaluation (Выражение) (рис. 2.6). Кроме того, некоторые из них можно найти и на других панелях семейства Math.

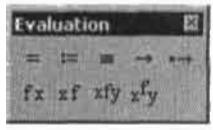


Рис. 2.6. Панель Evaluation

К группе операторов выражения мы будем относить следующие.

- Оператор численного вывода (Evaluate Numerically) «=».
- Оператор присваивания (Definition) «:=». Сочетание клавиш — Shift+«;».
- Оператор символьного вывода (Evaluate Symbolically) «→». Также вводится сочетанием Ctrl+«.».
- Оператор глобального присваивания (Global Definition). «≡». Вводится сочетанием Shift+«`».

Три первых оператора данной группы уже были довольно подробно рассмотрены ранее, поэтому останавливаться на них мы не будем. Разберем лишь различия между операторами простого и глобального присваивания.

Основное же различие между операторами присваивания состоит в том, что для того, чтобы найти, например, численное значение функции в некоторой точке, определение переменной с помощью оператора простого присваивания (Definition) должно быть сделано строго выше или левее ее самой. В том случае, если это сделать ниже функции, то ее значение не посчитается и будет выдано сообщение об ошибке: This variable is undefined (Эта переменная не определена). Если же в качестве оператора присваивания используется глобальное присваивание (Global Definition), то определение переменной можно сделать совершенно в любой точке документа.

Пример 2.25. Использование оператора глобального присваивания

$$f(x) := \sin(x)$$

$$f(x) \rightarrow \frac{1}{2} \cdot 3^{\frac{1}{2}}$$

$$x \equiv \frac{\pi}{3}$$

Глобально определять можно не только переменные, но и функции.

Очень важно также понимать, как глобальное присваивание сочетается с обычным. В том случае, если определение переменной с помощью «≡» осуществляется ниже,

чем присваивание с помощью «:=», то действовать обычное определение будет на участке документа ниже и правее себя вплоть до глобального. Если же оператор обычного присваивания расположить ниже глобального определения, то работать он будет абсолютно стандартно, и влияние Global Definition никак не будет проявляться. Правда, если новый оператор и функция определяются глобально через заданную выше как с помощью глобального, так и обычного присваивания переменную, то, независимо от той последовательности, в которой это задание было произведено, определение будет сделано исходя из глобального присваивания. Чтобы понять, почему так происходит, нужно знать, что Mathcad «читает» документ на предмет присваиваний дважды. В начале текст анализируется исходя из присутствующих глобальных определений, и лишь потом распознаются обычные присваивания.

Пример 2.26. Совместное использование оператора простого и глобального присваивания

$x \equiv 1$ $x = 1$

$x := 2$ $x = 2$

$y := x - 1$ $y = 1$

$z \equiv x - 1$ $z = 0$

Согласитесь, что правила использования оператора глобального присваивания довольно запутанны и не слишком понятны. Поэтому применять данный оператор следует максимально осторожно, иначе ошибки просто неизбежны. Впрочем, на практике глобальное определение используется очень редко, и, на мой взгляд, этого лучше не делать хотя бы для сохранения логической структуры документа.

При желании глобальное присваивание, аналогично обычному, можно отобразить с помощью символа простого равенства. Чтобы это сделать, в контекстном меню формулы (вызывается щелчком правой кнопкой мыши на области оператора) откройте всплывающее меню View Definition As (Видеть присвоение как), в котором переместите флажок со строки Triple Equal (Тройное равенство) на строку Equal (Равенство).

Помимо описанных выше, на панели Evaluation располагаются еще несколько операторов, которые по выполняемому смыслу скорее могут быть отнесены к заготовкам для создания операторов пользователя. О них мы поговорим позже.

2.5.2. Арифметические операторы

Операторы, выполняющие все основные арифметические действия, расположены на панели Calculator (Калькулятор) (рис. 2.7).

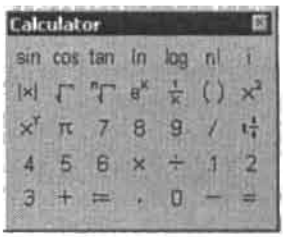


Рис. 2.7. Панель Calculator

Арифметические операторы — это наиболее часто используемые (после операторов вывода) и хорошо известные (хотя бы по обычному калькулятору) вычислительные символы. Поэтому специально описывать каждый из них нет никакого смысла. Стоит отметить лишь два довольно необычных оператора.

- Оператор смешанной дроби:

  Дает возможность задавать смешанные дроби в виде единого целого, а не произведения целой части на дробь десятичной части.

- Оператор линейного деления:

  Данный оператор полезен, так как, используя его, можно представлять в более компактной форме выражения типа «многоэтажной» дроби. Например:

$$\frac{x^2 + 5x + 7}{56 + x} \div \frac{x^3 - 1}{x^4 + 8x + 3} \text{ expand } \rightarrow \frac{x^6 + 8\cdot x^3 + 43\cdot x^2 + 5\cdot x^5 + 71\cdot x + 7\cdot x^4 + 21}{56\cdot x^3 - 56 + x^4 - x}$$

Такой важный оператор, как оператор умножения, в связи с существованием в математике нескольких форм записи этой арифметической процедуры можно отобразить в Mathcad по-разному. По умолчанию символ умножения отображается как точка с небольшим расстоянием до сомножителей (режим Narrow Dot). Чтобы сменить режим отображения знака умножения в выражении, следует щелкнуть на нем правой кнопкой мыши. При этом появится контекстное меню, в котором следует открыть меню View Multiplication as (Видеть умножение как). Здесь тип отображения оператора умножения можно выбрать из шести вариантов.

- Dot (Точка). Оператор умножения в виде точки.
- Narrow Dot (Близкая точка). Расстояние от точки до сомножителей несколько сужено по сравнению с режимом Dot. Режим, принятый по умолчанию.
- Large Dot (Большая точка). Символ умножения в виде толстой точки.
- X. Оператор умножения отображается в виде крестика.
- Thin Space (Небольшой промежуток). Символ умножения не отображается ни в какой форме, однако расстояние между сомножителями сохраняется довольно значительным.
- No Space (Без промежутка). Не отображается ни символ умножения, ни промежуток между сомножителями.

При выделении формулы оператор умножения будет визуализирован в том стиле, который определен по умолчанию.

Пример 2.27. Варианты отображения оператора умножения

$3 \cdot 2 = 6$ $\quad 3\,2 = 6$

$3 \times 2 = 6$ $\quad 3 \bullet 2 = 6$

$3 \cdot 2 = 6$ $\quad 32 = 6$

В том случае, если все умножения в документе должны быть отображены в стиле, отличном от установленного настройками по умолчанию, нужно эти настройки изменить. Для этого следует открыть, выполнив команду Tools ▸ Worksheet Options (Инструменты ▸ Параметры документа), специальную вкладку Display (Отображение) появившегося окна Worksheet Options. Настройкам вида оператора умножения здесь соответствует меню Multiplication (Умножение).

2.5.3. Вычислительные операторы

Операторы данной группы расположены на панели Calculus (Вычисления) (рис. 2.8).

Рис. 2.8. Панель Calculus

Вычислительные операторы служат для выполнения различных операций математического анализа: нахождения производной, суммы ряда, определенного и неопределенного интегралов, пределов, произведений. Весьма подробно вопрос о проведении расчетов с помощью операторов панели Calculus рассматривается в соответствующих главах. Поэтому останавливаться на технических деталях их применения сейчас мы не будем и лишь приведем несколько примеров.

Пример 2.28. Использование вычислительных операторов

$$\frac{d}{dx}\ln\left(x + a\cdot x^2\right) \rightarrow \frac{(1 + 2\cdot a\cdot x)}{\left(x + a\cdot x^2\right)} \qquad \int \frac{1}{\sqrt{1 - x}}\,dx \rightarrow -2\cdot(1 - x)^{\frac{1}{2}}$$

$$\int_0^{\frac{\pi}{8}} \frac{1}{\sin(x)}\,dx \rightarrow \infty \qquad \sum_{i=1}^{\infty} \frac{1}{e^i} \rightarrow \frac{1}{(-1 + \exp(1))} \qquad \lim_{x \rightarrow 0} \frac{\sin(x)}{x} \rightarrow 1$$

2.5.4. Матричные операторы

Операторы, позволяющие проводить основные матричные и векторные преобразования, расположены на панели Matrix (Матричные) (рис. 2.9).

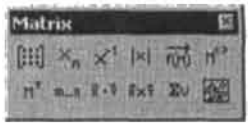


Рис. 2.9. Панель Matrix

Подробно вопрос об использовании матричных операторов будет рассмотрен в гл. 3 (там же разбирается то, как матрицы создаются и редактируются). А пока приведем лишь несколько примеров их применения.

Пример 2.29. Операторы матричных преобразований

$$M := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 0 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad M^T \rightarrow \begin{pmatrix} 1 & 4 & 7 \\ 2 & 0 & 8 \\ 3 & 6 & 9 \end{pmatrix} \quad M^{-1} \rightarrow \begin{pmatrix} \frac{-4}{5} & \frac{1}{10} & \frac{1}{5} \\ \frac{1}{10} & \frac{-1}{5} & \frac{1}{10} \\ \frac{8}{15} & \frac{1}{10} & \frac{-2}{15} \end{pmatrix}$$

$$M_{1,1} = 0 \qquad M^{\langle 2 \rangle} \rightarrow \begin{pmatrix} 3 \\ 6 \\ 9 \end{pmatrix} \qquad |M| = 60$$

2.5.5. Логические операторы

Очень важными операторами в Mathcad являются логические операторы, поскольку они используются при задании условий целого ряда задач. Расположены эти операторы на панели Boolean (Булевы) (рис. 2.10).

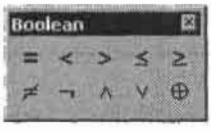


Рис. 2.10. Панель Boolean

Всего в Mathcad имеется 10 логических операторов. Такие операторы, как логические И (∧), Или (∨), Исключающее Или (⊕), Не (¬), полезны прежде всего при написании программ. Более широкие функции выполняет оператор логического равенства Bold equal «=» (помимо панели, вводится сочетанием Ctrl+=). Он используется в выражениях уравнений при их решении с помощью оператора solve или блока Given-Find. Также мы будем встречаться с операторами сравнения <, >, ≥, (Ctrl+0), ≤ (Ctrl+9), так как они используются при задании неравенств и написании алгоритмов.

Результатом применения логических операторов является одно из двух чисел: либо 0, если высказывание оказывается ложным, либо 1, если оно истинно. Таким образом происходит кодирование в форму, приемлемую для восприятия результата другими операторами или функциями.

Пример 2.30. Логические выражения

$$a := 1 \qquad b := 2$$

$$a = b = 0 \qquad a \neq b = 1 \qquad a > b = 0$$

$$a < b = 1 \qquad a \geq b = 0 \qquad a \leq b = 1$$

$$a \wedge b = 1 = 0 \qquad a \vee b \neq 6 = 1 \qquad a \oplus b \neq 6 = 0$$

$$\neg a \neq b = 1$$

Обратите внимание на различие при использовании простого («∨») и исключающего («⊕») ИЛИ. В первом случае условие считается выполненным, если оно справедливо

хотя бы для одного из элементов. Во втором же требуется выполнение условия строго для одного элемента. Если же условие выполняется для обоих элементов, то в качестве ответа оператор «⊕» выдаст 0.

Аналогично некоторым другим важнейшим операторам Mathcad, оператор логического равенства (Bold Equal) может быть представлен в разных формах. По умолчанию он визуализируется жирным символом «равно», однако при необходимости вы можете сменить его тип отображения на обычное «=». Для этого зайдите в контекстное меню формулы (щелкнув на ней правой кнопкой мыши), где в списке View Equality As (Видеть равенство как) переставьте флажок со строки Bold Equal (Жирное равно) на строку Equal (Равно). Те же изменения, но для всего документа, вы можете произвести на уже знакомой вкладке Display (Отображение) окна Worksheet options (Параметры документа) меню Tools (Инструменты).

2.5.6. Символьные операторы

Символьные операторы служат для выполнения целого ряда аналитических преобразований, таких как упрощение выражений, разложение на множители, замена переменных и многих других. Расположены они на панели Symbolic (Символьные), и по количеству их гораздо больше, чем операторов любых других типов, — 21. Применение символьных операторов подробно рассматривалось в предыдущем разделе, так что сейчас останавливаться на этом вопросе мы не будем.

2.5.7. Операторы программирования

Одна из важнейших возможностей системы Mathcad — программирование — реализуется благодаря наличию специальных операторов, расположенных на панели Programming (Программирование). В этой книге мы будем очень активно использовать язык программирования Mathcad, самостоятельно реализуя очень многие из используемых программой численных методов и решая другие нетривиальные проблемы. Пока же, чтобы у вас сложилось впечатление о том, как выглядят программы в Mathcad, приведем реализацию функции, выполняющей скалярное умножение векторов.

Пример 2.31. Создание функции скалярного произведения векторов

```
scalar(v1,v2) := | S ← 0
                 | if last(v1) ≠ last(v2)
                 |     return "length of v1 must be equal length of v2"
                 |     break
                 | for i ∈ 0..last(v1)
                 |     s ← v1_i·v2_i
                 |     S ← S + s
                 | S
```

$$A := \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \qquad B := \begin{pmatrix} 6 \\ 7 \\ 8 \end{pmatrix} \qquad C := \begin{pmatrix} 9 \\ 0 \end{pmatrix}$$

scalar(A, B) = 44 A·B = 44

scalar(A, C) = "length of v1 must be equal length of v2"

2.5.8. Создание оператора пользователя

Аналогично заданию функций пользователя, в Mathcad имеется возможность создания и собственных операторов. Для этого используются специальные заготовки панели Evaluation (Выражение).

В Mathcad существуют операторы двух типов: унарные и бинарные. Действие унарных зависит от одной переменной, и к ним относятся, например, оператор квадратного корня или модуля числа. Бинарный оператор определяется в общем случае функцией двух переменных.

Чтобы создать собственный унарный оператор, выполните следующую последовательность действий.

1. Задайте имя будущего оператора. Сделать это можно по тем же правилам, что и при определении функции. Однако учитывая то, что в математике операторы принято обозначать специальными символами, лучше для этого использовать подходящий значок, нежели слово. Некоторые специальные символы можно ввести и с клавиатуры, правда, в большинстве случаев для этого придется применять довольно тонкие технические ходы (что связано с тем, что в формульном режиме сочетание клавиш, соответствующее, например, символу $, вводит оператор ранжированной суммы), которые были рассмотрены в подразд. 2.2.3. Однако таким образом можно задать лишь несколько нематематических символов. В том случае, если вам нужно определить оператор в виде, например, символа суммы, можно попробовать перейти в текстовом режиме на шрифт Symbol (Символ). Однако можно поступить и проще. Чтобы ввести в документ некоторый редкий математический значок, нужно обратиться к разделу QuickSheets (Шпаргалки) Ресурсов Mathcad. Здесь в списке следует задействовать ссылку Extra Math Symbols (Дополнительные математические символы). При этом откроется таблица, содержащая 55 символов, применяющихся для обозначения различных операций в математике. Скопировав нужный значок, вставьте его в документ.
2. Откройте скобки и пропишите переменную, преобразованием которой определяется действие оператора. Создадим, например, оператор пересчета угла в радианах (а именно в таком форме работает по умолчанию с углами Mathcad) в градусы. Для обозначения задаваемого оператора используем символ в виде острого угла рассмотренной выше таблицы Extra Math Symbols (Дополнительные математические символы):

$$\angle(x)$$

3. Далее введите оператор присваивания и пропишите то математическое действие, которое должен выполнять создаваемый оператор. В нашем случае выражение функции, описывающей оператор:

$$\angle(x) := \frac{x}{\pi} \cdot 180$$

4. Описанные выше действия привели к заданию функции пользователя, пересчитывающей радианы в градусы, но ни в коей мере не оператора. Чтобы создать непосредственно оператор, введите специальную заготовку в виде двух черных маркеров, расположенную на панели Evaluation (Выражение):

$$\blacksquare\blacksquare$$

Заготовка эта имеет два типа, отличающиеся взаимным расположением оператора и переменной. Первый тип, Prefix Operator (Оператор-приставка), «fx», требует введения имени описывающей оператор функции на первом месте. Второй тип, Postfix Operator (Оператор-суффикс), требует обратной последовательности задания элементов (рис. 2.11).

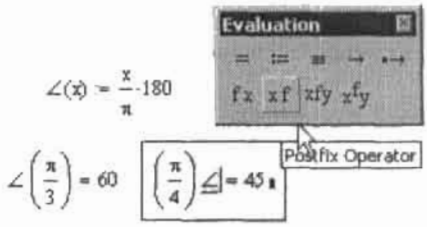


Рис. 2.11. Задание унарного оператора

Задание бинарного оператора ничем принципиально не отличается от определения унарного. Рассмотрим его на примере определения оператора, вычисляющего длину гипотенузы прямоугольного треугольника по длине катетов.

1. Сначала нужно задать функцию двух переменных, описывающую действие создаваемого оператора. Ее имя, аналогично созданию унарного оператора, лучше обозначить специальным символом. В нашем случае:

$$\nabla(x,y) := \sqrt{x^2 + y^2}$$

2. Далее следует ввести одну из двух возможных форм заготовки бинарного оператора пользователя. Первая форма, Infix Operator, продолжает идею унарных заготовок и содержит три маркера:

■ ■ ■

В первый вводится величина переменной x, во второй — имя описывающей оператор функции, в третий — значение переменной y.

Вторая форма бинарной заготовки, Tree Operator, более необычна и служит для представления оператора в форме графа:

При такой форме определения оператора имя функции вводится в вершину графа, величины переменных — в маркеры ветвей.

Пример 2.32. Создание бинарного оператора

$$\nabla(x,y) := \sqrt{x^2 + y^2}$$

$$3 \nabla 2 = 3.606 \qquad \begin{matrix} \nabla \\ 3 \quad 2 \end{matrix} = 3.606$$

2.6. Управление вычислениями

В том случае, если вы когда-нибудь писали на каком-либо языке программирования, вы, наверное, привыкли к тому, что редактирование текста программы и ее выполнение разнесены во времени. В Mathcad же по умолчанию и редактирование, и выполнение алгоритма происходят одновременно (слово алгоритм вполне приемлемо для характеристики расчетов в Mathcad, поскольку пусть и в не совсем традиционной, предназначенной для восприятия не программистами, а математиками, форме Mathcad является очень серьезной системой программирования). Это не всегда удобно, поэтому порой приходится использовать возможность непосредственного управления пользователем ходом вычислений. О том, как это делается, мы и поговорим в данном разделе.

2.6.1. Режимы вычислений

В Mathcad существует два режима вычислений: автоматический и ручной. В вышеприведенных примерах предполагалось, что включен определенный по умолчанию автоматический режим. В большинстве случаев использование его более оправдано, так как при этом результаты вычислений появляются в режиме реального времени, что позволяет производить корректировки и находить ошибки непосредственно при создании алгоритма решения задачи. Однако иногда автоматический режим может быть весьма неудобен. Проблемы могут возникать в связи с тем, что любое изменение в предшествующих условиях приводит к пересчету последующих выражений документа. Это обстоятельство, конечно, не играет никакой роли, если решаемая вами задача не слишком сложна в вычислительном плане — в этом случае пересчет займет неуловимые доли секунды. Однако если вы используете численные алгоритмы, требующие значительного числа операций (например, кратного интегрирования или решения жесткой системы дифференциальных уравнений), то время расчета может быть заметным даже на мощном современном компьютере. Естественно, что в этом случае ждать после замены одной буквы или числа при редактировании, пока просчитается весь документ, совершенно неэффективно. Поэтому стоит перейти в ручной режим (manual mode) выполнения расчетов. Чтобы это сделать, в подменю Calculate (Вычислить) меню Tools (Инструменты) снимите флажок Automatic Calculation (Автоматические вычисления).

При работе в ручном режиме, при задании выражений, вместо ответов после операторов вывода будут отображаться черные маркеры. Чтобы произвести расчет, системе нужно дать соответствующую команду. Однако сделать это можно по-разному.

- Чтобы пересчитать все формулы документа, используйте команду Calculate Worksheet (Подсчитать документ) подменю Calculate меню Tools.
- Зачастую на одном документе располагается несколько не связанных напрямую вычислительных алгоритмов. При этом, естественно, совершенно нецелесообразно пересчитывать их все, если внесенные изменения коснулись лишь одного. В подобной ситуации вместо команды Calculate Worksheet лучше использовать команду Calculate Now (Подсчитать). Данная команда пересчитывает формулы только видимой части документа. Регулировать ее работу можно, изменяя формат листа. Кстати, задействовать команду Calculate Now можно и не обращаясь к соответствующему меню. Для этого следует либо нажать клавишу F9 на клавиатуре, либо использовать специальную кнопку панели Standard (Стандартные) (рис. 2.12).

Рис. 2.12. Кнопка Calculate на панели Standard

Режим вычислений определяется независимо для каждого документа, поэтому при необходимости можно просто разнести малосовместимые в технике подсчета алгоритмы на разные листы.

2.6.2. Прерывание вычислений

Как правило, проследить процесс выполнения вычислений в Mathcad невозможно по причине их предельной быстроты. Однако в некоторых случаях время расчета может быть довольно ощутимым. В этом случае подсчитываемая формула будет взята в зеленую рамку, а курсор приобретет вид мигающей желтой лампочки. Иногда, особенно если ваш компьютер не слишком мощный или параллельно запущено несколько приложений, можно увидеть, как зеленая рамка расчета перескакивает с формулы на формулу.

Очень часто бывает необходимым прервать процесс вычислений. Это может быть связано с тем, что вы обнаружите ошибку в условии или поймете, что выбранный алгоритм в данных условиях не самый эффективный. Длительные вычисления созданной вами программы могут означать то, что вы что-то не предусмотрели при ее написании и ее цикл работает как бесконечный. Кроме того, в момент проведения расчетов в Mathcad невозможно осуществлять никаких других операций, что иногда создает неудобства.

Чтобы прервать вычисления, нужно нажать клавишу Esc. При этом вычисляемые выражения будут окрашены в красный цвет и появится специальная панель Interrupt Processing (Прерывание обработки). Если вы решили остановить процесс подсчета, нажмите OK.

Возобновить прерванный процесс вычислений можно, использовав одну из двух описанных выше команд подменю Calculate меню Tools.

2.6.3. Отключение вычислений отдельных выражений

Часто приходится проводить отладку части вычислительного алгоритма, проверяя его правильность многочисленными пересчетами. При этом может возникать очень существенная проблема, связанная с длительным временем расчета некоторых формул. Конечно, можно отключить автоматический режим и использовать команду Calculate Now (Вычислить). Однако иногда бывает невозможным (даже чисто по эстетическим причинам) разделить различные формулы алгоритма по разным листам, что делает описанный ход совершенно неэффективным. В подобных случаях можно использовать возможность Mathcad отключать вычисление отдельных формул.

Чтобы перевести определенное выражение в режим не вычисляемого, следует, открыв его контекстное меню, выбрать пункт Disable Evaluation (Не вычисляемое). При этом формула, переведенная в подобный режим, будет помечена черным прямоугольником, расположенным в ее правом верхнем углу. Например:

$$\sin(\pi) \rightarrow 0$$

Чтобы снова сделать формулу вычисляемой, в той же строке контекстного меню следует выбрать команду Enable Evaluation (Вычисляемая).

Помимо решения проблем, связанных с длительным вычислением формул, режим Disable Evaluation можно использовать в том случае, если на листе содержатся два взаимоисключающих выражения.

2.6.4. Оптимизация

Как уже отмечалось, огромные возможности открывает использование аналитических расчетов в Mathcad. В тех случаях, когда символьное решение существует, результат будет получен много точнее и быстрее, чем при численных расчетах. Особенно важно время расчета в случае решения задач, численные алгоритмы подсчета которых требуют значительного числа шагов. Наиболее яркий пример задачи такого типа — кратный определенный интеграл. Так, символьный результат в том случае, если все первообразные существуют, будет получен за доли секунды. Например:

$$\int_{-10}^{10}\int_{-100}^{100}\int_{-1000}^{1000}\frac{x^2}{y+z}\,dz\,dy\,dx \rightarrow \frac{-400000}{3}\cdot i\cdot\pi$$

Численный же ответ будет подсчитываться для интеграла такого вида десятки минут даже на современном компьютере.

Из приведенного примера следует вывод: при решении многих задач стоит попробовать подсчитать их аналитически и лишь при неудаче прибегать к использованию численных алгоритмов. Автоматизировать же такой подход к вычислениям можно, использовав такую возможность Mathcad, как оптимизация.

По умолчанию оптимизация выражений не проводится. Чтобы ее включить, следует установить флажок WorkSheet (Документ) в подменю Optimize (Оптимизировать) меню Tools (Инструменты). При этом все формулы документа сначала будут анализироваться на предмет символьного упрощения и лишь затем рассчитываться численно.

Применить оптимизацию можно и по отношению к отдельно взятой формуле или выражению. Для этого следует задействовать команду Optimize (Оптимизировать) контекстного меню формулы.

В том случае, если системе удастся успешно провести оптимизацию, она поставит в правом нижнем углу вычисляемой формулы красную звездочку. В случае неудачи звездочка будет синего цвета.

Пример 2.33. Успешно проведенная оптимизация

$$I := \int_{-10}^{10}\int_{-100}^{100}\int_{-1000}^{1000}\frac{x^2}{y+z}\,dz\,dy\,dx \quad I = -4.189i\times 10^5$$

При проведении оптимизации очень важна форма задания выражения. Например, если просто ввести тройной интеграл и поставить «=», возможности оптимизации использованы не будут. Чтобы оптимизация была осуществлена, интеграл должен быть задан как значение некоторой переменной.

При желании можно проанализировать и символьный результат, на основе которого был получен численный ответ. Для этого следует задействовать команду Show Optimization (Показать оптимизацию) контекстного меню оптимизированной формулы. В открывшемся окне Optimized result (Оптимизированный результат) будет отображен результат символьного подсчета.

2.6.5. Настройка системных вычислительных параметров

Все наиболее общие параметры вычислительного процесса в Mathcad можно настроить с помощью команд специальной вкладки Calculation (Вычисления) окна Worksheet Options (Параметры документа) (открывается с помощью соответствующей команды меню Tools (Инструменты).

Всего в Mathcad имеется пять вычислительных параметров.

- Recalculate automatically (Пересчитывать автоматически). По своим функциям полностью совпадает с параметром меню Tools (Инструменты) Automatic Calculation (Автоматические вычисления). В том случае, если в окошке рассматриваемой настройки установлен флажок, то любое изменение в документе будет приводить к пересчету того фрагмента, которого эти изменения коснулись.
- Use strict singularity checking for matrices (Использовать проверку матриц на сингулярность). Этот параметр очень важен при работе с некоторыми численными методами, оперирующими при расчетах матрицами. При его включении перед тем, как работать с новой матрицей, система будет проверять ее на сингулярность, что может помочь избежать в некоторых случаях получения ошибочного результата.
- Optimize expressions before calculating (Оптимизировать выражения перед вычислением). Параметр, отвечающий за включение процесса оптимизации. Соответствует команде Optimize (Оптимизировать) меню Tools (Инструмены).
- Use exact equality for boolean comparisons (Использовать точное равенство для булевых сравнений). По умолчанию, если два числа сравниваются посредством оператора логического равенства, они считаются равными, если отличие в них начинается с 12 знака мантиссы. Это оправдано ввиду погрешностей численных расчетов (вспомните пример из подразд. 2.3.2, когда алгебраически идентичные выражения давали близкие, но не одинаковые значения). Однако иногда важно установить, что два числа совпадают в точности. Чтобы с этой работой мог справиться оператор логического равенства, следует активизировать данную настройку.
- Use ORIGIN for string indexing (Использовать ORIGIN для индексирования в строках). Системная переменная ORIGIN определяет, с какого числа начинается отсчет индексов в векторах и матрицах. Если включить данную настройку, то ORIGIN будет задавать и то, как будут индексироваться символы в строках. По умолчанию отсчет символов ведется с нуля.

2.7. Математические константы

Наиболее распространенные математические константы в Mathcad являются предопределенными и отображаются своими традиционными символами. Ввести их можно либо с некоторых панелей (Calculator (Калькулятор) для π, e, мнимой единицы и Calculus (Вычислительные) для символа бесконечности), либо с помощью специальных сочетаний клавиш. Всего в Mathcad имеется шесть встроенных математических констант.

- π — число «Пи». Для его задания лучше использовать сочетание Ctrl+Shift+P.
- e — основание натурального логарифма. Вводится клавишей соответствующей буквы латинского алфавита.
- i (или j). Мнимая единица. Вводится последовательным нажатием клавиш 1+I (или 1+J).
- ∞ — символ бесконечности. В численных расчетах принимается равным числу 10^{307} (машинной бесконечности). Сочетание клавиш — Ctrl+Shift+Z.
- % — символ процента. В вычислениях принимается равным 0.01. Вводится с клавиатуры.
- NaN. «Не число». Новая константа, появившаяся только в Mathcad 12. Используется, если при импорте внешних данных некоторое значение не может быть преобразовано в корректное число.

Пример 2.34. Значения математических констант

$$e = 2.718 \qquad i = i \qquad \% = 0.01$$

$$\pi = 3.142 \qquad j = i \qquad NaN = NaN$$

$$\infty = 1 \times 10^{307}$$

Восприятие математических констант системой очень сильно зависит от используемого типа вычислений. При численных расчетах они воспринимаются как числа. При символьных же вычислениях происходит анализ непосредственно математического значения введенного выражения, содержащего константы. Особо следует отметить способность Mathcad использовать математические константы в результатах аналитических вычислений, что значительно повышает их корректность по сравнению с ответами численных алгоритмов.

Пример 2.35. Использование математических констант в условиях и в ответах

$$\sin\left(\frac{\pi}{12}\right) \rightarrow \frac{1}{4}\cdot 6^{\frac{1}{2}}\cdot\left(1-\frac{1}{3}\cdot 3^{\frac{1}{2}}\right) \qquad \int_{-\infty}^{\infty} e^{-x^2}\,dx \rightarrow \pi^{\frac{1}{2}} \qquad \ln(1+i) = 0.347 + 0.785i$$

2.8. Системные переменные

Системные переменные служат для управления точностью некоторых численных методов, определяют особенности задания массивов, а также параметры ввода-вывода данных. Чтобы изменить величины важнейших системных переменных для всего документа, следует обратиться к вкладке Built-In Variables (Системные переменные) уже знакомого нам окна Worksheet Options (Параметры документа) меню Tools (Инструменты) (рис. 2.13).

Всего с вкладки Built-In Variables можно задать шесть системных переменных.

- Array Origin (ORIGIN) (Начальный индекс массива). С помощью этой переменной можно определить, с какого целого числа системе начинать нумерацию строк и столбцов в массивах. По умолчанию переменная ORIGIN равна 0. В нашей же математике не принято выделять нулевые столбцы и строки, поэтому очень многие пользователи

предпочитают сменить ее значение на 1. В общем случае данная переменная может быть любым целым числом.

- Convergence Tolerance (TOL) (Точность сходимости). Самая важная для практики системная переменная Mathcad. В общем случае служит для задания точности численных методов решения уравнений, систем уравнений, поиска экстремума и интегрирования. Подробно использование TOL рассматривается в главах 8 (при решении уравнений и систем уравнений) и 10 (при интегрировании). Минимальное значение TOL соответствует точности численных вычислений в Mathcad и равно 10^{-17}. По умолчанию TOL равна 0.001.
- Constraint Tolerance (CTOL) (Граничная точность). Критерий точности для дополнительных условий при численном решении систем уравнений с помощью блока Given-Find (подробнее о CTOL читайте в гл. 8).
- Seed Value for random numbers (Начальная величина для случайных чисел). Параметр, определяющий работу некоторых генераторов случайных чисел. Подробно о нем мы поговорим в разд. 15.9.
- PRNPRECISION — параметр формата данных при выводе в файл.
- PRNCOLWIDTH — установка формата столбца при выводе в файл.

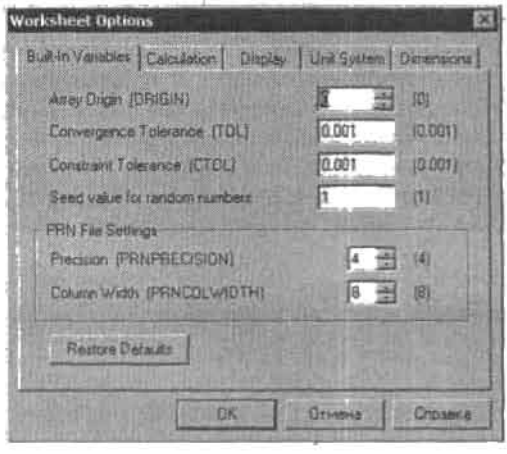


Рис. 2.13. Вкладка Built-In Variables

Более удобно, нежели при использовании описанной вкладки, изменять значения встроенных переменных простым их переопределением на рабочем листе.

Помимо описанных, в Mathcad есть еще несколько системных переменных. По причине специфичности они не были вынесены на вкладку Built-In Variables. Перечислим их.

- FRAME. Переменная, значением которой является номер текущего кадра. Используется при создании анимированных графиков.
- ERR. Переменная, характеризующая величину ошибки при приближенном решении систем уравнений с использованием блока Given-Minerr.
- 1L, 1M, 1S, 1T, 1C, 1K. Эти переменные хранят базовые размерности используемой системы единиц (к примеру, 1M соответствует единице массы, 1T — температуры и т. д.).

Глава 3. Матричные вычисления

Матричная алгебра была создана в середине XIX века английским математиком Артуром Кэли. Далеко не сразу исследователи смогли принять тот факт, что привычные арифметические действия (сложение, вычитание, умножение, возведение в степень) можно производить не только с числами, но и со специальными прямоугольными таблицами — матрицами. Тогда это казалось безумным и бесполезным изобретением, созданным лишь для того, чтобы еще больше запутать уже изрядно озадаченных появлением неевклидовых геометрий и комплексных чисел ученых. Но время показало, что Артур Кэли совершил воистину революционный прорыв в математике. Без него не были бы возможны те великие достижения в науке и технике, которыми может гордиться уже ушедший XX век. Особенно велика роль матричного исчисления в компьютерной математике: практически все численные методы на том или ином этапе работы своего алгоритма сводятся к решению систем линейных уравнений, которое производится матричными методами. Вообще, нельзя назвать ни одной области использования компьютера, в алгоритмах которой (в большей или меньшей степени) не использовались бы матрицы. При работе же с Mathcad владение методами создания и редактирования массивов данных — это одно из важнейших условий успеха при решении большинства задач. Поэтому эту тему мы рассмотрим особенно подробно.

Матричные вычисления в Mathcad можно условно разделить на три основных типа. К первому относятся такие элементарные действия над матрицами, как создание, извлечение из них данных, их умножение, сложение или скалярное произведение (в случае векторов). Для их реализации служат специальные операторы трех панелей семейства Math (Математические): Calculator (Калькулятор), Matrix (Матричные) (рис. 3.1) и Symbolic (Символьные).

Рис. 3.1. Рабочая панель Matrix

Ко второму типу можно отнести те матричные преобразования, которые требуют использования специальных функций и встроенных алгоритмов матричной алгебры,

таких как, например, функции вычисления матричных норм или сортировки элементов векторов по возрастанию. Функции этой группы можно найти в разделе Vector and Matrix (Векторные и матричные) списка Insert Function (Вставить функцию).

И, наконец, к третьему типу матричных вычислений следует отнести те задачи, решить которые можно только используя возможности системы программирования Mathcad. Например, только написав соответствующую программу, можно просуммировать элементы вложенного массива.

В этой главе мы рассмотрим все возможные операции с матрицами, исходя из принципа «от простого к сложному», но прежде чем рассматривать особенности решения матричных задач, разберемся, как матрицы создаются и редактируются.

3.1. Создание матриц и извлечение из них данных

Массивом (Array) называется любая упорядоченная последовательность элементов, адресуемых посредством целочисленных индексов. В качестве элементов массива могут использоваться числа, строки и другие массивы. Огромным достоинством Mathcad является то, что массивы в нем отображаются в традиционном для математики представлении в виде прямоугольных таблиц — матриц.

Исходя из структуры, все массивы Mathcad могут быть разделены на три большие группы.

- Векторы (Vectors). В Mathcad этим термином принято обозначать матрицы-столбцы, хотя иногда под ним понимают и матрицы-строки (что особенно характерно для нашей математики).
- Матрицы (Matrices). Двумерный массив. Представляет собой множество элементов, организованных по принципу шахматной доски.
- Тензоры (Nested Arrays). Так называемые вложенные массивы. Матрицы или векторы, элементы которых также являются массивами.

Если же разделять массивы по принципу особенностей задания их элементов, то можно выделить две группы.

- Векторы, матрицы или тензоры, при задании которых не существует прямой связи между величиной элемента и его индексами.
- Ранжированные переменные (range variables). Векторы, величина элементов которых напрямую определяется индексом.

Пример 3.1. Типы массивов

$$\text{Vector} := \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \qquad \text{Matrix} := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad \text{Tenzor} = \left[\begin{pmatrix} 1 \\ 2 \end{pmatrix} \quad \begin{pmatrix} 3 \\ 4 \end{pmatrix} \right]$$

Массив является основным элементом, на основе которого строится практически любая программа или численный алгоритм. Учитывая это, совсем не удивительно, что в Mathcad можно реализовать целых восемь способов задания массивов.

- Задание матрицы или вектора вручную с помощью команды Insert Matrix (Вставить матрицу).

- Определение матрицы последовательным заданием каждого элемента.
- Использование ранжированных переменных.
- Задание с помощью языка программирования.
- Применение встроенных функций.
- Через связь с другим приложением, например Excel.
- Создание таблицы данных.
- Чтение из внешнего файла.

Наиболее простым способом задания матрицы является использование специального окна Insert Matrix (Вставить матрицу) (рис. 3.2). Чтобы его вызвать, нажмите на панели Matrix (Матричные) одноименную кнопку с изображением квадратной матрицы с маркерами вместо элементов.

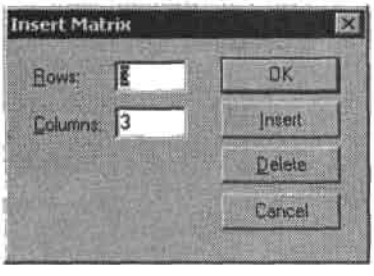


Рис. 3.2. Окно Insert Matrix (Вставить матрицу)

Чтобы открыть панель Matrix (Матрица), можно также воспользоваться сочетанием клавиш Ctrl+M или соответствующей командой меню Insert (Вставить).

Параметры создаваемой матрицы можно определить в окошках Rows (Строки) и Columns (Колонки). При этом количество элементов будущей матрицы не может превышать 100. Реально же в Mathcad можно работать с массивами любой объективно встречаемой при расчетах размерности. Данное же ограничение связано с тем, что заполнять вручную матрицы с числом элементов, большим 100, нецелесообразно — для этого используются некоторые другие методы из приведенного выше списка.

Определившись с размерами матрицы, нажмите OK (или Enter). При этом в документ будет вставлена заготовка с черными маркерами вместо элементов. Последовательно перемещая курсор с помощью мыши или клавиш управления курсором, введите в маркеры нужные значения (рис. 3.3). Элементы матрицы можно представить и символически: как в виде переменной, так и в виде выражения.

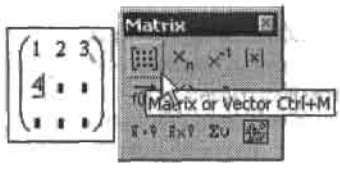


Рис. 3.3. Заполнение матрицы значениями

На практике обычно оперируют не матрицами, а их именами. Это связано с тем, что, во-первых, многие преобразования можно провести только с помощью имен, а во-вторых,

очень важной может оказаться задача экономии места документа, что особенно актуально в случае больших матриц. Чтобы вместо матрицы использовать имя, нужно определить ее как значение некоторой переменной. Делается это точно так же, как при задании переменной как числа: введением матричной заготовки с помощью окна Insert Matrix (Вставить матрицу) в маркер оператора присваивания.

Элементы матрицы могут быть как числами, так и строками или выражениями. Правила их определения при этом сохраняются те же, что и при задании простых переменных и функций. То есть если среди выражений или символов, выступающих в качестве элементов матрицы, есть неизвестные или параметры, то они должны обязательно быть численно определены выше. В противном случае матрица должна быть задана как функция. В случае определения элементов матрицы строковыми выражениями их текст обязательно должен быть взят в кавычки.

Пример 3.2. Задание матриц с элементами-переменными и элементами-функциями

$$a := 1 \qquad b := 2$$

$$M := \begin{pmatrix} \frac{a+b}{2} & \frac{a-b}{2} \\ \frac{2}{a+b} & \frac{2}{a-b} \end{pmatrix} \qquad M = \begin{pmatrix} 1.5 & -0.5 \\ 0.667 & -2 \end{pmatrix}$$

$$N(x,y) := \begin{pmatrix} \sin(x+y) & \cos(x-y) \\ \sin\left(\frac{x}{2}+y\right) & \cos\left(\frac{x}{2}-y\right) \end{pmatrix} \qquad N\left(\pi, \frac{\pi}{3}\right) \rightarrow \begin{pmatrix} \frac{-1}{2}\cdot 3^{\frac{1}{2}} & \frac{-1}{2} \\ \frac{1}{2} & \frac{1}{2}\cdot 3^{\frac{1}{2}} \end{pmatrix}$$

Есть только одно принципиальное отличие между определением значений переменных (или функций) и элементов матрицы — непосредственно в матрицу нельзя добавить матрицу (о том, как задаются тензоры (вложенные матрицы), мы поговорим ниже). В случае необходимости можно добавить к уже созданной и заполненной матрице строку или столбец. Чтобы это сделать, поставьте курсор к тому элементу, правее которого должен быть введен столбец (или ниже которого — в случае строки). Одним из описанных выше способов откройте окно Insert Matrix (Вставить матрицу), в окошках параметров которого введите 0 — для строк и 1 — для столбцов (или наоборот — в случае добавления строки). Нажмите Enter.

В случае заданной матрицы всегда можно получить значение любого ее элемента, используя его матричные индексы. Матричные индексы равняются номеру строки и столбца, на пересечении которых элемент находится. В нашей математике отсчет строк и столбцов принято начинать с 1. В программировании же начальные индексы обычно равняются 0. По умолчанию в Mathcad строки и столбцы также отсчитываются с 0. В том случае, если такая система вам неудобна или непривычна, вы можете изменить точку отсчета индексов на 1 (или любую другую). Чтобы это сделать, откройте окно Worksheet Options (Параметры рабочего листа) меню Tools, на вкладке Built-In Variables (Системные переменные) внесите соответствующие коррективы в величину параметра ORIGIN

(Точка отсчета). Впрочем, можно поступить и проще, сделав переопределение данной системной переменной вверху документа:

$$ORIGIN := 1$$

Однако, даже если вы привыкли к отсчету строк и столбцов с единицы, лучше все-таки используйте принятые по умолчанию правила. Это даст преимущества хотя бы в том, что вы лучше будете понимать примеры из справочной системы и алгоритмы других пользователей.

Итак, чтобы получить значение какого-то матричного элемента, нужно ввести имя матрицы с соответствующими индексами и поставить «=» (или, если элемент задан символьным выражением, «→»). Для задания индексов на панели Matrix (Матричные) имеется специальная кнопка Subscript (Индекс), которой соответствует клавиша «[». Нажав ее, вы увидите, что на месте будущего индекса, чуть ниже текста имени матрицы, появится черный маркер. В него через запятую следует ввести значения индексов. На первом месте при этом должен стоять номер строки, а на втором — столбца. При выделении элемента вектора нужно задать только индекс строки. Вообще, индексы могут быть определены и через выражения или специальные функции. В том случае, если элементы матрицы заданы как функции, то извлечены они должны быть в таком же виде.

Пример 3.3. Выделение элементов матрицы

$$V := \begin{pmatrix} \text{"Vector"} \\ 0 \end{pmatrix} \qquad M := \begin{pmatrix} 1 & 2 \\ \sin(\pi) & \frac{1}{2} \end{pmatrix} \qquad N(x,y) := \begin{pmatrix} \sin(x+y) & \cos(x-y) \\ \sin\left(\frac{x}{2}+y\right) & \cos\left(\frac{x}{2}-y\right) \end{pmatrix}$$

$$V_0 = \text{"Vector"} \qquad M_{0,0} = 1 \qquad M_{0,1} = 2 \qquad N(x,y)_{0,0} \to \sin(x+y)$$

$$V_1 = 0 \qquad M_{1,0} = 0 \qquad M_{1,1} = 0.5 \qquad N(\pi, 6\pi)_{0,0} \to 0$$

Аналогично извлечению отдельных элементов матрицы, используя индексы, можно задать новую матрицу. Для этого просто нужно проделать операцию присвоения для каждого элемента по отдельности. При этом размерность матрицы будет определяться элементом с наибольшими индексами. В том случае, если вы присвоите значения не всем элементам создаваемой матрицы, все остальные будут автоматически определены как 0.

Пример 3.4. Поэлементное задание матриц

$$M_{0,0} := 0 \qquad M_{0,1} := 1 \qquad N_{0,0} := \text{"Start"}$$

$$M_{1,0} := 2 \qquad M_{1,1} := 3 \qquad N_{2,1} := \text{"End"}$$

$$M = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \qquad N = \begin{pmatrix} \text{"Start"} & 0 \\ 0 & 0 \\ 0 & \text{"End"} \end{pmatrix}$$

Кстати, переопределяя значения элементов матриц, можно производить их редактирование. Особенно важен этот способ в случае больших массивов, не отображаемых на мониторе, а также при создании различного рода программ.

Пример 3.5. Форматирование матрицы переопределением элементов

$$V := (1 \quad 2 \quad 3 \quad 4)$$

$$V_{0,0} := \text{"Start"} \qquad V_{0,3} := \text{"End"}$$

$$V = (\text{"Start"} \quad 2 \quad 3 \quad \text{"End"})$$

Поэлементное задание — единственный способ определения тензоров (вложенных массивов) в Mathcad. Чтобы задать тензор, просто присвойте соответствующим элементам основного массива значения в виде матриц или векторов (или, что тоже возможно, других вложенных массивов). При этом совершенно не обязательно, чтобы все элементы тензора были определены как матрицы. При желании можно создать массив, содержащий данные всех типов.

Пример 3.6. Поэлементное задание тензора

$$T_{0,0} := (1 \quad 2) \qquad T_{0,1} := \pi$$

$$T_{1,0} := \text{"Tenzor"} \qquad T_{1,1} := \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$T = \begin{pmatrix} \{1,2\} & 3.142 \\ \text{"Tenzor"} & \{2,1\} \end{pmatrix}$$

Соответствующие матрицы тензора можно задать выше в виде переменных, а в качестве элементов тензора прописать их имена.

Пример 3.7. Альтернативный способ задания тензора

$$a := (1 \quad 2) \qquad b := \pi$$

$$c := \text{"Tenzor"} \qquad d := \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$T := \begin{pmatrix} a & b \\ c & d \end{pmatrix} \qquad T = \begin{pmatrix} \{1,2\} & 3.142 \\ \text{"Tenzor"} & \{2,1\} \end{pmatrix}$$

Внимательно изучив приведенные примеры, вы обнаружите, что при визуализации тензоров Mathcad отображает не сами элементы-матрицы, а цифры, отражающие их размерность, заключенные в фигурные скобки. В большинстве случаев такой подход вполне оправдан в связи с тем, что обычно вложенные массивы изучаются не напрямую, а извлечением нужных элементов. Однако иногда все же бывает полезно раскрыть тензор. Чтобы это сделать, обратитесь к уже хорошо нам знакомому окну Result Format (Формат результата). Здесь, на вкладке Display Options (Параметры отображения) установите флажок Expand Nested Arrays (Раскрывать вложенные массивы). При этом все тензоры будут визуализированы в развернутом виде. Так, вложенный массив примера 3.7 приобретет следующий вид:

$$T = \begin{bmatrix} (1 \quad 2) & 3.142 \\ \text{"Tenzor"} & \begin{pmatrix} 1 \\ 2 \end{pmatrix} \end{bmatrix}$$

Извлечь матрицу тензора можно точно так же, как элемент простого массива. Столь же просто можно узнать значение любого элемента его вложенной матрицы. Для этого следует просто использовать два набора индексов: первый должен соответствовать положению самой матрицы, второй — ее выделяемому элементу. Единственной тонкостью при этом является то, что выражение, соответствующее матрице, следует обязательно взять в скобки.

Пример 3.8. Выделение элементов матриц тензора

$$T = \begin{bmatrix} (1\ \ 2) & (3\ \ 4) \\ (6\ \ 4) & \begin{pmatrix} 1 \\ 2 \end{pmatrix} \end{bmatrix}$$

$$T_{1,0} = (6\ \ 4) \qquad T_{1,1} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$\left(T_{1,0}\right)_{0,0} = 6 \qquad \left(T_{1,1}\right)_{1,0} = 2$$

Помимо одного элемента, можно очень просто выделить и целые матричные столбцы. Чтобы это сделать, нужно использовать специальный оператор панели Matrix (Матричные) Matrix Column (Столбец матрицы) (также вводится сочетанием Ctrl+6). В том случае, если требуется выделить строку, матрицу необходимо транспонировать (оператор Matrix Transpose (Матричное транспонирование) той же рабочей панели).

Пример 3.9. Выделение из матрицы строки и столбца

$$M := \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

$$M^{\langle 0 \rangle} = \begin{pmatrix} 1 \\ 3 \end{pmatrix} \qquad M^{\langle 1 \rangle} = \begin{pmatrix} 2 \\ 4 \end{pmatrix} \qquad \left(M^T\right)^{\langle 0 \rangle} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \qquad \left(M^T\right)^{\langle 1 \rangle} = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$$

3.1.1. Ранжированные переменные

Одной из разновидностей задания массивов является использование так называемых ранжированных переменных.

Ранжированная переменная (от англ. range — ряд) — это разновидность вектора, особенностью которого является непосредственная связь между индексом элемента и его величиной. В Mathcad ранжированные переменные очень активно используются как аналог программных операторов цикла (например, при построении графиков).

Простейшим примером ранжированной переменной является вектор, значение элементов которого совпадает с их индексами. Для задания такой ранжированной переменной выполните следующую последовательность действий.

1. Введите имя переменной и оператор присваивания.
2. Поставив курсор в маркер значения переменной, нажмите кнопку Range Variable (Ранжированная переменная) панели Matrix (Матрица). При этом будет введена заготовка в виде двух маркеров, разделенных точками:

$$I := \blacksquare .. \blacksquare$$

Кроме того, вставить данную заготовку можно с помощью клавиши «;». Задать оператор ранжированной переменной, просто введя с клавиатуры последовательно две точки, нельзя.

3. В левый маркер заготовки ранжированной переменной введите ее первое значение (в нашем случае 0), в правый — последнее (например, 2):

$$I := 0..2$$

4. Выведите результат, поставив «=» после имени переменной:

$$I =$$

0
1
2

Обратите внимание, что шаг изменения ранжированной переменной при ее задании с помощью описанного способа постоянен и равен 1. Однако при необходимости его можно сделать и произвольным. Для этого нужно, поставив после левой границы интервала запятую, ввести второе значение ранжированной переменной. Разность между первым и вторым ее значением и определит шаг. Границы изменения ранжированной переменной задаются произвольным образом. Так, например, если точка начала изменения ранжированной переменной равна 5 и она должна уменьшаться с шагом 3 до –1, то ее определение необходимо сделать следующим образом:

$$k := 5, 2..-1$$

Результатом будет следующий вектор:

$$k =$$

5
2
-1

Нужно отметить, что во многом использование ранжированных переменных основано на том, что большинство математических действий в Mathcad над векторами осуществляется точно так же, как над простыми числами. Так, например, существует возможность вычисления значений практически любой встроенной и пользовательской функции от вектора. При этом в качестве результата будет выдан вектор, составленный из значений функции при величинах переменных, равных соответствующим элементам исходного вектора.

Пример 3.10. Вычисление вектора значений встроенной и пользовательской функции

$$X := \begin{pmatrix} \frac{\pi}{6} \\ \frac{\pi}{2} \\ \frac{\pi}{3} \end{pmatrix} \qquad \sin(X) = \begin{pmatrix} 0.5 \\ 1 \\ 0.866 \end{pmatrix} \qquad \frac{\sin(X)}{1 + \cos(X)} = \begin{pmatrix} 0.268 \\ 1 \\ 0.577 \end{pmatrix}$$

Аналогичным образом можно использовать вектор значений переменной, определенный с помощью ранжированной переменной. У такого способа есть огромное достоинство перед заданием вектора вручную: действительно, если, например, нужно построить график некоторой функции на промежутке от –10 до 10 с шагом 0,1, то на то, чтобы просчитать «вручную» все 200 значений, потребовалось бы весьма значительное время.

При необходимости шаг изменения переменной, по которой вычисляется вектор значений определенной зависимости, можно задать и непостоянным. Для этого нужно просто определить его с помощью конкретной функции, изменяющейся по некоторому закону, исходя из значений ранжированной переменной.

Пример 3.11. Вычисление вектора значений функции от неравномерно изменяющейся переменной

$$i := 0..3 \qquad x_i := i^2\pi - \frac{\pi}{3} \qquad f(x) := \cos(x)$$

$$i = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \end{pmatrix} \qquad x = \begin{pmatrix} -1.047 \\ 2.094 \\ 11.519 \\ 27.227 \end{pmatrix} \qquad f(x) = \begin{pmatrix} 0.5 \\ -0.5 \\ 0.5 \\ -0.5 \end{pmatrix}$$

Обратите внимание, что вектор переменной в примере 3.11 задан не совсем как функция. Скорее, его способ определения можно отнести к поэлементному заданию. Все дело в том, что ранжированная переменная работает наподобие цикла в программировании (кстати, для замены циклов она и была введена в Mathcad). То есть все выражения, которые к ней относятся, просчитываются по отдельности непосредственно при получении ее нового значения, а не тогда, когда будет вычислен весь ее вектор. В этом заключается принципиальное отличие ранжированной переменной от простых векторов.

Используя две или более ранжированные переменные, можно имитировать вложенные циклы. Это позволяет, например, формировать матрицы значений функций двух переменных. Подобные матрицы активно используются при построении поверхностей. В примере 3.12 показано, как можно создать матрицу размерности N×N, содержащую числа от 0 до N^2-1.

Пример 3.12. Задание матрицы с помощью ранжированных переменных

$$i := 0..3 \qquad j := 0..3 \qquad S_{i,j} := 4 \cdot i + j$$

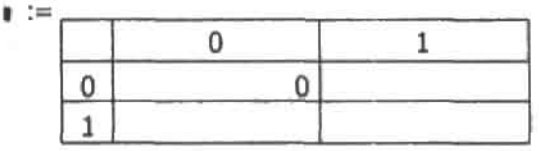
$$S = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{pmatrix}$$

3.1.2. Таблицы

Все экспериментальные данные обрабатываются в Mathcad в виде матриц. Однако использовать описанные выше стандартные методы задания массивов в этом случае

крайне неудобно. Более того, можно утверждать, что это просто невозможно: так, если размерность матрицы данных больше 10·10 (а в статистике так оно чаще всего и бывает), то использовать окно Insert Matrix (Вставить матрицу) невозможно, а поэлементное определение потребует много времени. Кроме того, очень большие матрицы в Mathcad просто не визуализируются.

Разрешить все описанные проблемы можно, используя так называемую таблицу ввода (Input Table). Чтобы ее вызвать, задействуйте команду Insert ▸ Data ▸ Table (Вставить ▸ Данные ▸ Таблица) главного меню или же активизируйте команду Insert ▸ Table (Вставить ▸ Таблица) контекстного меню рабочего поля (соответствующая этой команде кнопка имеется и на панели Standard).В том случае, если вы правильно выполните описанные действия, в ваш документ будет введена следующая заготовка:



Присвоив будущей матрице определенное имя, попробуйте определиться с ее размерами. Если она не очень большая, можно сразу расширить пустую таблицу до нужной величины. Для этого следует использовать специальные черные маркеры, появляющиеся на контуре таблицы при ее выделении. Сам процесс форматирования величины табличной заготовки абсолютно стандартен для Windows и выполняется протаскиванием при нажатой левой кнопке мыши. Никаких ограничений на размеры таблица ввода не имеет.

Чаще таблицу ввода не разворачивают в полную величину, а введение значений осуществляют с помощью клавиш управления курсором. Это помогает не только сэкономить место на документе, но и ускорить процесс задания матрицы.

В том случае, если вы когда-нибудь работали в Excel, процесс заполнения таблицы ввода вам покажется очень знакомым. Однако даже если вы никогда не сталкивались с этой программой, технику введения значений вы моментально усвоите по причине ее предельной простоты. Во многом создание таблицы повторяет заполнение обычных матриц, однако одно существенное отличие все же имеется: в таблицах нельзя использовать формулы.

Так как таблицы являются для Mathcad такими же матрицами, как заданные стандартными способами, с ними можно проводить все те же преобразования, что и со стандартными по виду массивами. Кстати, если вы захотите отобразить содержание таблицы через ее имя, оно визуализируется (при стандартных настройках) именно как простая матрица (рис. 3.4).

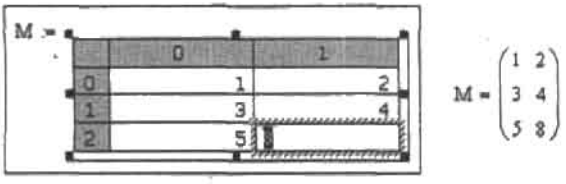


Рис. 3.4. Задание матрицы с помощью таблицы ввода

Схожесть таблицы ввода Mathcad с Excel не случайна: одно из основных ее применений связано с организацией согласованной работы этих программ. Так, вы можете очень просто произвести обработку данных в Mathcad в том случае, если они сохранены в формате Excel. Для этого достаточно просто скопировать их и вставить затем (с помощью команды контекстного меню Paste Table (Вставить таблицу)) в пустую таблицу ввода.

Существует и другой способ импорта данных из Excel. Однако о нем мы поговорим в гл. 16.

Использование таблиц предельно упрощает задачу выделения из матрицы отдельных столбцов и особенно строк и подматриц. Для этого, аналогично все тому же Excel, нужно с помощью левой кнопки мыши выделить нужный фрагмент и скопировать его с помощью команды Copy Selection (Копировать выделение) контекстного меню. Затем скопированный фрагмент массива нужно вставить либо в пустую таблицу ввода, либо в маркер переменной (при этом он отобразится в виде простой матрицы).

3.1.3. Способы отображения массивов

В Mathcad существует два типа визуализации массивов: таблицы и матрицы.

Пример 3.13. Типы отображения матриц

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

M =

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

По умолчанию данные обычно отображаются в виде матриц (исключение составляют ранжированные переменные, векторы которых всегда представляют собой таблицы). Если вы хотите изменить стиль определенной матрицы, то дважды щелкните на ней мышью. При этом откроется уже знакомое нам окно Result Format (Формат результата), на вкладке Display Options (Параметры отображения) которого выберите список Matrix Display Style (Стиль отображения матриц).

В данном списке содержится три строки.

- Automatic (Автоматически). Стиль отображения матриц и векторов определяется установками по умолчанию системы.
- Matrix (Матрица). Массив будет представлен в виде матрицы.
- Table (Таблица). Массив отображается как таблица.

Соответственно вам нужно выбрать наиболее подходящий тип визуализации и нажать OK.

В том случае, если вы работаете с типом отображения Table (Таблица), вы можете произвольным образом определять расположение имени матрицы относительно ее самой. Для этого вам нужно открыть контекстное меню таблицы, в котором затем следует выбрать список Alignment (Выравнивание).

В списке вам нужно определиться с выбором между пятью вариантами стиля таблицы.

- Top (Верх). Имя матрицы располагается на уровне первой строки таблицы ввода.
- Center (Центр). Имя массива отображается посередине таблицы.
- Bottom (Низ). Уровень имени матрицы соответствует последней строке таблицы.

- Above (Выше). Имя матрицы располагается выше самой таблицы.
- Below (Ниже). Имя массива отобразится ниже таблицы ввода.

По умолчанию таблицы ввода всегда окаймлены специальными серыми метками, содержащими порядковые номера их строк и столбцов. Однако при желании эти элементы оформления можно и удалить. Для этого с помощью соответствующей команды контекстного меню таблицы откройте окно Properties (Свойства).

В открывшемся диалоговом окне необходимо убрать метку на параметре Show column/row labels (Отображать пометки к столбцам/строкам). В меню Font (Шрифт) рассматриваемого окна вы можете настроить оптимальный стиль шрифта, используемого в таблицах.

Пример 3.14. Отображение таблицы с метками и без

M =

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

M =

1	2	3
4	5	6
7	8	9

3.2. Элементарные матричные вычисления

Все простейшие операции матричной алгебры реализованы в системе Mathcad с помощью операторов. Вид каждого из них полностью соответствует принятым в математике обозначениям.

Мы рассмотрим операции как над матрицами, так и над векторами. Вектором называется матрица размерности N×1 (то есть содержащая N строк и только 1 столбец), или матрица-столбец. Многие матричные операции универсальны: они аналогичны как для матриц, так и для векторов (сложение, вычитание, умножение на число). Другие же операции могут быть применимы только к квадратным матрицам (размерностью N×N) (например, оператор вычисления обратной матрицы) или же только к векторам (векторное произведение или суммирование элементов). Некоторые операторы по-разному действуют на матрицы и векторы (например, оператор Determinant (Определитель) является оператором вычисления определителя в случае матриц и одновременно оператором модуля вектора).

В математике иногда вектором считают и матрицу-строку. Однако в Mathcad все операторы векторных преобразований работают только в случае матриц-столбцов. Поэтому, если возникает необходимость произвести какое-то действие над вектором, представленным матрицей-строкой, ее следует просто предварительно транспонировать.

3.2.1. Сложение и перемножение матрицы и скаляра

В Mathcad к матрице можно прибавлять (или отнимать от нее) любое число. При этом оно будет прибавлено ко всем (или вычтено из всех) элементов исходной матрицы. При умножении матрицы на скаляр на него умножается каждый элемент исходной матрицы. Аналогично умножению, матрицу можно разделить на скаляр. Во всех операциях матрица и скаляр могут быть представлены и символически: как в виде буквы, так и в виде выражения. При этом в качестве оператора вывода следует использовать оператор символьного вывода «→».

Пример 3.15. Сложение и перемножение матрицы и скаляра

$$3+\begin{pmatrix}1&2\\3&4\end{pmatrix}=\begin{pmatrix}4&5\\6&7\end{pmatrix}\qquad a\cdot\begin{pmatrix}a+1&a^2\\b\cdot a&a-1\end{pmatrix}\rightarrow\begin{bmatrix}(a+1)\cdot a&a^3\\b\cdot a^2&(a-1)\cdot a\end{bmatrix}$$

3.2.2. Сложение и вычитание матриц

Чтобы сложить или вычесть матрицы, используются привычные символы «+» или «–» (вводятся с клавиатуры или с помощью соответствующих команд меню Calculator (Калькулятор)), которые помещаются между соответствующими матрицами (или именами матриц). При этом к каждому элементу M_{ij} первой матрицы прибавится (или вычтется из него) элемент $M1_{ij}$ второй матрицы. Результатом будет третья матрица, элементы которой являются суммой (разностью) соответствующих элементов суммируемых (вычитаемых) матриц. Естественно, матрицы должны быть одинаковой размерности, иначе будет выдано сообщение об ошибке. Кроме того, в выражениях матричного сложения или вычитания можно использовать и коэффициенты.

Пример 3.16. Сложение и вычитание матриц

Даны матрицы:

$$A:=\begin{pmatrix}1&-2&6\\4&3&-8\\2&-2&5\end{pmatrix}\qquad B:=\begin{pmatrix}1&0&-2\\6&1&-10\\6&4&-3\end{pmatrix}$$

Найти: A+B; матрицу X, удовлетворяющую условию 3A–2X=B.

$$A+B=\begin{pmatrix}2&-2&4\\10&4&-18\\8&2&2\end{pmatrix}$$

$$X:=\frac{3A-B}{2}\qquad X=\begin{pmatrix}1&-3&10\\3&4&-7\\0&-5&9\end{pmatrix}$$

3.2.3. Матричное умножение

Матричное умножение выполняется следующим образом: все элементы нулевой (как вы помните, по умолчанию отсчет строк и столбцов в Mathcad начинается с 0) строки первой матрицы умножаются на соответствующие элементы нулевого столбца второй матрицы, и затем эти произведения суммируются. Полученное значение определяется как первый элемент нулевой строки матрицы-результата. Далее нулевая строка первой матрицы аналогично умножается на первый столбец второй матрицы, и значение заносится как второй элемент верхней строки матрицы-результата. При умножении следующей строки первой матрицы на столбцы второй будет сформирована первая строка результирующей матрицы. И так далее — до тех пор, пока не будут перемножены все строки. Так, при умножении матрицы размерности N×M на матрицу размерности M×K будет получена матрица размерности N×K. Естественно, перемножать матрицы можно лишь в том случае, если количество столбцов первой равно числу строк второй.

Перемножить матрицы можно либо воспользовавшись клавишей «*», либо с помощью специальной команды Dot Product (Умножение) панели Matrix (Матричные). При этом знак умножения, представляемый по умолчанию как «·», можно изменить на принятый для матриц знак «×». Для этого следует воспользоваться командой контекстного меню View Multiplication As (Видеть произведение как).

Перемножать матрицы можно и в том случае, когда элементы их представлены символами или выражениями.

Пример 3.17. Матричное умножение

Найти произведение ABC для трех матриц:

$$A := \begin{pmatrix} 2 & 5 & 0 \\ 0 & 1 & 3 \end{pmatrix} \qquad B := \begin{pmatrix} 1 & -3 \\ 5 & 0 \\ 10 & -8 \end{pmatrix} \qquad C := \begin{pmatrix} -1 & 1 \\ 0 & -2 \end{pmatrix}$$

$$A \cdot B \cdot C = \begin{pmatrix} -27 & 39 \\ -35 & 83 \end{pmatrix}$$

Символьное умножение матриц:

$$\begin{pmatrix} \sqrt{a} & b - x & c \\ d + a & g^2 & f \end{pmatrix} \cdot \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix} \rightarrow \begin{bmatrix} a^{\frac{3}{2}} & (b - x) \cdot b & c^2 \\ (d + a) \cdot a & g^2 \cdot b & f \cdot c \end{bmatrix}$$

Если вы попытаетесь перемножить матрицы несоответствующего размера, будет выдано следующее сообщение об ошибке: These array dimensions are incompatible (Размеры этих массивов несовместимы), а само произведение окрасится красным цветом.

3.2.4. Транспонирование матриц

Транспонированием называется матричная операция, переводящая матрицу размерности M×N в матрицу размерности N×M. Иначе говоря, при транспонировании строки исходной матрицы превращаются в столбцы, а столбцы — в строки. Оператор транспонирования (Transpose) находится на панели Matrix (Матричные), а также его можно вставить с помощью сочетания клавиш Ctrl+1 (перед тем как ввести оператор транспонирования, матрицу следует выделить).

Транспонирование можно провести и для матриц, чьи элементы определены символически. При этом следует использовать оператор «→», расположенный на панели Symbolic (Символьные).

Пример 3.18. Транспонирование матриц

Даны матрицы:

$$A := \begin{pmatrix} -2 & 1 \\ 3 & 4 \\ -5 & -7 \end{pmatrix} \qquad B := \begin{pmatrix} 1 & 5 & -6 \\ 2 & -4 & 3 \end{pmatrix}$$

Найти матрицу $X=3A+B^T$:

$$X := 3A + B^T$$

$$B^T = \begin{pmatrix} 1 & 2 \\ 5 & -4 \\ -6 & 3 \end{pmatrix} \qquad X = \begin{pmatrix} -5 & 5 \\ 14 & 8 \\ -21 & -18 \end{pmatrix}$$

Символьное транспонирование:

$$\begin{pmatrix} 0 & b & c \\ c & 0 & b \\ b & c & 0 \end{pmatrix}^T \rightarrow \begin{pmatrix} 0 & c & b \\ b & 0 & c \\ c & b & 0 \end{pmatrix}$$

3.2.5. Определитель матрицы

Определитель — это число (или выражение), которое прежде всего характеризует линейную независимость строк (или столбцов) матрицы. Значение определителя в математике огромно, а вычисление его порой бывает весьма сложным. Поэтому наличие такого оператора в системе Mathcad следует оценить как ее очень большой плюс.

Ввести оператор определителя (Determinant) можно либо с помощью панели Matrix (Матричные), либо сочетанием клавиш Shift+«\» (предварительно матрица должна быть выделена). Вид определителя в Mathcad соответствует принятому в математике.

Поскольку в Mathcad для нахождения определителя и модуля используется один и тот же оператор, в его контекстном меню при вычислении определителя следует выбрать пункт Square Matrix Determinant (Определитель квадратной матрицы).

Пример 3.19. Вычисление определителя четвертого порядка

$$\left|\begin{pmatrix} 4 & 6 & 3 & 4 \\ 5 & 2 & 1 & 8 \\ 1 & 3 & 1 & 2 \\ 3 & 1 & 2 & 1 \end{pmatrix}\right| = 17$$

Вычислить определитель можно как численно, так и символически. В приведенном ниже примере подсчитывается в символьном виде якобиан для преобразования тройного интеграла в сферическую систему координат.

Пример 3.20. Перейдя к сферическим координатам, вычислить тройной интеграл

$$\int\int\int_{(V)} \sqrt{\left(x^2 + y^2 + z^2\right)^4}\, dx\, dy\, dz,$$

где (V) — верхняя половина шара

$$x^2 + y^2 + z^2 \le R^2$$

Переходим к сферическим координатам:

$$X(r,\phi,\psi) := r \cdot \sin(\phi) \cdot \cos(\psi)$$
$$Y(r,\phi,\psi) := r \cdot \sin(\phi) \cdot \sin(\psi)$$
$$Z(r,\phi,\psi) := r \cdot \cos(\phi)$$

Вычисляем якобиан в символьном виде:

$$J := \left| \begin{pmatrix} \frac{d}{dr}X(r,\phi,\psi) & \frac{d}{d\phi}X(r,\phi,\psi) & \frac{d}{d\psi}X(r,\phi,\psi) \\ \frac{d}{dr}Y(r,\phi,\psi) & \frac{d}{d\phi}Y(r,\phi,\psi) & \frac{d}{d\psi}Y(r,\phi,\psi) \\ \frac{d}{dr}Z(r,\phi,\psi) & \frac{d}{d\phi}Z(r,\phi,\psi) & \frac{d}{d\psi}Z(r,\phi,\psi) \end{pmatrix} \right| \qquad J \text{ simplify } \rightarrow \sin(\phi)\cdot r^2$$

Задаем пределы интегрирования.

Поскольку область (V) ограничена верхней половиной шара, угол ω изменяется в пределах от 0 до π/2:

$$0 \le \phi \le \frac{\pi}{2} \qquad 0 \le \psi \le 2\pi \qquad 0 \le r \le R$$

Вычисляем тройной интеграл:

$$\int_0^{2\pi}\int_0^{\frac{\pi}{2}}\int_0^{R} \sqrt{\left[(X(r,\phi,\psi))^2 + (Y(r,\phi,\psi))^2 + (Z(r,\phi,\psi))^2\right]^4} \cdot J \, dr\,d\phi\,d\psi \rightarrow \frac{2}{7}\cdot\pi\cdot\left(R^8\right)^{\frac{1}{2}}\cdot R^3$$

$$\frac{2}{7}\cdot\pi\cdot\left(R^8\right)^{\frac{1}{2}}\cdot R^3 \;\left|\begin{array}{l}\text{assume},R=\text{real}\\ \text{simplify}\end{array}\right. \rightarrow \frac{2}{7}\cdot\pi\cdot R^7$$

Для наглядного представления якобиана пришлось прибегнуть к оператору simplify (Упростить) рабочей панели Symbolic (Символьные), так как если этого не сделать, то определитель будет выдан в виде суммы 4 слагаемых. А вообще, число членов в сумме для определителя равно N!, где N – размерность квадратной матрицы, для которой этот определитель вычисляется. В случае же якобиана для сферической системы координат число членов такой суммы получается равным 4, поскольку производная координаты Z по одному из углов равна 0.

Заметьте, что при подсчете интеграла мы получили довольно громоздкое выражение, которое затем пришлось упростить, задействовав два оператора. С помощью оператора simplify (Упростить), как вы помните, можно эффективно упрощать выражения, содержащие степень, однако в нашем случае его применение приводит к появлению за-

гадочной функции csgn. Связано это с тем, что Mathcad по умолчанию рассматривает все числа как комплексные. Оператор assume позволяет ввести ограничения на параметр R, поскольку радиус может быть только действительным числом (что задается с помощью служебного слова real). Так, путем несложных преобразований мы пришли к корректному ответу.

В Mathcad определитель находится с помощью LU-разложения. Как это делается, демонстрируется в гл. 8 (в разделе, посвященном решению систем линейных уравнений).

3.2.6. Модуль вектора

Модулем (или абсолютной величиной) вектора называется длина изображающего его отрезка. Модуль вектора (Vector Magnitude) по определению равен квадратному корню из суммы квадратов его элементов. Геометрический смысл модуля вектора — это длина отрезка, соединяющего точку начала координат и точку, координаты которой численно равны соответствующим элементам вектора.

В случае вектора оператор Determinant (Определитель) является оператором модуля вектора, на что следует указать в его контекстном меню, выбрав пункт Absolute Value (Абсолютная величина).

Модуль вектора может быть вычислен и символически.

Пример 3.21. Модуль вектора

$$\left|\begin{pmatrix}1\\4\\5\end{pmatrix}\right| = 6.481 \qquad \left|\begin{pmatrix}x\\y\\z\end{pmatrix}\right| \rightarrow \left[(|x|)^2 + (|y|)^2 + (|z|)^2\right]^{\frac{1}{2}}$$

3.2.7. Векторное произведение

Векторным произведением (Cross Product), по определению, называется вектор, длина которого равна произведению длин исходных векторов и синуса угла между ними, а направление его совпадает с направлением перпендикуляра к плоскости этих двух векторов (по правилу «буравчика»). Символически векторное произведение находится как определитель следующей матрицы (для случаев векторов в трехмерном пространстве):

$$\begin{pmatrix}\vec{i} & \vec{j} & \vec{k}\\ x1 & x2 & x3\\ y1 & y2 & y3\end{pmatrix}$$

где i, j, k — единичные, взаимно перпендикулярные векторы, x1, x2, x3, y1, y2, y3 — координаты (элементы) перемножаемых векторов.

Обозначается векторное произведение в Mathcad символом «×» (соответствует принятому в математике и физике). Ввести оператор векторного произведения можно либо с панели Matrix (Матричные) (кнопка Cross Product), либо сочетанием клавиш Ctrl+8.

Пример 3.22. Сила F = (1, –3, –4) приложена в точке A(3, 4, 5). Найти величину и направляющие косинусы момента этой силы относительно точки B(1, 8, 9)

$$F := \begin{pmatrix} 1 \\ -3 \\ -4 \end{pmatrix} \qquad A := \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix} \qquad B := \begin{pmatrix} 1 \\ 8 \\ 9 \end{pmatrix}$$

Момент силы относительно точки B вычисляется как векторное произведение a·F, где a — вектор, направленный к точке приложения силы:

$$a := A - B \qquad a \times F = \begin{pmatrix} 4 \\ 4 \\ -2 \end{pmatrix}$$

Величина момента силы определяется как модуль вектора a·F:

$$|a \times F| = 6$$

Направляющие косинусы момента силы (косинусы углов α, β, γ, образованных вектором с осями координат) вычисляются как отношения соответствующих координат вектора к его длине:

$$\cos_{\alpha\beta\gamma} := \frac{a \times F}{|a \times F|} \qquad \cos_{\alpha\beta\gamma} = \begin{pmatrix} \frac{2}{3} \\ \frac{2}{3} \\ -\frac{1}{3} \end{pmatrix}$$

Чтобы отобразить результат в виде простых дробей, выполните на нем двойной щелчок. В открывшемся окне Result Format (Формат результата) перейдите на вкладку Number Format (Формат чисел), в меню Format (Формат) выберите пункт Fraction (Дробь).

Векторное произведение вычисляется системой Mathcad и символически:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \times \begin{pmatrix} 2x \\ 4y \\ \frac{1}{2z} \end{pmatrix} \rightarrow \begin{pmatrix} \frac{1}{2}\cdot\frac{y}{z} - 4\cdot z\cdot y \\ 2\cdot z\cdot x - \frac{1}{2}\cdot\frac{x}{z} \\ 2\cdot x\cdot y \end{pmatrix}$$

3.2.8. Скалярное произведение векторов

Скалярным произведением (Vector inner product) векторов называется число (или выражение), равное произведению длин перемножаемых векторов и косинуса угла между ними. Иначе скалярное произведение можно найти, сложив попарно перемноженные координаты (элементы) векторов. Скалярное произведение в математике принято обозначать символом «·». Аналогично оно представляется и в Mathcad. Чтобы

перемножить скалярно два вектора, можно либо воспользоваться специальной кнопкой Dot Product панели Matrix (Матричные), либо просто ввести с клавиатуры оператор умножения («*»).

Пример 3.23. Дан треугольник с вершинами A(1, 0, –5), B(–1, 1, –4), C(2, –5, –1). Найти внешний угол j при вершине B (рис. 3.5)

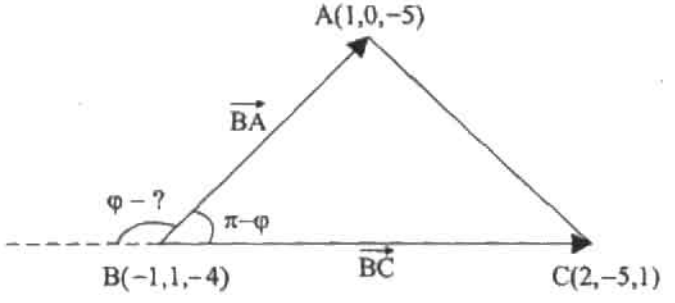


Рис. 3.5. Чертеж к задаче о нахождении внешнего угла треугольника

Задаем точки треугольника в векторной форме:

$$A := \begin{pmatrix} 1 \\ 0 \\ -5 \end{pmatrix} \qquad B := \begin{pmatrix} -1 \\ 1 \\ -4 \end{pmatrix} \qquad C := \begin{pmatrix} 2 \\ -5 \\ -1 \end{pmatrix}$$

Пользуясь определением скалярного произведения векторов, вычислим косинус угла ∠ABC, образованного сторонами треугольника — векторами BA и BC. Затем с помощью встроенной функции acos (арккосинус) определим величину внутреннего угла. Так как сумма смежных углов равна 180°, внешний угол найдем как разность π – ∠ABC.

$$\varphi := \pi - \operatorname{acos}\left[\frac{(A - B)\cdot(C - B)}{|A - B|\cdot|C - B|}\right] \qquad \varphi \to \frac{2}{3}\cdot\pi$$

Скалярное произведение можно вычислить и символьно

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}\cdot\begin{pmatrix} x - y \\ y \\ x \\ z - 4 \end{pmatrix} \to x\cdot(x - y) + \frac{y^2}{x} + z\cdot(z - 4)$$

3.2.9. Обратная матрица

Матрица A^{-1} называется обратной к матрице A, если $A\cdot A^{-1}=A^{-1}\cdot A=E$, где E — единичная матрица (матрица, у которой элементы главной диагонали равны 1, а все остальные — 0). Матрица имеет обратную только в том случае, если она квадратная и ее определитель не равен 0. Определение обратной матрицы — одна из основных задач матричной алгебры, поскольку в подавляющем большинстве доказательств и выводов этого раздела математики (имеющего огромное практическое значение) обратную

матрицу приходится использовать. С помощью Mathcad можно предельно упростить эту очень трудоемкую и порой весьма сложную задачу.

Оператор нахождения обратной матрицы (Inverse) можно ввести с помощью специальной кнопки панели Matrix (Матричные). Однако можно обойтись и без обращения к рабочей панели: достаточно просто выделить матрицу и возвести ее в степень –1 абсолютно аналогично числам или выражениям.

Находить обратную матрицу можно как для матриц с элементами-числами, так и для матриц, элементы которых определены символьно.

Пример 3.24. Вычисление обратной матрицы

$$\begin{pmatrix} 2 & 3 & 4 \\ 5 & 6 & 7 \\ 8 & 9 & 0 \end{pmatrix}^{-1} = \begin{pmatrix} -2.1 & 1.2 & -0.1 \\ 1.867 & -1.067 & 0.2 \\ -0.1 & 0.2 & -0.1 \end{pmatrix}$$

$$\begin{pmatrix} \sin(\alpha) & -\cos(\alpha) \\ \cos(\alpha) & \sin(\alpha) \end{pmatrix}^{-1} \text{ simplify } \rightarrow \begin{pmatrix} \sin(\alpha) & \cos(\alpha) \\ -\cos(\alpha) & \sin(\alpha) \end{pmatrix}$$

Обратите внимание, оператор simplify корректно упростил элементы обратной матрицы. Однако не следует забывать о том, что символьный процессор Mathcad может правильно преобразовать лишь простейшие тригонометрические выражения.

Для нахождения матрицы, обратной A, Mathcad использует численный метод, в основе которого лежит решение систем линейных уравнений. Действительно, чтобы найти обратную матрицу, необходимо, по сути, решить матричное уравнение вида $A \cdot X = E$. Для вычисления $X = A^{-1}$ Mathcad последовательно ищет корни систем уравнений $A \cdot X_i = E_i$ ($i = 1,2,..N$ – размерность матрицы), где E_i – i-й столбец единичной матрицы или матрицы перестановки (если таковая требуется). Вектор решений X_i каждой системы есть не что иное, как i-й столбец обратной матрицы. Объединяясь, они формируют конечный результат. Обратите внимание, чтобы найти столбцы обратной матрицы, нужно решить одну и ту же систему уравнений с различными векторами правых частей. Для упрощения этой процедуры проводится LU-разложение, поскольку алгоритм, используемый в этом случае, затрачивает на поиск решения меньшее число шагов, чем метод Гаусса. В основе его работы лежат следующие операции.

1. После того как проведено LU-разложение, исходную систему уравнений представляем как $LUX_i = E_i$.
2. Полагая, что $UX_i = Y_i$, находим решения Y_i нижних треугольных систем $LY_i = E_i$ методом прямой подстановки.
3. Методом обратной подстановки ищем интересующие нас решения X_i верхних треугольных систем $UX_i = Y_i$.
4. Объединяем полученные вектора X_i в конечную матрицу A^{-1}.

Помимо описанного выше способа найти обратную матрицу в Mathcad можно, задействовав встроенную функцию geninv(M). Для матриц, определитель которых мало отличен от нуля, алгоритм, использующий LU-разложение, может и не сойтись. С помощью же функции geninv(M) можно получить матрицу, обратную любой почти сингулярной матрице. Правда, точность таких решений оставляет желать лучшего. В остальных случаях использование функции geninv(M) и оператора Inverse панели Matrix абсолютно идентично.

3.2.10. Сумма элементов вектора

В некоторых случаях приходится находить сумму элементов вектора. Для этого в Mathcad существует специальный оператор Vector Sum (Сумма вектора), который расположен на панели Matrix (Матричные). Также его можно ввести сочетанием клавиш Ctrl+4.

Иногда возникает необходимость просуммировать элементы, стоящие на главной диагонали квадратной матрицы. Такая сумма называется следом (trace) матрицы. Для ее вычисления в Mathcad существует специальная функция tr(M), где M — матрица.

Пример 3.25. Сумма элементов вектора и след матрицы

$$\sum\begin{pmatrix}3\\4\\5\end{pmatrix}=12 \qquad \mathrm{tr}\left(\begin{pmatrix}1&3&5\\4&5&6\\5&6&7\end{pmatrix}\right)=13$$

3.2.11. Возведение матрицы в степень и матричные уравнения

Система Mathcad позволяет проводить операции возведения матрицы в степень. Если матрица квадратная, ее можно возвести в степень n, где n — любое целое число (или 0). При этом:

- если n=0, то (по аналогии с традиционной алгеброй) $M^n=E$, где E — единичная матрица;
- если n=1, то $M^1=M$ (матрица, возведенная в первую степень, равна сама себе);
- если n=N, где N — любое натуральное число, равное или большее 2, то степень матрицы определяется как произведение соответствующего числа матриц в первой степени. Подобное произведение возможно, поскольку при перемножении две квадратные матрицы равной размерности дают третью с той же размерностью;
- если n=−1, то M^{-1} — это матрица, обратная данной;
- если n=R, где R — любое целое отрицательное число, меньшее либо равное −2, то аналогично возведению в положительную степень, M^R определяется как произведение нужного числа обратных матриц.

Задается степень матрицы аналогично случаям с числами или выражениями: либо с помощью кнопки Raise to Power (Возвести в степень) рабочей панели Calculator (Калькулятор), либо сочетанием клавиш Shift+6 (предварительно матрицу следует выделить).

Пример 3.26. Возведение матрицы в степень

$$\begin{pmatrix}1&0&0\\0&2&0\\0&0&3\end{pmatrix}^4=\begin{pmatrix}1&0&0\\0&16&0\\0&0&81\end{pmatrix} \qquad \begin{pmatrix}3&4&5\\6&7&8\\5&7&8\end{pmatrix}^{-5}=\begin{pmatrix}-71.222&24.889&17\\11.202&-54.695&49.778\\37.905&30.49&-54.222\end{pmatrix}$$

Большое значение в линейной алгебре и, особенно, теории оптимизации имеют так называемые матричные уравнения. Решить их можно очень просто с помощью блока Given-Find. Подробно особенности использования данного блока описываются в гл. 8.

Пример 3.27. Найти матрицу X, удовлетворяющую условию $X^2-X=M$, где M — некоторая квадратная матрица

Задаем матрицу M:

$$M := \begin{pmatrix} 30 & 36 & 42 \\ 66 & 81 & 96 \\ 102 & 126 & 150 \end{pmatrix}$$

Указываем начальное приближение для используемого Mathcad численного алгоритма:

$$X := 0.001M$$

Указываем вид уравнения в блоке Given-find и присваиваем матрицу с решением переменной R:

Given

$$X^2 - X = M$$

$$R := \text{find}(X)$$

Проверяем решение на верность:

$$R = \begin{pmatrix} -2.106 & -2.626 & -2.146 \\ -4.355 & -4.22 & -6.084 \\ -5.604 & -7.814 & -9.023 \end{pmatrix} \qquad R^2 - R = \begin{pmatrix} 30 & 36 & 42 \\ 66 & 81 & 96 \\ 102 & 126 & 150 \end{pmatrix}$$

Аналогично численному решению простых уравнений, решение матричных уравнений зависит от величины параметра TOL, а также начального приближения. При поиске корней матричных уравнений система использует те же алгоритмы, что и для решения систем нелинейных уравнений. Символьно решать матричные уравнения Mathcad не может.

3.2.12. Векторизация матриц

Самым оригинальным и интересным оператором матричных преобразований является оператор векторизации (Vectorize) панели Matrix (Матричные). Этот оператор позволяет производить некоторые преобразования над каждым элементом матрицы в отдельности. Задачи подобного рода возникают очень часто, и в общем случае для их решения приходится задавать циклы или даже писать программы. Оператор векторизации помогает справиться со многими такими задачами гораздо проще.

Пример 3.28. Произведение элементов двух матриц

Пусть стоит следующая задача: есть две соразмерные матрицы M и N. Следует задать матрицу P, элементы которой образованы произведениями соответствующих элементов матриц M и N.

Определяем исходные матрицы:

$$M := \begin{pmatrix} 2 & 3 & 3 \\ 3 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad N := \begin{pmatrix} 3 & 2 & 4 \\ 6 & 7 & 5 \\ 7 & 8 & 9 \end{pmatrix}$$

Перемножаем их с использованием векторизации:

$$P := \overrightarrow{(M \cdot N)}$$

Выводим результат:

$$P = \begin{pmatrix} 6 & 6 & 12 \\ 18 & 35 & 30 \\ 49 & 64 & 81 \end{pmatrix}$$

Ввести оператор векторизации можно либо с панели Matrix (Матричные), либо сочетанием клавиш Ctrl+←. Перед тем как ввести этот оператор, следует выделить преобразуемое выражение.

Применять оператор векторизации можно в очень разнообразных ситуациях. Следующий пример демонстрирует это.

Пример 3.29. Разнообразие векторизации

С помощью операции векторизации можно возвести одну матрицу в степень, которая представляется в виде другой матрицы. И это совсем не абсурд: просто каждый элемент одной матрицы будет возведен в степень, равную значению соответствующего ему элемента второй матрицы:

$$M := \begin{pmatrix} 1 & 2 & 1 \\ 2 & 3 & 1 \\ 2 & 3 & 3 \end{pmatrix} \qquad N := \begin{pmatrix} 2 & 3 & 4 \\ 2 & 4 & 5 \\ 6 & 3 & 2 \end{pmatrix} \qquad \overrightarrow{M^N} = \begin{pmatrix} 1 & 8 & 1 \\ 4 & 81 & 1 \\ 64 & 27 & 9 \end{pmatrix}$$

Используя векторизацию, можно получить матрицу значений любой функции в точках, равных по значению элементам исходной матрицы. А это может значительно облегчить задачу обработки данных и построения графиков:

$$\overrightarrow{\sin(M)} = \begin{pmatrix} 0.841 & 0.909 & 0.841 \\ 0.909 & 0.141 & 0.841 \\ 0.909 & 0.141 & 0.141 \end{pmatrix}$$

3.3. Использование матричных функций

Работа с матрицами — это наиболее важная и наиболее часто используемая область компьютерной математики. Поэтому совсем не удивительно, что специальных матричных функций в Mathcad очень много — более 50 (значительно больше, чем любых других). Аналогично другим встроенным функциям, для задания матричных функций нужно воспользоваться командой Insert ▸ Function (Вставить функцию) главного меню либо соответствующей ей кнопкой панели Standard. Также матричные функции можно просто набрать с клавиатуры. Первый способ предпочтительнее в том плане, что запомнить синтаксис такого большого количества функций очень сложно, а также тем, что вы всегда сможете прочитать описание нужной вам матричной функции в случае, если нет уверенности в собственных знаниях.

Все матричные функции можно разделить на несколько основных типов: редактирование, создание матриц специального вида, поиск матричных норм и др. Поэтому в этой

главе матричные функции рассматриваются не по отдельности, а функциональными группами.

3.3.1. Задание матриц специального вида

Существует целый ряд матриц так называемого специального вида (верхние и нижние треугольные, единичные, трапециевидные, скалярные, нулевые и некоторые другие). В Mathcad имеются функции, которые позволяют быстро и просто задавать матрицы специального вида.

- identity(N). Эта функция служит для задания единичной матрицы размерности N.

Пример 3.30. Создание единичной матрицы

$$\text{identity}(3) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- diag(V). Функция создает диагональную матрицу, элементы главной диагонали которой равны соответствующим элементам вектора V.

3.3.2. Функции определения размерности матрицы

- Функции rows(M) и cols(M). Служат для определения количества строк и соответственно столбцов некоторой матрицы M.

Пример 3.31. Определение размерности матрицы

$$M := \begin{pmatrix} 2 & 3 & 4 \\ 6 & 7 & 6 \end{pmatrix}$$

$$\text{rows}(M) = 2 \qquad \text{cols}(M) = 3$$

- length(V). Функция определяет количество элементов вектора.
- last(V). Функция служит для определения индекса последнего элемента некоторого вектора V. По умолчанию last(V)=length(V)–1. Эти две функции очень важны при решении задач программными методами, поэтому мы с ними еще не раз встретимся.

Пример 3.32. Определение размерности вектора

$$V := \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix} \qquad M := \begin{pmatrix} 2 & 3 \\ 4 & 5 \\ 3 & 5 \end{pmatrix}$$

$$\text{length}(V) = 3 \qquad \text{last}\left(M^{\langle 1 \rangle}\right) = 2$$

3.3.3. Функции сортировки матриц

При решении очень многих задач возникает необходимость отсортировать полученные векторы или матрицы исходя из величины их элементов или по какому-то дру-

гому принципу. При решении большинства сложных задач с помощью программирования, как правило, приходится использовать функции сортировки. Поэтому их значение очень велико и функции эти нужно постараться запомнить.

- sort(V). Сортировка элементов вектора по возрастанию. Одна из наиболее важных и часто используемых на практике матричных функций. Может быть использована и в том случае, если часть элементов вектора определена комплексными числами. При этом сортировка проводится исходя из действительной части числа, а мнимая часть просто игнорируется.

Пример 3.33. Сортировка элементов вектора

$$V := \begin{pmatrix} 1 \\ -3 \\ 5 - 8\cdot i \end{pmatrix} \qquad \text{sort}(V) = \begin{pmatrix} -3 \\ 1 \\ 5 - 8i \end{pmatrix}$$

- csort(M,i). Функция выполняет сортировку по возрастанию элементов i-го столбца некоторой матрицы M.

Пример 3.34. Сортировка выбранного столбца квадратной матрицы

$$M := \begin{pmatrix} 4 & 5 & 1 \\ 1 & 0 & 3 \\ 2 & -41 & 2 \end{pmatrix} \qquad \text{csort}(M, 1) = \begin{pmatrix} 2 & -41 & 2 \\ 1 & 0 & 3 \\ 4 & 5 & 1 \end{pmatrix}$$

Также очень просто можно отсортировать все столбцы матриц. Для этого следует задать цикл с помощью ранжированных переменных. Обратите внимание на использование функции last(V) в этом примере.

Пример 3.35. Сортировка столбцов матрицы

$$M := \begin{pmatrix} 4 & 5 & 1 \\ 1 & 0 & 3 \\ 2 & -41 & 2 \end{pmatrix} \qquad i := 0..\,\text{last}\left(M^{\langle 0 \rangle}\right)$$

$$M1^{\langle i \rangle} := \text{sort}\left(M^{\langle i \rangle}\right) \qquad M1 = \begin{pmatrix} 1 & -41 & 1 \\ 2 & 0 & 2 \\ 4 & 5 & 3 \end{pmatrix}$$

- rsort(M,i). Функция сортирует в порядке возрастания элементы i-й строки матрицы M за счет перестановки соответствующих столбцов. В своем роде абсолютно идентична функции csort(M,i).

Пример 3.36. Сортировка выбранной строки матрицы

$$M := \begin{pmatrix} 4 & 5 & 1 \\ 1 & 0 & 3 \\ 2 & -41 & 2 \end{pmatrix} \qquad \text{rsort}(M, 0) = \begin{pmatrix} 1 & 4 & 5 \\ 3 & 1 & 0 \\ 2 & 2 & -41 \end{pmatrix}$$

Если необходимо отсортировать все строки матрицы, то проще всего действовать исходя из следующего алгоритма.

Пример 3.37. Сортировка строк матрицы

Транспонируем исходную матрицу:

$$M := \begin{pmatrix} 4 & 5 & 1 \\ 1 & 0 & 3 \\ 2 & -41 & 2 \end{pmatrix} \qquad M := M^T$$

Задаем с помощью ранжированной переменой цикл. Задействовав функцию sort(V), поочередно сортируем все столбцы транспонированной матрицы:

$$i := 0..\,last\left(M^{\langle 0 \rangle}\right) \qquad M1^{\langle i \rangle} := sort\left(M^{\langle i \rangle}\right)$$

Транспонируем полученную матрицу (при этом отсортированные столбцы становятся строками) и получаем необходимый результат:

$$M1 := M1^T \qquad M1 = \begin{pmatrix} 1 & 4 & 5 \\ 0 & 1 & 3 \\ -41 & 2 & 2 \end{pmatrix}$$

Если же попытаться отсортировать матрицу, задав цикл непосредственно для функции rsort(M,i), то в результате получится вложенный массив, содержащий в качестве элементов матрицы, у которых отсортировано только по одной строке. Конечно, проблему эту обойти можно — но решение при этом очень усложнится. Так что в таких случаях лучше создавать алгоритмы, подобные вышеописанному.

- reverse(V). Эта функция переставляет элементы вектора в обратном порядке. Используется, как правило, в том случае, если элементы отсортированного вектора или матрицы должны располагаться по принципу «чем ниже — тем меньше». Использование же reverse(V) в данном случае необходимо, поскольку функции сортировки расставляют их в противоположном порядке.

В основе работы функций sort(V), csort(M,i) и rsort(M,i) лежит алгоритм так называемой пирамидальной сортировки. Суть его заключается в представлении массива в виде бинарного сортирующего дерева — пирамиды (рис. 3.6).

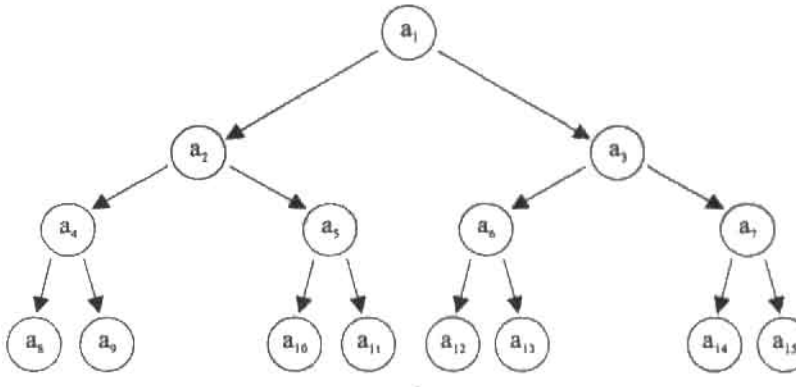


Рис. 3.6. Бинарное дерево последовательности, состоящей из 15 элементов

Любое бинарное дерево содержит только один корневой узел, не имеющий предшественников. Другие же узлы имеют одного предшественника и одного или двух преемников. Бинарное дерево будет являться пирамидой в том случае, если выполняется следующее условие: $a_i \geq a_{2i}$ и $a_i \geq a_{2i+1}$. Другими словами, каждый элемент пирамиды должен быть меньше своего предшественника, либо равняться ему. Тогда максимальный элемент массива будет помещен в корневой узел. Любой же фрагмент «хвостовой» части пирамиды также является пирамидой.

Алгоритм пирамидальной сортировки требует порядка $N \cdot \log_2 N$ операций (где N — размер массива) и включает в себя реализацию двух этапов: построение бинарной пирамиды и непосредственно сортировка ее элементов. Обсудим детально каждый их них.

Начнем построение пирамиды с элементов $a_{N/2}...a_N$, поскольку часть массива при $i \geq N/2$ удовлетворяет свойству пирамиды, так как его элементы не имеют преемников. Будем наращивать пирамиду, добавляя на каждом шаге по одному элементу, стоящему перед уже готовой частью. Допустим, что построение пирамиды началось с элемента a_8 (см. рис. 3.6). В этом случае нам необходимо рассмотреть предшественника — элемент a_4 — и двух его преемников — элементы a_8 и a_9, из которых выбираем наибольший. Если этот преемник (предположим, a_8) оказывается больше своего предшественника a_4, то их меняют местами («просеивают» a_4 к основанию пирамиды). Таким образом, элемент a_8 попадает на предыдущий уровень. Аналогичные операции проводятся с оставшимися элементами массива $a_{N/2}...a_N$. Далее переходим к следующему шагу — повторяем все вышеописанные действия с элементами на уровне, находящемся перед рассмотренным, а «опустившиеся» элементы «просеиваем» по веточке бинарного дерева к основанию до тех пор, пока весь массив не превратится в пирамиду (то есть пока не будет соблюдаться условие $a_i \geq a_{2i}$ и $a_i \geq a_{2i+1}$).

После того как пирамида построена, переходим к следующему этапу — непосредственной сортировке ее элементов. Как вы помните, первый элемент, находящийся в корневом узле, — максимальный. Меняем его местами с последним элементом пирамиды $a_1...a_N$ и больше не возвращаемся к нему. Затем «просеиваем» новый верхний элемент и получаем пирамиду $a_1...a_{N-1}$, с которой повторяем ту же операцию до тех пор, пока размер массива не уменьшится до одного элемента. Таким образом, на каждом шаге в конце массива оказывается максимальный элемент текущей пирамиды. В результате формируется упорядоченная последовательность элементов по возрастанию.

3.3.4. Функции слияния и разбиения матриц

Выделение части матрицы

Для выбора из матрицы элемента с известными индексами в Mathcad существует специальная функция `hlookup(i,M,j)`, где `i` — номер строки, `j` — номер столбца, `M` — некоторая матрица.

При решении многих задач возникает необходимость выделить в отдельные пары соответствующие элементы двух матриц. Наиболее быстро и просто сделать это можно с помощью специальной функции `lookup(r,M,N)`, где `r` — известный скаляр, `M` и `N` — соразмерные матрицы. Функция эта выводит значение того элемента матрицы `N`, который занимает в ней такое же положение, как скаляр `r` в матрице `M`. При этом, если несколько элементов матрицы `M` равны `r`, то в качестве ответа выдается вектор с соответствующими всем им значениями элементов матрицы `N`.

Пример 3.38. Выделение элементов из матриц

$$M := \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 5 \\ 5 & 6 & 7 \end{pmatrix} \qquad N := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

$$\text{lookup}(2, M, N) = \begin{pmatrix} 2 \\ 5 \end{pmatrix} \qquad \text{hlookup}(3, M, 1) = (5)$$

Очень часто на практике возникает задача, обратная рассмотренной: по известному элементу определить его положение в матрице. Так как иногда приходится работать с очень большими матрицами, визуальный поиск нельзя считать универсальным. Поэтому для выполнения такой задачи в Mathcad существует специальная функция match(z,M), где z — скаляр, положение которого определяется, M — матрица, в которой ведется поиск.

Пример 3.39. Определение положения элемента матрицы

$$M := \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \qquad \text{match}(3, M) = \left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} \right]$$

Для выделения из матрицы произвольной подматрицы в Mathcad существует специальная функция submatrix(M,r1,r2,c1,c2), где M — некоторая матрица, r1 и r2 — верхняя и нижняя граничные строки, c1 и c2 — соответственно граничные столбцы извлекаемой подматрицы. С помощью этой функции, правильно определив границы, можно выделить отдельные строки или столбцы матрицы.

Пример 3.40. Использование функции submatrix

$$M := \begin{pmatrix} 2 & 3 & 4 \\ 6 & 7 & 8 \\ 6 & 7 & 4 \end{pmatrix}$$

$$M1 := \text{submatrix}(M, 0, 1, 0, 1) \qquad M1 = \begin{pmatrix} 2 & 3 \\ 6 & 7 \end{pmatrix}$$

Выделение отдельного столбца или строки:

$$\text{Col} := \text{submatrix}(M, 0, 2, 1, 1) \qquad \text{Row} := \text{submatrix}(M, 0, 0, 0, 2)$$

$$\text{Col} = \begin{pmatrix} 3 \\ 7 \\ 7 \end{pmatrix} \qquad \text{Row} = (2 \quad 3 \quad 4)$$

Функции слияния матриц

Иногда при решении задач возникают проблемы, требующие слияния нескольких векторов или матриц. На практике функции слияния матриц используются очень часто, поэтому их нужно постараться запомнить.

- augment(A,B,...), где A, B — некоторые матрицы. Функция эта служит для слияния матриц слева направо. Естественно, соединяемые матрицы должны иметь одинаковое количество строк.
- stack(A,B,...). Функция отвечает за слияние матриц сверху вниз. В случае ее использования матрицы должны иметь равное число столбцов.

Пример 3.41. Слияние матриц

$$A := \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \qquad N := \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix}$$

$$\text{augment}(A, N) = \begin{pmatrix} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \end{pmatrix} \qquad \text{stack}(A, N) = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 2 & 2 \\ 2 & 2 \end{pmatrix}$$

3.3.5. Функции вычисления матричных норм

Одной из важнейших характеристик матрицы является ее норма. Понятие это очень широко используется во многих задачах линейной алгебры (например, при нахождении чисел обусловленности матриц или при оценке сходимости итерационных методов при решении больших систем линейных алгебраических уравнений). Нестрого говоря, прежде всего, норма матрицы отражает средний порядок величины матричных элементов. Так, например, норма вектора — это не что иное, как его длина.

В линейной алгебре существует довольно значительное количество различных алгоритмов вычисления нормы. Если векторное пространство ограничено, то все они дают приблизительно одинаковый результат — как правило, различие не выходит за пределы одного порядка. А так как матричные нормы есть величины оценочные, то нет разницы, которую из них использовать.

В Mathcad существуют четыре функции, которые отвечают за вычисление матричных норм.

- norm1(M) — норма вычисляется в пространстве *L1* (о том, что это такое, можно прочитать в любой математической энциклопедии или учебнике по линейной алгебре). Алгоритм ее вычисления следующий: складываются все элементы в столбцах (вернее, модули значений элементов), и наибольшая сумма определяется как норма данной матрицы.
- norm2(M) — матричная норма в пространстве *L2*.
- normi(M) — так называемая ∞-норма, или норма на бесконечности (infinity norm). Вычисляется так же, как norm1, но суммируются элементы не столбцов, а строк.
- norme(M). Норма в евклидовом пространстве. По аналогии с векторами, вычисляется как квадратный корень из суммы квадратов всех элементов матрицы.

Так как нормы определяются довольно простыми выражениями, на практике можно специальные функции и не использовать. Поэтому в следующем примере нормы вычисляются обоими способами (обратите внимание на то, что результаты получаются очень близкими вне зависимости от выбранного алгоритма).

Пример 3.42. Вычисление норм матриц

$$M := \begin{pmatrix} 1 & 2 & 2 \\ 2 & 3 & 8 \\ 9 & 6 & 7 \end{pmatrix}$$

norm1(M) = 17 normi(M) = 22

norm2(M) = 15.224 norme(M) = 15.875

$$\max\left(\left(\begin{array}{c} \sum_{i=0}^{2} M_{i,0} \\ \sum_{i=0}^{2} M_{i,1} \\ \sum_{i=0}^{2} M_{i,2} \end{array}\right)\right) = 17 \qquad \max\left(\left(\begin{array}{c} \sum_{j=0}^{2} M_{0,j} \\ \sum_{j=0}^{2} M_{1,j} \\ \sum_{j=0}^{2} M_{2,j} \end{array}\right)\right) = 22$$

$$\sqrt{\sum_{i=0}^{2}\sum_{j=0}^{2}\left(M_{i,j}\right)^2} = 15.875$$

3.3.6. Вычисление ранга матрицы

Ранг матрицы равен порядку наибольшего отличного от нуля минора этой матрицы. Величина эта служит прежде всего для характеристики матрицы системы уравнений, а также как параметр, определяющий, базисом пространства какой размерности может быть данное множество векторов.

Для вычисления ранга матрицы в Mathcad существует специальная функция rank(M).

В следующем примере находится величина ранга для трех матриц. У первой (M) все три строки линейно независимы, у второй (M1) имеются две линейно зависимые строки, у третьей (M2) все строки линейно зависимы.

Пример 3.43. Вычисления ранга матрицы

$$M := \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 6 \\ 4 & 5 & 9 \end{pmatrix} \qquad M1 := \begin{pmatrix} 1 & 2 & 3 \\ 5 & 10 & 15 \\ 2 & 3 & 5 \end{pmatrix} \qquad M2 := \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 4 & 8 & 12 \end{pmatrix}$$

rank(M) = 3 rank(M1) = 2 rank(M2) = 1

3.3.7. Функции вычисления собственных значений и собственных векторов

Очень важной с практической точки зрения является задача о собственных векторах и собственных значениях матрицы, описывающей некоторую однородную систему. Это направление линейной алгебры находит широкое применение в экономике, программировании и особенно в инженерном деле (например, без него немыслима теория повреждений при максимальных нагрузках или теория о деформациях). Поэтому, естественно, в такой совершенной математической системе, как Mathcad, оно не могло не найти отражения.

Собственные векторы X_1, X_2,... и соответствующие им собственные значения λ_1, λ_2,... матрицы M являются решениями матричного уравнения вида

$$M \cdot X = \lambda \cdot X \qquad (1)$$

В принципе, для небольших матриц решить уравнение такого вида совсем несложно, однако, если, к примеру, вам требуется найти корни однородной системы линейных дифференциальных уравнений, образованной 10 уравнениями, тут без помощи компьютера обойтись трудно. В системе Mathcad существует ряд функций, позволяющих весьма эффективно справляться с такой работой.

- eigenvals(M). Функция возвращает вектор, содержащий собственные значения матрицы M. Собственные значения расположены в векторе в соответствии с их величиной.
- eigenvecs(M). Функция возвращает матрицу, столбцы которой являются собственными векторами для матрицы M. Между функциями eigenvals(M) и eigenvecs(M) существует следующая связь: n-му элементу вектора собственных значений соответствует n-й столбец матрицы собственных векторов.
- eigenvec(M,λ). Функция вычисляет собственный вектор матрицы M, соответствующий определенному собственному значению λ.

В следующем примере рассматривается вычисление собственных значений и собственных векторов для матрицы размерности n=3. Также проводится проверка правильности вычислений. Для этого, исходя из общего вида матричного уравнения (1), перемножаются собственный вектор и собственное значение матрицы M. Если расчеты были сделаны правильно, то полученный вектор должен равняться вектору, образованному перемножением самой матрицы M на ее собственный вектор.

Пример 3.44. Вычисление собственных значений и собственных векторов матрицы

$$M := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad \text{eigenvals}(M) = \begin{pmatrix} 16.117 \\ -1.117 \\ 0 \end{pmatrix}$$

$$\text{eigenvecs}(M) = \begin{pmatrix} 0.232 & 0.786 & -0.408 \\ 0.525 & 0.087 & 0.816 \\ 0.819 & -0.612 & -0.408 \end{pmatrix} \qquad \text{eigenvec}(M, 0) = \begin{pmatrix} 0.408 \\ -0.816 \\ 0.408 \end{pmatrix}$$

Проверка результатов вычислений:

$$M \cdot \text{eigenvecs}(M)^{\langle 0 \rangle} = \begin{pmatrix} 3.739 \\ 8.467 \\ 13.194 \end{pmatrix} \qquad \text{eigenvecs}(M)^{\langle 0 \rangle} \cdot \text{eigenvals}(M)_0 = \begin{pmatrix} 3.739 \\ 8.467 \\ 13.194 \end{pmatrix}$$

Также в линейной алгебре существует проблема поиска собственных значений и собственных векторов для случая более общей задачи, определяемой уравнением $M \cdot X = \lambda \cdot N \cdot X$, где N — некоторая квадратная матрица. Технически решение такой задачи находится намного труднее, чем предыдущей, так что возможности Mathcad тут могут быть просто незаменимы.

Для решения задачи поиска обобщенных собственных значений существуют две специальные функции.

- genvals(M,N). Функция вычисляет вектор собственных значений матрицы M (аналогично функции eigenvals(M)).
- genvecs(M,N). Функция, аналогичная eigenvecs(M). Возвращает матрицу, содержащую в качестве столбцов собственные векторы матрицы M при некоторой заданной матрице N.

В следующем примере рассматривается использование функций genvals(M,N) и genvecs(M,N), а также проводится проверка на верность вычислений.

Пример 3.45. Решение обобщенной задачи на собственные значения

$$M := \begin{pmatrix} 3 & 4 & 5 \\ 5 & 0 & 7 \\ 5 & 6 & 3 \end{pmatrix} \qquad N := \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

$$\text{genvecs}(M, N) = \begin{pmatrix} 0.803 & 0.831 \\ -0.466 & 0.534 \\ -0.371 & -0.158 \end{pmatrix} \qquad \text{genvals}(M, N) = \begin{pmatrix} -3.038 \\ 5.705 \end{pmatrix}$$

Проверка на верность расчетов:

$$M \cdot \text{genvecs}(M, N)^{\langle 0 \rangle} = \begin{pmatrix} -1.312 \\ 1.416 \\ 0.104 \end{pmatrix} \qquad N \cdot \text{genvals}(M, N)_0 \cdot \text{genvecs}(M, N)^{\langle 0 \rangle} = \begin{pmatrix} -1.312 \\ 1.416 \\ 0.104 \end{pmatrix}$$

3.3.8. Функции матричных разложений

Одной из главных задач линейной алгебры является задача о представлении матрицы в виде произведения двух или более матриц специального вида. Во многих случаях, если такое представление (разложение) удается, решение той или иной проблемы значительно упрощается. Матричные разложения используются прежде всего при решении систем линейных алгебраических уравнений, а также в задачах, связанных с поиском собственных значений матриц.

LU-разложение

Суть этого разложения сводится к тому, что любую (не сингулярную) квадратную матрицу можно представить в виде произведения верхней и нижней треугольных матриц (R·M=L·U, где M — разлагаемая матрица, R — матрица перестановки (если же таковая не требуется, просто единичная матрица), L и U — нижняя и верхняя треугольные матрицы). LU-разложение широко используется при построении альтернативного методу Гаусса алгоритма решения систем линейных алгебраических уравнений. Достоинство такого способа проявляется в тех случаях, когда требуется решить одну и ту же систему с различными векторами правых частей (а задача такая может возникнуть при решении систем с параметрами). В этом случае число шагов алгоритма значительно сокращается. При решении же единственной системы уравнений нет разницы, какой из методов использовать.

Разложить матрицу на произведение нижней и верхней треугольных матриц в системе Mathcad можно с помощью специальной функции lu(M).

Особенностью этой функции является то, что результат выдается в виде одной общей матрицы, полученной слиянием матрицы перестановки R, нижней треугольной матрицы L и верхней треугольной матрицы U. Чтобы результат можно было использовать в дальнейшей работе, придется воспользоваться уже рассмотренной выше функцией submatrix. Как это сделать, демонстрируется в примере 3.46. Также в нем осуществляется проверка правильности разложения.

Пример 3.46. LU-разложение

$$M := \begin{pmatrix} 3 & 4 & 5 \\ 5 & 0 & 7 \\ 5 & 6 & 3 \end{pmatrix} \qquad N := lu(M)$$

$$N = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 5 & 6 & 3 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & -6 & 4 \\ 1 & 0 & 0 & 0.6 & -0.067 & 1 & 0 & 0 & 3.467 \end{pmatrix}$$

$$R := submatrix(N, 0, rows(M) - 1, 0, cols(M) - 1) \qquad R = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$L := submatrix(N, 0, rows(M) - 1, cols(M), 2cols(M) - 1) \qquad L = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0.6 & -0.067 & 1 \end{pmatrix}$$

$$U := submatrix(N, 0, rows(M) - 1, 2cols(M), 3cols(M) - 1) \qquad U = \begin{pmatrix} 5 & 6 & 3 \\ 0 & -6 & 4 \\ 0 & 0 & 3.467 \end{pmatrix}$$

Проверка на верность разложения:

$$R \cdot M - L \cdot U = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

То, как в Mathcad осуществляется LU-разложение, очень подробно описывается в гл. 8 (в разделе, посвященном решению систем линейных уравнений). Там же имеется соответствующий алгоритм на языке программирования Mathcad.

Разложение Холецкого

Разложением Холецкого называется представление матрицы в виде произведения некоторой треугольной матрицы и матрицы, полученной транспонированием последней ($M=L \cdot L^T$). Обязательным условием возможности разложения Холецкого является то, что разлагаемая матрица должна быть симметричной (то есть $M=M^T$) и положительно определенной (то есть ее определитель должен быть положительным числом).

Для реализации разложения Холецкого в Mathcad существует специальная встроенная функция cholesky(M), результатом работы которой является матрица L.

QR-разложение

QR-разложением матрицы M называется представление ее в виде произведения ортогональной (то есть $Q^T=Q^{-1}$) и верхней треугольной матриц. Разложение имеет практическое значение в связи с использованием его при решении задач на собственные значения матриц.

Для выполнения QR-разложения в Mathcad существует специальная встроенная функция qr(M). Аналогично функции LU-разложения, qr(M) представляет результат в виде единой матрицы. Поэтому, как и при выполнении LU-разложения, придется использовать функцию submatrix.

Пример 3.47. QR-разложение

$$M := \begin{pmatrix} 2 & 3 & 5 \\ 4 & 5 & 6 \\ 7 & 8 & 6 \end{pmatrix} \qquad N := qr(M)$$

$$N = \begin{pmatrix} 0.241 & -0.84 & 0.487 & 8.307 & 9.872 & 9.149 \\ 0.482 & -0.332 & -0.811 & 0 & -0.742 & -3.613 \\ 0.843 & 0.43 & 0.324 & 0 & 0 & -0.487 \end{pmatrix}$$

$$Q := submatrix(N, 0, rows(M) - 1, 0, cols(M) - 1)$$

$$Q = \begin{pmatrix} 0.241 & -0.84 & 0.487 \\ 0.482 & -0.332 & -0.811 \\ 0.843 & 0.43 & 0.324 \end{pmatrix} \qquad Q \cdot Q^T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R := \text{submatrix}(N, 0, \text{rows}(M) - 1, \text{cols}(M), 2\,\text{cols}(M) - 1) \quad R = \begin{pmatrix} 8.307 & 9.872 & 9.149 \\ 0 & -0.742 & -3.613 \\ 0 & 0 & -0.487 \end{pmatrix}$$

Проверка разложения:

$$M - Q \cdot R = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Сингулярное разложение

Сингулярным разложением матрицы M размера $n \times m$ ($n \geq m$) называется разложение вида $M = U \cdot s \cdot V^T$, где U и V — диагональные матрицы размерности соответственно $n \times n$ и $m \times m$, s — диагональная матрица с сингулярными числами матрицы M на диагонали.

Для решения задачи сингулярного разложения в Mathcad существуют следующие встроенные функции.

- svds(M). Определяет вектор, содержащий сингулярные числа матрицы M.
- svd(M). Функция сингулярного разложения. Результат выдает в виде единой матрицы.

Сингулярное разложение полезно в связи с тем, что оно зачастую дает возможность решать практически вырожденные системы линейных уравнений, с которыми не справляются стандартные методы.

3.3.9. Функции max и min

Одной из самых распространенных матричных задач является задача выбора наибольшего элемента в массиве. С подобной проблемой мы уже сталкивались, когда при определении нормы матрицы требовалось выбрать наибольшую сумму элементов строк или столбцов. Также к задаче такого рода сводится поиск экстремума функции (локального или глобального) матричным методом. Естественно, Mathcad не был бы самой совершенной математической программой, если бы не позволял делать этого предельно быстро и просто. Для выбора наибольшего или наименьшего элемента существуют две специальные функции:

- max(M) — выдает значение наибольшего элемента матрицы;
- min(M) — функция выбора наименьшего элемента.

Если элементы представлены комплексными числами, то при использовании функций min и max результат выдается исходя из величины их действительной части.

Пример 3.48. Использование функций min и max

$$M := \begin{pmatrix} 1 & 2 & 5 \\ 4 & 5 & 6 \\ 6 & 7 & 8 \end{pmatrix} \qquad \max(M) = 8 \qquad \min(M) = 1$$

Глава 4. Программирование

Большинство людей, никогда не сталкивавшихся в своей практике с решением задач с помощью программирования, считают эту область человеческого знания сложной, малопонятной и доступной лишь специалистам. Это так и не так в одно и то же время. Конечно, посидеть вечерок над учебником для того, чтобы быть способным написать программу управления сложной автоматической системой — это, конечно, нереально. Но за относительно небольшой отрезок времени освоить тот минимум, который необходим для успешного решения большинства математических задач — это, в принципе, возможно, причем в случае любого современного языка программирования высокого уровня.

Позиция компании Mathsoft относительно того, как должна быть построена математическая система, была и остается следующей: Mathcad должен быть настолько прост, чтобы быть доступным пониманию всех, кому приходится сталкиваться с расчетами, будь то ученый, инженер или простой студент. Поэтому в первых версиях системы Mathcad языка программирования не было вовсе: авторы посчитали, что введение такого сложного (в массовом понимании) элемента, как язык программирования, не впишется в общую концепцию пакета. Однако на практике без элементов программирования обойтись довольно сложно (это потребовало бы введения огромного количества встроенных функций, предусматривающих все возможные и невозможные запросы пользователя). Поэтому создатели Mathcad, желая сохранить простоту своего детища, пошли на довольно нестандартный шаг — они создали систему программирования без программирования. Звучит довольно странно, не правда ли? Попробуем объяснить, что означает этот термин. Все дело в том, что любая программа строится на своего рода трех китах алгоритмизации: следовании, цикле и условии. Последовательное выполнение команд (следование) обеспечивалось логической активностью рабочих листов Mathcad. Вместо оператора цикла for в Mathcad было предложено использовать ранжированные переменные, цикл while имитировался посредством функции until (последняя со времен Mathcad 2000 считалась устаревшей, но пользователи продолжали ее активно применять, поэтому в Mathcad 12 разработчики «воскресили» ее), для введения условия существовала встроенная функция if. Куда больших изощрений требовало задание рекуррентных соотношений, но и это было возможно. Построенные по описанным принципам алгоритмы получались резко отличными по своей форме от выработанных норм и традиций, поэтому программами их можно было назвать с очень зна-

чительными оговорками. При этом желаемого упрощения достигнуто не было, скорее наоборот — алгоритмы в Mathcad получались запутанными и причудливо-сложными. Поэтому можно вполне смело утверждать, что эксперимент Mathsoft, связанный с попыткой создать качественную математическую систему, которая бы могла эффективно работать вообще без использования программирования, оказался неудачным.

В следующем примере вы можете сравнить два способа программного решения задачи в Mathcad. Внимательно изучив его, вы наверняка поймете преимущества использования языка программирования.

Пример 4.1. Решение квадратного уравнения альтернативными программными методами

$$f(x) := x^2 - 5x + 4$$

$$A := f(x)\ \text{coeffs}, x \rightarrow \begin{pmatrix} 4 \\ -5 \\ 1 \end{pmatrix}$$

$$A := \text{reverse}(A)$$

$$D := (A_1)^2 - 4 \cdot A_0 \cdot A_2$$

$$\text{Solution} := \text{if}\left(D \geq 0, \begin{bmatrix} \dfrac{(-A)_1 - \sqrt{D}}{2 \cdot A_0} \\ \dfrac{(-A)_1 + \sqrt{D}}{2 \cdot A_0} \end{bmatrix}, \text{"No real roots"}\right)$$

$$\text{Solution} = \begin{pmatrix} 1 \\ 4 \end{pmatrix}$$

$$\text{SolutionProg}(a, b, c) := \left| \begin{array}{l} D \leftarrow b^2 - 4 \cdot a \cdot c \\ \text{if } D \geq 0 \\ \quad \left| \begin{array}{l} x_0 \leftarrow \dfrac{-b - \sqrt{D}}{2a} \\ x_1 \leftarrow \dfrac{-b + \sqrt{D}}{2a} \\ \text{return } x \end{array} \right. \\ \text{"No real roots"} \end{array} \right.$$

$$\text{SolutionProg}(1, 1, 1) = \text{"No real roots"}$$

$$\text{SolutionProg}(1, -5, 4) = \begin{pmatrix} 1 \\ 4 \end{pmatrix}$$

Осознав, что отсутствие возможности программирования сильно ослабляет позиции пакета Mathcad, компания Mathsoft дополнила его профессиональную версию соответствующим элементом. И, следует признать, сделала это очень качественно.

Язык программирования Mathcad содержит все элементы языка высокого уровня, необходимые для математических расчетов. Будучи дополненным сотнями встроенных функций и операторов системы, возможностями численного и символьного расчета различных величин, он по эффективности не уступает профессиональным системам программирования. Кроме того, у него есть одно очень крупное преимущество: язык программирования Mathcad предельно прост (а по изящности и наглядности в оформлении алгоритмов вообще не имеет аналогов). Конечно, можно за несколько вечеров освоить все, что нужно для решения математических задач, например в Delphi, но для этого потребуется, чтобы у вас были хотя бы начальные знания об основных ходах и приемах в программировании. Для системы программирования Mathcad вам не понадобится ровным счетом никаких познаний в этой области — будет достаточно одной лишь интуиции. Возможности же, которые вы получите, освоив эту систему, ограничены лишь конечной производительностью компьютера!

В данной главе мы рассмотрим операторы и важнейшие приемы программирования в Mathcad. Практику программирования мы отработаем при решении задач в остальных

разделах (так, мы самостоятельно реализуем несколько ключевых численных методов).

4.1. Создание программ

Первым делом, приступая к разговору о создании алгоритмов в среде Mathcad, откроем специальную панель, содержащую все операторы и элементы языка программирования. Панель эта называется Programming (Программирование) и относится она к панели Math (Математические) (кнопка в виде блок-схемы).

Открыв указанную панель (рис. 4.1), вы обнаружите, что язык программирования Mathcad имеет предельно малое количество операторов — всего 8. Впрочем, это никоим образом не сказывается на эффективности создаваемых алгоритмов.

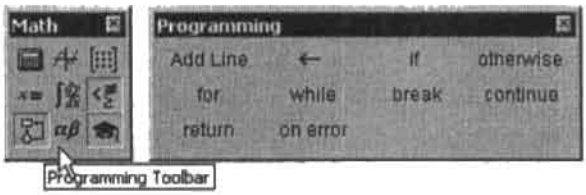
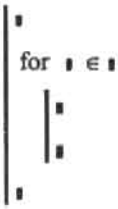


Рис. 4.1. Панель Programming (Программирование)

Чтобы написать программу, прежде всего для нее должен быть создан специальный обособленный от остального документа блок. Выглядит он как черная вертикальная линия с маркерами, в которые заносятся те или иные выражения алгоритма. Чтобы построить единичный элемент программного блока, нажмите кнопку команды Add Line (Добавить линию) панели Programming (Программирование). Или, лучше, воспользуйтесь «горячей» клавишей «]». При этом в области курсора появится следующий объект:

Обычно программа содержит больше чем две строки, поэтому лучше сразу задать блок из 5–6 маркеров. Сделать это можно, последовательно нажав нужное количество раз соответствующую кнопку панели Programming или «горячую» клавишу.

Программный блок можно создать и внутри уже заданного блока. Для этого используйте один из стандартных способов, поставив курсор в маркер любого из операторов программирования:



Созданный таким образом блок выглядит как параллельная главному блоку линия. Выражения, внесенные в него, будут обособлены от остальной программы, и выполне-

ние соответствующих им действий будет связано только с оператором, к которому относится внутренний блок.

Иногда при написании программы бывает нужным добавить строку к уже созданному блоку. Чтобы это сделать, поставьте курсор в ту строку блока, выше или ниже которой должна быть введена строка, и нажмите клавишу Пробел. При этом строка будет выделена и можно будет произвести добавление одним из стандартных способов. Положение вставляемого маркера определяется положением вертикальной черты курсора. Если она находится слева от выделенного выражения, то маркер будет добавлен выше выделенной строки, если справа — то ниже. Чтобы развернуть курсор в нужную сторону, нажмите клавишу Insert. Чтобы добавить строку к целому блоку, его следует выделить, дважды нажав клавишу Пробел. В том случае, если программа содержит блоки различных уровней, то для добавления строки, например, к первому блоку, нажмите клавишу Пробел несколько раз: при каждом нажатии будут выделяться блоки более низкого уровня.

Для присвоения значений переменным и функциям в программах Mathcad используется специальный оператор: «←» (Local Definition — Локальное присваивание), расположенный на панели Programming (Программирование) (также вводится сочетанием клавиш Shift+«[»). Использовать оператор обычного присваивания «:=» в программах нельзя. Однако вид уже введенного оператора присваивания может быть сменен с «←» на «=». Для этого нужно щелкнуть на соответствующей строке программы правой кнопкой мыши и в появившемся контекстном меню открыть меню View Definition As (Видеть присваивание как). В данном меню нужно выбрать пункт Equal (Равенство) (по умолчанию выбран пункт Left Arrow (Левая стрелка)).

Присваивание значений в программах имеет ряд особенностей. Важнейшим из них является то, что присвоение величин используемым алгоритмом функциям и переменным может быть произведено как в самой программе, так и выше нее. Данные два подхода весьма существенно разнятся. Если значение переменной или функции присваивается в программе посредством оператора «←», то такая переменная или функция будет являться локальной. То есть она будет видимой только в рамках программы. Как-то повлиять на объекты вне программы она не сможет (равно как извне к ней нельзя будет получить доступ). Если переменная или функция задается выше программы с помощью оператора «:=», то она будет обладать глобальной видимостью. То есть такая переменная или функция будет доступна любому нижележащему объекту, в том числе и коду программ. Однако программа может только прочитать значение глобальной переменной или вызвать глобальную функцию. Как-то изменить значение глобальной переменной или функции программа не может. Это очень важно учитывать при написании алгоритмов. Если программа должна осуществлять какую-то модификацию объекта (например, возводить все элементы массива в квадрат), то результат своей работы она должна возвращать (см. пример 4.2).

Пример 4.2. Использование значения внешней переменной в программе

Массив, элементы которого нужно возвести в квадрат:

$$M := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Программа, перебирающая все элементы M и заменяющая первоначальные значения элементов их квадратами. Полученный результат программа возвращает.

$$\left|\begin{array}{l} \text{for } i \in 0..2 \\ \quad \text{for } j \in 0..2 \\ \quad\quad M_{i,j} \leftarrow \left(M_{i,j}\right)^2 \\ M \end{array}\right. = \begin{pmatrix} 1 & 4 & 9 \\ 16 & 25 & 36 \\ 49 & 64 & 81 \end{pmatrix}$$

В результате выполнения программы массив M изменен не был. Это означает, что при обращении из программы к глобальной переменной копированием создается аналогичная локальная переменная. Именно с хранимой ею величиной работают команды программы. Как-то изменить глобальную переменную программа не может.

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Локальные переменные и функции имеют приоритет над глобальными в рамках «родной» программы. Это означает, что если имеется локальная и глобальная переменные (или функции) с одним именем, то обращение по этому имени будет адресоваться к локальной переменной (или функции).

Пример 4.3. Одноименные локальная и глобальная переменные

$$a := 2$$

$$\left|\begin{array}{l} a \leftarrow 3 \\ a \end{array}\right. = 3$$

$$a = 2$$

Возможность создания локальных функций появилась только в Mathcad 12. До этого попытка объявить функцию в программе вызывала ошибку. Создаются локальные функции точно так же, как обычные (только в качестве оператора присваивания используется «←»). Вызвать локальную функцию можно только из нижележащих строк программы. Вне программы она не доступна.

Пример 4.4. Задание в программе локальной функции

$$\left|\begin{array}{l} f(x,y,z) \leftarrow \sin(x) + \sin(y) + \sin(z) \\ value \leftarrow f\left(\frac{\pi}{3}, \frac{\pi}{5}, \frac{\pi}{2}\right) + f\left(\pi, \frac{\pi}{8}, \frac{\pi}{4}\right) + f\left(\frac{\pi}{10}, 0, \frac{\pi}{7}\right) \\ value \end{array}\right. = 4.287$$

Локальные функции нужно создавать тогда, когда определенное выражение должно вычисляться сразу в нескольких местах программы. Это позволяет избежать дублирования выражения, а следовательно, способствует уменьшению длины программы

и увеличению ее читабельности. Конечно, можно вынести выражение и во внешнюю функцию. Однако применение локальной функции техничнее, так как при этом сохраняется целостность программы, а, следовательно упрощается ее повторное использование.

Иногда в программах используется значительное число локальных переменных. Если при этом каждая переменная объявляется в отдельной строке, то программа может стать очень длинной. Соответственно, ее будет сложнее отлаживать, возникнут проблемы с распечаткой. В подобных случаях довольно заметно уменьшить длину программы можно, используя следующий прием. Представим, что нам нужно задать пять переменных. Для этого создадим в маркере программного блока матрицу-строку из пяти элементов. После этого определим каждую переменную в маркерах данной матрицы (см. пример 4.5).

Пример 4.5. Объявление нескольких переменных в одной матрице-строке

$$\begin{vmatrix} (a \leftarrow 1 \quad b \leftarrow \pi \quad c \leftarrow e \quad d \leftarrow 0 \quad f \leftarrow -1) \\ a + b + c + d + f \end{vmatrix} = 5.86$$

В матрице-строке могут быть прописаны совершенно любые действия, а не только операции присваивания. Это можно использовать для сокращения длины программы тогда, когда одновременно должно быть выполнено несколько однотипных действий.

Также сократить длину программы позволяет проведение присваивания в строке через запятую. Для этого поставьте курсор в маркер программного блока и последовательным нажатием клавиши «,» введите необходимое количество маркеров, после чего в каждом из них задайте переменную либо пропишите требуемое действие.

Пример 4.6. Объявление нескольких переменных в одной строке через запятую

$$\begin{vmatrix} a \leftarrow 4, b \leftarrow 5, c \leftarrow 8, d \leftarrow 9 \\ a \cdot b + c + 3d \end{vmatrix} = 55$$

При написании алгоритмов очень часто встречаются малопонятные для новичков в программировании выражения вроде n←n+1. На самом же деле ничего противоречащего логике в таких выражениях нет. Важно понимать, что оператор присваивания радикально отличается от оператора равенства, хотя в математике они традиционно обозначаются символом «=». Запись n←n+1 не означает, что n тождественно равно n+1. Она лишь показывает, что текущее значение n нужно увеличить на 1, а затем полученную величину присвоить n. Математически это можно записать рекуррентным выражением вида $n_{k+1}=n_k+1$. Выражению n←n+1 соответствует целых три разделенных во времени действия. Сначала считывается текущее значение переменной. Затем оно увеличивается на 1. И в последнюю очередь происходит переопределение n. Именно то, что эти действия происходят не одновременно, делает выражения рассматриваемого вида имеющими смысл.

Практически любая программа строится с использованием специальных управляющих операторов, вроде оператора цикла for или оператора условия if. Применение каждого из них мы рассмотрим весьма подробно ниже, а пока выделим лишь наиболее общие принципы, связанные с их заданием.

- Чтобы задать нужный оператор, используйте соответствующие кнопки панели Programming (Программирование). Просто набрать оператор с клавиатуры нельзя — он будет воспринят системой Mathcad как неизвестная функция. Кроме того, каждый оператор программирования имеет свое сочетание клавиш, узнать которое можно, подведя курсор к соответствующей ему кнопке. Постарайтесь запомнить эти сочетания клавиш, так как их использование значительно ускоряет набор программ.
- Такие операторы как, if, for, while, активируют код, помещенный в их левый маркер, в том случае, если выполняется условие в правом. Для задания условия используются такие операторы панели Boolean (Булевы), как «=», «>», «<», «≠», «≥», «≤». Можно задать и комплекс условий. Если некоторое действие должно быть выполнено, когда истину возвращают оба условия, то для их объединения нужно задействовать оператор логического И «∧» панели Boolean (Булевы). Если же действие должно быть проделано тогда, когда выполняется хотя бы одно условие из двух, для их объединения следует использовать оператор логического ИЛИ «∨». В тех же случаях, когда важно, чтобы выполнилось только одно условие из двух, применяется оператор исключающего ИЛИ «⊕». В принципе, в комплексе может быть и больше двух условий. Только стоит помнить о том, что для их объединения должен использоваться один и тот же оператор. Если же часть условий объединяется с логикой И, а часть — ИЛИ, то лучше их разнести по разным операторам.

Пример 4.7. Задание комплекса условий

Данная программа перемножает две соразмерные матрицы так, что каждый элемент матрицы-результата есть произведение соответствующих элементов перемножаемых матриц. Перед тем как запустить основной алгоритм, программа проверяет матрицы на соразмерность.

$$M1 := \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 2 \\ 3 & 2 & 1 \end{pmatrix} \qquad M2 := \begin{pmatrix} 3 & 2 & 1 \\ 2 & 1 & 2 \\ 1 & 2 & 3 \end{pmatrix}$$

```
if last(M1^⟨0⟩) = last(M2^⟨0⟩) ∧ last[(M1^T)^⟨0⟩] = last[(M2^T)^⟨0⟩]
    for i ∈ 0.. last(M1^⟨0⟩)
        for j ∈ 0.. last[(M1^T)^⟨0⟩]
            result_{i,j} ← M1_{i,j} · M2_{i,j}
    return result
return "Error!!!"
```

$$= \begin{pmatrix} 3 & 4 & 3 \\ 4 & 1 & 4 \\ 3 & 4 & 3 \end{pmatrix}$$

В качестве результата работы программы выводится значение того выражения, которое находится в последнем маркере главного программного блока. Ошибившись с его определением (или забыв задать его вовсе), можно получить совсем не тот ответ, который соответствует решению данной задачи. Впрочем, прервать работу программы и возвратить значение можно при выполнении любой ее строки. Для этого предназначен особый оператор return (см. пример 4.7).

Если код программы не возвратит никакого значения, то по умолчанию будет выведено значение 0. Например:

$$a := 1$$

$$\begin{vmatrix} \text{return "True"} & \text{if } a > 1 & = 0 \\ \text{return "False"} & \text{if } a < 1 & \end{vmatrix}$$

Из общих сведений об особенностях написания программ мы не обсудили еще одну важную тему: форму вывода результата. Нам уже хорошо знаком непосредственный вывод результата с помощью «=». Также программу можно присваивать переменной. При этом результат ее вычисления будет значением данной переменной. Наиболее же важен для практики третий способ вывода результата. Дело в том, что он позволяет создавать свои собственные функции, подобные встроенным функциям Mathcad. Суть его заключается в том, что любая программа может быть задана как функция. Причем параметры ее могут быть определены в общем виде (по своим особенностям они будут схожи с локальными переменными). Подставив в маркеры созданной таким образом функции какие-то числовые значения, можно получить непосредственно результат программного расчета. Такой подход позволяет значительно экономить время в том случае, если программа должна быть использована много раз.

Пример 4.8. Создание функции расчета вероятности нахождения нормально распределенной случайной величины в заданном интервале

$$\text{Ver}(\alpha,\beta,\sigma,m) := \begin{vmatrix} A \leftarrow \dfrac{1}{\sigma\cdot\sqrt{2\pi}}\cdot\displaystyle\int_{\alpha}^{\beta} e^{\frac{-(x-m)^2}{2\sigma^2}}\,dx \\ A \end{vmatrix}$$

$$\text{Ver}(-3,3,3,1) = 0.656 \qquad \text{Ver}(-6,6,3,1) = 0.942$$

$$\text{Ver}(-9,9,3,1) = 0.996 \qquad \text{Ver}(-100,100,3,1) = 1$$

Внимательно прочитав данный раздел и тем самым познакомившись с наиболее важными программными операторами, вы уже наверняка будете способны написать любую не очень сложную программу. Однако в связи с тем что любой оператор имеет ряд технических особенностей, рассмотрим использование каждого из них по отдельности.

4.2. Операторы цикла (for, while, break, continue)

В языке программирования Mathcad (как, впрочем, и в большинстве С-подобных языков высокого уровня) имеются два оператора, предназначенных для задания цикла. С помощью первого — оператора простого цикла for — можно организовать выполнение операции или проверку условия для ряда конкретных значений переменной. Задать оператор for можно, равно как и все остальные операторы, с помощью команды панели Programming (Программирование) или сочетанием клавиш (Ctrl+Shift+«'»). Оператор for имеет три маркера:

for ▪ ∈ ▪

▪

В двух верхних маркерах, соединенных символом принадлежности, задается имя переменной, по которой организуется цикл, и ряд принимаемых ею значений. В нижнем маркере определяется операция или комплекс операций, которые должны быть выполнены для каждого значения переменной. Задать ряд значений можно разными способами. Например, можно организовать ряд, члены которого представляют собой последовательность целых чисел. Для этого в правый верхний маркер введите оператор ранжированной переменной (Range Variable) (сделать это можно либо с панели Matrix (Матрица), либо использовав клавишу «;»). Данный оператор имеет два маркера, в которых следует определить первое и последнее значения переменной.

Пример 4.9. Организация цикла по целочисленной переменной

```
Factorial(n) := | s ← 1
                | return 1 if n = 0
                | return "Error" if n < 0 ∨ trunc(n) ≠ n
                | for i ∈ 1..n
                |   s ← s·i
                | s
```

Factorial(3) = 6 Factorial(5) = 120

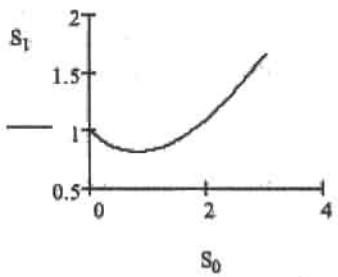
Factorial(2.3) = "Error" Factorial(−2) = "Error"

Описанный метод позволяет задать ряд значений с шагом, равным 1. Однако зачастую значения переменной должны изменяться с меньшим или большим шагом. Конечно, этого можно добиться и при задании целочисленной последовательности введением коэффициента, однако можно поступить и проще. Задать произвольным образом шаг можно и непосредственно в операторе for. Для этого в правом маркере оператора ранжированной переменной введите через запятую первое и второе значения в ряде переменной (разница между ними и задаст шаг).

Пример 4.10. Организация цикла с произвольным шагом переменной (на примере решения задачи Коши по методу Эйлера)

$$\text{Koshi}(Df, t0, y0, h, tmax) := \left| \begin{array}{l} y_0 \leftarrow y0 \\ n \leftarrow 0 \\ \text{for } t \in t0, t0 + h .. tmax - h \\ \quad \left| \begin{array}{l} y_{n+1} \leftarrow y_n + h \cdot Df(t + h, y_n) \\ T_n \leftarrow t \\ n \leftarrow n + 1 \end{array} \right. \\ k \leftarrow \begin{pmatrix} T \\ y \end{pmatrix} \end{array} \right.$$

$$f(t,y) := \frac{t-y}{2} \qquad S := \text{Koshi}(f, 0, 1, 0.0001, 3)$$



Ряд значений переменной, определенный описанными способами с использованием оператора ранжированной переменной, может быть задан и непосредственно в рабочей зоне с помощью того же оператора. Поэтому в большинстве случаев задача, которая решается программно организацией простого цикла, может быть решена и без использования языка программирования.

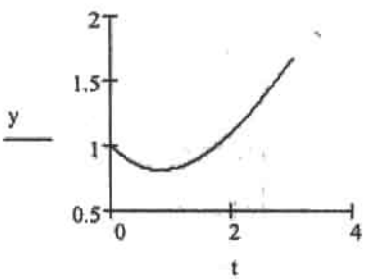
Пример 4.11. Решение задачи Коши по методу Эйлера без использования языка программирования

$$t_0 := 0 \qquad y_0 := 1 \qquad tmax := 3 \qquad h := 0.001$$

$$f(t,y) := \frac{t-y}{2} \qquad i := 0 .. \frac{tmax - t_0}{h}$$

$$t_{i+1} := t_i + h$$

$$y_{i+1} := y_i + h \cdot f(t_i, y_i)$$



В том случае, если операция или комплекс операций должны быть просчитаны при ряде некоторых конкретных значений переменной, причем ряд этот нельзя задать математически в общем виде, его можно непосредственно определить в правом верхнем маркере оператора for в виде вектора.

Пример 4.12. Задание цикла переменной по вектору значений

```
n ← 0                                   = (1  9  25  49  121)
for i ∈ (1  3  5  7  11)
    s_(0,n) ← i^2
    n ← n + 1
s
```

С помощью второго оператора цикла while (Пока) (сочетание клавиш Ctrl+«]») можно организовать цикл, который будет работать до тех пор, пока выполняется некоторое условие. Оператор while имеет два маркера, в которые вводятся соответственно условие работы цикла и выражения операций, которые должны быть проделаны на каждом его витке:

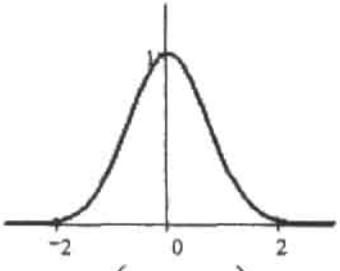
```
while ■
    ■
```

Важным преимуществом цикла while по сравнению с циклом for является то, что количество его витков не нужно определять явно. Итерации будут совершаться до тех пор, пока будет выполняться условие в правом маркере. В принципе, условие может быть истинным всегда. В этом случае цикл будет работать бесконечно долго (в Mathcad нет ограничения на время выполнения кода). Однако пользы от такого цикла не будет. Бесконечные циклы — это одна из наиболее распространенных причин сбоев в программах. Поэтому, если вы обнаружите, что написанный вами алгоритм выполняется слишком долго, проверьте на корректность условие в маркере цикла while.

Пример 4.13. Использование оператора while

```
Integral(f, TOL) := (I ← 0  n ← 1)
                    (x_0 ← 0  x_1 ← TOL)
                    (Δx_0 ← 0  ΔI ← 1)
                    while ΔI > TOL
                        Δx_n ← 2TOL / f(x_(n-1))
                        x_n ← x_(n-1) + (Δx_(n-1) + Δx_n)/2
                        ΔI ← Δx_n · f(x_n)
                        (I ← I + ΔI  n ← n + 1)
                    I-2
```

$$func(x) := e^{-x^2}$$



$Integral(func, 10^{-3}) = 1.77161$

$Integral(func, 10^{-5}) = 1.77245$

$$\int_{-\infty}^{\infty} func(x)\,dx = 1.77245$$

В примере 4.13 рассмотрен один из способов численного подсчета интеграла с бесконечными пределами. Произвести такого рода расчет с помощью стандартных методов

численного интегрирования невозможно, поскольку при этом пришлось бы производить суммирование бесконечно большого количества членов. Однако с учетом того, что рассматриваемый интеграл является сходящимся и на практике нас всегда интересует результат ограниченной точности, можно, в принципе, производить подсчет до тех пор, пока площади элементарных прямоугольников (трапеций или криволинейных трапеций, ограниченных параболой, – в зависимости от выбранной методики) не достигнут по величине какого-то минимума, которым можно пренебречь. В случае рассматриваемой программы достижение этого минимума и является условием остановки работы цикла:

while $\Delta I > TOL$

Однако если вы попробуете подсчитать интеграл с помощью метода средних прямоугольников с фиксированной шириной интервала разбиений, то получить правильный ответ будет весьма проблематично. Это связано с тем, что функция на всем своем промежутке изменяется весьма неравномерно. Так, если шаг будет выбран слишком большим, то значительная ошибка будет допущена при интегрировании функции в окрестности нуля (а именно эта область и определяет основное значение интеграла). Если же шаг выбрать достаточно малым, то подсчет будет вестись очень долго. Поэтому, очевидно, каким-то образом следует связать значение функции в центре промежутка с его шириной, а также с уровнем точности. Нетрудно догадаться, что чем больше значение функции в центре отрезка разбиения, тем меньше должна быть его ширина. С точностью же должна быть пропорциональная связь: чем выше точность, тем меньше шаг. Наиболее просто можно осуществить связь между параметрами в следующем виде:

$$\Delta x_n \leftarrow \frac{2TOL}{f(x_{n-1})}$$

Написанная программа позволяет с произвольной точностью подсчитывать интегралы любых симметричных четных функций с экстремумом в x=0 (естественно, нужно учитывать, что чем выше точность, тем больше время расчета). Например:

$$func(x) := (|x|)^{-|x|}$$

$$Integral(func, 10^{-5}) = 3.9909 \qquad \int_{-\infty}^{\infty} func(x)\,dx = 3.99091$$

Довольно часто в алгоритмах нужно предусматривать ситуации, в которых работа цикла должна быть досрочно прервана. К примеру, если численный метод сходится в случае некоторых задач медленно, может накапливаться ошибка, искажающая результат. Чтобы избежать получения ошибочного ответа, нужно прервать работу цикла, если будет превышен лимит на количество итераций. Одной из технических возможностей реализации этого является использование оператора break (Прервать) (вводится сочетанием клавиш Ctrl+Shift+«[»). Ввиду того, что цикл бывает нужно остановить при выполнении некоторого условия, оператор break почти всегда используется с условным оператором if (или, реже, с оператором ошибки on error).

Пример 4.14. Использование оператора break

$$\text{Integral2}(f,a,b,N) := \left| \begin{array}{l} \left(\Delta x \leftarrow \frac{b-a}{N} \quad x_0 \leftarrow a \quad Z \leftarrow 0\right) \\ \text{for } i \in 1..N \\ \quad \left| \begin{array}{l} x_i \leftarrow x_{i-1} + \Delta x \\ \text{if } \left|f(x_i)\right| > 10^{15} \\ \quad \left| \begin{array}{l} Z \leftarrow \text{"Singularity"} \\ \text{break} \end{array} \right. \\ Z \leftarrow \frac{f(x_i)+f(x_{i-1})}{2}\cdot\Delta x + Z \end{array} \right. \\ Z \end{array} \right.$$

$f1(x) := \frac{1}{x^2}$ $\qquad f2(x) := x^2$

Integral2(f1, −5, 5, 1000) = "Singularity"

Integral2(f2, −5, 5, 1000) = 83.3335

Integral2(f2, −5, 5, 10000) = 83.33333

$$\int_{-5}^{5} f2(x)\,dx = 83.33333$$

В примере 4.14 рассмотрена программа вычисления интеграла по методу трапеций. Использовать данный метод в случае существования точек разрыва второго рода нельзя, поэтому следует предусмотреть такого рода ситуации. А сделать это можно очень просто: в том случае, если величина функции на границе элементарной трапеции превысит некоторое значение, то цикл должен быть остановлен (оператор break) и в качестве ответа должно быть выдано сообщение о проблеме при вычислении:

$$\left| \begin{array}{l} \text{if } \left|f(x_i)\right| > 10^{15} \\ \quad \left| \begin{array}{l} Z \leftarrow \text{"Singularity"} \\ \text{break} \end{array} \right. \end{array} \right.$$

Иногда алгоритм содержит несколько условий, причем при выполнении одного из них рассмотрение остальных приведет к ошибке. В этом случае нужно после выполнения операций, прописанных в первом условии, сразу же перейти к рассмотрению следующего значения циклической переменной. Чтобы это сделать, следует использовать специальный оператор continue (Продолжить) (сочетание клавиш Ctrl+«]»). Употребление оператора continue аналогично оператору break.

Пример 4.15. Использование оператора continue

$$\text{Sum} := \left| \begin{array}{l} S \leftarrow 0 \\ \text{for } n \in 0..1000 \\ \quad \left| \begin{array}{l} \text{continue} \quad \text{if } \text{trunc}\left(\frac{n}{3}\right) = \frac{n}{3} \\ S \leftarrow S + n \end{array} \right. \\ S \end{array} \right. \qquad \text{Sum1} := \left| \begin{array}{l} S \leftarrow 0 \\ \text{for } n \in 0..1000 \\ \quad \left| \begin{array}{l} S \leftarrow S + n \quad \text{if } \text{trunc}\left(\frac{n}{3}\right) \neq \frac{n}{3} \\ S \end{array} \right. \\ S \end{array} \right.$$

$\text{Sum} = 3.33667 \times 10^5$ $\qquad \text{Sum1} = 3.33667 \times 10^5$

В примере 4.15 приведены алгоритмы подсчета суммы всех неделящихся без остатка на 3 чисел от 0 до 1000. В одном из них используется оператор continue, с помощью которого пропускаются все неподходящие под условие значения. Само же условие задается с помощью встроенной функции trunc(n), которая служит для определения целой части числа.

Циклы программ Mathcad работают довольно быстро. Убедиться в этом можно, используя появившуюся в Mathcad 12 функцию time. Данная функция возвращает машинное время в секундах. Записав в переменную, когда начал работать цикл, можно найти время его работы как разность значения, возвращенного time по завершении прокрутки цикла, и величины этой переменной. Например, проверим, за сколько будет просуммирован 1 000 000 членов гармонического ряда:

$$\begin{aligned}&\text{time0} \leftarrow \text{time}(0) \qquad = 1.412\\&S \leftarrow 0\\&\text{for } i \in 1..1000000\\&\quad S \leftarrow S + \frac{1}{i}\\&\text{time}(1) - \text{time0}\end{aligned}$$

Итак, миллион членов ряда был просуммирован всего за полторы секунды. Столь высокая производительность дает возможность реализовывать в Mathcad алгоритмы, связанные с очень объемными вычислениями.

Функция time также может использоваться для учета задач, для которых реализуемый алгоритм будет плохо сходящимся. Если окажется, что подсчет ведется неоправданно долго, работа программы должна быть остановлена и выведено сообщение об ошибке.

4.3. Условные операторы (if, otherwise)

Из всех программных операторов оператор условия if (Если) (сочетание клавиш Shift+«]») является, пожалуй, наиболее важным. Его приходится использовать практически во всех создаваемых алгоритмах.

Условный оператор if имеет два маркера:

■ if ■

В правый маркер вводится условие, в левый — операция, которая должна быть проделана в случае, если условие выполнится (если же оно не выполняется, система просчитывает программу, пропуская данный фрагмент). Как уже говорилось, в маркер оператора может быть внесено несколько условий.

Оператор otherwise (Иначе) предназначен для определения того действия, которое должно быть выполнено, если условие оператора if (Если) окажется неистинным.

Одновременно может быть использовано несколько условных операторов if (Если). Оператор otherwise (Иначе) в таком случае будет задействован, если не выполнятся условия всех операторов if (Если).

Пример 4.16. Решение системы линейных уравнений

$$M1 := \begin{pmatrix} 2 & 3 & 4 \\ 4 & 5 & 6 \\ 4 & 5 & 6 \end{pmatrix} \quad N1 := \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \qquad M2 := \begin{pmatrix} 3 & 4 & 6 \\ 6 & 7 & 3 \\ 7 & 2 & 1 \end{pmatrix} \quad N2 := \begin{pmatrix} 11 \\ 23 \\ 78 \end{pmatrix}$$

$$\text{Linear}(M, N) := \begin{cases} r \leftarrow \text{"Matrix is singular"} \quad \text{if } |M| = 0 \\ r \leftarrow M^{-1} \cdot N \quad \text{otherwise} \\ r \end{cases}$$

$$\text{Linear}(M1, N1) = \text{"Matrix is singular"} \qquad \text{Linear}(M2, N2) = \begin{pmatrix} 13.491 \\ -8.642 \\ 0.849 \end{pmatrix}$$

Для закрепления материала подробно разберем решение несколько более сложной задачи, сочетающей как условия, так и циклы.

Пример 4.17. Вектор коэффициентов полинома

$$\text{coeffs}(\text{polinom}) := \begin{cases} D(x,n) \leftarrow \dfrac{d^n}{dx^n}\text{polinom}(x) \\ n \leftarrow 0 \\ \text{while } D(0,n) \neq 0 \vee D(1,n) \neq 0 \\ \quad \begin{cases} t_n \leftarrow \dfrac{D(0,n)}{n!} \\ n \leftarrow n + 1 \\ \text{if } n > 5 \\ \quad \begin{cases} t \leftarrow \text{"Too high power"} \\ \text{break} \end{cases} \end{cases} \\ t \end{cases}$$

$$f1(x) := x^4 + 3x^2 + x + 1$$

$$\text{coeffs}(f1) = \text{"Too high power"}$$

$$f2(x) := x^3 + x - 7$$

$$\text{coeffs}(f2) = \begin{pmatrix} -7 \\ 1 \\ 0 \\ 1 \end{pmatrix} \qquad f2(x)\ \text{coeffs}, x \rightarrow \begin{pmatrix} -7 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

В примере 4.17 приведена программа, которая решает ту же задачу, что и оператор coeffs панели Symbolic (Символьные): создает вектор из коэффициентов полинома. Эта программа последовательно вычисляет производные членов полинома. При этом на каждом шаге появляется свободный член, который можно вычислить как значение функции производной в нуле. Нетрудно догадаться, что, поделив полученное значение свободного члена на факториал порядка той производной, при вычислении которой этот свободный член был получен, можно узнать величину коэффициента.

Как вы видите, математически все очень просто. Однако даже в такой простой задаче есть несколько технических тонкостей.

- Условие продолжения работы цикла. В нашей программе оно определено тем, что значения функции производной рассматриваемого порядка в точках $x=0$ и $x=1$ не должны одновременно равняться 0 (такое равенство возможно в подавляющем большинстве случаев, только если сама производная есть 0). Важным моментом является то, в какой форме будет записан комплекс условий. Цикл должен продол-

жать работу, даже если одно из условий не выполняется. Значит, для связки условий нужно использовать логическое ИЛИ «∨».

- Производные в программе вычисляются локальной функцией D(x,n). Однако такую функцию можно создать только в Mathcad 12. Если бы данная программа писалась в Mathcad 11, то функцию D(x,n) пришлось бы задать как глобальную выше программы.
- Использование оператора кратного дифференцирования ограничивает возможности программы, поскольку высшая степень производной, которая может быть вычислена, равна пяти. В том случае, если степень полинома окажется выше четырех, будет выдано сообщение об ошибке: This Number must be between 0 and 5 (Это число должно быть между 0 и 5). Дабы избежать ошибки, с помощью оператора if прописываем условие, что если n превышает пять, то работа программы должна быть остановлена (оператор break) и в качестве ответа выдано сообщение Too high power (Слишком высокая степень). Для того же, чтобы написать программу, способную создавать вектор из коэффициентов полинома более высокой степени, нужно совместить несколько операторов кратного дифференцирования.

Все рассмотренные выше программы по уровню сложности можно отнести к элементарным и легким. В дальнейшем мы будем писать куда более сложные алгоритмы. И если сейчас вам что-то кажется недоступным и непонятным — не стоит огорчаться! Немного практики — и вскоре вы без особого труда будете писать даже самые нестандартные программы.

4.4. Возврат значений (return)

С помощью оператора return (Возвратить) можно прервать работу программы и возвратить некоторое значение. Обычно данный оператор используется при описании действий алгоритма в случае ошибочной ситуации. Впрочем, зачастую применение return избавляет от необходимости использовать оператор otherwise (Иначе), способствуя тем самым упрощению программы.

Пример 4.18. Использование оператора return (на примере создания функции, вычисляющей сумму n членов гармонического ряда)

$$\text{Garm}(n) := \begin{cases} \text{return "Error"} \quad \text{if } n \le 0 \vee \text{trunc}(n) \ne n \\ Z \leftarrow \ln(n) + 0.5772 \quad \text{otherwise} \\ Z \end{cases}$$

$\text{Garm}(-6) = \text{"Error"}$ $\quad \text{Garm}(1.1) = \text{"Error"}$

$\text{Garm}(1000) = 7.485$ $\quad \sum_{n=1}^{1000} \frac{1}{n} = 7.485$

4.5. Перехват ошибок (on error)

Любая вычислительная схема или математическая модель может работать корректно только при определенных условиях и ограничениях. Так, например, в большинстве методов численного решения уравнений обязательным требованием на каждом шаге работы алгоритма является то, что производная не может принимать значение, равное 0. Если же окажется так, что очередное приближение приведет к необходимости произвести подсчет вблизи экстремума или точки разрыва, то возникнет неопределенность, работа программы будет остановлена и будет выдано сообщение об ошибке.

Однако избежать подобных ситуаций все же можно. Для этого следует использовать специальный оператор обхода ошибки on error (вводится также сочетанием Ctrl+«'»):

■ on error ■

Данный оператор по своему синтаксису полностью соответствует условному оператору if. В правый его маркер следует ввести величину или выражение, ошибка в вычислении которого должна быть зарегистрирована. В левом маркере следует прописать условие, которое должно быть выполнено при возникновении ошибочной ситуации. В лучшем случае таким образом можно исправить или обойти возникшую проблему. Так, при численном решении уравнения, если приближение оказывается в окрестности экстремума и возникает ошибка при делении на производную, можно, используя оператор on error, не допустить остановки работы алгоритма из-за такой незначительной трудности. Для этого нужно просто прибавить к значению приближения небольшую величину (данное условие прописывается в левом маркере оператора ошибки, а в правый соответственно заносится сама расчетная формула). Это, с одной стороны, ничего не изменит в эффективности определения результата, с другой — позволит предельно просто разрешить трудную ситуацию.

Пример 4.19. Создание функции численного решения уравнений с использованием оператора on error

$$\begin{aligned}
&\text{Nev}(f, x0, TOL) := \left|\begin{array}{l}
D(x) \leftarrow \frac{d}{dx} f(x) \\
\left(n \leftarrow 0 \quad x_0 \leftarrow x0\right) \\
\text{while } \left|f\left(x_n\right)\right| \geq TOL \\
\quad \left|\begin{array}{l}
x_n \leftarrow x_n + 0.0001 \text{ on error } \frac{f\left(x_n\right)}{D\left(x_n\right)} \\
x_{n+1} \leftarrow x_n - \frac{f\left(x_n\right)}{D\left(x_n\right)} \\
n \leftarrow n + 1
\end{array}\right. \\
x_n
\end{array}\right. \\
&f(x) := x^3 - \sin(x)\cdot x + 1 \\
&\text{Nev}(f, 1, 0.0001) = -0.77246 \\
&f(-0.77246) = -2.18 \times 10^{-5}
\end{aligned}$$

В некоторых случаях (когда причина сбоя точно известна) оператор ошибки может быть заменен условным оператором. Так, условие обхода проблемной ситуации в примере 4.19 можно задать и как:

$$x_n \leftarrow x_n + 0.0001 \text{ if } \frac{f\left(x_n\right)}{D\left(x_n\right)}$$

В тех случаях, когда разрешить возникшую проблему невозможно, следует ограничиться выводом сообщения, раскрывающего причину остановки программы. Сделать это можно очень просто, присвоив переменной, значение которой выводится как ответ, соответствующую текстовую строку.

Пример 4.20. Вывод сообщения об ошибке

$$T(x) := \text{"can't devide by zero"} \quad \text{on error } \frac{1}{x^2 - 1}$$

$$T(\pi) = 0.113 \qquad T(0.3) = -1.099$$

$$T(1) = \text{"can't devide by zero"}$$

Обратите внимание, как интересно задана функция в примере 4.20: оператор on error возвращает значение самого выражения в том случае, если в нем нет ошибки.

Если вас не устраивает форма сообщения в виде текстовой строки, то его можно вывести в принятой в Mathcad форме, то есть в виде желтой всплывающей подсказки. Для этого нужно использовать специальную функцию error(«S»), где S — текст сообщения (рис. 4.2).

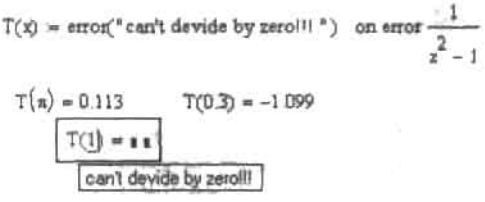


Рис. 4.2. Использование функции error

4.6. Поиск ошибок в программах

До тех пор пока вы не приобретете достаточно опыта, ваши программы вряд ли будут работать с первого же запуска. Почти наверняка они будут содержать множество ошибок и незамеченных опечаток. Конечно, большинство из них может быть легко найдено, но если вы сталкиваетесь с ошибкой такого типа впервые, определить ее расположение будет непросто. Особенно это касается больших программ, запутаться в которых очень и очень легко даже опытному программисту.

В универсальных языках программирования высокого уровня ошибки обычно находит сама система при процедуре, называемой компиляцией. В Mathcad подобной процедуры нет, однако возможность быстро и легко определять ошибки в синтаксисе алгоритма имеется.

Рассмотрим конкретную ошибочную ситуацию (рис. 4.3): при попытке использования программы, вычисляющей интеграл по методу трапеций, система выдает сообщение This variable is undefined (Эта переменная не определена). При проверке всех используемых в программе переменных они оказываются заданными верно. В чем же реальная причина сбоя? У авторов этой книги, столкнувшихся с данной проблемой при написании раздела об интегрировании, самостоятельно найти ошибку не получилось. Если не получится и у вас, то стоит обратиться за помощью к системе. Для этого щелкните правой кнопкой мыши на функции, в которой возникла ошибка. При этом откроется ее контекстное меню, в котором следует выбрать команду, расположенную в его верхней строке, Trace Error (Поиск ошибки). Если вы это сделаете, то откроется специальная одноименная панель, содержащая различные параметры поиска ошибки.

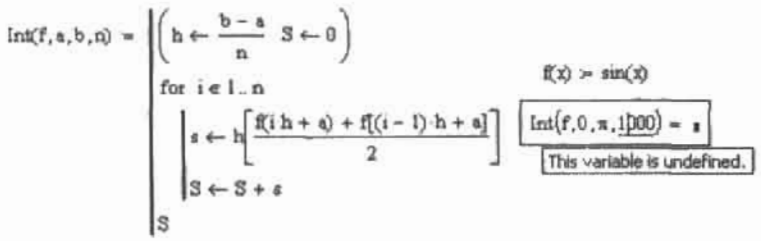


Рис. 4.3. Ошибочная ситуация

Панель Trace Error (Поиск ошибки) (рис. 4.4) содержит несколько кнопок.

Рис. 4.4. Панель Trace Error

- First (Первая). Если нажать эту кнопку, то тот фрагмент программы, который вызывает ошибку, будет выделен курсором (рис. 4.5). В нашем случае будет выделен сомножитель *h* в первом выражении в цикле for. Проанализировав, где именно в выделенном выражении может быть ошибка, обнаруживаем, что между *h* и скобкой с полусуммой функций нет знака умножения. А это означает, что *h* рассматривается как функция со значением аргумента, равным величине в скобках. Однако, так как *h* не является функцией одного переменного, запись такая некорректна, что приводит к появлению сообщения об ошибке. Кстати, с учетом того, что знаки умножения по умолчанию в Mathcad не отображаются, найти ошибку такого рода очень сложно.

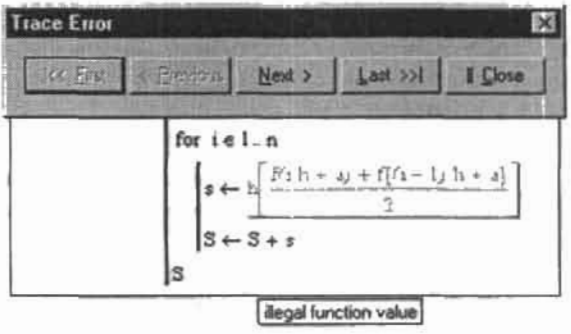


Рис. 4.5. В программе найдена ошибка

- Previous (Предыдущая). Так как влияние ошибки распространяется иногда уровнями (например, при неправильном задании индекса), бывает нужным произвести исправления в разных фрагментах программы. Чтобы подняться на уровень выше, и используется данная команда.
- Next (Следующая). Команда, идентичная предыдущей по функциям, но противоположная по направлению.

- Last (Последняя). Команда помещает курсор на последний уровень распространения ошибки.
- Close (Закрыть). Команда служит для сворачивания окна Trace Error (Поиск ошибки).

Использовать панель Trace Error (Поиск ошибки) можно не только в случае программ, но и везде, где есть последовательность выполнения каких-то действий.

Объективно говоря, производить отладку программ Mathcad довольно сложно. Панель Trace Error (Поиск ошибки) и всплывающие сообщения могут помочь лишь тогда, когда допущена синтаксическая ошибка. Если же ошибка имеется в самом алгоритме, то система ее, естественно, не обнаружит. Как же действовать в подобном случае? В первую очередь нужно внимательно проанализировать код алгоритма. Вполне вероятно, что причина сбоя — это обычная описка. Если же ничего явно неверного в коде обнаружено не будет, программу следует исследовать в работе. Для этого обычно нужно проверять, как меняются переменные, когда срабатывают условия, или сколько итераций делают циклы. Полученную информацию необходимо записывать в вектор, который должен быть возвращен как результат работы программы.

4.7. Символьное вычисление программ

Аналогично большинству операторов и встроенных функций, программы Mathcad могут быть просчитаны как численно, так и символьно. Однако аналитический расчет алгоритмов пользователя не согласован с операторами on error и return, поэтому он может быть произведен лишь в случае программ, в коде которых данных операторов нет. Также символьно невозможно рассчитывать рекурсивные алгоритмы. В качестве примера аналитически просчитываемой программы можно привести создание функции, находящей ряд Фурье для произвольной функции.

Пример 4.21. Символьное вычисление программ

$$\text{Fur}(f,n) := \left| \begin{array}{l} s \leftarrow \frac{1}{2\cdot\pi}\cdot\int_0^{2\cdot\pi} f(t)\,dt \\ m \leftarrow 1 \\ \text{while } m \le n \\ \quad \left| \begin{array}{l} s \leftarrow s + \frac{1}{\pi}\cdot\int_0^{\pi} f(t)\cdot\cos(m\cdot t)\,dt\cdot\cos(m\cdot t) + \frac{1}{\pi}\cdot\int_0^{\pi} f(t)\cdot\sin(m\cdot t)\,dt\cdot\sin(m\cdot t) \\ m \leftarrow m + 1 \\ s \end{array} \right. \\ s \end{array} \right.$$

$$f(t) := t + \pi$$

$$\text{Fur}(f,2) \rightarrow 2\cdot\pi - \frac{2}{\pi}\cdot\cos(t) + 3\cdot\sin(t) - \frac{1}{2}\cdot\sin(2\cdot t)$$

$$\text{Fur}(f,4) \rightarrow 2\cdot\pi - \frac{2}{\pi}\cdot\cos(t) + 3\cdot\sin(t) - \frac{1}{2}\cdot\sin(2\cdot t) - \frac{2}{9\cdot\pi}\cdot\cos(3\cdot t) + \sin(3\cdot t) - \frac{1}{4}\cdot\sin(4\cdot t)$$

4.8. Рекурсия

Рекурсией в программировании называется вызов подпрограммой самой себя. Рекурсия является одной из самых важных структур управления, так как с ее использованием максимально просто реализуется ряд чрезвычайно полезных алгоритмов.

В том, что функция может вызвать саму себя, нет ничего удивительного. Чтобы понять это, нужно четко отделить программу, определяющую функцию (она одна), и активацию функции (их может быть несколько). Активация — это особый объект данных, создаваемый в памяти при вызове функции на основании ее определения. Проще говоря, активация — это выполняющаяся функция. Одновременно в памяти может быть несколько активаций — они образуют стек (очередь) в том порядке, в котором они были вызваны. Активация, сформированная последней, выполняется первой — и наоборот. При вызове функции, вызывающей другую функцию, создается два объекта активации: первая функция «ждет», пока выполнится вторая, и лишь затем выполняется сама. Никаких принципиальных различий от того, вызовет ли функция свое определение или определение другой функции, нет: в любом случае будет создано два объекта активации.

Классическим примером рекурсивной функции, приводимым практически в любой книге по программированию, является функция, вычисляющая факториал числа ($n!=1\cdot2\cdot3\cdot...\cdot(n-2)\cdot(n-1)\cdot n$). Не будем нарушать традицию и мы (см. пример 4.22).

Пример 4.22. Рекурсивное вычисление факториала

$$\text{factorial}(n) := \begin{cases} \text{error("Parameter is defined incorrectly")} & \text{if } n < 0 \vee \text{trunc}(n) \neq n \\ \text{return } 1 & \text{if } n = 0 \\ \text{return } n \cdot \text{factorial}(n-1) & \end{cases}$$

$$\text{factorial}(0) = 1 \qquad \text{factorial}(3) = 6 \qquad \text{factorial}(5) = 120$$

Проанализируем работу созданной программы на примере вычисления факториала от 3.

Итак, интерпретатор Mathcad добирается до строки factorial(3). При этом создается объект активации и выполняются прописанные в определении функции действия. Так как 3 – это целое положительное число, то сразу вычисляется выражение n·factorial(n–1). При этом происходит вызов функции factorial() со значением параметра, равным 2. Создается второй объект активации и располагается в стеке первым. Его выполнение сопровождается появлением еще двух объектов активации — со значением n=1 и n=0. При проделывании кода для верхнего в стеке объекта активации выполняется условие n=0, что сопровождается разрушением самого объекта и возвращением в точку вызова — нижележащую активацию функции — значения 1. При этом выражение 1·factorial(0) получает конечное значение, и уже данный объект активации выгружается, возвратив активации с n=2 значение 1. Следующие два шага характеризуются разрушением еще двух активаций, которое сопровождается умножением передаваемой величины на 2 и 3. В результате в точку вызова функции factorial() возвращается значение 6.

Можно легко написать алгоритм, при выполнении которого рекурсивная цепь объектов активации будет неограниченной. Естественно, что это приведет к исчерпанию ре-

сурсов машины и, как следствие, к зависанию Mathcad или даже операционной системы. Чтобы этого не произошло, существуют ограничения на длину рекурсивной цепи. Она не может быть длиннее 100 объектов активации. Иначе система выведет сообщение об ошибке: This evaluation can not be evaluated. It may have resulted in an overflow or infinite recursion (Это выражение не может быть вычислено. Возможно, причина в переполнении или бесконечной рекурсии).

В программировании рекурсия обычно используется в тех случаях, в которых, выражаясь языком теории алгоритмов, нужно обойти граф (дерево) неизвестной степени вложенности. Например, с ее помощью можно просмотреть все элементы вложенного массива с неизвестной структурой или решить задачу Эйлера о ходе коня (конь должен обойти шахматную доску, побывав в каждой клетке только один раз). Впрочем, очень часто посредством рекурсии весьма изящно можно решить задачи, которые при более «приземленном» подходе решаются с помощью циклов (кстати, такой задачей является задача о вычислении факториала). В качестве примера такой задачи решим знаменитую задачу Леонардо Фибоначчи, которую он сформулировал и решил в 1202 году. Ее условие следующее. Сколько пар кроликов можно получить от одной пары за год (за два года, за n лет)? При этом используются такие допущения: каждая пара кроликов производит новую пару раз в месяц, новая пара начинает размножаться через месяц, кролики не дохнут.

Пример 4.23. Решение задачи Фибоначчи посредством рекурсии

```
Fibonacci(n) := | return 1 if n = 0 ∨ n = 1
                | s ← 0
                | while n ≥ 2
                |     | s ← s + Fibonacci(n − 2)
                |     | n ← n − 1
                | s
```

Fibonacci(0) = 1 Fibonacci(1) = 1 Fibonacci(2) = 1

Fibonacci(3) = 2 Fibonacci(4) = 3 Fibonacci(5) = 5

Fibonacci(6) = 8 Fibonacci(12) = 144

Идея написанного алгоритма довольно очевидна. Количество месяцев, которое отводится одной паре кроликов на размножение, передается функции Fibonacci в качестве параметра n. В течение этого времени она произведет n−2 новых пар кроликов. Каждая пара сама начнет размножаться. Имитировать этот процесс можно рекурсивным вызовом функции Fibonacci. Однако важно учесть, что у каждой последующей пары в потомстве будет времени на один месяц меньше, чем у предыдущей. Сделать это можно, уменьшая значение n в цикле while.

Справочная система Mathcad предлагает более простую с точки зрения кода, но куда более сложную для понимания программу, решающую задачу Фибоначчи. Попробуйте самостоятельно понять, как она работает.

Пример 4.24. Решение задачи Фибоначчи из справки Mathcad

$$\text{fibonacci}(n) := \begin{cases} \text{return } \text{fibonacci}(n-1) + \text{fibonacci}(n-2) \text{ if } n > 1 \\ \text{return } 1 \text{ if } n = 1 \\ \text{return } 0 \text{ if } n = 0 \end{cases}$$

$$i := 0..10 \qquad f_i := \text{fibonacci}(i) \qquad f^T = (0\ 1\ 1\ 2\ 3\ 5\ 8\ 13\ 21\ 34\ 55)$$

Фибоначчи был крупнейшим математиком средневековья. И открытая им последовательность чисел — 0, 1, 1, 2, 3, 5, 8... — имеет исключительно важное значение. Большое количество интереснейших алгоритмов сводится к числам Фибоначчи. Например, отношение двух соседних чисел Фибоначчи при $n \to \infty$ дает не что иное, как золотое сечение:

$$\frac{\text{fibonacci}(10)}{\text{fibonacci}(9)} = 1.618 \qquad \frac{1+\sqrt{5}}{2} = 1.618$$

Заметьте также, что каждое последующее число Фибоначчи есть сумма двух предыдущих.

Многие задачи решаются с использованием рекурсии очень красиво. Однако порой важно учитывать, что рекурсия куда более требовательна к ресурсам машины по сравнению с циклами. Поэтому в случае алгоритмов, требующих объемных вычислений, отдавать предпочтение все же лучше циклам, применяя рекурсию тогда, когда альтернатив ей нет.

4.9. Пример программирования: решение задачи Эйлера о ходе коня

В уже изученных нами разделах этой главы было немало примеров программирования в Mathcad. Однако почти все они были или элементарными или узкоспециальными, иллюстрирующими какую-то одну возможность. На практике же приходится создавать значительно более сложные программы. Чтобы писать подобные программы, необходим определенный опыт — одного лишь знания теории тут недостаточно. Поэтому, чтобы «набить руку», мы решим посредством языка программирования Mathcad классическую задачу теории алгоритмов, известную как задача Эйлера о ходе коня. Условие ее следующее: конь, начав движение из произвольной клетки, должен обойти всю доску, побывав в каждой клетке только один раз. В традиционной постановке задачи считается, что доска имеет размеры 8×8 клеток, но ее можно решать и для доски любых других размеров.

Задача о коне имеет долгую историю — около трех столетий. В XVIII веке ее решением занимались многие известные математики, однако особенно преуспел в этом Эйлер, посвятивший ей объемное сочинение. Он создал алгоритм, дающий возможность находить маршруты подбором за приемлемое время. Позже были открыты более эффективные алгоритмы: правило Вандсорфа, рамочный метод Мунка и Коллини, метод деления на четверти Полиньяка и Роже и пр. Однако полностью задача о коне не решена до сих пор. Пока никому не удалось найти не то что все возможные маршруты, но даже оценить их число. Достоверно известно лишь, что число маршрутов не превосходит числа сочетаний из 168 по 63 (приблизительно 10^{47}), но больше 30 млн. Столь значительное количество маршрутов дает право уверенно утверждать, что из какой бы клет-

ки конь ни начал движение, у него будет множество возможностей обойти всю доску, побывав в каждой клетке только один раз.

Мы решим задачу Эйлера двумя способами: прямым перебором и используя правило Вансдорфа. Однако, прежде чем мы приступим к реализации алгоритмов, необходимо провести небольшую подготовительную работу. В первую очередь нужно решить, как мы будем описывать шахматную доску. Очевидно, что ей нужно поставить в соответствие матрицу. Каждый элемент этой матрицы будет описывать клетку доски. Однако адресоваться элементы будут, естественно, иначе, чем клетки. Если в случае реальной доски столбцы адресуются буквами от a до h, а строки — числами от 1 до 8, причем отсчет начинается с нижнего левого угла, то в случае матрицы ячейки будут адресоваться индексами от 0 до 7, а отсчет начнется с верхнего левого угла. Это означает, что, например, клетке a3 будет соответствовать элемент матрицы с индексами i=5, j=0.

На то, что конь уже побывал в данной клетке, будет указывать значение соответствующего элемента описывающей доску матрицы: это номер хода, на котором конь находился в клетке. Если же конь еще ни разу не был в клетке, значением элемента будет нуль. Поэтому изначально в матрице доски должны быть одни нули. Создадим функцию, которая будет формировать подобные матрицы для досок произвольных размеров:

$$\text{chess\_field}(n) := \left| \begin{array}{l} \text{for } i \in 0..n-1 \\ \quad \text{for } j \in 0..n-1 \\ \quad\quad \text{field}_{i,j} \leftarrow 0 \\ \text{field} \end{array} \right. \qquad \text{chess\_field}(5) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Далее нужно придумать, как можно представить в форме, понятной языку программирования, правила, по которым ходит конь. В общем случае имеется восемь вариантов хода (рис. 4.6).

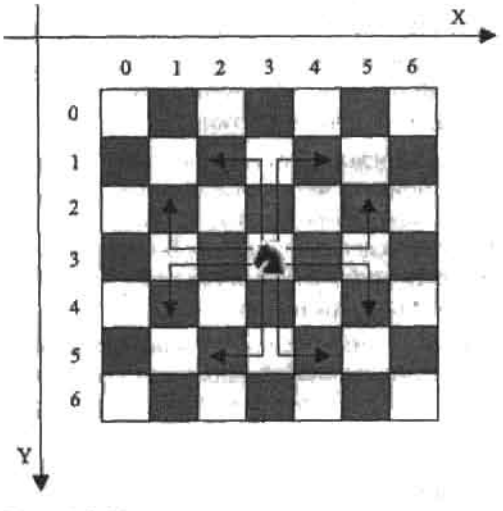


Рис. 4.6. Варианты хода коня

Во всех вариантах хода есть одно общее: конь смещается на две клетки по одному измерению и на одну клетку по другому. В зависимости от направления движения смещение будет иметь знак «+» или «–». Таким образом, каждый возможный ход можно описать парой чисел, показывающей, на сколько клеток переместится конь по горизонтали и вертикали. Чтобы узнать, в какой клетке окажется конь, сделав ход, нужно к индексам клетки, в которой он находится в данный момент, прибавить соответствующие числа из описывающей ход пары. Сами же пары удобно занести в векторы, которые в свою очередь являются элементами вектора:

$$step_0 := \begin{pmatrix} 1 \\ -2 \end{pmatrix} \quad step_1 := \begin{pmatrix} 2 \\ -1 \end{pmatrix} \quad step_2 := \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad step_3 := \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$step_4 := \begin{pmatrix} -1 \\ 2 \end{pmatrix} \quad step_5 := \begin{pmatrix} -2 \\ 1 \end{pmatrix} \quad step_6 := \begin{pmatrix} -2 \\ -1 \end{pmatrix} \quad step_7 := \begin{pmatrix} -1 \\ -2 \end{pmatrix}$$

Как бы мы решали на бумаге задачу Эйлера о коне, не зная никаких специальных правил и методик? Почти наверняка мы использовали бы метод перебора с возвратом (именно его применял Эйлер). Суть этого метода заключается в следующей последовательности действий.

1. Организуется иерархия шагов коня. Каждому варианту шага присваивается свой индекс.
2. То, куда будет сделан ход, зависит от того, какие варианты шага приведут коня в «пустую» клетку. Если все варианты возможны, используется шаг с наименьшим индексом. Если сделать этот шаг нельзя, осуществляется попытка сделать шаг с индексом, на единицу большим, — и так до тех пор, пока свободная клетка не будет найдена.
3. Если в результате очередного шага конь попал в клетку, из которой невозможно сделать шаг в клетку, в которой он не был ранее, маршрут признается тупиковым. При этом конь должен сделать один шаг назад и «попробовать» походить в другую клетку (последовательно используя варианты шага, расположенные в иерархии позже тупикового). Если это невозможно, делается еще один шаг возврата и попытка осуществляется снова — и так до тех пор, пока «пустая» клетка не будет найдена.
4. Если решение найдено не будет, то конь вернется в исходную клетку, опробовав все возможные маршруты. О том же, что искомый маршрут обнаружен, можно судить по тому, что на поле не останется ни одной непосещенной клетки.

Визуально алгоритм перебора с возвратом можно представить как последовательный обход графа (дерева) неизвестной структуры в поисках ветви максимальной длины. Пример подобного обхода для небольшого графа показан на рис. 4.7. Чтобы найти наиболее длинную ветвь из шести узлов, понадобилось сделать 17 шагов. В случае графа, соответствующего обходу конем шахматной доски, чтобы найти ветвь из 64 узлов, может понадобиться сделать миллионы или даже миллиарды шагов.

Каждый из возможных маршрутов коня может быть визуализирован как ветвь графа. Клеткам маршрута будут соответствовать узлы. Так как любой маршрут будет иметь часть, общую с другими маршрутами, ветви при движении сверху вниз сливаются, сходясь к одному узлу, соответствующему клетке начала движения. Чтобы найти ветвь графа требуемой длины, нужно организовать последовательный обход его ветвей слева направо или справа налево (см. рис. 4.7). Наиболее простой способ это сделать с помощью средств программирования связан с использованием рекурсии.

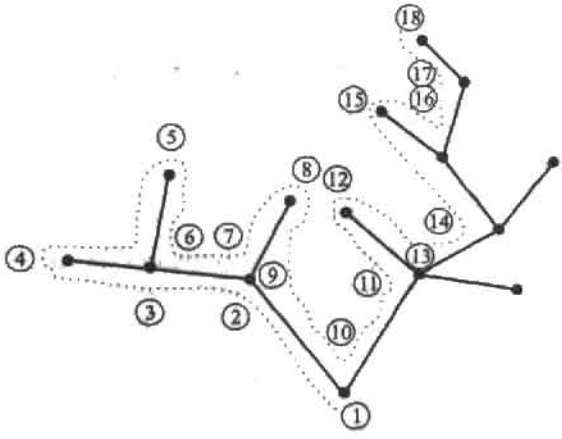


Рис. 4.7. Обход графа посредством последовательного перебора с возвратом

Алгоритм рекурсивно вызываемой функции, выполняющей обход графа маршрутов, будет несложен. Каждая ее активация будет соответствовать отдельному узлу графа. Основная цель активации — выполнить очередной шаг вверх к следующему узлу ветви. Для этого посредством цикла проверяется, имеется ли вариант хода, который приведет коня в непосещенную ранее клетку. Если такой обнаруживается, осуществляется рекурсивный вызов, создающий соответствующую этой клетке активацию. Текущая же активация ожидает результата дальнейшего исследования ветви, который будет возвращен в конце концов новой активацией. Если, просмотрев все варианты хода, код активации обнаружит, что возникла тупиковая ситуация, активация должна быть разрушена, а в точку соответствующего ей вызова, относящегося к предыдущей активации, должно быть возвращено информационное сообщение об этом. Получив это сообщение, нижележащая активация проверит, можно ли посредством хода с более низким приоритетом, чем осуществленный ранее, перейти в «пустую» клетку. Если такой вариант хода имеется, действия повторятся. Если же все варианты уже исчерпаны, активация разрушится, передав сообщение о тупике нижележащей активации. И так до тех пор, пока не будет найден нужный маршрут. Об успехе будет говорить то, что очередная активация окажется в стеке в качестве элемента под номером n^2, где n — размерность доски. При этом активация должна быть разрушена, а в точку вызова возвращено сообщение, говорящее об успешном завершении обхода. Данное сообщение должно передаваться сверху вниз от активации к активации, разрушая их. В качестве такого сообщения имеет смысл использовать матрицу с найденным маршрутом.

Приведенное выше описание алгоритма, который нам предстоит реализовать, скорее всего, покажется вам запутанным и малопонятным. Это неудивительно ввиду чрезвычайной абстрактности рекурсивных алгоритмов. Вернитесь к описанию алгоритма обхода после того, как мы создадим соответствующую ему функцию. При этом вы легко поймете то, что трудно понять сейчас.

Функцию, предназначенную для рекурсивного обхода графа маршрутов коня, мы назовем brain. Она будет принимать четыре параметра: индексы клетки, из которой должен быть сделан ход (i и j), матрицу с информацией о том, в каких клетках конь уже побывал (field), количество «заполненных» на момент вызова клеток (n). Чтобы запустить работу алгоритма, активировав brain, необходимо предварительно создать описывающую доску матрицу нужной размерности посредством функции chess_field, а затем заменить

в ней значение 0 элемента, соответствующего клетке начала движения, на 1. Исполнять роль такого «детонатора» у нас будет функция recursive_solver(n, i, j), где n — размерность доски, i и j — индексы клетки, из которой конь должен начать движение. Ее код:

$$\text{recursive\_solver}(n,i,j) := \left| \begin{array}{l} \text{field} \leftarrow \text{chess\_field}(n) \\ \text{field}_{i,j} \leftarrow 1 \\ \text{brain}(i,j,\text{field},1) \end{array} \right.$$

Первое действие, которое должна выполнить функция brain, это проверить, не закончил ли конь обход доски на прошлом шаге. На то, что это действительно так, будет указывать отсутствие нулей в описывающей поле матрице field. Наименьшим значением в ней будет 1. Выделить же минимальный элемент из матрицы можно, воспользовавшись функцией min. Если проверка покажет, что обход был успешно завершен, то текущая активация должна быть разрушена, а в точку вызова возвращена матрица field, хранящая найденный маршрут. Осуществить эти действия позволяет оператор return.

$$\text{brain}(i,j,\text{field},n) := \left| \text{return field if } \min(\text{field}) = 1 \right.$$

Если на поле есть еще непосещенные клетки, нужно проверить, можно ли сделать шаг в одну из них из клетки, в которой конь находится в данный момент. Для этого следует запустить цикл от 0 до 7, последовательно перебирая все варианты хода. Чтобы вычислить индексы клетки, в которую конь переместится, сделав некоторый ход, нужно к индексам данной клетки прибавить смещения по соответствующим направлениям (как вы помните, смещения для всех вариантов шага хранятся в векторах вложенного вектора step). Вычислив индексы, нужно проверить, корректны ли они. Индекс не может быть меньше нуля и больше или равен номеру последнего столбца (строки) матрицы поля. Если окажется, что значение индекса некорректно, необходимо сразу перейти к рассмотрению следующего варианта хода, для чего нужно задействовать оператор continue. Аналогично нужно поступить, если клетка, в которую перешел бы конь в результате хода, уже была посещена им ранее (об этом говорит то, что значение соответствующего элемента матрицы поля не равно 0).

$$\left| \begin{array}{l} \text{for } k \in 0..7 \\ \left| \begin{array}{l} \left[i1 \leftarrow i + \left(\text{step}_k\right)_0 \quad j1 \leftarrow j + \left(\text{step}_k\right)_1 \right] \\ \text{continue} \quad \text{if } \neg\left(0 \le i1 < \text{rows}(\text{field}) \wedge 0 \le j1 < \text{cols}(\text{field}) \wedge \text{field}_{i1,j1} = 0\right) \end{array} \right. \end{array} \right.$$

Обратите внимание, как был задан комплекс условий, выполнение которых необходимо, чтобы ход можно было осуществить.

- В Mathcad возможны двойные сравнения типа A≤x≤B. Это довольно необычно: в универсальных языках программирования сначала бы вычислилось условие A≤x, а полученный при этом результат сравнился затем с B. По этой причине в таких языках, как C++, двойные сравнения невозможны (их необходимо разбивать на отдельные операции сравнения, связанные посредством операторов логического И и ИЛИ). В Mathcad же можно задавать не только двойные, но и более сложные сравнения, содержащие сколь угодно много операций. Например:

$$2 < 3 < 4 < 5 < 6 = 1 \qquad 6 > 4 < 5 < 7 \le 7 \ge 7 = 1$$

- Посредством двойных сравнений мы проверяем, существует ли клетка, в которую переместился бы конь в результате хода. Ее индексы не должны быть отрицательными и превышать количество строк (столбцов) в матрице. Определить размерность матрицы можно, воспользовавшись функцией rows (количество рядков) или cols (количество столбцов).
- Отдельные условия объединяем в комплекс условий посредством операторов логического И «∧», так как для того, чтобы конь мог сделать ход, необходимо одновременное выполнение всех условий. Если бы было достаточно выполнения хотя бы одного условия, мы бы задействовали оператор логического ИЛИ «∨». Важной особенностью логических операторов является то, что они имеют более низкий приоритет, чем операторы сравнения (подобно тому, как операторы суммы и разности уступают по приоритету умножению и делению). Поэтому двойные сравнения не нужно брать в скобки, объединяя их посредством оператора И. На момент вычисления оператора И выражения справа и слева его уже будут вычислены в 0 или 1.
- Комплекс условий записан так, что он вычисляется в истину тогда, когда конь может сделать ход в клетку. Однако оператор continue должен быть задействован тогда, когда условия вычисляются в ложь. По этой причине нужно перед комплексом условий ввести оператор логического НЕ «¬», который преобразует ложь в истину, а истину в ложь.

Если все необходимые условия выполнятся, нужно сделать ход. Это будет выражаться в том, что значение соответствующего клетке элемента матрицы field должно быть изменено с 0 на номер данного хода. Определить этот номер очень просто, так как номер предыдущего хода передается функции brain в качестве четвертого параметра.

$$field_{i1,\,j1} \leftarrow n + 1$$

Сделав ход, нужно выполнить рекурсивный вызов функции brain. Результатом этого вызова будет или сообщение «Bad way», означающее, что данный путь тупиковый, или матрица с найденным маршрутом. Если маршрут будет найден, дальнейшую работу функции продолжать нет смысла. При этом активация должна быть разрушена, а в качестве результата ее работы возвращена матрица с маршрутом. Если же данный путь оказался тупиковым, необходимо сделать шаг назад, что будет выражаться в том, что значение заполненного последним элемента матрицы field должно быть заменено нулем.

```
res_field ← brain(i1, j1, field, n + 1)
return res_field  if  res_field ≠ "Bad way"
field_{i1, j1} ← 0
```

Важным и тонким моментом является то, как активации функции brain обмениваются информацией. Почему для того, чтобы следующая активация «знала», какие клетки уже были посещены конем, матрица field должна быть передана ей в качестве параметра? Почему для того, чтобы «спустить» матрицу с маршрутом от последней активации к первой, она должна последовательно возвращаться активациями, по мере их разрушения, как результат их работы? На это имеется две причины. Во-первых, функция не может изменить объект данных, который не относится непосредственно к ее коду. При обращении к внешней переменной из кода функции создается копия ее значения. И именно с копией работает код функции. Поэтому, даже если объект данных, хранимый во внешней переменной, подвергается модификациям по мере работы кода функции, после разрушения активации никаких изменений не останется, так как все операции проводились над копией. Именно поэтому матрицы field и res_field нельзя вынести

во внешние переменные. Во-вторых, функция, вызванная из кода другой функции, не видит ее локальных переменных. Поэтому нельзя создать матрицу field при вызове функции recursive_solver, а затем лишь модифицировать ее по мере роста рекурсивной цепи. Код функции может обращаться лишь к глобальным переменным.

Если все варианты шага будут просмотрены и не один из них не приведет к нахождению искомого маршрута, направление признается тупиковым. В этом случае функция brain должна возвратить строку «Bad way».

Функция brain готова. Ее полный код:

```
brain(i, j, field, n):= return field if min(field) = 1
                        for k ∈ 0..7
                          [i1 ← i + (step_k)_0   j1 ← j + (step_k)_1]
                          continue if ¬(0 ≤ i1 < rows(field) ∧ 0 ≤ j1 < cols(field) ∧ field_(i1,j1) = 0)
                          field_(i1,j1) ← n + 1
                          res_field ← brain(i1, j1, field, n + 1)
                          return res_field if res_field ≠ "Bad way"
                          field_(i1,j1) ← 0
                        "Bad way"
```

Как видите, функции brain соответствует довольно небольшой код. Это очень характерно для рекурсивных алгоритмов: их абстрактность и сложность для понимания компенсируется компактностью, доходящей порой до изящества. Именно поэтому рекурсивные алгоритмы столь популярны в программировании.

Проверим написанный код в работе. Мы увидим, что он весьма эффективен: он находит маршрут вне зависимости от того, какая клетка выбрана начальной. Однако, увы, за приемлемое время маршрут обнаруживается лишь для досок размером 6×6 клеток или более маленьких. Для стандартной же доски 8×8 клеток поиск будет вестись столь долго, что результата будет просто невозможно дождаться. Это связано с колоссальным количеством вариантов, которые необходимо перебрать алгоритму.

$$\text{recursive\_solver}(5,0,0)=\begin{pmatrix}1 & 16 & 21 & 10 & 25\\ 20 & 11 & 24 & 15 & 22\\ 17 & 2 & 19 & 6 & 9\\ 12 & 7 & 4 & 23 & 14\\ 3 & 18 & 13 & 8 & 5\end{pmatrix}\quad \text{recursive\_solver}(6,3,3)=\begin{pmatrix}31 & 34 & 29 & 20 & 11 & 36\\ 28 & 19 & 32 & 35 & 22 & 13\\ 33 & 30 & 21 & 12 & 5 & 10\\ 18 & 27 & 6 & 1 & 14 & 23\\ 7 & 2 & 25 & 16 & 9 & 4\\ 26 & 17 & 8 & 3 & 24 & 15\end{pmatrix}$$

Увы, но написанную программу невозможно оптимизировать так, чтобы она стала работать существенно быстрее. Не получится найти решение для доски 8×8 клеток, даже если программу перевести в нерекурсивную форму или реализовать ее посредством компилируемого языка вроде C++. Дело в том, что в основе нашей программы лежит

универсальный, но неэффективный алгоритм. Количественно эффективность можно оценить, зная функцию зависимости числа необходимых для нахождения решения шагов от входных параметров. Чаще всего точно определить такую функцию нельзя. Однако очень часто можно вывести ее общий вид. Исходя из него, алгоритмы можно условно разделить на линейные, полиномиальные и экспоненциальные (возможны и другие варианты, например «логарифмические» алгоритмы). Количество необходимых шагов, а следовательно, и время выполнения линейных алгоритмов пропорционально величине параметра. В случае полиномиальных алгоритмов $N=P^z$, где N — число шагов, P — величина параметра, z — показатель степени. Чем больше z, тем менее эффективен алгоритм. В случае экспоненциальных алгоритмов $N=Z^P$, где Z — некоторое число. Понятно, что время выполнения экспоненциальных алгоритмов увеличивается по мере возрастания параметра гораздо быстрее, чем время выполнения линейных и полиномиальных алгоритмов. Поэтому алгоритмы экспоненциального типа являются наименее эффективными. Созданный же нами алгоритм обхода шахматной доски посредством перебора с возвратом — экспоненциальный. Довольно просто оценить верхнюю границу для числа вариантов, которые придется перебрать компьютеру при поиске маршрута коня на доске размером n×n клеток:

$$N_{max}=8^{n^2}$$

Количество необходимых шагов алгоритма с увеличением размера доски возрастает чрезвычайно быстро. Этим объясняется то, почему для доски 6×6 клеток маршрут обнаруживается довольно быстро, а уже для доски 7×7 он ищется многие часы. Сделаем вывод. Задачу Эйлера мы решили, но лишь частично. Чтобы справиться с ней в случае больших досок, нужно использовать более эффективный алгоритм, чем перебор с возвратом. И такие алгоритмы за триста лет существования задачи Эйлера были найдены. Наиболее известным из них является алгоритм, основывающийся на правиле Вандорфа.

Вандорф сформулировал свое знаменитое правило в 1823 году в брошюре, посвященной его опыту решения задачи Эйлера о коне. Оно гласит: «На каждом ходу необходимо ставить фигуру в то поле, из которого можно совершить наименьшее количество ходов на еще непосещенные поля. Если же два или несколько полей равноценны, ход можно сделать в любое из них».

Алгоритм, построенный на основании правила Вандорфа, будет очень быстрым. Несложно догадаться, что он будет исполняться за полиномиальное время, а точнее, за время, пропорциональное n^2, где n — размер доски. Чтобы найти маршрут в случае доски 8×8, ему понадобится всего 64 шага. То есть маршрут будет определен за доли секунды.

Реализация метода Вандорфа не требует использования рекурсии, и поэтому соответствующий код более прост для написания и изучения. Поэтому мы будем комментировать его кратко. Функцию chess_field и вектор step, чтобы не выполнять лишнюю работу, просто скопируем из рекурсивного алгоритма.

Итак, приступим. Первое, что мы сделаем, это создадим функцию, которая будет оценивать, сколько можно осуществить ходов из данной клетки. Называться она будет checker, а в качестве параметров принимать индексы клетки, которую нужно проверить на количество возможных из нее ходов, и матрицу с информацией о том, какие клетки уже были посещены (аналогична матрице field созданного ранее рекурсивного алгоритма). Алгоритм функции checker будет следующим.

1. Создаем счетчик (переменная free_position), в котором будет храниться число возможных ходов. Изначально он должен быть равен 0.

2. Запускаем цикл и перебираем все восемь вариантов хода.
3. На каждой итерации цикла производим следующие действия. Сначала вычисляем индексы клетки, в которую переместится конь, сделав ход. Затем проверяем, существует ли данная клетка и является ли она «пустой». Если какое-то из условий не выполняется, необходимо перейти к следующей итерации (посредством оператора continue). Если условия выполнятся, нужно увеличить счетчик возможных ходов на 1.
4. По завершении работы цикла счетчик должен быть возвращен программой как результат ее работы.

В коде функция checker будет иметь следующий вид:

```
checker(i, j, field) := | free_position ← 0
                        | for k ∈ 0..7
                        |   [ i1 ← i + (step_k)_0   j1 ← j + (step_k)_1 ]
                        |   continue   on error field_{i1, j1}
                        |   free_position ← free_position + 1  if field_{i1, j1} = 0
                        | free_position
```

Обратите внимание, как в коде функции checker учитывается, что вычисленные индексы могут соответствовать не существующей реально клетке. Для этого используется оператор on error. Как вы помните, выражение в левом маркере данного оператора проделывается тогда, когда выполнение выражения в правом маркере приводит к возникновению ошибки. Если индексы не определяют конкретного элемента field, система зарегистрирует ошибку. При этом будет активизирован оператор continue и цикл перейдет к следующей итерации.

Далее мы должны создать функцию, которая будет определять, в какую из свободных клеток нужно сделать ход с учетом правила Вансдорфа. Называться она будет optim_step. Параметры у нее будут следующие: индексы клетки, в которой находится конь, и матрица с информацией о предыдущих ходах фигуры. Алгоритм функции optim_step несложен.

1. Запускаем цикл, перебирающий все варианты хода.
2. Перейдя к очередному варианту хода, вычисляем индексы и проверяем, соответствуют ли они существующей и непосещенной еще клетке. Если необходимые условия не выполняются, переходим сразу на следующую итерацию.
3. Если клетка существует и свободна, вычисляем ее «рейтинг» посредством созданной ранее функции checker. Далее сравниваем полученное значение со значением переменной min_res, в которой хранится наименьший рейтинг из определенных на предыдущих шагах (изначально min_res необходимо присвоить значение, заведомо превосходящее наибольший из возможных рейтингов, то есть 8). Если рейтинг клетки оказывается меньше значения min_res, данную переменную следует переопределить, присвоив ей его значение. Также при этом в переменную best_step необходимо занести индекс вектора, описывающего данный ход.
4. В качестве результата работы функции нужно возвратить переменную best_step, хранящую индекс вектора вложенного вектора step, который соответствует ходу в клетку, из которой можно сделать наименьшее число ходов.

На языке программирования функция optim_step может быть реализована следующим образом:

```
optim_step(i, j, field) := | min_res ← 10
                           | for k ∈ 0..7
                           |     [ i1 ← i + (step_k)_0   j1 ← j + (step_k)_1 ]
                           |     continue  on error field_{i1, j1}
                           |     res ← checker(i1, j1, field)  if field_{i1, j1} = 0
                           |     continue  otherwise
                           |     (best_step ← k  min_res ← res)  if res < min_res
                           | best_step
```

Написанные выше функции лишь вспомогательные. Непосредственно маршрут будет определять функция solver. У нее будет три параметра: размерность матрицы, которую нужно обойти, и индексы клетки, с которой необходимо начать движение. Алгоритм функции solver будет следующим.

1. Сначала, используя созданную еще для рекурсивного алгоритма функцию chess_field, создаем описывающую поле матрицу field. Далее переопределяем значение элемента field, соответствующего клетке, с которой начинается обход, с 0 на 1.
2. Запускаем цикл из n^2-1 итераций, где n — размерность матрицы (не n^2, так как первый ход делается вручную).
3. На каждой итерации осуществляем один шаг вперед. Для этого с помощью функции optim_step определяем, в каком векторе вложенного вектора step находятся смещения, описывающие нужный ход. Зная смещения, легко найти индексы клетки. Ход в нее должен заключаться в том, что значение соответствующего ей элемента матрицы field должно переопределяться с 0 на номер данного хода.
4. В качестве результата работы функции возвращаем матрицу field, содержащую маршрут обхода поля.

Код функции solver:

```
solver(n, i_first, j_first) := | field ← chess_field(n)
                               | field_{i_first, j_first} ← 1
                               | (i ← i_first  j ← j_first)
                               | for k ∈ 2..n^2
                               |     next_step ← optim_step(i, j, field)
                               |     [ i ← i + (step_{next_step})_0   j ← j + (step_{next_step})_1 ]
                               |     field_{i, j} ← k
                               | field
```

Готово. Проверяем эффективность созданной программы для матрицы 8×8 в случае использования в качестве начальных двух разных клеток:

$$\text{solver}(8,4,0)=\begin{pmatrix}55&64&13&32&47&28&11&30\\14&33&54&57&12&31&48&27\\63&56&61&46&53&50&29&10\\34&15&58&51&60&43&26&49\\1&62&45&42&19&52&9&40\\16&35&20&59&44&41&6&25\\21&2&37&18&23&4&39&8\\36&17&22&3&38&7&24&5\end{pmatrix}\quad \text{solver}(8,3,3)=\begin{pmatrix}27&30&17&34&45&64&15&36\\18&33&28&57&16&35&44&61\\29&26&31&46&63&60&37&14\\32&19&58&1&56&47&62&43\\25&2&55&52&59&42&13&38\\20&51&22&41&48&53&10&7\\3&24&49&54&5&8&39&12\\50&21&4&23&40&11&6&9\end{pmatrix}$$

Правило Вансдорфа доказало, что его не зря помнят уже почти 200 лет. Решение было найдено верно для обеих клеток, причем сделано это было за мгновение. Аналогично можно найти маршрут обхода и очень больших полей, например 100×100 клеток. Времени на это потребуется также довольно немного.

Правило Вансдорфа очень часто дает возможность завершить маршрут обхода, даже если первая часть пути была найдена не на его основании. Именно поэтому начинать обход можно с любой клетки поля. Если же действовать строго исходя из правила Вансдорфа, маршрут должен начинаться в одной из четырех угловых клеток (так как из них можно сделать наименьшее количество ходов).

Правило Вансдорфа эффективно в подавляющем большинстве случаев. Но не всегда. Иногда оно дает сбои. Дело в том, что вторая часть правила неверна. Если есть несколько клеток с одинаковым рейтингом, то имеет значение, в какую из них походить. Неверно выбранный путь приведет к тому, что решение не будет найдено. Впрочем, такой вариант на практике встречается весьма редко. Но что делать, если правило Вансдорфа все же даст сбой? В этом случае необходимо поменять последовательность векторов ходов в векторе step. Смысл этого действия заключается в том, что при наличии клеток с одинаковым рейтингом ход делается именно в ту, которая соответствует верному маршруту. Всего вариантов организации ходов в иерархию 8!=40 320. Если запастись терпением, всегда можно найти такую последовательность ходов, при которой правило Вансдорфа будет работать безупречно.

Глава 5. Комплексные числа

Одним из крупных достоинств системы Mathcad является наличие полной поддержки работы с комплексными числами в их традиционной форме представления (калькуляторы и электронные таблицы обычно рассматривают комплексные числа как пару действительных). Более того, комплексные числа — это основной тип чисел в Mathcad. Большинство математических операций программа по умолчанию проводит именно над комплексными числами. К примеру, при аналитическом и численном решении уравнений определяются как действительные, так и мнимые корни. Функция комплексного переменного может быть легко проинтегрирована. Большинство встроенных функций могут принимать как действительные, так и комплексные аргументы. Многие встроенные функции сами могут возвращать комплексные значения при определенных величинах аргументов. Например, логарифм отрицательного числа в Mathcad существует и равен соответствующему комплексному числу.

Пример 5.1. Комплексные числа в расчетах различных типов

Mathcad позволяет найти квадратный корень и логарифм отрицательного числа:

$$\sqrt{-2} = 1.414i \qquad \ln(-10) = 2.303 + 3.142i$$

При аналитическом и численном решении уравнений могут быть получены комплексные корни.

Аналитическое решение:

$$x^2 + 1 = 0 \text{ solve}, x \rightarrow \begin{pmatrix} i \\ -i \end{pmatrix}$$

Численное решение:

$$x := 2 + i \qquad \text{root}\left(x^2 + 1, x\right) = i$$

При проведении математических операций следует всегда помнить, что для большинства из них имеют смысл и комплексные операнды. Поэтому, получив, например, мнимое значение интеграла, не удивляйтесь, а тщательно проверьте правильность определения пределов.

$$\int_{-e}^{e} \frac{\ln(x)}{\sqrt{x}}\,dx \text{ simplify } \rightarrow -2\cdot e^{\frac{1}{2}} + 2\cdot i\cdot e^{\frac{1}{2}} + 2\cdot e^{\frac{1}{2}}\cdot\pi$$

$$\int_{-e}^{e} \frac{\ln(x)}{\sqrt{x}}\,dx = 7.062 + 3.297i$$

Предельно внимательным нужно быть и в случае некоторых арифметических операций. Так, абсолютно идентичные с алгебраической точки зрения записи кубического корня из отрицательного числа, в виде степени 1/3 и специального значка, в Mathcad различаются принципиально: в первом случае число рассматривается как комплексное, во втором — как действительное. В результате для их подсчета реализуются различные алгоритмы, что приводит к получению несхожих ответов.

$$\sqrt[3]{(-3)} = -1.442 \qquad (-3)^{\frac{1}{3}} = 0.721 - 1.249i$$

Большим достоинством Mathcad является то, что комплексное число может быть не только непосредственно получено при вычислениях, но и быть использовано в них в качестве переменной или параметра. Эффективность расчета при этом совершенно не снизится.

$$\ln(3\cdot i + 1) = 1.151 + 1.249i \qquad \sin\left(i + 5\cdot\frac{\pi}{8}\right) = 1.426 - 0.45i$$

$$\int_{-1+i}^{1+i} e^{-x}\,dx \rightarrow -e^{-1-i} + e^{1-i}$$

Введение комплексного числа в Mathcad имеет некоторые особенности.

- Проще всего можно ввести мнимую единицу, использовав соответствующую кнопку панели Calculator (Калькулятор).
- Чтобы задать мнимую единицу с клавиатуры, нельзя просто нажать клавишу I, так как при этом она будет воспринята системой как неизвестная переменная. Для корректного задания мнимой единицы сначала введите цифру 1 и лишь затем i (но не 1·i).
- В западной научной литературе иногда принято обозначать мнимую единицу не как i, а как j. В Mathcad имеется возможность замены символа мнимой единицы с помощью параметра Imaginary Value (Мнимая единица) на вкладке Display Options (Параметры отображения) окна Result Format (Формат результата), вызываемого из меню Format (Формат). Однако изменения, произведенные описанным параметром, коснутся лишь выражений ответа. Для задания же исходных данных с мнимым числом в виде буквы j нужно просто непосредственно ввести его с клавиатуры.

Пример 5.2. Отображение мнимой единицы различными символами

$$e^{j} = 0.54 + 0.841j \qquad e^{i} = 0.54 + 0.841i$$

5.1. Основные характеристики комплексных чисел

Имеется несколько величин, позволяющих охарактеризовать комплексное число z=a+b·i. Для их определения в Mathcad используются специальные функции и операторы. Кратко опишем, как можно найти в изучаемой программе основные характеристики комплексного числа (рис. 5.1).

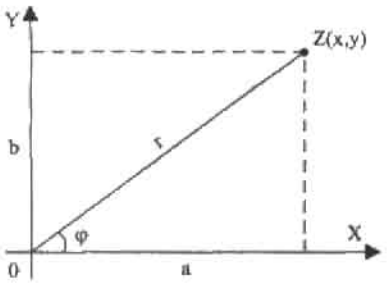


Рис. 5.1. Геометрическая интерпретация комплексного числа. Число представляется точкой, лежащей в плоскости XOY. Длина вектора r, соединяющего начало координат и точку числа, равна модулю этого числа; проекция этого вектора на ось X соответствует действительной части числа a, проекция на ось Y — мнимой части b; наклон вектора к X определяет аргумент числа j

- Действительная часть a. Для ее определения служит функция Re(z).
- Мнимая часть b. Находится с использованием функции Im(z).
- Модуль комплексного числа |z|. Ему соответствует длина вектора, соединяющего начало координат и точку числа на комплексной плоскости. Значение модуля комплексного числа позволяет определить оператор модуля панели Calculator (Калькулятор).
- Сопряженное число z^*. Это такое комплексное число, при умножении на которое z получается действительное число. Несложно догадаться, что $z^*=a-b\cdot i$, а также что $z\cdot z^*=|z|^2$. Найти z^* в Mathcad позволяет особый оператор комплексного сопряжения. Ввести данный оператор можно с помощью сочетания клавиш Shift+«'». Обязательное условие при этом — курсор должен находиться на тексте преобразуемого выражения, иначе будет вставлена текстовая область, вводимая тем же сочетанием. Кстати, оператор сопряженного числа по-своему уникален: это единственный из операторов в Mathcad, который не был вынесен на одну из панелей семейства Math (Математические).
- Аргумент комплексного числа. Представляет собой угол наклона φ вектора, соединяющего в комплексной плоскости начало координат и точку числа. Численно равен арктангенсу отношения мнимой части числа к действительной части: arg(z)=arctg(b/a). Для определения аргумента комплексного числа в Mathcad служит функция arg(z). Строго говоря, с учетом периодичности, аргумент комплексного числа имеет бесконечное множество значений, определяемых соотношением $\varphi=\arg(z)+2\cdot p\cdot k$, где k — любое целое число.

Все описанные выше функции и операторы работают как в режиме аналитических, так и численных расчетов.

Пример 5.3. Расчет основных характеристик комплексного числа

Присваиваем число переменной и пересчитываем его в форму z=a+b·i:

$$z := \frac{1+i}{1-i} + \frac{3}{1+2\cdot i} + 5 \qquad z \to \frac{28}{5} - \frac{1}{5}\cdot i$$

Находим действительную и мнимую части числа, его аргумент, модуль, а также сопряженное число:

$$\mathrm{Im}(z) \to \frac{-1}{5} \qquad \mathrm{Re}(z) \to \frac{28}{5}$$

$$\arg(z) \to -\mathrm{atan}\left(\frac{1}{28}\right) \qquad \arg(z) = -0.036 \qquad |z| \to \frac{1}{5}\cdot 785^{\frac{1}{2}} \qquad |z| = 5.604$$

$$\overline{z} \to \frac{28}{5} + \frac{1}{5}\cdot i \qquad \sqrt{z\cdot\overline{z}} \to \frac{1}{5}\cdot 785^{\frac{1}{2}} \qquad \sqrt{z\cdot\overline{z}} = 5.604$$

5.2. Формы представления комплексных чисел

Известны три формы представления комплексных чисел.

- Алгебраическая форма. Наиболее простая и традиционная: z=a+b·i.
- Тригонометрическая форма. Число в данном случае определяется через соотношение z=r·(cos(φ)+i·sin(φ)), где r — модуль комплексного числа, φ — его аргумент.
- Экспоненциальная форма: $z=r\cdot e^{i\cdot\varphi}$.

Mathcad умеет полноценно работать только с комплексными числами в алгебраической форме. Именно в ней чаще всего представляются результаты аналитических расчетов, и только в ней — численных. В тригонометрическую форму программа переводит лишь числа, представленные в экспоненциальной форме и упрощаемые с помощью оператора complex панели Symbolic (Символьные). И нет такой операции, в результате которой было бы получено комплексное число в экспоненциальной форме (если не считать операций вроде символьного интегрирования e^x от –i до i).

Что же делать, если результат необходимо представить именно в тригонометрической или экспоненциальной форме? Тут имеется два пути. Во-первых, вы можете составить нужное выражение самостоятельно, подсчитав модуль и аргумент комплексного числа. Во-вторых, можно создать функцию, которая будет записывать строку с ответом. У этого пути есть недостатки. Так, полученный ответ нельзя будет использовать в дальнейших расчетах, а также он будет «испорчен» скобками. Кроме того, подобная функция не может быть просчитана аналитически. Однако ничего лучшего для автоматизации преобразования комплексных чисел из одной формы в другую в Mathcad сделать нельзя. В методических целях продемонстрируем оба подхода к переводу комплексных чисел из алгебраической формы в тригонометрическую и экспоненциальную.

Пример 5.4. Представить число m=1+i в тригонометрической и экспоненциальной форме

Чтобы решить задачу первым способом, находим модуль числа и его аргумент, а затем подставляем их в формулы:

$$m := 1 + i$$

$$\arg(m) \to \frac{1}{4}\cdot\pi \qquad |m| \to 2^{\frac{1}{2}}$$

Тригонометрическая форма: $\sqrt{2}\cdot\left(\cos\left(\frac{\pi}{4}\right) + i\cdot\sin\left(\frac{\pi}{4}\right)\right)$

Экспоненциальная форма: $\sqrt{2}\cdot e^{i\cdot\frac{\pi}{4}}$

Выполняем проверку (система автоматически пересчитывает число из любого представления в алгебраическое):

$$\sqrt{2}\cdot\left(\cos\left(\frac{\pi}{4}\right) + i\cdot\sin\left(\frac{\pi}{4}\right)\right) = 1 + i \qquad \sqrt{2}\cdot e^{i\cdot\frac{\pi}{4}} = 1 + i$$

Второй способ заключается в том, что формируется строка с ответом. Для этого используется функция concat, объединяющая подстроки в одну строку. Для перевода чисел в строки применяем функцию num2str.

```
trigC(z):=concat(num2str(|z|),"*(cos(" ,num2str(arg(z)),")+i*sin(" ,num2str(arg(z)),"))" )

expC(z) := concat(num2str(|z|),"*(e^i*" ,num2str(arg(z)),")" )

trigC(1+i) = "1,4142135623731*(cos(0,785398163397448)+i*sin(0,785398163397448)

expC(1 + i) = "1,4142135623731*(e^i*0,785398163397448)"
```

Довольно часто комплексные числа получаются в форме, в которой действительная и мнимая части не разделены. Чтобы привести такое число к стандартному виду алгебраического представления z=a+b·i, следует задействовать символьный оператор complex. Также данный оператор осуществляет перевод числа из экспоненциальной формы в тригонометрическую.

Пример 5.5. Использование оператора complex

$$\frac{5 + i}{2 + 3\cdot i}\ \text{complex} \to 1 - i \qquad 5\cdot e^{4\cdot i}\ \text{complex} \to 5\cdot\cos(4) + 5\cdot i\cdot\sin(4)$$

5.3. Операции над комплексными числами

Такие арифметические операции, как сложение, вычисление разности, умножение, деление, возведение в степень над комплексными числами в системе Mathcad можно

осуществлять точно так же, как над действительными числами. Причем расчет может быть проведен как аналитически, так и численно. Сложнее дело обстоит с извлечением из комплексного числа корня. Ввиду нетривиальности данной проблемы рассмотрим ее более подробно.

Чему равен квадратный корень из 4? Большинство читателей моментально ответят: 2. Тот же результат выдаст и Mathcad, причем независимо от формы записи корня:

$$4^{\frac{1}{2}} = 2 \qquad \sqrt{4} = 2$$

Поставим вопрос по-другому: чему равняются корни уравнения $x^2=4$. Даже троечник, подумав, сообразит, что у данного уравнения два корня: +2 и −2. Но почему корень из числа только один, а у корня из переменной два значения?

Строго говоря, корню n-й степени из любого числа будет удовлетворять n значений. Однако n−2 из них для четной степени и n−1 для нечетной будут комплексными. В случае четной степени два значения, равных по модулю, но обратных по знаку, будут действительными. Для нечетной степени будет только одно действительное значение. При извлечении же корня работают следующие ограничения: результат должен быть действительным и иметь такой же знак, как исходное число. Очевидно, что этим ограничениям будет удовлетворять только одно значение из n. При извлечении корня из 4 таким значением будет 2.

Если число рассматривается, как комплексное, описанные выше ограничения не действуют. Извлекая из такого числа корень n-й степени, нужно получить все n значений. Как это сделать в Mathcad? Тут имеется два пути.

- Пусть комплексное число, из которого нужно извлечь корень n-й степени, равно z. Обозначим результат данной операции переменной x. Очевидно, что при возведении x в степень n мы получим z:

  $$x^n = z$$

  Аналитически решив данное уравнение с помощью оператора solve (Решить) панели Symbolic (Символьные), мы получим n искомых корней z.

- Несложно догадаться, что значения корня n-й степени из комплексного числа z будут лежать на окружности с центром в начале координат и радиусом $|z|^{1/n}$, деля эту окружность на равные части. Из этого факта можно получить следующую формулу, использующуюся для извлечения корня из комплексного числа, записанного в тригонометрической форме:

  $$\sqrt[n]{r\cdot(\cos(\phi)+i\cdot\sin(\phi))} = \sqrt[n]{r}\cdot\left(\cos\left(\frac{\phi+2k\cdot\pi}{n}\right)+i\cdot\sin\left(\frac{\phi+2k\cdot\pi}{n}\right)\right)$$

  Здесь r — модуль числа, ϕ — его аргумент, $k=0, 1, 2, \ldots, n-1$.

  Применяя данную формулу, можно определять корни комплексного числа тогда, когда в отчете должен быть виден механизм расчета. В остальных случаях удобнее применять первый способ.

Пример 5.6. Операции над комплексными числами

Элементарные операции (сложение, вычитание, произведение, деление, возведение в степень).

$$z1 := i + 1 \qquad z2 := 3 - 12\cdot i$$

$z1 + z2 = 4 - 11i$ $z1 - z2 = -2 + 13i$ $z1 \cdot z2 = 15 - 9i$ $\frac{z1}{z2} \to \frac{-1}{17} + \frac{5}{51}\cdot i$

$z1^3 = -2 + 2i$ $z2^7 \to -43842789 - 6359796\cdot i$

Извлечение корня. Найдем корни третьей степени из 1 обонми описанными способами.

Способ первый. Аналитически решаем уравнение:

$$x^3 = 1 \text{ solve}, x \to \begin{pmatrix} 1 \\ \frac{-1}{2} + \frac{1}{2}\cdot i\cdot 3^{\frac{1}{2}} \\ \frac{-1}{2} - \frac{1}{2}\cdot i\cdot 3^{\frac{1}{2}} \end{pmatrix}$$

Способ второй. Запускаем цикл по k от 0 до n–1 с помощью ранжированной переменной и по приведенной выше формуле подсчитываем корни, записывая их в вектор.

$n := 3$ $z := 1$ $k := 0 .. n - 1$

$$\theta_k := \sqrt[n]{|z|}\cdot\left[\cos\left(\frac{\arg(z) + 2\cdot k\cdot\pi}{n}\right) + i\cdot\left(\sin\left(\frac{\arg(z) + 2\cdot k\cdot\pi}{n}\right)\right)\right]$$

$$\theta^T \to \begin{pmatrix} 1 & \frac{-1}{2} + \frac{1}{2}\cdot i\cdot 3^{\frac{1}{2}} & \frac{-1}{2} - \frac{1}{2}\cdot i\cdot 3^{\frac{1}{2}} \end{pmatrix}$$

Проверяем, правильно ли были подсчитаны значения корней:

$\left(\frac{-1}{2} + \frac{1}{2}\cdot i\cdot\sqrt{3}\right)^3$ expand $\to 1$ $\left(\frac{-1}{2} - \frac{1}{2}\cdot i\cdot\sqrt{3}\right)^3$ expand $\to 1$

Глава 6. Графики

Очень трудно представить научные доклады, статьи или диссертации, в которых не использовалась бы графическая форма представления данных — кривые, диаграммы, гистограммы, поверхности, векторные поля. И причина этому вполне очевидна: ведь гораздо проще сделать нужные выводы о существовании, например, локального экстремума функции двух переменных, просто взглянув на соответствующую ей поверхность, чем анализировать матрицу из тысяч или десятков тысяч элементов. Однако если начертить схематично кривую способен даже школьник, то построить вручную поверхность сложной, несимметричной функции могут лишь люди маниакального трудолюбия и недюжинных художественных способностей. Выполнить же такую работу с помощью компьютера можно очень просто. Поэтому использование Mathcad не только значительно облегчает расчетную и оформительскую работу, но и открывает широкие, недоступные еще лет 10 назад, возможности в области визуализации данных.

Построение разнообразных графиков — одна из самых сильных сторон системы Mathcad. Особенно впечатляют возможности художественного оформления трехмерных объектов — пожалуй, в этом вопросе Mathsoft однозначно превзошла всех своих конкурентов. А так как задание графиков — тема очень важная, обширная и интересная, то вполне оправданным будет рассмотреть ее в качестве отдельной главы.

Данная глава разделена на три части, исходя из рассматриваемых вопросов: в первой мы поговорим об особенностях задания графиков функций одной переменной — в декартовой и полярной системах координат, во второй — о построении и форматировании поверхностей, в третьей части будет рассмотрено создание анимации в Mathcad.

Все основные типы графиков и инструменты работы с ними расположены на рабочей панели Graph (Графические) семейства Math (Математические). Здесь (рис. 6.1) вы можете найти ссылки на семь типов графиков.

Рис. 6.1. Панель Graph (Графические)

- График кривой в двумерной декартовой системе координат (X-Y Plot).
- График кривой в полярной системе координат (Polar Plot).
- Поверхность (Surface).
- Контурный график (Contour Plot).
- Столбчатая трехмерная (3D) диаграмма (3D Bar Plot).
- Точечный трехмерный (3D) график (3D Scatter Plot).
- Векторное поле (Vector Field).

Всего же типов графиков в Mathcad — восемь, просто один из видов отображения трехмерной зависимости — так называемая «кусочечная» поверхность (Patch Plot) — не был вынесен на панель Graph (Графические). О том, как построить такой график, мы поговорим в разделе, посвященном форматированию трехмерных изображений.

Аналогичный панели Graph (Графические) список всех типов графиков Mathcad расположен в одноименном ей подменю меню Insert (Вставить).

Деление графиков на восемь типов очень условно, поскольку, изменяя различные параметры и установки, можно построить и не имеющую аналогов на панели или в меню Graph (Графические) зависимость. Так, например, изменяя тип отображения ряда данных в двумерной декартовой системе координат, можно построить не только тривиальную кривую, но и гистограмму, ступенчатый график и график с отложенными ошибками. Аналогично существуют очень широкие возможности форматирования и в трехмерной системе координат.

Таким образом, можно смело утверждать, что типов графиков в Mathcad реально намного больше, чем восемь. В том, как суметь сориентироваться в этом многообразии и построить максимально наглядную кривую или поверхность, мы попробуем разобраться в данной главе.

6.1. Двумерные графики

В данном разделе мы разберем особенности создания и редактирования изображений в декартовых системах координат, а также кратко затронем вопрос об особенностях задания полярных графиков. Кроме того, мы поговорим об использовании инструментов исследования двумерных кривых (трассировке и масштабировании).

6.1.1. Задание X-Y-зависимостей в декартовой системе координат

В Mathcad существует несколько способов задания кривых в декартовой системе координат, однако первый шаг для всех будет один и тот же. Этим первым шагом является введение специальной заготовки для будущего графика — так называемой графической области. Ввести графическую область как для декартового, так и для любого другого графика можно либо с панели Graph (Графические), либо командой одноименного подменю меню Insert (Вставка). Также графические области имеют свои сочетания клавиш. Так, заготовку для декартовой двумерной системы координат можно ввести, нажав Shift+2.

Графическая область представляет собой две вложенные рамки. Во внутренней отображаются непосредственно кривые зависимостей. Пространство между рамками служит для визуализации разного рода служебной информации. Графическую область можно

увеличивать и уменьшать с помощью специальных маркеров, расположенных на ее внешней рамке. Перемещать по документу и удалять графические области можно точно так же, как простые формулы.

Окно форматирования вида графической области — Properties (Свойства) — также полностью совпадает с аналогичным окном для формул. Открыть его можно с помощью одноименной команды контекстного меню графика (вызывается щелчком правой кнопкой мыши на графической области).

В окне Properties (Свойства) для подавляющего большинства пользователей объективно могут быть полезны два параметра, расположенных на вкладке Display.

- Highlight Region (Цветная область). Установив этот флажок, вы сможете на палитре Choose Color (Выбор цвета) определить наиболее подходящий цвет заливки для вашей графической области.
- Show Border (Показать границу). Параметр отвечает за отображение внешней границы графической области. По умолчанию она не визуализируется.

После того как графическая область будет введена, в общем случае требуется задать два соразмерных вектора, определяющих значения координат точек. Сделать это можно различными способами.

Наиболее простым и часто используемым методом задания координатной сетки является так называемый быстрый метод. При его применении пользователь задает только имя переменной и вид функции, а шкалы осей и величину шага между узловыми точками автоматически определяет система. Чтобы построить кривую функции по быстрому методу, выполните следующую последовательность действий.

1. Введите графическую область.
2. В специальном маркере, расположенном в центре под внутренней рамкой графической области, задайте имя переменной.
3. В центральный маркер, расположенный слева от внутренней рамки, введите функцию или имя функции (если функцию определить раньше переменной, то работа даже упрощается, так как переменная будет задана автоматически).

К недостаткам рассматриваемого метода относится прежде всего то, что область изменения переменной для всех функций определяется одна и та же: от –10 до 10. В большинстве же случаев такие пределы будут неприемлемы по целому ряду причин. Например, если мы хотим определить местоположение точек локального экстремума функции:

$$f(x) := x^5 - x^3 + x^2 - 3x + 12$$

и используем для выполнения этой работы быстрый способ построения графика, то системой будет прочерчена кривая, изображенная на рис. 6.2.

Никакой полезной информации из данного графика почерпнуть не удастся, поскольку амплитуда экстремумов столь мала по сравнению с изменением величины функции на промежутке, что они становятся просто незаметными.

Чтобы справиться с возникшими трудностями, нужно просто уменьшить интервал изменения либо переменной, либо функции. Для этого выделите графическую область щелчком левой кнопкой мыши. При этом визуализируются все элементы, которые до этого были скрыты. Непосредственно под крайними значениями (для оси X) или слева от них (для оси Y) появятся цифры, отражающие максимальные и минимальные величины координат узловых точек графика. Чтобы изменить их значения, просто удалите (точно так же, как при редактировании формул) старые величины и введите но-

вые. Изменения пределов по оси X вызывает автоматический пересчет крайних значений по Y. Однако если вы переопределите область по оси Y, то область изменения переменной останется старой.

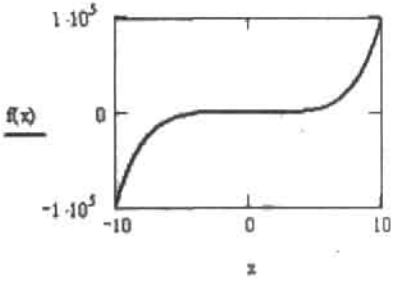


Рис. 6.2. График с неприемлемо большими пределами построения

На практике же, как правило, приходится настраивать пределы сразу по обеим осям. Это связано с тем, что хорошо подобрать интервал по оси значений функции системе удается далеко не всегда. Исходя из вышесказанного, попробуем произвести настройку вида графика рассматриваемой функции, изменяя диапазон как по оси X, так и по оси Y. Потратив немного времени на подбор наиболее удачных параметров, получим рис. 6.3.

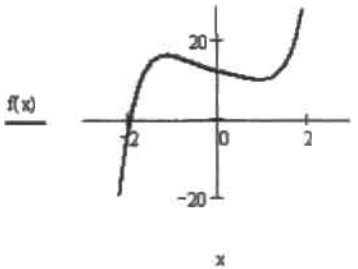


Рис. 6.3. Конечный вариант графика для задачи на экстремум

В ряде случаев (например, если в графике приходится очень часто менять диапазоны по осям) гораздо удобнее задать векторы данных самостоятельно. Сделать это можно с помощью оператора ранжированной переменной (вводится он либо с панели Matrix (Матричные), либо с помощью клавиши «;»).

Чтобы задать вектор значений переменной с помощью оператора Range Variable (Ранжированная переменная), выполните следующую последовательность действий.

1. Введите имя переменной вместе с оператором присваивания.
2. Задайте левую границу интервала построения и поставьте запятую.
3. Введите оператор ранжированной переменной.
4. В левом маркере введенного оператора задайте вторую точку на промежутке (тем самым вы определите шаг).
5. В правый маркер оператора ранжированной переменной введите значение правой границы на интервале.

В результате переменная и функция будут заданы в виде двух соразмерных векторов, по которым будет построен график (рис. 6.4).

$$y(x) := e^{x}\cos(12x) \qquad x := 0, 0.001.. 5$$

x =	y(x) =
0	1
$1 \cdot 10^{-3}$	1.001
$2 \cdot 10^{-3}$	1.002
$3 \cdot 10^{-3}$	1.002
$4 \cdot 10^{-3}$	1.003
$5 \cdot 10^{-3}$	1.003
$6 \cdot 10^{-3}$	1.003
$7 \cdot 10^{-3}$	1.003
$8 \cdot 10^{-3}$	1.003

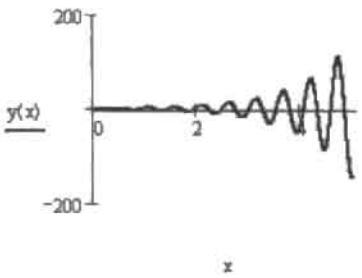


Рис. 6.4. Построение графика с использованием ранжированной переменной

Работа непосредственно с векторами значений может быть безальтернативным способом задания графика в целом ряде задач того же программирования. Однако при этом могут возникнуть проблемы, связанные с недостаточно малой величиной шага. Так, например, в случае функций, имеющих точки разрыва, вполне возможна такая ситуация, что две крайние узловые точки будут соединены линией, в результате чего вид графика будет искажен (рис. 6.5).

$$y(x) := \frac{x}{1 + x} \qquad x := -3, -2.99.. 2$$

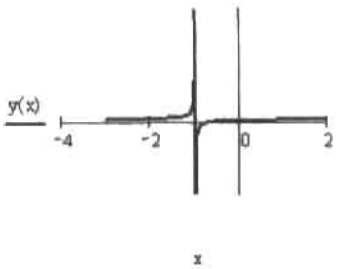


Рис. 6.5. Ошибка при построении графика

Преодолеть такую проблему в некоторых случаях можно, просто уменьшив шаг ранжированной переменной (для рассматриваемой задачи, как, впрочем, и для многих других, этот ход окажется неэффективным). Для изображения кривых проблемных функций лучше используйте автоматическое построение графика, так как при этом трудностей с точками разрыва практически никогда не возникает. Способ же задания графика с ранжированными переменными стоит применять только тогда, когда исследуемая функция непрерывна, или в том случае, если быстрым способом воспользоваться нельзя.

Использование способа построения графика с помощью оператора ранжированной переменной имеет очень важное преимущество перед быстрым методом, поскольку позволяет задавать произвольным образом шаг между узловыми точками. Конечно,

это, как правило, не совсем важно при отображении зависимости в виде кривой, но является принципиальным условием качественного построения графиков других типов — гистограмм, диаграмм, кривой с отложенными ошибками. Для таких зависимостей расстояние между точками должно быть довольно значительным, иначе элементы, их образующие, просто сольются. А так как автоматически система задает шаг довольно малым, то единственным приемлемым способом задания нетривиальных графиков Mathcad является использование оператора ранжированной переменной.

Построить график в Mathcad можно и по готовым векторам или таблицам данных, полученных, например, при эксперименте или выполнении лабораторной работы. Естественно, при этом векторы должны быть соразмерны.

О том, как произвести более корректное построение графика по экспериментальным данным, мы поговорим в гл. 16 (в разделе, посвященном интерполяции).

6.1.2. Построение нескольких графиков

В Mathcad на одну графическую область можно поместить до 16 кривых. Чтобы добавить к уже имеющемуся графику еще один, выполните следующую последовательность действий.

1. Установите курсор справа от выражения, определяющего координаты последнего ряда данных по оси Y (предварительно выделив его).
2. Нажмите клавишу «,». При этом курсор опустится на строку ниже и появится чистый маркер.
3. В появившийся маркер введите выражение для новой функции или имя функции.

С помощью описанного метода можно построить графики функций одной переменной. Если же кривые, которые нужно отобразить на одной области, зависят от различных переменных, то их, полностью аналогично добавлению новых функций, следует ввести через запятую в нижний маркер в том же порядке, что и соответствующие им функции.

Иногда возникает ситуация, когда в одной области необходимо отобразить графики функций, амплитуда изменения которых сильно различается на выбранном промежутке. Так, если попытаться максимально удачно настроить вид одной из кривых, варьируя пределы по оси Y, особенности поведения второй могут остаться незаметными при данных настройках. Подобных трудностей легко избежать, если задать требуемый интервал изменения функции, используя вспомогательную ось Y (Secondary Y Axis). Для этого дважды щелкните на графике мышью. В открывшемся окне Formatting Currently Selected X-Y Plot (Форматирование выделенной декартовой плоскости) на вкладке X-Y Axes (Оси X и Y) установите флажок Enable secondary Y axis (Задействовать вспомогательную ось Y). При этом в правой части графической области появятся три маркера: в центральном нужно указать вид зависимости, в крайних — подходящие пределы изменения. Как и в случае главной оси Y, по вспомогательной оси Y можно построить несколько графиков, правда, для функций, зависящих только от одной переменной — последней из указанных в маркере оси X.

Отобразить в одной графической области можно и кривые, заданные совершенно различными способами. Так, например, нет никакой трудности в том, чтобы визуализировать совместно график, заданный с помощью ранжированной переменной, и график, определенный через таблицу значений (рис. 6.6).

$$f(x) := \cos(\sin(x)) \qquad x := -3, -2.999.. 3$$

M :=

	0	1
0	-3	0
1	-2	1
2	-1	2
3	0	3
4	1	3
5	2	2
6	3	1
7	4	0

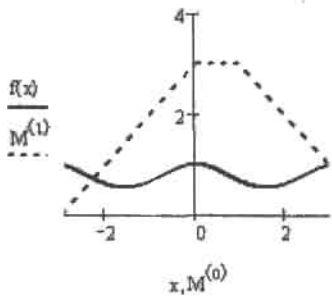


Рис. 6.6. Построение зависимостей от данных разного типа

6.1.3. Форматирование шкалы графика

Внести изменения в вид координатных осей вашего графика можно, обратившись к специальной вкладке X-Y Axes (Оси X и Y) диалогового окна Formatting Currently Selected X-Y Plot (Форматирование выделенной декартовой плоскости). Открыть данное окно можно тремя стандартными способами.

- Дважды щелкнув левой кнопкой мыши на области графика.
- Выполнив команду Format ▸ Graph ▸ X-Y Plot (Формат ▸ График ▸ X-Y зависимость).
- Воспользовавшись пунктом Format (Формат) контекстного меню графика.

Для настройки вида осей на вкладке X-Y Axes (Оси X и Y) имеются три идентичных списка параметров (рис. 6.7).

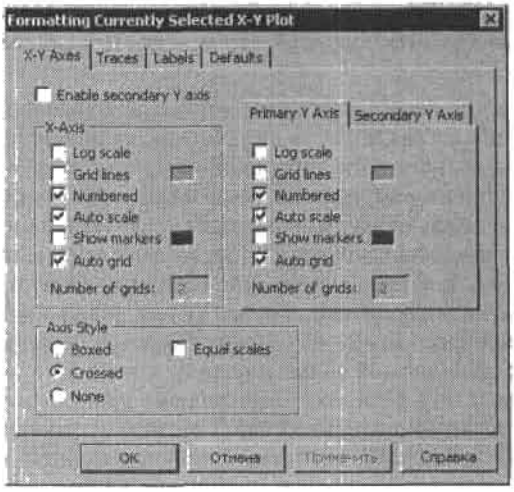


Рис. 6.7. Окно Formatting Currently Selected X-Y Plot (Форматирование выделенной декартовой плоскости)

- Log Scale (Логарифмическая шкала). Если вы выберете этот параметр, то координаты по форматируемой оси будут прологарифмированы (по десятичному логарифму). Это может быть удобно в том случае, если значения координат точек вашего

графика отличаются на несколько порядков. Обязательным условием использования логарифмической шкалы является то, что координаты кривой по выбранной оси должны быть только положительны (рис. 6.8).

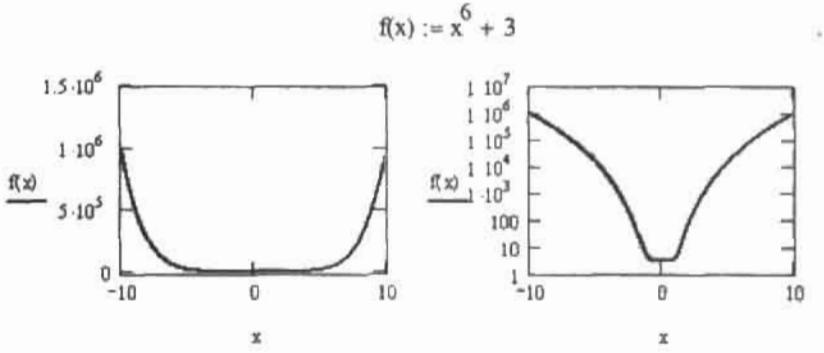


Рис. 6.8. Использование логарифмической шкалы (правый график)

- Grid lines (Линии сетки). При активизации этого параметра будет визуализирована сетка из вспомогательных линий.
- Numbered (Нумерованные). Параметр отвечает за отражение нумерации шкалы. Величина же деления шкалы определяется параметром Number of grids (Количество разбиений) и по умолчанию задается системой автоматически.
- Auto scale (Автошкала). Выбор диапазона изменения оси производится автоматически системой.
- Show markers (Показать маркеры). Используя эту команду, можно выделить с помощью специальных подписанных линий по две важные точки (корни, экстремумы, разрывы) по каждой из осей.

  Чтобы выделить какие-то значения переменной или функции, установите флажок Show markers (Показать маркеры). При этом возле форматируемой оси появятся два черных прямоугольника — маркеры, в которые и следует ввести соответствующие координаты выделяемых точек (рис. 6.9).

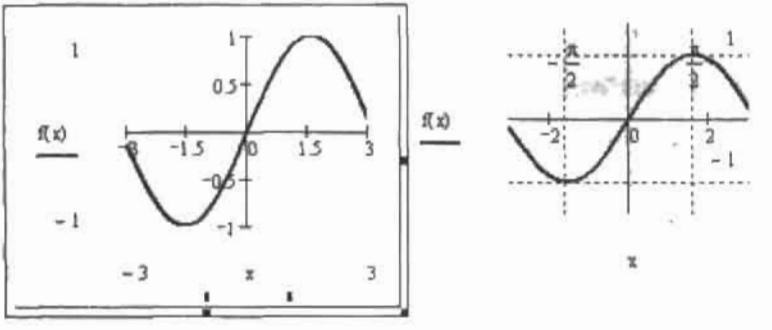


Рис. 6.9. Создание маркеров

- Auto grid (Автоматическая сетка). Параметр отвечает за автоматический выбор числа разбиений форматируемой оси. Отключив его, в окошке Number of grid (Количество разбиений) шаг нумерации можно определить произвольным образом.

Чтобы провести форматирование оси, необязательно открывать окно Formatting Currently Selected X-Y Plot (Форматирование выделенной декартовой плоскости). Сделать это можно и с помощью специального окна форматирования оси Axis Format (Формат оси), вызвать которое можно, выполнив на ней двойной щелчок мышью. Содержимое данного окна полностью повторяет рассмотренный выше список вкладки X-Y Axis (Оси X и Y).

Ниже списка параметров отображения шкалы расположено меню Axes Style (Стиль осей), отвечающее за особенности отображения координатной системы (см. рис. 6.7). Меню содержит четыре пункта.

- Boxed (Прямоугольные). Оси пересекаются в наименьших точках диапазона. Параметр, определенный по умолчанию.
- Crossed (Пересеченные). Оси пересекаются в точке (0,0). Стандартный вид декартовой системы координат.
- None (Нет). График отображается без осей.
- Equal Scales (Равные шкалы). Разбиение осей производится в равном масштабе, то есть их отрезки, численно равные по модулю, будут равны и по длине. Параметр может быть полезен в том случае, если правильность и неискаженность вида кривой принципиальна — например, при построении окружности.

6.1.4. Форматирование графиков

С помощью второй вкладки Traces (Ряды данных) диалогового окна Formatting Currently Selected X-Y Plot (Форматирование выделенной декартовой плоскости) можно произвести настройку вида кривой (более правильно — ряд данных, так как кривая — это лишь одна из восьми возможных форм отображения двумерной зависимости в Mathcad).

Главным элементом вкладки Traces (Ряды данных) (рис. 6.10) является список, содержащий 16 строк (по максимально возможному количеству графиков в одной графической области). Каждая строка отвечает за отображение одного ряда данных. Расположены строки настройки вида рядов данных в том же порядке, в котором они были прописаны в соответствующем маркере графической области. Перейти от одной строки списка к другой можно либо с помощью мыши, либо используя клавиши управления курсором.

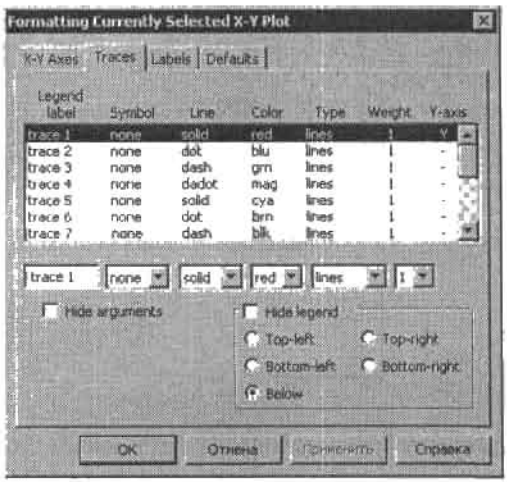


Рис. 6.10. Вкладка Traces (Ряды данных)

Каждая строка ряда данных содержит семь пунктов настройки. Разберем их по порядку.

- Legend Label (Легенда). Своеобразное «имя» набора параметров, описывающих отображение зависимости на графике. Если отключить расположенный ниже рассматриваемого списка параметр Hide Legend (Спрятать легенду), то на графической области появятся специальные строки с текстом легенды и фрагментом линии, с помощью которой визуализируется соответствующая ей кривая. Расположение легенды можно регулировать с помощью пяти пунктов меню Hide Legend. В зависимости от настройки легенда будет помещена в одном из углов области построения графика (Top-left, Top-right, Bottom-left, Bottom-right) либо внизу, в служебной зоне (Below).
- Symbol (Символ). Параметр определяет символ, которым отображаются узловые точки графика (рис. 6.11). Таких символов пять:
  - x's — точки отображаются в виде латинской буквы x;
  - +'s — в качестве символа узловой точки выступает знак «плюс»;
  - box (Прямоугольник) — точки на графике визуализируются в виде маленьких квадратиков;
  - dmnd (от англ. diamond — бриллиант) — точки по своей форме стилизованы под алмаз;
  - o's — точки графика отображаются кружками.

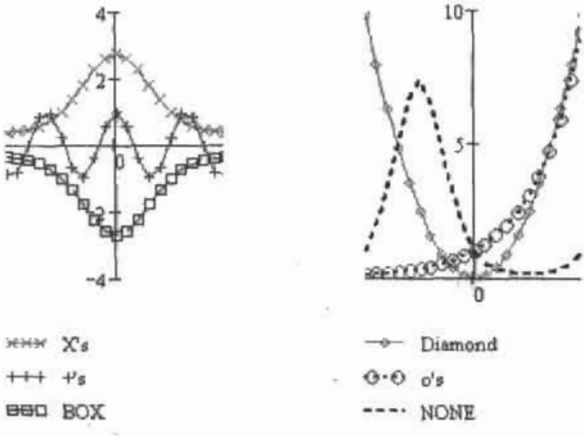


Рис. 6.11. Отображение узловых точек символами различного типа

По умолчанию для типа графика Lines (Кривые) узловые точки не визуализируются (что соответствует пункту None (Нет) списка Symbol (Символ)).

- Line (Линия). Параметр определяет, линией какого типа будет прочерчена зависимость. Всего же таких типов четыре (рис. 6.12).
  - Solid (Сплошная). Кривая проводится непрерывной линией.
  - Dot (Пунктир). Здесь для отображения зависимости используется пунктирная линия.
  - Dash (Штрих). График будет прочерчен штриховой линией.
  - Dadot (от англ. Dash+Dot). Штрихпунктирная линия.

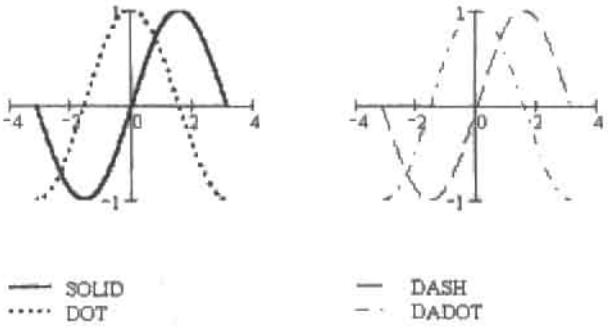


Рис. 6.12. Возможные типы линий графиков в Mathcad

- Color (Цвет). С помощью данной настройки можно задать наиболее подходящий цвет линий и точек. Всего на палитре восемь оттенков.
- Type (Тип). Параметр определяет тип представления ряда данных. В Mathcad возможно построить декартовы графики восьми видов.
  - Lines (Линия). График в виде обычной кривой. Все графики, с которыми мы сталкивались до этого, были построены именно по этому типу. Параметр, определенный по умолчанию.
  - Points (Точки). Параметр отвечает за задание точечного графика. Этот тип представления ряда данных характеризуется тем, что при его выборе будут визуализированы только узловые точки, без соединительных линий.
  - Error (Погрешность). График с отложенной погрешностью. Имеет ряд особенностей в задании и большое практическое значение, поэтому далее мы рассмотрим его построение отдельно.
  - Bar (Столбец). График отображается в виде столбчатой диаграммы (гистограммы). Данный тип отображения ряда данных широко применяется в статистике (о нем мы поговорим в гл. 15).
  - Step (Шаг). Параметр отвечает за задание довольно близкого к гистограмме шагового графика (шаговый график есть не что иное, как контур обычной гистограммы).
  - Draw (Рисунок). Кривая прорисовывается между узловыми точками.
  - Stem (Стержень). Выбрав этот параметр, вы построите довольно оригинальную стержневую диаграмму.
  - Solidbar (Сплошные столбцы). Разновидность гистограммы, отличающаяся от задаваемой параметром Bar (Столбцы) тем, что ее столбцы залиты выбранным оттенком.
- Weight (Толщина). Параметр определяет толщину линий и величину символов точки. Может изменяться от 1 до 9, а также принимать значение p. Величина параметра Weight определяется тем, во сколько раз задаваемая им линия толще (или символ больше) минимального уровня соответствующей характеристики. Выбрав значение p, вы построите график, толщина линии которого будет равна одному пикселу. Однако нужно учитывать, что при распечатке документа на принтере с низким разрешением такой график может и не отобразиться.

- Y-Axis (Ось Y). Параметр отображает, по какой из осей ординат — главной (Y) или вспомогательной (Y2) — построен график зависимости. Изменить данную настройку непосредственно в строке ряда данных нельзя, для этого вы должны переместить выражение, описывающее функцию, из маркера оси Y в маркер оси Y2 или наоборот.

Итак, внимательно рассмотрев вкладку Traces (Ряды данных), вы увидите, что на ней остался только один неописанный нами параметр — Hide Arguments (Спрятать аргументы), расположенный ниже изученного списка форматирования вида кривой. Установив данный флажок, вы очистите служебную зону графической области от имен функций и переменных. Это может быть очень удобно в том случае, если на графике расположено много кривых.

6.1.5. Создание заголовка графика и подписи оси

Чтобы озаглавить ваш график, откройте вкладку Labels (Пометки) диалогового окна Formatting Currently Selected X-Y Plot (Форматирование выделенной декартовой плоскости). Данная вкладка (рис. 6.13) содержит специальную строку Title (Заглавие), в которую следует ввести текст будущего названия. Определить расположение заглавия относительно окна кривой можно с помощью параметров Above (Выше) и Below (Ниже). Чтобы название вашего графика было визуализировано, установите флажок Show Title (Показать заглавие). В противном случае заглавие будет отражаться только при выделении графика.

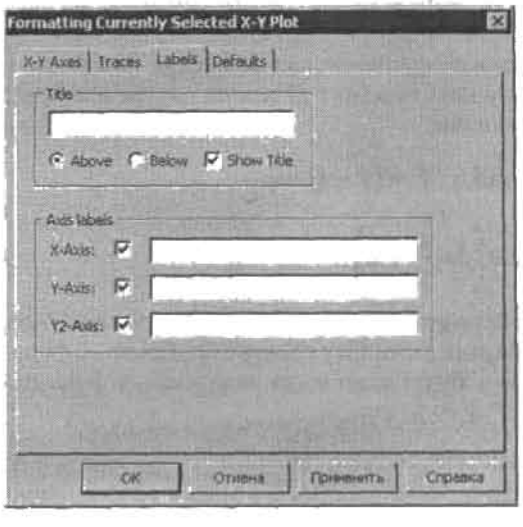


Рис. 6.13. Вкладка Labels (Пометки)

Меню Axis Labels (Пометки к осям) рассматриваемой вкладки служит для задания подписей к осям. Аналогично созданию заглавия, текст подписи следует ввести в специальную строку (X-Axis для оси X, Y-Axis для главной оси Y, Y2-Axis для вспомогательной оси Y). Текст подписи будет отображен в том случае, если в окошках возле строк ввода текста установлены флажки.

6.1.6. Изменение установок по умолчанию

С помощью последней вкладки Defaults (По умолчанию) диалогового окна Formatting Currently Selected X-Y Plot (Форматирование выделенной декартовой плоскости) можно внести некоторые изменения в установки вида графика, принятые по умолчанию.

Вкладка содержит всего два параметра настройки.

- Кнопка Change to Defaults (Изменить, как по умолчанию). Нажав ее, вы вернетесь к виду графика, определенному настройками, принятыми в системе по умолчанию.
- Окошко Use for Defaults (Использовать, как по умолчанию). Если вы установите этот флажок, то настройки форматируемого графика будут приняты системой как настройки по умолчанию для всего документа.

6.1.7. Создание графика с отложенной погрешностью

При оформлении лабораторных и других работ по физике на полученных или использованных графиках принято откладывать погрешность эксперимента. Делается это, в частности, с помощью вертикальных отрезков, проводимых, как правило, с равным расстоянием между ними. Откладываются они, исходя из физического смысла погрешности, как выше, так и ниже экспериментальной кривой. Длина отрезков определяется величиной погрешности.

В Mathcad существует возможность построения графика с отложенными ошибками подобного типа. Причем создание такой зависимости имеет целый ряд особенностей и отличий от остальных типов отображения рядов данных. Поэтому рассмотрим задание данного графика пошагово.

1. Для начала следует задать исходные данные: функцию, описывающую физический процесс, или таблицу измерений, величины констант. В нашем случае это будет уравнение колебаний пружинного маятника:

$$A := 2\cdot m \qquad \phi := \frac{\pi}{4}\cdot rad \qquad T := (\pi - 1)\cdot sec$$

$$X(t) := A\cdot\cos\left(2\pi\cdot\frac{t}{T} + \phi\right)$$

2. Далее следует задать выражения для определения погрешности. Их должно быть два: для верхней и нижней ошибки. Верхнюю ошибку следует прибавить к функции, нижнюю — отнять. Таким образом, будет задано два ряда данных функций погрешности:

$$\Delta := 0.2\cdot sec$$

$$ErrUP(t) := X(t) + \left(\frac{d}{dt}X(t)\right)\cdot\Delta \qquad ErrDn(t) := X(t) - \left(\frac{d}{dt}X(t)\right)\cdot\Delta$$

В том случае, если функция задана таблицей измерений, то вектор погрешности проще всего провести через экспериментальные точки. Для этого нужно задать вектор ошибок и прибавить или отнять его от вектора опытных значений. В результате будут получены векторы верхней и нижней погрешности, действовать с которыми нужно точно так же, как с ошибкой, заданной через функцию.

3. Если погрешность задается через функцию, обязательным является определение ряда данных с помощью ранжированной переменной с относительно большим шагом. Иначе отрезки ошибки будут расположены столь близко друг к другу, что просто сольются. Естественно, никакой информации из такого графика почерпнуть не удастся.

$$t := 0, 0.15.. \frac{2\pi}{\sqrt{3}}$$

4. Когда ряды данных функции и погрешности заданы, можно непосредственно приступать к построению графика. Для начала, аналогично заданию нескольких кривых в одной графической области, следует определить в маркере функций все три ряда данных. Затем на вкладке Traces (Ряды данных) следует изменить тип отображения функций ошибки с Lines (Кривая) на Error (Ошибка). В результате для рассматриваемой задачи получим график, изображенный на рис. 6.14.

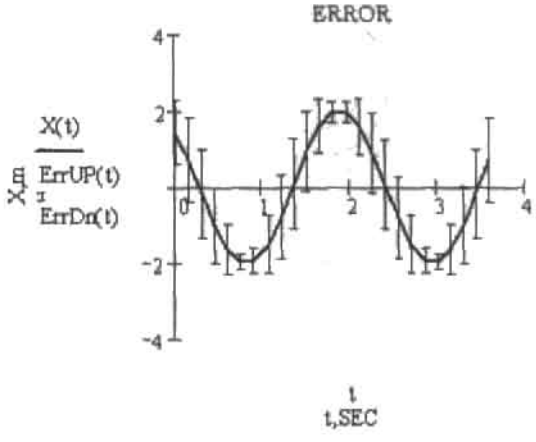


Рис. 6.14. График с отложенной погрешностью

Построить только верхнюю или только нижнюю ошибку нельзя: тип отображения Error (Ошибка), в отличие от всех остальных, требует задания двух рядов данных. При задании графика с отложенной погрешностью цвет рядов данных ошибки желательно сделать одинаковым из чисто эстетических соображений.

6.1.8. Создание графиков в полярных координатах

Задание графиков в полярной системе координат с чисто технической точки зрения не имеет ровным счетом никаких принципиальных отличий от создания графиков на декартовой плоскости. Поэтому этот вопрос мы рассмотрим довольно кратко.

Ход создания графика в полярных координатах в точности повторяет задание кривой в декартовой системе. Для начала нужно ввести графическую область. Сделать это можно либо с помощью специальной кнопки Polar Plot (Полярный график) панели Graph (Графические), либо сочетанием клавиш Ctrl+7. Как и в случае зависимости X-Y, для полярного графика существует два основных метода построения: быстрый способ построения и использование ранжированных переменных (рис. 6.15). Причем последний имеет

большее значение ввиду того, что поменять стандартную величину изменения угла (от 0 до 360°) непосредственно на графической области нельзя. При задании полярной системы координат по быстрому методу система автоматически определит область изменения угла от 0 до 360°.

В отличие от области изменения угла, величину диапазона полярного радиуса можно задать произвольным образом непосредственно на графической области точно так же, как меняется диапазон осей в случае декартовой плоскости.

Как и при построении в декартовой системе координат, на одну полярную графическую область можно поместить до 16 кривых. Правила их задания одинаковы в обоих случаях.

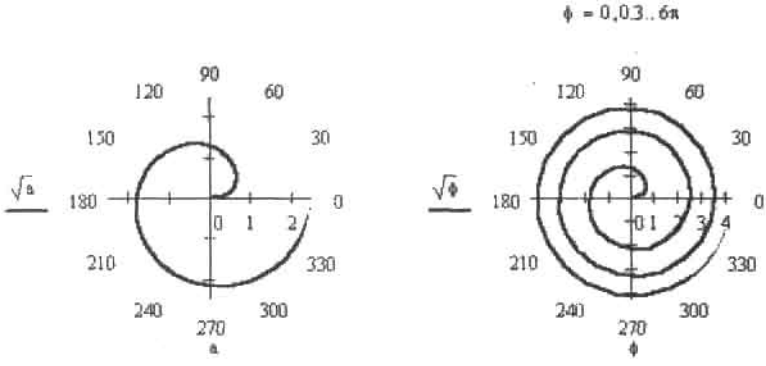


Рис. 6.15. Задание спирали Ферма быстрым способом (слева) и с помощью оператора ранжированной переменной

Чтобы внести изменения в вид системы координат или кривых, следует, аналогично декартовым графикам, обратиться к диалоговому окну Formatting Currently Selected Polar Plot (Форматирование выделенного полярного графика).

Существенное отличие между окнами форматирования декартовой и полярной систем координат имеются только на вкладке Polar Axes (Полярные оси), поэтому мы остановимся только на ней (рис. 6.16).

На данной вкладке, в отличие от аналогичной вкладки в случае форматирования декартовой плоскости, списки параметров, отвечающих за вид осей, не идентичны. Так, если список настроек полярного радиуса практически полностью повторяет аналогичный список для форматирования осей декартовой плоскости (за исключением параметра AutoScale (Автошкала)), то набор параметров для полярного угла куда более скромен. Из практически важных параметров стоит выделить создание сетки из вспомогательных линий (Grid Lines) и выделение специальных значений (Show Markers).

С помощью меню Axis Style (Стиль осей) рассматриваемой вкладки можно определить вид отображения системы координат. Меню содержит три параметра.

- Perimeter (Периметр). Ось со значениями полярного радиуса не отображается, а визуализируется только ее шкала в виде вертикального столбца.
- Crossed (Пересеченные). Система отображается в виде пересечения двух перпендикулярных прямых наподобие декартовой.
- None (Нет). Система координат не отображается.

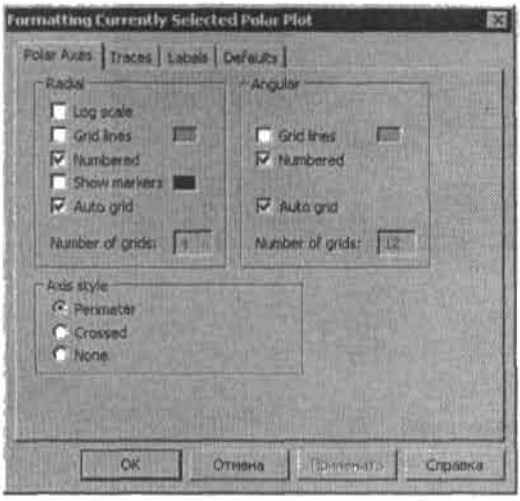


Рис. 6.16. Вкладка Polar Axes (Полярные оси)

Настройку вида осей, аналогично декартовому графику, можно произвести и с помощью специального меню, вызываемого двойным щелчком правой кнопкой мыши в области форматируемой оси.

При построении полярного графика можно использовать все те же виды линий, точек, типов отображения рядов данных, что и в случае зависимости X-Y. Никаких отличий в технике и возможностях форматирования вида кривой в полярной и декартовой системах координат нет. Однако строить, например, статистическую гистограмму в полярной системе координат — это, мягко говоря, бессмысленно.

6.1.9. Увеличение фрагмента графика

Зачастую на практике требуется увеличить фрагмент графика. В Mathcad это можно сделать очень просто благодаря специальному инструменту Zoom (Масштаб) панели Graph (Графические).

Чтобы увеличить фрагмент графика, выполните следующую последовательность действий.

1. Выделите график и введите панель инструмента Zoom (Масштаб) (без выделения графической области инструменты панели Graph будут недоступны).
2. Аналогично стандартному выделению нескольких объектов в Windows, мышью возьмите фрагмент кривой, требующий увеличения, в штрихованный прямоугольник. Точность выделения зоны увеличения можно контролировать благодаря тому, что в специальных окошках Min (Минимум) и Max (Максимум) отражаются значения координат вершин прямоугольника по обеим осям (рис. 6.17).

$$f(x) := (|x|)^{-|x|}$$

3. В том случае, когда область увеличения выделена, нажмите кнопку с изображением лупы с плюсом в центре. При этом выделенный фрагмент займет всю область кривой (рис. 6.18).

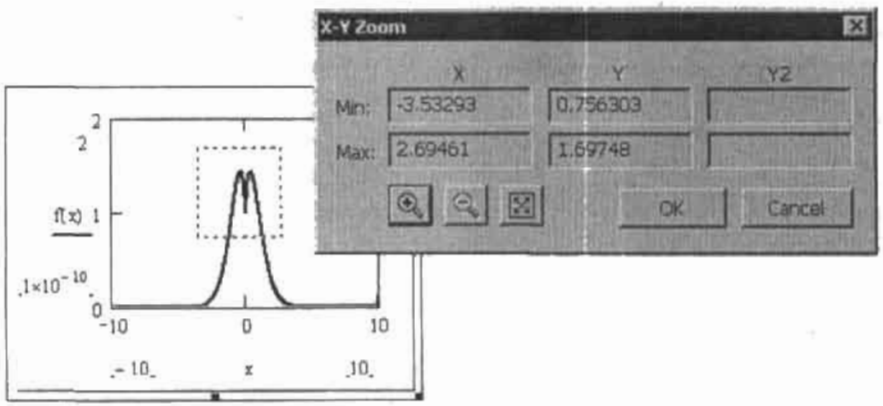


Рис. 6.17. Использование инструмента Zoom (Масштаб)

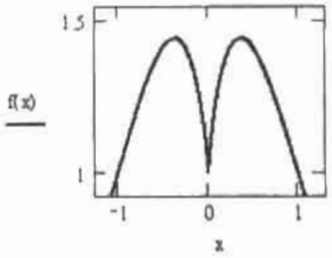


Рис. 6.18. Увеличенный фрагмент графика

Если какой-то шаг при последовательном увеличении фрагмента кривой был сделан неверно, то вернуться на предыдущий этап масштабирования можно с помощью кнопки с изображением лупы с минусом в центре.

Используя последнюю из трех кнопок, можно вернуться к первоначальному виду графика.

6.1.10. Трассировка графиков

Пожалуй, еще чаще, чем к инструменту масштабирования, приходится прибегать к использованию второго инструмента панели Graph (Графические) — Trace (Трассировка). При его помощи можно довольно точно определить координаты интересующей вас точки (например, корня или экстремума).

Для трассировки выполните следующую последовательность действий.

1. Выделите график и откройте панель инструмента Trace (Трассировка).
2. В результате на графике появится своеобразный «датчик» в виде пересекающихся штрихованных прямых (рис. 6.19). Координаты точки пересечения этих прямых отображаются на панели инструмента в строках X-, Y- и Y2-Value (Координаты по X, Y и Y2). Перемещать датчик можно, просто изменяя положение курсора.

$$f(x) := x^3 - 3x - 1$$

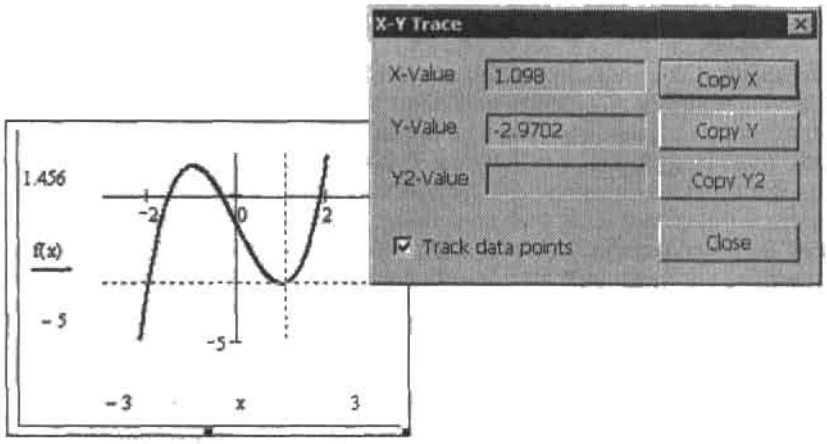


Рис. 6.19. Трассировка графика

3. Когда нужная точка будет найдена, нажмите кнопку Copy X/Y/Y2 (Копировать X/Y/Y2), в зависимости от того, какая координата вас интересует. При этом ее численное значение будет скопировано в буфер.

Значительно облегчается работа при включенном параметре Track Data Points (Следовать точкам данных), так как при этом датчик инструмента Trace (Трассировка) будет скользить исключительно по кривой (см. рис. 6.19).

Совместное использование инструментов панели Graph (Графические) позволяет весьма точно решать такие задачи, как определение координат корней, точек экстремумов, разрывов различных функций.

6.2. 3D-графики

Со всей смелостью и ответственностью можно определить тему данного раздела — трехмерные графики (или 3D-графики) — как задачу недоступной для человека сложности. Конечно, очень легко проанализировать на предмет определяемых ими поверхностей такие классические уравнения, как уравнение шара, параболоида, эллипсоида. Но для зависимости нестандартной, тем более содержащей какие-либо специальные функции, построить поверхность вручную на бумаге очень сложно, практически невозможно. И если даже удалось бы определить координаты тех 500–1000 узловых точек, необходимых для задания среднего, более или менее гладкого трехмерного графика, то как их затем правильно спроецировать в объем, соединить и придать вид поверхности — это большой вопрос! Естественно, даже для человека очень незаурядных способностей и трудолюбия задача такая окажется практически не решаемой. С помощью же Mathcad такие графики можно строить очень просто, причем широкие возможности форматирования полученных изображений позволяют получать поверхности очень высоких художественных качеств.

Тема 3D-графиков очень объемная и довольно сложная, она содержит множество методик, специальных функций и параметров. В данном разделе мы попытаемся прежде всего систематизировать это многообразие и на примерах продемонстрировать единый принцип, лежащий в основе всех способов построения трехмерных объектов.

6.2.1. Способы задания 3D-графиков

Как уже говорилось выше, способов задания поверхности в Mathcad существует великое множество. Рассмотрим для начала наиболее простой и практически важный, быстрый метод построения трехмерного графика (QuickPlot). В его основе лежит тот же принцип, что и при быстром задании двумерной зависимости: пользователь определяет только вид функции, а все параметры построения, такие как шаг между узловыми точками, диапазон шкал осей и система координат, задаются автоматически системой. Впрочем, даже быстрый метод в случае трехмерной системы координат имеет ряд тонкостей, поэтому ход построения этим методом поверхности рассмотрим по пунктам.

1. Для начала введем графическую область 3D-графика. Аналогично зависимостям X-Y, сделать это можно тремя стандартными способами: либо нажав кнопку Surface Plot (Поверхность) панели Graph (Графические), либо использовав одноименную команду меню Insert (Вставка), либо с помощью сочетания клавиш Ctrl+2.

   Проводя аналогию с построением двумерной декартовой системы координат, вполне логичным будет предположить, что графическая область поверхности должна иметь три маркера для определения вида функции и ее переменных. На самом же деле маркер есть только один. В общем случае в нем должен быть прописан массив, содержащий координаты узловых точек по всем трем осям. Тому, как этот массив можно задать и что он собой представляет, и будет посвящен, во многом, данный раздел.

   Никаких принципиальных различий, кроме описанного выше, между графическими областями двумерных и трехмерных объектов нет. Поэтому на особенностях форматирования графической области при создании поверхности мы останавливаться не будем: оно абсолютно идентично двумерному случаю.

2. После того как графическая область введена, следует задать вид функции, определяющей поверхность. В отличие от X-Y-зависимостей, просто ввести ее выражение в маркер нельзя — при этом будет выдано сообщение об ошибке: This variable is undefined (Данная переменная не определена).

   Мы будем строить поверхность следующей функции:

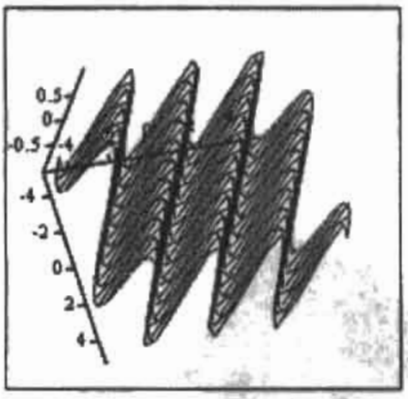
$$f(x,y) := \sin(x + 2y)$$

3. В маркер графической области следует ввести имя заданной выше функции. Однако, в отличие от двумерного случая, прописан должен быть лишь непосредственно текст имени, без переменных в скобках. Это различие связано с тем, что трехмерный график, как уже говорилось выше, должен быть задан через массив данных, содержащий численные значения координат по всем трем осям. Вполне очевидно, что матрица с числами никак не может быть одновременно и функцией двух переменных. При быстром же построении поверхности мы имеем дело с таким же массивом, как если бы мы просто набрали его вручную, однако в данном случае параметры его определяются системой автоматически.

   Для рассматриваемой функции системой Mathcad будет нарисован график, изображенный на рис. 6.20.

При использовании данной методики поверхность задается на стандартном интервале от –5 до 5 по обеим переменным. Естественно, такой диапазон во многих случаях может быть неприемлем (так, для нашей функции поверхность быстрого построения получилась слишком сжатой, что делает ее неглядной и неудобной для изучения). Изменить же интервал по выбранной оси так, как это делалось в двумерном случае,

нельзя. Для форматирования параметров графиков быстрого построения существует специальная вкладка Quick Plot Data (Данные графика быстрого построения) окна форматирования трехмерных графиков 3D-Plot Format. Открывается это окно либо двойным щелчком левой кнопкой мыши на графической области, либо с помощью команды Format (Формат) ее контекстного меню (рис. 6.21).



f

Рис. 6.20. 3D-график быстрого построения

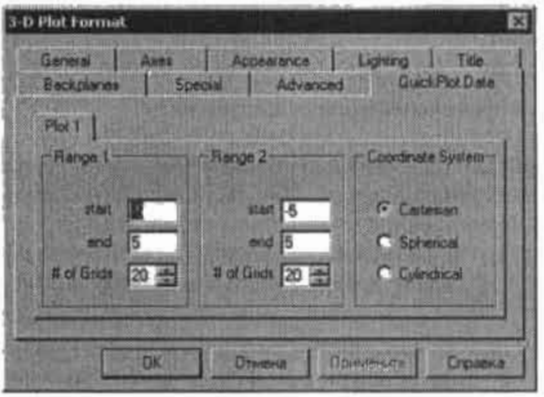


Рис. 6.21. Вкладка QuickPlot Data (Данные графика быстрого построения)

Все параметры настройки графика быстрого построения расположены на вкладке Plot 1 (График 1). В общем случае таких вкладок может быть несколько, что связано с тем, что на одной графической области может быть размещено несколько поверхностей — чтобы это сделать, просто введите через запятую имена функций, графики которых должны быть построены.

Вкладка Plot 1 (График 1) содержит три меню настройки, две из которых, Range 1 и Range 2 (Ряд 1 и Ряд 2), идентичны друг другу. Эти меню отвечают за характеристики сетки построения поверхности вдоль каждой из осей переменных (соответствие переменной ряду определяется последовательностью введения ее при задании имени функции) и содержат следующие параметры настройки.

- Start (Начало). В окошке данного параметра вы можете произвольным образом задать начальную точку построения прямоугольника по данной оси.
- End (Конец). Здесь вы можете определить конечную точку интервала.
- # of Grids (Количество линий сетки). Параметр определяет, на сколько отрезков будет разбит интервал построения по выбранной переменной (что соответствует числу отображенных линий сетки) (рис. 6.22). Величина, обратная шагу. Очень важный параметр, особенно в случае не очень гладких функций и разрывных функций.

$$f(x,y) := \frac{1}{x} + \frac{1}{y}$$

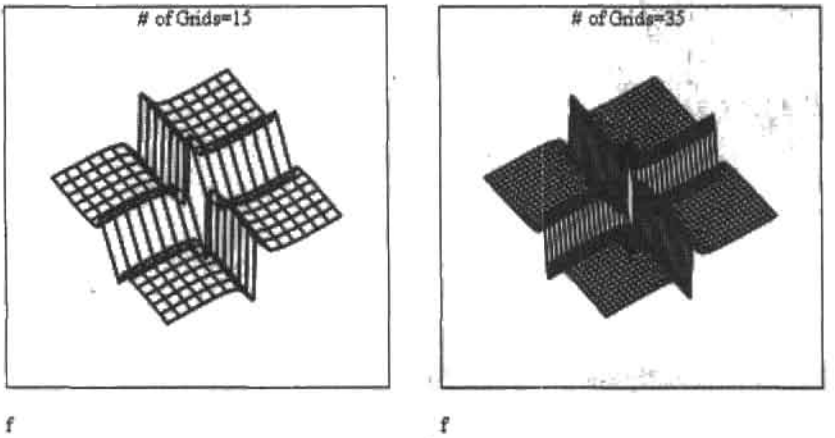


Рис. 6.22. Вид графика при различных величинах ячейки сетки построения

Количество разбиений интервала может быть принципиальным в том случае, если на выбранном промежутке функция имеет точки разрыва. Так, если вы попытаетесь построить по быстрой методике график функции, представленной на рис. 6.22, то при стандартных настройках система не начертит поверхность, а выдаст сообщение об ошибке: Divide by zero in function evaluation (В выражении функции присутствует деление на нуль).

И действительно, если интервал построения по каждой переменной (а относительно них функция симметрична) изменяется от –5 до 5 с шагом 0.5 (# of Grids=20), то десятая линия сетки по каждой из переменных пройдет через 0. При этом и возникнет неопределенность, на которую ссылается программа. Преодолеть же проблему такого рода можно очень просто: для этого шаг нужно сделать таким, чтобы координаты узловых точек не принимали нулевые значения (что и было сделано при построении графиков на рис. 6.22).

Внимательно рассмотрев полученный при изменении шага график, можно сделать вывод, что он, в принципе, построен неверно, так как факт существования точек разрыва на нем не отображен. Там, где график должен уходить на бесконечность, он просто перегибается. То есть разрывная функция визуализируется непрерывной поверхностью. Чтобы понять причину ошибки и попробовать как-то ее исправить, разберемся в тех принципах, по которым Mathcad строит трехмерные объекты.

А принципы эти по своей идее предельно просты и сводятся к следующему алгоритму.

1. Аналогично двумерному случаю, интервал по каждой из осей переменных разбивается на заданное количество отрезков. Границы этих отрезков дают координаты узловых точек. При этом, если число разбиений по X равно N, а по Y — M, то для каждого значения X будет существовать M точек с различными координатами по Y, и, наоборот, каждому Y будет соответствовать N значений X. Визуально это можно представить в виде сетки, величина ячейки которой определяется шагом по каждой из переменных, а в узлах находятся точки, относительно которых производится построение. Увидеть такую сетку для конкретного графика можно, повернув его строго перпендикулярно монитору.
2. Когда сетка разбиений задана, вычисляются значения функции в ее узлах. Если остановиться на этом этапе и визуализировать только точки, то будет построен так называемый точечный график (Data Points).
3. Каждая точка соединяется с соседними с помощью отрезков прямых, применяются сглаживание и другие графические эффекты, в результате чего, в зависимости от величины ячейки сетки, получается более или менее гладкая поверхность.

Зная алгоритм, совсем не сложно понять, отчего графики разрывных функций получаются в Mathcad не совсем верными. Все дело в том, что при верном подборе шага система просто пропускает точки разрыва (при неверном же, как было показано выше, график не будет построен). То есть Mathcad соединяет отрезком две точки, лежащие слева и справа от разрыва, в результате чего вместо разрыва отображается перегиб. Кстати, с аналогичной проблемой мы уже сталкивались, когда разбирали построение X-Y-зависимостей с помощью оператора ранжированной переменной.

К сожалению, преодолеть возникшие проблемы, связанные с разрывами функции, аналогично двумерному случаю (уменьшив шаг по переменным), вряд ли получится. Поэтому при построении поверхностей, описываемых разрывными функциями, к результату следует относиться весьма осторожно. А еще более правильным будет анализировать поведение такого рода функций с помощью двумерных графиков, приняв одну из переменных за константу.

Третье меню вкладки Plot 1 (График 1) — Coordinate System (Система координат) — определяет, в какой системе координат следует отобразить данную зависимость. Возможны следующие варианты.

- Cartesian (Декартова). График отображается в декартовой системе координат.
- Spherical (Сферическая). Сферическая система координат.
- Cylindrical (Цилиндрическая). Цилиндрическая система координат.

Относительно преобразования изображения из одной системы координат в другую следует дать некоторые пояснения. Все дело в том, что если мы изучаем параметры какого-то реального объекта, то, вне зависимости от того, в какой системе координат мы это делаем, характеристики его будут получены одни и те же. То же самое касается, в частности, и поверхностей: какие бы замены мы ни произвели, переходя из одной системы в другую, никаких изменений в виде поверхности произойти не должно. Так, например, несмотря на то, что вид системы уравнений, задающей поверхность цилиндра, весьма значительно отличается в различных системах координат, каждая из них описывает один и тот же объект — просто подход к такому описанию различный. Но с точки зрения человека какая-то система окажется все же лучше, поскольку уравнения в ней имеют более простой вид. Так, упомянутая поверхность цилиндра в цилиндрической системе координат описывается системой уравнений, определяющей в декартовой системе наиболее простой трехмерный объект — плоскость (рис. 6.23).

Естественно, что работать с такой системой гораздо проще, особенно если расчеты производятся вручную.

$$f(\rho, z) := \begin{pmatrix} R\cdot\sin(\rho) \\ R\cdot\cos(\rho) \\ z \end{pmatrix} \qquad g(x, y) := \begin{pmatrix} 1 \\ x \\ y \end{pmatrix}$$

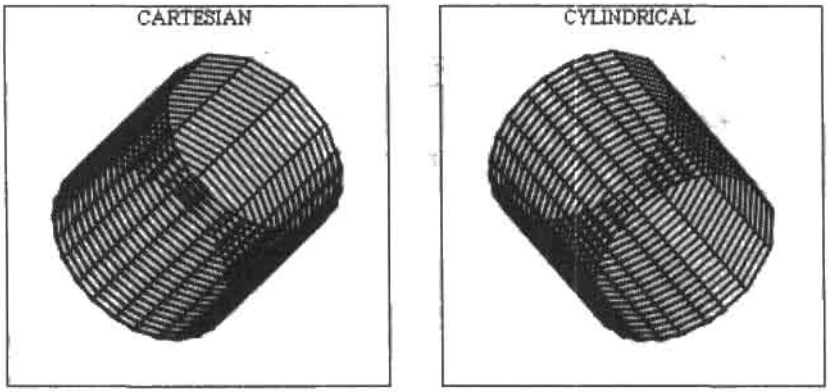


Рис. 6.23. Задание поверхности цилиндра в декартовой и цилиндрической системах координат

Таким образом, уравнение одного вида в различных системах координат описывает почти наверняка различные поверхности, и наоборот, два различных уравнения в разных системах координат могут дать одну и ту же поверхность. Второе свойство очень широко используется, например, для придания геометрического смысла заменам переменных при интегрировании. Кстати, на самом деле, любая замена переменных является в то же время переходом к новой системе координат. И в Mathcad существует возможность построения поверхности в произвольной системе координат — но для этого нужно использовать специальную функцию CreateMesh, о которой мы поговорим немного позже.

Теперь, зная методы настройки вида поверхности быстрого построения, попробуем подобрать наиболее подходящие для нашей периодической функции параметры. Также, из чисто теоретического интереса, попробуем построить определяемую ею поверхность, например, в цилиндрической системе координат (рис. 6.24).

$$f(x,y) := \sin(x + 2y)$$

Согласитесь, что после изменения интервала построения и шага поверхность стала куда более наглядной. А вид графика функции в цилиндрической системе координат получился очень необычным и интересным.

В подавляющем большинстве случаев быстрый метод построения поверхности, с учетом возможности форматирования параметров сетки разбиений и системы координат, вполне приемлем для задания графиков любых функций. Однако иногда задать нужный объект с помощью обычного уравнения вида Z=f(x,y) невозможно по причине отсутствия такового для данной поверхности. Целая группа трехмерных объектов опре-

деляется с помощью систем параметрических уравнений, и поэтому для их задания нужно искать другой подход. Не менее важным является также построение поверхностей исходя из таблиц экспериментальных данных, которое имеет целый ряд совсем неочевидных особенностей.

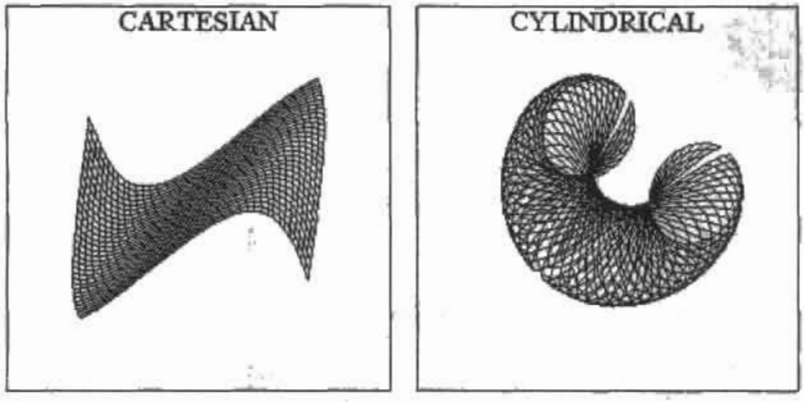


Рис. 6.24. Поверхность быстрого построения с измененными параметрами

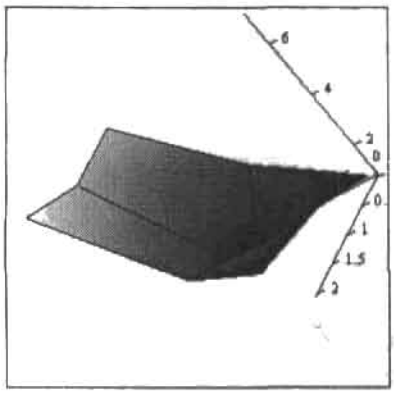
Способов построения поверхности, различающихся между собой какой-то технической деталью, но использующих один и тот же принцип, можно придумать великое множество. Описывать их все, естественно, нет никакого смысла, поэтому мы остановимся лишь на наиболее важных или интересных. Чтобы разговор наш был максимально эффективным, все эти способы будут приложены к решению одной и той же задачи — построению сферы.

Способ 1. Матрица значений

Если представить такую гипотетическую ситуацию, что уравнение сферы неизвестно, то построить ее изображение можно, произведя измерение значений координат нескольких сотен точек реального шара и организовав их тем или иным образом в виде массива данных. Наиболее очевидным будет создать таблицу из трех столбцов: в первом будут расположены координаты экспериментальных точек по X, во втором — по Y, в третьем — по Z. Например:

$$z := \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 3 \\ 6 & 6 & 7 \end{pmatrix}$$

Внимательно изучив рис. 6.25, можно обнаружить, что Mathcad построила поверхность совсем по другому принципу, чем мы рассчитывали. Вместо того чтобы принять каждый из столбцов за вектор с координатами по соответствующей оси, программа поступила довольно неожиданно: все значения элементов матрицы были приняты как координаты по Z, а в качестве координат по X и Y были использованы значения индексов элементов. Естественно, что подобного рода подход вряд ли может быть эффективно использован для решения поставленной задачи.

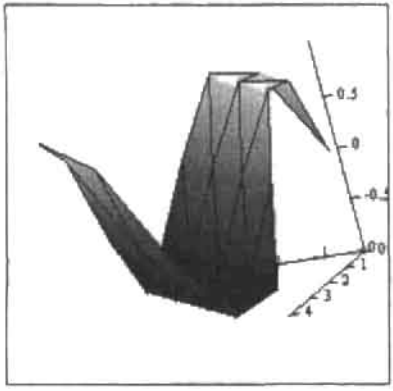


z

Рис. 6.25. Задание поверхности с помощью матрицы значений

Для задания поверхности по методике, схожей с описанной, в Mathcad существует специальная функция matrix(m,n,f) (матрица). Функция формирует матрицу, элементы которой равны значениям функции f(x,y), исходя из того условия, что x=i, y=j (то есть переменные определяются равными соответствующим матричным индексам данного элемента). Количество строк создаваемой матрицы определяется в первом маркере имени функции (параметр m), количество столбцов — во втором (параметр n). Большого практического значения функция matrix не имеет (прежде всего из-за того, что поверхность с ее помощью может быть задана только для положительных значений переменной), хотя в ряде случаев может быть довольно полезной (рис. 6.26).

$$f(x,y) := \sin(x+y) \qquad M := \text{matrix}(5,5,f)$$



$$M = \begin{pmatrix} 0 & 0.841 & 0.909 & 0.141 & -0.757 \\ 0.841 & 0.909 & 0.141 & -0.757 & -0.959 \\ 0.909 & 0.141 & -0.757 & -0.959 & -0.279 \\ 0.141 & -0.757 & -0.959 & -0.279 & 0.657 \\ -0.757 & -0.959 & -0.279 & 0.657 & 0.989 \end{pmatrix}$$

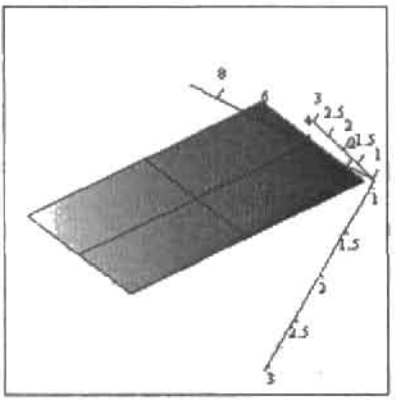
M

Рис. 6.26. Использование функции matrix (матрица)

Чтобы понять, каким же образом должны быть организованы координаты экспериментальных точек, чтобы по ним можно было построить поверхность (рис. 6.27), попробуем проанализировать, по каким принципам представляет данные, содержащие коор-

динаты узловых точек, сама система при использовании изученного нами выше быстрого метода задания 3D-графика. Как вы помните, визуально разбиение интервала при этом легче всего представить в виде сетки. Через значения функции в узлах этой сетки и проводится поверхность. Математически же множество координат узловых точек сетки по каждой из осей можно представить в виде трех матриц. Нетрудно догадаться, что каждый отдельно взятый столбец матрицы X и каждая отдельная строка матрицы Y будет содержать одинаковые значения координат. Для того же, чтобы матрицы рассматривались совместно, нужно просто объединить их в один массив. Наиболее просто это можно сделать, введя имена матриц со значениями координат узловых точек через запятую в маркер графической области (при этом они должны быть обязательно взяты в скобки).

$$X := \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix} \qquad Y := \begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{pmatrix} \qquad Z := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$



(X,Y,Z)

Рис. 6.27. Задание поверхности, исходя из массива данных

Конечно, описанный способ задать поверхность может мало помочь при построении сферы, однако его стоило рассмотреть для того, чтобы в дальнейшем нам было легче разбираться в более сложных методиках. Тем более что при решении практических инженерных задач он имеет огромное значение, так как данные, например при автоматическом определении дефекта детали датчиком Холла, организуются именно так, как это было сделано нами на рис. 6.27.

Способ 2. Ранжированные переменные

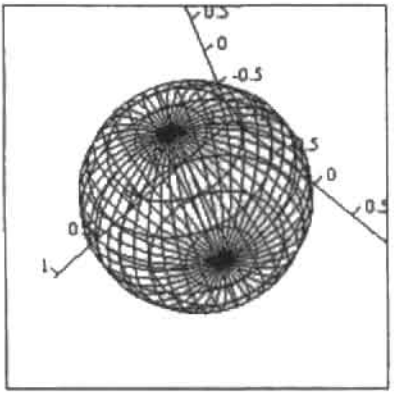
Аналогично двумерному случаю, задать поверхность можно, используя оператор ранжированной переменной (что, как вы помните, аналогично заданию циклов в программировании). Так как с этим вопросом мы уже довольно хорошо знакомы по разделу о создании X-Y-графиков, то подробно на технических деталях останавливаться не стоит и мы сразу приведем пример задания сферы с помощью двух ранжированных переменных (рис. 6.28).

$$N := 20 \qquad M := 20$$

$$i := 0..N \qquad j := 0..M$$

$$\phi_i := 2\pi \cdot \frac{i}{N} \qquad \theta_j := \pi \cdot \frac{j}{M}$$

$$X_{i,j} := \sin(\phi_i) \cdot \cos(\theta_j) \qquad Y_{i,j} := \sin(\phi_i) \cdot \sin(\theta_j) \qquad Z_{i,j} := \cos(\phi_i)$$



(X,Y,Z)

Рис. 6.28. Задание сферы с помощью ранжированных переменных

Внимательно изучив предложенный алгоритм, нельзя не согласиться, что во многом данный способ повторяет те ходы, к которым мы прибегали, когда создавали график по готовым матрицам. Однако если вы попытаетесь открыть матрицу значений одного из уравнений, вы обнаружите, что все же есть одно принципиальное отличие: данные по координатам организованы не в виде матриц, а в виде векторов (рис. 6.29).

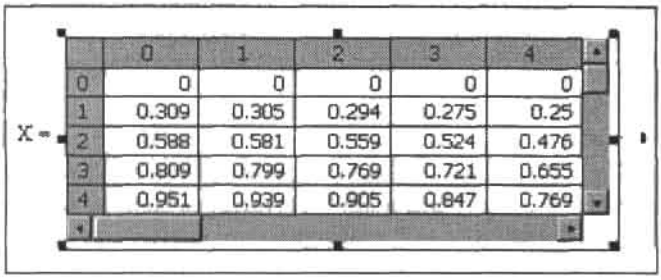
$X_{i,j}$ =

	0
20	0
21	0.309
22	0.305
23	0.294
24	0.275
25	0.25

Рис. 6.29. Вектор значений уравнения для X

Различие это связано прежде всего с формой записи имени матрицы. Если сохранить индексы (что и было сделано), то по сути мы заставляем систему вывести каждый элемент в отдельности. Сделано это будет последовательным прочтением строк (слева

направо и сверху вниз — точно так же, как вы читаете эту книгу). В результате и получается вектор, фрагмент которого приведен на рис. 6.29. Однако в общий массив мы объединяем, если вы обратили внимание, названия координат без индексов. При этом данные рассматриваются именно в виде матриц (рис. 6.30).



X =

	0	1	2	3	4
0	0	0	0	0	0
1	0.309	0.305	0.294	0.275	0.25
2	0.588	0.581	0.559	0.524	0.476
3	0.809	0.799	0.769	0.721	0.655
4	0.951	0.939	0.905	0.847	0.769

Рис. 6.30. Таблица значений координаты X для сферы

Итак, никаких принципиальных различий между такими, на первый взгляд, разными способами задания поверхности, как использование экспериментальных данных и ранжированных переменных, нет. Равно как нет их и между всеми остальными методами. В основе всех способов построения поверхности лежит одна и та же идея: создание вложенного массива, описывающего некоторую сетку разбиений. Зная это, всегда можно задать совершенно любую зависимость, даже не зная никаких специальных функций или ходов.

Объединить матрицы или уравнения в один массив можно еще до введения его в маркер графической области. Таким образом, в нашем случае массив данных можно представить в виде:

$$M := \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \qquad M = \begin{pmatrix} \{21,21\} \\ \{21,21\} \\ \{21,21\} \end{pmatrix}$$

Описанный способ имеет скорее теоретическое, нежели практическое значение в связи с тем, что в Mathcad существуют функции, значительно облегчающие построение параметрически определенных трехмерных графиков. Однако изучить его было весьма полезно для понимания тех механизмов, которые лежат в основе построения поверхностей в Mathcad. Тем более что построить таким образом несколько графиков — это лучший способ разобраться в особенностях организации данных в Mathcad, что очень важно, особенно при решении задач программными методами.

Способ 3. Использование уравнения полусферы

Аналитически сфера определяется уравнением $X^2+Y^2+Z^2=R^2$, где R — ее радиус. Однако для того, чтобы задать какую-то поверхность уравнением, оно должно иметь стандартный вид: $Z=f(X,Y)$.

В случае уравнения сферы координату Z можно выразить как

$$Z(X,Y) := \sqrt{R^2 - X^2 - Y^2}$$

или как

$$Z(X,Y) := -\sqrt{R^2 - X^2 - Y^2}$$

В первом случае уравнение задает верхнюю полусферу, во втором — нижнюю. Воспользоваться распространенным в математике сочетанием «плюс-минус» не получится, поскольку в Mathcad его никак нельзя определить. Однако выход есть: поместим на одну графическую область сразу две эти поверхности. Слившись, они дадут шар.

Последовательность действий следующая.

1. Вводим два уравнения полусфер.
2. Задаем циклы для переменных и определяем область изменения X и Y от –R до R (или сразу строим график быстрым способом и необходимые настройки делаем с помощью параметров вкладки QuickPlot окна форматирования поверхности).
3. Строим график.

Но на последнем этапе вдруг возникает проблема: поверхность не отображается и выдается сообщение об ошибке: Function must return real scalar (Функция должна возвращать действительный скаляр). В чем же дело?

А проблема заключается в следующем. Как вы помните, при создании образующей сетки каждому значению X ставятся в соответствие все возможные значения Y (и наоборот). При этом, с учетом заданной области изменения переменных, обязательно найдутся точки, сумма квадратов координат которых превысит квадрат радиуса сферы. При этом подкоренное значение станет отрицательной величиной, а так как Mathcad рассматривает все числа как комплексные, то координате Z в таких точках будет присвоено мнимое значение. Естественно, что построить при этом поверхность будет невозможно.

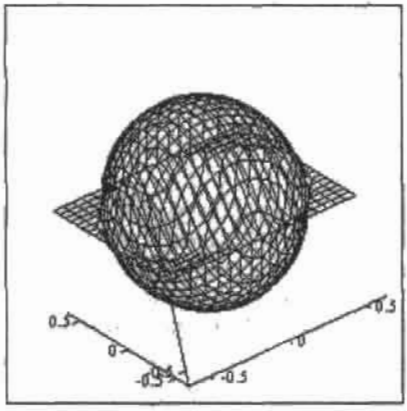
Проблему можно было бы легко решить, если бы существовала возможность перехода от прямоугольной сетки разбиений к круглой. Но, к сожалению, в рамках данной методики это невозможно. Что же делать? Проявив немного фантазии, выход находим довольно оригинальный. Попробуем скомпенсировать недостатки системы ее достоинствами: используем широкие возможности работы Mathcad с комплексными числами. Для ограничения решений действительной областью воспользуемся функцией Re (от real — действительная), которая возвращает действительную часть комплексного числа. В нашем случае ее использование даст то, что мнимые значения координаты Z будут заменены на 0.

Чтобы использовать вышеизложенную идею, уравнения полусфер переписываем в следующем виде:

$$f(X,Y) := -\mathrm{Re}\left(\sqrt{R^2 - X^2 - Y^2}\right)$$

$$f1(X,Y) := -f(X,Y)$$

Попробуем теперь построить по полученным уравнениям сферу. Сфера наша получилась с недостатком в виде прямоугольной каемки (рис. 6.31). К сожалению, избавиться от нее никак нельзя. Она образована теми точками, мнимую часть которых мы заменили на 0. Но все равно, мы получили результат там, где получить что-то довольно трудно. Поэтому данную методику можно смело отнести к способам задания сферы в Mathcad, имеющим определенное познавательное значение.



f,f1

Рис. 6.31. Сфера, построенная по уравнениям полусфер с ограничением на мнимую часть

Способ 4. Использование функции CreateMesh

Довольно громоздкую запись параметрического задания сферы, рассмотренную в качестве второго способа построения, можно заменить более короткой и простой с помощью специальной матричной функции CreateMesh (Создать сетку). Алгоритм ее использования следующий.

1. Задаем систему параметрических уравнений сферы:

$$X(\phi,\theta) := \sin(\phi)\cdot\cos(\theta)$$
$$Y(\phi,\theta) := \sin(\phi)\cdot\sin(\theta)$$
$$Z(\phi,\theta) := \cos(\phi)$$

2. Объединяем матрицы данных в один массив:

$$F(\phi,\theta) := \begin{pmatrix} X(\phi,\theta) \\ Y(\phi,\theta) \\ Z(\phi,\theta) \end{pmatrix}$$

3. Вводим функцию CreateMesh(F,s0,s1,t0,t1,sgrid,tgrid,fmap) в маркер графической области. Она имеет восемь пустых маркеров, в которые последовательно вводятся:
   - имя матрицы значений или функции F. Единственный обязательный параметр. Остальные могут быть определены как пользователем, так и заданы автоматически системой;
   - начальное значение первой переменной s0;
   - начальное значение второй переменной s1;
   - конечное значение первой переменной t0;
   - конечное значение второй переменной t1;
   - число линий сетки по первой переменной sgrid;

- количество линий сетки по второй переменной tgrid;
- карта отображения fmap. Что это такое, мы поговорим ниже.

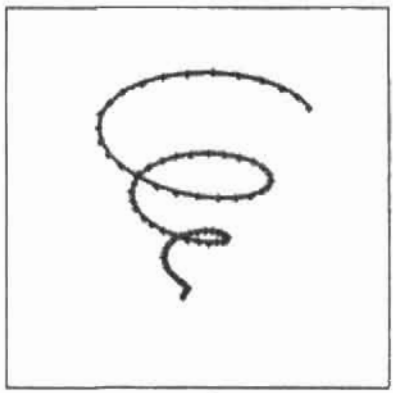
Для нашего случая параметры определяем следующим образом: CreateMesh(F,0,2π,0,π,50,50). График построен.

Согласитесь, такой способ — наиболее простой из всех рассмотренных, хотя и наименее наглядный для тех, кто не владеет системой Mathcad.

Помимо поверхностей, параметрически в пространстве можно задавать и разного рода линии. Для этого существует специальная функция CreateSpace(F,t0,t1,tgrid,fmap) (Создать пространственные). Она имеет пять маркеров, в которые последовательно вводятся имя массива данных или системы параметрических уравнений, начальное и конечное значения параметра, количество разбиений промежутка параметра, карта отображения.

В качестве примера использования функции CreateSpace приведем построение расходящейся спирали (в качестве типа графика следует выбрать Scatter Plot (График рассеяния) или Data Points (Точечный)) (рис. 6.32).

$$F(t) := \begin{pmatrix} \sin(t)\cdot t \\ \cos(t)\cdot t \\ t \end{pmatrix}$$



CreateSpace(F,0,6π,80)

Рис. 6.32. Использование функции CreateSpace

Способ 5. Параметрическое закручивание

Представим такую ситуацию: параметрических уравнений сферы вы не помните, а график с дефектами, получаемый исходя из аналитического уравнения, вас не устраивает. Что же делать? В такой сложной ситуации следует прежде всего хорошенько подумать и попробовать подойти к проблеме нестандартно. И решение будет обязательно найдено!

Если вы знаете хотя бы уравнение окружности — этого уже вполне достаточно, чтобы решить поставленную задачу не только правильно, но и очень красиво. Впрочем, с помощью этого уравнения (аналогично случаю со сферой) мы сможем построить только половину окружности. Но и этого будет достаточно!

Последовательность действий при использовании алгоритма параметрического закручивания следующая.

1. Задаем уравнение полуокружности:

$$f(x) := \sqrt{9 - x^2}$$

Для наглядности построим график определенной выше функции (рис. 6.33).

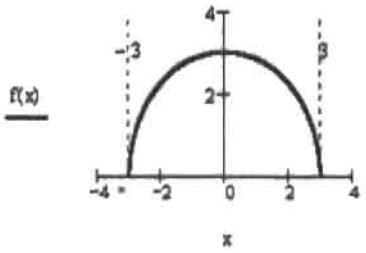


Рис. 6.33. Полуокружность

2. Задаем систему параметрического закручивания и объединяем ее в один массив:

$$A(u,v) := u$$

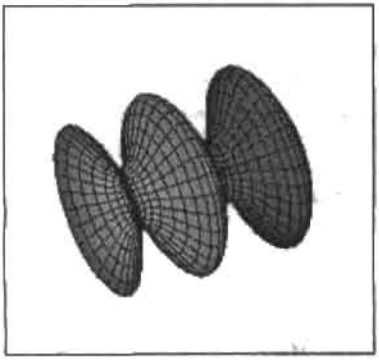
$$B(u,v) := f(u)\cdot\cos(v)$$

$$C(u,v) := f(u)\cdot\sin(v)$$

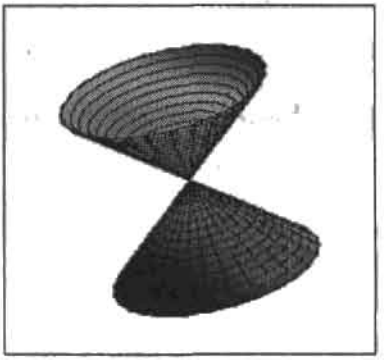
$$M(u,v) := \begin{pmatrix} A(u,v) \\ B(u,v) \\ C(u,v) \end{pmatrix}$$

3. Вносим в маркер графической области следующую запись: CreateMesh(M,-3,3,0,2π,30,30). Результат — обычная, без каких-либо дефектов или недостатков сфера.

Аналогичным образом можно построить самые разнообразные поверхности вращения. Приведем в качестве примера поверхности, образованные в результате вращения синусоиды y(x)=sin(x)+2 и прямой y(x)=x вокруг оси x (рис. 6.34).



CreateMesh(M,-π,π,0,4.3π,30,30)



CreateMesh(F,-π,π,0,4.3π,30,30)

Попробуем разобраться в идее работы алгоритма параметрического закручивания.

Функция CreateMesh по принципу своей работы практически абсолютно повторяет второй из разобранных нами способов с применением ранжированных переменных. И в случае рассматриваемого параметрического закручивания аналогично задается два цикла изменения соответствующих переменных: главный и внутренний. Шаг такого изменения определяется в самой функции заданием количества линий сетки (величины, обратной шагу). Тут же устанавливаются границы интервалов.

Первым вступает в работу главный цикл по переменной U. При этом ей присваивается первое на промежутке значение: U=–3. Затем включается внутренний цикл, ставя в соответствие первому значению U все возможные значения V. При этом заполняется первая строка матриц координат. Далее цикл повторяется до тех пор, пока U не достигнет своего максимального значения: U=3.

Проанализируем выражения для координат. Первый элемент массива M содержит матрицу из 31 строки и 31 столбца с X-координатами узловых точек. Ее определяет выражение A(u,v)=u, то есть элементы не зависят от V и поэтому они изменяются только исходя из главного цикла. А это означает, что в любой строке элементы будут одинаковыми и не будут зависеть от номера столбца. Например, в первой строке будет 31 элемент со значением –3, во второй — столько же со значением –2.8 и т. д.

Чтобы понять смысл дальнейших действий, мысленно рассеките нашу сферу плоскостью. Если не считать случаев крайних точек, получится окружность (рис. 6.35). А параметрическая система для окружности имеет следующий вид: X=Rsin(φ), Y=Rcos(φ) (или наоборот — никакой разницы от этого не будет), где φ — угол, изменяющийся от 0 до 2π.

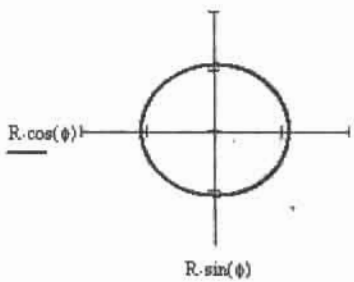


Рис. 6.35. Параметрическое задание окружности

График на рис. 6.35 показывает, что система уравнений выбрана верно. Осталось только определиться с R. А радиусом такой окружности будет не что иное, как значение функции в выбранной точке области X. В случае нашей задачи для каждого значения U (а как было показано выше, в ходе работы внутреннего цикла по V значение U не меняется) с помощью внутреннего цикла изменения V как бы прорисовываются подобные окружности (реально же просто ставятся в соответствие друг другу элементы матриц массива данных с одинаковыми индексами). В результате получается шар.

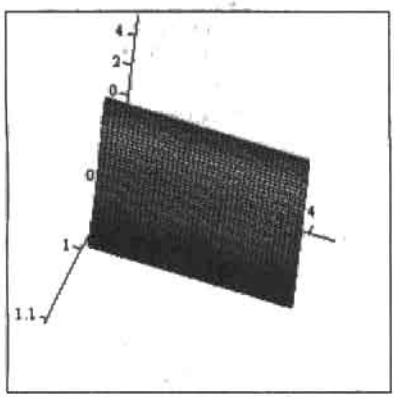
Как видите, все очень просто в том случае, если хорошо понимать, по каким принципам строятся поверхности в Mathcad.

Способ 6. Преобразование системы координат

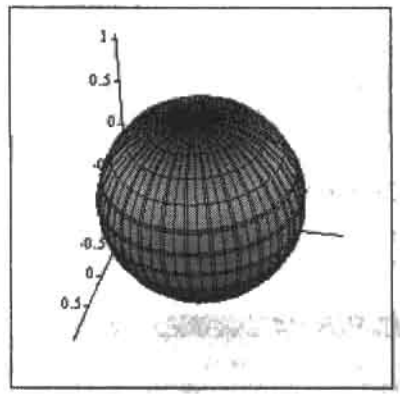
Последним мы рассмотрим самый простой и быстрый способ построения сферы. Зададим быстрым методом плоскость, а затем преобразуем ее из декартовой системы координат в сферическую (рис. 6.36), выбрав пункт Spherical (Сферическая) меню Coordinate

System (Система координат) на вкладке Quick Plot Data (Данные графика быстрого построения) окна форматирования трехмерных графиков 3D-Plot Format.

$$M(x,y) := \begin{pmatrix} 1 \\ x \\ y \end{pmatrix}$$



M

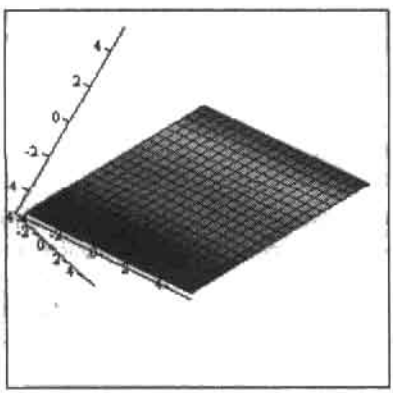


M

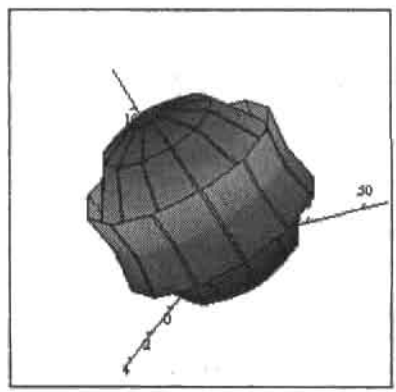
Рис. 6.36. Задание сферы преобразованием системы координат

Построить поверхность можно в совершенно любой системе координат. Для этого нужно воспользоваться функцией CreateMesh, задав параметр fmap самостоятельно как функцию. Пример того, как это можно сделать, приведен на рис. 6.37.

$$z(\phi,\theta) := \begin{pmatrix} \phi \\ \theta \\ \phi \end{pmatrix} \qquad FMAP(x,y,r) := \begin{pmatrix} r\cdot\sin(x)\cos(y) \\ 15\cdot r\sin(x)\cdot\sin(y) \\ 5\cdot r\cos(x) \end{pmatrix}$$



CreateMesh(z)



CreateMesh(z,FMAP)

Рис. 6.37. Построение поверхности в произвольной системе координат

Как вы, наверное, убедились, способов задания поверхности можно придумать много. Однако очень широки также возможности форматирования вида полученных трехмерных графиков. Этому вопросу посвящен следующий подраздел.

6.2.2. Форматирование 3D-графиков

Для настройки вида поверхностей, аналогично двумерным графикам, существует специальное диалоговое окно 3D-Plot Format (Форматирование 3D-графиков). С содержанием одной вкладки данного окна мы уже познакомились, когда обсуждали изменение параметров графика быстрого построения. Однако, помимо вкладки QuickPlot Data (Данные графика быстрого построения), окно 3D-Plot Format (Форматирование 3D-графиков) содержит еще восемь (!) вкладок, отвечающих за настройку разнообразных параметров вида трехмерных графиков. Разберем их содержимое и назначение по порядку.

Вкладка General (Общие)

Вкладка содержит наиболее общие и часто используемые параметры настройки вида графика и системы координат (рис. 6.38).

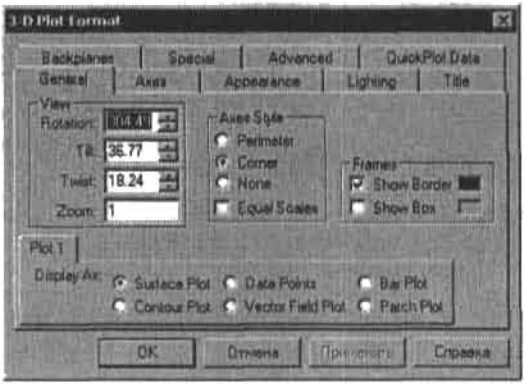


Рис. 6.38. Вкладка General (Общие)

За расположение графика в пространстве отвечает меню View (Вид). Углы поворота поверхности относительно стандартно расположенной системы координат отображаются в окошках трех параметров.

- Rotation (Вращение).
- Tilt (Наклон).
- Twist (Кручение).

Меняя угол для данных параметров, можно добиться наиболее удобного расположения изображения. Однако на практике обычно поступают по-другому. Поменять расположение вашей поверхности можно и с помощью мыши. Для этого при нажатой правой кнопке манипулятора «протащите» график в нужном направлении. Таким образом можно расположить поверхность так, как вы хотите, причем делается это очень легко. Кстати, при желании вы можете заставить вашу поверхность вращаться в нужном направлении. Для этого следует выполнить точно такое же протаскивание, как и при изменении ее положения в пространстве, только при нажатой клавише Shift. Скорость вращения будет зависеть от того, насколько резко вы выполните это движение.

С помощью параметра Zoom (Масштаб) меню View (Вид) вы можете изменить масштаб отображения вашей поверхности, причем это можно сделать как в сторону увеличения, так и в сторону уменьшения. Нормальный размер графика, при котором он занимает всю внутреннюю рамку графической области, принят за единицу. Соответственно, чтобы увеличить масштаб в два раза, в окошко параметра Zoom (Масштаб) следует ввести 2. Чтобы уменьшить поверхность в два раза, единицу заменяем на 0.5. На практике же удобнее изменять масштаб протаскиванием графика левой кнопкой мыши при нажатой клавише Ctrl (или же использовав колесо прокрутки мыши).

Меню Axes Style (Стиль осей) вкладки General (Общие) содержит настройки, отвечающие за тип отображения системы координат. Возможны следующие варианты (рис. 6.39).

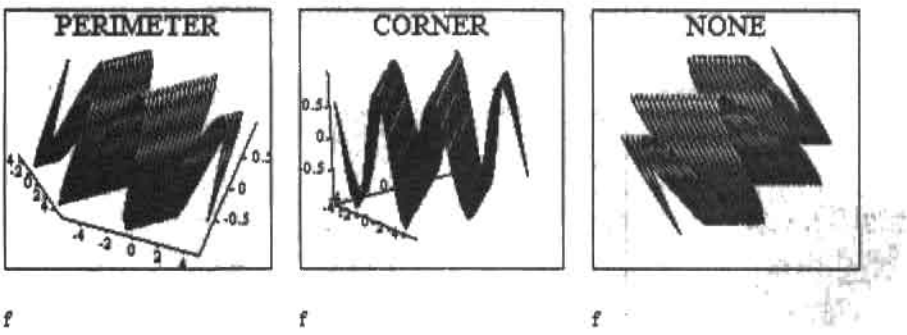


Рис. 6.39. Варианты отображения системы координат

- Perimeter (Периметр). Взаимное расположение осей не статично. То есть при изменении положения поверхности меняется и точка пересечения осей так, чтобы они были наиболее доступны наблюдению.
- Corner (Угол). Оси пересекаются в точках наименьших значений шкал, и их положение не меняется при повороте поверхности. Параметр, определенный по умолчанию.
- None (Нет). График отображается без осей. В принципе, в большинстве случаев пересекающиеся оси портят вид поверхности, поэтому, если функциональной необходимости в них нет, их лучше убрать.
- Equal Scales (Равные шкалы). Если вы установите этот флажок, то оси будут отображены в равном масштабе. Параметр важен в том случае, если поверхность должна быть отображена без искажений (например, при задании сферы).

С помощью меню Frames (Рамки) можно задать наличие или отсутствие двух элементов оформления.

- Show border (Показать границу). Параметр отвечает за отображение внутренней рамки графической области. По умолчанию включен. Цвет рамки может быть задан произвольным образом на палитре левее окошка параметра.
- Show Box (Показать коробку). Параметр окаймляет область поверхности кубической рамкой. Цвет рамки также может быть выбран непосредственно пользователем.

В меню Display As (Отобразить как) вы можете определить, графиком какого типа должна быть задана ваша поверхность. Всего имеется шесть типов трехмерных графиков, пять из которых могут быть заданы непосредственно командами меню Graph (Графические). Однако использовать для каждого отдельного типа графика специальную графическую область нет особого смысла, так как все они предельно легко переводятся

друг в друга командами данного меню. Правда, перевести поверхность в векторное поле таким образом вряд ли удастся, поскольку форма задания у них разная. Созданию векторного поля мы посвятим отдельный подраздел.

Как уже говорилось выше, имеется шесть типов отображения трехмерных объектов.

- Surface (Поверхность). Наиболее важный и часто используемый тип трехмерного графика. Все зависимости, с которыми мы сталкивались до этого, были построены именно по этому типу.
- Contour Plot (Контурный график). Представляет собой проекцию поверхности на плоскость XOY. Очень полезен при изучении поведения функций двух переменных (поверхности зачастую бывают трудны для изучения).
- Data Points (Точки данных). На данном типе графика отображаются только узловые точки (рис. 6.40).

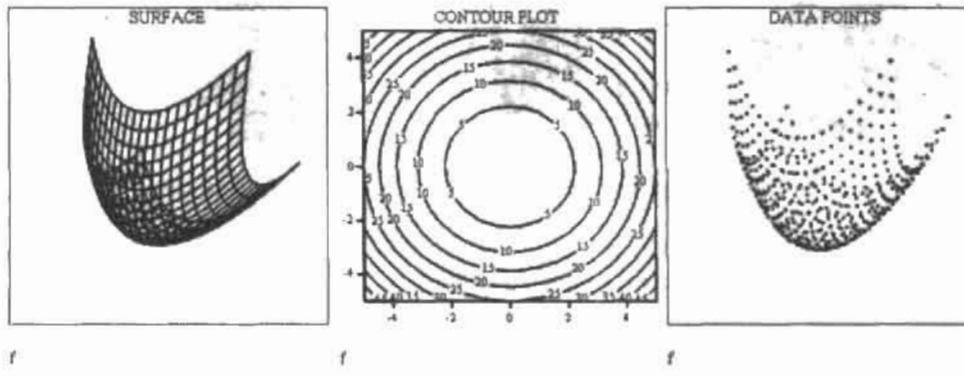


Рис. 6.40. Типы графиков Surface, Contour Plot, Data Points

- Vector Field Plot (Векторное поле). Очень необычный тип графика, имеющий большое значение при решении физических задач.
- Bar Plot (Диаграмма). Трехмерный график в виде столбчатой диаграммы. Площадь основания столбика определяется величиной ячейки сетки разбиения.
- Patch Plot («Кусочечный» график). Тип графика, представляющий собой нечто среднее между поверхностью и точечной зависимостью (рис. 6.41).

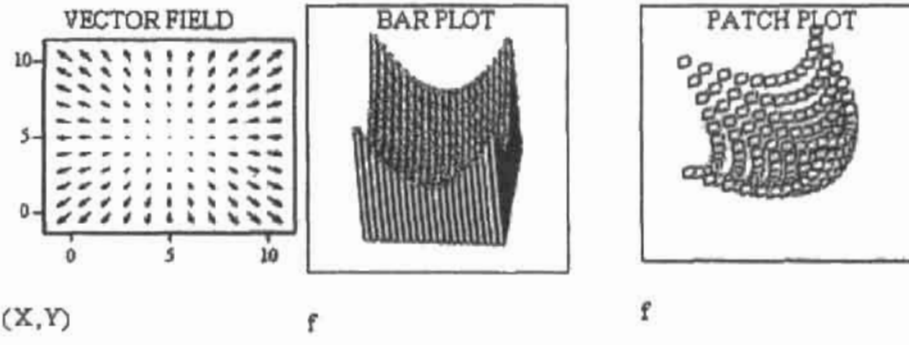


Рис. 6.41. Типы графиков Vector Field, Bar Plot, Patch Plot

Рассматривать по отдельности форматирование каждого типа трехмерной зависимости мы не будем, так как, за исключением некоторых деталей, настройка вида производится абсолютно идентично для любого из них. Поэтому все особенности форматирования 3D-графиков мы разберем на примере наиболее часто используемого типа трехмерного изображения – поверхности.

Вкладка Backplanes (Задний план)

Данная вкладка содержит параметры форматирования вида заднего плана вашего графика. Под этим термином понимаются три противоположные наблюдателю грани виртуального куба, в котором находится поверхность. Соответственно вкладка Backplanes (Задний план) содержит три вкладки, отвечающие за настройку вида каждой из граней: X-Y, Y-Z, X-Z Backplane (рис. 6.42). По своему содержанию данные вкладки абсолютно идентичны.

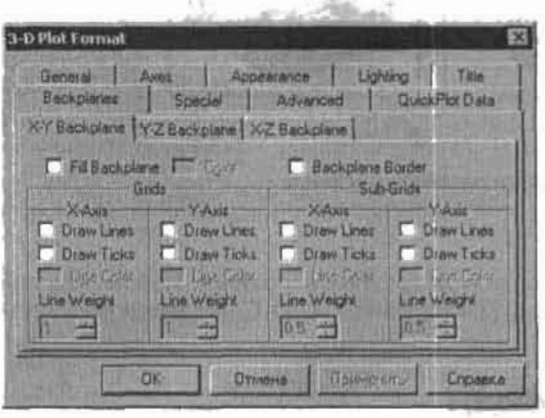


Рис. 6.42. Вкладка Backplane (Задний план)

За визуализацию непосредственно выбранного заднего плана отвечают два параметра.

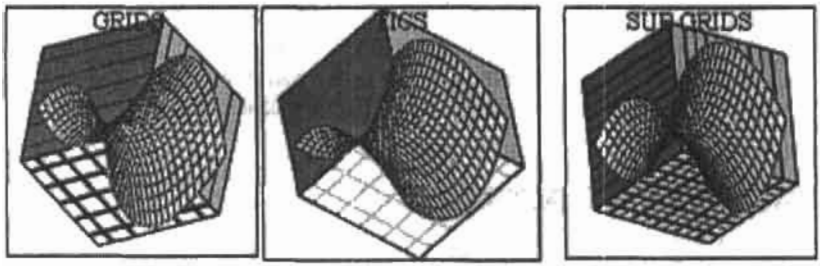
- Fill Backplane (Залить задний план). Установив данный флажок, вы сможете выбрать на соответствующей палитре наиболее подходящий цвет для заднего плана вашей поверхности.
- Backplane Border (Граница заднего плана). С помощью данного параметра вы можете окаймить одну из граней заднего плана рамкой.

Расположенное чуть ниже параметра Fill Backplane (Залить задний план) меню Grids (Вспомогательные линии) содержит параметры отображения линий вспомогательной сетки. Меню разбито на два подменю, каждое из которых содержит параметры визуализации вспомогательных линий по одной из двух ограничивающих грань осей. Содержание подменю одинаково для всех осей и образовано четырьмя следующими параметрами.

- Draw Lines (Рисовать линии). Установив этот флажок, вы визуализируете сетку из вспомогательных линий по выбранной оси (рис. 6.43, слева).
- Draw Ticks (Рисовать метки). Параметр отвечает за отображение меток в начале вспомогательных линий (рис. 6.43, в центре).
- Line Color (Цвет линии). На этой палитре вы можете выбрать наиболее подходящий для вашего графика цвет вспомогательных линий.

- Line Weight (Толщина линий). В данном списке можно задать толщину линий вспомогательной сетки.

Расположенное справа от рассмотренного меню Grids (Вспомогательные линии) меню Sub Grids (Дополнительные вспомогательные линии) отвечает за отображение дополнительной вспомогательной сетки, линии которой будут проведены между основными вспомогательными линиями (рис. 6.43, справа). По своему содержанию данное меню полностью повторяет меню Grids (Вспомогательные линии).



f f f

Рис. 6.43. Варианты отображения сетки из вспомогательных линий

Вкладка Axes (Оси)

Данная вкладка содержит параметры вида осей системы координат. Соответственно для каждой из трех осей есть своя вкладка (X-Axis, Y-Axis, Z-Axis). Содержание всех вкладок идентично (рис. 6.44).

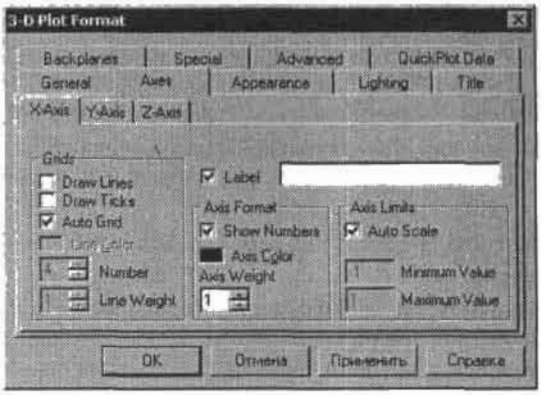


Рис. 6.44. Вкладка Axis (Оси)

На вкладке расположено одноименное и практически аналогичное уже рассмотренному нами меню Grids (Вспомогательные линии) вкладки Backplanes (Задний план) меню. Правда, есть одно существенное отличие в том, каким образом проводятся вспомогательные линии при использовании команды Draw Lines (Рисовать линии) меню Grids (Вспомогательные линии) этих двух вкладок. Если вы используете параметр рассматриваемой вкладки, то линии будут прочерчены перпендикулярно оси через две грани заднего плана.

Меню Grids (Вспомогательные линии) содержит параметр Number (Количество), позволяющий произвольным образом задавать количество линий сетки для выбранной оси. Чтобы это сделать, отключите параметр Auto grid (Автоматическая сетка) и введите нужное число в окошко параметра Number (Количество). Кстати, проведенные изменения будут восприниматься и меню Grids (а также Sub Grids) вкладки Backplanes (Задний план).

Непосредственно вид оси можно настроить с помощью параметров меню Axis Format (Формат оси).

- Show Numbers (Показать нумерацию). Параметр отвечает за отображение нумерации на осях.
- Axis Color (Цвет оси). На данной палитре вы можете выбрать цвет осей вашего графика.
- Axis Weight (Толщина оси). Этот параметр определяет толщину выбранной оси. Может изменяться как в сторону увеличения, так и в сторону уменьшения (1 — стандартная толщина, 0.1 — минимум, 10 — максимум).

Меню Axis Limits (Пределы осей) данной вкладки определяет диапазон построения поверхности по выбранной оси.

- Auto Scale (Автоматическая шкала). Пределы построения по данной оси определяет автоматически система. Сняв данный флажок, вы сможете самостоятельно задать диапазон вашего графика, введя нужные значения в строки параметров Minimum Value (Минимальная величина) и Maximum Value (Максимальная величина).

Чтобы создать подпись к оси, установите флажок Label и справа от него введите необходимый текст.

Вкладка Special (Специальные)

На данной вкладке (рис. 6.45) расположены те настройки некоторых специальных графиков, которые не являются универсальными для всех типов трехмерных изображений и поэтому не отражены на других вкладках.

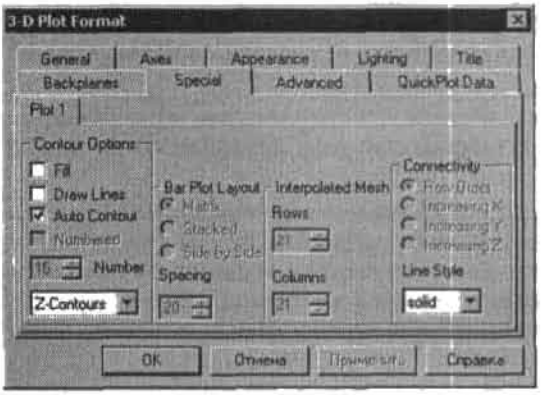


Рис. 6.45. Вкладка Special (Специальные)

Меню Contour Options (Параметры контура) предназначено, прежде всего, для форматирования вида контурного графика (однако с учетом того, что линии уровня есть и на поверхностях, его можно использовать для редактирования вида последних). Это меню содержит следующие параметры.

- Fill (Залить). График будет залит исходя из цветов выбранной палитры (как можно поменять палитру, мы поговорим, когда будем разбирать вкладку Advanced (Дополнительные)). При этом пространство между двумя линиями уровня будет закрашено одним оттенком.
- Draw Lines (Рисовать линии). Параметр отвечает за отображение на графике линий уровня.
- AutoContour (Автоконтур). Если данный флажок установлен, то количество линий уровня вашего графика будет определено системой автоматически. Убрав его, вы сможете задать количество линий уровня произвольным образом, изменив значение параметра Number (Количество) (рис. 6.46, слева и в центре).
- Numbered (Пронумерованные). С помощью данного параметра вы сможете визуализировать нумерацию линий уровня вашего графика (рис. 6.46, справа).

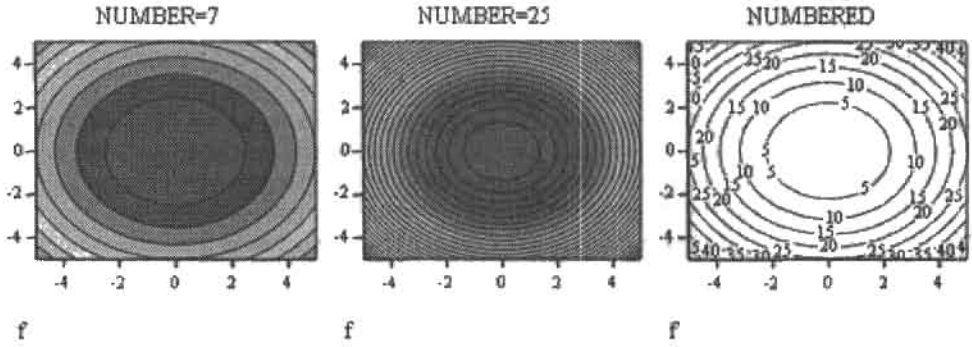


Рис. 6.46. Форматирование вида контурного графика

В списке в конце рассматриваемого меню можно определить, вдоль какой оси проводятся линии уровня. По умолчанию это делается вдоль оси Z (Z-Contours).

С помощью параметров меню Bar Plot Layout (Оформление столбчатой диаграммы) можно определить тип диаграммы, исходя из которого она должна быть визуализирована. Всего таких типов три.

- Matrix (Матрица). С этим типом диаграммы мы уже сталкивались, когда разбирали вкладку General (Общие). В общем полностью повторяет поведение соответствующей поверхности.
- Stacked (Соединенные). Задать данный тип диаграммы с помощью функции нельзя (да и не имеет смысла). Он строится исходя из матрицы значений. При этом каждый вектор визуализируется в виде столбца, разделенного на участки по количеству элементов в векторе. Длина участка определяется величиной соответствующего ему элемента. Если какой-то элемент оказывается отрицательным, то участок столбца вектора строится ниже нулевого уровня.
- Side by Side (Шаг за шагом). Идея данной диаграммы та же, что и в предыдущем случае, только участки столбца не состыкованы, а располагаются рядом (рис. 6.47).

$$M := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ -7 & 8 & 9 \end{pmatrix}$$

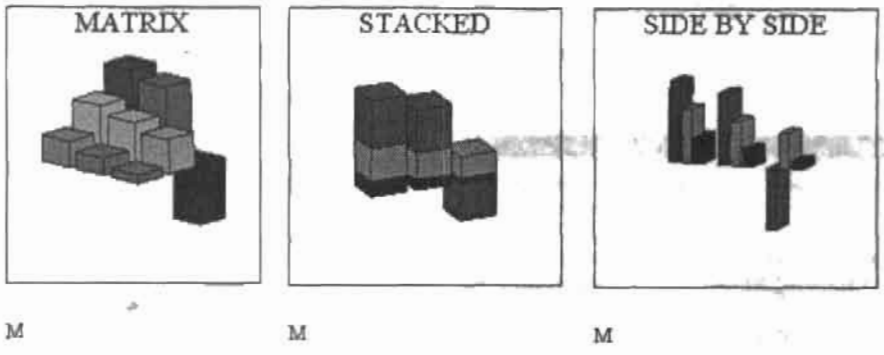


Рис. 6.47. Типы столбчатых диаграмм

Вы, наверное, обратили внимание на то, что столбцы диаграммы не прилегают плотно друг к другу. Величину расстояния, разделяющего столбцы, можно регулировать с помощью параметра Spacing (Разделение) рассматриваемого меню. Цифра в его окошке означает, какой процент от площади области построения графика занимает пространство между столбцами. Соответственно она может меняться от 0 до 100.

Меню Interpolated Mesh (Сетка интерполяции) рассматриваемой вкладки служит для настройки вида поверхности, которая строится по экспериментальным данным. Более подробно о такого рода интерполяции мы поговорим в гл. 16.

Меню Connectivity (Соединение) вкладки Special (Специальные) полезно в случае построения точечного графика. При его помощи можно определить, каким образом будут соединяться линиями точки графика. Возможны четыре варианта.

- Row Order (По порядку). Точки будут соединены по такому же принципу, как они заносились в массив. То есть матрица будет прочитана слева направо и сверху вниз (рис. 6.48).

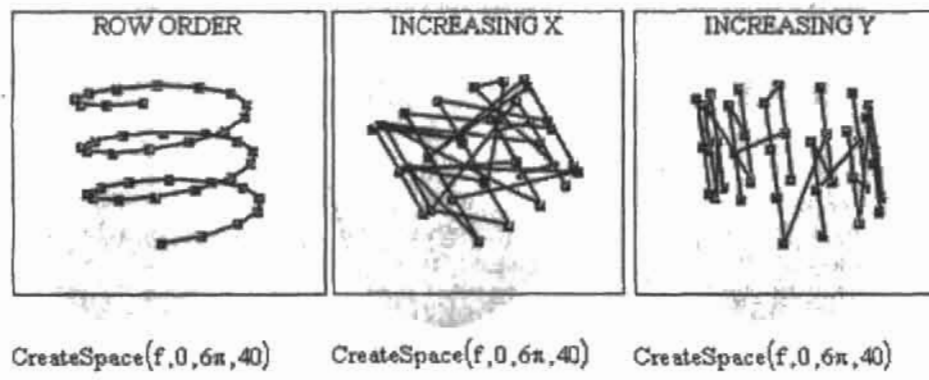


Рис. 6.48. Типы соединений при построении точечного графика

- Increasing X, Y, Z (По возрастанию X, Y, Z). Точки будут соединены исходя из возрастания координаты X, Y или Z.

Также в данном меню, в списке Line Style (Стиль линий), вы можете определить тип линии, которой соединяются узловые точки вашего графика.

Вкладка Appearance (Вид)

На данной вкладке расположены наиболее важные параметры художественного оформления поверхностей (рис. 6.49).

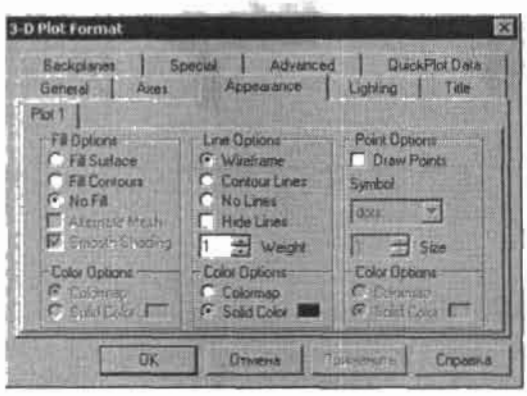


Рис. 6.49. Вкладка Appearance (Вид)

С помощью команд меню Fill Options (Параметры заливки) вы можете закрасить прозрачный каркас, получаемый при построении поверхности, выбранным цветом или цветовой гаммой. Данное меню содержит следующие команды (рис. 6.50).

- Fill Surface (Залить поверхность). Заливка поверхности по этому типу характеризуется тем, что индивидуально заливается каждая ячейка сетки разбиений, причем окрашена она может быть не только одним цветом, но и целой цветовой гаммой. Очень эффектный и полезный параметр.
- Fill Contours (Залить контуры). Поверхность будет залита уровнями, причем каждый уровень будет окрашен в свой цвет (если заливка производится палитрой).
- No Fill (Без заливки). Обычный каркас трехмерного графика. Параметр, определенный по умолчанию.

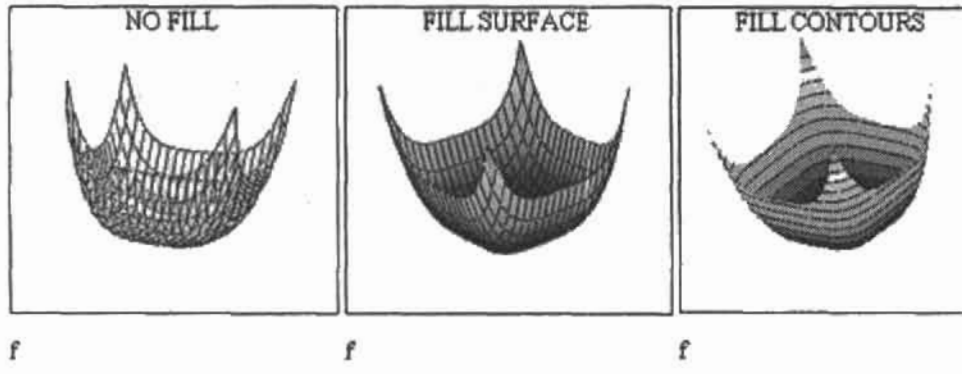


Рис. 6.50. Типы заливки поверхности

Используя параметры подменю Color Options (Параметры цвета), можно определить, каким образом будет произведена окраска поверхностей. Возможны два варианта.

- Solid Color (Постоянный цвет). При выборе данного параметра поверхность будет залита одним оттенком, выбрать который можно на палитре рассматриваемого параметра.
- Colormap (Палитра). Поверхность закрашивается палитрой, в которой каждый цвет соответствует некоторому значению функции. Палитр в Mathcad несколько. О том, как можно выбрать из них наиболее подходящую, мы поговорим, когда будем разбирать вкладку Advanced (Дополнительные).

Помимо рассмотренных, в меню Fill Options (Параметры заливки) есть два очень интересных и полезных параметра.

- Alternate Mesh (Альтернативная сетка). Если вы установите данный флажок, то прямоугольники сетки разбиений будут разделены на две части. Одна половина будет закрашена, а одна останется прозрачной. В результате поверхность приобретает очень необычный ажурный вид.
- Smooth Shading (Сглаживание). Если данный параметр задействован, между ячейками поверхности цветовой переход осуществляется постепенно. В противном случае каждая ячейка будет залита индивидуальным цветом.

Следующее меню вкладки Appearance (Вид) — Line Options (Параметры линии) отвечает за вид отображаемых линий поверхности (рис. 6.51).

- Wireframe (Каркас). На графике отображается образующая сетка. Параметр, определенный по умолчанию.
- Contour Lines (Линии контура). На графике визуализируются линии уровня.
- No Lines (Нет линий). График отображается без какой-либо вспомогательной сетки.
- Hide Lines (Спрятать линии). Параметр доступен в том случае, если над графиком не производилось операции заливки. Установив данный флажок, вы как бы зальете каркас белым цветом, сделав поверхность непрозрачной.

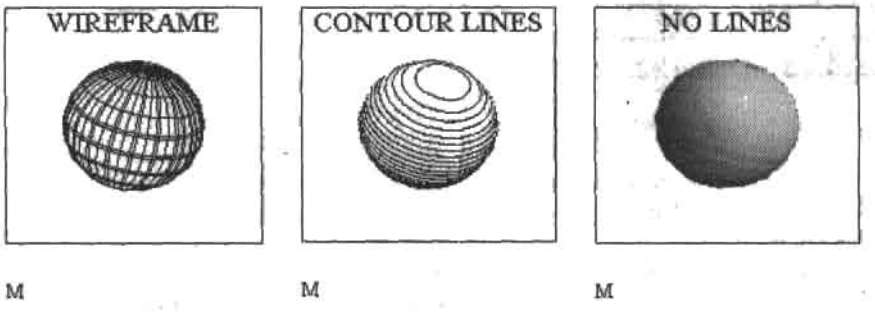


Рис. 6.51. Варианты отображения линий поверхности

Изменить размер линий поверхности можно с помощью параметра Weight (Толщина) рассматриваемого меню.

В подменю Color Options (Параметры цвета) вы можете выбрать наиболее подходящий цвет для вспомогательной сетки вашей поверхности.

В меню Point Options (Настройки точек) вкладки Appearance (Вид) содержатся параметры отображения узловых точек поверхности. По умолчанию узловые точки на трехмерном графике не визуализируются. Чтобы их отобразить, требуется установить флажок Draw Points (Рисовать точки).

В списке Symbol (Символ) данного меню вы можете определить, каким символом будут обозначены узловые точки вашей поверхности. Возможны следующие варианты:

- dots (Точки) – узловые точки отображаются в виде маленьких шариков;
- x's – символ точки в виде двух пересекающихся под острым углом маленьких отрезков;
- +'s – символ в виде трех пересекающихся под прямым углом отрезков;
- boxes (Кубы) – точки отображаются в виде маленьких кубиков;
- diamonds (Бриллианты) – по форме символ точки стилизован под бриллиант.

Размер символа точки можно изменить, используя параметр Size (Размер).

Аналогично уже рассмотренным меню, для того чтобы изменить цвет узловых точек, следует обратиться к подменю Color Options (Параметры цвета).

Вкладка Advanced (Дополнительные)

Данная вкладка (рис. 6.52) содержит наиболее удивительные и технически «продвинутые» параметры вида трехмерного графика.

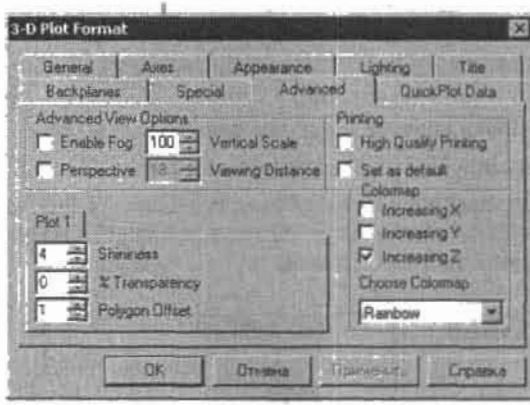
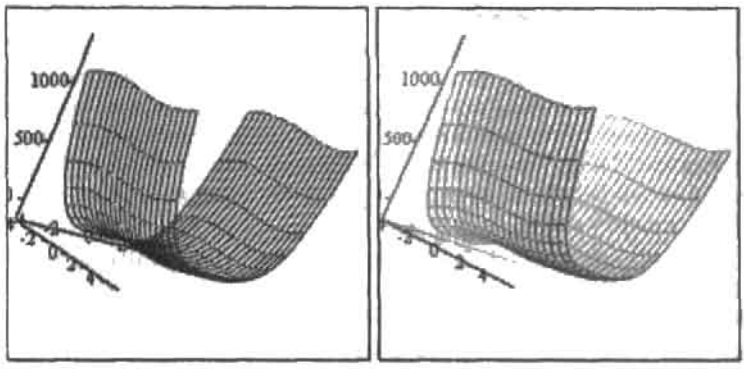


Рис. 6.52. Вкладка Advanced (Дополнительные)

Очень красивого эффекта графика, окутанного туманом (рис. 6.53), можно добиться, задействовав параметр Enable Fog (Туман). Пожалуй, это самый необычный графический фильтр в системе.

Правее рассмотренного параметра Enable Fog (Туман) расположено окошко параметра Vertical Scale (Вертикальная шкала). С помощью данного параметра можно произвольным образом сжать поверхность вдоль оси Z. Величина такого сжатия определяется в процентах в специальном окошке рассматриваемого параметра: 100 – график нормального вида, 50 – график сжат наполовину, 0 – поверхность превращается в плоскость.

Еще одной удивительной возможностью построения поверхностей в Mathcad является учет такой особенности человеческого зрения, как перспектива. Чтобы использовать эту возможность, следует установить флажок Perspective (Перспектива) рассматриваемой вкладки. При этом те участки графика, которые расположены дальше от наблюдателя, будут уменьшены пропорционально своему удалению. Располагающийся в той же строке, что и Perspective (Перспектива), параметр Viewing Distance (Дистанция наблюдения) определяет условное расстояние между наблюдателем и поверхностью.

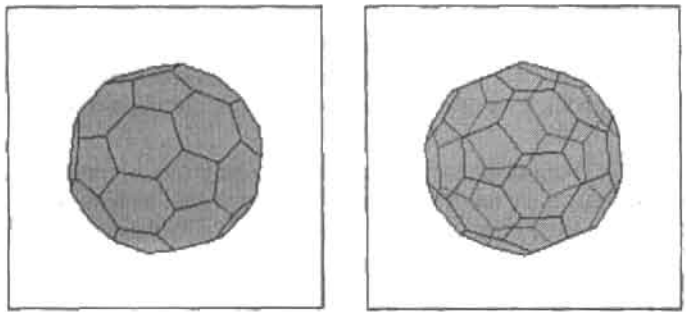


f f

Рис. 6.53. Эффект тумана

Чуть ниже рассмотренного меню Advanced view options (Дополнительные параметры вида) расположено меню Plot 1 (График 1), содержащее три настройки вида графика с заливкой.

- Shininess (Яркость). Параметр определяет степень отражения прямого (Specular) света от поверхности графика. Если степень отражения достаточно велика, то на графике может проявиться такой оптический эффект, как блик (light spot). Этот эффект может очень эффективно подчеркнуть особенности формы поверхности в том случае, если нежелательно отображение сетки. Параметр Shininess (Яркость) может изменяться от 0 до 128.
- Transparency (Прозрачность). С помощью этого параметра можно задать степень прозрачности графика (рис. 6.54). По умолчанию заливка поверхности абсолютно непрозрачна (Transparency=0), однако, в принципе, ее можно сделать полностью прозрачной (Transparency=100), правда, при этом она станет невидимой.



Polyhedron("truncated icosahedron") Polyhedron("truncated icosahedron")

Рис. 6.54. Вид многогранника при различной степени прозрачности его заливки

- Polygon Offset (Выделение многоугольника). Данный параметр определяет степень выделенности сетки поверхности на фоне заливки. Может изменяться от 0 до 10. При значении 0 создается эффект погруженности линий графика в окружающую заливку.

Меню Colormap (Палитра) содержит параметры, отвечающие за особенности заливки поверхности цветовой палитрой. Три первых параметра данного меню (Increasing X, Y, Z (По возрастанию X, Y, Z)) определяют, вдоль какой из осей будет произведена заливка. В рассматриваемом списке можно установить одновременно два или даже все три флажка. В первом случае заливка будет произведена вдоль линии, проходящей через начало координат под углом 45° к выбранным осям. Во втором случае заливка будет произведена вдоль линии, проходящей также через начало координат и под углом 45°, однако одновременно ко всем трем осям.

В Mathcad встроено довольно значительное количество цветовых палитр (10), выбрать из которых наиболее подходящую можно в меню Choose Colormap (Изменить палитру). По умолчанию поверхности заливаются палитрой Rainbow (Радуга).

Кстати, палитру можно задать и произвольным образом. Для этого следует использовать специальную функцию SaveColormap("Name", M). Name — это имя, под которым создаваемая палитра будет отображена в соответствующем списке, M — специальным образом организованная матрица, определяющая цветовую гамму в формате RGB. Матрица должна содержать три столбца: в первом задается интенсивность красного цвета, во втором — зеленого, в третьем — голубого. Интенсивность может быть определена числом от 0 до 255. Сочетаясь, эти три оттенка дают все остальные возможные в формате RGB (всего таким образом может быть образовано около полутора миллионов оттенков). В каждой строке матрицы палитры может быть задан один оттенок. Количество же таких строк, в принципе, не ограничивается. Поэтому отображение палитры производится таким образом, чтобы распределить равномерным образом все заданные цвета на поверхности графика.

Чтобы задать свою палитру, выполните следующую последовательность действий.

1. Задайте матрицу размерности N×3, содержащую элементы со значениями от 0 до 255 (если вы когда-нибудь работали с графикой, то для вас не составит никакого труда выполнить эту работу так, чтобы сочетание цветов было более или менее гармоничным).
2. Введите функцию SaveColormap("Name", M).
3. Заполните необходимые параметры в имени заданной функции (имя палитры должно быть занесено как строка в кавычках).
4. Поставьте «=» после функции SaveColormap. При этом созданная палитра будет занесена в список Choose Colormap (Выбрать палитру), а непосредственно на листе будет выведено количество строк в матрице палитры. Кстати, палитра сохраняется в специальном формате — CMP.

Чтобы залить поверхность созданной палитрой, просто выберите ее в списке Choose Color. Кстати, сохраняются созданные палитры (рис. 6.55) в данном списке не только на время создания одного документа, так что доступ к ним вы будете иметь постоянно.

$$M := \begin{pmatrix} 125 & 225 & 1 \\ 125 & 225 & 1 \\ 125 & 225 & 1 \end{pmatrix} \qquad M1 := M^T \qquad M2 := M + M1$$

SaveColormap("User" , M) = 3

SaveColormap("User1" , M1) = 3

SaveColormap("User2" , M2) = 3

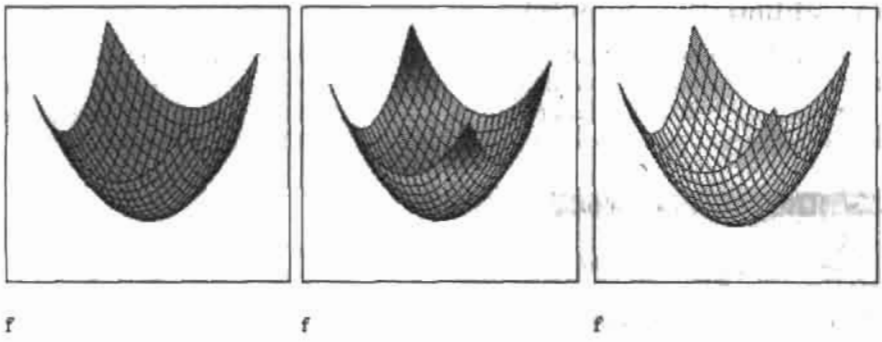


Рис. 6.55. Создание палитр

Черно-белый формат данной книги практически не позволяет показать возможности цветового оформления поверхностей, в том числе и созданными нами на рис. 6.55 палитрами. Так, хотя и выглядят три поверхности практически одинаково, на самом деле первая из них темно-зеленая, вторая – серая, третья – желто-салатовая.

Создать новую палитру можно и с помощью некоторых изменений в старой. Для этого ее нужно открыть с помощью специальной функции `LoadColormap("Name")`, где `Name` – имя редактируемой палитры. Пример такого редактирования приведен на рис. 6.56.

M := LoadColormap("Fire") $N^{\langle 0 \rangle} := M^{\langle 2 \rangle}$ $N^{\langle 1 \rangle} := M^{\langle 1 \rangle}$ $N^{\langle 2 \rangle} := M^{\langle 0 \rangle}$

M =

	0	1	2
0	0	0	0
1	144	10	229
2	147	9	227

N =

	0	1	2
0	0	0	0
1	229	10	144
2	227	9	147

SaveColormap("Fire1", N) = 256

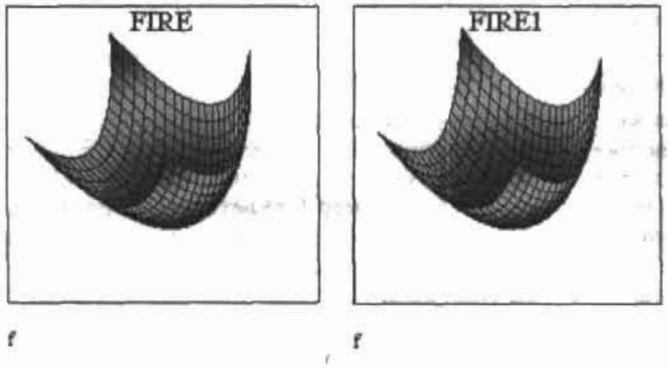


Рис. 6.56. Редактирование палитры

С помощью меню Printing (Печать) вкладки Advanced (Дополнительные) можно задать параметры распечатки созданных поверхностей.

Вкладка Lighting (Освещение)

Самой интересной и необычной возможностью художественного оформления поверхностей является возможность применения эффекта освещения. При этом на графике появятся тени и полутени, блики, световые переходы различного рода. Расположены же параметры освещения на специальной вкладке Lighting (Освещение) (рис. 6.57).

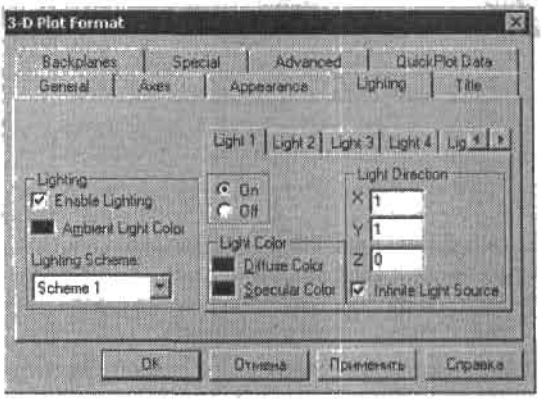


Рис. 6.57. Вкладка Lighting (Освещение)

По умолчанию освещение поверхности не производится. Чтобы его задействовать, установите флажок Enable Lighting (Освещаемый).

Освещение графиков может осуществляться двояко. Оно может производиться одинаково со всех сторон. Чтобы реализовать такую схему, замените в окошке Ambient Light Color (Цвет окружающего света) черный цвет на любой другой. При этом проявятся эффекты светотени, однако окраска самого графика принципиально не изменится (если только в качестве заливки не был выбран белый (или любой светлый) оттенок — в этом случае поверхность будет окрашена исходя из цвета окружающего света).

Вторым возможным способом освещения графиков в Mathcad является имитация точечных источников. Все параметры таких источников располагаются на вкладках Light N (Источник N). Рассмотрим эти параметры.

- Прежде всего, чтобы задействовать любой источник, поставьте метку в окошко On (Включить) выбранной вкладки Light N (Источник N).
- Положение источника можно задать в меню Light Direction (Расположение света). Это положение определяется относительно трех осей: если в окошке одной из них стоит 1, а в окошках двух остальных 0, то источник будет расположен на отмеченной единицей оси, если для двух осей выбран параметр 1, то источник будет расположен на линии, проходящей под углом 45° между этими двумя осями. И, наконец, если 1 стоит в окошках всех осей, то источник будет расположен на линии, проходящей под равным углом ко всем трем осям. Если в окошках всех трех осей стоит 0, то освещение данным источником не производится. Обычно используются три активных источника освещения, однако, в принципе, можно активировать гораздо большее их количество.
- Свет, которым освещается поверхность, может быть двух типов (рис. 6.58).
  - Diffuse Color (Диффузный свет). Такой свет создается бесконечно удаленным точечным источником. В общем случае при таком освещении интенсивность све-

тового потока, падающего на каждую точку малой, расположенной перпендикулярно линии, соединяющей ее центр и источник, площадки, будет одинаковым. При таком освещении проявляются эффекты теней и полутеней, однако нет бликов.

- Specular Color (Зеркальный свет). Эффект освещения, получаемого от источника, расположенного на сопоставимом с размером поверхности расстоянии. Пример из жизни — прожектор освещения в театре. Свет падает круглым пятном, причем интенсивность его уменьшается от центра к краям. При этом для хорошо отражающих свет объектов проявляется такой оптический эффект, как блик. Степень яркости заливки поверхности определяется параметром Shininess (Яркость) вкладки Advanced (Дополнительные). По умолчанию освещение таким типом света не производится. Чтобы его задействовать, замените черный цвет в палитре рассматриваемого параметра на любой другой.

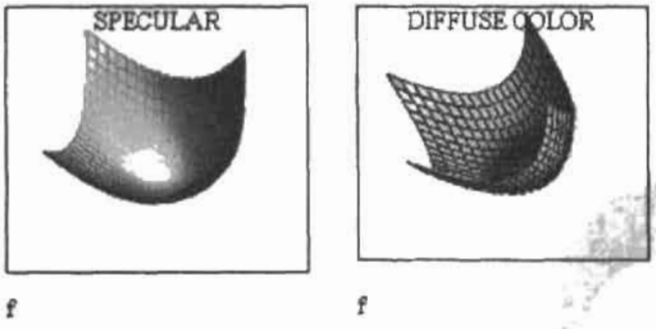


Рис. 6.58. Типы источников света

- Infinite Light Source (Бесконечно удаленный источник света). Этот параметр отвечает за проявление эффектов, связанных с бесконечным удалением источника света. Определяет освещение диффузным светом.
- Определить схему расположения источников света можно в списке параметра Lighting Scheme (Схема освещения).

Вкладка Title (Заголовок)

За задание заглавия графика отвечают параметры вкладки Title (Заголовок).

Поместить название можно либо выше (Above), либо ниже (Below) поверхности. В том случае, если отмечен параметр Hide (Спрятать), название не отображается.

Быстрое редактирование вида поверхности можно также провести с помощью ее контекстного меню (вызывается щелчком правой кнопкой мыши на области графика), которое содержит несколько наиболее важных параметров отображения.

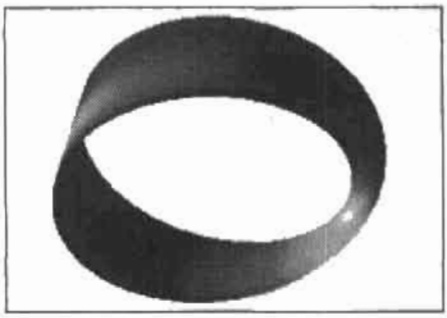
6.2.3. Интересные поверхности

Заканчивая разговор о построении и форматировании 3D-графиков, рассмотрим несколько примеров оригинальных поверхностей, задаваемых с помощью систем параметрических уравнений. При изучении примеров обратите внимание на то, каких удивительных графических эффектов можно добиться, искусно используя возможности Mathcad в визуализации трехмерных объектов.

Лента Мебиуса

Лента Мебиуса является простейшей односторонней поверхностью, впервые описанной в 1865 году профессором Лейпцигского университета Августом Мебиусом. Получается она при склеивании двух противоположных сторон прямоугольника, одна из которых перевернута. Лента Мебиуса (рис. 6.59) обладает замечательными свойствами: двигаясь по ее поверхности вдоль края в одном направлении, всегда попадаешь в исходную точку, но в обращенном положении относительно предыдущего; если же разрезать ленту Мебиуса по средней линии, мы получим не две части, а новую поверхность, аналогичную поверхности цилиндра, но перекрученную два раза вокруг себя.

$$\text{Moebius\_band}(u,v) := \begin{pmatrix} \cos(u) + v\cdot\cos\left(\frac{u}{2}\right)\cdot\cos(u) \\ \sin(u) + v\cdot\cos\left(\frac{u}{2}\right)\sin(u) \\ v\cdot\sin\left(\frac{u}{2}\right) \end{pmatrix}$$



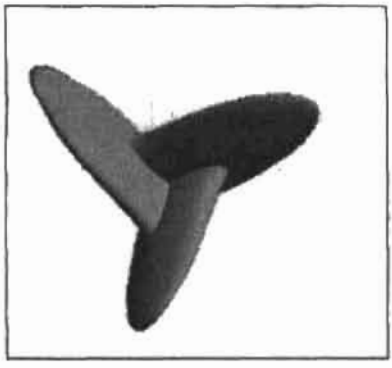
CreateMesh(Moebius_band,-5,2,-0.1,0.1,40,40)

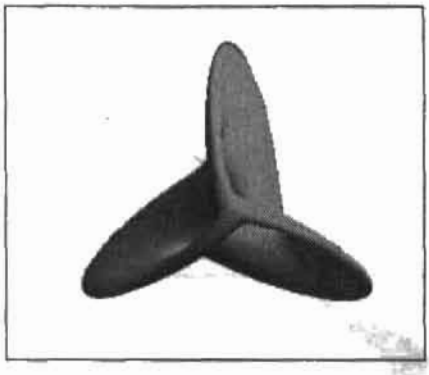
Рис. 6.59. Лента Мебиуса

Поверхность Боя

Поверхность Боя (рис. 6.60) — типичный представитель класса односторонних поверхностей (к которому, в частности, относится и знаменитая бутылка Клейна), которые невозможно отобразить в трехмерном пространстве без линий самопересечения (без самопересечения такие поверхности можно представить только в четырехмерном пространстве).

$$\text{Boy's\_surface}(u,v) := \begin{bmatrix} \frac{\left(\cos(u)\cdot\cos(2\cdot v) + \sqrt{2}\cdot\sin(u)\cdot\cos(v)\right)\cdot\cos(u)}{\sqrt{2} - \sin(2\cdot u)\cdot\sin(3\cdot v)} \\ \frac{\left(\cos(u)\cdot\sin(2\cdot v) - \sqrt{2}\cdot\sin(u)\cdot\sin(v)\right)\cdot\cos(u)}{\sqrt{2} - \sin(2\cdot u)\cdot\sin(3\cdot v)} \\ \frac{\sqrt{2}\cdot\cos(u)^2}{\sqrt{2} - \sin(2u)\sin(3v)} \end{bmatrix}$$





CreateMesh(Boy's_surface,0,2,-π,π,100,100) CreateMesh(Boy's_surface,0,2,-π,π,100,100)

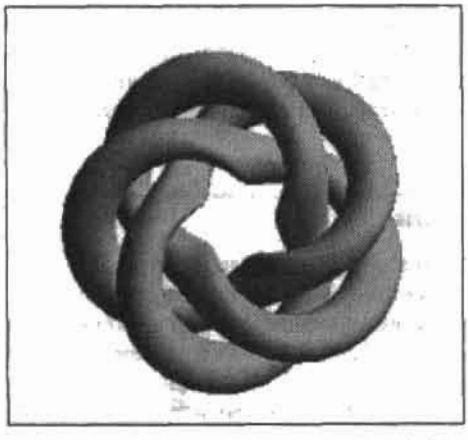
Рис. 6.60. Поверхность Боя

Сплетенная трубка

Сплетенная трубка представлена на рис. 6.61.

$$a := 5 \qquad b := 1.25 \qquad c := 2 \qquad m := 3 \qquad n := 5$$

$$\text{Knotted\_tube}(\theta, \psi) := \begin{bmatrix} (a + b\cdot\cos(\theta) + c\cdot\sin(n\cdot\psi))\cdot\cos(m\cdot\psi) \\ (a + b\cdot\cos(\theta) + c\cdot\sin(n\cdot\psi))\cdot\sin(m\cdot\psi) \\ b\cdot\sin(\theta) + c\cdot\cos(n\cdot\psi) \end{bmatrix}$$



CreateMesh(Knotted_tube,0,10,0,10,200,200)

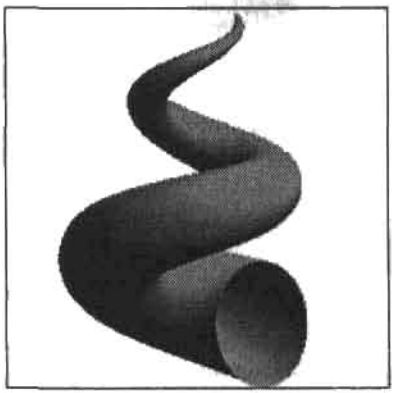
Рис. 6.61. Сплетенная трубка

Ракушка

Ракушка представлена на рис. 6.62.

$$a := 0.2 \qquad b := 1 \qquad c := 0.1 \qquad n := 2$$

$$\text{Shell}(u,v) := \begin{bmatrix} a\cdot\left(1-\frac{v}{2\pi}\right)\cdot\cos(n\cdot v)\cdot(1+\cos(u)) + c\cdot\cos(n\cdot v) \\ a\cdot\left(1-\frac{v}{2\pi}\right)\cdot\sin(n\cdot v)\cdot(1+\cos(u)) + c\cdot\sin(n\cdot v) \\ b\cdot\frac{v}{2\pi} + a\cdot\left(1-\frac{v}{2\pi}\right)\sin(u) \end{bmatrix}$$



CreateMesh(Shell,0,7,0,6,100,100)

Рис. 6.62. Ракушка

Большое количество примеров самых необычных, поражающих воображение поверхностей вы можете найти по адресу http://www.mathcad.com/resources/gallery/, а также в электронной книге Creating Amazing Images with Mathcad (Создание удивительных изображений в Mathcad), которую можно скачать со страницы http://www.mathcad.com/library/ebooks/images.asp.

6.2.4. Построение многогранников

Практически всем из нас в школе или вузе приходилось выполнять чертежи многоугольников, к примеру, при решении тех же задач по геометрии. Описать такие пространственные фигуры, как куб, пирамида или октаэдр с помощью уравнения или системы уравнений практически невозможно. Следовательно, их нельзя построить в системе Mathcad с помощью стандартных методов. Однако упростить себе работу, связанную с выполнением чертежей, все же возможно благодаря тому, что в программу встроено средство получения большинства встречающихся на практике многогранников.

Чтобы построить многогранник (рис. 6.63), следует использовать специальную функцию Polyhedron(S), где S — это либо порядковый номер, либо название, либо код (так называемый Wythoff-символ).

Проще всего можно построить нужный вам многогранник, используя его порядковый номер. Для этого введите соответствующее число в виде строки (то есть его нужно взять в кавычки), поставив перед ним символ номера (#). Учитывая то, что всего встроенных многогранников 80, то и вводимое число должно лежать между 1 и 80 (см. рис. 6.63).

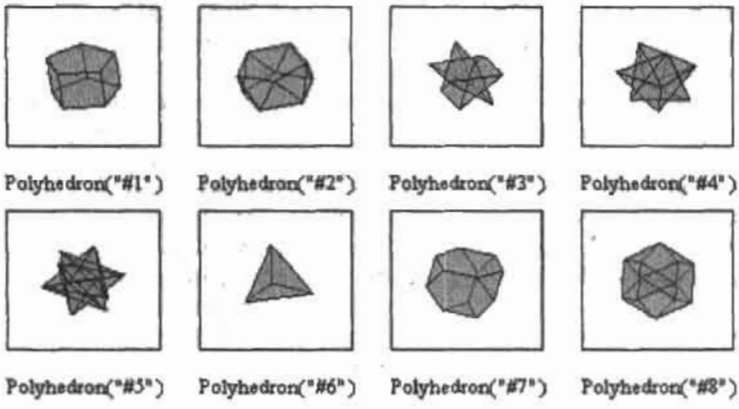


Рис. 6.63. Восемь первых многогранников, имеющихся в системе Mathcad

Более глубокие знания (по крайней мере, в английском языке) требуются для построения многогранника по его названию. А самостоятельно написать код (Wythoff-symbol) смогут лишь специалисты. Впрочем, получить подобного рода информацию можно, используя специальную функцию PolyLookup (S). Аналогично функции Polyhedron(S), в ее скобки можно ввести как порядковый номер, так и код или название. Как результат функция PolyLookup (S) выдает вектор, содержащий информацию о многограннике: варианты названия и код. Пример использования данной функции, а также построения по полученным данным многогранника, приведен на рис. 6.64.

$$\text{PolyLookup("\#35")} = \begin{pmatrix} \text{"small ditrigonal icosidodecahedron"} \\ \text{"small triambic icosahedron"} \\ \text{"3|5/2 3"} \end{pmatrix}$$

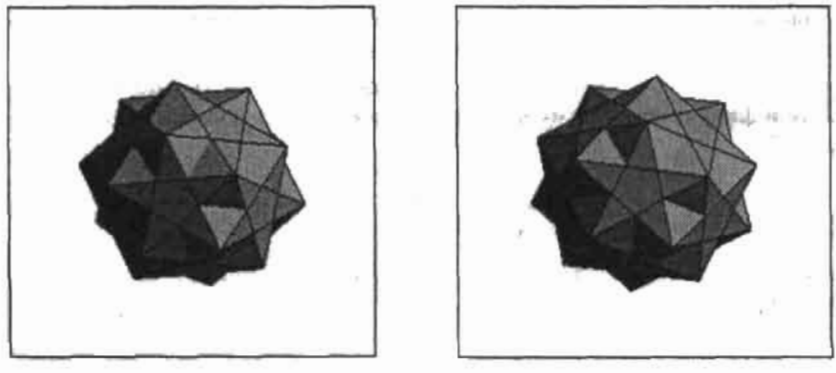


Рис. 6.64. Использование функции PolyLookup (S)

Все основные возможности оформления трехмерных графиков могут быть применены и для многогранников. Особенно эффектным оказывается применение к ним графических фильтров освещения.

Наиболее значительной трудностью при построении многогранников является то, что довольно проблематично узнать, какой порядковый номер имеет нужный вам объект. Конечно, можно просто просмотреть все 80 встроенных многогранников, однако на это может потребоваться слишком много времени. Для наиболее простых и известных фигур (например, куба или тетраэдра) английское название можно найти в словаре. Но что делать, если нужно построить, например, пятиугольную призму (термин, подобный этому, не будет переведен правильно почти наверняка, даже при использовании самых современных переводчиков). В таком случае вам следует обратиться к Ресурсам Mathcad (Mathcad Resources). Зайдите в раздел Reference Tables (Справочные таблицы), затем в группе статей Geometry (Геометрия) выберите статью Polyhedra (Многогранники). В ней вы найдете изображение и краткую информацию обо всех многогранниках, имеющихся в Mathcad. Значительно облегчит поиск то обстоятельство, что все многогранники разбиты на группы исходя из принятой в геометрии классификации.

6.2.5. Построение векторного поля

Хотя векторное поле формально и относится к трехмерным объектам, его вид, а главное, физический и математический смысл сильно отличаются от той же поверхности. К тому же векторное поле задается по совершенно другим правилам, чем все остальные трехмерные графики. Поэтому мы рассмотрим его построение в качестве отдельной темы.

Чтобы лучше понять принципы, лежащие в основе задания векторного поля, изучим этот вопрос, решая какую-нибудь конкретную задачу. Построим, например, векторное поле напряженности электростатического поля, создаваемого точечным зарядом.

Прежде всего, приступая к решению поставленной задачи, требуется задать все константы, которые будут нами использованы. В данном случае это величина точечного заряда и электрическая постоянная:

$$q := 10^{-5} \qquad \varepsilon_0 := 8.854187818 \cdot 10^{-12}$$

Размерности констант лучше не использовать, так как они, не меняя ничего в самом графике, могут создать дополнительные трудности в ходе его задания.

Далее сформулируем закон, на основе которого должно быть построено векторное поле. По известной физической формуле, напряженность электрического поля определяется формулой:

$$E(r) = \frac{1}{4\pi\varepsilon_0} \cdot \frac{q}{r^2}$$

Если принять условие, что заряд находится в точке начала координат, то данную формулу можно переписать:

$$E(x, y) := \frac{1}{4\pi\varepsilon_0} \cdot \frac{q}{x^2 + y^2}$$

Аналогично заданию поверхности, теперь следует задать сетку разбиений, в узлах которой будут прорисованы соответствующие векторы поля. Сделать это проще всего с помощью ранжированных переменных, причем никаких отличий выполнения этой задачи для векторного поля от рассмотренных нами ранее нет. Единственное, шаг должен быть задан довольно большим: иначе векторы просто сольются.

$$X_{start} := -2 \quad X_{end} := 2 \qquad Y_{start} := -2 \quad Y_{end} := 2$$

$$N := 11 \qquad M := 11$$

$$i := 0..N \qquad j := 0..M$$

$$x_i := X_{start} + \frac{X_{end} - X_{start}}{N} \cdot i \qquad y_j := Y_{start} + \frac{Y_{end} - Y_{start}}{M} \cdot j$$

Чтобы легче понять дальнейшие шаги построения, разберемся, что такое векторное поле в принципе. В самом общем случае векторное поле — это вид графика, который описывает поведение функции двух переменных с помощью направленных стрелок векторов. В зависимости от того, какой математический (или физический) смысл положен в основу векторного поля, направление векторов и их величина могут вычисляться по-разному. Чаще всего в качестве такого базиса используется понятие градиента — вектора, составленного из частных производных функции. В этом случае направление вектора поля определяется векторной суммой векторов отдельных частных производных в рассматриваемой точке (так как эти векторы параллельны соответствующим осям, то это предельно просто реализуется технически). Модуль же вектора поля определяется по теореме Пифагора как корень квадратный из суммы квадратов частных производных. Математический смысл построенного таким образом графика довольно очевиден: направление стрелок характеризует поведение функции (возрастание или убывание), а также соотношение между приращениями по разным переменным, длина — быстроту изменения функции.

В случае рассматриваемой задачи, аналогично построению векторного поля градиента функции, также требуется задать векторы, сложением которых будет определено направление векторов графика. А так как модуль значения напряженности поля в любой точке пространства может быть легко вычислен, то очень просто можно задать необходимый базис, спроектировав вектор E на координатные оси:

$$X_{i,j} := E(x_i, y_j)\cos(\operatorname{atan2}(x_i, y_j)) \qquad Y_{i,j} := E(x_i, y_j)\sin(\operatorname{atan2}(x_i, y_j))$$

Функция atan2(x,y) служит для вычисления угла между линией, соединяющей точку (x,y) и начало координат, и осью X. Впрочем, даже не прибегая к тригонометрии и не используя встроенных функций, можно догадаться, что соответствующие модули векторов базиса легко можно определить как:

$$X_{i,j} := E(x_i, y_j)\frac{x_i}{\sqrt{(x_i)^2 + (y_j)^2}} \qquad Y_{i,j} := E(x_i, y_j)\frac{y_j}{\sqrt{(x_i)^2 + (y_j)^2}}$$

Никакой принципиальной разницы от того, каким образом вы определите базис, не будет. Единственное, при задании ранжированных переменных следует учесть то, что в некоторых точках функция может не существовать. Так, в нашем случае нельзя задать M и N как 10, так как при этом придется вычислять значение функции в точке (0,0), что, естественно, невозможно.

После того как координаты векторов базиса будут заданы, можно непосредственно строить векторное поле (рис. 6.65). Для этого матрицы X и Y объединяем в один массив (взяв их имена в скобки в маркере графической области).

В результате проделанной работы векторное поле получилось, и получилось правильно. Однако нельзя не согласиться, что вид его, при всей математической корректности,

оставляет желать лучшего. Лишь четыре наиболее близких к заряду вектора имеют вид стрелки, остальные же столь малы, что отображаются в виде точки. Это связано с тем, что функция напряженности убывает очень быстро, а так как существует пропорциональность между размерами вектора поля и величиной функции в данной точке, то одни векторы оказываются несопоставимо большими, чем другие.

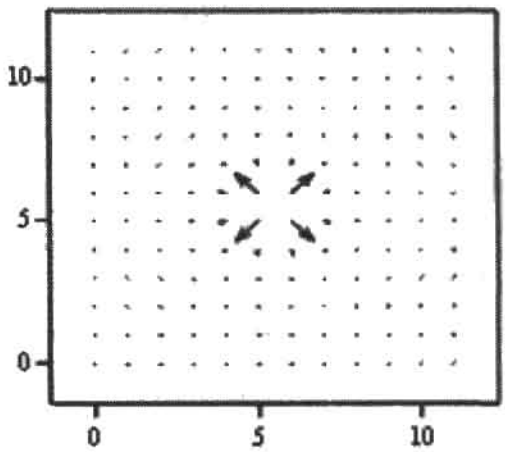


Рис. 6.65. Векторное поле (без логарифмирования)

Чтобы решить возникшую проблему, вспомним, как мы справились с аналогичными трудностями при построении двумерных кривых. Тогда для этого была введена логарифмическая шкала. А что если попробовать прологарифмировать функцию напряженности? Ведь она как раз принимает только положительные значения:

$$E(x,y) := \log\left(\frac{1}{4\pi\varepsilon_0}\cdot\frac{q}{x^2+y^2},6\right)$$

Попробуем построить векторное поле теперь (рис. 6.66).

Согласитесь, что полученное векторное поле гораздо лучше того варианта, который был построен без использования логарифмирования. Кстати, меняя основание логарифма, можно сделать разницу между стрелками середины и краев графика значительной в той степени, в которой вы посчитаете нужной.

Скажем несколько слов об оформлении векторного поля. Из существующих возможностей оформления стоит выделить построение цветного графика и изменение толщины стрелок поля.

Чтобы окрасить стрелки векторного поля выбранным цветом или палитрой, задействуйте параметр Fill Arrows (Залить стрелки) вкладки Appearance (Вид).

Толщина векторов поля регулируется с помощью параметра Weight (Толщина) меню Line Options (Параметры линии) вкладки Appearance (Вид).

Также к векторному полю могут быть применены все параметры вкладки Advanced (Дополнительные) и Lighting (Освещение). Однако нет смысла, например, окутывать векторное поле туманом или освещать направленным светом.

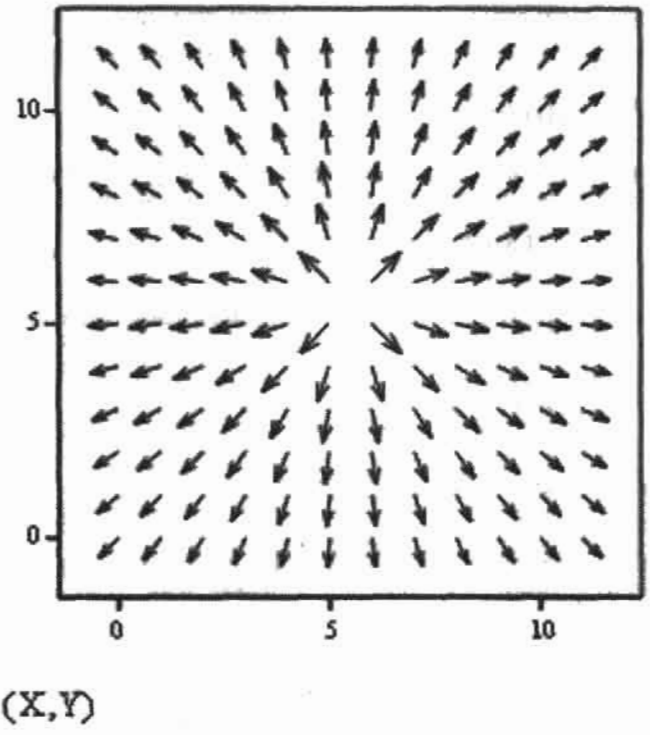


Рис. 6.66. Вид векторного поля при использовании логарифмирования

6.3. Анимация графиков

Без сомнения, самой необычной, интересной и удивительной возможностью визуализации в Mathcad является анимация. В данном разделе мы кратко рассмотрим основные ходы, применяемые при ее создании (более обстоятельный разговор невозможен по той причине, что просто в формате книги очень и очень проблематично привести результат проделанной работы).

Создается анимация в Mathcad точно так же, как, например, на студии Уолта Диснея. То есть она представляет собой быстро сменяющие друг друга кадры с более или менее значительными изменениями. А из-за того, что зрение наше довольно инертно, создается впечатление непрерывного процесса.

Требующиеся изменения в кадр вносятся с помощью специальной переменной FRAME (Кадр). Значение этой переменной равно по умолчанию номеру кадра, и, следовательно, может изменяться от 0 (до запуска процесса анимирования) до предельного значения 9999. Подставив переменную FRAME в функцию, вы заставите ее изменяться по мере того, как будут сменяться кадры (если увеличение переменной на 1 для каждого

кадра — это слишком много, то шаг можно уменьшить с помощью коэффициента). В результате при совмещении нескольких кадров получится анимация.

Технические особенности задания анимации рассмотрим на примере следующей задачи: попробуем, меняя значение точки касания, заставить касательную как бы скользить по кривой. Для этого выполняем следующую последовательность действий.

1. Задаем вид функции, касательную к которой мы заставим двигаться. Например:

$$y(x) := \frac{\sin(x)\cdot x}{3}$$

2. Определяем начальное положение и шаг изменения точки касания при переходе от кадра к кадру. Проще всего это сделать непосредственно с помощью переменной FRAME:

$$t := \frac{FRAME}{5} + \frac{\pi}{4}$$

3. Задаем уравнение, описывающее касательную. В качестве точки касания определяем параметр t:

$$D(x) := \frac{d}{dx} y(x)$$

$$tangent(t, x) := y(t) + D(t)\cdot(x - t)$$

4. Строим график функции и касательной.
5. С помощью команды Tools ▸ Animation ▸ Record Animation (Инструменты ▸ Анимация ▸ Записать анимацию) открываем специальное диалоговое окно, содержащее параметры анимации.
6. Выделяем с помощью пунктирного прямоугольника (точно так же, как при масштабировании фрагмента кривой) ту часть рабочей области, которая должна быть отображена в анимационном ролике. Обычно выделяется только внутренняя часть графической области.
7. В меню For Frame (Параметры кадра) настраиваем параметры будущей анимации. В окошке From (От) следует задать первое значение целочисленной переменной FRAME, в окошке To (До) — последнее. Количество кадров, сменяющихся в секунду, можно определить в окошке параметра At: Frames/Sec (Со скоростью: кадров в секунду).

   Для нашей задачи параметры должны быть определены следующим образом: FRAME изменяется от 0 до 30 со скоростью 10 кадров в секунду. В общем же случае, безусловно, шаг лучше делать по возможности более маленьким, а скорость прокрутки — высокой. При этом можно добиться очень высокого качества анимации, правда, лишь в том случае, если ваш компьютер достаточно мощный — иначе анимирование будет производиться очень и очень долго.
8. Нажимаем кнопку Animate (Анимировать). При этом начнется непосредственно процесс анимирования, за ходом которого вы сможете следить с помощью специальной зоны диалогового окна Animate. Количество проработанных кадров отображается под данной зоной в строке FRAME (Кадр).
9. После того как анимация будет завершена, на листе появится окно прокрутки видеороликов, нажав кнопку Play (Проиграть) которого, вы сможете просмотреть созданную анимацию (*рис. 6.67*).

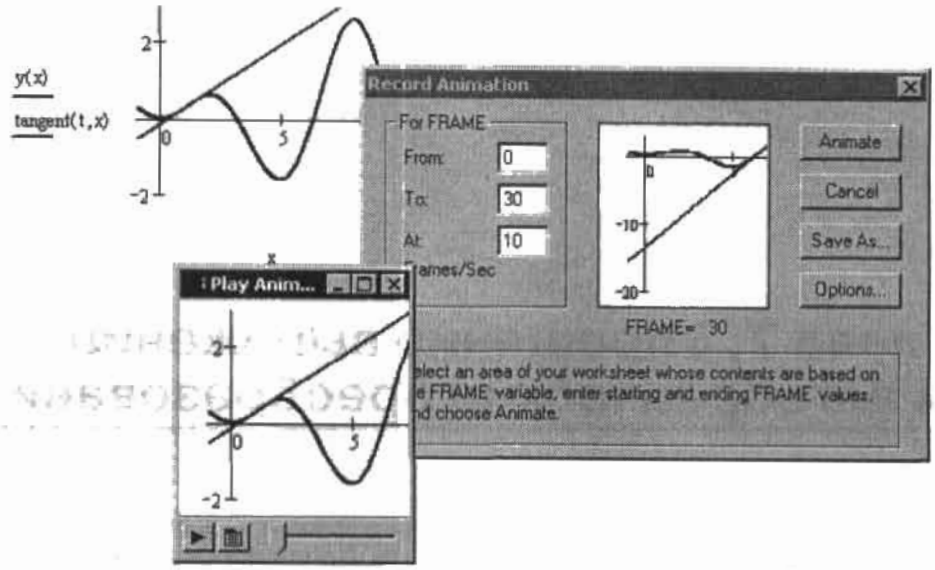


Рис. 6.67. Просмотр созданной анимации

Созданные анимации можно сохранять с помощью кнопки Save As (Сохранить как) в стандартном видеоформате, что позволит просматривать их в дальнейшем и без использования среды Mathcad.

Чтобы просмотреть созданную ранее анимацию с помощью Mathcad, задействуйте команду Tools ▸ Animation ▸ Playback (Инструменты ▸ Анимация ▸ Воспроизведение). В появившемся окне прокрутки видеороликов нажмите кнопку Open AVI и выберите необходимый файл.

Глава 7. Упрощение выражений и алгебраические преобразования

Пожалуй, трудно найти более рутинную, однако и более часто встречающуюся математическую операцию, чем упрощение выражений. Практически в любом выводе промежуточный результат приходится преобразовывать в форму, более удобную для дальнейшего использования. Разложение на множители, приведение общих слагаемых, переход от двойного аргумента к одинарному в случае тригонометрических функций — принципы проведения всех этих операций проходят еще в средней школе. Однако как не хочется порой терять время на то, чтобы, к примеру, в очередной раз перемножать полиномы или приводить две дроби к одному знаменателю! Значительную помощь при этом может оказать символьный процессор Mathcad. Конечно, наивно рассчитывать на то, что программа сможет без вашей помощи решать задачи уровня сборника М. И. Сканави. Однако проблемы логически менее нагруженные, но практически сколь угодно сложные с вычислительной точки зрения для нее вполне по силам. В этой главе мы научимся эффективно использовать возможности символьного процессора в области упрощения выражений, проанализировав на примерах, какие задачи можно полностью возложить на Mathcad, а в каких требуется ваше активное участие.

7.1. Разложение выражений

Под разложением алгебраического выражения понимается математическое преобразование, переводящее степени и произведения в более простые для анализа суммы (например, $(a-b)\cdot(a+b)=a^2-b^2$). В случае разложения тригонометрических выражений функции кратного аргумента преобразуются в соответствующие им выражения от одинарного аргумента (например, $\sin(2\cdot x)=2\cdot\sin(x)\cdot\cos(x)$); разлагаются также функции от суммы переменных (к примеру, по формуле $\sin(x+y)=\sin(x)\cdot\cos(y)+\cos(x)\cdot\sin(y)$). При разложении логарифмических выражений используются преобразования типа $\ln(a\cdot b)=\ln(a)+\ln(b)$, $\ln(a/b)=\ln(a)-\ln(b)$, $\log_a(a^x)=x$. Для проведения разложения выражений в Mathcad существует специальный оператор expand (Разложить) панели Symbolic (Символьные). В левый маркер данного оператора заносится упрощаемое выражение, в правый — переменная (или выражение), относительно которой разложение проводится. Объективно говоря, в подавляющем большинстве случаев заполнение

правого маркера — это избыточная операция и проводить ее не обязательно. Поэтому обычно правый маркер просто удаляют.

Пример 7.1. Разложение выражений различных типов

Разложение алгебраического выражения. Система перемножает выражения в числителе и производит приведение слагаемых к общему знаменателю. Затем раскрывается степень в знаменателе, после чего числитель делится на знаменатель.

$$\frac{\left(x^2+3x+1\right)\cdot\left(x+\frac{1}{x}\right)}{(x+1)^5}\ \text{expand}\ \rightarrow \frac{x^4+2\cdot x^2+3\cdot x^3+3\cdot x+1}{x^6+5\cdot x^5+10\cdot x^4+10\cdot x^3+5\cdot x^2+x}$$

Разложение тригонометрического выражения. Система переходит от sin(2x) и tg(3x) к функциям от x, после чего делит числитель на знаменатель.

$$\frac{\sin(2x)}{\tan(3x)}\ \text{expand}\ \rightarrow \frac{2}{3\cdot\tan(x)-\tan(x)^3}\cdot\sin(x)\cdot\cos(x)-\frac{6}{3\cdot\tan(x)-\tan(x)^3}\cdot\sin(x)\cdot\cos(x)\cdot\tan(x)^2$$

Разложение логарифмического выражения. Обратите внимание на то, что десятичный логарифм представляется через отношение натуральных логарифмов.

$$\frac{\log\left(b^x\right)}{\ln\left(e^b\right)}\ \text{expand}\ \rightarrow x\cdot\frac{\ln(b)}{\ln(10)\cdot b}$$

Внимательный читатель может задать вопрос: зачем у оператора expand имеется правый маркер, если то, заполнен он или нет, никак не сказывается на результате. Действительно, в большинстве случаев указывать, исходя из какой переменной (или выражения) должно производиться разложение выражения, совсем необязательно. Однако иногда то, заполнен ли правый маркер или же нет, может весьма существенно сказываться на результате. Дело в том, что, когда правый маркер удален, аналитический процессор пытается разложить выражение «по максимуму», что не всегда приемлемо. Например, пусть имеется выражение вроде $(\sin(2x)+1)^3$. Если использовать оператор expand без заполнения правого маркера, то сначала система возведет выражение в третью степень, а затем перейдет от sin(2x) к функциям от x. Однако может оказаться так, что в рамках решаемой задачи нужно лишь произвести возведение в степень, не переходя от удвоенного к одинарному аргументу. В этом случае в качестве параметра разложения в правом маркере оператора expand следует указать 2x. При этом система «поймет», что 2x должно входить в результат, и не будет разлагать синус (см. пример 7.2). Описанный подход применим и к выражениям других типов. Общий принцип следующий: если выражение образовано несколькими частями и одна из его частей не должна быть разложена, то ее следует прописать в правом маркере оператора expand. При этом она будет сохранена аналитическим процессором в первоначальном виде. Также довольно тонкое различие в формате ответа при заполнении правого маркера и его удалении обнаруживается, если упрощаемое выражение является дробью. Так, если переменная разложения прописана, то система разделит числитель на знаменатель. Если же правый маркер оператора expand был удален, то никакого деления проводиться не будет.

Пример 7.2. Различие в результате разложения при заполнении правого маркера оператора expand и его удалении

По умолчанию оператор expand производит как возведение в степень, так и приведение тригонометрических функций к одинарному аргументу. Чтобы отменить последнюю операцию, прописываем удвоенный аргумент в правом маркере.

$$(1 + \sin(2x))^3 \text{ expand } \rightarrow 1 + 6\cdot\sin(x)\cdot\cos(x) + 12\cdot\sin(x)^2\cdot\cos(x)^2 + 8\cdot\sin(x)^3\cdot\cos(x)^3$$

$$(1 + \sin(2x))^3 \text{ expand}, 2x \rightarrow 1 + 3\cdot\sin(2\cdot x) + 3\cdot\sin(2\cdot x)^2 + \sin(2\cdot x)^3$$

Чтобы разложить лишь одну часть выражения, подлежащую сохранению, его часть указываем в правом маркере оператора expand.

$$(x + y)^2 + (x + y + 1)^2 \text{ expand } \rightarrow 2\cdot x^2 + 4\cdot x\cdot y + 2\cdot y^2 + 2\cdot x + 2\cdot y + 1$$

$$(x + y)^2 + (x + y + 1)^2 \text{ expand}, x + y + 1 \rightarrow x^2 + 2\cdot x\cdot y + y^2 + (x + y + 1)^2$$

Если правый маркер expand заполнен, то, в случае выражений в виде дроби, числитель будет разделен на знаменатель.

$$\frac{(x + 1)^3}{(x + 2)^4} \text{ expand } \rightarrow \frac{x^3 + 3\cdot x^2 + 3\cdot x + 1}{x^4 + 8\cdot x^3 + 24\cdot x^2 + 32\cdot x + 16}$$

$$\frac{(x + 1)^3}{(x + 2)^4} \text{ expand}, x \rightarrow \frac{1}{(x + 2)^4}\cdot x^3 + \frac{3}{(x + 2)^4}\cdot x^2 + \frac{3}{(x + 2)^4}\cdot x + \frac{1}{(x + 2)^4}$$

Эффективно использовать оператор expand в случае логарифмических выражений можно далеко не всегда. Причина — результат генерируется так, что в нем присутствуют только натуральные логарифмы. Логарифмы же по другим основаниям при этом приводятся к натуральным по формуле $\log_a(x)=\ln(x)/\ln(a)$. Увы, но обойти этот недостаток системы невозможно. Поэтому при решении соответствующих задач просто заменяйте отношения натуральных логарифмов нужными логарифмами по основанию a.

В Mathcad нет оператора, обратного expand. Имеется оператор factor, преобразующий алгебраические суммы в произведения. Он является обратным к expand в случае тождеств вроде $x^2-y^2=(x-y)\cdot(x+y)$. Однако «понять», что $2\cdot\sin(x)\cdot\cos(x)$ есть не что иное, как $\sin(2x)$ (или что $\ln(x)+\ln(y)$ тождественно равняется $\ln(x\cdot y)$), ни оператор factor, ни оператор simplify не смогут. Проведение подобных преобразований — это довольно значительная проблема в Mathcad. Обычно с ней справляются банальной подстановкой формулы — и ничего лучшего пока не придумано. Пример этой операции имеется в разделе, посвященном оператору simplify.

Кстати, помимо своего прямого назначения, оператор expand может быть использован в качестве своеобразного справочника математических формул по символьной алгебре (особенно тригонометрии). Например, с его помощью легко узнать, чему равняется $\sin(5\cdot x)$ или $\cos(x+y+z)$.

7.2. Разложение на множители и приведение к общему знаменателю

Произвести разложение выражения на множители в системе Mathcad можно с помощью оператора factor (от англ. factoring — разложение на множители). По функциям данный оператор не является полной противоположностью оператору expand. Так, с помощью оператора factor нельзя преобразовать логарифмическое или тригонометрическое выражение. Разложить на множители можно только не очень сложное алгебраическое выражение. Особенностью использования оператора factor является то, что он содержит лишний правый маркер. Если вам требуется произвести разложение выражения на множители, этот маркер нужно затереть. Иначе, если аналогично оператору expand в правый маркер ввести имя переменной (или выражение), по которой должно вестись разложение, даже в самом простом случае результат не будет получен.

При разложении алгебраического выражения на множители в качестве таковых могут выступать не только линейные множители, но и полиномы более низкой степени, чем у исходного выражения. В отличие от символьного решения уравнений, аналитический процессор не ищет комплексных множителей, ограничиваясь только действительными. Разложение на множители с использованием оператора factor может быть проведено и в том случае, если в выражения входят специальные функции (см. третье преобразование в примере 7.3).

Пример 7.3. Разложение многочленов на множители

$$x^3 - 1 \text{ factor} \rightarrow (x-1)\cdot\left(x^2 + x + 1\right)$$

$$x^4 - 10\cdot x^3 + 35\cdot x^2 - 50\cdot x + 24 \text{ factor} \rightarrow (x-1)\cdot(x-2)\cdot(x-3)\cdot(x-4)$$

$$\sin(x)^2 - \cos(x)^2 \text{ factor} \rightarrow -(\cos(x) - \sin(x))\cdot(\cos(x) + \sin(x))$$

$$x^2 + 2x\cdot y + 2x\cdot z + y^2 + 2y\cdot z + z^2 \text{ factor} \rightarrow (x + y + z)^2$$

Второй по важности задачей, для решения которой используется оператор factor, является разложение целых чисел на простые множители. Эта возможность символьного процессора активно применяется при поиске корней полиномов, решении учебных задач по криптографии (алгоритм RSA), а также упрощении выражений.

Пример 7.4. Определить, является ли число $11^{11}+13$ простым

$$11^{11} + 13 \text{ factor} \rightarrow 2^5\cdot 3^4\cdot 197\cdot 293\cdot 1907$$

Вывод: данное число простым не является, так как его можно получить перемножением пяти различных чисел.

Довольно интересной возможностью оператора factor является преобразование десятичных дробей в дроби простые. Естественно, что такая операция возможна лишь в случае отдельных десятичных дробей. Самое важное требование — дробь должна быть конечной (например, 0,25, но не 0,3333333333...). Описанную возможность можно использовать для приведения выражения в более простую для аналитического процессора форму (простые дроби он «понимает» лучше, чем десятичные), а также для «улучшения» ответа, полученного при численном решении уравнений или численном интегрировании.

Пример 7.5. Преобразование десятичных дробей в обыкновенные

$$0.125\,\text{factor} \rightarrow \frac{1}{8} \qquad 1.674\,\text{factor} \rightarrow \frac{837}{500} \qquad 0.45 + 0.54\cdot i\,\text{factor} \rightarrow \frac{9}{20} + \frac{27}{50}\cdot i$$

Нижележащий пример демонстрирует, как результат численного интегрирования можно перевести в символьную форму.

$$\int_1^5 \frac{x^4}{2500}\,dx = 0.24992 \qquad 0.24992\,\text{factor} \rightarrow \frac{781}{3125} \qquad \int_1^5 \frac{x^4}{2500}\,dx \rightarrow \frac{781}{3125}$$

Очень важной для практики операцией, для проведения которой используется оператор factor, является приведение к общему знаменателю выражения из двух или более дробей. Причем, в числитель и знаменатель дробей могут входить совершенно любые функции и их сочетания. Приведение к общему знаменателю — это один из основных ходов, применяемых при упрощении выражений.

Пример 7.6. Приведение дробей к общему знаменателю

$$1 + \frac{1}{x} + \frac{1}{x^2} + \frac{1}{x+y}\,\text{factor} \rightarrow \frac{x^3 + x^2\cdot y + 2\cdot x^2 + x\cdot y + x + y}{x^2\cdot(x+y)}$$

В задачниках для поступающих в вузы довольно часто встречаются задачи, в которых требуется так преобразовать арифметическое выражение, чтобы в знаменателе исчезла иррациональность (то есть как-то все корни нужно перенести в числитель). Задачи такого рода требуют весьма значительных выкладок, однако оператор factor справляется с ними с легкостью.

Пример 7.7. Избавиться от иррациональности в знаменателе арифметического выражения

$$\frac{6}{\sqrt{2}+\sqrt{3}+\sqrt{5}}\,\text{factor} \rightarrow \frac{-1}{2}\cdot 2^{\frac{1}{2}}\cdot 3^{\frac{1}{2}}\cdot 5^{\frac{1}{2}} + \frac{3}{2}\cdot 2^{\frac{1}{2}} + 3^{\frac{1}{2}}$$

7.3. Вынесение общего множителя за скобку

Вынести общий множитель за скобку в выражении можно с помощью символьного оператора collect (Собрать). Определить, по какой переменной нужно производить данное преобразование, можно, введя ее имя в правый маркер оператора. Вынесение общего множителя может быть осуществлено как по одной, так и по нескольким переменным. В последнем случае имена переменных нужно указать через запятую в маркере оператора collect в той последовательности, в которой они должны быть вынесены за скобку. В случае дробных выражений вынесение общего множителя за скобку будет проводиться как в числителе, так и в знаменателе. Если слагаемые должны быть распределены по нескольким группам, то выражение придется разделить на части и опе-

рировать с каждой из них по отдельности (или, что вероятнее всего, выполнить эту работу придется самостоятельно). Также нельзя вынести за скобку произведение двух переменных. В выражениях из функций вынести общий множитель за скобку можно, прописав нужную функцию в правом маркере оператора collect. Если же необходимо вынести за скобку несколько множителей-функций, их, как и в случае переменных, необходимо последовательно прописать через запятую в маркере оператора collect.

Оператор collect объединяет в одну группу члены, в которые переменная, по которой проводится преобразование, входит в одинаковой степени. Поэтому, например, выражение x^2+x+1 не будет представлено им как $x\cdot(x+1)+1$. Вообще же, группировка членов многочлена «лучшим образом» — это чрезвычайно нагруженная интеллектуально задача. Поэтому оператор collect, в общем, довольно редко используется на практике. Из-за того, что нельзя указать, какое именно выражение должно быть вынесено за скобку, он малополезен для большинства задач. Увы, но их пока придется решать на бумаге.

Пример 7.8. Вынесение общего множителя за скобку

$$x^3\cdot y^2\cdot z^3 + x^3\cdot y^2\cdot z^2 + x^2\cdot y^3\cdot z^2 + x^2\cdot y^2\cdot z^2 \text{ collect, } x \rightarrow \left(y^2\cdot z^3 + y^2\cdot z^2\right)\cdot x^3 + \left(y^3\cdot z^2 + y^2\cdot z^2\right)\cdot x^2$$

$$x^3\cdot y^2\cdot z^3 + x^3\cdot y^2\cdot z^2 + x^2\cdot y^3\cdot z^2 + x^2\cdot y^2\cdot z^2 \text{ collect, } y, x, z \rightarrow x^2\cdot y^3\cdot z^2 + \left[\left(z^3 + z^2\right)\cdot x^3 + x^2\cdot z^2\right]\cdot y^2$$

$$\sin(x)\cos(x)^2 + \sin(x) \text{ collect, } \sin(x) \rightarrow \left(\cos(x)^2 + 1\right)\cdot\sin(x)$$

$$2x^2 e^x\cos(x) + 5e^x\sin(x) + 3x\cdot e^x\cdot\cos(x) - \frac{e^x}{x}\cdot\sin(x) \text{ collect, } \cos(x), \sin(x), e^x \rightarrow$$

$$\rightarrow \left(2\cdot x^2 + 3\cdot x\right)\cdot e^x\cdot\cos(x) + \left(5 - \frac{1}{x}\right)\cdot e^x\cdot\sin(x)$$

7.4. Разложение на элементарные дроби

Одной из самых объемных и трудных в вычислительном плане задач символьной алгебры является разложение какого-то сложного дробного выражения на более простые, в идеале — линейные, дроби. Эта задача важна прежде всего при подсчете интеграла от отношения двух полиномов и встречается в любом вузовском типовом расчете по математическому анализу. Кроме того, разложение на элементарные дроби — это одна из основных операций, использующихся при упрощении выражений.

В Mathcad имеется специальный символьный оператор convert,parfac (от англ. Convert to Partial Fraction — разложить на элементарные дроби), проводящий рассматриваемую операцию. Вводится он нажатием кнопки parfac панели Symbolic (Символьные). В его левом маркере прописывается подлежащее преобразованию выражение, в правом — переменная, исходя из которой преобразование должно проводиться (программа не может самостоятельно различить переменную и параметры).

Обычно на элементарные дроби раскладывается выражение, представляющее собой отношение полиномов. Однако можно разложить и выражение, в котором в качестве переменных полинома выступают функции (см. третье преобразование в примере 7.9).

Пример 7.9. Разложение на элементарные дроби выражений различных типов

$$\frac{x^2+5}{x^3-3x+2}\ \text{convert},\text{parfrac},x \rightarrow \frac{1}{x+2}+\frac{2}{(x-1)^2}$$

$$\frac{-3x^2+(2\cdot b+2a+2c)\cdot x-b\cdot c-a\cdot b-a\cdot c}{x^3+(-a-b-c)\cdot x^2+(b\cdot c+a\cdot b+a\cdot c)x-a\cdot b\cdot c}\ \text{convert},\text{parfrac},x \rightarrow \frac{-1}{x-a}-\frac{1}{x-c}-\frac{1}{x-b}$$

$$\frac{\cos(x)+2\cdot\sin(x)}{\sin(x)\cdot(\cos(x)+\sin(x))}\ \text{convert},\text{parfrac},\sin(x) \rightarrow \frac{1}{\sin(x)}+\frac{1}{\cos(x)+\sin(x)}$$

Очень полезен оператор convert parfac для упрощения интегрирования рациональных функций. Конечно, найти первообразную в Mathcad можно и сразу, используя оператор неопределенного интеграла панели Calculus (Исчисление). Однако такой подход не всегда приемлем. Например, если вы студент и решаете домашнее задание, то без промежуточных выкладок не обойтись. При интегрировании же рациональных функций самый трудоемкий и скучный этап — это разложение отношения полиномов на элементарные дроби. Именно его стоит доверить Mathcad.

Пример 7.10. Интегрирование рациональной функции

Пусть стоит задача найти интеграл от функции вида:

$$\frac{x^3+5x^2-13x-9}{x^4-10x^2+9}$$

Данная функция является рациональной, поэтому к форме, пригодной для прямого интегрирования, она приводится разложением на элементарные дроби.

$$\frac{x^3+5x^2-13x-9}{x^4-10x^2+9}\ \text{convert},\text{parfrac},x \rightarrow \frac{1}{x-1}-\frac{1}{x+3}+\frac{1}{2\cdot(x-3)}+\frac{1}{2\cdot(x+1)}$$

Проинтегрировать полученное выражение очень просто, помня, что первообразной для 1/a является ln(a). Итак, ответом будет:

$$\ln(x-1)-\ln(x+3)+\frac{1}{2}\cdot\ln(x-3)+\frac{1}{2}\cdot\ln(x+1)+C=\ln\left[\frac{(x-1)\cdot\sqrt{(x+1)\cdot(x-3)}}{x+3}\right]+C$$

Аналогичный результат будет получен и при использовании оператора неопределенного интеграла:

$$\int \frac{x^3+5x^2-13x-9}{x^4-10x^2+9}\,dx \rightarrow \ln(x-1)-\ln(x+3)+\frac{1}{2}\cdot\ln(x-3)+\frac{1}{2}\cdot\ln(x+1)$$

7.5. Выполнение подстановки и замены переменных

Очень часто при символьных расчетах возникают ситуации, требующие замены переменной в выражении на другую переменную или некоторое выражение. Например, замена может понадобиться для упрощения вида выражения или уменьшения его размеров. Наиболее просто подстановку можно осуществить, воспользовавшись специальным символьным оператором substitute (Заместить). Для этого проделайте следующую последовательность действий.

1. Выбрав соответствующую команду панели Symbolic (Символьные), введите оператор substitute (Заместить).
2. В левый маркер оператора внесите выражение или его имя.
3. В правом маркере сделайте запись вида a=b, где a — замещаемая переменная, b — подставляемое значение. Если заместить нужно две переменные, то через запятую делается еще одно аналогичное присваивание. В качестве знака равенства в данном случае следует использовать логическое равенство (Bold Equal — Ctrl+=).

Оператор substitute позволяет замещать не только переменные, но и функции и даже целые выражения. Например, чтобы заместить sin(x) на t, в правом маркере оператора substitute достаточно набрать «sin(x)=t».

Пример 7.11. Упрощение выражения с подстановкой

Задача: упростить выражение вида

$$\frac{1-b}{\sqrt{b}}\cdot x^2 - 2x + \sqrt{b}$$

при условии, что

$$x = \frac{\sqrt{b}}{1-\sqrt{b}}$$

Решение: выполняем подстановку выражения для x с помощью оператора substitute, а затем упрощаем выражение с использованием оператора simplify.

$$\frac{1-b}{\sqrt{b}}\cdot x^2 - 2x + \sqrt{b} \;\left|\begin{array}{l}\text{substitute},\ x = \dfrac{\sqrt{b}}{1-\sqrt{b}}\\ \text{simplify}\end{array}\right. \rightarrow 0$$

При проведении аналитических преобразований операция подстановки весьма востребована. Причина этого заключаетя в следующем. То, насколько эффективно символьный процессор будет проводить операции вроде разложения на множители или поиска корней, очень сильно зависит от вида выражения. Так, если в выражении есть корни, то система с ним работает неважно. Однако эффективность работы аналитического процессора можно резко повысить, если выполнить такую замену, чтобы корни исчезли. Сильно упрощенный пример, демонстрирующий описанный подход, приведен ниже. С более реалистичными примерами мы встретимся в следующем разделе.

Пример 7.12. Приведение выражения к более простому виду посредством замены переменных

Если попытаться разложить выражение в его первоначальном виде, то должного упрощения произведено не будет.

$$\frac{\sqrt{x}+\sqrt{y}}{x-y}\ \text{expand}\ \rightarrow \frac{1}{x-y}\cdot x^{\frac{1}{2}}+\frac{1}{x-y}\cdot y^{\frac{1}{2}}$$

Символьный процессор плохо справляется с выражениями с корнями, поэтому нужно осуществить такую подстановку, чтобы корни исчезли.

$$\frac{\sqrt{x}+\sqrt{y}}{x-y}\ \text{substitute}\ ,\sqrt{x}=m,\sqrt{y}=n,x=m^2,y=n^2\ \rightarrow \frac{m+n}{m^2-n^2}$$

Теперь выражение будет разложено так, как нужно.

$$\frac{m+n}{m^2-n^2}\ \text{expand}\ \rightarrow \frac{-1}{n-m}$$

Выполняем обратную подстановку, получая окончательный ответ.

$$\frac{1}{m-n}\ \text{substitute}\ ,m=\sqrt{x},n=\sqrt{y}\ \rightarrow \frac{1}{x^{\frac{1}{2}}-y^{\frac{1}{2}}}$$

7.6. Комплексное упрощение выражений

На практике упрощение выражений редко сводится к проведению операций какого-то одного рода. Чаще для того, чтобы преобразовать выражение в более простую форму, требуется задействовать несколько операций. Например, над результатом символьного дифференцирования может понадобиться сначала провести операцию приведения к подобным слагаемым, а уж затем попытаться вынести общие множители за скобку. Упрощение, для осуществления которого нужно провести несколько операций различного рода, мы будем называть комплексным. В простейших случаях аналитический процессор Mathcad способен проводить комплексное упрощение выражений. Служит для этого специальный символьный оператор simplify (Упростить).

По большому счету, возможности оператора simplify представляют собой частичную сумму возможностей остальных операторов алгебраических преобразований. Он может, подобно оператору expand, проводить разложение выражений. Аналогично оператору factor, simplify может осуществить приведение двух дробей к общему знаменателю. Как и collect, simplify способен вынести общий множитель за скобку. Кроме того, simplify «умеет» приводить подобные слагаемые и производить арифметические расчеты. Главное же достоинство рассматриваемого оператора — он способен совмещать все эти операции так, чтобы упрощение было максимальным.

Пример 7.13. Упрощение выражений

$$\frac{x^3 - y^3}{x^2 - y^2} \text{ simplify } \rightarrow \frac{x^2 + x\cdot y + y^2}{x + y}$$

$$\frac{x^2 - 2a\cdot x + a^2}{x^2 - (a+3)\cdot x + 3\cdot a} \text{ simplify } \rightarrow \frac{x - a}{x - 3} \qquad e^{\ln\left(\frac{x-y}{x^2-y^2}\right)} \text{ simplify } \rightarrow \frac{1}{x + y}$$

Очень часто глубина упрощения выражения и результат этой операции зависят от того, какие значения могут принимать входящие в выражения величины. По умолчанию Mathcad рассматривает все неизвестные как комплексные числа. По этой причине довольно интересный результат будет получен при попытке извлечь корень из квадрата переменной:

$$\sqrt{x^2} \text{ simplify } \rightarrow \text{csgn}(x)\cdot x$$

Функция csgn — это функция знака комплексного числа. Она равна 0 — если x=0; 1 — если действительная часть числа больше 0 (или если действительная часть равна 0, а мнимая положительна); –1 — если действительная часть отрицательна (или если действительная часть равна 0, а мнимая отрицательна). Хотя полученный ответ вполне корректен при предположении, что x — это комплексное число, в подавляющем большинстве случаев он неприемлем. Необходимо как-то указать аналитическому процессору, что x следует рассматривать как действительное число. Сделать это позволяет символьный оператор assume (Принимает).

Оператор assume имеет два маркера. В левый вводится оператор, результат работы которого должен быть получен исходя из некоторого условия (в нашем случае это будет оператор simplify). В правом маркере определяется само условие.

Условие может быть ограничивающим либо область определения переменной, либо ее тип. Ввести ограничения на область можно с помощью логических операторов панели Boolean (Булевы). Для того же, чтобы задать тип переменной или параметра, нужно обратиться к панели специальных ключевых слов Modifier (Модификация) панели Symbolic (Символьные).

Панель Modifier содержит четыре ключевых слова. Вводить их можно и просто набором с клавиатуры.

- assume (Принимает). Слово-заготовка для создания оператора модификации.
- real (Действительное). При использовании этого служебного слова система будет воспринимать переменную как действительную.
- RealRange(a,b) (Действительная область). С помощью этого оператора вы можете ограничить область изменения переменной или константы отрезком между действительными числами *a* и *b*. Обратите внимание на то, что при вызове данного оператора с панели Modifier его название начинается со строчной буквы. Поскольку Mathcad чувствителен к регистру, следует вручную заменить строчные буквы «r» в названии оператора на прописные «R» так, чтобы его вид соответствовал изображенному на панели Modifier. Легче же всего просто ввести данный оператор с клавиатуры.

- trig (Тригонометрические). Параметр или переменная рассматривается как тригонометрическая величина. На практике используется редко, так как реально практически ни на что не влияет.

Итак, попробуем найти корень из квадрата переменной, указав, что она принимает только действительные значения:

$$\sqrt{x^2}\ \left|\begin{array}{l}\text{assume},\ x = \text{real}\\ \text{simplify}\end{array}\right.\ \rightarrow \text{signum}(x)\cdot x$$

Функция signum является функцией знака действительного числа. Она возвращает –1 — если число отрицательно, 1 — если оно положительно — и 0 — если оно равно 0. Нетрудно заметить, что signum(x)·x есть не что иное, как модуль x.

Если известно, что переменная может быть только положительной или только отрицательной, можно придти к еще более простому результату. Для этого область изменения x должна быть указана в правом маркере оператора assume с помощью соответствующих неравенств:

$$\sqrt{x^2}\ \left|\begin{array}{l}\text{assume},\ x \geq 0\\ \text{simplify}\end{array}\right.\ \rightarrow x \qquad \sqrt{x^2}\ \left|\begin{array}{l}\text{assume},\ x < 0\\ \text{simplify}\end{array}\right.\ \rightarrow -x$$

Если переменная изменяется в какой-то ограниченной области, то наиболее эффективно использовать модификатор RealRange:

$$\sqrt{x^2}\ \left|\begin{array}{l}\text{assume},\ x = \text{RealRange}(3,5)\\ \text{simplify}\end{array}\right.\ \rightarrow x$$

Не забывайте использовать оператор assume в тех случаях, в которых результат упрощения зависит от значения переменной (это прежде всего выражения с корнем, а также некоторые логарифмические выражения и выражения с модулем). Помимо оператора simplify, оператор assume часто применяется с операторами взятия определенного и неопределенного интегралов.

Пожалуй, по логической нагруженности оператор simplify занимает первое место среди остальных символьных операторов. Действительно, ведь зачастую очень трудно четко определить, какое из двух выражений более простое, а следовательно, и задаться верным направлением в упрощении. Например, какое выражение проще: $\sin(2\cdot x)$ или $2\cdot\sin(x)\cdot\cos(x)$? Очевидно, что ответ зависит только от особенностей решаемой задачи. Но Mathcad эти особенности не известны! И это, увы, не может не сказываться на качестве работы оператора simplify: упростить в Mathcad можно лишь наиболее простые, как правило, алгебраические или логарифмические, выражения. Так, из 10 первых примеров по данной теме из известного задачника Сканави оператором simplify правильно не был упрощен ни один! А ведь это совсем несложные задачи, которые решают школьники в 8 классе. И если алгебраические выражения хотя бы в наиболее простых случаях могут быть успешно упрощены оператором simplify, то тригонометрическое выражение не будет правильно преобразовано почти наверняка (в лучшем случае система ограничится приведением подобных и упрощением, исходя из основного тригонометрического равенства $\sin^2(x)+\cos^2(x)=1$). Эффективность упрощения алгебраических выражений очень сильно зависит от того, есть ли в них корни и каким образом они заданы. Выражения, содержащие корни, упрощаются очень плохо, даже если и имеют простой вид.

Пример 7.14. Неэффективное упрощение выражений

$$\frac{\sqrt{x}+1}{x-1}\ \text{simplify}\ \rightarrow \frac{1}{x^{\frac{1}{2}}+1}$$

Неверно. Правильный ответ $-\frac{1}{\sqrt{x}-1}$.

$$\frac{\sin(2x)}{2\cdot\cos(x)}\ \text{simplify}\ \rightarrow \frac{1}{2}\cdot\frac{\sin(2\cdot x)}{\cos(x)}$$

Неверно. Правильный ответ — $\sin(x)$.

Глядя на приведенные примеры, трудно не согласиться с утверждением, что операция символьного упрощения выражений — это одно из самых слабых мест системы Mathcad. Впрочем, для этого есть вполне объективные и пока трудноразрешимые причины (увы, но компьютеры не умеют мыслить — они лишь очень быстро считают). Поэтому не стоит требовать от программы невозможное, и если вам понадобится решить пример из того же задачника Сканави, пока вам придется сделать это самостоятельно. Вернее, самостоятельно вам придется продумать стратегию упрощения. Mathcad же выполнит за вас вычислительную работу. Продемонстрируем данный подход на примере.

Пример 7.15. Упрощение сложного алгебраического выражения

Пусть нам необходимо упростить громоздкое алгебраическое выражение вида

$$\frac{\left(\sqrt[4]{m}+\sqrt[4]{n}\right)^2+\left(\sqrt[4]{m}-\sqrt[4]{n}\right)^2}{2\cdot(m-n)}\div\frac{1}{\sqrt{m^3}-\sqrt{n^3}}-3\cdot\sqrt{m\cdot n}$$

Оперировать столь большим выражением довольно непросто. Поэтому разделим его на три части и будем проводить упрощение поэтапно. Очевидно, что в первую очередь нужно раскрыть квадраты в числителе первой части выражения:

$$\left(\sqrt[4]{m}+\sqrt[4]{n}\right)^2+\left(\sqrt[4]{m}-\sqrt[4]{n}\right)^2\ \text{expand}\ \rightarrow 2\cdot m^{\frac{1}{2}}+2\cdot n^{\frac{1}{2}}$$

Раскрыв квадраты в числителе, обнаруживаем, что полученная дробь может быть довольно легко упрощена. Однако оператор simplify «не ладит» с корнями и дробными степенями. Поэтому, чтобы от них избавиться, выполним замену $m=x^2$, $n=y^2$. Чтобы в ответе «не всплыли» функции csgn и signum, указываем, что x и y могут быть только положительными (это соответствует истине, так как в исходном выражении имеются корни четвертой степени из m и n).

$$\frac{2\cdot m^{\frac{1}{2}}+2\cdot n^{\frac{1}{2}}}{2(m-n)}\ \Big|\ \begin{matrix}\text{substitute},m=x^2,n=y^2\\ \text{assume},x\geq 0,y\geq 0\\ \text{simplify}\end{matrix}\ \rightarrow \frac{1}{x-y}$$

Заменяем m и n на x и y во второй части исходного выражения, а затем упрощаем его совместно с преобразованной первой частью:

$$\frac{1}{\sqrt{m^3}-\sqrt{n^3}}\ \text{substitute}, m=x^2, n=y^2 \rightarrow \frac{1}{\left(x^6\right)^{\frac{1}{2}}-\left(y^6\right)^{\frac{1}{2}}}$$

$$\frac{1}{x-y}+\frac{1}{\left(x^6\right)^{\frac{1}{2}}-\left(y^6\right)^{\frac{1}{2}}}\ \left|\begin{array}{l}\text{assume}, x\geq 0, y\geq 0\\ \text{simplify}\end{array}\right. \rightarrow x^2+x\cdot y+y^2$$

Выполнив замену, подставляем в выражение третью часть и проводим упрощение:

$$x^2+x\cdot y+y^2-3\sqrt{m\cdot n}\ \left|\begin{array}{l}\text{substitute}, m=x^2, n=y^2\\ \text{assume}, x\geq 0, y\geq 0\\ \text{simplify}\end{array}\right. \rightarrow x^2-2\cdot x\cdot y+y^2$$

Разлагаем полученное выражение на множители и возвращаемся к исходным переменным:

$$x^2-2\cdot x\cdot y+y^2\ \left|\begin{array}{l}\text{factor}\\ \text{substitute}, x=\sqrt{m}, y=\sqrt{n}\end{array}\right. \rightarrow \left(m^{\frac{1}{2}}-n^{\frac{1}{2}}\right)^2$$

Выражения, содержащие функции (логарифмы, тригонометрические), обычно упрощаются довольно плохо. Однако очень часто эффективность упрощения можно резко увеличить, сведя посредством замен выражение к чисто алгебраическому виду. Максимально упростив его алгебраическими приемами, уже будет проще действовать, опираясь на специфические свойства входящих в него функций. Для примера покажем, как можно с помощью Mathcad упростить логарифмическое выражение.

При упрощении логарифмических выражений имеется две сложности. Во-первых, для задания логарифма от x по основанию a в Mathcad служит функция log(a,x). Однако в математике такие логарифмы принято обозначать с использованием нижнего индекса: $\log_a(x)$. Если вид решения важен, то можно создать собственную функцию логарифма по некоторому основанию. Единственная тонкость здесь заключается в том, как ввести в имя нижний индекс (естественно, что использовать оператор извлечения элементов из матрицы нельзя). А сделать это очень просто, нажав клавишу «.» (точка). Во-вторых, зачастую символьный процессор выдает неприемлемый результат по причине того, что все логарифмы он преобразует в отношения натуральных. Избежать этого никак нельзя. Поэтому, если в ответ должны входить десятичные логарифмы или логарифмы по основанию a, то соответствующее преобразование ответа придется проделать самостоятельно.

Пример 7.16. Упрощение логарифмических выражений

Пусть нам необходимо упростить логарифмическое выражение следующего вида:

$$\sqrt{\left(\sqrt{\log_b(a)^4 + \log_a(b)^4 + 2}\right) + 2 - \log_b(a) - \log_a(b)}$$

Чтобы задать его в Mathcad в традиционном представлении, делаем следующие определения:

$$\log_a(x) := \log(x, a) \qquad \log_b(x) := \log(x, b)$$

Созданные пользовательские функции используем в выражении.

Для начала данное выражение должно быть преобразовано, как алгебраическое. Для этого введем замену $t=\log_b(a)$. Очевидно, что тогда $\log_a(b)=1/t$. В итоге получим алгебраический аналог упрощаемого выражения, с которым без проблем «справится» оператор simplify:

$$\sqrt{\sqrt{t^4 + \frac{1}{t^4} + 2} + 2 - t - \frac{1}{t}} \left| \begin{array}{l} \text{assume}, t = \text{real} \\ \text{simplify} \end{array} \right. \rightarrow \frac{\text{signum}(t)\cdot t^2 + \text{signum}(t) - t^2 - 1}{t}$$

Обратите внимание на то, что для того, чтобы символьный процессор смог провести упрощение, необходимо указать, что t принимает действительные значения. Иначе выражение упрощено не будет.

Полученное выражение зависит от знака t. В зависимости от того, положителен t или нет, дальнейшее упрощение даст разный результат. Для положительных t (то есть для a>1 при b>1 и a<1 при b<1):

$$\frac{\text{signum}(t)\cdot t^2 + \text{signum}(t) - t^2 - 1}{t} \left| \begin{array}{l} \text{assume}, t > 0 \\ \text{simplify} \end{array} \right. \rightarrow 0$$

Для отрицательных t (то есть для a<1 при b>1 и a>1 при b<1):

$$\frac{\text{signum}(t)\cdot t^2 + \text{signum}(t) - t^2 - 1}{t} \left| \begin{array}{l} \text{assume}, t < 0 \\ \text{simplify} \end{array} \right. \rightarrow -2\cdot\frac{t^2 + 1}{t}$$

Во второе выражение делаем обратную подстановку $t=\log_b(a)$:

$$-2\cdot\frac{t^2 + 1}{t} \left| \begin{array}{l} \text{substitute}, t = \log_b(a) \\ \text{expand} \end{array} \right. \rightarrow -2\cdot\frac{\ln(a)}{\ln(b)} - \frac{2}{\ln(a)}\cdot\ln(b)$$

Учитывая, что $\ln(m)/\ln(n)=\log_n(m)$, преобразуем ответ так, чтобы в нем присутствовали те же логарифмы, что и в исходном выражении:

$$-2\left(\log_b(a) + \log_a(b)\right)$$

Программа Mathcad может также помочь вам в упрощении тригонометрических выражений. Символьный процессор умеет приводить подобные слагаемые, раскладывать функции от суммы переменных, переходит от функций кратного аргумента к функциям аргумента одинарного. Однако упрощения вроде $2\cdot\sin(x)\cdot\cos(x)=\sin(2x)$ или $\sin(x)\cdot\cos(y)+\cos(x)\cdot\sin(y)=\sin(x+y)$ оператор simplify никогда не осуществляет. Подобные преобразования вы должны проводить самостоятельно. Вообще, главное условие эффективного использования аналитического процессора Mathcad — это четкое понимание того, в чем он может помочь, а что нужно делать «вручную». Пример упрощения тригонометрического выражения совместными силами Mathcad и человека приведен ниже.

Пример 7.17. Упрощение тригонометрического выражения

Пусть нам необходимо упростить следующее тригонометрическое выражение

$$\cos(2\cdot\alpha) - \sin(4\cdot\alpha) - \cos(6\cdot\alpha)$$

Условие упрощения: итоговое выражение должно представлять собой произведение тригонометрических функций.

Для начала нужно привести входящие в выражение функции к одному аргументу — 2α:

$$\cos(2\alpha) - \sin(4\alpha) - \cos(6\alpha)\ \text{simplify} \rightarrow 4\cdot\cos(2\cdot\alpha) - 2\cdot\sin(2\cdot\alpha)\cdot\cos(2\cdot\alpha) - 4\cdot\cos(2\cdot\alpha)^3$$

Исходя из условия упрощения, следующим этапом логично будет сделать разложение на множители:

$$4\cdot\cos(2\cdot\alpha) - 2\cdot\sin(2\cdot\alpha)\cdot\cos(2\cdot\alpha) - 4\cdot\cos(2\cdot\alpha)^3\ \text{factor} \rightarrow$$

$$\rightarrow -2\cdot\cos(2\cdot\alpha)\cdot\left(-2 + \sin(2\cdot\alpha) + 2\cdot\cos(2\cdot\alpha)^2\right)$$

Чтобы упростить выражение в скобках, косинус следует заменить синусом, а затем выполнить разложение на множители:

$$-2 + \sin(2\cdot\alpha) + 2\cdot\cos(2\cdot\alpha)^2\ \text{substitute}, \cos(2\alpha)^2 = 1 - \sin(2\alpha)^2 \rightarrow \sin(2\cdot\alpha) - 2\cdot\sin(2\cdot\alpha)^2$$

$$\sin(2\cdot\alpha) - 2\cdot\sin(2\cdot\alpha)^2\ \text{factor} \rightarrow -\sin(2\cdot\alpha)\cdot(-1 + 2\cdot\sin(2\cdot\alpha))$$

В итоге получим следующее выражение:

$$2\cdot\cos(2\cdot\alpha)\cdot\sin(2\cdot\alpha)\cdot(-1 + 2\cdot\sin(2\cdot\alpha))$$

С учетом того, что $2\cdot\sin(\alpha)\cdot\cos(\alpha)=\sin(2\cdot\alpha)$, выражение можно немного упростить:

$$\sin(4\alpha)\cdot((-1 + 2\cdot\sin(2\cdot\alpha))$$

Чтобы преобразовать сумму в скобках в произведение, осуществим следующие тождественные преобразования:

$$-1 + 2\cdot\sin(2\cdot\alpha) = 2\left(\sin(2\alpha) - \frac{1}{2}\right) = 2\cdot\left(\sin(2\alpha) - \sin\left(\frac{\pi}{6}\right)\right)$$

Формулу преобразования разности синусов в произведение тригонометрических функций можно найти в любом справочнике:

$$2\cdot\cos\left(\frac{x+y}{2}\right)\cdot\sin\left(\frac{x-y}{2}\right)\ \text{substitute}, x = 2\alpha, y = \frac{\pi}{6} \rightarrow 2\cdot\cos\left(\alpha + \frac{1}{12}\cdot\pi\right)\cdot\sin\left(\alpha - \frac{1}{12}\cdot\pi\right)$$

Окончательно имеем:

$$4\sin(4\alpha)\cdot\left(\cos\left(\frac{1}{12}\cdot\pi + \alpha\right)\cdot\sin\left(\alpha - \frac{1}{12}\cdot\pi\right)\right)$$

С помощью оператора simplify можно упрощать значения численных выражений. При этом если какое-нибудь число в выражении содержит десятичную точку, то ответ будет также найден в виде десятичной дроби. Чтобы этого не произошло, все десятичные дроби нужно перевести в обыкновенные с использованием оператора factor.

Пример 7.18. Упрощение численного выражения

Если в выражение входит десятичная дробь, расчет будет произведен приблизительно:

$$\frac{2^{-2}+5^{0}}{\left(\frac{1}{2}\right)^{-2}-5\cdot(-2)^{-2}+\left(\frac{2}{5}\right)^{-2}+\sqrt{2}}+4.35\ \text{simplify}\ \rightarrow 4.4700282664181472302$$

Чтобы получить ответ в аналитическом виде, переводим десятичные дроби в простые:

$$4.35\ \text{factor}\ \rightarrow \frac{87}{20}$$

$$\frac{2^{-2}+5^{0}}{\left(\frac{1}{2}\right)^{-2}-5\cdot(-2)^{-2}+\left(\frac{2}{5}\right)^{-2}+\sqrt{2}}+\frac{87}{20}\ \text{simplify}\ \rightarrow \frac{1}{20}\cdot\frac{808+87\cdot 2^{\frac{1}{2}}}{9+2^{\frac{1}{2}}}$$

Глава 8. Решение уравнений и систем уравнений

Эта глава довольно необычная. В ней мы не только изучим то, как в Mathcad можно решать уравнения и системы уравнений, но и ознакомимся с алгоритмами, используемыми программой. Дело в том, что если не понимать основных идей этих алгоритмов, эффективно использовать соответствующие функции практически невозможно. Знакомясь с очередным численным методом, мы, как правило, будем самостоятельно реализовывать его на языке программирования Mathcad. Этим мы убьем сразу двух зайцев. Во-первых, при этом идеи численного метода усвоятся лучше. Во-вторых, это будет очень и очень неплохой практикой программирования.

8.1. Решение уравнений

В Mathcad реализовано три принципиально отличающихся друг от друга подхода к решению уравнений: использование символьных преобразований, применение численных алгоритмов и графический метод. В этой главе мы подробно разберем их все.

Почти наверняка все задачи, с которыми вам приходилось сталкиваться в школе или изучать в университете, решались символьно. То есть вы тем или иным образом преобразовывали и упрощали выражения, использовали какие-то стандартные формулы и методы, умножали, делили, сокращали — и в результате приходили к какому-то несложному аналитическому результату. Так, например, при решении квадратного уравнения вы использовали формулы Виета; пытаясь найти корни кубического уравнения, вы разлагали выражение на линейные множители (или, в крайнем случае, использовали формулу Кардано); для бикубических уравнений прибегали к замене. Решение задач аналитически имеет массу преимуществ перед решением численным. Во-первых, в этом случае ответ может быть вычислен без какой-либо погрешности. Во-вторых, при получении результата в виде аналитического выражения имеются куда более широкие возможности его последующего использования (например, в качестве формулы). В-третьих, числовой результат, полученный в символьном виде, куда более нам понятен, чем десятичная дробь, получаемая при использовании численных методов.

Увы, но аналитическим решением обладает очень ограниченное количество задач. Чтобы найти корни уравнения в виде выражения, требуется выразить одну переменную

через все остальные (или коэффициенты). Сделать же это обычно можно только в том случае, если уравнение включает переменные невысокой степени и не содержит разнородных функций. Такие уравнения специально подбираются в учебниках, и их можно более или менее просто решить на бумаге. Но на практике часто существует необходимость находить корни таких уравнений, пытаться решать которые с помощью традиционных приемов символьной алгебры совершенно бесперспективно. Численно же можно решить практически любое уравнение. Однако получаемое при использовании численного метода значение корня в виде числа с плавающей точкой куда менее информативно, чем выражение аналитического решения. Опыт показывает, что простые уравнения лучше решать символьно, более сложные — численно. Обычно численный метод используется, если Mathcad не сможет решить уравнение аналитически. Впрочем, с учетом склонности символьного процессора делать ошибки (чаще всего он теряет часть корней), в ответственных случаях аналитическое решение стоит проверять с помощью численного метода как значительно более надежного.

Изредка попадаются такие уравнения, которые нельзя решить ни аналитически (так как они слишком сложны), ни численно (чаще всего потому, что соответствующая функция не является непрерывной). В таких случаях решение ищут по графику, используя специальные инструменты панели Graph (Графические). Данный способ довольно трудоемок, однако он способен обеспечить точность, мало уступающую точности численных методов.

8.1.1. Аналитическое решение уравнений

Для аналитического решения уравнений в системе Mathcad существует специальный оператор solve (Решить). Чтобы найти с его использованием корни уравнения, выполните следующую последовательность действий.

1. Введите оператор solve (Решить) с помощью одноименной команды панели Symbolic (Символьные).
2. В левом маркере задайте вид решаемого уравнения. В качестве знака равенства следует использовать логическое равенство (Bold Equal — вводится сочетанием Ctrl+=). Если уравнение приведено к стандартному виду, то достаточно будет определить лишь его левую часть. При этом выражение будет приравнено к нулю автоматически. Также в левый маркер можно внести и имя функции — в этом случае будут найдены выражения, определяющие ее нули. Форма записи уравнения через функцию удобна в том случае, если оно имеет большую длину.
3. В правый маркер внесите переменную, относительно которой должно быть решено уравнение.

Ответ оператор solve возвращает в виде выражения (численного или буквенного), которое вполне можно использовать в дальнейших вычислениях. Если решений имеется несколько, то возвращается содержащий их вектор.

При символьном решении уравнений нет особой разницы, сколько переменных содержит уравнение. Ответ ищется в виде выражения, и поэтому для системы неважно, будет ли оно содержать буквенные или численные элементы. Исходя из этого вы можете найти корни как уравнения нескольких переменных, так и уравнения с параметрами или буквенными коэффициентами.

Рассмотрим особенности решения каждого из типов уравнений, встречающихся на практике.

Лучше всего Mathcad справляется с поиском корней алгебраических полиномов. Причем находятся все корни — как действительные, так и мнимые. Общее их количество, исходя из знаменитой теоремы алгебры (теорема Гаусса), равно n, где n — степень полинома. Например, в случае полинома третьей степени мы получим три решения.

Пример 8.1. Поиск всех решений уравнения, являющегося алгебраическим полиномом

$$(x+1)\cdot\left(x^2+2\right)+(x+2)\cdot\left(x^2+1\right)=2 \text{ solve},x \rightarrow \begin{pmatrix} -1 \\ \frac{-1}{4}+\frac{1}{4}\cdot i\cdot 15^{\frac{1}{2}} \\ \frac{-1}{4}-\frac{1}{4}\cdot i\cdot 15^{\frac{1}{2}} \end{pmatrix}$$

На практике обычно бывает необходимо найти только действительные корни полинома. Увы, но указать Mathcad, что поиск решений должен быть осуществлен только в пределах действительной области, невозможно. Бессмысленно для этого пытаться использовать оператор assume (подробно о нем читайте в гл. 7), присвоив переменной константу real или задав область поиска от $-\infty$ до ∞ посредством модификатора RealRange. Дело в том, что операторы assume и solve не сочетаются — при попытке их совместить solve прекратит работу. Это одна из самых известных недоработок системы Mathcad — и обойти ее невозможно. Так что, если комплексные корни вас не интересуют, просто не обращайте на них внимания.

В алгебре доказано, что аналитические выражения существуют лишь для корней полиномов до пятой степени. Однако это не означает, что оператор solve не сможет найти решения полиномиального уравнения более высокой степени. Mathcad может решать уравнения любой степени — но приблизительно. Если символьный процессор обнаружит, что старший член в полиноме возведен в степень, превышающую четыре, им будет задействован численный метод. В результате корни будут найдены, но не в форме выражений, а в форме чисел с плавающей точкой. Чтобы понять разницу, изучите пример 8.2.

Пример 8.2. Решение полиномиального уравнения низкой и высокой степеней

Уравнение второй степени:

$$(x+2)^3-(x+13)^3=1 \text{ solve},x \rightarrow \begin{pmatrix} \frac{-15}{2}+\frac{1}{22}\cdot i\cdot 4895^{\frac{1}{2}} \\ \frac{-15}{2}-\frac{1}{22}\cdot i\cdot 4895^{\frac{1}{2}} \end{pmatrix}$$

Уравнение пятой степени:

$$(x-1)\cdot(x-2)\cdot(x-3)\cdot(x-4)\cdot(x-5)=1 \text{ solve}, x \rightarrow \begin{pmatrix} 1.0459203853766219542 \\ 1.8490158267250284186 \\ 3.2758341933170920482 \\ 3.7907343035445203650 \\ 5.0384952910367372140 \end{pmatrix}$$

В Mathcad имеется функция polyroots (о ней мы поговорим ниже), которая служит для нахождения корней полинома. В основе нее лежит численный метод, поэтому она позволяет находить корни полиномов любой степени. В общем, в случае полиномов степени 5 и выше между использованием polyroots и оператора solve нет столь уж ощутимых различий. Единственное, при применении solve решение находится более точно — вплоть до 20-го знака мантиссы. Точность же работы polyroots ограничена 15-ю знаками. Кроме того, polyroots не может оперировать с очень большими или очень малыми (по абсолютной величине) значениями.

Без особых трудностей справляется Mathcad и с алгебраическими уравнениями более сложного вида, содержащими разного рода корни. Главная проблема, которая при этом может возникнуть, это чрезвычайно большие выражения ответа, которые могут занимать несколько страниц и не поддаваться оптимизации посредством оператора simplify. Если вы столкнетесь с такой сложностью, то пересчитайте ответ в число с плавающей точкой с нужным уровнем точности. Служит для этого оператор float (или аналогичная команда меню Symbolics), принимающий в качестве параметра количество знаков мантиссы, которые должны быть вычислены (предел точности — 4000 знаков мантиссы).

Пример 8.3. Решение алгебраических уравнений сложного вида

$$\sqrt[3]{9-\sqrt{x+1}}+\sqrt[3]{7+\sqrt{x+1}}=4 \text{ solve}, x \rightarrow 0$$

Ответ в следующем уравнении получается слишком громоздким, поэтому пересчитываем его в десятичную дробь с точностью до 40 знаков

$$\frac{1}{x+\sqrt{x+2}}=\sqrt{x+2}+1 \left| \begin{matrix} \text{solve}, x \\ \text{float}, 40 \end{matrix} \right. \rightarrow -.6823278038280193273694837397110482568 9$$

В некоторых случаях ответ может получаться столь большим, что его отображение станет невозможным. При этом будет выдано сообщение об ошибке: Discarding huge result (Выбраковка огромного результата), а сам результат будет помещен в текстовой форме в буфер обмена. Однако как-то воспользоваться им будет практически невозможно. Поэтому, если подобная ситуация возникнет, для решения уравнения лучше применить один из численных алгоритмов (или пересчитать ответ в число с плавающей точкой с помощью оператора float).

Для решения уравнений оператором solve проводятся аналитические преобразования, по причине чего зачастую корни можно найти в общем виде (то есть выразить их через буквенные коэффициенты). Аналогично solve справится с несложными уравнениями с параметрами. Эта его возможность особенно полезна при решении физических и технических задач, так как в соответствующие уравнения обычно входят многочисленные константы.

Пример 8.4. Решение уравнения с параметром

$$a\cdot x^4 - x^3 + a^2\cdot x - a \text{ solve}, x \rightarrow \begin{bmatrix} (-a)^{\frac{1}{3}} \\ \frac{-1}{2}\cdot(-a)^{\frac{1}{3}} - \frac{1}{2}\cdot i\cdot 3^{\frac{1}{2}}\cdot(-a)^{\frac{1}{3}} \\ \frac{-1}{2}\cdot(-a)^{\frac{1}{3}} + \frac{1}{2}\cdot i\cdot 3^{\frac{1}{2}}\cdot(-a)^{\frac{1}{3}} \\ \frac{1}{a} \end{bmatrix}$$

Весьма неплохо, хотя и заметно хуже по сравнению с алгебраическими уравнениями, справляется символьный процессор с показательными и логарифмическими уравнениями. Для решения логарифмических уравнений нужно запомнить, что натуральный логарифм задается функцией ln, десятичный — функцией log. Для задания логарифма по основанию x также служит функция log. Однако в этом случае она принимает два параметра: первый соответствует величине, от которой нужно найти логарифм, второй предназначен для указания основания.

Решая логарифмическое или показательное уравнение, ответ оператор solve обычно выдает в виде сложного выражения из чисел и логарифмов от чисел. Чтобы привести его к более простому виду, следует использовать оператор simplify.

Пример 8.5. Решение логарифмических и показательных уравнений

$$\log\left(9 - 2^x, 2\right) = 10^{\log(3-x)} \left| \begin{matrix} \text{solve}, x \\ \text{simplify} \end{matrix} \right. \rightarrow 0$$

$$7^x\cdot\left(\sqrt{2}\right)^{2x^2-6} - \left(\frac{7}{4}\right)^x = 0 \left| \begin{matrix} \text{solve}, x \\ \text{simplify} \end{matrix} \right. \rightarrow \begin{pmatrix} 1 \\ -3 \end{pmatrix}$$

Ответ для нижележащего уравнения определяется в неявной форме, однако с легкостью можно подсчитать его приблизительное значение:

$$\frac{2\cdot x + 10}{4} = \frac{8}{2^{x-2}} \text{ solve}, x \rightarrow \frac{-5\cdot\ln(2) + W(2048\cdot\ln(2))}{\ln(2)}$$

$$\frac{-5\cdot\ln(2) + W(2048\cdot\ln(2))}{\ln(2)} \text{ float} \rightarrow 3.0000000000000000$$

В последнем уравнении примера получен весьма странный результат. Что такое «W», откуда оно взялось и как из него можно получить более приемлемое для анализа выражение — ответы на эти вопросы совсем не очевидны. Поэтому для их выяснения обратимся к справочной системе Mathcad.

В специальной строке раздела поиска вводим «W». В результате система находит целый список разделов, где упоминается данная функция. Однако, исходя из темы данного раздела, более всего нам подойдет статья Special functions and syntax used in symbolic results (Специальные функции и синтаксис, используемые в символьных результатах). Данная статья содержит список всех специальных функций и констант, которые могут встречаться в ответах при проведении операций символьных преобразований. Таких функций и констант довольно много — 22. В данном списке имеется и упоминание про функцию W. Оказывается, данная функция есть не что иное, как функция Ламберта. Чтобы получить про нее более детальную информацию, открываем шпаргалку Special symbolic functions, ссылка на которую имеется внизу статьи. В этой шпаргалке описываются все неинтегральные символьные функции. Из нее мы узнаем, что функция Ламберта — это функция, обратная функции $f(x)=x\cdot e^x$. Чтобы найти значение W, можно использовать функцию root (корень). Эта функция реализует, в зависимости от формы записи, два численных алгоритма решения уравнений, и о ней мы поговорим весьма подробно ниже. Однако применение root — это не лучший из возможных подходов. Гораздо техничнее использовать оператор float, так как это проще, а также при этом точность найденного значения может быть сколь угодно высокой.

Специальные функции могут быть получены в ответах не только при аналитическом решении уравнений, но при проведении интегрирования, интегральных преобразованиях. В любом случае для приведения ответа к числовой форме используется оператор float.

Хуже всего символьный процессор Mathcad решает тригонометрические уравнения. Как известно, большинство таких уравнений имеет бесконечное множество корней и описываются они с помощью специальных выражений, содержащих некоторый целочисленный параметр. Например, решение уравнения $\sin(x)=0$ запишется как $x=\pi\cdot N, N\in R$ (где R — множество целых чисел). Mathcad же находит корни только на промежутке одного периода соответствующей уравнению тригонометрической функции. Естественно, удовлетворительным такое решение считать вряд ли возможно. Но в некоторых случаях им все же можно воспользоваться.

Пример 8.6. Решение тригонометрических уравнений

$$\sin(x)=0 \text{ solve}, x \rightarrow 0 \qquad \cos(x)=0 \text{ solve}, x \rightarrow \frac{1}{2}\cdot\pi$$

$$\sin(a\cdot x)+\cos(b\cdot x)=0 \text{ solve}, x \rightarrow \frac{-1}{2}\cdot\frac{\pi}{a-b}$$

Главная проблема, связанная с решением тригонометрических уравнений в Mathcad, заключается отнюдь не в том, что система находит лишь один корень из бесконечного множества. Она связана даже не с тем, что ответ зачастую выдается в виде огромного выражения или содержит специальные функции. Все гораздо хуже: решая тригонометрические уравнения, Mathcad нередко ошибается. Особенно вероятна ошибка при наличии параметров и разнородных функций.

Пример 8.7. Неверное аналитическое решение уравнения

$$\sin(x)^3+\cos(a\cdot x)^3=1 \left|\begin{matrix}\text{solve}, x\\ \text{float}, 10\end{matrix}\right. \rightarrow \begin{pmatrix} .9168683843 \\ -.3269639712+.6736886743\cdot i \\ -.3269639712-.6736886743\cdot i \end{pmatrix}$$

Проверяем верность решения, передавая полученные значения корней соответствующей уравнению функции. Если они были найдены правильно, значение функции должно быть равно 0:

$$f(x,a) := \sin(x)^3 + \cos(a\cdot x)^3 - 1$$

$$f(.9168683843,1) = -0.275$$

$$f(-.3269639712+.6736886743\cdot i,\ 2) = -1.991 + 7.398i$$

Функция не равна 0 даже приблизительно. Вывод: корни были определены неверно.

Чтобы убедиться, что в приведенном выше примере уравнение было решено неверно, совсем не обязательно было создавать соответствующую уравнению функцию. Достаточно было обратить внимание на то, что параметр a не входит в выражения ответа. То же, что корни должны зависеть от параметра, очевидно. Вообще же, рекомендую вам воздержаться от символьного решения тригонометрических уравнений напрямую. Это одна из тех интеллектуально нагруженных задач, с которыми компьютеры справляются плохо. Если же вы все-таки используете оператор solve для решения тригонометрического уравнения, то обязательно проверяйте ответ так, как мы это сделали выше (а еще лучше строить график).

То, что Mathcad не силен в решении тригонометрических уравнений, — это печальная реальность. Но это не значит, что такие уравнения придется решать на бумаге. Конечно, в одно действие и полностью отключив голову тригонометрическое уравнение в Mathcad вы вряд ли решите. Но если вы примете деятельное участие в процессе решения, разрабатывая стратегию самостоятельно и доверяя Mathcad лишь вычислительную часть работы, то с самыми сложными уравнениями удастся совладать с гораздо меньшими усилиями, чем при проведении всех операций на бумаге. Основная идея следующая: используя Mathcad, сводим преобразованиями разного рода сложное уравнение к элементарному, вроде cos(x)=0, а затем, уже без участия программы, самостоятельно решаем его. То, как это делается на практике, показано в примере 8.8.

Пример 8.8. Поэтапное решение тригонометрических уравнений

Задача 1. Решить уравнение вида

$$\sin^6(2t) + \cos^6(2t) - \frac{3}{2}\cdot\left(\sin^4(2t) + \cos^4(2t)\right) + \frac{1}{2}\cdot(\sin(t) + \cos(t)) = 0$$

Данное уравнение содержит функции от 2t и t. Нужно перейти к функциям одного аргумента, а затем упростить полученное выражение. Одновременно обе эти операции выполнит оператор simplify:

$$E := \sin(2t)^6 + \cos(2t)^6 - \frac{3}{2}\cdot\left(\sin(2t)^4 + \cos(2t)^4\right) + \frac{1}{2}\cdot(\sin(t) + \cos(t))$$

$$E\ \text{simplify} \rightarrow \frac{-1}{2} + \frac{1}{2}\cdot\sin(t) + \frac{1}{2}\cdot\cos(t)$$

Мы получили очень простое уравнение, которое далее нужно решать самостоятельно. Для начала сократим 1/2, после чего перенесем единицу вправо:

$$\sin(t) + \cos(t) = 1$$

Возведем обе части уравнения в квадрат и воспользуемся основным тригонометрическим тождеством:

$$1 + 2\cdot\sin(t)\cdot\cos(t) = 1$$

Сократим единицы и используем формулу для синуса двойного аргумента:

$$\sin(2t) = 0$$

Так как синус равен нулю в точках $\pi \cdot k$ (k — любое целое число), то решением уравнения будет $t=(\pi \cdot k)/2$.

Задача 2. Решить уравнение вида:

$$\sqrt{3}\cdot\sin(t) = \sqrt{2\cdot\sin(t)^2 - \sin(2t) + 3\cdot\cos(t)^2}$$

Первым нашим шагом будет то, что функции двойного аргумента мы представим через функции одинарного аргумента. Сделать это можно, задействовав оператор simplify:

$$E := \sqrt{3}\cdot\sin(t) - \sqrt{2\cdot\sin(t)^2 - \sin(2t) + 3\cdot\cos(t)^2}$$

$$E := E \text{ simplify } \rightarrow 3^{\frac{1}{2}}\cdot\sin(t) - \left(-2\cdot\sin(t)\cdot\cos(t) + \cos(t)^2 + 2\right)^{\frac{1}{2}}$$

Чтобы в уравнение входили лишь функции одного типа, заменим косинусы выражением, содержащим только синус:

$$E := E \text{ substitute}, \cos(t) = \sqrt{1-\sin(t)^2} \rightarrow 3^{\frac{1}{2}}\cdot\sin(t) - \left[-2\cdot\left(1-\sin(t)^2\right)^{\frac{1}{2}}\cdot\sin(t) + 3 - \sin(t)^2\right]^{\frac{1}{2}}$$

Mathcad плохо решает тригонометрические уравнения. А вот с алгебраическими уравнениями программа справляется очень хорошо. Отсюда вытекает следующая стратегия: решаем полученное уравнение как алгебраическое относительно sin(t). При этом мы узнаем, какие значения принимает синус. А уж самостоятельно справиться с системой уравнений типа $\sin(t)=y$ не составит труда.

$$E \text{ solve}, \sin(t) \rightarrow \frac{1}{2}\cdot 2^{\frac{1}{2}}$$

Нам повезло. Уравнение имеет только одно решение. Значит, осталось только интерпретировать полученный результат. Сделав это, получим следующий ответ: $t1=\pi/4+2\cdot\pi\cdot k$ и $t2=3\cdot\pi/4+2\cdot\pi\cdot k$ (k — любое целое число).

Иногда помощь символьному процессору Mathcad нужно оказывать и при решении алгебраических, показательных и логарифмических уравнений. Если система не справится с задачей, преобразованиями и заменами нужно постараться свести выражение к тождественному, но более простому. Например, если в уравнение входят неизвестные в дробной степени, следует подобрать такую замену, чтобы иррациональность исчезла (Mathcad очень «не любит» иррациональные выражения). Если программа не способна решить показательное уравнение, его левую и правую части стоит прологарифмировать. Зачастую отличный эффект дает банальное упрощение выражения уравнения с помощью оператора simplify с указанием области изменения переменной посредством оператора assume. В общем, различных приемов можно придумать очень много. Главное — сразу не сдаваться, если оператор solve не сможет найти корни уравнения. Помогая программе, направляя ее, вы сможете решить 99 % уравнений, которые имеют аналитическое решение.

Пример 8.9. Поэтапное решение алгебраических и показательных уравнений

Задача 1. Решить алгебраическое уравнение вида:

$$8.4\cdot\sqrt[12]{x^{-7}} - 0.2\cdot\sqrt[4]{x^{-1}\cdot\sqrt[3]{x^{2}}} = \sqrt[12]{x^{11}}$$

Если мы попытаемся решить данное уравнение, не проводя никаких преобразований, то аналитического ответа программа найти не сможет. На это есть две причины. Во-первых, в качестве множителей в левой части уравнения выступают десятичные дроби. Символьный процессор плохо оперирует такими числами. Поэтому десятичные дроби нужно перевести в обыкновенные. Сделать это позволяет оператор factor. Во-вторых, выражение уравнения слишком сложно. Его необходимо упростить, задействовав оператор simplify. Чтобы упрощение было эффективным, нужно указать область изменения x (по умолчанию система считает все неизвестные комплексными величинами). Так как в уравнении несколько раз вычисляются корни четной степени из x в нечетной степени, то, следовательно, x не может быть отрицательной величиной.

$$8.4\cdot\sqrt[12]{x^{-7}} - 0.2\cdot\sqrt[4]{x^{-1}\cdot\sqrt[3]{x^{2}}} - \sqrt[12]{x^{11}} \;\Bigg|\; \begin{matrix}\text{factor}\\ \text{assume},x>0\\ \text{simplify}\end{matrix} \rightarrow \frac{-1}{5}\cdot\frac{-42+x^{\frac{1}{2}}+5\cdot x^{\frac{3}{2}}}{x^{\frac{7}{12}}}$$

Полученное в результате проведенных преобразований уравнение Mathcad решит без проблем:

$$\frac{-1}{5}\cdot\frac{-42+x^{\frac{1}{2}}+5\cdot x^{\frac{3}{2}}}{x^{\frac{7}{12}}}\ \text{solve},x\rightarrow 4$$

Задача 2. Решить алгебраическое уравнение вида:

$$5\cdot\sqrt[3]{x\cdot\sqrt[5]{x}} + 3\cdot\sqrt[5]{x\cdot\sqrt[3]{x}} = 8$$

Если попробовать решить данное уравнение напрямую, то Mathcad сможет найти пять корней, четыре из которых комплексные и только один — действительный:

$$5\cdot\sqrt[3]{x\cdot\sqrt[5]{x}} + 3\cdot\sqrt[5]{x\cdot\sqrt[3]{x}} = 8 \;\Bigg|\; \begin{matrix}\text{solve},x\\ \text{float},5\end{matrix} \rightarrow \begin{pmatrix} 1. \\ -2.0657 + 5.4484\cdot i \\ 2.0657 - 5.4484\cdot i \\ -2.0657 - 5.4484\cdot i \\ 2.0657 + 5.4484\cdot i \end{pmatrix}$$

Однако проверка показывает, что найденное Mathcad решение верно лишь для корня x=1. Комплексные же корни определены неправильно, в чем можно легко убедиться, выполнив подстановку:

$$5\cdot\sqrt[3]{x\cdot\sqrt[5]{x}}+3\cdot\sqrt[5]{x\cdot\sqrt[3]{x}}\ \text{substitute},x=\frac{512}{78125}\cdot\left(-20+10\cdot i\cdot 6^{\frac{1}{2}}\right)^{\frac{1}{2}}\cdot\left(\frac{788}{5}+\frac{26}{5}\cdot i\cdot 6^{\frac{1}{2}}\right)\ \text{float},3 \rightarrow 13.5+6.23\cdot i$$

Проверка по графику показывает, что x=1 это не единственный корень уравнения. Также корнем, вероятно, является x=−1 (рис. 8.1).

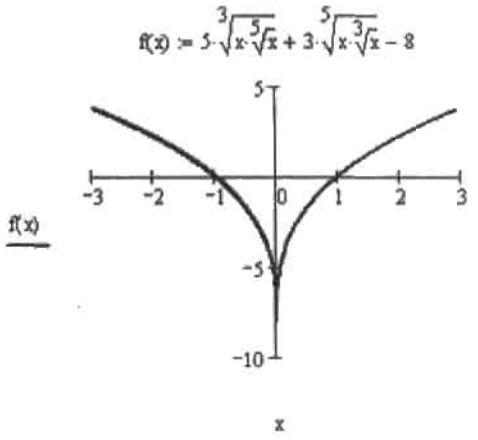


Рис. 8.1. Соответствующая уравнению функция четная (то есть ее участок правее оси Y является зеркальным отображением участка, лежащего левее данной оси). Это означает, что у корня x=1 будет парный корень x=−1

Можно ли преобразовать решаемое уравнение так, чтобы символьный процессор с ним справился? Сложность данного уравнения заключается в его иррациональности. Mathcad довольно плохо оперирует с выражениями, содержащими корни и дробные степени, поэтому от них нужно стремиться избавляться. Для этого нужно найти такую замену, которая сделала бы иррациональное выражение рациональным (или хотя бы упростила его). В нашем случае хорошей заменой будет $x=t^{15/2}$.

$$5\cdot\sqrt[3]{x\cdot\sqrt[5]{x}}+3\cdot\sqrt[5]{x\cdot\sqrt[3]{x}}-8\ \text{substitute},x=t^{\frac{15}{2}} \rightarrow 5\cdot\left[\left(\frac{15}{t^{2}}\right)^{\frac{6}{5}}\right]^{\frac{1}{3}}+3\cdot\left[\left(\frac{15}{t^{2}}\right)^{\frac{4}{3}}\right]^{\frac{1}{5}}-8$$

Полученное в результате замены выражение следует упростить вручную, так как оператор simplify (даже в сочетании с assume) с этой задачей не справится (вероятно, из-за слишком большого количества степеней).

$$\frac{15}{2}\cdot\frac{6}{5}\cdot\frac{1}{3} \rightarrow 3 \qquad \frac{15}{2}\cdot\frac{4}{3}\cdot\frac{1}{5} \rightarrow 2$$

$$5\cdot t^3 + 3\cdot t^2 = 8$$

В результате замены мы получили простое алгебраическое уравнение третьей степени, которое Mathcad легко решит.

$$5\cdot t^3 + 3\cdot t^2 = 8 \text{ solve}, t \rightarrow \begin{pmatrix} 1 \\ \frac{-4}{5} + \frac{2}{5}\cdot i\cdot 6^{\frac{1}{2}} \\ \frac{-4}{5} - \frac{2}{5}\cdot i\cdot 6^{\frac{1}{2}} \end{pmatrix}$$

Ограничившись действительным корнем, находим, при каких x переменная t принимает значение 1.

$$x^{\frac{2}{15}} = 1 \text{ solve}, x \rightarrow \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

Найденные корни соответствуют правильному решению уравнения.

Задача 3. Решить показательное уравнение следующего вида:

$$\left(2+\sqrt{3}\right)^{x^2-2x+1} + \left(2-\sqrt{3}\right)^{x^2-2x-1} = \frac{4}{2-\sqrt{3}}$$

Оператор solve способен решить данное уравнение, но лишь приблизительно (корни выдаются в формате чисел с плавающей точкой):

$$\left(2+\sqrt{3}\right)^{x^2-2x+1} + \left(2-\sqrt{3}\right)^{x^2-2x-1} = \frac{4}{2-\sqrt{3}} \text{ solve}, x \rightarrow \begin{pmatrix} 1. \\ 1. \end{pmatrix}$$

То, что оператор solve определил корень только приблизительно — это не беда, так как он имеет целочисленное значение. Хуже другое. Проверка по графику показывает, что у уравнения должно быть три корня, а не один (рис. 8.2).

$$f(x) := \left(2+\sqrt{3}\right)^{x^2-2x+1} + \left(2-\sqrt{3}\right)^{x^2-2x-1} - \frac{4}{2-\sqrt{3}}$$

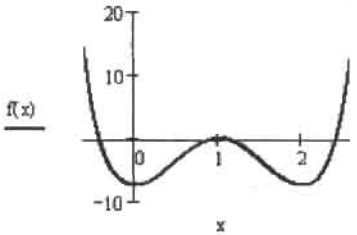


Рис. 8.2. Кривая дважды пересекает ось X, а в точке x=1 касается ее. Таким образом, у уравнения будет три корня. Так как кривая симметрична относительно прямой x=1, два корня будут сопряжены и будут иметь вид 1+a и 1–a, где a — некоторое число

Чтобы упростить уравнение, используем тот факт, что подстепенные выражения в нем являются сопряженными и поэтому их перемножение даст целое число:

$$\left(2+\sqrt{3}\right)\cdot\left(2-\sqrt{3}\right)\text{expand}\rightarrow 1$$

Отсюда:

$$2-\sqrt{3}=\frac{1}{2+\sqrt{3}}$$

Используя данное равенство, так преобразуем уравнение, чтобы в нем был только один вид подстепенного выражения. Упрощенное таким образом уравнение оператор solve без проблем решит:

$$\left(2+\sqrt{3}\right)^{x^2-2x+1}+\left(2-\sqrt{3}\right)^{x^2-2x-1}=\frac{4}{2-\sqrt{3}}\left|\begin{array}{l}\text{substitute},2-\sqrt{3}=\dfrac{1}{2+\sqrt{3}}\\ \text{solve},x\\ \text{simplify}\end{array}\right.\rightarrow\begin{pmatrix}1\\ 1\\ \frac{1}{2}\\ 2^{\frac{1}{2}}+1\\ 1-2^{-\frac{1}{2}}\end{pmatrix}$$

Если, несмотря на все ухищрения, аналитически решить уравнение не удастся, следует попытаться найти его решения хотя бы численно. Как это сделать, подробно описывается в следующем разделе.

8.1.2. Численное решение уравнений

Далеко не все уравнения можно решить аналитически. Это можно сделать лишь в случае отдельных, «удобных» уравнений. На практике же приходится работать с уравнениями, включающими разнородные функции (что, как правило, автоматически означает невозможность символьного решения) или с очень неудобными коэффициентами. Справиться со многими из них привычным преобразованием или заменами никак не получится. Что же делать?

Естественно, если вам попадется такое «нерешаемое» уравнение, вы попытаетесь просто подобрать корни. Для этого вы будете подставлять какие-то значения переменной (выбор которых, скорее всего, в основном будет определяться вашей интуицией) в надежде на то, что какое-то из них обратит уравнение в нуль. К сожалению, такой способ весьма малоэффективен на практике: подставив десяток-другой значений, вы, почти наверняка, предпочтете удовлетвориться мыслью о том, что данное уравнение не имеет решений вовсе, чем продолжать эту чрезвычайно неинтересную работу.

Но если у вас есть компьютер с системой Mathcad, то никаких проблем с поиском решения не будет, вне зависимости от сложности уравнения. Конечно, аналитическое решение компьютер не всегда найдет даже для очень простых уравнений (в этом пока человек значительно превосходит машину), но зато, ввиду колоссальной, по сравнению с человеческой, скоростью обработки данных, очень эффективными становятся так называемые численные методы.

В основе всех численных методов решения уравнений лежит принцип подбора. Но, в отличие от подбора возможных корней человеком, в численных методах этот процесс является строго направленным. Основным понятием численных методов является итерация. Итерация — это, буквально, «повторение», то есть все численные методы решения уравнений построены на принципе повторения одного и того же действия или последовательности действий, результатом которых является большее или меньшее приближение некоторого промежуточного значения переменной к корню. Поэтому численные методы решения уравнений называются еще итерационными. Количество необходимых итераций определяется скоростью сходимости алгоритма (эффективностью) и особенностями уравнения. Эффективность различных численных алгоритмов может различаться весьма значительно, но это не значит, что всегда стоит избирать из них наиболее быстрый: чем выше скорость сходимости к решению, тем, как правило, выше чувствительность алгоритма к разного рода трудностям (многочисленным экстремумам, разрывам или недифференцируемости в некоторых точках).

Для численного поиска решений уравнений с одним неизвестным в Mathcad существует специальная встроенная функция root (с англ. — корень). Функция эта может использоваться в двух различных формах, при этом реализуются разные численные алгоритмы. Так, если определена только одна точка приближения к корню, поиск решений будет осуществляться так называемым методом секущих (или, в случае неудачи использования метода секущих, его более эффективной модификацией — методом Мюллера (Muller)). Если же задан интервал, на котором предположительно локализовано решение, то поиск его будет осуществлен с применением двух модификаций метода Больцано деления пополам (метода бисекции) — методов Риддера (Ridder) и Брента (Brent). Идеи всех этих методов, их достоинства и слабые места мы подробно обсудим в этом разделе.

Если необходимо найти корень некоторого уравнения, причем известен интервал, в котором он локализован, проще всего использовать функцию root с четырьмя аргументами: root(f(x),x,a,b), где f(x) — функция, определяющая уравнение, x — переменная, a и b — границы интервала локализации. Обязательным условием является то, что значения функции на концах интервала должны быть противоположных знаков. Это связано с особенностью используемых root алгоритмов. Если нарушить это условие, система выдаст сообщение об ошибке. Также функция f(x) должна быть ограниченной и непрерывной на промежутке (a, b). Если же она имеет точку разрыва на этом промежутке, то численный метод может не сойтись.

Пример 8.10. Численное решение уравнения при известном интервале локализации корня

Попробуем протестировать функцию root и для этого найдем корни какого-нибудь уравнения, точное решение которого очевидно. К примеру, возьмем уравнение

$$\cos\left(\frac{x}{2}\right) = 0$$

все корни которого имеют вид $\pi/4+(\pi/2)\times N$ ($N \in R$, где R — множество целых чисел). Попробуем найти первое положительное решение. Очевидно, что таковым будет $x=\pi/4$ (что очень хорошо видно на рис. 8.3).

Посмотрим, однако, найдет ли этот корень функция root. Интервал локализации определим от 0 до $\pi/3$:

$$\text{root}\left(f(x), x, 0, \frac{\pi}{3}\right) = 0.25\pi$$

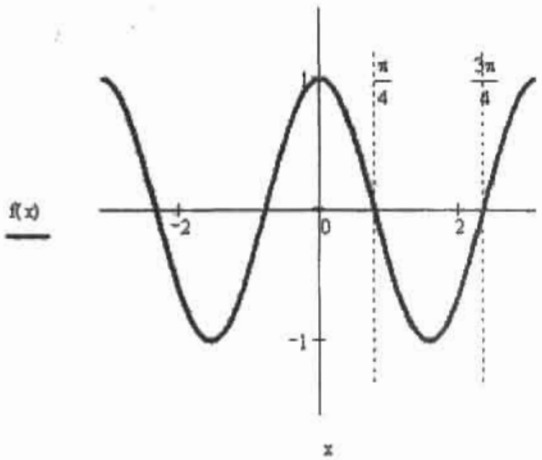


Рис. 8.3. График функции, соответствующей уравнению

Программа не подвела: решение найдено в точности. Попробуем решить это уравнение с другим интервалом локализации:

$$\text{root}\left(f(x), x, 0, \frac{5\pi}{3}\right) = 1.25\pi$$

В приведенном выше примере при использовании второго интервала приближения полученный результат отличается от решения для первого интервала, хотя корень $x=0.25\pi$ принадлежит обоим интервалам. Все дело в том, что в этом же промежутке находится еще один корень, и заранее предсказать, какой именно из них будет выдан в качестве ответа, затруднительно. Поэтому следует ввести еще одно ограничение для применения функции root в рассматриваемой форме: на промежутке должен быть локализован только один корень. В тех случаях, когда определить границы такой локализации невозможно, следует применять функцию root с одной точкой приближения (то есть перейти от методов интервалов локализации корня к методам, основанным на методе секущих). Хотя практически всегда определить нужный промежуток можно и чисто визуально, предварительно построив график. Вообще, строить график всегда желательно, когда вы используете численные методы: это самый надежный способ избежать ошибок и не потерять корни.

Очень важной характеристикой решения является его точность. В Mathcad можно регулировать величину погрешности решения, изменяя значение специальной системной переменной TOL (от англ. tolerance — точность). В общем случае, чем меньше TOL, тем точнее будет найден корень, но и тем больше времени уйдет на его определение (а также будет выше риск, что численный метод не сойдется к решению).

Строго говоря, TOL — это параметр, количественно определяющий условия прекращения итераций. Судить о том, что численный метод сошелся к решению, можно по двум критериям. Во-первых, можно реализовать численный метод так, что цикл алгоритма

остановит свою работу и выдаст последнее значение приближения к корню p_0, если $f(p_n)$ примет значение, меньшее по модулю, чем TOL. Во-вторых, можно считать, что достаточная точность уже достигнута, если два последних приближения различаются на величину, по абсолютному значению меньшую TOL: $|p_n - p_{n-1}| < TOL$. Какой из подходов лучше? По большому счету, они равнозначны. Если используется критерий $|f(p_n)| < TOL$, то решение будет локализовано в горизонтальной области, ограниченной параллельными оси X прямыми $B_{high}(x)=TOL$, $B_{low}(x)=-TOL$ (рис. 8.4). Если функция изменяется быстро, то она пересечет границы области достаточно близко от корня. Это гарантирует то, что точность приближения к корню будет хорошей. Однако если функция изменяется очень медленно, может оказаться так, что точка, выданная как приближение к корню, реально располагается очень далеко от него (см. нижнюю часть кривой на рис. 8.4).

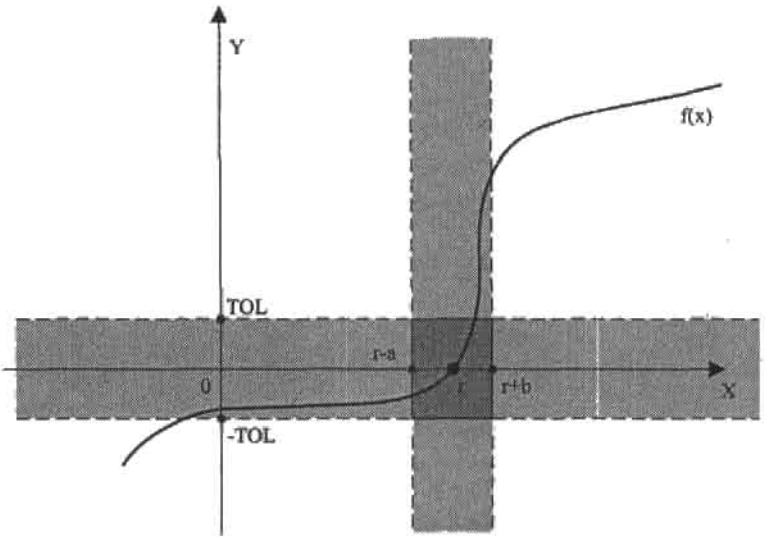


Рис. 8.4. Геометрическая интерпретация TOL

Если в качестве критерия сходимости численного метода используется модуль разности двух последних приближений, то решение будет находиться в вертикальной области, ограниченной параллельными оси Y прямыми с абсциссами r–a и r+b, где r — истинная координата корня. Численно найти a и b непросто, но можно, если знать, какой алгоритм по отношению к какой функции используется. Рассматриваемый критерий хорош, если функция вблизи корня изменяется относительно медленно. Если же она изменяется очень быстро, то может оказаться так, что найденному решению будет соответствовать очень большое значение функции, которое даже с учетом округления не удовлетворит обязательному условию существования корня $f(r)=0$ (см. верхнюю часть кривой на рис. 8.4). Важно отметить то, что a и b вполне могут оказаться больше TOL, так как разность двух последних приближений и расстояние от них до корня прямо никак не связаны. Если алгоритм сходится медленно (по причине своей неэффективности или «неудобности» функции), то расстояние между точками двух последних приближений может быть весьма малым даже тогда, когда они находятся еще очень далеко от корня.

Наиболее достоверным ответ будет, если работу численного алгоритма контролировать с учетом обоих критериев. При этом последнее приближение будет локализовано в прямоугольной области, образующейся на пересечении соответствующих критериям бесконечных вертикальной и горизонтальной областей (см. рис. 8.4).

В случае функции root критерием сходимости к корню является только величина функции в точке приближения. То, насколько отличаются два последних приближения, не учитывается. Такой подход увеличивает скорость работы численных алгоритмов, однако снижает надежность получаемого результата. Мы же, реализуя применяемые root численные алгоритмы, будем учитывать оба критерия.

Изменить величину TOL можно либо с помощью окна, открываемого командой Tools ▸ Worksheet Options (Инструменты ▸ Параметры документа), либо выполнив соответствующее присваивание непосредственно слева или сверху функции численного решения. Присвоить TOL можно совершенно любое значение. По умолчанию TOL равняется 10^{-3}. Однако в случае функции root наибольшим корректным (при стандартных настройках отображения результата) значением TOL является 10^{-5}. Это ограничение связано с тем, что стандартное значение $TOL=10^{-3}$ может обеспечить точность определения корня до третьего знака после запятой далеко не всегда (в этом численные методы решения уравнений отличаются от используемых в Mathcad численных методов интегрирования, ошибка которых всегда меньше TOL). Теоретически наиболее высокую точность дает значение TOL, равное 10^{-16}. Однако нижней границей для TOL в случае функции root является 10^{-12}-10^{-13}. Делать TOL меньше нет смысла, так как ошибки округления компьютерной арифметики и погрешности при расчете значений трансцендентных функций все равно не дадут достигнуть предельной возможной точности — вплоть до 16 знака мантиссы. Также, если TOL очень мала, резко повышается вероятность того, что численный алгоритм не сойдется к решению.

Величина TOL должна задаваться исходя из того, насколько в абсолютном выражении приближение может отличаться от истинного значения корня. Например, если решение нужно получить с точностью до 0.00001, TOL должен быть меньшим или равным 10^{-5}. Впрочем, это правило не очень строгое. Обычно точность решения на 2–3 порядка превышает TOL. Но в случае функций, изменяющихся в области корня существенно медленнее, чем y(x)=x, точность решения может оказаться ниже TOL. Это особенно характерно, если используется метод секущих или Мюллера. Поэтому, обращаясь к соответствующим методам, нужно обязательно строить график и анализировать поведение функции в области пересечения с осью X. Если она изменяется медленнее, чем y(x)=x, то для того, чтобы с высокой надежностью получить результат с абсолютной погрешностью, не превышающей 10^{-n}, TOL следует задать на несколько порядков меньшим, чем 10^{-n}. Если же функция изменяется быстро, то очень точное решение можно получить и при невысоком TOL. Также о скорости изменения функции можно судить по абсолютному значению производной в окрестности точки корня. Если оно больше 1, то функция изменяется быстрее, чем y(x)=x, если меньше — функция изменяется медленно. В случае очень медленно изменяющихся функций нужно применять методы интервалов локализации корня, а не метод секущих. Если же использовать методы локализации корня невозможно (к примеру, корню соответствует касание кривой функции оси X, а не ее пересечение), численному методу решения уравнений стоит предпочесть графический. Впрочем, есть очень хороший прием, позволяющий улучшить результаты работы численных методов в случае медленно изменяющихся в области корня функций. Он заключается в замене функции f(x) на g(x), которая получается делением f(x) на ее производную:

$$g(x) = \frac{f(x)}{\frac{df(x)}{dx}}$$

Функция g(x) будет иметь нули там же, где и f(x). Однако там, где f(x) изменяется очень медленно, g(x) будет изменяться быстрее. Наоборот, там, где f(x) изменяется быстро, g(x) будет изменяться медленнее. Переход от f(x) к g(x) чем-то напоминает нормирование: функция подгоняется под некий средний уровень.

В отдельных случаях выражение уравнения можно улучшить тождественными преобразованиями. Например, если левая часть уравнения соответствует чрезвычайно медленно изменяющейся показательной функции, то стоит произвести логарифмирование. Полученная логарифмическая функция будет иметь нули там же, где и исходная показательная, однако изменяться она будет гораздо быстрее (см. пример 8.11).

Описанные приемы улучшения медленно изменяющейся функции особенно действенны при поиске ее нулей методом секущих (или Мюллера). Без их применения найденное решение может отстоять от точки корня на расстояние, значительно превышающее TOL. По сравнению с методом секущих (Мюллера), методы интервалов локализации корня чувствительны к виду функции в меньшей степени. Однако рассматриваемые приемы позволяют увеличить и их эффективность (см. пример 8.11).

Пример 8.11. Поиск нулей медленно изменяющейся функции

Пусть стоит задача найти нули следующей функции при положительных x:

$$f(x) := x^{-x} - 0.000001$$

По графику определяем, что на интересующем промежутке функция имеет только один нуль, расположенный в окрестности точки x=7. Находим его:

$$\text{root}(f(x), x, 5, 10) = 7.0657934747539093$$

Чтобы получить результат более точно, перейдем от функции f(x) к g(x), которая получается делением f(x) на ее производную:

$$g(x) := f(x) \div \frac{d}{dx} f(x) \qquad \text{root}(g(x), x, 5, 10) = 7.0657967283224243$$

Еще лучшую точность можно получить, прологарифмировав входящие в f(x) выражения:

$$lf(x) := -x \cdot \ln(x) + 6 \cdot \ln(10) \qquad \text{root}(lf(x), x, 5, 10) = 7.0657967282996212$$

Чтобы определить, насколько каждый из полученных корней точен, найдем решение уравнения аналитически:

$$x^{-x} - 10^{-6} \;\Big|\; \begin{matrix}\text{solve}, x \\ \text{simplify}\end{matrix} \rightarrow 6 \cdot \frac{\ln(2) + \ln(5)}{W(6 \cdot \ln(2) + 6 \cdot \ln(5))}$$

Найденное решение содержит W — функцию Ламберта. Определить ее точное значение нельзя, однако можно получить ее сколь угодно точную приближенную величину, задействовав оператор float. Найдем значение корня с точностью до 40 знаков мантиссы:

$$6 \cdot \frac{\ln(2) + \ln(5)}{W(6 \cdot \ln(2) + 6 \cdot \ln(5))}\ \text{float}, 40 \rightarrow 7.065796728299620611060102564664602690090$$

Итак, ответ, выданный системой при поиске нуля для немодифицированной f(x), точен до пятого знака после запятой. Переход от f(x) к g(x) увеличил точность на пять знаков. Наиболее же результативным оказалось логарифмирование. Полученный при этом ответ точен до 15-го (!) знака.

Очевидно, что входить в ответ «ненадежные» знаки не должны. Однако как определить, какие знаки лишние, если уровень погрешности зависит не только от TOL, но и от вида функции и прочих факторов? Алгоритм решения этой проблемы следующий: чтобы убедиться, что в рамках выбранной точности решение найдено верно, нужно округлить результат до нужного знака, а затем уменьшить TOL на 1–2 порядка. Если при этом ответ изменится, значит, он был получен с погрешностью. Также нужно попробовать изменить начальные приближения. Если при этом ответ поменяется, то можно быть уверенным, что в нем имеется серьезная неточность, обусловленная, скорее всего, слишком низкой скоростью изменения функции в районе точки корня.

Пример 8.12. Степень влияния TOL на точность численного решения уравнений в случае методов интервалов локализации корня

Задача: найти первый положительный корень уравнения $\cos(2x+1)^3=0$.

Задаем на основании уравнения функцию:

$$f(x):=\cos(2x+1)^3$$

Значение первого положительного корня очевидно. На него мы будем ориентироваться, определяя точность выдаваемых системой ответов.

$$\frac{\pi}{4}-\frac{1}{2}=0.285398163397448$$

TOL=0,1. Ответ точен до 1-го знака после запятой.

$$TOL:=10^{-1}\qquad root\left(f(x),x,0,\frac{\pi}{2}\right)=0.228536607221123$$

TOL=0,0001. Ответ точен до 4-го знака после запятой.

$$TOL:=10^{-4}\qquad root\left(f(x),x,0,\frac{\pi}{2}\right)=0.285323404126252$$

TOL=0,000000000000001. Ответ точен до 15-го знака после запятой.

$$TOL:=10^{-15}\qquad root\left(f(x),x,0,\frac{\pi}{2}\right)=0.285398163397448$$

Чтобы эффективно использовать функцию root, нужно знать основные идеи применяемых ею численных методов. При задании в качестве начального приближения интервала локализации корня, решение ищется с помощью метода Риддера (Ridder). Если данному методу не удается найти корень, используется метод Брента (Brent). И метод Риддера, и метод Брента относятся к одному и тому же типу методов численного решения уравнений — методам интервалов локализации корня. Родоначальником этой группы методов является метод Больцано деления пополам (бисекции). Остальные

методы группы основаны на той же основной идее, однако они используют разного рода хитрые приемы, увеличивающие скорость сходимости алгоритма. Поэтому есть смысл рассмотреть сначала метод Больцано в его классическом варианте, а затем обсудить открытые Риддером и Брентом улучшения.

Алгоритм метода Больцано следующий.

1. Задается функция, нули которой необходимо найти. Определяется интервал от a_0 до b_0, на котором локализован корень. На то, что корень находится именно на этом интервале, будет указывать то, что функция на его границах принимает значения разных знаков. Этот признак будет работать, если функция непрерывна на промежутке $[a_0, b_0]$, а также если корень не является корнем типа касания.
2. Определяется координата середины интервала $[a_0, b_0]$ c_0.
3. Проверяется, в какой из границ интервала функция принимает противоположный знак тому знаку, который она имеет в средней точке. Тут возможны три варианта.
   - Если $f(a_0)$ и $f(c_0)$ имеют разные знаки, то корень находится на промежутке $[a_0, c_0]$. Значит, можно сузить интервал локализации, взяв в качестве его правой границы c_0 вместо b_0.
   - Если $f(b_0)$ и $f(c_0)$ имеют разные знаки, то корень находится на интервале $[c_0, b_0]$. Сужаем интервал, взяв в качестве левой границы c_0 вместо a_0.
   - Если $c_0=0$, то c_0 является искомой точкой корня. Возвращаем ее как результат.
4. Повторяем описанные выше действия до тех пор, пока длина интервала $[a_n, b_n]$ не уменьшится до 2·TOL. При этом, если взять в качестве приближения к корню среднюю точку интервала, погрешность не превысит TOL. Узнать же, сколько итераций должен совершить алгоритм, чтобы интервал локализации корня сузился до 2·TOL очень просто. На каждой итерации ширина интервала уменьшается в два раза. Исходя из этого можно записать следующее неравенство (здесь N — необходимое количество итераций):

$$\frac{b-a}{2^N} \le 2 \cdot TOL$$

Решив данное элементарное неравенство, получим:

$$N \ge \frac{\ln(b-a) - \ln(TOL)}{\ln(2)} + 1$$

Таким образом, точность поиска корней методом Больцано не зависит от особенностей функции, а определяется только количеством итераций.

Крупным недостатком метода Больцано является его медленная сходимость. Так, к примеру, чтобы найти корень с точностью до 0.00001 при ширине интервала 10, необходима 21 итерация. Это довольно много. Существуют методы, в которых проблема медленной сходимости метода Больцано решается за счет того, что учитываются особенности поведения функции. Это метод ложного положения, а также используемые в Mathcad методы Риддера и Брента. Увы, но в случае этих методов уже нельзя однозначно оценить, сколько необходимо итераций для достижения требуемой точности приближения корня. Поэтому в качестве критериев сходимости используются величина функции в точке последнего приближения и (или) разность двух последних приближений.

Чтобы вам было проще разобраться в сути метода Больцано, приведем его реализацию на языке программирования Mathcad:

$$\text{Bolcano}(f,a,b,TOL) := \left|\begin{array}{l} \text{error("Bad interval!")} \ \text{if} \ \frac{f(a)}{|f(a)|} = \frac{f(b)}{|f(b)|} \vee f(a) = 0 \vee f(b) = 0 \\ N \leftarrow \text{floor}\left(\frac{\ln(b-a) - \ln(TOL)}{\ln(2)} + 1\right) \\ \text{for} \ i \in 1..N+1 \\ \quad \left|\begin{array}{l} c \leftarrow (a+b) \div 2 \\ \text{return} \ c \ \text{if} \ i = N+1 \\ \text{return} \ c \ \text{if} \ f(c) = 0 \\ b \leftarrow c \ \text{if} \ \frac{f(a)}{|f(a)|} \neq \frac{f(c)}{|f(c)|} \\ a \leftarrow c \ \text{otherwise} \end{array}\right. \end{array}\right.$$

Проверка показывает, что функция Bolcano работает ничуть не хуже функции root:

$$f(x) := \sin(x)$$

$$\text{Bolcano}\left(f,1,4,10^{-5}\right) = 1.00000010371853\pi \quad \text{Bolcano}\left(f,1,4,10^{-10}\right) = 1.00000000001226\pi$$

В приведенной реализации метода Больцано есть несколько технических моментов, на которые нужно обратить внимание.

- Ключевым шагом метода Больцано является определение того, принимает функция в двух точках значения одного или разных знаков. Чтобы это выяснить, разделим значение функции в точке на модуль этого значения. В результате мы получим 1 или –1, то есть, по сути, выделим знак. Проделав аналогичную операцию для второй точки и сравнив полученные значения, мы узнаем, имеется ли на промежутке корень. Выделить знак числа также можно, используя встроенную функцию Mathcad sign.
- Определяя количество необходимых для достижения требуемой точности итераций по выведенной выше формуле, результат необходимо округлить до ближайшего меньшего целого числа (подумайте, почему). Выполняет эту операцию функция floor. Соответственно, округление до ближайшего большего целого числа производит функция ceil.
- Перед тем как запустить работу алгоритма Больцано, нужно проверить, принимает ли функция на концах интервала значения разных знаков. Если это не так, останавливаем работу кода и возвращаем сообщение об ошибке посредством функции error.

Метод Больцано имеет важное историческое и теоретическое значение, однако на практике для решения уравнений он не используется. Причина – его медленная сходимость. То есть, чтобы получить посредством него решение нужной точности, следует проделать значительный объем вычислительной работы. Существует несколько методов, которые, используя ту же основную идею, что и метод Больцано, обладают лучшей сходимостью. Наиболее простой из них – метод ложного положения (Regula Falsi).

Отличие метода ложного положения от метода Больцано заключается в том, как вычисляется очередное приближение. В методе Больцано для этого находится среднее арифметическое границ интервала локализации корня: $c_n=(a_n+b_n)/2$. В методе ложного положения через точки $(a_n, f(a_n))$ и $(b_n, f(b_n))$ проводится секущая (рис. 8.5). Точкой

c_n, в которой секущая пересекает ось X, заменяют одну из границ интервала локализации корня, сужая его. Приближение считается достаточно точным, если $|f(c_n)| < TOL$ и (или) $|c_n - c_{n-1}| < TOL$ (простой формулы, позволяющей определить, сколько необходимо итераций для достижения требуемой точности, в случае метода ложного положения, в отличие от метода Больцано, нет).

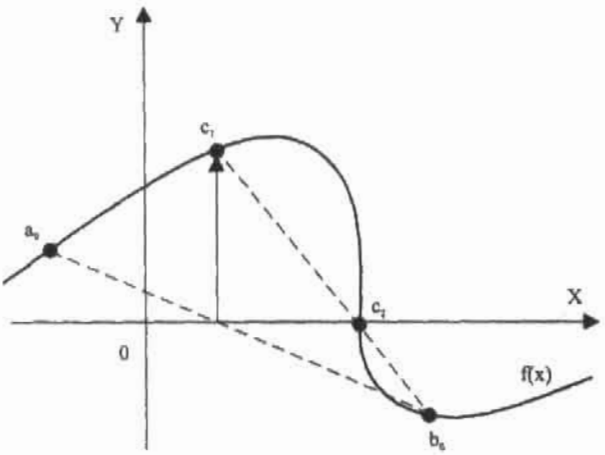


Рис. 8.5. Иллюстрация метода ложного положения. Неплохая точность достигается уже на второй итерации

Метод ложного положения сходится быстрее метода Больцано, но есть и более эффективные методы. По умолчанию функция root применяет метод Риддера. Основное улучшение в нем, по сравнению с методом Regula Falsi, довольно простое, и основано оно на распространенном в теории численных методов приеме — использовании интерполирующего функцию полинома (точно так же осуществляется, например, переход от метода секущих к методу Мюллера, от метода трапеций — к методу Симпсона, от метода Эйлера — к методу Рунге–Кутта). Если излагать суть метода Риддера в деталях, то на каждой итерации проделываются следующие шаги. Сначала вычисляется значение функции в средней точке интервала $c_n=(a_n+b_n)/2$. Затем через имеющиеся три точки (граничные точки интервала и средняя точка) проводится интерполирующая функцию парабола. Так как через K точек проходит только один полином степени K–1 (попробуйте это доказать), то интерполирующая парабола будет уникальной. Несложно вывести, что данная парабола будет определяться следующей функцией (подобный вывод приводится в гл. 10 при рассмотрении метода Симпсона):

$$P_n(x) = f(b_n)\cdot x^2 - 2\cdot f(c_n)\cdot x + f(a_n)$$

Если интервал (a_n, b_n) достаточно узок, то поведение интерполирующей параболы будет весьма схоже с поведением функции. Поэтому логично в качестве приближения к корню использовать точку, в которой парабола пересекает ось X. Найти эту точку несложно, положив $P_n(x)=0$ и решив полученное квадратное уравнение. В результате будет получено два корня, однако только один из них будет принадлежать интервалу (a_n, b_n). Риддер вывел формулу, которая дает возможность сразу вычислить нужный корень (при ее выводе считалось, что уравнение задано не относительно переменной x, а относительно экспоненциальной величины e^Q). Данная формула имеет вид:

$$r_n = c_n + (c_n - a_n)\cdot\frac{f(c_n)\cdot \mathrm{sign}(f(c_n) - f(b_n))}{\sqrt{f(c_n)^2 - f(a_n)\cdot f(b_n)}}$$

После нахождения приближения, проверяется, соответствует ли оно используемым критериям сходимости к корню (в Mathcad это $|f(r_n)| < TOL$). Если нет, то осуществляются стандартные для всех методов интервалов локализации корня шаги. А именно: приближением заменяется та граница интервала (a_n, b_n), которая имеет с ним один и тот же знак. После этого алгоритм переходит на новую итерацию.

Выведенное Риддером уравнение имеет несколько замечательных свойств. Во-первых, приближение всегда оказывается внутри интервала (a_n, b_n). Во-вторых, вычисленная на основании него последовательность приближений сходится довольно быстро. Каждое последующее приближение будет в среднем на две значимые цифры более точным по сравнению с предыдущим. В-третьих, метод Риддера очень устойчив.

Реализовать метод Риддера на языке программирования Mathcad несложно. Напишем соответствующий алгоритм так, чтобы в качестве результата он возвращал не только приближение к корню, но и число, показывающее, за сколько итераций было найдено решение. Это необходимо для объективной оценки эффективности программы.

$$\begin{aligned}
&\mathrm{Ridder}(f,a,b,TOL) := \left|\begin{array}{l}
\mathrm{error}(\text{"Bad Interval!"}) \ \text{ if } \ \mathrm{sign}(a) = \mathrm{sign}(b) \vee f(a) = 0 \vee f(b) = 0 \\
\left(r_{prev} \leftarrow a \quad n \leftarrow 0\right) \\
\text{while } 1 \\
\quad \left|\begin{array}{l}
c \leftarrow (a + b) \div 2 \\
r \leftarrow c + (c - a)\cdot\dfrac{f(c)\cdot \mathrm{sign}(f(c) - f(b))}{\sqrt{f(c)^2 - f(a)\cdot f(b)}} \\
\text{return } (r \ \ n) \ \text{ if } \left(|f(r)| < TOL \wedge |r - r_{prev}| < TOL\right) \vee f(r) = 0 \\
b \leftarrow r \ \text{ if } \ \mathrm{sign}(a) \neq \mathrm{sign}(r) \\
a \leftarrow r \ \text{ otherwise} \\
\left(r_{prev} \leftarrow r \quad n \leftarrow n + 1\right)
\end{array}\right.
\end{array}\right.
\end{aligned}$$

Решим с помощью написанной программы уравнение с очевидным корнем при разном значении TOL:

$$f(x) := x^3 - 1$$

$\mathrm{Ridder}\left(f, 0, 4, 10^{-3}\right) = (1.00000468261462\ 3)$ $\mathrm{Ridder}\left(f, 0, 4, 10^{-5}\right) = (1.00000009556532\ 4)$

$\mathrm{Ridder}\left(f, 0, 4, 10^{-8}\right) = (1.0000000000398\ 6)$ $\mathrm{Ridder}\left(f, 0, 4, 10^{-10}\right) = (1.000000000000817)$

Проанализируем полученные результаты. В первую очередь, стоит отметить высокую эффективность алгоритма Риддера: на то, чтобы найти решение с точностью до 14 знаков мантиссы, понадобилось всего семь итераций. Стандартная же точность в четыре знака мантиссы потребует всего двух оборотов алгоритма. Также обратите внимание

на то, что каждая итерация действительно увеличивает точность на две значимые цифры. Так, например, если алгоритм проделывает шесть оборотов, ошибка проявляется в 12-м знаке мантиссы. Однако нельзя достигнуть предельной точности в 16 знаков мантиссы, выполнив 8 или больше итераций. Это связано с погрешностью при проведении арифметических операций, вычислении значений функций, приближенности при представлении бесконечных дробей, иррациональных чисел и математических констант.

Код функции Ridder несложен. В нем есть только два неочевидных момента.

- Оборвать работу цикла нужно тогда, когда приближение будет соответствовать критериям сходимости к корню. По идее, эта проверка должна осуществляться в заголовке цикла. Однако тут есть одна проблема. Дело в том, что приближение вычисляется кодом, относящимся к блоку цикла. Это означает, что на момент первой итерации никакого приближения еще нет. Соответственно, проверка условия сходимости к корню вызовет ошибку. Преодолеть эту проблему можно несколькими способами. Во-первых, можно вычислить первое приближение до вступления в работу цикла. Однако при этом алгоритм станет менее изящным и компактным. Во-вторых, можно запустить бесконечный цикл, оборвав его работу посредством оператора return в тот момент, когда приближение достигнет необходимой точности. Именно такой прием используется в программе Ridder. Чтобы сделать цикл бесконечным, в его правый маркер нужно ввести условие, которое всегда вычисляется в истину. Или же, проще, можно ввести в правый маркер 1 (в Mathcad, если условие истинно, операторы сравнения и логические операторы возвращают 1).

  В универсальных C-подобных языках проблема, описанная выше, решается проще благодаря наличию оператора цикла do-while. Отличием этого цикла от цикла while является то, что на первой итерации сначала выполняется код в блоке цикла, а уж затем проверяется условие в заголовке.

- Приближение считается достигшим нужной точности, если $|f(r)| < TOL$ и $|r - r_{prev}| < TOL$. Однако возможна такая ситуация, что приближение r совпадет с корнем. При этом проверка комплекса условий сходимости может дать ложь, так как не выполнится условие для близости двух последних приближений. Дальнейшая же работа алгоритма станет невозможной, так как функция не будет принимать значения разных знаков на границах интервала. Чтобы этого избежать, необходимо дополнительно осуществлять проверку на точное попадание приближения в точку корня:

$$\text{return } (r \quad n) \text{ if } \left(|f(r)| < TOL \wedge |r - r_{prev}| < TOL\right) \vee f(r) = 0$$

  Обратите внимание на то, что комплекс критериев сходимости к корню должен быть взят в скобки, так как он должен быть рассмотрен как единое целое.

Методы ложного положения и Риддера сходятся быстрее метода Больцано. Однако они зависят от вида функции и поэтому менее универсальны. Существуют функции, поиск нулей которых займет у методов Regula Falsi и Риддера существенно больше времени, чем уйдет на решение этой задачи у метода Больцано. Такими функциями часто являются функции со сложным поведением: имеющие множественные экстремумы, разрывы или очень быстро изменяющиеся в окрестности корня. А возможно ли сочетать в одном методе высокую сходимость метода Риддера и универсальность метода Больцано? В принципе, да. Стратегия следующая. Используем прием, обеспечивающий высокую сходимость до тех пор, пока он оправдывает себя. При возникновении сложностей переходим к методу бисекции, сходящемуся медленно, но устойчиво.

Методом, наиболее эффективно использующим описанную стратегию, является метод Брента (Brent). Данный метод применяется функцией root в том случае, если за опре-

деленное количество итераций приближение к корню не будет получено методом Риддера.

На начальном этапе поиска корня метод Брента действует точно так же, как метод Риддера. То есть через точки $(a_n, f(a_n))$, $(b_n, f(b_n))$, $(c_n, f(c_n))$ (где a_n и b_n — границы интервала локализации корня, c_n — средняя точка интервала) проводится интерполирующая функцию парабола. Для этого записывается соответствующий полином Лагранжа. Формула Лагранжа интенсивно применяется при разработке численных методов, так как она позволяет автоматически записать полином степени N, проходящий через N+1 известную точку. В случае метода Брента полином Лагранжа будет иметь следующий вид:

$$P_n(x) = \frac{f(a)\cdot(x-b)\cdot(x-c)}{(a-b)\cdot(a-c)} + \frac{f(b)\cdot(x-a)\cdot(x-c)}{(b-a)\cdot(b-c)} + \frac{f(c)\cdot(x-a)\cdot(x-b)}{(c-a)\cdot(c-b)}$$

Приближением к корню считается точка, в которой полином пересекает ось X. Найти ее можно, положив $P_n(x)=0$ и решив полученное уравнение. Результатом будет довольно объемное выражение, сократить которое можно, введя следующие замены:

$$R = \frac{f(b)}{f(c)} \qquad S = \frac{f(b)}{f(a)} \qquad T = \frac{f(a)}{f(c)}$$

$$P = S\cdot[\,T\cdot(R-T)\cdot(c-b) - (1-R)\cdot(b-a)\,] \qquad Q = (T-1)\cdot(R-1)\cdot(S-1)$$

С учетом замен формула для вычисления приближений примет вид:

$$x = b + \frac{P}{Q}$$

После вычисления приближения проверяется, достаточно ли оно близко к корню. Если нет, то интервал локализации корня сужается посредством замены приближением той границы, которая имеет с ним один и тот же знак.

Квадратичная интерполяция хорошо работает, если функция гладкая. Если же на промежутке имеются экстремумы, то приближение, вычисленное по приведенной выше формуле, может оказаться за пределами интервала локализации корня. В этом случае в методе Брента «плохое» приближение отбрасывается, а новое приближение вычисляется так же, как в методе бисекции. Кроме того, метод бисекции будет использован, если окажется, что вычисленное с помощью квадратичной интерполяции приближение не сужает интервал локализации корня в достаточной степени.

Сделаем выводы. Метод Брента сочетает в себе универсальность и стабильность метода Больцано и высокую сходимость метода Риддера. Это делает его одним из наиболее эффективных методов среди методов численного решения уравнения. Именно поэтому использующая его функция root так редко дает сбои.

Главная сложность в использовании методов интервалов локализации корня состоит в том, что нужно определить, на каком промежутке находится корень. Причем функция на границах промежутка должна принимать значения с разными знаками (поэтому корни типа касания такими методами найдены быть не могут), она должна быть непрерывной, на промежутке должен быть только один корень. Это серьезные ограничения, но если удовлетворяющий им интервал будет найден, то за определенное количество итераций (может быть, очень большое) численный метод со 100 %-ной вероятностью сойдется к корню. Поэтому методы интервалов локализации корня называют глобально сходящимися. В отличие от них методы вроде метода секущих, Мюллера

или Ньютона (о них мы поговорим чуть ниже) являются локально сходящимися. Это означает, что то, сойдется метод к решению или нет, очень сильно зависит от выбора начального приближения. В этом они уступают глобально сходящимся методам. Скорость же сходимости лучших методов интервалов локализации корня вроде метода Риддера или метода Брента близка к скорости сходимости метода секущих и лишь немного уступает скорости сходимости метода Мюллера. Поэтому, если корень не имеет особенностей, для его определения лучше использовать метод Риддера или метод Брента.

Однако есть несколько ситуаций, в которых от использования методов интервалов локализации корня лучше отказаться. Во-первых, корню может соответствовать не пересечение функцией оси X, а лишь ее касание. Во-вторых, таким образом нельзя определять комплексные корни. В-третьих, методы вроде метода Больцано неудобно использовать, если процесс решения уравнения является частью более сложной вычислительной схемы, реализованной в виде программы. Во всех этих случаях лучше применять методы, отличительной особенностью которых является то, что они основываются на приближении к корню, а не на интервале его локализации. Этим методам соответствует другая форма функции root, имеющая только два аргумента (имя функции и переменной). Приближение к корню присваивается переменной выше функции root (см. пример 8.13).

Пример 8.13. Численное решение уравнения при наличии точки приближения к корню

Задаем функцию, описывающую уравнение, и находим точное значение первого положительного корня.

$$f(x) := \cos(2x + 1)$$

$$\cos(2x+1) = 0 \text{ solve}, x \rightarrow \frac{-1}{2} + \frac{1}{4}\cdot\pi \qquad \frac{-1}{2} + \frac{1}{4}\cdot\pi \text{ float}, 20 \rightarrow .285398163397448309$$

Задаем начальное приближение и ищем корень при различных значениях TOL.

$$x := 0.2$$

$$TOL := 10^{-5} \qquad \text{root}(f(x), x) = 0.285394105343086$$

$$TOL := 10^{-10} \qquad \text{root}(f(x), x) = 0.285398163399349$$

Функция root с двумя аргументами по умолчанию ищет корни посредством метода секущих. Если этот метод не сходится за определенное количество итераций к решению, применяется метод Мюллера. О сути этих методов мы поговорим чуть позже.

Крупным преимуществом метода секущих и метода Мюллера является то, что с их помощью можно находить комплексные корни. Обязательным условием при этом является то, что приближение также должно быть комплексным (см. пример 8.14).

Пример 8.14. Численное определение комплексных корней

$$f(x) := x^2 + x + 1$$

$$f(x)\begin{vmatrix}\text{solve}, x \\ \text{float}, 4\end{vmatrix} \rightarrow \begin{pmatrix} -.5000 + .8660i \\ -.5000 - .8660i \end{pmatrix} \qquad r1 := -4 + 7\cdot i \qquad r2 := -4 - 7\cdot i$$

$$\text{root}(f(r1), r1) = -0.5 + 0.866i \qquad \text{root}(f(r2), r2) = -0.5 - 0.866i$$

Кстати, довольно часто бывает так, что, основываясь на действительном приближении, функция root находит комплексный корень. Это связано с тем, что в Mathcad все числа рассматриваются как комплексные.

При использовании метода секущих и метода Мюллера появляются дополнительные сложности, которых нет при применении методов интервалов локализации корня. Так, при наличии у уравнения нескольких корней бывает проблематично так подобрать приближение, чтобы был найден именно нужный корень. Причина — при этом не работает, казалось бы, очевидное правило, что определяться должен тот корень, который ближе к точке приближения. Это правило справедливо лишь, если точка приближения находится в окрестности корня. Иначе то, какой корень будет найден, будет зависеть в первую очередь от особенностей поведения функции. Решение этой проблемы простое: приближение нужно не «брать с потолка», а определять по графику.

Во-вторых, корень уравнения методом секущих (Мюллера) может быть не найден, если начальное приближение было задано в районе экстремума или разрыва, а также в области, в которой кривая функции имеет очень маленький наклон.

Пример 8.15. Поиск корня уравнения при разных приближениях

Первый положительный корень уравнения, описываемого следующей функцией, равен π/3:

$$f(x) := \cos(3x)^5 + 1 \qquad f\left(\frac{\pi}{3}\right) = 0 \qquad \frac{\pi}{3} = 1.0471975511966$$

То, как ведет себя функция в области корня, показано на рис. 8.6.

Если в качестве приближения к корню взять точку x=0 (экстремум), то численный метод к решению не сойдется. Приближение, очень близкое к экстремуму, также не приведет к положительному результату. Ближайшая к экстремуму точка, при использовании которой в качестве приближения функция root найдет корень, это x=0.001. Однако определенный при этом корень будет расположен очень далеко от искомого корня x=π/3:

$$x := 0.001 \qquad root(f(x), x) = 114.144104502797$$

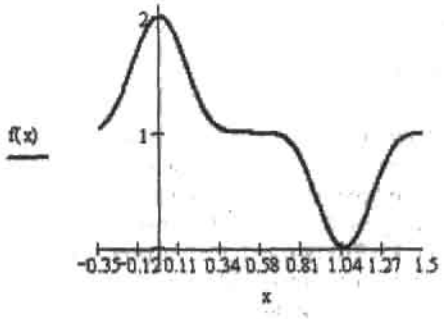


Рис. 8.6. Поведение функции вблизи первого положительного корня

Большинство точек в области от 0.001 до 0.1 при использовании их в качестве приближения к корню или не дадут сходящейся к решению последовательности, или эта последовательность будет сходиться к удаленному корню. Лишь отдельные точки дадут верное решение.

Большая часть точек в области от 0.1 до 0.3 дадут верное решение. Это связано с тем, что кривая функции в этом интервале имеет достаточный наклон по направлению к корню. Однако часть

точек будут давать или не сходящуюся последовательность, или последовательность, сходящуюся к удаленному корню.

$$x := 0.1 \qquad root(f(x), x) = 1.04761194953047$$

$$x := 0.22 \qquad root(f(x), x) = -8.00373139717741 \times 10^{3}$$

В области от 0.3 до 0.8 почти все точки дают последовательность, не сходящуюся к корню. Отдельные точки приводят к решению, однако выдаваемые при этом корни очень далеки от $\pi/3$. Данные сложности обусловлены тем, что кривая в интервале (0.3, 0.8) почти горизонтальна.

$$x := 0.57 \qquad root(f(x), x) = -3.00231601851947 \times 10^{3}$$

Все точки интервала (0.8, $\pi/3$), взятые в качестве приближения к корню, дают сходящуюся к необходимому корню последовательность. Это связано с тем, что данная область непосредственно прилегает к корню, а также с тем, что соответствующий ей участок кривой имеет хороший наклон и лишен экстремумов и разрывов.

$$x := 0.9 \qquad root(f(x), x) = 1.0467392869927$$

Область кривой справа от корня симметрична области слева. Поэтому с точки зрения выбора приближений они идентичны.

Пример 8.15 показывает, сколь сильно то, даст ли приближение сходящуюся к нужному корню последовательность или нет, зависит от особенностей поведения функции. То, почему так происходит, вы поймете, когда мы разберем принципы метода секущих и метода Мюллера.

В отличие от методов интервалов локализации корня, методы секущих и Мюллера склонны находить корни там, где их нет. Такие корни называются ложными.

Пример 8.16. Неверное решение уравнения методом секущих

Попытаемся решить уравнение, заведомо не имеющее корней — $e^{-x}=0$. Система находит корень. Причем, его величина резко увеличивается с уменьшением TOL.

$$f(x) := e^{-x} \qquad x := 1$$

$$TOL := 10^{-5} \qquad root(f(x), x) = 11.625$$

$$TOL := 10^{-12} \qquad root(f(x), x) = 28.26$$

Приведенный пример показывает, насколько «глупой» может быть функция root. Попробуем разобраться, в чем же причина возникших трудностей и как можно избежать ошибок. На самом деле, все просто: как вы помните, критерий сходимости к корню у функции root — это $|f(r_n)| < TOL$. Функция же e^{-x} имеет одно очень вредное свойство: бесконечно близко приближаться к оси X, никогда ее не достигая. Из этого следует, что *если в какой-то точке* r_n *выполняется условие* $|f(r_n)| < TOL$, *то оно будет справедливо для* всех точек, лежащих правее r_n. А точек таких бесконечно много! Избежать же обнаружения ложных корней очень часто можно, введя дополнительно, в качестве критерия сходимости к корню, условие близости двух последних приближений $|r_n - r_{n-1}| < TOL$. Увы, но функция root этого не делает.

Чтобы не найти корень там, где функция просто сближается с осью X, опять же нужно строить график. Вообще, действовать вслепую, численно решая уравнения, ни в коем случае нельзя.

В общем, шанс получить ошибочный ответ при использовании метода секущих (или Мюллера) очень велик. Как же этого избежать? В первую очередь, для этого нужно

понимать, как именно функция root ищет корни. Разберем для начала более простой метод секущих. Он заключается в следующих шагах.

- В качестве первой точки приближения x_1 определяется начальное приближение, вводимое пользователем.
- Точка второго приближения x_2 определяется как сумма первого приближения и величины, равной заданной точности: $x_2 = x_1 + TOL$.
- Через точки $(x_1, f(x_1))$ и $(x_2, f(x_2))$ проводится секущая. Абсцисса точки $(r, f(r))$, в которой она пересекает ось X, определяется как третье приближение x_3. Найти x_3 несложно. Для этого нужно записать уравнение прямой, проходящей через точки $(x_1, f(x_1))$ и $(x_2, f(x_2))$, приравнять его нулю и выразить x. В результате получим следующую формулу:

$$x_3 = x_2 - \frac{f(x_2)\cdot(x_2 - x_1)}{f(x_2) - f(x_1)}$$

- Если значение функции в точке третьего приближения удовлетворяет критериям сходимости к корню, то эта точка определяется как корень. В противном случае соответственно точкам третьего и второго приближения проводится еще одна секущая, которая дает четвертое приближение. Процесс повторяется до тех пор, пока полученная точка приближения не удовлетворит условию корня. В общем случае формула для получения приближения на основании двух предыдущих приближений будет иметь вид:

$$x_n = x_{n-1} - \frac{f(x_{n-1})\cdot(x_{n-1} - x_{n-2})}{f(x_{n-1}) - f(x_{n-2})}$$

Метод секущих можно гораздо проще понять, если отобразить ход поиска корня на графике (рис. 8.7).

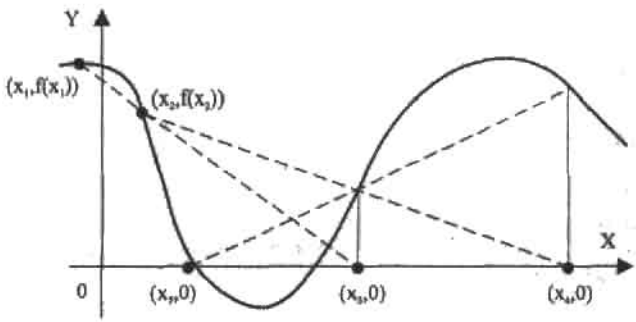


Рис. 8.7. Иллюстрация метода секущих. Приемлемое приближение определяется за три итерации

Реализовать метод секущих самостоятельно, используя язык программирования Mathcad, проще простого. Однако нужно учесть, что, в отличие от глобально сходящихся методов интервалов локализации корня, метод секущих при неудачном выборе начального приближения может породить расходящуюся последовательность. Поэтому, если количество итераций превысит некоторый максимум (несколько десятков), работу программы нужно остановить и вывести сообщение об ошибке.

Программу мы напишем так, чтобы в качестве результата она возвращала вектор из двух элементов. Первым элементом будет найденное решение, второй элемент будет содержать вектор со всеми вычисленными в ходе поиска корня приближениями. Анализируя, насколько быстро последовательность приближений сходится к корню, мы сможем оценить эффективность метода секущих.

$$\text{Secant}(f,e,TOL) := \left| \begin{array}{l} \left(p_0 \leftarrow e \quad p_1 \leftarrow e + TOL\right) \\ \text{while } \neg\left(\left|p_{last(p)} - p_{last(p)-1}\right| < TOL \wedge \left|f\left(p_{last(p)}\right)\right| < TOL\right) \\ \quad \left| \begin{array}{l} \left(a \leftarrow p_{last(p)-1} \quad b \leftarrow p_{last(p)}\right) \\ p_{last(p)+1} \leftarrow b - \dfrac{f(b)\cdot(b-a)}{f(b)-f(a)} \\ \text{error("Dont converge to a solution")} \quad \text{if } last(p) > 50 \end{array} \right. \\ \left(p_{last(p)} \quad p\right) \end{array} \right.$$

Программа Secant несложна, но в ней есть моменты, которые стоит пояснить.

- Приближения записываются в вектор p. Чтобы получить новое приближение, нужно подставить в формулу метода секущих значения последних двух приближений. Чтобы адресовать их, используем функцию last, возвращающую индекс последнего элемента в векторе. Также можно применить функцию length, определяющую количество элементов в векторе.
- Итерации должны совершаться до тех пор, пока не выполнятся условия сходимости к корню. Добиться этого можно, внеся соответствующий комплекс условий в заголовок цикла while и выполнив над ним операцию логического отрицания посредством оператора ¬.

Решим с помощью функции Secant уравнение с известными корнями при разном значении TOL:

$$f(x) := x^2 - 2x - 3 \qquad f(x)\ \text{solve},x \rightarrow \begin{pmatrix} -1 \\ 3 \end{pmatrix}$$

$$\text{Secant}\left(f, 0.7, 10^{-5}\right) = (-0.999999999991711 \quad \{11,1\}) \qquad \text{Secant}\left(f, 0.7, 10^{-8}\right) = (-1 \quad \{12,1\})$$

Итак, метод секущих сходится весьма быстро. Если приближение выбрано достаточно близко к корню, то каждая итерация уточняет результат приблизительно на полторы значимые цифры. Если же первое приближение удалено от корня, то несколько итераций уйдет на то, чтобы метод «стал на верный курс». Анализируя векторы приближений, возвращаемые функцией Secant, можно обнаружить, что скорость сходимости возрастает по мере продвижения к корню:

$$\left(\text{Secant}\left(f,4,10^{-4}\right)_{0,1} - 3\right)^T = \left(1 \quad 1 \quad 0.17 \quad 0.03 \quad 1.28\times 10^{-3} \quad 1.02\times 10^{-5} \quad 3.28\times 10^{-9}\right)$$

$$\left(\text{Secant}\left(f,-2,10^{-4}\right)_{0,1} + 1\right)^T = \left(-1 \quad -1 \quad -0.167 \quad -0.032 \quad -1.28\times 10^{-3} \quad -1.024\times 10^{-5} \quad -3.275\times 10^{-9}\right)$$

Кстати, написанная нами программа превосходит root, с точки зрения надежности функции, что связано с тем, что мы используем два критерия сходимости к корню, а root —

только один. Так, например, если мы попытаемся решить уравнение $e^x=0$ с использованием функции Secant, то будет выдано сообщение об ошибке: Dont convert to a solution. Функция же root в такой ситуации выдаст ложный корень (см. пример 8.16).

Зная то, как работает метод секущих, легко понять, почему так важно правильно выбрать начальное приближение.

- Начальное приближение должно быть достаточно близко к корню для того, чтобы первая секущая была направлена в его сторону (первая секущая, ввиду малости TOL, ведет себя практически так же, как касательная к точке начального приближения). Если между приближением и корнем имеется экстремум или разрыв, то направление функции в окрестности приближения может быть таким, что секущая «уйдет в сторону».
- Если приближение находится близко к экстремуму, метод секущих может не сойтись или же будет получен корень, весьма удаленный от искомого. Это связано с тем, что вблизи экстремума кривая имеет очень малый наклон. Из-за этого малый наклон будет иметь и первая секущая. Соответственно точка, в которой она пересечет ось X, будет очень удалена от первого приближения.
- Аналогично случаю с экстремумом, решение можно не получить также, если в окрестности первого приближения функция изменяется очень медленно (см. пример 8.15).
- Возможны и более сложные случаи, в которых метод секущих не сойдется. К примеру, корень может соседствовать с бесконечным разрывом.

В общем, подобрать подходящее начальное приближение для метода секущих может быть непросто. Поэтому его стоит использовать лишь в оговоренных выше особых случаях. В остальных случаях лучше применять глобально сходящиеся методы интервалов локализации корня.

Если функции root не удается найти решение посредством метода секущих, она пытается это сделать посредством более совершенного, но и сложного метода Мюллера. По сути, данный метод является модифицированным с целью увеличения скорости сходимости методом секущих. В этом методе секущая заменяется параболой. Соответственно, для построения этой параболы используются три последних приближения. Точка, в которой парабола пересекает ось X, считается новым приближением. Метод Мюллера сходится быстрее метода секущих, давая, в среднем, по две значимые цифры на одну итерацию. Однако он более чувствителен к начальному приближению и виду функции.

8.1.3. Определение корней полинома

Одним из основных недостатков функции root является то, что одновременно она способна найти только один корень уравнения. Чтобы найти остальные, придется проделать те же операции, но для других приближений. В случае большого количества корней это может быть очень утомительно. Конечно, можно написать программу, выполняющую эту работу, — однако это слишком сложно для большинства пользователей. Но все значительно упрощается в том случае, если ваше уравнение представлено алгебраическим полиномом. Для этого частного случая разработаны очень эффективные алгоритмы решения, и использовать для поиска корней полинома, например, метод секущих было бы не совсем рационально. Поэтому совсем не удивительно, что в Mathcad существует специальная функция поиска решений полинома polyroots(V), где V — вектор, составленный из коэффициентов полинома. Важной особенностью задания V

является то, что коэффициенты располагаются в векторе сверху вниз соответственно увеличению степеней членов полинома, к которым они относятся. То есть первым сверху нужно поместить коэффициент при x^0 (свободный член), нижним — коэффициент при x^n, где n — порядок полинома. Ответ выдается в виде вектора, содержащего как действительные, так и комплексные корни.

Пример 8.17. Поиск корней полинома посредством численного метода

Пусть стоит задача найти корни уравнения следующего вида:

$$x^4 + x^3 + x + 1 = 1$$

Решение: составляем вектор коэффициентов и используем функцию polyroots.

$$\text{koef} := \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} \qquad \text{polyroots(koef)} = \begin{pmatrix} -1.466 \\ 0 \\ 0.233 - 0.793i \\ 0.233 + 0.793i \end{pmatrix}$$

Переписывать коэффициенты из выражения в вектор не очень интересное и приятное занятие, тем более что при этом очень легко либо допустить ошибку в самом числе, либо просто перепутать знаки. Для решения этой проблемы нужно использовать специальный оператор coeffs, расположенный на панели Symbolic (Символьные). Причем полином может быть представлен в неразвернутой форме: к стандартному виду оператор coeffs приведет его автоматически (см. пример 8.18).

Пример 8.18. Использование оператора coeffs

Пусть нам необходимо решить уравнение следующего вида:

$$\left(x^2 + x + 1\right)^2 + (x - 1)^3 + (x + 1)^4 = 0$$

Решение: используем функцию polyroots, вектор коэффициентов для которой создаем с помощью оператора coeffs.

$$\text{koefs} := \left(x^2 + x + 1\right)^2 + (x - 1)^3 + (x + 1)^4 \text{ coeffs}, x \rightarrow \begin{pmatrix} 1 \\ 9 \\ 6 \\ 7 \\ 2 \end{pmatrix} \qquad \text{polyroots(koefs)} = \begin{pmatrix} -2.981 \\ -0.2 + 1.169i \\ -0.2 - 1.169i \\ -0.119 \end{pmatrix}$$

Найти корни полинома с использованием polyroots можно с помощью двух методов: Лаггера (LaGuerre) и сопровождающей матрицы (Companion matrix). Поменять используемый метод можно в контекстном меню функции polyroots (вызывается щелчком правой кнопкой мыши). По умолчанию выбран метод Лаггера. Нетрудно проверить, что оба метода очень точны — однако точность метода сопровождающей матрицы зачастую все-таки несколько выше. Однако функцией polyroots по умолчанию все же используется метод Лаггера, так как он более стабилен.

Точность работы функции polyroots не зависит от TOL — она всегда максимальна.

Пример 8.19. Проверка точности функции polyroots

Попробуем применить функцию polyroots к полиному, корни которого известны. Это даст возможность оценить величину погрешности.

$$\text{koefs} := (x-1)\cdot(x-4)\cdot(x+45)\cdot(x-2)\cdot(x+1001)\ \left|\begin{matrix}\text{expand},x\\ \text{coeffs},x\end{matrix}\right.\rightarrow \begin{pmatrix}-360360\\ 622262\\ -300679\\ 37737\\ 1039\\ 1\end{pmatrix}$$

LaGuerra

$$\text{polyroots(koefs)}-\begin{pmatrix}-1001\\ -45\\ 1\\ 2\\ 4\end{pmatrix}=\begin{pmatrix}-1.478\times10^{-12}\\ 3.243\times10^{-11}\\ -4.483\times10^{-12}\\ -1.595\times10^{-12}\\ -2.479\times10^{-11}\end{pmatrix}$$

Companion Matrix

$$\text{polyroots(koefs)}-\begin{pmatrix}1001\\ -45\\ 1\\ 2\\ 4\end{pmatrix}=\begin{pmatrix}-2.002\times10^{3}\\ -1.421\times10^{-14}\\ -3.109\times10^{-15}\\ 2.665\times10^{-15}\\ 0\end{pmatrix}$$

Функция polyroots способна находить корни полиномов от 2 до 99 степени. В общем случае, чем выше степень полинома, тем с меньшей точностью будут найдены корни.

Объективно говоря, чаще всего использовать функцию polyroots для поиска корней полинома не совсем технично. На практике лучше всегда применять оператор solve, так как при этом для полиномов невысокой степени ответ может быть получен в более информативном и точном символьном виде. Если же символьный ответ будет слишком громоздок, его всегда можно пересчитать в числа с плавающей точкой посредством оператора float с той точностью, с которой необходимо. Однако важно понимать, что если степень полинома превышает четыре, то для поиска его корней оператор solve также будет использовать приближенный метод (скорее всего, тот же, что применяет функция polyroots). При этом заметное различие между использованием оператора solve и функции polyroots исчезнет (но в случае solve ответ будет на пять порядков точнее, так как символьный процессор использует не связанную с аппаратным уровнем арифметику длинных мантисс).

Численные методы, используемые функцией polyroots, очень своеобразны. Поэтому стоит с ними познакомиться. Начнем мы с метода Лаггера.

Метод Лаггера — это самый стабильный из известных методов определения корней полинома. Наверное, единственный его недостаток, это то, что он требует использования арифметики комплексных чисел, даже если все корни полинома действительны. Впрочем, так как Mathcad поддерживает комплексные числа, даже этот недостаток исчезает. Метод Лаггера требует начальное приближение к корню, однако в подавляющем большинстве случаев не имеет значения, что это будет за приближение (именно поэтому функция polyroots принимает только один параметр). В случае действительных корней метод сходится к решению, какое бы число ни было взято в качестве приближения. Крайне редко проблемы со сходимостью возникают, если у полинома

есть комплексные корни. В этом случае нужно просто сменить неудачное приближение.

Скорость сходимости метода Лаггера к корню весьма высока. Так, она почти в два раза выше скорости сходимости метода секущих и в полтора раза выше скорости сходимости метода Мюллера.

Метод Лаггера основывается на нескольких фактах. Основной из них — любой полином можно представить в виде произведения линейных множителей. Этот факт вытекает из фундаментальной теоремы алгебры, утверждающей, что у полинома степени N будет N корней.

$$P_n(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_1 \cdot x + a_0 = (x - x_1)(x - x_2) \cdot \ldots \cdot (x - x_{n-1})(x - x_n)$$

Логарифмируя разложенный на множители полином, а затем дифференцируя полученную функцию, обнаружим следующие важные связи:

$$G = \frac{d}{dx}\left|\ln(P_n(x))\right| = \frac{1}{x - x_1} + \frac{1}{x - x_2} + \ldots + \frac{1}{x - x_n} = \frac{\frac{d}{dx}P_n(x)}{P_n(x)}$$

Взятие второй производной от логарифмированного полинома дает следующие соотношения:

$$H = -\frac{d^2}{dx^2}\left|\ln(P_n(x))\right| = \frac{1}{(x - x_1)^2} + \frac{1}{(x - x_2)^2} + \ldots + \frac{1}{(x - x_n)^2} = \left(\frac{\frac{d}{dx}P_n(x)}{P_n(x)}\right)^2 - \frac{\frac{d^2}{dx^2}P_n(x)}{P_n(x)}$$

Базируясь на данных формулах, Лаггер сделал предположение, которое на первый взгляд может показаться просто безумным. Тем не менее оно работает. Лаггер предложил считать, что первый корень располагается на расстоянии a от текущего приближения x, в то время как все остальные корни располагаются от него на одинаковом расстоянии b. С учетом этих допущений выражения для G и H существенно упростятся:

$$G = \frac{1}{a} + \frac{n-1}{b} \qquad H = \frac{1}{a^2} + \frac{n-1}{b^2}$$

Данную систему уравнений довольно просто решить относительно a:

$$a = \frac{n}{G \pm \sqrt{(n-1)\cdot(n \cdot H - G^2)}}$$

Полученное уравнение является основным уравнением метода Лаггера. Выбор знака перед дискриминантом должен быть осуществлен таким образом, чтобы величина знаменателя была максимальной.

Поиск корня исходя из уравнения Лаггера — это итеративный процесс. То есть на основании начального приближения x_0 находится поправка a_0. Следующее приближение определяется как $x_1 = x_0 - a_0$. Подстановка x_1 в уравнение Лаггера дает вторую поправку a_1, на основании которой находится третье приближение x_2. Этот процесс продолжается до тех пор, пока не выполнятся условия сходимости к корню. Эти условия те же, что и для других численных методов решения уравнений: $|f(x_n)| < TOL$ и (или) $|x_n - x_{n-1}| < TOL$.

В случае функции polyroots TOL имеет всегда одно и то же значение — порядка 10^{-12}. Мы же напишем реализующую метод Лаггера программу так, что значение TOL можно будет задавать произвольно.

Формула Лаггера дает возможность найти только один корень полинома исходя из данного приближения. Чтобы найти другие корни, необходимо повторять процесс при иных значениях приближений. Однако на практике этот путь, ввиду его неэффективности, не используют. Действительно, так как заведомо неизвестно, какое приближение какой корень даст, можно многие тысячи раз впустую прокрутить алгоритм, получая один и тот же корень. Более технично, найдя корень, разделить полином на соответствующий ему линейный множитель. При этом будет получен полином на единицу меньшей степени, чем исходный полином. К этому полиному нужно повторно применить алгоритм Лаггера (начальное приближение менять не обязательно). Получив второй корень, делим полином на соответствующий ему линейный множитель. И так до тех пор, пока все корни не будут найдены.

Итак, прежде чем реализовывать непосредственно алгоритм Лаггера, нужно создать функцию, посредством которой можно будет делить полиномы на соответствующие найденным корням линейные множители. Чтобы это сделать, необходимо вспомнить, как производится деление полинома на полином «столбиком». В общем, полином делится на полином по тем же правилам, что и число на число. Покажем это на примере деления полинома четвертой степени на его линейный множитель:

$$
\begin{array}{l|l}
x^4-10x^3+35x^2-50x+24 & x-4 \\
\underline{-\,x^4-4x^3} & \overline{x^3-6x^2+11x-6} \\
\quad -6x^3+35x^2-50x+34 & \\
\quad \underline{-\,-6x^3+24x^2} & \\
\quad\quad 11x^2-50x+24 & \\
\quad\quad \underline{-\,11x^2-44x} & \\
\quad\quad\quad -6x+24 & \\
\quad\quad\quad \underline{-\,-6x+24} & \\
\quad\quad\quad\quad 0 &
\end{array}
$$

Как видите, поделить полином на его линейный множитель очень просто. Соответственно, несложно написать и выполняющую эту операцию программу. Представлять полином для этой функции наиболее удобно в той же форме, что и для функции polyroots: вектора коэффициентов. Вторым параметром данной функции будет корень полинома a, на соответствующий которому линейный множитель необходимо разделить полином.

```
pol_dev(V,a):= | n ← last(V)
               | r ← V_n
               | V ← submatrix(V,0,n-1,0,0)
               | for i ∈ n-1..0
               |     | s ← V_i
               |     | V_i ← r
               |     | r ← s + r·a
               | V
```

$V := (-126 \;\; -31 \;\; 77 \;\; -17 \;\; 1)^T$

$\text{pol\_dev}(V,7)^T = (18 \;\; 7 \;\; -10 \;\; 1)$

$$\frac{x^4 - 17\cdot x^3 + 77\cdot x^2 - 31\cdot x - 126}{x-7} \text{ simplify} \rightarrow x^3 - 10\cdot x^2 + 7\cdot x + 18$$

Программа, реализующая метод Лаггера, будет довольно сложна. Поэтому имеет смысл описать ее создание поэтапно.

Назовем мы нашу программу LaGuerra. В качестве параметров она будет принимать вектор коэффициентов полинома V и уровень необходимой точности TOL:

$$\text{LaGuerra}(V, TOL) := \blacksquare$$

В первую очередь на основании вектора коэффициентов V нужно задать функцию полинома P. Затем необходимо объявить использующиеся в формуле Лаггера функции G и H. Так как в выражениях функций G и H функция P присутствует в знаменателях дробей, то крайне важно предусмотреть возможность возникновения ошибки деления на нуль. Чтобы данная ошибка не вызвала сбой в работе программы, нужно сделать так, чтобы функция полинома P никогда не была равна 0. Для этого необходимо отслеживать значение P. Если P окажется равным 0, в качестве значения функции следует возвратить очень малую, однако отличную от 0 величину. Логичным шагом будет увязать данную величину и TOL (к примеру, как $0.1 \cdot TOL$). Технически описанный механизм проще всего реализовать, используя функцию if. Данная функция по своему назначению аналогична конструкции if-otherwise и, соответственно, она принимает три параметра. Первый параметр — это некоторое условие. Второй параметр — величина, которая должна быть возвращена, если условие окажется истинным. Третий параметр — значение, которое следует возвратить, если условие даст ложь.

$$P(V,x) \leftarrow \text{if}\left[\sum_{j=0}^{\text{last}(V)} \left(V_j \cdot x^j\right) \neq 0,\ \sum_{j=0}^{\text{last}(V)} \left(V_j \cdot x^j\right), 0.1 \cdot TOL\right]$$

$$G(V,x) \leftarrow \frac{\frac{d}{dx}P(V,x)}{P(V,x)},\ H(V,x) \leftarrow G(V,x)^2 - \frac{\frac{d^2}{dx^2}P(V,x)}{P(V,x)}$$

Далее нужно создать цикл, на каждом обороте которого будет вычисляться один корень. Соответственно, количество итераций, совершаемых данным циклом, должно быть равно степени полинома.

$$\text{for } i \in 0..\text{last}(V) - 1$$

$$\blacksquare$$

Перейдя к нахождению очередного корня, нужно задать начальное приближение a. Чаще всего не имеет значения, что это будет за приближение. Однако в случае комплексных корней оно может влиять на то, сойдется алгоритм или нет. Чтобы исключить предвзятость, будем задавать начальное приближение случайным образом в интервале от –100 до 100, используя генератор равномерно распределенных случайных чисел rnd. Задав a, необходимо объявить еще две переменные. Переменная a_{last} будет хранить значение предыдущего приближения. Изначально ей можно присвоить любое значение, заметно отличающееся от a. Переменная k — это счетчик итераций, совершенных при поиске корня. Первоначально она должна быть равна 0.

$$a \leftarrow -100 + \text{rnd}(200), a_{last} \leftarrow 2 \cdot a, k \leftarrow 0$$

Теперь следует создать цикл, который будет реализовывать итеративный механизм постепенного приближения к корню по формуле Лаггера. Так как количество необходимых итераций заведомо неизвестно, нужно использовать цикл while. В качестве условий его остановки необходимо задать то, что значение полинома в точке приближения по модулю должно быть меньше TOL, а также разность между данным и предыдущим приближениями по абсолютной величине должна не превышать TOL.

$$\text{while } \neg\left(\left|P(V,a)\right| < TOL \wedge \left|a - a_{last}\right| < TOL\right)$$

Как уже указывалось выше, в большинстве случаев не имеет значения, какое число взять в качестве начального приближения. Но изредка встречаются и исключения (некоторые комплексные корни). Поэтому метод Лаггера может и не сойтись. При этом нужно сменить начальное приближение и осуществить попытку определения корня заново. Судить о том, что последовательность приближений к корню не сходится, можно по величине счетчика итераций k. Так как скорость сходимости метода Лаггера очень высока, то даже при самом малом значении TOL и самом неудачном выборе начального приближения на то, чтобы найти корень, не должно уйти больше, чем несколько десятков итераций. Если же значение k превысит установленный лимит, то нужно сгенерировать новое приближение, обнулить переменные k и a_{last}, после чего перейти на новую итерацию посредством оператора continue. Если же значение k невелико, следует просто увеличить его на 1, зафиксировав очередную итерацию.

$$\begin{cases} \text{if } k > 100 \\ \quad \begin{cases} a \leftarrow -100 + \text{rnd}(200), a_{last} \leftarrow 2 \cdot a, k \leftarrow 0 \\ \quad \text{continue} \end{cases} \\ k \leftarrow k + 1 \quad \text{otherwise} \end{cases}$$

Далее нужно написать код, который будет, используя формулу Лаггера, на основании текущего приближения находить приближение следующее. Величина же старого приближения будет присваиваться переменной a_{last}.

$$\begin{cases} n \leftarrow \text{last}(V), G_a \leftarrow G(V,a), H_a \leftarrow H(V,a) \\ m_0 \leftarrow G_a + \sqrt{(n-1)\cdot\left(n\cdot H_a - G_a^{\,2}\right)}, m_1 \leftarrow G_a - \sqrt{(n-1)\cdot\left(n\cdot H_a - G_a^{\,2}\right)} \\ a_{last} \leftarrow a, a \leftarrow a - \dfrac{n}{\text{if}(\max(m) \neq 0, \max(m), 0.1\cdot TOL)} \end{cases}$$

В приведенном фрагменте кода есть два неочевидных момента. Во-первых, нужно ответить на вопрос, зачем создается массив m. Это необходимо, чтобы минимальным количеством кода определить, при каком знаке перед корнем выражение знаменателя дроби из формулы Лаггера примет максимальное значение. Занося оба варианта в массив m, мы можем определить, какой из них больше посредством функции max. Во-вторых, нужно предусмотреть то, что наибольший элемент массива m может быть и нулем. Чтобы избежать ошибки деления на нуль, действуем точно так же, как в случае локальной функции P.

Когда критерии сходимости к корню выполнятся и цикл while прекратит работу, значение последнего приближения должно быть занесено в вектор результатов res. Затем полином должен быть разделен на соответствующий найденному корню линейный множитель. Для этого следует задействовать написанную ранее функцию pol_dev. После выполнения этих действий алгоритм будет готов перейти к поиску очередного корня.

$$\begin{aligned}&\text{res}_i \leftarrow a\\&V \leftarrow \text{pol\_dev}(V, a)\end{aligned}$$

Когда все корни будут найдены, вектор res должен быть возвращен как результат работы функции. Однако перед этим корни стоит отсортировать в порядке возрастания, воспользовавшись функцией sort.

$$\text{sort}(\text{res})$$

Функция LaGuerra готова. Приведем ее полный код:

$$\begin{aligned}
\text{LaGuerra}(V, TOL) := \;& P(V,x) \leftarrow \text{if}\left[\sum_{j=0}^{\text{last}(V)}\left(V_j \cdot x^j\right) \neq 0,\ \sum_{j=0}^{\text{last}(V)}\left(V_j \cdot x^j\right), 0.1 \cdot TOL\right]\\
& G(V,x) \leftarrow \frac{\frac{d}{dx}P(V,x)}{P(V,x)},\ H(V,x) \leftarrow G(V,x)^2 - \frac{\frac{d^2}{dx^2}P(V,x)}{P(V,x)}\\
& \text{for } i \in 0..\text{last}(V) - 1\\
& \quad a \leftarrow -100 + \text{rnd}(200),\ a_{\text{last}} \leftarrow 2 \cdot a,\ k \leftarrow 0\\
& \quad \text{while } \neg\left(\left|P(V,a)\right| < TOL \wedge \left|a - a_{\text{last}}\right| < TOL\right)\\
& \quad\quad \text{if } k > 100\\
& \quad\quad\quad a \leftarrow -100 + \text{rnd}(200),\ a_{\text{last}} \leftarrow 2 \cdot a,\ k \leftarrow 0\\
& \quad\quad\quad \text{continue}\\
& \quad\quad k \leftarrow k + 1 \quad \text{otherwise}\\
& \quad\quad n \leftarrow \text{last}(V),\ G_a \leftarrow G(V,a),\ H_a \leftarrow H(V,a)\\
& \quad\quad m_0 \leftarrow G_a + \sqrt{(n-1)\cdot\left(n \cdot H_a - G_a^2\right)},\ m_1 \leftarrow G_a - \sqrt{(n-1)\cdot\left(n \cdot H_a - G_a^2\right)}\\
& \quad\quad a_{\text{last}} \leftarrow a,\ a \leftarrow a - \frac{n}{\text{if}(\max(m) \neq 0, \max(m), 0.1 \cdot TOL)}\\
& \quad \text{res}_i \leftarrow a\\
& \quad V \leftarrow \text{pol_dev}(V,a)\\
& \text{sort}(\text{res})
\end{aligned}$$

Проверим эффективность написанной программы, применив ее по отношению к полиному, корни которого заведомо известны. Сравним выданный ею результат с результатом работы функции polyroots.

$$Z := (x+1)\cdot(x-9)^2\cdot(x-7)\cdot\left(x^2+4\right)\ \text{coeffs},x \rightarrow \begin{pmatrix} -2268 \\ -1440 \\ 161 \\ -456 \\ 186 \\ -24 \\ 1 \end{pmatrix}$$

$$\text{LaGuerra}\left(Z, 10^{-13}\right) = \begin{pmatrix} -0.99999999999998685 \\ -1.9999999999999915i \\ 1.9999999999999849i \\ 6.9999999999999973 \\ 8.9999998133418515 \\ 9.0000001866581449 \end{pmatrix} \quad \text{polyroots}(Z) = \begin{pmatrix} -1.0000000230252142 \\ -2.0000000149387862i \\ 2.0000000153338786i \\ 7.0000000388097439 \\ 8.9998382448970879 \\ 9.0001617261872831 \end{pmatrix}$$

Как видите, наша программа если и уступает функции polyroots, так только в скорости работы. Точность же определения ею корней полинома может даже (при малых TOL) превосходить точность, показываемую функцией polyroots. Она способна находить все виды корней: действительные, комплексные, кратные.

Зная суть алгоритма Лаггера, несложно понять, отчего точность определения корней с повышением степени полинома резко падает. Это связано с тем, что нахождению очередного корня предшествует операция деления полинома на соответствующий определенному ранее корню линейный множитель. Так как приближение к корню имеет ограниченную точность, коэффициенты нового полинома будут найдены с ошибкой. Соответственно, ошибка при вычислении нового корня будет складываться из общей погрешности метода и погрешности, связанной с неточностью коэффициентов. Поэтому данный корень будет менее точен по сравнению с предыдущим. Таким образом по мере вычисления корней точность будет неуклонно падать. Это можно заметить даже в случае полиномов не очень высокой степени. Так, если определить корни полинома степени 8, то часть из них будет точна до 13–14-го знака мантиссы. Но два-три корня будут иметь ошибку уже в 4–5-м знаке.

Учитывая описанную выше особенность метода Лаггера, его не стоит использовать для поиска корней полиномов высокой степени. Для этого лучше применять функцию root, находя каждый корень индивидуально. Или же можно обратиться к методу сопровождающей матрицы, который менее устойчив, однако не склонен к такому выраженному накоплению ошибки.

Метод сопровождающей матрицы мы не будем описывать так подробно, как метод Лаггера, однако изложим его основные идеи.

Из курса линейной алгебры известно, что любой полином может быть представлен в матричном виде как следующий определитель:

$$P(x) = \left|A - x\cdot 1\right|$$

Здесь A — так называемая сопровождающая матрица (companion matrix), x — переменная, I — единичная матрица, соразмерная A.

Сопровождающая матрица, по сути, хранит коэффициенты полинома. В случае полинома степени m сопровождающая матрица — это матрица размерности mxm, имеющая следующий вид:

$$A = \begin{pmatrix} -\frac{a_{m-1}}{a_m} & -\frac{a_{m-2}}{a_m} & \cdots & -\frac{a_1}{a_m} & -\frac{a_0}{a_m} \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}$$

Можно показать, что корням полинома соответствуют собственные значения сопровождающей матрицы. Следовательно, задача определения корней сводится к задаче нахождения собственных значений. Существуют специальные численные методы, которые позволяют решать эту задачу с высокой точностью. Один из этих методов реализован в Mathcad в форме функции eigenvals. Мы воспользуемся данной функцией, чтобы написать программу, реализующую метод сопровождающей матрицы. Код этой программы будет предельно прост:

```
comp_matr(vector) :=
  for i ∈ last(vector) − 1 .. 0
    I[0, last(vector)−1−i] ← −(vector[i] + vector[last(vector)])
  for j ∈ 0 .. last(vector) − 2
    I[j+1, j] ← 1
  eigenvals(I)
```

Несложно показать, что точность программы comp_matr совпадает с точностью встроенного алгоритма сопровождающей матрицы. Это означает, что они абсолютно идентичны.

$$Z := (24 \quad -50 \quad 35 \quad -10 \quad 1)^T$$

$$\text{comp\_matr}(Z) = \begin{pmatrix} 4.0000000000000338 \\ 2.9999999999999565 \\ 2.0000000000000164 \\ 0.99999999999999734 \end{pmatrix} \qquad \text{polyroots}(Z) = \begin{pmatrix} 0.99999999999999734 \\ 2.0000000000000164 \\ 2.9999999999999565 \\ 4.0000000000000338 \end{pmatrix}$$

8.1.4. Графическое решение уравнений

Графическое решение уравнения заключается в определении по графику функции, соответствующей левой части уравнения, при какой величине аргумента данная функция принимает значение, равное правой части уравнения. В Mathcad графический метод используется относительно редко. Однако есть случаи, в которых без него не

обойтись. Дело в том, что порой встречаются такие уравнения, которые нельзя решить ни аналитически, с использованием оператора solve, ни численно, применяя функцию root. Невозможность аналитического решения объясняется сложностью соответствующих уравнений. То же, что найти корни этих уравнений невозможно посредством численных методов, связано с тем, что описывающие их функции принимают не непрерывный, а дискретный набор значений. Приведем три примера таких функций:

$$f1(n) := \int_0^{\pi} \frac{d^n}{dx^n}\Gamma(x)\,dx \qquad f2(n) := \frac{n^2}{n!} \qquad f3(n) := \sum_{k=0}^{n}\left(\frac{2^k - 2^{-k}}{2^k + 2^{-k}}\right)$$

Функция f1(n) определена лишь при целых неотрицательных n, так как только такие значения могут соответствовать порядку производной. При аналогичных значениях n существует f2(n), что связано с тем, что в знаменателе ее выражения вычисляется факториал. Функция f3(n) дискретна, так как она задается суммированием членов ряда от 0 до n.

Чтобы решить уравнение графически, следует построить график функции, соответствующей его левой части. Затем нужно провести вспомогательную линию на уровне, определяемом правой частью уравнения. Область, в которой вспомогательная линия и кривая функции пересекаются, следует увеличить, используя инструмент Zoom (Масштаб) (подробно данный инструмент описывается в главе, посвященной графикам). Если решение нужно найти максимально точно, эту операцию следует проделать несколько раз. Затем необходимо определить координаты точки пересечения вспомогательной линии и кривой, задействовав инструмент Trace X-Y (Следовать X-Y) при включенном параметре Track Data Points (Следовать точкам данных).

Пример 8.20. Графическое решение уравнения

Пусть стоит задача с точностью до 10^{-10} найти сумму следующего бесконечного ряда:

$$\sum_{n=0}^{\infty} \frac{n^2}{n!}$$

Так как данный ряд является абсолютно сходящимся, то точность его приближения усеченным рядом не превышает величины первого отброшенного члена. Соответственно, нужно определить, при каком n член ряда принимает значение, меньшее 10^{-10}. Сделать это, используя оператор solve или функцию root, не получится. Но эту задачу можно решить графически (рис. 8.8).

Итак, необходимо просуммировать где-то 15—16 членов ряда. В том, что при этом будет достигнута требуемая точность, можно убедиться, вычислив сумму бесконечного ряда аналитически.

$$\sum_{n=0}^{\infty} \frac{n^2}{n!} \to 2\cdot\exp(1) \qquad \sum_{n=0}^{15} \frac{n^2}{n!} - 2e = -1.31\times 10^{-11}$$

В отдельных случаях функции дискретного аргумента можно преобразовать в идентичные, но непрерывные функции. При этом становится возможным использование численных и, порой, даже аналитических методов решения уравнений. Так, если в выражении дискретной функции присутствует факториал, то его можно заменить Г-функцией Эйлера. Данная функция является обобщением понятия факториала на всю область действительных чисел. Она непрерывна при положительных значениях аргумента, принимая при целочисленном значении аргумента те же

значения, что и факториал. Используя Г-функцию, поставленную выше задачу можно решить проще:

$$r(n) := \frac{n^2}{\Gamma(n)} - 10^{-10}$$

$$\text{root}(r(n), n, 1, 50) = 16.256$$

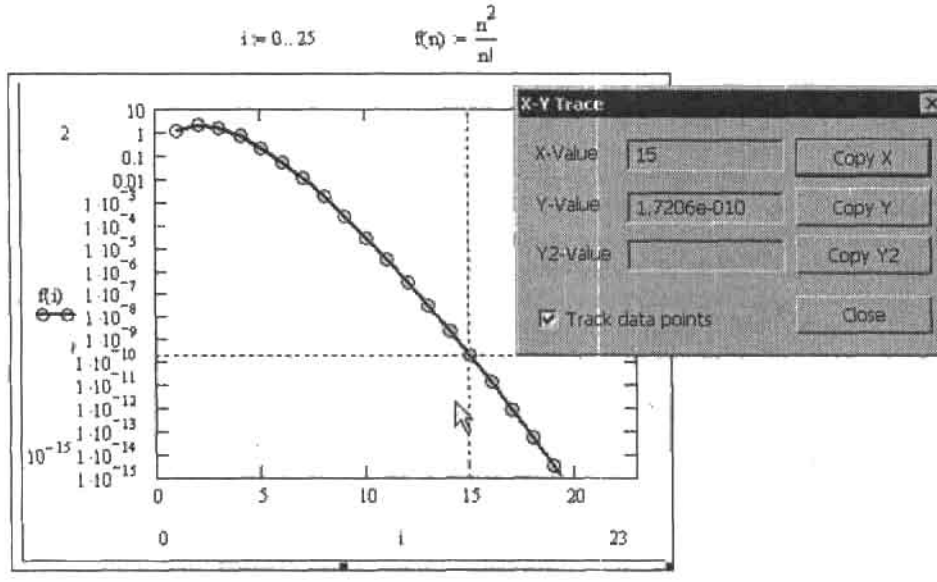


Рис. 8.8. Графическое решение уравнения. Так как функция изменяется крайне быстро и ее значения для разных n различаются на много порядков, то на оси Y следует использовать логарифмическую шкалу

8.2. Решение систем уравнений

Принципы, лежащие в основе как аналитического, так и численного решения систем уравнений схожи с принципами, на которых основывается решение одинарных уравнений. Поэтому, прежде чем приступить к изучению этого раздела, прочитайте раздел предыдущий. Многие описывающиеся в нем идеи мы не будем повторять в этом разделе, считая, что они уже прочно усвоены. К примеру, мы не будем подробно останавливаться на системной переменной TOL, так как в случае решения систем уравнений она исполняет ту же роль, что и в случае уравнений одинарных.

Данная глава разделена на две части. В первой мы изучим способы решения систем линейных уравнений, во второй — нелинейных. Однако вы должны понимать, что линейные уравнения есть не более чем подмножество нелинейных. Поэтому способы решения систем нелинейных уравнений могут быть использованы для решения систем линейных уравнений. Однако на практике проще использовать специализированные средства решения линейных уравнений, так как при этом не придется задавать начальные приближения, а ответ будет получен быстрее и точнее.

8.2.1. Решение систем линейных уравнений

Все методы решения систем линейных алгебраических уравнений можно разделить на две основные группы. К первой относятся так называемые прямые методы, например, Крамера или Гаусса. Во вторую группу входят довольно специфичные итеративные методы. В системе Mathcad реализованы методы обеих групп. При использовании прямых методов расчет можно вести как численно, так и символьно. Итеративные методы применяются лишь в численных расчетах.

Наиболее просто можно решить систему линейных уравнений, вспомнив некоторые правила линейной алгебры. Нижележащий пример демонстрирует аналитическое и численное решение системы уравнений с использованием обратной матрицы.

Пример 8.21. Решить систему линейных уравнений следующего вида:

$$x + y + z + p = 10$$

$$x - y - z - p = -8$$

$$x + y - z + p = 4$$

$$-x - y + z + p = 4$$

Прежде всего следует переписать систему в матричном виде. Для этого нужно составить матрицу коэффициентов, вектор неизвестных и вектор правых частей.

$$KOF := \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & -1 \\ 1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \end{pmatrix} \quad X := \begin{pmatrix} x \\ y \\ z \\ p \end{pmatrix} \quad PR := \begin{pmatrix} 10 \\ -8 \\ 4 \\ 4 \end{pmatrix}$$

Теперь данную систему уравнений можно заменить тождественным ей матричным выражением KOF·X=PR. В справедливости его можно убедиться, выполнив матричное умножение. Чтобы выразить вектор решений, умножим слева обе части уравнения на матрицу, обратную KOF. В результате получим конечную формулу $X = KOF^{-1} \cdot PR$, по которой и будем вести расчет.

$$KOF^{-1} \cdot PR \rightarrow \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \qquad KOF^{-1} \cdot PR = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

Находить решение системы линейных уравнений через обратную матрицу вполне допустимо, если количество уравнений в системе невелико и расчет должен быть произведен только один раз. Если же система большая или же одновременно нужно решить много систем, то такой подход не является оптимальным. В Mathcad встроен более быстрый и точный метод решения систем линейных уравнений, основанный на LU-разложении (о его сути мы поговорим чуть ниже). Чтобы его использовать, следует обратиться к встроенной функции lsolve(M,v), где M — матрица коэффициентов, v — вектор правых частей. Расчет данная функция может вести как численно, так и аналитически.

Пример 8.22. Полином степени 3 проходит через точки (0, 5), (4, –23), (8, –3), (11, 100). Найти уравнение данного полинома

В общем виде уравнение полинома степени 3 может быть записано следующим образом:

$$P_3(x) = a \cdot x^3 + b \cdot x^2 + c \cdot x + d$$

Нам известны четыре точки, через которые проходит полином. На основании этой информации можно записать систему линейных уравнений, в которой неизвестными будут коэффициенты уравнения параболы a, b, c, d. Матрица коэффициентов и вектор правых частей для этой системы будут иметь следующий вид:

$$A := \begin{pmatrix} 0 & 0 & 0 & 1 \\ 4^3 & 4^2 & 4 & 1 \\ 8^3 & 8^2 & 8 & 1 \\ 11^3 & 11^2 & 11 & 1 \end{pmatrix} \qquad B := \begin{pmatrix} 5 \\ -23 \\ -3 \\ 100 \end{pmatrix}$$

Решаем полученную систему линейных уравнений, используя функцию lsolve. Расчет ведем численно и аналитически. На основании полученных данных задаем функцию полинома $P_3(x)$.

$$K := \text{lsolve}(A, B) \rightarrow \begin{pmatrix} \frac{113}{462} \\ \frac{-221}{154} \\ \frac{-1195}{231} \\ 5 \end{pmatrix} \qquad \text{lsolve}(A, B) = \begin{pmatrix} 0.245 \\ -1.435 \\ -5.173 \\ 5 \end{pmatrix}$$

$$P_3(x) := K_0 \cdot x^3 + K_1 \cdot x^2 + K_2 \cdot x + K_3$$

Строим график полинома, коэффициенты которого были определены, чтобы убедиться, что он действительно проходит через данные в условии задачи точки (рис. 8.9).

$$data_x := (0 \ \ 4 \ \ 8 \ \ 11)^T \qquad data_y := (5 \ \ -23 \ \ -3 \ \ 100)^T$$

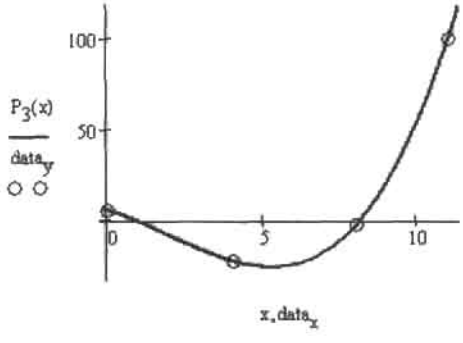


Рис. 8.9. Кубическая парабола, проходящая через четыре точки

В каких случаях решение СЛАУ нужно искать численно, а в каких аналитически? Аналитический расчет лучше в том отношении, что ответ выдается в виде «красивых» выражений и не содержит погрешности (главный недостаток такого подхода, связанный с возможностью появления громоздких дробей, легко решается их пересчетом в числа с плавающей точкой посредством оператора float). Также при этом допустимо, чтобы коэффициенты были не числами, а символами или символьными выражениями. Однако символьный расчет не стоит применять, если система очень большая (его при этом весьма сложно реализовать технически, так как в Mathcad напрямую не может быть задана матрица размерности больше, чем 10×10). Также лучше предпочесть численный метод, если коэффициентами являются числа с плавающей точкой. Если задействовать при этом оператор символьного вывода «→», то ответ будет получен, однако для его нахождения будет задействован тот же численный алгоритм, как если бы был использован оператор вывода «=». Правда, так как символьный процессор применяет не связанную с аппаратным уровнем арифметику длинных мантисс, то ответ при этом будет на пять знаков точнее. Однако и времени на его нахождение уйдет больше. Невозможно использовать символьный расчет также, если решение СЛАУ является частью алгоритма, реализованного в виде программы на языке Mathcad.

Пример 8.23. Решение СЛАУ, заданной в общем виде

$$A:=\begin{pmatrix} a & b & c \\ d & z & f \\ g & i & k \end{pmatrix}\quad B:=\begin{pmatrix} x1 \\ x2 \\ x3 \end{pmatrix}\quad \text{lsolve}(A,B)\rightarrow \begin{bmatrix} \frac{c\cdot x2\cdot i - x1\cdot f\cdot i - b\cdot k\cdot x2 + k\cdot z\cdot x1 - x3\cdot z\cdot c + b\cdot x3\cdot f}{-g\cdot z\cdot c + f\cdot g\cdot b + d\cdot c\cdot i - d\cdot b\cdot k + z\cdot a\cdot k - a\cdot i\cdot f} \\ \frac{-(g\cdot c\cdot x2 - g\cdot x1\cdot f + k\cdot d\cdot x1 - x3\cdot d\cdot c + f\cdot a\cdot x3 - x2\cdot a\cdot k)}{-g\cdot z\cdot c + f\cdot g\cdot b + d\cdot c\cdot i - d\cdot b\cdot k + z\cdot a\cdot k - a\cdot i\cdot f} \\ \frac{-d\cdot b\cdot x3 + x2\cdot g\cdot b - g\cdot z\cdot x1 + z\cdot a\cdot x3 + d\cdot x1\cdot i - a\cdot i\cdot x2}{-g\cdot z\cdot c + f\cdot g\cdot b + d\cdot c\cdot i - d\cdot b\cdot k + z\cdot a\cdot k - a\cdot i\cdot f} \end{bmatrix}$$

Матричные методы решения СЛАУ не могут быть использованы, если количество уравнений и количество неизвестных в системе не совпадает (при этом матрица коэффициентов не будет квадратной). Тут возможно два случая. Если количество неизвестных превышает количество уравнений, то система будет иметь бесконечное множество решений. При этом часть неизвестных можно выразить через остальные. Для этого нужно использовать оператор solve (см. пример 8.24). Если количество уравнений превышает количество неизвестных, то, как правило, часть из них является избыточными. Однако просто отбросить часть уравнений нельзя, так как среди оставшихся уравнений некоторые могут являться просто линейными комбинациями друг друга. Выкидывая избыточные уравнения, нужно обязательно проверять остаточную систему на линейную независимость, например, определяя ранг матрицы коэффициентов или вычисляя определитель. Впрочем, можно обойтись без всех этих операций, если использовать для решения системы оператор solve (аналитический расчет) или же блок Given-Find (численный расчет). Подробно о данном блоке мы поговорим ниже, а пока ограничимся лишь примером.

Пример 8.24. Решение СЛАУ в случае неравенства количество уравнений и количества неизвестных

Если количество неизвестных превышает количество уравнений, то одни неизвестные можно выразить как функциональные зависимости от других неизвестных:

$$\begin{pmatrix} x+y+z+k=4 \\ x-y+z-k=0 \\ -x+y+z+k=2 \end{pmatrix} \text{solve},x,y,z \rightarrow (1 \quad -k+2 \quad 1)$$

Если количество уравнений превышает количество неизвестных, то найти корни аналитически можно, использовав оператор solve.

$$\begin{pmatrix} x+y+z=3 \\ x-y+z=1 \\ -x+y+z=1 \\ -x-y-z=-3 \end{pmatrix} \text{solve},x,y,z \rightarrow (1 \quad 1 \quad 1)$$

Численно решить систему, в которой количество уравнений превышает количество неизвестных, позволяет блок Given-Find.

$$x := 0 \qquad y := 0 \qquad z := 0$$

Given

$$x-y+z=1 \qquad x+y+z=3 \qquad -x+y+z=1 \qquad -x-y-z=-3$$

$$\text{find}(x,y,z) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Говоря о численном решении СЛАУ, важно затронуть вопрос о точности используемых в Mathcad алгоритмов. В случае обычных систем, вроде приводимых в задачниках, точность функции lsolve или блока Given-Find очень высока. Ответ обычно верен вплоть до 14–15-го знака мантиссы, что близко к предельной теоретической точности численных методов на машинах с 64-битовыми числами с плавающей точкой. Однако с увеличением размеров системы точность обычно падает. Также крайне важно, насколько разнородные уравнения образуют систему. Если коэффициенты двух или нескольких уравнений близки, то при малых возмущениях в матрице коэффициентов A или векторе правых частей B произойдут большие изменения в векторе неизвестных $X=A^{-1}\cdot B$. Это означает, что численный метод будет неустойчив и, следовательно, резко повысится вероятность получения ответа, содержащего большую погрешность.

Системы, содержащие практически линейно зависимые уравнения, называются плохо обусловленными. Количественной мерой обусловленности может являться определитель: чем он ближе к нулю, тем хуже обусловлена система. Однако есть и более объективная характеристика, называемая числом обусловленности (condition number). Число обусловленности определяется как произведение нормы матрицы коэффициентов A на норму обратной ей матрицы. Чем больше число обусловленности, тем выше вероятность получения ответа со значительной погрешностью. В Mathcad есть функции, позволяющие определять число обусловленности. Данные функции различаются тем, на основании каких норм находится число обусловленности. Перечислим их: cond1 (норма L1), cond2 (норма L2), conde (эвклидова норма), condi (бесконечная норма). Расчет данные функции могут вести как численно, так и аналитически.

Пример 8.25. Влияние обусловленности системы на точность результата

Классической плохо обусловленной системой линейных уравнений является система, матрица коэффициентов которой является матрицей Гильберта. Матрица Гильберта A — это матрица, значение элементов которой определяется формулой $A_{i,j}=1/(1+i+j)$, где i и j — индексы элемента (отсчет строк и столбцов матрицы ведется с нуля). Создадим функцию, которая будет формировать матрицы Гилберта произвольной размерности.

$$A(N) := \left| \begin{array}{l} \text{for } i \in 0..N-1 \\ \quad \text{for } j \in 0..N-1 \\ \quad\quad M_{i,j} \leftarrow \dfrac{1}{j+i+1} \\ M \end{array} \right. \qquad A(3) \rightarrow \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{pmatrix}$$

Очевидно, что чем больше будет матрица Гильберта, тем в меньшей степени будут отличаться нижние рядки. Соответственно, тем хуже будет матрица обусловлена. Посмотрим, как зависят определитель матрицы Гильберта и ее число обусловленности от ее размерности.

$$|A(2)| = 0.083 \qquad |A(3)| = 4.63\times10^{-4} \qquad |A(4)| = 1.653\times10^{-7}$$

$$\text{conde}(A(2)) = 19.333 \quad \text{conde}(A(3)) = 526.159 \quad \text{conde}(A(4)) = 1.561\times10^{4}$$

Наше предположение подтвердилось. Стоит ожидать, что с увеличением размерности матрицы Гильберта погрешность определения корней будет возрастать. Проверим это. В первую очередь создадим функцию, которая будет формировать вектор правых частей, соразмерный матрице Гильберта. Так как нас интересует только погрешность, не имеет значения, какие у системы будут корни. Поэтому мы будем заполнять вектор правых частей одними единицами.

$$B(N) := \left| \begin{array}{l} \text{for } i \in 0..N-1 \\ \quad V_i \leftarrow 1 \\ V \end{array} \right.$$

Чтобы можно было оценить погрешность численного метода, необходимо решить систему точно. В нашем случае это возможно, так как значения элементов матрицы Гильберта могут быть представлены простыми дробями, а в векторе правых частей имеются только целые числа. Результат, возвращенный функцией lsolve, присвоим переменной T_c.

$$N := 3 \qquad T_c := \text{lsolve}(A(N), B(N)) \rightarrow \begin{pmatrix} 3 \\ -24 \\ 30 \end{pmatrix}$$

Далее находим среднюю ошибку численного метода. Для этого от вектора с определенными им корнями отнимаем вектор T_c. При этом будет получен вектор абсолютных ошибок. Чтобы найти ошибки относительные, данный вектор делим на вектор T_c, применив оператор векторизации. Затем умножаем вектор относительных ошибок на 100, чтобы получить вектор процентных ошибок. И, наконец, находим среднее арифметическое ошибок, задействовав функцию mean. Все описанные действия легко совместить в одной формуле.

$$err\% := \left| mean\left(\left[\left(\overrightarrow{lsolve(A(N), B(N)) - T_c} \right) \div T_c \right]^2 \right) \cdot 100 \right| \qquad err\% = 2.536 \times 10^{-13}$$

Изменяя значения N, находим среднюю ошибку и число обусловленности для матриц Гильберта разной размерности. Полученные данные заносим в табл. 8.1.

Таблица 8.1. Зависимость средней ошибки и числа обусловленности от размерности матрицы Гильберта

Размерность, N	3	5	7	9	11	13
Число обусловленности	526.1	$4.8 \cdot 10^5$	$4.8 \cdot 10^8$	$5.0 \cdot 10^{11}$	$5.3 \cdot 10^{14}$	$5.7 \cdot 10^{17}$
Средняя ошибка	$2.5 \cdot 10^{-13}$	$7.2 \cdot 10^{-11}$	$3.3 \cdot 10^{-7}$	0.0003	0.173	732.877

Как видно из табл. 8.1, с увеличением числа обусловленности погрешность работы численного метода резко возрастает. Ошибка определения корня может составлять сотни процентов, делая результат абсолютно бесполезным. В нашем случае это наблюдается, когда размерность матрицы Гильберта достигает 13.

Пример 8.25 показывает, сколь важно оценивать степень обусловленности матрицы коэффициентов. Если система обусловлена плохо, то численному решению нужно предпочесть аналитическое. Если же это невозможно, то к результату следует относиться крайне осторожно. В отдельных случаях точность численного решения можно повысить, используя специальные техники вроде сингулярного разложения (о нем мы поговорим ниже).

Традиционно основным методом решения СЛАУ считается метод Гаусса. Его изучают на занятиях по математике во всех вузах, его описывают во всех книгах по численным методам. Однако на практике метод Гаусса не применяется, так как существуют более эффективные методы. В основе функции lsolve лежит алгоритм, основанный на LU-разложении матрицы. Мы, следуя установленному в этой главе принципу обучения, реализуем данный метод самостоятельно.

LU разложением квадратной матрицы A называется ее представление в виде произведения нижней треугольной матрицы L (то есть матрицы, все ненулевые элементы которой лежат только на главной диагонали и ниже ее) и верхней треугольной матрицы U (матрицы, элементы которой располагаются на главной диагонали и выше нее). К примеру, LU-разложение матрицы размерности 3×3 можно записать следующим образом:

$$\begin{pmatrix} L_{0,0} & 0 & 0 \\ L_{1,0} & L_{1,1} & 0 \\ L_{2,0} & L_{2,1} & L_{2,2} \end{pmatrix} \cdot \begin{pmatrix} U_{0,0} & U_{0,1} & U_{0,2} \\ 0 & U_{1,1} & U_{1,2} \\ 0 & 0 & U_{2,2} \end{pmatrix} = \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} \\ A_{1,0} & A_{1,1} & A_{1,2} \\ A_{2,0} & A_{2,1} & A_{2,2} \end{pmatrix}$$

Зная разложение матрицы коэффициентов A на треугольные матрицы L и U, решить систему очень просто. Действительно, если справедливо A·X=B (здесь X — вектор неизвестных, B — вектор правых частей), то исполняться будет и L·U·X=B. Произведение U·X даст вектор. Обозначим его Y. Найти вектор Y можно, решив систему L·Y=B. Так как матрица L является нижней треугольной, то это несложно сделать методом прямой подстановки. Данный метод заключается в следующих действиях.

- Переменная Y_0 равна $B_0/L_{0,0}$, так как в верхней строке матрицы L все остальные элементы — это нули.
- Перемножение второй строки матрицы коэффициентов L и вектора неизвестных Y даст уравнение $L_{1,0}\cdot Y_0+L_{1,1}\cdot Y_1=B_1$. Так как значение Y_0 было получено на прошлом шаге, то из данного уравнения легко найти Y_1: $Y_1=(B_1-L_{1,0}\cdot Y_0)/L_{1,1}$.
- В общем случае умножение n-й строки матрицы L на вектор Y даст следующее уравнение: $L_{1,0}\cdot Y_0+L_{1,1}\cdot Y_1+\ldots+L_{n,n-1}\cdot Y_{n-1}+L_{n,n}\cdot Y_n=B_n$. Так как неизвестные от Y_0 до Y_{n-1} были найдены ранее, то в данном уравнении будет только одно неизвестное — Y_n. Выразив его, получим рабочую формулу метода прямой подстановки:

$$Y_n=\frac{B_n-\sum_{i=0}^{n-1}L_{n,i}\cdot Y_i}{L_{n,n}}$$

Найдя вектор Y, решаем систему $U\cdot X=Y$, определяя корни исходной системы $A\cdot X=B$. Так как матрица U — верхняя треугольная, то это легко сделать с помощью метода обратной подстановки. Идея данного метода точно такая же, как у метода прямой подстановки. Отличие состоит в том, что неизвестные определяются в противоположном порядке. Рабочая формула метода обратной подстановки следующая:

$$X_n=\frac{Y_n-\sum_{i=n}^{last(Y)}U_{n,i}\cdot X_i}{U_{n,n}}$$

Итак, польза от представления матрицы коэффициентов в виде произведения треугольных матриц заключается в том, что при этом становится возможным нахождение корней посредством подстановок. Написать же программы прямой и обратной подстановки, зная рабочие формулы данных методов, проще простого:

$$\text{direct}(L,B):=\left|\begin{array}{l} Y_0\leftarrow B_0\div L_{0,0} \\ \text{for } n\in 1..\text{last}(B) \\ \quad Y_n\leftarrow\dfrac{B_n-\sum_{i=0}^{n-1}L_{n,i}\cdot Y_i}{L_{n,n}} \\ Y \end{array}\right.$$

$$\text{invert}(U,Y):=\left|\begin{array}{l} N\leftarrow \text{last}(Y) \\ X_N\leftarrow Y_N\div U_{N,N} \\ \text{for } n\in N-1..0 \\ \quad X_n\leftarrow\dfrac{Y_n-\sum_{i=n}^{N}U_{n,i}\cdot X_i}{U_{n,n}} \\ X \end{array}\right.$$

Как же можно на практике произвести LU-разложение? В Mathcad для этого используется весьма эффективный алгоритм Краута (Crout), описать который можно следующими шагами.

- Пусть дана невырожденная квадратная матрица размерности N. Назовем ее A. Будем предполагать, что данная матрица может быть представлена в виде произведения нижней треугольной матрицы L и верхней треугольной матрицы U.
- Перемножим матрицы L и U и поставим в соответствие полученным для элементов матрицы A выражениям их числовые значения. В результате получим систему из N^2 уравнений, каждое из которых описывается следующей общей формулой (i=0, 1...N–1, j=0, 1...N–1):

$$L_{i,0} \cdot U_{0,j} + L_{i,1} \cdot U_{1,j} + \ldots + L_{i,i} \cdot U_{i,j} = A_{i,j}$$

  Так как и в матрице L, и в матрице U главная диагональ заполнена, то в системе будет N^2+N неизвестных. Чтобы уменьшить количество неизвестных до N^2, Краут предложил считать, что все элементы главной диагонали матрицы L равны 1. Если принять это предположение, то система легко решается.
- Краут обнаружил, что выведенная им система решается обычными подстановками в том случае, если неизвестные ищутся в правильном порядке. Детали следующие.
  - Начинаем поиск, перебирая столбцы матрицы A, а не строки (то есть верхним циклом должен быть цикл по j, а не по i).
  - Перейдя к очередному столбцу, последовательно перебираем все элементы в нем посредством цикла по i.
  - Если i<j, то на основании $A_{i,j}$ и определенных ранее элементов матриц L и U находим $U_{i,j}$. Для этого используется следующая формула:

$$U_{i,j} = A_{i,j} - \sum_{k=0}^{i-1} L_{i,k} \cdot U_{k,j}$$

    В данной формуле сумма вычисляется, если i>0. Если же i=0, то сумму полагают равной нулю.
  - Если i>j, то вычисляется элемент матрицы U с индексами i,j. Для этого применяется следующая зависимость:

$$L_{i,j} = \frac{A_{i,j} - \sum_{k=0}^{j-1} L_{i,k} \cdot U_{k,j}}{U_{j,j}}$$

    В этой формуле сумма просчитывается, если j>0. Если j=0, то $L_{i,j} = A_{i,j} / U_{j,j}$.

Проделав мысленно несколько итераций алгоритма Краута, вы обнаружите, что используемые в приведенных выше формулах элементы матриц L и U всегда создаются до того, как будут востребованы. Также заметьте, что каждый отдельный элемент разлагаемой матрицы A участвует в вычислениях только один раз, давая элемент с такими же индексами в матрице L или в матрице U. В общем же, алгоритму Краута нужно проделать совсем немного вычислений, чтобы произвести LU-разложение.

На языке программирования Mathcad алгоритм Краута реализуется следующим кодом:

$$
\begin{aligned}
&\text{lu\_brain}(A) := \\
&\quad n \leftarrow \text{rows}(A) \\
&\quad (L \leftarrow \text{identity}(n)\ \ U \leftarrow L - L) \\
&\quad \text{for } j \in 0..n-1 \\
&\qquad \text{for } i \in 0..j \\
&\qquad\quad U_{i,j} \leftarrow A_{i,j} - \text{if}\left(i > 0, \sum_{k=0}^{i-1} L_{i,k} \cdot U_{k,j}, 0\right) \\
&\qquad \text{for } i \in j..n-1 \\
&\qquad\quad L_{i,j} \leftarrow \left(A_{i,j} - \text{if}\left(j > 0, \sum_{k=0}^{j-1} L_{i,k} \cdot U_{k,j}, 0\right)\right) \div U_{j,j} \\
&\quad (L\ \ U)
\end{aligned}
$$

Программа lu_brain довольно простая, но в ней есть несколько моментов, которые стоит пояснить.

- В алгоритме Краута считается, что на диагонали матрицы L лежат единицы. Чтобы учесть это, в качестве основы матрицы L нужно взять единичную матрицу соответствующей размерности. Создать такую матрицу можно, задействовав встроенную функцию identity.
- В качестве основы матрицы U должна быть взята нулевая матрица той же размерности, что и матрица L. Создать такую матрицу можно, отняв от матрицы L ее же.
- В рабочих формулах метода Краута суммы вычисляются лишь, если i>0 (формула для $U_{i,j}$) или j>0 (формула для $L_{i,j}$). В противном случае суммы полагаются равными 0. Чтобы это учесть, используя минимальный объем кода, применяем функцию if(cond, true, false), где cond — условие, true — значение, которое должно быть возвращено, если условие выполнится, false — значение, возвращаемое, если условие не выполнится.

Проверим программу lu_brain:

$$M := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad LU := \text{lu\_brain}(M)$$

$$LU_{0,0} = \begin{pmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 7 & 2 & 0 \end{pmatrix} \qquad LU_{0,1} = \begin{pmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & 0 & 0 \end{pmatrix} \qquad LU_{0,0} \cdot LU_{0,1} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Разложение было осуществлено верно. Однако любая ли невырожденная матрица может быть разложена методом Краута? Оказывается, нет. К примеру, при попытке разложить следующую матрицу Mathcad выдаст сообщение об ошибке: Divide by zero in function evaluation (При вычислении функции происходит деление на нуль).

$$\begin{pmatrix} 1 & 2 & 6 \\ 4 & 8 & -1 \\ -2 & 3 & 5 \end{pmatrix}$$

Проблема заключается в том, что если значением диагонального элемента матрицы U окажется 0, то при вычислении элемента матрицы L может возникнуть ошибка деления на 0. Что же делать, если это произойдет? То, что диагональный элемент оказался равным 0, это не более чем результат неудачного сочетания элементов в исходной матрице A. Но можно ли это сочетание изменить в более удачную сторону, не изменив саму систему? Конечно. С учетом того, что нет никакой разницы, в какой последовательности записаны уравнения в системе, мы можем поменять строки матрицы A местами. Если аналогичную модификацию произвести с вектором правых частей и вектором неизвестных, то система не изменится. Однако сочетание элементов в матрице коэффициентов, с точки зрения метода Краута, после перестановки может оказаться более удачным.

Технически перестановка строк в матрице A осуществляется с помощью так называемой матрицы перестановок. Матрица перестановок — это соразмерная матрице A матрица, в каждой строке и столбце которой имеется только один ненулевой элемент, равный 1. Несложно догадаться, что количество возможных матриц перестановок для матрицы размерности N составляет N!-1 (единичная матрица не является матрицей перестановки). Так, если N=3, то матриц перестановок будет пять:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Исходя из правил матричного умножения очевидно, что, при умножении матрицы A слева на матрицу перестановок, если в строке i ненулевой элемент находится в столбце j, то j-я строка матрицы A переместится на позицию i.

Итак, нам необходимо написать программу, которая будет генерировать матрицы перестановок произвольной размерности. Сделать это в Mathcad не так уж и просто. Мы используем для этого следующий алгоритм.

- Пусть нужно создать матрицу перестановок размерности N. Для начала сгенерируем вектор v, содержащий числа от 0 до N-1. Каждое число будет адресовать один столбец в матрице перестановок.
- Запускаем цикл от 0 до N-1 и последовательно перебираем строки матрицы перестановок.
- Перейдя к очередной строке, случайным образом выбираем из вектора v элемент. Его значение укажет, в элемент, принадлежащий какому столбцу, должна быть помещена единица.
- Так как в каждом столбце может быть только один ненулевой элемент, вырезаем из вектора v уже использованное значение индекса столбца. Это гарантирует, что данное значение не будет разыграно повторно.
- После того, как матрица будет сгенерирована, проверяем, не является ли она единичной. Если нет, то возвращаем ее как результат. Если же матрица получилась единичной, то генерируем матрицу перестановок заново.

На языке программирования Mathcad описанный алгоритм можно реализовать следующим образом:

```
per(N) := | for i ∈ 0.. N − 1
          |   clear_v_i ← i
          | while 1
          |   v ← clear_v
          |   for j ∈ 0.. N − 1
          |     (md_index ← round(rnd(last(v)))  a ← v_md_index)
          |     M_j,a ← 1
          |     break if last(v) = 0
          |     (v ← submatrix(v, 1, last(v), 0, 0)  continue ) if md_index = 0
          |     (v ← submatrix(v, 0, last(v) − 1, 0, 0)  continue ) if md_index = last(v)
          |     v ← stack(submatrix(v,0,md_index−1,0,0),submatrix(v,md_index+1,last(v),0,0))
          |     break if M ≠ identity(N)
          | M
```

В контексте написанных ранее программ функция per не содержит никаких новых приемов. Единственное, стоит пояснить, как производится случайный выбор элемента из вектора v. Для этого с помощью функции rnd генерируется случайное число от 0 до n, где n — индекс последнего в векторе элемента. Так как сгенерированное число может быть дробью, оно округляется до ближайшего целого, для чего используется функция round.

Проверим, как работает функция per, сгенерировав три матрицы перестановки размерности 3:

$$per(4) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad per(4) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad per(4) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Итак, функция per работает так, как было задумано. Теперь мы должны так модифицировать код, производящий LU-разложение, чтобы он при возникновении ошибки деления на нуль осуществлял перестановку строк матрицы. Проще это сделать, не переписывая код функции lu_brain, а создав новую функцию, использующую lu_brain как подпрограмму. Назовем мы эту функцию LU. Ее алгоритм будет заключаться в следующих действиях.

- Пытаемся произвести LU-разложение, вызвав функцию lu_brain. Если при этом возникает ошибка («отловить» ее можно, используя оператор on error), то генерируем матрицу перестановок и умножаем на нее разлагаемую матрицу. Полученную матрицу передаем функции lu_brain. Описанные действия повторяются до тех пор, пока разложение не удастся осуществить.

- Вполне может оказаться так, что матрица вырождена и, соответственно, ее нельзя представить в виде произведения треугольных матриц. Это означает, что ошибка деления на нуль будет возникать, какая бы матрица перестановок ни была использована. Поэтому важно ограничить количество итераций, в течение которых разложение должно быть произведено, например величиной в 2·N, где N — размерность матрицы. Если лимит на количество итераций окажется превышенным, нужно возвратить сообщение об ошибке, информирующее пользователя о сингулярности матрицы.
- Если у матрицы коэффициентов A при произведении LU-разложения были переставлены строки, то при решении системы уравнений такая же перестановка должна быть осуществлена и над вектором правых частей B. Чтобы это можно было сделать, необходимо знать, какая матрица перестановок была использована алгоритмом LU-разложения. Поэтому соответствующая матрица должна возвращаться функцией LU как третий элемент вектора ответа. Если же матрица перестановок не применялась, то возвращена должна быть единичная матрица. Кстати, именно так работает встроенная функция Mathcad lu, отвечающая за LU-разложение. Она возвращает матрицу перестановок, L и U матрицы, слитые в одну матрицу.

Код функции LU следующий:

```
LU(A) := | (n ← 0  N ← rows(A))
         | while 1
         |   on error lu_brain(A)
         |     error("Error! Probably, matrix is singular!")  if n > 2·N
         |     perm_matr ← per(N)
         |     A ← perm_matr·A
         |     (n ← n + 1  continue)
         |   (answer ← lu_brain(A)  break)
         | answer_{0,2} ← perm_matr  if n > 0
         | answer_{0,2} ← identity(N)  otherwise
         | answer
```

Проверим эффективность функции LU, разложив с ее помощью матрицу, с которой не справилась функция lu_brain:

$$M := \begin{pmatrix} 1 & 2 & 6 \\ 4 & 8 & -1 \\ -2 & 3 & 5 \end{pmatrix} \qquad res := LU(M)$$

$$res = \left[\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 4 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 6 \\ 0 & 7 & 17 \\ 0 & 0 & -25 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \right] \qquad \left(res_{0,2}\right)^{-1} \cdot res_{0,0} \cdot res_{0,1} = \begin{pmatrix} 1 & 2 & 6 \\ 4 & 8 & -1 \\ -2 & 3 & 5 \end{pmatrix}$$

Проверка показывает, что функция LU справляется со своей задачей хорошо. Значит, нам осталось объединить возможности функций LU, direct и invert в программе, непосредственно производящей решение СЛАУ. Ее код будет очень прост. Единственная тонкость состоит в том, что нужно не забыть умножить вектор правых частей на матрицу перестановок, использованную при вычислении LU-разложения. Также стоит проверить матрицу коэффициентов на сингулярность, вычислив ее ранг (для этого можно использовать встроенную функцию rank).

$$\text{lu\_solver}(A,B) := \left|\begin{array}{l} \text{error("Matrix is singular")} \ \text{if} \ \text{rank}(A) \neq \text{rows}(A) \\ \left(M \leftarrow \text{LU}(A) \quad L \leftarrow M_{0,0} \quad U \leftarrow M_{0,1} \quad \text{per\_m} \leftarrow M_{0,2}\right) \\ B \leftarrow \text{per\_m} \cdot B \\ (Z1 \leftarrow \text{direct}(L,B) \quad Z2 \leftarrow \text{invert}(U,Z1)) \\ Z2 \end{array}\right.$$

Решаем посредством функции lu_solver систему линейных уравнений и сравниваем полученный ответ с точным аналитическим решением:

$$A := \begin{pmatrix} 1 & 2 & 2 \\ 4 & 1 & -1 \\ -2 & 4 & 5 \end{pmatrix} \quad B := \begin{pmatrix} 4 \\ 7 \\ 2 \end{pmatrix} \quad K := \text{lsolve}(A,B) \rightarrow \begin{pmatrix} \frac{14}{9} \\ 1 \\ \frac{2}{9} \end{pmatrix} \quad \text{lu\_solver}(A,B) - K = \begin{pmatrix} -2.22 \times 10^{-16} \\ 6.661 \times 10^{-16} \\ -6.106 \times 10^{-16} \end{pmatrix}$$

Итак, написанный нами алгоритм работает очень неплохо. Точность решения системы из трех уравнений находится на уровне предельной точности численных расчетов. Причем ответ рассчитывается моментально.

Почему мы разбираем особенности решения систем линейных уравнений столь подробно? Дело в том, что это, пожалуй, наиболее важный вопрос в теории численных методов. Алгоритмы решения СЛАУ используются доброй половиной всех практически важных численных методов. К примеру, на них основывается нахождение обратной матрицы, они применяются при решении систем нелинейных уравнений, краевых задач, дифференциальных уравнений в частных производных; без них невозможны многие алгоритмы интерполяции и регрессии, методы отыскания собственных значений матриц и многое-многое другое. По причине основополагающей роли методов решения СЛАУ, в курсах численных методов именно с них принято начинать изложение материала.

Продемонстрируем, как в Mathcad, на основании алгоритма решения СЛАУ, вычисляется обратная матрица.

Пусть дана квадратная матрица M и нужно найти матрицу, обратную данной. По определению обратной матрицы, $M \cdot M^{-1} = I$, где I — единичная матрица соответствующей размерности. Исходя из того, как производится матричное умножение, можно записать $M \cdot (M^{-1})_j = I_j$, где $(M^{-1})_j$ — j-й столбец обратной матрицы, I_j — j-й столбец единичной матрицы. По сути, $M \cdot (M^{-1})_j = I_j$ является системой линейных уравнений, где M — матрица коэффициентов, $(M^{-1})_j$ — вектор неизвестных, I_j — вектор правых частей. Решив данную систему, можно найти j-й столбец обратной матрицы. Таким образом, чтобы найти обратную матрицу для матрицы размерности N, нужно N раз решить систему из N линейных уравнений.

На языке программирования Mathcad реализовать функцию, вычисляющую обратную матрицу, можно следующим кодом:

$$\text{inverter}(M) := \left| \begin{array}{l} (N \leftarrow \text{rows}(M) \quad B \leftarrow \text{identity}(N)) \\ \text{error("Matrix must be squarte!")} \ \text{ if } \text{rows}(M) \neq \text{cols}(M) \\ \text{error("Matrix is singular!")} \ \text{ if } \text{rank}(M) \neq N \\ \text{for } i \in 0..N-1 \\ \quad Z^{\langle i \rangle} \leftarrow \text{lu\_solver}\left(M, B^{\langle i \rangle}\right) \\ Z \end{array} \right. \qquad A := \begin{pmatrix} 1 & 2 & 2 \\ 4 & 1 & -1 \\ -2 & 4 & 5 \end{pmatrix}$$

$$\text{inverter}(A) = \begin{pmatrix} 1 & -0.222 & -0.444 \\ -2 & 1 & 1 \\ 2 & -0.889 & -0.778 \end{pmatrix}$$

LU-разложение квадратной матрицы, которому мы посвятили столько внимания, полезно не только при решении СЛАУ. В Mathcad оно используется для нахождения определителя матрицы. В линейной алгебре доказано, что определитель равен произведению диагональных элементов верхней треугольной матрицы U. Попробуем, воспользовавшись данным фактом, самостоятельно создать функцию, вычисляющую определитель матрицы.

$$\det(M) := \left| \begin{array}{l} \text{error("Matrix must be squarte!")} \ \text{ if } \text{rows}(M) \neq \text{cols}(M) \\ U \leftarrow \text{LU}(M)_{0,1} \\ \prod_{i=0}^{\text{rows}(M)-1} U_{i,i} \end{array} \right. \qquad A := \begin{pmatrix} 1 & 2 & 2 \\ 4 & 1 & -1 \\ -2 & 4 & 5 \end{pmatrix} \quad \det(A) = 9 \quad |A| = 9$$

Функция lsolve использует для решения СЛАУ LU-разложение матрицы коэффициентов. Однако существуют и другие матричные разложения, полезные при решении систем линейных уравнений: сингулярное разложение, QR-разложение, разложение Холецкого. Каждое из них предназначено для эффективного решения отдельного типа СЛАУ.

Наиболее востребованным на практике матричным разложением является так называемое сингулярное разложение. Дело в том, что такой подход дает возможность решать системы уравнений, матрицы коэффициентов которых практически вырождены. Во многих случаях сингулярное разложение позволяет найти решение там, где метод LU-разложения или метод Гаусса оказываются бессильными.

Сингулярное разложение заключается в представлении матрицы коэффициентов в виде произведения трех квадратных матриц той же размерности: $A=U \cdot \text{diag}(S) \cdot V^T$. Здесь diag(S) — диагональная матрица, ненулевыми элементами которой являются так называемые сингулярные числа разлагаемой матрицы. В Mathcad вектор сингулярных чисел возвращает встроенная функция svds. Матрицы U и V можно найти, используя функцию svd (они возвращаются слитыми в одну матрицу).

Чтобы найти на основании сингулярного разложения решение СЛАУ, нужно последовательно решить три системы линейных уравнений, в качестве матриц коэффициентов которых выступают полученные в результате проведения разложения матрицы U, V и diag(S). Так как обычно данные матрицы обусловлены куда лучше исходной матрицы, то для решения соответствующих систем можно применять стандартные методы вроде метода LU-разложения. В случае матрицы diag(S) решение можно полу-

чить и проще, поделив элементы вектора правых частей на соответствующие им элементы главной диагонали.

$$\text{sing\_solver}(A,B) := \left| \begin{array}{l} (m \leftarrow \text{svd}(A) \quad n \leftarrow \text{rows}(m) \quad k \leftarrow \text{cols}(m)) \\ U \leftarrow \text{submatrix}\left(m, 0, \frac{n}{2} - 1, 0, k-1\right) \\ V \leftarrow \text{submatrix}\left(m, \frac{n}{2}, n-1, 0, k-1\right) \\ \left(Z1 \leftarrow \text{lu\_solver}(U,B) \quad Z2 \leftarrow \overrightarrow{(Z1 \div \text{svds}(A))} \quad Z3 \leftarrow \text{lu\_solver}\left(V^T, Z2\right)\right) \\ Z3 \end{array} \right.$$

Проверьте эффективность программы sing_solver самостоятельно, решив плохо обусловленную систему (например, матрица коэффициентов которой является матрицей Гильберта).

Интересной особенностью сингулярного разложения является то, что оно может быть использовано по отношению к неквадратной матрице. Это дает возможность решать системы уравнений, в которых количество уравнений и количество неизвестных не совпадает. Также на основании сингулярного разложения может быть реализован алгоритм расчета линейной регрессии.

В Mathcad имеются функции и других матричных разложений, которые могут быть использованы для повышения эффективности решения специфических видов СЛАУ. Так, если матрица коэффициентов системы линейных уравнений является симметричной (то есть $A=A^T$) и положительно определенной (все элементы больше или равны нулю), то вместо LU-разложения гораздо техничнее применять так называемое разложение Холецкого. Данное разложение заключается в представлении матрицы коэффициентов A в виде произведения нижней треугольной матрицы L и ее транспонированной формы: $A=L\cdot L^T$. Чтобы найти решение СЛАУ на основании разложения Холецкого, нужно решить две системы линейных уравнений, матрицами коэффициентов в которых являются L и L^T. Так как данные матрицы являются соответственно нижней треугольной и верхней треугольной, то для этого можно использовать методы прямой и обратной подстановки.

Как это ни удивительно, но симметричные матрицы встречаются в приложениях довольно часто (к примеру, такой матрицей является матрица Гильберта). Поэтому разложение Холецкого имеет заметное практическое значение.

В Mathcad разложение Холецкого можно провести, обратившись к встроенной функции cholecky. Результатом работы данной функции является матрица L.

$$\text{chol}(A,B) := \left| \begin{array}{l} m \leftarrow \text{cholesky}(A) \\ z1 \leftarrow \text{direct}(m,B) \\ z2 \leftarrow \text{invert}\left(m^T, z1\right) \\ z2 \end{array} \right. \qquad A := \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{pmatrix} \quad B := \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \qquad \text{chol}(A,B) = \begin{pmatrix} 27 \\ -192 \\ 210 \end{pmatrix}$$

В отдельных случаях может быть полезно так называемое QR-разложение. Данное разложение заключается в представлении матрицы в виде произведения ортогональной матрицы Q (то есть $Q^T=Q^{-1}$) и верхней треугольной матрицы R. Впрочем, на практике QR-разложение для решения СЛАУ применяется нечасто, так как оно требует почти в два раза больше операций по сравнению с LU-разложением. Большее значение QR-разложение имеет при решении специфических задач (например, определении собственных значений и векторов матрицы).

Все способы решения СЛАУ, которые мы рассмотрели выше, являются прямыми. Это означает, что для них заведомо известно количество действий, которое должен проделать алгоритм, чтобы найти корни. Точность работы таких алгоритмов зависит только от особенностей системы уравнений. В этом есть свои преимущества, однако есть и недостатки. Так, решая посредством прямых методов СЛАУ, мы не можем задать точность решения. Поэтому, в общем случае, нельзя быть уверенным, что корни были определены с достаточной точностью. Если система плохо обусловлена, погрешность может быть очень высока. А можно ли создать для решения СЛАУ итеративный алгоритм, в котором бы использовались явные критерии сходимости к корню и итерации совершались до тех пор, пока корни не удовлетворят данным критериям? Такие методы есть. Опишем самый простой итеративный алгоритм решения СЛАУ — алгоритм Якоби.

Суть работы алгоритма Якоби заключается в следующем: из каждого уравнения системы выражается по одному неизвестному. На первом шаге находится первое приближение простой подстановкой определенных пользователем начальных приближений к переменным. На втором круге работы алгоритма в каждое выражение подставляются полученные на шаг ранее приближения. Таким образом вычисляются следующие, более близкие к корням, приближения. Если полученные значения не удовлетворяют условиям точности, то цикл повторяется до тех пор, пока нужное приближение достигнуто не будет.

Несложно догадаться, что рабочей формулой метода Якоби будет следующая формула (здесь x — вектор приближений к корню, A — матрица коэффициентов, B — вектор правых частей):

$$x_i = \frac{B_i - A_{i,0}\cdot x_0 - A_{i,1}\cdot x_1 - \ldots - A_{i,i-1}\cdot x_{i-1} - A_{i,i+1}\cdot x_{i+1} - \ldots - A_{i,n-1}\cdot x_{n-1}}{A_{i,i}}$$

Критерием того, что приближения к корню уже достигли нужной точности, может быть то, что подстановка их в уравнения дает величины, отклоняющиеся от соответствующих элементов вектора B не более чем на TOL.

Согласитесь, что метод Якоби — это весьма оригинальный алгоритм. То, что при таких, на первый взгляд, довольно странных действиях корни все-таки находятся, можно объяснить тем, что в ходе работы программы происходит своеобразное смешение уравнений, в результате чего наступает усреднение, которое шаг за шагом приближает функции к их общей точке. Правда, для этого должно выполняться условие: матрица системы должна быть строго диагонально доминирующей (то есть величина элемента главной диагонали должна превышать сумму остальных элементов строки). В противном случае алгоритм может не сойтись (требование диагонального доминирования является достаточным условием сходимости метода Якоби, но не необходимым).

Реализовать алгоритм Якоби на языке программирования Mathcad можно следующим кодом (A — матрица коэффициентов, B — вектор правых частей, L — вектор начальных приближений, TOL — точность):

$$\text{Jacobi}(A, B, L, TOL) := \left| \begin{array}{l} (x \leftarrow L \quad N \leftarrow \text{rows}(A) - 1 \quad M \leftarrow \text{cols}(A) - 1) \\ \text{while} \sum_{i=0}^{N} \left(\left| \sum_{j=0}^{M} A_{i,j} \cdot x_j - B_i \right| < TOL \right) \neq M + 1 \\ \quad \text{for } i \in 0..N \\ \quad\quad x_i \leftarrow \dfrac{B_i - \text{if}\left(i > 0, \sum_{j=0}^{i-1} A_{i,j} \cdot x_j, 0\right) - \text{if}\left(i < M, \sum_{k=i+1}^{M} A_{i,k} \cdot x_k, 0\right)}{A_{i,i}} \\ x \end{array} \right.$$

Проверку того, насколько хорошо работает функция Jacobi, проведите самостоятельно.

Из встроенных средств Mathcad итеративный алгоритм для решения СЛАУ применяет блок Given-Find (в настройках функции find должен быть активирован пункт Linear). Данным алгоритмом является широко известный симплекс-метод. В своей классической формулировке симплекс-метод используется не для решения СЛАУ, а для поиска экстремумов линейных функций (задачи линейного программирования). Однако если немного преобразовать СЛАУ, то симплекс-метод может быть применен и для определения корней системы линейных уравнений.

Ввиду того что симплекс-метод не является специализированным средством решения СЛАУ, корни он, как правило, находит менее точно, чем функция lsolve. Кроме того, он может и не сойтись к решению, что происходит довольно часто. Поэтому единственным преимуществом блока Given-Find в случае решения СЛАУ является то, что система уравнений может быть записана не в матричной, а в естественной форме.

8.2.2. Аналитическое решение систем нелинейных уравнений

С помощью символьного процессора Mathcad можно получить аналитическое решение системы уравнений. Сделать это можно двумя способами. Во-первых, можно воспользоваться оператором solve (решить). В этом случае система должна быть внесена в его левый маркер в виде вектора. Переменные, значение которых должно быть найдено, следует ввести через запятую в правый маркер оператора solve. Ответ будет возвращен в виде матрицы, в строках которой будут записаны корни найденных решений. Их порядок будет таким же, каким был порядок соответствующих переменных в правом маркере оператора solve.

Пример 8.26. Решение системы уравнений с помощью оператора solve

$$\begin{pmatrix} \sqrt[4]{u+v} - \sqrt[4]{u-v} = 2 \\ \sqrt{u+v} - \sqrt{u-v} = 8 \end{pmatrix} \text{solve}, u, v \rightarrow (41 \quad 40)$$

Во-вторых, можно использовать так называемый вычислительный блок. Вычислительным блоком в Mathcad мы будем называть систему из вводного слова (Given (Дано)) и функции той или иной математической операции (например, Find — решение систем

уравнений, Odesolve — решение систем дифференциальных уравнений, Minerr — поиск точки минимальной невязки системы).

Чтобы решить систему уравнений с помощью вычислительного блока, выполните следующую последовательность действий.

1. Наберите вводное слово Given.
2. Строго под вводным словом задайте систему уравнений. Делается это, в отличие от случая использования оператора solve, точно так же, как при ее решении на бумаге. В качестве знаков равенства следует использовать логическое равенство (Bold Equal — Ctrl+=). Если система приведена к стандартному виду, то, аналогично поиску корней одиночных уравнений, можно определить лишь левые части ее уравнений.
3. Введите функцию решения систем уравнений find(x1,x2,...). В скобках через запятую задайте переменные в том порядке, в котором должны быть расположены в ответе соответствующие им корни.
4. В качестве оператора вывода результата работы функции find(x1,x2,...) используйте оператор символьного вывода «→». Если же вы примените оператор численного вывода «=», то для решения системы, при условии добавления начальных приближений, будет запущен один из численных алгоритмов.

Пример 8.27. Решение системы уравнений с помощью вычислительного блока

GIVEN

$$\sqrt{x+y}+\sqrt{y+z}=3 \qquad \sqrt{y+z}+\sqrt{z+x}=5 \qquad \sqrt{z+x}+\sqrt{x+y}=4$$

$$\text{find}(x,y,z)\rightarrow\begin{pmatrix}3\\-2\\6\end{pmatrix}$$

Существенных различий между решением системы уравнений с помощью оператора solve и вычислительного блока Given-Find нет. Однако есть небольшая, но совсем неочевидная разница в форме представления результата. Она заключается в том, что значения корней, относящиеся к одному решению, в случае использования вычислительного блока располагаются в столбцах, а при применении оператора solve — в строках матрицы ответа. Если этого не знать, то можно очень легко запутаться и сделать ошибку (особенно если количество решений и переменных совпадает).

Пример 8.28. Различия в форме ответа при использовании оператора solve и вычислительного блока

Решаем систему с использованием оператора solve:

$$\begin{pmatrix}x+y=1\\3x^2-2+y=0\end{pmatrix}\text{solve},x,y\rightarrow\begin{pmatrix}\frac{1}{6}-\frac{1}{6}\cdot13^{\frac{1}{2}} & \frac{5}{6}+\frac{1}{6}\cdot13^{\frac{1}{2}}\\ \frac{1}{6}+\frac{1}{6}\cdot13^{\frac{1}{2}} & \frac{5}{6}-\frac{1}{6}\cdot13^{\frac{1}{2}}\end{pmatrix}$$

Решаем систему с помощью блока Given-Find:

Given

$$x + y = 1 \qquad 3x^2 - 2 + y = 0$$

$$\text{find}(x,y) \rightarrow \begin{pmatrix} \frac{1}{6} - \frac{1}{6}\cdot\sqrt{13} & \frac{1}{6} + \frac{1}{6}\cdot\sqrt{13} \\ \frac{5}{6} + \frac{1}{6}\cdot\sqrt{13} & \frac{5}{6} - \frac{1}{6}\cdot\sqrt{13} \end{pmatrix}$$

Mathcad может решать самые разнообразные системы. Лучше всего решаются системы алгебраических уравнений, хуже всего — системы, содержащие сильно разнородные по своей природе функции (например, экспоненту и синус). Неплохо справляется Mathcad и с системами алгебраических уравнений с параметрами. В общем же, трудности, которые связаны с видом уравнений при решении систем, очень схожи с аналогичными проблемами при поиске корней одного уравнения, о которых весьма подробно говорилось в начале главы, поэтому останавливаться на этом вопросе не будем. Приведем лишь несколько примеров решения в Mathcad систем различных типов.

Пример 8.29. Аналитическое решение систем нелинейных уравнений различных типов

Система алгебраических уравнений с параметрами:

$$\begin{bmatrix} x + y + z = 0 \\ c\cdot x + a\cdot y + b\cdot z = 0 \\ (x+b)^2 + (y+c)^2 + (z+a)^2 = a^2 + b^2 + c^2 \end{bmatrix} \text{solve}, x, y, z \rightarrow \begin{pmatrix} 0 & 0 & 0 \\ -b+a & -c+b & c-a \end{pmatrix}$$

Система логарифмических уравнений:

$$\begin{bmatrix} \log\left[\sqrt{(x+y)^2}\right] = 1 \\ \log(y) - \log(|x|) = \log(2) \end{bmatrix} \text{solve}, x, y \rightarrow \begin{pmatrix} \frac{10}{3} & \frac{20}{3} \\ -10 & 20 \end{pmatrix}$$

Система тригонометрических уравнений:

$$\begin{pmatrix} \cos(x)^2 + \cos(y)^2 = \frac{1}{4} \\ x + y = \frac{5\pi}{6} \end{pmatrix} \text{solve}, x, y \rightarrow \begin{pmatrix} \frac{1}{3}\cdot\pi & \frac{1}{2}\cdot\pi \\ \frac{4}{3}\cdot\pi & \frac{-1}{2}\cdot\pi \\ \frac{1}{2}\cdot\pi & \frac{1}{3}\cdot\pi \\ \frac{3}{2}\cdot\pi & \frac{-2}{3}\cdot\pi \end{pmatrix}$$

8.2.3. Численное решение систем нелинейных уравнений

Как и в случае одинарных уравнений, к численным методам решения системы уравнений следует обращаться тогда, когда с ней не справится аналитический процессор Mathcad. Реально это приходится делать очень часто, так как лишь редкие системы нелинейных уравнений могут быть решены символьно.

В Mathcad для численного решения систем уравнений служит блок Given-Find. Правила его задания схожи с правилами, использующимися при символьном решении систем уравнений, однако есть и особенности. Отметим их.

- Аналогично численным методам решения уравнений с одним неизвестным, сначала следует определить начальные приближения. В случае систем уравнений приближение должно быть определено для каждой переменной. Например:

$$x := 10^{-3} \qquad y := 10^{3} \qquad z := 1$$

- Для вывода результата после функции find следует ввести оператор численного вывода ◆=▸.

Используя блок Given-Find, можно решать системы, содержащие до 250 нелинейных уравнений и до 1000 линейных. Впрочем, на практике редко приходится встречаться с такими огромными системами, чтобы можно было проверить эти данные. Однако замечено, что уже на системах из 5-7 уравнений Given-Find начинает давать сбои. Поэтому к декларируемым в справочной системе возможностям нужно относиться с долей скепсиса.

Пример 8.30. Численное решение системы нелинейных уравнений

$$x := 10 \qquad y := 5$$

Given

$$\sqrt{x} + \sqrt{y} = 9 \qquad \sqrt[3]{x} + \sqrt[3]{y} = 5$$

$$\text{find}(x, y) = \begin{pmatrix} 64 \\ 1 \end{pmatrix}$$

Редкое нелинейное уравнение имеет только один корень. Многие из них (периодические функции) могут иметь бесконечное множество решений. В случае же систем нелинейных уравнений ситуация еще более усложняется тем, что для большинства из них весьма проблематично определить начальные приближения и даже количество корней (доказательство их существования для систем с большим, чем 2, количеством уравнений – это задача очень сложная и часто не решаемая в случае нелинейных разнородных функций). С этими проблемами можно было очень просто справиться в случае уравнения с одним неизвестным, определяя нужные характеристики чисто визуально на графике. Аналогично можно поступить, в принципе, и для систем из двух уравнений, но лишь в тех редких случаях, когда одна переменная однозначно выражается через другую. Например, для системы

$$x^2 + y^2 = 1$$
$$x^2 - 6y = 0$$

начальные приближения (и количество решений) действительно можно найти по декартовому графику, так как каждое из уравнений может быть представлено кривой

на плоскости (окружностью и параболой). Точки пересечения этих кривых — это и есть решения системы (рис. 8.10).

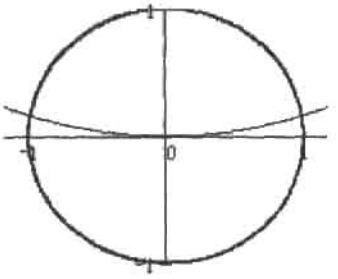


Рис. 8.10. Определение начальных приближений для численного метода решения системы уравнений с помощью графика

Но далеко не всегда можно представить каждое уравнение системы в виде функциональной зависимости одной переменной от остальных. В большинстве случаев разделение переменных невозможно. При этом единственный универсальный способ прикинуть приближения для системы из двух уравнений — это построить поверхности, представив каждое уравнение как функцию двух переменных. Однако точно определить приближения по трехмерному графику довольно сложно (это можно сделать лишь очень и очень приблизительно). А если в системе 3 или 4 неизвестные? При этом, естественно, никаких графиков построить нельзя. В этом случае приближения нужно брать такие, чтобы они были близки к ожидаемому результату. Поиск же всех решений системы уравнений с помощью численного метода — это довольно сложная и трудоемкая задача. Обычно с ней справляются, решая систему для целой области приближений.

Впрочем, если в системе только два уравнения, то очень часто приближения все же можно определить весьма точно. Для этого нужно построить график, на котором будут отображены кривые, получаемые при пересечении соответствующих уравнениям поверхностей плоскостью X0Y. Чтобы нарисовать такие кривые, нужно так преобразовать функции, чтобы они принимали только два значения: одно, когда функция больше 0, и второе, когда она меньше 0. Затем на основании этих функций следует построить контурный график (рис. 8.11).

Если окажется, что Mathcad не сможет найти решения при данных приближениях, попробуйте сменить их на другие. Вполне вероятна такая ситуация, что корни существуют, однако численный метод, «подбираясь» к ним, «попал в капкан» разрыва или экстремума. При этом дальнейшее вычисление приближений окажется невозможным. Также довольно распространенной ошибкой является попытка поиска комплексных корней при действительных приближениях. Чтобы найти мнимые корни, приближения также должны быть мнимыми.

Между численными методами, используемыми для решения одного уравнения и систем уравнений, нет принципиальных различий. Следовательно, основным параметром, определяющим точность решения, является порог, при достижении которого функция считается равной нулю. Как вы помните, при использовании функции root этому порогу соответствовала системная переменная TOL. Ту же роль исполняет данная переменная и при решении систем уравнений. Уменьшая TOL, вы увеличиваете точность, однако при этом возрастает время расчета и чувствительность к виду функции. Минимальное

значение TOL — 10^{-17}, стандартное — 10^{-3}. Стоит воздержаться от «профилактического» назначения TOL минимального значения. Дело в том, что количество итераций, которые может совершить численный метод, ограничено ввиду накопления ошибки. Поэтому при очень малом значении TOL имеется вероятность, что алгоритм просто не доберется до корня, исчерпав лимит на количество итераций. Величина TOL должна задаваться исходя из того, сколько знаков в ответе действительно значимо. В большинстве случаев TOL=10^{-3} обеспечивает вполне приемлемую точность.

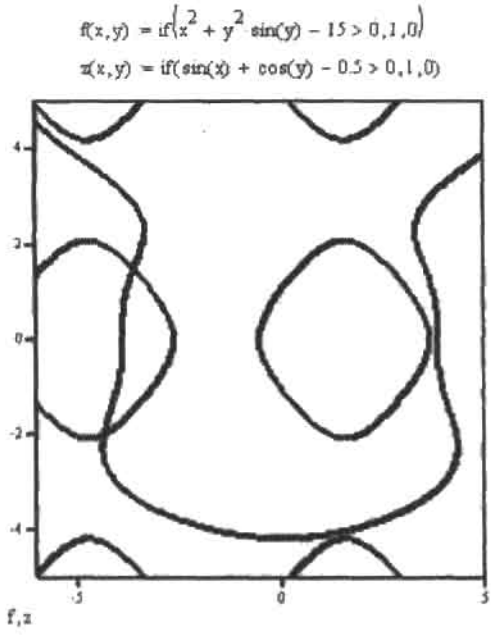


Рис. 8.11. Линии нулевого уровня для системы $x^2+y^2\cdot\sin(y)=15$ и $\sin(x)+\sin(y)=0.5$

При решении линейных уравнений в блоке Given-Find TOL не оказывает влияния на точность результата.

В Mathcad реализовано несколько алгоритмов численного решения систем уравнений. Если алгоритм, используемый по умолчанию, не справится с задачей, можно попытаться сменить его на другой. Для этого щелкните правой кнопкой мыши на тексте функции find. При этом откроется контекстное меню, в котором среди стандартных команд имеются три пункта, определяющие тип используемого метода.

- Пункт AutoSelect (Автоматический выбор). Численный алгоритм и его настройки будут определены автоматически. Если система линейная, то будет использован соответствующий метод. Если она нелинейная, то сначала будет осуществлена попытка найти решения методом сопряженных градиентов. Если она окажется неудачной, то будет задействован метод Левенберга. Если и он окажется неэффективным, то система применит квазиньютоновский метод. Чаще всего нет никакого смысла менять пункт AutoSelect на какой-то другой, так как при его активности Mathcad самостоятельно будет пытаться найти решения всеми доступными способами. Сменить его стоит лишь, если вы хотите применить тонкие настройки или же получить решение каким-то конкретным способом.

- Пункт Linear (Линейный) отвечает за реализацию итеративного метода (симплекс-метод) решения системы линейных уравнений. Обычно Mathcad самостоятельно относит систему уравнений к той или иной группе, поэтому специально настраивать этот параметр нет необходимости. Более подробно вопрос о решении СЛАУ был рассмотрен ранее.
- При подведении курсора к строке Nonlinear откроется всплывающее меню, содержащее ссылки на три различных численных метода решения систем уравнений: сопряженных градиентов (Conjugate gradient), Левенберга (Levenberg-Marquardt), квазиньютоновский (Quasi-Newton). Вдаваться в подробности каждого из них мы не будем по одной простой причине: очень трудно предсказать, какой из методов окажется эффективнее для данной системы уравнений. Чаще всего их эффективность оказывается близка, поэтому гораздо более принципиальное значение имеет правильный выбор начальных приближений и точности.

Чтобы сравнить эффективность встроенных в Mathcad численных методов, решим систему уравнений с заведомо известными корнями и сопоставим точности полученных результатов.

Пример 8.31. Точность решения системы уравнений различными методами

Выберем систему так, чтобы ее корни были равны, например, (1, 2, 3). Решать ее будем при стандартной точности (TOL=10^{-3}):

$$x := 0 \qquad y := 0 \qquad z := 0$$

Given

$$x^2 + y^2 + z^2 = 14 \qquad x + y + z = 6 \qquad x \cdot y \cdot z = 6$$

Метод сопряженных градиентов:

$$\text{find}(x, y, z) - \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -2.03226304051896 \times 10^{-6} \\ 2.35318717667354 \times 10^{-6} \\ -3.20924135155387 \times 10^{-7} \end{pmatrix}$$

Метод Левенберга:

$$\text{find}(x, y, z) - \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 0.000000000000000 \\ -2.88657986402541 \times 10^{-15} \\ 1.77635683940025 \times 10^{-15} \end{pmatrix}$$

Квазиньютоновский метод:

$$\text{find}(x, y, z) - \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -4.06755012072324 \times 10^{-6} \\ 4.76041154673368 \times 10^{-6} \\ -6.92861425233282 \times 10^{-7} \end{pmatrix}$$

Проверка показала более высокую эффективность метода Левенберга. Впрочем, это неудивительно, ведь в системе Mathcad он считается основным. Но и остальные алгоритмы показали неплохой результат. В общем же случае (ввиду того что на практике, как правило, суперточности и не требуется) нет разницы, какой из методов использовать, и прибегать к их смене стоит только в случае, если алгоритм, установленный по умолчанию, не смог найти решения. Если же система содержит много сложных уравнений и корни необходимо определить максимально точно, стоит сменить режим AutoSelect на режим Nonlinear/Levenberg-Marquardt.

В описываемом всплывающем меню, помимо выбора самого численного метода решения систем уравнений, можно настроить некоторые параметры его работы. Сделать это можно в специальном окне Advanced Options (Дополнительные параметры). Здесь имеется пять строк, в которых находятся по два переключателя, ответственных за ту или иную настройку (рис. 8.12).

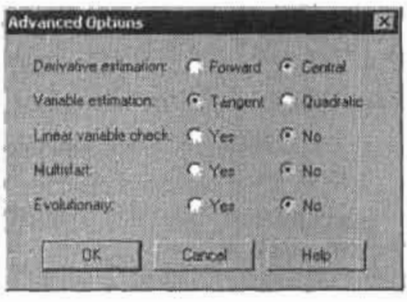


Рис. 8.12. Окно тонких настроек функции find

В первой строке (Derivative Estimation) можно настроить тип аппроксимации производной: либо центральной (Central), либо правой (Forward) конечной разностью. По умолчанию производная вычисляется по трехточечной симметричной схеме: это хотя и несколько более медленно, зато дает лучшие результаты. Так, при замене в методе сопряженных градиентов центральной схемы на правую двухточечную рассмотренная выше система решена не была, хотя при использовании параметра Central корни были найдены легко и с большой точностью. Поэтому не стоит менять схему вычисления производной, даже если корни не были найдены: это почти наверняка приведет лишь к худшим результатам.

Во второй строке можно определить тип аппроксимации функции: либо касательной (Tangent), либо параболой (Quadratic). Квадратичная аппроксимация более точная, но и более чувствительная к виду функции. Поэтому, дабы избежать лишних ошибок, по умолчанию определено использование касательных. Но если решение найдено не было, можно попробовать использовать квадратичное приближение.

Строка Linear Variable Check отвечает за проверку линейности вашей функции относительно переменных. Это может помочь сэкономить время за счет того, что частные производные будут приняты на первом шаге за константы и не будут вычисляться при каждой итерации. Впрочем, задачи, для которых этот параметр может быть полезен, крайне специфичны и редки, поэтому менять установки в этой строке не стоит.

Строка Multistart интересна только в том случае, если в блоке Given используется не функция find, а функции Maximize и Minimize (они служат для определения минимума и максимума функции). При активации данного параметра система будет пытаться найти глобальный экстремум, а не ограничится поиском ближайшего локального экстремума.

Настройка Evolutionary может помочь найти решения в случае функций с многочисленными экстремумами, точками и областями недифференцируемости. Используемые блоком Given-Find градиентные методы очень чувствительны к виду функции, поэтому столь важно правильно выбирать начальное приближение. Модификация же, активируемая настройкой Evolutionary, позволяет значительно снизить данную чувствительность. Также она помогает найти из нескольких решений оптимальное, даже если оно расположено далеко от точки начального приближения.

Настройки окна Advanced Options работают только для метода сопряженных градиентов и квазиньютоновского метода. На алгоритм метода Левенберга они никак не влияют.

Нельзя не упомянуть о еще одной очень полезной возможности блока Given-Find: в состав системы могут входить не только уравнения, но и неравенства (в которых можно использовать любые логические операторы). В некоторых случаях это может помочь ограничить количество решений, и тем самым облегчить поиск нужного.

Пример 8.32. Решение системы уравнений при наличии ограничений на величины корней

$$x := 1 \qquad y := 1$$

Given

$$x^2 - y^2 = 8$$

$$x^3 + y^3 = 27$$

X должен лежать в пределах от –10 до 10, y не должен быть равен 1:

$$-10 \le x \le 10 \wedge y \ne 1$$

$$\text{find}(x, y) = \begin{pmatrix} 2.972 \\ 0.912 \end{pmatrix}$$

Ограничения на величину корней не должны соблюдаться абсолютно точно. Аналогично тому, как функция считается равной нулю, если ее величина по модулю меньше TOL, так и ограничивающее неравенство будет считаться соблюденным, если отклонение от него не превышает определенного порога. Порог этот задается специальной системной константой CTOL (Constraint Tolerance — Точность ограничения). К примеру, при CTOL=10^{-3} неравенство $x \ge 5$ окажется невыполненным, если x=5,099. По умолчанию CTOL равняется 10^{-3}, ее предельная величина — 10^{-17}. Задать ее можно точно так же, как TOL: переопределением на рабочем поле или открыв вкладку Build-in Variables (Встроенные переменные) окна Worksheet Options (Параметры документа). Менять CTOL в сторону уменьшения нужно тогда, когда результат должен быть получен с максимальной точностью. Однако нужно учитывать, что при этом может возрасти время расчета вплоть до того, что система посчитает, что корня не существует. Поэтому стоит воздержаться от «профилактического» присваивания CTOL предельного значения.

Довольно очевиден следующий вывод: простые уравнения и системы лучше решать символьно, более сложные — численно. Вообще, численные методы куда более универсальны и надежны. Однако у них есть один весьма существенный недостаток: корни определяются не точно, а с заданной погрешностью вычислений. Предельно высоким уровнем точности работы функции find является 10^{-15}–10^{-16}. Для очень многих практически важных задач такой точности явно недостаточно. Кроме того, для всех итерационных численных методов характерно накопление ошибки, которое может довольно

серьезно исказить результат. Так, например, при решении систем уравнений аналитической химии для определения pH какого-то вещества или смеси веществ, искомое значение концентрации ионов водорода обычно весьма мало. Поэтому, при использовании численных методов, оно зачастую просто округляется до нуля. А так как pH — это отрицательный десятичный логарифм от концентрации H^+, то вполне понятно, что при таком округлении его значение правильно вычислено быть не может. Если же для решения этой задачи использовать символьный процессор, то ответ будет найден без каких-либо трудностей. Это связано с тем, что символьный процессор при приблизительном решении систем алгебраических уравнений использует арифметику длинных мантисс и предельный уровень точности увеличивается до 10^{-20}. Однако, увы, такой подход применим только к простейшим, алгебраическим системам.

В следующем примере предложено решение подобной задачи. Изучив его, вы поймете преимущество используемой символьным процессором арифметики длинных мантисс, а также важность верного выбора численного метода. Если вы не владеете элементами химии, то просто рассмотрите решение данной системы с точки зрения трудностей, которые могут возникнуть при малой величине параметров или корней уравнений.

Пример 8.33. Определение pH миллимолярного раствора гидрофосфата натрия

Начальные условия (k_3 — константа диссоциации гидрофосфата, k_2 — дигидрофосфата, k_w — константа диссоциации воды, C — концентрация гидрофосфата):

$$k_3 := 4.8 \cdot 10^{-13} \qquad k_2 := 6.2 \cdot 10^{-8} \qquad k_w := 10^{-14} \qquad C := 10^{-3}$$

Приближения для численного метода (z — концентрация фосфата, y — гидрофосфата, x — дигидрофосфата, o — OH^-, h — H^+):

$$z := 0 \qquad o := 0 \qquad h := 0 \qquad x := 0 \qquad y := 0$$

Решение системы уравнений, описывающей происходящие процессы диссоциации и гидролиза:

Given

$$\frac{x \cdot h}{y} = k_2 \qquad \frac{y \cdot h}{z} = k_3 \qquad o \cdot h = k_w$$

$$x + y + z = C \qquad x + 2y + 3z - o - 2C = 0$$

Ответ численного метода, использующегося по умолчанию (неверный в рамках рассматриваемой задачи):

$$\text{find}(x, y, z, o, h) = \begin{pmatrix} 3.333 \times 10^{-4} \\ 3.519 \times 10^{-4} \\ 3.148 \times 10^{-4} \\ -1.852 \times 10^{-5} \\ 0 \end{pmatrix}$$

Ответ метода Левенберга (точность в рамках данной задачи приемлемая):

$$\text{find}(x,y,z,o,h) = \begin{pmatrix} 4.842\times10^{-4} \\ 1.389\times10^{-6} \\ 5.144\times10^{-4} \\ 3.013\times10^{-5} \\ 1.778\times10^{-10} \end{pmatrix}$$

Решение символьного процессора (наиболее точное):

$$\text{find}(x,y,z,o,h) \rightarrow \begin{pmatrix} 5.315218722837805624\cdot10^{-4} & 9.967676533828310467\cdot10^{-4} & 4.718737001435861087\cdot10^{-4} \\ -1.390508258306147061\cdot10^{-6} & -1.618414241584358057\cdot10^{-7} & 1.387188384649792091\cdot10^{-6} \\ 4.698686359745255846\cdot10^{-4} & 3.394188041327389045\cdot10^{-6} & 5.267391114717640991\cdot10^{-4} \\ -6.165323630925497779\cdot10^{-5} & -9.933734653415036577\cdot10^{-4} & 5.486541328177990344\cdot10^{-5} \\ -1.621974870846944633\cdot10^{-10} & -1.006670738538620276\cdot10^{-11} & 1.822641944700806661\cdot10^{-10} \end{pmatrix}$$

Определение pH:

$$pH := \log\left(1.822641944700806607\cdot10^{-10}\right) \qquad pH = 9.739$$

Правильный выбор используемого метода играет первостепенную роль для верного и легкого решения уравнения или системы уравнений. Однако решаете ли вы задачу численно или используете возможности символьного процессора, очень внимательно относитесь к результату вычислений и при малейших сомнениях обязательно делайте проверку. Этот принцип проверки очень важен, поэтому мы еще не раз его повторим в этой книге. Постарайтесь ему следовать, и вы не допустите ошибку или оплошность почти наверняка.

Все употребляемые в Mathcad алгоритмы решения систем нелинейных уравнений относятся к так называемым градиентным, то есть связаны с приближением функций с помощью производных. Однако они могут отличаться целым рядом технических тонкостей (влияющих на эффективность алгоритмов): способом вычисления производной или решения возникающей на определенном этапе системы линейных уравнений, условием принятия точки в качестве корня или приемом разрешения проблемы сингулярности якобиана. Но все алгоритмы решения систем уравнений в Mathcad можно отнести к модификациям метода Ньютона.

Чтобы разобраться, в чем заключается данный метод и с чем могут быть связаны проблемы при его использовании, сначала рассмотрим простейший случай поиска корней одного уравнения с одним неизвестным. В этом случае принцип работы метода Ньютона довольно схож с уже рассмотренным ранее методом секущих и заключается в следующем: через определенную пользователем начальную точку проводится касательная к кривой функции, нуль которой нужно найти. В точке, в которой касательная пересекает ось X, вычисляется значение функции. Если оно меньше (по модулю) установленного минимума точности TOL, то данная точка определяется как корень. В противном случае через точку функции, которая соответствует полученному приближению,

проводится вторая касательная. Точка ее пересечения с осью X определяется как второе приближение. Если значение f(x) в ней больше по модулю, чем TOL, проводится третья касательная. И так далее до тех пор, пока условие корня не будет выполнено (рис. 8.13).

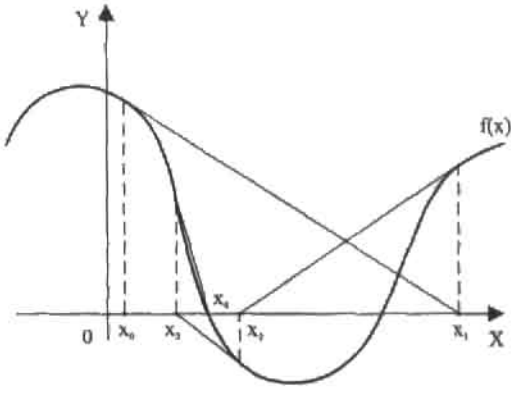


Рис. 8.13. Иллюстрация метода Ньютона. Достаточное приближение к корню достигается на пятой итерации

Теперь, зная идею метода Ньютона, попробуем самостоятельно написать программу поиска корней нелинейных уравнений. Для этого сначала следует решить проблемы чисто технического плана: во-первых, нужно придумать, каким образом мы будем проводить касательные через точки приближений, и, во-вторых, как затем находить точки их пересечения с осью X.

В общем случае условие пересечения прямой линии (каковой и является касательная) с осью X можно определить как:

$$k \cdot x + b = 0$$

Из этого уравнения требуется найти x, однако чтобы это сделать, нужно знать величины констант k и b. Определить одну из них предельно просто: ведь, как известно, k численно равна тангенсу угла наклона прямой к оси X. В случае же касательной тангенс этот равен значению производной функции в точке касания. То есть:

$$k = \frac{d}{dx} f(x_n)$$

Зная k, можно очень легко найти и b, исходя из того факта, что в точке касания значения функции касательной и исследуемой функции совпадают. То есть:

$$\frac{d}{dx} f(x_n) \cdot x_n + b = f(x_n)$$

Выражаем b:

$$b = f(x_n) - \frac{d}{dx} f(x_n) \cdot x_n$$

Подставляем полученные выражения для констант в первую формулу:

$$\frac{d}{dx} f(x_n) \cdot (x - x_n) + f(x_n) = 0$$

Из этого простого уравнения с одним неизвестным находим требуемую точку пересечения касательной с осью X:

$$x = x_n - \frac{f(x_n)}{\frac{d}{dx} f(x_n)}$$

Данная формула носит название формулы Ньютона. Зная ее, реализовать соответствующий численный метод предельно просто:

$$\text{Newton}(f, a, TOL) := \left| \begin{array}{l} (n \leftarrow 0 \quad x_n \leftarrow a) \\ df(x) \leftarrow \frac{d}{dx} f(x) \\ \text{while } 1 \\ \quad \left| \begin{array}{l} x_n \leftarrow x_n + TOL \text{ on error } 1 \div df(x_n) \\ x_{n+1} \leftarrow x_n - f(x_n) \div df(x_n) \\ n \leftarrow n + 1 \\ \text{return } (x_n \quad n)^T \text{ if } |f(x_n)| < TOL \wedge |f(x_n) - f(x_{n-1})| < TOL \\ \text{return error("Dont converge to a solution!") if } n > 100 \end{array} \right. \end{array} \right.$$

В коде программы Newton есть несколько тонкостей, которые стоит прокомментировать.

- Производную от f(x) необходимо задать как локальную функцию df(x). Это связано с тем, что если использовать оператор дифференцирования в выражении, соответствующем формуле Ньютона, напрямую, то возникнет ошибка (так как система воспримет $f(x_n)$ как постоянную величину, а не функцию от x).
- Если очередное приближение совпадет с экстремумом, то возникнет ошибка деления на нуль. Обойти эту проблему просто. Для этого нужно, при возникновении ошибки, прибавить к текущему приближению небольшую величину (например, TOL). Точка нового приближения сместится в сторону от точки экстремума, и поэтому значение производной в ней уже не будет равно нулю.
- Метод Ньютона является локально сходящимся. Это означает, что то, сойдется он к корню или нет, зависит от правильности выбора начального приближения. Важно предусмотреть ситуацию, в которой неудачное приближение породит расходящуюся последовательность приближений. В программе Newton отслеживается количество проделанных итераций. Если оно превышает 100, то возвращается сообщение об ошибке.

Проверим, насколько эффективна программа Newton, решив уравнение, корень которого можно точно найти с использованием оператора solve:

$$f(x) := \sin(x) - \frac{1}{x}$$

$$\sin(x) - \frac{1}{x} \text{ solve }, x \rightarrow 2.772604708265991234$$

$$\text{Newton}\left(f, 4, 10^{-3}\right) = \begin{pmatrix} 2.7726047083006953 \\ 5 \end{pmatrix} \quad \text{Newton}\left(f, 4, 10^{-15}\right) = \begin{pmatrix} 2.772604708265991 \\ 7 \end{pmatrix}$$

Проверка показывает, что метод Ньютона сходится очень быстро. Каждая итерация увеличивает точность в среднем на два знака мантиссы. Это больше, чем в случае таких методов, как метод секущих или Больцано. Но этот вывод вовсе не означает, что для решения всех уравнений лучше использовать вычислительный блок Given-Find (реализующий для нахождения корней одного уравнения с одним неизвестным схожий с описанным алгоритм). Все дело в том, что для метода Ньютона характерны те же недостатки, что и для метода секущих (трудности в поиске решений при наличии экстремумов, перегибов, точек разрыва и особенностей поведения функции на бесконечности). Однако, помимо этого, к описанным недостаткам добавляются еще и многочисленные проблемы, возникающие в связи с использованием производной. Поэтому в случае сложных уравнений лучше не рисковать и применять для решения более надежный алгоритм секущих (а еще лучше глобально сходящиеся методы интервалов локализации корня). Вообще, на практике метод Ньютона редко используется (ввиду описанных недостатков) для решения одиночного уравнения. Однако очень важно его расширение для систем уравнений. Конечно, в нем не исчезают присущие этому алгоритму слабости и недостатки, но для поиска решений систем уравнений просто пока не создано ничего лучшего.

Попробуем самостоятельно расширить метод Ньютона на случай систем уравнений. Для начала разберемся с наиболее простым случаем и придумаем идею алгоритма решения системы из двух нелинейных уравнений. Для этого вспомним, что любое такое уравнение — это частный случай (точка нуля) некоторой функции z=f(x, y), которую можно представить в пространстве как поверхность. Очевидно, что решением системы уравнений будет являться точка на линии пересечения (или просто точка касания) поверхностей, соответствующих каждому из уравнений. С другой стороны, значения обеих функций в такой точке должны быть равны нулю.

Итак, с тем, какой геометрический смысл имеет точка корня системы из двух уравнений, мы разобрались. Но как можно найти такую точку? Чтобы ответить на этот вопрос, вспомним, как мы справились с этой проблемой в двумерном случае. Тогда удалось найти корни благодаря тому, что сложные кривые были заменены на простую и техничную прямую. Первая мысль, которая возникает, — это просто повторить этот проверенный и такой эффективный шаг. Однако проблема заключается в том, что поверхность нельзя аппроксимировать прямой! Это то же самое, что заменить кривую точкой: глупо и бессмысленно. Поверхность должна быть заменена каким-то, по возможности более простым и легко исследуемым, но объемным объектом. Очевидно, что таким объектом является плоскость. Не менее очевидно, что две плоскости, приближающие в данной окрестности некоторые поверхности, почти наверняка пересекутся. Результатом их пересечения будет прямая. А точка, в которой данная прямая пересечет плоскость XOY, и будет либо корнем (в лучшем случае), либо приближением.

Нарисовать или тем более представить работу алгоритма, который бы реализовывал описанную выше идею, очень сложно. Поэтому просто поверим собственной интуиции.

Первый вопрос, на который мы должны ответить, это как можно провести касательную плоскость к поверхности в некоторой точке. Сделать это очень просто: для этого записываем разложение соответствующей функции в ряд Тейлора по двум первым членам. Математической тонкостью этого действия в случае функции двух переменных является то, что разложение должно быть проведено по обеим переменным.

$$f(x,y) = f(x_n,y_n) + \frac{\partial}{\partial x} f(x_n,y_n)(x - x_n) + \frac{\partial}{\partial y} f(x_n,y_n)(y - y_n)$$

Аналогичное выражение можно записать и для второго уравнения системы:

$$p(x,y) = p(x_n,y_n) + \frac{\partial}{\partial x} p(x_n,y_n)(x - x_n) + \frac{\partial}{\partial y} p(x_n,y_n)(y - y_n)$$

Так как значение обеих функций в точке корня равно 0, то имеем следующую систему линейных уравнений:

$$-p(x_n,y_n) = \frac{\partial}{\partial x} p(x_n,y_n)(x - x_n) + \frac{\partial}{\partial y} p(x_n,y_n)(y - y_n)$$

$$-f(x_n,y_n) = \frac{\partial}{\partial x} f(x_n,y_n)(x - x_n) + \frac{\partial}{\partial y} f(x_n,y_n)(y - y_n)$$

Из этой системы нам нужно найти значения x и y, которые являются координатами точки следующего приближения (или корня). Однако, раскрывая скобки, мы лишь усложним вид системы. Поэтому на данном этапе сделаем замены:

$$Y = y - y_n$$
$$X = x - x_n$$

Произведя замену, перепишем полученную систему из двух линейных уравнений в матричном виде:

$$\begin{pmatrix} \frac{\partial}{\partial x} p(x_n,y_n) & \frac{\partial}{\partial y} p(x_n,y_n) \\ \frac{\partial}{\partial x} f(x_n,y_n) & \frac{\partial}{\partial y} f(x_n,y_n) \end{pmatrix} \cdot \begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} -p(x_n,y_n) \\ -f(x_n,y_n) \end{pmatrix}$$

Левая матрица в произведении — это так называемый якобиан. В данном случае он играет роль матрицы коэффициентов.

Решить систему можно или через обратную матрицу, или (техничнее) используя функцию lsolve. Когда корни системы будут найдены, выражаем из них координаты точки нового приближения. Если это приближение удовлетворит критериям нуля функций f(x,y) и p(x,y), то возвращаем его как решение. В противном случае переходим к следующей итерации.

Важно учесть то, что якобиан может быть вырожден. Это означает, что в выбранной точке частные производные двух функций равны, а следовательно, аппроксимирующие плоскости будут параллельны и никакой точки приближения получено не будет. Также возможна ситуация, что одна из плоскостей будет параллельна плоскости XOY. В этом случае одна из строк якобиана будет нулевой, и он не будет нести необходимой информации. Как же таких проблем избежать? Наиболее простой вариант — не попадать

в точки, в которых якобиан вырожден. А сделать это можно очень просто: в том случае, если такое попадание все-таки происходит, следует просто изменить рабочие значения приближений, прибавляя к ним некоторую (небольшую) величину, например TOL.

Рассмотрим теперь, как можно решить систему из N уравнений. В общем-то, это можно сделать точно так же, как в случае системы из двух уравнений: решая систему из N линейных уравнений, в качестве матрицы коэффициентов которой выступает якобиан, вектора неизвестных — вектор разностей новых и старых приближений, вектора правых частей — вектор значений соответствующих уравнениям функций в точке текущего приближения, взятых с противоположным знаком.

Язык программирования Mathcad слишком примитивен, чтобы на нем можно было реализовать функцию, способную решать системы из произвольного количества уравнений. Поэтому мы ограничимся созданием функции, позволяющей решать системы из двух нелинейных уравнений.

$$
\begin{aligned}
&\text{sys}(f,p,x0,y0,TOL) := \left|
\begin{array}{l}
\left(n \leftarrow 0 \;\; x_0 \leftarrow x0 \;\; y_0 \leftarrow y0\right) \\
\left[dF_x(z,x,y) \leftarrow \frac{\partial}{\partial x} z(x,y) \;\; dF_y(z,x,y) \leftarrow \frac{\partial}{\partial y} z(x,y)\right] \\
\text{while } \left|f\left(x_n,y_n\right)\right| > TOL \vee \left|p\left(x_n,y_n\right)\right| > TOL \\
\quad \left|
\begin{array}{l}
\text{return error("Dont converge to a solutions")} \;\; \text{if } n > 100 \\
J \leftarrow \begin{pmatrix} dF_x\left(f,x_n,y_n\right) & dF_y\left(f,x_n,y_n\right) \\ dF_x\left(p,x_n,y_n\right) & dF_y\left(p,x_n,y_n\right) \end{pmatrix} \\
\left(x_{n+1} \leftarrow x_n + TOL \;\; y_{n+1} \leftarrow y_n + TOL \;\; n \leftarrow n+1 \;\; \text{continue}\right) \text{ if } |J| = 0 \\
Z \leftarrow \text{lsolve}\left[J, \begin{pmatrix} f\left(x_n,y_n\right) \\ p\left(x_n,y_n\right) \end{pmatrix}\right] \\
\left(x_{n+1} \leftarrow Z_0 - x_n \;\; y_{n+1} \leftarrow Z_1 - y_n \;\; n \leftarrow n+1\right)
\end{array}
\right. \\
\left(x_n \;\; y_n \;\; n\right)^T
\end{array}
\right.
\end{aligned}
$$

Попробуем решить с помощью созданного алгоритма следующую систему уравнений:

$$x^2 + y^2 = 16$$

$$y = \sin(x)$$

Для данной системы довольно легко подобрать начальные приближения. Это связано с тем, что в обоих уравнениях можно выразить одну переменную через другую, по причине чего система может быть визуализирована двумя кривыми. Корням же будут соответствовать пересечения этих кривых (рис. 8.14).

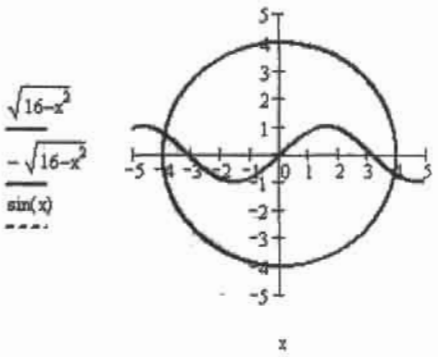


Рис. 8.14. У системы будет два решения. Первое будет расположено вблизи точки (4, –1), второе — (–4, 1)

С определением корней рассматриваемой системы, ввиду того, что мы знаем приблизительное расположение решений, программа sys легко справится:

$$f(x,y) := 16 - x^2 - y^2 \qquad p(x,y) := y - \sin(x)$$

$$\mathrm{sys}\left(f,p,4,-1,10^{-13}\right) = \begin{pmatrix} 3.9358754812 \\ -0.7133611965 \\ 4 \end{pmatrix} \quad \mathrm{sys}\left(f,p,-4,1,10^{-13}\right) = \begin{pmatrix} -3.9358754812 \\ 0.7133611965 \\ 4 \end{pmatrix}$$

В основе функции find лежат куда более сложные алгоритмы, чем созданный нами. Однако базовая концепция у них та же — метод Ньютона. Поэтому мы можем легко предсказать, какие у них будут слабые места. Чтобы решение было найдено, начальные приближения должны быть близки к корням. Если между приближением и корнем окажется экстремум, разрыв или область медленного изменения функции, то численный метод, скорее всего, не сойдется. Главная сложность заключается в том, чтобы такие приближения найти. Задача эта трудная и, если в системе больше двух уравнений, практически нерешаемая. По этой причине использовать численные методы решения систем уравнений крайне сложно. Увы, но универсальных и надежных алгоритмов пока не существует.

8.2.4. Приближенное решение систем уравнений

Очень часто приходится сталкиваться с системами, не имеющими решения. Однако иногда необходимо выяснить, при каких значениях переменных система не согласована наименее, иначе говоря, имеет минимальную невязку.

Для решения задач такого рода в Mathcad имеется специальная функция minerr(x1,...). По особенностям своего применения она абсолютно идентична функции find, то есть также требует ввода ключевого слова Given и определения начальных приближений. При своей работе функция minerr использует те же численные алгоритмы, что и функция find. Таким образом, те выводы, которые были сделаны нами в предыдущих подразделах для функции find, можно абсолютно спокойно распространить и на функцию

minerr, единственным отличием принципов работы которой является другое условие прекращения итераций. Так, функция minerr возвратит последние приближения при исчерпании лимита на итерации и вычислительные ошибки вне зависимости от того, удовлетворяют ли они критериям сходимости к корню или же нет.

Очень часто функция minerr оказывается способной найти те корни, которые не может определить find. Поэтому ее рекомендуется применять тогда, когда find оказывается не на высоте. Однако при этом необходимо обязательно осуществлять проверку найденных корней.

Сравним работу функций find и minerr при решении уравнения $x^2+y^2=0$. Уравнение это имеет только один корень (x=0, y=0) и является частным случаем функции, дающей поверхность в виде параболоида (рис. 8.15).

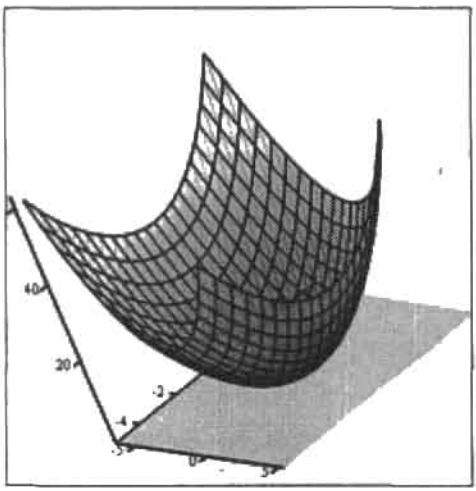


Рис. 8.15. Поверхность имеет с плоскостью XOY лишь одну общую точку

Особенностью нуля данной функции является то, что он относится по типу к касанию. Это вообще-то довольно трудная ситуация для любого из численных методов (особенно градиентных) и будет хорошей проверкой для обеих функций:

$$x := 1 \qquad y := 1$$

Given

$$x^2 + y^2 = 0$$

$$\text{find}(x, y) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$x := 1 \qquad y := 1$$

Given

$$x^2 + y^2 = 0$$

$$\text{minerr}(x, y) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

На этот раз find и minerr оказались на высоте и нашли решение верно. Но если попробовать решить аналогичное данному уравнение вида $k \cdot x^2+y^2=0$ при большом значении k (например, 1000), то упомянутый параболоид столь исказится, что найти корень с помощью функции find не получится ни при каком приближении, кроме x=0, y=0 (но это нельзя назвать успехом, поскольку при таких начальных точках происходит «прямое попадание»). А функция minerr справится с этой задачей легко.

Однако решение уравнений с двумя неизвестными — это не главная задача функции minerr. Куда более важной с практической точки зрения является возможность ее применения для поиска точки минимальной невязки несовместной системы. Практически важный случай таких систем связан с решением задач регрессии по методу наименьших квадратов.

Пример 8.34. Определение параметров нелинейной зависимости методом наименьших квадратов

Существует система, совершающая затухающие колебания. Нужно определить, каким законом можно описать данную систему, если известно общее уравнение затухающих колебаний. Оно имеет вид:

$$Y = A \cdot \sin(k \cdot t) \cdot e^{-b \cdot t}$$

Итак, необходимо узнать, чему равны параметры A, k, t. Для этого воспользуемся следующими экспериментальными данными:

$$t := (0\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 10)^T$$

$$y := (0\ \ 4.381\ \ 1.084\ \ -2.336\ \ -1.252\ \ 1.023\ \ 1.298\ \ -0.537\ \ -0.954\ \ 0.024\ \ 0.749)^T$$

Задаем функцию четырех переменных на основании общей формулы затухающих колебаний:

$$Y(t, A, k, b) := A \cdot \sin(k \cdot t) \cdot e^{-b \cdot t}$$

Задаем функцию метода наименьших квадратов. Она определяется как сумма квадратов отклонений экспериментальных данных от истинных значений зависимости при данных величинах аргумента.

$$Y\_err(t, A, k, b) := \sum_{i=0}^{last(t)} \left(Y\left(t_i, A, k, b\right) - y_i\right)^2$$

Если экспериментальные данные были получены без погрешности, то правильный подбор параметров A, k, b приведет к тому, что функция Y_err будет равна 0. Если же погрешность имеется, то Y_err при этом минимизируется. В первом случае параметры могут быть определены как с помощью функции find, так и Minerr. Во втором случае можно использовать только Minerr. Так как в наших данных есть довольно заметная погрешность, применяем Minerr.

$$A := 4 \qquad k := 1 \qquad b := 0.5$$

Given

$$Y\_err(t, A, k, b) = 0$$

$$Minerr(A, k, b) = \begin{pmatrix} 5.418 \\ 1.403 \\ 0.222 \end{pmatrix} \qquad \begin{pmatrix} A \\ k \\ b \end{pmatrix} := \begin{pmatrix} 5.418 \\ 1.403 \\ 0.222 \end{pmatrix}$$

Строим график, чтобы убедиться, что экспериментальные данные хорошо ложатся на кривую, параметры которой были найдены функцией Minerr (рис. 8.16).

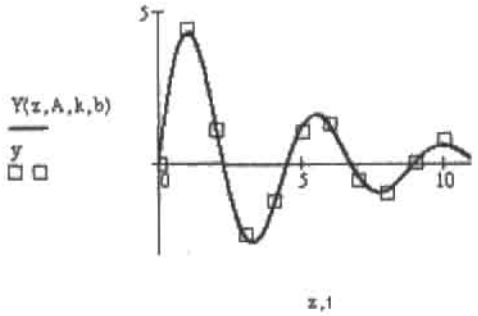


Рис. 8.16. Регрессивная кривая и экспериментальные точки

В Mathcad 12 появилась новая системная переменная ERR. Данная переменная хранит сумму квадратов ошибок решений, найденных Minerr. По значению данной переменной можно судить, насколько приблизительно были найдены корни. Если корни были определены точно, то ERR равна 0.

Глава 9. Решение неравенств

Неравенства — это, пожалуй, самая сложная тема школьной программы по математике. Научиться хорошо их решать очень и очень непросто. До определенной степени решение неравенств схоже с поиском корней уравнений. Однако в случае неравенств дело осложняется тем, что, по сути, необходимо исследовать поведение функции на всей числовой оси, что требует очень хорошего знания математических приемов и развитой логики. Система Mathcad способна помочь в непростом деле решения неравенств. Однако ни в коем случае нельзя полагаться на программу целиком. Неравенства — это слишком тонкий предмет, чтобы можно было требовать многого от линейно мыслящей машины. Решая неравенства, Mathcad нередко ошибается или просто не справляется с задачей. Поэтому, получив ответ, его обязательно нужно проверить по графику. Если же программа окажется в тупике, следует попытаться ей помочь, вручную преобразовав неравенство в более простую форму. Сложные неравенства обычно оказываются Mathcad «не по зубам». В таких случаях вы должны вспомнить, чему вас учили на занятиях, и попытаться решить задачу самостоятельно. Mathcad же может вам при этом помочь, выполняя такие рутинные операции, как упрощение выражений, решение промежуточных уравнений или построение графика. В этой небольшой главе мы попытаемся на примерах разобраться, на что же реально способен Mathcad в области решения неравенств.

Для аналитического решения неравенств в Mathcad используется тот же самый оператор панели **Symbolic** (Символьные), что и для решения уравнений и систем уравнений, — **solve**. Неравенство прописывается в его правом маркере, переменная, относительно которой оно должно быть решено, — в левом. Операторы сравнения можно ввести как с клавиатуры (знаки «<» и «>» можно задать напрямую, знак «≥» — сочетанием Ctrl+0, знак «≤» — сочетанием Ctrl+9), так и с помощью панели **Boolean** (Булевы). Приведем примеры решения нескольких неравенств, выраженных простыми алгебраическими выражениями. С такими неравенствами Mathcad справляется эффективнее всего.

Пример 9.1. Решение неравенств, выраженных полиномами и отношениями полиномов

$$\frac{(x+3)\cdot(5-x)}{2x-5} > 0 \text{ solve}, x \rightarrow \begin{bmatrix} x < -3 \\ \left(\frac{5}{2} < x\right)\cdot(x < 5) \end{bmatrix}$$

Ответ в стандартной форме: $x \in (-\infty; -3) \cup (2{,}5; 5)$.

$$\frac{x^2 \cdot (2x-9) \cdot (x-1)^3}{(x+4)^5 \cdot (2x-6)^4} \le 0 \text{ solve}, x \rightarrow \begin{bmatrix} x < -4 \\ x = 0 \\ (1 \le x) \cdot (x < 3) \\ (3 < x) \cdot \left(x \le \frac{9}{2}\right) \end{bmatrix}$$

Ответ в стандартной форме: $x \in (-\infty; -4) \cup 0 \cup [1; 3) \cup (3; 4{,}5]$.

$$x^2 + x + 1 > 0 \text{ solve}, x \rightarrow x$$

Ответ в стандартной форме: $x \in R$.

$$\frac{1}{a^2 - 4a + 4} > \frac{2}{a^3 - 8} \text{ solve}, a \rightarrow \begin{pmatrix} a < 2 \\ 2 < a \end{pmatrix}$$

Ответ в стандартной форме: $a \ne 2$ или $a \in (-\infty; 2) \cup (2; \infty)$.

Как вы уже, наверное, заметили, Mathcad выдает ответы в несколько отличном от принятого в нашей математике виде. Поэтому зачастую самой трудной частью работы при символьном решении неравенств является интерпретация результата. Тут нужно запомнить несколько правил.

- Ответ оператор **solve** возвращает в виде вектора, содержащего элементарные неравенства. Каждое такое неравенство описывает область, в которой решаемое неравенство справедливо. Если область открытая (то есть одной из ее границ является бесконечность), то задающее ее элементарное неравенство будет иметь вид $x>a$ или $x<a$. В стандартном виде такие области запишутся как $x \in (a; \infty)$ или $x \in (-\infty; a)$. Если область замкнута и ее границам соответствуют значения аргумента a и b, то она будет описана элементарным неравенством вида $(a<x)\cdot(x<b)$. В стандартном виде эта запись будет выглядеть как $x \in (a;b)$.
- Области в векторе ответа будут расположены строго в направлении числовой оси. Поэтому преобразовывать в стандартную форму его можно чисто механически, сохраняя исходный порядок областей. Для объединения обозначений областей в одно выражение используется символ «∪». К примеру, следующий вектор элементарных неравенств

$$\begin{bmatrix} x < 1 \\ (2 < x) \cdot (x < 3) \\ 4 < x \end{bmatrix}$$

в стандартной форме запишется как $x \in (-\infty; 1) \cup (2; 3) \cup (4; \infty)$. Знак умножения, стоящий в решении между элементарными неравенствами, эквивалентен логическому «И». Mathcad же, анализируя логическое выражение, возвращает 1, если оно оказывается истинным, и 0 в случае невыполнения его условия. Так, если оба элементарных неравенства верны, решение исходного неравенства также верно: $1 \cdot 1 = 1$. Если же условие хотя бы одного из них не выполняется, решение неравенства будет ложным: $1 \cdot 0 = 0$, $0 \cdot 1 = 0$.

- Нужно внимательно следить, используются ли в описывающем область элементарном неравенстве операторы «больше» и «меньше» («<» и «>») или же «больше или равно» и «меньше или равно» («≤» и «≥»). В первом случае точку границы не нужно включать в область, во втором — нужно. Если граничная точка входит в область, то в стандартной форме ее записи используется квадратная скобка, если не входит — круглая. Например, неравенство (3<x)·(x≤9) в стандартную форму записи решения неравенства преобразуется как $x \in (3;9]$.
- Если в решение неравенства входит отдельная точка a, то Mathcad использует обозначение x=a (см. второе неравенство в примере 9.1).
- Если неравенство выполняется при любых значениях аргумента, то в качестве ответа программа возвратит сам аргумент (см. третье неравенство в примере 9.1).
- Если неравенство не выполняется при любых значениях аргумента, то Mathcad возвратит сообщение об ошибке: **No solution was found** (Не было найдено решения). Однако полностью полагаться на это сообщение ни в коем случае нельзя. Вполне вероятно, что программа просто не смогла справиться с задачей. В подобных случаях обязательно необходимо проводить дополнительные исследования, например, с помощью графика.
- В выдаваемых ответах оператор **solve** никогда не использует знак «≠». Даже если неравенству не соответствует только одна точка (см. четвертое неравенство в примере 9.1), области решения описываются полностью. Однако в таких случаях все же техничнее давать ответ в форме x≠a.

Объективно говоря, полностью доверять символьному процессору Mathcad можно лишь при решении очень простых неравенств, выраженных алгебраическими полиномами или отношением полиномов. Во всех остальных случаях нужно быть очень внимательным и обязательно проверять решение. Так, программа сделает ошибку практически в любом неравенстве, содержащем корни (см. пример 9.2).

Пример 9.2. Неверное решение неравенств

$$\frac{\sqrt{x-3}}{x-2} > 0 \text{ solve}, x \rightarrow \begin{pmatrix} x < 2 \\ 9 < x \end{pmatrix}$$

Преобразуем ответ в стандартный вид: $x \in (-\infty; 2) \cup (9; \infty)$. Анализируя его, обнаруживаем, что он не до конца верный. Так как в выражении проводится операция извлечения корня из x, последний должен быть больше либо равен нулю. Следовательно, первая область решения найдена некорректно, а верный ответ будет иметь вид $x \in (0; 2) \cup (9; \infty)$. В его правильности можно убедиться, построив график (рис. 9.1).

Аналогичную описанной выше ошибку сделает Mathcad и при попытке решить другое неравенство с корнем:

$$\sqrt{x+61} > x+5 \text{ solve}, x \rightarrow x < 3$$

Ответ в стандартной форме: $x \in (-\infty; 3)$. Полученный результат неверен, так как подкоренное выражение не может быть меньше нуля. Правильный ответ — $x \in (-61; 3)$.

Достаточно неплохо справляется оператор **solve** с показательными и логарифмическими неравенствами. Однако при их решении нужно, как и в случае неравенств с корнями, обязательно делать проверку по графику. Если в качестве ответа система выдаст громоздкое выражение, его следует попытаться упростить с помощью оператора **simplify**.

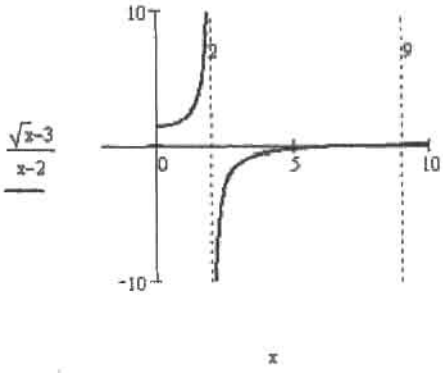


Рис. 9.1. Соответствующая неравенству функция отрицательна на промежутке от 2 до 9

Пример 9.3. Решение показательного неравенства

$$\frac{\left(\frac{1}{3}\right)^{8+x} - 81}{x^2 + 2x + 5} < 0 \quad \left| \begin{array}{l} \text{solve, x} \\ \text{simplify} \end{array} \right. \rightarrow -12 < x$$

Ответ в стандартном виде: $x \in (-12; \infty)$.

Проверяем верность найденного решения с помощью графика (рис. 9.2).

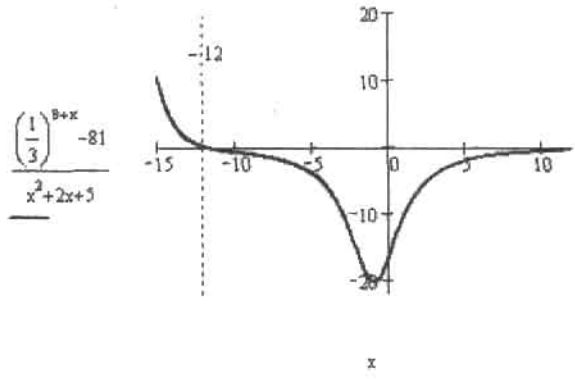


Рис. 9.2. Функция отрицательна, если x>12

Вывод: неравенство было решено верно.

Компьютер лишен интеллекта. Глупо рассчитывать на то, что при решении задачи Mathcad применит хитроумный прием или незаурядный ход. Программа решает простые неравенства стандартными способами — и не больше. Поэтому очень часто пользователь должен «помогать» символьному процессору, приводя задачу к более простому виду. Как упростить неравенство? Во-первых, можно произвести замену переменных. Во-вторых, можно путем алгебраических преобразований привести выражение к более легкому для программы виду. В-третьих, можно производить логарифмирова-

ние, возведение в степень или другие математические операции, не нарушающие условия неравенства. Подобноя «помощь» аналитическому процессору показана в примере 9.4.

Пример 9.4. Решение неравенства с предварительным его упрощением

Пусть перед нами стоит задача решить неравенство следующего вида:

$$x^{\frac{2x-1}{3-x}} > 1$$

Попытка решить его без проведения каких-либо преобразований окажется неудачной (рис. 9.3).

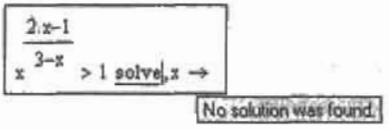


Рис. 9.3. Mathcad не справился с данным неравенством

Главная сложность данного неравенства заключается в том, что x возводится в степень, содержащую x. Сгладить эту трудность можно, прологарифмировав обе части неравенства. Знак в неравенстве при этом не изменится (подумайте, почему).

$$\ln\left(x^{\frac{2\cdot x-1}{3-x}}\right)\left|\begin{array}{l}\text{expand, x}\\ \text{simplify}\end{array}\right.\rightarrow -\ln(x)\cdot\frac{2\cdot x-1}{x-3}\qquad \ln(1)=0$$

С упрощенным неравенством оператор solve справится без каких-либо проблем:

$$\ln(x)\cdot\frac{2\cdot x-1}{3-x}>0\ \text{solve, x}\rightarrow\begin{bmatrix} x<\frac{1}{2}\\ (1<x)\cdot(x<3)\end{bmatrix}$$

Ответ в стандартной форме: $x\in(-\infty;\,0{,}5)\cup(1;\,3)$.

Данный ответ верен не до конца. Нужно учесть, что возводить отрицательное число в произвольную степень или вычислять от него логарифм нельзя. Поэтому x может быть только больше или равен нулю. Исходя из этого, правильный ответ будет иметь вид: $x\in(0;\,0{,}5)\cup(1;\,3)$.

Можно решать с помощью символьного процессора Mathcad и неравенства с параметром, правда, лишь самые простые. Так, даже с элементарным квадратным неравенством $a\cdot x^2+b\cdot x+c>0$, решение которого известно любому школьнику, программа не справляется. Наиболее же слабым местом символьного процессора является тригонометрия. Так, даже самое элементарное неравенство типа $\sin(x)\geq 0$ решено им не будет. Также практически наверняка неравенство не будет решено, если в нем присутствуют какие-то специальные функции. Как поступать в таких случаях, будет показано ниже.

В наиболее простых случаях с помощью оператора solve можно решить и системы неравенств. Для этого, аналогично решению систем уравнений, входящие в систему неравенства нужно объединить в вектор (см. пример 9.5).

Пример 9.5. Решение системы неравенств

$$\begin{pmatrix} \frac{x+4}{2x-3} > 0 \\ \frac{1}{x-1} < 0 \end{pmatrix} \text{solve},x \rightarrow x < -4$$

Ответ в стандартной форме: $x \in (-\infty; -4)$.

Чаще всего оператор solve не справляется с поиском совместного решения системы неравенств. В этой ситуации нужно действовать следующим образом: искать решение каждого неравенства как независимого, а затем находить общие для всех решений области. Продемонстрируем описанный подход на примере.

Пример 9.6. Пошаговое решение системы неравенств

Пусть перед нами стоит задача решить следующую систему неравенств:

$$\left(\frac{2}{3}\right)^{x} \cdot \left(\frac{8}{9}\right)^{-x} > \frac{27}{64}$$

$$2^{x^2-6\cdot x-\frac{7}{2}} < 8\cdot\sqrt{2}$$

Сначала попытаемся возложить всю работу на программу и решить неравенства совместно. При этом будет получен ответ: $x>-1$. Проверка по графику показывает, что это решение абсолютно неверно. Значит, необходимо решать неравенства раздельно, а затем сопоставлять найденные решения.

С первым неравенством в его исходном виде Mathcad не справится. Следовательно, его нужно попытаться упростить. Для этого прологарифмируем обе его части, а затем последовательно используем формулы $\log(a\cdot b)=\log(a)+\log(b)$ и $\log(a\cdot n)=n\cdot\log(a)$. Полученное в результате неравенство будет успешно решено программой:

$$x\cdot\ln\left(\frac{2}{3}\right) - x\cdot\ln\left(\frac{8}{9}\right) > \ln\left(\frac{27}{64}\right) \left|\begin{matrix}\text{solve},x \\ \text{simplify}\end{matrix}\right. \rightarrow x < 3$$

Ответ в стандартной форме: $x \in (-\infty; 3)$.

Второе неравенство системы Mathcad решит без особых сложностей:

$$2^{x^2-6x-\frac{7}{2}} < 8\sqrt{2}\ \text{solve},x \rightarrow \begin{bmatrix} 3 - \left[\frac{1}{\ln(2)}\cdot(13\cdot\ln(2)+\ln(8))\right]^{\frac{1}{2}} < x \\ x < 3 + \left[\frac{1}{\ln(2)}\cdot(13\cdot\ln(2)+\ln(8))\right]^{\frac{1}{2}} \end{bmatrix}$$

Чтобы привести громоздкие выражения, выданные в качестве ответа, к более простому виду, используем оператор simplify:

$$3-\left[\frac{1}{\ln(2)}\cdot(13\cdot\ln(2)+\ln(8))\right]^{\frac{1}{2}}\ \text{simplify}\rightarrow -1 \qquad 3+\left[\frac{1}{\ln(2)}\cdot(13\cdot\ln(2)+\ln(8))\right]^{\frac{1}{2}}\ \text{simplify}\rightarrow 7$$

Итак, решение второго неравенства можно записать как $x\in(-1;7)$.

Если первое неравенство выполняется на промежутке от $-\infty$ до 3, а второе на промежутке от -1 до 7, то, очевидно, им обоим удовлетворяет промежуток от -1 до 3. Итак, окончательный ответ: $x\in(-1;3)$.

Так как решенная система является довольно сложной, нужно обязательно выполнить проверку по графику (рис. 9.4).

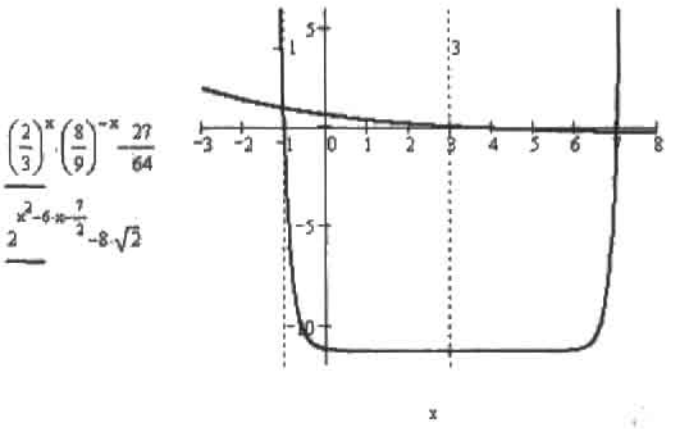


Рис. 9.4. Область решения системы заключена между вертикальными штриховыми линиями

Проверка показывает, что решение было найдено абсолютно верно. Аналогичным образом можно решить подавляющее большинство из встречающихся в задачниках систем неравенств.

В случае неравенств повышенной сложности (например, тригонометрических) Mathcad окажется не на высоте даже при условии максимального упрощения вами задачи. В таких случаях неравенство придется решать самостоятельно. При этом Mathcad сможет оказать существенную помощь, выполняя простые, но рутинные операции вроде приведения тригонометрического выражения к одному аргументу или упрощения выражения. Для примера продемонстрируем, как можно быстро и просто решить довольно трудное тригонометрическое неравенство.

Пример 9.7. Решение тригонометрического неравенства

Пусть стоит задача решить неравенство вида $2\cdot\sin^3(x)-\sin(x)+\sin(3\cdot x)<1$. Попытка решить его напрямую окажется неудачной. Следовательно, необходимо решать его пошагово, используя те же ходы, которые мы бы применяли при его решении на бумаге. Для начала приведем выражение к одному аргументу. Для этого нужно представить $\sin(3x)$ через синус и косинус от x. Выполнить это преобразование позволяет оператор expand.

$$2\cdot\sin(x)^2-\sin(x)+\sin(3\cdot x)\ \text{expand},x\rightarrow 2\cdot\sin(x)^2-2\cdot\sin(x)+4\cdot\sin(x)\cdot\cos(x)^2$$

Полученное выражение содержит как синусы, так и косинусы. Чтобы в дальнейшем можно было его рассматривать как алгебраическое, нужно оставить в нем лишь один вид функций. Очевидно, что проще избавиться от косинусов. Выразив косинус через синус из основного тригонометрического равенства, подставляем полученное выражение в неравенство (оператор substitute), а затем упрощаем его.

$$2\cdot\sin(x)^2-2\cdot\sin(x)+4\cdot\sin(x)\cdot\cos(x)^2 \left|\begin{matrix}\text{substitute},\cos(x)^2=1-\sin(x)^2\\ \text{collect},\sin(x)\end{matrix}\right. \rightarrow -4\cdot\sin(x)^3+2\cdot\sin(x)^2+2\cdot\sin(x)$$

Далее нужно, решив полученное неравенство относительно sin(x), определить, при каких значениях синуса оно выполняется.

$$-4\sin(x)^3+2\sin(x)^2+2\sin(x)-1<0\ \text{solve},\sin(x)\ \rightarrow \begin{bmatrix}\left(\frac{-1}{2}\cdot 2^{\frac{1}{2}}<\sin(x)\right)\cdot\left(\sin(x)<\frac{1}{2}\right)\\ \frac{1}{2}\cdot 2^{\frac{1}{2}}<\sin(x)\end{bmatrix}$$

Итак, решаемое неравенство справедливо, если $\sin(x)\in(-2^{1/2}/2;\ 1/2)\cup(2^{1/2}/2;\ 1)$. Чтобы найти, каким должен быть x, чтобы это условие соблюдалось, проанализируем график синуса на промежутке одного периода (рис. 9.5).

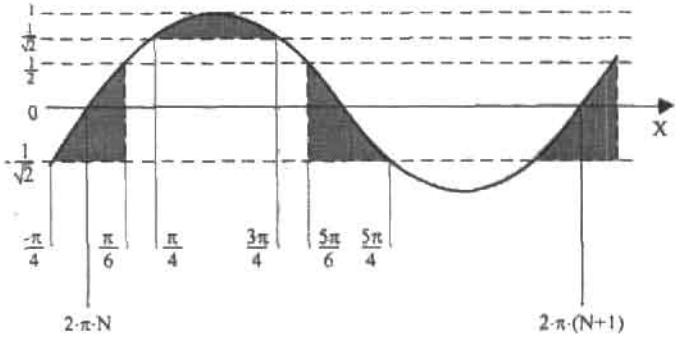


Рис. 9.5. Функция y(x)=sin(x) на промежутке, равном одному периоду

Отметив на оси Y границы областей решения, проведем вспомогательные линии. Заключенные между ними фрагменты кривой содержат точки, удовлетворяющие неравенству. Таких фрагментов на промежутке периода три. Глядя на полученную схему, совсем не сложно найти координаты их границ по X и записать окончательный ответ: $x\in(-\pi/4+2\cdot\pi\cdot N;\ \pi/6+2\cdot\pi\cdot N)\cup(\pi/4+2\cdot\pi\cdot N;\ 3\cdot\pi/4+2\cdot\pi\cdot N)\cup(5\cdot\pi/6+2\cdot\pi\cdot N;\ 5\cdot\pi/4+2\cdot\pi\cdot N)$, $N\in Z$.

Глава 10. Вычисление интегралов

В этой главе рассматриваются особенности проведения такой важнейшей математической операции, как интегрирование. Наряду с производной, интеграл является одним из основных инструментов современной математики. На практике очень часто ставится задача найти функцию или ее значение в точке, если известна ее производная. К подобной задаче сводятся многие физические и химические проблемы. Например, получив из эксперимента уравнение скорости химической реакции, можно с легкостью определить, сколько вещества прореагирует за определенный промежуток времени. Если тело можно описать функцией, то интегрирование позволит найти его объем и площадь поверхности. Подобных примеров можно привести очень и очень много.

Аналогично дифференцированию, интегрирование в Mathcad может быть как численным, так и символьным. При символьном интегрировании система ищет первообразную, при численном — приближенно подсчитывает ограниченную кривой функции площадь. Вычисление определенного интеграла может быть как символьным, так и численным, неопределенного — только символьным. В случае интегрирования численный подход имеет куда большее значение, чем в случае дифференцирования. На это есть две причины. Во-первых, далеко не все функции имеют первообразную, которую можно выразить в элементарных функциях. Во-вторых, определение первообразной — это куда более сложная задача, чем нахождение производной, поэтому Mathcad с ней не всегда справляется. Конечно, аналитический ответ куда информативнее и не содержит погрешности — поэтому сначала всегда нужно стремиться провести интегрирование символьно. И лишь в случае неудачи (или если ответ будет уж очень громоздок) следует обращаться к численным методам.

Помимо обычных интегралов, в Mathcad можно подсчитывать двойные и тройные интегралы, а также несобственные интегралы. В подсчете таких интегралов есть важные особенности, поэтому им мы посвятим отдельные подразделы.

10.1. Нахождение неопределенного интеграла

Панель Calculus (Вычисления) содержит два оператора интегрирования. Первый, Indefinite Integral (Неопределенный интеграл), позволяет определить вид первообразной интегрируемой функции (рис. 10.1). Помимо панели, вводится данный оператор сочетанием Ctrl+I.

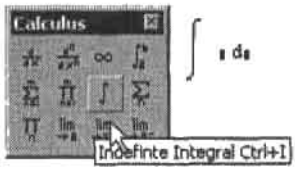


Рис. 10.1. Оператор неопределенного интеграла

Оператор неопределенного интеграла содержит два маркера, которые заполняются полностью в соответствии с принятыми в математике традициями: в левый вводится функция (или имя функции), под знак дифференциала — переменная, по которой должно быть проведено интегрирование. Использовать с оператором неопределенного интеграла можно только оператор символьного вывода «->».

Пример 10.1. Вычисление первообразных

Интегрирование сложной рациональной функции:

$$\int \frac{10x^2 + 8x + 16}{x^2 - 15x - 21}\,dx \to 10\cdot x + 79\cdot\ln\left(x^2 - 15\cdot x - 21\right) - \frac{2822}{309}\cdot 309^{\frac{1}{2}}\cdot \operatorname{atanh}\left[\frac{1}{309}\cdot(2\cdot x - 15)\cdot 309^{\frac{1}{2}}\right]$$

Интегрирование функции с иррациональностью:

$$\int \frac{\sqrt{x}}{x^2 + 1}\,dx \text{ factor} \to \frac{-1}{2}\cdot 2^{\frac{1}{2}}\cdot\left[\operatorname{atan}\left(\frac{2^{\frac{1}{2}}\cdot x^{\frac{1}{2}}}{x - 1}\right) + \ln\left[\frac{x + 2^{\frac{1}{2}}\cdot x^{\frac{1}{2}} + 1}{\left(x^2 + 1\right)^{\frac{1}{2}}}\right]\right]$$

Интегрирование тригонометрической функции с неопределенными коэффициентами:

$$\int e^{\alpha\cdot x}\cdot\cos(x)^2\,dx \text{ simplify} \to \frac{1}{2}\cdot\exp(\alpha\cdot x)\cdot\frac{\alpha^2\cdot\cos(2\cdot x) + 2\cdot\sin(2\cdot x)\cdot\alpha + \alpha^2 + 4}{\left(\alpha^2 + 4\right)\cdot\alpha}$$

Зачастую результат интегрирования представляет собой громоздкое выражение. В этом случае его стоит упрощать. Наиболее универсальный инструмент, который для этого используется, — это, конечно, оператор Simplify (Упростить). Однако иногда выражение можно сделать проще, приведя подобные слагаемые (оператор Collect), разложив степени (оператор Expand) или приведя дроби к общему знаменателю (оператор Factor). Чтобы задействовать нужный символьный оператор, следует выделить выражение интеграла и нажать соответствующую кнопку на панели Symbolic (Символьные). Применять к результату интегрирования можно и сразу несколько символьных операторов.

Строго говоря, неопределенный интеграл — это множество всех первообразных функции. Неопределенный интеграл отличается от выражения первообразной наличием произвольной постоянной (обычно она обозначается буквой C). Однако в выражения, возвращаемые оператором неопределенного интегрирования Mathcad, произвольная постоянная не входит, так как в случае большинства практических задач C — это довольно бесполезный балласт. Однако порой произвольную постоянную нужно учиты-

вать (например, если вы решаете дифференциальное уравнение прямым интегрированием). При этом C следует добавлять в выражение ответа самостоятельно.

Находить первообразные Mathcad умеет довольно неплохо. 99 % встречающихся на практике задач будет успешно решено изучаемой программой (естественно, при условии, что интегрируемая функция имеет первообразную в элементарных функциях). В отличие от решения уравнений, одинаково успешно можно проинтегрировать рациональную функцию, иррациональную функцию, функцию с логарифмом или тригонометрическую функцию. Находить первообразную можно и для функций с буквенными коэффициентами. В общем, без всяких преувеличений, возможности Mathcad в области аналитического интегрирования заслуживают похвалы. Однако это не значит, что можно навсегда забыть про утомительный поиск нужной первообразной в толстых математических справочниках. Увы, но в ряде случаев с них придется сдувать пыль.

- В Mathcad встроена обширная библиотека неопределенных интегралов. Однако она все же уступает по объему лучшим сборникам формул. Также вполне вероятна такая ситуация, что программа просто не сможет соотнести функцию в том виде, в котором вы ее предоставите, с имеющейся в библиотеке формулой (все-таки это довольно интеллектуальная задача). Поэтому, если первообразная не будет найдена, попробуйте тождественными преобразованиями и заменой переменных упростить вид функции. Если же это невозможно — доставайте с полки справочник.
- Результат интегрирования в Mathcad может представлять собой огромное и не поддающееся упрощению выражение. Также в него могут входить функции, мало что говорящие обычному инженеру или физику (особенно символьный процессор «любит» обратные гиперболические функции). Формулы же в справочниках обычно оптимизированы и содержат лишь базовые функции. Во многих случаях упрощение может быть проведено и в среде Mathcad путем замены сложных для восприятия функций на аналитические выражения для них.
- Mathcad совершенно не умеет интегрировать функции, первообразные для которых представляют собой рекуррентные соотношения, ряды, произведения.

Таким образом, если окажется, что Mathcad не сможет найти первообразную или выданное программой выражение слишком сложно, не опускайте сразу руки. Очень даже вероятно, что старый добрый справочник поможет вам лучше современной программы. Если же возиться со справочником не хочется или необходимой формулы в нем не имеется, можно попробовать «подсказать» Mathcad. Для этого можно выполнить замену переменных (например, чтобы избавиться от иррациональности) или, если ответ содержит сложные для восприятия функции, подставить вместо них их аналитические выражения. В примере 10.2 показаны наиболее типичные случаи неэффективного нахождения Mathcad первообразной, а также то, как в подобных случаях нужно действовать.

Пример 10.2. Случаи неэффективного нахождения первообразной

Следующий интеграл программа не смогла подсчитать и возвратила в качестве ответа исходное выражение без каких-либо изменений:

$$\int \ln\left(x+\sqrt{x^2-a^2}\right)dx \rightarrow \int \ln\left[x+\left(x^2-a^2\right)^{\frac{1}{2}}\right]dx$$

Значение данного интеграла находим в обычном справочнике довольно небольшого объема [16]:

$$\int \ln\left(x+\sqrt{x^2-a^2}\right)dx = x\cdot\ln\left(x+\sqrt{x^2-a^2}\right)-\sqrt{x^2-a^2}+C$$

Если справочника с интегралами под рукой нет или если первообразной для нужной функции в нем не имеется, можно попробовать «помочь» символьному процессору, заменой переменных приведя подынтегральное выражение к более простой форме. В нашем случае замену необходимо осуществить так, чтобы из выражения исчезла иррациональность. Для этого введем новую переменную t:

$$t = x+\sqrt{x^2-a^2}$$

Выразим переменную x через переменную t:

$$x+\sqrt{x^2-a^2}=t \text{ solve},x \rightarrow \frac{1}{2}\cdot\frac{a^2+t^2}{t}$$

Найдем связь между dx и dt. Очевидно, что $dx=d(x(t))=x'(t)\cdot dt$. Определим $x'(t)$:

$$\frac{d}{dt}\frac{1}{2}\cdot\frac{a^2+t^2}{t} \text{ simplify} \rightarrow \frac{-1}{2}\cdot\frac{a^2-t^2}{t^2}$$

Заменяем соответствующее выражение от x на t, а dx на $x'(t)\cdot dt$. Находим первообразную:

$$\int \frac{-1}{2}\cdot\frac{\left(a^2-t^2\right)}{t^2}\cdot\ln(t)\,dt \text{ factor} \rightarrow \frac{1}{2}\cdot\frac{a^2\cdot\ln(t)+a^2+t^2\cdot\ln(t)-t^2}{t}$$

Выполняем обратную подстановку, возвращаясь к переменной x:

$$\frac{1}{2}\cdot\frac{a^2\cdot\ln(t)+a^2+t^2\cdot\ln(t)-t^2}{t} \text{ substitute},t=x+\sqrt{x^2-a^2} \rightarrow$$

Полученное в результате подстановки выражение мы не приводим, так как оно слишком громоздко для книжной страницы. Однако буквально несколькими действиями его можно преобразовать к той простой форме, которая используется в справочнике.

Для следующего интеграла Mathcad результат находит — но в виде чрезвычайно громоздкого и сложного выражения. Упростить его посредством оператора simplify не получится. Однако, заглянув в справочник, мы обнаружим для данной функции простую и удобную первообразную.

$$\int \frac{\sin(x)}{\cos(x)^n}dx \rightarrow \frac{\frac{1}{n-1}-\frac{1}{n-1}\cdot\tan\left(\frac{1}{2}\cdot x\right)^2}{1+\tan\left(\frac{1}{2}\cdot x\right)^2}\cdot\left[\frac{-\left(-1+\tan\left(\frac{1}{2}\cdot x\right)^2\right)}{1+\tan\left(\frac{1}{2}\cdot x\right)^2}\right]^{-n}$$

Формула из справочника:

$$\int \frac{\sin(x)}{\cos(x)^n}\,dx = \frac{1}{(n-1)\cdot\cos(x)^{n-1}} + C$$

Впрочем, в случае приведенного интеграла результат все же можно упростить. Для этого нужно правильно сделать замену. Очевидно, что следует перейти от половинного аргумента к целому, от «тяжелого» тангенса — к простым синусу и косинусу. Для этого используем известную «школьную» формулу $\operatorname{tg}(\alpha/2)=(1-\cos(\alpha))/\sin(\alpha)$.

$$\int \frac{\sin(x)}{\cos(x)^n}\,dx \;\left|\begin{array}{l}\text{substitute}, \tan\left(\frac{x}{2}\right)=\frac{1-\cos(x)}{\sin(x)}\\ \text{simplify}\end{array}\right. \rightarrow \frac{\cos(x)^{1-n}}{n-1}$$

При вычислении следующего интеграла первообразная выражается с использованием функции atanh — гиперболического арктангенса (более корректное название этой функции — ареатангенс). Функция эта достаточно экзотическая, так что ее наличие в первообразной мало информативно для специалиста нематематического профиля.

$$\int \frac{1}{x^3\cdot\sqrt{1-x^2}}\,dx \rightarrow \frac{-1}{2\cdot x^2}\cdot\left(1-x^2\right)^{\frac{1}{2}} - \frac{1}{2}\cdot\operatorname{atanh}\left[\frac{1}{\left(1-x^2\right)^{\frac{1}{2}}}\right]$$

Ответ в более простой форме можно найти в справочнике. Однако можно попытаться упростить первообразную и самостоятельно, заменив ареатангенс на явное выражение из более элементарных функций. Для этого можно опять же порыться в справочнике. Но лучше справиться с проблемой своими силами. Так как ареатангенс — это функция, обратная гиперболическому тангенсу, то ее можно найти, выразив из задающего его выражения аргумент (проще говоря, найдя зависимость x(y) из известной зависимости y(x)). Легче всего это сделать с помощью оператора solve, решив соответствующее уравнение:

$$\frac{e^y-e^{-y}}{e^y+e^{-y}}=z \;\left|\begin{array}{l}\text{solve}, y\\ \text{simplify}\end{array}\right. \rightarrow \begin{bmatrix} -\ln\left[\frac{1}{1+z}\cdot[-(1+z)\cdot(-1+z)]^{\frac{1}{2}}\right] \\ -\ln\left[\frac{-1}{1+z}\cdot[-(1+z)\cdot(-1+z)]^{\frac{1}{2}}\right] \end{bmatrix}$$

Мы получили два решения. Но какое из них является выражением ареатангенса? Это очень просто определить, зная, что аргумент ареатангенса (то есть гиперболический тангенс) изменяется в интервале от −1 до 1. При этом условии второй из полученных корней является некорректным,

так как логарифмируемое выражение в нем является комплексным для любых z. Наоборот, логарифмируемое выражение в первом корне всегда действительное и положительное — следовательно, нужно использовать именно его.

Путем элементарных преобразований выражение ареатангенса можно упростить:

$$\operatorname{atanh}(z) = \frac{1}{2}\cdot\ln\left(\frac{1+z}{1-z}\right)$$

Подставив полученное выражение вместо функции atanh в выражение первообразной, получим следующую довольно простую формулу:

$$\int \frac{1}{x^3\cdot\sqrt{1-x^2}}\,dx = \frac{-1}{2\cdot x^2}\cdot\left(1-x^2\right)^{\frac{1}{2}} - \frac{1}{4}\cdot\ln\left(\frac{\sqrt{1-x^2}+1}{\sqrt{1-x^2}-1}\right)$$

Убедиться, что мы не сделали ошибки, можно, продифференцировав полученную первообразную.

Подобно тому, как мы избавились от ареасинуса, из результатов символьного интегрирования можно убирать и другие «сложные» функции. Исключение составляют так называемые интегральные функции, которые не выражаются через сочетания функций элементарных. О данном типе функций мы поговорим чуть ниже.

В качестве последнего примера, демонстрирующего то, что Mathcad заменяет справочник и голову далеко не всегда, мы приведем интеграл, первообразная которого выражается через сумму общего вида. Mathcad рассчитать такой интеграл не может:

$$\int \frac{x^n}{x+1}\,dx \rightarrow \int \frac{x^n}{x+1}\,dx$$

В справочнике же первообразная для данного интеграла приводится:

$$\int \frac{x^n}{x+1}\,dx = \sum_{k=0}^{n-1}\left[\frac{(-1)^k\cdot x^{n-k}}{n-k} + (-1)^n\cdot\ln(|x+1|)\right] + C$$

Первообразные для функций подобного типа нужно сразу брать из справочника. Mathcad их не найдет даже при наличии помощи с вашей стороны.

Для всех ли функций можно найти первообразную? В математическом анализе доказывается, что для любой непрерывной функции существует первообразная. Однако ее далеко не всегда можно выразить как алгебраическое сочетание элементарных функций — показательной, логарифма, косинуса, синуса и т. д. Если первообразная не выражается через элементарные функции, соответствующий интеграл называется «неберущимся».

Какой результат будет получен, если попытаться найти в Mathcad неберущийся интеграл? Тут возможны два варианта. Во-первых, система может возвратить исходное выражение без каких-либо изменений. Во-вторых, ответ может быть выражен через подходящие интегральные функции.

Интегральные функции появились задолго до Mathcad и компьютеров. Они были введены для описания наиболее важных неберущихся интегралов. Существовали таблицы, в которых приводились значения интегральных функций для различных величин аргумента. В случае сложного неберущегося интеграла его сводили к сочетанию интегралов, которым соответствуют интегральные функции. Точно так же действует и Mathcad.

Для примера приведем несколько интегралов, первообразные для которых выражаются через интегральные функции. Описание данных интегральных функций, а также список остальных используемых символьным процессором Mathcad функций можно найти в статье Special Functions and Syntax Used in Symbolic Results справочной системы программы.

Пример 10.3. Вычисление неберущихся интегралов

Интегральные синус (Si) и косинус (Ci):

$$\int \frac{\sin(x)}{x}\,dx \rightarrow Si(x) \qquad \int \frac{\cos(x)}{x}\,dx \rightarrow Ci(x)$$

Интегральная показательная функция (Ei) и интеграл вероятности (erf):

$$\int \frac{e^x}{x}\,dx \rightarrow -Ei(1,-x) \qquad \frac{2}{\sqrt{\pi}}\cdot\int e^{-x^2}\,dx \rightarrow erf(x)$$

Интегралы Френеля (синус – FresnelS, косинус – FresnelC):

$$\int \sin\left(x^2\right)dx \rightarrow \frac{1}{2}\cdot 2^{\frac{1}{2}}\cdot\pi^{\frac{1}{2}}\cdot FresnelS\left(\frac{2^{\frac{1}{2}}}{\pi^{\frac{1}{2}}}\cdot x\right) \qquad \int \cos\left(x^2\right)dx \rightarrow \frac{1}{2}\cdot 2^{\frac{1}{2}}\cdot\pi^{\frac{1}{2}}\cdot FresnelC\left(\frac{2^{\frac{1}{2}}}{\pi^{\frac{1}{2}}}\cdot x\right)$$

Интегральные функции «знает» только символьный процессор Mathcad. В численных расчетах их использовать нельзя. Но что делать, если нужно определить значение первообразной в точке или найти ее нули, построить график ее функции? Проще всего можно найти величину интегральной функции в точке. Для этого следует присвоить аргументу нужное значение и задействовать оператор Float панели Symbolic (Символьные). Сложнее определить нули первообразной, в выражение которой входят интегральные функции, а также построить ее график. Чтобы решить эти задачи, следует заменить интегральные функции соответствующими им явными выражениями (найти эти выражения можно в справочной системе Mathcad, а также в сборниках математических формул). Обычно интегральной функции соответствует определенный интеграл с неявным верхним пределом, подсчитать который можно численно. Реже интегральные функции выражают через бесконечные ряды. В этом случае создать функцию, с нужной точностью приближающую первообразную, можно, заменив бесконечный ряд конечным. Количество членов в этом ряду должно зависеть от точности, с которой первообразную следует приблизить. Найти подходящее количество членов можно,

и не обращаясь к специальным формулам. Для этого следует последовательно увеличивать количество членов ряда на 1 и сравнивать результаты, полученные при прошлом и данном вычислениях. Показателем того, что нужная точность достигнута, будет то, что результаты окажутся равными вплоть до последнего интересующего знака.

Пример 10.4. Найти первообразную для функции e^{-x}/x. Определить с точностью до 5 знаков мантиссы: a) значение функции первообразной в точке x=1; b) при каких x первообразная равна –1. Построить график функции первообразной

Искомый интеграл является неберущимся. Первообразная для него выражается с помощью интегральной показательной функции Ei:

$$\int \frac{e^{-x}}{x}\,dx \rightarrow -Ei(1,x)$$

Найдем приближенное значение Ei в точке x=1 с точностью до пяти знаков:

$$-Ei(1,1)\ \text{float},5 \rightarrow -.21938$$

Чтобы решить остальные поставленные в задаче проблемы, нужно создать аппроксимирующую Ei функцию. Для этого необходимо знать соответствующее Ei явное выражение. Найти его можно в справочной системе Mathcad:

$$Ei(1,x) = -\gamma - \ln(x) - \sum_{n=1}^{\infty} \frac{(-1)^n \cdot x^n}{n \cdot n!}$$

Постоянная γ в данном выражении — это константа Эйлера. Вычислить ее приближенное значение можно с использованием оператора float:

$$\gamma\ \text{float},5 \rightarrow .57722$$

Описанным выше способом определяем, что точности в пять знаков мантиссы можно достичь, заменив бесконечный ряд в выражении для Ei на ряд из восьми членов. Тогда функция, приближающая интегральную показательную функцию, будет иметь вид:

$$EI(x) := -0.57722 - \ln(x) - \sum_{n=1}^{8} \frac{(-1)^n \cdot x^n}{n \cdot n!}$$

Данная функция позволяет вычислять приближенные значения Ei столь же эффективно, что и оператор float:

$$-EI(1) = -0.21938$$

Чтобы найти, при какой величине аргумента первообразная принимает значение –1, численно решаем соответствующее уравнение, применяя функцию find. Так как мы должны получить ответ с точностью до пяти знаков, уменьшаем значение TOL до 10^{-6}:

$$f(z) := EI(z) - 1$$

$$TOL := 10^{-5} \qquad z := 1$$

$$root(f(z), z) = 0.264736 \qquad EI(0.264736) = 1$$

Строим график первообразной на промежутке от 0 до 2 (рис. 10.2).

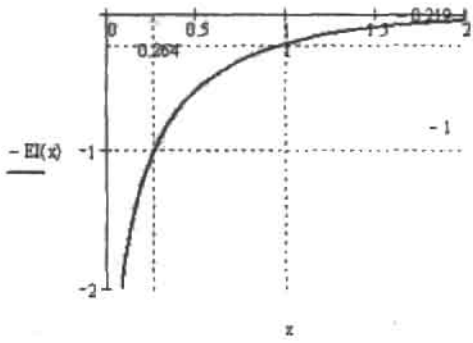


Рис. 10.2. График первообразной функции e^x/x

Объективно говоря, в случае численных расчетов интегральные функции ничем не отличаются от функций элементарных. Для подсчета как первых, так и вторых применяются приближенные методы вроде использования аппроксимирующего ряда или численного интегрирования. Вообще, с приходом эры компьютеров разделение функций на элементарные и сложные потеряло смысл. Для машины все функции равны — и вычисление синуса обычного ничуть не проще, чем вычисление синуса интегрального. Разница между сложными и элементарными функциями имеется лишь при проведении расчетов аналитически.

10.2. Аналитическое вычисление определенного интеграла

Помимо нахождения неопределенного интеграла, в Mathcad существует возможность вычисления и определенного интеграла. Для этого следует использовать специальный оператор Definite Integral (Shift+7) панели Calculus (Вычисления):

$$\int_{\blacksquare}^{\blacksquare} \blacksquare \, d\blacksquare$$

Оператор содержит четыре маркера, заполняются которые в полном соответствии с принятой в математике формой.

В отличие от неопределенного интеграла, определенный может быть вычислен как аналитически, так и численно. Естественно, символьный метод точнее численного, выдаваемый им аналитический ответ информативнее десятичной дроби численного результата, однако он применим в случае далеко не всех функций. Используемые же в Mathcad численные методы практически универсальны — но им присущи существенные недостатки по сравнению с аналитическим подходом. Поэтому численно подсчитывать определенный интеграл стоит лишь, если найти его значение не сможет аналитический процессор. Даже если первообразная выдается в виде громоздкого выражения, лучше пересчитать его в десятичную дробь с помощью оператора float или оператора численного вывода «=», чем проводить интегрирование численно (точность при этом будет выше, а также уменьшится время расчета (что важно в случае кратных интегралов)).

В этом разделе мы обсудим особенности символьного вычисления определенного интеграла. Применению численных методов будет посвящен следующий раздел.

Чтобы вычислить определенный интеграл аналитически, необходимо заполнить все маркеры соответствующего оператора, а затем ввести оператор символьного вывода «->». В случае аналитического интегрирования пределы могут быть как числовыми, так и буквенными или даже быть представленными целыми выражениями. Кроме того, можно использовать символ бесконечности (Ctrl+Shift+Z) (об особенностях вычисления несобственных интегралов мы поговорим чуть ниже).

Если выражение ответа, возвращенное оператором определенного интеграла, слишком громоздкое, его следует попытаться упростить с помощью оператора simplify. В отдельных случаях для упрощения нужно задействовать операторы factor, expand и collect.

Пример 10.5. Аналитическое вычисление определенных интегралов

Нахождение определенных интегралов с численными пределами:

$$\int_0^{\pi} \sin(x)\cdot\cos\left(\frac{x}{2}\right) dx \rightarrow \frac{4}{3} \qquad \int_{\frac{\pi}{2}}^{\pi} \frac{1}{1+\cos(x)^2} dx \rightarrow \frac{1}{4}\cdot 2^{\frac{1}{2}}\cdot\pi$$

Вычисление определенного интеграла с неявно заданными пределами. Чтобы получить ответ в компактной форме, возвращенное оператором интегрирования выражение следует упростить с помощью оператора simplify:

$$\int_{\alpha-\beta}^{\alpha+\beta} \frac{1}{1+\sin(x)} dx \text{ simplify } \rightarrow 2\cdot\frac{\sin(\beta)}{\cos(\beta)+\sin(\alpha)}$$

При вычислении определенного интеграла всегда следует помнить, что все числа и функции Mathcad рассматривает в комплексной области, поэтому, неправильно задав пределы, можно получить комплексное значение. Например:

$$\int_{-1}^{1} \frac{e^{\sqrt{x}}}{\sqrt{x}} dx \rightarrow 2\cdot\exp(1) - 2\cdot\exp(i) = 4.356 - 1.683i$$

Если выражение результата получается слишком сложным или если оно содержит функции, то его нужно пересчитать в десятичную дробь. Для этого можно или задействовать оператор float, или ввести после ответа оператор «=»:

$$\int_{-1}^{1} \frac{1}{\left(x^2+9\right)\cdot\sqrt{x^2+4}} dx \rightarrow \frac{2}{15}\cdot\text{atanh}\left(\frac{1}{3}\right)\cdot 5^{\frac{1}{2}} = 0.103$$

Аналитически вычисляя определенный интеграл, Mathcad использует теорему Ньютона–Лейбница. То есть программа находит функцию первообразной F(x), затем вычисляет разность F(b)–F(a), где a и b — пределы интегрирования, после чего упрощает

полученное выражение. Это означает, что все особенности нахождения неопределенного интеграла, которые мы обсудили выше, можно перенести на случай аналитического вычисления определенного интеграла. Например, если символьный процессор не находит значения интеграла, нужно ему «помочь», выполнив упрощающую подстановку. Также в результат могут входить интегральные функции, оперировать с которыми нужно точно так же, как при нахождении неопределенного интеграла.

Пример 10.6. Вычисление определенного интеграла от функции, первообразная для которой не может быть найдена Mathcad в замкнутой форме

Пусть перед нами стоит задача аналитически рассчитать следующий определенный интеграл:

$$\int_0^2 \frac{\sqrt{x}}{\sqrt{8-x^3}}\,dx$$

Символьный процессор выдает в качестве результата интегрирования следующее громоздкое выражение (которое, правда, может быть пересчитано в десятичную дробь оператором float):

$$-3^{\frac{1}{4}}\cdot\frac{\left(3^{\frac{1}{2}}-1\right)\cdot\text{EllipticK}\left(\frac{1}{4}\cdot 2^{\frac{1}{2}}\cdot 3^{\frac{1}{2}}-\frac{1}{4}\cdot 2^{\frac{1}{2}}\right)-\text{EllipticPi}\left(\frac{1}{2}-\frac{1}{3}\cdot 3^{\frac{1}{2}},\frac{1}{4}\cdot 2^{\frac{1}{2}}\cdot 3^{\frac{1}{2}}-\frac{1}{4}\cdot 2^{\frac{1}{2}}\right)}{-3+2\cdot 3^{\frac{1}{2}}}\ \text{float},6\rightarrow 1.04721$$

В полученном ответе присутствуют практически неописанные в справочной системе Mathcad интегральные функции EllipticK и EllipticPi. Чтобы узнать, что это за функции, нужно обратиться к документации Maple (разработчик Mathcad компания Mathsoft не стала создавать свой символьный процессор, а купила процессор системы Maple). Здесь мы прочитаем, что EllipticK и EllipticPi — это полные эллиптические интегралы второго и соответственно третьего рода.

Чтобы преобразовать ответ в более понятную для «простого смертного» форму, можно функции EllipticK и EllipticPi заменить явными выражениями полных эллиптических интегралов второго и третьего рода в форме Лежандра:

$$K(k):=\int_0^{\frac{\pi}{2}}\frac{1}{\sqrt{1-k^2\cdot\sin(\psi)^2}}\,d\psi \qquad Pi(h,k):=\int_0^{\frac{\pi}{2}}\frac{1}{\left(1-h\cdot\sin(\psi)^2\right)\sqrt{1-k^2\cdot\sin(\psi)^2}}\,d\psi$$

$$-3^{\frac{1}{4}}\cdot\frac{\left(3^{\frac{1}{2}}-1\right)\cdot K\left(\frac{1}{4}\cdot 2^{\frac{1}{2}}\cdot 3^{\frac{1}{2}}-\frac{1}{4}\cdot 2^{\frac{1}{2}}\right)-Pi\left(\frac{1}{2}-\frac{1}{3}\cdot 3^{\frac{1}{2}},\frac{1}{4}\cdot 2^{\frac{1}{2}}\cdot 3^{\frac{1}{2}}-\frac{1}{4}\cdot 2^{\frac{1}{2}}\right)}{-3+2\cdot 3^{\frac{1}{2}}}=1.047$$

Вполне вероятно, то, что ответ был выражен Mathcad в незамкнутой форме с использованием эллиптических интегралов, связано с тем, что интегрируемая функция имеет слишком сложный вид. Стоит попробовать выполнить такую замену переменных, которая бы упростила ее. Введем переменную t, связанную с x следующим соотношением:

$$\sqrt{8 - x^3} = t$$

Из этого равенства, ограничившись действительным решением, находим x:

$$x = \left(8 - t^2\right)^{\frac{1}{3}}$$

Так как dx=d(x(t))=x'(t)·dt, то нам нужно найти выражение производной x(t) по t:

$$\frac{d}{dt}\left(8 - t^2\right)^{\frac{1}{3}} \rightarrow \frac{-2}{3\cdot\left(8 - t^2\right)^{\frac{2}{3}}}\cdot t$$

Заменяем в исходном выражении x на t, умножаем его на производную x'(t), а затем производим упрощение:

$$\frac{\frac{-2}{3\cdot\left(8 - t^2\right)^{\frac{2}{3}}}\cdot t\cdot\sqrt{\left(8 - t^2\right)^{\frac{1}{3}}}}{t}\ \text{simplify} \rightarrow \frac{-2}{3\cdot\left(8 - t^2\right)^{\frac{1}{2}}}$$

В связи с тем что была произведена замена переменных, нужно пересчитать величины пределов интегрирования. Мы должны определить, какие значения принимает t, если x равен 0 и 2. Для этого присвоим выражению, связывающему x с t, значения пределов и решим аналитически полученные уравнения:

$$\left(8 - t^2\right)^{\frac{1}{3}} = 0\ \text{solve}, t \rightarrow \begin{pmatrix} 2\cdot 2^{\frac{1}{2}} \\ -2\cdot 2^{\frac{1}{2}} \end{pmatrix} \qquad \left(8 - t^2\right)^{\frac{1}{3}} = 2\ \text{solve}, t \rightarrow 0$$

Для случая x=0 мы получили два значения t. Какое из них должно быть использовано в качестве нижнего предела интегрирования? Это очень легко определить. Нижний предел интегрирования должен быть меньше верхнего. Для верхнего же предела было получено значение 0. Следовательно, использовать нужно отрицательное решение.

Упрощенная в результате замены переменных функция будет с легкостью проинтегрирована Mathcad:

$$\int_{-2\cdot\sqrt{2}}^{0} \frac{-2}{3\cdot\left(8 - t^2\right)^{\frac{1}{2}}}\,dt \rightarrow \frac{-1}{3}\cdot\pi \qquad \frac{-1}{3}\cdot\pi = -1.047$$

Убедиться в том, что задача была решена верно, можно, проведя интегрирование численно.

Приведенный пример показывает, как важно, используя Mathcad, не забывать математику. Ваше активное участие в вычислениях может помочь получить результат в тех случаях, в которых символьный процессор оказывается недостаточно интеллектуальным.

Довольно тонкая особенность связана с интегрированием функций, зависящих от параметра. Дело в том, что то, будет ли найдено выражение результата и, если да, то в какой форме, зачастую определяется тем, какие значения может принимать параметр. Например, следующий интеграл при $0<|k|<1$ является неберущимся. Если же $|k|>1$, то его значение можно найти в замкнутой форме:

$$\int_1^2 \frac{1}{\sqrt{\left(1-x^2\right)\left(1-k^2\cdot x^2\right)}}\,dx$$

Подобных примеров можно привести очень много. Поэтому всегда, когда вы проводите интегрирование функции с параметром, указывайте, какие значения он принимает. Сделать это можно с помощью оператора **assume** (Принимает) панели **Symbolic** (Символьные). Информация о параметре указывается в его правом маркере следующими способами.

- Если параметр *a* принимает действительные значения (по умолчанию все неизвестные воспринимаются Mathcad как комплексные величины), то в правом маркере assume нужно набрать: «*a=real*». В качестве знака равенства следует использовать логическое равенство (Bold Equal — Ctrl+=). Модификатор real вводится с панели Modifiers (Модификаторы), которая открывается нажатием одноименной кнопки панели Symbolic.
- Если параметр *a* всегда больше (меньше) действительного числа *b*, то в правом маркере оператора assume нужно набрать: «*a>b*» («*a<b*»).
- Если параметр *a* принимает значения в действительной области от *b* до *c*, то в правом маркере оператора assume указываем: «*a=RealRange*(*b*,*c*)». Здесь знак равенства — логическое равенство. Модификатор RealRange вводится нажатием соответствующей кнопки панели Modifiers.

Приведем несколько примеров, показывающих, как важно в случае интегрирования функции с параметром указывать область изменения параметра.

Пример 10.7. Интегрирование функций с параметрами

При вычислении следующего интеграла без задания области изменения параметра a выражение результата получается громоздким и сложным для интерпретации:

$$\int_0^{\pi} \frac{1}{a^2-\cos(x)^2}\,dx \rightarrow \frac{1}{2}\cdot\pi\cdot\frac{\operatorname{csgn}\left[\left(\overline{a+1}\right)\cdot\left[(a+1)\cdot(a-1)\right]^{\frac{1}{2}}\right]+\operatorname{csgn}\left[\left(\overline{a-1}\right)\cdot\left[(a+1)\cdot(a-1)\right]^{\frac{1}{2}}\right]}{a\cdot\left(a^2-1\right)^{\frac{1}{2}}}$$

Но если нам известно, что параметр a всегда больше 1, ответ можно получить в куда более простой форме:

$$\int_0^{\pi} \frac{1}{a^2 - \cos(x)^2}\,dx \text{ assume}, a > 1 \rightarrow \frac{\pi}{a \cdot \left(a^2 - 1\right)^{\frac{1}{2}}}$$

Еще более простой результат будет получен, если сообщить системе, что параметр a локализован в промежутке от −1 до 1:

$$\int_0^{\pi} \frac{1}{a^2 - \cos(x)^2}\,dx \text{ assume}, a = \text{RealRange}(-1, 1) \rightarrow 0$$

Одновременно можно указывать области изменения сразу нескольких параметров подынтегральной функции. Примером может быть интеграл, задающий знаменитую B-функцию Эйлера. Если не указать, какие значения принимают параметры, Mathcad данную функцию не распознает:

$$\int_0^1 x^{\alpha-1} \cdot (1-x)^{\beta-1}\,dx \text{ assume}, \alpha > 0, \beta > 0 \rightarrow \text{Beta}(\alpha, \beta)$$

Наиболее чувствительны к тому, как изменяется параметр подынтегральной функции, несобственные интегралы. Следующий интеграл, являющийся разновидностью интеграла Эйлера, будет подсчитан, лишь если указать, что параметр α принимает значения от 0 до 1:

$$\int_0^{\infty} \frac{x^{\alpha-1}}{\beta + x}\,dx \text{ assume}, \alpha = \text{RealRange}(0, 1) \rightarrow \frac{\pi}{\sin(\pi \cdot \alpha) \cdot \beta^{-\alpha+1}}$$

Если от функции с параметром вычисляется неопределенный интеграл, то указывать область изменения параметра не нужно. Это либо никак не повлияет на результат, либо будет выдано сообщение об ошибке: No symbolic result was found (Не было найдено символьного результата).

В теореме Ньютона–Лейбница есть одно очень важное условие, которое символьным процессором Mathcad не всегда учитывается. Чтобы найти определенный интеграл для функции f(x) на промежутке [a, b] как разность F(b)–F(a), первообразная F(x) должна быть непрерывной на этом промежутке. Каким бы это ни показалось странным, но области определения функции f(x) и ее первообразной F(x) могут не совпадать. Например, f(x)=1/x определена на всей числовой оси, исключая точку 0. Ее же первообразная F(x)=ln(x) принимает действительные значения только при положительных x. Первообразная для f(x)=ctg(x) F(x)=ln(sin(x)) не определена на промежутке от π до 2·π – подобных примеров можно привести еще очень много. Прежде чем использовать теорему Ньютона–Лейбница, следует проверить, существует ли первообразная на всем промежутке интегрирования. Если этого не сделать, вероятность ошибки очень высока. Увы, но иногда Mathcad свойственно делать подобные ошибки. Рассмотрим конкретный пример. Подумаем, какое значение должен принимать следующий интеграл:

$$\int_{-1}^{1} \frac{1}{\sqrt[3]{x}}\,dx$$

Так как участки кривой симметричны относительно точки разрыва в x=0, но противоположны по знаку, то, очевидно, данный интеграл должен быть равен 0. Этот результат будет получен при проведении интегрирования численно:

$$\int_{-1}^{1} \frac{1}{\sqrt[3]{x}}\,dx = 0$$

Однако при аналитическом интегрировании результатом будет довольно странное комплексное значение:

$$\int_{-1}^{1} \frac{1}{\sqrt[3]{x}}\,dx \to \frac{9}{4} - \frac{3}{4}\cdot i\cdot 3^{\frac{1}{2}}$$

Чтобы понять, в чем кроется причина ошибки программы, найдем первообразную для интегрируемой функции, а затем построим ее график (рис. 10.3).

$$\int \frac{1}{\sqrt[3]{x}}\,dx \to \frac{3}{2}\cdot x^{\frac{2}{3}}$$

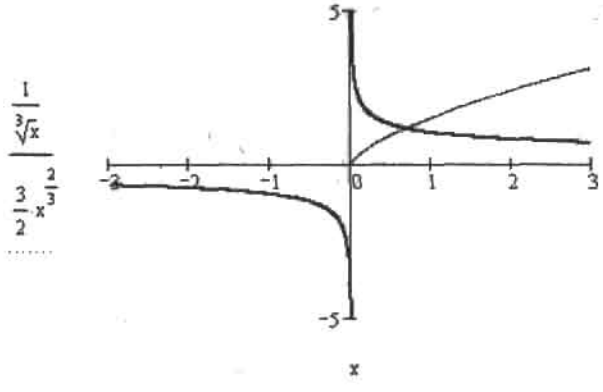


Рис. 10.3. Интегрируемая функция определена на всей числовой оси, кроме точки x=0. Ее первообразная существует лишь при неотрицательных значениях аргумента

Итак, использовать теорему Ньютона–Лейбница в случае рассматриваемого интеграла нельзя, так как первообразная не определена в действительной области при x=–1. Mathcad же этого, судя по всему, не учитывает:

$$F(x) := \frac{3}{2}\cdot x^{\frac{2}{3}} \qquad F(1) - F(-1)\ \text{simplify} \to \frac{9}{4} - \frac{3}{4}\cdot i\cdot 3^{\frac{1}{2}}$$

Как избежать ошибок, подобных описанной? Во-первых, строить график интегрируемой функции и, по возможности, ее первообразной. Во-вторых, сопоставлять результаты аналитического и численного интегрирования. В-третьих, главное, не полагаться полностью на программу и относиться осторожно к полученным результатам. И помните, что невозможно эффективно использовать Mathcad, посредственно владея математикой. Найти и исправить ошибки при этом просто нереально.

10.3. Численное вычисление определенного интеграла

В том случае, если системе не удается найти первообразную функции, приходится использовать методы численного интегрирования. Численное интегрирование в Mathcad — это куда более тонкая операция, чем интегрирование аналитическое. Зачастую, чтобы получить правильный ответ, нужно верно задать точность, выбрать наиболее эффективный алгоритм, проанализировать поведение функции и, при наличии точек разрыва, представить интеграл в виде суммы интегралов... Неплохо также иметь общие представления об алгоритмах численного интегрирования, чтобы понимать, в каких случаях они могут быть использованы, а в каких нет. Этот раздел посвящен рассмотрению важнейших принципов численного интегрирования в Mathcad.

Если вы хотите использовать численный метод, прежде всего присвойте всем параметрам, входящим в интегрируемую функцию, конкретные значения. Пределы интегрирования, естественно, также должны быть числами. В качестве оператора вывода следует использовать оператор численного вывода «=».

Пример 10.8. Численный расчет неберущихся интегралов

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}\cdot\left(1-\frac{1}{2}\cdot x^2\right)}\,dx = 4.443 \qquad \int_{0}^{\frac{3\pi}{2}} \frac{\sin(x)}{x}\,dx = 1.608$$

$$\int_{0}^{10} \frac{e^x}{x}\,dx = 2.501\times 10^3 \qquad \int_{0}^{1} e^x\cdot\ln(x)\,dx = -1.318$$

В систему Mathcad разработчиками было встроено несколько численных методов интегрирования. Необходимость этого связана с тем, что алгоритмы численного интегрирования не столь универсальны, как алгоритм дифференцирования. Каждый из них подходит для определенной группы функций или типа интеграла. Так, например, такие классические методы, как метод трапеций, средних прямоугольников или Симпсона, хороши лишь в случае гладких непрерывных функций и небольшого интервала интегрирования. Для того же, чтобы подсчитать интеграл с бесконечным пределом или интеграл от функции, имеющей точку разрыва второго рода на границе промежутка, нужно использовать принципиально другие подходы, отличающиеся от банального разбиения на равные элементарные отрезки.

Чтобы произвести смену численного метода, щелкните правой кнопкой мыши на операторе интегрирования. При этом откроется его контекстное меню, содержащее список вариантов возможных алгоритмов интегрирования.

- **AutoSelect** (Автоматический выбор). Метод интегрирования выбирается системой автоматически. Лучше всего, если по умолчанию у вас будет отмечен именно этот пункт. В большинстве случаев этого вполне достаточно для того, чтобы интеграл был подсчитан максимально правильно и без вашего участия. Однако опыт показывает, что в некоторых ситуациях (прежде всего при наличии точек разрыва), система может не справляться с правильным выбором численного метода (это связано с тем, что, как правило, автоматически Mathcad выбирает либо адаптивный метод (**Adaptive**), либо метод бесконечного предела (**Infinite Limit**)). Если вы столкнулись с проблемой при вычислении интеграла, попробуйте сменить алгоритм подсчета. Для этого вам нужно сделать правильный выбор из следующих четырех методов.
- **Romberg** (Ромберга). Весьма эффективный метод, применяемый для вычисления интегралов от функций, не имеющих особенностей (разрывов первого и второго рода, областей резкого изменения). Является основным в Mathcad. Упрощенно его можно описать следующими шагами:
  1) в качестве самого первого приближения вычисляется значение площади трапеции, основания которой проведены через границы промежутка интегрирования;
  2) затем запускается цикл. На каждом его круге шаг уменьшается вдвое. На первом обороте цикла вычисляется приближение по формуле трапеций из того условия, что интервалов интегрирования уже 2. На втором круге шаг будет уменьшен в четыре раза относительно первоначального, и соответственно трапеций также будет просуммировано четыре. И так далее до тех пор, пока не будет выполнено условие остановки цикла;
  3) в качестве условия остановки цикла в методе Ромберга обычно используют критерий разности двух последних приближений. В том случае, если разность по модулю окажется меньше TOL, то цикл будет остановлен и в качестве ответа будет выдано последнее приближение.

Данное выше описание метода Ромберга является предельно упрощенным. Реально же данный метод, за счет использования ускоряющих сходимость и увеличивающих точность математических приемов, куда более сложен (обстоятельно его мы разберем в разд. 10.6). Но несколько важных выводов можно сделать даже из такого примитивного описания.

- Точность результата численного интегрирования всегда выше или равна TOL. Поэтому, если нужно получить ответ, правильный до пятого знака мантиссы, TOL нужно присвоить значение 10^{-5}.
- Чем выше TOL, тем больше оборотов должен будет проделать цикл алгоритма. Следовательно, время расчета резко возрастает с увеличением TOL. Это может быть существенно в случае кратных интегралов или при интегрировании функции на широком интервале.
- В любых численных расчетах на компьютере имеется погрешность, у которой есть свойство накапливаться. Если расчет интеграла потребует слишком большого количества итераций, то погрешность может значительно превысить уровень желаемой точности. Чтобы этого не произошло, в реализацию метода Ромберга Mathcad введено ограничение на количество итераций. Если оно будет превышено, то система возвратит сообщение об ошибке: **Can't converge to a solution** (Не сходится к решению). Это означает, что не стоит стремиться присвоить TOL минимальное значение,

дабы получить максимально точный результат. При этом интегрирование почти наверняка не будет успешным. Обычно минимальное значение TOL для метода Ромберга лежит в пределах от 10^{-5} до 10^{-10}.

- Очевидно, что алгоритм Ромберга не предназначен для интегрирования функций с разрывами. Причем это касается как разрывов 2-го рода (в случае которых численное интегрирование в принципе возможно лишь, если первообразная в соответствующих точках непрерывна или имеет разрыв первого рода), так и разрывов 1-го рода, при наличии которых, казалось бы, подсчет интеграла не должен представлять особой сложности. Чтобы проинтегрировать функцию с разрывами первого рода, используйте адаптивный алгоритм (если для этого применить метод Ромберга, то ответ будет получен с большой погрешностью или же алгоритм попросту не сойдется). Об особенностях интегрирования при наличии точек разрыва 2-го рода мы поговорим в следующем разделе.

Численные методы интегрирования схожи в основных своих идеях. Поэтому те выводы, которые мы сделали на основании анализа метода Ромберга, можно автоматически перенести и на остальные реализованные в Mathcad методы. Особенно важно запомнить сведения о ключевой роли системной константы TOL для получения точного результата. На примере продемонстрируем как, уменьшая TOL, можно добиться нужной точности ответа.

Пример 10.9. Влияние TOL на точность численного интегрирования

Точность численного интегрирования мы будем оценивать, сравнивая результаты, полученные численными методами, с точным аналитическим решением:

$$\text{standard} := \int_{\frac{\pi}{6}}^{100\pi} \frac{\sin(x)^2}{1+\cos(x)}\,dx \rightarrow \frac{1}{2} + \frac{599}{6}\cdot\pi$$

Последовательно уменьшая величину TOL, посмотрим, как это будет влиять на точность результата:

$$TOL := 10^{-2} \qquad \int_{\frac{\pi}{6}}^{100\pi} \frac{\sin(x)^2}{1+\cos(x)}\,dx - \text{standard} = -0.021318500079929$$

$$TOL := 10^{-3} \qquad \int_{\frac{\pi}{6}}^{100\pi} \frac{\sin(x)^2}{1+\cos(x)}\,dx - \text{standard} = -4.92834374199447\times 10^{-7}$$

$$TOL := 10^{-12} \qquad \int_{\frac{\pi}{6}}^{100\pi} \frac{\sin(x)^2}{1+\cos(x)}\,dx - \text{standard} = 3.82556208933238\times 10^{-11}$$

Если уменьшить TOL до 10^{-13}, алгоритм не сойдется (рис. 10.4). Следовательно, полученный при $TOL=10^{-12}$ ответ является максимально точным в случае использования метода Ромберга.

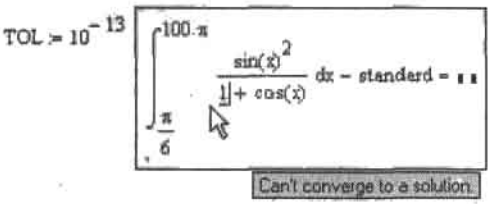


Рис. 10.4. Если TOL присвоено слишком малое значение, то численный алгоритм не сойдется

- Adaptive (Адаптивный). Метод, предназначенный для вычисления интегралов от функций, быстро изменяющихся на промежутке. Главная идея, лежащая в его основе, заключается в том, что ширина интервала разбиения не постоянна, как в случае метода Ромберга, а изменяется в зависимости от скорости изменения функции. В большинстве случаев данный алгоритм дает более точный результат, чем метод Ромберга (поэтому он используется Mathcad по умолчанию). Впрочем, как правило, различия между ответами этих численных алгоритмов начинаются после 9–10-го знака, поэтому нет принципиальной разницы, какой из них использовать при решении обычных практических задач. Впрочем, одно существенное преимущество у адаптивного метода есть. С его использованием можно интегрировать функции с разрывами первого рода (метод Ромберга в таких случаях дает большую погрешность или не сходится), а также с разрывами второго рода при условии, что первообразная в соответствующих точках непрерывна (или имеет разрыв первого рода).

Пример 10.10. Численное интегрирование с использованием адаптивного метода

При использовании адаптивного метода результат обычно получается немного точнее, чем при применении метода Ромберга:

$$\text{standart} := \int_2^3 \frac{\ln\left(x^2\right)}{x^2}\,dx \rightarrow \frac{-2}{3}\cdot\ln(3) + \frac{1}{3} + \ln(2)$$

Адаптивный метод:

$$\int_2^3 \frac{\ln\left(x^2\right)}{x^2}\,dx - \text{standart} = 0$$

Метод Ромберга:

$$\int_2^3 \frac{\ln\left(x^2\right)}{x^2}\,dx - \text{standart} = -2.803\times 10^{-14}$$

Адаптивный метод дает гораздо лучший результат, чем метод Ромберга, если на промежутке интегрирования имеются точки разрыва первого рода. Например:

Адаптивный метод:

$$\int_0^{100} \frac{x^5}{\left|x^5\right|} + \frac{x}{|x|}\,dx = 200$$

Метод Ромберга:

$$\int_0^{100} \frac{x^5}{\left|x^5\right|} + \frac{x}{|x|}\,dx = 199.941$$

Адаптивный метод, в отличие от метода Ромберга, может быть использован для интегрирования функций с разрывами второго рода (при непрерывности первообразной в точках разрывов):

$$\int_{-2}^{1} \frac{1}{\sqrt[3]{x^2}}\,dx = 6.78$$

- Infinite Limit (Бесконечный предел). Этот метод используется для подсчета несобственных интегралов с неограниченными пределами. Подробно его мы разберем в следующем разделе.
- Singular Endpoint (Сингулярный предел). Данный метод применяется для определения несобственных интегралов от неограниченных функций, если точка разрыва совпадает с границей интегрирования.

Обычно численный подсчет интеграла не вызывает особых сложностей. Но что делать, если Mathcad не сможет решить задачу? Во-первых, проверьте правильность задания интегрируемой функции и пределов интегрирования. Во-вторых, попробуйте сменить численный метод. Иногда адаптивный метод справляется с интегрированием таких функций, перед которыми пасует метод Ромберга. В-третьих, увеличьте значение TOL. Если установленный уровень точности слишком высок, то численный алгоритм может просто не успевать сходиться в рамках отведенного на количество итераций лимита. В-четвертых, проанализируйте поведение функции на интервале интегрирования с помощью графика. Вполне возможно, что она имеет точки разрыва второго рода. Об особенностях интегрирования таких функций мы поговорим в следующем разделе.

10.4. Особенности вычисления несобственных интегралов

В классическом математическом анализе при введении понятия определенного интеграла обычно предполагается, что интегрируемая функция ограничена, а интервал интегрирования конечен. Однако оказывается, что при соблюдении ряда условий определенный интеграл может иметь смысл и в случае функций, имеющих разрывы второго рода (то есть бесконечно возрастающих по абсолютному значению при приближении к некоторой точке), а также при бесконечных пределах интегрирования. Подобные интегралы называются несобственными, и в их вычислении имеется масса особенностей по сравнению с обыкновенными (собственными) интегралами.

Есть два типа несобственных интегралов: от неограниченных функций и с бесконечными пределами. Вычисляются они совершенно по-разному, поэтому стоит рассмотреть их по отдельности. Начнем мы с интегралов от неограниченных функций.

Рассмотрим функцию f(x), имеющую разрыв второго рода в точке x=b. Возможно ли, и если да, то при каких условиях, найти определенный интеграл в интервале от некоторой точки x=a до точки разрыва? Оказывается, что это реально, если первообразная для f(x) в области точки разрыва конечна (но не обязательно непрерывна). При этом соответствующий интеграл будет называться сходящимся, и по смыслу он будет совпадать с обычным определенным интегралом. Если же первообразная будет иметь в точке b разрыв второго рода, то несобственный интеграл будет называться расходящимся и равняться бесконечности. Также важным условием для того, чтобы можно было най-

ти определенный интеграл от неограниченной функции, является то, что функция должна быть непрерывной и конечной в окрестности точки разрыва.

Поясним вышесказанное на примерах. Функция

$$f(x) = \frac{1}{\sqrt{1 - x^2}}$$

имеет разрыв второго рода в точке x=1. Сходится ли интеграл от нее в интервале от 0 до 1? Очевидно, что данная функция ограничена и непрерывна на промежутке интегрирования. Поэтому, чтобы ответить на поставленный вопрос, мы должны выяснить только одно: конечна ли первообразная f(x) в левой окрестности точки x=1. Проверим это:

$$\int \frac{1}{\sqrt{1 - x^2}}\,dx \rightarrow \operatorname{asin}(x) \qquad \lim_{x \to 1^-} \operatorname{asin}(x) \rightarrow \frac{1}{2}\cdot\pi$$

Первообразная конечна, следовательно, интеграл должен иметь конечное значение. Причем найти его можно как численно, так и аналитически:

$$\int_0^1 \frac{1}{\sqrt{1 - x^2}}\,dx = 1.571 \qquad \int_0^1 \frac{1}{\sqrt{1 - x^2}}\,dx \rightarrow \frac{1}{2}\cdot\pi$$

Рассмотрим еще одну функцию, имеющую разрыв второго рода в точке x=0:

$$f(x) = \frac{1}{x^2}$$

Будет ли сходиться интеграл от данной функции, вычисленный на интервале от 0 до 1? Посмотрим, конечна ли первообразная в правой окрестности точки x=0:

$$\int \frac{1}{x^2}\,dx \rightarrow \frac{-1}{x} \qquad \lim_{x \to 0^+} \frac{-1}{x} \rightarrow -\infty$$

Первообразная не ограничена. Следовательно, расходиться будет и сам интеграл.

$$\int_0^1 \frac{1}{x^2}\,dx \rightarrow \infty$$

Задача несколько усложняется, если точка разрыва лежит не на границе, а внутри интервала интегрирования. Чтобы подсчитать интеграл в этом случае, его нужно представить в виде суммы двух интегралов, у которых точка разрыва является верхним и, соответственно, нижним пределами интегрирования. Если оба эти интеграла сходятся, то сходится и интересующий интеграл. Если один из них сходится, а второй — нет, то однозначно можно сказать, что их сумма будет равна бесконечности. Если же оба интеграла расходятся, то все зависит от того, одного ли они знака. Если да, то суммарный интеграл также будет равен бесконечности с соответствующим знаком. Если же

у них знаки противоположные, то возникает неопределенность типа ∞–∞. Интеграл при этом может быть как конечным, так и равняться бесконечности обоих знаков. Говорить что-то определенное про такие интегралы без дополнительного исследования нельзя. Однако если кривая симметрична относительно точки разрыва, то можно мысленно сокращать идентичные области. Рассмотрим, к примеру, интеграл от функции $f(x)=1/x$ на промежутке от –1 до 2. Очевидно, что площади, ограниченные кривой на промежутках от –1 до 0 и от 0 до 1 будут одинаковы, но противоположны по знаку. Значит, их можно отбросить и считать интеграл на промежутке от 1 до 2. При этом мы получим так называемое главное значение несобственного интеграла в смысле Коши. Но сам несобственный интеграл не существует.

В Mathcad можно подсчитывать интегралы от неограниченных функций как численно, так и аналитически. Аналитический подход имеет преимущества перед численным, так как он позволяет получить явный ответ в случае расходящихся интегралов (численный метод при этом выдаст сообщение об ошибке, которое не несет никакой явной информации). Также при аналитическом интегрировании интеграл не нужно разделять на два, если точка разрыва лежит в середине промежутка. Кроме того, символьный процессор «знает» значения значительного количества несобственных интегралов от функций, не имеющих первообразных в элементарных функциях.

Пример 10.11. Аналитическое вычисление несобственных интегралов от неограниченных функций

Вычисление сходящихся интегралов в случае, когда точка разрыва находится в центре интервала интегрирования (формула слева) и на его границе (формула справа):

$$\int_{-1}^{1} \frac{1}{\sqrt[3]{x^2}}\,dx \to 6 \qquad \int_{1}^{e} \frac{1}{x\cdot\sqrt{\ln(x)}}\,dx \to 2$$

Вычисление расходящихся интегралов с точкой разрыва на границе и в середине интервала. Во втором случае подсчет возможен, так как функция с обеих сторон точки разрыва имеет один и тот же знак.

$$\int_{0}^{2} \frac{1}{x^2}\,dx \to \infty \qquad \int_{-2}^{2} \frac{1}{x^2}\,dx \to \infty$$

Символьный процессор может находить значения несобственных интегралов от ряда функций, не имеющих первообразных в замкнутой форме:

$$\int_{0}^{1} \frac{e^x}{x^2}\,dx \to \infty \qquad \int_{0}^{\frac{\pi}{2}} \ln(\cos(x))\,dx \to \frac{-1}{2}\cdot\pi\cdot\ln(2) \qquad \int_{0}^{1} \frac{\operatorname{atan}(x)}{x}\,dx \to \text{Catalan}$$

Константа Catalan, которая была выдана системой при вычислении правого интеграла, это константа Каталана. Найти ее приблизительное значение с нужной точностью позволяет оператор float:

$$\text{Catalan float},10 \to .9159655942$$

Как уже упоминалось выше, при определении значений интегралов от функций, имеющих на интервале интегрирования точку разрыва второго рода и стремящихся к бесконечности разных знаков слева и справа от этой точки, возникает неопределенность. Поэтому Mathcad не вычисляет такие интегралы, возвращая исходное выражение без изменений. Впрочем, в отдельных случаях символьный процессор возвращает для подобных интегралов конечное значение. По смыслу данная величина является главным значением интеграла в смысле Коши. Найти ее можно, если считать, что площади под идентичными участками кривой сокращаются. Пример:

$$\int_{\frac{-\pi}{4}}^{\frac{\pi}{6}} \cot(x)\,dx \rightarrow \frac{-1}{2}\cdot\ln(2)$$

Если представить данный интеграл в виде суммы двух интегралов с пределами от $-\pi/4$ до 0 и от 0 до $\pi/6$, то получим неопределенность типа $\infty-\infty$:

$$\int_{\frac{-\pi}{4}}^{0} \cot(x)\,dx \rightarrow -\infty \qquad \int_{0}^{\frac{\pi}{6}} \cot(x)\,dx \rightarrow \infty$$

Найти главное значение данного интеграла можно, если считать, что площади под кривой на интервалах от $-\pi/6$ до 0 и от 0 до $-\pi/6$ равны по значению, но противоположны по знаку. Значит, их сумма равна нулю, а интеграл будет равен площади, ограниченной кривой на промежутке от $-\pi/4$ до $-\pi/6$:

$$\int_{\frac{-\pi}{4}}^{\frac{-\pi}{6}} \cot(x)\,dx \rightarrow \frac{-1}{2}\cdot\ln(2)$$

Объективно говоря, то, что в некоторых случаях результатом вычисления расходящихся интегралов являются их главные значения в смысле Коши, объясняется не интеллектуальностью аналитического процессора Mathcad. Причина этого в том, что при аналитическом вычислении интеграла программа не проверяет важнейшее условие применимости теоремы Ньютона–Лейбница, а именно непрерывность первообразной на интервале интегрирования. Подробно данная проблема описана в разд. 10.2.

Если нужно найти несобственный интеграл от функции, не имеющей первообразной в замкнутой форме, символьное интегрирование зачастую неэффективно. В этом случае нужно использовать численный алгоритм. В Mathcad встроен особый алгоритм, предназначенный для вычисления интегралов с точкой разрыва на границе интервала интегрирования (в основе его лежит модифицированный алгоритм Ромберга, о котором вы можете прочитать в справочной системе Mathcad). Чтобы его задействовать, в контекстном меню оператора интегрирования выберите пункт Singular Endpoint (Неопределенный предел).

Есть несколько важных особенностей численного подсчета несобственного интеграла от неограниченной функции.

- Интеграл должен быть сходящимся. В случае расходящегося интеграла будет выдано сообщение об ошибке. Явно то, что интеграл, равный бесконечности, при численном интегрировании узнать нельзя.
- Если точка разрыва находится посередине интервала интегрирования, то интеграл стоит представить в виде суммы интегралов так, чтобы она оказалась на верхней и, соответственно, нижней границах интегрирования. При этом в полной мере проявятся достоинства модифицированного метода Ромберга. Хотя объективности ради стоит заметить, что в большинстве случаев без этой операции можно и обойтись. Однако точность при этом будет ниже.
- Обычно сходящиеся интегралы от неограниченных функций можно подсчитать и с помощью адаптивного метода или (реже) простого метода Ромберга, однако точность результата при этом будет немного ниже.

Пример 10.12. Численное определение несобственного интеграла от неограниченной функции

Вычисление несобственного интеграла с точкой разрыва на границе интервала:

Модифицированный метод Ромберга:

$$\int_0^1 e^x \cdot \ln\left(x^2\right) dx = -2.63580430290876$$

Адаптивный метод:

$$\int_0^1 e^x \cdot \ln\left(x^2\right) dx = -2.63580430290876$$

Вычисление несобственного интеграла с точкой разрыва посередине интервала:

$$\int_{-1}^1 e^x \cdot \ln\left(x^2\right) dx = -4.22900350150292$$

Очень многие интегралы нельзя вычислить ни символьно, ни численно. Это связано с отсутствием у них первообразной и расходимостью в точке сингулярности. Поэтому при использовании численного метода будет выдано сообщение об ошибке (которое не говорит ничего конкретно), а при попытке символьного вычисления такого интеграла система возвратит первоначальное выражение. Чтобы однозначно доказать, что подобный интеграл равен бесконечности с соответствующим знаком, можно использовать метод сравнения (то есть сравнить функцию с заведомо более быстро приближающейся к асимптоте). Если интеграл функции сравнения расходится, то расходится и интеграл сравниваемой функции. Например, то, что интеграл

$$\int_0^1 \frac{\sin(x)}{\ln(x)} dx$$

равен $-\infty$, можно показать, используя следующий интеграл сравнения:

$$\int_0^1 \frac{1}{10 \cdot \ln(x)} dx \to -\infty$$

Вторым типом несобственных интегралов являются интегралы с бесконечными пределами интегрирования. Как и интегралы от неограниченных функций, они бывают сходящимися и расходящимися. Подсчитать сходящийся интеграл можно как аналитически, так и численно. Если же интеграл расходится, в этом можно достоверно убе-

диться или выполнив интегрирование символьно, или воспользовавшись методом сравнения.

Аналитический процессор Mathcad подсчитывает несобственные интегралы с бесконечными пределами, используя теорему Ньютона–Лейбница. Кроме того, ему известны значения довольно значительного количества несобственных интегралов от функций, не имеющих первообразных в элементарных функциях. Обычно, чтобы найти значение такого интеграла, требуется применять весьма изощренные приемы, поэтому многие из них именные (Дирихле, Пуассона, Лапласа и т. д.). Нередко результатом вычисления несобственных интегралов с бесконечными пределами являются интегральные функции (например, Г-функция Эйлера) или их сочетания.

Пример 10.13. Аналитический подсчет несобственных интегралов с неограниченными пределами

Сходящиеся «простые» и именные несобственные интегралы:

$$\int_0^{\infty} \frac{1}{1+x^3}\,dx \rightarrow \frac{2}{9}\cdot\pi\cdot 3^{\frac{1}{2}} \qquad \int_0^{\infty} \frac{\sin(x)}{x}\,dx \rightarrow \frac{1}{2}\cdot\pi \qquad \int_0^{\infty} \sin\left(x^2\right)dx \rightarrow \frac{1}{4}\cdot 2^{\frac{1}{2}}\cdot\pi^{\frac{1}{2}}$$

$$\int_{-\infty}^{\infty} e^{-x^2}\,dx \rightarrow \pi^{\frac{1}{2}} \qquad \int_0^{\infty} e^{-a\cdot x}\cdot\sin(b\cdot x)\,dx \text{ assume }, a > 0 \rightarrow \frac{b}{a^2+b^2}$$

Случай полной неопределенности:

$$\int_0^{\infty} \sin(x)\,dx \rightarrow \text{undefined}$$

Пример интеграла, результат для которого выражается через символьные функции:

$$\int_0^{\infty} e^{-x}\cdot x^t\,dx \text{ assume }, t > 0 \rightarrow \Gamma(t)\cdot t$$

Если аналитически найти значение несобственного интеграла с бесконечными пределами не получается, расчет следует провести численно. Для этого в Mathcad имеется специальный метод Infinite Limit (Бесконечный предел). Самостоятельно выбирать данный метод вам не придется, поскольку система автоматически переключается на него при введении в оператор интегрирования символа бесконечности. Точность данного алгоритма весьма значительно зависит от значения TOL.

Пример 10.14. Вычисление интеграла с бесконечным пределом

Точное аналитическое решение:

$$\int_0^{\infty} e^{-x}\cdot\sin(x)\,dx \rightarrow \frac{1}{2} = 0.5$$

Обычная точность ($TOL=10^{-3}$):

$$\int_0^{\infty} e^{-x}\cdot\sin(x)\,dx = 0.499999999986754$$

$TOL=10^{-13}$:

$$\int_0^{\infty} e^{-x}\cdot\sin(x)\,dx = 0.5$$

Использование для вычисления интегралов с бесконечными пределами обычных методов (Ромберга или адаптивного) либо (в лучшем случае) приведет к сообщению об ошибке, либо (в худшем случае) будет выдан неверный результат.

Имеется ряд ограничений на применение численного интегрирования для определения несобственных интегралов с неограниченными пределами. Так, использовать его для нахождения расходящегося интеграла не стоит в любом случае. Также не получится произвести расчет, если интеграл сходится неравномерно. Это прежде всего относится к периодическим функциям, например вида f(x)=sin(x)/x. Впрочем, многие несобственные интегралы от такого рода функций могут быть подсчитаны символьно.

10.5. Вычисление кратных интегралов

Чтобы подсчитать кратный интеграл, выполните следующую последовательность действий.

1. Введите командой панели Calculus (Вычисления) или соответствующим сочетанием клавиш нужный оператор интегрирования.
2. Поставьте курсор в маркер подынтегральной функции и введите второй оператор. Для вычисления тройного интеграла повторите эту операцию два раза.
3. В маркере последнего из заданных операторов пропишите интегрируемую функцию.
4. Заполните маркеры дифференциалов в том порядке, в котором должно вестись интегрирование.
5. Если вычисляется определенный интеграл, заполните соответственно маркеры пределов интегрирования.

Все те выводы и рассуждения, которые были сделаны нами относительно символьного и численного определения интеграла от функции одной переменной, в полной мере могут быть перенесены и на случай кратного интегрирования. Эта возможность объясняется тем, что для вычисления интегралов обоих типов используются одни и те же алгоритмы.

Пример 10.15. Символьное и численное вычисление кратного интеграла

$$\int_a^b \int_{1-7x}^{1-x} \frac{1}{x} \cdot y \, dy \, dx \to -12 \cdot b^2 + 6 \cdot b + 12 \cdot a^2 - 6 \cdot a$$

$$\int_{-R}^{R} \int_{-\sqrt{R^2-x^2}}^{\sqrt{R^2-x^2}} \int_{-\sqrt{R^2-x^2-y^2}}^{\sqrt{R^2-x^2-y^2}} 1 \, dz \, dy \, dx \to \frac{4}{3} \cdot \pi \cdot R^3$$

$$\int_1^2 \int_2^3 \int_3^4 x \, dx \, dx \, dx \to \frac{7}{2} \qquad \int_0^\infty \int_0^\infty e^{-x-y} \, dx \, dy = 1$$

Очень внимательно при задании кратного интеграла следует относиться к последовательности определения пределов интегрирования. Неправильно согласовав их с пере-

менными под знаками дифференциалов, вы получите неверный ответ. Ошибка такого рода — самая распространенная при вычислении кратных интегралов.

При вычислении кратных интегралов осторожно нужно относиться к использованию численных методов. Связано это с тем, что если кратность интеграла равна трем или более и интервалы интегрирования относительно широкие, то количество шагов численного алгоритма, которые требуются для получения более или менее точного значения, просто огромно. А это означает, что время расчета такого интеграла численным методом будет весьма заметным даже на мощном современном компьютере. Так, посмотрим, сколько времени в секундах может понадобиться для вычисления «проблемного» тройного интеграла:

$$\begin{aligned} &\text{time0} \leftarrow \text{time}(0) && = 349.863 \\ &\text{INT} \leftarrow \int_{-100}^{100}\int_{-200}^{200}\int_{-1000}^{1000} \sin(x+y+z)\,dx\,dy\,dz \\ &\text{time}(1) - \text{time0} \end{aligned}$$

Из данного примера следует вывод: численные методы применять тогда и только тогда, когда с поставленной задачей не справится символьный процессор.

10.6. Численные методы интегрирования

Численное интегрирование — пожалуй, одна из самых важных задач прикладной математики. Методов ее решения было создано великое множество, и каждый из них имеет свои достоинства и недостатки. Естественно, описывать их все в рамках данной книги нет никакого смысла, поэтому мы остановимся лишь на некоторых основополагающих идеях, а также опишем один из методов, использующихся системой Mathcad.

Так как, в отличие от численного решения уравнений, успех интегрирования не так сильно зависит от вашего знания идей используемого метода, то читать данный раздел совсем не обязательно, если в своей практике вы не сталкиваетесь с необходимостью самостоятельно создавать численные алгоритмы. Однако это будет полезно как с точки зрения кругозора, так и как очень неплохая тренировка техники программирования в Mathcad. Тем более численные методы интегрирования входят в курс математики технических и естественных специальностей вузов, поэтому в самостоятельной реализации численного метода может заключаться домашнее задание.

Суть любого численного метода интегрирования состоит в приближении функции другой кривой, площадь под которой можно более или менее легко подсчитать аналитически. Обычно в качестве таких кривых используются кривые алгебраических полиномов.

Самым ранним исторически и наиболее известным численным методом интегрирования является метод средних прямоугольников. Идея его вытекает из самого определения интеграла как предела суммы произведений ширины интервала разбиения на *значение* функции в его середине. То есть гладкая криволинейная трапеция заменяется на ступенчатую фигуру, площадь которой приближается к площади трапеции при стремлении ширины интервала разбиения к нулю. Иллюстрация метода средних прямоугольников показана на рис. 10.5, *а*.

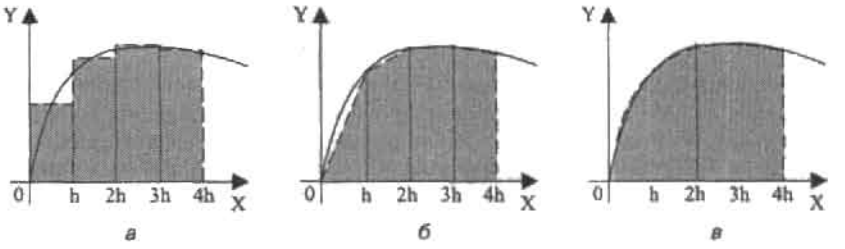


Рис. 10.5. Иллюстрация основных методов численного интегрирования

Совсем не трудно вывести общую формулу данного метода:

$$I = \sum_{i=1}^{n} \left[h \cdot \left(f\left(a + h \cdot i + \frac{h}{2} \right) \right) \right]$$

Здесь n – количество разбиений интервала интегрирования, h – ширина среднего прямоугольника. Для того же, чтобы реализовать в системе Mathcad такой простой алгоритм, которым является метод средних прямоугольников, даже не придется прибегать к использованию языка программирования:

$$Pr(f,a,b,n) := \sum_{i=1}^{n} \left[\frac{b-a}{n} \cdot \left[f\left[a + \frac{b-a}{n} \left(i - \frac{1}{2} \right) \right] \right] \right]$$

Проверим, правильно ли работает заданная функция:

$$Pr(f,a,b,n) := \sum_{i=1}^{n} \left[\frac{b-a}{n} \cdot \left[f\left[a + \frac{b-a}{n} \left(i - \frac{1}{2} \right) \right] \right] \right]$$

$$func(x) := \sin(x) \cdot x^2$$

$$\int_0^{\frac{\pi}{3}} func(x)\,dx \text{ float}, 10 \rightarrow .265488009 \qquad Pr\left(func, 0, \frac{\pi}{3}, 1000 \right) = 0.2654879007$$

Проверка показывает, что при больших n метод средних прямоугольников может быть весьма точен. Так, в рассмотренном примере 1000 разбиений дали результат, верный до пяти знаков. Для большинства практических задач это более нежели достаточная точность. А вычислить функцию 1000 раз и просуммировать полученные значения – это для современного компьютера дело, требующее долей секунды. Поэтому можно было бы остановиться на методе средних прямоугольников как на вполне приемлемом для подсчета интегралов на небольшом интервале. Однако если с помощью рассматриваемого алгоритма понадобилось бы вычислить с той же точностью тройной интеграл, то количество разбиений возросло бы до 10 000 000. А подсчитать столько раз значение сложной функции – это уже дело нескольких часов (а лет 20 назад это была бы вообще малоразрешимая задача). Приведенные цифры иллюстрируют очевидную необходимость найти какой-то другой метод численного интегрирования, ко-

торый, при сохранении точности, требовал бы меньшего количества узловых значений функции.

Уменьшить количество разбиений можно прежде всего, увеличивая точность приближения функции легко интегрируемой аналитически кривой. Наиболее простая из таких кривых — это прямая.

Метод, основанный на приближении фрагментов кривой функции секущими, проходящими через границы интервалов разбиения, называется методом трапеций (см. рис. 10.5, *б*). Точность данного метода близка к точности метода средних прямоугольников, и величина ошибки пропорциональна квадрату ширины интервала разбиения. Общая формула метода трапеций будет вполне очевидна, если вспомнить, каким образом в геометрии вычисляется площадь этой фигуры:

$$I = \sum_{i=1}^{n} \left[\frac{h}{2} [f(a + h \cdot i) + f[a + h \cdot (i - 1)]] \right]$$

Задать в Mathcad функцию, вычисляющую интеграл по методу трапеций, столь же просто, как и в случае метода средних прямоугольников:

$$\mathrm{Int}(f, a, b, n) := \sum_{i=1}^{n} \left[\left(\frac{b - a}{n} \right) \cdot \left[\frac{f\left(\frac{b - a}{n} \cdot i + a \right) + f\left[\frac{b - a}{n} \cdot (i - 1) + a \right]}{2} \right] \right]$$

$$\mathrm{func}(x) := \sin(x) \cdot x^2$$

$$\int_0^{\frac{\pi}{3}} \mathrm{func}(x)\, dx \;\mathrm{float}, 10 \rightarrow .265488009 \qquad \mathrm{Int}\left(\mathrm{func}, 0, \frac{\pi}{3}, 1000 \right) = 0.2654882245$$

Точность результата вычисления интеграла по методу трапеций, полностью в соответствии с теоретическими выводами, оказалась крайне близкой к точности метода средних прямоугольников. Соответственно точность данного метода нас вряд ли может удовлетворить.

Невысокая точность метода трапеций легко объясняется тем, что все-таки приближение кривой функции ломаной из отрезков секущих является довольно грубым даже для значительного количества разбиений. Чтобы уменьшить ошибку интерполяции, попробуем использовать для приближения функции полином второй степени — параболу.

Идея вычисления интеграла с помощью парабол довольно проста: промежуток разбивается на нешироких интервалы и на каждом из них вычисляется площадь, ограниченная интерполирующей функцию параболой (см. рис. 10.5, *в*). Чтобы задать приближающую параболу, нужно знать коэффициенты ее уравнения. Конечно, можно просто разложить в средней точке интервала функцию в ряд Тейлора — но это, пожалуй, слишком сложно и не очень надежно. Гораздо проще можно справиться с данной задачей, использовав возможности Mathcad по решению систем уравнений. Так как уравнение параболы имеет три коэффициента, то, исходя из известного правила алгебры, чтобы их найти, нужно задать систему соответственно из трех уравнений. Сделать это можно,

вычислив значение приближаемой функции в трех точках интервала. Наиболее просто выбрать в качестве таковых границы и середину интервала разбиения. В результате получим следующую систему уравнений:

$$A \cdot x0^2 + B \cdot x0 + C = f0$$

$$A \cdot (x0 + h)^2 + B \cdot (x0 + h) + C = f1$$

$$A \cdot (x0 + 2 \cdot h)^2 + B \cdot (x0 + 2 \cdot h) + C = f2$$

Здесь x0 — левая граница интервала; h — шаг, соответствующий расстоянию между узловыми точками; f0, f1, f2 — значения функции на левой границе, в середине, на правой границе интервала; A, B, C — коэффициенты уравнения параболы.

Система является линейной относительно коэффициентов, поэтому у нее будет только одно решение. Это означает, что через три точки можно провести одну единственную параболу (справедливо и обобщение — через n точек проходит только один полином степени n–1). Решить заданную систему очень просто или посредством оператора solve, или матричным способом. Ввиду объемности выражений для коэффициентов, создадим переменные A, B и C, которые будут на них ссылаться:

$$COF := \begin{bmatrix} A \cdot x0^2 + B \cdot x0 + C = f0 \\ A \cdot (x0 + h)^2 + B \cdot (x0 + h) + C = f1 \\ A \cdot (x0 + 2 \cdot h)^2 + B \cdot (x0 + 2 \cdot h) + C = f2 \end{bmatrix} \text{solve}, A, B, C \rightarrow$$

$$A := (COF^T)_0 \qquad B := (COF^T)_1 \qquad C := (COF^T)_2$$

После того как коэффициенты приближающей параболы будут определены, элементарно можно найти площадь, которую ограничивает ее фрагмент на интервале интегрирования:

$$\int_{x0}^{x0+2 \cdot h} A \cdot x^2 + B \cdot x + C \, dx \text{ simplify} \rightarrow \frac{1}{3} \cdot h \cdot (f0 + 4 \cdot f1 + f2)$$

Полученная общая формула называется формулой Симпсона, и с ее помощью можно находить значение интеграла, и не решая систему уравнений для каждого интервала разбиения. Достаточно вычислить три узловых значения интегрируемой функции.

Реализация метода Симпсона в Mathcad будет выглядеть следующим образом:

$$Simp(f, a, b, n) := \frac{1}{2} \cdot \sum_{i=1}^{n} \left[\frac{b-a}{3 \cdot n} \cdot \left[f\left[\frac{b-a}{n} \cdot (i-1) + a \right] \ldots + \left[f\left(\frac{b-a}{n} \cdot i + a \right) \ldots + 4 f\left[\left[\frac{b-a}{n} \cdot \left(i - \frac{1}{2} \right) + a \right] \right] \right] \right] \right]$$

$$func(x) := \sin(x) \cdot x^2$$

$$\int_0^{\frac{\pi}{3}} \text{func}(x)\,dx \text{ float}, 15 \rightarrow .26548800861814 \qquad \text{Simp}\left(\text{func}, 0, \frac{\pi}{3}, 100\right) \approx 0.265488008580604$$

При количестве разбиений, в 10 раз меньшем, чем было использовано для расчета интеграла методами средних прямоугольников и трапеций, точность результата оказалась на 5 (!) порядков выше. А это означает, что метод Симпсона можно уже использовать и для вычислений кратных интегралов.

Приближать интегрируемую функцию можно полиномами и более высокой степени. Обычно для этого используются полиномы 3-й и 4-й степени (описываются соответственно формулой Симпсона 3/8 и формулой Буля). Получить формулы для них можно точно так же, как мы получили формулу Симпсона. В общем случае, чем выше степень интерполирующего полинома, тем точнее приближение. Так, ошибка при вычислении интеграла по формуле Буля убывает пропорционально 6-й степени величины интервала разбиений (формула Симпсона на два порядка менее точная). Из-за ряда технических сложностей полиномы выше 4-й степени используются редко.

Чтобы более объективно сравнить эффективность работы рассмотренных методов, определим подбором то количество разбиений, которое требуется для каждого из них, чтобы приблизить интеграл с заведомо известным аналитическим решением с точностью до 10-го знака мантиссы:

$$f(x) := \sin(x) \qquad \int_0^{\pi} f(x)\,dx \rightarrow 2$$

$$\text{Pr}(f, 0, \pi, 100000) - 2 = 8.2235551701614895 \times 10^{-11}$$

$$\text{Int}(f, 0, \pi, 130000) - 2 = -9.7301278145778269 \times 10^{-11}$$

$$\text{Simp}(f, 0, \pi, 170) - 2 = 8.099 \times 10^{-11}$$

Полученные цифры не могут не впечатлять: при использовании метода параболы разбиений требуется почти в 1000 раз меньше, чем для методов средних прямоугольников и трапеций. Для достижения же стандартной точности (0.001) будет достаточным разбить промежуток всего на 4 (!) интервала:

$$\text{Simp}(f, 0, \pi, 4) = 2.00026917$$

Благодаря такой высокой эффективности формулы Симпсона до появления компьютерной техники ее использовали для подсчета интегралов, которые нельзя было вычислить аналитически, на бумаге. Кстати, если использовать, например, формулу Буля, то разбиений потребуется еще меньше.

В теории численных методов имеются формулы, используя которые можно предсказать, какой будет порядок у ошибки при вычислении интеграла от функции f(x) на промежутке от a до b с шагом h. Соответствующая формула для метода трапеций имеет вид:

$$O(h^2) = \frac{(b-a)\cdot\left(\overline{\frac{d^2}{dx^2} f(x)}\right)\cdot h^2}{12}$$

Формула для ошибки метода Симпсона будет схожей:

$$O\left(h^4\right) = \frac{(b-a)\cdot\left(\overline{\frac{d^4}{dx^4}f(x)}\right)\cdot h^4}{180}$$

Данные формулы показывают, что ошибка зависит прежде всего от величины шага. Также на нее будет влиять ширина интервала интегрирования (чем больше шагов придется проделать алгоритму, тем сильнее будут сказываться накапливающиеся погрешности) и скорость изменения функции (чем она выше, тем больше будет погрешность). Количественно скорость изменения функции можно узнать, вычислив производную. Однако производная дает мгновенную скорость, а нам нужно знать среднюю скорость изменения функции на промежутке интегрирования. Поэтому необходимо найти среднее значение производной на промежутке. Именно эта величина фигурирует в приведенных выше формулах.

Формулы ошибки методов численного интегрирования важны в связи с тем, что они позволяют определить, насколько малым должен быть шаг, чтобы была достигнута необходимая точность. Например, попробуем узнать, какой величины нужно сделать шаг, чтобы вычислить интеграл от $f(x)=\sin(x)$ на промежутке от 0 до π методом Симпсона с точностью до 5-го знака.

Выразив из формулы ошибки метода Симпсона h, получим следующее неравенство:

$$h \le \sqrt[4]{\frac{180\,TOL}{(b-a)\cdot\overline{\frac{d^4}{dx^4}f(x)}}}$$

Все входящие в данное выражение величины, кроме среднего значения производной, нам известны. Чтобы найти недостающую величину, рассчитаем четвертую производную аналитически, после чего найдем среднее значение полученной функции на промежутке, используя следующее замечательное свойство интеграла:

$$\overline{f(x)} = \frac{1}{b-a}\cdot\int_a^b f(x)\,dx$$

Расчет:

$$\frac{d^4}{dx^4}\sin(x) \rightarrow \sin(x) \qquad \frac{1}{\pi}\cdot\int_0^{\pi}\sin(x)\,dx \rightarrow \frac{2}{\pi}$$

Зная среднее значение производной на промежутке, находим величину шага и количество шагов:

$$\frac{d^4}{dx^4}\sin(x) \rightarrow \sin(x) \qquad \frac{1}{\pi}\cdot\int_0^{\pi}\sin(x)\,dx \rightarrow \frac{2}{\pi}$$

Проводим интегрирование по Симпсону, опираясь на полученный выше результат. Ответ проверяем посредством аналитического интегрирования:

$$f(x) := \sin(x)$$

$$\text{Simp}(f, 0, \pi, 101) = 2.00000000065 \qquad \int_0^{\pi} \sin(x)\,dx \to 2$$

На практике при определении необходимой величины шага среднее значение производной обычно не вычисляется ввиду того, что для этого необходимо провести интегрирование. Вместо него в формулу для ошибки подставляется наибольшее значение (по модулю) производной на промежутке. Неравенство для h от этого не нарушается, только величина шага получается слегка заниженной. Так, с учетом, что на промежутке от 0 до π максимальное значение синуса — 1, мы получим следующий результат:

$$\sqrt[4]{\frac{180\,10^{-8}}{\pi}} = 0.028 \qquad \frac{\pi}{0.028} = 112.2$$

Как видите, величина шага при точном нахождении среднего значения производной и замене его наибольшим ее значением различается мало — всего на 10%.

Проверка показала, что формулы для определения шага интегрирования вполне эффективны. Однако их применение при реализации алгоритмов численного интегрирования невозможно из-за сложностей, с которыми связано вычисление среднего значения производной или даже ее максимальной величины. В первом случае нужно проводить интегрирование, во втором — применять методы численной оптимизации. И то, и другое само по себе является серьезной задачей. Но можно ли как-то вычислить интеграл с заданной точностью, не применяя формул для ошибки?

Решение поставленной проблемы можно легко найти, если вспомнить, что с ростом количества разбиений результаты вычислений любым численным методом образуют равномерно сходящуюся (в случае отсутствия погрешности округления) последовательность. Данная последовательность более или менее быстро, в зависимости от используемого алгоритма, сходится к истинному значению интеграла. В принципе, абсолютная точность приближения наступает лишь при достижении ширины интервала разбиения бесконечно малого значения. Однако на практике нас всегда интересует определенная, обычно не очень высокая точность результата. Очевидно, что чем ближе находятся два соседних значения интеграла (вычисленные соответственно при n и 2·n разбиениях) к истинному значению, тем на меньшую величину они отличаются. А что если использовать их разность в качестве критерия достижения требуемой точности?

Описанная идея реализуется в программе AutoInt, которая вычисляет интеграл по методу трапеций. Данная программа генерирует последовательность приближений к интегралу, удваивая на каждом обороте количество разбиений промежутка. Технически это реализуется с помощью бесконечного цикла while, который прекращает работу в том случае, если два последних приближения к интегралу отличаются на величину, меньшую TOL. Сделать же цикл бесконечным очень просто: для этого выражение в его правом маркере должно всегда давать истину (то есть 1). В принципе, можно просто ввести в правый маркер 1.

$$\text{AutoInt}(f,a,b,TOL) := \left|\begin{array}{l} n \leftarrow 1 \\ \text{INT\_PREV} \leftarrow (a-b)\cdot\left(\dfrac{f(a)+f(b)}{2}\right) \\ \text{while } 1 \\ \quad \left|\begin{array}{l} n \leftarrow n\cdot 2 \\ h \leftarrow \dfrac{b-a}{n} \\ \text{INT} \leftarrow 0 \\ \text{for } i \in 1..n \\ \quad \text{INT} \leftarrow \text{INT} + h\cdot\left[\dfrac{f[a+h\cdot(i-1)]+f(a+h\cdot i)}{2}\right] \\ \text{break if } (|\text{INT}-\text{INT\_PREV}|<TOL) \\ \text{INT\_PREV} \leftarrow \text{INT otherwise} \end{array}\right. \\ \text{INT} \end{array}\right.$$

Проверим, насколько эффективно работает написанная программа:

$$f(x) := \sin(x)$$

$$\text{AutoInt}\left(f,0,\pi,10^{-5}\right)-2 = -1.5687316193\times10^{-6}$$

$$\text{AutoInt}\left(f,0,\pi,10^{-8}\right)-2 = -1.5319761015\times10^{-9}$$

$$\text{AutoInt}\left(f,0,\pi,10^{-12}\right)-2 = -1.2079226508\times10^{-13}$$

Программа работает неплохо. Конечно, она не так эффективна, как встроенные в Mathcad алгоритмы, однако ее можно улучшить. Главный ее недостаток заключается в том, что при каждом удвоении количества разбиений значения функции в узловых точках пересчитываются «с нуля», хотя из 2·n значений половина уже была вычислена на предыдущей итерации цикла while. Также, если n>1, то лучше использовать не формулу трапеций, а формулу Симпсона или Буля. Исправить описанные недостатки, сохранив главную идею программы AutoInt можно, применяя алгоритм Ромберга.

Метод Ромберга довольно сложен для понимания, однако мы попробуем максимально просто описать его идею. Для этого лучше всего будет изложить последовательность действий при решении задач этим методом.

1. В качестве самого первого приближения вычисляется значение площади трапеции, основания которой проведены через границы промежутка интегрирования. Полученное значение заносится в матрицу приближений как:

$$R_{0,0} \leftarrow \frac{(b-a)}{2}(f(a)+f(b))$$

2. Затем запускается бесконечный цикл. На каждом его круге шаг уменьшается вдвое. На первом обороте цикла вычисляется приближение по формуле трапеций из того условия, что интервалов интегрирования уже 2. Полученное, более точное, приближение интеграла заносится в первую строку нулевого столбца. На втором круге шаг

будет уменьшен в четыре раза относительно первоначального, и соответственно трапеций также будет просуммировано 4. Новое приближение будет занесено во вторую строку. И так далее до тех пор, пока не будет выполнено условие остановки цикла. На языке программирования описанную последовательность действий можно задать так:

$$\text{while } 1$$

$$\left|\begin{array}{l} J \leftarrow J + 1 \\ h \leftarrow \dfrac{(b-a)}{2^J} \\ R_{J,0} \leftarrow \dfrac{R_{J-1,0}}{2} + h \cdot \displaystyle\sum_{p=1}^{2^{J-1}} f[a + h\cdot(2p-1)] \end{array}\right.$$

Изучая данный фрагмент программы, вы не найдете уже хорошо нам известную формулу метода трапеций. Здесь она заменена на более простую с точки зрения вычислительной работы рекуррентную формулу. Формула позволяет, использовав значение интеграла, полученное на шаг раньше при вдвое большей ширине интервала разбиения, найти новое приближение. Достоинство этой формулы в том, что она приводит к результату при вдвое меньшем количестве вычислительных операций, чем простая формула трапеций.

3. Как было показано выше, точность формул Симпсона и Буля значительно выше, чем точность метода трапеций. Поэтому вполне логичным было бы использовать их, а не ждать, пока в ходе работы цикла количество разбиений промежутка возрастет до тех десятков тысяч, при которых алгоритм трапеций может дать достаточную точность.

   В алгоритме Ромберга действительно производится переход к более точным интерполирующим полиномам высокой степени. Причем происходит он таким образом, что никаких приближений интеграла, кроме приближения по методу трапеций, вычислять не приходится.

   Все дело в том, что в математике доказана связь между приближениями интеграла, рассчитанными различными численными методами. Так, зная два приближения, полученные по методу трапеций, можно вычислением совершенно простого арифметического выражения получить приближение, которое по точности соответствует формуле Симпсона. Обязательным условием (на основе этого строился вывод) для используемых двух приближений более низкой степени точности является то, что шаг, при котором они были получены, должен отличаться в два раза:

$$S(f,h) = \frac{4T(f,h) - T(f,2h)}{3}$$

   Не стоит пугаться этой общей формулы, на самом деле она предельно проста. Она лишь показывает, что для того чтобы получить приближение по Симпсону с шагом h, следует умножить приближение, полученное при этом шаге по методу трапеций, на 4, отнять от него элемент того же столбца, соответствующий вычислению интеграла методом трапеций при шаге 2h, и поделить затем результат на 3. То есть вычисление приближения более высокой степени в том случае, если есть два значения интеграла, рассчитанные по более простому методу, сводится к элементарной арифметической операции из трех действий.

Вычислив приближения по Симпсону, можно получить приближения еще на два порядка более точные, если применить рекуррентную формулу Буля:

$$B(f,h) = \frac{16S(f,h) - S(f,2h)}{15}$$

В общем виде формула уточнения имеет вид:

$$R(j,k) = \frac{4^k \cdot R(j,k-1) - R(j-1,k-1)}{4^k - 1}$$

Проще всего понять ее смысл можно, представив себе соответствующую таблицу приближений. Первый столбец из j элементов будет полностью заполнен приближениями, полученными с помощью рекуррентной формулы трапеций. Приближений по Симпсону будет на одно меньше, поскольку первое приближение алгоритма трапеций не имеет верхней, вычисленной при вдвое большем шаге, пары. Соответственно в столбце приближений, вычисленных по формуле Буля, не будет двух верхних элементов, в следующем столбце — трех и т. д. В результате будет получена нижняя треугольная матрица, элемент нижнего правого угла которой и является наиболее точным приближением (рис. 10.6).

Рис. 10.6. Схема интегрирования по Ромбергу для пяти приближений

Если отнять полученную при вычислении интеграла по Ромбергу матрицу от соразмерной, все ненулевые элементы которой — истинные значения подсчитываемого интеграла, поделить на величину истинного значения и умножить на 100, то в результате получится характеристическая матрица процентных ошибок приблизительно следующего вида:

$$\begin{pmatrix} 38.7894 & 0 & 0 & 0 & 0 & 0 \\ 21.46018 & -4.71976 & 0 & 0 & 0 & 0 \\ 5.19406 & -0.22799 & 0.07146 & 0 & 0 & 0 \\ 1.28842 & -0.01346 & 8.43455\times10^{-4} & -2.775\times10^{-4} & 0 & 0 \\ 0.32148 & -8.2955\times10^{-4} & 1.2375\times10^{-5} & -8.15\times10^{-7} & 2.7\times10^{-7} & 0 \\ 0.08033 & -5.167\times10^{-5} & 1.9\times10^{-7} & -5\times10^{-9} & 0 & 0 \end{pmatrix}$$

Из этой матрицы видно, сколь быстро убывают ошибки последовательных приближений, вычисляемых с помощью описанных рекуррентных формул. Самих же вычислений при этом нужно сделать очень немного. Поэтому метод Ромберга не только очень точен, но и весьма быстр.

4. В программной реализации алгоритма Ромберга удобно осуществлять одновременное нахождение всех возможных для данного шага приближений (то есть на каждом круге цикла while заполняется одна строка приведенной выше матрицы, начиная со второй):

$$\text{for } k \in 1..J$$

$$R_{J,k} \leftarrow \frac{4^k \cdot R_{J,k-1} - R_{J-1,k-1}}{4^k - 1}$$

5. В качестве условия остановки цикла в методе Ромберга обычно используют критерий разности двух последних приближений, лежащих на диагонали матрицы ($R_{J,J}$ и $R_{J-1,J-1}$). В том случае, если разность по модулю окажется меньше TOL, то цикл будет остановлен и в виде ответа будет выдано последнее приближение:

$$\text{break if } \left|R_{J,J} - R_{J-1,J-1}\right| < TOL$$

Как видите, реализация метода Ромберга, в общем, несложна. Однако все равно стоит привести соответствующую программу в «собранном» виде:

$$\begin{aligned}
&\text{Romberg}(f,a,b,TOL) := \left|\begin{array}{l}
J \leftarrow 0 \\
R_{0,0} \leftarrow \dfrac{(b-a)}{2}(f(a)+f(b)) \\
\text{while } 1 \\
\quad \left|\begin{array}{l}
J \leftarrow J+1 \\
h \leftarrow \dfrac{(b-a)}{2^J} \\
R_{J,0} \leftarrow \dfrac{R_{J-1,0}}{2} + h \cdot \displaystyle\sum_{p=1}^{2^{J-1}} f[a+h\cdot(2p-1)] \\
\text{for } k \in 1..J \\
\quad R_{J,k} \leftarrow \dfrac{4^k \cdot R_{J,k-1} - R_{J-1,k-1}}{4^k - 1} \\
\text{break if } \left|R_{J,J} - R_{J-1,J-1}\right| < TOL
\end{array}\right. \\
R_{J,J}
\end{array}\right.
\end{aligned}$$

Проверим, насколько качественно работает написанная программа. Для этого сравним результат расчета интеграла, произведенного ею, с точным аналитическим значением данного интеграла:

$$f(x) := \sin(x)\cdot x^2 \qquad \int_0^{\pi} f(x)\,dx \rightarrow \pi^2 - 4$$

$$\text{Romberg}(f, 0, \pi, 10^{-3}) - (\pi^2 - 4) = 4.558\times 10^{-7}$$

$$\text{Romberg}(f, 0, \pi, 10^{-5}) - (\pi^2 - 4) = -1.659\times 10^{-10}$$

$$\text{Romberg}(f, 0, \pi, 10^{-8}) - (\pi^2 - 4) = 1.51\times 10^{-14}$$

Как видно из приведенных примеров, эффективность написанной программы очень высока и точность результата значительно превышает точность уровня TOL. Кроме того, алгоритм очень быстро сходится и расчет даже с предельной точностью занимает доли секунды. А это означает, что метод Ромберга можно применять для вычисления интегралов любого типа, а также кратных интегралов.

Описывать остальные встроенные в Mathcad методы мы не будем, прежде всего по причине их специфичности. При желании информацию о них вы сможете найти в справочной системе программы или в книге, посвященной численным методам.

Глава 11. Вычисление производных

Производная – это одно из основных понятий современной математики, это инструмент, имеющий огромное практическое значение. С помощью производной можно охарактеризовать скорость изменения функции, найти ее максимальные и минимальные значения, а также точки перегибов. К вычислению производной сводятся решения значительного количества научных и производственных задач (например, нахождение скорости химической реакции или определение оптимальной формы бака для топлива). В этой главе мы обсудим особенности вычисления производных в Mathcad. В ее первой части приведены необходимые теоретические сведения. Во второй части главы мы разберем, как решаются наиболее важные задачи, связанные с дифференцированием (построение касательной, нормали). Однако задачи, направленные на исследование функции (определение экстремумов, перегибов, асимптот), решаться в данной главе не будут. Подробному описанию этой объемной темы посвящена гл. 13.

11.1. Принципы расчета производных в Mathcad

Аналогично большинству остальных наиболее важных математических операций, в Mathcad существует численное и символьное дифференцирование.

С вычислительной точки зрения аналитическое определение производной почти любой встречаемой на практике функции – дело, в общем-то, технически очень простое. Однако, в случае сложных выражений и дробей, аналитическое дифференцирование может потребовать громоздких выкладок и быть чрезвычайно трудоемким. Очевидно, что такая работа вряд ли кому-то покажется интересной. Поэтому возможности системы Mathcad в этой области могут помочь сэкономить время и силы. Ключевым моментом здесь является то, что, в отличие от символьного интегрирования или решения уравнений, аналитически можно просчитать производную любой функции. Зачем же тогда нужно численное дифференцирование? По большому счету, в 99% случаев без численного дифференцирования можно обойтись. Однако в ряде специфических задач оно более удобно, чем аналитическое дифференцирование. Так, если нужно получить значение производной в точке (к примеру, чтобы провести касательную), а аналитическое выражение производной слишком громоздко, дифференцирование проводить стоит численно. Иногда приходится дифференцировать зависимости, которые не могут быть выражены каким-то аналитическим выражением. Наиболее типичный

случай — интерполяция экспериментальной зависимости квадратичным или кубическим сплайном (подробно соответствующая задача разбирается в гл. 16). Естественно, что аналитическое дифференцирование в случае подобных зависимостей не имеет смысла. Также зачастую немаловажно то, что значение производной в точке численно определяется гораздо быстрее, чем аналитически. Конечно, при вычислении одной производной это не играет ровным счетом никакой роли: время, которое потребуется современному компьютеру для дифференцирования любой функции, покажется вам неуловимым мгновением, независимо от используемого подхода. Однако, если при решении задачи производные придется вычислить сотни или тысячи раз (например, при поиске корней системы уравнений), использование численных методов более эффективно. В то же время у численного дифференцирования есть значительные недостатки, главным из которых является значительная погрешность, возникающая при вычислении производных большой степени.

Подводя итог, можно сказать: выбор метода дифференцирования должен зависеть от типа решаемой задачи. Символьный метод имеет преимущество в том плане, что результат можно получить в виде функции, которую можно будет использовать в дальнейших расчетах (а погрешность вычисления, исходя из значения производной в интересующей точке, будет ограничена только ошибкой округления или же, если символьный подсчет возможен, будет вообще отсутствовать). Численный же подход имеет преимущества в некоторых специфических задачах. Особенности как символьного, так и численного подхода мы продемонстрируем ниже на примерах.

Mathcad позволяет вычислять как обычную производную, так и производные более высоких порядков, а также частные производные. Об особенностях расчета каждого из этих типов производной мы поговорим по отдельности.

11.1.1. Определение первой производной

Одним из наиболее важных достоинств системы Mathcad является то, что в ее основе лежит принцип WYSIWYG (What you see is what you get — что вы видите, то и получите). Исходя из этого принципа, запись производной в программе полностью соответствует принятым в математике правилам, и для того чтобы верно ввести какое-то выражение, не нужно знать специального синтаксиса. Сделать это можно точно так же, как если бы вы проводили данную операцию на бумаге. Поэтому, приступая к вычислению производной какой-либо функции, следует найти такой знакомый любому студенту, инженеру или ученому оператор дифференцирования. Оператор первой производной (Derivative) расположен на панели Calculus (Вычисления) и, помимо того, вводится сочетанием клавиш Shift+«/» (рис. 11.1).

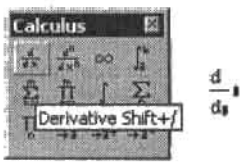


Рис. 11.1. Оператор простого дифференцирования на панели Calculus

Оператор первой производной имеет два маркера, принцип заполнения которых абсолютно очевиден: в верхний вводится функция, в нижний — переменная, по которой проводится дифференцирование.

Когда оператор будет заполнен, следует решить, в какой форме необходимо получить ответ. Если в результате дифференцирования должна быть получена функция производной, следует обратиться к возможностям символьного процессора. Для этого в качестве оператора вывода следует использовать оператор символьного вывода (→). При символьном дифференцировании можно оперировать функциями нескольких переменных и функциями с параметрами. Также оператор дифференцирования может сочетаться с любым вычислительным или символьным оператором. Особенно полезен оператор simplify, так как выражение производной выдается в неупрощенном виде. Иногда для упрощения ответа стоит использовать операторы collect (приводит подобные слагаемые), factor (раскладывает выражение на множители) и expand (раскрывает скобки).

Пример 11.1. Аналитическое вычисление производных

Чтобы в этом примере выражение производной получилось компактным, его нужно упростить, а затем в знаменателе выполнить приведение подобных слагаемых по cos(x).

$$\frac{d}{dx}\left(\frac{\cos(x) - x\cdot\sin(x)}{\sin(x) + x\cdot\cos(x)}\right)\ \left|\begin{array}{l}\text{simplify}\\ \text{collect, cos(x)}\end{array}\right.\rightarrow \frac{-\left(2 + x^2\right)}{\left(x^2 - 1\right)\cdot\cos(x)^2 + 2\cdot\sin(x)\cdot x\cdot\cos(x) + 1}$$

Пример вычисления производной функции от функции. Условие получения компактного ответа – использование оператора simplify.

$$\frac{d}{dx}\left[\frac{1}{3}\ln\left[\frac{(x-2)\cdot(x+1)^5\cdot(x+2)^2}{(x-1)^2}\right]\right]\ \text{simplify}\ \rightarrow \frac{2\cdot x^3 - 3\cdot x^2 - 5\cdot x + 10}{(x+2)\cdot(x-2)\cdot(x+1)\cdot(x-1)}$$

Пример вычисления производной от функции с буквенными параметрами.

$$\frac{d}{dx}(x\cdot\sin(a\cdot x)) + \cos(b\cdot x)^2 \rightarrow \sin(a\cdot x) + x\cdot\cos(a\cdot x)\cdot a + \cos(b\cdot x)^2$$

Пример совмещения оператора дифференцирования с другим вычислительным оператором (иллюстрация правила Лопиталя–Бернулли – раскрытие неопределенности типа 0/0).

$$\lim_{x\to 0}\frac{\frac{d}{dx}(\sin(4x) + x)}{\frac{d}{dx}(2x - \sin(3x))} \rightarrow -5$$

Чтобы получить численное значение производной в нужной точке исходя из результатов символьного расчета, нужно поступить следующим образом.

1. Найти функцию производной, используя оператор символьного вывода (→).
2. Присвоить переменной соответствующее численное значение.
3. Скопировать полученное выражение для производной и вычислить его символьно. Также данное выражение можно просчитать, используя оператор simplify. При этом результат будет получен в удобной и не имеющей погрешности символьной форме. Однако порой ответ выходит громоздким, и его приходится пересчитать в десятичную дробь (поставив после выражения оператор численного вывода или, лучше, использовав оператор float).

Чтобы найти производную функции, не обязательно вводить ее выражение непосредственно в оператор дифференцирования. Можно первоначально определить производную как некоторую функцию пользователя и в дальнейшем получать численные значения, просто вводя нужные величины переменной в скобки. Такой способ более удобен тем, что позволяет получать одновременно значения производной во многих точках. Просчитывать такую функцию можно только численно. Если те же расчеты нужно провести аналитически, определите, используя оператор символьного вывода («→»), выражение производной в общем виде, а затем задайте его ниже как функцию. Указывая в скобках необходимые значения переменной, вы получите значения производной в различных точках в символьной форме.

Пример 11.2. Определение значения производной в точке

Численный расчет:

$$r(\phi) := \frac{\phi + \tan(\phi)}{1 + \tan(\phi)} \qquad r'(\phi) := \frac{d}{d\phi} r(\phi)$$

$$r'(-\pi) = 5.142 \qquad r'(\pi) = -1.142$$

Аналитический расчет:

$$\frac{d}{d\phi}\frac{\phi + \tan(\phi)}{1 + \tan(\phi)} \text{ simplify } \rightarrow \frac{-\left(-2 - \tan(\phi) - \tan(\phi)^2 + \phi + \phi\cdot\tan(\phi)^2\right)}{(1 + \tan(\phi))^2}$$

$$r'(\phi) := \frac{-\left(-2 - \tan(\phi) - \tan(\phi)^2 + \phi + \phi\cdot\tan(\phi)^2\right)}{(1 + \tan(\phi))^2}$$

$$r'(-\pi) \rightarrow 2 + \pi \qquad r'(\pi) \rightarrow 2 - \pi$$

Существенным недостатком численного определения производной является то, что алгоритм находит значения в точках, где функция производной терпит разрыв. Разумеется, такие результаты абсолютно некорректны и не имеют смысла. В случае же символьного расчета при попытке найти производную в точке, где она не существует, система выдаст сообщение об ошибке: Can't divide by zero (Невозможно деление на нуль). Чтобы убедиться в сказанном, попробуйте найти значение производной обоими методами в точке $-\pi/4$.

Чтобы численно получить значение производной в конкретной точке, после оператора дифференцирования можно сразу ввести оператор численного вывода («=»). Естественно, при этом выше самой производной следует определить соответствующее значение переменной.

Пример 11.3. Численное дифференцирование

$$\phi := -\frac{\pi}{4} \qquad \frac{d}{d\phi}\frac{\phi + \tan(\phi)}{1 + \tan(\phi)} = 0.5$$

При численном определении значения производной в точке ни в коем случае нельзя задавать величину переменной в самом операторе дифференцирования: при этом в пер-

вую очередь будет подсчитано значение самой функции, и производная уже будет вычисляться от некоторой постоянной величины. Естественно, что, независимо от вида функции, ответ будет один и тот же: 0. Кстати, это относится и к символьному дифференцированию.

Пример 11.4. Ошибка при расчете производной

$$x := 0$$

$$\frac{d}{dx}\cos\left(\frac{\pi}{2}\right) = 0 \qquad \frac{d}{dx}\cos\left(\frac{\pi}{2}\right) \rightarrow 0$$

Попробуем определить, с какой точностью находится значение производной в точке при использовании численного дифференцирования. Для этого сравним ответы, выданные при численном и аналитическом расчете производной. Естественно, что результат аналитического расчета будет эталонным (так как, после пересчета в десятичную дробь, в нем будет присутствовать ошибка лишь на уровне 14–15-го знака).

Пример 11.5. Точность численного дифференцирования

Находим выражение производной:

$$\frac{d}{dx}\frac{\sin(x)+\cos(x)}{\sin(x)-\cos(x)} \rightarrow \frac{\cos(x)-\sin(x)}{\sin(x)-\cos(x)} - \frac{(\sin(x)+\cos(x))^2}{(\sin(x)-\cos(x))^2}$$

Присваиваем переменной нужное значение:

$$x := \frac{\pi}{3}$$

Вычисляем значение производной в точке аналитически:

$$\frac{\cos(x)-\sin(x)}{\sin(x)-\cos(x)} - \frac{(\sin(x)+\cos(x))^2}{(\sin(x)-\cos(x))^2}\ \text{float},20 \rightarrow -14.928203230275509$$

Проводим численное дифференцирование:

$$\frac{d}{dx}\frac{\sin(x)+\cos(x)}{\sin(x)-\cos(x)} = -14.92820323029353$$

Отличия в ответах начинаются с 13-го знака мантиссы.

Как видно из примера 11.5, точность численного метода, используемого Mathcad для определения значения производной в точке, очень высока — ошибка появилась только в 13-м знаке мантиссы. Впрочем, в зависимости от вида функции, точность может быть как выше (вплоть до 15-го знака мантиссы), так и существенно ниже. Разработчики рекомендуют доверять 7–8 знакам результата, полученного при численном определении значения производной первого порядка. Точность вычисления производных более высоких порядков обычно ниже. Также ответ, содержащий большую ошибку, может быть получен при определении значения производной в точке, лежащей недалеко от точки разрыва или вблизи других особенностей функции.

Важно понимать, что если оператор дифференцирования входит в сложное выражение, которое подсчитывается численно, то будет задействован численный алгоритм. Если

же выражение вычисляется аналитически, то и производная будет определена символьно.

По определению, производной функции f(x) называется предел отношения приращения этой функции к приращению аргумента, когда последнее стремится к нулю. В принципе, в Mathcad можно подсчитывать производные и без использования оператора дифференцирования, а только на основании определения производной. Это может быть полезно при оформлении докладов и статей, а также при решении некоторых специфических задач.

Пример 11.6. Вычисление функции производной исходя из ее определения

Вычислить производную для f(x)=ln(a·x+b).

$$\lim_{h \to 0} \frac{\ln[a\cdot(x+h)+b]-\ln(a\cdot x+b)}{h} \to \frac{a}{a\cdot x+b}$$

11.1.2. Производные высших порядков

Чтобы подсчитать производную выше первого порядка, следует использовать специальный оператор Nth Derivative панели Calculus (Вычисления). Помимо кнопки панели, его можно ввести сочетанием клавиш Ctrl+Shift+«/». Оператор содержит четыре маркера, которые заполняются полностью в соответствии с принятыми в математике правилами. Кстати, как только вы введете значение степени в один маркер порядка, в другом оно появится автоматически.

Порядок производной должен быть обязательно числом. Аналитический подсчет производной с неявно заданным порядком невозможен даже в простейших случаях. Наоборот, вид функции, от которой находится производная, может быть очень сложным. Также возможен аналитический подсчет производной от функции нескольких переменных или от функции с неявно заданными параметрами.

Пример 11.7. Вычисление производных высших порядков

Вычисляя производную, символьный процессор не упрощает полученное выражение. Поэтому, если оно получается громоздким, нужно использовать оператор simplify.

$$\frac{d^2}{dx^2}\frac{x^2+3x+1}{x^3-5x-6}\ \text{simplify} \to 2\cdot\frac{-29+x^6+21\cdot x^4+9\cdot x^5+18\cdot x+93\cdot x^2+57\cdot x^3}{\left(x^3-5\cdot x-6\right)^3}$$

Пример нахождения производной высокой степени от функции с неявным параметром.

$$\frac{d^{10}}{dx^{10}}e^{n\cdot x} \to n^{10}\cdot e^{n\cdot x}$$

Численное определение значения производной второго порядка:

$$x := 0 \qquad y := 1$$

$$\frac{d^2}{dx^2}\ln(y-x) = -1$$

Аналогично обычным производным, производные высших порядков можно вычислить как аналитически, так и численно. Однако при численном дифференцировании порядок производной не может превышать 5 (при попытке задать больший порядок программа выдаст соответствующее сообщение об ошибке). Причина данного ограничения связана с особенностями используемого Mathcad численного метода и заключается в том, что в ходе вычисления производной высокого порядка происходит накопление ошибок на каждом круге приближений. Поэтому чем выше порядок производной, тем меньше точность результата. Очевидно, что если потеря точности на каждый порядок дифференцирования составляет по одному десятичному порядку, а надежность используемого Mathcad численного метода для первой производной равна 7–8 знакам после запятой, то доверять результату расчета этим методом производной, например, восьмого порядка, ни в коем случае нельзя. Описанную проблему демонстрирует следующий пример.

Пример 11.8. Накопление ошибки по мере увеличения порядка производной при использовании численного метода дифференцирования

Производная любого порядка от e^x при x=1 равна e. Используем этот факт для проверки точности ответов.

$$x := 1$$

$$e - \frac{d^2}{dx^2}e^x = 3.317 \times 10^{-13} \qquad e - \frac{d^3}{dx^3}e^x = 5.348 \times 10^{-11}$$

$$e - \frac{d^4}{dx^4}e^x = 5.285 \times 10^{-11} \qquad e - \frac{d^5}{dx^5}e^x = -2.203 \times 10^{-7}$$

Из примера 11.8 видно, что при подсчете производной пятого порядка ошибка численного метода началась с 4-го знака после запятой. Это, в принципе, приемлемо для обычной точности в Mathcad, хотя для ряда специфических задач такое приближение может оказаться недопустимо грубым. Поэтому производные высоких порядков все же лучше вычислять символьно. Благо, при этом точность будет высокой, независимо от вида дифференцируемых функций и порядка производной. Единственным недостатком такого метода можно назвать то, что при вычислении производных высоких порядков приходится иногда работать с весьма громоздкими выражениями (но обычно их можно упростить, используя операторы simplify, collect, factor и expand).

При желании можно попробовать вычислить производную более пятого порядка и численным методом (это можно сделать в том случае, если особая точность вас не интересует). Для этого нужно просто последовательно ввести два (или несколько) оператора дифференцирования. Однако не рекомендую вам использовать данный ход, так как погрешность при этом может быть очень высокой.

Пример 11.9. Численное определение значения производной седьмого порядка

$$x := 1$$

$$e - \frac{d^5}{dx^5}\frac{d^2}{dx^2}e^x = 9.969 \times 10^{-5}$$

11.1.3. Частные производные

В том случае, если производная вычисляется для функции нескольких переменных, она называется частной. Никаких принципиальных отличий между заданием простой и частной производных нет. Единственное, важным моментом является то, каким образом можно преобразовать оператор простого дифференцирования к виду частной производной. Чтобы это сделать, нужно щелкнуть правой кнопкой мыши на операторе производной. При этом откроется контекстное меню, в котором следует выбрать список View Derivative As (Видеть производную как). В появившемся подменю установите флажок Partial Derivative (Частная производная). Если вам понадобится вернуться к стандартному виду оператора дифференцирования, то в том же меню выберите пункт Derivative.

Можно задавать в Mathcad и частные производные высших порядков. Правда, определенные сложности могут возникнуть со смешанными производными. Дело в том, что традиционную форму их представления Mathcad не поддерживает. Однако задать смешанные производные все же можно, поместив в маркер выражения одного оператора дифференцирования другой оператор. В большинстве случаев порядок взятия производных при вычислении смешанной производной не влияет на результат, поэтому объединять их можно в любой последовательности. Различия могут возникнуть, лишь если смешанные производные не являются непрерывными.

Пример 11.10. Найти все частные производные второго порядка для функции $f(x,y,z)=x^2 \cdot e^{x+z}-y^3$

Задаем функцию:

$$f(x,y,z) := x^2 \cdot e^{x+z} - y^3$$

Находим несмешанные частные производные второго порядка:

$$\frac{\partial^2}{\partial x^2} f(x,y,z) \text{ factor } \rightarrow e^{x+z} \cdot \left(2 + 4 \cdot x + x^2\right) \qquad \frac{\partial^2}{\partial y^2} f(x,y,z) \rightarrow -6 \cdot y$$

$$\frac{\partial^2}{\partial z^2} f(x,y,z) \rightarrow x^2 \cdot e^{x+z}$$

Определяем смешанные производные. Из каждой пары производных достаточно найти одну – у второй значение будет таким же (так как условие непрерывности соблюдается).

$$\frac{\partial^2}{\partial x^2}\frac{\partial}{\partial y} f(x,y,z) \rightarrow 0 \qquad \frac{\partial}{\partial y}\frac{\partial^2}{\partial z^2} f(x,y,z) \rightarrow 0 \qquad \frac{\partial}{\partial x}\frac{\partial}{\partial y}\frac{\partial}{\partial z} f(x,y,z) \rightarrow 0$$

$$\frac{\partial}{\partial x}\frac{\partial^2}{\partial z^2} f(x,y,z) \text{ factor } \rightarrow x \cdot e^{x+z} \cdot (2 + x)$$

11.2. Задачи, связанные с вычислением производной

Задач, в основе решения которых лежит вычисление производных, имеется немало. Некоторые из них особенно важны для практики, и поэтому они встречаются в любом

задачнике по математическому анализу. Это, прежде всего, задачи, связанные с исследованием функций: поиском экстремумов, перегибов, асимптот. В этом разделе мы не будем разбирать примеры такого рода, так как описанию особенностей их решения посвящена гл. 13. Сейчас же мы обсудим, как решаются задачи так называемой дифференциальной геометрии (построение касательных к линиям и касательных плоскостей к поверхностям, нормалей, огибающих), а также задачи, связанные с дифференцированием сложных функций.

11.2.1. Построение касательной и нормали к плоской кривой

Построить касательную к кривой очень просто, если знать геометрический смысл производной — это тангенс угла наклона касательной к оси X. То есть производной соответствует коэффициент k в уравнении касательной y(x)=k·x+b. Коэффициент же b легко найти из условия y(0)=b. Путем несложных преобразований можно заключить, что касательная к кривой функции f(x) в точке $M(x_0,y_0)$ задается следующим уравнением:

$$y(x) = y_0 + \left(\frac{d}{dx}f(x_0)\right)\cdot(x - x_0)$$

Уравнение нормали очень просто вывести из уравнения касательной, если догадаться, что при повороте на 90° касательная переходит в нормаль. Запишем уравнение касательной через тангенс ее наклона к оси X:

$$y(x) = y_0 + \tan(\alpha)\cdot(x - x_0)$$

Очевидно, что уравнение нормали тогда будет выглядеть следующим образом:

$$y(x) = y_0 + \tan\left(\alpha + \frac{\pi}{2}\right)\cdot(x - x_0)$$

Так как tg(a+p/2)=–ctg(a), а ctg(a)=1/tg(a), то окончательно имеем:

$$y(x) = y_0 - \frac{1}{\frac{d}{dx}f(x_0)}\cdot(x - x_0)$$

Задачи на построение касательных обычно довольно несложные. Типичная из них рассматривается в примере 11.11.

Пример 11.11. Составить уравнение касательной и нормали к линии, заданной уравнением $y(x)=x^4-3\cdot x^3+4\cdot x^2-5\cdot x+1$ в точке M(0, 1)

Задаем переменные с координатами точки M и функцию линии:

$$x_0 := 0 \qquad y_0 := 1$$

$$y(x) := x^4 - 3x^3 + 4x^2 - 5x + 1$$

Определяем выражение производной и задаем ее как функцию:

$$\frac{d}{dx}y(x) \to 4\cdot x^3 - 9\cdot x^2 + 8\cdot x - 5$$

$$y'(x) := 4\cdot x^3 - 9\cdot x^2 + 8\cdot x - 5$$

Находим уравнения касательной и нормали по приведенным выше формулам:

$$tang(x) := y_0 + y'(x_0)\cdot(x - x_0) \qquad norm(x) := y_0 - \frac{1}{y'(x_0)}\cdot(x - x_0)$$

$$tang(x) \to 1 - 5\cdot x \qquad norm(x) \to 1 + \frac{1}{5}\cdot x$$

Строим график (рис. 11.2).

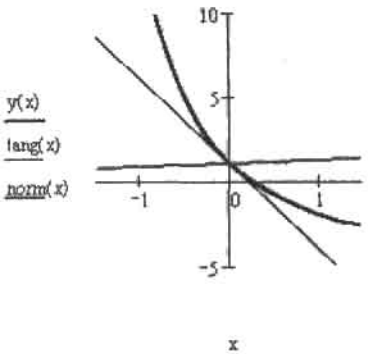


Рис. 11.2. Кривая с касательной и нормалью (из-за неравного масштаба осей график несколько искажен, поэтому угол между касательной и нормалью кажется меньше 90°)

Столь элементарно уравнения для нормали и касательной можно найти лишь в случае явно заданной кривой. Если же кривая описывается неявным уравнением, разделить переменные в котором невозможно, или же ее задает система параметрических уравнений, то построение нормали и касательной значительно усложняется. Покажем на примере, как можно решить задачу такого рода.

Пример 11.12. Написать уравнение касательной, имеющей общую точку с кривой при x=1. Кривая задается следующими параметрическими уравнениями:

$$x = \frac{3t}{1 + t^3} \qquad y = \frac{3\cdot t^2}{1 + t^3}$$

Начнем мы решение этой задачи с того, что зададим x и y как функции от t:

$$x(t) := \frac{3t}{1 + t^3} \qquad y(t) := \frac{3t^2}{1 + t^3}$$

В математическом анализе доказывается, что если функция Y=f(x) задана параметрически с помощью уравнений x=x(t) и y=y(t), где x(t), y(t) — дифференцируемые функции, то производная Y по x находится по следующей формуле (естественно, что производная от x(t) не должна быть равна 0):

$$\frac{d}{dx}Y(x) = \frac{\frac{d}{dt}y(t)}{\frac{d}{dt}x(t)}$$

Используя данную формулу, задаем функцию производной:

$$\frac{\frac{d}{dt}y(t)}{\frac{d}{dt}x(t)}\ \text{simplify} \rightarrow t\cdot\frac{-2+t^3}{-1+2\cdot t^3} \qquad Y'_x(t) := t\cdot\frac{-2+t^3}{-1+2\cdot t^3}$$

Нам нужно построить касательную в точке x=1. Но кривая задана параметрическими уравнениями, зависящими от t. От t зависит и функция производной. Следовательно, нам нужно найти, какое t соответствует x, равному 1. Для этого следует решить уравнение x(t)=1. Проще это сделать аналитически:

$$R := \frac{3t}{1+t^3} = 1 \left|\begin{matrix}\text{solve},t\\ \text{float},5\end{matrix}\right. \rightarrow \begin{pmatrix}1.5321+.2e{-}4\cdot i\\ -1.8795\\ .34735-.2e{-}4\cdot i\end{pmatrix}$$

Итак, мы получили три корня, два из которых комплексные. Естественно, что t может быть только действительной, поэтому первой мыслью будет отбросить комплексные корни. Однако обратите внимание на то, что порядок мнимой части корней находится на уровне точности расчета. А может, ее существование обусловлено только погрешностью? Чтобы проверить это предположение, увеличим точность до 10 знаков (заменив значение в правом маркере оператора float). При этом порядок мнимой части в первом корне уменьшится до уровня 10^{-10}, в третьем же корне она исчезнет вовсе. Это означает, что мнимая часть — это действительно просто погрешность, возникающая при пересчете аналитических выражений в десятичные дроби. Следовательно, учитывать ее не нужно. Чтобы избавиться от мнимой части, используем функцию Re, выделяющую из комплексного числа действительную часть:

$$t1 := \mathrm{Re}(R_0) \qquad t2 := R_1 \qquad t3 := \mathrm{Re}(R_2)$$

Мы получили три значения t, соответствующие x=1. Но как такое возможно? Очень просто. Параметрически заданные функции могут иметь несколько точек для одного значения аргумента, что исключено в случае явно заданных функций. Собственно, поэтому для описания сложных кривых и применяют параметрическую форму уравнений. Таким образом, наша кривая имеет три точки для x=1. Убедиться в этом можно, построив график. Следовательно, мы должны найти три уравнения касательных (в качестве переменной используем z, так как выше уже была задана функция с именем x):

$$tang1(z) := y(t1) + Y'_x(t1)\cdot(z - x(t1)) \qquad tang2(z) := y(t2) + Y'_x(t2)\cdot(z - x(t2))$$

$$tang3(z) := y(t3) + Y'_x(t3)\cdot(z - x(t3))$$

Приближенно находим, какой вид будет иметь каждое уравнение:

$$tang1(z)\ \text{float},3 \rightarrow 1.14+.395\cdot z \qquad tang2(z)\ \text{float},3 \rightarrow -.742-1.14\cdot z$$

$$tang3(z)\ \text{float},3 \rightarrow -.395+.742\cdot z$$

Строим графики кривой и касательных (рис. 11.3).

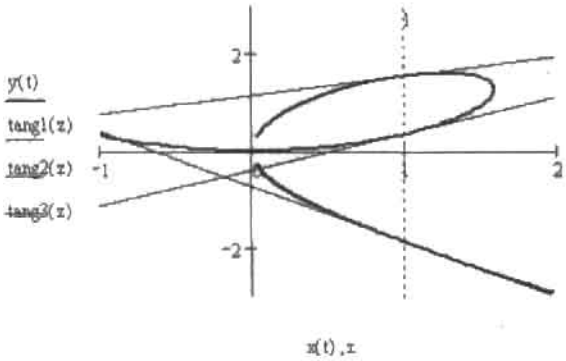


Рис. 11.3. Параметрическая кривая и касательные к ней

11.2.2. Построение касательной плоскости и нормали к поверхности

Одной из самых интересных областей применения частных производных является решение задач аналитической геометрии. В любом практикуме по высшей математике есть примеры на нахождение касательных плоскостей и нормалей к поверхностям, а также касательных линий и нормальных плоскостей к пространственным линиям. Ввиду наличия готовых формул данные задачи довольно просты с математической точки зрения. Однако их решение зачастую связано с подсчетом большого количества частных производных и сложных определителей, поэтому в чисто вычислительном плане они могут быть крайне трудоемкими. Значительную часть работы при решении таких задач может взять на себя Mathcad. В этом подразделе мы покажем, как решаются задачи подобного типа на примере построения касательной плоскости и нормали к сфере.

Сложность задач на нахождение уравнений касательной плоскости и нормали очень сильно зависит от того, уравнениями в какой форме задается поверхность. Если поверхность описывается явным (например, параболоид $z(x,y)=x^2+y^2$) или неявным (к примеру, шар $x^2+y^2+z^2=R^2$) уравнением, то рассчитать уравнения касательной плоскости и нормали несложно даже на бумаге. Однако задача становится на порядок труднее, если поверхность описывает система параметрических уравнений. А так как наиболее интересные поверхности обычно задаются только в параметрической форме, то подобные задачи не являются редкостью. В Mathcad же параметрическая форма описания поверхностей является основной, так как зачастую только исходя из нее можно строить качественные графики. В нижележащем примере показано, как можно найти уравнения касательной плоскости и нормали для параметрически заданной поверхности. Если вы разберетесь с данным примером, то вы с легкостью сможете решать задачи, в которых поверхность описывается явным или неявным уравнением.

Пример 11.13. Имеется сфера с радиусом R=5 и центром в начале координат. Построить касательные плоскости, проходящие через точки сферы, которым соответствуют значения аргументов x=2, y=3, а также нормали к этим точкам

Сферу можно описать неявным уравнением $x^2+y^2+z^2=R^2$. Однако работать с таким уравнением в Mathcad неудобно. Решать поставленную задачу мы будем, задав сферу параметрическими уравнениями:

$$R := 5$$

$$x(u,v) := R\cdot\cos(u)\cdot\cos(v)$$

$$y(u,v) := R\cdot\cos(u)\cdot\sin(v)$$

$$z(u,v) := R\cdot\sin(u)$$

Нам известны координаты x и y точек сферы, через которые нужно провести касательные плоскости и нормали. Однако параметрические уравнения зависят от переменной u (азимутальный угол, изменяется от $-\pi/2$ до $\pi/2$) и v (полярный угол, изменяется от 0 до $2\cdot p$). Нам нужно найти, какие значения u и v соответствуют данным значениям x и y. Для этого решим систему уравнений x(u,v)=2, y(u,v)=3. Так как система довольно несложная, это можно попробовать сделать аналитически:

$$R := \begin{pmatrix} R\cdot\cos(u)\cdot\cos(v) = 2 \\ R\cdot\cos(u)\cdot\sin(v) = 3 \end{pmatrix} \text{solve},u,v \rightarrow \begin{pmatrix} \text{acos}\left(\frac{1}{5}\cdot 13^{\frac{1}{2}}\right) & \text{atan}\left(\frac{3}{2}\right) \\ \pi - \text{acos}\left(\frac{1}{5}\cdot 13^{\frac{1}{2}}\right) & \text{atan}\left(\frac{3}{2}\right) - \pi \end{pmatrix}$$

Mathcad нашел две пары u и v, удовлетворяющих системе. Это корректное решение: несложно прикинуть, что две точки сферы радиусом R=5 с центром в начале координат будут иметь координаты x=2, y=3. Однако нельзя слепо доверять выданному ответу. Во-первых, значение полярного угла для обеих точек должно быть одинаковым. В ответе же для v получены разные значения. Очевидно, что в дальнейших расчетах следует использовать только значение v=arctg(3/2), так как оно соответствует точке (2, 3), в то время как v=arctg(3/2)–p описывает точку (–2, –3). Во-вторых, нужно учесть, что отсчет азимутального угла u ведется не с 0, а с $-\pi/2$. Для этого $\pi/2$ нужно отнять от полученных значений u.

Введем переменные u1, v1, z1 и u2, v2, z2, в которых будут храниться значения координат, соответствующих точкам, через которые должны быть проведены касательные плоскости и нормали:

$$u1 := R_{0,0} - \frac{\pi}{2} \qquad v1 := R_{0,1} \qquad u2 := R_{1,0} - \frac{\pi}{2} \qquad v2 := R_{0,1}$$

$$z1 := z(u1, v1) \qquad z2 := z(u2, v2)$$

$$z1 = -3.606 \qquad z2 = 3.606$$

Координаты по z для точек сферы с одинаковыми координатами по x и y оказались, как и положено, равными по модулю и противоположными по знаку. Следовательно, u и v для этих точек были найдены верно.

Уравнение касательной плоскости для параметрически заданной поверхности можно найти, раскрыв следующий определитель и выделив Z в левую часть равенства:

$$\left|\begin{pmatrix} X - x_0 & Y - y_0 & Z - z_0 \\ x_u & y_u & z_u \\ x_v & y_v & z_v \end{pmatrix}\right| = 0$$

Здесь X, Y, Z — переменные уравнения плоскости; x_0, y_0, z_0 — координаты точки, в которой поверхность и плоскость соприкасаются; x_u, y_u, z_u — значения частных производных параметрических уравнений по переменной u в точке соприкосновения; x_v, y_v, z_v — значения частных производных параметрических уравнений по v.

Рассчитать приведенный определитель в Mathcad несложно. Но тут есть одна тонкость. Мы должны использовать в нем значения частных производных параметрических уравнений в точке соприкосновения касательной плоскости и поверхности, а не их аналитические выражения. Имеется несколько способов решить эту проблему. Мы будем действовать следующим образом: вычислим определитель с символьными выражениями для частных производных, а затем подставим вместо переменных u и v их значения с помощью оператора substitute. Далее мы выразим Z посредством оператора solve и пересчитаем полученное громоздкое аналитическое выражение в более простую приблизительную форму, применив оператор float.

$$\left|\begin{pmatrix} X-2 & Y-3 & Z-z1 \\ \frac{\partial}{\partial u}x(u,v) & \frac{\partial}{\partial u}y(u,v) & \frac{\partial}{\partial u}z(u,v) \\ \frac{\partial}{\partial v}x(u,v) & \frac{\partial}{\partial v}y(u,v) & \frac{\partial}{\partial v}z(u,v) \end{pmatrix}\right| = 0 \left|\begin{array}{l} \text{substitute}, u = u1, v = v1 \\ \text{solve}, Z \\ \text{float}, 3 \end{array}\right. \rightarrow .533\cdot X - 7.06 + .800\cdot Y$$

На основании полученного выражения создаем функцию:

$$Z1(X,Y) := .533\cdot X - 7.06 + .800\cdot Y$$

Обратите внимание, что в уравнении касательной плоскости мы использовали переменные X, Y, Z. Ввиду наличия для идентификаторов в Mathcad чувствительности к регистру, это позволяет избежать конфликта с определенными выше параметрическими функциями x(t), y(t), z(t).

Аналогичным образом находим уравнение касательной плоскости и для второй точки:

$$\left|\begin{pmatrix} X-2 & Y-3 & Z-z2 \\ \frac{\partial}{\partial u}x(u,v) & \frac{\partial}{\partial u}y(u,v) & \frac{\partial}{\partial u}z(u,v) \\ \frac{\partial}{\partial v}x(u,v) & \frac{\partial}{\partial v}y(u,v) & \frac{\partial}{\partial v}z(u,v) \end{pmatrix}\right| = 0 \left|\begin{array}{l} \text{substitute}, u = u2, v = v2 \\ \text{solve}, Z \\ \text{float}, 3 \end{array}\right. \rightarrow -.533\cdot X + 7.06 - .800\cdot Y$$

$$Z2(X,Y) := -.533\cdot X + 7.06 - .800\cdot Y$$

Уравнения касательных найдены. Осталось вычислить уравнения для нормалей. В справочной литературе можно найти следующую формулу для нормали к параметрически заданной поверхности:

$$\frac{X - x_0}{\left|\begin{pmatrix} y_u & z_u \\ y_v & z_v \end{pmatrix}\right|} = \frac{Y - y_0}{\left|\begin{pmatrix} z_u & x_u \\ z_v & x_v \end{pmatrix}\right|} = \frac{Z - z_0}{\left|\begin{pmatrix} x_u & y_u \\ x_v & y_v \end{pmatrix}\right|}$$

Здесь X, Y, Z — переменные уравнения; x_0, y_0, z_0 — координаты общей точки поверхности и нормали; x_u, y_u, z_u и x_v, y_v, z_v — значения частных производных параметрических уравнений по u и v в точке пересечения нормалью поверхности.

От данной формулы очень просто перейти к системе параметрических уравнений X(t), Y(t), Z(t), задающих нормаль (это нужно сделать, так как только линии, описанные в такой форме, можно построить в Mathcad). Для этого выражение для каждой переменной нужно приравнять к параметру t и выразить затем через него саму переменную.

Чтобы не оперировать громоздкими выражениями, сначала рассчитаем входящие в формулу определители:

$$Dl_x := \left| \begin{pmatrix} \frac{\partial}{\partial u} y(u,v) & \frac{\partial}{\partial u} z(u,v) \\ \frac{\partial}{\partial v} y(u,v) & \frac{\partial}{\partial v} z(u,v) \end{pmatrix} \right| \left| \begin{array}{l} \text{substitute}, u = u1, v = v1 \\ \text{simplify} \end{array} \right. \rightarrow \frac{-24}{13} \cdot 13^{\frac{1}{2}}$$

$$Dl_y := \left| \begin{pmatrix} \frac{\partial}{\partial u} z(u,v) & \frac{\partial}{\partial u} x(u,v) \\ \frac{\partial}{\partial v} z(u,v) & \frac{\partial}{\partial v} x(u,v) \end{pmatrix} \right| \left| \begin{array}{l} \text{substitute}, u = u1, v = v1 \\ \text{simplify} \end{array} \right. \rightarrow \frac{-36}{13} \cdot 13^{\frac{1}{2}}$$

$$Dl_z := \left| \begin{pmatrix} \frac{\partial}{\partial u} x(u,v) & \frac{\partial}{\partial u} y(u,v) \\ \frac{\partial}{\partial v} x(u,v) & \frac{\partial}{\partial v} y(u,v) \end{pmatrix} \right| \left| \begin{array}{l} \text{substitute}, u = u1, v = v1 \\ \text{simplify} \end{array} \right. \rightarrow 2 \cdot 13^{\frac{1}{2}} \cdot 3^{\frac{1}{2}}$$

Функция CreateSpace, служащая в Mathcad для построения пространственных параметрически заданных кривых (подробно она описана в гл. 6), принимает систему уравнений, описывающих линию, в виде вектора. Поэтому имеет смысл получить этот вектор сразу, для чего объединим уравнения в систему и решим их одновременно:

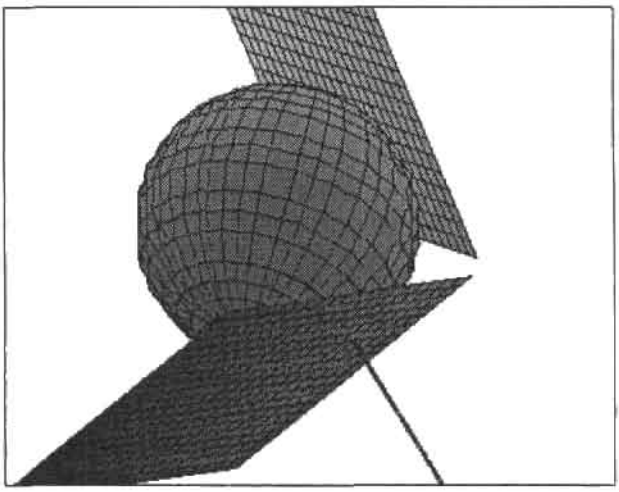
$$\begin{pmatrix} \frac{X-2}{Dl_x} = t \\ \frac{Y-3}{Dl_y} = t \\ \frac{Z-z1}{Dl_z} = t \end{pmatrix} \left| \begin{array}{l} \text{solve}, X, Y, Z \\ \text{float}, 3 \end{array} \right. \rightarrow (2.01 - 6.67 \cdot t \;\; 3.01 - 10.0 \cdot t \;\; -3.60 + 12.5 \cdot t)$$

Транспонируем выданную solve матрицу-строку и создаем на основании нее функцию:

$$norml(t) := (2.01 - 6.67 \cdot t \;\; 3.01 - 10.0 \cdot t \;\; -3.60 + 12.5 \cdot t)^T$$

Аналогично получаются уравнения нормали для второй точки. В целях экономии места приводить соответствующие формулы не будем.

Строим шар, касательные плоскости и нормаль (рис. 11.4) (чтобы понять, как была нарисована линия нормали, перечитайте раздел гл. 6, посвященный функции CreateSpace).



(x,y,z),Z1,Z2,CreateSpace(norml,-0.5,0.3,100)

Рис. 11.4. Шар, касательные плоскости и нормаль к нему

11.2.3. Дифференцирование сложной функции

Дифференцируемая функция $U(v_1, v_2, v_3 \ldots v_n)$ называется сложной, если ее аргументы в свою очередь являются функциями некоторых независимых переменных: $v1=f(x_1, x_2, x_3 \ldots x_m)$. Очень часто на практике возникает задача нахождения частных производных сложной функции по переменным, от которых зависят ее аргументы. Например, перед вами может стать задача дифференцирования функции, заданной в сферической системе координат, по переменным x, y, z. В любом задачнике есть примеры на перевод оператора Лапласа или оператора градиента из декартовой системы координат в цилиндрическую и сферическую системы. Подобного рода задачи требуют весьма объемных математических выкладок и хорошей техники вычислений — поэтому стоит использовать Mathcad для их решения.

Рассмотрим функцию, заданную в сферической системе координат. Она зависит от трех переменных, каждая из которых в свою очередь является функцией декартовых координат:

$$U = f(\rho(x, y, z), \theta(x, y, z), \phi(x, y, z))$$

Частные производные U по x, y, z можно найти по следующим формулам:

$$\frac{\partial}{\partial x}u = \left(\frac{\partial}{\partial \phi}u\right)\cdot\left(\frac{\partial}{\partial x}\phi\right) + \left(\frac{\partial}{\partial \rho}u\right)\cdot\left(\frac{\partial}{\partial x}\rho\right) + \left(\frac{\partial}{\partial \theta}u\right)\cdot\left(\frac{\partial}{\partial x}\theta\right)$$

$$\frac{\partial}{\partial y}u = \left(\frac{\partial}{\partial \phi}u\right)\cdot\left(\frac{\partial}{\partial y}\phi\right) + \left(\frac{\partial}{\partial \rho}u\right)\cdot\left(\frac{\partial}{\partial y}\rho\right) + \left(\frac{\partial}{\partial \theta}u\right)\cdot\left(\frac{\partial}{\partial y}\theta\right)$$

$$\frac{\partial}{\partial z}u = \left(\frac{\partial}{\partial \phi}u\right)\cdot\left(\frac{\partial}{\partial z}\phi\right) + \left(\frac{\partial}{\partial \rho}u\right)\cdot\left(\frac{\partial}{\partial z}\rho\right) + \left(\frac{\partial}{\partial \theta}u\right)\cdot\left(\frac{\partial}{\partial z}\theta\right)$$

То есть чтобы найти частную производную сложной функции по одной из независимых переменных, нужно вычислить произведения частных производных этой функции по промежуточным аргументам и частных производных промежуточных аргументов по независимым переменным, а затем сложить их.

Решение неэлементарных задач на дифференцирование сложной функции требует активного использования возможностей символьного процессора. Однако нужно помнить, что они весьма ограничены, поэтому, к примеру, найти в пару действий лапласиан для сферической системы координат вряд ли получится. Запомните несколько советов, которые позволят вам проводить аналитические расчеты в Mathcad более эффективно.

- Не обращайте внимания на сообщения об ошибках. Если формула стала красной, это не значит, что она не будет вычислена. Просто среда разработки настроена на синтаксис численных расчетов, поэтому ее «смущает», если, например, функция задается без скобок с аргументами после имени. Также сообщение об ошибке появляется, если в выражении присутствуют не заданные численно переменные, хотя аналитический процессор «умеет» отлично работать с такими выражениями.
- Можно работать лишь с реальными, а не абстрактными объектами. К примеру, нельзя проводить аналитические преобразования, оперируя некоей гипотетической, обобщенной функцией, как это обычно делается при выводах на бумаге. Функция должна быть задана как конкретная зависимость, иначе символьный процессор с ней работать не будет. Аналогично, нельзя оперировать «пустыми» операторами дифференцирования или отдельными дифференциалами.
- Применяйте уникальные идентификаторы, если одна и та же величина выступает в разных ипостасях. Поясним этот принцип на примере. В приведенные выше выражения для частных производных сложных функций по независимым переменным входят произведения, в которых промежуточный аргумент сначала фигурирует, как переменная, а затем как функция. Для человека разница ролей, которые исполняет в данном случае промежуточный аргумент, очевидна. Однако машина лишена интеллекта! Для нее величина — это или переменная, или функция. Поэтому, чтобы использовать данные формулы в Mathcad, промежуточный аргумент как переменная и как функция должен иметь различные идентификаторы (например, x и X или y и y′).

Для примера мы решим довольно сложную с вычислительной точки зрения задачу: найдем лапласиан для функции, заданной в сферической системе координат, при условии, что известна лишь формула оператора Лапласа для декартовых координат:

$$\Delta u = \frac{\partial^2}{\partial x^2}u + \frac{\partial^2}{\partial y^2}u + \frac{\partial^2}{\partial z^2}u$$

Оператор Лапласа довольно активно используется в физике. В частности, он входит в гамильтониан — оператор полной энергии, поэтому задачи, подобные решаемой ниже, не редкость в классической и особенно квантовой механике. Уравнение Лапласа также применяется для описания полей со сферической симметрией.

Пример 11.14. Найти лапласиан функции $u(\rho, \theta, \phi)=\rho^2 \cdot \sin(\theta) \cdot \cos(\phi)$ в сферической системе координат

Начнем мы с того, что зададим саму функцию u:

$$u := \rho^2 \cdot \sin(\theta) \cdot \cos(\phi)$$

Далее введем выражения, связывающие декартовы координаты x, y, z со сферическими координатами ρ, θ, ϕ:

$$X := \rho \cdot \sin(\theta) \cdot \cos(\phi)$$

$$Y := \rho \cdot \sin(\theta) \cdot \sin(\phi)$$

$$Z := \rho \cdot \cos(\theta)$$

Для обозначения декартовых координат мы использовали буквы в верхнем регистре. Это необходимо, чтобы система могла «различить» случаи, когда они выступают как переменные (при этом мы будем обозначать их символами в нижнем регистре) и как функции от сферических координат.

Функцию $u(\rho, \theta, \phi)$ мы можем представить как сложную, в которой промежуточными аргументами являются ρ, θ, ϕ, а независимыми переменными — x, y, z. Используя приведенные выше формулы, мы найдем частные производные $u(\rho, \theta, \phi)$ по декартовым координатам. Повторное дифференцирование полученных функций как сложных даст входящие в выражение лапласиана частные производные второго порядка по x, y и z.

Итак, в первую очередь мы должны представить сферические координаты как функции от координат декартовых. Для начала попробуем объединить уравнения, их связывающие, в систему и решить ее с помощью оператора solve относительно ρ, θ и ϕ. Однако полученный при этом результат окажется мало приемлемым, так как в нем будет использоваться функция atan2, не имеющая аналогов в «бумажной» математике. Следовательно, выразить ρ, θ и ϕ через x, y и z мы должны самостоятельно.

Чтобы найти выражение для ρ, возведем все три уравнения в квадрат и сложим их:

$$X^2 + Y^2 + Z^2 \text{ simplify } \rightarrow \rho^2$$

Таким образом, r зависит от декартовых координат следующим образом:

$$\rho' := \sqrt{x^2 + y^2 + z^2}$$

Так как идентификаторы r как переменной и как функции от x, y, z должны различаться, мы использовали обозначение «ρ'».

Формулу для ϕ легко найти, разделив выражение для y на выражение для x:

$$\frac{Y}{X} \text{ simplify } \rightarrow \frac{\sin(\phi)}{\cos(\phi)}$$

Отсюда:

$$\phi' := \operatorname{atan}\left(\frac{y}{x}\right)$$

Формулу для θ находим, выразив данную переменную из выражения для Z (учтя полученную ранее формулу для ρ):

$$\theta' := \operatorname{acos}\left(\frac{z}{\sqrt{x^2 + y^2 + z^2}}\right)$$

Уже известны все величины для того, чтобы можно было найти частные производные $u(\rho, \theta, \phi)$ по x, y, z:

$$u'_x := \left(\frac{\partial}{\partial \phi} u\right)\cdot\left(\frac{\partial}{\partial x} \phi'\right) + \left(\frac{\partial}{\partial \rho} u\right)\cdot\left(\frac{\partial}{\partial x} \rho'\right) + \left(\frac{\partial}{\partial \theta} u\right)\cdot\left(\frac{\partial}{\partial x} \theta'\right)$$

$$u'_y := \left(\frac{\partial}{\partial \phi} u\right)\cdot\left(\frac{\partial}{\partial y} \phi'\right) + \left(\frac{\partial}{\partial \rho} u\right)\cdot\left(\frac{\partial}{\partial y} \rho'\right) + \left(\frac{\partial}{\partial \theta} u\right)\cdot\left(\frac{\partial}{\partial y} \theta'\right)$$

$$u'_z := \left(\frac{\partial}{\partial \phi} u\right)\cdot\left(\frac{\partial}{\partial z} \phi'\right) + \left(\frac{\partial}{\partial \rho} u\right)\cdot\left(\frac{\partial}{\partial z} \rho'\right) + \left(\frac{\partial}{\partial \theta} u\right)\cdot\left(\frac{\partial}{\partial z} \theta'\right)$$

Теперь данные выражения нужно аналитически вычислить. При этом необходимо учесть, что в выданные в качестве ответа выражения могут входить как сферические координаты, так и декартовы. Последние нужно заменить, используя заданные выше выражения перехода.

$$u'_x := u'_x \left|\begin{array}{l}\text{substitute}, x = X, y = Y, z = Z\\ \text{assume}, \rho \geq 0, \theta = \text{RealRange}(0, \pi) \rightarrow\\ \text{simplify}\end{array}\right.$$

$$\rightarrow \rho\cdot\left(1 - \cos(\phi)^2 + 2\cdot\sin(\theta)^2\cdot\cos(\phi)^2 + \cos(\phi)^2\cdot\cos(\theta)^2\right)$$

$$u'_y := u'_y \left|\begin{array}{l}\text{substitute}, x = X, y = Y, z = Z\\ \text{assume}, \rho \geq 0, \theta = \text{RealRange}(0, \pi) \rightarrow\\ \text{simplify}\end{array}\right.$$

$$\rightarrow \cos(\phi)\cdot\sin(\phi)\cdot\rho\cdot\left(-1 + 2\cdot\sin(\theta)^2 + \cos(\theta)^2\right)$$

$$u'_z := u'_z \left|\begin{array}{l}\text{substitute}, x = X, y = Y, z = Z\\ \text{assume}, \rho \geq 0, \theta = \text{RealRange}(0, \pi) \rightarrow\\ \text{simplify}\end{array}\right.$$

$$\rightarrow \rho\cdot\cos(\phi)\cdot\cos(\theta)\cdot\frac{-1 + 2\cdot\sin(\theta)^2 + \cos(\theta)^2}{\sin(\theta)}$$

Чтобы максимально упростить результат аналитического расчета, следует задействовать оператор simplify. Однако по умолчанию при этом в ответы войдут функции знака выражения – csgn (знак комплексного выражения) и signum (знак действительного выражения). Это происходит из-за того, что в ряде алгебраических преобразований знак должен учитываться. Чтобы убрать данные функции, символьному процессору нужно показать, в каких пределах изменяются переменные. Сделать это можно с помощью оператора assume. Анализ выражений ответа показывает, что в качестве аргументов функций знаков выступают ρ и $\sin(\theta)$. Однако ρ может быть только

больше или равна 0 (см. уравнение, связывающее ρ и декартовы координаты). То же присуще и sin(θ), так как азимутальный угол в случае использованной формы задания сферической системы координат изменяется от 0 до π. Используя операторы неравенства и модификатор RealRange, «сообщаем» эти данные аналитическому процессору. При этом функции знака исчезнут.

В той форме, в которой мы получили частные производные функции u(ρ, θ, ϕ) по x, y, z, они также являются сложными функциями. Чтобы вычислить частные производные второго порядка, дифференцируем их, используя уже хорошо знакомые формулы:

$$u''_x := \left(\frac{d}{d\phi}u'_x\right)\cdot\left(\frac{d}{dx}\phi'\right) + \left(\frac{d}{d\rho}u'_x\right)\cdot\left(\frac{d}{dx}\rho'\right) + \left(\frac{d}{d\theta}u'_x\right)\cdot\left(\frac{d}{dx}\theta'\right)$$

$$u''_y := \left(\frac{d}{d\phi}u'_y\right)\cdot\left(\frac{d}{dy}\phi'\right) + \left(\frac{d}{d\rho}u'_y\right)\cdot\left(\frac{d}{dy}\rho'\right) + \left(\frac{d}{d\theta}u'_y\right)\cdot\left(\frac{d}{dy}\theta'\right)$$

$$u''_z := \left(\frac{d}{d\phi}u'_z\right)\cdot\left(\frac{d}{dz}\phi'\right) + \left(\frac{d}{d\rho}u'_z\right)\cdot\left(\frac{d}{dz}\rho'\right) + \left(\frac{d}{d\theta}u'_z\right)\cdot\left(\frac{d}{dz}\theta'\right)$$

Чтобы найти лапласиан, аналитически вычисляем сумму частных производных второго порядка. Так как в выражение ответа могут входить декартовы координаты, их нужно заменить соответствующими им выражениями в сферической системе. Чтобы исключить из ответа функции знака, показываем символьному процессору области изменения r и q.

$$u''_x + u''_y + u''_z \left|\begin{array}{l}\text{substitute},x = X, y = Y, z = Z\\ \text{simplify}\\ \text{assume},\rho \geq 0, \theta = \text{RealRange}(0,\pi)\\ \text{simplify}\end{array}\right. \rightarrow 4\cdot\cos(\phi)\cdot\sin(\theta)$$

Обратите внимание, что при расчете последнего выражения оператор simplify был задействован дважды. Это довольно эффективный ход — проводить упрощение не один раз, а после каждой операции — поэтому его нужно запомнить.

Полученный результат весьма компактен. Однако верен ли он? Чтобы проверить это, воспользуемся готовым выражением оператора Лапласа в сферической системе координат:

$$\Delta u := \frac{1}{\rho^2}\cdot\frac{\partial}{\partial\rho}\left(\rho^2\cdot\frac{\partial}{\partial\rho}u\right) + \frac{1}{\rho^2\cdot\sin(\theta)}\cdot\frac{\partial}{\partial\theta}\left(\sin(\theta)\cdot\frac{\partial}{\partial\theta}u\right) + \frac{1}{\rho^2\cdot\sin(\theta)^2}\cdot\frac{d^2}{d\phi^2}u$$

$$\Delta u \text{ simplify } \rightarrow 4\cdot\cos(\phi)\cdot\sin(\theta)$$

Ответы совпали. Следовательно, задача была решена верно.

11.3. Численные методы дифференцирования

В отличие от численного решения уравнений, о котором мы говорили в гл. 8, алгоритм численного дифференцирования, в общем-то, совсем не сложно построить и самостоятельно. Для этого достаточно вспомнить знакомую со школы формулу определения производной:

$$\frac{d}{dx}f(x) = \lim_{h \to 0} \frac{f(x+h) - f(h)}{h}$$

Очевидно, что, выбрав точку, производная в которой должна быть вычислена, и определившись с величиной приращения, по данной формуле вполне можно вести расчет. Столь же очевидно, что чем меньше величина h, тем точнее будет приближение производной. Впрочем, последнее утверждение не лишним будет и проверить. Для этого построим таблицу влияния размера шага на точность вычисления производной (табл. 11.1). Для того же, чтобы можно было контролировать величину ошибки, будем дифференцировать функцию в точке, значение производной в которой хорошо известно. Выберем для этого, например, синус в точке π. Точная величина производной для этого случая хорошо известна: –1.

$$f(x) := \sin(x)$$

$$D(a,h) := \frac{f(a+h) - f(a)}{h}$$

Таблица 11.1. Влияние величины приращения на точность численного дифференцирования

h	F(π+h)	D(π,h)	D(π,h) –1
0.1	-0.099833416646828	-0.998334166468282	1.66583353171768E-3
0.01	-9.99983333416633E-3	-0.999983333416645	1.66665833547519E-5
0.001	-9.99999833333109E-4	-0.999999833333231	1.66665833547519E-5
0.0001	-9.99999998334219E-5	-0.999999998335444	1.66665833547519E-5
0.00001	-9.99999999977639E-6	-0.999999999989884	1.01155750442672E-11
0.000001	-1.00000000001715E-6	-1.00000000013961	1.39611433525033E-10
0.0000001	-9.99999997138813E-8	-0.99999999836342	1.63658053775606E-9
0.00000001	-9.99999981676465E-9	-0.999999993922529	6.07747097092215E-9
0.000000001	-9.99999960279736E-10	-1.00000008274037	8.27403709990904E-8
0.000000000000001	0	-0.888178419700125	0.111821580299875

Анализ полученной таблицы дает довольно неожиданные результаты: оказывается, точность численного дифференцирования совершенно не обязательно тем выше, чем меньше выбранное приращение. При шаге 10^{-15} ошибка более чем в 100 раз выше, чем при максимальном приращении 10^{-1}. Максимум же точности получается при шаге величиной 10^{-5}. В чем же причина такой странной зависимости точности от приращения?

Действительно, в идеальном случае, чем меньше приращение h, тем точнее приближается производная. Однако при использовании компьютера возникает проблема, связанная с тем, что величины и функции вычисляются не абсолютно точно, а существует некоторая расчетная ошибка. Чем меньшее значение принимает функция, тем сильнее сказывается на относительной точности погрешность численных расчетов при малых h разность f(x+h)–f(x) столь невелика, что значение производной будет зависеть от округления весьма заметно. При h, близких по порядкам к точности округления системы, значение производной будет определяться, прежде всего, именно тем, что относительная ошибка при вычислении разности f(x+h)–f(x) сопоставима с самой разностью. Из-за этого и возникает такая на первый взгляд парадоксальная ситуация

значительной неточности определения производной при минимально возможном приращении.

В теории численных методов доказывается, что значение приращения, при котором достигается максимальное приближение к производной, является функцией порога округления системы. Так, для используемой нами формулы предела отношений приращений $h_{опт} \equiv e^{1/3}$, где e — порог точности округления. Так как e в Mathcad равняется 10^{-17}, то, очевидно, дифференцирование следует проводить при $h=10^{-5}$. Вывод этот полностью подтверждается полученной нами ранее таблицей. При $h=10^{-5}$ ошибка начинается только с 11-го знака после запятой, что является очень и очень неплохим приближением.

Для иллюстрации вышесказанного попробуем построить график зависимости величины ошибки от значения приращения. Так как и приращения, и ошибки различаются на много порядков, график следует строить в логарифмическом режиме по обеим шкалам (рис. 11.5).

$$D(a,h) := \frac{f(a+h) - f(a)}{h}$$

$$i := 0..1000 \qquad h_i := 10^{-15+1.5\cdot 10^{-2}\cdot i}$$

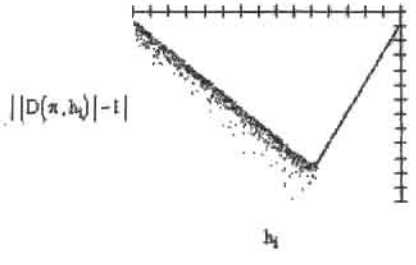


Рис. 11.5. График зависимости величины ошибки от значения приращения

Данный график подтверждает наличие максимума точности в окрестности точки приращения 10^{-5}. Помимо того, из него видно, насколько хаотично распределены точки значений ошибки при малых h. Данная хаотичность объясняется влиянием ошибки округления, имеющей, до определенной степени, случайный характер.

Существует довольно значительное количество формул, позволяющих более точно и легко, чем по методу конечных разностей, выполнять численное дифференцирование. Как правило, все они создавались с целью повышения точности вычислений при больших h. Например:

$$\frac{d}{dx} f(x) = \frac{-f(x+2\cdot h) - 8f(x-h) + 8f(x+h) + f(x-2\cdot h)}{12\cdot h}$$

Данная формула относится к группе формул центральных разностей, и в общем случае она на два порядка точнее рассмотренной нами ранее формулы конечной разности. Однако она еще более чувствительна к ошибкам округления, поэтому максимум ее точности также смещен влево и подчиняется условию $h \equiv e^{1/5}$. То есть лучших ре-

зультатов от данной формулы стоит ожидать при $h=10^{-3}$. Ошибка при этом составит $1.43995926293883\times10^{-13}$, что почти в 100 раз меньше, чем полученный нами наилучший результат по более простой формуле.

В Mathcad для численного определения значения производной в точке используется очень эффективный метод Риддера (Ridder). Об его основных идеях вы можете прочитать в справочной системе программы. Главное же, что о нем нужно знать, это то, что его точность не зависит от TOL или какой-то другой системной константы. Кстати, метод Риддера применяется для вычисления значения производной и реализации градиентных методов решения систем уравнений (блок Given-Find). Поэтому общие представления о нем могут помочь вам более эффективно использовать средства для численного решения уравнений.

Глава 12. Ряды и пределы

В системе Mathcad существует возможность проведения всех основных операций математического анализа: вычисления интегралов и производных, разложения в ряды Тейлора и Фурье, определения предела функции или последовательности, осуществления интегральных преобразований. Кроме того, с помощью специальных операторов можно весьма эффективно находить конечные и бесконечные суммы и произведения. Данная глава будет посвящена особенностям решения в Mathcad задач, связанных с пределами и рядами. Интегрирование и дифференцирование рассматриваются, ввиду объемности и сложности этих вопросов, в отдельных главах (см. гл. 10 и 11). Преобразования Лапласа и Фурье разбираются в гл. 14, посвященной решению дифференциальных уравнений.

12.1. Пределы

Вычисление пределов функций или последовательностей — одна из важнейших задач математического анализа. Такое вычисление порой весьма сложно с технической точки зрения, поэтому использование компьютера может значительно сэкономить силы и время. Тем более Mathcad решает задачи подобного рода весьма эффективно. Любой предел, который можно встретить в задачниках, будет подсчитан программой с легкостью, причем ей не придется «помогать», например, используя правило Лопиталя–Бернулли. Неплохо находятся и очень сложные пределы. Например, в Mathcad можно вычислить производную и определенный интеграл, основываясь только на их определениях (см. пример 12.1).

Чтобы найти предел, следует обратиться к панели Calculus (Вычисления). Данная панель содержит три вида операторов предела: предел в точке или двусторонний предел (Two-sided Limit) (также вводится сочетанием клавиш Ctrl+L), левосторонний предел (Limit from Below) (Ctrl+Shift+B), правосторонний предел (Limit from Above) (Ctrl+Shift+A).

Очень многие пределы вычисляются при условии стремления переменной к бесконечности. Символ бесконечности (Infinity) в Mathcad можно ввести либо с панели Calculus (Вычисления), либо сочетанием клавиш Ctrl+Shift+Z.

В качестве оператора вывода при вычислении пределов можно использовать только оператор символьного вывода «→». Если же вы введете оператор численного вывода «=», то будет выдано сообщение об ошибке.

В том случае, если система предел вычислить не может, в качестве ответа выдается само выражение с ключевым словом Limit и указанием на точку предела:

$$\lim_{n\to\infty}\sum_{k=1}^{n}\cos\left(\frac{x}{k^3}\right)\rightarrow \text{Limit}\left(\sum_{k=1}^{n}\cos\left(\frac{x}{k^3}\right), n\leftarrow\infty\right)$$

Если ошибка заключена в условии задачи, то будет выведено сообщение вроде: No symbolic result was found (Символьный результат не был найден). В некоторых точках функция может быть не определена. Если левосторонний и правосторонний пределы для подобной точки будут иметь разные значения, то предела в этой точке не будет. Если вы попытаетесь его вычислить, в качестве ответа будет выдано слово undefined (*Неопределен*).

Пример 12.1. Вычисление пределов различных типов

Вычисление предела в точке:

$$\lim_{x\to 1}\frac{\sqrt[3]{x}-1}{x^2-4x+3}\rightarrow\frac{-1}{6}\qquad\lim_{x\to\infty}\sqrt{x^2+8x+9}-x\rightarrow 4$$

Вычисление правостороннего и левостороннего пределов:

$$\lim_{x\to 0^-}\frac{\sqrt{\cos(x)}-\sqrt[3]{\cos(x)}}{|x|\cdot\sin(x)}\rightarrow\frac{1}{12}\qquad\lim_{x\to 0^+}\frac{\sqrt{\cos(x)}-\sqrt[3]{\cos(x)}}{|x|\cdot\sin(x)}\rightarrow\frac{-1}{12}$$

Пример предела, не существующего в точке. Доказательством того, что данный факт был определен Mathcad верно, является то, что левосторонний и правосторонний пределы имеют разные значения:

$$\lim_{x\to 0^-}\frac{1}{x}\rightarrow-\infty\qquad\lim_{x\to 0^+}\frac{1}{x}\rightarrow\infty\qquad\lim_{x\to 0}\frac{1}{x}\rightarrow\text{undefined}$$

Пределы от функций с неявно заданными параметрами:

$$\lim_{x\to 0}\frac{\ln(1+m\cdot x)}{n\cdot x}\rightarrow\frac{m}{n}\qquad\lim_{x\to 0}\frac{(1+x)^{\alpha}-1}{x}\rightarrow\alpha$$

Mathcad умеет вычислять пределы и для более сложных выражений. Например, можно найти предел бесконечной суммы, произведения. В выражение могут входить операторы интегрирования и дифференцирования. Для примера приведем формулу Валлиса, представляющую собой сходящееся бесконечное произведение. Его предел равен π, поэтому долгое время оно использовалось для вычисления данного важного числа с точностью до десятков знаков.

$$\lim_{k\to\infty}\left(\prod_{n=1}^{k}\frac{2\cdot n}{2n-1}\right)^2\cdot\frac{2}{2k+1}\rightarrow\pi$$

Показателем того, насколько хорошо Mathcad считает пределы, является то, что в изучаемой программе можно вычислить производную только исходя из ее определения как предела отношения приращения функции к приращению аргумента, когда это приращение стремится к 0:

$$\lim_{\delta \to 0} \frac{\ln(\sin(x+\delta)) - \ln(\sin(x))}{\delta} \to \frac{\cos(x)}{\sin(x)} \qquad \frac{d}{dx}\ln(\sin(x)) \to \frac{\cos(x)}{\sin(x)}$$

Аналогично производной, исходя из одного лишь определения, можно вычислить и определенный интеграл. Как вы помните, это, упрощенно, предел суммы площадей прямоугольников, на которые можно разбить криволинейную трапецию. Естественно, что этот предел будет наблюдаться при возрастании количества прямоугольников до бесконечности. Зная это, найдем значение интеграла от f(x)=sin(x) на промежутке от π/2 до π:

$$\lim_{n \to \infty} \sum_{k=0}^{n} \sin\left(\frac{\pi}{2} + \frac{\pi}{2 \cdot n} \cdot k\right) \cdot \frac{\pi}{2 \cdot n} \to 1 \qquad \int_{\frac{\pi}{2}}^{\pi} \sin(x)\,dx \to 1$$

То, что Mathcad способен вычислять столь сложные пределы, можно использовать для строгих математических выводов, а также при написании статей или курсовых, в которых важна формальность используемых формул.

Интересной возможностью символьного процессора является то, что пределы могут быть найдены и в комплексной области.

12.2. Вычисление суммы ряда

Не менее эффективно, чем с пределами, способен Mathcad справляться с такой важной задачей математического анализа, как вычисление суммы ряда. Служит для этого специальный оператор Summation (Суммирование) панели Calculus (Вычисления) (также его можно ввести нажатием сочетания Ctrl+Shift+4):

$$\sum_{\blacksquare=\blacksquare}^{\blacksquare} \blacksquare$$

Оператор суммы ряда содержит четыре маркера, которые заполняются точно в соответствии с принятыми в математике правилами. Пределы суммирования могут быть заданы как числами, так и неизвестными, и даже выражениями. В качестве верхнего предела также может выступать бесконечность. Соответственно, имеется возможность находить суммы как конечных (численно и аналитически), так и бесконечных (только аналитически) рядов (причем ряды могут быть как числовыми, так и функциональными). Бесконечные ряды важны в разного рода аналитических преобразованиях, конечные – в численных расчетах. Рассмотрим особенности их вычисления по отдельности.

В Mathcad встроена довольно большая библиотека формул для суммы рядов. Поэтому практически любой ряд из задачника или справочника будет просуммирован без каких-либо сложностей. Правда, иногда ответ получается громоздким, но часто его можно упростить, задействовав оператор simplify (Упростить) панели Symbolic (Символьные). Отлично умеет программа восстанавливать функции на основании соответствующих им степенных рядов. Во многих случаях удается просуммировать и более сложные функциональные ряды. Совместно с оператором суммирования можно использовать и другие вычислительные операторы, например почленно дифференцируя или интегрируя ряд. Единственное слабое место оператора суммирования – это бесконечные

тригонометрические ряды. Ответить на вопрос, к какой функции сходится тригонометрический ряд, он не сможет даже в случае простейших рядов.

Пример 12.2. Вычисление сумм бесконечных рядов различных типов

Нахождение сумм числовых рядов с положительными членами:

$$\sum_{n=0}^{\infty} \frac{1}{(2n-1)\cdot(2n+1)} \to \frac{-1}{2} \qquad \sum_{n=0}^{\infty} \frac{1}{n!} \to \exp(1)$$

Нахождение сумм знакочередующихся числовых рядов (поразмышляйте над удивительным результатом, полученным при суммировании ряда 1+1–1+1–1...):

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{n!} \to \exp(-1) \qquad \sum_{k=1}^{\infty} \frac{(-1)^{\left(2k^2-k-1\right)}}{k\cdot 3^{k-1}} \to 3\cdot\ln\left(\frac{4}{3}\right) \qquad \sum_{n=0}^{\infty} (-1)^n \to \frac{1}{2}$$

Иногда сумма бесконечного ряда выражается через специальные функции. Чаще всего невозможно получить точное аналитическое значение подобной функции в точке. Однако его можно рассчитать приблизительно, задействовав оператор float.

Сумма приведенного ниже ряда соответствует Z-функции Римана. Для данной функции существуют аналитические значения при четном показателе степени. Если же показатель нечетный или нецелочисленный, то значение Z-функции можно рассчитать лишь приблизительно.

$$\sum_{n=1}^{\infty} \frac{1}{n^p} \to \mathrm{Zeta}(p) \qquad \mathrm{Zeta}(2) \to \frac{1}{6}\cdot\pi^2 \qquad \mathrm{Zeta}(3)\ \mathrm{float},5 \to 1.2021$$

Примеры восстановления функций на основании степенных рядов. Обычно при такого рода расчетах ответ выдается громоздким и его нужно упрощать с использованием оператора simplify. Иногда бывает также необходимым указывать область изменения аргумента (см. ряд для ln(x)):

$$2\sum_{n=0}^{\infty} \frac{(x-1)^{2n+1}}{(2n+1)\cdot(x+1)^{2n+1}} \left|\begin{array}{l}\text{assume},x>1\\ \text{simplify}\end{array}\right. \to \ln(x) \qquad 1+\sum_{n=1}^{\infty} \frac{\prod_{k=0}^{n-1}(\alpha-k)}{n!}\cdot x^n \ \text{simplify} \to (1+x)^{\alpha}$$

Функциональные ряды в Mathcad можно почленно интегрировать и дифференцировать. Для примера возьмем степенной ряд, соответствующий синусу:

$$\sum_{n=0}^{\infty} \frac{d}{dx}\left[\frac{(-1)^{n-1}\cdot x^{2n-1}}{(2n-1)!}\right] \to \cos(x) \qquad \sum_{n=0}^{\infty} \int \frac{(-1)^{n-1}\cdot x^{2n-1}}{(2n-1)!}\,dx \to -\cos(x)$$

Иногда программа использует стандартные формулы довольно некорректно. Например, формула для суммы сходящейся бесконечной геометрической прогрессии будет возвращена по умолчанию, хотя основание прогрессии q потенциально может быть и больше 1 (при этом сумма прогрессии будет равна бесконечности):

$$\sum_{n=1}^{\infty} a\cdot q^{n-1} \ \text{assume},q>0 \to \frac{-a}{q-1}$$

В отличие от сумм бесконечных, суммы конечные можно вычислять как численно, так и символьно. Аналитически конечные суммы приходится находить не так уж и часто, но иногда эта возможность бывает очень полезной. Так, используя оператор суммирования, можно «вывести» формулы для арифметической и геометрической прогрессии, суммы всех нечетных (или четных) чисел от 0 до n, суммы гармонического ряда.

Пример 12.3. Аналитическое вычисление конечных сумм

Суммы типа арифметической прогрессии. Обратите внимание на то, что для того, чтобы получить результат в простой форме, не обойтись без операторов simplify и factor:

$$\sum_{k=1}^{n} k \left| \begin{matrix} \text{simplify} \\ \text{factor} \end{matrix} \right. \rightarrow \frac{1}{2}\cdot n\cdot(n+1) \qquad \sum_{k=1}^{n} (2k-1)\ \text{simplify} \rightarrow n^2$$

$$\sum_{k=1}^{n} k^4\ \text{factor} \rightarrow \frac{1}{30}\cdot n\cdot(2\cdot n+1)\cdot(n+1)\cdot\left(3\cdot n^2+3\cdot n-1\right)$$

Сумма n членов гармонического ряда:

$$\sum_{k=1}^{n} \frac{1}{k} \rightarrow \text{Psi}(n+1)+\gamma$$

В данном выражении γ — это константа Эйлера, Psi — ψ-функция, явное выражение которой имеет следующий вид:

$$\text{Psi}(x) = \frac{d}{dx}\Gamma(x)$$

Чтобы найти численное значение суммы гармонического ряда, полученное выше выражение должно быть приближенно подсчитано с использованием оператора float. К примеру, найдем сумму 100 первых членов ряда:

$$\text{Psi}(101)+\gamma\ \text{float},4 \rightarrow 5.187$$

Сумму членов конечного гармонического ряда можно вычислить и непосредственным суммированием. Правда, если провести эту операцию аналитически, будет получена уж очень громоздкая простая дробь:

$$\sum_{k=1}^{100} \frac{1}{k} \rightarrow \frac{14466636279520351160221518043104131447711}{2788815009188499086581352357412492142272},\text{float}\ 5 \rightarrow 5.1874$$

Наиболее простой вид суммирования — это нахождение суммы конечного ряда численно. Обычно он используется в разного рода приблизительных расчетах. Например, если интегральная функция описывается бесконечным рядом, то, чтобы найти ее значение с ограниченной точностью, достаточно просуммировать лишь определенное количество членов этого ряда. К сумме конечного ряда сводятся все основные методы численного интегрирования. В конце концов, оператор суммирования используется при нахождении банального среднего арифметического последовательности значений.

Пример 12.4. Задачи, сводящиеся к численному нахождению конечных сумм

Задача 1. Не применяя численных методов интегрирования, найти значение интеграла

$$\int_0^{\pi} \frac{\sin(x)}{x}\,dx$$

с точностью до 0,00001.

Данный интеграл является неберущимся, поэтому вычислить его значение аналитически, применив теорему Ньютона–Лейбница, невозможно. Однако мы можем рассчитать его приближенно, используя ряды.

Разложим подынтегральную функцию в ряд Тейлора, используя оператор series (Ряд) панели Symbolic (*Символьные*):

$$\frac{\sin(x)}{x}\ \text{series},x,9\ \rightarrow 1-\frac{1}{6}\cdot x^2+\frac{1}{120}\cdot x^4-\frac{1}{5040}\cdot x^6$$

Попробуем вывести формулу для общего члена данного ряда. Сразу заметим, что величина в знаменателе — это факториал числа, на единицу большего, чем степень x. Далее отметим, что в ряду имеются лишь члены, которым соответствует четное n. Ряд знакочередующийся — следовательно, в числителе формулы общего члена обязательно должен быть множитель $(-1)^p$, где p — это некое выражение от n. Соединяя все эти факты воедино, находим необходимую формулу:

$$\frac{\sin(x)}{x}=\sum_{n=0}^{\infty}\frac{(-1)^n\cdot x^{2n}}{(2n+1)!}$$

В том, что полученная формула верна, можно убедиться, подсчитав сумму аналитически:

$$\sum_{n=0}^{\infty}\frac{(-1)^n\cdot x^{2n}}{(2n+1)!}\rightarrow\frac{1}{x}\cdot\sin(x)$$

Проинтегрируем выведенную формулу общего члена ряда. В результате мы получим бесконечный ряд, сумма которого описывает первообразную для f(x)=sin(x)/x.

$$\int\frac{(-1)^n\cdot x^{2n}}{(2n+1)!}\,dx\rightarrow\frac{x^{2\cdot n+1}}{2\cdot n+1}\cdot\frac{(-1)^n}{(2\cdot n+1)!}\qquad\int\frac{\sin(x)}{x}\,dx=\sum_{n=0}^{\infty}(-1)^n\cdot\frac{x^{2n+1}}{(2n+1)\cdot(2n+1)!}$$

Теперь мы можем, перейдя от бесконечного ряда к конечному, применить теорему Ньютона–Лейбница. Но сколько членов ряда взять, чтобы получить результат с нужной точностью? Мы воспользуемся правилом, гласящим, что погрешность в определении суммы сходящегося ряда *не* превышает величины первого отброшенного члена.

$$f(z):=(-1)^z\cdot\frac{\pi^{2z+1}}{(2z+1)(2z+1)!}$$

$$f(1)=-1.723\qquad f(3)=-0.086\qquad f(5)=-6.7\times10^{-4}\qquad f(7)=-1.461\times10^{-6}$$

Итак, чтобы достигнуть точности в пять знаков после запятой, нужно взять восемь членов ряда первообразной. Тот же результат будет получен, если z мы найдем, определив посредством численного метода решения уравнений, при каком значении аргумента f(z) принимает значение 10^{-6}:

$$TOL := 10^{-15}$$

$$f(z) := \left\lceil \left| \frac{(-1)^z \cdot \pi^{2z+1}}{(2z+1)\cdot\Gamma(2z+1)} \right| \right\rceil - 10^{-6}$$

$$z := 4 \qquad root(f(z), z) = 7.933$$

При решении данного уравнения был применен довольно тонкий и важный прием, на который стоит обратить внимание. В знаменатель исходного выражения f(z) входит факториал от 2z+1. Следовательно, функция f(z) не является непрерывной и дифференцируемой, так как она определена только при целых неотрицательных z. Значит, использовать для поиска ее нулей численные методы не получится (а аналитически решить столь сложное уравнение невозможно). Но мы можем «схитрить», заменив факториал Г-функцией Эйлера. При целых положительных значениях аргумента данная функция равна факториалу, так что тождественность не нарушится. Однако Г-функция непрерывна на промежутке (0, ∞), поэтому в результате замены ею факториала функция f(z) станет непрерывной на промежутке (1/2, ∞). Найти же нуль непрерывной функции численный метод сможет с легкостью. Однако, нужно задать максимальный уровень точности (присвоив TOL значение 10^{-17}), так как порядок величин очень мал.

Найдя оптимальное количество членов приближающего ряда, подсчитываем интеграл:

$$\sum_{n=0}^{7} (-1)^n \cdot \frac{\pi^{2n+1}}{(2n+1)\cdot(2n+1)!} = 1.85194$$

Проверяем верность результата с помощью численного интегрирования:

$$\int_0^{\pi} \frac{\sin(x)}{x}\,dx = 1.85194$$

Задача 2. Функция f(x) задается бесконечным тригонометрическим рядом следующего вида:

$$f(x) := \frac{\pi}{2} - \frac{4}{\pi}\cdot\left(\cos(x) + \frac{\cos(3x)}{9} + \frac{\cos(5x)}{25} + \frac{\cos(7x)}{49} + \ldots\right)$$

С точностью до пяти знаков рассчитать значение производной данной функции в точке x=1 и найти величину интеграла на промежутке от 0 до p. Построить график функции.

Для начала найдем формулу общего члена ряда, задающего функцию. Она довольно очевидна (главное, обратить внимание на то, что в ряду имеются члены лишь для нечетных n):

$$f(x) = \frac{\pi}{2} - \frac{4}{\pi}\cdot\sum_{n=0}^{\infty} \frac{\cos[(2n+1)\cdot x]}{(2n+1)^2}$$

Чтобы можно было вести численные расчеты, следует перейти от бесконечного ряда к конечному, суммируя n первых его членов и не учитывая все остальные. Но какое значение нужно присвоить n? Очевидно, что количество суммируемых членов напрямую определяется требуемым уровнем точности. По условию задачи мы должны найти производную и интеграл с пятью вер-

ными знаками после запятой. Это означает, что функция должна быть аппроксимирована с точностью до 7–8 знаков после запятой, так как два-три разряда будут потеряны за счет погрешности как численных методов, так и компьютерной арифметики. Чтобы найти n исходя из этого условия, подберем для нашего функционального ряда мажорантный числовой ряд (это такой ряд, члены которого всегда больше или равны соответствующим членам данного ряда):

$$\frac{4}{\pi}\cdot\frac{\cos[(2n+1)\cdot x]}{(2n+1)^2} \le \frac{4}{\pi}\cdot\frac{1}{(2n+1)^2}$$

Очевидно, что если n взять таким, что сумма мажорантного ряда вычислится с точностью до 8-го знака, то такой же или более высокий уровень точности будет достигнут при суммировании n членов ряда, задающего f(x) (так как он быстрее сходится). А так как выбранный мажорантный ряд является абсолютно сходящимся числовым рядом, то погрешность вычисления его суммы не превышает величины первого отброшенного члена. На основании этого свойства можно найти n, решив соответствующее уравнение:

$$\frac{4}{\pi}\cdot\frac{1}{(2n+1)^2} = 10^{-8} \;\left|\begin{array}{l}\text{solve},n\\ \text{float},5\end{array}\right.\rightarrow \begin{pmatrix}5641.6\\ -5642.6\end{pmatrix}$$

Итак, чтобы аппроксимировать $f(x)$ с точностью до 8-го знака после запятой, следует просуммировать порядка 5600 членов задающего ее ряда. Согласитесь, это довольно много (разумеется для человека; для компьютера это мизер). Вообще, тригонометрические ряды сходятся намного медленнее, чем ряды Тейлора, что долго ограничивало их использование в приблизительных расчетах.

Задаем функцию, аппроксимирующую f(x):

$$f(x) := \frac{\pi}{2} - \frac{4}{\pi}\cdot\sum_{n=0}^{5642}\frac{\cos[(2n+1)\cdot x]}{(2n+1)^2}$$

Численно подсчитываем производную и интеграл:

$$x := 1 \qquad \frac{d}{dx}f(x) = 1 \qquad \int_0^{\pi} x\,dx = 4.9348$$

Строим график (рис. 12.1). Обратите внимание, что на это потребуется в общем не очень много времени, что удивительно с учетом того, сколько точек должно быть посчитано.

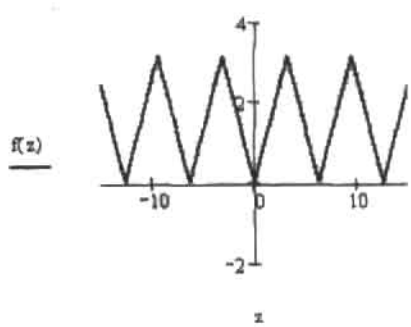


Рис. 12.1. График функции, заданной тригонометрическим рядом

Убедиться в том, что задача была решена верно, можно, зная, что ряд, задающий f(x), есть не что иное, как результат разложения в ряд Фурье функции p(x)=|x| на промежутке от $-\pi$ до π:

$$x := 1 \qquad p(x) := |x|$$

$$\frac{d}{dx}p(x) = 1 \qquad \int_0^{\pi} p(x)\,dx = 4.9348$$

12.3. Произведения

Полностью аналогично сумме ряда можно вычислить произведение его членов. Для этого нужно воспользоваться специальным оператором Iterated Product (Результат произведения), который, помимо нажатия кнопки панели Calculus (Вычисления), можно ввести сочетанием клавиш Ctrl+Shift+3:

$$\prod_{\blacksquare=\blacksquare}^{\blacksquare}\blacksquare$$

В Mathcad можно вычислять конечные (численно и аналитически) и бесконечные (только аналитически) произведения. Наиболее интересны сходящиеся бесконечные произведения. Для примера приведем несколько подобных произведения.

Пример 12.5. Вычисление бесконечных сходящихся произведений

Произведения, сходящиеся к числам:

$$\prod_{n=2}^{\infty}\left(1-\frac{1}{n^2}\right) \rightarrow \frac{1}{2} \qquad \prod_{k=2}^{\infty}\frac{(k-1)\cdot(k+2)}{k\cdot(k+1)} \rightarrow \frac{1}{3}$$

Произведения, сходящиеся к функциям (приведенные произведения могут использоваться для вычисления значений синуса и косинуса вместо рядов Тейлора):

$$\prod_{k=1}^{\infty}\left(1-\frac{1}{k^2\cdot\pi^2}\right) \rightarrow \sin(1) \qquad \prod_{k=1}^{\infty}\left[1-\frac{4}{(2k-1)^2\cdot\pi^2}\right] \rightarrow \cos(1)$$

Произведение, позволяющее вычислять число π:

$$\prod_{k=1}^{\infty}\left(1-\frac{1}{4\cdot k^2}\right) \rightarrow \frac{2}{\pi}$$

Конечные произведения не столь изящны, как бесконечные, однако они куда более важны для практики. Они могут использоваться при решении таких задач, как вычисление биноминальных коэффициентов, геометрического среднего, определения значений различных математических функций и констант.

Пример 12.6. Вычисление конечных произведений

Функция, находящая биноминальные коэффициенты (число сочетаний из n по k):

$$\text{binom}(n,k) := \prod_{i=0}^{k-1} (n-i) \div \prod_{i=1}^{k} i \qquad \text{binom}(5,3) = 10 \qquad \text{binom}(2,1) = 2$$

Функция, позволяющая определять значения синуса. Аналогична встроенной функции sin:

$$\text{new\_sin}(x) := x \cdot \prod_{k=1}^{1000} \left(1 - \frac{x^2}{k^2 \cdot \pi^2}\right) \qquad \sin\left(\frac{\pi}{3}\right) = 0.866 \qquad \text{new\_sin}\left(\frac{\pi}{3}\right) = 0.866$$

12.4. Ранжированные суммы и произведения

Помимо стандартных, широко распространенных в математике операторов обычного суммирования и умножения, в Mathcad есть два довольно интересных оператора так называемых ранжированных сумм (Range Variable Summation — Суммирование по ранжированной переменной) и произведений (Range Variable Iterated Product — Произведение по ранжированной переменной). Ввести их можно либо с панели Calculus, либо сочетаниями клавиш (Shift+4 — для суммы и Shift+3 — для произведения):

$$\sum_{\blacksquare}^{\blacksquare} \qquad \prod_{\blacksquare}^{\blacksquare}$$

При использовании данных операторов в первую очередь нужно задать ранжированную переменную, которая покажет, какие значения должна принимать переменная цикла или произведения. Определив ранжированную переменную, ее имя нужно ввести в нижний маркер оператора. При этом будут просуммированы или перемножены все элементы последовательности, соответствующие заданным значениям ранжированной переменной. Если ранжированная переменная принимает только целочисленные значения, такое суммирование или умножение не имеет никаких отличий от использования простых операторов ряда. Дополнительные возможности ранжированного суммирования или произведения проявляются тогда, когда переменная должна изменяться не с целым шагом. Но взаимозаменимы операторы суммирования и произведений всегда.

Операторы ранжированной суммы или произведения обычно используются тогда, когда необходимо произвести обработку матрицы или вектора. В качестве примера покажем, как с их помощью можно вычислить три вида среднего: арифметическое, геометрическое и среднеквадратичное.

Пример 12.7. Вычисление различных средних для выборки

$$\text{data} := (1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 10)^T \qquad \text{NN} := \text{length}(\text{data})$$

$$i := 0..\ \text{NN} - 1$$

$$\frac{1}{\text{NN}} \cdot \sum_i \text{data}_i = 5.5 \qquad \sqrt[\text{NN}]{\prod_i \text{data}_i} = 4.529 \qquad \sqrt{\frac{\sum_i (\text{data}_i)^2}{\text{NN}}} = 6.205$$

Ранжированные суммы и произведения можно вычислять и аналитически. Причем при этом не обязательно задавать ранжированную переменную. Если в оператор ранжированной суммы или произведения в качестве переменной ввести неопределенную величину k, то ответ будет рассчитан из условия изменения переменной от 0 (или 1, если возникает ошибка деления на 0) до k с шагом 1.

Пример 12.8. Использования операторов ранжированной суммы и произведения в аналитических вычислениях

$$\sum_k \frac{1}{k\cdot(k-1)} \rightarrow \frac{-1}{(k-1)} \qquad \prod_k \frac{1}{k} \rightarrow \frac{1}{\Gamma(k)}$$

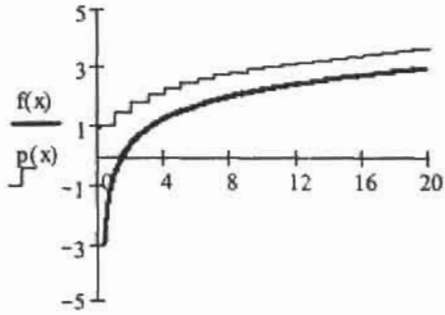
$$\prod_k \frac{1}{k\cdot(k-1)} \rightarrow \frac{1}{\Gamma(k)\cdot\Gamma(k-1)} \qquad \sum_k \frac{1}{k} \rightarrow \mathrm{Psi}(k)$$

Установить однозначное соответствие между результатами символьных вычислений с помощью ранжированных и обычных операторов суммирования и произведения зачастую бывает не столь просто. Например, после суммирования k элементов гармонического ряда в примере 12.8 был получен результат в виде функции Psi(k). Функция эта имеет вид

$$\mathrm{Psi}(x) = \frac{d}{dx}\ln(\Gamma(x))$$

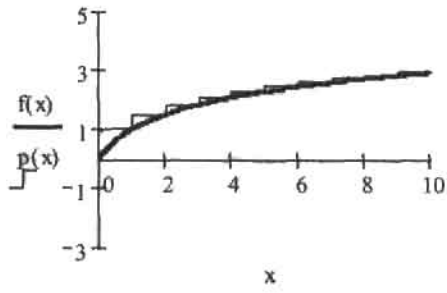
и сама по себе лишь повторяет поведение ломаной гармонического ряда, абсолютно не совпадая с ней:

$$p(x) := \sum_{k=1}^{\mathrm{ceil}(x)} \frac{1}{k} \qquad f(x) := \mathrm{Psi}(x)$$



Чтобы через функцию Psi точно выразить сумму гармонического ряда, к этой функции следует прибавить специальную величину — так называемую константу Эйлера γ, приблизительно равную 0.57722. Также нужно учесть, что функция Psi «отстает» от функции суммы ряда на единицу.

$$p(x) := \sum_{k=1}^{ceil(x)} \frac{1}{k} \qquad \gamma := 0.57722 \qquad f(x) := Psi(x+1) + \gamma$$



Таким образом, результат ранжированного суммирования членов гармонического ряда был получен не совсем корректный. Однако выражение для суммы гармонического ряда в случае использования оператора простого суммирования будет определено верно:

$$\sum_{i=1}^{k} \frac{1}{i} \rightarrow Psi(k+1) + \gamma$$

Сделаем вывод: использование операторов ранжированного суммирования и умножения в случае аналитических расчетов имеет ряд особенностей и может привести к неожиданному или даже неверному результату. Поэтому по возможности (а возможность такая есть всегда) используйте все-таки стандартные операторы ряда.

12.5. Разложение функций в ряды Тейлора

С помощью символьного процессора Mathcad можно выполнить и такую важную с практической точки зрения операцию математического анализа, как разложение функции в ряд Тейлора. Для этого следует воспользоваться оператором series (Ряд) панели Symbolic (Символьные):

■ series , ■ , ■ →

Данный оператор содержит три маркера. В первый вводится функция или ее имя. Во второй — переменная, по которой должно быть проведено разложение. В третий — количество членов ряда (что соответствует порядку точности).

По умолчанию разложение производится в окрестности 0 (то есть вычисляется так называемый ряд Маклорена). Если же ряд нужно найти для окрестности другой точки, то следует переменной во втором маркере оператора series присвоить с помощью логического равенства (Ctrl+=) нужное значение (оно может быть как численным, так и символьным).

Зачастую приходится аппроксимировать рядами функции нескольких переменных. В Mathcad найти такой ряд можно, введя в оператор series не одну переменную, а две (или любое количество). Сделать это можно, просто введя после второго маркера запятую. При этом появится еще один маркер, в котором можно задать имя второй переменной.

Используя оператор series, можно произвести разложение в ряд длиной до 100 членов.

Пример 12.9. Вычисление рядов Тейлора различных типов

Разложение функции в ряд Маклорена:

$$\ln(x+\sin(x))\ \text{series},x,7\ \rightarrow \ln(2)+\ln(x)-\frac{1}{12}\cdot x^2+\frac{1}{1440}\cdot x^4+\frac{1}{18144}\cdot x^6$$

Разложение функции в ряд Тейлора в окрестности точки x=a:

$$\frac{1}{x}\ \text{series},x=a,5\ \rightarrow \frac{1}{a}+\frac{-1}{a^2}\cdot(x-a)+\frac{1}{a^3}\cdot(x-a)^2+\frac{-1}{a^4}\cdot(x-a)^3+\frac{1}{a^5}\cdot(x-a)^4$$

Разложение в ряд Тейлора функции двух переменных:

$$\sin(x+y)\ \text{series},x=\frac{\pi}{4},y=\frac{\pi}{2},2\ \rightarrow \frac{1}{2}\cdot 2^{\frac{1}{2}}-\frac{1}{2}\cdot 2^{\frac{1}{2}}\cdot\left(x-\frac{1}{4}\cdot\pi\right)-\frac{1}{2}\cdot 2^{\frac{1}{2}}\cdot\left(y-\frac{1}{2}\cdot\pi\right)$$

Обратите внимание на то, что ответ при использовании оператора **series** выводится без остаточного члена $O(x^n)$. Это сделано для того, чтобы результат вычислений можно было без проблем использовать в дальнейших расчетах. Если же формула должна быть *абсолютно верной*, то разложение нужно осуществить не посредством оператора **series**, а командой **Expand to Series** (Разложить в ряд) подменю **Variable** (Переменная) меню **Symbolics** (Символьные). При этом остаточный член будет включен в ответ. Например, при разложении f(x)=sin(x) в ряд Маклорена из пяти членов получим следующий результат:

$$1\cdot x-\frac{1}{6}\cdot x^3+O\left(x^5\right)$$

Mathcad не умеет находить формулы общих членов рядов Тейлора. При необходимости эту работу придется производить самостоятельно, анализируя и сопоставляя несколько первых членов разложения. Или же, в крайнем случае, можно воспользоваться справочником.

В математике ряды используются прежде всего для численных расчетов, и, в частности, для аппроксимации функции в окрестности точки, в которой известно ее точное аналитическое значение. В следующем примере рассматривается влияние порядка разложения на точность приближения функции. Обратите также внимание на то, сколь сильно влияет на результат правильность выбора точки разложения.

Пример 12.10. Используя разложение в ряд Тейлора, найти значение функции f(x)=sin(x) в точке x=1 с точностью до 10^{-10}

Чтобы получить максимальную точность при минимальном количестве членов в ряду, мы должны произвести разложение в окрестности точки, для которой характерно, с одной стороны, то, что для синуса в ней известно аналитическое значение, а, с другой, то, что она находится близко к точке x=1. Наилучшим образом этим условиям удовлетворяет точка $x=\pi/3$.

С точкой разложения мы определились. Но сколько членов должно быть в ряду, чтобы была достигнута желаемая точность? Очевидно, что необходимое количество членов разложения зависит от близости точки, в которой должно быть вычислено значение функции, к точке разложения. Чем они будут дальше друг от друга, тем больше членов придется рассчитать. Также, несомненно, влияние оказывают и особенности поведения функции в окрестности точки разложения. В об-

щем, определение необходимого количества членов разложения — это довольно нетривиальная задача. К счастью, имеются удобные формулы, которые позволяют оценить порядок остаточного члена разложения (погрешность будет определяться именно этой величиной). Широко известны формулы для остаточного члена в форме Лагранжа, Коши, Пеано. Менее известна формула остаточного члена в интегральной форме. В рамках нашей задачи наиболее удобна именно она:

$$R_n(x) = \frac{1}{n!}\cdot\int_a^x f^{(n+1)}(t)\cdot(a-t)^n\,dt$$

Здесь n — количество членов разложения, $f^{(n+1)}(t)$ — производная порядка n+1 разлагаемой функции, t — переменная подынтегральной функции (она исчезает после проведения интегрирования), a — точка разложения (в нашем случае $\pi/3$), x — точка, в которой вычисляется значение функции (в нашем случае 1).

Нам нужно найти минимальное n, при котором остаточный член по модулю окажется меньше 10^{-10}. Увы, но если написать на основании формулы для $R_n(x)$ уравнение относительно n, то его не удастся решить ни аналитически (оно будет слишком сложным), ни численно ($R_n(x)$ не является непрерывной функцией от n). Не получится решить его и графически, так как порядок величин слишком мал (не поможет даже использование логарифмической шкалы). Так что остается одно: подобрать n вручную. Для этого представим формулу для остаточного члена в виде функции от n и x:

$$R(n,x) := \frac{1}{n!}\cdot\int_{\frac{\pi}{3}}^x \frac{d^{n+1}}{dt^{n+1}}\sin(t)\cdot\left(\frac{\pi}{3}-t\right)^n dt$$

Подбираем n для точки x=1. Расчет значения функции R(n,x) необходимо выполнить аналитически, а затем пересчитать результат в десятичную дробь. Почему? Во-первых, при этом будет достигаться куда большая точность, чем при численном расчете. Во-вторых, это позволит работать с очень малыми величинами (при численном расчете они будут округляться до 0). В-третьих, главное, в выражение R(n,x) входит производная порядка n+1. А так как численно можно рассчитать производную лишь до пятого порядка, то численно значение R(n,x) можно найти лишь при n<5.

$$R(5,1)\ \text{float},40 \rightarrow -1.32435289842492418209678024020\cdot10^{-11}$$

Итак, мы должны разложить синус в ряд из шести-семи членов (не пяти, так как нужно учесть и погрешность при проведении арифметических расчетов):

$$f(x) := \sin(x)\ \text{series},\left(x=\frac{\pi}{3}\right),6 \rightarrow \left[\frac{1}{2}\cdot3^{\frac{1}{2}}+\frac{1}{2}\cdot\left(x-\frac{1}{3}\cdot\pi\right)+\frac{-1}{4}\cdot3^{\frac{1}{2}}\cdot\left(x-\frac{1}{3}\cdot\pi\right)^2\right]\ldots$$

$$+\frac{1}{12}\cdot\left(x-\frac{1}{3}\cdot\pi\right)^3+\frac{1}{48}\cdot3^{\frac{1}{2}}\cdot\left(x-\frac{1}{3}\cdot\pi\right)^4+\frac{1}{240}\cdot\left(x-\frac{1}{3}\cdot\pi\right)^5$$

Находим значение синуса в точке x=1, отвечая на вопрос задачи:

$$f(1) = 0.8414709848$$

Проверяем верность ответа с помощью встроенной функции Mathcad sin:

$$\sin(1) = 0.8414709848$$

Ответы сошлись. Следовательно, задача была решена нами абсолютно верно.

Особенностью рядов Тейлора является то, что они (при фиксированном количестве членов) тем лучше приближают функцию, чем ближе точка располагается к точке разложения. Соответственно, чем дальше точка находится от точки разложения, тем больше членов ряда придется просуммировать, чтобы достичь нужной точности. В принципе, сколь угодно точное значение функции можно найти в сколь угодно удаленной от точки разложения точке — но количество членов ряда Тейлора, которые придется для этого просуммировать, может быть весьма значительным. Продемонстрируем это на примере. Посмотрим, сколько нужно взять членов ряда, полученного разложением синуса в точке a=π/3, чтобы вычислить значение этой функции в точке x=2 с точностью до 10^{-10}:

$$R(12, 2)\ \text{float}, 40 \rightarrow 3.021105652360807083921567500049 5 \cdot 10^{-11}$$

Как и следовало ожидать, с удалением от точки разложения длина аппроксимирующего ряда возросла. Посмотрим, будет ли результат точен до 10-го знака, если мы возьмем 13 членов разложения:

$$f(2) = 0.9092974268 \qquad \sin(2) = 0.9092974268$$

Возьмем еще более удаленную точку x=10:

$$R(40, 10)\ \text{float}, 40 \rightarrow -2.93146035076434073576993836808991325817 6 \cdot 10^{-11}$$

$$f(10) = -0.5440211109 \qquad \sin(10) = 0.5440211109$$

Чтобы вычислить в этой точке синус с нужной точностью, придется просуммировать 41 член разложения! Представляете, как сложно было бы выполнить эту работу на бумаге? Делаем вывод: очень важно правильно выбрать точку разложения — это может уменьшить объем вычислительной работы на порядок.

Заканчивая обсуждение данной задачи, приведем график функции, аппроксимирующей синус, которая представляет собой ряд из 41 члена (рис. 12.2). Посмотрите, на каком широком промежутке полином ведет себя точно так же, как синус. Удивительно, не правда ли?

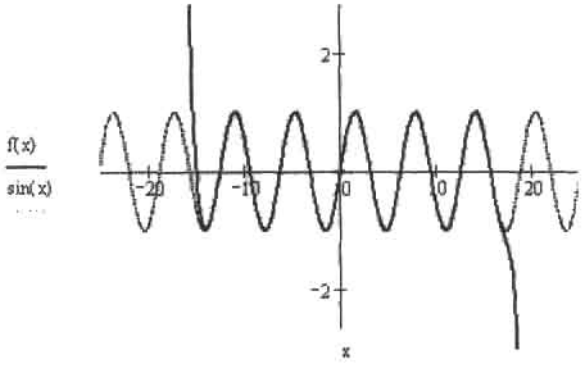


Рис. 12.2. Кривая синуса и график аппроксимирующего ее полинома

Помимо приблизительного нахождения значений функций в неаналитических точках, ряды Тейлора активно используются при вычислении пределов. Данный подход хорош

своей универсальностью и простотой, поэтому именно его использует символьный процессор Mathcad при подсчете пределов. Приведем и мы пример такого применения рядов Тейлора.

Пример 12.11. Нахождение пределов с использованием разложения в ряд Тейлора

Пусть стоит задача без применения оператора lim найти следующий предел:

$$\lim_{x \to 0} \frac{e^{\frac{-x^2}{2}} - \cos(x)}{x^3 \cdot \sin(x)}$$

Чтобы найти данный предел, нужно раскрыть неопределенность $-1/0$, что довольно непросто сделать посредством стандартных средств. Однако если разложить функцию в ряд Тейлора, задача будет решена довольно просто:

$$\frac{e^{\frac{-x^2}{2}} - \cos(x)}{x^3 \cdot \sin(x)} \text{ series}, x, 9 \rightarrow \frac{1}{12} - \frac{1}{180} \cdot x^2 + \frac{29}{30240} \cdot x^4$$

Полагая x равным 0, находим, что предел будет равен 1/12. Этот же ответ будет выдан и оператором lim Mathcad:

$$\lim_{x \to 0} \frac{e^{\frac{-x^2}{2}} - \cos(x)}{x^3 \cdot \sin(x)} \rightarrow \frac{1}{12}$$

Приведем примеры расчета с использованием ряда Тейлора еще двух пределов:

$$\lim_{x \to 0} \left[\left(\cos(x) + \frac{x^2}{2} \right)^{\frac{1}{x \cdot (\sin(x) - x)}} \right] \rightarrow e^{\frac{-1}{4}} \qquad \left(\cos(x) + \frac{x^2}{2} \right)^{\frac{1}{x \cdot (\sin(x) - x)}} \text{ series}, x, 5 \rightarrow e^{\frac{-1}{4}}$$

$$\lim_{x \to 0} \left(1 + x^2\right)^{\frac{1}{e^x - 1 - x}} \rightarrow e^2 \qquad \left(1 + x^2\right)^{\frac{1}{e^x - 1 - x}} \text{ series}, x, 3 \rightarrow e^2$$

12.6. Разложение функций в ряды Фурье

Разложение в ряд Тейлора периодических функций — это не очень эффективный ход, так как при этом хороший уровень аппроксимации будет достигнут лишь в окрестности точки разложения. При отдалении же от нее приближение будет плохим, хотя участки

кривой будут идентичны участку, на котором проводилось разложение. Очевидно, что для того, чтобы хорошо приближать периодическую функцию, аппроксимирующий полином сам должен быть периодической функцией с таким же периодом, как у приближаемой функции. Чтобы удовлетворить этому условию, нужно разложить функцию не по степеням x, а по тригонометрическим функциям. Полученный при этом ряд называется рядом Фурье. Он имеет следующий общий вид (для функций с периодом 2π):

$$f(x) = \frac{a_0}{2} + \sum_{n=0}^{\infty} \left(a_n \cdot \cos(n \cdot x) + b_n \cdot \sin(n \cdot x)\right)$$

В математическом анализе доказано, что погрешность приближения функции рядом Фурье минимальна тогда и только тогда, когда коэффициенты a_0, b_n и a_n определяются следующим образом:

$$a_0 = \frac{1}{\pi} \cdot \int_{-\pi}^{\pi} f(x)\,dx \qquad a_n = \frac{1}{\pi} \cdot \int_{-\pi}^{\pi} f(x) \cdot \cos(n \cdot x)\,dx \qquad b_n = \frac{1}{\pi} \cdot \int_{-\pi}^{\pi} f(x) \cdot \sin(n \cdot x)\,dx$$

В Mathcad нет оператора наподобие series, с помощью которого можно было бы производить разложение функций в ряд Фурье. Однако это не значит, что данная проблема не решаема. Используя вычислительные возможности программы, с ней вполне можно справиться. Как это сделать, показано в примере 12.12.

Пример 12.12. Разложение функции в ряд Фурье

Пусть стоит задача — разложить на промежутке от $-\pi$ до π в ряд Фурье функцию $f(x)=x$. Для этого создадим функцию Fourier(x, k), основываясь на приведенных выше формулах:

$$f(x) := x$$

$$\text{Fourier}(x, k) := \frac{1}{2\pi} \cdot \int_{-\pi}^{\pi} f(x)\,dx + \frac{1}{\pi} \cdot \sum_{n=1}^{k} \left(\int_{-\pi}^{\pi} f(x) \cdot \cos(n \cdot x)\,dx \cdot \cos(n \cdot x) + \int_{-\pi}^{\pi} f(x) \cdot \sin(n \cdot x)\,dx \cdot \sin(n \cdot x) \right)$$

Чтобы получить ряд из k членов в символьном виде, функцию Fourier следует вычислить аналитически. Найдем девять членов разложения $f(x)=x$ в ряд Фурье:

$$z(x) := \text{Fourier}(x, 9) \text{ factor } \rightarrow 2 \cdot \sin(x) - \sin(2 \cdot x) + \frac{2}{3} \cdot \sin(3 \cdot x) - \frac{1}{2} \cdot \sin(4 \cdot x) \ldots$$
$$+ \frac{2}{5} \cdot \sin(5 \cdot x) - \frac{1}{3} \cdot \sin(6 \cdot x) + \frac{2}{7} \cdot \sin(7 \cdot x) - \frac{1}{4} \cdot \sin(8 \cdot x) + \frac{2}{9} \cdot \sin(9 \cdot x)$$

Тригонометрические ряды сходятся медленно, а полученный — вообще на уровне знакопеременного гармонического. Поэтому точность приближения им функции при суммировании девяти членов разложения будет невысока:

$$z(1) = 0.988 \qquad z(2) = 2.044$$

То, что девяти членов разложения явно недостаточно для качественной аппроксимации, можно увидеть, построив график z(x) (рис. 12.3).

Что же делать, если функцию нужно аппроксимировать с высокой точностью? Идею получить необходимый для этого ряд в явном виде отбросим сразу (оперировать выражением из десятков, а то и сотен элементов очень сложно). Можно попробовать просчитать функцию Fourier численно. Посмотрим, какую точность обеспечит суммирование 20, 50 и 100 членов разложения:

$$\text{Fourier}(-1, 20) = -0.9447022 \qquad \text{Fourier}(-1, 50) = -0.9946389 \qquad \text{Fourier}(-1, 100) = -1.0003202214$$

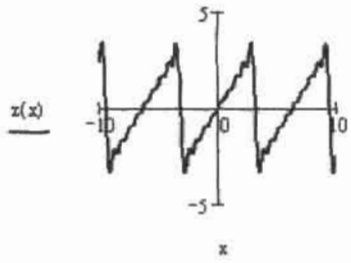


Рис. 12.3. Из-за медленной сходимости ряда девять членов — это слишком мало для хорошего приближения функции (обратите внимание на колебания, которые совершает кривая)

В общем, если функцию нужно приблизить с точностью до 0,001–0,0001, численный расчет вполне приемлем. Однако более высокой точности он обеспечить не может, что связано с погрешностью численных методов интегрирования. Ее, конечно, можно уменьшить, присвоив TOL минимальное значение. Однако при этом численные алгоритмы интегрирования перестанут сходиться уже при n порядка 50.

Если стоит задача аппроксимировать функцию рядом Фурье максимально точно, нужно вывести формулу общего члена и использовать ее в сочетании с оператором суммы. При этом можно будет суммировать тысячи и даже миллионы членов разложения буквально за секунды. В нашем случае найти формулу общего члена будет очень просто, если вынести за скобку 2:

$$f(x) = 2 \cdot \left[\sin(x) - \frac{\sin(2x)}{2} + \frac{\sin(3x)}{3} - \frac{\sin(4x)}{4} + \frac{\sin(5x)}{5} + \ldots + (-1)^{n+1} \cdot \frac{\sin(n \cdot x)}{n} \right]$$

На основании данной формулы создадим функцию, которая позволит суммировать произвольное количество членов ряда:

$$F(x,k) := 2 \cdot \sum_{n=0}^{k} \frac{(-1)^{n+1} \cdot \sin(n \cdot x)}{n}$$

Проверим, какую точность аппроксимации обеспечит суммирование 1000, 10 000 и 10 000 000 (!) членов ряда:

$$F(-1,1000) = -0.998866428398192 \qquad F(-1,10000) = -1.00008257593928$$

$$F(-1,10000000) = -1.00000000750972$$

Суммирование 1000 членов ряда Фурье для f(x)=x дает достаточную точность, чтобы можно было построить «правильный» график (рис. 12.4).

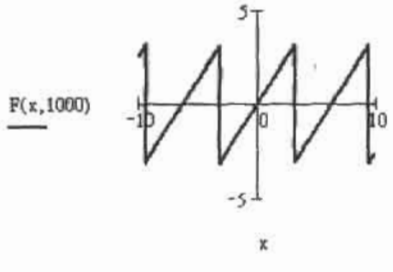


Рис. 12.4. «Периодическая линейная» функция

Глава 13. Исследование функций и оптимизация

Исследование функций различной степени сложности и построение их графиков является важнейшей задачей математического анализа, с которой приходится сталкиваться любому старшекласснику и студенту естественной специальности вуза.

В основе методов исследования поведения функций как одной, так и многих переменных лежит вычисление производных (первого и второго порядков), пределов, а также (для функций нескольких переменных) решение систем уравнений. Изучив материал предыдущих глав, вы наверняка убедились в том, что с подобными операциями Mathcad под вашим чутким руководством справляется весьма эффективно, поэтому длительную и, в общем-то, не требующую аналитического подхода процедуру исследования некоторой функции вы сможете провести с его помощью, затратив всего пару минут. Благодаря удивительным графическим возможностям системы вы без труда построите кривую или поверхность любой сложности, описывающую анализируемую функцию — график наглядно отражает особенности ее поведения, что может значительно упростить проведение исследования.

13.1. Исследование функций одной переменной

В данном разделе мы рассмотрим наиболее часто встречающиеся в практикуме по высшей математике задачи, связанные с исследованием функций одной переменной: определением точек экстремума и перегиба, промежутков монотонности, выпуклости и вогнутости, нахождением асимптот. К сожалению, в Mathcad имеются встроенные средства лишь для отыскания максимумов и минимумов функций (особенностям их использования мы уделим внимание в соответствующем разделе), поэтому большинство этапов исследования функции в Mathcad придется проводить, используя классические формулы и определения.

Пример 13.1. Исследовать функцию

$$f(x) = \sqrt[3]{(x+3)x^2}$$

и построить ее график

Поскольку алгоритм исследования функции включает в себя большое количество пунктов, разберем каждый из них по отдельности, прибегая, в случае необходимости, к теоретическим пояснениям.

Для облегчения поставленной задачи построим в первую очередь график функции (рис. 13.1). Заметьте, при проведении исследования на бумаге такой возможности у вас нет.

$$f(x) := \sqrt[3]{(x+3)x^2}$$

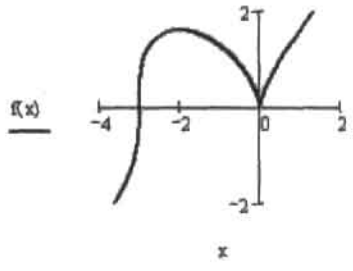


Рис. 13.1. График исследуемой функции

Во-первых, найдем область определения и область значений функции. X может принимать любые значения (в выражении функции отсутствуют корни четной степени и знаменатель), поэтому функция определена на всей числовой оси. Очевидно, область значений функции принадлежит промежутку $(-\infty; \infty)$.

Во-вторых, определим, является ли функция четной, нечетной или периодической.

Видно, что график несимметричен относительно оси ординат и начала координат, значит это функция общего вида. Приведем аналитическое подтверждение.

$$f(a) \to \left[(a+3)\cdot a^2\right]^{\frac{1}{3}} \qquad f(-a) \to \left[(-a+3)\cdot a^2\right]^{\frac{1}{3}} \qquad -f(a) \to -\left[(a+3)\cdot a^2\right]^{\frac{1}{3}}$$

Как видно из полученных выражений, условия четности ($f(-a)=f(a)$) и нечетности ($f(-a)=-f(a)$) не соблюдаются, значит, это функция общего вида.

Докажем непериодичность исследуемой функции. Функция является периодической, если для любого x существует такое действительное число T, отличное от нуля, что $f(x+T)=f(x)$. Доказательством периодичности/непериодичности функции является нахождение величины T из уравнения $f(x+T)=f(x)$. Если T является действительным числом, функция периодична. В случае непериодичности T будет являться функцией от x.

$$\sqrt[3]{(x+3)x^2} = \sqrt[3]{(x+T+3)(x+T)^2}\ \text{solve},T \to \begin{bmatrix} \frac{-3}{2}\cdot x - \frac{3}{2} + \frac{1}{2}\cdot\left(-3\cdot x^2 - 6\cdot x + 9\right)^{\frac{1}{2}} \\ \frac{-3}{2}\cdot x - \frac{3}{2} - \frac{1}{2}\cdot\left(-3\cdot x^2 - 6\cdot x + 9\right)^{\frac{1}{2}} \\ 0 \end{bmatrix}$$

При решении уравнения f(x+T)=f(x) относительно T мы получили три значения. По определению T≠0, остальные же выражения зависят от x, следовательно, функция является непериодической.

В-третьих, точек разрыва функция не имеет, поскольку непрерывна на промежутке $(-\infty; \infty)$.

В-четвертых, найдем точки пересечения графика функции с осями координат.

$$\sqrt[3]{(x+3)x^2}\ \text{solve},x \rightarrow \begin{pmatrix} -3 \\ 0 \\ 0 \end{pmatrix}$$

График функции пересекает ось Ox в точках (-3; 0), (0; 0).

$$f(0) = 0$$

График функции пересекает ось Oy в точке (0; 0).

В-пятых, определим точки экстремума, значения функции в этих точках, а также интервалы возрастания и убывания функции.

Необходимым условием существования экстремума в точке x_0 является равенство нулю или несуществование (равенство бесконечности) производной в данной точке. Достаточным условием существования экстремума в точке x_0 является смена знака производной при переходе через точку x_0.

$$\frac{d}{dx}f(x)\ \text{simplify} \rightarrow x\cdot\frac{x+2}{\left[(x+3)\cdot x^2\right]^{\frac{2}{3}}} \qquad x\cdot\frac{x+2}{\left[(x+3)\cdot x^2\right]^{\frac{2}{3}}}\ \text{solve},x \rightarrow -2$$

Производная равна нулю при x=−2, при x=−3 и x=0 она обращается в бесконечность. Значит, данные критические точки являются точками перегиба или экстремума. Чтобы проверить последнее утверждение, найдем интервалы возрастания и убывания функции. Для этого вспомним достаточное условие монотонности функции, утверждающее, что дифференцируемая и возрастающая (убывающая) в интервале (a, b) функция f(x) имеет во всех точках этого интервала неотрицательную (неположительную) производную.

$$x\cdot\frac{x+2}{\left[(x+3)\cdot x^2\right]^{\frac{2}{3}}} > 0\ \text{solve},x \rightarrow \begin{bmatrix} x < -3 \\ (-3 < x)\cdot(x < -2) \\ 0 < x \end{bmatrix}$$

$$x\cdot\frac{x+2}{\left[(x+3)\cdot x^2\right]^{\frac{2}{3}}} < 0\ \text{solve},x \rightarrow \begin{bmatrix} x < -3 \\ (-2 < x)\cdot(x < 0) \end{bmatrix}$$

Как вы помните, во избежание ошибки при решении неравенств в Mathcad полученный результат следует обязательно проверять по графику (подробно о правилах интерпретации результатов, полученных при решении неравенств, см. в гл. 9). В нашем случае решение обеих неравенств было найдено с учетом того, что в точке x=−3 функция производной терпит разрыв, однако с помощью графика мы можем указать корректные промежутки знакопостоянства исследуемой функции: в интервале $(-\infty; -2) \cup (0; \infty)$ функция возрастает, в интервале $(-2; 0)$ — убывает.

Итак, при переходе через точку x=−2 производная меняет знак с плюса на минус, значит, в данной точке функция имеет максимум:

$$f(-2) = 1.587$$

При переходе же через x=0 знак производной меняется с минуса на плюс, следовательно, в данной точке расположен минимум:

$$f(0) = 0$$

Заметьте, при x=−2 экстремум гладкий (f'(−2)=0), а при x=0 — острый (f'(0)=∞).

При x=−3 экстремума нет, поскольку знак производной сохраняется. Поведение функции в этой точке требует дополнительного исследования. Взглянув на график, легко убедиться в том, что x=−3 является точкой перегиба, однако это необходимо подтвердить аналитически, определив интервалы вогнутости и выпуклости.

В-шестых, найдем интервалы вогнутости и выпуклости графика функции, а также точки перегиба.

Достаточным условием вогнутости (выпуклости) кривой на промежутке (a; b) является положительный (отрицательный) знак второй производной на данном промежутке.

$$\frac{d^2}{dx^2} f(x) \text{ simplify} \rightarrow \frac{-2}{(x+3)\cdot\left[(x+3)\cdot x^2\right]^{\frac{2}{3}}}$$

Промежутки монотонности функции второй производной можно определить из соответствующих неравенств, проверяя результаты по графику (рис. 13.2).

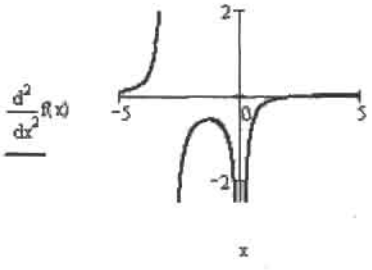


Рис. 13.2. График функции второй производной

$$\frac{-2}{(x+3)\cdot\left[(x+3)\cdot x^2\right]^{\frac{2}{3}}} > 0 \text{ solve}, x \rightarrow x < -3$$

$$\frac{-2}{(x+3)\cdot\left[(x+3)\cdot x^2\right]^{\frac{2}{3}}} < 0 \text{ solve}, x \rightarrow \begin{bmatrix} x < -3 \\ (x < 0)\cdot(-3 < x) \\ 0 < x \end{bmatrix}$$

Глядя на график, можно сделать вывод о том, что первое неравенство было решено системой верно: $f''(x)>0$ на промежутке $(-\infty; -3)$, а вот со вторым неравенством Mathcad справился немного хуже — полученный результат верен частично: $f''(x)<0$ на промежутке $(-3; 0) \cup (0; \infty)$. Исходя из этого, в интервале $(-\infty; -3)$ график исследуемой функции является вогнутым, в интервале же $(-3; 0) \cup (0; \infty)$ — выпуклым.

При переходе через точку $x=-3$ вторая производная меняет знак, значит, $x=-3$ является точкой перегиба графика функции.

Обратите внимание, в данном случае достаточное условие точки перегиба не выполняется, поскольку в ней вторая производная равна не 0, а бесконечности. Тем не менее поведение функции второй производной (которая не является непрерывной) в окрестности этой точки дает нам право считать ее точкой перегиба (что, в общем-то, очевидно из графика).

При $x=0$ вторая производная также обращается в бесконечность, но не меняет знак в окрестности данной точки, поэтому при $x=0$ перегиба нет.

В-седьмых, найдем асимптоты графика функции

Асимптотой кривой называется прямая, к которой неограниченно приближается точка этой кривой при неограниченном удалении от начала координат.

Исследуемая нами функция не имеет вертикальных асимптот, поскольку непрерывна на всей числовой оси. Невертикальные асимптоты будем искать в виде $y=kx+b$ (рис. 13.3):

$$k := \lim_{x \to \infty} \frac{f(x)}{x} \to 1 \qquad b := \lim_{x \to \infty} (f(x) - k \cdot x) \to 1$$

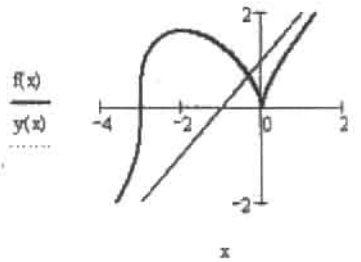


Рис. 13.3. Асимптота $y=x+1$ графика исследуемой функции

$y=x+1$ является наклонной асимптотой. Расстояние от нее до точек графика функции неограниченно уменьшается с ростом x:

$$\lim_{x \to \infty} [f(x) - (x + 1)] \to 0$$

В-восьмых, найдем предельные значения функции на концах области определения.

$$\lim_{x \to -\infty} \sqrt[3]{(x + 3) \cdot x^2} \to \infty \qquad \lim_{x \to \infty} \sqrt[3]{(x + 3) \cdot x^2} \to \infty$$

Строго говоря, предельное значение функции на левой границе области определения равно не ∞, а $-\infty$, в чем легко убедиться исходя из графика.

Проведенное нами исследование функции в очередной раз доказывает, что полностью доверять результатам, полученным при символьных расчетах, нельзя: проверка любого из них по графику обязательна.

13.2. Исследование функций нескольких переменных

Как и в случае функций одной переменной, Mathcad существенно облегчает исследование функций многих переменных благодаря возможности построения описывающих их поверхностей и линий уровня, с помощью которых доказать непрерывность функции, найти экстремумы, седловые точки, а также точки или линии разрыва намного проще.

В данном разделе мы разберем, как средствами Mathcad можно определить экстремумы функции двух переменных.

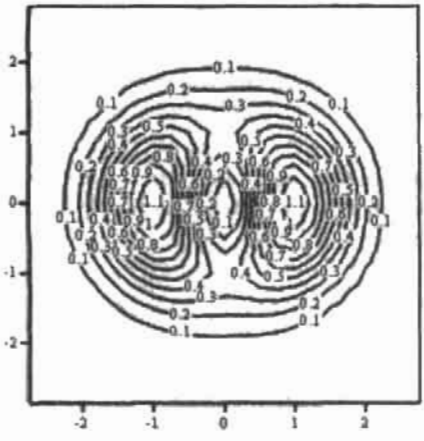
Пример 13.2. Найти экстремум функции двух переменных

$$f(x, y) = e^{-x^2-y^2} \cdot \left(3x^2 + y^2\right)$$

Построить линии уровня функции

Построим в первую очередь линии уровня исследуемой функции (подробно о способах задания поверхностей и контурных графиков рассказано в гл. 6) (рис. 13.4):

$$f(x, y) := e^{-x^2-y^2} \cdot \left(3x^2 + y^2\right)$$



f

Рис. 13.4. Линии уровня исследуемой функции двух переменных

Необходимым условием экстремума функции нескольких переменных в точке (x_0, y_0) является равенство нулю ее частных производных в данной точке. Достаточное условие экстремума заключается в следующем. Пусть в точке (x_0, y_0) $f'_x(x_0, y_0)=0$, $f'_y(x_0, y_0)=0$, и функция имеет в этой точке непрерывные частные производные второго порядка:

$$f''_{xx}(x_0, y_0)=A \qquad f''_{xy}(x_0, y_0)=f''_{yx}(x_0, y_0)=B \qquad f''_{yy}(x_0, y_0)=C$$

$$\Delta=AC-B^2$$

Тогда если $\Delta>0$, функция имеет экстремум в точке (x_0, y_0): максимум при $A<0$ и минимум при $A>0$. Если $\Delta<0$, экстремума в точке нет — она является седловой.

Исследование функции двух переменных на экстремум также проведем по пунктам.

Во-первых, найдем частные производные функции:

$$\frac{\partial}{\partial y} f(x,y) \rightarrow -2\cdot y\cdot e^{-x^2-y^2}\cdot\left(3\cdot x^2+y^2\right)+2\cdot y\cdot e^{-x^2-y^2}$$

$$\frac{\partial}{\partial x} f(x,y) \rightarrow -2\cdot x\cdot e^{-x^2-y^2}\cdot\left(3\cdot x^2+y^2\right)+6\cdot x\cdot e^{-x^2-y^2}$$

Во-вторых, найдем критические точки функции, решив систему уравнений $f'_x(x_0, y_0)=0$, $f'_y(x_0, y_0)=0$:

Given

$$-2\cdot x\cdot e^{-x^2-y^2}\cdot\left(3\cdot x^2+y^2\right)+6\cdot x\cdot e^{-x^2-y^2}=0$$

$$-2\cdot y\cdot e^{-x^2-y^2}\cdot\left(3\cdot x^2+y^2\right)+2\cdot y\cdot e^{-x^2-y^2}=0$$

$$\text{Find}(x,y)\rightarrow\begin{pmatrix}0 & 0 & 0 & 1 & -1\\ 0 & 1 & -1 & 0 & 0\end{pmatrix}$$

Обратите внимание, в данном случае система нелинейных уравнений была решена символьно. К сожалению, в Mathcad это возможно далеко не всегда, поэтому в случае, если критические точки символьно определить не удастся, для каждой из них в отдельности придется задавать начальное приближение исходя из контурного графика.

В-третьих, найдем частные производные второго порядка, вычислим их значения в каждой критической точке и с помощью достаточного условия сделаем вывод о наличии экстремумов (для экономии места исследуем две точки из полученных пяти).

$$A(x,y):=\frac{\partial^2}{\partial x^2}f(x,y) \qquad B(x,y):=\frac{\partial}{\partial x}\frac{\partial}{\partial y}f(x,y) \qquad C(x,y):=\frac{\partial^2}{\partial y^2}f(x,y)$$

$$A(1,0)\cdot C(1,0)-B(1,0)^2=6.496 \qquad A(1,0)=-4.415$$

В точке (1, 0) $\Delta>0$ и $A<0$, значит, это точка максимума. Аналогичным образом можно доказать, что (−1, 0) является точкой максимума, а (0, 0) — точкой минимума функции.

Исследуем точку (0, 1).

$$A(0,1)\cdot C(0,1)-B(0,1)^2=-2.165$$

Δ оказалось отрицательным, следовательно, в данной точке экстремума нет, она является седловой. То же самое можно утверждать, исследовав точку (0, −1).

В-четвертых, найдем значения функции в точках экстремума:

$$f(0,0)\rightarrow 0 \qquad f(1,0)\rightarrow 3\cdot e^{-1} \qquad f(-1,0)\rightarrow 3\cdot e^{-1}$$

13.3. Численное определение экстремумов функций

Очень многие математические задачи решаются с помощью определения точки экстремума соответствующей функции. Учитывая важность данной проблемы, разработчики Mathcad внесли в программу специальные функции, которые, используя возможности численных алгоритмов решения уравнений, могут весьма эффективно с ней справляться. Однако использование градиентных методов (а именно такие алгоритмы лежат в основе поиска экстремумов функций) сопровождается многочисленными сложностями, связанными с правильным выбором начальных приближений, точности настроек метода. Поэтому в этом разделе мы попробуем разобраться, как можно быстро и просто найти точки минимума или максимума с помощью системы Mathcad и как при этом не допустить ошибок.

13.3.1. Экстремум функции одной переменной

Поиск экстремума функции — это задача, технически очень схожая с поиском корней уравнения. Критерием существования корня в некоторой точке является то, что величина функции в ней по модулю должна быть меньше некоторого, определенного в системе или пользователем, параметра точности. При поиске же экстремума меньше этого параметра должна быть не сама функция, а ее первая производная. Минимум же это или максимум, можно определить по знаку второй производной.

В программе Mathcad существуют две функции, отвечающие за поиск экстремумов: Minimize(f,x1,x2....) (определяет локальные минимумы) и Maximize (f,x1,x2....) (ищет локальные максимумы). Синтаксис их довольно схож с синтаксисом функций решения уравнений (root, find) и имеет два возможных варианта записи. Единственным отличием в синтаксисе функций Maximize, Minimize и root является то, что при поиске экстремумов в скобках соответствующей встроенной функции должно быть заключено лишь имя исследуемой функции, без переменных, от которых она зависит.

Если вам известно приблизительное расположение нужной вам точки экстремума, то можно воспользоваться более простым стандартным вариантом употребления функций Minimize и Maximize. В этом случае требуется задать начальное приближение.

Пример 13.3. Поиск экстремумов с помощью функций Minimize и Maximize при заданном начальном приближении

$$f(x) := x^3 - 3x - 1$$

$$x := 1$$

$$\text{Maximize}(f, x) = -1 \qquad \text{Minimize}(f, x) = 1$$

Убедиться в правильности полученных результатов можно, построив график.

Если вы щелкнете правой кнопкой мыши на тексте функции Minimize или Maximize, то откроется контекстное меню, практически полностью аналогичное меню функции find. Отличие в них есть только одно: поиск экстремумов нельзя произвести с помощью метода Левенберга. Так что, если функция не справится эффективно со своей задачей, вы можете поменять используемый метод или самостоятельно произвести его настройку.

Впрочем, как показывает опыт, это вряд ли поможет (ввиду того, что эффективность как квазиньютоновского метода, так и метода сопряженных градиентов практически совпадает).

Эффективность работы функций Minimize и Maximize сильно зависит от выбора точки начального приближения. Особенно это проявляется в тех случаях, когда количество локальных минимумов или максимумов велико.

Пример 13.4. Исследуем функцию (рис. 13.5)

$$X(t) := 2e^{-t}\cdot\cos(6t)$$

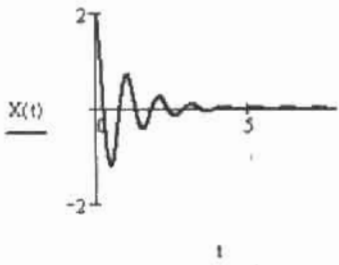


Рис. 13.5. График функции, описывающей затухающие колебания

Функции, подобные этой, в теоретической механике применяются для описания затухающих колебаний. Поэтому график отображен только начиная с 0, поскольку время не может быть отрицательной величиной.

Экстремумов, равно как и корней, данная функция имеет бесконечное множество. Посмотрим, какой результат дадут функции Maximize и Minimize при различных приближениях:

$$t := 0$$

$$\text{Maximize}(X,t) = -0.028 \qquad \text{Minimize}(X,t) = 0.496$$

Отрицательная координата максимума была получена потому, что, чисто с математической точки зрения, данная функция существует и в отрицательной области, причем там она совершает колебания с увеличивающейся амплитудой по направлению к бесконечности.

$$t := 10$$

$$\text{Maximize}(X,t) = 2.067 \qquad \text{Minimize}(X,t) = 2.59$$

Интересно, что существуют локальные экстремумы, которые расположены ближе к 10, чем полученные с помощью функций Maximize и Minimize. То, что они не были обнаружены, связано, скорее всего, с их малой амплитудой.

$$t := 10000$$

$$\text{Maximize}(X,t) = 1\times 10^{4} \qquad \text{Minimize}(X,t) = 1\times 10^{4}$$

На этот раз действительно получился странный результат: если верить Mathcad, данная функция имеет в одной точке одновременно и минимум, и максимум! Естественно, реально такого быть никогда не может. Попробуем объяснить причину допущенной ошибки.

Как вы помните, главным условием существования в точке экстремума является условие равенства производной нулю. Рассматриваемая функция является убывающей, и при значении переменной 10 000 с точностью, значительно превышающей границу нуля в Mathcad, равняется нулю. Производная же по величине в данном случае должна быть близка к функции. Таким образом, получается, что для всех точек, начиная с некоторого значения $t_д$, будет выполняться условие экстремума. Тип же экстремума определяется обычно по знаку второй производной, которая в данной ситуации также должна бесконечно близко приближаться по значению к нулю. Из-за этого возникает неопределенность, и обе функции относят данную точку к экстремуму своего типа.

И, наконец, если мы попробуем ввести большое по модулю приближение из отрицательной области, то никаких значений экстремумов найдено не будет. Причина же этого довольно очевидна: при таких значениях t функция принимает слишком большие по модулю значения.

Делая вывод, можно сказать, что правильный выбор начальной точки играет принципиальную роль при поиске экстремумов. А осложнить выполнение этой работы могут те же особенности поведения исследуемой функции, что и при решении уравнений: множественность экстремумов, разрывы, особенности поведения функции на бесконечности. Поэтому к полученному результату нужно относиться очень внимательно и стараться всегда проверять его верность с помощью графика. Благо, в случае функции одной переменной это можно сделать очень легко.

Второй возможной формой употребления функций **Minimize** и **Maximize** является запись их в составе вычислительного блока, подобно функции **find**. При такой форме синтаксиса появляются некоторые дополнительные возможности. Например, записав ниже ключевого слова **Given** систему неравенств, вы сможете произвести поиск экстремумов на определенном интервале изменения переменной. Также здесь вы сможете ввести некоторые ограничения на значения минимума или максимума.

Пример 13.5. Рассмотрим функцию (рис. 13.6)

$$f(x) := -x^7 + 4x^3 - 3x$$

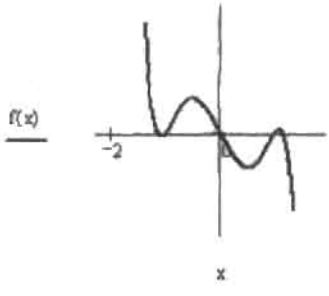


Рис. 13.6. График исследуемой функции

При использовании стандартной записи функции Maximize при начальном приближении x=3 система выдает значения максимума, более близкого к этой точке:

$$f(x) := -x^7 + 4x^3 - 3x$$

$$x := 3$$

$$\text{Maximize}(f, x) = 1.077$$

Но при использовании записи с вычислительным блоком при этом же приближении мы сможем получить значение и из отрицательной области. Для этого требуется лишь ввести ограничения на положительные величины точки максимума под ключевым словом Given:

$$x := 3$$

$$f(x) := -x^7 + 4x^3 - 3x$$

$$\text{Given}$$

$$x < 0$$

$$\text{Maximize}(f, x) = -0.51$$

Впрочем, если взять начальное приближение, равное, например, 10, то никаких экстремумов не будет найдено вообще. Связано это, прежде всего, с очень высокой скоростью возрастания функции (то есть производная принимает слишком большие для эффективной работы численного алгоритма значения).

Равно как и при стандартной записи, функции экстремумов с вычислительным блоком крайне чувствительны к величинам начальных приближений. Так, попробуем найти максимумы рассматриваемой функции, выделив интервал их локализации:

$$x := 0$$

$$f(x) := -x^7 + 4x^3 - 3 \cdot x$$

$$\text{Given}$$

$$-3 < x < 3$$

$$\text{Maximize}(f, x) = -0.51$$

$$x := 1$$

$$f(x) := -x^7 + 4x^3 - 3 \cdot x$$

$$\text{Given}$$

$$-3 < x < 3$$

$$\text{Maximize}(f, x) = -3$$

При начальном приближении x=0 один из локальных максимумов был найден верно. Но при попытке определить экстремум из положительной области (x=1.077) был выдан довольно неожиданный результат (x=–3), хотя точка приближения была выбрана предельно близко к нему. Фактической ошибки, конечно, при этом допущено не было, так как на левой границе выбранного интервала функция действительно принимает свое наибольшее значение. Но как же в такой ситуации определить именно локальный максимум? Испробовав десяток начальных приближений, приходим к выводу, что при данных условиях найти правый экстремум весьма проблематично, хотя при стандартной записи без каких-либо дополнительных условий функция Maximize при использовании той же начальной точки находила его очень легко.

Таким образом, делая вывод, можно утверждать, что, если на интервале расположено несколько локальных экстремумов, лучше не рисковать и использовать стандартную

запись функций Maximize и Minimize. Если же экстремум только один, то использование вычислительного блока — очень эффективный и простой способ определения максимума или минимума. Но в любом случае проверка результата обязательна.

Грубой ошибкой функций Maximize и Minimize является и то, что в качестве экстремумов они выдают абсциссы всех точек, в которых первая и вторая производные имеют нулевые значения, однако, как вы помните, для ряда функций такие точки могут быть и не экстремальными.

Пример 13.6. Рассмотрим функцию (рис. 13.7)

$$f(x) := x^3 - 1$$

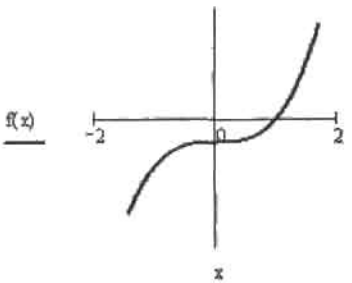


Рис. 13.7. График рассматриваемой функции

Функция эта возрастает на всем промежутке своего определения и никаких экстремумов не имеет. Но в точке x=0 ее первая производная ($3x^2$) и вторая производная (6x) равны 0. А это означает, что алгоритм Mathcad мало того, что определит ее в качестве точки экстремума, так и к тому же отнесет ее одновременно и к минимуму, и к максимуму:

$$x := 0$$

$$f(x) := x^3 - 1$$

$$\text{Maximize}(f, x) = 0 \qquad \text{Minimize}(f, x) = 0$$

13.3.2. Экстремум функции нескольких переменных

Различие между поиском экстремума функции одной переменной и функции нескольких переменных приблизительно такое же, как между решением уравнения с одним неизвестным и системы уравнений. То есть если максимум или минимум функции, зависящей от одной переменной, можно определить, найдя нули ее производной, то для того чтобы узнать координаты точки экстремума функции нескольких переменных, придется решить систему уравнений (необходимым условием существования в данной точке экстремума функции нескольких переменных является равенство всех ее частных производных 0).

Функции Maximize и Minimize могут быть использованы и для нахождения экстремумов функций нескольких переменных.

Пример 13.7. Найти экстремумы функции двух переменных (рис. 13.8)

$$f(x,y) := x^3 + y^3 - [5(x+y) + 4]$$

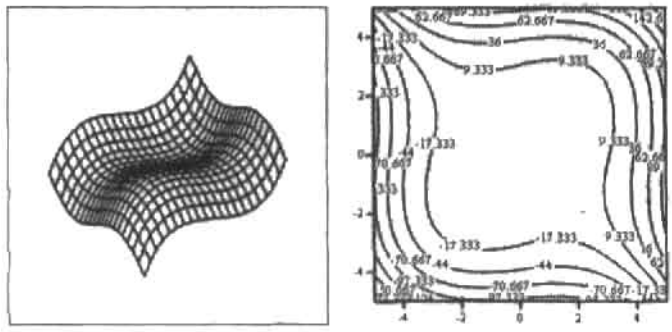


Рис. 13.8. График исследуемой функции двух переменных

Функция эта симметрична относительно своих переменных, то есть ее частные производные равны при любых значениях переменных. Из графиков следует, что она имеет локальный минимум и локальный максимум. Найти их можно очень просто и без помощи компьютера: частные производные равны $3r^2-5$ (r=x, y), откуда локальные экстремумы должны иметь координаты (1.291, 1.291) или (–1.291, –1.291). Попробуем теперь их найти с помощью функций Maximize и Minimize.

$$x := 1 \qquad y := 1$$

$$\text{Maximize}(f,x,y) = \begin{pmatrix} -1.291 \\ -1.291 \end{pmatrix} \qquad \text{Minimize}(f,x,y) = \begin{pmatrix} 1.291 \\ 1.291 \end{pmatrix}$$

Результат полностью соответствует расчетам, сделанным нами ранее.

Аналогично функциям одного переменного, Maximize и Minimize могут быть встроены в вычислительный блок. При этом можно вводить целый ряд дополнительных условий, что может быть весьма удобно для определения нужного значения экстремума. Однако, задав какие-то дополнительные условия, нужно быть очень внимательным к результату.

Пример 13.8. Рассмотрим функцию (рис. 13.9)

$$f(x,y) := \sin(x) + \sin(y)$$

Функция эта имеет бесконечное множество минимумов и максимумов. Попробуем, однако, найти минимум из того условия, что x+y=0:

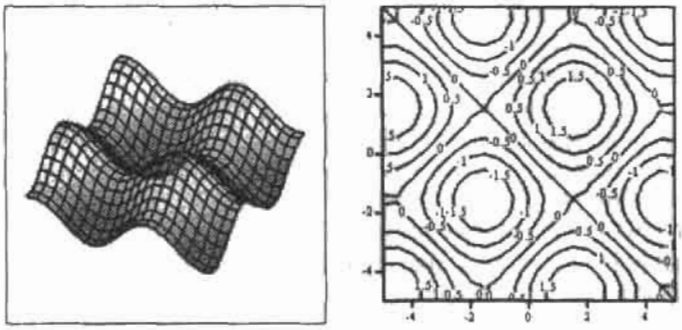
$$f(x,y) := \sin(x) + \sin(y)$$

$$x := 1 \qquad y := 1$$

$$\text{Given}$$

$$x + y = 0$$

$$\text{Minimize}(f,x,y) = \begin{pmatrix} 0.1 \\ -0.1 \end{pmatrix}$$



f f

Рис. 13.9. Поверхность и контурный график функции f(x,y)=sin(x)+sin(y)

Итак, в качестве точки локального минимума была определена точка M=(0,1, −0,1). Однако обе частные производные в этой точке равны косинусу соответствующей переменной и принимают значение 0.995, то есть не выполняется необходимое условие локального экстремума. Поэтому, естественно, в данной точке никакого минимума нет и быть не может. Так что в этом случае Minimize допустила ошибку, причем весьма грубую.

Анализируя результаты проделанной работы, можно сделать вывод: следует стараться не усложнять решаемую с помощью Mathcad проблему. Так, введение дополнительного условия в рассмотренном выше примере, привело к получению довольно грубой и глупой ошибки. В то же время совсем не сложно доказать, что вычислить экстремум, удовлетворяющий условию x+y=0, нельзя (подумайте, почему?).

Впрочем, если записать более согласованную систему, то функции экстремумов придут все-таки к правильным выводам.

Пример 13.9. Найти условный экстремум функции f(x,y)=sin(x)−sin(y) при x+y=0

$$f(x,y) := \sin(x) - \sin(y)$$

$$x := 1 \qquad y := 1$$

Given

$$x + y = 0$$

$$\text{Minimize}(f,x,y) = \begin{pmatrix} -1.571 \\ 1.571 \end{pmatrix}$$

В заключение еще раз повторю, что к результату работы функций Maximize и Minimize следует относиться предельно осторожно и по возможности проводить проверку, иначе частые ошибки практически неизбежны.

13.4. Линейное программирование (решение задач оптимизации)

Линейным программированием называется раздел математики, который занимается проблемами минимизации линейной функции конечного количества переменных,

удовлетворяющих набору ограничений в виде линейных уравнений или неравенств. Линейное программирование является основным аппаратом в теории оптимального принятия решений в первую очередь в задачах экономического планирования.

Впервые задачи линейного программирования сформулировал советский математик Л. В. Канторович, за что впоследствии получил Нобелевскую премию по экономике. В настоящее время для решения задач данного типа разработано сложнейшее программное обеспечение, позволяющее эффективно анализировать входные данные больших объемов и получать достоверный результат.

В общем виде задача линейного программирования формулируется следующим образом: необходимо определить неизвестные величины $x_1, x_2, ...x_n$, при которых функция

$$f(x) = c_1 \cdot x_1 + c_2 \cdot x_2 + ... + c_n \cdot x_n$$

называемая целевой, принимает минимальное значение при условии принадлежности переменных некоторой области, определяемой системой линейных равенств

$$a_{11} \cdot x_1 + a_{12} \cdot x_2 + ... + a_{1n} \cdot x_n = b_1$$

$$a_{21} \cdot x_1 + a_{22} \cdot x_2 + ... + a_{n2} \cdot x_n = b_2$$

.....

$$a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + ... + a_{mn} \cdot x_n = b_m$$

и их неотрицательности:

$$x_1 \geq 0 \qquad x_2 \geq 0 \qquad ... \qquad x_n \geq 0$$

Описанная формулировка задачи линейного программирования называется канонической. Именно к такому виду должна быть приведена задача, чтобы для ее решения можно было применить один из численных методов оптимизации. На практике же ограничения для переменных могут быть представлены в форме равенств, неравенств различных знаков и даже комбинации равенств и неравенств. Однако все они путем несложных приемов сводятся к ограничениям двух видов: равенствам и условиям неотрицательности.

Наиболее распространенным и универсальным методом решения задач оптимизации является так называемый симплекс-метод, позволяющий найти оптимальное решение за (2–3)m итерации, где m — количество равенств-ограничений.

Решение задач линейного программирования в Mathcad имеет несколько тонкостей, связанных прежде всего с заданием условия. В остальном же проблемы, которые могут возникнуть при работе с ними, связаны со слабостями функций экстремумов Maximize и Minimize и будут характерны для всех задач, в которых функции эти используются. Поэтому останавливаться на них, ввиду того, что способы разрешения таких трудностей были весьма подробно рассмотрены в двух предыдущих подразделах, мы не будем.

Следуя избранному стилю, изучим решение задач линейного программирования, просто найдя условие оптимизации конкретной практической задачи.

Одним из самых распространенных типов задач линейного программирования являются задачи, связанные с минимизацией транспортных или складских расходов, максимизацией прибыли за счет удачного распределения товара. Типичная задача оптимизации ставит условие приблизительно следующим образом.

Пусть в районе есть три свинофермы (в колхозе «Путь Ильича», в совхозе «Подольский» и у фермера Забутько). Предприниматель Иванов хочет купить у них мясо и перераба-

тать его в тушенку для последующей продажи службам армии. Сделать это можно на двух мясокомбинатах, расположенных в соседних районах. Мясокомбинаты государственные, поэтому и цена переработки одинакова. И совхоз, и колхоз, и фермер согласны продать свинину по одной и той же цене. Проблема выбора переработчика в том, что один из них, расположенный ближе к производителям, может принять очень ограниченный заказ (всего 20 т). Другой же, хоть и может переработать любое количество мяса, находится довольно далеко. Перевозка же свиней на 1 км увеличивает стоимость сырья на $1,2 на каждую тонну. Задача состоит в том, чтобы распределить купленных свиней между перерабатывающими предприятиями так, чтобы транспортные расходы были минимальными.

Информацию о количестве мяса, купленного у каждого производителя, и расстоянии от последних до каждого из мясокомбинатов можно узнать из табл. 13.1.

Таблица 13.1. Данные о поставщиках и переработчиках

Показатели	Колхоз	Совхоз	Фермер
Куплено мяса, тонн	12	19	24
Расстояние до первого мясокомбината, км	32	12	54
Расстояние до второго мясокомбината, км	160	144	199

Итак, попробуем найти для предпринимателя оптимальный путь распределения сырья с помощью Mathcad.

1. Для начала представим исходные данные в наиболее приемлемом для обработки программой виде — как матрицы или векторы. В первую очередь зададим вектор F распределения сырья и вектор MK объемов возможной приемки данного сырья мясокомбинатами. Очевидно, что, так как первый мясокомбинат ближе ко всем производителям, то везти на него сырье выгоднее, поэтому загрузить его следует по максимуму. То мясо, которое не сможет взять первый мясокомбинат, придется везти на второй.

$$F := \begin{pmatrix} 12 \\ 19 \\ 24 \end{pmatrix} \qquad MK := \begin{pmatrix} 20 \\ 35 \end{pmatrix}$$

$$\sum F = 55 \qquad \sum MK = 55$$

2. Далее зададим матрицу расстояний, в строках которой будут расположены соответствующие расстояния от производителей до данного мясокомбината. На этом же этапе определим коэффициент транспортных расходов, равный увеличению стоимости продукции при перевозке тонны сырья на 1 км:

$$TR := 1.2$$

$$RasP := \begin{pmatrix} 32 & 12 & 54 \\ 160 & 144 & 199 \end{pmatrix}$$

3. Когда все исходные данные определены, нужно задать функцию, минимизация которой будет производиться. Функция эта, исходя из условий задачи, должна определять транспортные расходы. Ее переменные, очевидно, должны быть определены как объемы сырья, перевезенные от каждого производителя к каждому переработчику. Нетрудно догадаться, что переменных таких будет 6 (с каждой из трех ферм свиньи могут быть перевезены на два мясокомбината). Сумма же произведений их значений и величин стоимости перевозки 1 т по данному маршруту определит полные транспортные расходы:

$$f(x) := \sum_{i=0}^{last(MK)} \sum_{j=0}^{last(F)} RasP_{i,j} \cdot x_{i,j} \cdot TR$$

4. Далее создаем вычислительный блок введением ключевого слова Given. В нем требуется определить все условия распределения переменных. Так, первые три уравнения устанавливают связь между двумя переменными, соответствующими объемам сырья, отправленным каждому из производителей, с общим количеством купленного мяса:

$$x_{0,0} + x_{1,0} = F_0$$

$$x_{0,1} + x_{1,1} = F_1$$

$$x_{0,2} + x_{1,2} = F_2$$

Еще два уравнения связывают количество сырья, которое предстоит переработать каждому из мясокомбинатов, с переменными, отвечающими за долю каждого из производителей в этом количестве:

$$x_{0,0} + x_{0,1} + x_{0,2} = MK_0$$

$$x_{1,0} + x_{1,1} + x_{1,2} = MK_1$$

Также при определении условий требуется ввести ограничение на отрицательные значения переменных. Если этого не сделать, то почти наверняка матрица решений будет содержать отрицательные элементы, что делает такое решение заведомо неверным:

$$x_{0,0} \geq 0 \qquad x_{0,1} \geq 0 \qquad x_{0,2} \geq 0$$
$$x_{1,0} \geq 0 \qquad x_{1,1} \geq 0 \qquad x_{1,2} \geq 0$$

5. Как вы помните, обязательным условием работы вычислительного блока является определение начальных значений переменных. Однако в случае работы с системами подобного рода совсем не нужно приближать все шесть переменных. Достаточно это сделать для одной (выражаясь более корректно, для одного элемента матрицы переменных) — с наибольшими значениями индексов. Это поможет одновременно и определить порядок неизвестных, и задать размерность матрицы ответа. Так, в случае данной задачи выше ключевого слова Given указываем:

$$x_{1,2} := 12$$

6. Далее, используя функцию Minimize, определяем матрицу, содержащую информацию о наиболее выгодном распределении сырья:

$$S := \text{Minimize}(f, x)$$

$$S = \begin{pmatrix} 0 & 0 & 20 \\ 12 & 19 & 4 \end{pmatrix}$$

Проверим для начала, соответствует ли такое распределение имеющемуся реально сырью:

$$\sum\left(S^{T}\right)^{\langle 0\rangle} + \sum\left(S^{T}\right)^{\langle 1\rangle} = 55$$

Да, это действительно так. Определим теперь сумму транспортных расходов. Наиболее просто это можно сделать, перемножив соответствующие друг другу элементы матрицы решения и матрицы расстояний, и умножив это произведение на коэффициент транспортных расходов. Очевидно, что такие перемножения нельзя задать с помощью традиционного перемножения матриц. Для того чтобы произвести эту операцию нужным нам образом, воспользуемся специальным оператором векторизации (Vectorize) панели Matrix (Матричные) (подробнее об этом операторе вы можете прочитать в гл. 3):

$$Pay := \left(\overrightarrow{(RasP \cdot S)}\right) \cdot TR$$

$$Pay = \begin{pmatrix} 0 & 0 & 1.296 \times 10^{3} \\ 2.304 \times 10^{3} & 3.283 \times 10^{3} & 955.2 \end{pmatrix}$$

$$SumPay := \sum\left(Pay^{T}\right)^{\langle 0\rangle} + \sum\left(Pay^{T}\right)^{\langle 1\rangle}$$

$$SumPay = 7.838 \times 10^{3}$$

Анализируя матрицу ответа, нельзя не согласиться, что распределить сырье по такому принципу было бы весьма разумно.

При решении задач линейного программирования принципиальным условием успеха является то, какой численный алгоритм вы используете. Впрочем, правильный выбор сделать совсем не сложно. Само прилагательное «линейное» подсказывает, какой из методов следует выбрать в контекстном меню функции Minimize: Linear. Если же вы предпочтете решать задачу такого рода нелинейными методами, то возможен целый ряд дополнительных проблем, связанных с их большей чувствительностью к начальному приближению и меньшей надежностью нахождения решения.

Конечно, решенная задача — очень простая по сравнению со многими из тех проблем, которые могут быть решены с помощью приемов линейного программирования. В случае многих задач нам бы пришлось сканировать промежуток одной или нескольких переменных, чтобы из множества минимумов или максимумов выбрать наибольший. Но разбирать такие примеры в этой книге мы не будем: все ходы, которые при этом нужно использовать, уже отлично вам знакомы, и в случае необходимости вы напишете соответствующий алгоритм самостоятельно.

Глава 14. Дифференциальные уравнения

Пожалуй, мы будем не столь уж не правы, назвав дифференциальные уравнения самой интересной, но и при этом самой сложной проблемой, решаемой прикладной математикой. Вся современная физика, техника и даже, в некоторой мере, экономика используют их для моделирования реальных процессов. И те точность и феноменальная предсказательная способность, которыми обладают дифференциальные уравнения, не могут не поражать.

Все дифференциальные уравнения можно разделить на две большие группы. Если в уравнение входят производные только по одной переменной, то оно называется обыкновенным дифференциальным уравнением (ordinary differential equation) — ОДУ. Если же его образуют производные по разным переменным, то оно называется дифференциальным уравнением в частных производных или, в более традиционной терминологии, уравнением математической физики (partial differential equation). В Mathcad имеются встроенные функции для решения дифференциальных уравнений двух типов.

В этой главе мы обсудим встроенные средства Mathcad для решения ОДУ и их систем, поговорим о краевых задачах, продемонстрируем, как можно в некоторых случаях найти решение в аналитическом виде. Отдельным вопросом разберем особенности жестких систем дифференциальных уравнений и дифференциальные уравнения математической физики. Кроме того, чтобы понять специфику решения различных уравнений средствами Mathcad, мы уделим особое внимание численным методам, лежащим в основе работы некоторых встроенных функций.

14.1. Аналитическое решение ОДУ и систем ОДУ

Возможности символьного решения дифференциальных уравнений в Mathcad, к сожалению, довольно ограничены. Как известно, решением любого дифференциального уравнения является функция, обращающая это уравнение в тождество, поэтому найти напрямую, подобно алгебраическому уравнению, корни ОДУ, каким бы оно ни было простым, нельзя в принципе. Однако высочайший вычислительный потенциал символьного процессора в области интегральных преобразований, решения алгебраических уравнений и интегрирования позволяет в некоторых случаях довольно красиво и очень легко находить корни несложных ОДУ и даже систем ОДУ. О том, как это делается, мы и поговорим в этом разделе.

14.1.1. Решение дифференциальных уравнений с применением преобразования Лапласа

Если дифференциальное уравнение, которое вам нужно решить, имеет постоянные коэффициенты, то предельно просто можно найти его корни, используя преобразование Лапласа.

Математический смысл преобразования Лапласа заключается в установлении взаимно однозначного соответствия между функцией действительной переменной f(t) (в терминологии операционного исчисления — оригиналом) и некоторой функцией комплексной переменной F(s), называемой изображением. Функция F(s) определяется равенством, которое получило название интеграла Лапласа:

$$F(s) = \int_0^{\infty} f(x) \cdot \exp(-s \cdot x)\, dx$$

Преобразование Лапласа широко используется в связи с тем, что многие математические операции проще проводить над изображением, чем над оригиналом (например, дифференцированию в оригинале соответствует умножение в изображении).

Преобразование Лапласа переводит дифференциальное уравнение в соответствующее алгебраическое, аналитически решив которое и применив к нему операцию обратного преобразования Лапласа, можно получить решение ОДУ в общем виде. Именно эту идею мы и используем для того, чтобы найти корни дифференциального уравнения с постоянными коэффициентами (если они будут зависеть от каких-то функций, то, скорее всего, символьный процессор просто не сможет решить полученное алгебраическое уравнение — если, конечно, системе удастся провести само преобразование).

В принципе, возможности решения ОДУ и систем ОДУ с помощью преобразования Лапласа довольно ограничены, на что есть вполне объективные причины. Интеграл Лапласа является несобственным и сходится лишь в том случае, когда функция f(x) удовлетворяет следующим условиям:

- f(x), равно как и ее производные, непрерывна для всех неотрицательных x, однако на любом конечном интервале допускается наличие конечного количества точек разрыва 1-го рода, в которых это условие не выполняется;
- при x<0 функция принимает нулевые значения: f(x)=0;
- f(x) возрастает медленнее экспоненциальной функции $M \cdot e^{ax}$. При x>0 существуют положительные M и a, при которых выполняется неравенство $|f(x)| \le M \cdot e^{ax}$.

Несмотря на довольно большое количество ограничений, с помощью преобразования Лапласа можно найти корни практически всех дифференциальных уравнений, которые могут встретиться в практикуме по высшей математике технического или гуманитарного вуза. Убедимся в этом на конкретных примерах.

Пример 14.1. Найти корни неоднородного дифференциального уравнения второго порядка X''+4X'=12cos(2t)

Приступая к решению уравнения, получим, прежде всего, его изображение с помощью прямого преобразования Лапласа. Для этого используем специальный оператор laplace (Лапласа) панели Symbolic (Символьные):

▪ laplace, ▪ →

В левом маркере этого оператора должно быть определено преобразуемое выражение (обязательно без «=», то есть все члены из правой части уравнения должны быть перенесены в левую), в правом — переменная, по которой будет осуществляться преобразование (в нашем случае — t):

$$\frac{d^2}{dt^2}X(t) + 4X(t) - 12\cos(2t)\ \text{laplace},t \rightarrow$$

$$\rightarrow s\cdot(s\cdot\text{laplace}(X(t),t,s) - X(0)) - \left|\begin{matrix} t \leftarrow 0 \\ \frac{d}{dt}X(t)\end{matrix}\right. + 4\cdot\text{laplace}(X(t),t,s) - 12\cdot\frac{s}{\left(s^2+4\right)}$$

Далее следует упростить полученное громоздкое выражение. Для этого переменную laplace(X(t),t,s) (которая соответствует корню ОДУ) заменим, например, на z, более воспринимаемую визуально. Кроме того, в изображении присутствует довольно странное определение начального условия для первой производной:

$$\left|\begin{matrix} t \leftarrow 0 \\ \frac{d}{dt}X(t)\end{matrix}\right.$$

Проводить дальнейшие преобразования при наличии такого рода элементов невозможно, поэтому введем в выражение обычную константу, например X'0, и изменим обозначение начального условия для функции с X(0) на X0. В результате получим:

$$s\cdot(s\cdot z - X0) - X'0 + 4\cdot z - 12\cdot\frac{s}{\left(s^2+4\right)}$$

После того как изображение ОДУ приведено в приемлемый для дальнейших операций вид, его нужно разрешить относительно z. Для этого воспользуемся специальным оператором символьного решения уравнений solve панели Symbolic (Символьные):

$$s\cdot(s\cdot z - X0) - X'0 + 4\cdot z - 12\cdot\frac{s}{\left(s^2+4\right)}\ \text{solve},z \rightarrow$$

$$\rightarrow \frac{\left(s^3\cdot X0 + 4\cdot s\cdot X0 + s^2\cdot X'0 + 4\cdot X'0 + 12\cdot s\right)}{\left(s^2+4\right)^2}$$

Чтобы получить решение дифференциального уравнения, к полученному решению-изображению нужно применить операцию обратного преобразования Лапласа. Сделать это можно с помощью оператора invlaplace панели Symbolic (Символьные):

$$\frac{\left(s^3\cdot X0 + 4\cdot s\cdot X0 + s^2\cdot X'0 + 4\cdot X'0 + 12\cdot s\right)}{\left(s^2+4\right)^2}\ \text{invlaplace},s \rightarrow$$

$$\rightarrow 3\cdot t\cdot\sin(2\cdot t) + \frac{1}{2}\cdot X'0\cdot\sin(2\cdot t) + X0\cdot\cos(2\cdot t)$$

В полученном громоздком выражении желательно привести подобные слагаемые, что можно сделать, задействовав оператор collect панели Symbolic (Символьные):

$$3 \cdot t \cdot \sin(2 \cdot t) + \frac{1}{2} \cdot X'0 \cdot \sin(2 \cdot t) + X0 \cdot \cos(2 \cdot t) \text{ collect}, \sin(2 \cdot t) \rightarrow$$

$$\rightarrow \left(3 \cdot t + \frac{1}{2} \cdot X'0\right) \cdot \sin(2 \cdot t) + X0 \cdot \cos(2 \cdot t)$$

Константы X0 и X'0 входят в независимые произвольные постоянные, определяющие вид искомой функции. По сути, X0 и X'0 есть начальные условия. Если задать их численно, мы получим задачу Коши, которая, как известно, имеет единственное решение. Параллельно записав полученное выражение как функцию от t, можно построить соответствующий график (рис. 14.1).

$$X0 := 0 \qquad X'0 := 1$$

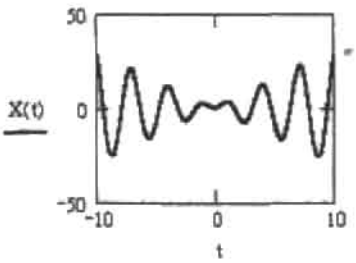


Рис. 14.1. Интегральная кривая аналитического решения ОДУ

Как вы уже убедились, с помощью преобразования Лапласа можно найти корни уравнения со специальной правой частью, подобного рассмотренному нами. С решением однородных уравнений приведенным способом Mathcad справляется ничуть не хуже.

Уже при решении уравнений второго порядка приходится оперировать огромными выражениями. Как же поступить, если перед нами стоит задача найти корни уравнения высшего, например, 4-го порядка? Ведь выражение, полученное при попытке преобразовать подобное уравнение, вряд ли попадет даже в самое широкое поле зрения. В этом случае придется воспользоваться свойством линейности преобразования Лапласа, благодаря которому можно значительно упростить решение дифференциальных уравнений высших порядков. (Например, для линейной комбинации функций $ay(x)+bf(x)$ существует аналогичная линейная комбинация их изображений: $aY(s)+bF(s)$).

Пример 14.2. Решить линейное однородное дифференциальное уравнение $y^{IV}-4y'''+6y''-4y'+y=0$

Производим над функцией y(t) и каждой из ее производных, присутствующих в уравнении, преобразование Лапласа:

$$\frac{d^4}{dt^4}y(t) \text{ laplace}, t \rightarrow s \cdot \left[s \cdot \left[s \cdot (s \cdot \text{laplace}(y(t), t, s) - y(0)) - \left| \begin{matrix} t \leftarrow 0 \\ \frac{d}{dt}y(t) \end{matrix} \right| \right] - \left| \begin{matrix} t \leftarrow 0 \\ \frac{d^2}{dt^2}y(t) \end{matrix} \right| \right] - \left| \begin{matrix} t \leftarrow 0 \\ \frac{d^3}{dt^3}y(t) \end{matrix} \right|$$

$$\frac{d^3}{dt^3}y(t)\ \text{laplace},t\ \rightarrow s\cdot\left[s\cdot(s\cdot\text{laplace}(y(t),t,s)-y(0))-\left|\begin{matrix}t\leftarrow 0\\ \frac{d}{dt}y(t)\end{matrix}\right.\right]-\left|\begin{matrix}t\leftarrow 0\\ \frac{d^2}{dt^2}y(t)\end{matrix}\right.$$

$$\frac{d^2}{dt^2}y(t)\ \text{laplace},t\ \rightarrow s\cdot(s\cdot\text{laplace}(y(t),t,s)-y(0))-\left|\begin{matrix}t\leftarrow 0\\ \frac{d}{dt}y(t)\end{matrix}\right.$$

$$\frac{d}{dt}y(t)\ \text{laplace},t\ \rightarrow s\cdot\text{laplace}(y(t),t,s)-y(0)$$

$$y(t)\ \text{laplace},t\ \rightarrow \text{laplace}(y(t),t,s)$$

Аналогично предыдущему примеру, заменим функцию laplace(y(t), t, s) на z и запишем начальные условия для производных в классическом виде. Для удобства дальнейших вычислений представим полученные выражения в виде вектора:

$$V:=\begin{bmatrix}s\cdot[s\cdot[s\cdot(s\cdot z-y0)-y'0]-y''0]-y'''0\\ s\cdot[s\cdot(s\cdot z-y0)-y'0]-y''0\\ s\cdot(s\cdot z-y0)-y'0\\ s\cdot z-y0\\ z\end{bmatrix}$$

Упростим элементы вектора, вызвав команду simplify меню Symbolic:

$$V:=\begin{pmatrix}s^4\cdot z-s^3\cdot y0-s^2\cdot y'0-s\cdot y''0-y'''0\\ s^3\cdot z-s^2\cdot y0-s\cdot y'0-y''0\\ s^2\cdot z-s\cdot y0-y'0\\ s\cdot z-y0\\ z\end{pmatrix}$$

Обратите внимание, как простым переопределением элементов можно упростить огромные выражения прямого преобразования Лапласа. Осуществлять же такого рода операцию нужно не только исходя из эстетики решения или экономии места, но и в связи с тем, что чем проще форма выражения, тем более эффективно с ней работает символьный процессор.

Поскольку преобразование Лапласа обладает свойством линейности, комбинация упрощенных изображений представляет собой алгебраическое уравнение, соответствующее дифференциальному. Решив его относительно z, мы получим изображение корня дифференциального уравнения:

$$V_0-4V_1+6V_2-4V_3+V_4\ \text{solve},z\ \rightarrow$$

$$\rightarrow\frac{\left(-4\cdot y0+s^3\cdot y0+s^2\cdot y'0+s\cdot y''0+y'''0-4\cdot s^2\cdot y0-4\cdot s\cdot y'0-4\cdot y''0+6\cdot s\cdot y0+6\cdot y'0\right)}{\left(-4\cdot s^3+6\cdot s^2-4\cdot s+s^4+1\right)}$$

Применим обратное преобразование Лапласа и приведем подобные слагаемые (последнюю операцию следует выполнять после того, как решение будет получено в развернутом виде, ведь предсказать заранее, какие слагаемые окажутся подобными, нельзя):

$$\frac{\left(-4\cdot y0 + s^3\cdot y0 + s^2\cdot y'0 + s\cdot y''0 + y'''0 - 4s^2\cdot y0 - 4s\cdot y'0 - 4y''0 + 6s\cdot y0 + 6y'0\right)}{\left(-4\cdot s^3 + 6s^2 - 4s + s^4 + 1\right)}\ \Bigg|\begin{matrix}\text{invlaplace, s}\\ \text{collect, exp(t), t}^3, \text{t}^2\end{matrix}\rightarrow$$

$$\rightarrow \left[\left(\frac{-1}{2}\cdot y''0 + \frac{1}{6}\cdot y'''0 - \frac{1}{6}\cdot y0 + \frac{1}{2}\cdot y'0\right)\cdot t^3 + \left(\frac{1}{2}\cdot y0 + \frac{1}{2}\cdot y''0 - y'0\right)\cdot t^2 + (y'0 - y0)\cdot t + y0\right]\cdot \exp(t)$$

Выражения в круглых скобках являются независимыми произвольными постоянными функции решения, обозначаемыми в математике как C_n. Найденная в результате функция является записанным пусть и не в самой удобной форме, но все же вполне корректным решением уравнения.

Абсолютно аналогичным способом можно найти корни и неоднородных дифференциальных уравнений высших порядков. Достаточно лишь преобразовать правую часть, а при решении алгебраического уравнения перенести соответствующее ей изображение в левую часть.

Используя преобразование Лапласа, можно решать системы ОДУ. Чтобы это сделать, вы должны найти корни системы алгебраических уравнений, составленной из полученных изображений уравнений системы ОДУ. Применив затем к найденным корням операцию обратного преобразования Лапласа, вы получите искомые аналитические решения системы.

Задачу Коши для ОДУ или системы ОДУ можно решить и несколько упрощенным способом, используя вычислительный блок Given-Find, но для этого нам придется вспомнить, как формируются изображения для производных функции-оригинала.

Пример 14.3. Найти решение задачи Коши для системы дифференциальных уравнений $x'=4x-3y+2e^t$, $y'=3x-2y-t^2$, $x(0)=-1$, $y(0)=0$

Вводим ключевое слово Given:

Given

Выполняем операцию преобразования Лапласа над производными и свободными членами уравнений (функциям x(t) и y(t) соответствуют изображения x и y):

$$\frac{d}{dt}x(t)\ \text{laplace, t} \rightarrow s\cdot \text{laplace}\,(x(t), t, s) - x(0)$$

$$\frac{d}{dt}y(t)\ \text{laplace, t} \rightarrow s\cdot \text{laplace}(y(t), t, s) - y(0)$$

$$2e^t\ \text{laplace, t} \rightarrow \frac{2}{(s-1)} \qquad t^2\ \text{laplace, t} \rightarrow \frac{2}{s^3}$$

В изображениях производных присутствуют начальные условия x(0) и y(0). Их необходимо указать в алгебраических уравнениях, к которым сводятся уравнения дифференциальные:

$$s\cdot x + 1 = 4x - 3y + \frac{2}{s-1} \qquad s\cdot y = 3x - 2y - \frac{2}{s^3}$$

Решаем полученную систему уравнений относительно x и y:

$$\text{find}(x,y) \rightarrow \begin{bmatrix} \dfrac{\left(-s^5 + 6\cdot s^3 + 6\cdot s + s^4 - 6\right)}{s^3\cdot\left(s^3 - 3\cdot s^2 + 3\cdot s - 1\right)} \\ \dfrac{\left(10\cdot s - 3\cdot s^4 - 2\cdot s^2 - 8 + 9\cdot s^3\right)}{s^3\cdot\left(s^3 - 3\cdot s^2 + 3\cdot s - 1\right)} \end{bmatrix}$$

Для получения функций решения проводим над полученными выражениями обратное преобразование Лапласа, группируя затем подобные слагаемые.

Функция x(t):

$$\frac{\left(-s^5 + 6\cdot s^3 + 6\cdot s + s^4 - 6\right)}{s^3\cdot\left(s^3 - 3\cdot s^2 + 3\cdot s - 1\right)} \left|\begin{matrix}\text{invlaplace},s\\ \text{collect},\exp(t)\end{matrix}\right. \rightarrow$$

$$\rightarrow \left(3\cdot t^2 + 5\cdot t - 19\right)\cdot\exp(t) + 3\cdot t^2 + 12\cdot t + 18$$

Функция y(t):

$$\frac{\left(10\cdot s - 3\cdot s^4 - 2\cdot s^2 - 8 + 9\cdot s^3\right)}{s^3\cdot\left(s^3 - 3\cdot s^2 + 3\cdot s - 1\right)} \left|\begin{matrix}\text{invlaplace},s\\ \text{collect},\exp(t)\end{matrix}\right. \rightarrow$$

$$\rightarrow \left(3\cdot t^2 + 3\cdot t - 20\right)\cdot\exp(t) + 4\cdot t^2 + 14\cdot t + 20$$

14.1.2. Решение линейных неоднородных дифференциальных уравнений с применением преобразования Фурье

Специфическим, но все же альтернативным преобразованию Лапласа способом можно решать линейные неоднородные дифференциальные уравнения, используя преобразование Фурье. Достоинством преобразования Лапласа является то, что с его помощью можно решать как однородные, так и неоднородные линейные уравнения. Однако у этого метода есть и существенный недостаток: при его использовании приходится оперировать огромными выражениями и делать множество замен и подстановок. К тому же полученное решение всегда представлено в довольно неудобной нестандартной форме. При использовании же преобразования Фурье этих проблем не возникает: процесс решения компактен, а искомая функция отображается в привычном виде.

Сущность преобразования Фурье заключается в переходе от функции действительной переменной f(x) — оригинала к изображению — комплексной функции F(y). Функция F(y) ставится в соответствие функции f(x) согласно формуле

$$F(y) = \frac{1}{\sqrt{2\pi}}\cdot\int_{-\infty}^{\infty} f(x)\, e^{i\cdot y\cdot x}\, dx$$

которая и носит название преобразования Фурье.

При преобразовании Фурье операция дифференцирования переходит в умножение на независимую переменную. Другими словами, если F(y) — изображение функции f, а G(y) — функции f', то G(y)=-iy·F(y). К функции f(x) предъявляются следующие требования: она должна быть гладкой (кусочно-гладкой) и абсолютно интегрируемой на всей числовой оси.

Как известно, общее решение линейного неоднородного уравнения есть сумма общего решения соответствующего однородного уравнения и частного решения неоднородного уравнения. К сожалению, с помощью преобразования Фурье нельзя найти общее решение однородного дифференциального уравнения, однако этим способом можно легко получить частное решение неоднородного уравнения любого порядка, не оперируя громоздкими выражениями, как в случае применения преобразования Лапласа. Общее же решение однородного уравнения можно найти, вычислив корни соответствующего характеристического многочлена.

Чтобы продемонстрировать преимущества использования преобразования Фурье, рассмотрим конкретное уравнение.

Пример 14.4. Проинтегрировать линейное неоднородное дифференциальное уравнение $y^{V}-3y'''+4y''+2y'-4y=\sin(x)$

Приступим к поиску частного решения. В первую очередь нам нужно получить изображение уравнения. Для этого воспользуемся оператором прямого преобразования Фурье fourier (Фурье) панели Symbolic (Символьные). Как и в случае преобразования Лапласа, все члены уравнения должны быть перенесены в левую часть, которая указывается в левом маркере оператора fourier. В правом маркере определяется переменная, по которой проводится преобразование.

$$\frac{d^5}{dx^5}y(x) - 3\cdot\frac{d^3}{dx^3}y(x) + 4\frac{d^2}{dx^2}y(x) + 2\frac{d}{dx}y(x) - 4\,y(x) - \sin(x)\ \text{fourier},x \rightarrow$$

$$\rightarrow i\cdot\omega^5\cdot\text{fourier}(y(x),x,\omega) + 3\cdot i\cdot\omega^3\cdot\text{fourier}(y(x),x,\omega) - 4\cdot\omega^2\cdot\text{fourier}(y(x),x,\omega) +$$

$$+ 2\cdot i\cdot\omega\cdot\text{fourier}(y(x),x,\omega) - 4\cdot\text{fourier}(y(x),x,\omega) + i\cdot\pi\cdot\Delta(\omega-1) - i\cdot\pi\cdot\Delta(\omega+1)$$

Здесь Δ — знаменитая дельта-функция Дирака (она определена для всех x и равна бесконечности в одной точке и нулю во всех остальных).

Для удобства заменим в последнем выражении переменную fourier(y(x),x,ω) на z, а затем решим полученное уравнение относительно z.

$$i\cdot\omega^5\cdot z + 3\cdot i\cdot\omega^3\cdot z - 4\cdot\omega^2\cdot z + 2\cdot i\cdot\omega\cdot z - 4\cdot z + i\cdot\pi\cdot\Delta(\omega-1) - i\cdot\pi\cdot\Delta(\omega+1)\ \text{solve},z \rightarrow$$

$$\rightarrow i\cdot\pi\cdot\frac{-\Delta(\omega-1)+\Delta(\omega+1)}{i\cdot\omega^5 + 2\cdot i\cdot\omega - 4 + 3\cdot i\cdot\omega^3 - 4\cdot\omega^2}$$

Найдем частное решение уравнения. Для этого применим к решению-изображению обратное преобразование Фурье, задействовав оператор invfourier панели Symbolic. Чтобы результат не был представлен в комплексной форме, воспользуемся оператором complex панели Symbolic.

$$i\cdot\pi\cdot\frac{-\Delta(\omega-1)+\Delta(\omega+1)}{i\cdot\omega^5 + 2\cdot i\cdot\omega - 4 + 3\cdot i\cdot\omega^3 - 4\cdot\omega^2}\ \left|\begin{array}{l}\text{invfourier},\omega\\ \text{complex}\end{array}\right. \rightarrow \frac{-2}{25}\cdot\sin(t) - \frac{3}{50}\cdot\cos(t)$$

Частное решение найдено. Теперь необходимо найти общее решение однородного уравнения, соответствующего нашему неоднородному: $y^v-3y'''+4y''+2y'-4y=0$. Составим для него характеристический многочлен и найдем его корни:

$$\text{roots} := x^5 - 3\cdot x^3 + 4\cdot x^2 + 2\cdot x - 4 \text{ solve}, x \rightarrow \begin{pmatrix} 1 \\ -2 \\ -1 \\ 1+i \\ 1-i \end{pmatrix}$$

Из курса высшей математики известно, что общее решение однородного уравнения определяется формулой

$$y = \sum_{i=1}^{n} C_n \cdot e^{\lambda_n \cdot x}$$

где n — порядок дифференциального уравнения, λ_n — корни характеристического многочлена (формула справедлива для случая отсутствия кратных корней). Имея вектор корней характеристического уравнения, несложно написать программку, которая представляла бы общее решение в таком виде:

$$\text{General\_solution(pol)} := \left| \begin{array}{l} a \leftarrow 0 \\ \text{for } i \in 0..\text{last(pol)} \\ \quad a \leftarrow a + C_i \cdot e^{\text{pol}_i \cdot x} \\ a \end{array} \right.$$

$$\text{General\_solution(roots)} \rightarrow C_0\cdot e^{x} + C_1\cdot e^{-2\cdot x} + C_2\cdot e^{-x} + C_3\cdot e^{(1+i)\cdot x} + C_4\cdot e^{(1-i)\cdot x}$$

Избавимся от комплексности в ответе, приведем подобные слагаемые и переопределим константы, чтобы записать общее решение однородного уравнения в стандартной форме.

$$C_0\cdot e^{x} + C_1\cdot e^{-2\cdot x} + C_2\cdot e^{-x} + C_3\cdot e^{(1+i)\cdot x} + C_4\cdot e^{(1-i)\cdot x} \left| \begin{array}{l} \text{complex} \\ \text{collect}, \cos(x), \sin(x), e^{x} \end{array} \right. \rightarrow$$

$$\rightarrow \left(C_3 + C_4\right)\cdot e^{x}\cdot\cos(x) + i\cdot\left(C_3 - C_4\right)\cdot e^{x}\cdot\sin(x) + C_0\cdot e^{x} + C_1\cdot e^{-2\cdot x} + C_2\cdot e^{-x}$$

$$C_3 + C_4 = k1 \qquad i\cdot\left(C_3 - C_4\right) = k2$$

$$k1\cdot e^{x}\cdot\cos(x) + k2\cdot e^{x}\cdot\sin(x) + C_0\cdot e^{x} + C_1\cdot e^{-2\cdot x} + C_2\cdot e^{-x}$$

Чтобы получить конечный результат —общее решение неоднородного уравнения, прибавим к общему решению однородного уравнения частное решение неоднородного.

$$\text{Solution} := k1 \cdot e^{x} \cdot \cos(x) + k2 \cdot e^{x} \cdot \sin(x) + C_0 \cdot e^{x} + C_1 \cdot e^{-2 \cdot x} + C_2 \cdot e^{-x} - \frac{2}{25} \cdot \sin(x) - \frac{3}{50} \cdot \cos(x)$$

Проверка показывает, что найденный нами ответ абсолютно верен:

$$\frac{d^5}{dx^5} y(x) - 3 \cdot \frac{d^3}{dx^3} y(x) + 4 \frac{d^2}{dx^2} y(x) + 2 \frac{d}{dx} y(x) - 4\, y(x) - \sin(x) \left| \begin{array}{l} \text{substitute}, y(x) = \text{Solution} \\ \text{simplify} \end{array} \right. \rightarrow 0$$

14.1.3. Интегрирование дифференциальных уравнений

С помощью преобразований Лапласа и Фурье можно решать уравнения лишь с постоянными коэффициентами, однако средствами Mathcad можно значительно упростить поиск корней дифференциальных уравнений, коэффициенты которых являются функциями. Как известно, решение любого дифференциального уравнения сводится к интегрированию, которое также выполняется системой очень просто.

Пример 14.5. Проинтегрировать дифференциальное уравнение

$$\left(1 + x^2\right) dy + \left(xy - \sqrt{1 + x^2} \cdot \sin(x)\right) dx = 0$$

Преобразуем уравнение в следующую форму:

$$\frac{dy}{dx} = -\frac{xy - \sqrt{1 + x^2} \sin(x)}{1 + x^2}$$

Обратим внимание на то, что в левой части получено выражение, задающее y'. В правой части для упрощения выражения стоит разделить числитель на знаменатель:

$$y' = -\frac{x}{1 + x^2} y + \frac{\sin(x)}{\sqrt{1 + x^2}}$$

Введем следующие замены:

$$p(x) = \frac{x}{1 + x^2} \qquad q(x) = \frac{\sin(x)}{\sqrt{1 + x^2}}$$

Мы получили линейное дифференциальное уравнение первого порядка общего вида y'+p(x)y–q(x)=0. Из курса высшей математики известно, что общее решение подобных уравнений ищется в виде

$$y(x) = e^{-\int p(x)\,dx} \left(\int q(x)\, e^{\int p(x)\,dx}\, dx + C \right)$$

Приступим к непосредственному решению уравнения. Подставим в формулу выражения для p(x) и q(x):

$$y(x) := e^{-\int \frac{x}{1+x^2}dx}\left(\int \frac{\sin(x)}{\sqrt{1+x^2}}\cdot e^{\int \frac{x}{1+x^2}dx}dx + C\right)$$

Символьно вычисляем корни уравнения. Геометрически они представляют собой семейство интегральных кривых, зависящее от параметра C:

$$y(x) \rightarrow \frac{1}{\left(1+x^2\right)^{\frac{1}{2}}}\cdot(-\cos(x)+C)$$

Решение однородных уравнений первого порядка сводится к замене переменных, превращающих их в уравнения с разделяющимися переменными. Последние удается проинтегрировать в Mathcad так же легко, как, в частности, и уравнения в полных дифференциалах.

Пример 14.6. Проинтегрировать дифференциальное уравнение

$$xdy - \left(1-\sqrt{x}\right)^3 dx = 0.$$

Построить интегральную кривую, проходящую через точку M(1; 7/3)

В два действия данное уравнение приводится к виду

$$dy = \frac{\left(1-\sqrt{x}\right)^3}{x}dx$$

Проинтегрировав обе части, мы найдем общее решение, определяющее однопараметрическое семейство интегральных кривых:

$$y(x) := \int \frac{\left(1-\sqrt{x}\right)^3}{x}dx$$

$$y(x) \rightarrow \ln(x) - 6\cdot x^{\frac{1}{2}} + 3\cdot x - \frac{2}{3}\cdot x^{\frac{3}{2}}$$

$$y(x) = \ln(x) - 6\cdot x^{\frac{1}{2}} + 3\cdot x - \frac{2}{3}\cdot x^{\frac{3}{2}} + C$$

Чтобы найти требуемое частное решение, необходимо определить фиксированное значение параметра C. Для этого подставим в последнее выражение координаты точки M, а затем разрешим полученное уравнение относительно C:

$$\frac{7}{3} - \ln(1) + 6 \cdot 1^{\frac{1}{2}} - 3 + \frac{2}{3} 1^{\frac{3}{2}} - C \text{ solve}, C \rightarrow 6$$

Записываем функцию решения с численным значением параметра C:

$$y(x) := \ln(x) - 6 \cdot x^{\frac{1}{2}} + 3 \cdot x - \frac{2}{3} \cdot x^{\frac{3}{2}} + 6$$

Строим интегральную кривую (рис. 14.2).

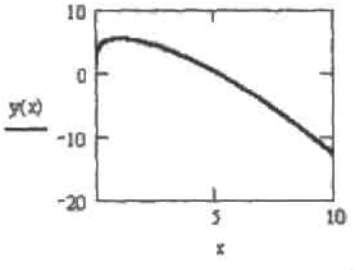


Рис. 14.2. Интегральная кривая решения, проходящая через точку M(1; 7/3)

14.2. Численное решение ОДУ в форме задачи Коши

Основная форма задания исходных условий, которая используется при моделировании реальных процессов с помощью дифференциальных уравнений, связана с определением значений всех низших производных в начальной точке интервала изменения переменной. Так, в физике ОДУ обычно описывают изменение изучаемой характеристики с течением времени, и начальные условия определяются в момент t=0. Заданные таким образом дифференциальные уравнения и системы дифференциальных уравнений называются задачами Коши.

Методы численного решения ОДУ в форме задачи Коши разработаны весьма досконально. Самыми популярными из них заслуженно являются алгоритмы Рунге–Кутта, успешно используемые для решения подавляющего большинства дифференциальных уравнений. Однако для некоторых задач (например, жестких систем химической кинетики) они неприменимы (или неэффективны), и тогда приходится применять более тонкие и сложные методы. Функции, реализующие наиболее важные из них, являются в Mathcad системными, что позволяет очень быстро и просто находить решение практически для любого ОДУ или системы ОДУ.

Решение линейных дифференциальных уравнений и их систем реализовано в Mathcad в двух формах: в виде вычислительного блока и в виде встроенных функций. Первая форма предпочтительнее с точки зрения наглядности представления решения и технической простоты, вторая же открывает куда более широкие возможности для влияния

на ход его получения и изучения полученных результатов. Так как, вне зависимости от выбора типа оформления, система использует для решения ОДУ одни и те же алгоритмы, то принципиального различия от того, какой из них вы предпочтете, в результатах вычислений не будет.

Далее мы обсудим принципы решения ОДУ и систем ОДУ, заданных в форме задачи Коши обоими альтернативными методами, а также отдельно поговорим о жестких задачах.

14.2.1. Вычислительный блок Given-Odesolve

С такой формой проведения расчетов, как вычислительный блок (Solve Block), мы уже встречались, когда разбирали методы численного решения алгебраических уравнений (функция find), минимизации невязки системы (функция minerr), а также определения экстремумов (функции Maximize и Minimize). Самый простой и наглядный способ численного решения линейных ОДУ и систем линейных ОДУ в Mathcad также использует вычислительный блок. Чтобы его реализовать, выполните следующую последовательность действий.

- Залайте вводное слово Given (Дано).
- Ниже вводного слова определите вид дифференциального уравнения или системы уравнений. Сделать это нужно, соблюдая следующие правила.
  - Дифференциальное уравнение (в том числе входящее в систему) должно быть строго линейным (то есть высшая производная в нем не должна иметь никаких сомножителей или степенных показателей). Подобное ограничение связано с особенностями используемого (по умолчанию) системой при решении ОДУ в форме вычислительного блока Given-Odesolve метода Рунге–Кутта 4-го порядка точности (о нем мы поговорим ниже). Если же ваше уравнение нелинейное, то использовать вычислительный блок нельзя.
  - Производные в выражении дифференциального уравнения могут быть заданы с помощью специальных операторов панели Calculus (Вычисления) (Shift+/ для оператора производной первого порядка и Shift+Ctrl+/ для оператора дифференцирования произвольного порядка). Подобная запись дифференциальных уравнений более принята в физике, а в математике традиционно производные в ОДУ обозначаются с помощью специальных меток-штрихов. При задании вычислительного блока можно использовать и такую форму отображения дифференциальных уравнений (соответствующий штрих вводится с помощью сочетания Ctrl+F7).
  - Искомые функции должны быть определены явно с указанием имени переменной, например f(x).
  - В качестве знака равенства внутри вычислительного блока следует использовать, как и в случае решения простых алгебраических уравнений, только логическое равенство (Bold Equal – Ctrl+=).
- Помимо самого дифференциального уравнения, внутри вычислительного блока необходимо задать и соответствующие начальные или граничные (в случае решения краевых задач) условия. В курсе высшей математики доказывается, что для того, чтобы получить явное решение ОДУ n-го порядка, для него следует определить n начальных условий. В случае задачи Коши в качестве начальных условий необходимо задать значения производных порядков n–1,n–2,...,0 на левой границе интервала изменения переменной. При определении производных в начальной точ-

ке используются те же правила, что и при задании выражения самого дифференциального уравнения.

- Когда ОДУ, начальные или граничные условия будут заданы, можно непосредственно приступать к вычислению значений искомой функции на нужном промежутке. Для этого следует задействовать специальную функцию odesolve([vector],t,b,[step]), где:
  - vector — вектор функций, относительно которых решается система дифференциальных уравнений. Вектор должен содержать имена функций без имени переменной. Данный параметр используется, когда вы решаете систему ОДУ, в случае же решения одиночного уравнения его следует опустить;
  - t — переменная, от которой зависит искомая функция. Необходимость задания этого параметра связана с тем, что, как это ни покажется странным, результат использования функции odesolve также является функцией, а не вектором некоторых числовых значений. Те возможности, которые открывает это обстоятельство, мы продемонстрируем немного ниже;
  - b — правая граница интервала поиска решения. Соответственно левая граница определяется вами при задании начальных условий. Обязательным условием является то, что b должно лежать правее a — это общее требование для корректно заданной задачи Коши при использовании всех численных методов;
  - step — этот параметр определяет количество шагов, используемых численным методом Рунге–Кутта. Задание step не является обязательным, и по умолчанию этот параметр определяется таким образом, чтобы длина шага была равна 0.1. В случае сложных уравнений правильное определение step может играть самую принципиальную роль для получения корректного решения. Однако стремиться во всех случаях сделать step минимальным не стоит, так как для большинства ОДУ это приведет лишь к резкому увеличению времени расчетов без значимого повышения их точности.

Наряду с дифференциальными уравнениями в блоке Given-Odesolve можно задавать и алгебраические ограничения, например, y(t)+z(t)=w(t), что позволяет ввести в систему дополнительную неизвестную функцию w(t), которая должна быть определена odesolve наряду с y(t) и z(t). Ограничения же в виде неравенств задавать нельзя.

Приведем пример решения ОДУ первого порядка. Выберем уравнение таким образом, чтобы оно было не решаемо аналитическими методами (следует заметить, что всевозможными символьными преобразованиями можно привести к удобному для интегрирования виду предельно ограниченную группу дифференциальных уравнений).

Пример 14.7. Численное решение ОДУ первого порядка (рис. 14.3)

Given

$$\frac{d}{dt}y(t) = \left(y(t) - y(t)^2 + t\right)\cdot\sin\left(t + \frac{1}{y(t)}\right)\cdot\cos\left(t - \frac{1}{y(t)}\right)$$

$$y(0) = 1$$

$$y := \text{odesolve}(t, 40)$$

$y(6.11115) = 2.5930254520046692$ $y(39.89679) = 6.6616193367807526$

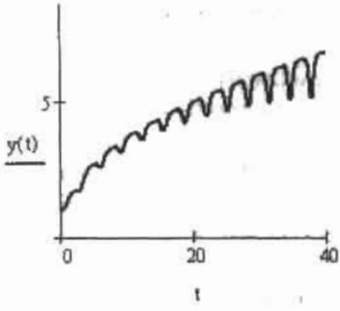


Рис. 14.3. График численного решения ОДУ первого порядка

Обратите внимание, что при решении дифференциального уравнения с помощью функции odesolve существует возможность определения значений искомой функции и в тех точках, которые не являются узловыми (то есть непосредственно в ходе работы численного алгоритма эти значения просчитаны не были). Достигается же это за счет задания между каждой парой точек интерполирующего полинома (точнее, кубического сплайна). Очевидно, что точность такого предсказания будет тем выше, чем меньше величина шага и чем более плавно и предсказуемо изменяется функция решения.

Кстати, и график решения ОДУ вычислительным блоком Given-Odesolve задается таким образом, как будто оно является непрерывной функцией (см. рис. 14.3).

Аналогичным образом, как и ОДУ первого порядка, решаются и линейные дифференциальные уравнения, содержащие производные более высокого порядка.

Пример 14.8. Решение ОДУ третьего порядка (рис. 14.4)

Given

$$y'''(t) + y''(t) + y'(t) + y(t) = -t \cdot \sin(t)$$
$$y''(0) = 2 \qquad y'(0) = 1 \qquad y(0) = 0$$

$$y := \text{odesolve}(t, 30, 30000)$$

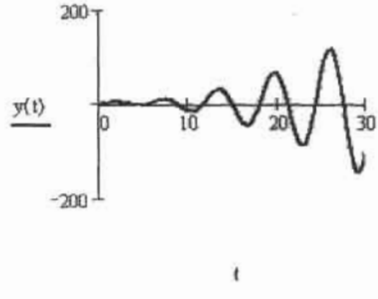


Рис. 14.4. Модель резонирующей колебательной системы

Изучая пример 14.8, вы, наверное, заметили, что дифференциальное уравнение в нем задано иначе, чем в примере 14.7. Вообще же, обе эти формы абсолютно эквивалентны,

и выбор одной из них должен определяться прежде всего вашими предпочтениями и той областью, к которой относится решаемое уравнение.

Важным условием успеха при численном решении дифференциальных уравнений является правильный выбор величины шага. Так, если он будет определен недостаточно малым, то найденное решение может быть весьма и весьма далеким от истинного. Приведем пример ошибочного решения ОДУ при слишком большой ширине шага.

Пример 14.9. Осцилляция при недостаточно малой длине шага (рис. 14.5)

Given

$$y'(t) = 1 - y(t)$$

$$y(5) = 5$$

y := odesolve(t, 50, 10) y1 := odesolve(t, 50, 1000)

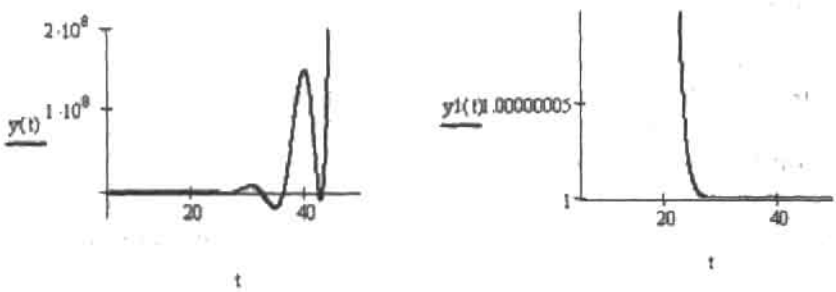


Рис. 14.5. Ошибочное (слева) и верное (справа) решение дифференциального уравнения

Конечно, в приведенном примере возникшая ошибка (см. рис. 14.5) связана с искусственно завышенной длиной шага, и даже при использовании его величины, принятой по умолчанию (step=0.1), решение было бы найдено верно.

На практике бывает необходимо решать такие дифференциальные уравнения, нужное значение шага для которых может быть совсем не очевидно (и step=0.1 для них будет порождать еще более значительные колебания или отклонения, чем step=4.5 в примере 14.9). В тех случаях, когда у вас возникают сомнения в верности заданной вами длине шага, вы можете руководствоваться следующим простым правилом: ее можно принять как корректную, если уменьшение ее величины в 10 раз не приводит к изменению результата в некоторой точке в пределах количества десятичных знаков, отвечающего нужному уровню точности.

Как уже было отмечено выше, по умолчанию вычислительный блок Given-Odesolve использует популярный метод Рунге–Кутта 4-го порядка с постоянным шагом. Однако Mathcad позволяет сменить его при необходимости на так называемый адаптивный алгоритм (adaptive) или алгоритм решения жестких систем (stiff). Чтобы это сделать, щелкните правой кнопкой мыши на функции odesolve. В открывшемся при этом контекстном меню переместите флажок из строки Fixed (Фиксированный) в строку Adaptive (Адаптивный) или Stiff (Жесткий). Отличие адаптивного метода от простого алгоритма Рунге–Кутта состоит в том, что используемая им длина шага не постоянна, а зависит от скорости изменения функции результата (что определяется, в частности, величиной

первой производной). Такой подход может быть более эффективен в случае не слишком хорошо подобранного пользователем шага, а также для решения сложных дифференциальных уравнений с быстро изменяющимися и осциллирующими функциями решений. Если же вам нужно решить жесткую систему уравнений или систему, содержащую алгебраические ограничения, следует воспользоваться методом Stiff (рис. 14.6).

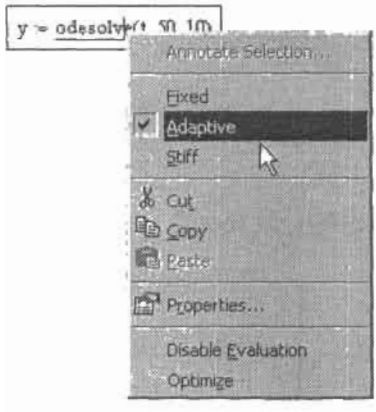


Рис. 14.6. Смена используемого численного метода

14.2.2. Решение ОДУ с помощью встроенных функций

В более старых версиях Mathcad для решения ОДУ использовались специальные встроенные функции. Однако с появлением более наглядного и простого способа, связанного с применением вычислительного блока Given-Odesolve, они уже не представляют особой практической важности. Несмотря на это, о возможности их применения для решения одинарного ОДУ нужно иметь четкое представление, так как в некоторых случаях (например, в программировании) использовать вычислительный блок нельзя.

Так как для решения одного дифференциального уравнения соответствующие встроенные функции применяются редко, то подробно описывать их в этом разделе мы не будем и сразу приведем соответствующий пример (в случае необходимости читатель без труда в нем разберется). Обстоятельно же об указанных функциях мы поговорим в разделе, посвященном решению систем ОДУ.

Пример 14.10. Альтернативный метод решения ОДУ (рис. 14.7)

$$y0 := 1 \qquad M := 100$$

$$D(t, y) := \sin(t) \cdot y$$

$$x1 := 0 \qquad x2 := 10$$

$$y1 := \text{rkfixed}(y0, x1, x2, M, D) \qquad y2 := \text{Bulstoer}(y0, x1, x2, M, D)$$

$$y3 := \text{Rkadapt}(y0, x1, x2, M, D)$$

y1=

	0	1
0	0	1
1	0.1	1.005
2	0.2	1.02
3	0.3	1.046

y2=

	0	1
0	0	1
1	0.1	1.005
2	0.2	1.02
3	0.3	1.046

y3=

	0	1
0	0	1
1	0.1	1.005
2	0.2	1.02
3	0.3	1.046

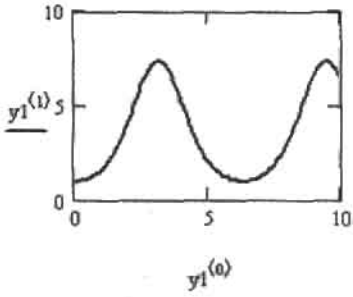


Рис. 14.7. График решения ОДУ, полученного с помощью встроенной функции rkfixed

Применять приведенные встроенные функции можно лишь для решения дифференциальных уравнений первого порядка, как линейных, так и нелинейных. Найти же с их помощью решение уравнения более высокого порядка, в отличие от использования блока Given-Odesolve, непосредственным образом невозможно. Впрочем, это можно сделать, сведя указанное уравнение к системе дифференциальных уравнений. Приведем пример решения ОДУ второго порядка с помощью вычислительного блока и встроенной функции rkfixed (реализующей метод Рунге–Кутта четвертого порядка с фиксированным шагом).

Пример 14.11. Альтернативные способы решения ОДУ второго порядка (рис. 14.8)

Вычислительный блок:

Given

$$y''(t) + y'(t) = 2y(t)$$

$$y(0) = 1 \qquad y'(0) = 3$$

$$z := \text{odesolve}(t, 5, 100)$$

Сведение к системе ОДУ:

$$y := \begin{pmatrix} 1 \\ 3 \end{pmatrix} \qquad D(t, y) := \begin{pmatrix} y_1 \\ -y_1 + 2y_0 \end{pmatrix}$$

$$r := \text{rkfixed}(y, 0, 5, 100, D)$$

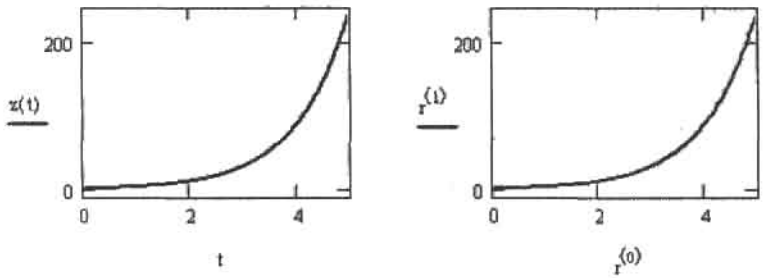


Рис. 14.8. Альтернативные решения ОДУ, полученные с помощью вычислительного блока и встроенной функции

Изучив пример 14.11, вы, наверное, согласитесь, что решать дифференциальные уравнения, особенно высокого порядка, гораздо проще и лучше с помощью вычислительного блока, чем встроенных функций. Действительно, зачем выполнять лишнюю, иногда весьма сложную и объемную, работу по разложению уравнения в систему ОДУ, если это отлично может сделать и сама программа. К тому же решение при использовании вычислительного блока получается гораздо более наглядным и доступным пониманию для не владеющего Mathcad человека, чем при применении системных функций с их запутанным синтаксисом. Поэтому, на мой взгляд, прибегать к встроенным функциям для решения ОДУ стоит лишь в том случае, если использовать блок Given-Odesolve невозможно.

14.2.3. Системы ОДУ

Начиная с 11 версии Mathcad возможности вычислительного блока Given-Odesolve значительно расширились. Теперь с его помощью можно находить численные решения не только ОДУ, но и их систем. Как и к ОДУ, к уравнениям системы предъявляются следующие требования: каждое из них должно быть линейным (высшая производная не должна содержать каких-либо сомножителей) и иметь соответствующее количество начальных или граничных условий.

К сожалению, вычислительный блок не может решать системы нелинейных дифференциальных уравнений. Однако в Mathcad имеются специальные встроенные функции, позволяющие находить решения как линейных, так и нелинейных систем ОДУ. Всего таких функций три.

- Rkfixed(y0,t0,t1,M,D). Задействовав эту системную функцию, можно решить задачу Коши с помощью, пожалуй, самого популярного из численных алгоритмов: метода Рунге–Кутта 4-го порядка с фиксированным шагом. Эта функция подходит для качественного и быстрого решения подавляющего большинства систем ОДУ.
- Rkadapt(y0,t0,t1,M,D). Эта функция реализует адаптивный алгоритм Рунге–Кутта. Ее принципиальным отличием от rkfixed является то, что при вычислении соответствующих приближений она использует не постоянный, а зависящий от скорости изменения функций решения шаг дискретизации переменной. Однако, несмотря на это, в качестве ответа Rkadapt возвращает значения функций в равномерно распределенных, исходя из заданной пользователем величины промежутка, точках интервала. Использовать же рассматриваемую функцию следует в случае более жестких и слож-

ных систем уравнений — это позволит повысить точность расчета или сэкономить время по сравнению с применением простого метода Рунге–Кутта.

- Bulstoer(y0,t0,t1,M,D). Метод Булирша–Штера. Использовать этот алгоритм стоит в том случае, если вы уверены, что функции решения вашей системы достаточно гладкие и плавно изменяющиеся. При выполнении этого условия функция Bulstoer позволяет получать более точные решения, чем rkfixed, затрачивая на это меньше времени.

Для корректного использования описанных выше функций система дифференциальных уравнений должна быть записана в векторном виде:

$$Y'(x) = D(Y(x), x)$$

где $Y'(x)$ — вектор первых производных системы, $D(Y(x),x)$ — вектор-функция, каждая строка которой содержит левую часть соответствующего уравнения системы.

Кроме того, в векторной форме должны быть определены начальные условия:

$$Y(x_0) = y0$$

Разобравшись с принципом приведения системы ОДУ к векторной форме, вы без труда сможете задать и параметры для рассматриваемых встроенных функций:

- y0 — вектор начальных условий; в нем вы должны определить числовые значения искомых функций на левой границе интервала изменения переменной;
- t0 — начальная точка для переменной;
- t1 — конечная точка расчета;
- M — количество шагов, при котором численный метод будет решать систему;
- D — вектор-функция, содержащая левые части уравнений системы. Должна быть задана как функция двух переменных: скаляра t и вектора y (то есть все искомые функции системы должны быть представлены как элементы одного вектора).

Результатом работы приведенных функций является матрица, в первом столбце которой содержатся узловые величины переменной t, а в остальных — значения неизвестных функций системы, рассчитанные в этих точках. При этом порядок расположения столбцов с найденными величинами искомых функций определяется последовательностью, в которой они были занесены в вектор y.

Аппарат дифференциальных уравнений широко используется для описания не только физических, химических или биологических процессов, но и различных явлений в медицине, экономике, демографии и множестве других современных наук. Зачастую при попытке решения той или иной системы уравнений можно получить совершенно неожиданный результат. В этом разделе мы рассмотрим наиболее интересные примеры решения систем ОДУ: экологическую модель «хищник — жертва», некоторые виды динамических систем, а также движение ракеты в поле тяготения небесных тел.

Пример 14.12. Модель «хищник — жертва» (Лотки–Вольтерра)

Модель «хищник — жертва» была предложена независимо друг от друга американским физиком Альфредом Лоткой в 1925 году и итальянским математиком Вито Вольтерра в 1926 году. Она описывает эволюцию численности взаимодействующих популяций хищников и жертв на протяжении определенного промежутка времени. Рассмотрим количественные изменения, происходящие

в популяциях рысей и зайцев. Зайцы, питаясь растительностью, размножаются с постоянной скоростью A, в результате чего их численность возрастает:

$$\frac{d}{dt}x(t) = Ax$$

Рыси поедают зайцев, что уменьшает их численность, причем скорость этого процесса By пропорциональна количеству хищников y:

$$\frac{d}{dt}x(t) = -Bxy$$

Общая динамика популяции зайцев описывается уравнением:

$$\frac{d}{dt}x(t) = Ax - Bxy$$

В результате количество зайцев становится настолько большим, что приводит к резкому увеличению количества рысей. Популяция хищника растет пропорционально имеющейся добыче:

$$\frac{d}{dt}y(t) = Bxy$$

Умирают рыси естественной смертью или под влиянием внешних факторов:

$$\frac{d}{dt}y(t) = -Cy$$

В итоге количество хищников определяется уравнением

$$\frac{d}{dt}y(t) = -Cy + Bxy$$

На определенном этапе рысей становится так много, что количество зайцев быстро уменьшается вследствие интенсивного поедания хищниками. Резкое сокращение запасов пищи вызывает снижение численности рысей, что возобновляет рост популяции зайцев. В конечном счете, процесс повторяется заново. Ниже приведено решение и фазовые портреты полученной системы уравнений для двух начальных условий (рис. 14.9).

$$A := 1.5 \qquad B := 1 \qquad C := 3$$

$$G(t, y) := \begin{pmatrix} A \cdot y_0 - B \cdot y_0 \cdot y_1 \\ -C \cdot y_1 + B \cdot y_0 \cdot y_1 \end{pmatrix} \qquad y0 := \begin{pmatrix} 10 \\ 5 \end{pmatrix} \qquad y1 := \begin{pmatrix} 3 \\ 5 \end{pmatrix}$$

$$r1 := \text{rkfixed}(y0, 0, 100, 5000, G) \qquad r2 := \text{rkfixed}(y1, 0, 100, 5000, G)$$

r1 =

	0	1	2
0	0	10	5
1	0.02	9.258	5.71
2	0.04	8.45	6.419
3	0.06	7.606	7.099
4	0.08	6.757	7.718
5	0.1	5.935	8.251

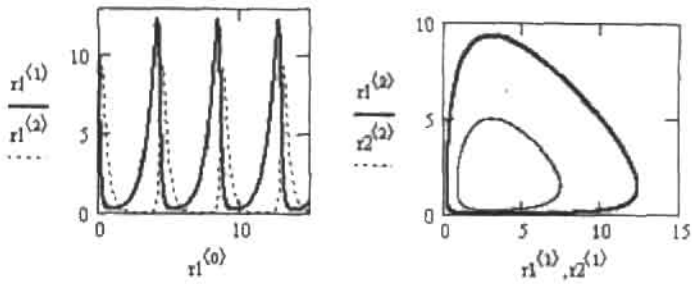


Рис. 14.9. Колебания численности двух взаимодействующих популяций (кривые решений и фазовый портрет)

В критических точках при

$$\frac{d}{dt}x(t)=0$$

скорость размножения зайцев равна скорости их поедания хищниками, при

$$\frac{d}{dt}y(t)=0$$

рождаемость рысей равна их смертности.

Конечно, модель Лотки–Вольтерра является идеализированной. Несмотря на это, она широко применяется не только в экологии, поскольку удивительно точно описывает периодические процессы, происходящие в природе. Приведенный пример демонстрирует биологические колебания. С таким же успехом можно моделировать и химические колебания, например периодические реакции.

Чтобы визуализировать результаты решения системы дифференциальных уравнений (см. рис. 14.9), из полученной с помощью встроенной функции матрицы следует выделить столбцы переменной и интересующей вас функции в виде отдельных векторов. Сделать это можно, используя специальный матричный оператор Matrix Column (Матричный столбец), вызываемый сочетанием клавиш Ctrl+6 непосредственно в соответствующих маркерах графической области. В том случае, если вам нужно узнать значение функции решения в одной из точек, то либо непосредственно найдите нужный рядок в таблице с помощью строк прокрутки, либо используйте стандартную для Mathcad форму вывода с применением матричных индексов (при этом нужно учитывать, что точек, в которых были вычислены величины функций корней, на одну больше, чем было задано шагов).

Пример 14.13. Аттрактор Лоренца

В 1963 году Лоренц, занимаясь исследованием конвективных потоков воздуха, описал трехмерную динамическую систему, которая вызвала большой интерес благодаря непредсказуемости своего состояния. Поведение траекторий такой системы крайне нерегулярно, но со временем они притягиваются к некоему устойчивому множеству, которое получило название хаотического аттрактора Лоренца, или странного аттрактора. Аттрактор (от англ. to attract — притягивать) существует в некоторой области фазового пространства, координаты которого определяются системой трех нелинейных дифференциальных уравнений первого порядка со строгим набором параметров и начальных условий. Система описывает вызванное тепловой конвекцией турбулентное течение жидкости в тороидальном сосуде, расположенном вертикально (рис. 14.10).

$$Pr := 10 \qquad b := \frac{8}{3} \qquad r := 28$$

$$D(t,x) := \begin{bmatrix} Pr \cdot (x_1 - x_0) \\ -x_1 + r \cdot x_0 - x_0 \cdot x_2 \\ x_0 \cdot x_1 - b \cdot x_2 \end{bmatrix}$$

$$D := \text{rkfixed}\left[\begin{pmatrix} 5 \\ 6 \\ 7 \end{pmatrix}, 0, 50, 5000, D\right]$$

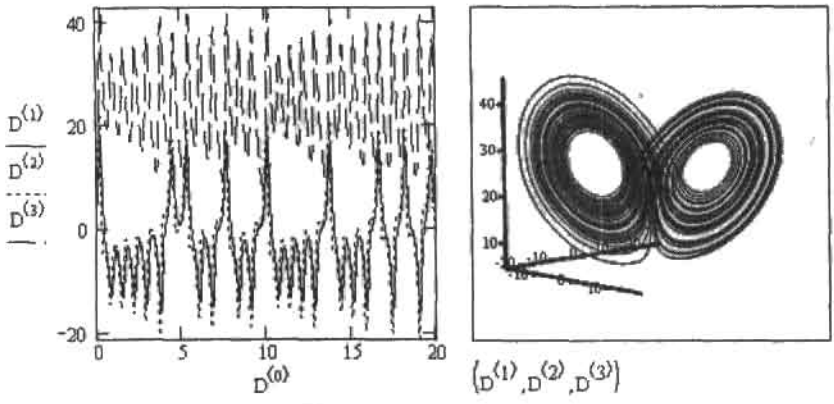


Рис. 14.10. Иллюстрация решения системы Лоренца

Пример 14.14. Модель предсказуемого аттрактора

Аттракторы — предмет изучения теории хаоса, которая предполагает взаимный переход хаоса и порядка в любой динамической системе. Теория помогает моделировать случайные системы, но не позволяет предсказать их поведение в будущем. Помимо хаотических, существуют и предсказуемые аттракторы. Например, траектория квазипериодического движения, слагаемого из двух независимых колебаний, в фазовом пространстве может быть представлена как тор.

$$a := 1.5 \qquad b := 0.3 \qquad B := 1 \qquad p := 1.2$$

$$z(t) := B \cdot \sin(p \cdot t)$$

$$Y(t,y) := \begin{bmatrix} y_1 \\ a \cdot \left[1 - b \cdot (y_0)^2\right] \cdot y_1 - y_0 + z(t) \end{bmatrix}$$

$$Y := \text{rkfixed}\left[\begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}, 0, 500, 5000, Y\right]$$

$$i := 0 .. 5000 \qquad f_i := z(0.1 \cdot i)$$

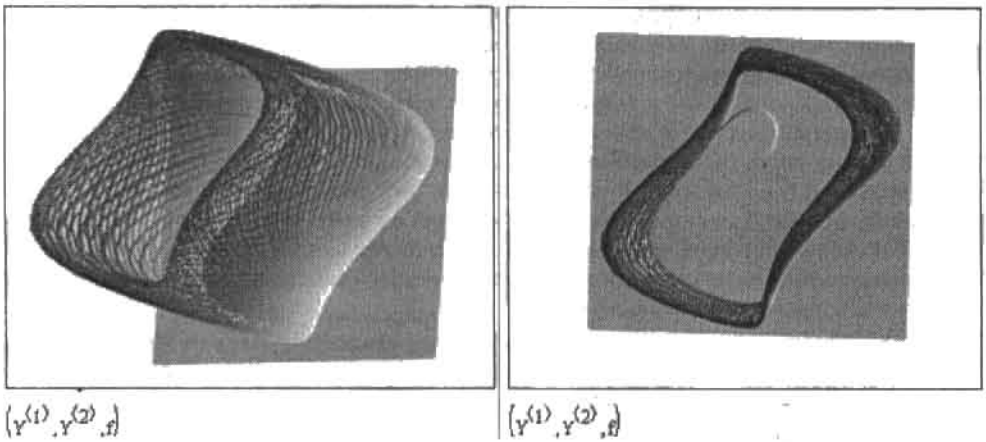


Рис. 14.11. Аттрактор тороидальной формы

Время оборота на внутреннем круге тора является частотой одного колебания, а на внешнем круге – частотой другого колебания. Примером квазипериодического движения могут служить модуляции колебаний в замкнутом колебательном контуре, состоящем из катушки индуктивности, конденсатора, сопротивления и источника тока.

Принципиальным условием успешного решения системы дифференциальных уравнений является правильный выбор величины шага (особенно это важно для функции rkfixed). В общем случае для его определения можно использовать то же правило, что и при решении одного уравнения с помощью вычислительного блока (см. подразд. 14.2.1).

Кстати, точность работы функции rkadapt зависит, помимо определенного пользователем количества шагов M, и от значения системной переменной TOL. Более того, меняя ее величину, гораздо проще можно прийти к корректному решению, чем при подборе количества разбиений интервала. В следующем примере мы попробуем оценить степень влияния величины TOL на точность полученного результата.

Пример 14.15. Влияние величины TOL на точность адаптивного алгоритма

$$y0 := \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad D(x, y) := \begin{bmatrix} x^2 + (y_0)^2 \\ x^2 - (y_1)^2 \end{bmatrix}$$

$z1 := \text{Rkadapt}(y0, 1, 2, 2, D)$ $\qquad z2 := \text{Rkadapt}(y0, 1, 2, 10000, D)$

$z1_{2,1} = 6.7029358244372448$ $\qquad z2_{10000,1} = 6.703786022251724$

$$TOL := 10^{-16}$$

$z1 := \text{Rkadapt}(y0, 1, 2, 2, D)$ $\qquad z2 := \text{Rkadapt}(y0, 1, 2, 10000, D)$

$z1_{2,1} = 6.7037860222954926$ $\qquad z2_{10000,1} = 6.703786022251724$

Как можно заключить из приведенного примера, TOL определяет точность результата в конечной точке гораздо значительнее, чем количество шагов. Однако и последний параметр оказывает определенное влияние на качество расчета. Поэтому, решая численно систему ОДУ, подберите эти параметры так, чтобы их изменение в 10 раз не приводило к изменению значений искомых функций в крайней точке (конечно, в пределах интересующих вас десятичных знаков). Стремиться же сделать сразу количество шагов по максимуму большим не стоит: при этом может резко возрасти как погрешность аппроксимации из-за влияния ошибки округления и других факторов, так и время расчетов. Так, использованное в примере 14.15 количество шагов, равное 10 000, является неоправданно завышенным, и того же уровня точности можно было достигнуть и при гораздо меньшем количестве разбиений.

В столь же значительной степени, как и функция Rkadapt, зависит от TOL и функция Bulstoer. Метод же Рунге–Кутта в стандартной интерпретации, реализуемый функцией rkfixed, в противоположность им находит решение только исходя из длины шага.

Для функций, реализующих адаптивный метод и алгоритм Булирша–Штера, в Mathcad существует и другая форма, позволяющая пользователю более значительно влиять на ход поиска решений. Отличается же эта форма от стандартной только набором параметров и тем, что имя соответствующих функций начинается с маленькой литеры (в связи с этим обстоятельством к заданию функций для решения систем ОДУ следует подходить очень внимательно):

- rkadapt(y0,x0,x1,acc,D,k,s) — метод Рунге–Кутта с переменным шагом;
- bulstoer(y0,x0,x1,acc,D,k,s) — метод Булирша–Штера.

Набор параметров для приведенных функций идентичен и содержит целых семь пунктов:

- y0 — вектор начальных условий. Задается точно так же, как для рассматриваемых функций в стандартной интерпретации;
- x0 — начальная точка интервала изменения переменной;
- x1 — конечная точка для значений переменной;
- acc — погрешность вычисления. Чем меньшим вы определите этот параметр, тем более точным будет результат, однако тем дольше будет вестись расчет. По своим функциям acc полностью аналогичен встроенной переменной TOL в случае использования функций Rkadapt и Bulstoer;
- D — вектор-функция, определяющая систему ОДУ;
- k — максимальное количество шагов, на которые алгоритм может разбить интервал изменения переменной. Необходимость введения этого параметра связана с тем, что в случае некоторых систем при малых значениях acc количество оборотов цикла алгоритма может быть определено программой столь большим, что времени, которое нужно будет потратить на расчет, потребуется недопустимо много. Кроме того, работать в Mathcad с очень большими матрицами невозможно;
- s — минимальная величина шага. Этот параметр необходимо определить в связи с тем, что, в частности, при слишком малой величине шага погрешность разностной аппроксимации может быть весьма значительна.

Использование приведенных функций имеет ряд особенностей.

1. Вы заранее не можете определить, в скольких точках Mathcad вычислит решение вашей системы. Вы можете лишь задать верхнюю границу их количества. А это может быть не всегда удобно — например, при не слишком высоком уровне точности acc решение может быть получено по столь малому количеству разбиений, что по-

строить на основании его гладкий график можно будет лишь при условии использования специальных функций интерполяции.

2. Очень важным является правильное согласование параметра максимального количества шагов и минимальной их длины. Так, если последний параметр вы определите слишком маленьким, то встроенная функция может не досчитать решения до правой границы заданного интервала в связи с тем, что у нее просто кончится лимит шагов.
3. Поскольку длина шага в случае использования описываемых функций может различаться для разных фрагментов функций решения системы на порядки, полученный результат далеко не всегда можно использовать для построения кривых или фазового портрета. Для выполнения этой работы гораздо лучше использовать функции Rkadapt и Bulstoer.
4. Чаще всего рассматриваемые функции используются для получения значения функций решения в конкретной точке (обычно — в конце промежутка). Однако так как заведомо размерность матрицы решений неизвестна, то для определения количества ее строк можно использовать функцию length(V) (или last(V)), выделив предварительно с помощью соответствующего матричного оператора один из ее столбцов.

Приведем пример использования функций rkadapt и bulstoer. Чтобы существовала возможность оценки точности этих функций в зависимости от различных параметров, расчет будем производить для ОДУ второго порядка, решение которого элементарно находится аналитически. Кроме того, в нем мы попытаемся продемонстрировать все изложенные выше трудности при применении рассматриваемых функций.

Пример 14.16. Функции rkadapt и bulstoer

Решаемое уравнение второго порядка с начальными условиями:

$$y''(x) + 10y'(x)\cdot x + 10y(x) = 0$$

$$y'(0) = 0 \qquad y(0) = 1$$

Его точное аналитическое решение:

$$y(x) := e^{-5x^2}$$

Преобразование ОДУ второго порядка в систему ОДУ в векторной форме (переносим в правую часть уравнения все слагаемые, кроме содержащего вторую производную, и вводим замены $y_0=y(x)$, $y'_0=y_1$):

$$D(x,y) := \begin{bmatrix} y_1 \\ -10\cdot(x\cdot y_1 + y_0) \end{bmatrix} \qquad y0 := \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Численное решение полученной системы:

$$r1 := \text{rkadapt}(y0, 0, 1, 10^{-4}, D, 100, 0.0001) \qquad b1 := \text{bulstoer}(y0, 0, 1, 10^{-4}, D, 100, 0.0001)$$

Оценка количества шагов, потребовавшихся для поиска решения с заданной точностью:

$$\text{last}(r1^{\langle 1 \rangle}) = 8 \qquad \text{last}(b1^{\langle 1 \rangle}) = 4$$

Матрицы решений:

$$r1 = \begin{pmatrix} 0 & 1 & 0 \\ 0.01 & 1 & -0.1 \\ 0.06 & 0.982 & -0.589 \\ 0.31 & 0.618 & -1.917 \\ 0.503 & 0.282 & -1.418 \\ 0.703 & 0.084 & -0.593 \\ 0.829 & 0.032 & -0.267 \\ 0.929 & 0.013 & -0.124 \\ 1 & 6.711\times 10^{-3} & -0.067 \end{pmatrix} \qquad b1 = \begin{pmatrix} 0 & 1 & 0 \\ 0.01 & 1 & -0.1 \\ 0.11 & 0.941 & -1.035 \\ 0.615 & 0.151 & -0.927 \\ 1 & 6.696\times 10^{-3} & -0.067 \end{pmatrix}$$

Оценка погрешности исходя из максимального отклонения от аналитического решения:

$$\max\left(\left|r1^{\langle 1\rangle} - y\left(r1^{\langle 0\rangle}\right)\right|\right) = 1.855\times 10^{-4} \qquad \max\left(\left|b1^{\langle 1\rangle} - y\left(b1^{\langle 0\rangle}\right)\right|\right) = 4.365\times 10^{-5}$$

Оценка влияния на точность решения уменьшения величины acc на три порядка:

$$r2 := \text{rkadapt}\left(y0, 0, 1, 10^{-7}, D, 100, 0.0001\right) \qquad b2 := \text{bulstoer}\left(y0, 0, 1, 10^{-7}, D, 100, 0.0001\right)$$

$$\text{last}\left(r2^{\langle 1\rangle}\right) = 26 \qquad \text{last}\left(b2^{\langle 1\rangle}\right) = 5$$

$$\max\left(\left|r2^{\langle 1\rangle} - y\left(r2^{\langle 0\rangle}\right)\right|\right) = 3.433\times 10^{-7} \qquad \max\left(\left|b2^{\langle 1\rangle} - y\left(b2^{\langle 0\rangle}\right)\right|\right) = 8.803\times 10^{-9}$$

Оценка значимости правильного согласования параметров:

$$r3 := \text{rkadapt}\left(y0, 0, 1, 10^{-15}, D, 100, 0\right) \qquad b3 := \text{bulstoer}\left(y0, 0, 1, 10^{-15}, D, 100, 0\right)$$

$$\text{last}\left(r3^{\langle 1\rangle}\right) = 99 \qquad r3_{98,0} = 0.178 \qquad \text{last}\left(b3^{\langle 1\rangle}\right) = 11 \qquad b3_{10,0} = 0.957$$

$$\max\left(\left|r3^{\langle 1\rangle} - y\left(r3^{\langle 0\rangle}\right)\right|\right) = 2.109\times 10^{-14} \qquad \max\left(\left|b3^{\langle 1\rangle} - y\left(b3^{\langle 0\rangle}\right)\right|\right) = 3.015\times 10^{-14}$$

Решения системы ОДУ, полученные с различной точностью с помощью функций rkadapt и bulstoer, представлены на рис. 14.12.

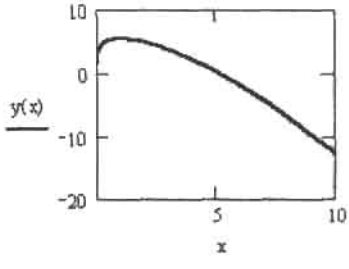


Рис. 14.12. Решения системы ОДУ, полученные с различной точностью с помощью функций rkadapt и bulstoer

Изучая приведенный пример, обратите внимание на следующее.

- Точность метода Булирша–Штера в случае нашей системы оказалась выше, чем у адаптивного алгоритма Рунге–Кутта при том, что и шагов он использовал меньше. Этот факт можно объяснить тем, что искомые решения являются гладкими и не осциллирующими функциями. При решении же более сложных систем эффективность адаптивного алгоритма может оказаться выше, чем метода Булирша–Штера.
- Чтобы найти решение при всех рассмотренных уровнях точности, понадобилось неожиданно мало шагов (особенно это касается функции bulstoer). Учитывая это, не спешите делать количество разбиений при решении ОДУ с помощью стандартных функций большим — для получения результата обычной точности (0.001) для этого бывает вполне достаточно всего несколько десятков шагов.
- Если вы хотите построить график решений системы ОДУ, сделайте величину acc поменьше (10^{-10}–10^{-15}). В этом случае система сгенерирует достаточно точек для задания гладкой кривой. При обычной же точности по полученным данным вы сможете построить лишь весьма грубую ломаную.
- Очень важно правильно согласовать параметры точности, максимального количества шагов и минимальной их длины. Так, в приведенном примере при acc=10^{-15} и отсутствии ограничений на размер шага адаптивный алгоритм смог просчитать функции решения только на 1/10 длины интервала (см. правый график рисунка из примера 14.16). Для того же, чтобы получить с помощью него корректное решение, лимит количества шагов должен быть увеличен до 2000. Зато при том же наборе параметров функция bulstoer нашла результат всего при 11 разбиениях — более чем в 100 раз меньшем их количестве, чем понадобилось rkadapt. Этот факт как нельзя лучше демонстрирует, как сильно определяет успех расчетов правильный выбор алгоритма.

Чтобы проиллюстрировать, как различается шаг на промежутке при адаптивном его определении, построим, исходя из полученных в примере 14.16 значений, диаграмму (рис. 14.13).

$$i := 0..\,last\left(r2^{\langle 1\rangle}\right) \qquad j := 0..\,last\left(b2^{\langle 1\rangle}\right)$$

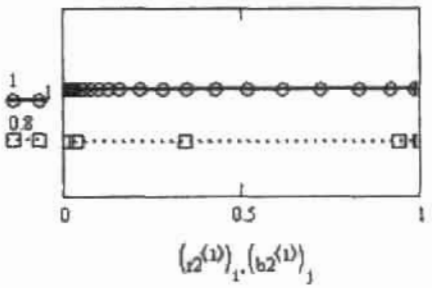


Рис. 14.13. Переменный шаг, использованный функциями rkadapt и bulstoer

Из приведенной диаграммы следует, что шаг, используемый численными методами, зависит как от скорости изменения функций системы, так и от их величин.

Чтобы продемонстрировать все возможности Mathcad в решении систем ОДУ, рассмотрим на примере нахождения траектории ракеты, насколько эффективно справляется с поставленной задачей вычислительный блок Given-Odesolve.

Пример 14.17. Движение ракеты в поле тяготения небесных тел

Согласно открытому Ньютоном закону всемирного тяготения, все тела притягиваются друг к другу с силой, прямо пропорциональной произведению их масс и обратно пропорциональной квадрату расстояния между ними (в данном случае объекты рассматриваются как материальные точки, то есть размерами тел по сравнению с расстоянием между ними можно пренебречь):

$$F = G \cdot \frac{m \cdot M}{R^2}$$

В нашем случае m — масса ракеты, M — масса планеты, R — расстояние между ними, G — гравитационная постоянная (чтобы не оперировать большими величинами масс и расстояний, *не будем* учитывать ее в примере, поскольку на характер зависимости это никак не повлияет). С другой стороны, второй закон Ньютона гласит, что сила есть не что иное, как произведение массы тела на его ускорение:

$$F = m \cdot a$$

Приравняв выражения и отбросив G, получим:

$$a = \frac{M}{R^2}$$

Используя теорему Пифагора, несложно показать, что расстояние R между двумя точками с координатами (x, y, z) и (x1, y1, z1) определяется следующим выражением:

$$R = \sqrt{(x1 - x)^2 + (y1 - y)^2 + (z1 - z)^2}$$

Для упрощения расчетов найдем проекции вектора ускорения *a* на координатные оси x, y, z. Чтобы наглядно представить все рассуждения, начертим вспомогательную схему (рис. 14.14).

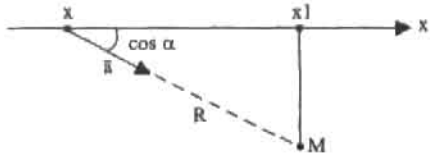


Рис. 14.14. Схема нахождения проекции вектора ускорения a на ось x

Проекция расстояния R между ракетой и планетой M является катетом прямоугольного треугольника *Mx1x* (см. рис. 14.14), а само расстояние — гипотенузой. Поскольку косинус угла есть отношение прилежащего катета к гипотенузе, то проекцию вектора *a* мы можем найти как

$$a_x = a \cdot \cos(\alpha)$$

Подставим в полученную формулу выражение для *a* и воспользуемся определением косинуса, в результате получим

$$a_x = \frac{M}{R^2} \cdot \frac{(x1 - x)}{R} = \frac{M \cdot (x1 - x)}{\left[\sqrt{(x1 - x)^2 + (y1 - y)^2 + (z1 - z)^2}\right]^3}$$

Абсолютно аналогичные выражения можно записать и для проекций вектора *a* на оси y и z.

Полученная нами зависимость описывает траекторию полета ракеты в поле тяготения одной планеты. В случае же нескольких планет необходимо учитывать принцип суперпозиции грави-

тационных полей. Согласно этому принципу гравитационное поле, создаваемое массой, действует *вне зависимости от наличия других масс*. Другими словами, гравитационные поля, возбуждаемые несколькими небесными телами, накладываются друг на друга, оставаясь при этом неизменными, и управляют движением как искусственных, так и естественных тел в космическом пространстве. В нашем примере поле тяготения каждой планеты будет вносить вклад в ускорение ракеты. Сумма же этих вкладов и будет результирующим ускорением. В дифференциальной форме для составляющей вектора ускорения по оси x сказанное выше запишется как:

$$\frac{d^2x}{dt^2} = \frac{M_1\cdot(x1-x)}{R_1^{\,3}} + \frac{M_2\cdot(x2-x)}{R_2^{\,3}} + \ldots + \frac{M_n\cdot(xn-x)}{R_n^{\,3}}$$

По такому же принципу составляются уравнения и для проекций вектора *a* по осям y и z.

Теперь, когда даны необходимые теоретические пояснения, приступим к решению полученной системы дифференциальных уравнений.

Траектория полета определяется рядом факторов, важнейшие из которых — начальная скорость ракеты, масса, координаты планет, а также их количество. Варьируя эти параметры, рассмотрим особенности траектории ракеты в наиболее интересных случаях.

Начнем рассмотрение задачи с простейшего случая, когда ракета, имеющая нулевую начальную скорость, движется в направлении одной планеты.

Если начальная скорость ракеты равна нулю, то она будет двигаться в направлении планеты по прямой линии. Аналогичная траектория будет наблюдаться и в том случае, когда начальная скорость ракеты направлена точно к центру планеты.

Изучая траекторию движения ракеты, мы будем увеличивать количество планет до пяти, поэтому для задания масс и координат планет по мере их увеличения воспользуемся таблицами ввода данных (в общем случае количество планет может быть произвольным).

В данном примере приводятся дифференциальные уравнения для универсальной модели, описывающей движение ракеты в поле тяготения n планет. Если же задать в таблице параметры одной планеты, то решение найдено не будет, поскольку для корректной работы модели массы и координаты должны быть представлены в форме векторов, что возможно, когда минимальное количество планет равно двум. Поэтому добавим в систему еще одну планету малой массы, влиянием которой на движение ракеты можно пренебречь.

M:=

	0
0	$1\cdot10^{3}$
1	$1\cdot10^{-8}$

x:=

	0
0	70
1	20

y:=

	0
0	20
1	50

z:=

	0
0	10
1	30

Определяем координаты исходной точки и составляющие скорости ракеты по осям x, y, z.

$$Vx := 0 \qquad Vy := 0 \qquad Vz := 0$$

$$x0 := 0 \qquad y0 := 0 \qquad z0 := 0$$

Вводим ключевое слово Given, начальные условия и общие уравнения движения.

Given

$$x(0) = x0 \qquad x'(0) = Vx$$
$$y(0) = y0 \qquad y'(0) = Vy$$
$$z(0) = z0 \qquad z'(0) = Vz$$

$$\frac{d^2}{dt^2}x(t) = \sum_{i=0}^{last(x)} \frac{M_i \cdot (x_i - x(t))}{\left[\sqrt{(x_i - x(t))^2 + (y_i - y(t))^2 + (z_i - z(t))^2}\right]^3}$$

$$\frac{d^2}{dt^2}y(t) = \sum_{i=0}^{last(y)} \frac{M_i \cdot (y_i - y(t))}{\left[\sqrt{(x_i - x(t))^2 + (y_i - y(t))^2 + (z_i - z(t))^2}\right]^3}$$

$$\frac{d^2}{dt^2}z(t) = \sum_{i=0}^{last(z)} \frac{M_i \cdot (z_i - z(t))}{\left[\sqrt{(x_i - x(t))^2 + (y_i - y(t))^2 + (z_i - z(t))^2}\right]^3}$$

Находим решение системы

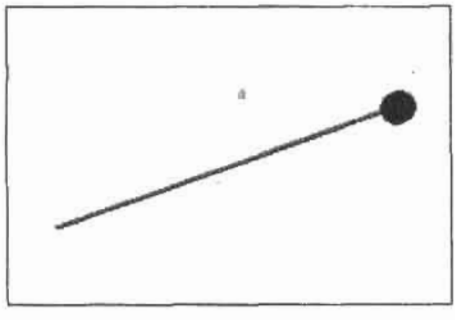
$$R := Odesolve\left[\begin{pmatrix} x \\ y \\ z \end{pmatrix}, t, 150, 4000\right]$$

Чтобы воспользоваться функцией CreateSpace для визуализации решения, зададим входящий в нее параметр — вектор-функцию F.

$$X := R_0 \qquad Y := R_1 \qquad Z := R_2$$

$$F(t) := \begin{pmatrix} X(t) \\ Y(t) \\ Z(t) \end{pmatrix}$$

В том, что система была решена нами верно, легко убедиться, взглянув на рис. 14.15.



(x,y,z) CreateSpace(F,0,50,2000)

Рис. 14.15. Траектория движения ракеты в гравитационном поле одной планеты при нулевой начальной скорости

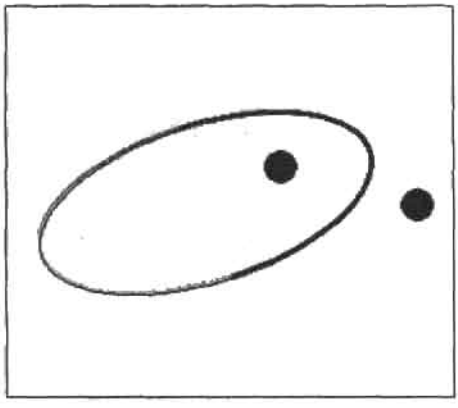
Рассмотрим теперь движение ракеты, когда ее начальная скорость не направлена на планету.

Когда начальная скорость ракеты не направлена на планету, траектория полета искривляется притяжением. Если скорость ракеты не превышает некоторой величины, то по закону Кеплера траектория будет представлять собой эллипс, который лежит в плоскости, проведенной через начальное направление скорости и центр планеты. Центр же притяжения будет находиться в одном из фокусов эллипса. Разумеется, чем больше начальная скорость, тем больше будет ось орбиты.

В случае слишком большой скорости ракета сможет уйти от центра притяжения. Согласно закону Кеплера, она также будет двигаться по одному из конических сечений — параболе или гиперболе.

Решение ищется аналогичным образом (рис. 14.16). Изменим лишь составляющие начальной скорости.

$$Vx := 1 \qquad Vy := 2 \qquad Vz := 3$$



(x,y,z),CreateSpace(F,0,150,2000)

Рис. 14.16. Траектория движения ракеты в поле тяготения одной планеты при ненулевой начальной скорости, направленной не к центру планеты

Перейдем к более сложному случаю, когда ракета, имеющая нулевую начальную скорость, движется в поле гравитационных сил двух планет.

Особенностью такого движения является то, что траектория ракеты всегда будет находиться в одной плоскости с центрами тяжести планет независимо от их массы и расположения.

Изменим входные условия для масс и координат. Остальные выражения останутся неизменными.

M:=

	0
0	$1 \cdot 10^3$
1	$2 \cdot 10^3$

x:=

	0
0	70
1	40

y:=

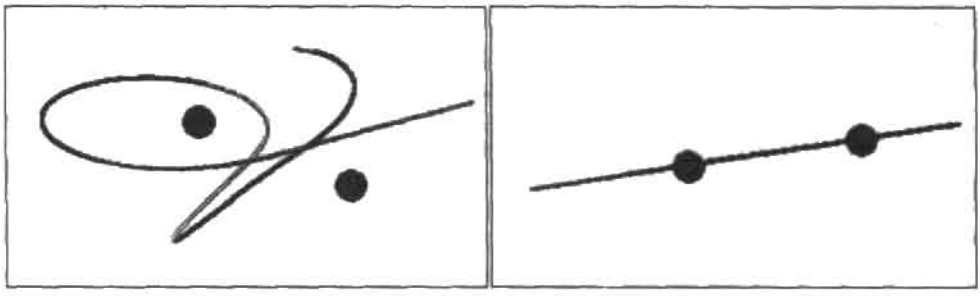
	0
0	20
1	80

z:=

	0
0	10
1	20

Решение системы (рис. 14.17) полностью подтверждает сказанное выше.

И, наконец, рассмотрим последний случай, иллюстрирующий путь ракеты с ненулевой начальной скоростью в поле сил притяжения пяти планет.



(x,y,z) .CreateSpace(F,0,100,2000) (x,y,z) .CreateSpace(F,0,100,2000)

Рис. 14.17. Движение ракеты с нулевой начальной скоростью в поле сил гравитации двух планет

Введем массы и координаты планет, а также начальную скорость ракеты. В маркере функции Odesolve укажем конечную точку интегрирования 2500. Остальные параметры и уравнения оставим без изменения.

M:=

	0
0	$1 \cdot 10^3$
1	$2 \cdot 10^3$
2	$3 \cdot 10^3$
3	$2 \cdot 10^3$
4	$1 \cdot 10^3$

x:=

	0
0	900
1	700
2	500
3	300
4	100

y:=

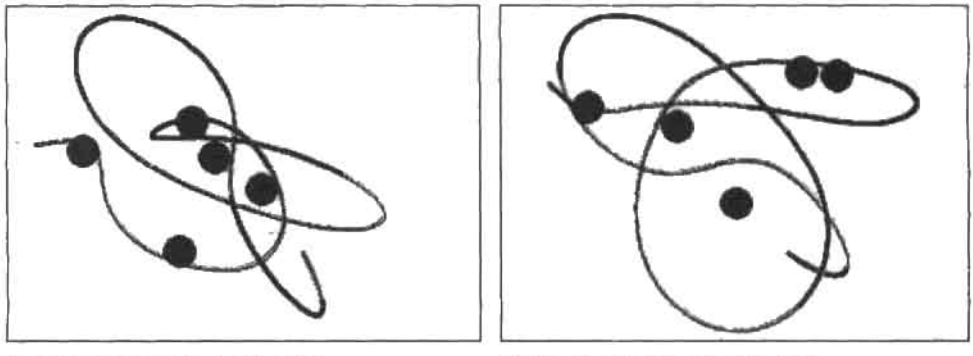
	0
0	200
1	600
2	100
3	400
4	50

z:=

	0
0	100
1	200
2	500
3	300
4	100

Vx:= 1 Vy := 2 Vz:= 3

Траектория ракеты будет представлять собой сложную кривую (рис. 14.18).



(x,y,z) ,CreateSpace(F,0,2300,2000) (x,y,z) ,CreateSpace(F,0,2300,2000)

Рис. 14.18. Путь ракеты в поле сил притяжения пяти планет (начальная скорость не равна нулю)

Заканчивая разговор о решении систем ОДУ в Mathcad, хотелось бы посоветовать: относитесь к результатам работы встроенных функций и вычислительного блока критично и осторожно. Всегда старайтесь искать решение с помощью двух различных алгоритмов и принимайте результат как истинный, только если их ответы совпадут. Параметры встроенных функций настраивайте так, чтобы их изменение в 2–10 раз не приводи-

ло к изменению в ответе. Руководствуясь описанным подходом, вы сможете избежать многих неприятных сюрпризов. В следующем разделе мы подробно разберем численные методы, лежащие в основе работы описанных выше встроенных функций.

14.2.4. Численные методы, применяемые встроенными функциями для решения ОДУ и их систем

Методы Рунге–Кутта

Метод Рунге–Кутта 4-го порядка наиболее популярен при решении задачи Коши, поскольку является достаточно точным и не требует вычисления производных высших порядков, в отличие, например, от метода Тейлора, что значительно сокращает время расчетов. Чтобы понять, каким образом функция rkfixed, реализующая метод, находит численное решение, и насколько оно оказывается верным, попытаемся проанализировать сам алгоритм. Очевидно, самый лучший способ сделать это — самостоятельно написать соответствующую программу.

Рассмотрим уравнение $y'(t)=t^2-y$, для которого имеется аналитическое решение, чтобы сравнить, насколько близки к истинным значения, полученные численным методом. Зададим начальные условия $y(t_0)=y0$ и интервал поиска $[t_0;t_1]$, который разобьем на множество подынтервалов $[t_i;t_{i+1}]$. Их количество определяется величиной выбранного шага h. Именно на этих промежутках мы будем вести поиск интегральной кривой, описывающей функцию решения. Сущность его заключается в нахождении семейства тангенсов угла наклона кривой на данном промежутке, которое затем используется для приближенного расчета приращения функции. Сначала определим направление движения на заданном подынтервале. Для этого необходимо найти наклон интегральной кривой в точке t_i. Очевидно, что он будет равен тангенсу угла наклона касательной к нашей кривой в этой точке:

$$k1 = tg\alpha = f(t_i, y_i)$$

Затем, двигаясь по касательной, сделаем половинный шаг и попадем в точку с абсциссой $t_i+h/2$. Понятно, что приращение искомой функции составит $h/2 \cdot k1$. Теперь определим наклон кривой в новой точке

$$k2 = f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2} \cdot k1\right)$$

и вновь сделаем половинный шаг из исходной точки, правда, в направлении, заданном k2. Абсцисса останется той же, а ордината примет вид $y_i+h/2 \cdot k2$. Повторим уже знакомую нам операцию: найдем тангенс угла наклона касательной к функции в этой точке:

$$k3 = f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2} \cdot k2\right)$$

который используем для определения наклона функции на правой границе интервала разбиения:

$$k4 = f(t_i + h, y_i + h \cdot k3)$$

Итак, мы получили семейство тангенсов угла наклона кривой в трех точках интервала: k1 — в начальной точке, k2 и k3 — в середине и k4 — в конце. Эти значения необходимы для того, чтобы провести численное интегрирование функции, задающей наклон

на промежутке $[t_i;t_{i+1}]$ методом Симпсона, о котором мы говорили ранее. Функция в этом случае интерполируется параболой. Чтобы определить ее вид, достаточно трех точек подынтервала, в которых значение функции известно. Выше мы определили четыре таких точки: k1, k2, k3 и k4, поэтому в середине промежутка наклон кривой зададим как среднее арифметическое k2 и k3. Применив формулу Симпсона с половинными шагом, мы найдем приращение искомой функции на данном подынтервале.

$$y_{i+1} - y_i = \int_{t_i}^{t_{i+1}} f(t,y)\,dt = \frac{h}{6}\left[k1 + 4\cdot\frac{(k2+k3)}{2} + k4\right]$$

Эта идея лежит в основе метода Рунге–Кутта 4-го порядка, который позволяет подобным образом рассчитать приращение функции на всем интервале $[t_0;t_1]$. Исходя из условия $y(t_0)=y0$, мы нашли решение y_i в точке t_i. Аналогично, приняв за начальное условие $y(t_i)=y_i$, можно определить функцию в точке t_{i+1} и т. д. В общем виде

$$y_{i+1} = y_i + \frac{h}{6}\cdot(k1 + 2k2 + 2k3 + k4)$$

Чтобы наглядно представить всю последовательность действий, проведенных нами выше, напишем программу, реализующую данный метод.

Пример 14.18. Программная реализация метода Рунге–Кутта 4-го порядка (рис. 14.19)

$$f(t,y) := t^2 - y$$

$$\begin{aligned}
Dif(f,a,b,y0,M) := \Bigg| & \left(h \leftarrow \frac{b-a}{M} \quad y_0 \leftarrow y0 \quad t \leftarrow a\right) \\
& \text{for } i \in 0..M \\
& \quad k1 \leftarrow f(t, y_i) \\
& \quad k2 \leftarrow f\left(t + \frac{h}{2}, y_i + \frac{h}{2}\cdot k1\right) \\
& \quad k3 \leftarrow f\left(t + \frac{h}{2}, y_i + \frac{h}{2}\cdot k2\right) \\
& \quad k4 \leftarrow f(t + h, y_i + h\cdot k3) \\
& \quad y_{i+1} \leftarrow y_i + h\cdot\frac{(k1 + 2\cdot k2 + 2\cdot k3 + k4)}{6} \quad \text{if } i \le M-1 \\
& \quad (T_i \leftarrow t \quad t \leftarrow t + h) \\
& V \leftarrow \begin{pmatrix} T \\ y \end{pmatrix}
\end{aligned}$$

$$C := Dif(f,0,10,1,100)$$

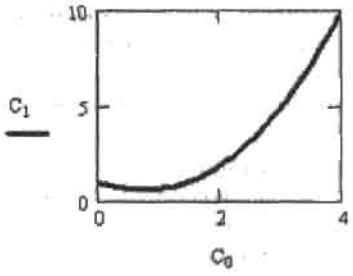


Рис. 14.19. Решение ОДУ с помощью приведенного алгоритма

В корректности работы программы легко убедиться, сравнив решения, полученные с помощью встроенных функций и нашего алгоритма.

На каждом шаге алгоритм аппроксимирует дифференциальное уравнение по приведенной схеме, поэтому попытаемся проанализировать, какие факторы влияют на точность полученных с его помощью результатов. Очевидно, возникает ошибка приближения искомой функции параболой при использовании формулы Симпсона, а также погрешность усреднения значений k2 и k3 в центральной точке подынтервала. Такой подход к оценке погрешности можно назвать геометрическим. С аналитической точки зрения метод Рунге–Кутта 4-го порядка использует для нахождения решения только первые пять слагаемых, полученных при разложении искомой функции в ряд Тейлора в окрестности заданной точки. Если на отдельном шаге погрешность пренебрежимо мала, то на интервале происходит накопление ошибки, что отражается на точности результата. Поэтому важно выбрать оптимальное соотношение между длиной шага и их количеством, при котором погрешность аппроксимации окажется минимальной, а время расчета приемлемым. Как правило, с уменьшением длины шага точность возрастает до определенной величины, которая не меняется при дальнейшем разбиении интервала. Проиллюстрируем сказанное. В табл. 141 приведены истинные решения задачи Коши $y'(t)=t^2-y$, $y(0)=1$ на интервале [0;10], а также рассчитанные нашим алгоритмом при различной длине шага.

Таблица 14.1. Решения задачи Коши $y'(t)=t^2-y$, $y(0)=1$ на интервале [0;10] методом Рунге–Кутта 4-го порядка

t_i	y_i			Истинное решение	Ошибка		
	h=1	h=0.5	h=0.1		h=1	h=0.5	h=0.1
0	1.0000	1.0000	1.0000	1.0000	0.0000	0.0000	0.0000
1	0.6458	0.6329	0.6321	0.6321	0.0137	0.0008	0.0000
2	1.8880	1.8659	1.8647	1.8647	0.0233	0.0012	0.0000
3	4.9788	4.9517	4.9502	4.9502	0.0286	0.0015	0.0000
4	10.0129	9.9833	9.9817	9.9817	0.0312	0.0016	0.0000
5	17.0257	16.9949	16.9933	16.9933	0.0324	0.0016	0.0000
6	26.0305	25.9992	25.9975	25.9975	0.0330	0.0017	0.0000
7	37.0323	37.0007	36.9991	36.9991	0.0332	0.0016	0.0000
8	50.0329	50.0013	49.9997	49.9997	0.0332	0.0016	0.0000
9	65.0332	65.0015	64.9999	64.9999	0.0333	0.0016	0.0000
10	82.0333	82.0016	82.0000	82.0000	0.0333	0.0016	0.0000

Как видно из таблицы, при длине шага, равной 0.1 (используемой системой по умолчанию), решения оказываются точными до четвертого знака. Уменьшение длины шага до 0.01 увеличивает точность до седьмого знака. К сожалению, подобная закономерность реализуется далеко не для всех функций. Выбор желаемого уровня точности остается за пользователем, но зачастую оценить величину ошибки при той или иной длине шага просто невозможно. В этом случае альтернативным оказывается метод Рунге–Кутта с переменным шагом (адаптивный метод), лежащий в основе работы функций rkadapt и Rkadapt. В каждой узловой точке интервала этот алгоритм находит два варианта решения, используя схемы Рунге–Кутта 4-го и 5-го порядков. Если разность между полученными значениями оказывается меньше допустимой величины ошибки (которую можно заранее задать через системную переменную TOL), то длина шага увеличивается. Если же разность превысит ошибку, то следующую точку интервала алгоритм определит с меньшей длиной шага. В случае равенства обоих параметров шаг остается постоянным. Адаптивный метод целесообразно использовать для нахождения неоднородных решений, меняющихся с различной скоростью на том или ином участке интервала. Это позволит сократить время расчета, поскольку для некоторых функций хороший результат может быть получен и при использовании алгоритмом небольшого количества шагов.

Метод Булирша–Штера

Алгоритм Булирша–Штера (Bulirsch–Stoer), реализованный во встроенных функциях bulstoer и Bulstoer, дает хорошие результаты для гладких функций решения. В его основе лежит численное приближение решения на промежутке [x;x+H] методом рациональной экстраполяции. Серия значений, полученных на интервале интегрирования, экстраполируется в конечной точке интервала к истинному решению полиномом, приближающим функцию (рис. 14.20).

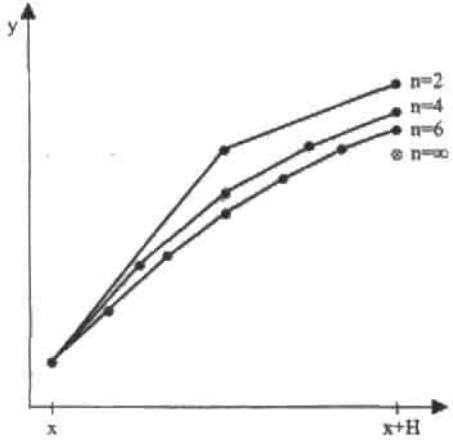


Рис. 14.20. Иллюстрация метода Булирша–Штера

Приближение оказывается тем точнее, чем больше шагов было выбрано внутри интервала. Понятно, что при стремлении их количества к бесконечности оно превращается в точное аналитическое решение, но при этом неограниченно возрастает и время расчета. Сократить его можно, задав требуемую точность решения TOL через параметр acc.

На промежутке последовательно выбираются n=2, 4, 6, 8, 10 и т. д. шагов интегрирования с постоянно уменьшающейся длиной до тех пор, пока разность между полученными на конце интервала значениями не окажется равной TOL. Последний результат заносится в матрицу решений, выбирается следующий промежуток, и все операции повторяются заново. Проанализируем конкретный пример, чтобы оценить, каким образом влияют на точность решения выбранное алгоритмом количество шагов и величина TOL.

Пример 14.19. Влияние количества шагов и величины TOL на точность алгоритма Булирша–Штера

$$Sys(t,y) := \begin{pmatrix} 2 \cdot y_1 \\ -y_0 - 15y_1 \end{pmatrix} \qquad y0 := \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$x := bulstoer(y0, 0, 10, 10^{-3}, Sys, 10, 0.1)$ $\qquad$ $y := bulstoer(y0, 0, 10, 10^{-16}, Sys, 10, 0.1)$

$x_{5,1} = 9.32674444016026$ $\qquad$ $y_{5,1} = 9.99770030614651$

$last(x^{\langle 1 \rangle}) = 9$ $\qquad$ $last(y^{\langle 1 \rangle}) = 9$

$x := bulstoer(y0, 0, 10, 10^{-3}, Sys, 10000, 0.1)$ $\qquad$ $y := bulstoer(y0, 0, 10, 10^{-16}, Sys, 10000, 0.1)$

$x_{5,1} = 9.32674444016026$ $\qquad$ $y_{5,1} = 9.99770030614651$

$last(x^{\langle 1 \rangle}) = 35$ $\qquad$ $last(y^{\langle 1 \rangle}) = 50$

Как видно, точность решения, полученного методом Булирша–Штера, зависит не от количества разбиений всего интервала поиска, а от заданной величины TOL, которая определяет внутреннее количество шагов каждого промежутка [x;x+H].

14.2.5. Жесткие системы ОДУ

Понятие о жесткости (Stiff) системы дифференциальных уравнений появилось в середине XX века в связи с решением систем, описывающих протекание катализируемых химических реакций. Оказалось, что такие, казалось бы, эффективные и абсолютно надежные численные методы, как алгоритм Рунге–Кутта (да и все другие разработанные на том этапе явные алгоритмы) не способны решать системы подобного рода. Это неприятное открытие послужило толчком для развития обширнейшего и важнейшего направления в математике численных методов, и сейчас алгоритмы, способные решать жесткие системы, играют на практике куда большую роль, чем классические явные методы.

Точного определения жесткой системы ОДУ в общем не существует. Обычно жесткой считается система, описывающая сильно разномасштабные процессы. Трудности, связанные с решением таких систем явными численными методами, объясняются тем, что условие устойчивости требует брать шаг слишком мелким. А это означает, что для того, чтобы найти корректный результат, алгоритм придется прокрутить миллионы или даже миллиарды раз, что может быть непосильной задачей даже для современного компьютера.

Рассмотрим следующее простейшее линейное ОДУ:

$$\frac{d}{dt}y(t) = -a \cdot y(t)$$

При небольшом значении коэффициента a вычислительный блок Given-Odesolve (реализующий метод Рунге–Кутта 4-го порядка с фиксированным шагом) без проблем решает это уравнение (рис. 14.21). Например:

$$a := 3$$

Given

$$\frac{d}{dt}y(t) = -a \cdot y(t) \qquad y(0) = 1$$

$$f := \text{Odesolve}(t, 3, 10)$$

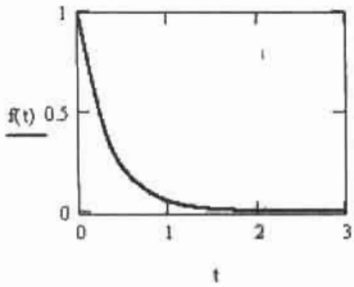


Рис. 14.21. Верное решение ОДУ

При увеличении же значения коэффициента a до 30 в решении появляется неустойчивость (рис. 14.22).

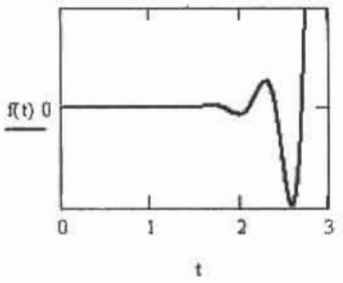


Рис. 14.22. Неустойчивость численного решения ОДУ

Приведенный пример показывает, что одни и те же уравнение или система уравнений могут быть жесткими или нет в зависимости от значения (точнее, соотношения) входящих в них параметров. Как правило, чем более различаются коэффициенты при разных членах уравнений системы ОДУ, тем более сложно решить ее с помощью обычных численных методов. Это связано с тем, что чем жестче система или уравнение, тем больше требуется шагов для устойчивой работы численного алгоритма. Так, например,

чтобы получить верный результат для рассматриваемого уравнения при a=30, следует разбить промежуток не на 10, а на 50 отрезков. Однако в случае многих жестких систем невозможно прийти к верному решению обычным уменьшением длины шага, поскольку такой подход потребует столь огромного количества оборотов цикла, что задача будет не решаема в вычислительном плане.

Разрешена же данная проблема была с помощью так называемых неявных абсолютно устойчивых разностных методов. Особенностью этих методов является то, что они устойчивы при любой величине шага, поэтому последний может определяться только исходя из соображений точности. В Mathcad жесткие системы ОДУ можно решать двумя способами: с помощью вычислительного блока Given-Odesolve, выбрав в контекстном меню функции Odesolve неявный алгоритм Stiff, либо задействовав встроенные функции, реализующие наиболее популярные неявные методы.

- Stiffb(y0,t0,t1,M,F,J). Метод Булирша–Штера для жестких систем ОДУ;
- Stiffr(y0,t0,t1,M,F,J). Метод Розенброка для решения жестких систем ОДУ;
- Radau(y0, t0, t1, M, F). Метод RADAU5 (в терминологии справочной системы Mathcad) для жестких систем ОДУ.

Данные функции требуют задания следующих параметров:

- y0 — вектор начальных условий. Задается точно так же, как при решении простых систем ОДУ;
- t0 — начальная точка для переменной;
- t1 — правая граница для интервала изменения переменной;
- M — количество шагов, которое должен использовать численный метод;
- F — векторная функция, определяющая систему ОДУ. Задается точно по таким же правилам, как и в случае использования функций решения нежестких систем;
- J — матричная функция размерности Nr(N+1), где N — количество уравнений в системе. В первом столбце J должны находиться функции производных соответствующих строк F по независимой переменной. В остальных N столбцах должен быть расположен якобиан (матрица частных производных всех соответствующих уравнениям функций по всем входящим в них переменным) решаемой системы дифференциальных уравнений. Аналогично F, J должна быть задана как функция двух переменных.

Результатом работы перечисленных функций, как и всех остальных встроенных функций, предназначенных для решения систем ОДУ, является матрица, в первом столбце которой содержатся значения переменной, а в остальных — соответствующие им величины искомых функций.

Обратите внимание, что функция Radau не требует задания якобиана, однако при ее использовании необходимо повысить точность вычислений, уменьшив значение встроенной константы TOL. Точность расчета с помощью Stiffr и Stiffb также определяется TOL.

Рассмотрим решение жесткой системы ОДУ на примере задачи химической кинетики. Пусть исходное вещество S под действием фермента E превращается в конечный продукт P. Фермент E, действуя на вещество S, образует с ним промежуточный комплекс ES, после распада которого происходит образование продукта реакции P и регенерация фермента E. Комплекс ES не является абсолютно устойчивым и может разрушаться с образованием исходных реагентов. Каждая из перечисленных стадий реакции

протекает с определенной скоростью, которая характеризуется константой скорости kn. Схематично все процессы можно представить следующим образом:

$$E + S \underset{k2}{\overset{k1}{\rightleftharpoons}} ES \xrightarrow{k3} P + E$$

Под скоростью реакции понимается изменение (уменьшение) концентрации реагирующих веществ в единицу времени. Чем больше kn, тем соответственно больше и скорость реакции. Кроме того, нетрудно догадаться, что реакция будет протекать тем быстрее, чем больше концентрация исходных компонентов. Учитывая эти факты, запишем уравнения, описывающие изменение концентрации веществ в реакционной системе с течением времени.

Концентрация исходного реагента S уменьшается по мере образования промежуточного комплекса ES и восполняется за счет его частичного распада:

$$\frac{d[S]}{dt} = -k1\cdot[E]\cdot[S] + k2\cdot[ES]$$

Концентрация фермента E уменьшается по мере образования промежуточного комплекса ES. Противоположный процесс наблюдается при регенерации фермента E в результате образования продукта P, а также при разрушении комплекса ES:

$$\frac{d[E]}{dt} = -k1\cdot[E]\cdot[S] + (k2 + k3)\cdot[ES]$$

Концентрация промежуточного комплекса ES увеличивается за счет связывания фермента E с веществом S. При образовании конечного продукта P и при разрушении комплекса концентрация ES в реакционной системе падает.

$$\frac{d[ES]}{dt} = k1\cdot[E]\cdot[S] - (k2 + k3)\cdot[ES]$$

Конечный продукт P накапливается в системе за счет распада комплекса ES:

$$\frac{d[P]}{dt} = k3\cdot[ES]$$

Итак, мы получили систему уравнений, решение которой представлено в примере 14.20. Чтобы получить задачу Коши, в систему необходимо добавить начальные условия — исходные концентрации фермента E и вещества S. Разумеется, в момент начала реакции концентрации промежуточного комплекса ES и продукта P будут равны нулю.

Изучая пример, обратите особое внимание на особенности задания параметра J.

Пример 14.20. Решение жесткой системы ОДУ с помощью встроенных функций

$$k1 := 10^6 \qquad k2 := 10^2 \qquad k3 := 10^{-3}$$

$$y0 := \begin{pmatrix} 1 \\ 0.1 \\ 0 \\ 0 \end{pmatrix} \qquad F(t, y) := \begin{bmatrix} -k1\cdot y_1\cdot y_2 + k2y_0 \\ -k1\cdot y_1\cdot y_2 + (k2 + k3)\cdot y_0 \\ k1\cdot y_1\cdot y_2 - (k2 + k3)\cdot y_0 \\ k3\cdot y_0 \end{bmatrix}$$

$$J(t,y) := \begin{bmatrix} \frac{d}{dt}F(t,y)_0 & \frac{d}{dx}F\left[x,\begin{pmatrix} x \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}\right]_0 & \frac{d}{dx}F\left[x,\begin{pmatrix} y_0 \\ x \\ y_2 \\ y_3 \end{pmatrix}\right]_0 & \frac{d}{dx}F\left[x,\begin{pmatrix} y_0 \\ y_1 \\ x \\ y_3 \end{pmatrix}\right]_0 & \frac{d}{dx}F\left[x,\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ x \end{pmatrix}\right]_0 \\ \frac{d}{dt}F(t,y)_1 & \frac{d}{dx}F\left[x,\begin{pmatrix} x \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}\right]_1 & \frac{d}{dx}F\left[x,\begin{pmatrix} y_0 \\ x \\ y_2 \\ y_3 \end{pmatrix}\right]_1 & \frac{d}{dx}F\left[x,\begin{pmatrix} y_0 \\ y_1 \\ x \\ y_3 \end{pmatrix}\right]_1 & \frac{d}{dx}F\left[x,\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ x \end{pmatrix}\right]_1 \\ \frac{d}{dt}F(t,y)_2 & \frac{d}{dx}F\left[x,\begin{pmatrix} x \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}\right]_2 & \frac{d}{dx}F\left[x,\begin{pmatrix} y_0 \\ x \\ y_2 \\ y_3 \end{pmatrix}\right]_2 & \frac{d}{dx}F\left[x,\begin{pmatrix} y_0 \\ y_1 \\ x \\ y_3 \end{pmatrix}\right]_2 & \frac{d}{dx}F\left[x,\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ x \end{pmatrix}\right]_2 \\ \frac{d}{dt}F(t,y)_3 & \frac{d}{dx}F\left[x,\begin{pmatrix} x \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}\right]_3 & \frac{d}{dx}F\left[x,\begin{pmatrix} y_0 \\ x \\ y_2 \\ y_3 \end{pmatrix}\right]_3 & \frac{d}{dx}F\left[x,\begin{pmatrix} y_0 \\ y_1 \\ x \\ y_3 \end{pmatrix}\right]_3 & \frac{d}{dx}F\left[x,\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ x \end{pmatrix}\right]_3 \end{bmatrix}$$

$$J(t,y) \rightarrow \begin{pmatrix} 0 & 100 & -1000000\cdot y_2 & -1000000\cdot y_1 & 0 \\ 0 & \frac{100001}{1000} & -1000000\cdot y_2 & -1000000\cdot y_1 & 0 \\ 0 & \frac{-100001}{1000} & 1000000\cdot y_2 & 1000000\cdot y_1 & 0 \\ 0 & \frac{1}{1000} & 0 & 0 & 0 \end{pmatrix}$$

$$J(t,y) := \begin{pmatrix} 0 & 100 & -1000000\cdot y_2 & -1000000\cdot y_1 & 0 \\ 0 & \frac{100001}{1000} & -1000000\cdot y_2 & -1000000\cdot y_1 & 0 \\ 0 & \frac{-100001}{1000} & 1000000\cdot y_2 & 1000000\cdot y_1 & 0 \\ 0 & \frac{1}{1000} & 0 & 0 & 0 \end{pmatrix}$$

s1 := Stiffb(y0, 0, 1000, 5000, F, J) s2 := Stiffr(y0, 0, 1000, 5000, F, J)

Полученное нами решение абсолютно адекватно физически (рис. 14.23): с течением времени концентрация исходного вещества S падает (s1<1>) по мере накопления в реакционной системе конечного продукта P (s1<4>). На начальном этапе, когда количество E и S в системе велико, происходит их интенсивное взаимодействие, вследствие чего концентрация комплекса ES резко возрастает. Затем за счет образования продукта P и частичного разрушения комплекса концентрация ES постепенно снижается (s1<3>). Процесс превращения S в P протекает с участием фермента E с последующей его регенерацией, поэтому концентрация E в ходе реакции практически не меняется (s1<2>).

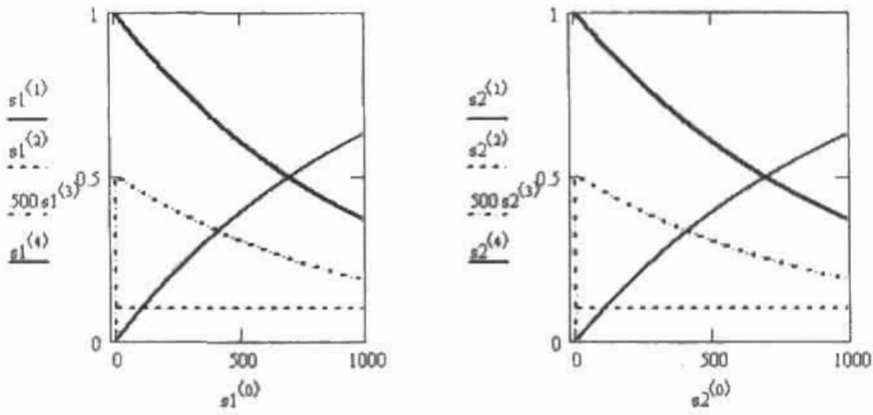


Рис. 14.23. Решение жесткой системы дифференциальных уравнений

Обратите внимание, каким интересным и необычным способом был подсчитан нами якобиан. Мы использовали ту особенность решаемой системы, что в ней нет никаких специальных функций, и все искомые элементы входят в нее в первой степени. А это означает, что при вычислении производной по определенной неизвестной, последняя будет исключена из уравнения. Поэтому, не сделав никакой фактической ошибки, мы просто заменили соответствующие элементы вектора y на переменную x. Необходимость этой операции связана с тем, что оператор дифференцирования Mathcad не может вычислять производной по матричному элементу.

Безусловно, вы можете сказать, что приведенный в рассматриваемом примере способ задания якобиана чрезмерно громоздок и гораздо легче было бы просто подсчитать его элементы в голове. Конечно, в случае таких простых для вычисления производных функций, какими являются правые части уравнений химической кинетики, следует поступить именно так. Однако для более сложных систем представленный способ аналитического вычисления якобиана является наиболее простым в реализации среди всех возможных в Mathcad. Поэтому его стоит запомнить.

Продолжая разговор о якобиане, поясним, для чего он нужен встроенным функциям группы Stiff. Все дело в том, что в теории численных методов доказано, что чем более вырожденным является якобиан системы ОДУ, тем она более жесткая. То есть матрица Якоби является относительной мерой жесткости, и это ее свойство может быть использовано для наиболее оптимальной настройки численного метода. Когда определитель якобиана, вне зависимости от значений функций и переменной, равен нулю, система имеет предельно высокую степень жесткости.

Интересно, что в решении этой же системы, полученном с помощью вычислительного блока Given-Odesolve (методом Stiff) наблюдается осцилляция (рис. 14.24), указывающая на неустойчивость алгоритма, а при попытке решить систему с помощью функции Radau решение вовсе расходится.

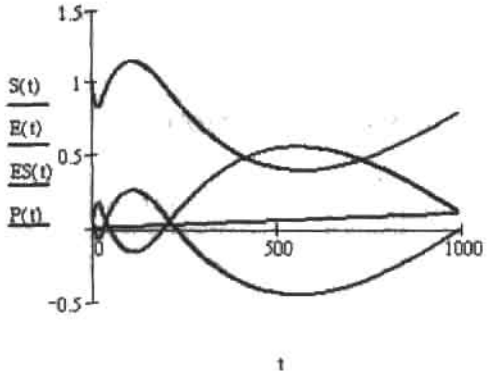


Рис. 14.24. Осцилляция решения жесткой системы ОДУ, полученного с помощью вычислительного блока Given-Odesolve

Возникшие проблемы связаны в первую очередь с тем, что оба метода не требуют задания якобиана в символьном виде, что, безусловно, снижает эффективность их работы. Используя блок Given-Odesolve и функцию Radau, можно быстро получить решение менее жестких систем, не тратя время на составление громоздкого якобиана. Однако найти решение особо жестких систем с их помощью не удастся, в таких случаях нужно пользоваться функциями семейства Stiff.

Следует помнить, что для корректной работы функций Stiffb и Stiffr также необходим тщательный подбор входных условий. Так, снижение начальной концентрации фермента E в рассмотренном примере с 0,1 до 0,01 увеличивает жесткость системы, и ее решение оказывается непосильным даже для Stiffb и Stiffr при любом количестве шагов.

Аналогично функциям для решения простых систем ОДУ, для встроенных функций Stiffr и Stiffb имеются модификации, позволяющие определять решение в одной (конечной) точке (например, в задачах химической кинетики интерес представляют концентрации реагентов именно в определенный момент времени):

- stiffr(y0,t0,t1,acc,F,J,k,s) — метод Розенброка;
- stiffb(y0,t0,t1,acc,F,J,k,s) — метод Булирша–Штера;
- radau(y0, t0, t1, acc, F, k, s) — метод RADAU5.

Данные функции имеют весьма значительное количество параметров — восемь, что, однако, позволяет пользователю активно влиять на ход поиска решения. Особенности определения параметров в случае применения функций stiffb и stiffr точно такие же, как при использовании функций bulstoer и rkadapt, поэтому подробно останавливаться на них мы не будем. Единственное же отличие состоит в том, что вам необходимо задать, в связи со спецификой алгоритмов для решения жестких ОДУ, функцию якобиана J, что делается точно так же, как для функций семейства stiff в стандартной интерпретации.

14.3. Краевые задачи

Краевые задачи отличаются от задач Коши тем, что начальные условия для них задаются на обеих границах интервала поиска решений. В данном разделе мы рассмотрим способы решения краевых задач для ОДУ и их систем с помощью встроенных алгоритмов, а также подробно разберем принципы построения разностных схем для краевых задач средствами программирования.

14.3.1. Решение краевых задач для ОДУ с помощью встроенных средств

Краевая форма для ОДУ и систем ОДУ используется в основном в физике и технике в тех случаях, когда определить все начальные значения на левой границе промежутка невозможно. При этом найти решение для заданных таким образом дифференциальных уравнений значительно сложнее, чем для задач Коши.

В Mathcad имеется возможность решения ОДУ и систем ОДУ, заданных в форме краевых задач, с помощью специального численного метода, называемого методом пристрелки или методом стрельбы (shooting method). При этом, аналогично задачам Коши, можно использовать либо вычислительный блок, либо специальные встроенные функции.

Так как эффективное использование встроенных средств для решения краевых задач требует определенного представления о лежащем в их основе методе стрельбы (shooting method), то нелишне будет разобраться в его принципах.

Прежде всего нужно отметить, что метод стрельбы не предназначен для непосредственного решения дифференциальных уравнений. С его помощью можно лишь преобразовать краевую задачу в форму Коши, определив недостающие на левой границе начальные условия. Сделать же это можно только подбором (естественно, направленным) параметров исходя из того условия, что соответствующие решения полученной задачи Коши в правой крайней точке интервала должны совпадать с исходными краевыми условиями с определенным уровнем точности.

Представим, что нам нужно найти закон, по которому совершает колебания некоторая система, описываемая следующим дифференциальным уравнением второго порядка:

$$\frac{d^2}{dt^2}X(t) + X(t) = 0$$

Известно, что в начальный момент времени (t=0) отклонение системы от точки равновесия равно нулю, а амплитудного значения X=1 оно достигает при t=π/2. На языке математики эти факты можно выразить следующими краевыми условиями:

$$X(0) = 0 \qquad X\left(\frac{\pi}{2}\right) = 1$$

Заданное нами выше уравнение является примером двухточечной линейной (так как при необходимости оно может быть преобразовано в систему из двух линейных ОДУ) нежесткой краевой задачи, и наиболее оптимальный способ ее решения связан с переходом к задаче Коши с помощью метода пристрелки.

Реализуя метод стрельбы, прежде всего нужно определиться с недостающим начальным условием на левой границе интервала. В общем случае считается, что это может быть сделано произвольным образом, однако на практике, естественно, можно найти

довольно точное приближение за счет того, что каждый из подбираемых параметров имеет вполне реальный физический смысл. Так, например, в нашем случае первая производная по X(t) — это мгновенная линейная скорость колебательной системы. Естественно, что хоть ее значение и будет максимальным в момент времени t=0, по величине оно не может быть очень большим (такую оценку можно сделать, сопоставив имеющиеся сведения об амплитудном значении отклонения и периоде колебаний).

Учитывая приведенные выше доводы, предположим, что X'(0) равняется, например, трем. Решая заданную таким образом задачу Коши с помощью блока Given-Odesolve, получим функцию решения, график которой приведен на рис. 14.25.

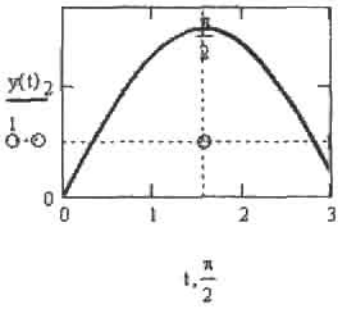


Рис. 14.25. Первое приближение для решения краевой задачи

Как видно из графика, функция решения принимает в точке t=π/2 большее значение, чем оно должно быть исходя из начальных условий. А это означает, что величину ее первой производной нужно сделать несколько меньшей, положив, например, X'(0)=2. Решив при таких значениях начальных условий дифференциальное уравнение, обнаружим, что реально скорость колебательной системы в исходной точке должна быть еще меньше (рис. 14.26). При величине же X'(0)=0.5 окажется, что максимум функции решения лежит ниже заданного изначально уровня. И лишь определив искомый параметр как 1, находим подходящее приближение.

Описанная последовательность действий демонстрирует (конечно, в очень упрощенной форме) основной принцип метода стрельбы. То есть, зная значение функций решения в точке правой границы интервала, очень часто можно, решив последовательность задач Коши (исходя из некоторого начального приближения), найти с определенным уровнем точности и решение самой системы ОДУ.

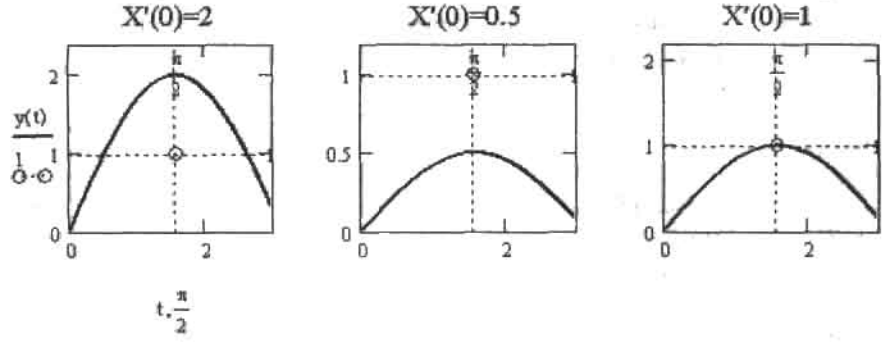


Рис. 14.26. Последовательность приближений в методе стрельбы

Техническая реализация метода стрельбы в Mathcad совсем не сложна, и написать соответствующую программу можно всего за несколько минут. Однако делать этого нет никакого смысла, так как в системе уже имеются встроенные функции, позволяющие преобразовывать краевые задачи в форму задач Коши.

Приведем решение описанной выше краевой задачи о колебаниях с помощью вычислительного блока Given-Odesolve.

Пример 14.21. Решение краевой задачи о колебаниях с помощью вычислительного блока (рис. 14.27)

Given

$$X(0) = 0 \qquad X\left(\frac{\pi}{2}\right) = 1$$

$$\frac{d^2}{dt^2}X(t) + X(t) = 0$$

$$y := \text{Odesolve}(t, 10)$$

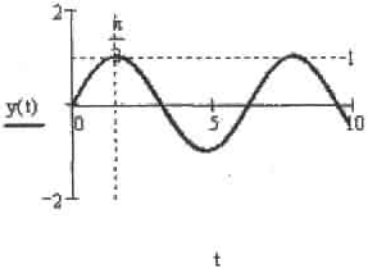


Рис. 14.27. *График решения краевой задачи о колебаниях*

Как вы можете заметить из приведенного примера, никаких особых отличий между решением ОДУ в краевой форме и форме Коши с помощью вычислительного блока нет. Единственное, вы должны корректно задать начальные условия: так, например, вы не можете при решении уравнения второго порядка на одной границе определить значение функции, а на другой — ее производной. Соответствующие начальные условия должны быть определены либо только функциями, либо только производными.

При решении ОДУ более высоких порядков для задания краевых условий появляется довольно значительное количество вариантов. Это связано с тем, что различным может быть как распределение условий между границами, так и относительное количество производных различных порядков. Так, например, можно найти кривую корня ОДУ 4-го порядка, если три начальных условия определены на правой границе, и только один — на левой. Всего же вариантов задания граничных условий для такого уравнения могут быть десятки.

Как и для систем ОДУ, заданных в форме Коши, вычислительный блок Given-Odesolve может находить решения для систем дифференциальных уравнений произвольного порядка с краевыми условиями. Приведем пример простейшей краевой системы ОДУ, описывающей траекторию движения тела, брошенного под углом к горизонту. В лю-

...о действует только сила притяжения F=mg, где m — масса ...одного падения (9,81 м/с²). Тогда, с учетом второго закона Нью... ...а оси x и y можно представить в виде следующих уравнений:

$$m\cdot\frac{d^2x}{dt^2}=0 \qquad m\cdot\frac{d^2y}{dt^2}=-m\cdot g$$

...частии уравнений на m, мы получим проекции вектора ускорения на оси x ...отовую систему дифференциальных уравнений осталось дополнить краевыми: пусть время полета составило пять секунд, за которые тело перемес... ...на расстояние 100 метров. Решение полученной краевой задачи представлено ...мере 14.22.

...ример 14.22. Решение краевой задачи о теле, брошенном под углом ...к горизонту, с помощью вычислительного блока (рис. 14.28)

Given

$$x(0)=0 \qquad x(5)=100$$

$$y(0)=0 \qquad y(5)=0$$

$$\frac{d^2}{dt^2}x(t)=0 \qquad \frac{d^2}{dt^2}y(t)=-9.81$$

$$\begin{pmatrix}x\\y\end{pmatrix}:=\text{Odesolve}\left[\begin{pmatrix}x\\y\end{pmatrix},t,5\right]$$

$$t:=0,0.01..5$$

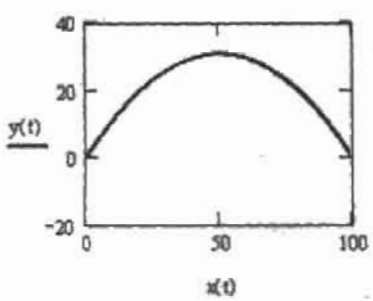


Рис. 14.28. Траектория движения тела, брошенного под углом к горизонту

Помимо вычислительного блока, для решения краевых задач в Mathcad имеются две встроенные функции, реализующие метод пристрелки. Первая из них, более простая и важная практически функция sbval(z,x0,x1,D,load,score), осуществляет поиск недостающих параметров для двухточечных краевых задач. Данная функция требует определения весьма значительного количества параметров:

- z. Вектор приближений, в котором вы должны определить исходные значения для недостающих на левой границе условий. К заданию этого параметра стоит подойти

предельно аккуратно, так как верный выбор прибл... значительное влияние на точность результата;

- x0. Левая граница интервала решения;
- x1. Правая граница интервала решения;
- D(x,y). Векторная функция, описывающая линейное дифф... или систему линейных дифференциальных уравнений. За... при использовании рассмотренных нами ранее встроенны... ченных для решения ОДУ и систем ОДУ в форме задачи Ко...
- load(x0,z). Векторная функция двух переменных, описывающа... на левой границе промежутка. Представляет собой вектор из N... количество уравнений системы), каждый из которых является ... соответствующей функции вектора y параметра D. Если начальн... некоторой функции неизвестно, в качестве элемента load следует ... величину из вектора приближений z;
- score(x1,y). Векторная функция, служащая для задания правых границ... При ее определении следует учитывать одну очень важную и совершенно неочевид... ную техническую особенность: элементы вектора score должны быть заданы как разности начальных значений и соответствующих им значений функций. Все дело в том, что алгоритм, лежащий в основе функции sbval, использует текущие величины score в качестве меры точности подобранных приближений.

Результатом работы функции sbval является вектор с найденными значениями недостающих для представления системы дифференциальных уравнений в форме задачи Коши начальных условий. Для того же, чтобы решить непосредственно ОДУ или систему ОДУ, следует использовать одну из стандартных встроенных функций, например rkfixed. При этом в качестве ее параметра y0 можно использовать уже определенную выше функцию load (обозначив, естественно, вектор результата как z) (см. пример 14.23).

Читая описание параметров функции sbval, вы, наверное, испытали легкую дрожь от чрезмерного обилия переменных, векторов, функций и постоянных. Однако на самом деле в решении краевой задачи для ОДУ или системы ОДУ с использованием данной функции нет ничего сложного. Убедимся в этом на конкретном примере — решим рассмотренную выше краевую задачу о движении тела, брошенного под углом к горизонту.

Пример 14.23. Решение краевой задачи о движении тела, брошенного под углом к горизонту, с помощью функции sbval

Чтобы воспользоваться функцией sbval, сведем нашу систему дифференциальных уравнений второго порядка (см. пример 14.22) к системе дифференциальных уравнений первого порядка, левые части которых содержат только производную. Правые же части нам нужно будет задать как векторную функцию D (правой части каждого уравнения ставится в соответствие элемент вектора y, являющегося параметром D).

Первое уравнение можно преобразовать к системе из двух уравнений первого порядка, введя новую функцию z(t), равную первой производной x(t) по t:

$$\frac{d}{dt}z(t) = 0 \qquad y_0$$

$$\frac{d}{dt}x(t) = z(t) \qquad y_1$$

По аналогичному принципу преобразуется и второе уравнение:

$$\frac{d}{dt}p(t) = -g \qquad y_2$$

$$\frac{d}{dt}y(t) = p(t) \qquad y_3$$

Теперь, когда система задана в корректной форме, приступим к ее решению с помощью функции sbval. Краевые условия определим такими же, как в примере 14.22.

$$x(0) = 0 \qquad x(5) = 100$$

$$y(0) = 0 \qquad y(5) = 0$$

Задаем вектор приближений для искомых неизвестных начальных условий на левой границе интервала и границы самого интервала:

$$z := \begin{pmatrix} 20 \\ 20 \end{pmatrix} \qquad t0 := 0 \qquad t1 := 5$$

Определяем вектор-функцию D для системы уравнений:

$$D(t, y) := \begin{pmatrix} 0 \\ y_0 \\ -9.81 \\ y_2 \end{pmatrix}$$

Задаем вектор значений функций системы на левой границе интервала:

$$load(t0, z) := \begin{pmatrix} z_0 \\ 0 \\ z_1 \\ 0 \end{pmatrix}$$

Задаем краевые начальные условия на правой границе интервала:

$$score(t1, y) := \begin{pmatrix} y_1 - 100 \\ y_3 \end{pmatrix}$$

Вычисляем недостающие начальные условия на левой границе интервала:

$$z := sbval(z, t0, t1, D, load, score) \qquad z = \begin{pmatrix} 20 \\ 24.525 \end{pmatrix}$$

Решение полученной задачи Коши с помощью метода Рунге–Кутта с фиксированным шагом (рис. 14.29):

$$y0 := load(t0, z)$$

$$R := rkfixed(y0, t0, 5, 1000, D)$$

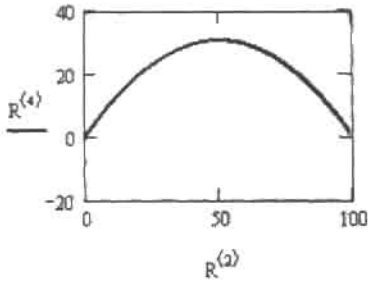


Рис. 14.29. Траектория движения тела, брошенного под углом к горизонту (решение краевой задачи идентично полученному с помощью вычислительного блока)

Обычный метод пристрелки, реализуемый функцией sbval, эффективен лишь в случае непрерывности первых производных функций решения. Многие же физические модели подразумевают наличие точек разрыва на кривых искомых функций, и использовать для решения таких задач стандартный метод стрельбы нельзя.

Однако решить подобные системы все же возможно, если вести пристрелку сразу из обеих границ интервала. В Mathcad имеется специальная встроенная функция bvalfit(z1,z2,x0,x1,xf,D,load1,load2,score), реализующая эту идею.

Так как принципы, лежащие в основе работы bvalfit, довольно непросты, то и параметров для нее нужно задать рекордное количество — целых девять:

- z1 — вектор исходных приближений для недостающих начальных условий на левой границе интервала; задается точно так же, как в случае функции sbval;
- z2 — вектор приближений для недостающих условий на правой границе интервала (одно из принципиальных отличий bvalfit от sbval заключается в том, что она определяет недостающие для корректного задания задачи Коши параметры для обеих границ расчетного промежутка);
- x0 — левая граница промежутка решения;
- x1 — правая граница промежутка решения;
- xf — точка внутри расчетного интервала, в которой происходит сшивка левого и правого решений. Исходя из математического смысла встроенной функции bvalfit, чаще всего xf — это точка, в которой производная функции решения имеет разрыв;
- D(x,y) — вектор-функция, описывающая дифференциальное уравнение или систему дифференциальных уравнений. Задается так же, как для всех рассмотренных нами ранее функций решения систем ОДУ;
- load1(x0,z1) — векторная функция, описывающая начальные условия на левой границе интервала. Количество ее элементов должно совпадать с количеством уравнений в системе. Известные приближения должны быть заданы численно, искомые — соответствующими элементами вектора z1;
- load2(x1,z2) — вектор-функция для условий на правой границе интервала. Задается точно так же, как load1;
- score(xf,y) — векторная функция, описывающая поведение функций решений в точке xf. Обычно является просто условием сшивки и определяется равной вектору y. Однако возможны и более сложные условия, задающие, например, разрыв на искомой кривой.

Результатом работы функции bvalfit является матрица из двух столбцов, в первом из которых содержатся искомые приближения для правой границы интервала, а во втором — соответственно для левой.

Чтобы затем корректно визуализировать результат расчета, соответствующие кривые должны быть составлены из двух различных фрагментов, сшитых в точке xf. Сделать это можно простым решением двух задач Коши: одной — на промежутке от x0 до xf, другой — на отрезке от xf до x1. Естественно, что при этом нужно использовать различные начальные приближения.

Воспользуемся функцией bvalfit, чтобы решить следующую задачу. Пружинный маятник совершает колебания. В определенный момент времени к нему подвешивают груз. Необходимо определить, как изменится характер колебаний маятника вследствие изменения его массы.

На пружинный маятник массой m, совершающий колебания, действуют две противоположно направленные силы: сила притяжения F=mg и сила Гука F=–kx (где k — коэффициент жесткости пружины, x — смещение маятника). С учетом второго закона Ньютона запишем суперпозицию этих сил как

$$m \cdot a = -k \cdot x + m \cdot g$$

Сократим обе части равенства на m:

$$a = -\frac{k}{m} \cdot x + g$$

Представив ускорение a в дифференциальной форме, мы получим уравнение колебаний пружинного маятника.

$$\frac{d^2}{dt^2} x(t) = -\frac{k}{m} \cdot x + g$$

Чтобы получить краевую задачу, функция решения которой имеет разрыв, зададим краевые условия и условия изменения характера колебаний. Пусть колебания маятника наблюдались в течение десяти секунд. В начальный момент времени смещение маятника было равно 5, в конечный — 10. Груз, подвешенный на четвертой секунде наблюдения, увеличил массу маятника в 15 раз. Определить, как меняется характер колебаний на протяжении всего времени наблюдения. Решение полученной задачи представлено в примере 14.24.

Пример 14.24. Использование функции bvalfit в решении задачи об изменении характера колебаний маятника

Задаем параметры колебательной системы — жесткость пружины и исходную массу маятника:

$$k := 100 \qquad m := 5$$

Определяем приближения для неизвестных начальных условий — скорости колебаний:

$$z1_0 := 0 \qquad z2_0 := 0$$

Задаем границы временного интервала:

$$t0 := 0 \qquad t1 := 10$$

Определяем точку сшивки tf:

$$tf := 4$$

Поскольку решаемое дифференциальное уравнение имеет второй порядок, его нужно преобразовать в систему дифференциальных уравнений первого порядка по описанному в примере 14.23 принципу. Полученную систему затем задаем в виде вектор-функции D, внутри которой указываем условие изменения массы маятника.

$$D(t,y) := \begin{bmatrix} -y_1 \cdot \dfrac{k}{m \cdot [1 + 14(t > 4)]} + 9.81 \\ y_0 \end{bmatrix}$$

Определяем векторные функции начальных условий для обеих границ временного интервала:

$$load1(t0,z1) := \begin{pmatrix} z1_0 \\ 5 \end{pmatrix} \qquad load2(t1,z2) := \begin{pmatrix} z2_0 \\ 10 \end{pmatrix}$$

Задаем условие сшивки:

$$score(tf,y) := y$$

Используя функцию bvalfit, определяем недостающие начальные условия:

$$z := bvalfit(z1,z2,t0,t1,tf,D,load1,load2,score)$$

$$z = (-16.755 \quad 6.386)$$

На основании полученных данных решаем соответствующие задачи Коши методом Рунге–Кутта с фиксированным шагом (рис. 14.30):

$$R0 := rkfixed\left[\begin{pmatrix} z_{0,0} \\ 5 \end{pmatrix}, 0, 4, 100, D\right] \qquad R1 := rkfixed\left[\begin{pmatrix} z_{0,1} \\ 10 \end{pmatrix}, 10, 4, 100, D\right]$$

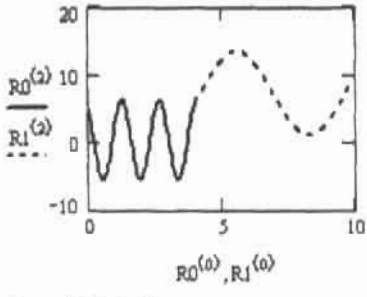


Рис. 14.30. Изменение характера колебаний пружинного маятника с увеличением массы

Проанализируем полученное решение. Исходя из графика, груз был подвешен к маятнику в момент его максимального отклонения от положения равновесия. Вследствие увеличения массы скорость колебаний уменьшилась, а значит, увеличился их период. Амплитуда колебаний также незначительно возросла, а новое положение равновесия сместилось относительно предыдущего в область более сильных растяжений пружины.

Изучив приведенный пример, вы, наверное, согласитесь, что, несмотря на обилие используемых параметров, системы дифференциальных уравнений решаются с помощью функции bvalfit, в общем, совсем не сложно. Правда, на некоторые моменты все же стоит обратить внимание.

- Если вы не читали гл. 2, то вам, скорее всего, будет непонятна использованная в правой части дифференциального уравнения запись условия изменения массы:

$$m\cdot[1 + 14(t > 4)]$$

  Все дело в том, что Mathcad умеет анализировать логические выражения и возвращает 0, если они ложны, и 1, если их условие оказывается верным. Таким образом, приведенная запись означает, что выражение принимает значение m при отрицательных t и 15m — при положительных.

- Приближения для начальных значений должны быть заданы как векторы. Если же, как в нашем случае, на каждой границе имеется только по одному неизвестному параметру, то приближения для них должны быть определены как первые векторные элементы (но ни в коем случае не как простые переменные):

$$z1_0 := 0 \qquad z2_0 := 0$$

- Функция rkfixed, в отличие от вычислительного блока Given-Odesolve, способна находить решения системы ОДУ, даже если граничные значения определены на правой границе интервала. Этот факт был использован нами для построения графика функций решения (особое внимание обратите на последовательность задания параметров):

$$R1 := \text{rkfixed}\left[\begin{pmatrix} z_{0,1} \\ 10 \end{pmatrix}, 10, 4, 100, D\right]$$

Особенности решения краевых задач в Mathcad были изложены нами в этом разделе довольно подробно. К сожалению, в Mathcad нет встроенной функции, которая реализовывала бы для решения краевых задач одну из многочисленных, созданных для этой цели разностных схем, однако при необходимости вы сможете без проблем задать соответствующий алгоритм и самостоятельно, используя возможности программирования системы. Об этом мы и поговорим в следующем разделе.

14.3.2. Разностная схема для решения краевых задач

К сожалению, далеко не все краевые задачи можно решить методом пристрелки, реализованном во встроенных функциях sbval и bvalfit: при решении уравнений, содержащих быстро растущие или быстро убывающие решения, например $e^{\lambda t}$, алгоритм оказывается крайне неточным. Для решения подобных задач существует ряд численных методов, в основе которых лежит построение той или иной разностной схемы. Но в Mathcad нет собственных алгоритмов, позволяющих находить решение конечно-разностным методом. Однако построить разностную схему совсем не сложно, воспользовавшись программными средствами. Чтобы убедиться в сказанном, рассмотрим принцип метода конечных разностей на конкретном примере.

Решим краевую задачу

$$\frac{d^2}{dt^2}x(t) - 0.05\cdot\frac{d}{dt}x(t) + 5\cdot x(t) - \sin(0.05\cdot t) = 0$$

на интервале (0, 20) со следующими граничными условиями:

$$x(0) = 0 \qquad x(20) = 1$$

Для построения разностной схемы на расчетный интервал необходимо нанести сетку линий. В нашем случае уравнение содержит производную по одной переменной, поэтому сетка окажется одномерной. Для уравнений в частных производных, рассмотренных ниже, интервал поиска покрывается двумерной или трехмерной сеткой. Точки пересечения линий называются узлами, в которых ищутся значения функции. Полученные решения называются сеточными функциями. Понятно, что они не могут быть непрерывными и представляют собой множество, поскольку вычислены при определенных значениях аргумента. Поэтому при построении графика для получения гладкой кривой вычисленные ранее приближения интерполируются подходящим полиномом.

Расстояния между узлами определяются выбранным шагом сетки h. На каждом шаге дифференциальное уравнение аппроксимируется конечно-разностным алгебраическим, где в качестве переменной выступает искомое значение функции в узле i. Заметим, что в нашем уравнении содержится производная первого и второго порядков. Вспомнив ее определение, знакомое читателю еще со школы, приближенно можно записать:

$$x' = \frac{x_{i+1} - x_{i-1}}{2h}$$

$$x'' = \frac{\frac{x_{i+1} - x_i}{h} - \frac{x_i - x_{i-1}}{h}}{h} = \frac{x_{i-1} - 2 \cdot x_i + x_{i+1}}{h^2}$$

Если граничные условия заданы в форме производных, их также необходимо заменить разностными аналогами. Учитывая это, с помощью простых преобразований мы получим систему линейных алгебраических уравнений, называемую разностной схемой. Количество шагов N определяет количество алгебраических уравнений в системе: N+1 (с учетом граничных условий). К разностной схеме предъявляются следующие требования: во-первых, она должна наилучшим образом аппроксимировать дифференциальное уравнение, во-вторых, при стремлении длины шага к нулю решение разностной задачи должно сходиться к решению дифференциальной задачи. Безусловно, для обеспечения сходимости необходима также устойчивость разностной схемы.

Подставив левое граничное условие в первое уравнение, а правое — в последнее, можно обнаружить, что матрица коэффициентов при неизвестных A симметрична и трехдиагональна. Таким образом, задача сводится к решению системы

$$A \cdot x = B$$

где B и x — вектор правых частей и неизвестных соответственно. Программа, реализующая изложенную идею, представлена ниже:

Пример 14.25. Программная реализация разностной схемы для приведенной краевой задачи

Задаем интервал интегрирования, величину шага, коэффициенты уравнения, граничные условия и узловые точки расчетной сетки.

$$N := 500 \qquad a := 0 \qquad b := 20 \qquad h := \frac{b - a}{N}$$

$$u(t) := 0.05 \qquad v(t) := -5 \qquad w(t) := \sin(0.05 \cdot t)$$

$$x_a := 0 \qquad x_b := 1$$

$$i := 1 .. N - 1 \qquad t_i := a + i \cdot h$$

Определяем для системы разностных уравнений трехдиагональную матрицу коэффициентов при неизвестных.

$$A_{i,i} := 2 + h^2 \cdot v(t_i) \qquad A_{i,i+1} := \frac{h}{2} \cdot u(t_i) - 1 \qquad A_{i,i-1} := \frac{-h}{2} \cdot u(t_i) - 1$$

$$A := \text{submatrix}(A, 1, N - 1, 1, N - 1)$$

Задаем вектор правых частей

$$B := \left| \begin{array}{l} B_0 \leftarrow -h^2 \cdot w(t_1) + x_a \cdot \left(\frac{h}{2} \cdot u(t_1) + 1 \right) \\ \text{for } i \in 1 .. N - 3 \\ \quad B_i \leftarrow -h^2 \cdot w(t_{i+1}) \\ B_{N-2} \leftarrow -h^2 \cdot w(t_{N-1}) + x_b \cdot \left(\frac{-h}{2} \cdot u(t_{N-1}) + 1 \right) \\ B \end{array} \right.$$

Решаем систему разностных уравнений

$$x := \text{lsolve}(A, B)$$

$$x := \text{stack}(x_a, x, x_b)$$

Результат работы разностной схемы представлен на рис. 14.31.

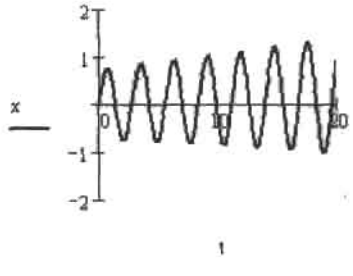


Рис. 14.31. Результат работы разностной схемы

Отметим некоторые особенности построенной разностной схемы. Вторая производная должна быть в первой степени и не содержать каких-либо сомножителей, краевые условия должны быть поставлены строго на границах интервала поиска. Коэффициенты при остальных слагаемых, а также свободный член уравнения в общем случае могут являться функциями, поэтому в программе заданы в виде u(t), v(t) и w(t). Точность варьируется в зависимости от количества шагов, однако не следует забывать, что при слишком

большом их увеличении резко возрастает время расчетов, значительная часть которого используется функцией lsolve. Связано это с тем, что встроенному алгоритму приходится работать с матрицами больших размеров. В нашем случае, когда количество узловых точек равно 500, матрица A имеет размерность 500·500. Нетрудно понять, что при увеличении N в арифметической прогрессии скорость вычислений будет падать в геометрической прогрессии. Например, уже при N=1000 расчет длится недопустимо долго.

К счастью, решить возникшую проблему довольно просто. Как известно, самым популярным алгоритмом решения систем линейных алгебраических уравнений является метод последовательного исключения Гаусса. В классической форме для N–1 уравнений системы он требует порядка N^3 арифметических операций. Однако для систем, содержащих матрицы особой структуры имеется множество вариантов этого алгоритма, которые позволяют во много раз повысить его быстродействие. Выше мы упоминали, что матрица A имеет ленточную структуру: только главная диагональ и две соседние содержат ненулевые элементы (рис. 14.32).

A =

	0	1	2	3	4	5	6	7	8
0	1.992	-0.999	0	0	0	0	0	0	0
1	-1.001	1.992	-0.999	0	0	0	0	0	0
2	0	-1.001	1.992	-0.999	0	0	0	0	0
3	0	0	-1.001	1.992	-0.999	0	0	0	0
4	0	0	0	-1.001	1.992	-0.999	0	0	0
5	0	0	0	0	-1.001	1.992	-0.999	0	0
6	0	0	0	0	0	-1.001	1.992	-0.999	0
7	0	0	0	0	0	0	-1.001	1.992	-0.999
8	0	0	0	0	0	0	0	-1.001	1.992
9	0	0	0	0	0	0	0	0	-1.001

Рис. 14.32. Матрица коэффициентов системы уравнений, формирующих разностную схему

Для решения систем с подобными матрицами существует модификация алгоритма Гаусса, называемая методом прогонки. Замечателен он тем, что для нахождения корней системы N–1 уравнений требует порядка N операций, в отличие от стандартного алгоритма. Попытаемся разобраться в сущности метода прогонки и напишем соответствующую программу, которая наверняка окажется полезной для читателя, столкнувшегося с необходимостью решения систем уравнений с трехдиагональной матрицей.

Любое решение, полученное методом прогонки, описывается уравнением

$$x_i = \alpha_{i+1} \cdot x_{i+1} + \beta_{i+1}$$

где α_{i+1}, β_{i+1} – прогоночные коэффициенты. Они определяются на первом этапе работы алгоритма, называемого прямым ходом прогонки. В классическом варианте метода Гаусса этому этапу соответствует приведение матрицы к треугольному виду. Обратим внимание, что, согласно уравнению, предыдущий корень вычисляется исходя из последующего. Такая последовательность называется обратным ходом алгоритма прогонки, в процессе которого непосредственно ищется вектор решений. Описанные действия

представлены ниже в виде программы. Заметим, что скорость расчета значительно возрастает при замене функции lsolve в примере на наш алгоритм.

Пример 14.26. Алгоритм прогонки

$$
\begin{aligned}
x := \Bigg| & \left(\alpha_1 \leftarrow \frac{-A_{0,1}}{A_{0,0}} \quad \beta_1 \leftarrow \frac{B_{0,0}}{A_{0,0}} \right) \\
& \text{for } i \in 1..N-3 \\
& \quad \alpha_{i+1} \leftarrow \frac{-A_{i,i+1}}{A_{i,i} + A_{i+1,i} \cdot \alpha_i} \\
& \quad \beta_{i+1} \leftarrow \frac{-A_{i+1,i} \cdot \beta_i + B_{i,0}}{A_{i,i} + A_{i+1,i} \cdot \alpha_i} \\
& x_{N-2} \leftarrow \frac{-A_{N-2,N-3} \cdot \beta_{N-2} + B_{N-2,0}}{A_{N-2,N-2} + A_{N-2,N-3} \cdot \alpha_{N-2}} \\
& \text{for } i \in N-3..0 \\
& \quad x_i \leftarrow \alpha_{i+1} \cdot x_{i+1} + \beta_{i+1} \\
& x
\end{aligned}
$$

Метод прогонки также используется для решения систем разностных уравнений, аппроксимирующих дифференциальные уравнения в частных производных. Об этой группе дифференциальных уравнений мы поговорим в следующем разделе.

14.4. Дифференциальные уравнения в частных производных

Пожалуй, мы будем не столь уж не правы, назвав дифференциальные уравнения в частных производных самым сложным разделом не только теории численных методов, но и всей математики в целом. Уравнения эти, называемые также уравнениями математической физики, с удивительной точностью описывают самые разнообразные реальные процессы, начиная от колебаний струны и заканчивая всевозможными явлениями в биосфере. Поэтому подавляющее большинство современных разработок, которые ведутся специалистами по численным методам, относятся именно к области дифференциальных уравнений в частных производных.

Для численного решения дифференциальных уравнений в частных производных существует множество разностных схем, вид которых определяется тем или иным типом уравнения. Чтобы понять принципы работы встроенных средств Mathcad, возможности которых в решении уравнений математической физики, начиная с 11 версии программы, значительно расширились, мы приведем дополнительно примеры построения разностных схем для наиболее известных уравнений.

В общем виде дифференциальные уравнения второго порядка, которые описывают огромное количество разнообразных физических явлений, можно записать следующим образом:

$$A\cdot\frac{\partial^2 U}{\partial x^2}+B\cdot\frac{\partial^2 U}{\partial y^2}+C\cdot\frac{\partial U}{\partial x}+D\cdot\frac{\partial U}{\partial y}+E\cdot U=F$$

Параметры реальных процессов, как правило, зависят от времени, поэтому оно может выступать в качестве одной из переменных функции U. Уравнение (как и сам процесс) в таком случае называется нестационарным (или неустановившимся).

Когда коэффициенты при производных — постоянные величины, уравнение линейно и в простейших случаях разрешимо аналитически. Если коэффициенты являются функциями того или иного параметра, уравнение называется нелинейным и, как правило, не имеет аналитического решения.

Линейные уравнения математической физики подразделяются на три типа в зависимости от соотношения входящих в них коэффициентов, которое определяется физическим смыслом решаемой задачи:

- если $b^2-ac=0$, уравнение называется параболическим;
- если $b^2-ac>0$, уравнение называется гиперболическим;
- если $b^2-ac<0$, уравнение называется эллиптическим.

В данном разделе мы рассмотрим простейшие примеры уравнений математической физики, относящиеся к трем указанным выше видам соответственно: одномерное уравнение теплопроводности, волновое уравнение и уравнение Пуассона. Но прежде поясним специфику постановки задач. Решением любого дифференциального уравнения в частных производных является функция нескольких аргументов, для каждого из которых необходимо задать область определения. Так как переменные характеризуют параметры реальных процессов, для них область определения всегда положительна. Если в качестве переменных функции U выступают пространственная и временная координаты x и t, то начальные условия определяются как зависимости U(x) или ее первой производной на левой границе интервала существования t. Граничные условия, напротив, представляют собой зависимость V от времени t в крайних точках области определения пространственной координаты.

Принципы, лежащие в основе построения разностных схем существенно не отличаются от изложенных в подразд. 14.3.2. Правда, задача несколько усложняется в связи с тем, что уравнение содержит производные функции по нескольким переменным. Поэтому даже в простейшем случае, когда функция U зависит всего от двух координат x и t, расчетная область покрывается двумерной сеткой, в узлах которой вычисляются соответствующие друг другу значения x и t. Теоретически разностную схему можно построить и для уравнений с тремя и четырьмя независимыми переменными. Например, волновая функция, являющаяся решением временного уравнения Шредингера,

$$i\hbar\cdot\frac{\partial\psi}{\partial t}=\frac{-\hbar^2}{2m}\cdot\left(\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2}+\frac{\partial^2\psi}{\partial z^2}\right)+V(x,y,z,t)\,\psi$$

описывает движение частицы в пространстве (x,y,z) и времени t под действием потенциала V, то есть зависит от четырех аргументов. Тогда сетка будет представлена в четырехмерном пространстве. Естественно, в таком случае во много раз возрастет коли-

чество разностных уравнений, аппроксимирующих искомую функцию на каждом шаге, а значит, и время расчетов. Но можно поступить и проще, перейдя от временного уравнения к стационарному, которое описывает положение частицы в фиксированный момент времени. Еще более упрощаются вычисления, если использовать набор дискретных значений одной из пространственных координат. Безусловно, выигрыш в сокращении времени расчетов окажется значительным, однако полученные таким образом решения уже не будут отражать реальной динамики квантового объекта.

14.4.1. Параболические уравнения

Рассмотрим в качестве примера параболического уравнения одномерное уравнение теплопроводности

$$\frac{\partial T}{\partial t} = a^2 \cdot \frac{\partial^2 T}{\partial x^2}$$

описывающее процесс распределения тепла в неравномерно нагретом металлическом стержне, поверхность которого покрыта теплоизоляционным материалом. Будем пренебрегать толщиной стержня, предполагая, что в любой точке произвольно выбранного сечения температура постоянна (рис. 14.33).

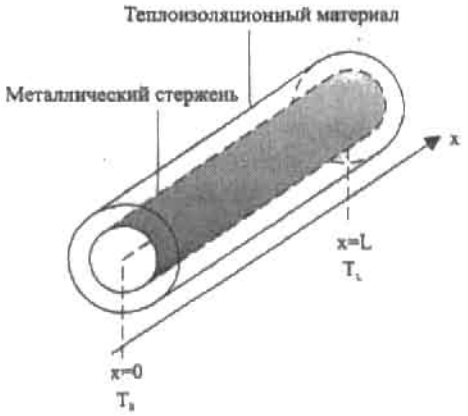


Рис. 14.33. Модель распространения тепла в неравномерно нагретом стержне

В течение определенного промежутка времени на концах стержня поддерживаются постоянные температуры T_0 и T_L, что необходимо указать в граничных условиях:

$$T(t,0) = T_0$$

$$T(t,L) = T_L \qquad 0 \le t \le A$$

Начальное условие задает температуру стержня в момент времени t=0:

$$T(0,x) = T_{исх}(x) \qquad 0 \le x \le L$$

Коэффициент температуропроводности *a* является характеристикой материала, учитывающей его плотность, теплоемкость и теплопроводность. Естественно, эти параметры

могут меняться при различных температурных режимах, и в реальных условиях a зависит от температуры. Однако для упрощения задачи флуктуациями свойств вещества можно пренебречь.

Теперь, когда задано уравнение, описывающее процесс диффузии тепла, и все необходимые условия, можно приступить к построению разностной схемы.

Для этого нанесем на расчетную область сетку, в узлах которой будем искать функцию $T_{j,i}$ — температуру участка i в момент времени j. Заметим, что длина шага k и h по временной и пространственной координатам j и i может и не совпадать. Другими словами

$$T_{j+1} = T_j + k$$

$$T_{i+1} = T_i + h$$

Далее запишем исходное дифференциальное уравнение в конечно-разностной форме (аналогично решению краевой задачи в предыдущем разделе):

$$\frac{T_{j+1,i} - T_{j,i}}{k} = \frac{a^2 \cdot T_{j,i-1} - 2 \cdot T_{j,i} + T_{j,i+1}}{h^2}$$

Общая температура стержня при t=j будет решением уравнения на временном слое j. Обратите внимание на то, что температуру в следующий момент времени j+1 можно выразить, исходя из ранее найденных значений T в трех соседних узлах i предыдущего временного слоя j. Такого рода схемы называются явными.

$$T_{j+1,i} = c \cdot T_{j,i-1} + (1 - 2 \cdot c) \cdot T_{j,i} + c \cdot T_{j,i+1}, \text{ где } c = \frac{a^2 \cdot k}{h^2}$$

Более наглядно явная схема аппроксимации для уравнения теплопроводности изображена в виде шаблона на рис. 14.34.

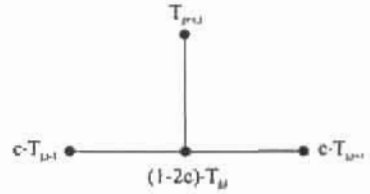


Рис. 14.34. Шаблон явной разностной схемы для решения уравнения теплопроводности

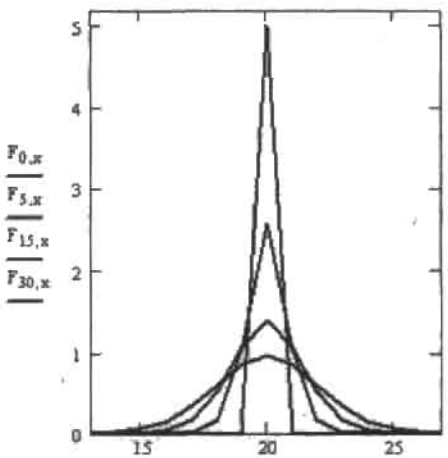
Решение, полученное с помощью приведенной разностной схемы, представлено ниже в виде программы.

Отметим одну очень важную особенность, которую следует учитывать при вводе параметров задачи. При определенном их сочетании явная разностная схема становится неустойчивой. Это связано с тем, что ошибка аппроксимации, полученная на одном слое, увеличивается при переходе к следующему. Когда же разностная схема устойчива, в процессе вычислений ошибка уменьшается. В нашем случае показателем сходимости решения является множитель c. Схема устойчива, если он принимает значения от 0 до 5. В противном случае решение расходится (см. второй рисунок в примере 14.27).

Пример 14.27. Решение одномерного уравнения теплопроводности (рис. 14.35)

$$
F := \left|
\begin{array}{l}
(A \leftarrow 30 \quad L \leftarrow 40 \quad n \leftarrow 30 \quad m \leftarrow 500) \\
\left(h \leftarrow \dfrac{A}{n-1} \quad k \leftarrow \dfrac{L}{m-1} \quad a \leftarrow 1 \quad c \leftarrow \dfrac{a^2 \cdot k}{h^2} \right) \\
\text{for } j \in 0..A-1 \\
\quad \text{for } i \in 1..L-1 \\
\quad\quad \left|
\begin{array}{l}
T_{0,20} \leftarrow 5 \\
T_{j,0} \leftarrow 0 \\
T_{j,40} \leftarrow 0 \\
T_{j+1,i} \leftarrow c \cdot \left(T_{j,i-1} - 2 \cdot T_{j,i} + T_{j,i+1} \right) + T_{j,i}
\end{array}
\right. \\
T
\end{array}
\right.
$$

$x := 0..40$



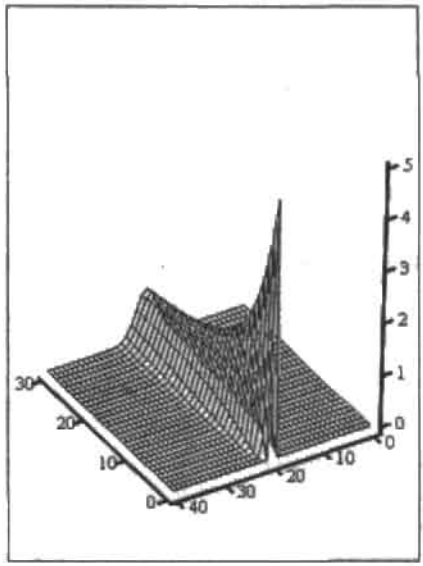
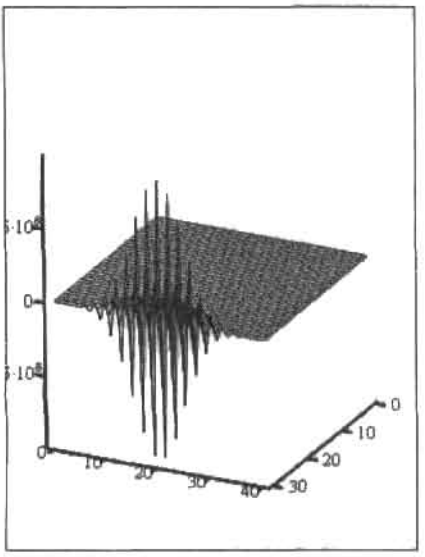


Рис. 14.35. Профили температуры на разных временных слоях ($c<1/2$) и поверхность решения уравнения теплопроводности

Неверное решение, полученное с помощью разностной схемы при $c>1/2$, представлено на рис. 14.36.



F

Рис. 14.36. Неверное решение, полученное с помощью разностной схемы при c>1/2

К сожалению, применение явных разностных схем ограничивается слишком малой длиной шага, используемой ими. Безусловной альтернативой, лишенной такого недостатка, является неявная разностная схема, обеспечивающая сходимость решения при любом значении параметра *c*.

Запишем конечно-разностную формулу, аппроксимирующую согласно неявной схеме уравнение теплопроводности:

$$\frac{T_{i,j+1} - T_{i,j}}{k} = a^2 \cdot \frac{T_{i-1,j+1} - 2 \cdot T_{i,j+1} + T_{i+1,j+1}}{h^2}$$

Введем коэффициент *c* и перепишем разностное уравнение в следующем виде:

$$c = \frac{a^2 \cdot k}{h^2}$$

$$-c \cdot T_{i-1,j+1} + (1 + 2c) T_{i,j+1} - c \cdot T_{i+1,j+1} = T_{i,j}$$

Принципиальным отличием неявных разностных схем является то, что на данном временном слое для поиска решения используются значения функции, рассчитанные на этом же слое, а не на предыдущем, как при вычислениях с помощью явных схем.

Сказанное выше иллюстрирует шаблон, изображенный на рис. 14.37.

Легко заметить, что для i=0,1,2...n и j=0,1,2...n последнее равенство представляет собой систему линейных алгебраических уравнений, матрица которой, как и в случае краевых задач, симметрична и имеет трехдиагональную форму.

С учетом этих особенностей представим программный вариант решения уравнения теплопроводности, используя неявную разностную схему.

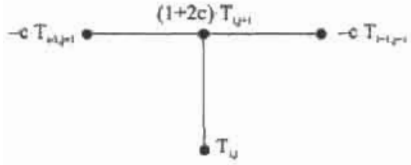


Рис. 14.37. Шаблон аппроксимации уравнения теплопроводности по неявной разностной схеме

Пример 14.28. Программная реализация неявной разностной схемы (рис. 14.38)

$$T(A,L,n,m) := \left| \begin{array}{l} \left(h \leftarrow \frac{A}{n-1} \quad k \leftarrow \frac{L}{m-1} \quad a \leftarrow 1 \quad c \leftarrow \frac{a^2 \cdot k}{h^2} \right) \\ \text{for } i \in 0..n-1 \\ \quad \text{for } j \in 1..n-1 \\ \quad\quad \left| \begin{array}{l} b \leftarrow 1 + 2 \cdot c \\ \left(M_{0,0} \leftarrow b \quad M_{n,n} \leftarrow b \quad M_{j,j} \leftarrow b \right) \\ \left(M_{j-1,j} \leftarrow -c \quad M_{j,j-1} \leftarrow -c \right) \\ \left(T_0 \leftarrow 0 \quad T_n \leftarrow 0 \right) \\ T_i \leftarrow 2 \cdot \sin\left(\pi \cdot \frac{i}{n} \right) \end{array} \right. \\ \text{for } t \in 0..n-1 \\ \quad T^{\langle t+1 \rangle} \leftarrow \text{lsolve}\left(M, T^{\langle t \rangle} \right) \\ T \end{array} \right.$$

$$T := T(30,30,30,10)$$

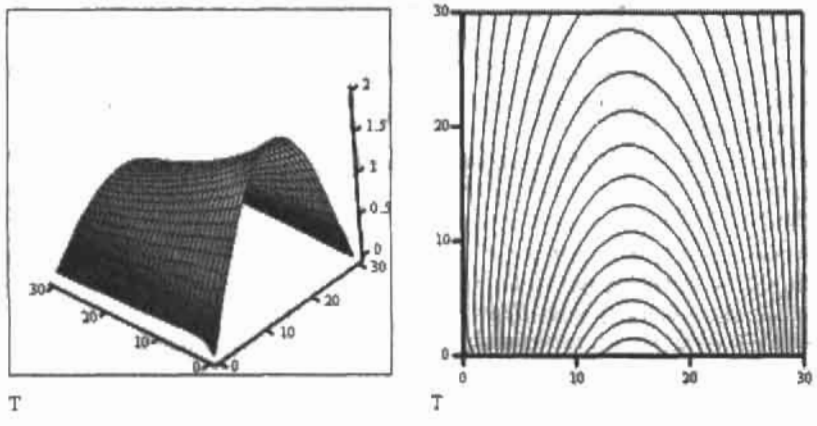


Рис. 14.38. Поверхность и контурный график решения уравнения теплопроводности с помощью неявной разностной схемы

В начале программы определяется произвольный шаг по пространственной и временной координате, а также значение коэффициента *a*. Далее вводятся элементы матрицы M системы линейных уравнений, граничные T_0, T_n и начальные T_i условия. В конце на каждом временном слое t непосредственно вычисляются значения функции в узлах сетки. Заметьте, в примере решение получено при *c*=3, что подтверждает безусловную устойчивость неявных разностных схем.

Для решения линейных (параболических и гиперболических) и нелинейных одномерных уравнений в частных производных или их систем в Mathcad имеется вычислительный блок Given-Pdesolve, в основу работы которого положен метод конечных разностей (или метод сеток).

Итогом работы встроенной функции Pdesolve (u, x, xrange, t, trange, [xpts], [tpts]) является скалярная или векторная функция переменных x и t, представляющая собой решение уравнения или системы уравнений в частных производных. Для получения конечного результата алгоритм интерполирует значения, найденные в узлах сетки.

Задействовав функцию Pdesolve, нужно указать следующие параметры:

- u — искомая функции или вектор функций (в случае системы уравнений), не содержащие имен переменных;
- x — пространственная координата;
- xrange — вектор, задающий интервал поиска по пространственной координате. Содержит значения аргумента x на левой и правой границах промежутка. Элементы вектора должны быть действительными числами;
- t — временная координата;
- trange — вектор, задающий интервал поиска по времени. Его элементы, представляющие собой значения переменной t на границах временного промежутка, могут быть только действительными числами;
- xpts — количество точек дискретизации по пространственной координате. Данный параметр позволяет пользователю влиять на процесс решения. Однако указывать его необязательно, в этом случае количество шагов в направлении x алгоритм определит автоматически наиболее оптимальным способом;
- tpts — количество точек дискретизации по времени. Параметр также не является обязательным для указания.

Прежде чем приступить к решению уравнения теплопроводности с помощью блока Given-Pdesolve, ознакомимся с некоторыми особенностями его использования.

- После слова Given вводится дифференциальное уравнение или система уравнений. Неизвестные функции в уравнениях должны задаваться явно и содержать имена аргументов, например u(x,t).
- В отличие от вычислительного блока Given-Odesolve, производная функции по той или иной переменной указывается в виде нижнего литерального индекса («горячая» клавиша «.»). Например, $u_t(x,t)$ для первой и $u_{xx}(x,t)$ для второй производной.
- Для каждой неизвестной функции задается начальное условие u(x,0) и граничные условия, количество которых равно степени дифференциального уравнения. Граничные условия задаются в форме Дирихле или Неймана. Например, u(0,t)=f(t) или $u_x(0,t)=f(t)$. Разумеется, границы интервалов, указанные вами в начальном и граничных условиях и в маркерах функции Pdesolve, должны совпадать. В противном случае будет выдано сообщение об ошибке.
- Знак равенства в уравнении, начальном и граничных условиях вводится с панели Boolean (*или сочетанием клавиш* Ctrl+=).

- Если уравнение содержит в левой части вторые частные производные, его необходимо представить в виде системы уравнений в первых производных. Например, чтобы корректно задать волновое уравнение $u_{tt}(x,t)=c^2 \cdot u_{xx}(x,t)$ в блоке Given-Pdesolve, следует выразить первую производную по времени через новую функцию $u_t(x,t)=w(x,t)$ и записать систему: $w_t(x,t)=c^2 \cdot u_{xx}(x,t)$, $u_t(x,t)=w(x,t)$. Для новой функции $w(x,t)$ также задается начальное условие, например, $w(x,0)=0$.
- В блоке Given-Pdesolve наряду с дифференциальными уравнениями можно задавать и алгебраические. Например, для системы уравнений с неизвестными функциями $u(x,t)$ и $v(x,t)$ ограничение в виде $u(x,t)+v(x,t)+w(x,t)=0$ позволяет ввести в систему дополнительную неизвестную функцию $w(x,t)$, которая должна быть определена Pdesolve наряду с $u(x,t)$ и $v(x,t)$. Ограничения же в виде неравенств задавать нельзя.
- При необходимости в контекстном меню функции Pdesolve вы можете сменить модификацию аппроксимирующего метода, чтобы увеличить скорость расчета.

Решение одномерного уравнения теплопроводности с применением вычислительного блока приведено в примере.

Пример 14.29. Решение уравнения теплопроводности с помощью вычислительного блока Given-Pdesolve (рис. 14.39)

$$D := 0.05$$

$$L := 1 \qquad T := 0.5$$

Given

$$u_t(x,t) = D \cdot u_{xx}(x,t)$$

$$u(x,0) = 0.02\sin\left(\frac{\pi x}{2}\right) + 20\sin(\pi x)$$

$$u(0,t) = 0 \qquad u(L,t) = 0$$

$$u := \text{Pdesolve}\left[u, x, \begin{pmatrix}0\\L\end{pmatrix}, t, \begin{pmatrix}0\\T\end{pmatrix}, 100, 100\right]$$

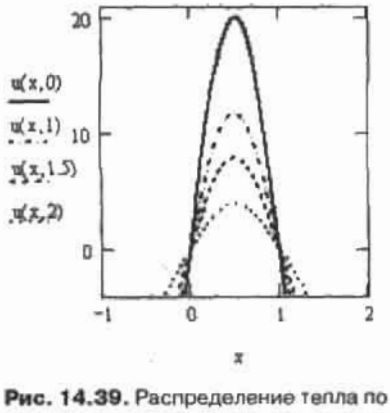


Рис. 14.39. Распределение тепла по стержню с течением времени

14.4.2. Гиперболические уравнения

В качестве примера гиперболического уравнения рассмотрим одномерное волновое уравнение, описывающее колебания струны с фиксированными концами.

$$\frac{\partial^2 V}{\partial t^2} = c^2 \cdot \frac{\partial^2 V}{\partial x^2}$$

Величина коэффициента c зависит от свойств материала струны.

Гиперболическое уравнение можно легко запрограммировать с помощью встроенных средств Mathcad, воспользовавшись все тем же методом конечных разностей. Для этого сначала определим начальные и граничные условия. При колебаниях концы струны не смещаются, поскольку закреплены. Это обстоятельство отражается в граничных условиях:

$$V(t,0) = V_0$$

$$V(t,L) = V_L \qquad 0 \le t \le A$$

Начальные условия задают положение струны и ее скорость в исходный момент времени t_0:

$$V(0, x) = V_{исх}(x)$$

$$V_t(0, x) = 0 \qquad 0 \le x \le L$$

Представим вторые производные, которые содержатся в обеих частях уравнения, в уже знакомой разностной форме:

$$\frac{V_{j,i+1} - 2 \cdot V_{j,i} + V_{j,i-1}}{h^2} = c^2 \cdot \frac{V_{j+1,i} - 2 \cdot V_{j,i} + V_{j-1,i}}{k^2}$$

и приведем формулу к более наглядному виду, выразив значение волновой функции на следующем временном слое j+1 исходя из значений на предыдущих слоях j и j–1:

$$V_{j+1,i} = a^2 \cdot (V_{j,i+1} + V_{j,i-1}) + 2 \cdot (1 - a^2) \cdot V_{j,i} - V_{j-1,i}$$

Нетрудно догадаться, что мы получили явную разностную схему, абсолютная устойчивость которой определяется параметром

$$a^2 = \frac{k^2}{c^2 \cdot h^2}$$

Решение сходится, если его значение не превышает 8. Четырехточечный шаблон рассмотренной схемы и узловые коэффициенты изображены на рис. 14.40.

Теперь, когда определены необходимые уравнения и параметры, приступим к реализации программы. Чтобы излишне ее не загромождать, не будем задавать величины шагов по обеим координатам, приняв известным значение параметра a и нулевую начальную скорость колебания струны.

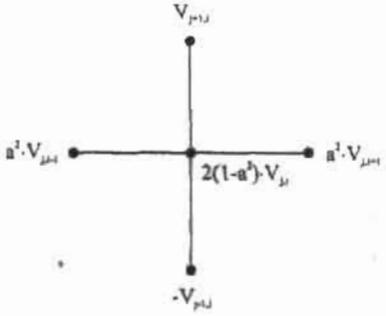
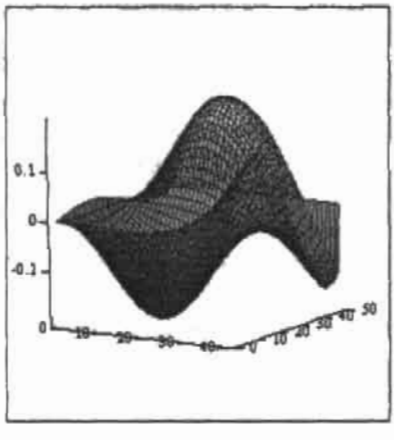


Рис. 14.40. Шаблон явной разностной схемы для волнового уравнения

Пример 14.30. Разностная схема для решения уравнения колебаний (рис. 14.41)

$$
\begin{aligned}
V := \;& a \leftarrow 0.1 \\
& \text{for } i \in 1..49 \\
& \quad \text{for } j \in 1..40 \\
& \quad\quad V_{0,i} \leftarrow 0.02\sin\left(\frac{\pi \cdot i}{30}\right) \\
& \quad\quad V_{j,0} \leftarrow 0 \\
& \quad\quad V_{j,50} \leftarrow 0 \\
& \quad\quad V_{j+1,i} \leftarrow a^2 \cdot \left(V_{j,i+1} + V_{j,i-1}\right) + 2 \cdot \left(1 - a^2\right) \cdot V_{j,i} - V_{j-1,i} \\
& V
\end{aligned}
$$



V

Рис. 14.41. Поверхность решения волнового уравнения

Так же успешно справляется с решением гиперболических и параболических уравнений и встроенная функция numol.

Функция numol (x_endpts, xpts, t_endpts, tpts, num_pde, num_pae, pde_func, pinit, bc_func) возвращает матрицу размерности xpts · tpts, которая содержит решения одномерного дифференциального уравнения в узловых точках сетки. В каждом столбце t_i представлены значения функции по одной пространственной координате для данного временного слоя t_i. В случае систем уравнений размерность матрицы увеличивается: xpts · (tpts·n), где n — общее количество уравнений. Решения для каждого уравнения системы по всем временным слоям располагаются соответственно слева направо. Заметьте, система может содержать не только дифференциальные уравнения в частных производных, но и алгебраические.

Функция numol содержит большое количество параметров, поэтому к их заданию следует подходить особенно внимательно:

- x_endpts, t_endpts — векторы, задающие интервал поиска решения по пространственной и временной координатам. Содержат два элемента, представляющих собой левую и правую границу промежутка;
- xpts, tpts — количество точек дискретизации расчетной области (x,t) (по координате x и t соответственно), в которых ищутся значения сеточных функций;
- num_pde, num_pae — количество дифференциальных и алгебраических уравнений в системе. Если решается одно дифференциальное уравнение, первый параметр задается как 1, второй — 0;
- pde_func — вектор функции переменных x, t, u, u_x и u_{xx}, количество элементов которого равно общему количеству уравнений в системе. Он должен содержать правые части уравнений в таком виде, чтобы в левых частях находились только первые производные функции по времени u_t. Для определения производных используется литеральный индекс «.», очередность функций системы обозначается математическим индексом «[»;
- pinit — вектор-функция переменной x размерности (n·1) (где n — общее количество уравнений), задающая начальные условия для каждой функции в системе;
- bc_func — матрица граничных условий размерности (num_pde·3). Задавая параметр bc_func, обратите внимание на следующие особенности:

  1) если уравнение, записанное в виде pde_func, содержит в правой части вторую производную, то в двух первых маркерах строки, соответствующей данному уравнению, указывается левое и правое граничное условие. Если они заданы в форме Дирихле, то в третий маркер необходимо ввести знак «D», для граничных условий в форме Неймана третий элемент строки записывается как «N»;

  2) если в правой части уравнения, записанного в форме pde_func, содержится только первая производная, то условие указывается только на одной границе интервала. В таком случае в соседний маркер вводится символ «NA», а в третий — «D»;

  3) если уравнение не содержит производных в правой части, то соответствующая ему строка матрицы bc_func должна выглядеть следующим образом: («NA» «NA» «D»).

Другими словами, количество граничных условий для каждого уравнения системы должно быть равно порядку производной по пространственной переменной.

Воспользуемся функцией numol для решения волнового уравнения. Обратите внимание на то, что в примере оно представлено в виде системы.

Пример 14.31. Решение волнового уравнения с помощью встроенной функции numol (рис. 14.42)

$$c := 5 \qquad L := \pi \qquad T := \pi$$

$$x\_endpts := \begin{pmatrix} 0 \\ L \end{pmatrix} \qquad t\_endpts := \begin{pmatrix} 0 \\ T \end{pmatrix}$$

$$xpts := 40 \qquad tpts := 30$$

$$num\_pde := 2 \qquad num\_pae := 0$$

$$pde\_func(x, t, u, u_x, u_{xx}) := \begin{pmatrix} u_1 \\ c^2 \cdot u_{xx_0} \end{pmatrix}$$

$$pinit(x) := \begin{pmatrix} 0 \\ \sin\left(\frac{\pi \cdot x}{L}\right) \end{pmatrix}$$

$$bc\_func(t) := \begin{pmatrix} "NA" & "NA" & "D" \\ pinit(0)_0 & pinit(L)_0 & "D" \end{pmatrix}$$

$$u := numol(x\_endpts, xpts, t\_endpts, tpts, num\_pde, num\_pae, pde\_func, pinit, bc\_func)$$

$$i := 0 .. xpts$$

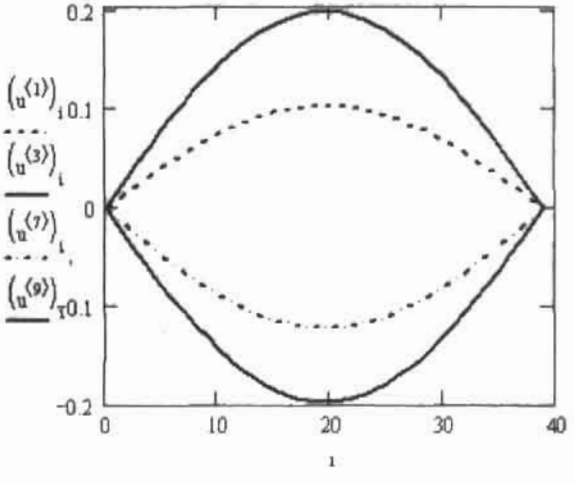


Рис. 14.42. Решение волнового уравнения на разных временных слоях

14.4.3. Эллиптические уравнения

Для дифференциальных уравнений в частных производных эллиптического типа в Mathcad имеются средства решения лишь одного из них – уравнения Пуассона, имеющего следующий вид:

$$\frac{d^2}{dx^2}T(x,y) + \frac{d^2}{dy^2}T(x,y) = -f(x,y)$$

Чаще всего уравнение Пуассона (в приведенной интерпретации) используют для описания некоторого стационарного процесса, например распространения температуры на плоскости вокруг источника (подчеркнем: здесь имеется в виду стационарный процесс распространения тепла; общее же уравнение теплопроводности записывается иначе). Впрочем, оно применимо и к целому ряду других физических явлений (например, в электростатике его используют для описания напряженности электрического поля в двумерной области).

В Mathcad решение уравнения Пуассона ищется в узлах сетки размерности (M+1)×(M+1) (где $M=2^n$) точек, в углах которой должны быть заданы соответствующие граничные условия. В том случае, если тепловой поток полностью рассеивается средой, то они должны быть определены как нули. Для описания этой наиболее простой ситуации в Mathcad существует специальная функция multigrid(F,ncycle), где:

- F – матрица размерности (M+1)×(M+1) ($M=2^n$), определяющая правую часть уравнения Пуассона. В случае описания стационарных процессов теплопроводности в ней вы должны определить положение и силу источников (или поглотителей) тепла. В крайних элементах матрицы F следует задать соответствующие начальные условия (необходимость этого определения связана с тем, что таким образом вы сообщаете системе размерность сетки решения);
- ncycle – этот параметр определяет количество циклов, используемых численным алгоритмом на каждую итерацию. В большинстве случаев его достаточно определить как 2, хотя в случае плохой аппроксимации решения его можно задать и большим (естественно, целым) числом.

Приведем пример модели рассеяния тепла от одного источника при нулевых граничных условиях.

Пример 14.32. Решение уравнения Пуассона для случая полного рассеяния тепла

Задаем размерность аппроксимирующей сетки и краевые условия:

$$M := 2^5$$

$$F_{M,M} := 0$$

Задаем координаты источника и его мощность:

$$Xsource := 13 \qquad Ysource := 13 \qquad Power := 123000$$

$$F_{Xsource,Ysource} := Power$$

Решаем уравнение Пуассона с помощью функции multigrid (рис. 14.43):

$$T := multigrid(-F, 2)$$

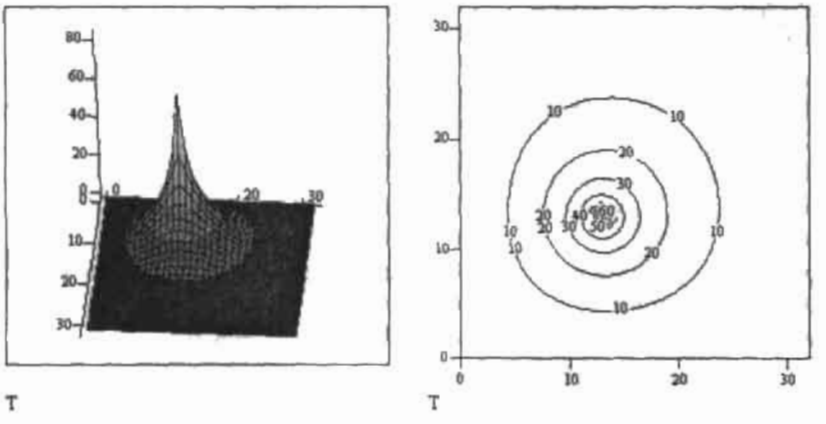


Рис. 14.43. Поверхность и контурный график решения уравнения Пуассона

Обратите внимание на то, что матрица правой части уравнения Пуассона F должна быть задана с обратным знаком, что связано с ее физическим смыслом.

С помощью функции multigrid можно промоделировать ситуацию, когда на одной плоскости располагаются несколько источников и поглотителей тепла. Для этого нужно просто правильно определить соответствующие элементы матрицы F.

Пример 14.33. Решение уравнения Пуассона при наличии нескольких источников и поглотителей тепла (рис. 14.44)

$$R := 2^5$$

$$\tau_{R,R} := 0$$

$$\tau_{\frac{R}{2},\frac{R}{2}} := 100 \qquad \tau_{R-2,\frac{R}{2}} := -165 \qquad \tau_{\frac{3}{4}R,\frac{3}{4}R} := 150 \qquad \tau_{\frac{3}{4}R,\frac{1}{2}R} := 50$$

$$T := \text{multigrid}(-\tau, 2)$$

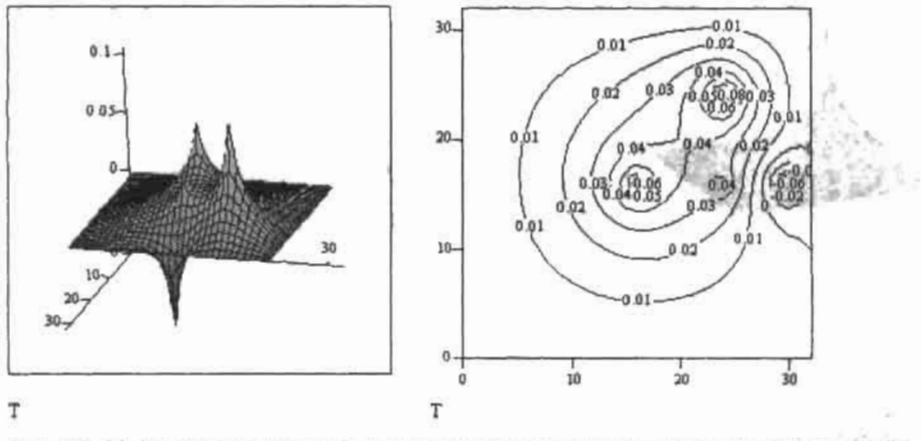


Рис. 14.44. Распределение тепла при наличии нескольких источников и поглотителей

В случаях, когда уравнение Пуассона имеет ненулевые значения краевых условий, найти его решение можно с помощью специальной встроенной функции relax(a,b,c,d,e,F,v,rjac), где:

- a,b,c,d,e — квадратные матрицы одинаковой размерности, содержащие коэффициенты аппроксимирующей дифференциальное уравнение пятиточечной разностной схемы;
- F — квадратная матрица, соответствующая правой части уравнения Пуассона;
- v — квадратная матрица, содержащая граничные условия и начальные приближения для решения;
- rjac — спектральный радиус итераций Якоби. Может принимать значение от 0 до 1 и служит для характеристики скорости сходимости итераций.

Функция relax использует для решения уравнений математической физики довольно распространенный алгоритм релаксации, описание которого приводится ниже. Приведем пример решения уравнения Пуассона с использованием схемы аппроксимации типа «крест».

Пример 14.34. Решение уравнения Пуассона с помощью функции relax (рис. 14.45)

$$M := 2^5$$

$$j := 0..M \qquad i := 0..M$$

$$a_{i,j} := 1$$

$$b := a \qquad c := a \qquad d := a \qquad e := -4a$$

$$F_{M,M} := 0 \qquad F_{23,12} := 20 \qquad F_{13,13} := -13$$

$$v_{i,j} := 0$$

$$Z := relax(a, b, c, d, e, -F, v, 0.5)$$

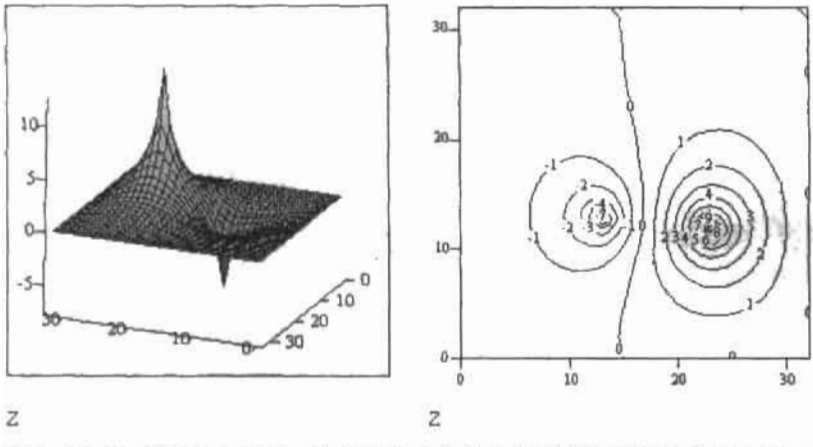


Рис. 14.45. Поверхность и контурный график решения уравнения Пуассона, построенные на основании расчета с помощью функции relax

С помощью функции relax, подобрав верные коэффициенты разностной схемы, можно решать и некоторые другие линейные дифференциальные уравнения в частных производных.

Прежде чем перейти к рассмотрению довольно необычного алгоритма релаксации, реализованного во встроенной функции relax, запишем разностную схему для уравнения Пуассона, описывающего процесс распространения тепла при наличии точечного источника f и поглотителя g.

$$\frac{\partial^2}{\partial x^2}T(x,y) + \frac{\partial^2}{\partial y^2}T(x,y) = -f(x,y) + g(x,y)$$

Оно содержит вторые производные по обеим переменным, каждая из которых аппроксимируется по собственной координате согласно известной формуле:

$$\frac{T_{i+1,j} - 2\cdot T_{i,j} + T_{i-1,j}}{h^2} + \frac{T_{i,j+1} - 2\cdot T_{i,j} + T_{i,j-1}}{h^2} = -f_{i,j} + g_{i,j}$$

После несложных преобразований расчетная формула принимает вид

$$T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 4\cdot T_{i,j} + h^2\cdot\left(f_{i,j} - g_{i,j}\right) = 0 \qquad (*)$$

Обратите внимание на то, что шаг по обеим координатам берется равный, поэтому в простейшем случае при одинаковых промежутках определения i и j расчетная область представляет собой квадрат (что является необходимым условием корректной работы функции relax).

Из приведенного равенства видно, что в точке (i, j) решение ищется как среднее арифметическое между значениями сеточных функций в четырех соседних узлах. Принципиально это возможно, когда заданы граничные условия по всему периметру квадратной области. Разностная схема такого типа получила название «крест». Ее шаблон представлен на рис. 14.46.

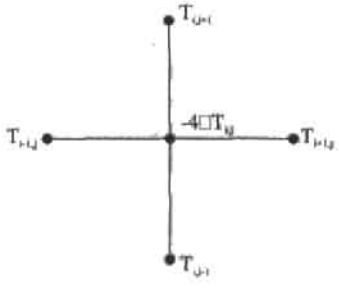


Рис. 14.46. Шаблон разностной схемы «крест» для уравнения Пуассона

Итак, для i=1...N–1 и j=1...N–1 (для всех точек, находящихся внутри расчетной области) мы имеем систему алгебраических уравнений, матрица коэффициентов которой содержит ненулевые элементы на пяти диагоналях: главной, двух верхних и нижних соответственно. При использовании функции relax эти элементы задаются в виде матриц a, b, c, d, e. Естественно, чтобы система была разрешима, нужно дополнить ее граничными условиями.

Пожалуй, основным и довольно существенным недостатком такого подхода является то, что количество уравнений в системе определяется величиной выбранного шага. Стремясь максимально уменьшить его, мы резко увеличим количество точек решетки, для каждой из которых прописывается отдельное уравнение. В результате возникнет необходимость решать систему с разреженной (пятидиагональной) матрицей большой размерности, что, безусловно, приведет к неприемлемым временным затратам. К счастью, из подобной ситуации есть два выхода. Для ускорения решения можно модифицировать для пятидиагональной матрицы алгоритм прогонки, предложенный для матриц с тремя диагоналями в подразд. 14.3.2., однако тогда придется программировать не только систему, но и сам алгоритм. Поэтому с позиции экономии времени и памяти самым оптимальным в данном случае оказывается итерационный метод релаксации. Он требует представления конечно-разностного уравнения в несколько иной форме. Выразим из равенства (*) $T_{i,j}$, а затем в правой части прибавим и вычтем $T_{i,j}$:

$$T_{i,j} = T_{i,j} + \frac{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 4 \cdot T_{i,j} + h^2 \cdot \left(f_{i,j} - g_{i,j}\right)}{4}$$

Согласно этой формуле, для каждого внутреннего узла последовательно высчитывается новое значение исходя из полученного на предыдущей итерации. Предварительно для точек задается начальное приближение с использованием граничных условий. Естественно, что при большом количестве узлов сетки решение сходится крайне медленно, поэтому в методе релаксации для ускорения сходимости вводится дополнительный множитель ω:

$$T_{i,j} = T_{i,j} + \omega \frac{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 4 \cdot T_{i,j} + h^2 \cdot \left(f_{i,j} - g_{i,j}\right)}{4}$$

Параметр ω задается формулой, указанной ниже в программе. Обычно его значения находятся в интервале $1<\omega<2$. Таким образом, оптимизированная формула последовательно слева направо и сверху вниз применяется к узлам сетки на каждой итерации. Вычисление продолжается до тех пор, пока разность между значениями во всех точках, полученными на данной и последующей итерации, не окажется ниже пороговой величины ошибки или вовсе не сойдется к точному. Амплитуда колебаний решения уменьшается, оно постепенно «релаксирует» к действительным значениям, отсюда и название метода.

В данном разделе мы рассмотрели лишь незначительную часть задач, связанных с линейными уравнениями в частных производных с постоянными коэффициентами. Их многообразие в теории уравнений математической физики связано с разнообразием явлений окружающего нас мира. В действительности существует очень малое количество задач, решаемых в явном виде, поскольку большинство физических явлений описывается нелинейными уравнениями в частных производных. Однако большое количество различных постановок задач, связанных с решением уравнений в частных производных, привело к тому, что теория численных методов, разрабатываемых в этой области, постоянно развивается и имеет множество направлений. Например, существуют алгоритмы, позволяющие решать одно- и многомерные уравнения параболического типа с переменными, в частности с нелинейно зависящими от решения коэффициентами.

Список литературы

1. *Гусак А. А.* Задачи и упражнения по высшей математике: в 2 ч. Ч. 1: Для вузов. — 2-е изд. — Мн.: «Вышэйшая школа», 1988. — 247 с.: ил.
2. *Гусак А. А.* Задачи и упражнения по высшей математике: в 2 ч. Ч. 2: Для вузов. — 2-е изд. — Мн.: «Вышэйшая школа», 1988. — 229 с.: ил.
3. *Гусак А. А.* Высшая математика. Учебник для студентов вузов: в 2 т. Т. 1. — 3-е изд. — Мн.: «ТетраСистемс», 2001. — 544 с.
4. *Гусак А. А.* Высшая математика. Учебник для студентов вузов: в 2 т. Т. 2. — 3-е изд. — Мн.: «ТетраСистемс», 2001. — 448 с.
5. *Гусак А. А., Бричкова Е. А.* Теория вероятностей. Справочное пособие к решению задач. — 3-е изд. — Мн.: «ТетраСистемс», 2002. — 288 с.
6. *Гмурман В. Е.* Теория вероятностей и математическая статистика. — 9-е изд. — М.: «Высшая школа», 2003. — 479 с.: ил.
7. *Гмурман В. Е.* Руководство к решению задач по теории вероятности и математической статистике. 4-е изд. — М.: «Высшая школа», 1997. — 400 с.: ил.
8. *Гурский Д. А.* Вычисления в MathCAD. — Мн.: «Новое знание», 2003. — 814 с.: ил.
9. *Гурский Д., Турбина Е.* Mathcad для студентов и школьников. Популярный самоучитель. — СПб.: Питер, 2005. — 400 с.: ил.
10. Сборник задач по математике для поступающих во ВТУЗы / *В. К. Егерев, В. В. Зайцев, Б. А. Кордемский и др.* Под ред. М. И. Сканави. — Мн.: «Вышэйшая школа», 1990. — 528 с.: ил.
11. *Ильин В. А., Позняк Э. Г.* Основы математического анализа. Ч. 1. — М.: «Наука», Гл. ред. физ.-мат. лит., 1971. — 600 с.: ил.
12. *Мацкевич И. П., Свирид Г. П., Булдык Г. М.* Сборник задач и упражнений по высшей математике: Теория вероятностей и математическая статистика. — Мн.: «Вышэйшая школа», 1996. — 318 с.: ил.
13. *Микулик Н. А., Рейзина Г. Н.* Решение технических задач по теории вероятности и математической статистике. — Мн.: «Вышэйшая школа», 1991. — 164 с.: ил.
14. *Мэтьюз Джон Г., Финк Куртис Д.* Численные методы. Использование MATLAB. — 3-е изд. — М.: Издательский дом «Вильямс», 2001. — 720 с.: ил.
15. *Бронштейн И. Н., Семендяев К. А.* Справочник по высшей математике для инженеров и учащихся втузов. — 13-е изд. — М.: «Наука», Гл. ред. физ.-мат. лит., 1986. — 544 с.
16. *Воднев В. Т., Наумович А. Ф., Наумович Н. Ф.* Основные математические формулы. Под ред. Ю. С. Богданова. — Мн.: «Вышэйшая школа», 1980. — 336 с.: ил.
17. Numerical Recipes in C. W. H. Press. *B. P. Flannery, S. A. Teukolsky, W. T. Vetterling.* — New York: Cambridge University Press, 1992.

Приложение. Встроенные функции

Функция	Аргументы	Назначение	Ссылка
a*(z)	z — аргумент	Приставка «а» служит для обозначения соответствующей обратной тригонометрической или гиперболической функции	18.3 18.5
Ai(x)	x — аргумент	Функция Эйри первого рода	18.1.3
Ai.sc(x)	x — аргумент	Нормированная функция Эйри первого рода (нормировочный множитель $\exp(\lvert\mathrm{Re}(2/3\cdot z^{3/2})\rvert)$)	18.1
angle(x,y)	(x,y) — координата точки	Угол между точкой и полуосью OX	18.5
antisymmetric tensor(i,j,k)	(i,j,k) — матричные индексы	Определяет, сколько перестановок нужно осуществить, чтобы преобразовать последовательность (0,1,2) в (i,j,k)	18.7
APPENDPRN («file», [M])	file — строка пути к файлу; M — матрица данных	Дозапись данных в уже существующий текстовый файл	16.7.1
arg(z)	z — комплексное число	Возвращает угол между точкой комплексного числа и действительной осью	5.1
atan2(x,y)	(x,y) — координаты точки	Определяет угол между точкой и осью X	18.5
augment(A,B...)	A,B... — матрицы или векторы	Слияние матриц слева направо	3.3.4
bei(n,x)	n — порядок x — аргумент	Мнимая часть функции Бесселя–Кельвина порядка n	18.1.5
ber(n,x)	n — порядок; x — аргумент	Действительная часть функции Бесселя–Кельвина порядка n	18.1.5
Bi(x)	x — аргумент	Функция Эйри 2-го рода	18.1.3
Bi.sc(x)	x — аргумент	Нормированная функция Эйри 2-го рода (нормировочный множитель $\exp(\lvert\mathrm{Re}(2/3\cdot z^{3/2})\rvert)$)	18.1
bspline(x,y,u,n)	x,y — векторы экспериментальных данных; u — вектор сшивок элементарных сплайнов; n — порядок сплайнов	Возвращает вектор коэффициентов интерполирующего сплайна произвольного порядка с сшивками в заданных пользователем точках	16.1.3

Функция	Аргументы	Назначение	Ссылка
Bulstoer(y0,t0, t1,M,D)	y0 — вектор начальных условий; (t0,t1) — область поиска решения; M — количество шагов интегрирования; D(t,y) — система ОДУ в векторной форме	Возвращает матрицу решений задачи Коши для системы линейных ОДУ по методу Булирша–Штера	14.2.3
bulstoer(y0,t0, t1,acc,D,k,s)	y0 — вектор начальных условий; (t0,t1) — область поиска решения; acc — точность; D(t,y) — система ОДУ в векторной форме; k — максимальное количество шагов; s — минимальный размер шаг	Возвращает решение задачи Коши для системы линейных ОДУ по методу Булирша–Штера, найденное с учетом указанных параметров (используется для определения значения функции решения в конкретной точке)	14.2.3
bvalfit(z1,z2,x0, x1,xf,D,load1, load2,score)	z1, z2 — векторы начальных приближений для недостающих краевых условий; x0, x1 — границы интервала изменения независимых переменных; D(x,y) — система ОДУ в векторной форме; load1(x0, z1), load2(x1,z2) — векторные функции, задающие условия на границах; score(xf,y) — вектор-функция, задающая условие сшивки в xf	Возвращает вектор недостающих краевых условий для системы линейных ОДУ со специальным условием в промежуточной точке	14.3.1
Ceil(x,y)	x,y — аргументы	Наименьшее целое число, большее либо равное x/y, умноженное на y	18.6
ceil(x)	x — аргумент	Наименьшее целое число, большее x	18.6
CFFT(y) cfft(y)	y — вектор значений сигнала	Численное прямое преобразование Фурье (комплексное) в двух различных нормировках	16.6.1

Функция	Аргументы	Назначение	Ссылка
cholesky(A)	A — квадратная матрица	Разложение Холецкого	3.3.8
cnorm(x)	x — аргумент	Нормированная функция нормального распределения (функция Лапласа)	15.3.3
cnper(rate,pv,fv)	rate — фиксированная учетная ставка; pv — текущий размер вклада; fv — необходимый размер вклада	Определяет количество периодов, требуемых для получения некоторого значения вклада, если известен его текущий размер и процент начислений	18.11
cols(A)	A — матрица или вектор	Определяет количество столбцов в матрице	3.3.2
combin(n,k)	n, k — количество элементов множеств	Число сочетаний из n по k	15.1
concat(S1,S2...)	S1, S2... — строки	Слияние нескольких строк в одну	18.10
cond1(A) cond2(A) conde(A) condi(A)	A — квадратная матрица	Числа обусловленности в нормах L1, L2, евклидовой и бесконечной	8.2.1
corr(A,B)	A, B — соразмерные векторы или матрицы	Вычисляет значения коэффициента корреляции двух выборок	16.4
cos(z)	z — аргумент	Косинус	185
cosh(z)	z — аргумент	Гиперболический косинус	18.3
cot(z)	z — аргумент	Котангенс	18.5
coth(z)	z — аргумент	Гиперболический котангенс	18.3
crate(nper, pv, fv)	nper — количество периодов. Должно быть целым положительным числом; pv — текущий размер вклада; fv — будущий размер вклада	Вычисляет фиксированную учетную ставку на период, если известен текущий размер вклада, количество периодов, требуемое для получения определенной прибыли, и будущая величина вклада	—
CreateMesh(F, s0,s1,t0,t1,sgrid, tgrid, fmap)	F(s,t) — вектор-функция из трех элементов; (s0,s1) — интервал по переменной s; (t0,t1) — интервал по переменной t; sgrid, tgrid — количество линий сетки по каждой переменной; fmap — функция, описывающая систему координат	Создает вложенный массив, содержащий значения координат по x,y,z параметрически заданной поверхности. Используется для построения трехмерных графиков	6.2.1

Функция	Аргументы	Назначение	Ссылка
CreateSpace(F, t0,t1,tgrid,fmap)	F(t) — вектор-функция из трех элементов; (t0,t1) — интервал изменения переменной; tgrid — количество точек пространственной кривой; fmap — функция, описывающая систему координат	Создает вложенный массив, содержащий значения координат по x,y,z параметрически заданной кривой	6.2.1
csc(z)	z — аргумент	Косеканс	18.5
csch(z)	z — аргумент	Гиперболический косеканс	18.3
csgn(z)	z — аргумент	Логическая функция комплексного числа	7.6
csort(A,i)	A — матрица; i — индекс столбца	Выполняет сортировку по возрастанию элементов i-го столбца некоторой матрицы A	3.3.3
cspline(Mx,My)	Mx, My — векторы или матрицы приближаемых данных	Вектор коэффициентов кубического сплайна со сшивками в экспериментальных точках с кубическим продолжением	16.1.2 16.1.4
cumint(rate, nper, pv, start, end, [type])/ cumprn(rate, nper, pv, start, end, [type])	rate — фиксированный процент по заему; nper — количество составных периодов; pv — текущее значение заема (положительная величина); start — начальный период накопления; end — конечный период накопления (положительное число, не меньшее start); type — принимает значение 0 для платежа в конце периода или 1 — в начале. По умолчанию type=0	Определяет совокупный процент/сумму по заему между начальным и конечным периодом при фиксированной процентной ставке, известном общем числе составных периодов и текущей стоимостью ссуды	—
cvar(A,B)	A, B — соразмерные векторы или матрицы	Определяет коэффициент ковариации двух выборок	16.4
cyl2xyz(r,q,f)	r, q, f — координаты точки в цилиндрической системе координат	Производит пересчет значений координат точки из цилиндрической системы координат в декартову	—

Функция	Аргументы	Назначение	Ссылка
dbeta(x,s1,s2)	x — переменная; s1, s2 — положительные параметры	Задает плотность вероятности бета-распределения	15.3.3
dbinom(k,n,p)	k, n — целые положительные параметры; p — вероятность однократного наступления события	Задает вероятность наступления некоторого события k раз в n испытаниях исходя из биноминального распределения	15.3.2
dcaushy(x,l,s)	x — переменная; l — параметр разложения; s — положительный параметр масштаба	Задает плотность вероятности распределения Коши	15.3.3
dchisq(x,d)	x — переменная; d — число степеней свободы	Задает плотность вероятности распределения «хи-квадрат»	15.3.3
dexp(x,r)	x — переменная; r — положительный параметр	Задает плотность вероятности экспоненциального распределения	15.3.3
dF(x,d1,d2)	x — переменная; d1, d2 — числа степеней свободы	Задает плотность вероятности распределения Фишера	15.3.3
dgamma(x,s)	x — переменная; s — положительный параметр формы	Задает плотность вероятности гамма-распределения	15.3.3
dgeom(k,p)	k — количество испытаний; p — вероятность успеха в одном испытании	Вероятность наступления события на k-м испытании исходя из геометрического распределения	15.3.2
dhypergeom(m,a,b,n)	m, a, b, n — целые положительные параметры	Вероятность события исходя из гипергеометрического распределения	15.3.2
diag(v)	v — вектор	Возвращает диагональную матрицу с элементами v на главной диагонали	3.3.1
dlnorm(x,mu,sigma)	x — переменная; mu — натуральный логарифм математического ожидания; sigma — натуральный логарифм среднеквадратичного отклонения	Задает плотность вероятности логнормального распределения	15.3.3

Функция	Аргументы	Назначение	Ссылка
dlogis(x,l,s)	x — переменная; l — математическое ожидание; s — положительный параметр масштаба	Задает плотность вероятности логистического распределения	15.3.3
dnbinom(k,n,p)	k, n — целые положительные параметры; p — вероятность однократного наступления отрицательного события	Вероятность события исходя из отрицательного биноминального распределения	15.3.2
dnorm(x,mu, sigma)	x — переменная; mu — математическое ожидание; sigma — среднеквадратичное отклонение	Задает плотность вероятности нормального распределения	15.3.3
dpois(k,i)	k — количество испытаний; i — положительный параметр	Вероятность наступления события исходя из распределения Пуассона	15.3.2
dt(x,d)	x — переменная; d — число степеней свободы	Вычисляет плотность вероятности распределения Стьюдента	15.3.3
dunif(x,a,b)	x — переменная; (a, b) — границы интервала	Определяет плотность равномерного распределения	15.3.3
dweibull(x,s)	x — переменная; s — положительный параметр формы	Функция плотности распределения Вейбулла	15.3.3
eff(rate,nper)	rate — номинальная процентная ставка, может быть задана как дробью, так и числом с символом процента; nper — общее количество составных периодов в год	Определяет эффективную годовую учетную ставку при известной номинальной процентной ставке и числе составных периодов в год	—
eigenvals(M)	M — квадратная матрица	Определяет собственные значения матрицы	3.3.7
eigenvec(M,z)	M — квадратная матрица; z — собственное значение	Собственный вектор матрицы, вычисленный исходя из заданного собственного значения	3.3.7

Функция	Аргументы	Назначение	Ссылка
eigenvecs(M)	M — квадратная матрица	Вычисляет собственные векторы матрицы	3.3.7
erf(x)	x — переменная	Функция ошибки	15.3.3
erfc(x)	x — переменная	Обратная функция ошибки	15.3.3
error(S)	S — строка	Возвращает строку S в виде сообщения об ошибке	18.10
exp(z)	z — переменная	Вычисляет значение экспоненты от z	18.4
expfit(vx,vy,vg)	vx, vy — векторы данных; vg — вектор начальных приближений для искомых параметров a, b, c	Вычисляет коэффициенты экспоненциальной регрессии ($a \cdot e^{bx}+c$)	16.3.1 16.3.5
FFT(y) ffi(y)	y — вектор значений сигнала	Вектор прямого преобразования Фурье (действительных данных) в двух различных нормировках	16.6.1
fhyper(a,b,c,x)	a, b, c — параметры; x — аргумент (по абсолютной величине меньший 1)	Гипергеометрическая функция Гаусса	18.9
find(Var1,Var2...)	Var1, Var2... — неизвестные	Служит для численного и символьного решения алгебраических уравнений и систем алгебраических уравнений, заданных в блоке Given. Может быть использована для решения матричных уравнений	8.2
Floor(x,y)	x, y — аргументы	Возвращает целую часть x, умноженную на y	18.6
floor(x)	x — аргумент	Наибольшее целое число, меньшее либо равное x	18.6
fv(rate, nper, pmt, [[pv],[type]])	rate — фиксированная процентная ставка за период; nper — количество составных периодов за год; pv — текущее значение заема; type — принимает значение 0 для платежа в конце периода или 1 — в начале. По умолчанию type=0	Вычисляет будущее значение вклада или кредита через определенное количество составных периодов при постоянных платежах и фиксированной процентной ставке	—

Функция	Аргументы	Назначение	Ссылка
fvadj(prin,v)	prin — ежегодная общая сумма; v — вектор процентных ставок, каждая из которых применяется с той же самой основной суммой и процентами с нее за период времени	Определяет будущую стоимость вклада после применения ряда ставок сложного процента	—
fvc(rate,v)	rate — фиксированная учетная ставка; v — вектор регулярных денежных потоков	Определяет будущую стоимость ряда потоков наличности, встречающихся равномерно при фиксированной процентной ставке	—
Gamma(a,z)	z — аргумент; a — положительный действительный параметр	Неполная гамма-функция Эйлера. Для подсчета простой гамма-функции используйте эту встроенную функцию, не определяя параметр a	18.9
gcd(A,B...)	A, B — векторы или матрицы с целыми положительными элементами	Определяет наибольший общий делитель для элементов матрицы или вектора A	15.4.5
genfit(vx,vy,vg,F)	vx, vy — векторы экспериментальных данных; vg — начальные приближения для искомых параметров регрессии; F — вектор-функция, на основании которой строится регрессия	Функция вычисления регрессии произвольного вида	16.3.5
geninv(A)	A — квадратная матрица	Возвращает обобщенную обратную матрицу	3.2.9
genvals(A,B)	A, B — квадратные соразмерные матрицы	Расчет обобщенных собственных значений	3.3.7
genvecs(A,B)	A, B — квадратные соразмерные матрицы	Расчет обобщенных собственных векторов	3.3.7
GETWAVINFO (file)	file — строка пути к звуковому файлу	Возвращает вектор с основной информацией о звуковом файле	—
gmean(x)	x — вектор или матрица с экспериментальной выборкой	Вычисляет геометрическое среднее выборки	15.4.6

Функция	Аргументы	Назначение	Ссылка
H1(m,x)	x — аргумент; m — порядок	Функция Ганкеля 1-го рода порядка m	18.1.4
H1.sc(m,x)	x — аргумент; m — порядок	Нормированная функция Ганкеля 1-го рода порядка m (нормировочный множитель exp((3 – 2m)·z·i))	18.1
H2(m,x)	x — аргумент; m — порядок	Функция Ганкеля 2-го рода порядка m	18.1.4
H2.sc(m,x)	x — аргумент; m — порядок	Нормированная функция Ганкеля 2-го рода порядка m (нормировочный множитель exp(–(3 – 2m)·z·i))	18.1
heaviside step(x)	x — действительное число	Функция «ступеньки» Хевисайда	18.7
Her(n,x)	x — аргумент; n — порядок	Полином Эрмита произвольного порядка	18.9
hist(intvls, A)	intvls — вектор границ сегментов гистограммы; A — матрица или вектор экспериментальной выборки	Используется для построения статистической гистограммы с произвольной шириной столбца	—
histogram(n,data)	n — количество столбцов гистограммы; data — вектор данных	Используется для построения гистограммы с фиксированной шириной столбца	15.7
hlookup(z,A,r)	z — номер столбца; A — матрица; r — вектор	Выборка элементов из матрицы	3.3.4
hmean(x)	x — вектор или матрица с экспериментальной выборкой	Вычисляет гармоническое среднее выборки	15.4.6
I0(x)	x — аргумент	Модифицированная функция Бесселя 1-го рода нулевого порядка	18.1.2
I0.sc(x)	x — аргумент	Нормированная модифицированная функция Бесселя 1-го рода нулевого порядка (нормировочный множитель exp(–|Re(z)|))	18.1
I1(x)	x — аргумент	Модифицированная функция Бесселя 1-го рода 1-го порядка	18.1.2

Функция	Аргументы	Назначение	Ссылка
II.sc(x)	x — аргумент	Нормированная модифицированная функция Бесселя 2-го рода нулевого порядка (нормировочный множитель exp(–|Re(z)|))	18.1
ibeta(a,x,y)	a — параметр; x, y — аргументы	Неполная бета-функция	18.9
ICFFT(v) icfft(v)	v — вектор Фурье-спектра	Обратное численное преобразование Фурье для комплексных данных в разных нормировках	16.6.1
identity(n)	n — размерность матрицы	Создает единичную матрицу размерности n	3.3.1
if(cond,x,y)	cond — некоторое логическое условие; x, y — величины, возвращаемые, если условие верно (ложно)	Функция условия	18.7
IFFT(v) ifft(V)	v — вектор Фурье-спектра	Обратное численное преобразование Фурье для действительных данных в разных нормировках	16.6.1
Im(z)	z — комплексное число (вектор или матрица с комплексными числами)	Мнимая часть комплексного числа (или соответствующие векторы и матрицы)	5.1
In(m,x)	m — порядок; x — аргумент	Модифицированная функция Бесселя произвольной степени	18.1.2
In.sc(m,x)	m — порядок; x — аргумент	Нормированная модифицированная функция Бесселя произвольной степени (нормировочный множитель exp(–|Re(z)|))	18.1
intercept(x,y)	x, y — векторы данных	Определяет свободный член линейной регрессии	16.3.1
interp(vs,Mx, My,x)	s — вектор вторых производных; Mx, My — векторы и матрицы экспериментальных данных; x — переменная	Задает интерполирующий сплайн на основании рассчитанного с помощью одной из специальных функций вектора вторых производных	16.1.2

Функция	Аргументы	Назначение	Ссылка
ipmt(rate, per, nper, pv, [[fv],[type]])	rate — фиксированная процентная ставка за период; per — период времени, для которого нужно найти процент (целое положительное число); nper — количество составных периодов за год; pv — текущее значение заема или вклада; fv — будущее значение заема или вклада. По умолчанию fv=0; type — принимает значение 0 для платежа в конце периода или 1 — в начале. По умолчанию type=0	Вычисляет процентный платеж по вкладу или заему за данный период, основанный на периодических и постоянных платежах с фиксированной учетной ставкой и точно установленной будущей стоимостью	—
irr(v,[guess])	v — вектор потоков наличности, образованный не менее чем двумя элементами, один из которых должен быть положительным, а другой — отрицательным числом; guess — необязательный параметр, приближение к ответу. По умолчанию равен 0.1 (10 %)	Определяет внутреннюю ставку возврата для ряда денежных потоков, распределенных равномерно	—
IsArray(x)	x — аргумент	Возвращает 1, если x — матрица или вектор, и 0 — во всех остальных случаях	18.2
IsNaN(x)	x — аргумент	Возвращает 1, если импортированные из внешнего файла данные содержат элементы, не интерпретируемые Mathcad как числа или строки, и 0 в противном случае	18.2
IsScalar(x)	x — аргумент	Возвращает 1, если x — комплексный или действительный скаляр, и 0 — во всех остальных случаях	18.2

Функция	Аргументы	Назначение	Ссылка
IsString(x)	x — аргумент	Возвращает 1, если x — строка, и 0 — во всех остальных случаях	18.2
ivawe(v)	v — вектор данных волнового спектра	Производит обратное волновое преобразование на основании фильтра Даубечи	16.6.2
J0(x)	x — аргумент	Функция Бесселя нулевого порядка	18.1.1
J0.sc(x)	x — аргумент	Нормированная функция Бесселя нулевого порядка (нормировочный множитель exp(–|Im(z)|))	18.1
J1(x)	x — аргумент	Функция Бесселя 1-го порядка	18.1.1
J1.sc(x)	x — аргумент	Нормированная функция Бесселя 1-го порядка (нормировочный множитель exp(–|Im(z)|))	18.1
Jac(n,a,b,x)	n — порядок полинома; a, b — параметры; x — аргумент	Полином Якоби	18.9
Jn(m,x)	m — порядок; x — аргумент	Функция Бесселя произвольного порядка	18.1.1
Jn.sc(m,x)	m — порядок; x — аргумент	Нормированная функция Бесселя произвольного порядка (нормировочный множитель exp(–|Im(z)|))	18.1
js(n,x)	n — порядок; x — аргумент	Сферическая функция Бесселя 1-го рода	18.1.6
K0(x)	x — аргумент	Модифицированная функция Бесселя 2-го рода нулевого порядка	18.1.2
K0.sc(x)	x — аргумент	Нормированная модифицированная функция Бесселя 2-го рода нулевого порядка (нормировочный множитель exp(z))	18.1
K1(x)	x — аргумент	Модифицированная функция Бесселя 2-го рода 1-го порядка	18.1.2
K1.sc(x)	x — аргумент	Нормированная модифицированная функция Бесселя 2-го рода 1-го порядка (нормировочный множитель exp(z))	18.1

Функция	Аргументы	Назначение	Ссылка
Kn(m,x)	m — порядок; x — аргумент	Модифицированная функция Бесселя 2-го рода произвольного порядка	18.1.2
Kn.sc(m,x)	m — порядок; x — аргумент	Нормированная модифицированная функция Бесселя 2-го рода произвольного порядка (нормировочный множитель exp(z))	18.1
kronecker(M,N)	M, N — квадратные матрицы	Произведение Кронекера двух квадратных матриц	18.8
Kronecker delta(x,y)	x, y — аргументы	Дельта-символ Кронекера. Возвращает 1, если x=y, и 0 во всех остальных случаях	18.7
ksmooth(vx,vy,b)	vx, vy — векторы данных b — ширина окна сглаживания	Сглаживание сигнала на основании функции Гаусса	16.5
kurt(A)	A — вектор или матрица экспериментальной выборки	Подсчитывает эксцесс выборки	15.5.2
Lag(n,x)	n — порядок; x — аргумент	Полином Лаггера	18.9
last(v)	v — вектор	Возвращает индекс последнего элемента вектора v	3.3.2
lcm(A)	A — матрица или вектор с целыми положительными элементами	Возвращает наименьшее общее кратное для элементов (целочисленных) вектора или матрицы A	15.4.5
Leg(n,x)	n — порядок; x — аргумент	Полином Лежандра	18.9
length(v)	v — вектор	Возвращает порядковый номер последнего элемента вектора v	3.3.2
lgsfit(x,y,g)	x, y — векторы данных g — вектор приближений для искомых параметров	Вычисляет коэффициенты регрессии логистической функцией $(a/(1+b\cdot e^{-cx}))$	16.3.5
line(vx,vy)	vx, vy — векторы данных	Возвращает вектор коэффициентов линейной регрессии	16.3.1
linfit(x,y,F)	x, y — векторы данных; F(x) — произвольная вектор-функция, на основании которой проводится регрессия	Регрессия линейной комбинацией функций пользователя	16.3.4

Функция	Аргументы	Назначение	Ссылка
linterp(vx,vy,x)	vx, vy — векторы данных; x — переменная	Линейная интерполяция экспериментальной выборки	16.1.1
ln(z)	z — аргумент	Натуральный логарифм	18.4
ln0(z)	z — аргумент	Аналогична ln(x), но для x=0 возвращает число, равное машинной бесконечности	18.4
lnGamma(z)	z — аргумент	Натуральный логарифм значения гамма-функции, вычисленного для заданного x	18.4
lnfit(x,y)	x, y — векторы данных	Регрессия логарифмической функцией (a·ln(x)+b)	16.3.1
LoadColormap (file)	file — строка пути к файлу (или имя палитры, если она располагается в специальной папке)	Читает цветовую палитру трехмерных графиков в файл	6.2.2
loess(Mx,My, span)	Mx, My — матрицы или векторы параметров выборки; span — параметр длины элементарного полинома	Вычисляет вектор коэффициентов для регрессии отрезками полиномов (используется совместно с interp)	16.3.2
log(z,[b])	z — аргумент; [b] — основание	Логарифм z по основанию b. Если b опущено, то основание принимается равным 10 (десятичный логарифм)	18.4
logfit(vx,vy,vb)	vx, vy — векторы данных; vb — вектор начальных приближений для искомых неизвестных параметров	Регрессия логарифмической функцией с тремя неизвестными параметрами (a·ln(x+b)+c)	16.3.5
lookup(z,A,B)	A, B — соразмерные матрицы; z — аргумент	Возвращает элемент, располагающийся в матрице B на той же позиции, что занимает z в A	3.3.4
lsolve(A,b)	A — матрица системы линейных алгебраических уравнений; b — вектор правых частей	Функция решения систем линейных алгебраических уравнений	8.2.1
lspline(x,y)	x, y — векторы данных	Функция линейной интерполяции (используется совместно с interp)	16.1.2 16.1.4

Функция	Аргументы	Назначение	Ссылка
lu(M)	M — квадратная матрица	Производит LU-разложение квадратной матрицы	3.3.8
match(z,A)	z — скаляр, матрица или строка	Возвращает индексы элемента матрицы A, равного z	3.3.4
matrix(M,N,f)	M — количество строк; N — количество столбцов; f — функция 2 переменных	Возвращает матрицу, элементами которой являются значения f(i,j) (i,j — индексы соответствующего элемента). Используется для построения поверхностей	6.2.1
max(A)	A — вектор или матрица	Определяет наибольший элемент матрицы или вектора A	3.3.9 15.4.4
maximize(f, var1,var2...)	f — функция; var1, var2... — переменные	Функция численного определения локального максимума. Требует задания начальных приближений. При необходимости может быть использована с группой дополнительных условий (заданных в вычислительном блоке Given)	13.3.1 13.3.2
mean(A)	A — матрица или вектор выборки	Рассчитывает среднее выборочное	15.4.1
medfit(vx,vy)	vx, vy — вектора данных	Определяет коэффициенты линейной регрессии исходя из медиан-медианного алгоритма	16.3.1
median(A)	A — матрица или вектор выборки	Рассчитывает точку медианы выборки	15.4.3
medsmooth(vy,n)	vy — вектор значений сигнала; n — количество окон сглаживания	Сглаживание сигнала на основании метода бегущих медиан	16.5
mhyper(a,b,x)	x — аргумент; a, b — параметры	Конфлюэнтная гипергеометрическая функция	18.9
min(A)	A — вектор или матрица	Определяет наибольший элемент матрицы или вектора A	3.3.9 15.4.4
minerr(Var1, Var2,...)	Var1, Var2,... — переменные	Минимизирует невязку несовместных систем уравнений и неравенств. Используется для их приближенного решения. Условия и приближения должны быть заданы в блоке Given	8.2.4

Функция	Аргументы	Назначение	Ссылка
minimize(f, Var1, Var2,...))	f — функция; Var1, Var2,... — переменные	Функция численного определения локального минимума. Требует задания начальных приближений. При необходимости может быть использована с группой дополнительных условий (заданных в вычислительном блоке Given)	13.3.1 13.3.2
mirr(v,fin_rate, rein_rate)	v — вектор потоков наличности, образованный не менее чем двумя элементами, один из которых должен быть положительным, а другой — отрицательным числом; fin_rate — финансовая ставка, подлежащая оплате на заимствованных потоках наличности; rein_rate — реинвестиционная ставка, заработанная на потоках наличности	Соответствует модифицированной процентной ставке возврата для серии денежных потоков с регулярными интервалами при условии, что ставка финансирования подлежит оплате в соответствии с суммой заимствования, а ставка реинвестирования приносит доход с суммы, которую повторно инвестируете	—
mod(x,y)	x, y — действительные числа	Остаток от деления x на y. Имеет тот же знак, что и x	—
mode(A)	A — матрица или вектор выборки	Определяет моду выборки — элемент, который встречается чаще других	15.4.3
multigrid(F, ncycle)	F — матрица правой части уравнения Пуассона; ncycle — параметр точности	Возвращает матрицу решения уравнения Пуассона на квадратной области с нулевыми граничными условиями	14.4.3
nom(rate,nper)	rate — фиксированная учетная ставка; nper — общее количество составных периодов за год	Определяет номинальную учетную ставку	—
norm1(M)	M — квадратная матрица	Определяет норму матрицы в пространстве L1	3.3.5
norm2(M)	M — квадратная матрица	Определяет норму матрицы в пространстве L2	3.3.5

Функция	Аргументы	Назначение	Ссылка
norme(M)	M — квадратная матрица	Определяет евклидову норму матрицы	3.3.5
normi(M)	M — квадратная матрица	Определяет ∞-норму матрицы	3.3.5
nper(rate,pmt,pv, [[fv],[type]])	rate — фиксированная процентная ставка; pmt — платеж, осуществляемый каждый период; pv — текущее значение заема или вклада; fv — будущее значение заема или вклада; type — принимает значение 0 для платежа в конце периода или 1 — в начале. По умолчанию type=0	Определяет количество периодов для капиталовложения или ссуды, основанной на периодических, постоянных платежах с фиксированной учетной ставкой и точно установленной текущей стоимостью	—
npv(rate,v)	rate — фиксированная процентная ставка; v — вектор денежных потоков	Находит значение вклада, если известны скидки и регулярные денежные потоки и оплата проводится в конце периода	—
num2str(z)	z — безразмерный скаляр	Переводит число из цифрового формата в строковый	18.10
numol (x_endpts, xpts, t_endpts, tpts, num_pde, num_pae, pde_func, pinit, bc_func)	x_endpts, t_endpts — векторы, задающие интервал поиска решения по пространственной и временной координатам; xpts, tpts — количество точек дискретизации расчетной области (x,t); num_pde, num_pae — количество дифференциальных и алгебраических уравнений в системе; pde_func — вектор функции переменных x, t, u, u_x и u_{xx}; pinit — вектор-функция переменной x; bc_func — матрица граничных условий	Возвращает матрицу размерности xpts × tpts, которая содержит решения в узловых точках сетки одномерного дифференциального уравнения в частных производных гиперболического или параболического типа	14.4.2

Функция	Аргументы	Назначение	Ссылка
odesolve(t,b, [step])	t — переменная (или вектор переменных), по которой проводится интегрирование уравнения; b — конечная точка области решения; step — величина шага (или количество шагов) численного метода. Необязательный параметр	Используется для решения обыкновенного дифференциального уравнения (по методу Рунге–Кутта), заданного как в форме задачи Коши, так и в виде краевой задачи. Порядок уравнения может быть любым. Начальные условия и само дифференциальное уравнение должны быть определены в блоке Given	14.2.1 14.2.3
p*(x,par)	x — случайное значение; par — список параметров конкретного теоретического распределения	Группа функций вероятности встроенных теоретических распределений (перечисление соответствующих им корней * было дано выше при описании функций плотности распределения d*)	15.3
Pdesolve (u, x, xrange, t, trange, [xpts], [tpts])	u — искомая функция или вектор функций; x — пространственная координата; xrange — вектор, задающий интервал поиска по пространственной координате; t — временная координата; trange — вектор, задающий интервал поиска по времени; xpts — количество точек дискретизации по пространственной координате; tpts — количество точек дискретизации по времени	Возвращает скалярную или векторную функцию переменных x и t, представляющую собой решение уравнения или системы уравнений в частных производных гиперболического или параболического типа	14.4.1
permut(n,k)	n, k — количество элементов в множествах	Число распределений из n по k	15.1

Функция	Аргументы	Назначение	Ссылка
pmt(rate,pmt,pv, [[fv],[type]])	rate — фиксированная процентная ставка; pmt — платеж, осуществляемый каждый период; pv — текущее значение заема или вклада; fv — будущее значение заема или вклада; type — принимает значение 0 для платежа в конце периода или 1 — в начале. По умолчанию type=0	Соответствует платежу по вкладу или заему, основанному на периодичности, постоянных платежах через данное количество составных периодов, использующих фиксированную процентную ставку и особое будущее значение	—
pol2xy(r,q)	r, q — полярные координаты	Преобразование полярных координат в декартовы	—
Polyhedron(S)	S — строка с именем, кодом или Withoff-символом многогранника	Служит для построения пространственных многогранников	6.2.4
PolyLookup(S)	S — строка с именем, кодом или Withoff-символом многогранника	Функция возвращает вектор, содержащий имя, код и Withoff-символ многогранника	6.2.4
polyroots(v)	v — вектор коэффициентов полинома	Возвращает вектор решений алгебраического полинома (как действительных, так и комплексных)	8.1.3
ppmt((rate,per,pmt, pv,[[fv],[type]]))	rate — фиксированная процентная ставка; per — период; pmt — платеж, осуществляемый каждый период; pv — текущее значение заема или вклада; fv — будущее значение заема или вклада; type — принимает значение 0 для платежа в конце периода или 1 — в начале. По умолчанию type=0	Определяет оплату по вкладу или заему для некоторого промежутка времени, основанную на периодических, постоянных платежах над заданными значениями составных периодов с фиксированной учетной ставкой и точно установленной будущей стоимостью	—

Функция	Аргументы	Назначение	Ссылка
predict(v,m,n)	v — вектор равномерно распределенных значений; m — количество элементов v, на основании которых проводится экстраполяция; n — количество элементов в векторе предсказания	Функция, строящая кривую предсказания на основании имеющейся выборки (реализует метод линейной экстраполяции)	16.2
Psi(z)	z — скаляр	Пси-функция	18.8
pspline(x,y)	x, y — векторы данных	Возвращает коэффициенты кубического сплайна с параболическим продолжением. Используется совместно с interp	16.1.2 16.1.4
pv(rate,nper,pmt, [[fv],[type]])	rate — фиксированная процентная ставка; per — период; nper — общее количество составных периодов за год; pmt — платеж, осуществляемый каждый период; pv — текущее значение заема или вклада; fv — будущее значение заема или вклада; type — принимает значение 0 для платежа в конце периода или 1 — в начале. По умолчанию type=0	Соответствует текущему значению вклада или заема, основанному на периодичности, постоянных платежах через данное количество составных периодов, использующих фиксированную процентную ставку и особый взнос	—
pwrfit(x,y,g)	x, y — векторы данных; g — начальные приближения для искомых параметров	Производит расчет коэффициентов регрессии степенной функцией ($a \cdot x^b + c$)	16.3.5
q*(v,par)	v — вероятность; par — набор параметров для конкретного теоретического распределения	Группа функций квантилей вероятности встроенных теоретических распределений (перечисление соответствующих им корней слова * было дано выше при описании функций плотности распределения d*)	15.3

Функция	Аргументы	Назначение	Ссылка
qr(A)	A — квадратная матрица	Проводит QR-разложение квадратной матрицы	3.3.8
Radau(y0, t0, t1, M, F)	y0 — вектор начальных условий; t0 — начальная точка для переменной; t1 — правая граница интервала изменения переменной; M — количество шагов, используемое численным методом; F — векторная функция, определяющая систему ОДУ	Возвращает матрицу, в первом столбце которой содержатся значения переменной, а в остальных — соответствующие им величины искомых функций решения жесткой системы ОДУ	14.2.5
radau(y0, t0, t1, acc, F, k, s)	y0 — вектор начальных условий; (t0,t1) — область поиска решения; acc — точность; D(t,y) — жесткая система ОДУ в векторной форме; k — максимальное количество шагов; s — минимальный размер шаг	Возвращает решение жесткой системы ОДУ, найденное с учетом указанных параметров (используется для определения значения функции решения в конкретной точке)	14.2.5
rank(A)	A — матрица	Ранг матрицы	3.3.6
rate(rate,per,pmt,pv, [[fv],[type],[Guess]])	rate — фиксированная процентная ставка; per — период; pmt — платеж, осуществляемый каждый период; pv — текущее значение заема или вклада; fv — будущее значение заема или вклада; type — принимает значение 0 для платежа в конце периода или 1 — в начале. По умолчанию type=0; guess — необязательный параметр, приближение к ответу. По умолчанию равен 0.1 (10 %)	Определяет учетную ставку на время капиталовложения или ссуды для точно установленного периода	—

Функция	Аргументы	Назначение	Ссылка
Re(z)	z — комплексное число	Возвращает действительную часть комплексного числа	5.1
r*(N,par)	N — количество генерируемых случайных чисел; par — набор параметров конкретного распределения	Группа генераторов случайных чисел с заданной статистикой (перечисление соответствующих им корней слова* было дано выше при описании функций плотности распределения d*)	15.3
READ_BLUE (file)	file — строка пути к файлу	Возвращает матрицу синей составляющей изображения исходя из RGB-модели	—
READ_GREEN (file)	file — строка пути к файлу	Возвращает матрицу зеленой составляющей изображения исходя из RGB-модели	—
READ_HLS(file)	file — строка пути к файлу	Создает матрицу, описывающую цветное изображение, исходя из HLS-модели	—
READ_HLS_HUE (file)	file — строка пути к файлу	Возвращает матрицу составляющей оттенка изображения исходя из HLS-модели	—
READ_HLS_ LIGHT(file)	file — строка пути к файлу	Возвращает матрицу составляющей освещенности изображения исходя из HLS-модели	—
READ_HLS_ SAT(file)	file — строка пути к файлу	Возвращает матрицу составляющей насыщенности изображения исходя из HLS-модели	—
READ_HSV(file)	file — строка пути к файлу	Создает матрицу, описывающую цветное изображение, исходя из HSV-модели	—
READ_HSV_ HUE(file)	file — строка пути к файлу	Возвращает матрицу составляющей оттенка изображения исходя из HSV-модели	—
READ_HSV_ SAT(file)	file — строка пути к файлу	Возвращает матрицу составляющей насыщенности изображения исходя из HSV-модели	—
READ_HSV_ VALUE(file)	file — строка пути к файлу	Возвращает матрицу составляющей значений изображения исходя из HSV-модели	—

Функция	Аргументы	Назначение	Ссылка
READ_IMAGE (file)	file — строка пути к файлу	Читает изображение в матрицу в серых полутонах	—
READ_RED(file)	file — строка пути к файлу	Возвращает матрицу красной составляющей изображения исходя из RGB-модели	—
READBMP(file)	file — строка пути к файлу	Создает матрицу, описывающую изображение в формате BMP (в оттенках серого)	—
READFILE(file, type)	file — строка пути к файлу; type — тип данных	Читает информацию из структурированного текстового файла в матрицу	16.7.1
READPRN(file)	file — строка пути к файлу	Читает информацию из структурированного текстового файла в матрицу	16.7.1
READRGB(file)	file — строка пути к файлу	Создает матрицу, описывающую цветное изображение, исходя из RGB-модели	—
READWAV(file)	file — строка пути к файлу	Создает матрицу, содержащую информацию об амплитуде звукового файла по 4 каналам	—
regress(x,y,k)	x, y — векторы данных; k — степень полинома	Вычисляет вектор коэффициентов для регрессии полиномом произвольной степени (используется совместно с interp)	16.3.2
relax(a,b,c,d,e,F,U, rjac)	a, b, c, d, e — матрицы коэффициентов разностной схемы; F — матрица правой части уравнения; U — матрица граничных условий; rjac — спектральный радиус итераций Якоби	Возвращает матрицу решений уравнения Пуассона на квадратной области с ненулевыми граничными условиями	14.4.3
reverse(A)	A — вектор или матрица	Выполняет перестановку элементов вектора (или столбцов матрицы) в обратном порядке	3.3.3
Rkadapt(y0,x0,x1, npoints,D)	y0 — вектор начальных условий; (x0, x1) — область поиска решения; M — количество шагов интегрирования; D(t, y) — система ОДУ в векторной форме	Возвращает матрицу с решением системы линейных ОДУ в форме задачи Коши по методу Рунге–Кутта с переменным шагом	14.2.3

Функция	Аргументы	Назначение	Ссылка
rkadapt(y0,t0, t1,acc,D,k,s)	y0 — вектор начальных условий; (t0, t1) — область поиска решения; acc — точность; D(t, y) — система ОДУ в векторной форме; k — максимальное количество шагов; s — минимальный шаг интегрирования	Используется для определения решения системы линейных ОДУ в форме задачи Коши по методу Рунге–Кутта в конкретной точке интервала. В большей степени, чем Rkadapt, позволяет влиять на ход поиска решений	14.2.3
rkfixed(y0,x0,x1, M,D)	y0 — вектор начальных условий; (x0, x1) — область поиска решения; M — количество шагов интегрирования; D(t, y) — система ОДУ в векторной форме	Возвращает матрицу с решением системы линейных ОДУ в форме задачи Коши по методу Рунге–Кутта с фиксированным шагом	14.2.3
rnd(x)	x — действительное число	Возвращает число, равномерно распределенное на промежутке от 0 до x (x может быть и отрицательным)	15.9
root(f(x),x,[a,b])	f(x) — уравнение в форме функции; x — переменная; [a, b] — интервал поиска	Функция численного решения алгебраических уравнений. Возможны 2 ее интерпретации, реализующие метод Больцано (поиск на ограниченном промежутке) и метод секущих (требует начальных приближений)	8.1.2
Round(x,y)	x, y — аргументы	Возвращает округленное до целого знака x/y, умноженное на y	18.6
round(x,n)	x — аргумент; n — количество десятичных знаков, до которых производится округление	Округление числа до нужного количества знаков	18.6
rows(A)	A — матрица	Определяет количество столбцов в матрице	3.3.2
rref(A)	A — матрица	Преобразование матрицы с помощью действия над сроками	—
rsort(A,i)	A — матрица; i — индекс строки	Функция сортирует элементы i-й строки в порядке возрастания	3.3.3

Функция	Аргументы	Назначение	Ссылка
SaveColormap (file,M)	file — строка пути к файлу; M — матрица из трех столбцов с целыми положительными числами от 0 до 255	Функция записывает файл палитры исходя из созданной матрицы	6.2.2
sbval(z,x0,x1,D, load, score)	z — вектор приближений для недостающих граничных условий; x0 — левая граница; x1 — правая граница; D(x, y) — ОДУ в векторной форме; load(x0,z) — вектор-функция начальных условий; score(x1,y) — векторная функция, служащая для задания правых граничных условий	Определяет недостающие начальные условия для краевой задачи для системы линейных ОДУ. Использует алгоритм пристрелки (Shooting method)	14.3.1
search(S,Subs,M)	S — строка; Subs — искомая подстрока; M — стартовая позиция для поиска	Определение стартовой позиции определенной подстроки в строке	18.10
sec(z)	z — аргумент	Секанс	18.5
sech(z)	z — аргумент	Гиперболический секанс	18.3
Seed(x)	x — значение Seed для случайных чисел	Меняет величину Seed на новое значение x и возвращает предыдущее	15.9
sign(x)	x — действительное число	Функция знака числа	18.7
signum(x)	z — аргумент	Функция сигнум	7.6
sin(z)	z — аргумент	Синус	18.5
sinfit(x,y,v)	x, y — векторы данных; v — вектор начальных приближений для искомых коэффициентов	Определяет коэффициенты синусоидальной регрессии (a·sin(x+b)+c)	16.3.5
sinh(z)	z — аргумент	Гиперболический синус	18.3

Функция	Аргументы	Назначение	Ссылка
SIUnitsOf(x)	x — размерное выражение	Возвращает размерность переданного ей выражения, выраженную в базовых единицах системы SI	17
skew(A)	A — матрица или вектор выборки	Вычисляет асимметрию эмпирического распределения	15.5.2
slope(x,y)	x, y — векторы данных	Вычисляет коэффициент a линейной регрессии (a·x+b)	16.3.1
sort(v)	v — вектор	Сортировка элементов вектора по возрастанию	3.3.3
sph2xyz(r,q,f)	r, q, f — координаты точки в сферической системе координат	Преобразование сферических координат в декартовы	—
stack(A,B…)	A, B… — матрицы с одинаковым количеством столбцов	Функция слияния матриц сверху вниз	3.3.4
stderr(x,y)	x, y — векторы данных	Возвращает стандартную ошибку, вычисленную на основании линейной регрессии	16.3.1
Stdev(A)	A — матрица или вектор с выборкой	Возвращает значение выборочного исправленного среднеквадратичного отклонения	15.4.2
stdev(A)	A — матрица или вектор с выборкой	Возвращает значение выборочного среднеквадратичного отклонения	15.4.2
Stiffb(y0,t0,t1, M,D,J)	y0 — вектор начальных условий; (t0, t1) — область поиска решения; M — количество шагов интегрирования; D(t, y) — система ОДУ в векторной форме; J(t, y) — якобиан для D(t,y)	Возвращает матрицу решения жесткой системы ОДУ в форме задачи Коши по методу Булирша–Штера	14.2.5
stiffb(y0,t0,t1, acc,D,J,k,s)	y0 — вектор начальных условий; (t0, t1) — область поиска решения; acc — точность; D(t, y) — система ОДУ в векторной форме; J(t, y) — якобиан для D(t,y); k — максимальное количество шагов; s — минимальный шаг интегрирования	Возвращает решение жесткой системы ОДУ в форме задачи Коши по методу Булирша–Штера в последней точке интервала	14.2.5

Функция	Аргументы	Назначение	Ссылка
Stiffr(y0,t0,t1,M, D,J)	См. Stiffb	Возвращает матрицу решения жесткой системы ОДУ в форме задачи Коши по методу Розенброка	14.2.5
stiffr(y0,t0,t1,acc, D,J,k,s)	См. stiffb	Возвращает решение жесткой системы ОДУ в форме задачи Коши по методу Розенброка в последней точке интервала	14.2.5
str2num(S)	S — строка	Переводит число из строкового формата в цифровой	18.10
str2vec(S)	S — строка	Возвращает вектор соответствующих S ASCII-кодов	18.10
strlen(S)	S — строка	Определяет количество знаков в строке	18.10
submatrix(M,ir,jr, ic,jc)	M — матрица; ir, jr — строки; ic, jc — столбцы	Выделяет из матрицы M подматрицу, лежащую между строками ir, jr и столбцами ic, jc (включая их)	3.3.4
substr(S,m,n)	S — строка; m, n — границы выделяемой подстроки	Подстрока, получаемая выделением из строки S, начиная с m-го знака и заканчивая n-м	18.10
supsmooth(x,y)	x, y — векторы данных	Сглаживание сигнала с помощью адаптивного алгоритма	16.5
svd(A)	A — квадратная матрица	Функция сингулярного разложения. Результат выдает в виде единой матрицы	3.3.8
svds(A)	A — квадратная матрица	Возвращает вектор из сингулярных чисел матрицы A	3.3.8
tan(z)	z — аргумент	Тангенс	18.5
tanh(z)	z — аргумент	Гиперболический тангенс	18.3
Tcheb(n,x)	x — действительный аргумент; n — порядок	Вычисляет значение полинома Чебышева 1-го рода порядка n	18.9
time(x)	x — аргумент	Возвращает машинное время в секундах	4.2
tr(A)	A — квадратная матрица	Вычисляет след матрицы	3.2.10
Trunc(x,y)	x, y — аргументы	Возвращает целую часть x/y, умноженную на y	18.6
trunc(x)	x — аргумент	Выделяет целую часть числа	18.6

Функция	Аргументы	Назначение	Ссылка
Ucheb(n,x)	x — действительный аргумент; n — порядок	Вычисляет значение полинома Чебышева 2-го рода порядка n	18.9
until	x, y — аргументы	Возвращает x до тех пор, пока выполняется условие y	18.7
Var(A)	A — матрица или вектор выборки	Вычисляет выборочную исправленную дисперсию	15.4.2
var(A)	A — матрица или вектор выборки	Вычисляет выборочную дисперсию	15.4.2
vec2str(v)	v — вектор ASCII-кодов	Переводит вектор ASCII-кодов в строковую форму	18.10
vlookup(z,M,c)	z — число; M — матрица; c — номер столбца	Возвращает тот элемент c-го столбца, который принадлежит той же строке, что и число z 1-го столбца	—
wave(v)	v — вектор данных	Прямое волновое преобразование сигнала	16.6.2
WRITE_HLS(file)	file — строка пути к файлу	Создает на основании матрицы графический файл исходя из цветовой модели HLS	—
WRITE_HSV(file)	file — строка пути к файлу	Создает на основании матрицы графический файл исходя из цветовой модели HSV	—
WRITEBMP(file)	file — строка пути к файлу	Создает на основании матрицы графический файл (в оттенках серого) в формате BMP	—
WRITEPRN(file)	file — строка пути к файлу	Записывает на основании матрицы текстовый файл	16.7.1
WRITERGB(file)	file — строка пути к файлу	Создает на основании матрицы графический файл исходя из цветовой модели RGB	—
WRITEWAV(file)	file — строка пути к файлу	Запись звукового файла на основании описывающей его матрицы	—
xy2pol(x,y)	x, y — координаты точки в декартовой плоскости	Переводит координаты точки из декартовой системы в полярную	—
xyz2cyl(x,y,z)	x, y, z — координаты точки в трехмерной декартовой системе координат	Преобразование декартовых координат в цилиндрические	—
xyz2sph(x,y,z)	x, y, z — координаты точки в трехмерной декартовой системе координат	Преобразование декартовых координат в сферические	—

Функция	Аргументы	Назначение	Ссылка
Y0(x)	x — аргумент	Функция Бесселя 2-го рода нулевого порядка	18.1.1
Y0.sc(x)	x — аргумент	Нормированная функция Бесселя 2-го рода нулевого порядка (нормировочный множитель exp(–|Im(z)|))	18.1
Y1(x)	x — аргумент	Функция Бесселя 2-го рода 1-го порядка	18.1.1
Y1.sc(x)	x — аргумент	Нормированная функция Бесселя 2-го рода 1-го порядка (нормировочный множитель exp(–|Im(z)|))	18.1
Yn(n,x)	x — аргумент n — степень	Функция Бесселя 2-го рода порядка n	18.1.1
Yn.sc(x)	x — аргумент	Нормированная функция Бесселя 2-го рода произвольного порядка (нормировочный множитель exp(–|Im(z)|))	18.1
ys(n,x)	x — аргумент n — степень	Сферическая функция Бесселя 2-го рода порядка n	12.1

Алфавитный указатель

Вычисления в MATHCAD 12

Книга посвящена Mathcad 12 — последней версии популярнейшего математического пакета компании Mathsoft. Более миллиона пользователей во всем мире избавились от рутины математических расчетов, приобретя Mathcad. И они не ошиблись, ведь Mathcad — это уникальная система, удачно объединившая в себе новейшие технологии аналитических расчетов и традиционные для компьютерной математики численные методы. Простой интерфейс, традиционная форма задания формул, сотни встроенных функций, уникальные возможности построения графиков и диаграмм как на плоскости, так и в пространстве, наличие языка программирования, высокий уровень интеграции с другими приложениями — все это поможет сделать ваши встречи с математикой более легкими и интересными!

Данную книгу характеризуют полнота изложения материала (описаны все основные возможности Mathcad), аналитический подход (объясняется не только как можно решить задачу, но и какие при этом могут возникнуть сложности, а также какие принципы лежат в основе работы соответствующей функции или оператора), сотни примеров, многие из которых являются нестандартными.

Книга не является повторением справочной системы, многие факты, которые вы найдете в ней, не описаны в справке.

ПИТЕР

Заказ книг:

197198, Санкт-Петербург, а/я 619
тел.: (812) 703-73-74, postbook@piter.com

61093, Харьков-93, а/я 9130
тел.: (057) 712-27-05, piter@kharkov.piter.com

www.piter.com — вся информация о книгах и веб-магазин

ISBN 5-469-00639-5

9 785469 006398