**Дмитрий Кирьянов**

# Mathcad 13

Санкт-Петербург
«БХВ-Петербург»
2006

УДК 681.3.06
ББК 32.973.26-018.2
К43

**Кирьянов Д. В.**

К43 Mathcad 13. — СПб.: БХВ-Петербург, 2006. — 608 с.: ил.

ISBN 5-94157-850-4

Книга посвящена методике решения задач высшей математики при помощи компьютерной программы Mathcad. Приводятся примеры расчета типовых задач линейной алгебры, математического анализа, теории дифференциальных уравнений, статистики и обработки данных. Объясняется работа численных алгоритмов, заложенных во встроенных функциях и операторах системы Mathcad. Предлагаются неочевидные приемы решения актуальных задач современной вычислительной науки. Описывается интерфейс Mathcad и его основные составные части, предоставляется необходимая справочная информация. Рассматриваются все новые возможности и отличительные черты Mathcad версии 13. Прилагаемый компакт-диск содержит листинги примеров в формате электронной книги Mathcad, а также учебник по вычислительной математике (его полную HTML-версию и демо-версию мультимедийного обучающего курса).

*Для широкого круга пользователей*

УДК 681.3.06
ББК 32.973.26-018.2

**Группа подготовки издания:**

| | |
|---|---|
| Главный редактор | *Екатерина Кондукова* |
| Зам. главного редактора | *Евгений Рыбаков* |
| Зав. редакцией | *Григорий Добин* |
| Редактор | *Алексей Семенов* |
| Компьютерная верстка | *Натальи Смирновой* |
| Корректор | *Наталия Першакова* |
| Дизайн серии | *Игоря Цырульникова* |
| Оформление обложки | *Елены Беляевой* |
| Зав. производством | *Николай Тверских* |

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 27.01.06.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 49,02.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.02.953.Д.006421.11.04
от 11.11.2004 г. выдано Федеральной службой по надзору
в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-850-4

# Оглавление

*Елене посвящаю эту книгу.*

# Введение

Mathcad — необычная программа. Она относится к классу приложений, называемых PSE (*problem solution environment — программная среда для решения задач*). Это подразумевает, что ее работа не определяется однозначно действиями пользователя (как, например, в текстовых редакторах и т. п.), а является (в большей степени) результатом работы встроенных алгоритмов, недоступных взору исследователя. Введя в редакторе Mathcad выражение, даже довольно простое, например, `df(x)/dx=`, и получив некоторый ответ, многие даже не задумываются о том, что для его вычисления проделывается довольно сложная работа, результат которой заранее не предопределен и зависит от целого ряда факторов, не представленных непосредственно на рабочей области документа (свойств функции `f`, параметров численного алгоритма дифференцирования, значений системных констант и т. д.). Поэтому, проводя даже очень простые расчеты, вам придется иногда сталкиваться с неочевидным поведением программы, которое нельзя понять без ясного представления об основах работы соответствующих алгоритмов, встроенных в Mathcad.

Приложение Mathcad компании MathSoft — самый популярный из компьютерных математических пакетов, остающийся, бесспорно, на протяжении многих последних лет лидером в своем классе математического и образовательного программного обеспечения (ПО). С его помощью можно решать самые разные математические задачи и оформлять результаты расчетов на высоком профессиональном уровне, и сейчас уже сложно представить современного ученого, не пользующегося Mathcad. При помощи этого пакета осуществляются не только простые и вспомогательные вычисления, но и довольно сложные расчеты и научные исследования, использующие комбинации самых разных численных алгоритмов и аналитических преобразований.

Суммируя сказанное, мне хочется отметить четыре основные цели, которые преследует эта книга (первые две из них являются главными).

1. Привести примеры решения в Mathcad стандартных задач и, по возможности, объяснить действие наиболее важных встроенных алгоритмов.

2. Предложить читателю не совсем очевидные приемы решения актуальных задач современной вычислительной науки (многие из которых разработаны автором в ходе научной и учебной деятельности), чтобы читатель мог использовать их в качестве идиом для своей работы.
3. Обозначить новые возможности и отличительные черты новой версии Mathcad 13.
4. Дать обширную справочную информацию, которая поможет читателю (уже накопившему опыт) быстро и эффективно работать в среде Mathcad.

Книга может использоваться как самоучитель, позволяющий "с нуля" освоить самые главные возможности вычислительной системы Mathcad и научиться с ее помощью решать все основные задачи математики. Тем не менее ее главная цель — изложить материал, делая акцент на решении конкретных математических проблем, причем я старался начинать рассказ с краткого определения математических понятий и терминов, конечно, имея в виду, что читатель обладает базовыми математическими знаниями. Книга не дает исчерпывающих сведений о всех деталях работы с Mathcad (с ними вы можете познакомиться, если обратитесь к моему Самоучителю, уже давно выходящему в издательстве "БХВ-Петербург"). Однако, следуя третьей из заявленных мной целей книги, несколько большее внимание уделено всем нововведениям 13-й версии, даже если они не являются важными с точки зрения работы с остальным материалом (бóльшая часть такой информации оформлена в виде примечаний).

Хочется сделать еще несколько замечаний по строению книги. Все листинги автономны и работают вне каких-либо дополнительных модулей. В листингах умышленно, чтобы не загромождать их, нет текстовых полей, — они содержат только расчеты по формулам. Все комментарии к ним находятся в тексте. Почти все графики вынесены в рисунки, причем, если они являются продолжением листингов, это помечено в подрисуночной подписи. Я старался сделать так, чтобы книга была максимально полезной пользователям всех пяти последних версий Mathcad: 2001—13. Все файлы на прилагаемом к книге компакт-диске, насколько это возможно, записывались в формате самой "старой" версии Mathcad — 2001. Функциональности, появившиеся в "новых" версиях, отмечены символами (11), (12) и (13).

Обозначенные символом ⓘ разделы содержат информацию, относящуюся к особенностям численных алгоритмов, а символом 🛠 — полезным советам, идиомам и программным решениям самого автора. Эти разделы при первом знакомстве с Mathcad могут быть пропущены. Пиктограмма 💿 говорит о

том, что соответствующий пример вынесен из текста книги на прилагаемый к ней компакт-диск (который, кстати, содержит почти все листинги и графики из книги, за исключением тривиальных). Знак  символизирует, что в данном разделе рассказывается о конкретном примере из области вычислительной математики, физики, химии, биологии или экономики.

Обо всех перечисленных возможностях я попытался в доступной форме рассказать в этой книге. Дополнительную информацию читатель может получить в Интернете на сервере производителя Mathcad **http://www.mathcad.com**, дистрибьютора Mathcad в России **http://www.mathcad.ru**, на обучающем ресурсе автора на сервере ИПМ РАН им. М. В. Келдыша **http://www.keldysh.ru/comma** и на личной странице автора **http://www.kirianov.orc.ru**.

# ЧАСТЬ I

## Основные математические операции

# Глава 1

# Основные сведения о Mathcad

В данной главе рассмотрены базовые приемы работы с Mathcad 2001—13. Мы дадим самые основные сведения, касающиеся интерфейса и возможностей Mathcad, пользуясь тем, что он интуитивен и похож на другие программы Windows (*см. разд. 1.3*).

## 1.1. Знакомство с Mathcad

Mathcad является математическим редактором, позволяющим проводить разнообразные научные и инженерные расчеты, начиная от элементарной арифметики и заканчивая сложными реализациями численных методов. С точки зрения классификации программного обеспечения, пакет Mathcad — типичный представитель класса PSE-приложений. Пользователи Mathcad — это студенты, ученые, инженеры, разнообразные технические специалисты и все, кому приходится проводить математические расчеты. Благодаря простоте применения, наглядности математических действий, обширной библиотеке встроенных функций и численных методов, возможности символьных вычислений, а также превосходному аппарату представления результатов (графики самых разных типов, мощных средств подготовки печатных документов и Web-страниц) Mathcad стал наиболее популярным математическим приложением.

### 1.1.1. Назначение Mathcad

Что из себя представляет система Mathcad? Следует хорошо представлять себе, что в состав Mathcad входят несколько интегрированных между собой компонентов:

- мощный текстовый редактор, позволяющий вводить, редактировать и форматировать как текст, так и математические выражения;

- вычислительный процессор, умеющий проводить расчеты по введенным формулам, используя встроенные численные методы;
- символьный процессор, позволяющий проводить аналитические вычисления и являющийся, фактически, системой искусственного интеллекта;
- огромное хранилище справочной информации, как математической, так и инженерной, оформленной в качестве интерактивной электронной книги.

Отличительной чертой Mathcad от большинства других современных математических приложений является его построение по принципу WYSIWYG ("What You See Is What You Get" — "что вы видите, то и получите"). Поэтому он очень прост в использовании, в частности, из-за отсутствия необходимости сначала писать программу, реализующую те или иные математические расчеты, а потом запускать ее на исполнение. Вместо этого достаточно просто вводить математические выражения с помощью встроенного редактора формул, причем в виде, максимально приближенном к общепринятому, и тут же получать результат. Кроме того, можно изготовить на принтере печатную копию документа или создать страницу в Интернете именно в том виде, который этот документ имеет на экране компьютера при работе с Mathcad, либо можно включить документ в структуру электронной книги Mathcad.

Создатели Mathcad сделали все возможное, чтобы пользователь, не обладающий специальными знаниями в программировании (а таких большинство среди ученых и инженеров), мог в полной мере приобщиться к достижениям современной вычислительной науки и компьютерных технологий. Для эффективной работы с редактором Mathcad достаточно базовых навыков пользователя. С другой стороны, профессиональные программисты (к которым относит себя и автор этих строк) могут извлечь из Mathcad намного больше, создавая различные программные решения, существенно расширяющие возможности, непосредственно заложенные в Mathcad.

В соответствии с проблемами реальной жизни, математикам приходится решать одну или несколько из следующих задач:

- ввод на компьютере разнообразных математических выражений (для дальнейших расчетов или создания документов, презентаций, Web-страниц или электронных книг);
- проведение математических расчетов (как аналитических, так и при помощи численных методов);
- подготовка графиков с результатами расчетов;
- ввод исходных данных и вывод результатов в текстовые файлы или файлы с базами данных в других форматах;

- подготовка отчетов работы в виде печатных документов;
- подготовка Web-страниц и публикация результатов в Интернете;
- получение различной справочной информации из области математики.

Со всеми этими (а также некоторыми другими) задачами с успехом справляется Mathcad:

- математические выражения и текст вводятся с помощью формульного редактора Mathcad, который по возможностям и простоте использования не уступает, к примеру, редактору формул, встроенному в Microsoft Word;
- математические расчеты производятся немедленно, в соответствии с введенными формулами;
- графики различных типов (по выбору пользователя) с богатыми возможностями форматирования вставляются непосредственно в документы;
- возможен ввод и вывод данных в файлы различных форматов;
- документы могут быть распечатаны непосредственно в Mathcad в том виде, который пользователь видит на экране компьютера, или сохранены в формате RTF для последующего редактирования в более мощных текстовых редакторах (например, Microsoft Word);
- возможно полноценное сохранение документов Mathcad в формате RTF-документов, а также Web-страниц: HTML и (начиная с 12-й версии) XML;

> **Примечание**
>
> Начиная с 12-й версии, файлы Mathcad имеют формат XMCD, являющийся разновидностью текстовой XML-разметки. Применение XML-стандарта оправдано, главным образом, тем, что формат файлов Mathcad становится общеупотребительным для целого ряда приложений и данных самого различного типа. В частности, Mathcad-документы с XML-разметкой теперь можно также просматривать и редактировать "вручную", в любом текстовом редакторе.

- имеется опция объединения разрабатываемых вами документов в электронные книги, которые, с одной стороны, позволяют в удобном виде хранить математическую информацию, а с другой — являются полноценными Mathcad-программами, способными осуществлять расчеты;
- символьные вычисления позволяют осуществлять аналитические преобразования, а также мгновенно получать разнообразную справочную математическую информацию;
- справочная система, а также многочисленные дополнительные материалы, оформленные в виде электронных книг (ресурсы Mathcad), помогают бы-

стро отыскать нужную математическую информацию или пример тех или иных расчетов.

Таким образом, следует хорошо представлять себе, что в состав Mathcad входит несколько интегрированных между собой компонентов — это мощный текстовый редактор для ввода и правки, как текста, так и формул, вычислительный процессор — для проведения расчетов согласно введенным формулам и символьный процессор, являющийся, по сути, системой искусственного интеллекта. Сочетание этих компонентов создает удобную вычислительную среду для разнообразных математических расчетов и, одновременно, документирования результатов работы.

## 1.1.2. Интерфейс пользователя

После того как Mathcad установлен на компьютере и запущен на исполнение, появляется основное окно приложения, показанное на рис. 1.1. Оно имеет ту же структуру, что и большинство приложений Windows. Сверху вниз располагаются заголовок окна, строка меню, панели инструментов (стандартная и форматирования) и *рабочий лист*, или *рабочая область*, документа (worksheet). Новый документ создается автоматически при запуске Mathcad. В самой нижней части окна находится строка состояния. Таким образом, интерфейс пользователя Mathcad сходен с другими приложениями Windows, и, помня о близости редактора Mathcad к обычным текстовым редакторам, вы интуитивно поймете назначение большинства кнопок на панелях инструментов.

> **Примечание**
>
> В версии Mathcad 13 разработчики предусмотрели специальную область окна Mathcad, называемую **Trace Window** (Окно отладки), а также дополнительную панель инструментов **Debug** (Отладка) (см. рис. 1.2). Окно отладки располагается под рабочей областью документа, примыкая сверху к строке состояния (см. рис. 1.1), и служит для облегчения процесса отладки Mathcad-программ. Чтобы закрыть его, достаточно нажать находящуюся в его правом верхнем углу кнопку закрытия окна.

Помимо элементов управления, характерных для типичного текстового редактора, Mathcad снабжен дополнительными средствами для ввода и редактирования математических символов, одним из которых является панель инструментов **Math** (Математика) (см. рис. 1.2). С помощью этой, а также ряда вспомогательных наборных панелей удобно осуществлять ввод уравнений.

Перечислим составные элементы интерфейса пользователя Mathcad:

- верхнее меню или строка меню (menu bar);

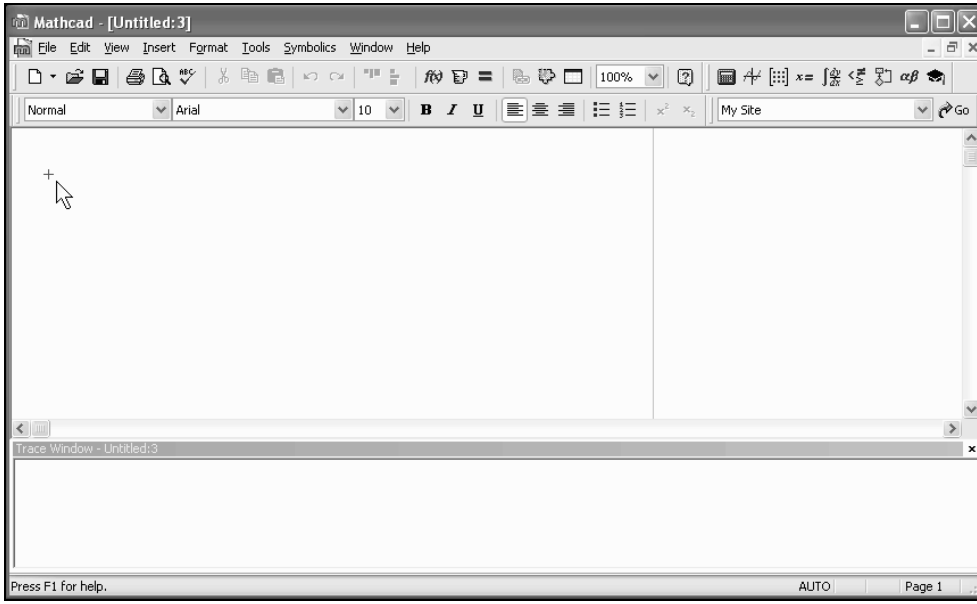



**Рис. 1.1.** Окно приложения Mathcad 13 с пустым документом

### *Примечание*

Далее в книге, говоря о совершении того или иного действия с помощью меню, последовательность выбора пунктов меню будем приводить сокращенно, разделяя их косыми чертами. Например, пункт **Toolbars** меню **View** обозначается как **View / Toolbars**.

- панели инструментов (toolbars) **Standard** (Стандартная), **Formatting** (Форматирование), **Resources** (Ресурсы), **Debug** (Отладка) и **Controls** (Элементы управления);
- панель инструментов **Math** (Математика) и доступные через нее дополнительные математические панели инструментов;
- рабочая область (worksheet);
- вспомогательное окно **Trace Window** (Окно отладки);
- строка состояния (status line, или status bar);
- всплывающие, или контекстные, меню (pop-up menus или context menus);
- диалоговые окна или диалоги (dialogs);
- окна ресурсов Mathcad (Mathcad Resources) со встроенными примерами и дополнительной информацией.

Большинство команд можно выполнить как с помощью меню (верхнего или контекстного), так и панелей инструментов или клавиатуры.

## 1.1.3. Панели инструментов

Панели инструментов служат для быстрого (в один щелчок мыши) выполнения наиболее часто применяемых команд. Все действия, которые можно выполнить с помощью панелей инструментов, доступны и через верхнее меню. На рис. 1.2 изображено окно Mathcad с основными панелями инструментов (три из них расположены непосредственно под строкой меню), а также дополнительными *математическими* (или *наборными*) панелями, о которых речь пойдет ниже. Перечислим основные панели.

- **Standard** (Стандартная) — служит для выполнения большинства операций, таких как действия с файлами, редакторская правка, вставка объектов и доступ к справочным системам;
- **Formatting** (Форматирование) — для форматирования (изменения типа и размера шрифта, выравнивания и т. п.) текста и формул;

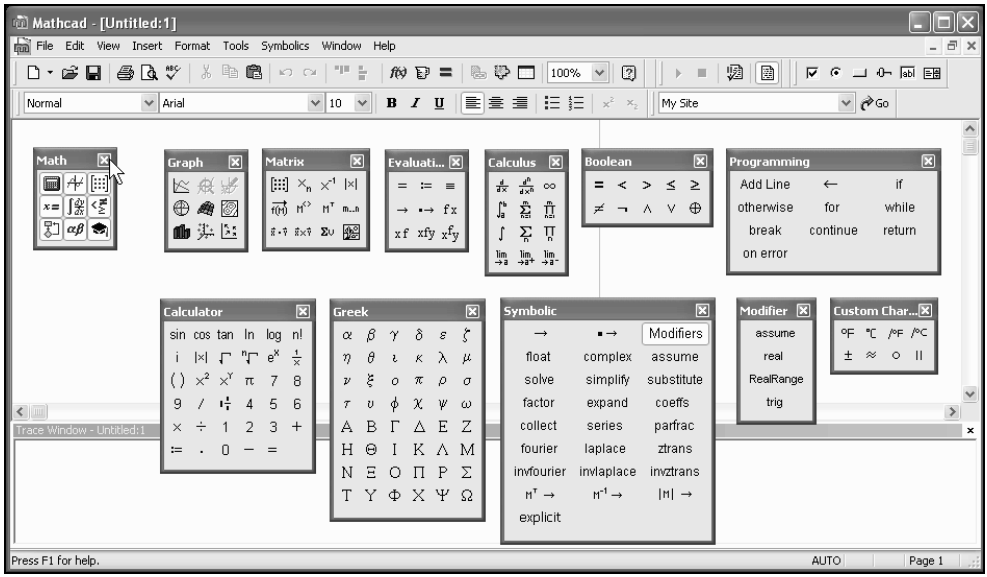



**Рис. 1.2.** Основные и математические (наборные) панели инструментов

- **Math** (Математика) — для вставки математических символов и операторов в документы;

- **Resources** (Ресурсы) — для быстрого вызова *ресурсов* Mathcad (примеров, учебников, электронных книг и т. п.);
- **Controls** (Элементы управления) — для вставки в документы стандартных элементов управления интерфейса пользователя (флажков проверки, полей ввода и т. п.). Эта панель на рис. 1.1 и 1.2 не показана.
- **Debug** (Отладка) — для управления отладкой Mathcad-программ.

> **Примечание**
>
> Панель инструментов **Debug** (Отладка) появилась в версии Mathcad 13.

Группы кнопок на панелях инструментов разграничены по смыслу вертикальными линиями — *разделителями*. При наведении указателя мыши на любую из кнопок рядом с кнопкой появляется *всплывающая подсказка* — короткий текст, поясняющий назначение кнопки. Наряду со всплывающей подсказкой, более развернутое объяснение готовящейся операции можно отыскать в строке состояния.

Панель **Math** (Математика) предназначена для вызова на экран еще девяти панелей (см. рис. 1.2), с помощью которых, собственно, и происходит вставка математических операций в документы. В прежних версиях Mathcad эти математические панели инструментов назывались *палитрами* (palettes) или *наборными панелями*. Чтобы вызвать какую-либо из них, нужно нажать соответствующую кнопку на панели **Math**.

Перечислим назначение математических панелей:

- **Calculator** (Калькулятор) — служит для вставки основных математических операций, получила свое название из-за схожести набора кнопок с кнопками типичного калькулятора;
- **Graph** (График) — для вставки графиков;
- **Matrix** (Матрица) — для вставки матриц и матричных операторов;
- **Evaluation** (Выражения) — для вставки операторов управления вычислениями;
- **Calculus** (Вычисления) — для вставки операторов интегрирования, дифференцирования, суммирования;
- **Boolean** (Булевы операторы) — для вставки логических (булевых) операторов;
- **Programming** (Программирование) — для программирования средствами Mathcad;

- **Greek** (Греческие символы) — для вставки греческих символов;
- **Symbolic** (Символика) — для вставки символьных операторов;
- **Modifier** (Модификатор) — для вставки некоторых операторов (например, преобразования числа);
- **Custom Characters** (Специальные символы) — для вставки специальных символов (единиц измерения температуры и т. п.).

> ***Примечание***
>
> Две последние (из перечисленных) панели используются довольно редко и потому вызываются не через панель **Math** (Математика), а посредством команды меню **View / Toolbars** (Вид / Панели инструментов).

> ⑬ ***Примечание***
>
> Панель инструментов **Custom Characters** (Специальные символы) появилась в версии Mathcad 13, вместе с дополнительными размерностями температуры (в частности, градусов шкалы Цельсия и Фаренгейта).

При наведении указателя мыши на многие из кнопок математических панелей появляется всплывающая подсказка, содержащая еще и сочетание *горячих клавиш*, нажатие которых приведет к эквивалентному действию. Ввод действий с клавиатуры часто удобнее нажатия кнопок панелей инструментов, но требует большего опыта.

Вызвать любую панель на экран или скрыть ее можно с помощью пункта **Toolbars** (Панели инструментов) меню **View** (Вид), выбирая в открывающемся подменю имя нужной панели. Убрать любую панель с экрана можно еще и посредством контекстного меню, которое вызывается щелчком правой кнопкой мыши в любом месте панели (например, на любой кнопке). В контекстном меню следует выбрать пункт **Hide** (Скрыть). Кроме того, если панель *плавающая*, т. е. не прикреплена к основному окну (как, например, все панели на рис. 1.2), то ее можно отключить кнопкой закрытия.

Математические панели, в отличие от основных, можно вызвать или скрыть нажатием соответствующей кнопки панели **Math** (Математика). Присутствие или отсутствие математических панелей показано в виде нажатой (или отжатой) соответствующей кнопки (см. рис. 1.2).

На некоторых рисунках этой главы (см., например, рис. 1.1) виден *курсор* ввода в виде небольшого крестика (на дисплее он имеет красный цвет). С его помощью отмечается незаполненное место в документе, куда в текущий момент можно вводить формулы или текст. Чтобы переместить курсор, достаточно щелкнуть указателем мыши в требуемом месте либо передвинуть его

клавишами-стрелками. Если выполнить щелчок в области формулы или начать ввод выражения на пустом месте, вместо курсора появятся линии редактирования, отмечающие место в формуле или тексте, редактируемое в данный момент (см., например, рис. 1.5 и 1.6 ниже).

## 1.1.4. Справочная информация

Вместе с Mathcad поставляется несколько источников справочной информации, доступ к которым осуществляется через меню **Help** (Справка).

- Справочные системы по вопросам использования Mathcad:
  - **Mathcad Help** (Справка) — система *справки* или *технической поддержки*;
  - **What's This** (Что это такое?) — контекстно-зависимая интерактивная справка;
  - **Developer's Reference** (Справка для разработчиков) — дополнительные главы справки для разработчиков собственных самостоятельных приложений на языке Mathcad;
  - **Author's Reference** (Справка для авторов) — дополнительные главы справки для авторов, разрабатывающих собственные электронные книги Mathcad.
- Ресурсы Mathcad — дополнительные материалы, организованные в специфическом формате электронных книг Mathcad с решением множества математических примеров (пример одной из страниц электронной книги показан на рис. 1.3):
  - **Tutorials** (Учебники) — библиотека электронных книг Mathcad с примерами, которые построены в форме обучающих курсов (от учебника для начинающих пользователей до учебника, адресованного математикам-профессионалам);
  - **QuickSheets** (Быстрые шпаргалки) — большое число документов Mathcad, организованных в виде электронной книги, которые удобно использовать в качестве шаблона для собственных расчетов;
  - **Reference Tables** (Справочный стол) — физические и инженерные таблицы, включающие перечни фундаментальных констант, единиц измерения величин, сводку разнообразных параметров веществ и т. п.;
  - **E-Books** (Электронные книги) — доступ к существующим библиотекам документов пользователя, примерам, а также встроенным электронным книгам, посвященным расширениям Mathcad.

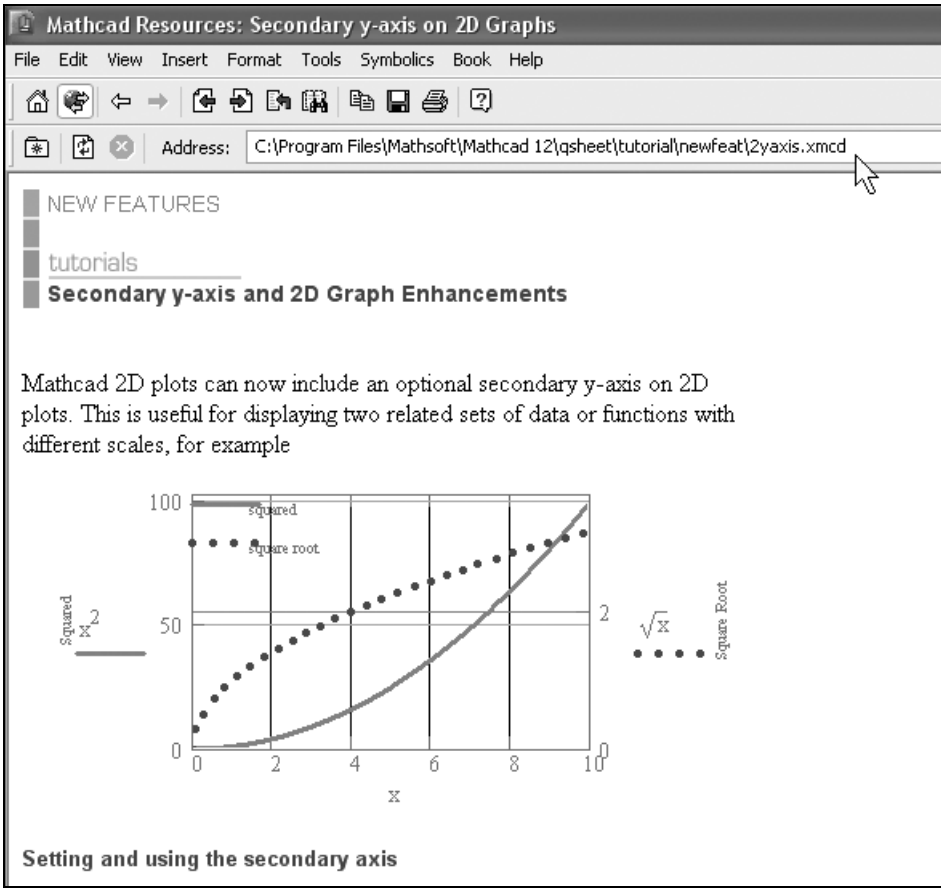



**Рис. 1.3.** Ресурсы Mathcad содержат большое количество справочной и учебной информации

Кроме поименованных, меню **Help** (Справка) содержит следующие пункты:

- Mathcad в сети Интернет:
  - **User Forums** (Форумы) — подключение к специальному интернет-сервису компании MathSoft, дающему возможность пользователям Mathcad общаться между собой, обмениваться программами и получать советы (как друг от друга, так и от разработчиков);
  - **Mathcad.com** — переход на официальный сайт приложения Mathcad;
  - **Mathcad Update** (Обновление Mathcad) — проверка сайта фирмы MathSoft на наличие обновлений Mathcad;

- **About Mathcad** (О программе) — вывод информационного окна со сведениями о текущей версии Mathcad и его разработчиках;
- **Register Mathcad** (Зарегистрировать Mathcad) — регистрация программы через Интернет.

Если в какой-либо момент работы с Mathcad вам потребовалась помощь, выберите **Help / Mathcad Help**, либо нажмите клавишу <F1>, либо кнопку **Help** со знаком вопроса на стандартной панели инструментов. Справка в Mathcad является контекстно-зависимой, т. е. ее содержание определяется тем, на каком месте документа она вызвана.

> ***Примечание***
>
> Помимо описанной стандартной справочной системы, построенной в гипертекстовом виде, Mathcad комплектуется также более полным руководством пользователя в формате PDF. Доступ к PDF-версии руководства пользователя осуществляется через главное меню Windows, т. е. кнопку **Start** (Пуск). Следует отыскать в разделе **Programs** (Программы) главного меню группу программ компании MathSoft и выбрать пункт **Mathcad User Guide**. В 13-й версии открыть PDF-руководство можно и через домашнюю страницу справочной системы, выбрав на ней пункт **User's guide** (Руководство пользователя).

В заключение этого раздела отметим, что как справочная система, так и ресурсы Mathcad представляют собой не просто статьи и примеры с описанием его возможностей. Они могут быть названы полноправными учебными пособиями по нескольким курсам высшей математики (в случае ресурсов, к тому же, еще и интерактивными). Там освещены и основные определения, и математический смысл многих операций, и алгоритмы численных методов. Причем, на взгляд автора, некоторые из тем объяснены лучше, чем где бы то ни было. Если вы в достаточной степени владеете английским языком, обязательно ознакомьтесь с ресурсами Mathcad.

## 1.2. Основы вычислений в Mathcad

Продемонстрируем, как можно быстро начать работу с Mathcad, научиться вводить математические выражения и получать результаты расчетов.

> ***Внимание!***
>
> Бóльшая часть содержания книги с одинаковым успехом применима к пяти последним версиям Mathcad: 2001, 2001i, 11, 12 и 13. Если определенные опции касаются только некоторых версий, на это делается соответствующее указание.

## 1.2.1. Операторы численного и символьного вывода

Для того чтобы выполнить простые расчеты по формулам, проделайте следующее:

1. Определите место в документе, где должно появиться выражение, щелкнув мышью в соответствующей точке документа.
2. Введите левую часть выражения.
3. Введите знак численного равенства = (клавишей <=>) или символьного равенства → (сочетанием клавиш <Ctrl>+<.>). В первом случае будет рассчитано численное значение выражения, а во втором (если это возможно) — аналитическое.

Оставим пока разговор о более надежных способах ввода математических символов и приведем пример простейших расчетов. Для вычисления арккосинуса какого-нибудь числа, например, 0, достаточно ввести с клавиатуры выражение `acos(0)=` или `acos(0)→`. После того как будет нажата клавиша со знаком равенства (или введен знак символьных вычислений →), с правой стороны выражения, как по мановению волшебной палочки, появится результат (листинги 1.1 и 1.2 соответственно).

**Листинг 1.1. Численный расчет простого выражения**

```
acos(0) = 1.571
```

**Листинг 1.2. Аналитический расчет простого выражения**

$$\mathrm{acos}(0) \rightarrow \frac{1}{2}\cdot\pi$$

***Примечание 1***

Здесь и далее во всей книге в листинги вынесено содержание рабочей области документа Mathcad вместе с полученными результатами вычислений. Почти все листинги выглядят в окне Mathcad рассматриваемых версий совершенно одинаково. Исключение составляют листинги, содержащие новые возможности той или иной версии Mathcad (в этих случаях они снабжены специальной ремаркой). Все листинги (а также рисунки) книги помещены также и на прилагаемом к ней компакт-диске, причем, по возможности, в формате Mathcad 2001, что позволяет просматривать их с одинаковым успехом при помощи любой из пяти версий программы (2001—13), которая установлена на вашем компьютере.

### ⑫ ⑬ Примечание 2

Одним из основных достоинств новых версий Mathcad 12 и 13 является новое ядро программы, позволяющее осуществлять вычисления с большей скоростью. Наиболее заметно это проявляется при расчетах с матрицами и векторами больших размеров, а также вложенными массивами (тензорами). Для таких задач разработчики Mathcad анонсируют повышение быстроты расчетов примерно в три раза по сравнению с предыдущими версиями. К тому же, некоторые преимущества в плане ускорения работы программы дает и архитектура Mathcad, построенная на платформе новой технологии .NET компании Microsoft.

Важно отметить, что по умолчанию вычисления в документе производятся в режиме реального времени, т. е. как только пользователь вводит в формулу оператор численного или символьного равенства, Mathcad пытается вычислить это выражение (и все остальные формулы, находящиеся ниже по тексту). Иногда, в основном в случае сложных и долгих расчетов, бывает полезно остановить их нажатием клавиши <Esc>, а затем (в нужный момент) возобновить нажатием клавиши <F9> или командой **Tools / Calculate** (Сервис / Вычислить) и **Tools / Calculate Worksheet** (Сервис / Вычислить во всем документе).

## 1.2.2. Математические выражения и встроенные функции

Описанным в предыдущем разделе образом можно проводить более сложные и громоздкие вычисления, пользуясь при этом всем арсеналом функций, которые заложены разработчиками в систему Mathcad и называются поэтому *встроенными функциями* (в отличие от *пользовательских* функций, конструируемых непосредственно при разработке Mathcad-программы). Легче всего вводить имена встроенных функций с клавиатуры, как в примере с вычислением арккосинуса, но, чтобы избежать возможных ошибок в их написании, лучше выбрать другой путь (тем более что многие из них весьма сложны и имеют несколько аргументов, так что сложно запомнить имена и параметры всех функций наизусть).

Чтобы ввести встроенную функцию в выражение:

1. Определите место в выражении, куда следует вставить функцию.
2. Нажмите кнопку с надписью ***f(x)*** на стандартной панели инструментов.
3. В списке **Function Category** (Категория функции) появившегося диалогового окна **Insert Function** (Вставить функцию) (рис. 1.4) выберите категорию, к которой принадлежит функция, — в нашем случае это категория **Trigonometric** (Тригонометрические).

4. В списке **Function Name** (Имя функции) выберите имя встроенной функции, под которым она фигурирует в Mathcad: в нашем примере — арккосинуса (**acos**). В случае затруднения с выбором ориентируйтесь на подсказку, появляющуюся при выборе функции в нижнем текстовом поле диалогового окна **Insert Function**.
5. Нажмите кнопку **OK** — функция появится в документе.
6. Введите недостающие аргументы введенной функции (в нашем случае это число 0) в *местозаполнителе*, обозначаемом черным прямоугольником) (нижнее выражение на рис. 1.5).

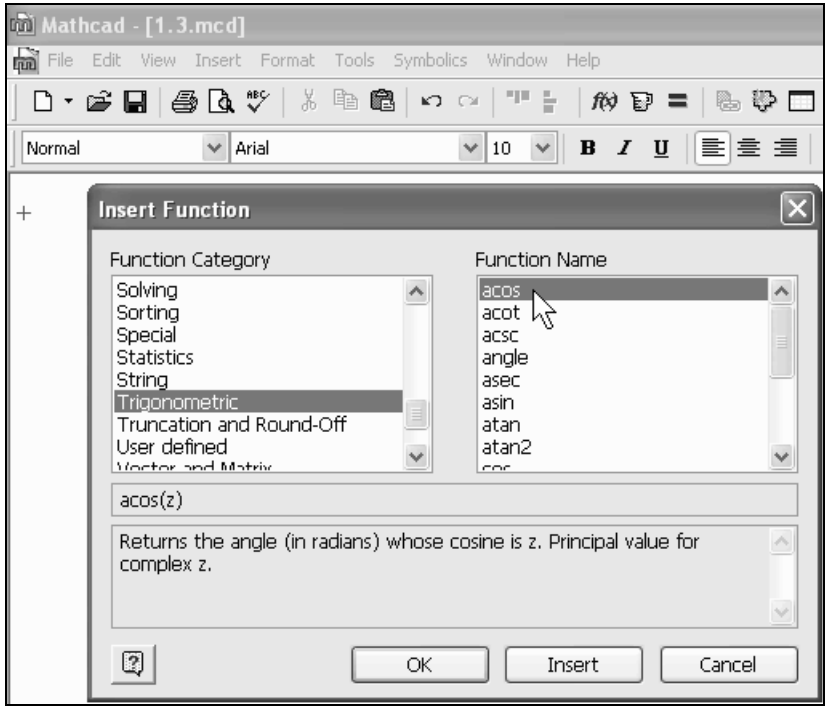



**Рис. 1.4.** Вставка встроенной функции (см. листинги 1.1 и 1.2)

Результатом будет введение выражения из листинга 1.1, для получения значения которого осталось лишь ввести знак (численного или символьного) вывода (оба примера приведены на рис. 1.5).

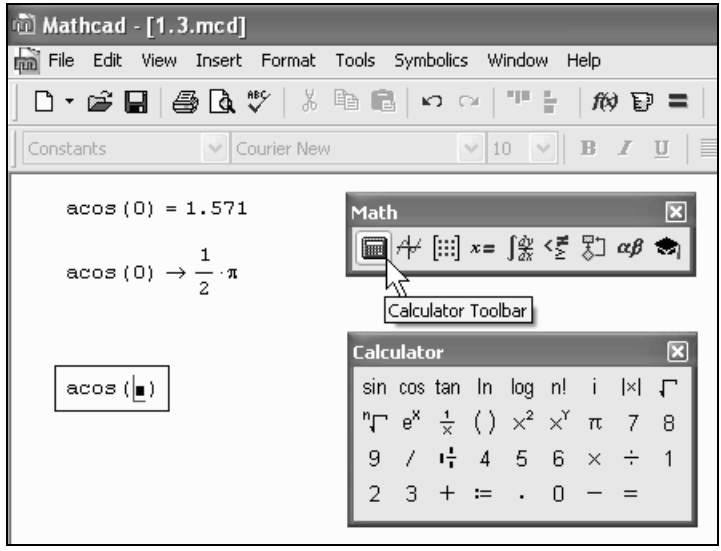



**Рис. 1.5.** Панель инструментов **Math** служит для вызова на экран остальных наборных панелей

### *Примечание*

Большинство численных методов, запрограммированных в Mathcad, реализовано в виде встроенных функций. Пролистайте на досуге списки в диалоговом окне **Insert Function** (Вставить функцию), чтобы представлять себе, какие специальные функции и численные методы можно использовать в расчетах (их подробный перечень приведен в *приложении 3*).

Не всякий символ можно ввести с клавиатуры. Например, неочевидно, как вставить в документ знак интеграла или дифференцирования. Для этого в Mathcad имеются специальные панели инструментов, очень похожие на средства формульного редактора Microsoft Word. Как мы уже говорили ранее, одна из них — панель инструментов **Math** (Математика), показанная на рис. 1.1. Она содержит инструменты для вставки в документы типично математических объектов (операторов, графиков, элементов программ и т. п.). Эта панель показана более крупным планом на рис. 1.5 уже на фоне редактируемого документа.

Панель содержит девять кнопок, нажатие каждой из которых приводит, в свою очередь, к появлению на экране еще одной панели инструментов. С помощью этих девяти дополнительных панелей можно вставлять в документы Mathcad разнообразные объекты. На рис. 1.5, как легко увидеть, на панели **Math** в нажатом состоянии находится только одна кнопка (левая, на которую

наведен указатель мыши). Поэтому на экране присутствует только одна математическая панель — **Calculator** (Калькулятор). Легко догадаться, какие объекты вставляются при нажатии кнопок на этой панели.

> ***Примечание***
>
> Подробнее о назначении этих и других наборных панелей инструментов рассказано ниже (*см. разд. 1.3 и 1.4*).

Большинство математических выражений можно ввести исключительно с помощью панели **Calculator** (Калькулятор), не пользуясь клавиатурой. Например, для расчета выражения `sin(1/2)` нужно сначала нажать кнопку **sin** (самую первую сверху), затем набрать выражение `1/2` в появившемся местозаполнителе внутри скобок. Для этого нажмите последовательно кнопки **1**, **/** и **2** на панели **Calculator** (Калькулятор) и затем, на ней же, кнопку **=**, чтобы получить ответ.

Как видите, вставлять в документы математические символы можно поразному, как и во многих других приложениях Windows. В зависимости от опыта работы с Mathcad и привычек работы на компьютере пользователь может выбрать любой из них.

> ***Совет***
>
> Если вы только начинаете осваивать редактор Mathcad, настоятельно рекомендую, где это только возможно, вводить формулы, пользуясь наборными панелями инструментов и описанной процедурой вставки функций с помощью диалога **Insert Function** (Вставить функцию). Это позволит избежать многих возможных ошибок.

## 1.2.3. Переменные и оператор присваивания

Описанные пока действия демонстрируют использование Mathcad в качестве обычного калькулятора с расширенным набором функций. Для математика же интерес представляет, как минимум, возможность задания переменных и операций с функциями пользователя. Для того чтобы присвоить некоторой переменной (например, переменной `x`) определенное значение, необходимо ввести выражение типа `x:=1`. Этот пример приведен в первой строке листинга 1.3, а в его второй строке осуществляется вычисление значения переменной `x` при помощи оператора численного вывода (знака равенства).

Как вы видите, присваивание обозначается не знаком равенства, а специальным символом, чтобы подчеркнуть его отличие от операции численного вывода. Оператор присваивания вводится нажатием клавиши-двоеточия <:> либо при помощи панели **Calculator** (Калькулятор). Символ равенства "="

говорит о вычислении значения слева направо, а символ ":=" — о присваивании значения справа налево.

### *Примечание 1*

Тем не менее пользователю позволено изменить внешнюю форму оператора на более привычный для математика символ обычного равенства (что категорически не рекомендуется делать, поскольку сильно ухудшает восприятие Mathcad-программы). Для этого (рис. 1.6) следует вызвать нажатием правой кнопки мыши из области оператора присваивания контекстное меню и выбрать в нем пункт **Equal** (Равно). Кстати, подобным образом можно выбирать написание и некоторых других операторов, допускающих обозначение разными символами (например, оператора умножения).

### *Примечание 2*

Если попытаться ввести знак численного вывода (обычного равенства) для переменной, впервые встречающейся в документе, он будет автоматически заменен символом присваивания.

**Листинг 1.3. Присваивание значения переменной и его использование в расчетах**

```
x := 1

x = 1

(x + 5)^2 = 36
```

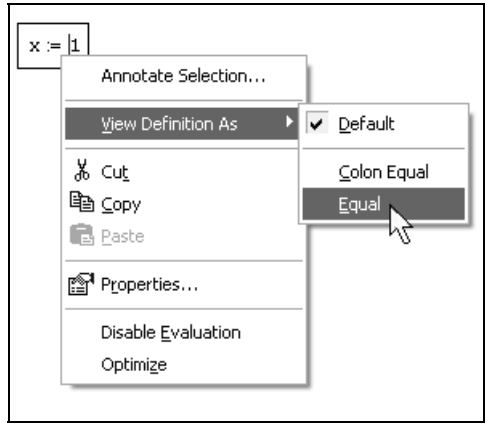



**Рис. 1.6.** Выбор вида оператора присваивания (см. листинг 1.3)

Для того чтобы вычислить значение выражения, содержащего некоторую переменную, следует просто ввести его, а затем применить оператор численного вывода (листинг 1.3, последняя строка). При этом необходимо, чтобы этой переменной ранее в документе было присвоено какое-либо значение.

> ***Примечание***
>
> Помимо оператора обычного присваивания, в Mathcad имеется еще один оператор *глобального присваивания* (≡). Если вставить его для задания значения переменной в любой части документа (например, в самом низу), то эта переменная автоматически будет определена в любой части документа.

В отличие от численных, для символьных вычислений задание значений всех переменных необязательно (листинг 1.4). Если некоторым переменным присвоены значения (как переменной `a` в листинге 1.4), то для получения результата используется это числовое значение. Если же переменной не было присвоено никакого значения (как переменной `k`), она воспринимается аналитически, просто как некоторое имя.

Символьные вычисления, позволяющие решить многие задачи аналитически, являются одной из самых впечатляющих возможностей Mathcad. Фактически Mathcad "знает" математику, по крайней мере, на уровне неплохого ученого. Умелое использование интеллекта символьного процессора Mathcad избавит вас от огромного количества рутинных вычислений, например, интегралов и производных. Обратите внимание на традиционную форму написания выражений (листинг 1.4), единственная особенность заключается в необходимости применения знака символьных вычислений → вместо знака равенства. Его, кстати, можно ввести в редакторе Mathcad с любой из панелей **Evaluation** (Выражения) или **Symbolic** (Символика), а символы интегрирования и дифференцирования — с панели **Calculus** (Вычисления).

**Листинг 1.4. Переменные в аналитических расчетах**

$$a := 3$$

$$\frac{d}{dx}\sin\left(\frac{k \cdot x}{a^2}\right) \rightarrow \frac{1}{9} \cdot \cos\left(\frac{1}{9} \cdot k \cdot x\right) \cdot k$$

## 1.2.4. Функции пользователя

Подобно присваиванию числовых значений переменным, можно определить функции пользователя одного или нескольких аргументов (листинги 1.5

и 1.6). В листинге 1.5 определяется функция f(x), а в листинге 1.6 — функция трех переменных g(a,y,ϕ).

**Листинг 1.5. Определение функции пользователя и расчет ее значений в точке**

$$f(x) := x^2 - 3 \cdot x - 2$$

$$f(0) = -2$$

$$f(10) = 68$$

**Листинг 1.6. Функция пользователя трех аргументов и ее вычисление в точке**

$$g(a, y, \phi) := a \cdot \sin(y + \phi)$$

$$g(1, 0, \pi) = 0$$

График функции f(x) показан на рис. 1.7. Чтобы построить его, следует нажать на панели **Graph** (График) кнопку с нужным типом графика (на нее на рисунке наведен указатель мыши) и в появившейся заготовке графика определить значения, которые будут отложены по осям. В нашем случае потребовалось ввести x в местозаполнитель возле оси X и f(x) — возле оси Y.

### *Примечание 1*

Сравните содержание листинга 1.5 и рис. 1.7. Такой стиль подачи материала будет сохранен во всей книге. Листинги представляют собой фрагменты рабочих областей документа, которые работают без какого-либо дополнительного кода (если это не оговорено особо). Можно ввести содержание любого листинга в новый (пустой) документ, и он будет работать точно так же, как в книге. Чтобы не загромождать листинги, графики выведены в отдельные рисунки. В отличие от рис. 1.6, в следующих рисунках код листингов не дублируется, а если имеется ссылка на листинг в подрисуночной подписи, то это подразумевает, что данный график может быть вставлен в документ после упомянутого листинга.

### *Примечание 2*

На том же графике на рис. 1.7 изображена и вторая кривая, представляющая собой двумерный график функции g(10,x,0). Для того чтобы нарисовать и этот график, потребовалось ввести имя функции g(10,x,0) через запятую после f(x) возле оси Y.

### (12) *Примечание 3*

В Mathcad 12 и 13 запрещено определять функции пользователя посредством рекуррентных выражений, например, f(x)=f(x)+1. При попытке вычис-

ления `f(x)` вместо ее нового (рекуррентного) присваивания, как происходило в прошлых версиях, будет организован бесконечный цикл, который на определенном шаге приведет к операции переполнения. Для организации рекуррентных вычислений используйте новое имя функции, например, `f1(f,x)=f(x)+1` (что даст, в частности, `f1(sin,0)=0`), либо *именной* оператор (см. примечание 4).

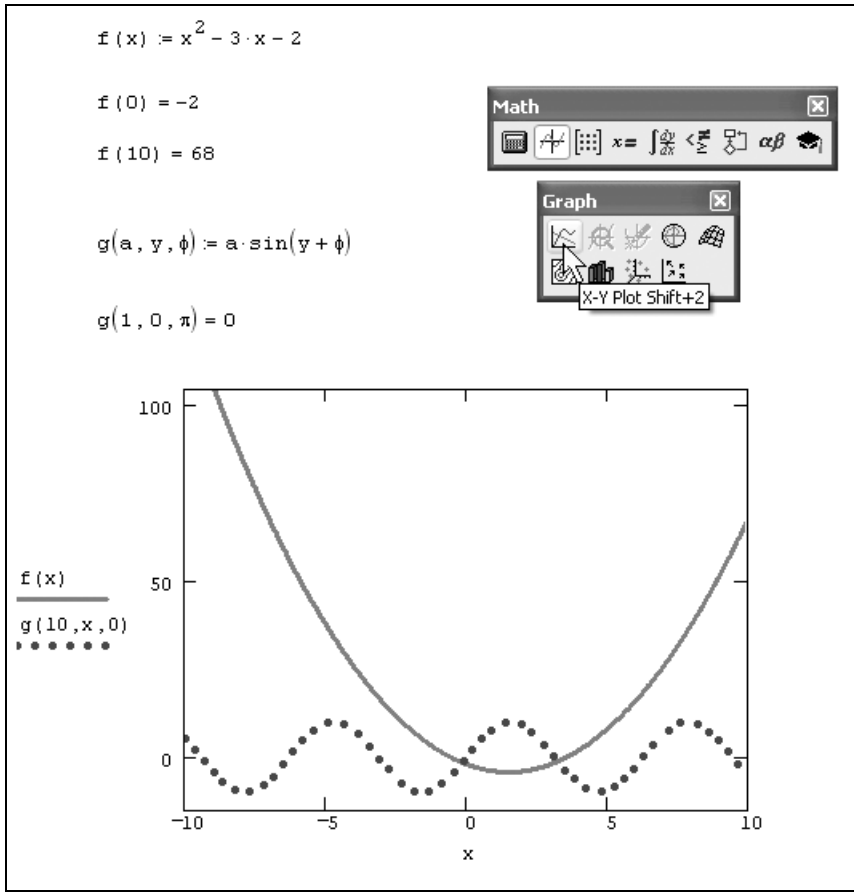



**Рис. 1.7.** Построение графика функции (продолжение листинга 1.5)

## 12 *Примечание 4*

В Mathcad 12 добавлена новая возможность переопределения переменных, размерностей и функций, причем как пользовательских, так и системных. Осуществляется это при помощи *именного оператора* (*namespace operator*), примером действия которого может служить переопределение встроенной функции синус

$\sin_{[mc]}(x):=\sin(x\cdot\pi/180)$ либо пользовательской функции $f_{[this]}(x)=f(x)+1$. Идентификатор [mc] указывает на подмену системного имени Mathcad, а [this] — на рекуррентное переопределение соответствующей функции.

## 1.2.5. Типы чисел

Перечислим основные типы переменных, которые используются в Mathcad.

### Действительные числа

Любое выражение, начинающееся с цифры, Mathcad интерпретирует как число. Поэтому для ввода числа просто начните его набирать на клавиатуре (листинг 1.7).

***Примечание 1***

Если вы продолжите листинг 1.7 последовательным выводом всех переменных, то с удивлением обнаружите, что некоторые из чисел выглядят по-иному (например, число d=0). Это связано с соответствующей настройкой формата результата численного вывода, которую можно поменять, используя команду **Format / Result** (Формат / Результат).

***Примечание 2***

Можно организовать ввод числа в других системах счисления: *двоичной* (binary), *восьмеричной* (octal) или *шестнадцатеричной* (hexadecimal) (листинг 1.8).

**Листинг 1.7. Ввод действительных чисел**

$a := 1000$ $b := 1.3474$

$c := 3124.1$ $d := 45.21 \cdot 10^{-5}$

**Листинг 1.8. Ввод чисел в других системах исчисления**

$a := 100010b$ $a = 34$

$b := 13o$ $c := 0f3h$

### Комплексные числа

Большинство операций в среде Mathcad по умолчанию осуществляются над *комплексными числами*. Комплексное число является суммой действительного и *мнимого числа*, получающегося путем умножения любого действитель-

ного числа на *мнимую единицу* (imaginary unit) `i`. По определению полагается, что $i = \sqrt{-1}$ или $i^2 = -1$.

Чтобы ввести мнимое число, например, `3i`:

1. Введите действительный сомножитель (`3`).
2. Введите символ "i" или "j" непосредственно после него.

### *Примечание*

Для ввода мнимой единицы надо нажать клавиши <1>, <i.>. Если просто ввести символ "i", то Mathcad интерпретирует его как переменную `i`. Кроме того, мнимая единица имеет вид `1i`, только когда соответствующая формула выделена. В противном случае мнимая единица отображается просто как `i` (рис. 1.8).

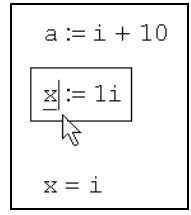



**Рис. 1.8.** Ввод мнимой единицы

Комплексное число можно ввести в виде обычной суммы действительной и мнимой частей или в виде любого выражения, содержащего мнимое число. Примеры ввода и вывода комплексных чисел иллюстрируются листингом 1.9. Для работы с комплексными числами имеется несколько простых функций и операторов, действие которых показано в листинге 1.10.

### *Примечание*

Можно выводить мнимую единицу в результатах вычислений не как `i`, а как `j`. Для смены представления выберите нужное в списке **Imaginary Value** (Мнимое значение) диалогового окна **Result Format** (Формат результата), доступного по команде **Format / Result / Display Options** (Формат / Результат / Опции отображения).

**Листинг 1.9. Ввод и вывод комплексных чисел**

```
a := 2i + 0.5          a = 0.5 + 2i
b := 1.77·e^(2i)       b = −0.737 + 1.609i
c := 25j + 12          c = 12 + 25i
```

**Листинг 1.10. Примеры простых вычислений с комплексными числами (продолжение листинга 1.9)**

```
Im(a) = 2          Re(a) = 0.5
|a| = 2.062        arg(a) = 1.326
arg(b) = 2         |b| = 1.77
```

## Встроенные константы

Некоторые имена в Mathcad зарезервированы под системные переменные, которые называются *встроенными константами* (built-in constants). Встроенные константы делятся на два типа: *математические* (math constants), хранящие значения некоторых общеупотребительных специальных математических символов, и *системные* (system variables), определяющие работу большинства численных алгоритмов, реализованных в Mathcad.

> ***Примечание***
>
> При желании можно изменить значение любой из перечисленных констант или использовать их в качестве переменных в расчетах. Разумеется, если присвоить константе новое значение, прежнее станет недоступным.

Математические константы по-разному интерпретируются при численных и символьных вычислениях. Вычислительный процессор просто воспринимает их как некоторые числа (листинг 1.11), а символьный распознает каждое из них, исходя из математического контекста, и способен выдавать математические константы в качестве результата. Перечислим математические константы:

- ∞ — символ бесконечности (вводится клавишами <Ctrl>+<Shift>+<z>);
- `e` — основание натурального логарифма (клавиша <e>);
- π — число "пи" (вводится клавишами <Ctrl>+<Shift>+<p>);
- `i`, `j` — мнимая единица (вводится клавишами <1>, <i> или <1>, <j>);
- `%` — символ процента, <%>, эквивалентный `0.01`.

**Листинг 1.11. Значения математических констант**

```
∞ = 1 × 10^307
e = 2.718
π = 3.142
```

i = i

j = i

% = 0.01 $100 \cdot 25 \cdot \% = 25$

Системные переменные определяют работу численных методов, заложенных во встроенные функции. Их предустановленные значения перечислены в листинге 1.12 (в принципе, допускается их менять в любой части документа). Системные переменные:

- TOL — точность численных методов;
- CTOL — точность выполнения выражений, используемая в некоторых численных методах;
- ORIGIN — номер начального индекса в массивах и строковых переменных;
- PRNPRECISION — установка формата данных при выводе в файл;
- PRNCOLWIDTH — установка формата столбца при выводе в файл;
- CWD — строковое представление пути к текущей рабочей папке.

**Листинг 1.12. Предустановленные значения системных переменных**

$TOL = 1 \times 10^{-3}$ $CTOL = 1 \times 10^{-3}$

ORIGIN = 0

## Строковые переменные

Значением переменной или функции может быть не только число, но и строка, состоящая из любой последовательности символов, заключенной в кавычки (листинг 1.13). Для работы со строками в Mathcad имеется несколько встроенных функций (*см. приложение 3*).

### *Примечание 1*

Аналогично листингам 1.5 и 1.6 (*см. разд. 1.2.4*) можно определять пользовательские функции строкового типа.

### *Примечание 2*

Системная константа ORIGIN может теперь устанавливать не только номер начального индекса массивов, но и начало отсчета для соответствующих встроенных функций строкового (текстового) аргумента. Если вы хотите включить эту опцию, установите флажок проверки **Use ORIGIN for string**

**indexing** (Использовать `ORIGIN` для индексирования строковых переменных) на вкладке **Calculations** (Вычисления) диалогового окна **Worksheet options** (Опции документа).

### (12) *Примечание 3*

Начиная с версии Mathcad 12, изменены требования к аргументу функций конвертации строковых переменных. Функция `str2num` теперь "умеет" переводить в числа текстовые строки, представляющие двоичную, восьмеричную или шестнадцатеричную запись числа. Аргументом функции `vec2str`, выдающей строку на основе кодировки ее символов, напротив, теперь может быть вектор, состоящий из чисел диапазона `32-255`.

**Листинг 1.13. Ввод и вывод строк**

```
s := "Hello,"                s = "Hello,"
concat(s," world!") = "Hello, world!"
```

## (12) НеЧисло

В версии Mathcad 12 введен новый тип данных, носящий имя NaN — *NotA-Number* (НеЧисло). Он предназначен, главным образом, для идентификации элементов массивов, содержащих пропущенные (по тем или иным причинам) данные. В частности (*см. разд. 13.3.3*), при импорте матрицы данных из внешнего файла элементам, соответствующим пропускам (пустым местам в файле), будет автоматически присвоено значение NaN. Если какие-либо элементы вектора или матрицы, имеющие тип NaN, будут откладываться на графике, то они станут просто игнорироваться при построении кривой. Тем самым, во-первых, повышается надежность импорта данных из файлов; во-вторых, улучшается качество построения графиков рядов данных при наличии пропусков; и в-третьих, пользователю предоставляются дополнительные средства по управлению вычислениями, т. к. любой переменной можно присвоить значение НеЧисло, например: `x:=NaN`.

Помните о том, что математическое выражение, включающее в себя число типа NaN, тоже имеет тип NaN. Идентифицировать значение переменной или выражения как НеЧисло можно при помощи новой служебной функции `isNaN`.

❒ `isNaN(x)` — возвращает `1`, если `x=NaN` и `0` в противоположном случае:

- `x` — аргумент.

## 1.2.6. Ранжированные переменные и матрицы

*Массивами* (arrays) называют упорядоченные последовательности чисел, или *элементов массива*. Доступ к любому элементу массива возможен по его *индексу*, т. е. номеру в последовательности чисел (в листинге 1.14 $a$ — это массив, $a_1$ — его элемент). Применение массивов чрезвычайно эффективно в математических расчетах.

**Листинг 1.14. Одномерный массив (вектор)**

$$a := \begin{pmatrix} 14 \\ 1.4 \\ 4.7 \end{pmatrix}$$

$$a_0 = 14$$

$$a_1 = 1.4$$

$$a_2 = 4.7$$

В Mathcad условно выделяются два типа массивов:

- *векторы* (одноиндексные массивы, листинг 1.14), *матрицы* (двухиндексные, листинг 1.15) и *тензоры* (многоиндексные);
- *ранжированные переменные* (range variables) — векторы, элементы которых определенным образом зависят от их индекса.

**Листинг 1.15. Двумерный массив (матрица)**

$$A := \begin{pmatrix} 0.1 & 0.2 & 0.3 \\ 4 & 5 & 6 \\ 7 & 8 & -9 \end{pmatrix}$$

$$A_{0,0} = 0.1$$

$$A_{2,0} = 7$$

Доступ ко всему массиву осуществляется обычным поименованием векторной переменной. Над элементами массива можно совершать действия, как над обычными числами. Нужно только правильно задать соответствующий

индекс или сочетание индексов массива. Например, чтобы получить доступ к нулевому элементу вектора `a` из листинга 1.14:

1. Введите имя переменной массива (`a`).
2. Нажмите кнопку **Subscript** (Нижний индекс) со значком $\mathbf{x_n}$ на панели **Matrix** (Матрица), либо введите `[`.
3. В появившийся справа снизу от имени массива местозаполнитель введите желаемый индекс (`0`).

Если после этого ввести знак численного вывода, то справа от него появится значение нулевого элемента вектора, как показано во второй строке листинга 1.14.

Чтобы получить доступ к элементу многоиндексного массива (например, элементу $a_{1,0}$ матрицы `a` из листинга 1.15):

1. Введите имя переменной массива (`a`).
2. Перейдите к вводу нижнего индекса, введя `[`.
3. Введите в местозаполнитель индекса первый индекс (`2`), запятую (`,`) и в появившийся после запятой местозаполнитель введите второй индекс (`0`).

В результате будет получен доступ к элементу, как показано в последней строке листинга 1.15.

### *Примечание 1*

В рассмотренных листингах нумерация индексов массивов начинается с нуля, иными словами, первый элемент массива имеет индекс `0`. Стартовый индекс массива задается системной переменной `ORIGIN`, которая по умолчанию равна нулю. Если вы привыкли нумеровать элементы векторов и матриц с единицы, присвойте этой переменной значение `1`. Обратите внимание, что в этом случае попытка выяснить значение нулевого элемента вектора приводит к ошибке, поскольку его значение не определено.

### *Примечание 2*

Помимо доступа к отдельным элементам массива, имеется возможность совершать действия над его подмассивами (например, векторами-столбцами, образующими матрицу). Делается это с помощью оператора со значком $\mathbf{x}^{\langle\rangle}$ на панели **Matrix** (Матрица) (листинг 1.16). Символ "T" во второй строке листинга 1.16 обозначает операцию транспонирования матрицы.

**Листинг 1.16. Доступ к подмассиву (продолжение листинга 1.15)**

$$A^{\langle 0 \rangle} = \begin{pmatrix} 0.1 \\ 4 \\ 7 \end{pmatrix}$$

$$\left(A^{T}\right)^{\langle 2 \rangle^{T}} = (7 \quad 8 \quad -9)$$

## 1.2.7. Размерные переменные

В Mathcad числовые переменные и функции могут обладать *размерностью*. Сделано это для упрощения инженерных и физических расчетов. В Mathcad встроено большое количество *единиц измерения*, с помощью которых и создаются размерные переменные.

Чтобы создать размерную переменную, определяющую, например, силу тока в `10 A`, введите выражение, присваивающее переменной `I` значение `10`: `I:=10`, и затем символ умножения <*>, а потом букву "A". Поскольку все символы, обозначающие единицы измерения, зарезервированы и имеют предустановленные значения (связанные с размерностью), то литера `A` будет распознана Mathcad как Ампер (листинг 1.17, первая строка). Если ранее вы переопределили переменную `A`, присвоив ей какое-либо значение, то восприниматься как единица силы тока она уже не будет.

**Листинг 1.17. Расчеты с размерными переменными**

$$I := 10 \cdot A$$

$$U := 110 \cdot V$$

$$R := \frac{U}{I}$$

$$R = 11\,\Omega$$

Вставить единицу измерения можно и по-другому, не вручную, а при помощи средств Mathcad. Для этого выберите команду **Insert / Unit** (Вставка / Единица), либо нажмите кнопку с изображением мерного стакана на стандартной панели инструментов, либо клавиши <Ctrl>+<U>. Затем в списке **Unit** (Единица измерения) открывшегося диалогового окна **Insert Unit** (Вставка единицы измерений) выберите нужную единицу измерения **Ampere (A)** и нажмите

кнопку **ОК**. Если вы затрудняетесь с выбором конкретной единицы измерения, но знаете, какова размерность переменной (в нашем случае это электрический ток), то попробуйте выбрать ее в списке **Dimension** (Размерность) диалогового окна **Insert Unit** (Вставка единицы измерений). Тогда в списке **Unit** (Единица измерения) появятся допустимые для этой величины единицы измерений, из которых выбрать нужную будет легче (рис. 1.9).

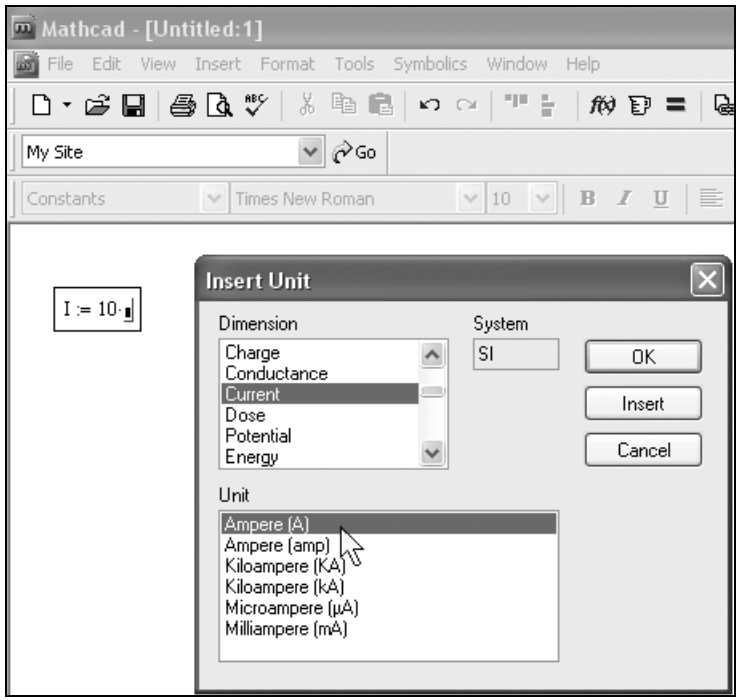



**Рис. 1.9.** Вставка единиц измерения размерной величины

Просмотреть вставку единиц измерения можно и без выхода из диалогового окна **Insert Unit**, нажимая вместо кнопки **ОК** кнопку **Insert** (Вставить). В этом случае вы увидите, что единица измерения появилась в нужном месте документа, и можете поменять ее, оставаясь в диалоге **Insert Unit**.

### *Примечание*

Многие единицы измерения можно представлять в виде различных символов. Например, ампер — как `A` или `amp`, Ом — как `ohm` или Ω и т. д.

Над размерными переменными можно производить любые разумные с физической точки зрения расчеты. Пример расчета сопротивления через отношение напряжения к току приведен в листинге 1.17. Работая с размерными переменными, приготовьтесь к тому, что Mathcad будет постоянно контролировать корректность расчетов. Например, нельзя складывать переменные разной размерности, в противном случае будет получено сообщение об ошибке "The units in this expression do not match" (Размерности в этом выражении не совпадают). Тем не менее позволяется складывать, например, амперы с килоамперами и величины, размерность которых выражена в разных системах измерения (например, СИ и СГС).

### (12) *Примечание 1*

В Mathcad 12 контроль за правильностью совместного применения различных размерных переменных стал еще жестче, что позволяет избежать случайных ошибок. В частности, примененная техника *статической проверки размерности* запрещает расчет функций, которые, в зависимости от значения аргумента, могут выдавать результат различной размерности, например, $f(x=0)=1\cdot m^2$, а $f(x=1)=1\cdot m^3$, и т. п. Запрещено также возводить размерные переменные в степень, не являющуюся целым числом, например $(1\cdot s)^{2.31}$, а также выполнять некоторые другие операции.

### *Примечание 2*

Можно включить автоматический перевод единиц измерения в более простые единицы, как это показано в листинге 1.17 (ответ автоматически переводится в омы). Для этого перейдите в диалоговое окно **Result Format** (Формат результата) на вкладку, посвященную размерностям, с помощью команды **Format / Result / Unit Display** (Формат / Результат / Отображение размерности). Установите в ней флажок **Simplify units when possible** (Упрощать единицы, когда это возможно).

### (13) *Примечание 3*

В Mathcad 13 появились новые единицы измерения температуры (а именно, градусы Цельсия и Фаренгейта, а также их разности). Пример применения этих единиц измерения иллюстрируется предпоследней строкой листинга 1.18. В ней сначала переменной `Temperature` присваивается значение разности температур 20°C, а затем оно выводится уже в кельвинах (разумеется, разность температур составляет также 20 K). О том, что разговор идет именно о разности, а не об абсолютном значении температуры, говорит символ Δ перед размерностью °C в операции присваивания.

⑫ Единицу измерения в системе СИ любой размерной переменной можно вывести при помощи встроенной функции `SIUnitsOf` (последняя строка листинга 1.18):

- `SIUnitsOf (a)` — возвращает единицу измерения переменной (в системе СИ):
  - `a` — переменная.

***Внимание!***

В прежних версиях Mathcad эта функция имела другое название — `UnitsOf`.

**Листинг 1.18. Вывод единицы измерения размерной величины в системе СИ**

$v := 40 \cdot \frac{km}{hr}$ $v = 11.111 \frac{m}{s}$

$SIUnitsOf(v) = 1 \frac{m}{s}$

$Temperature := 20 \cdot \Delta°C$ $Temperature = 20\ K$

$SIUnitsOf(Temperature) = 1\ K$

# 1.3. Ввод и редактирование формул

Формульный редактор Mathcad позволяет быстро и эффективно вводить и изменять математические выражения. Тем не менее некоторые аспекты его применения не совсем интуитивны, что связано с необходимостью избежания ошибок при расчетах по этим формулам. Поэтому не пожалейте немного времени на знакомство с особенностями формульного редактора, и впоследствии при реальной работе вы сэкономите гораздо больше.

## 1.3.1. Элементы интерфейса редактора формул

Перечислим элементы интерфейса редактора Mathcad:

- указатель мыши (mouse pointer) — играет обычную для приложений Windows роль, следуя за движениями мыши;

- курсор — обязательно находится внутри документа в одном из трех видов:
  - курсор ввода (crosshair) — крестик красного цвета, который отмечает пустое место в документе, куда можно вводить текст или формулу;
  - линии ввода (editing lines) — горизонтальная (underline) и вертикальная (insertion line) линии синего цвета, выделяющие в тексте или формуле определенную часть;
  - линия ввода текста (text insertion point) — красная вертикальная линия, аналог линий ввода для текстовых областей;
- местозаполнители (placeholders) — появляются внутри незавершенных формул в местах, которые должны быть заполнены символом или оператором:
  - местозаполнитель символа — черный прямоугольник;
  - местозаполнитель оператора — черная прямоугольная рамка.

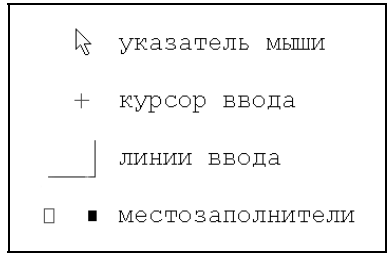



**Рис. 1.10.** Интерфейс редактирования

Курсоры и местозаполнители, относящиеся к редактированию формул, представлены на рис. 1.10.

## 1.3.2. Ввод формул

Бóльшую часть окна Mathcad занимает *рабочая область* документа Mathcad, в которую пользователь вводит математические выражения, текстовые поля и элементы программирования. Ввести математическое выражение можно в любом пустом месте документа Mathcad. Для этого поместите курсор ввода в желаемое место документа, щелкнув в нем мышью, и просто начинайте вводить формулу, нажимая клавиши на клавиатуре. При этом в документе создается *математическая область* (math region), которая предназначена для хра-

нения формул, интерпретируемых процессором Mathcad. Продемонстрируем последовательность действий на примере ввода выражения $x^{5+x}$ (рис. 1.11):

1. Щелкните мышью, обозначив место ввода.
2. Нажмите клавишу <x> — в этом месте вместо курсора ввода появится область с формулой, содержащей один символ x, причем он будет выделен линиями ввода.
3. Введите оператор возведения в степень, нажав клавишу <^> либо выбрав кнопку возведения в степень на панели инструментов **Calculator** (Калькулятор) — в формуле появится местозаполнитель для введения значения степени, а линии ввода выделят этот местозаполнитель.
4. Последовательно введите остальные символы <5>, <+>, <x>.

Таким образом, поместить формулу в документ можно, просто начиная вводить символы, числа или операторы, например, + или /. Во всех этих случаях на месте курсора ввода создается математическая область, иначе называемая *регионом*, с формулой и линиями ввода. В последнем случае, если пользователь начинает ввод формулы с оператора, в зависимости от его типа автоматически появляются и местозаполнители, без заполнения которых формула не будет восприниматься процессором Mathcad (рис. 1.12).

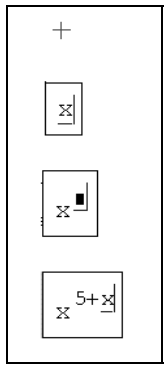


**Рис. 1.11.** Пример ввода формулы (коллаж)

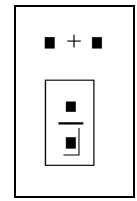


**Рис. 1.12.** Пример начала ввода операторов

## 1.3.3. Перемещение линий ввода внутри формул

Чтобы изменить формулу, щелкните на ней мышью, поместив таким образом в ее область линии ввода, и перейдите к месту, которое хотите исправить.

Перемещайте линии ввода в пределах формулы одним из двух способов:

- щелкая в нужном месте мышью;
- нажимая на клавиатуре клавиши: стрелки, пробел и <Ins>:
  - клавиши со стрелками имеют естественное назначение, переводя линии ввода вверх, вниз, влево или вправо;
  - клавиша <Ins> переводит вертикальную линию ввода с одного конца горизонтальной линии ввода на противоположный;
  - пробел предназначен для выделения различных частей формулы.

Если раз за разом нажимать клавишу пробела в формуле, пример которой рассмотрен выше (см. рис. 1.11), то линии ввода будут циклически изменять свое положение, как это показано на рис. 1.13. Если в ситуации, показанной сверху на этом рисунке, нажать стрелку <←>, то линии ввода переместятся влево (рис. 1.14). При нажатии пробела теперь линии ввода будут попеременно выделять одну из двух частей формулы.

### *Совет*

Привыкнув к использованию пробела для перемещения внутри формул, можно существенно облегчить себе работу с Mathcad.

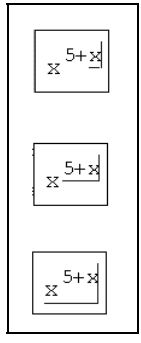



**Рис. 1.13.** Изменение положения линий ввода с помощью пробела (коллаж)

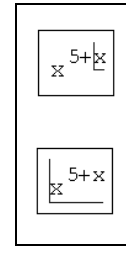



**Рис. 1.14.** Изменение положения линий ввода пробелом после сдвига стрелкой <←> (коллаж)

Таким образом, комбинация клавиш со стрелками и пробела позволяет легко перемещаться внутри формул. Накопив некоторый опыт, вы без труда освоите эту технику. Иногда поместить линии ввода в нужное место формулы с помощью указателя мыши непросто. Поэтому в Mathcad для этого лучше использовать клавиатуру.

## 1.3.4. Изменение формул

Редактируйте формулы в Mathcad так, как подсказывают вам интуиция и опыт работы с другими текстовыми редакторами. Большинство операций правки формул реализованы естественным образом, однако некоторые из них несколько отличаются от общепринятых, что связано с особенностью Mathcad как вычислительной системы. Рассмотрим основные действия по изменению формул.

### Вставка оператора

Операторы могут быть унарными (действующими на один операнд, как, например, оператор транспонирования матрицы или смены знака числа), так и бинарными (например, + или /, действующими на два операнда). При вставке нового оператора в документ Mathcad определяет, сколько операндов ему требуется. Если в точке вставки оператора один или оба операнда отсутствуют, Mathcad автоматически помещает рядом с оператором один или два местозаполнителя.

> ***Внимание!***
>
> То выражение в формуле, которое выделено линиями ввода в момент вставки оператора, становится его первым операндом.

Последовательность вставки оператора в формулу такова:

1. Поместите линии ввода на часть формулы, которая должна стать первым операндом.
2. Введите оператор, нажав кнопку на панели инструментов или сочетание клавиш.

> ***Примечание***
>
> Для того чтобы вставить оператор не после, а перед частью формулы, выделенной линиями ввода, нажмите перед его вводом клавишу <Ins>, которая передвинет вертикальную линию ввода вперед. Это важно, в частности, для вставки оператора отрицания.

На рис. 1.15 показано несколько примеров вставки оператора сложения в разные части формулы, создание которой мы подробно разбирали выше (см. рис. 1.11). В левой колонке рис. 1.15 приведены возможные размещения линий ввода в формуле, а в правой — результат вставки оператора сложения (т. е. нажатия клавиши <+>). Как видно, Mathcad сам расставляет, если это необходимо, скобки, чтобы часть формулы, отмеченная линиями ввода, стала первым слагаемым.

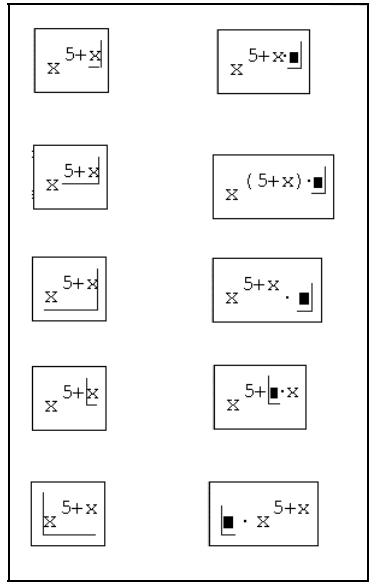


**Рис. 1.15.** Вставка оператора в разные части формулы (коллаж)

Некоторые операторы Mathcad вставит в правильное место независимо от положения линий ввода. Таков, например, оператор численного вывода =, который по смыслу выдает значение всей формулы в виде числа (*см. разд. 1.2.1*).

## Выделение части формулы

Чтобы выделить часть формулы в некоторой математической области (рис. 1.16):

1. Поместите ее между линиями ввода, пользуясь, при необходимости, клавишами-стрелками и пробелом.
2. Поместите указатель мыши на вертикальную линию ввода, нажмите и удерживайте левую кнопку мыши.
3. Удерживая кнопку мыши, протащите указатель мыши вдоль горизонтальной линии ввода, при этом часть формулы будет выделяться обращением цвета.
4. Отпустите кнопку мыши, когда будет выделена нужная часть формулы.

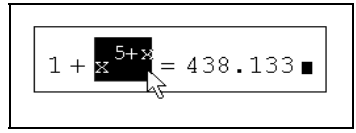



**Рис. 1.16.** Выделение части формулы

***Примечание***

Часть формулы можно выделить и без помощи мыши, нажимая клавиши со стрелками при удерживаемой клавише <Shift>. В этом случае вместо перемещения линий ввода происходит выделение соответствующей части формулы. Многие пользователи находят работу с клавиатурой при выделении части математических областей более удобной.

## Удаление части формулы

Чтобы удалить часть формулы:

1. Выделите ее.
2. Нажмите клавишу <Del>.
3. Кроме того, можно удалить часть формулы, помещая ее перед вертикальной линией ввода и нажимая клавишу <BackSpace>. В некоторых случаях, например, при работе со сложными формулами, для достижения желаемого эффекта может потребоваться повторное нажатие <BackSpace>.

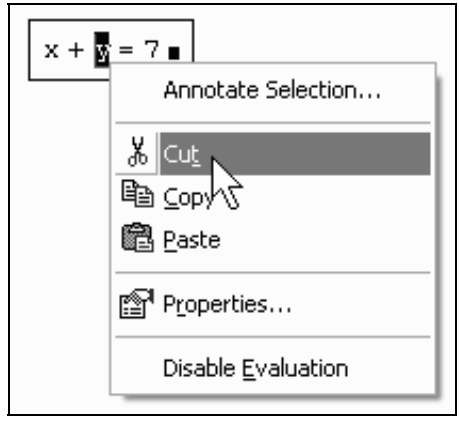



**Рис. 1.17.** Удаление части формулы в буфер обмена

***Примечание***

Имеется еще один способ удаления части формулы: выделите ее нужную часть, затем нажмите комбинацию клавиш <Ctrl>+<X>, тем самым вырезая и помещая ее в буфер обмена. Этот способ удобен в случае, если требуется использовать фрагмент формулы в дальнейшем. Особенно удобно работать с буфером обмена при помощи контекстного меню (рис. 1.17)

## 1.3.5. Программирование

Основными инструментами работы в Mathcad являются математические выражения, переменные и функции. Нередко записать формулу, использующую ту или иную внутреннюю логику (например, возвращение различных значений в зависимости от условий), в одну строку не удается. Назначение программных модулей как раз и заключается в определении выражений, переменных и функций в несколько строк, часто с применением специфических программных операторов.

### Принцип программирования в Mathcad

При помощи элементов программирования можно определять переменные и функции (как показано в листинге 1.19).

**Листинг 1.19. Функция условия, определенная с помощью программы**

```
f(x) := | "negative"   if x < 0
        | "positive"   if x > 0
        | "zero"   otherwise

f(1) = "positive"

f(-1) = "negative"

f(0) = "zero"
```

Традиционное программирование, упрощенный вариант которого применен в Mathcad и осуществляется при помощи панели инструментов **Programming** (Программирование), имеет ряд существенных преимуществ, которые в ряде случаев делают документ более простым и читаемым:

- возможность применения циклов и условных операторов;
- простота создания функций и переменных, требующих нескольких простых шагов;

- возможность создания функций, содержащих закрытый для остального документа код, включая преимущества использования локальных переменных и обработку исключительных ситуаций.

Как видно из листинга 1.19, программный модуль обозначается в Mathcad вертикальной чертой, справа от которой последовательно записываются операторы языка программирования. Чтобы начать создание программного модуля, следует (в случае листинга 1.19 после символа присваивания) нажать на панели **Programming** (Программирование) кнопку **Add Line** (Добавить линию). Затем, если приблизительно известно, сколько строк кода будет содержать программа, можно создать нужное количество линий повторными нажатиями кнопки **Add Line** (Добавить линию) (рис. 1.18).

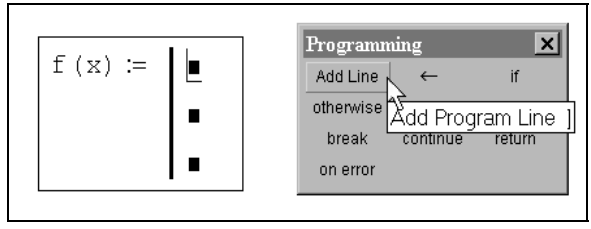



**Рис. 1.18.** Начало создания программного модуля

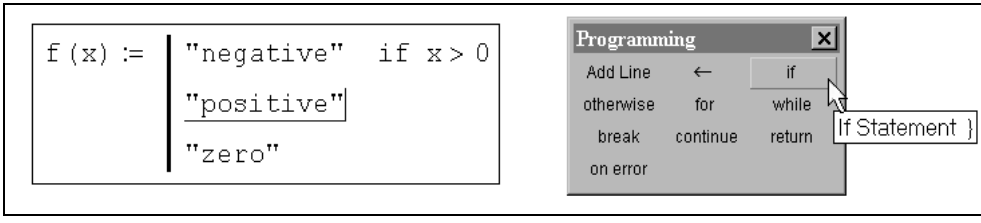



**Рис. 1.19.** Вставка программного оператора

В появившиеся местозаполнители введите желаемый программный код, используя программные операторы. В рассматриваемом примере в каждый местозаполнитель вводится строка, например, в средний — `"positive"` (рис. 1.19). Затем нажимается кнопка **If** (Если) на панели **Programming** (Программирование) и в возникший местозаполнитель вводится выражение `x>0`. После того как программный модуль полностью определен, и ни один место-

заполнитель не остался пустым, функция может использоваться обычным образом, как в численных, так и в символьных расчетах.

***Внимание!***

Не вводите с клавиатуры имена программных операторов. Для их вставки можно применять лишь сочетания клавиш, которые приведены в тексте всплывающей подсказки (рис. 1.18 и 1.19).

## Добавление строк программного кода

Вставить строку программного кода в уже созданную программу можно в любой момент с помощью той же самой кнопки **Add Line** (Добавить линию). Для этого следует предварительно поместить на нужное место внутри программного модуля линии ввода. Например, расположение линии ввода на строке, показанной на рис. 1.18, приведет к появлению новой линии с местозаполнителем перед этой строкой. Если передвинуть вертикальную линию ввода из начала строки (как это показано на рис. 1.20) в ее конец, то новая линия появится после строки. Если выделить строку не целиком, а лишь некоторую ее часть (рис. 1.20), то это повлияет на положение в программе новой строки кода (результат нажатия кнопки **Add Line** показан на рис. 1.21).

***Совет***

Не забывайте, что для желаемого размещения линий ввода внутри формулы можно использовать не только мышь и клавиши со стрелками, но и пробел. С помощью последовательных нажатий пробела линии ввода "захватывают" разные части формулы.

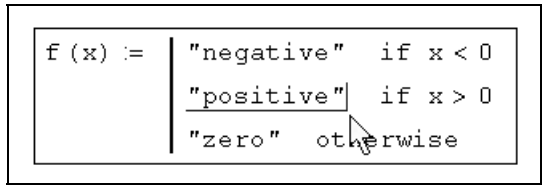



**Рис. 1.20.** Положение линий ввода влияет на положение создаваемой строки программы

Зачем может потребоваться вставка новой линии в положение, показанное на рис. 1.21? Новая вертикальная черта с двумя линиями выделяет фрагмент программы, который относится к условию `x>0`, находящемуся в его заголовке. Пример возможного дальнейшего программирования показан в листинге 1.20.

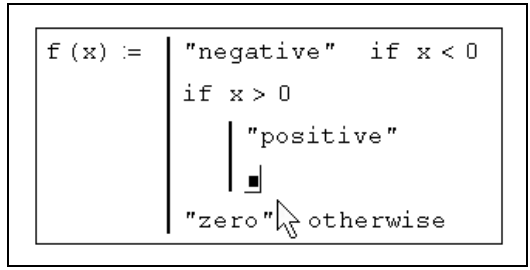



**Рис. 1.21.** Результат вставки новой линии в программу (из положения рис. 1.20)

**Листинг 1.20. Пример усовершенствования программы**

```
f(x) := | "negative"   if x < 0
        | if x > 0
        |     | "positive"
        |     | "big positive"   if x > 1000
        | "zero"   otherwise
f(1) = "positive"
f(10^5) = "big positive"
```

В режиме выполнения программы, а это происходит при любой попытке вычислить `f(x)`, выполняется последовательно каждая строка кода. Например, в предпоследней строке листинга 1.20 вычисляется `f(1)`. Рассмотрим работу каждой строки кода этого листинга.

1. Поскольку `x=1`, то условие `x<0` не выполнено, и в первой строке ничего не происходит.
2. Условие второй строки `x>0` выполнено, поэтому выполняются обе следующие строки, объединенные короткой вертикальной чертой в общий фрагмент.
3. Функции `f(x)` присваивается значение `f(x)="positive"`.
4. Условие `x>1000` не выполнено, поэтому значение `"big positive"` не присваивается `f(x)`, она так и остается равной строке `"positive"`.
5. Последняя строка не выполняется, т. к. одно из условий (`x>0`) оказалось истинным, и оператор `otherwise` (т. е. "иначе") не понадобился.

Таким образом, основной принцип создания программных модулей заключается в правильном расположении строк кода. Ориентироваться в их действии довольно легко, т. к. фрагменты кода одного уровня сгруппированы в программе с помощью вертикальных черт.

## Локальное присваивание (←)

Язык программирования Mathcad не был бы эффективным, если бы не позволял создавать внутри программных модулей локальные переменные, которые "не видны" извне, из других частей документа. Присваивание в пределах программ, в отличие от документов Mathcad, производится с помощью оператора **Local Definition** (Локальное присваивание), который вставляется нажатием кнопки с изображением стрелки (←) на панели **Programming** (Программирование).

> ***Внимание!***
>
> Ни оператор присваивания :=, ни оператор вывода = в пределах программ применять не разрешается.

> ***Внимание!***
>
> В Mathcad 12 и 13 переменным, которые впервые появляются в программных модулях, по умолчанию присваивается значение 0. В прежних версиях программы использование переменных в программах без предварительного присваивания им значений приводило к генерации ошибки (как в расчетах на рабочей области документов Mathcad).

Локальное присваивание иллюстрируется листингом 1.21. Переменная z существует только внутри программы, выделенной вертикальной чертой. Из других мест документа получить ее значение невозможно. На этом же листинге вы видите пример применения оператора цикла for.

**Листинг 1.21. Локальное присваивание в программе**

```
x := | z ← 0
     | for i ∈ 0 .. 5
     |   z ← z + i
x = 15
```

# 1.4. Графики

Одним из наиболее впечатляющих достоинств Mathcad, несомненно, являются развитые возможности построения графиков.

## 1.4.1. Типы графиков

В Mathcad встроено несколько различных типов графиков, которые можно разбить на две большие группы.

- Двумерные графики:
  - X-Y (декартов) график (X-Y Plot);
  - полярный график (Polar Plot).
- Трехмерные графики:
  - график трехмерной поверхности (Surface Plot);
  - график линий уровня (Contour Plot);
  - трехмерная гистограмма (3D Bar Plot);
  - трехмерное множество точек (3D Scatter Plot);
  - векторное поле (Vector Field Plot).

Деление графиков на типы несколько условно, т. к., управляя установками многочисленных параметров, можно создавать комбинации типов графиков, а также новые типы (например, двумерная гистограмма распределения является разновидностью простого X-Y графика).

## 1.4.2. Создание графика

Все графики создаются совершенно одинаково, с помощью панели инструментов **Graph** (График), различия обусловлены отображаемыми данными. Чтобы создать график, например, двумерный декартов:

1. Поместите курсор ввода в то место документа, куда требуется вставить график.
2. Если на экране нет панели **Graph** (График), вызовите ее нажатием кнопки с изображением графиков на панели **Math** (Математика).
3. Нажмите на панели **Graph** (График) кнопку **X-Y Plot** для создания декартова графика (рис. 1.22) или другую кнопку для иного желаемого типа графика.

4. В результате в обозначенном месте документа появится пустая область графика с одним или несколькими местозаполнителями (рис. 1.22, слева). Введите в местозаполнители имена переменных или функций, которые должны быть изображены на графике. В случае декартова графика это два местозаполнителя данных, откладываемых по осям `X` и `Y`.

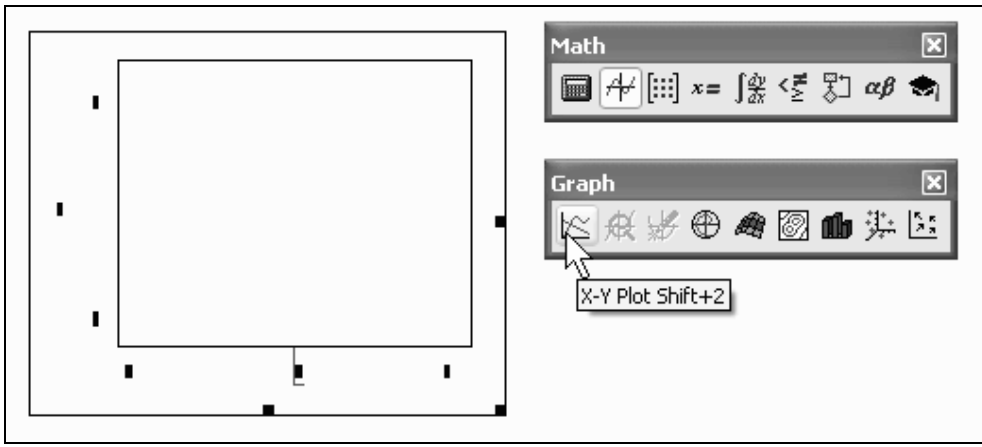



**Рис. 1.22.** Создание декартова графика при помощи панели **Graph**

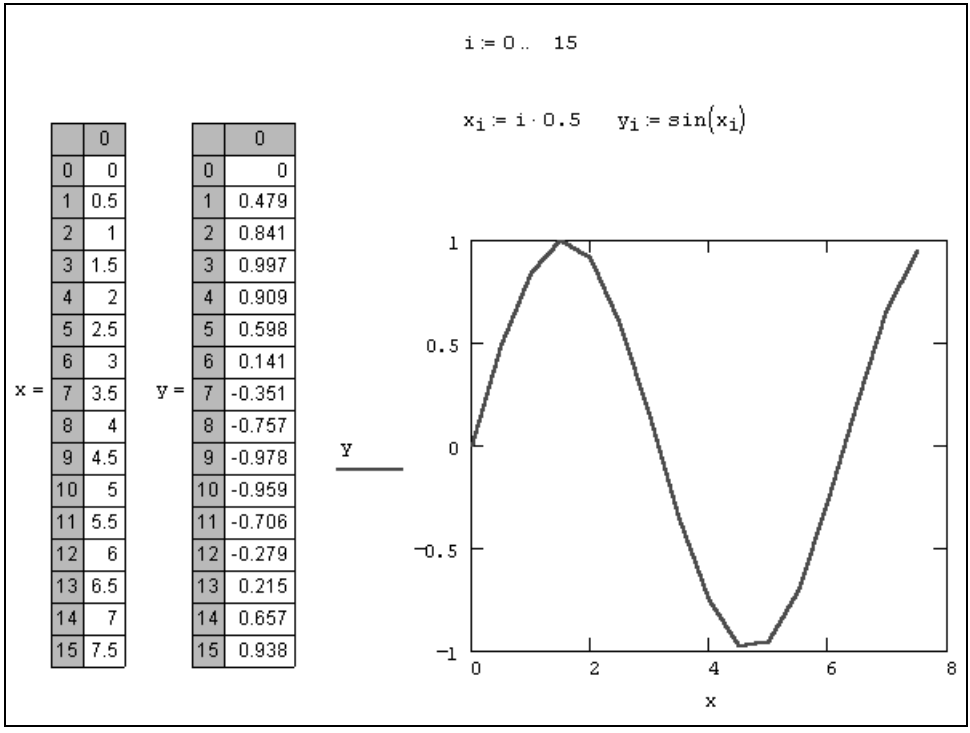
Если имена данных введены правильно, нужный график появится на экране. Созданный график можно изменить, в том числе меняя сами данные, форматируя его внешний вид или добавляя дополнительные элементы оформления.

> ***Внимание!***
>
> Некорректное определение данных приводит вместо построения графика к выдаче сообщения об ошибке.

Чтобы удалить график, щелкните в его пределах мышью и выберите в верхнем меню **Edit** (Правка) пункт **Cut** (Вырезать) или **Delete** (Удалить).

## 1.4.3. X-Y график двух векторов

Самый простой и наглядный способ получить декартов график — это сформировать два вектора данных, которые будут отложены вдоль осей `X` и `Y`. Последовательность построения графика двух векторов `x` и `y` показана на рис. 1.23.




| | 0 |
|---|---|
| 0 | 0 |
| 1 | 0.5 |
| 2 | 1 |
| 3 | 1.5 |
| 4 | 2 |
| 5 | 2.5 |
| 6 | 3 |
| 7 | 3.5 |
| 8 | 4 |
| 9 | 4.5 |
| 10 | 5 |
| 11 | 5.5 |
| 12 | 6 |
| 13 | 6.5 |
| 14 | 7 |
| 15 | 7.5 |

| | 0 |
|---|---|
| 0 | 0 |
| 1 | 0.479 |
| 2 | 0.841 |
| 3 | 0.997 |
| 4 | 0.909 |
| 5 | 0.598 |
| 6 | 0.141 |
| 7 | -0.351 |
| 8 | -0.757 |
| 9 | -0.978 |
| 10 | -0.959 |
| 11 | -0.706 |
| 12 | -0.279 |
| 13 | 0.215 |
| 14 | 0.657 |
| 15 | 0.938 |

**Рис. 1.23.** X-Y график двух векторов

В этом случае в местозаполнители возле осей вводятся просто имена векторов. Также допускается откладывать по осям элементы векторов, т. е. вводить в местозаполнители возле осей имена $x_i$ и $y_i$ соответственно. В результате получается график, на котором отложены точки, соответствующие парам элементов векторов, соединенные отрезками прямых линий. Образованная ими ломаная называется *рядом данных*, или *кривой* (trace).

### *Примечание*

Обратите внимание, что Mathcad автоматически определяет границы графика, исходя из диапазона значений элементов векторов.

Стоит отметить, что подобным образом легко создать и X-Y график столбцов или строк матрицы, применяя оператор выделения столбца и откладывая соответствующие выражения по осям графика (множество подобных примеров вы найдете на рисунках в последующих главах книги).

## 1.4.4. X-Y график функции

Нарисовать график любой скалярной функции `f(x)` можно двумя способами. Первый заключается в дискретизации значений функции, присвоении этих значений вектору и прорисовке графика вектора. Собственно, так и были получены графики синуса на рис. 1.23. Второй, более простой способ, называемый *быстрым построением графика*, заключается во введении функции в один из местозаполнителей (например, у оси `Y`), а имени аргумента — в местозаполнитель у другой оси (рис. 1.24).

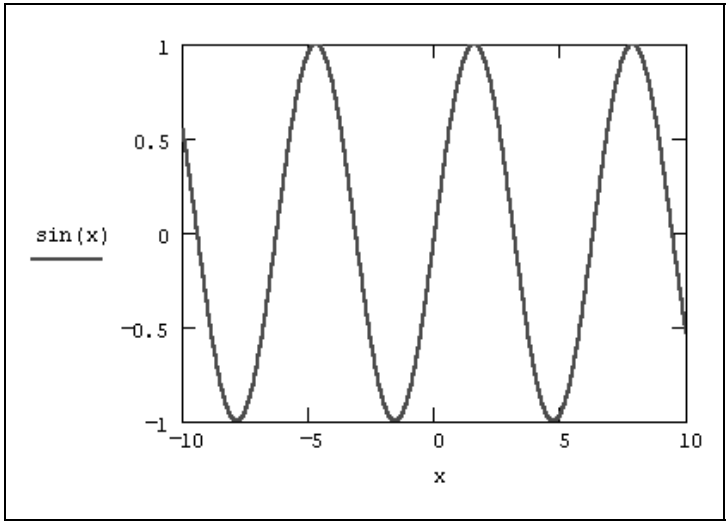



**Рис. 1.24.** Быстрое построение графика функции

В результате Mathcad сам создает график функции в пределах значений аргумента, по умолчанию принятых равными от –10 до 10. Разумеется, впоследствии можно поменять диапазон значений аргумента, и график автоматически подстроится под него.

Необходимо заметить, что если переменной аргумента функции было присвоено некоторое значение до построения в документе графика, то вместо быстрого построения графика будет нарисована зависимость функции с учетом этого значения.

## 1.4.5. Построение нескольких рядов данных

На одном графике может быть отложено до 16 различных зависимостей. Чтобы построить на графике еще одну кривую, необходимо выполнить следующие действия:

1. Поместите линии ввода таким образом, чтобы они целиком захватывали выражение, стоящее в надписи координатной оси Y (рис. 1.25).
2. Нажмите клавишу <,>.

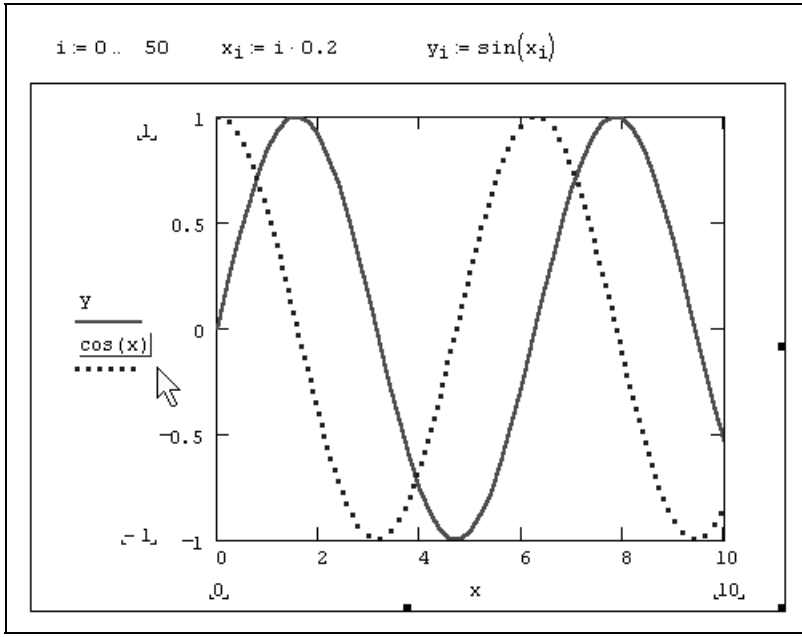



**Рис. 1.25.** Построение нескольких зависимостей на одном графике

3. В результате появится местозаполнитель, в который нужно ввести выражение для второй кривой.
4. Щелкните в любом месте вне этого выражения (на графике или вне его).

После этого вторая кривая будет отображена на графике. На рис. 1.25 уже нарисованы два ряда данных, а нажатие клавиши с запятой <,> приведет к появлению третьего местозаполнителя, с помощью которого можно определить третий ряд данных.

***Примечание***

Чтобы убрать один или несколько рядов данных с графика, удалите клавишами <BackSpace> или <Del> соответствующие им надписи у координатных осей.

Описанным способом будет создано несколько зависимостей, относящихся к одному аргументу. Чтобы на одном и том же графике отложить функции разного аргумента, следует ввести имена этих аргументов через запятую возле оси X (рис. 1.26).

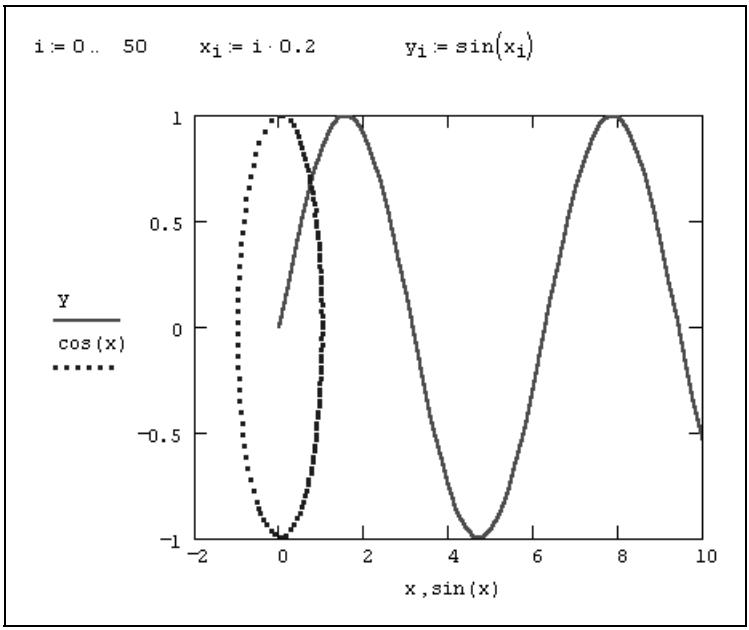



**Рис. 1.26.** Построение нескольких зависимостей от разного аргумента

## 1.4.6. Форматирование графиков

Возможности форматирования координатных осей графиков включают в себя управление их внешним видом, диапазоном, шкалой, нумерацией и отображением некоторых значений на осях при помощи маркеров.

### Масштаб осей

Когда график создается впервые, Mathcad выбирает представленный диапазон для обеих координатных осей автоматически.

Чтобы изменить этот диапазон:

1. Перейдите к редактированию графика, щелкнув в его пределах мышью.
2. График будет выделен, а вблизи каждой из осей появятся два поля с числами, обозначающими границы диапазона. Щелкните мышью в области одного из полей, чтобы редактировать соответствующую границу оси.
3. Пользуясь клавишами управления курсором и клавишами <BackSpace> и <Del>, удалите содержимое поля.
4. Введите новое значение диапазона.
5. Щелкните за пределами поля, и график будет автоматически перерисован в новых пределах.

Чтобы вернуть автоматический выбор какого-либо диапазона, удалите число из соответствующего поля и щелкните вне его. Граница шкалы будет выбрана Mathcad, исходя из значений данных, представляемых на графике.

## Свойства осей

Изменение внешнего вида шкалы, нанесенной на координатную ось, производится с помощью диалогового окна **Formatting Currently Selected X-Y Plot** (Форматирование выбранного графика), в котором следует перейти на вкладку **X-Y Axes** (Оси X-Y) (рис. 1.27). Вызвать диалог можно двойным щелчком мыши в области графика или выполнением команды **Format / Graph / X-Y Plot** (Формат / График / X-Y График) или выбором в контекстном меню команды **Format** (Формат).

С помощью флажков и переключателей легко поменять внешний вид каждой из осей. Перечислим доступные опции и поясним их действие:

- **Log scale** (Логарифмический масштаб) — график по данной оси будет нарисован в логарифмическом масштабе. Это полезно, если данные разнятся на несколько порядков;

  > **(12) *Примечание***
  >
  > В Mathcad 12 появилась новая возможность для построения графиков в логарифмическом масштабе (*см. разд. "Графики в логарифмическом масштабе" ниже*).

- **Grid lines** (Линии сетки) — показать линии сетки (пример сетки на рис. 1.28);
- **Numbered** (Нумерация) — показать нумерацию шкалы. Если убрать этот флажок, то числа, размечающие шкалу, пропадут;

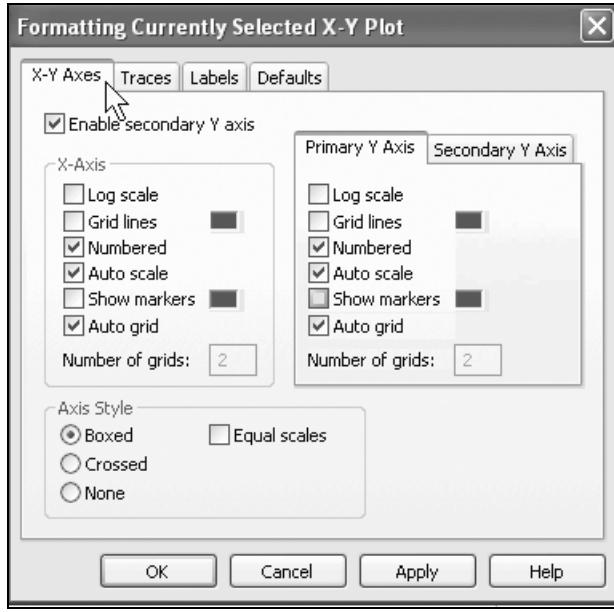



**Рис. 1.27.** Диалоговое окно
**Formatting Currently Selected X-Y Plot**

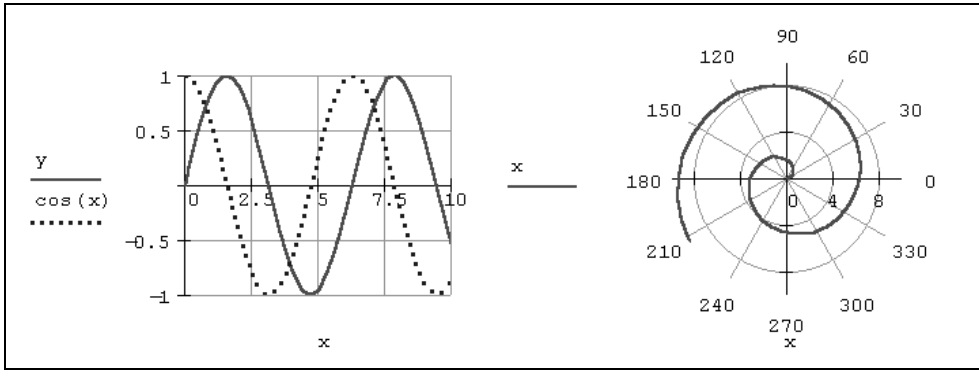



**Рис. 1.28.** Линии сетки на декартовом и полярном графиках,
вид осей — **Crossed**

- **Autoscale** (Автоматический масштаб) — выбор диапазона оси производится автоматически процессором Mathcad;
- **Show markers** (Показать маркеры) — выделение значений на осях (*см. разд. "Маркеры" далее в этой главе*);

- **AutoGrid** (Автоматическая шкала) — разбиение шкалы производится автоматически процессором Mathcad. Если этот флажок снят, в поле ввода рядом с ним следует указать желаемое количество меток шкалы;
- **Equal scales** (Одинаковый масштаб) — оси X и Y принудительно рисуются в одинаковом масштабе;
- **Axes Style** (Вид оси) — можно выбрать один из трех видов системы координат:
  - **Boxed** (Прямоугольник) — как показано на рис. 1.25, 1.26;
  - **Crossed** (Пересечение) — координатные оси в виде двух пересекающихся прямых (рис. 1.28);
  - **None** (Нет) — координатные оси не показываются на графике.

***Примечание***

Изменить описанные параметры можно и в диалоговом окне **Axis Format** (Формат оси), которое появляется, если щелкнуть дважды на самой оси.

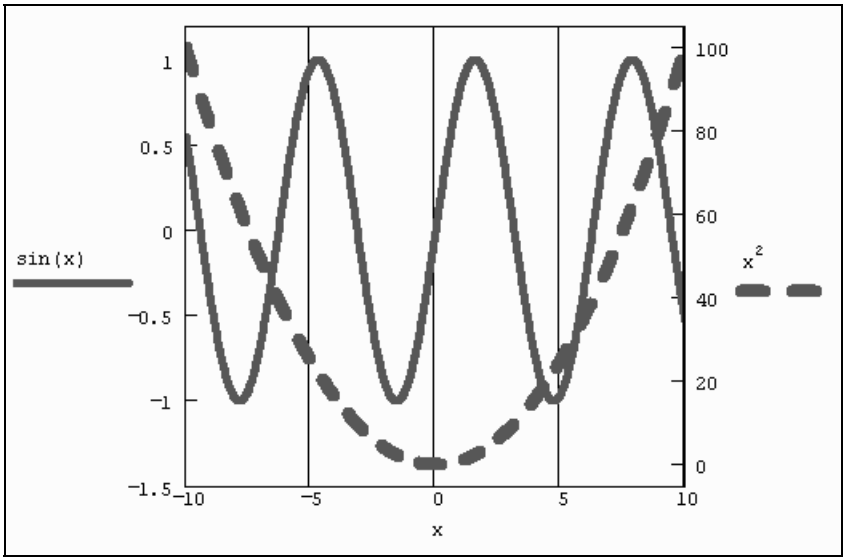
## ⑫ Вторая ось *Y*

В Mathcad 12 появилась дополнительная возможность добавления второй оси Y, обладающей собственной шкалой (рис. 1.29). Использование двух осей ординат очень удобно, когда на одном и том же графике представляются разнородные данные, не совпадающие по величине (например, данные различной размерности, либо различающиеся на несколько порядков, и т. п.).

Для того чтобы задать опцию рисования второй оси ординат:

1. Вызовите двойным щелчком диалоговое окно **Formatting Currently Selected X-Y Plot** (Форматирование выбранного графика) и откройте его вкладку **X-Y Axes** (Оси X-Y) (см. рис. 1.27).
2. Установите флажок проверки **Enable secondary Y axis** (Включить вторую ось Y).
3. Откройте вкладку **Secondary Y axis** (Вторая ось Y) и настройте в ней желаемые параметры второй оси, точно так же, как вы это делаете для первой оси.
4. Нажмите **OK**.
5. В появившиеся местозаполнители возле второй оси ординат введите желаемые имена переменных или выражения, которые вы хотите отложить на данной оси.

6. При желании настройте остальные параметры второй оси `Y` (пределы, маркеры и т. п.).




⑫ **Рис. 1.29.** Декартов график с двумя осями ординат

## ⑫ Графики в логарифмическом масштабе

Как уже говорилось, для построения графиков в логарифмическом масштабе необходимо установить опцию **Log scale** (Логарифмический масштаб) в диалоге **Formatting Currently Selected X-Y Plot** (Форматирование выбранного графика). В целях облегчения труда пользователя по подготовке таких графиков в Mathcad 12 добавлены встроенные функции `logspace` и `logpts`.

Функция `logspace` позволяет создать вектор из точек, отстоящих (в логарифмическом масштабе) друг от друга на равное расстояние, который будет использоваться в качестве аргумента. Например, рассмотрим функцию `f(x)`, которая на одном промежутке `x` меняется быстро, а на другом — медленно. Для того чтобы "красиво" и информативно построить график подобной функции, раньше приходилось создавать вектор `x` вручную, а теперь, благодаря функции `logspace`, этот процесс легко автоматизировать (рис. 1.30). Вторая функция, `logpts`, предназначена для генерации вектора, состоящего

из нескольких серий точек, расположенных линейно-равномерно в пределах каждой из серий (рис. 1.31).

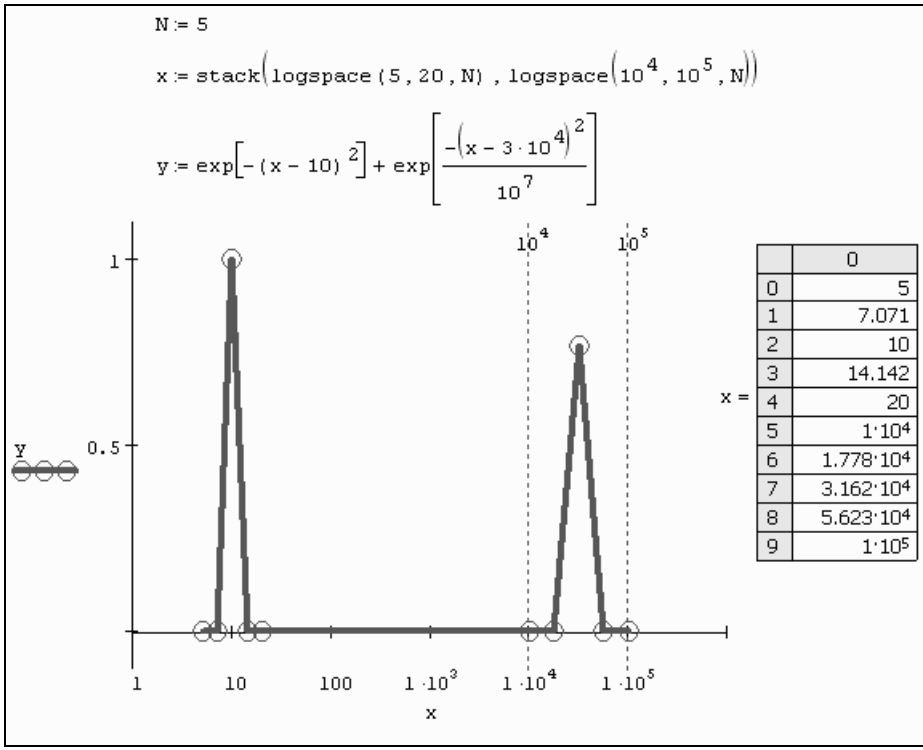
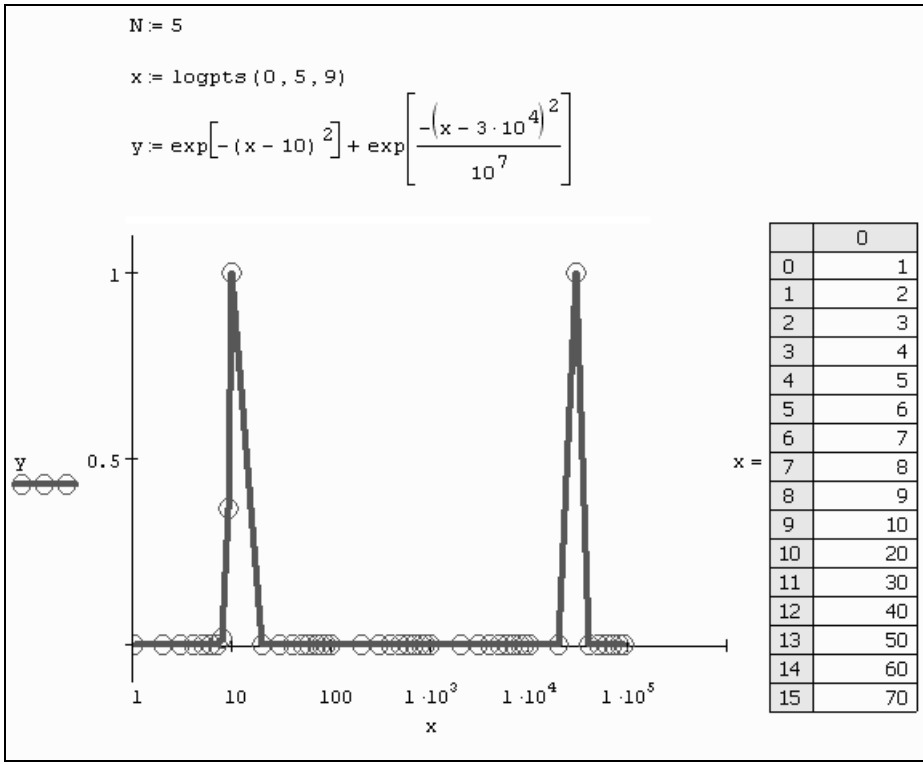



**Рис. 1.30.** Функция `logspace` выдает вектор равномерно-логарифмически расположенных точек

Приведем описание функций и их аргументов:

- `logspace(min,max,N)` — возвращает вектор из чисел, расположенных равномерно (в логарифмическом масштабе) на интервале `(min,max)`:
  - `min,max` — границы интервала;
  - `N` — количество генерируемых точек;
- `logpts(min,dec,N)` — возвращает вектор из чисел, расположенных линейно-равномерно в пределах каждой логарифмической декады, т. е. на интервалах `0.1 –10, 10 –100` и т. д., начиная с $10^{min}$:
  - `min` — показатель начальной граница интервала;

- `dec` — количество серий (декад);
- `N` — количество генерируемых точек в пределах каждой серии (декады).




| | 0 |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |
| 4 | 5 |
| 5 | 6 |
| 6 | 7 |
| 7 | 8 |
| 8 | 9 |
| 9 | 10 |
| 10 | 20 |
| 11 | 30 |
| 12 | 40 |
| 13 | 50 |
| 14 | 60 |
| 15 | 70 |

⑫ **Рис. 1.31.** Функция `logpts` выдает вектор точек, расположенных равномерно по декадам

## ⑬ Ряды данных

Чтобы отформатировать стиль построения кривых, представляющих ряды данных, следует перейти к вкладке **Trace** (Кривые) диалогового окна **Formatting Currently Selected X-Y Plot** (Форматирование выбранного графика) (рис. 1.32). На данной вкладке можно выбрать тип кривых (точки и/или линии), форму и размер маркеров точек, тип и толщину линий, а также задать цвет и легенду для каждой из кривых:

- **Legend label** (Метка легенды) — текст легенды, описывающий ряд данных;

- **Symbol Frequency** (Частота символов) — частота символов, отмечающих точки (этот параметр определяет, будет ли отмечаться каждая точка графика, или каждая 2-я, 3-я и т. д.);
- **Symbol** (Символ) — символ, которым обозначаются отдельные точки данных;
- **Symbol Weight** (Размер символа) — размер точек данных;
- **Line** (Линия) — стиль линии: сплошная, пунктирная, штриховая и т. п.;
- **Line Weight** (Толщина) — толщина линии и точек данных;
- **Color** (Цвет) — цвет линии и точек данных;

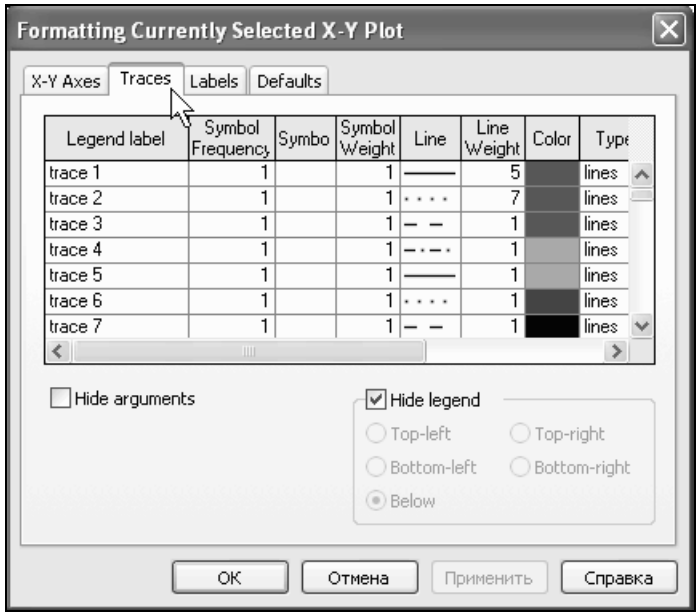



**Рис. 1.32.** Форматирование кривых на графике

- **Type** (Тип) — тип представления ряда данных:
  - **lines** (линии);
  - **points** (точки);
  - **error** (ошибка);
  - **bar** (столбец);

- **step** (шаг);
- **draw** (рисунок);
- **stem** (стержень);
- **solid bar** (гистограмма);

***Примечание***

Для некоторых типов графиков те или иные параметры недоступны (например, нельзя задать символ для шаговой кривой).

❒ **Y-axis** (Ось Y) — информация о том, на какой из двух осей `Y` откладывается ряд данных.

## Маркеры

*Маркером* на координатных осях отмечаются метки некоторых значений. Маркер представляет собой линию, перпендикулярную оси, снабженную числом или переменной.

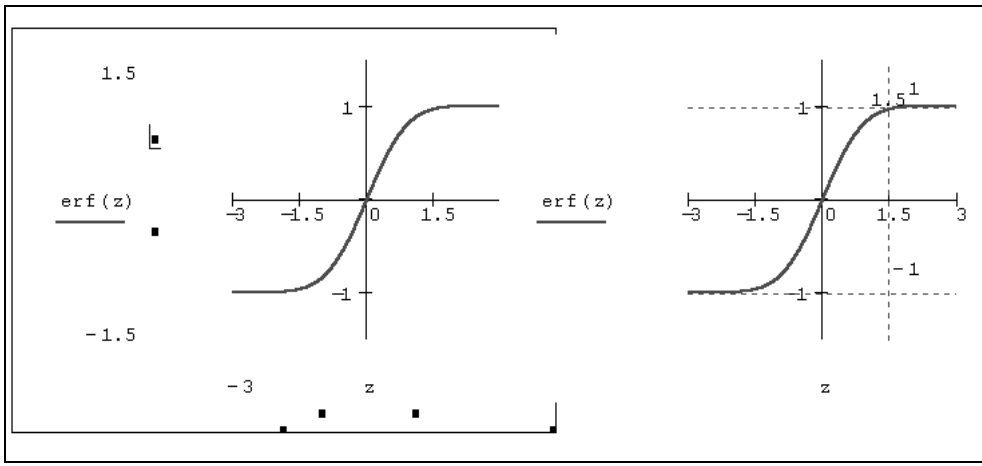



**Рис. 1.33.** Создание маркеров (слева) и готовые маркеры (справа)

Чтобы создать маркер:

1. Дважды щелкните на графике.
2. На вкладке **X-Y Axes** (Оси X-Y) диалога **Formatting Currently Selected X-Y Plot** (Форматирование выбранного графика) (см. рис. 1.27) установите флажок **Show markers** (Показать маркеры).

3. Нажмите кнопку **ОК**.
4. В появившийся местозаполнитель введите число или имя переменной, значение которой вы хотите отобразить на оси маркером (рис. 1.33, слева).
5. Щелкните вне маркера.

Готовые маркеры показаны на рис. 1.33, справа. На каждой из осей допускается установить по два маркера. Если определен лишь один из них, то второй виден не будет.

## 1.4.7. Трехмерные графики

Коллекция трехмерных графиков — настоящее чудо, которое Mathcad дарит пользователю. За считанные секунды вы можете создать великолепную презентацию результатов своих расчетов. Рамки данной книги не позволяют описать технику их создания и форматирования подробно, поэтому мы ограничимся лишь вводными замечаниями и простыми примерами, которые помогут ориентироваться в дальнейшем материале. Для этого рассмотрим на простом примере функции `z(x,y)` и матрицы `Z` (они заданы в листингах 1.22 и 1.23 соответственно) технику построения трехмерных графиков различных типов

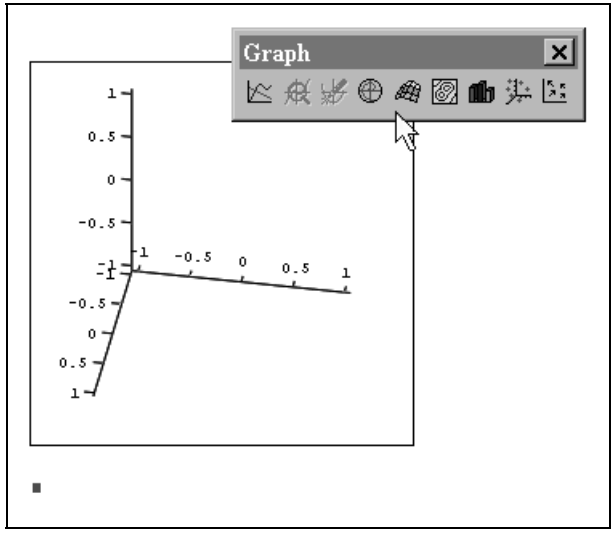



**Рис. 1.34.** Создание трехмерного графика

Чтобы создать трехмерный график, требуется нажать кнопку с изображением любого из типов трехмерных графиков на панели инструментов **Graph** (Гра-

фик). В результате появится пустая область графика с тремя осями (рис. 1.34) и единственным местозаполнителем в нижнем левом углу. В этот местозаполнитель следует ввести либо имя `z` функции `z(x,y)` двух переменных для *быстрого построения трехмерного графика* (рис. 1.35), либо имя матричной переменной `Z`, которая задаст распределение данных $Z_{x,y}$ на плоскости `XY` (рис. 1.36). Еще раз отметим, что для получения графиков (и этих, и последующих) не требуется никакого текста, кроме соответствующего листинга и введения имени соответствующей функции или матрицы в местозаполнитель.

**Листинг 1.22. Функция для быстрого построения трехмерных графиков**

$$z(x, y) := x^2 + y^2$$

**Листинг 1.23. Матрица для отображения на трехмерных графиках**

$$Z := \begin{pmatrix} 1 & 1 & 0 & 1.1 & 1.2 \\ 1 & 2 & 3 & 2.1 & 1.5 \\ 1.3 & 3.3 & 5 & 3.7 & 2 \\ 1.3 & 3 & 5.7 & 4.1 & 2.9 \\ 1.5 & 2 & 6.5 & 4.8 & 4 \end{pmatrix}$$

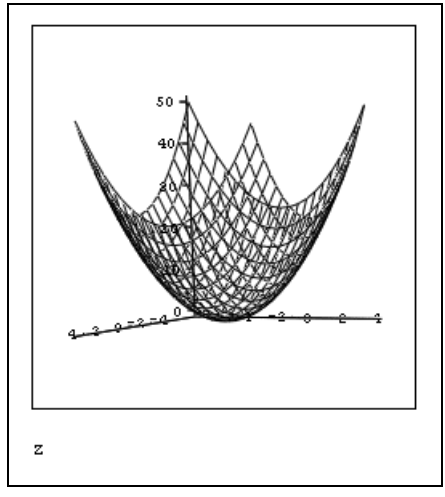



**Рис. 1.35.** Быстрое построение графика поверхности функции (продолжение листинга 1.22)

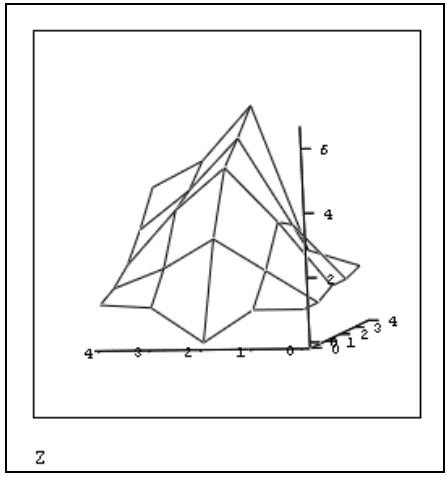



**Рис. 1.36.** График поверхности, заданный матрицей (продолжение листинга 1.23)

Помимо трехмерных графиков поверхности, нажатие соответствующих кнопок на панели **Graph** (График) приводит к созданию графика линий уровня (рис. 1.37), трехмерной гистограммы (рис. 1.38), трехмерного распределения точек (рис. 1.39) или векторного поля (рис. 1.40). Все эти графики представляют данные, составленные листингами 1.22 и 1.23.

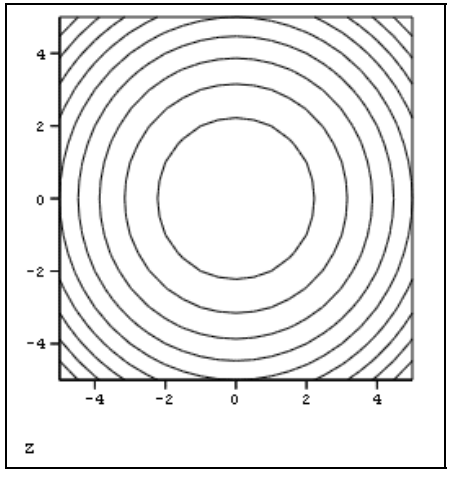



**Рис. 1.37.** Быстрое построение графика линий уровня (продолжение листинга 1.22)

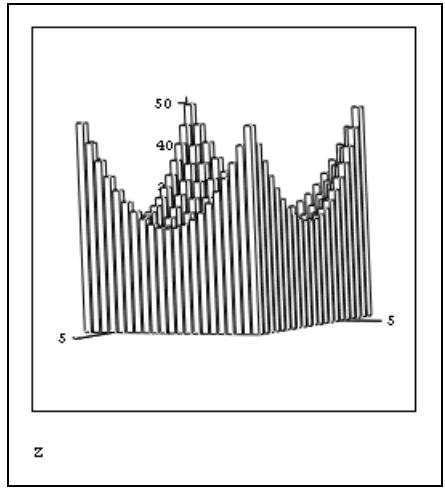



**Рис. 1.38.** Быстрое построение трехмерной гистограммы (продолжение листинга 1.22)

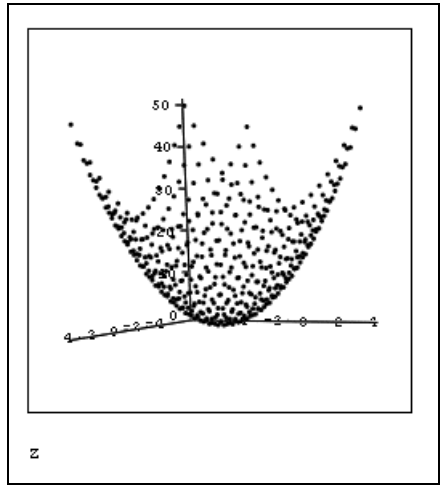



**Рис. 1.39.** Быстрое построение графика трехмерного распределения точек (продолжение листинга 1.22)

Форматирование трехмерных графиков выполняется с помощью диалогового окна **3-D Plot Format** (Форматирование 3-D графика), которое вызывается двойным щелчком мыши.

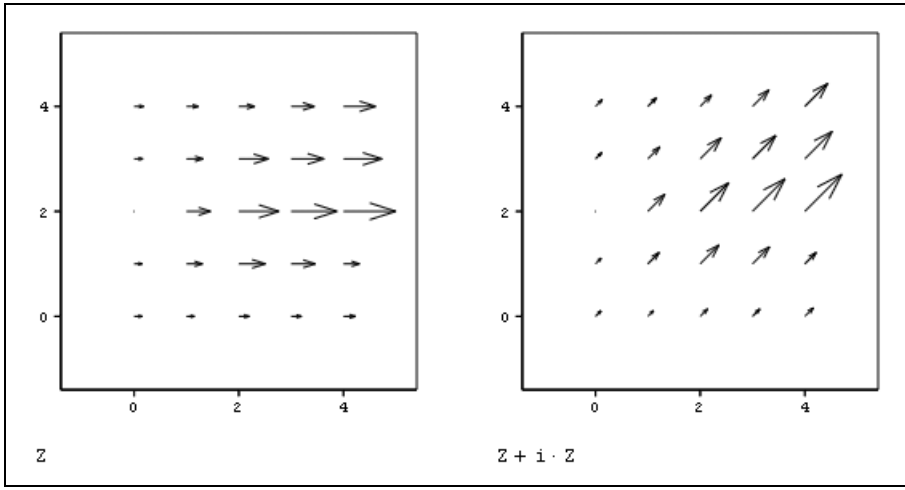



**Рис. 1.40.** Два графика векторных полей, заданных матрицами (продолжение листинга 1.23)

# 1.5. Отладка и комментирование программ

Программа Mathcad, несмотря на исключительную дружественность к пользователю и близость к обычным компьютерным редакторам, все-таки является полноценной системой программирования. Благодаря этому разработчики предусмотрели ряд инструментов, которые служат для отладки расчетов в среде Mathcad, а также для их комментирования.

## 1.5.1. Сообщения об ошибках

Когда процессор Mathcad по тем или иным причинам не может вычислить выражение, он вместо ответа выдает сообщение об ошибке (рис. 1.41). Если курсор находится вне формулы с ошибкой, то в ней имя функции или переменной, которая вызвала ошибку, отмечается красным цветом (сверху на рис. 1.41). При щелчке на такой формуле под ней появляется текстовое сообщение о типе ошибки, обрамленное черным прямоугольником (рис. 1.41, снизу).

Если некоторые выражения вызывают ошибку, они просто игнорируются, а следующие выражения в документе по-прежнему вычисляются. Конечно, если формулы, вызвавшие ошибку, влияют на значения нижеследующих формул, то они будут также интерпретированы как ошибочные. Поэтому,

встречая в документе сообщения об ошибках, найдите сначала самое первое из них. Часто ее устранение позволяет избавиться и от последующих ошибок.

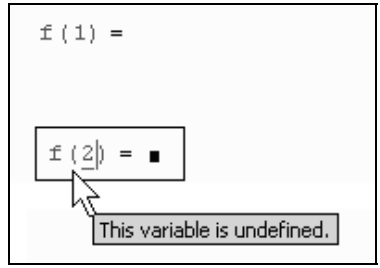



**Рис. 1.41.** Сообщение об ошибке

Помимо, собственно, сообщений об ошибках, которые приводят к аварийной остановке расчетов некоторых формул, последние версии Mathcad выдают пользователю дополнительные предупреждения о вероятных ошибках. Эти предупреждения выделяются при помощи подчеркивания волнистой зеленой линией и связаны, главным образом, с переопределением уже существующих переменных или функций, как встроенных, так и пользовательских (рис. 1.42).

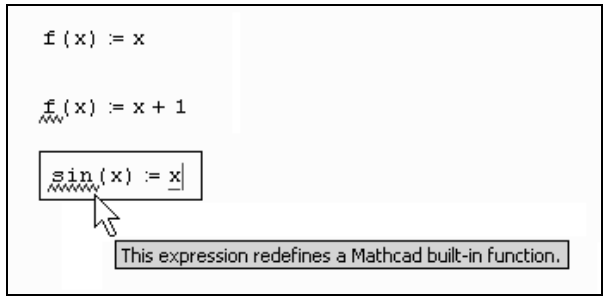



**Рис. 1.42.** Предупреждение о возможной нефатальной ошибке

## ⑬ 1.5.2. Отладка программ

Начиная с версии 13, Mathcad стал обладать расширенными возможностями по отладке программ. Для отладки предназначены три новых элемента Mathcad, связанные между собой: встроенные функции `trace` и `pause`, окно **Trace Window** (Окно отладки) и панель инструментов **Debug** (Отладка).

Встроенные функции отладки позволяют организовать в документах Mathcad определенные точки, в которых можно наблюдать значения любых переменных и функций, а также прерывать и продолжать расчеты. Сами значения наблюдаемых переменных выводятся в текстовом виде в окно **Trace Window** (Окно отладки), а управлять приостановом и продолжением вычислений, а также опциями отладки, можно при помощи панели **Debug** (Отладка).

Панель **Debug** (Отладка) содержит следующие кнопки (рис. 1.43):

- **Resume** (Продолжить) — для продолжения процесса вычислений, прерванного функцией `pause`;
- **Interrupt** (Прервать) — для отключения дальнейших вычислений (прерванных благодаря работе функции `pause`);
- **Toggle Debugging** (Включить отладку) — для включения (или отключения) опции отладки (т. е. работы функций отладки);
- **Trace Window** (Окно отладки) — для вызова на экран (или скрытия) окна отладки.

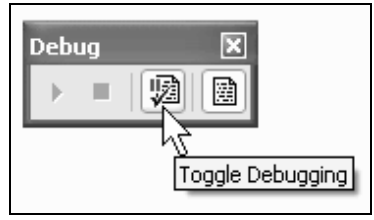



**Рис. 1.43.** Панель **Debug**

***Примечание***

Вызвать панель **Debug** (Отладка) на экран можно посредством команды меню **View / Toolbars / Debug** (Вид / Панели инструментов / Отладка).

В Mathcad 13 появились следующие функции отладки:

- `trace ([format_string],x])` — функция, осуществляющая вывод наблюдаемых переменных в окно отладки;
- `pause ([format_string],x])` — функция, осуществляющая приостановку процесса вычислений с одновременным выводом наблюдаемых переменных в окно отладки,

где:

- `format_string` — строка, определяющая формат вывода (*см. пример ниже*);
- `x` — наблюдаемая переменная (или список переменных через запятую).

Пример применения функции `trace` в документе приведен на рис. 1.44. Каждое обращение к функции `trace` приводит к появлению в окне **Trace Window** (Окно отладки) соответствующего числа. Использование первого строкового аргумента функции `trace` позволяет задать формат вывода и снабдить его соответствующими комментариями.

> ***Внимание!***
>
> Помните о том, что опция отладки (т. е. прерывание вычислений и вывод наблюдаемых величин в окно отладки) будет работать только, если кнопка **Toggle Debugging** (Включить отладку) на панели **Debug** (Отладка) находится в нажатом состоянии.

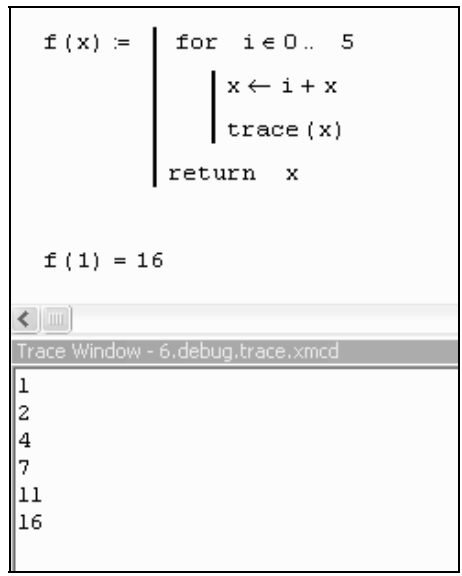



**Рис. 1.44.** Отслеживание значений переменных при помощи окна **Trace Window**

Организация точек останова вычислений в теле документа при помощи функции `pause` иллюстрируется рис. 1.45. Продолжить вычисления или пре-

рвать их можно нажатием кнопки **Resume** (Продолжить) или **Interrupt** (Прервать) соответственно на панели **Debug** (Отладка).

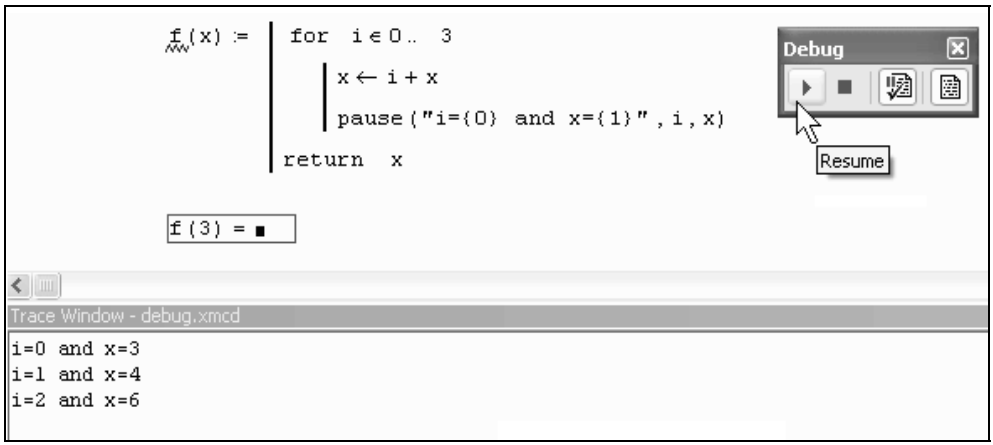



**Рис. 1.45.** Организация прерывания вычислений

Содержимое окна **Trace Window** (Окно отладки) можно скопировать в буфер обмена или попросту уничтожить (для того, чтобы возобновить затем процесс отладки "с чистого листа").

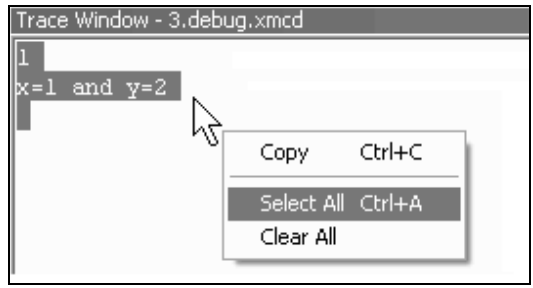



**Рис. 1.46.** Управление содержимым окна **Trace Window**

Для этого следует вызвать из области окна **Trace Window** (Окно отладки) контекстное меню и выбрать в нем соответствующую команду (рис. 1.46):

- **Copy** (Копировать) — для копирования выделенного фрагмента в буфер обмена;

- **Select all** (Выделить все) — для выделения всего содержимого окна отладки;
- **Clear all** (Удалить все) — для очистки окна отладки.

## 1.5.3. Обработка ошибок в программных модулях

Программирование в Mathcad позволяет осуществлять дополнительную обработку ошибок. Если пользователь предполагает, что выполнение кода в каком-либо месте программного модуля способно вызвать ошибку (например, деление на ноль), то эту ошибку можно перехватить с помощью оператора `on error`. Чтобы вставить его в программу, надо поместить линии ввода в ней в нужное положение и нажать кнопку с именем оператора `on error` на панели **Programming** (Программирование). В результате появится строка с двумя местозаполнителями и оператором `on error` посередине. В правом местозаполнителе следует ввести выражение, которое должно выполняться в данной строке программы. В левом — выражение, которое будет выполнено вместо правого выражения, если при выполнении последнего возникнет ошибка.

Приведем пример применения оператора `on error` (листинг 1.24) в программном модуле, который рассчитывает функцию обратного числа значению `n`. Если `n≠0`, то и присвоенное значение `z≠0`, поэтому в последней строке программы выполняется правое выражение расчета `1/z`. Так происходит при расчете `f(-2)`. Если попытаться вычислить `f(0)`, как в конце листинга, то выполнение программы, заложенной в `f(n)`, вызовет ошибку деления на ноль в последней строке программы. Соответственно, вместо выражения справа от оператора `on error` будет выполнено левое выражение, присваивающее функции `f(n)` строковое значение `"user error: can't divide by zero"` (пользовательская ошибка: деление на ноль невозможно). Конечно, этой строке можно присвоить и текст на русском языке.

**Листинг 1.24. Перехват ошибки деления на ноль**

```
f(n) := | z ← n
        |
        | "user error: can't divide by zero"   on error 1/z

f(-2) → -1/2

f(0) = "user error: can't divide by zero"
```

Оператор перехвата ошибок удобно применять в комбинации со встроенной функцией `error(S)`. Она приводит к генерации ошибки в обычной для Mathcad форме с сообщением `S`. Пример усовершенствования листинга 1.24 для такого стиля обработки ошибки деления на ноль показан на рис. 1.47. Обратите внимание, что сделанные изменения свелись к помещению текста сообщения об ошибке в аргумент функции `error`.

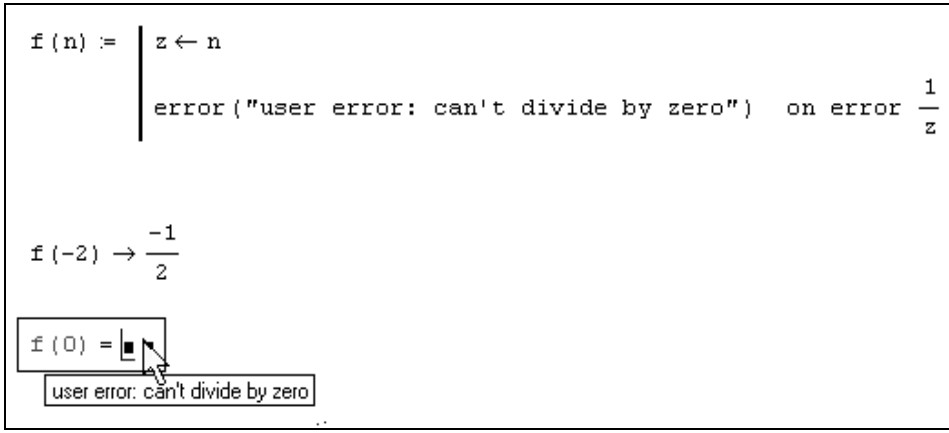



**Рис. 1.47.** Перехват ошибки деления на ноль

## 1.5.4. Текст и комментарии

Документы Mathcad могут содержать текстовые объекты, а также разного рода комментарии и примечания.

### Текстовые блоки

Для того чтобы ввести текст непосредственно в рабочую область документа Mathcad, достаточно непосредственно перед началом ввода текста нажать клавишу <">. В результате в месте расположения курсора ввода появится область с характерным выделением, обозначающая, что ее содержимое не будет восприниматься процессором Mathcad в качестве формул, а станет простым текстовым блоком (рис. 1.48, фразы "Рассмотрим квадратное уравнение" и "Детерминант"). Редактировать атрибуты текста в пределах блоков можно стандартными для текстовых редакторов средствами панели **Formatting** (Форматирование).

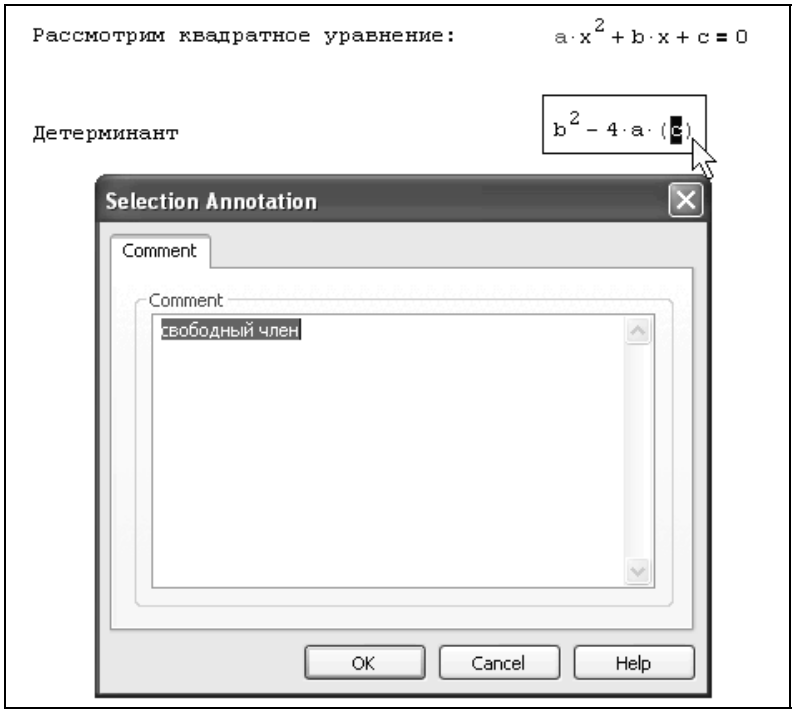



**Рис. 1.48.** Текст в рабочей области документа и комментарий к фрагменту формулы

## ⑫ Комментарии

Несколько нововведений, связанных с интерфейсом, помогут вам разрабатывать документы Mathcad с большим комфортом. Все они представляют собой дополнительные опции вставки в документы комментариев различного рода. Предусмотрено несколько типов комментариев, называемых *примечаниями* (annotation) и *метаданными* (metadata).

- Комментарии к файлу документа как единому целому, облегчающие его идентификацию как при работе в Mathcad, так и при поиске и отборе файлов средствами ОС Windows. Для добавления и редактирования комментариев ко всему документу выполните команду **File / Properties** (Файл / Свойства) и в открывшемся диалоговом окне установите стандартные свойства для данного файла (заголовок, автор, комментарии, ключевые слова).
- Примечания к отдельным выражениям, представляющие собой обычный текст небольшого размера. Для их добавления выделите выражение, вызо-

вите контекстное меню и выберите в нем пункт **Annotate Selection** (Добавить примечание к выделенному фрагменту). В открывшемся диалоге теперь можно ввести текст примечания, который впоследствии станет доступным по команде контекстного меню **View / Edit Annotation** (Просмотр / Правка примечания). Части формул, которые снабжены примечаниями, выделяются (при установке линий ввода в пределы выражения) дополнительными скобками зеленого цвета (как показано на рис. 1.48 для выделенной переменной).

> **Примечание**
>
> Отключить выделение примечаний можно командой верхнего меню **View / Annotations** (Вид / Примечания), а поменять назначенный им цвет — при помощи команды **Format / Color / Annotations** (Формат / Цвет / Примечания).

- Комментарии (метаданные) к отдельным элементам формул (переменным, функциям, выражениям), позволяющие задать для них несколько параметров. Для их создания выделите желаемую часть формулы, вызовите контекстное меню, выберите в нем пункт **Properties** (Свойства) и перейдите в открывшемся диалоге к вкладке **Custom** (Дополнительно). При помощи группы раскрывающихся списков можно добавить параметры разного вида и установить для них определенное значение того или иного типа (текст, число, дату, да/нет).
- Ключевые слова в документе (для глоссария), разметка которых производится в том же диалоге **Properties** (Свойства) на вкладке.

# 1.6. Управление файлами документов

Расчеты Mathcad организуются стандартным для Windows-приложений образом: в виде *документов* Mathcad, изначально пустых, на которые можно добавлять формулы и текст. Каждый документ представляет собой независимую серию математических расчетов и сохраняется в отдельном файле. Документ является одновременно и листингом Mathcad-программы, и результатом исполнения этой программы, получающимся при ее выполнении, и отчетом, пригодным для распечатки на принтере или публикации в Web.

## 1.6.1. Создание нового документа

Для того чтобы создать новый пустой документ, уже работая в Mathcad, следует выполнить одно из трех эквивалентных действий:

- одновременное нажатие клавиш <Ctrl>+<N>;

- нажатие кнопки **New** (Создать) на панели инструментов;
- ввод команды верхнего меню **File / New** (Файл / Создать).

Кнопка **New** (Создать) на стандартной панели состоит из двух частей (рис. 1.49). При нажатии ее левой половины (значка в виде чистого листа), как и при наборе комбинации клавиш <Ctrl>+<N>, создается новый пустой документ. Если же нажать правую часть (маленькую стрелку), это приведет к появлению выпадающего списка *шаблонов* нового документа.

Для создания пустого документа необходимо выделить в выпадающем списке (если вы используете панель инструментов) или диалоговом окне **New** (Создать) (если вы действуете через меню) пункт **Blank Worksheet** (Пустой документ) и нажать кнопку **OK**. В результате в окне **Mathcad** появляется пустой документ с условным названием **Untitled:2**, или **Untitled:3** и т. д., в зависимости от того, какой по счету новый документ создается

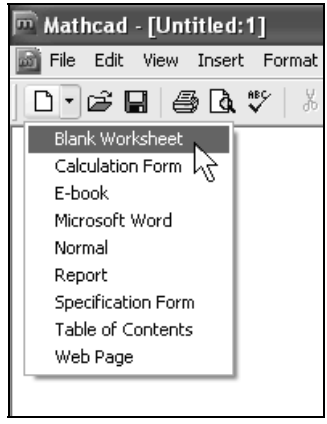



**Рис. 1.49.** Создание документа на основе шаблона

Помимо начала работы с пустым документом, можно использовать один из предустановленных шаблонов расчетов, который наилучшим образом подходит для вашей задачи. *Шаблон* (template) — это заготовка нового документа с введенными формулами, графиками, текстом, включая разметку, форматирование, выбор по умолчанию режима вычислений и т. д. Помимо пустого шаблона, который предназначен для начала редактирования документа "с чистого листа", в Mathcad имеется несколько предустановленных шаблонов, список которых отображен на рис. 1.49. Для создания нового документа на основе одного из этих шаблонов достаточно просто выбрать в списке нуж-

ный вам шаблон и начать работу с появившимся документом, на котором уже будут изначально размещены характерные для того или иного шаблона элементы оформления.

## 1.6.2. Открытие существующего документа

Чтобы открыть существующий документ для редактирования, выполните команду **File / Open** (Файл / Открыть) или нажмите комбинацию клавиш <Ctrl>+<O> (или кнопку **Open** (Открыть) на стандартной панели инструментов). В диалоговом окне **Open** (Открыть) выберите файл и нажмите кнопку **OK**. Таким способом можно открывать документы не только с дисков вашего компьютера, но и из локальной сети и даже с удаленных серверов Интернета.

Открыть файл можно и в обозревателе Windows, щелкнув дважды на его имени с расширением xmcd или mcd.

## 1.6.3. Сохранение документа

Для того чтобы сохранить документ в формате Mathcad, выберите **File / Save** (Файл / Сохранить), либо нажмите комбинацию клавиш <Ctrl>+<S> или кнопку **Save** (Сохранить) на стандартной панели инструментов. Если созданный документ сохраняется впервые, на экран будет выведено диалоговое окно **Save As** (Сохранить как), в котором потребуется определить его имя (рис. 1.50).

Начиная с версии 12, изменен формат файлов, использующийся по умолчанию для хранения документов Mathcad. Теперь они сохраняются в формате XMCD, являющемся разновидностью текстовой XML-разметки (и следующим шагом по сравнению с форматом MathML, примененным в версии 2001). Преимущества этого нововведения практически не заметны для рядового пользователя, процедура сохранения для которого осталась практически неизменной. Применение XML-формата оправдано, во-первых, тем, что он становится общеупотребительным для целого ряда приложений и данных самого различного типа. Во-вторых, удобство XML-файлов заключается в возможности использовать для просмотра и манипуляций с Mathcad-документами другие (предусматривающие это) приложения, например, MathSoft Calculation Management Framework, Microsoft Sharepoint и т. п. Их можно также просматривать и редактировать "вручную", в любом текстовом редакторе. Наконец, XMCD-файлы более надежно конвертируются в файлы других математических форматов и HTML.

Помимо формата XMCD поддерживаются форматы файлов MCD прежних версий Mathcad — от 2001i до 12, а также XML-формат со сжатием данных,

называемый XMCDZ. В Mathcad 12 и 13 сохранены опции экспорта в Web-страницы (HTML) и текстовые RTF-файлы, записывающие информацию о разметке документа и успешно читаемые такими редакторами, как Microsoft Word.

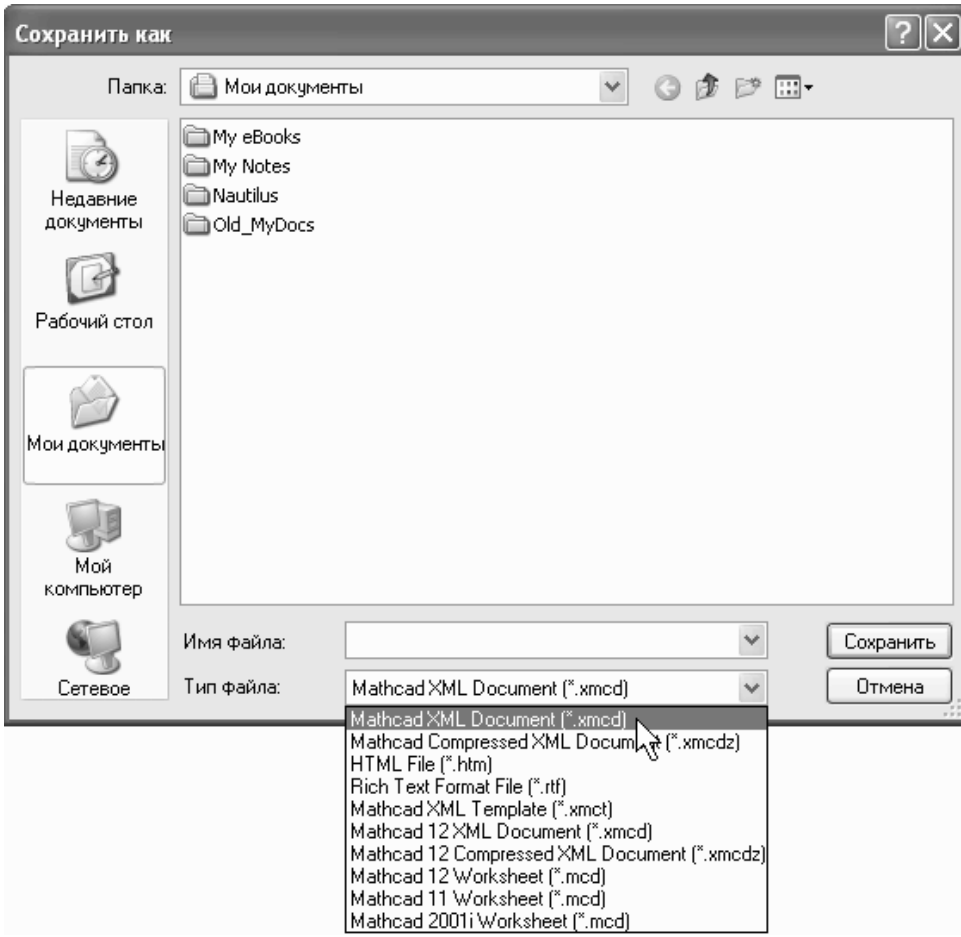



**Рис. 1.50.** Сохранение документа

Чтобы переименовать документ, сохраните его под другим именем с помощью команды **File / Save As** (Файл / Сохранить как) и появляющегося в результате диалогового окна **Save As** (Сохранить как). В этом случае файл со старым названием не изменяется.

На рис. 1.50 показан раскрывающийся список с возможными форматами сохраняемых файлов:

- **Mathcad XML Document (*.xmcd)** — последний и наиболее современный формат на основе XML-разметки, используется по умолчанию;
- **Mathcad Compressed XML Document (*.xmcdz)** — тот же XML-формат с дополнительной компрессией файла;
- **HTML (*.htm)** — формат Web-страницы, полностью сохраняющий информацию о расчетах и позволяющий впоследствии открывать их в Mathcad как обычные документы;
- **Rich Text Format (*.rtf)** — сохраняйте файлы в этом формате только для дальнейшего редактирования в текстовых редакторах с целью создания отчетов. В частности, сохранив документ в RTF-файле, можно загрузить его в Microsoft Word, OpenOffice.org или другой текстовый процессор, который поддерживает этот формат;
- **Mathcad XML Template (*.xmct)** — формат шаблона на основе XML-разметки;
- **Mathcad 12 XML Document (*.xmcd)** — формат на основе XML-разметки версии 12;
- **Mathcad 12 Compressed XML Document (*.xmcdz)** — тот же XML-формат (версии 12) с дополнительной компрессией файла;
- **Mathcad Template (*.mct)** — формат шаблона;
- **Mathcad 12 Worksheet (*.mcd)**, **Mathcad 11 Worksheet (*.mcd)**, **Mathcad 2001i Worksheet (*.mcd)** — форматы прежних версий Mathcad. Если вы работаете одновременно с несколькими версиями Mathcad (например, решаете с другими разработчиками общую задачу), то сохраняйте файлы в наиболее раннем формате из тех, с которыми приходится иметь дело. Однако помните, что возможности прежних версий (в частности, наборы встроенных функций) ограничены по сравнению с более поздними версиями Mathcad, поэтому некоторые из них будут недоступны.

### *Примечание*

Начиная с версии Mathcad 13 разработчики предусмотрели стандартную опцию *автоматического сохранения* (или, короче, *автосохранения*) файлов документов (в фоновом режиме). Если включить данную опцию на вкладке **Save** (Сохранение) диалогового окна **Preferences** (Настройки), то Mathcad будет автоматически сохранять в определенном месте на диске резервные копии документов. В случае возникновения нештатной ситуации (аварийного завершения работы программы) при последующем запуске Mathcad и обращении к несо-

храненным файлам будет выдано диалоговое окно с предложением вернуться к их автоматически сохраненным резервным копиям.

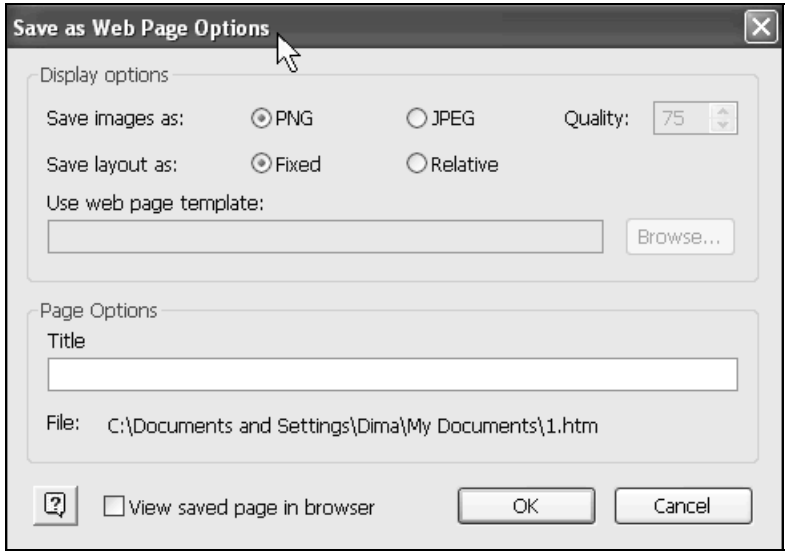



**Рис. 1.51.** Настройки HTML-экспорта

Для того чтобы экспортировать документ в файл формата HTML, выберите команду **File / Save As** (Файл / Сохранить как). После того как в открывшемся диалоговом окне вы определите имя файла, автоматически откроется еще одно диалоговое окно **Save as Web Page Options** (Опции HTML-сохранения), в котором можно настроить основные параметры текущего экспорта (рис. 1.51). Пара переключателей **Save images as** (Сохранять рисунки как) позволяет выбрать формат вспомогательных графических файлов для сохранения рисунков. Второй переключатель определяет стиль разметки HTML-документа: **Fixed** (Фиксированная) или **Relative** (Относительная).

Впоследствии расчеты Mathcad, сохраненные в виде Web-страницы, можно просматривать в окне любого браузера как стандартный гипертекстовый документ.

## 1.6.4. Расчеты в документах

Как мы уже говорили, документ Mathcad — это в полном смысле слова компьютерная программа, а сама система Mathcad — настоящая система программирования, правда, ориентированная на математика, а не на профессио-

нального программиста. Большинство других сред программирования (знакомых читателю по реализации таких языков, как Си, Фортран и т. п.) разделяют редактирование кода программ и их выполнение, которое можно вызвать предназначенными для этого командами. В Mathcad и код программы, и результат их выполнения объединены в документе. Тем не менее функции редактирования формул и их расчеты разделены, и пользователь имеет возможность управлять всеми важнейшими опциями вычислений.

Вообще говоря, имеются два режима вычислений:

- *автоматический режим* (automatic mode) — все вычисления выполняются автоматически по мере ввода формул;
- *ручной режим* (manual mode) — старт вычислений каждой формулы или всего документа производится пользователем.

Режим вычислений можно выбрать с помощью команды **Tools / Calculate / Automatic Calculation** (Сервис / Пересчитать / Считать автоматически), как показано на рис. 1.52. Если в этой строке меню установлен флажок проверки, значит, включен автоматический режим, если флажка нет, то редактируется документ в ручном режиме вычислений. Чтобы сменить режим, просто выберите этот пункт меню (например, нажав кнопку мыши в ситуации, показанной на рис. 1.52, включите ручной режим).

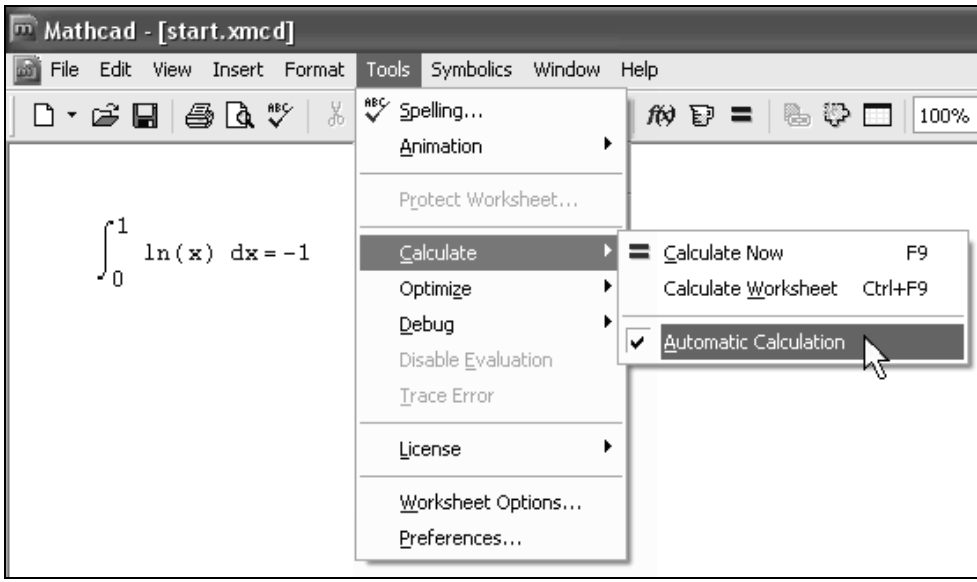



**Рис. 1.52.** Выбор режима вычислений

Mathcad осуществляет вычисления документа, как это принято в большинстве сред программирования: сверху вниз и слева направо. Чтобы прервать затянувшийся процесс вычислений, нажмите клавишу <Esc>. Появится диалоговое окно, в котором нужно подтвердить прерывание вычислений (нажать кнопку **OK**). В этом случае выражения, которые Mathcad не успел вычислить, будут помечены в документе красным цветом. Прерванные вычисления возобновляются нажатием клавиши <F9> или командой **Tools / Calculate / Calculate Now** (Математика / Пересчитать / Пересчитать).

Установки режимов вычислений для всего документа сведены на вкладке **Calculations** (Вычисления) диалогового окна **Worksheet Options** (Опции документа), вызываемого с помощью команды **Tools / Worksheet Options** (Сервис / Опции документа). Опции сгруппированы на нескольких вкладках (рис. 1.53), а их набор изменялся от версии к версии Mathcad:

- **Unit System** (Система измерения) — выбор системы, в которой отображаются размерные величины (СИ, СГС и т. д., в том числе пользовательская система);

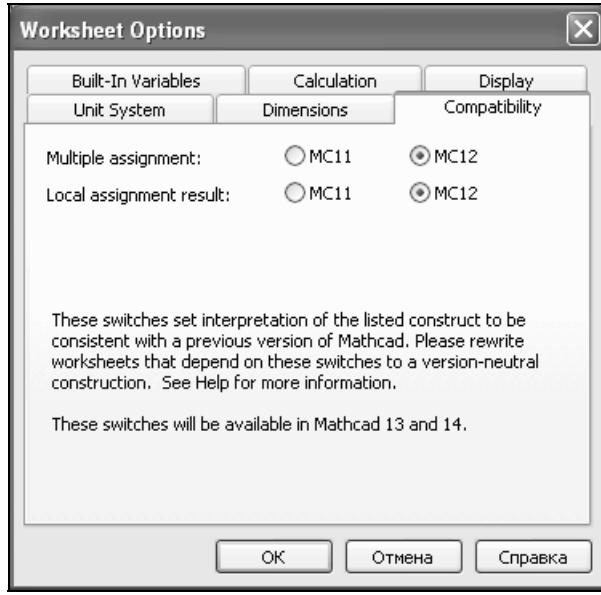



**Рис. 1.53.** Управление режимом вычислений документа в диалоговом окне **Worksheet Options**

- **Dimension** (Размерность) — задание набора единиц измерения основных размерных величин;
- **Compatibility** (Совместимость) — включение опций работы математического процессора по обработке команд присваивания в стиле версий Mathcad 11 или 12;
- **Built-In variables** (Встроенные переменные) — определение значений системных констант, таких, как `ORIGIN`, `TOL` или `CTOL`;
- **Calculating** (Вычисление) — выбор опций режима вычислений (например, результата деления 0/0, предварительной проверки матриц на сингулярность и др.);
- **Display** (Отображение) — выбор установок по умолчанию для символов операций, допускающих разное отображение на экране (например, присваивания, численного и символьного вывода, умножения).

# Глава 2

# Алгебраические вычисления

В данной главе рассматриваются простые вычисления, осуществляемые в Mathcad. Во-первых, приведено описание имеющихся встроенных операторов (*см. разд. 2.1*) и функций (*см. разд. 2.2*), при помощи которых можно рассчитать значение алгебраических выражений, построить графики и т. п. Во-вторых, составлен обзор наиболее простых символьных операций, реализующих в Mathcad аналитические преобразования для решения типичных задач алгебры. Они проводятся без применения численных методов и, соответственно, без погрешностей вычислений (*см. разд. 2.3*).

## 2.1. Операторы

Каждый оператор в Mathcad обозначает некоторое математическое действие в виде символа. В полном согласии с терминологией, принятой в математике, ряд действий (например, сложение, деление, транспонирование матрицы и т. п.) реализован в Mathcad в виде встроенных операторов, а другие действия (например, `sin`, `erf` и т. п.) — в виде встроенных функций. Каждый оператор действует на одно или два числа (переменную или функцию), которые называют *операндами*. Если в момент вставки оператора одного или обоих операндов не хватает, то недостающие операнды будут отображены в виде местозаполнителей. Символ любого оператора в нужное место документа вводится одним из двух основных способов:

- нажатием соответствующей клавиши (или сочетания клавиш) на клавиатуре;
- нажатием указателем мыши соответствующей кнопки на одной из математических панелей инструментов.

Напомним, что большинство математических панелей содержат сгруппированные по смыслу математические операторы, а вызвать эти панели на экран можно нажатием соответствующей кнопки на панели **Math** (Математика).

> ***Примечание***
>
> Везде в этом разделе будем рассматривать только второй способ вставки оператора. Те же, кто предпочитает использовать клавиатуру, найдут перечень горячих клавиш в *приложении 2*.

Выше мы рассмотрели особенности применения трех операторов: присваивания (*см. разд. 1.2.3*), численного и символьного вывода (*см. разд. 1.2.1*). Разберем в данном разделе более подробно действие этих и других операторов Mathcad.

## 2.1.1. Арифметические операторы

Операторы, обозначающие основные арифметические действия, вводятся с панели **Calculator** (Калькулятор), показанной на рис. 2.1:

- сложение и вычитание: + / –;
- умножение и деление: · / ÷ ;
- факториал: **!**;
- модуль числа: |**x**|;
- квадратный корень: $\sqrt{\ }$ ;
- корень *n*-й степени: $\sqrt[n]{\ }$ ;
- возведение x в степень y: $\mathbf{x}^{\mathbf{y}}$ ;
- изменение приоритета: скобки;
- численный вывод: = (все листинги).

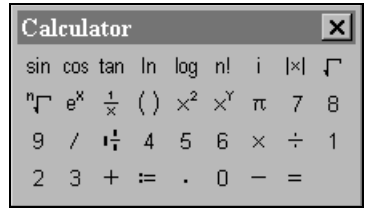



**Рис. 2.1.** Панель **Calculator**

## 2.1.2. Вычислительные операторы

Вычислительные операторы вставляются в документы при помощи панели инструментов **Calculus** (Вычисления) (рис. 2.2). При нажатии любой из кнопок в документе появляется символ соответствующего математического действия, снабженный несколькими местозаполнителями. Количество и расположение местозаполнителей определяется типом оператора и в точности соответствует их общепринятой математической записи. Например, при вставке оператора суммы (см. рис. 2.2) необходимо задать четыре величины: переменную, по которой надо произвести суммирование, нижний и верхний пределы, а также само выражение, которое будет стоять под знаком суммы. Для того чтобы вычислить неопределенный интеграл, следует заполнить два местозаполнителя: подынтегрального выражения и переменной интегрирования и т. д.

После ввода какого-либо вычислительного оператора имеется возможность вычислить его значение либо численно, нажатием клавиши <=>, либо аналитически, с помощью оператора символьного вывода.

***Примечание***

Дифференцированию и интегрированию, как более сложным операциям, посвящены отдельные главы этой книги (*см. главы 3 и 4 соответственно*). Суммирование и вычисление предела рассматривается ниже в этой главе (*см. соответственно разд. 2.3.8 и 2.3.11*).

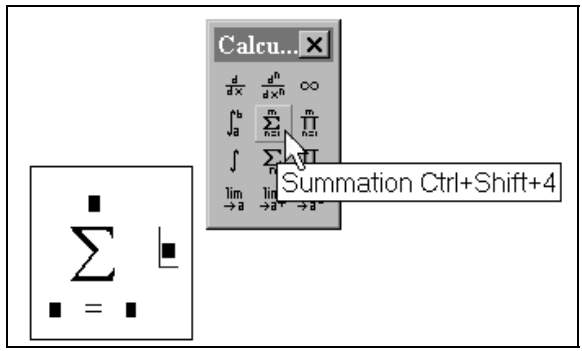



**Рис. 2.2.** Вставка оператора суммирования

## 2.1.3. Логические операторы

Результатом действия логических, или булевых, операторов являются только числа 1 (если логическое выражение, записанное с их помощью, истинно)

или 0 (если логическое выражение ложно). Чтобы вычислить значение логического выражения, например 1=1 (рис. 2.3):

1. Вставьте с панели **Boolean** (Булевы операторы) соответствующий оператор =.
2. В появившиеся местозаполнители вставьте операнды (две единицы).
3. Нажмите клавишу <=>, чтобы получить ответ.

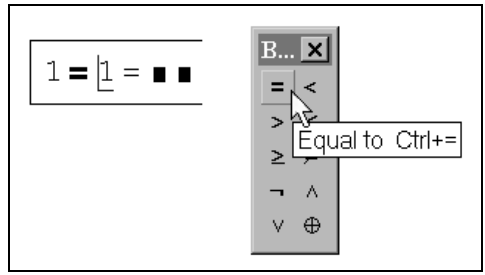



**Рис. 2.3.** Вставка логического оператора

Получается абсурдное на первый взгляд выражение 1=1=1. Однако на самом деле все правильно. Слева от оператора вывода записано логическое выражение 1=1 (обратите внимание, что логический знак равенства выглядит по-другому, нежели обычный), которое является истинным. Поэтому значение данного выражения равно 1, что и показано справа от знака равенства.

Перечислим логические операторы:

- больше (**Greater Than**) x>y;
- меньше (**Less Than**) x<y;
- больше или равно (**Greater Than or Equal**) x≥y;
- меньше или равно (**Less Than or Equal**) x≤y;
- равно (**Equal**) x=y;
- не равно (**Not Equal to**) x≠y;
- и (**And**) x∧y;
- или (**Or**) x∨y;
- исключающее или (**Exclusive or**) x⊕y;
- отрицание (**Not**) ¬x.

***Примечание***

Операнды в логических выражениях могут быть любыми числами. Однако если оператор по смыслу применим только к 0 и 1, то любое неравное нулю число по умолчанию принимается равным 1. Но в результате все равно может получиться либо 0, либо 1. Например, ¬(-0.33)=0.

Примеры действия логических операторов приведены в листингах 2.1 и 2.2.

***Примечание***

Логические операторы чрезвычайно важны при записи в редакторе Mathcad подлежащих решению алгебраических уравнений и неравенств (*см. главу 5*).

**Листинг 2.1. Операторы сравнения**

```
2 = 3 = 0        5 > 1 = 1        3 > 3 = 0
7 = 7 = 1        3 < ∞ = 1        3 ≥ 3 = 1
0 ≠ 0 = 0
```

**Листинг 2.2. Булевы операторы**

```
1 ∨ 0 = 1        1 ∧ 0 = 0        1 ⊕ 0 = 1        ¬1 = 0
0 ∨ 0 = 0        0 ∧ 0 = 0        0 ⊕ 0 = 0        ¬0 = 1
1 ∨ 1 = 1        1 ∧ 1 = 1        1 ⊕ 1 = 0
```

## 2.1.4. Матричные операторы

Матричные операторы предназначены для совершения различных действий над векторами и матрицами. Поскольку большинство из них реализует численные алгоритмы, о них будет подробно рассказано в разделах, посвященных линейной алгебре (*см. главу 7*). Остановимся в данном разделе лишь на вопросе вставки матриц в документы Mathcad.

Самый простой и наглядный способ создания вектора или матрицы заключается в следующем:

1. Нажмите кнопку **Matrix or Vector** (Матрица или вектор) на панели **Matrix** (Матрица), либо клавиши <Ctrl>+<M>, либо выберите пункт меню **Insert / Matrix** (Вставка / Матрица).

2. В диалоговом окне **Insert Matrix** (Вставка матрицы) (рис. 2.4) задайте целое число столбцов и строк матрицы, которую хотите создать. Например, для создания вектора 3×1 введите показанные на рис. 2.4 значения.
3. Нажмите кнопку **OK** или **Insert** (Вставить) — в результате в документ будет вставлена заготовка матрицы с определенным числом строк и столбцов (рис. 2.5).
4. Введите значения в местозаполнители элементов матрицы. Переходить от одного элемента матрицы к другому можно с помощью указателя мыши либо клавиш со стрелками.

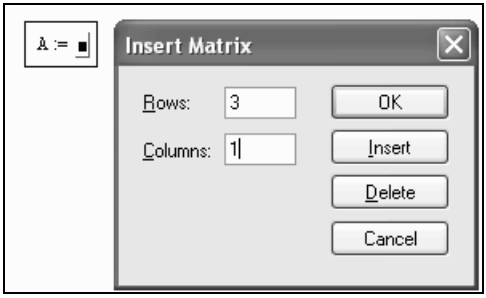



**Рис. 2.4.** Создание матрицы

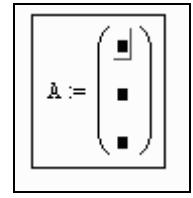



**Рис. 2.5.** Результат создания матрицы

Добавление в уже созданную матрицу строк или столбцов производится точно так же:

1. Выделите линиями ввода элемент матрицы, правее и ниже которого будет осуществлена вставка столбцов и (или) строк.
2. Вставьте в него матрицу, как было описано выше. При этом допускается задание числа столбцов или строк равным нулю.
3. Заполните местозаполнители недостающих элементов матрицы.

> ***Примечание***
>
> Определить новую матрицу можно и по-другому, например (впервые в документе), присвоив определенное значение ее некоторому элементу, например, $A_{3,3}:=1$, либо импортировав в матрицу данные из внешнего файла (*см. разд. 13.3*).

## 2.1.5. Операторы выражения

Вычислительные операторы сгруппированы на панели **Evaluation** (Вычисления) (*см. разд. 1.2.1*).

Перечислим их еще раз (без дополнительных комментариев):

- численный вывод (**Evaluate Numerically**) = ;
- символьный (аналитический) вывод (**Evaluate Symbolically**) → ;
- присваивание (**Definition**) := ;
- глобальное присваивание (**Global Definition**) ≡.

## 2.2. Функции

Mathcad содержит огромное количество встроенных функций. Некоторые из них просто рассчитывают определенное значение, а некоторые реализуют сложные численные алгоритмы. Учитывая, что большинство стандартных алгебраических функций Mathcad имеют общепринятую математическую форму, мы не станем приводить в данной главе все имеющиеся встроенные функции, а лишь перечислим их основные типы и приведем несколько характерных примеров, ограничившись минимальными комментариями. Полный перечень встроенных функций вы найдете в конце книги (*см. приложение 3*).

> ***Примечание***
>
> Напомним также (*см. главу 1*), что вставлять в документ не очень знакомую функцию легче всего, пользуясь диалоговым окном **Insert Function** (Вставить функцию), которое вызывается нажатием кнопки с надписью ***f(x)*** на стандартной панели инструментов. В этом диалоге функции разбиты на несколько групп, поэтому несложно выбрать из них нужную. При выделении какой-либо группы в левом списке упомянутого диалога справа обнаруживается список функций, принадлежащих этой группе.

### 2.2.1. Элементарные функции

Перечислим хорошо известные группы стандатртных функций (примеры некоторых из них приведены на рис. 2.6—2.8).

- **Exponential and logarithmic functions** (Логарифмы и экспонента) (рис. 2.6).
- **Complex** (Комплексные) (листинг 2.3).
- **Trigonometric** (Тригонометрические) (листинги 2.4 и 2.5).
- **Inverse trig** (Обратные тригонометрические) (листинги 2.4 и 2.5).
- **Hyperbolic** (Гиперболические) (листинг 2.6 и рис. 2.7).
- **Inverse hyperbolic** (Обратные гиперболические).
- **Sinc** (Sinc-функция) (рис. 2.8).

### Примечание

Sinc-функция появилась в версии Mathcad 11.

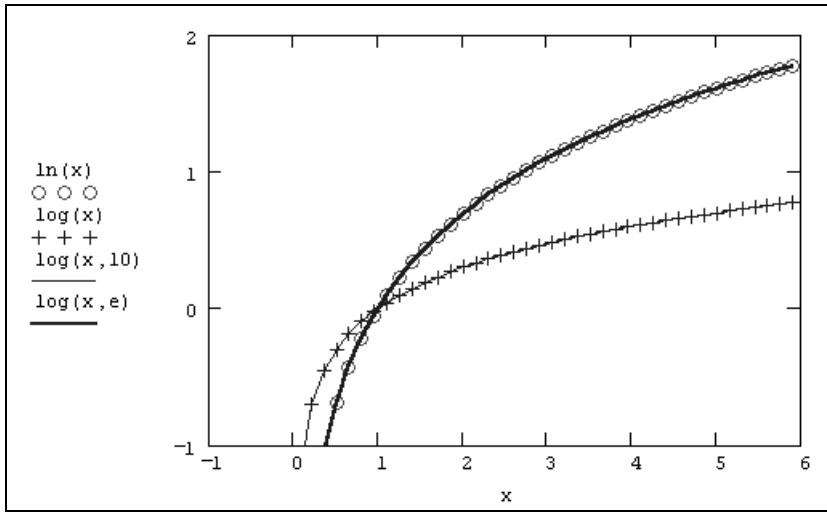



**Рис. 2.6.** Логарифмические функции

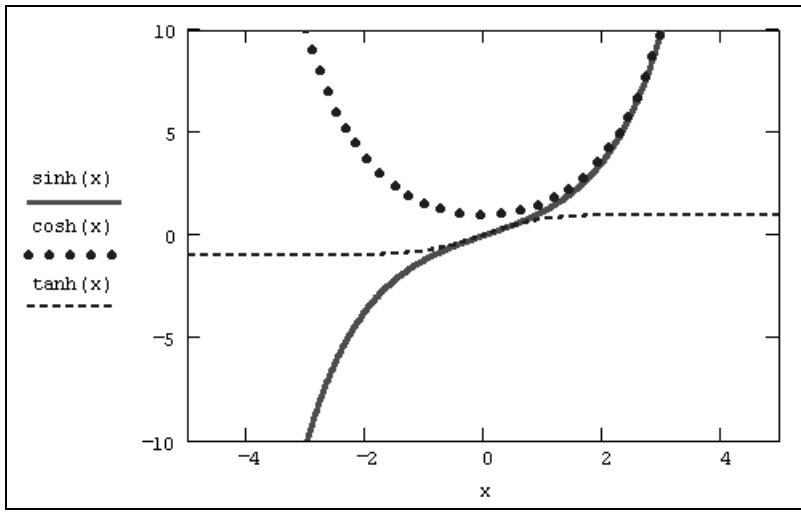
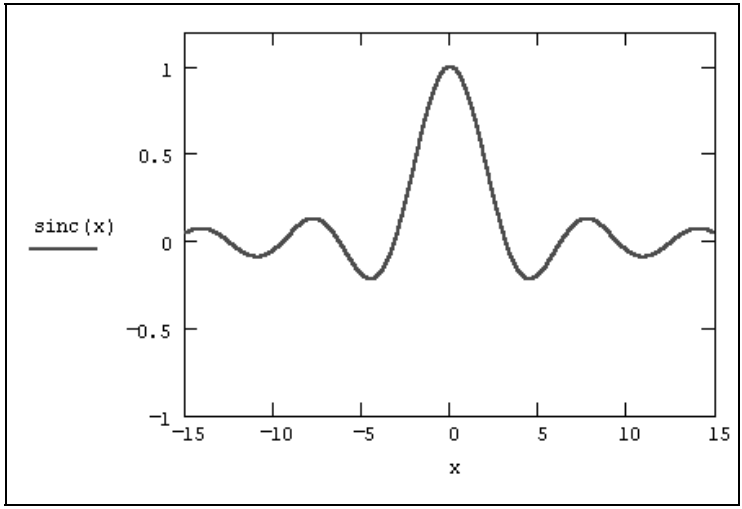



**Рис. 2.7.** Основные гиперболические функции




(11) **Рис. 2.8.** Sinc-функция

## Листинг 2.3. Некоторые функции работы с комплексными числами

$\mathrm{Re}(3.9 + 2.4i) = 3.9$ $\mathrm{Im}(3.9 + 2.4i) = 2.4$

$\left|1.7 \cdot e^{0.1i}\right| = 1.7$ $\arg\left(1.7 \cdot e^{0.1i}\right) = 0.1$

## Листинг 2.4. Примеры тригонометрических функций

$\sin(0.5) = 0.479$ $\dfrac{1}{\csc(0.5)} = 0.479$

$\mathrm{asin}(0.479) = 0.5$ $\mathrm{acsc}\left(\dfrac{1}{0.479}\right) = 0.5$

$\mathrm{acos}(0.682) \cdot \dfrac{180}{\pi} = 47$

$z := 47$

$\cos\left(\dfrac{\pi \cdot z}{180}\right) = 0.682$

**Листинг 2.5. Примеры расчета угла между прямой и осью ox**

```
atan2(1, 1) = 0.785        atan2(−1, −1) = −2.356
angle(1, 1) = 0.785        angle(−1, −1) = 3.927
```

**Листинг 2.6. Пример гиперболических функций**

```
z := 1.27
```

$$\frac{e^{z} + e^{-z}}{2} = 1.921$$

```
cosh(z) = 1.921
acosh(1.921) = 1.27
```

## 2.2.2. Вспомогательные функции

Кроме перечисленных, Mathcad включает целый ряд вспомогательных функций, во множестве ситуаций облегчающих вычисления.

- **Discontinuous functions** (Разрывные функции).
- **Round-off and truncation** (Сокращения и округления) (листинг 2.7).
- **Sorting** (Сортировки).
- **Strings** (Строковые).
- **Finance functions** (Финансовые).
- **Coordinate transform** (Преобразования координат) (листинг 2.8).
- **Conditional** (Условия) (листинг 2.9).
- **Expression type** (Типа выражения).

**Листинг 2.7. Функции сокращения и округления**

```
ceil(3.7) = 4      floor(3.7) = 3      trunc(3.7) = 3
ceil(−3.7) = −3    floor(−3.7) = −4    trunc(−3.7) = −3
ceil(3.7 − 2.1·i) = 4 − 2i
floor(3.7 − 2.1·i) = 3 − 3i
trunc(3.7 − 2.1·i) = 3 − 2i
round(1.23456789, 0) = 1          round(12.3456789, 0) = 12
```

```
round (12.3456789 , 1) = 12.3      round (12.3456789 , −1) = 10
round (12.3456789 , 2) = 12.35     round (12.3456789 , −2) = 0
round (12.3456789 , 5) = 12.34568
round (1.2345 + 6.789i , 1) = 1.2 + 6.8i
```

### Примечание

В прежних версиях Mathcad (вплоть до 11-й включительно) функции округления и сокращения могли выдавать результат, отличный от аналитического (что было связано с принципом представления чисел). В Mathcad 12 эти функции работают более правильно, выводя точный результат округления, совпадающий с символьным.

## Листинг 2.8. Функции преобразования координат на плоскости

$$\mathrm{xy2pol}\begin{pmatrix}1\\7\end{pmatrix}=\begin{pmatrix}7.071\\1.429\end{pmatrix} \qquad \mathrm{xy2pol}(1,7)=\begin{pmatrix}7.071\\1.429\end{pmatrix}$$

$$\mathrm{pol2xy}\begin{pmatrix}7.071\\1.429\end{pmatrix}=\begin{pmatrix}0.999\\7\end{pmatrix} \qquad \mathrm{pol2xy}(7.071,1.429)=\begin{pmatrix}0.999\\7\end{pmatrix}$$

$$\mathrm{xyz2cyl}(1,1,1)=\begin{pmatrix}1.414\\0.785\\1\end{pmatrix} \qquad \mathrm{cyl2xyz}(1,\pi,3.93)=\begin{pmatrix}-1\\0\\3.93\end{pmatrix}$$

$$\mathrm{xyz2sph}(1,1,1)=\begin{pmatrix}1.732\\0.785\\0.955\end{pmatrix} \qquad \mathrm{sph2xyz}\left(\sqrt{2},\frac{\pi}{4},\frac{\pi}{2}\right)=\begin{pmatrix}1\\1\\0\end{pmatrix}$$

## Листинг 2.9. Функции знака и условия

```
sign (−4) = −1      sign (1.3) = 1
if (1 > 3, 1, 3) = 3
if (1 > 3, "Yes" , "No" ) = "No"
```

### Примечание

Разработчики Mathcad 12 восстановили встроенную функцию `until`, которая вплоть до 2000-й версии служила для включения в документы циклов без помощи программирования. Функция `until(x,y)` служит для "непрограммной" имитации цикла: если `x<0`, происходит вычисление очередного `y`, затем опять

вычисляется новое x (так или иначе зависящего от y), снова проверяется условие x<0 и т. д. Финальное значение y, при котором x становится неотрицательным, выдается в качестве результата функции.

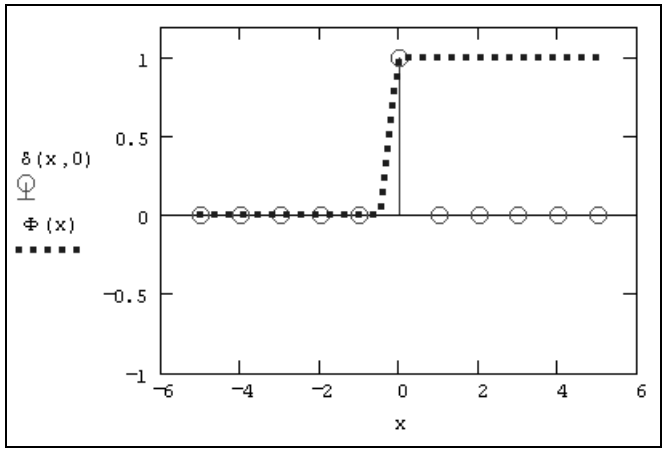



**Рис. 2.9.** Функции Хевисайда и устаревшая функция Кронекера

### ⑫ *Внимание!*

Начиная с Mathcad 12 функция (символ) Кронекера δ (рис. 2.9) изъята из списка встроенных функций.

## ⑫ 2.2.3. Функция вывода текущего времени

В версии Mathcad 12 появилась новая встроенная функция, которая иногда может быть очень полезной для хронометрирования процесса вычислений.

❒ `time(x)` — значение системной переменной текущего времени (в секундах):

- `x` — аргумент (нужен лишь для идентификации встроенной функции и не оказывает никакого влияния на результат).

Типичное использование функции требует ее двукратного вычисления: до и после расчетного фрагмента, который вы собираетесь хронометрировать (листинг 2.10). Одиночное вычисление `time(x)` выдает абсолютное значение системной переменной времени (первая строка листинга 2.10). Помните о том, что для получения значимой информации о времени расчетов необходимо пересчитать заново содержимое всего документа командой **Tools / Calculate Worksheet** (Сервис / Вычислить все).

**Листинг 2.10. Хронометрирование вычислений**

$time(0) = 1.089 \times 10^{9}$

$T := time(0)$

$i := 0 .. 10^{5}$

$v_i := \sqrt[3]{i}$

$time(1) - T = 1.542$

## 2.2.4. Спецфункции

В Mathcad встроено множество самых различных математических функций, которое пополняется от версии к версии. Большинство из них рассчитываются без привлечения специальных численных методов, но, тем не менее, играют важную роль, главным образом, в математической физике. Например, функции Бесселя по определению являются решениями различных краевых задач для некоторых обыкновенных дифференциальных уравнений (ОДУ). Конкретный вид соответствующих дифференциальных уравнений можно без труда отыскать в справочниках по спецфункциям или в справочной системе Mathcad.

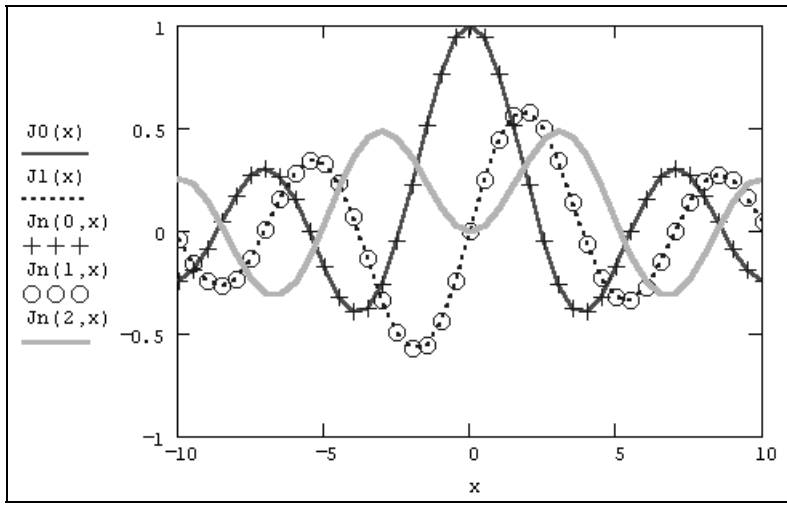
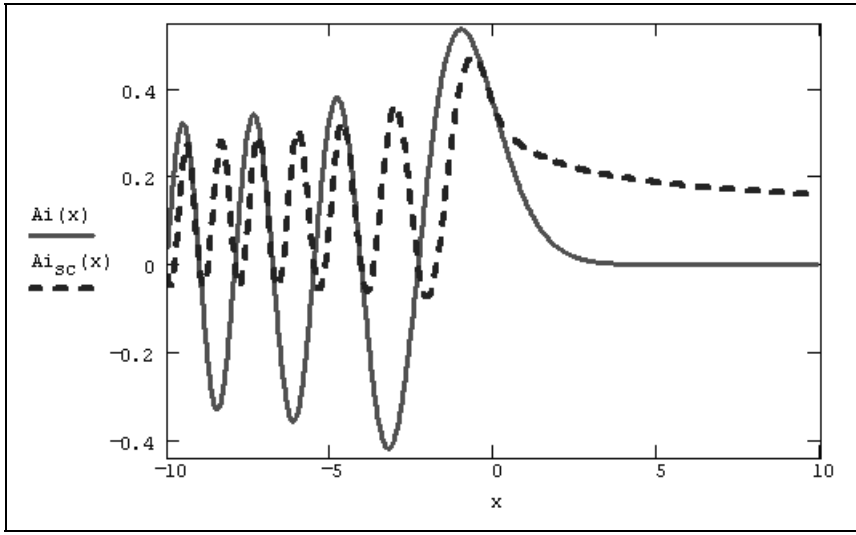



**Рис. 2.10.** Функции Бесселя первого рода




(12) **Рис. 2.11.** Функции Эйри (`Aisc` появилась в Mathcad 12)

Спецфункции в Mathcad разбиты на несколько групп:

- **Bessel** (Функции Бесселя) (рис. 2.10 и 2.11);
- **Error function and complementary error function** (Интегралы ошибок);
- **Special functions** (Остальные спецфункции).

# 2.3. Алгебраические преобразования

В этом разделе речь пойдет об алгебраических вычислениях, которые выполняются в Mathcad, главным образом, аналитически. Как ни странно, многие пользователи Mathcad не очень хорошо осведомлены об этих возможностях, тогда как они во многих ситуациях могут существенно сэкономить их время и силы по выполнению несложных, но рутинных преобразований.

## 2.3.1. О способах символьных вычислений

Символьные вычисления в Mathcad можно осуществлять в двух различных вариантах:

- с помощью команд меню;
- с помощью оператора символьного вывода →, ключевых слов символьного процессора и обычных формул (в справочной системе Mathcad этот

способ называется *символьными вычислениями в реальном времени* — live symbolic evaluation).

Первый способ более удобен, когда требуется быстро получить какой-либо аналитический результат для однократного использования, не сохраняя сам ход вычислений. Второй способ более нагляден, т. к. позволяет записывать выражения в традиционной математической форме и сохранять символьные вычисления в документах Mathcad. Кроме того, аналитические преобразования, проводимые через меню, касаются только одного, выделенного в данный момент, выражения.

### *Примечание*

В символьных вычислениях допускается использование большинства встроенных функций Mathcad, конечно, за исключением тех, которые реализуют численные методы.

Символьный процессор Mathcad умеет выполнять основные алгебраические преобразования, такие как упрощение выражений, разложение их на множители, символьное суммирование и перемножение.

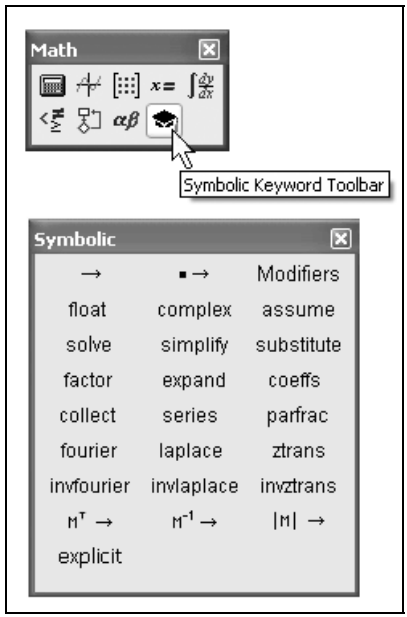



**Рис. 2.12.** Панель **Symbolic**

Для символьных вычислений при помощи команд предназначено главное меню **Symbolics** (Символика), объединяющее математические операции, которые Mathcad умеет выполнять аналитически. Для реализации второго способа применяются все средства Mathcad, пригодные для численных вычислений (например, панели **Calculator**, **Evaluation** и т. д.), и специальная математическая панель инструментов, которую можно вызвать на экран нажатием кнопки **Symbolic Keyword Toolbar** (Панель символики) на панели **Math** (Математика). На панели **Symbolic** (Символика) находятся кнопки, соответствующие специфическим командам символьных преобразований (рис. 2.12). Например, таким как разложение выражения на множители, приведение подобных слагаемых и другим операциям, которые в Mathcad нельзя проводить численно и для которых, соответственно, не предусмотрены встроенные функции.

> ***Примечание***
>
> Необходимо отметить, что приемы более сложных символьных вычислений описываются также в остальных главах данной книги в рамках рассказа о решении конкретных вычислительных задач.

## 2.3.2. Разложение выражений

Рассмотрим оба типа символьных вычислений на простом примере разложения на сомножители выражения `cos(4·x)`. В ходе операции символьного разложения, или *расширения*, раскрываются все суммы и произведения, а сложные тригонометрические зависимости разлагаются с помощью тригонометрических тождеств. Разложение выражений производится путем выбора команды **Symbolics / Expand** (Символика / Разложить) либо использованием вместе с оператором символьного вывода ключевого слова `expand`.

### Первый способ (разложение с помощью меню)

1. Введите выражение `cos(4·x)`.
2. Выделите его целиком (рис. 2.13).
3. Выберите в главном меню пункты **Symbolics / Expand** (Символика / Разложить).

После этого результат разложения выражения появится чуть ниже в виде еще одной строки (рис. 2.13).

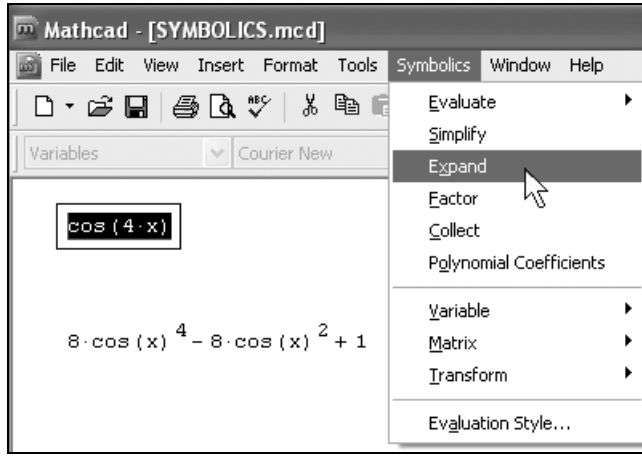



**Рис. 2.13.** Разложение выражения на множители при помощи команды меню **Symbolics / Expand**

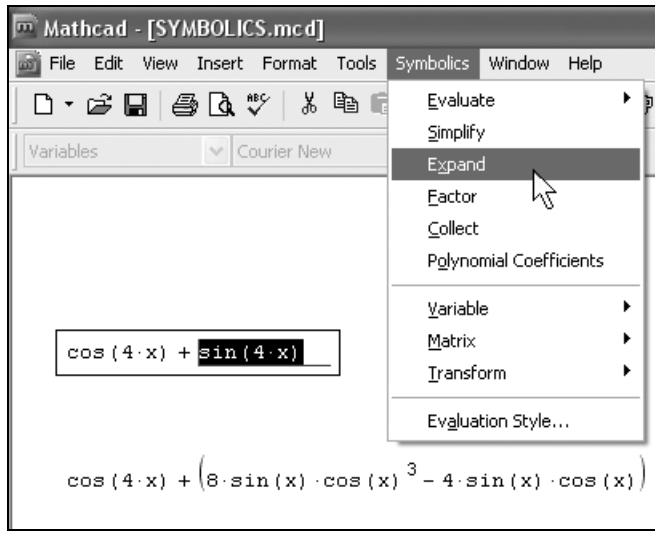



**Рис. 2.14.** Символьное разложение части выражения и его результат

### *Внимание!*

Символьные операции с помощью меню возможны лишь над каким-либо объектом (выражением, его частью или отдельной переменной). Для того чтобы правильно осуществить желаемое аналитическое преобразование, предварительно необходимо выделить тот объект, к которому оно будет относиться.

В данном случае преобразование было применено ко всему выражению cos(4·x). Если же выделить часть формулы, как показано на рис. 2.14, то соответствующее преобразование будет отнесено только к выделенной части (нижняя строка на этом рисунке).

## Второй способ (разложение с помощью оператора →)

1. Введите выражение, например, cos(4·x).
2. Нажмите кнопку **Expand** (Разложить) на панели **Symbolic** (Символика).

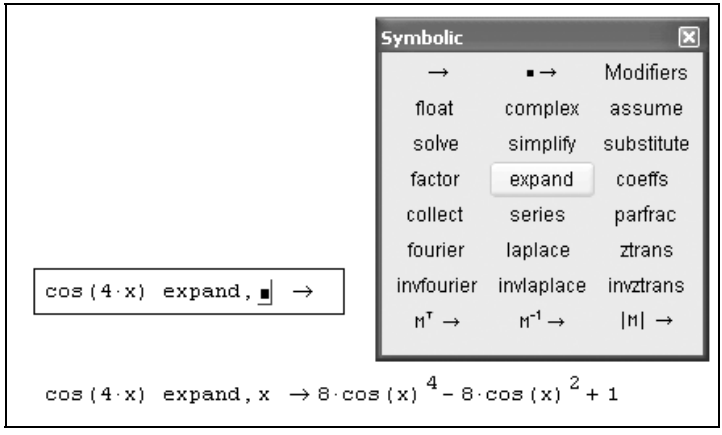



**Рис. 2.15.** Символьное разложение выражения

3. Введите в местозаполнитель после появившегося ключевого слова expand (рис. 2.15, сверху) имя переменной x либо нажмите клавишу <Del>, чтобы просто удалить местозаполнитель.
4. Введите оператор символьного вывода →.
5. Нажмите клавишу <Enter> либо просто щелкните мышью за пределами выражения.

### *Примечание*

Можно действовать и в другом порядке: сначала ввести ключевое слово, а уже затем впечатать в появившиеся местозаполнители выражение и переменную (рис. 2.16).

Оператор символьного вывода, как вы помните, можно ввести в редакторе Mathcad несколькими способами: нажатием кнопки → на любой из панелей **Evaluation** (Выражения) или **Symbolic** (Символика) либо сочетанием клавиш

<Ctrl>+<.>. Результат символьного разложения выражения показан на рис. 2.15, снизу.

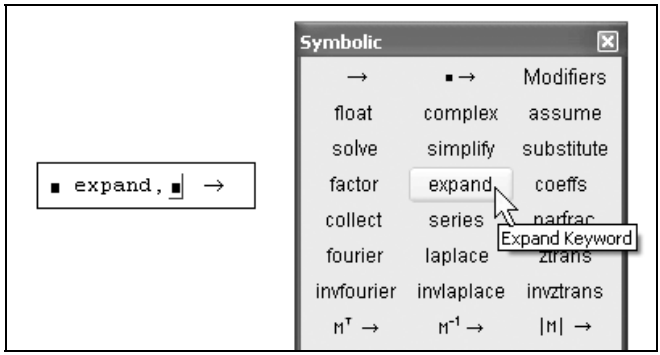



**Рис. 2.16.** Результат первоначального ввода ключевого слова

### *Совет*

Если вы можете выбрать способ символьных вычислений, рекомендую второй путь — с помощью оператора →, поскольку при этом в документе сохраняются действия пользователя. Наличие специального меню символьных вычислений — своего рода дань очень давним версиям Mathcad. В них аналитические преобразования были встроены не так гармонично и были доступны, главным образом, через меню.

Не всякое выражение поддается аналитическим преобразованиям. Если это так (либо в силу того, что задача вовсе не имеет аналитического решения, либо она оказывается слишком сложной для символьного процессора Mathcad), то в качестве результата выводится само выражение (листинг 2.11, внизу).

**Листинг 2.11. Символьные преобразования**

```
cos(2·x) expand, x → 2·cos(x)^2 − 1
cos(x) expand, x → cos(x)
```

### *Примечание*

Далее в этой главе, рассматривая символьные вычисления с помощью меню, будем иллюстрировать результаты рисунками, а символьные вычисления с применением оператора → приводить в виде листингов.

## 2.3.3. Упрощение выражений

*Упрощение выражений* — очень часто применяемая операция, противоположная по смыслу операции разложения, рассмотренной в предыдущем разделе. Символьный процессор Mathcad стремится так преобразовать выражение, чтобы оно приобрело более простую форму. При этом используются различные арифметические формулы, приведение подобных слагаемых, тригонометрические тождества, пересчет обратных функций и др. Чтобы упростить выражение с помощью меню (рис. 2.17):

1. Введите выражение.
2. Выделите выражение целиком или его часть, которую нужно упростить.
3. Выберите команду **Symbolics / Simplify** (Символика / Упростить).

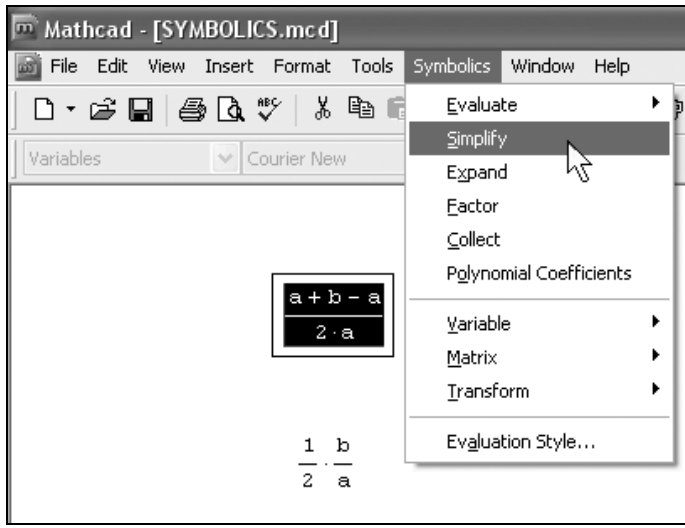



**Рис. 2.17.** Упрощение выражения

Для упрощения выражения при помощи оператора символьного вывода используйте ключевое слово `simplify` (листинг 2.12). Не забывайте, если некоторым переменным, входящим в выражение, ранее были присвоены некоторые значения, то они будут подставлены в него при выполнении символьного вывода (листинг 2.13).

**Листинг 2.12. Упрощение выражения**

$$\frac{a+b-a}{2\cdot a}\ \text{simplify}\ \rightarrow \frac{1}{2}\cdot\frac{b}{a}$$

$$\frac{a+b-a}{2\cdot a}\rightarrow \frac{1}{2}\cdot\frac{b}{a}$$

**Листинг 2.13. Упрощение выражения с подстановкой значения переменных**

$$a := 5 \qquad b := 10$$

$$\frac{a+b-a}{2\cdot a}\ \text{simplify}\ \rightarrow 1$$

$$\frac{a+b-a}{2\cdot a}\rightarrow 1$$

Упрощение выражений, содержащих числа, производится по-разному, в зависимости от наличия в числах десятичной точки. Если она есть, то выполняется непосредственное вычисление выражения (листинг 2.14).

**Листинг 2.14. Упрощение выражения с числами**

$$\sqrt{3.01}\ \text{simplify}\ \rightarrow 1.7349351572897472412$$

$$\text{acos}(0)\ \text{simplify}\ \rightarrow \frac{1}{2}\cdot\pi$$

## 2.3.4. Разложение на множители

Разложение выражений на простые множители производится при помощи команды **Symbolics / Factor** (Символика / Разложить на множители) (рис. 2.18) либо использованием вместе с оператором символьного вывода ключевого слова `factor` (листинг 2.15). Эта операция позволяет разложить полиномы на произведение более простых полиномов, а целые числа — на простые сомножители. Применяя команду меню, не забывайте перед ее вызовом выделить все выражение или его часть, которую планируете разложить на множители.

**Листинг 2.15. Примеры разложения на множители**

```
x^4 - 16 factor  → (x - 2) · (x + 2) · (x^2 + 4)

28 factor  → 2^2 · 7
```

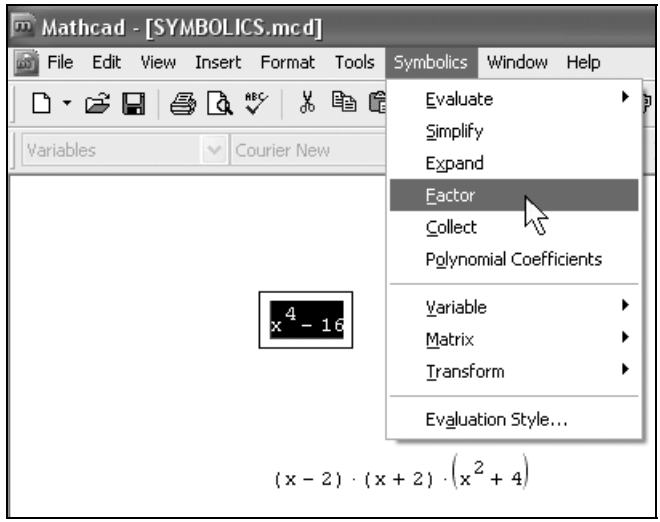



**Рис. 2.18.** Разложение выражения на множители

## 2.3.5. Приведение подобных слагаемых

Чтобы привести подобные слагаемые полинома с помощью меню (рис. 2.19):

1. Введите выражение.
2. Выделите в выражении имя переменной, относительно которой надо привести подобные слагаемые (в примере на рис. 2.19 это переменная `y`).
3. Выберите команду **Symbolics / Collect** (Символика / Привести подобные).

В результате появится строка с результатом приведения подобных слагаемых (нижняя строка на рис. 2.19).

Чтобы привести подобные слагаемые с помощью оператора символьного вывода (листинг 2.16):

1. Введите выражение.
2. Нажмите кнопку **Collect** на панели **Symbolic** (Символика).

3. Введите в местозаполнитель после вставленного ключевого слова `collect` имя переменной, относительно которой требуется привести подобные слагаемые (в первой строке примера из листинга 2.16 это переменная `x`, во второй — `y`).
4. Введите оператор символьного вывода →.
5. Нажмите клавишу <Enter>.

### *Примечание*

После ключевого слова `collect` допускается задание нескольких переменных через запятую. В этом случае, что иллюстрируется последней строкой листинга 2.6, приведение подобных слагаемых выполняется последовательно по всем переменным.

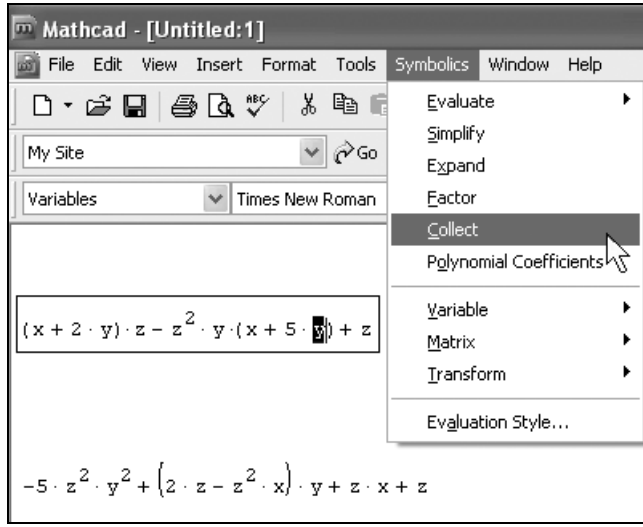



**Рис. 2.19.** Приведение подобных слагаемых

**Листинг 2.16. Приведение подобных слагаемых по разным переменным**

$$(x+2\cdot y)\cdot z-z^2\cdot y\cdot(x+5\cdot y)+z \text{ collect, } x \rightarrow \left(z-z^2\cdot y\right)\cdot x+2\cdot z\cdot y-5\cdot z^2\cdot y^2+z$$

$$(x+2\cdot y)\cdot z-z^2\cdot y\cdot(x+5\cdot y)+z \text{ collect, } y \rightarrow -5\cdot z^2\cdot y^2+\left(2\cdot z-z^2\cdot x\right)\cdot y+z\cdot x+z$$

$$(x+2\cdot y)\cdot z-z^2\cdot y\cdot(x+5\cdot y)+z \text{ collect, } x, y, z \rightarrow \left(z-z^2\cdot y\right)\cdot x+2\cdot z\cdot y-5\cdot z^2\cdot y^2+z$$

## 2.3.6. Вычисление коэффициентов полинома

Если выражение является полиномом относительно некоторой переменной x, заданным не в обычном виде $a_0+a_1x+a_2x^2+\ldots$, а как произведение других, более простых полиномов, то коэффициенты $a_0, a_1, a_2, \ldots$ легко определяются символьным процессором Mathcad. Коэффициенты сами могут быть функциями (подчас довольно сложными) других переменных.

Чтобы вычислить полиномиальные коэффициенты в выражении при помощи меню (рис. 2.20):

1. Введите выражение.
2. Выделите в нем имя переменной или выражение, для которого требуется рассчитать полиномиальные коэффициенты (в примере на рис. 2.20 это переменная z).
3. Выполните команду **Symbolic / Polynomial Coefficients** (Символика / Коэффициенты полинома).

В результате под выражением появится вектор, состоящий из полиномиальных коэффициентов. Первым элементом вектора является свободный член $a_0$, вторым — $a_1$ и т. д.

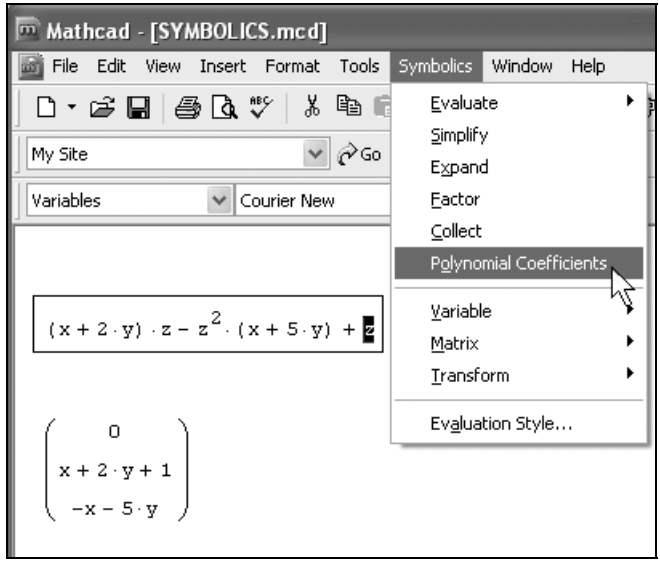



**Рис. 2.20.** Вычисление коэффициентов полинома

***Примечание***

Конкретная задача, требующая вычисления полиномиальных коэффициентов, приведена в разделе, посвященном численному отделению корней полинома (*см. разд. 5.2.3*).

Чтобы вычислить полиномиальные коэффициенты с помощью оператора символьного вывода:

1. Введите выражение.
2. Нажмите кнопку **Coeffs** на панели **Symbolic** (Символика).
3. Введите в местозаполнитель после вставленного ключевого слова `coeffs` аргумент полинома.
4. Введите оператор символьного вывода →.
5. Нажмите клавишу <Enter>.

Примеры вычисления коэффициентов полинома приведены в листингах 2.17 и 2.18. Листинг 2.17 показывает расчет коэффициентов для разных аргументов. Второй листинг демонстрирует возможность определения коэффициентов не только для отдельных переменных, но для более сложных выражений, входящих в рассматриваемую формулу в качестве составной части.

**Листинг 2.17. Вычисление коэффициентов полинома**

$$(x+2\cdot y)\cdot z - z^2\cdot y\cdot(x+5\cdot y) + z \text{ coeffs}, z \rightarrow \begin{pmatrix} 0 \\ x+2\cdot y+1 \\ -y\cdot x - 5\cdot y^2 \end{pmatrix}$$

$$(x+2\cdot y)\cdot z - z^2\cdot y\cdot(x+5\cdot y) + z \text{ coeffs}, x \rightarrow \begin{pmatrix} 2\cdot z\cdot y - 5\cdot z^2\cdot y^2 + z \\ z - z^2\cdot y \end{pmatrix}$$

**Листинг 2.18. Вычисление полиномиальных коэффициентов для простой переменной и выражения**

$$(x-4)\cdot(x-7)\cdot x + 99 \text{ coeffs}, x \rightarrow \begin{pmatrix} 99 \\ 28 \\ -11 \\ 1 \end{pmatrix}$$

$$(x-4)^3+(x-4)\cdot(x-7)\cdot x+99\ \text{coeffs},x-4 \rightarrow \begin{pmatrix} 99 \\ x^2-7\cdot x \\ 0 \\ 1 \end{pmatrix}$$

## 2.3.7. Разложение на простые дроби

Чтобы разложить сложную дробь на более простые дроби, следует либо выполнить команду **Symbolics / Variable / Convert to Partial Fractions** (Символика / Переменная / Разложить на элементарные дроби) (рис. 2.21), либо указать ключевое слово `parfrac` (листинг 2.19). Применяя первый способ (меню), не забывайте перед выбором его команды выделить переменную, по которой будет производиться разложение, а если используется второй способ (с оператором символьного вывода), то имя переменной следует указать после ключевого слова `parfrac`. В общем, последовательность действий при разложении на дроби та же самая, что и обычно (*см., например, разд. 2.3.4*).

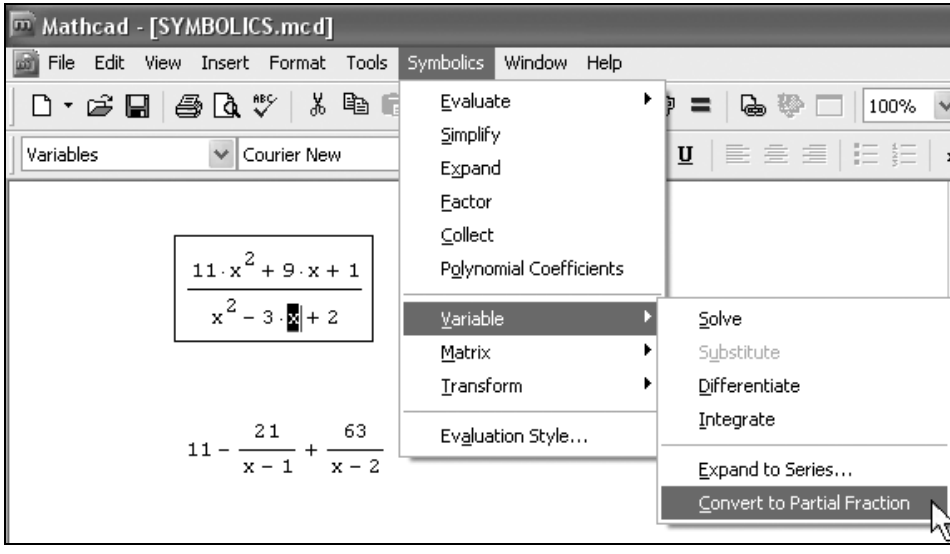



**Рис. 2.21.** Разложение сложной дроби на элементарные дроби

**Листинг 2.19. Разложение на элементарные дроби**

$$\frac{11\cdot x^2+9\cdot x+1}{x^2-3\cdot x+2}\ \text{convert},\text{parfrac},x\ \rightarrow 11-\frac{21}{(x-1)}+\frac{63}{(x-2)}$$

## 2.3.8. Вычисление рядов и произведений

Чтобы вычислить аналитически конечную или бесконечную сумму или произведение:

1. Введите выражение, используя панель **Calculus** (Вычисления) для вставки соответствующих символов суммирования или произведения. При необходимости введите в качестве предела ряда символ бесконечности (клавиши <Ctrl>+<Shift>+<Z>).
2. В зависимости от желаемого стиля символьных вычислений выберите команду **Symbolics / Simplify** (Символика / Упростить) или введите оператор символьного вывода →.

Примеры численного и символьного вычисления рядов и произведений приведены в листингах 2.20 и 2.21.

**Листинг 2.20. Символьные и численные расчеты рядов**

$$\sum_{i=0}^{10} 2^i = 2.047\times 10^3 \qquad \sum_{i=0}^{10} 2^i \rightarrow 2047$$

$$\sum_{i=0}^{\infty} a^i \rightarrow \frac{-1}{(a-1)}$$

$$\sum_{n=0}^{\infty} \frac{x^n}{2^n\cdot n!} \rightarrow \exp\left(\frac{1}{2}\cdot x\right) \qquad \sum_{n=0}^{\infty} \frac{1^n}{2^n\cdot n!} \rightarrow \exp\left(\frac{1}{2}\right) = 1.649$$

$$\sum_{n=0}^{100} \frac{1^n}{2^n\cdot n!} = 1.649$$

**Листинг 2.21. Символьный расчет произведения**

$$\prod_{n=1}^{\infty} \frac{1}{n^3+1} \to 0 \qquad \prod_{n=1}^{\infty} \sqrt{n} \to \infty$$

## 2.3.9. Подстановка переменной

Очень удобная возможность символьных вычислений — это операция подстановки значения переменной в выражение. При помощи меню подстановка производится следующим образом (рис. 2.22):

1. Выделите значение переменной, которое необходимо подставить в некоторое выражение. Значение переменной может быть любым выражением относительно любых переменных (на рис. 2.22 в качестве подстановки взята самая первая строка документа).
2. Скопируйте значение переменной в буфер обмена, например, нажатием клавиш <Ctrl>+<C> или кнопки **Copy** (Копировать) на панели инструментов **Standard** (Стандартная).
3. Выделите в выражении, в которое требуется подставить значение из буфера обмена, переменную, которая будет заменяться (во второй строке на рис. 2.22 выделена переменная `x`).
4. Выполните команду **Symbolics / Variable / Substitute** (Символика / Переменная / Подставить).

Результат этих действий иллюстрируется нижней строкой в документе на рис. 2.22.

Для осуществления той же операции в совокупности с оператором символьного вывода используйте ключевое слово `substitute`, которое вставляется в документ одноименной кнопкой на панели **Symbolic** (Символика). После ключевого слова `substitute` необходимо ввести в местозаполнители логическое выражение, показывающее, какую именно переменную какой формулой следует заменить (листинг 2.22).

**Листинг 2.22. Подстановка значения переменной**

$$\sin\left(k \cdot x^2 + b \cdot x\right) \text{ substitute , } k = a \cdot x^2 \to \sin\left(a \cdot x^4 + b \cdot x\right)$$

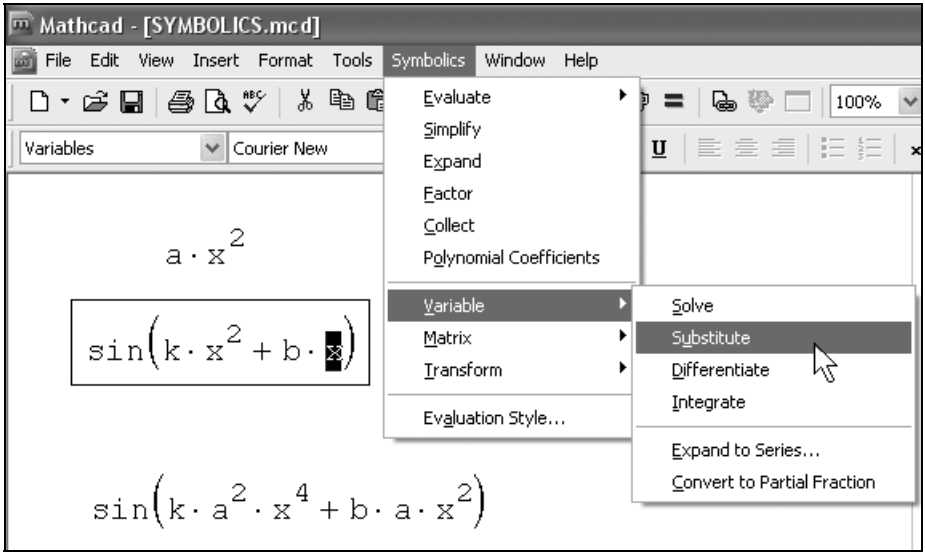



**Рис. 2.22.** Подстановка значения переменной

## 2.3.10. Получение численного значения выражения

С помощью символьного процессора можно рассчитать численное значение выражения (действительное или комплексное). Иногда такой путь представляется более удобным, чем применение численного процессора (т. е. знака обычного равенства). Чтобы рассчитать значение некоторого выражения (рис. 2.23), выберите команду **Symbolics / Evaluate / Symbolically** (Символика / Вычислить / Символьно) либо пункт **Symbolics / Evaluate / Floating Point** (Символика / Вычислить / С плавающей точкой). В последнем случае вам будет предложено с помощью диалога **Floating Point Evaluation** (Вычисления с плавающей точкой) задать точность вывода. В итоге применения данных команд Mathcad заменяет символьные результаты, где это возможно, значениями в виде чисел с плавающей точкой.

Еще один пункт меню **Symbolics / Evaluate / Complex** (Символика / Вычислить / Комплексно) позволяет представить выражение в виде `a+b·i`.

Аналогичные по действию ключевые слова `float` и `complex` можно использовать в документах, вводя их с панели **Symbolic** (Символика). Ключевое слово `float` применяется вместе со значением точности вывода результата с плавающей точкой (листинг 2.23). С помощью слова `complex` можно преобразовывать выражения как в символьном виде, так и с учетом численных

значений, если они были ранее присвоены переменным (несколько примеров приведено в листинге 2.24).

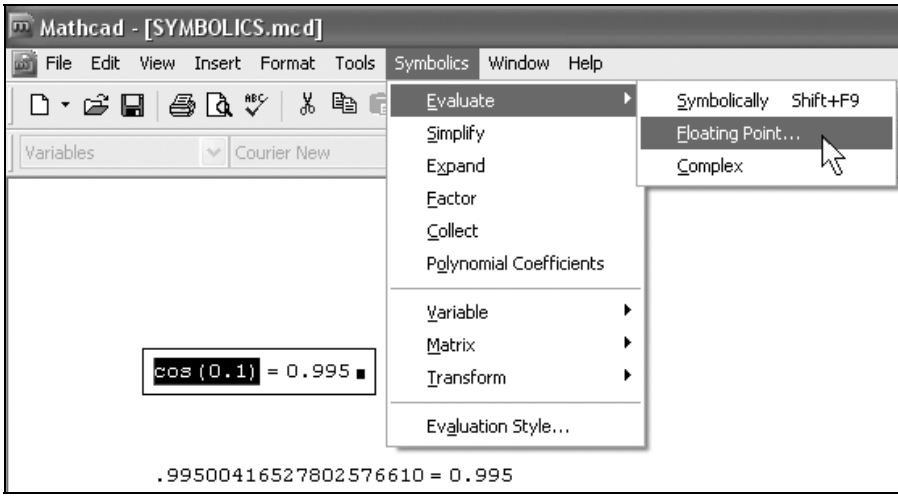



**Рис. 2.23.** Вычисление выражения с плавающей точкой

**Листинг 2.23. Вычисление выражения с плавающей точкой**

```
x := 3                    k := 2.4

cos(k·x) + 4·x^(2-k) float, 3  → 3.19

cos(k·x) + 4·x^(2-k) float, 10  → 3.185927374

cos(k·x) + 4·x^(2-k) float, 20  → 3.1859273744412716730
```

**Листинг 2.24. Комплексные преобразования выражений**

```
e^(z+2i) complex  → exp(z)·cos(2) + i·exp(z)·sin(2)

4.2·2i^(1.8-3i) complex  → 1193.4523970930846183 + 1107.3477730509390980·i

x := i

4·x^3 complex  → -4·i

4·x^3.1 complex  → .62573786016092347604 - 3.9507533623805509048·i
```

## ⑬ 2.3.12. Явные вычисления

В Mathcad 13 появилась возможность осуществления *явных* (*explicit*) символьных вычислений. Этим термином разработчики обозначили расчеты, в которых осуществляется простая подстановка в выражение численного значения той или иной переменной (без упрощения результата). Для организации явных вычислений предусмотрено специальное ключевое слово `explicit`, которое вводится с панели **Symbolic** (Символика). Сама формула вводится в левый местозаполнитель, а переменные, числовые значения которых должны быть подставлены в нее явно и без упрощения, — в правый местозаполнитель.

Пример вычислений, включающих ключевое слово `explicit`, приведен в листинге 2.25.

**Листинг 2.25. Явные вычисления**

$x := 2 \cdot m \qquad y := 0.5 \cdot sec$

$z := 1.5 \cdot \frac{m}{sec}$

$\frac{x}{y} + z \text{ explicit}, x \rightarrow \frac{2 \cdot m}{y} + z$

$\frac{x}{y} + z \text{ explicit}, x, y, z \rightarrow \frac{2 \cdot m}{0.5 \cdot sec} + 1.5 \cdot \frac{m}{sec}$

$\frac{x}{y} + z \text{ explicit}, z, y \rightarrow \frac{x}{0.5 \cdot sec} + 1.5 \cdot \frac{m}{sec}$

## 2.3.13. Вычисление предела

Наиболее ярким проявлением возможностей символьного процессора в Mathcad являются аналитические вычисления пределов, производных, интегралов и разложений в ряд, а также решение алгебраических уравнений. Все эти операции, при выполнении их посредством меню **Symbolics** (Символика), находятся в его подменю **Variable** (Переменная). Соответственно, требуется предварительное выделение в выражении переменной, относительно которой будет совершаться операция. Для выделения переменной достаточно поместить ее между линиями ввода, но для большей наглядности лучше выделить ее черным цветом путем протаскивания указателя мыши через нужную часть выражения.

***Примечание***

В отличие от других вычислительных операторов, пределы могут быть вычислены только символьно.

- Пределы (листинг 2.26):
  - двусторонний;
  - левый;
  - правый.

**Листинг 2.26. Операторы символьного вычисления пределов**

$$\lim_{x \to \infty} \frac{1 + 3 \cdot x}{x} \to 3$$

$$\lim_{x \to 0^+} \frac{1}{x} \to \infty$$

$$\lim_{x \to 0^-} \frac{1}{x} \to -\infty$$

## 2.3.14. О специфике аналитических вычислений

Выше в этой главе были разобраны основные приемы символьных вычислений в Mathcad. Они, как правило, были показаны на простых примерах, которые иллюстрировали ту или иную символьную операцию. Тем не менее при проведении разнообразных (и численных тоже) расчетов в Mathcad возможности символьного процессора можно использовать более эффективно. Отметим некоторые из них.

При проведении символьных вычислений с оператором символьного вывода функции пользователя и переменные, определенные ранее в документе Mathcad, воспринимаются символьным процессором корректно. Таким образом, имеется мощный аппарат включения символьных расчетов в программы пользователя. Примеры применения функции пользователя приведены в листингах 2.27 и 2.28. Сравните последние строчки этих листингов. Несмотря на их идентичность слева от знака символьного вывода, результаты получены различные. Это связано с тем, что в листинге 2.28 предварительно переменной `x` присвоено значение `4`. Поскольку значения переменных влияют на символьные вычисления, то результат учитывает подстановку вместо `x` числа `4`.

**Листинг 2.27. Функция пользователя в символьных вычислениях**

$$f(k,x) := \cos(k \cdot x) + 4 \cdot x^{2-k}$$

$$f(k,x)\ \text{substitute}, k = \sqrt{x} \rightarrow \cos\left(x^{\frac{3}{2}}\right) + 4 \cdot x^{2-x^{\frac{1}{2}}}$$

$$f(k,x)\ \text{series}, k, 2 \rightarrow 1 + 4 \cdot x^2 - 4 \cdot x^2 \cdot \ln(x) \cdot k$$

**Листинг 2.28. Значения переменных влияют на результат символьных вычислений**

$$f(k,x) := \cos(k \cdot x) + 4 \cdot x^{2-k}$$

$$x := 4$$

$$f(k,x)\ \text{series}, k, 2 \rightarrow 65 - 64 \cdot \ln(4) \cdot k$$

Напротив, при осуществлении символьных операций через меню **Symbolics** (Символика) символьный процессор "не видит" ничего, кроме выражения, в пределах которого находятся линии ввода. Поэтому ни функции пользователя, ни предварительно определенные значения каких-либо переменных никак не влияют на вычисления.

### *Совет*

Используйте меню **Symbolics** (Символика), если требуется "сиюминутно" провести некоторые аналитические действия с выражением и получить ответ в общем виде, не учитывающем текущие значения переменных, входящих в выражение.

# Глава 3

# Дифференцирование

Операция дифференцирования реализована в Mathcad как в численной, так и в аналитической форме и обозначается при помощи традиционного оператора, т. е. соответствующими математическими символами (подобно сложению или умножению). Если расчеты выполняются с помощью вычислительного процессора, необходимо хорошо представлять себе особенности численного алгоритма, действие которого остается для пользователя "за кадром". С помощью Mathcad можно вычислять производные скалярных функций любого количества аргументов, причем как функции, так и аргументы могут быть и действительными, и комплексными.

## 3.1. Аналитическое дифференцирование

Вне всякого сомнения, вы по достоинству оцените возможности символьного процессора, позволяющего с легкостью выполнять рутинные вычисления производных громоздких функций. В отличие от всех других операций, символьное дифференцирование выполняется успешно для подавляющего большинства аналитически заданных функций. Думаю, что многие читатели, которым иногда приходится аналитически дифференцировать функции, как и автор этих строк, единожды попробовав сделать это в среде Mathcad, уже никогда не возьмут в руки карандаша для расчета производных "вручную".

### 3.1.1. Аналитическое дифференцирование функции

Для того чтобы аналитически найти производную функции `f(x)` в Mathcad:

1. Задайте функцию `f(x)`.
2. Введите оператор дифференцирования нажатием кнопки **Derivative** (Производная) на панели **Calculus** (Вычисления) или введите с клавиатуры вопросительный знак <?>.

3. В появившихся местозаполнителях оператора дифференцирования (рис. 3.1) введите функцию, зависящую от аргумента `x`, т. е. `f(x)`, и имя самого аргумента `x`.
4. Введите оператор <→> символьного вычисления для получения ответа (листинг 3.1).

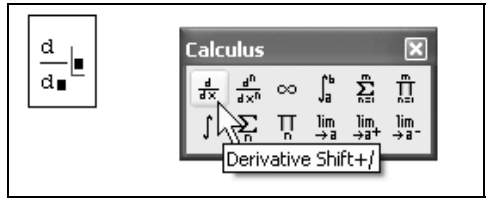



**Рис. 3.1.** Оператор дифференцирования

**Листинг 3.1. Пример аналитического дифференцирования**

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

$$\frac{d}{dx} f(x) \rightarrow \cos(x) \cdot \ln(x) + \frac{\sin(x)}{x}$$

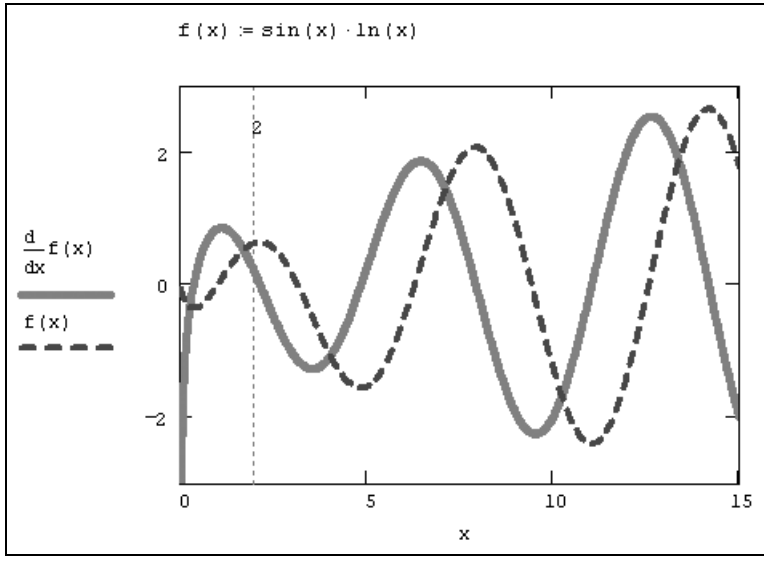



**Рис. 3.2.** График производной функции

### Примечание 1

Помните о том, что в описанном применении оператора дифференцирования его результатом является *функция* той же переменной `x`. Пример визуализации операции дифференцирования с помощью графика приведен на рис. 3.2.

### Примечание 2

Исходная функция может зависеть не только от аргумента `x`, но и от других аргументов, например, `f(x,y,z,t)` и т. п. В этом случае дифференцирование производится точно так же, причем становится более понятной необходимость определения переменной дифференцирования (в нижнем местозаполнителе оператора дифференцирования). Расчеты производных по разным аргументам (в этом случае говорят о *частных производных*), разумеется, будут давать совершенно разные результаты (*см. разд. 3.4*).

## 3.1.2. Вычисление производной функции в точке

Для того чтобы рассчитать производную в точке, необходимо предварительно задать значение аргумента в этой точке (листинг 3.2, вторая строка). Результатом дифференцирования в этом случае будет число — значение производной в этой точке. Если результат удается отыскать аналитически, то он приводится в виде числового выражения, а для того, чтобы получить его в форме числа, достаточно ввести после выданного выражения символ числового равенства <=> (последняя строка листинга 3.2).

**Листинг 3.2. Аналитическое дифференцирование функции в точке**

```
f(x) := sin(x)·ln(x)

x := 2

d/dx f(x) → cos(2)·ln(2) + 1/2·sin(2) = 0.166
```

Для того чтобы продифференцировать функцию, вовсе не обязательно предварительно присваивать ей какое-либо имя, как это сделано в листингах 3.1, 3.2. Можно определить функцию непосредственно в операторе дифференцирования (это демонстрирует первая строка листинга 3.3).

**Листинг 3.3. Правильное и неправильное использование оператора дифференцирования**

$$\frac{d}{dx}\sin(x) \rightarrow \cos(x)$$

$$\frac{d}{dx}\sin(2) \rightarrow 0$$

Как вы заметили, оператор дифференцирования, в основном, соответствует его общепринятому математическому обозначению, и поэтому его легко использовать интуитивно. Однако в некоторых случаях при вводе оператора дифференцирования следует проявить осторожность. Рассмотрим один показательный пример, приведенный во второй строке листинга 3.3, который демонстрирует неправильное применение оператора дифференцирования для вычисления производной в точке. Вместо вычисления производной `sin(x)` при `x=2`, как этого можно было ожидать, получено нулевое значение. Это случилось из-за того, что аргумент функции `sin(x)` введен не в виде переменной `x`, а в виде числа. Поэтому Mathcad воспринимает последнюю строку как вычисление сначала значения синуса в точке `x=2`, а затем дифференцирование этого значения (т. е. константы) также в точке `x=2`, в соответствии с требованием первой строки листинга. Поэтому ответ, на самом деле, неудивителен — в какой точке ни дифференцируй константу, результатом будет ноль.

***Примечание***

То же самое касается и операции численного дифференцирования, т. е. применения оператора <=> вместо <→> (*см. листинг 3.3.2 на компакт-диске*).

## 3.1.3. Определение функций пользователя через оператор дифференцирования

Разумеется, оператор дифференцирования, как и любой другой, можно применять для определения собственных функций пользователя. В листинге 3.4 через производную от `f(x)` определяется еще одна пользовательская функция `g(x)`, и затем, при помощи оператора символьного вывода, находится ее явный вид (предпоследняя строка листинга) и конкретное значение в точке `x=1` (последняя строка).

**Листинг 3.4. Определение функции посредством оператора дифференцирования**

$$f(x) := x^4 + 2x^3 - 7x^2 + 3x - 1$$

$$g(x) := \frac{d}{dx} f(x)$$

$$g(x) \rightarrow 4 \cdot x^3 + 6 \cdot x^2 - 14 \cdot x + 3$$

$$g(1) \rightarrow -1$$

## 3.1.4. Дифференцирование при помощи меню

Чтобы аналитически продифференцировать выражение по некоторой переменной, выделите в нем эту переменную и выберите команду **Symbolics / Variable / Differentiate** (Символика / Переменная / Дифференцировать) (рис. 3.3).

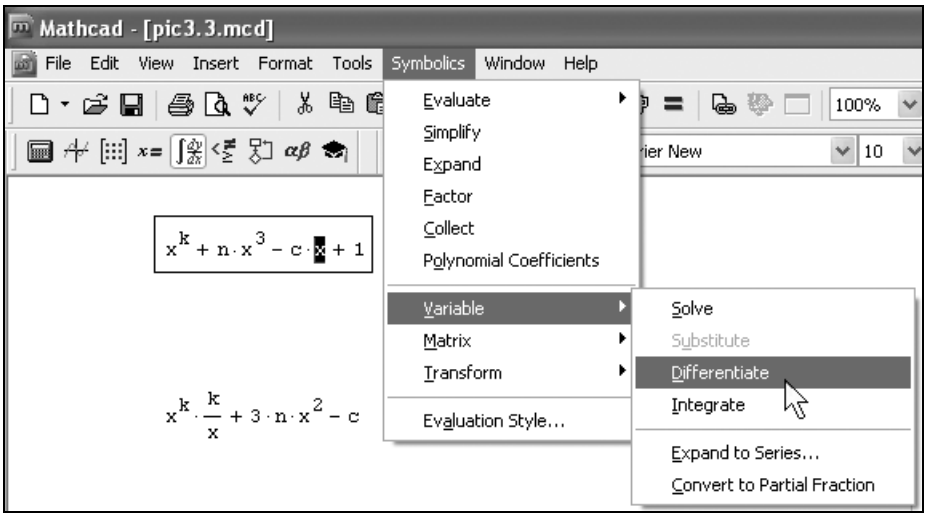



**Рис. 3.3.** Аналитическое дифференцирование по переменной x

В результате, в следующей строке за выражением появится значение ее производной. Для того чтобы найти вторую производную, повторно примените

эту последовательность действий, но уже к полученному результату дифференцирования. Так же находятся и производные высших порядков.

# 3.2. Численное дифференцирование

Вычислительный процессор Mathcad обеспечивает превосходную точность численного дифференцирования.

## 3.2.1. Дифференцирование в точке

Для того чтобы численно продифференцировать функцию `f(x)` в некоторой точке, следует использовать оператор численного вывода (вместо символьного):

1. Определите точку `x`, в которой будет вычислена производная, например, `x:=1`.
2. Введите оператор дифференцирования и обычным образом введите имена функции и аргумента в местозаполнители (см. рис. 3.1).
3. Введите оператор = численного вывода результата.

Пример дифференцирования функции `f(x)=sin(x)·ln(x)` приведен в листинге 3.5.

**Листинг 3.5. Численное дифференцирование функции в точке**

```
f(x) := sin(x)·ln(x)

x := 0.1

d/dx f(x) = -1.293
```

***Внимание!***

Не забывайте предварительно определять точку, в которой производится численное дифференцирование, как это сделано во второй строке листинга 3.5. Иначе будет выдано сообщение об ошибке, показанное на рис. 3.4, гласящее, что переменная или функция, входящая в выражение, ранее не определена. Между тем, символьное дифференцирование (*см. разд. 3.1*) не требует обязательного явного задания точки дифференцирования. В этом случае вместо значения производной (числа или числового выражения) будет выдана аналитическая зависимость (см. листинг 3.1).

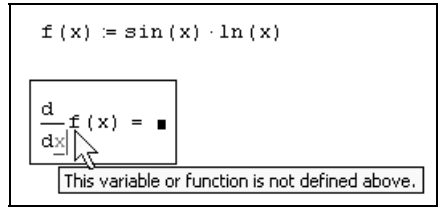



**Рис. 3.4.** Ошибка в применении оператора дифференцирования (не задан аргумент)

### 11 *Примечание*

В Mathcad 11 и выше для ускорения и повышения точности численного дифференцирования функций, заданных аналитически, автоматически задействуется символьный процессор. Сначала предпринимается попытка отыскать производные, входящие в выражение, аналитически, а уже затем, если она оказывается неудачной, включается в работу численный метод.

## 3.2.2. Об алгоритме дифференцирования

Для численного дифференцирования Mathcad применяет довольно сложный алгоритм, вычисляющий производную с колоссальной точностью до 7—8-го знака после запятой. Погрешность дифференцирования не зависит от констант `TOL` или `CTOL`, в противоположность большинству остальных численных методов, а определяется непосредственно алгоритмом. Этот алгоритм (метод Риддера) изложен во встроенной справочной системе Mathcad, доступной через меню **Help** (Справка). Мы не будем здесь его описывать, однако остановимся на важных аспектах численного определения производной функции `f(x)` на более простом примере. Несмотря на то, что простейшая разностная формула сильно отличается от метода Риддера, он все-таки поможет нам разобраться в некоторых вопросах, т. к. основан на базовом принципе численного дифференцирования, а именно на вычислении производной через значения функции `f(x)` в нескольких точках, расположенных на близком расстоянии друг от друга.

Исходя из определения производной функции, можно констатировать, что

$$\frac{df(x)}{dx} = \frac{f(x+\Delta) - f(x)}{\Delta} + o(\Delta). \qquad (3.1)$$

Основная проблема численного определения производной (как в этой простейшей формуле, так и в более сложных алгоритмах, в том числе Риддера) связана как раз с процедурой выбора значения $\Delta$, которая является далеко не

очевидной. На первый взгляд может показаться, что следует выбирать очень малые Δ, чтобы соблюсти желаемую точность, однако это не совсем так. Чтобы лучше разобраться в сути проблемы, используем Mathcad-программу, приведенную в листинге 3.6, которая рассчитывает (в зависимости от шага Δ) погрешность разностной формулы (3.1). График полученной зависимости изображен на рис. 3.5, причем для его обеих осей выбран логарифмический масштаб, а сама производная (ради примера), согласно листингу 3.6, считается в одной точке `x=1`.

**Листинг 3.6. Расчет зависимости точности разностной формулы от шага Δ**

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

$$x := 1$$

$$i := 0 .. 20$$

$$\Delta_i := 10^{-i}$$

$$\varepsilon_i := \left| \frac{d}{dx} f(x) - \frac{f(x + \Delta_i) - f(x)}{\Delta_i} \right|$$

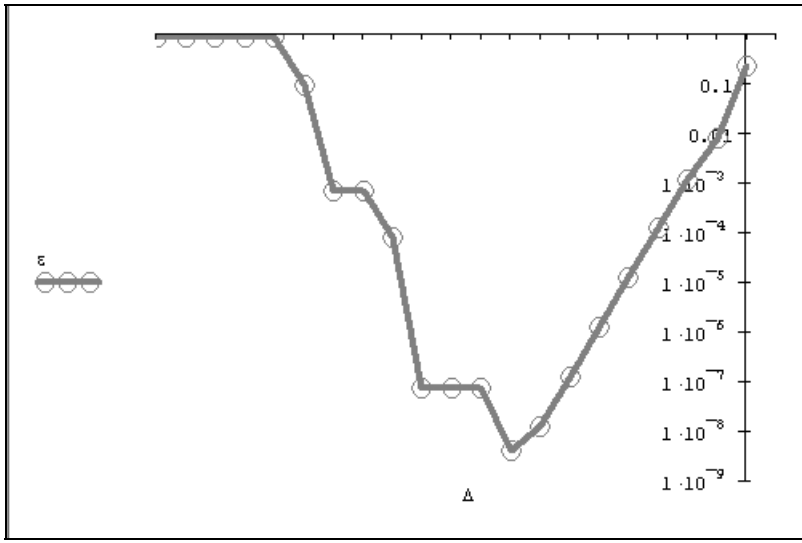



**Рис. 3.5.** График точности формулы (3.1) в зависимости от шага Δ (продолжение листинга 3.6)

Если увеличение ошибки на правом конце графика является совершенно очевидным, поскольку, согласно формуле (3.1), чем больше Δ, тем больше погрешность, то рост ошибки при очень малых Δ может, на первый взгляд, показаться неожиданным. Однако все дело в том, что, применяя разностную формулу, мы неявно полагали, что умеем точно вычислять значения функции `f(x)` в любой точке. Между тем, любые компьютерные вычисления сопряжены с неустранимыми погрешностями, в частности, обусловленными дискретным представлением чисел. Поэтому в реальности мы можем вычислить значение `f(x)` лишь с некоторой погрешностью δ, обусловленной (по крайней мере) заведомым округлением чисел при расчетах на компьютере.

В результате, при очень малом шаге разностные формулы означают вычитание друг из друга близких чисел. В этом случае ошибки вычисления функции `f(x)` становятся доминирующими и приводят к существенному росту суммарной погрешности вычисления разностной производной. Отсюда как раз и следует тот вывод, что значение шага следует выбирать "не очень малым", иначе ошибки вычисления `f(x)` неминуемо сделают результат дифференцирования неправильным. Глядя на рис. 3.5, легко сообразить, что в данном случае следует выбирать промежуточные значения Δ, которые обеспечат минимальную (или почти минимальную) погрешность.

Следует подчеркнуть, что в зависимости от характера дифференцируемой функции диапазон приемлемых значений Δ будет различным. Поэтому в каждом конкретном случае требуется совершать дополнительные шаги, тестирующие верность выбора шага для численного дифференцирования. Такая процедура, кстати говоря, заложена в адаптивном алгоритме дифференцирования, примененном в Mathcad, что делает его весьма надежным для численного расчета производной.

С учетом сказанного выше, с дифференцированием в Mathcad обычно не возникает сложных проблем. Исключение составляют функции, которые дифференцируются в окрестности сингулярной точки; например, для функции `f(x)=1/x` это будут точки вблизи `x=0`. При попытке найти ее производную при `x=0` (рис. 3.6) будет выдано сообщение об одной из ошибок деления на ноль "Can't divide by zero" (Деление на ноль невозможно) или "Found a singularity while evaluating this expression. You may be dividing by zero" (Найдена сингулярность при вычислении этого выражения. Возможно, вы делите на ноль).

Если попробовать численно определить производную очень близко к нулю, например, при $x=10^{-100}$, то, несмотря на существование производной, может появиться сообщение об ошибке "Can't converge to a solution" (Невозможно

найти решение). Новые версии Mathcad (начиная с 11-й) справляются с указанной трудностью, поскольку в них даже для численного дифференцирования сначала задействуется символьный процессор, поставляющий аналитическое решение, подстановка в которое аргумента дифференцирования дает верный результат. Встретившись с одной из упомянутых ошибок, присмотритесь внимательнее к дифференцируемой функции и убедитесь, что вы не имеете дело с точкой сингулярности.

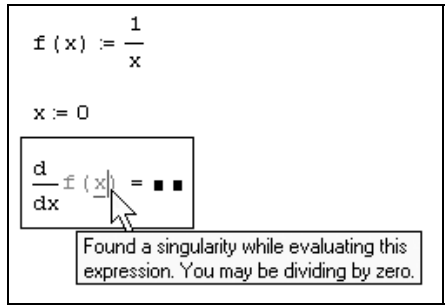



**Рис. 3.6.** Если производная функции в данной точке не существует, выдается сообщение об ошибке

## 3.3. Производные высших порядков

Mathcad позволяет численно определять производные высших порядков, от `0`-го до `5`-го включительно. Чтобы вычислить производную функции `f(x)` `N`-го порядка в точке `x`, нужно проделать те же самые действия, что и при взятии первой производной (*см. разд. 3.1 и 3.2*), за тем исключением, что вместо оператора производной необходимо применить оператор `N`-ой производной (**Nth Derivative**). Этот оператор вводится с той же панели **Calculus** (Вычисления), либо с клавиатуры нажатием клавиш <Ctrl>+<?>, и содержит еще два дополнительных местозаполнителя (рис. 3.7), в которые следует поместить число `N`. В полном соответствии с математическим смыслом оператора, определение порядка производной в одном из местозаполнителей приводит к автоматическому появлению того же числа в другом из них.

Очевидно, что "производная" при `N=0` по определению равна самой функции, при `N=1` получается обычная первая производная. Листинг 3.7 демонстрирует численное и символьное вычисление второй производной функции в заданной точке. Обратите внимание, что, как и при вычислении обычной производной, необходимо перед оператором дифференцирования присвоить аргу-

менту функции значение, для которого будет вычисляться производная. А вот для аналитического нахождения производных высших порядков при помощи оператора символьного вывода (*в полном соответствии с разд. 3.1*) вводить значения аргумента не следует (листинг 3.8).

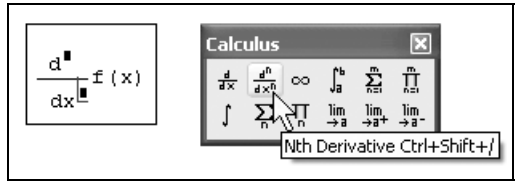



**Рис. 3.7.** Оператор производной высшего порядка

**Листинг 3.7. Пример вычисления второй производной функции в точке**

$$f(x) := \frac{1}{x}$$

$$x := 3$$

$$\frac{d^2}{dx^2} f(x) = 0.074$$

$$\frac{d^2}{dx^2} f(x) \rightarrow \frac{2}{27}$$

**Листинг 3.8. Пример аналитического поиска второй производной функции**

$$f(x) := \frac{1}{x}$$

$$\frac{d^2}{dx^2} f(x) \rightarrow \frac{2}{x^3}$$

### *Примечание*

Убедиться в том, что символьный процессор Mathcad в последней строке листинга 3.7 дает тот же результат, что и вычислительный процессор в предыдущей строке, можно, упростив его. Для этого следует выделить полученное по-

следнее выражение и выбрать в меню **Symbolics** (Символика) пункт **Simplify** (Упростить). После этого ниже появится еще одна строка с численным результатом выделенного выражения.

Повторимся, что численный метод предусматривает возможность вычисления производных до 5-го порядка, а символьный процессор умеет считать производные произвольного порядка (конечно, если аналитическое решение задачи в принципе существует). Сказанное иллюстрирует листинг 3.9, в котором аналитически вычисляется шестая производная функции, а попытка численного вывода результата того же выражения приводит к ошибке.

**Листинг 3.9. Численное и символьное вычисление шестой производной**

$$f(x) := \frac{1}{x}$$

$$\frac{d^6}{dx^6} f(x) \rightarrow \frac{720}{x^7}$$

$$x := 3$$

$$\frac{d^6}{dx^6} f(x) = \blacksquare$$

Чтобы вычислить производную порядка выше 5-го численно, можно последовательно применить несколько раз оператор N-ой производной (листинг 3.10), подобно тому, как производится отыскание кратных интегралов (*см. разд. 4.3.4*). Однако следует помнить о том, что численное определение производных высших порядков производится тем же вычислительным методом Риддера, что и для первых производных. Поскольку, как уже было сказано, для первой производной этот метод обеспечивает точность до 7—8 значащих разрядов числа, при повышении порядка производной на каждую единицу точность падает примерно на один разряд.

***Внимание!***

Из сказанного ясно, что падение точности при численном расчете высших производных может быть очень существенно. В частности, если попытаться определить шестую производную функции 1/x, то в качестве результата будет выдан ноль, в то время как истинное значение шестой производной может быть найдено при помощи символьного процессора (листинг 3.10).

**Листинг 3.10. Попытка численного поиска шестой производной функции в точке дает неправильный результат**

$$f(x) := \frac{1}{x}$$

$$x := 0.1$$

$$\frac{d}{dx}\left(\frac{d^5}{dx^5} f(x)\right) = 0$$

$$\frac{d^6}{dx^6} f(x) \rightarrow 7.20 \cdot 10^9$$

# 3.4. Частные производные

С помощью обоих процессоров Mathcad можно вычислять производные функций не только одного, но и любого количества аргументов. Как известно, производные функции нескольких аргументов по одному из них называются *частными*. Чтобы вычислить частную производную, необходимо, как обычно, ввести оператор производной с панели **Calculus** (Вычисления) и в соответствующем местозаполнителе напечатать имя переменной, по которой должно быть осуществлено дифференцирование.

## 3.4.1. Частные производные

Примеры отыскания частных производных функции двух переменных приведены в листингах 3.11 и 3.12. В первой строке обоих листингов определяется сама функция, а в последующих (символьным или численным образом) рассчитываются ее производные по обеим переменным — `x` и `k`. Чтобы определить частную производную в точке, необходимо предварительно задать значения всех аргументов, что и сделано в следующих строках листинга 3.12. Обратите внимание, что для символьного поиска производной функции нет необходимости задавать значения всех ее аргументов (третья строка листинга 3.12), а вот для численного дифференцирования (последняя строка листинга) должны быть предварительно определены все аргументы функции, иначе вместо результата появится сообщение об ошибке.

**Листинг 3.11. Аналитическое вычисление частных производных**

$$f(x,k) := k\cdot\sin(x)$$

$$\frac{\partial}{\partial x} f(x,k) \rightarrow k\cdot\cos(x)$$

$$\frac{\partial}{\partial k} f(x,k) \rightarrow \sin(x)$$

**Листинг 3.12. Символьное и численное вычисления частных производных в точке**

$$f(x,k) := k\cdot\sin(x)$$

$$x := 10$$

$$\frac{\partial}{\partial x} f(x,k) \rightarrow k\cdot\cos(10)$$

$$k := 1$$

$$\frac{\partial}{\partial x} f(x,k) = -0.839$$

Частные производные высших порядков рассчитываются точно так же, как и обычные производные высших порядков (*см. разд. 3.3*). Листинг 3.13 иллюстрирует расчет вторых производных функции по переменным `x` и `y`, а также смешанной производной.

**Листинг 3.13. Вычисление второй частной производной**

$$f(x,y) := y^2\cdot x^3 + y\cdot x^2$$

$$\frac{\partial^2}{\partial x^2} f(x,y) \rightarrow 6\cdot y^2\cdot x + 2\cdot y$$

$$\frac{\partial^2}{\partial y^2} f(x,y) \rightarrow 2\cdot x^3$$

$$\frac{\partial}{\partial x}\frac{\partial}{\partial y} f(x, y) \rightarrow 6 \cdot y \cdot x^2 + 2 \cdot x$$

Возможно, вы обратили внимание, что во всех трех листингах 3.11—3.13 оператор дифференцирования записан в традиционной форме частной производной (с округлыми символами дифференциала). Запись оператора не влияет на вычисления, а служит лишь более привычной формой представления расчетов.

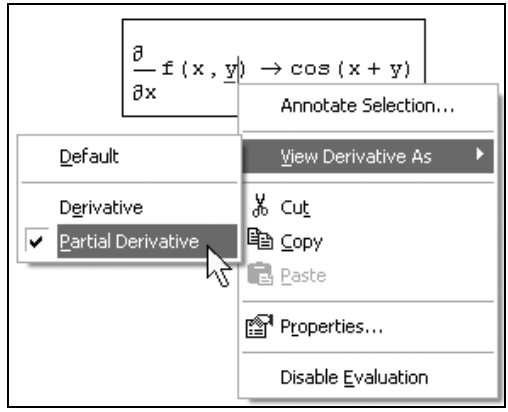



**Рис. 3.8.** Изменение вида оператора дифференцирования

Для того чтобы изменить вид оператора дифференцирования на представление частной производной, следует:

1. Вызвать контекстное меню из области оператора дифференцирования нажатием правой кнопки мыши.
2. Выбрать в контекстном меню верхний пункт **View Derivative As** (Показывать производную как).
3. В появившемся подменю (рис. 3.8) выбрать пункт **Partial Derivative** (Частная производная).

Чтобы вернуть вид производной, принятый по умолчанию, выберите в подменю пункт **Default** (По умолчанию) либо, для представления ее в обычном виде, — **Derivative** (Производная).

## 3.4.2. Примеры: градиент, дивергенция и ротор

Завершим разговор о частных производных несколькими примерами векторного анализа, которые нередко встречаются в вычислительной практике. Программная реализация первого из них, посвященная вычислению *градиента* функции двух переменных, приведена в листинге 3.14. В качестве примера взята функция `f(x,y)`, определяемая в первой строке листинга, график которой показан на рис. 3.9, в виде линий уровня. Как известно, градиент функции `f(x,y)` является векторной функцией тех же аргументов, что и `f(x,y)`, определенной через ее частные производные, согласно второй строке листинга 3.14. В его третьей строке производится аналитическое вычисление градиента, а в оставшейся части листинга задаются ранжированные переменные и матрицы, необходимые для подготовки графика линий уровня самой функции и графика векторного поля ее градиента (рис. 3.10).

**Листинг 3.14. Вычисление градиента функции двух переменных**

$$f(x,y) := 0.12\cdot x^2 + x\cdot y - 0.01\cdot y^4$$

$$grad(x,y) := \begin{pmatrix} \frac{\partial}{\partial x} f(x,y) \\ \frac{\partial}{\partial y} f(x,y) \end{pmatrix}$$

$$grad(x,y) \rightarrow \begin{pmatrix} .24\cdot x + y \\ x - 4.\cdot 10^{-2}\cdot y^3 \end{pmatrix}$$

$$N := 5$$

$$i := 0 .. 2\cdot N \qquad j := 0 .. 2\cdot N$$

$$F_{i,j} := f(i - N, j - N) \qquad V_{i,j} := grad(i - N, j - N)$$

$$X_{i,j} := (V_{i,j})_0 \qquad Y_{i,j} := (V_{i,j})_1$$

Как можно убедиться, сравнив графики на рис. 3.9 и 3.10, математический смысл градиента состоит в задании в каждой точке `(x,y)` направления на плоскости, в котором функция `f(x,y)` растет наиболее быстро. Абсолютное значение градиента (т. е. длина вектора в каждой точке) определяет локальную скорость изменения `f(x,y)`.

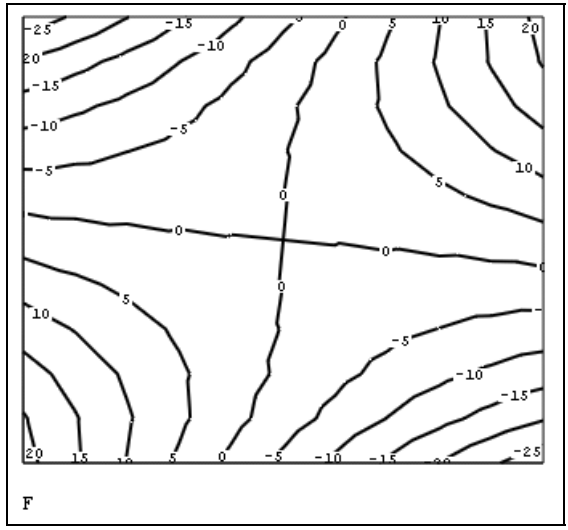



**Рис. 3.9.** Модельная функция двух переменных (продолжение листинга 3.14)

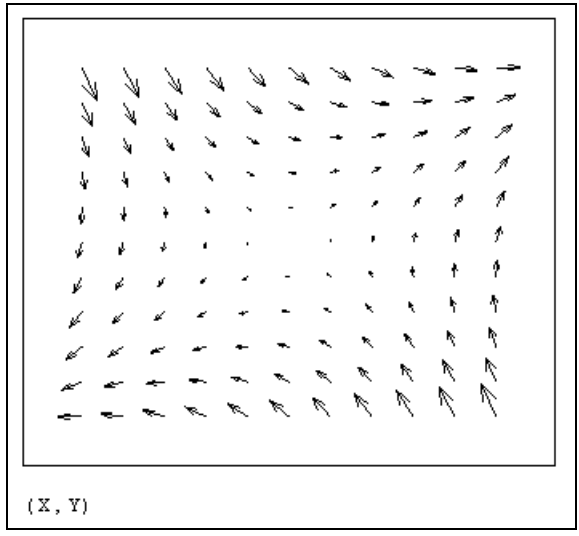



**Рис. 3.10.** Векторное поле градиента функции двух переменных (продолжение листинга 3.14)

Из сопоставления графиков ясно, что в центре показанной на них области `(x,y)` сама функция `f(x,y)` меняется медленно (соответственно, значения ее

градиента являются малыми), а в углах — быстро (там значения градиента максимальны).

Очень важно заметить, что градиент является не скалярной, а векторной функцией переменных `x,y`, поскольку фактически представляет собой комбинацию двух функций, задающих соответствующие проекции (горизонтальную и вертикальную) вектора в каждой точке. До сих пор в данной главе мы рассматривали дифференцирование скалярных функций, однако в математике часто приходится иметь дело и с вычислением производных векторных функций. Рассмотрим эти действия на примере операции поиска *дивергенции* (листинг 3.15 и рис. 3.11), применимой к *векторному полю*, т. е. векторной функции, зависящей от пространственных координат (на плоскости, как в нашем примере, или в трехмерном пространстве).

**Листинг 3.15. Вычисление дивергенции векторной функции**

$$f(x,y) := \begin{pmatrix} 0.24\cdot x + y \\ x - 4\cdot 10^{-2}\cdot y^{3} \end{pmatrix}$$

$$div(x,y) := \frac{\partial}{\partial x}\left(f(x,y)_0\right) + \frac{\partial}{\partial y}\left(f(x,y)_1\right)$$

$$div(x,y) \rightarrow .24 - \frac{3}{25}\cdot y^{2}$$

Если, как принято в математике, обозначить оператор взятия градиента символом $\nabla$, то дивергенцию вектор-функции можно формально определить как скалярное произведение $\nabla\cdot f$, а еще одну распространенную операцию векторного анализа — *ротор* (или, по-другому, *вихрь* или *завихренность*) — как векторное произведение $\nabla\times f$. Рисунок 3.11 иллюстрирует пример векторной функции `f(x,y)` (определяемой в первой строке листинга) и вычисление ее дивергенции (которое производится аналитически в третьей строке). Обратите внимание, что в качестве исходной вектор-функции взят результат предыдущих расчетов, показанный (в форме векторного поля) на рис. 3.10. Строки кода в верхней части рис. 3.11 нужны для подготовки графика вычисленной дивергенции (в виде трехмерной поверхности и линий уровня, соответственно сверху и снизу).

Точно такую же структуру имеют расчеты ротора той же векторной функции `f(x,y)` в листинге 3.16, причем определение операции взятия ротора приводится в его второй строке (как и в случае дивергенции для листинга 3.15).

Читателю, знакомому с векторным анализом, предлагается догадаться самому, почему в рассматриваемом примере (листинги 3.14—3.16) ротор получается тождественно равным нулю (последняя строка листинга 3.16).

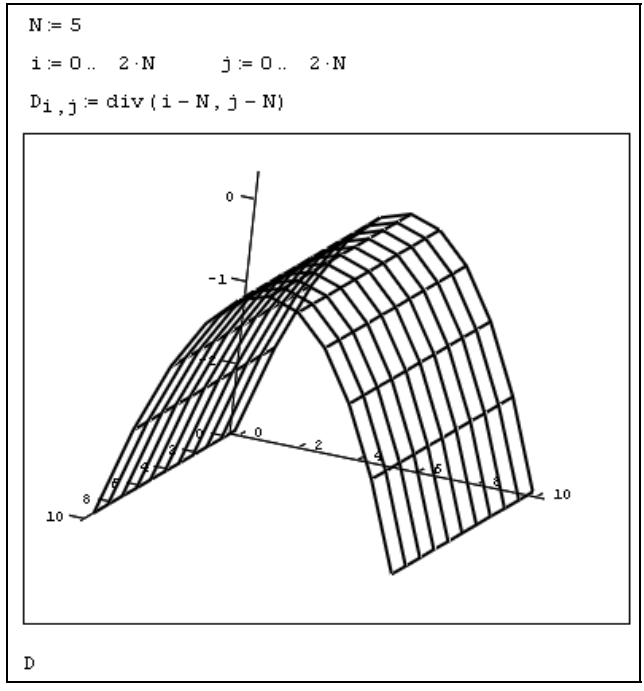



**Рис. 3.11.** График дивергенции векторной функции (продолжение листинга 3.15)

**Листинг 3.16. Вычисление ротора векторной функции**

$$f(x,y) := \begin{pmatrix} 0.24\cdot x + y \\ x - 4\cdot 10^{-2}\cdot y^{3} \end{pmatrix}$$

$$rot(x,y) := \frac{\partial}{\partial x}\left(f(x,y)_1\right) - \frac{\partial}{\partial y}\left(f(x,y)_0\right)$$

$$rot(x,y) \rightarrow 0$$

В заключение разговора о векторном анализе функций подчеркнем, что примеры в листингах 3.14—3.16 относились к функциям двух переменных, т. е.

описывали двумерный случай. Еще два листинга — 3.17 и 3.18 — показывают, как действуют перечисленные операции векторного анализа в трехмерном (пространственном) случае.

### *Примечание*

В электронной книге **Resource Center** (Центр ресурсов), поставляемой вместе с Mathcad, вы найдете дополнительные примеры вычисления градиента, дивергенции и ротора, относящихся к трехмерному случаю.

**Листинг 3.17. Градиент функции трех переменных**

$$f(x,y,z) := z\cdot\sin(x\cdot y)$$

$$\mathrm{grad}(x,y,z) := \begin{pmatrix} \frac{\partial}{\partial x} f(x,y,z) \\ \frac{\partial}{\partial y} f(x,y,z) \\ \frac{\partial}{\partial z} f(x,y,z) \end{pmatrix} \qquad \mathrm{grad}(x,y,z) \rightarrow \begin{pmatrix} z\cdot\cos(x\cdot y)\cdot y \\ z\cdot\cos(x\cdot y)\cdot x \\ \sin(x\cdot y) \end{pmatrix}$$

**Листинг 3.18. Дивергенция и ротор в трехмерном пространстве**

$$f(x,y,z) := \begin{pmatrix} x\cdot y \\ z \\ x+2z \end{pmatrix}$$

$$\mathrm{div}(x,y,z) := \frac{\partial}{\partial x}\left(f(x,y,z)_0\right) + \frac{\partial}{\partial y}\left(f(x,y,z)_1\right) + \frac{\partial}{\partial z}\left(f(x,y,z)_2\right)$$

$$\mathrm{rot}(x,y,z) := \begin{bmatrix} \frac{\partial}{\partial y}\left(f(x,y,z)_2\right) - \frac{\partial}{\partial z}\left(f(x,y,z)_1\right) \\ \frac{\partial}{\partial z}\left(f(x,y,z)_0\right) - \frac{\partial}{\partial x}\left(f(x,y,z)_2\right) \\ \frac{\partial}{\partial x}\left(f(x,y,z)_1\right) - \frac{\partial}{\partial y}\left(f(x,y,z)_0\right) \end{bmatrix}$$

$$\mathrm{div}(x, y, z) \rightarrow y + 2$$

$$\mathrm{rot}(x, y, z) \rightarrow \begin{pmatrix} -1 \\ -1 \\ -x \end{pmatrix}$$

## 3.4.3. Пример: якобиан

Еще одна задача, связанная с нахождением частных производных векторной функции, заключается в вычислении *якобиана* (или определителя *матрицы Якоби*) — матрицы, составленной из частных производных векторной функции по всем ее аргументам. Эта задача встречается в различных областях математики, например, применительно к жестким дифференциальным уравнениям (*см. разд. 9.4*). Приемы вычисления якобиана векторной функции `f(x)` векторного аргумента `x` демонстрируются в листинге 3.19. В нем для определения частных производных якобиана каждый `i`-й скалярный компонент $f(x)_i$ дифференцируется символьным процессором Mathcad.

**Листинг 3.19. Вычисление якобиана векторной функции векторного аргумента**

$$f(x) := \begin{bmatrix} x_0 \cdot x_1 \\ (x_1)^{x_0} \\ x_1 \cdot x_2 \end{bmatrix}$$

$$J(x, y, z) := \begin{bmatrix} \frac{\partial}{\partial x}\left[f\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)_0\right] & \frac{\partial}{\partial y}\left[f\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)_0\right] & \frac{\partial}{\partial z}\left[f\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)_0\right] \\ \frac{\partial}{\partial x}\left[f\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)_1\right] & \frac{\partial}{\partial y}\left[f\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)_1\right] & \frac{\partial}{\partial z}\left[f\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)_1\right] \\ \frac{\partial}{\partial x}\left[f\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)_2\right] & \frac{\partial}{\partial y}\left[f\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)_2\right] & \frac{\partial}{\partial z}\left[f\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)_2\right] \end{bmatrix}$$

$$J(x,y,z) \rightarrow \begin{pmatrix} y & x & 0 \\ y^x \cdot \ln(y) & y^x \cdot \frac{x}{y} & 0 \\ 0 & z & y \end{pmatrix}$$

Тот же самый якобиан можно вычислить и несколько по-другому, если определить функцию не одного векторного, а трех скалярных аргументов `f(x,y,z)` (листинг 3.20). Не забывайте, что для численного определения якобиана необходимо сначала определить точку, в которой он будет рассчитываться, т. е. вектор `x` в терминах листинга 3.19, или все три переменных `x,y,z` в обозначениях листинга 3.20.

**Листинг 3.20. Вычисление якобиана векторной функции трех скалярных аргументов**

$$f(x,y,z) := \begin{pmatrix} x \cdot y \\ y^x \\ y \cdot z \end{pmatrix}$$

$$J(x,y,z) := \begin{pmatrix} \frac{\partial}{\partial x} f(x,y,z)_0 & \frac{\partial}{\partial y} f(x,y,z)_0 & \frac{\partial}{\partial z} f(x,y,z)_0 \\ \frac{\partial}{\partial x} f(x,y,z)_1 & \frac{\partial}{\partial y} f(x,y,z)_1 & \frac{\partial}{\partial z} f(x,y,z)_1 \\ \frac{\partial}{\partial x} f(x,y,z)_2 & \frac{\partial}{\partial y} f(x,y,z)_2 & \frac{\partial}{\partial z} f(x,y,z)_2 \end{pmatrix}$$

$$J(x,y,z) \rightarrow \begin{pmatrix} y & x & 0 \\ y^x \cdot \ln(y) & y^x \cdot \frac{x}{y} & 0 \\ 0 & z & y \end{pmatrix}$$

# 3.5. Разложение функции в ряд Тейлора

Еще одна операция, тесно связанная с дифференцированием, представляет собой разложение функции в *ряд Тейлора* по любой переменной `x` в некоторой точке. Если эта точка `x=0`, то ряд называют также *рядом Маклорена*, и он

представим в окрестности точки x=0 суммой вида $a_0+a_1x+a_2x^2+a_3x^3+\ldots$. Здесь $a_i$ — некоторые коэффициенты, не зависящие от x, но, возможно, являющиеся функциями других переменных, от которых зависит исходная функция. Именно эти коэффициенты выражаются через производные функции. Если она имеет в точке x=0 особенность, то соответствующее разложение называют *рядом Лорана*.

При поиске разложения в ряд Тейлора нет необходимости явно рассчитывать все его коэффициенты, поскольку эта операция предусмотрена разработчиками Mathcad и выполняется при помощи символьного процессора. При этом можно использовать для ее осуществления как соответствующие встроенные функции, так и меню **Symbolics** (Символика).

## 3.5.1. Разложение в ряд при помощи меню

Чтобы разложить некоторое выражение в ряд:

1. Введите выражение.
2. Выделите значение переменной, по которой требуется получить разложение в ряд.
3. Выполните команду **Symbolics / Variable / Expand to Series** (Символика / Переменная / Разложить в ряд) (рис. 3.12).

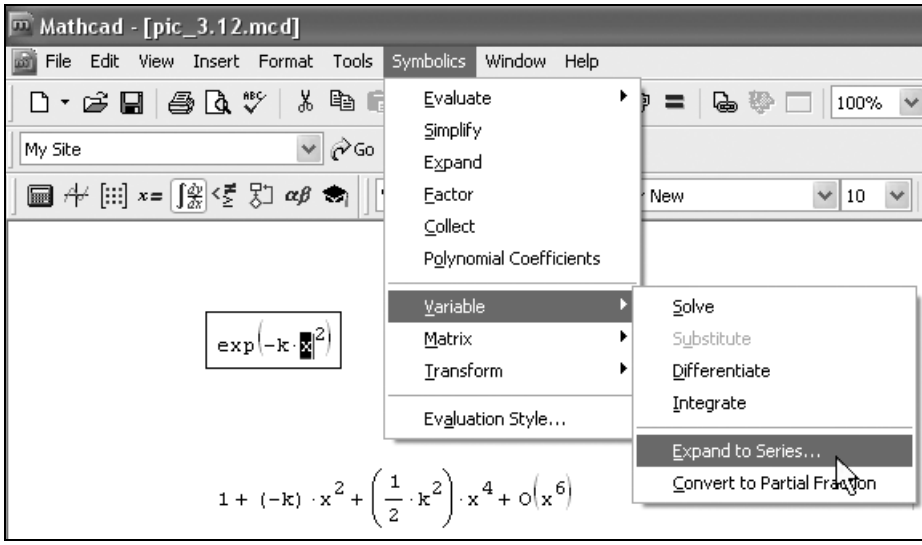



**Рис. 3.12.** Аналитическое разложение выражения в ряд по переменной x

4. В появившемся диалоговом окне **Expand to Series** (Разложить в ряд) введите желаемый порядок аппроксимации (**Order of Approximation**) и нажмите кнопку **ОК**.

Результат разложения появится под выражением (он показан на рис. 3.12, внизу).

***Примечание***

Не забывайте, что разложение строится только в точке `x=0`. Чтобы получить разложение в другой точке `x=a`, можно, к примеру, подставить вместо переменной `x` значение `x-a`.

## 3.5.2. Оператор разложения в ряд

Для разложения в ряд альтернативным способом, с помощью оператора символьного вывода, используйте ключевое слово `series`, вставляя его одноименной кнопкой панели **Symbolic** (Символика). После ключевого слова `series` через запятую указывается имя переменной, по которой производится разложение, и порядок аппроксимации (листинги 3.21 и 3.22).

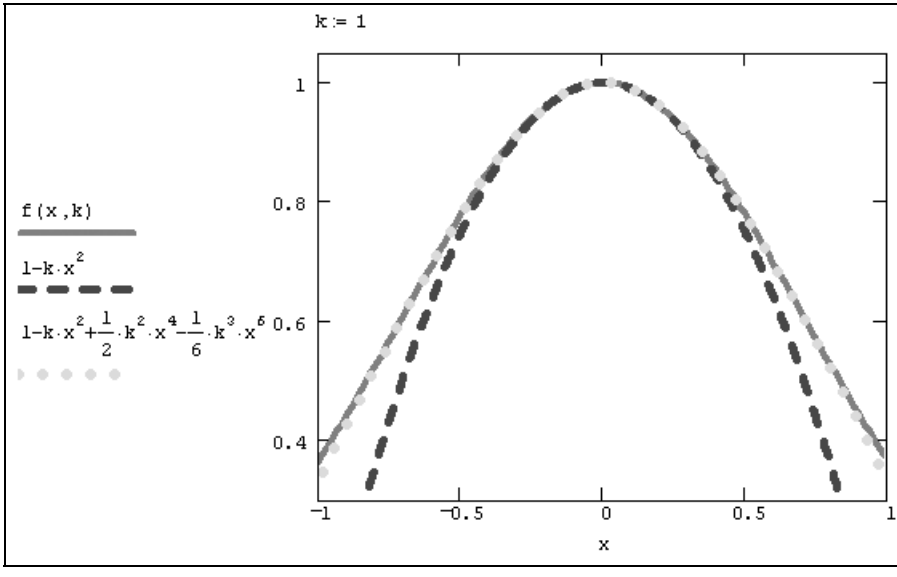



**Рис. 3.13.** График разложений функции в ряд в зависимости от порядка аппроксимации (продолжение листинга 3.21)

Сравнить поведение исходной функции и нескольких ее разложений в ряд Тейлора (с разными порядками аппроксимации) можно по графику, приведенному на рис. 3.13.

Видно, что разложение в ряд хорошо работает в окрестности точки x=0, а по мере удаления от нее все сильнее и сильнее отличается от функции. Естественно, что чем выше порядок аппроксимации, тем ближе к исходной функции располагается соответствующее разложение Тейлора.

**Листинг 3.21. Разложение функции в ряд с разным порядком аппроксимации**

$$f(x,k) := \exp\left(-k\cdot x^2\right)$$

$$f(x,k)\ \text{series},x,3 \rightarrow 1 - k\cdot x^2$$

$$f(x,k)\ \text{series},x,5 \rightarrow 1 - k\cdot x^2 + \frac{1}{2}\cdot k^2\cdot x^4$$

$$f(x,k)\ \text{series},x,7 \rightarrow 1 - k\cdot x^2 + \frac{1}{2}\cdot k^2\cdot x^4 - \frac{1}{6}\cdot k^3\cdot x^6$$

**Листинг 3.22. Разложение функции нескольких переменных в ряд по разным переменным**

$$f(x,k) := \exp\left(-k\cdot x^2\right)$$

$$f(x,k)\ \text{series},k,4 \rightarrow 1 - k\cdot x^2 + \frac{1}{2}\cdot k^2\cdot x^4 - \frac{1}{6}\cdot k^3\cdot x^6$$

$$f(x,k)\ \text{series},x,4 \rightarrow 1 - k\cdot x^2$$

# Глава 4

# Интегрирование

С одной стороны, численное интегрирование — одна из самых простых, с вычислительной точки зрения, операций, с другой — аналитически проинтегрировать можно далеко не каждую функцию. Всегда помните об этом, когда вы сталкиваетесь с численным или аналитическим интегрированием.

## 4.1. Определенный интеграл

Интегрирование в Mathcad реализовано в виде вычислительного оператора. Допускается вычислять интегралы от скалярных функций в пределах интегрирования, которые также должны быть скалярными. Несмотря на то, что пределы интегрирования обязаны быть действительными, подынтегральная функция может иметь и комплексные значения, поэтому и значение интеграла может быть комплексным.

### 4.1.1. Оператор интегрирования

Интегрирование, как и дифференцирование, и множество других математических действий, устроено в Mathcad по принципу "как пишется, так и вводится". Чтобы вычислить определенный интеграл, следует напечатать его обычную математическую форму в документе. Делается это с помощью панели **Calculus** (Вычисления) нажатием кнопки со значком интеграла или вводом с клавиатуры сочетания клавиш <Shift>+<7> (или символа "&", что то же самое). Появится символ интеграла с несколькими местозаполнителями (рис. 4.1), в которые нужно ввести нижний и верхний интервалы интегрирования, подынтегральную функцию и переменную интегрирования.

### Примечание

Если пределы интегрирования имеют размерность, то она должна быть одной и той же для обоих пределов.

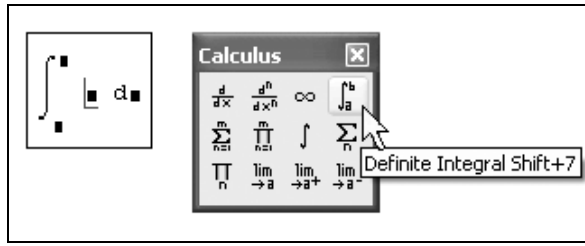



**Рис. 4.1.** Оператор интегрирования

Чтобы получить результат интегрирования, следует ввести знак равенства или символьного равенства. В первом случае интегрирование будет проведено численным методом, во втором — в случае успеха будет найдено точное значение интеграла с помощью символьного процессора Mathcad. Эти два способа иллюстрирует листинг 4.1. Конечно, символьное интегрирование возможно только для сравнительно небольшого круга несложных подынтегральных функций.

**Листинг 4.1. Численное и символьное вычисление определенного интеграла**

$$\int_0^{\pi} \exp\left(-x^2\right)\,dx = 0.886$$

$$\int_0^{\pi} \exp\left(-x^2\right)\,dx \rightarrow \frac{1}{2}\cdot \mathrm{erf}(\pi)\cdot \pi^{\frac{1}{2}} = 0.886$$

### Примечание 1

Можно вычислять интегралы с одним или обоими бесконечными пределами (листинг 4.2). Для этого на месте соответствующего предела введите символ бесконечности, воспользовавшись, например, той же самой панелью **Calculus** (Вычисления). Чтобы ввести $-\infty$ (минус бесконечность), добавьте знак минус к символу бесконечности, как к обычному числу.

**Листинг 4.2. Вычисление интеграла с бесконечными пределами**

$$\int_{-\infty}^{\infty} \exp\left(-x^2\right)\,dx \rightarrow \pi^{\frac{1}{2}}$$

### *Примечание 2*

Подынтегральная функция может зависеть от любого количества переменных. Именно для того чтобы указать, по какой переменной Mathcad следует вычислять интеграл, и нужно вводить ее имя в соответствующий местозаполнитель. Помните, что для численного интегрирования по одной из переменных предварительно следует задать значение остальных переменных, от которых зависит подынтегральная функция и для которых вы намерены вычислить интеграл (листинг 4.3).

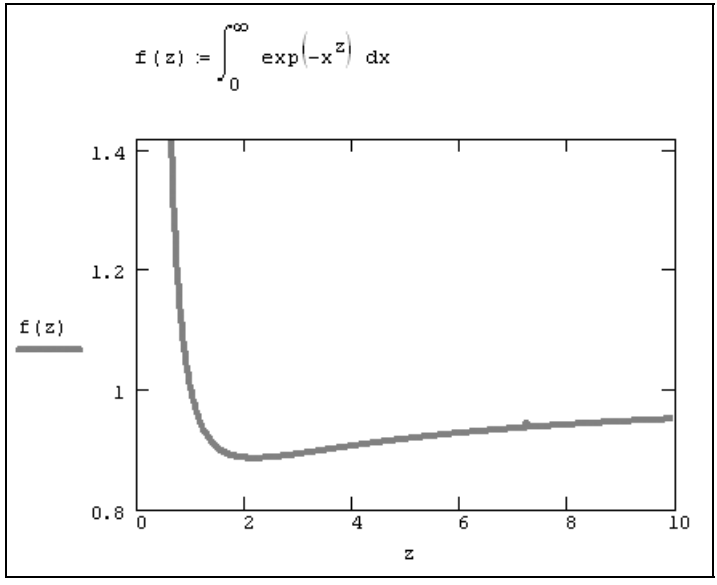



**Рис. 4.2.** Использование оператора интегрирования в функции пользователя

**Листинг 4.3. Интегрирование функции двух переменных по разным переменным**

$$\int_{a}^{b} \exp\left(-x\,z^2\right)\,dx \rightarrow \frac{-1}{z^2}\cdot\exp\left(-b\cdot z^2\right) + \frac{1}{z^2}\cdot\exp\left(-a\cdot z^2\right)$$

$$\int_a^b \exp\left(-x\,z^2\right)\,dz \rightarrow \frac{1}{2}\cdot \mathrm{erf}\left(b\cdot x^{\frac{1}{2}}\right)\cdot\frac{\pi^{\frac{1}{2}}}{x^{\frac{1}{2}}} - \frac{1}{2}\cdot \mathrm{erf}\left(a\cdot x^{\frac{1}{2}}\right)\cdot\frac{\pi^{\frac{1}{2}}}{x^{\frac{1}{2}}}$$

***Примечание 3***

Оператор интегрирования может использоваться точно так же, как и другие операторы: для определения функций, в циклах и при вычислении ранжированных переменных. Пример присваивания пользовательской функции `f(z)` значения определенного интеграла и вычисления нескольких ее значений приведен на рис. 4.2. На том же рисунке показано, как можно построить график результата интегрирования.

## ⓘ 4.1.2. О выборе алгоритма численного интегрирования

Результат численного интегрирования — это не точное, а приближенное значение интеграла, определенное с погрешностью, которая зависит от встроенной константы `TOL`. Чем она меньше, тем с лучшей точностью будет найден интеграл, но и тем больше времени будет затрачено на расчеты. По умолчанию `TOL=0.001`. Для того чтобы ускорить вычисления, можно установить большее значение `TOL`.

***Совет***

Если скорость расчетов имеет для вас принципиальное значение, например, при многократном вычислении интеграла внутри цикла, проявите осторожность, выбирая значение точности. Обязательно поэкспериментируйте на тестовом примере с характерной для ваших расчетов подынтегральной функцией. Посмотрите, как уменьшение константы `TOL` сказывается на погрешности интегрирования, вычислив интеграл для разных ее значений и выбрав оптимальное, исходя из соотношения точность/скорость вычислений.

Отдавайте себе отчет в том, что при вводе в редакторе Mathcad оператора численного интегрирования вы фактически создаете самую настоящую программу. Например, программой является первая строка листинга с рис. 4.2, просто основная ее часть скрыта от вашего взора разработчиками компании MathSoft. В большинстве случаев об этом не приходится специально задумываться, можно полностью положиться на Mathcad. Но иногда может потребоваться умение управлять параметрами этой программы, как мы уже рассмотрели на примере выбора константы `TOL`. Кроме нее пользователь имеет возможность выбирать сам алгоритм численного интегрирования.

Для этого:

1. Щелкните правой кнопкой мыши в любом месте на левой части вычисляемого интеграла.
2. В появившемся контекстном меню выберите один из имеющихся в наличии численных алгоритмов, например, **Romberg** (Ромберга) (рис. 4.3).

Обратите внимание, что перед тем как один из алгоритмов выбран впервые, как показано на рис. 4.3, флажок проверки в контекстном меню установлен возле пункта **AutoSelect** (Автоматический выбор). Это означает, что алгоритм определяется Mathcad, исходя из анализа пределов интегрирования и особенностей подынтегральной функции. Как только один из алгоритмов выбран, этот флажок сбрасывается, а избранный алгоритм отмечается точкой.

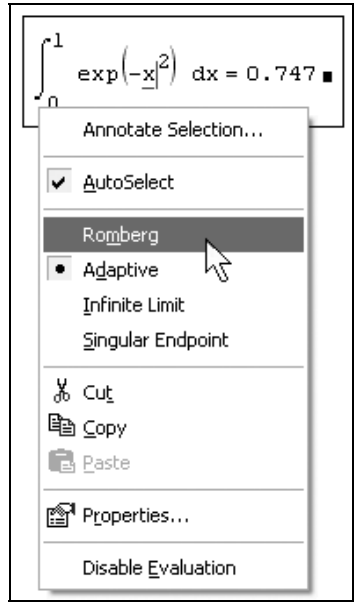



**Рис. 4.3.** Выбор алгоритма численного интегрирования производится при помощи контекстного меню

Разработчиками Mathcad запрограммированы четыре численных метода интегрирования:

- **Romberg** (Ромберга) — для большинства функций, не содержащих особенностей;
- **Adaptive** (Адаптивный) — для функций, быстро меняющихся на интервале интегрирования;

- **Infinite Limit** (Бесконечный предел) — для интегралов с бесконечными пределами;
- **Singular Endpoint** (Сингулярная граница) — для интегралов с сингулярностью на конце (применяется модифицированный алгоритм Ромберга для функций, не определенных на одном или обоих концах интервала интегрирования).

Старайтесь все-таки оставить выбор численного метода за Mathcad, установив флажок **AutoSelect** (Автоматический выбор) в контекстном меню. Попробовать другой метод можно, например, чтобы сравнить результаты расчетов в специфических случаях, когда у вас закрадываются сомнения в их правильности.

Если подынтегральная функция "хорошая", т. е. не меняется на интервале интегрирования слишком быстро, не имеет особенностей и не обращается в бесконечность, то численное решение интеграла не принесет никаких неприятных сюрпризов.

### 4.1.3. О традиционных алгоритмах интегрирования

Прежде чем перейти к изложению метода численного интегрирования, реализованного в Mathcad, скажем несколько слов об основных принципах численного интегрирования. Исходя из геометрического смысла определенного интеграла функции `f(x)` как площади фигуры, образованной графиком этой функции и осью `X`, можно предложить самый простой способ интегрирования "хорошей" функции — применить формулу прямоугольников. С ее помощью площадь упомянутой искомой фигуры подсчитывается как сумма элементарных прямоугольников, множеством которых заменяется подынтегральная функция `f(x)`.

Иллюстрация метода прямоугольников приведена на рис. 4.4. Для подсчета интеграла `I` интервал интегрирования `[a,b]` разбивается на `N` отрезков. На каждом `i`-м отрезке `f(x)` заменяется прямоугольником с шириной `h` и высотой $f(x_i)$. Площадь каждого из этих элементарных прямоугольников составляет $h \cdot f(x_i)$, а их сумма `S` может считаться приближением к искомому интегралу `I`. Несложно показать, что при $N \to \infty$ множество элементарных прямоугольников стремится к искомой фигуре, образованной подынтегральной функцией, а значение $S \to I$, причем погрешность (отличие `S` от точного значения `I`) составляет $o(h^2)$.

Можно воспринимать смысл алгоритма прямоугольников в замене исходной подынтегральной функции другой, близкой к ней (в данном случае кусочно-непрерывной) функцией, интеграл от которой легко подсчитать аналитически. Принцип более точных методов интегрирования как раз и состоит в интерполяции подынтегральной функции `f(x)` некоторой близкой зависимостью `y(x)` и расчете интеграла уже от этой функции. Важно, чтобы при этом, во-первых, интеграл от `y(x)` мог быть точно рассчитан аналитическими методами; и, во-вторых, функция `f(x)` была бы по возможности ближе к `y(x)`, чтобы уменьшить погрешность.

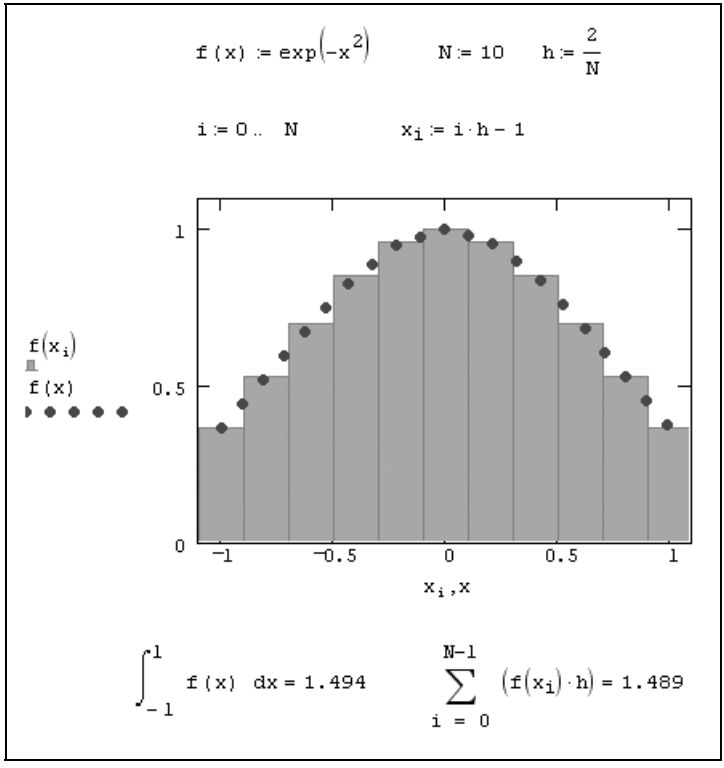



**Рис. 4.4.** Реализация алгоритма прямоугольников

Очевидно, что наиболее простой алгоритм заключается в интерполяции подынтегральной функции на каждом из `N` шагов интегрирования `f(x)` каким-либо полиномом `y(x)`. Известно, что могут быть предложены различные пути построения интерполирующих полиномов, отличающихся, в частности, порядком. Например, *полиномы Лагранжа* строятся при интерполяции `f(x)` в

n точках на каждом из N элементарных интервалов интегрирования. Семейство классических алгоритмов интегрирования в этом случае называется *методами Ньютона—Котеса*. Заметим, что при n=1 полиномом является прямая линия, и мы имеем метод трапеций; при n=2 интерполирующим полиномом на каждом шаге интегрирования будет квадратичная парабола, и мы получим *алгоритм Симпсона* и т. д.

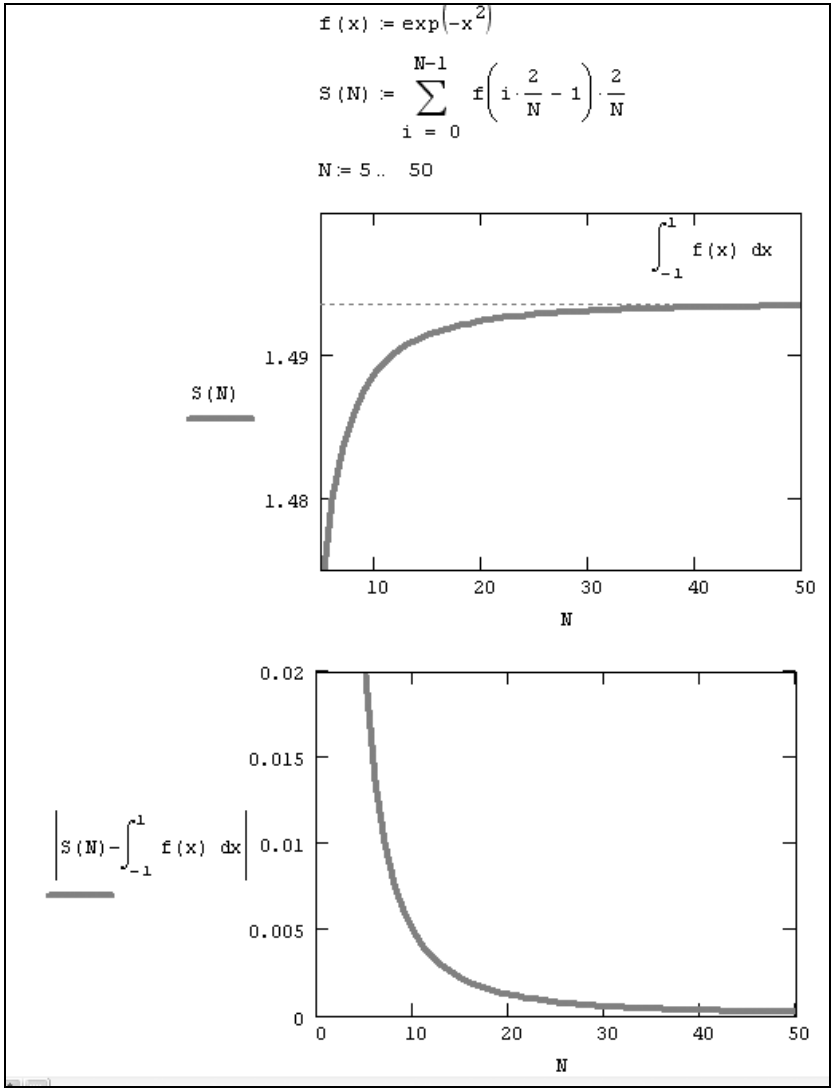



**Рис. 4.5.** Оценка погрешности алгоритма прямоугольников

***Примечание***

Дополнительные сведения об алгоритме полиномиальной интерполяции приведены в *главе 13*.

Недостатком перечисленных традиционных алгоритмов являются затруднения в количественной оценке погрешности. Аналитические формулы для погрешности включают, помимо множителя $h^K$, задающего, собственно, порядок аппроксимации метода, сомножитель, характеризующий величину производной (определенного высшего порядка) подынтегральной функции. Оценить ее значение при конкретных расчетах очень сложно, и поэтому, соответственно, сложно вычислить суммарную погрешность алгоритма. В то же время сведения о величине погрешности являются очень важными, и, чаще всего, желательно иметь их количественную оценку, чтобы оптимально выбрать число разбиений интервала интегрирования `N`.

Для апостериорной оценки погрешности можно, например, применять анализ зависимости `S(N)`, подсчитанной для нескольких значений `N` (рис. 4.5). Зная, что, с одной стороны, `S(N)` изменяется по определенному степенному закону $N^{-K}$, и, с другой, `S(N)`$\rightarrow$`I` (к точному значению интеграла), можно довольно точно определить погрешность метода. На нижнем графике рис. 4.5 приведена зависимость погрешности от `N` (правда, в данном случае мы использовали для наглядности графика точное значение интеграла, которое в практических случаях, конечно, неизвестно). Именно с подобной процедурой и связан алгоритм расчета определенных интегралов, использованный в Mathcad, который будет представлен в следующем разделе.

## 4.1.4. Алгоритм Ромберга

После сделанных вводных замечаний приведем основные идеи итерационного *алгоритма Ромберга*, который применяется в системе Mathcad для выполнения операции численного интегрирования.

- Сначала строится несколько интерполирующих полиномов, которые заменяют на интервале интегрирования подынтегральную функцию `f(x)`. В качестве первой итерации полиномы вычисляются по 1, 2 и 4 интервалам. Например, как уже отмечалось выше, первый полином, построенный по 1 интервалу, — это просто прямая линия, проведенная через две граничные точки интервала интегрирования, второй — квадратичная парабола и т. д.
- Интеграл от каждого полинома с известными коэффициентами легко вычисляется аналитически. Таким образом, определяется последовательность интегралов от интерполирующих полиномов: $I_1, I_2, I_4, \ldots$. Например, по правилу трапеций $I_1=(b-a)\cdot(f(a)+f(b))/2$ и т. д.

- Из-за интерполяции по разному числу точек вычисленные интегралы $I_1, I_2, \ldots$ несколько отличаются друг от друга. Причем чем больше точек используется для интерполяции, тем интеграл от интерполяционного полинома ближе к искомому интегралу $I$, стремясь к нему в пределе бесконечного числа точек. Поэтому определенным образом осуществляется экстраполяция последовательности $I_1, I_2, I_4, \ldots$ до нулевой ширины элементарного интервала. Результат этой экстраполяции $J$ принимается за приближение к вычисляемому интегралу.
- Осуществляется переход к новой итерации с помощью еще более частого разбиения интервала интегрирования, добавления нового члена последовательности интерполирующих полиномов и вычисления нового ($N$-го) приближения Ромберга $J^N$.
- Чем больше количество точек интерполяции, тем ближе очередное приближение Ромберга к вычисляемому интегралу и, соответственно, тем меньше оно отличается от приближения предыдущей итерации. Как только разница между двумя последними итерациями $|J^N - J^{N-1}|$ становится меньше погрешности $TOL$ или меньше $TOL \cdot |J^N|$, итерации прерываются, и $J^N$ появляется на экране в качестве результата интегрирования.

# 4.2. Неопределенный интеграл

Предыдущий раздел был посвящен проблеме поиска определенного интеграла, т. е. числового значения, равного площади фигуры, образованной графиком подынтегральной функции и осью $X$ (см. рис. 4.4). Задача нахождения неопределенного интеграла намного сложнее, поскольку связана с поиском функции, производная от которой равна исходной подынтегральной функции. Решение этой задачи целиком возложено на символьный процессор Mathcad.

## 4.2.1. Символьное интегрирование

Для того чтобы аналитически проинтегрировать некоторую функцию, следует ввести с панели **Calculus** (Вычисления) символ неопределенного интеграла, в появившемся в документе шаблоне заполнить местозаполнители и, наконец, ввести знак символьного равенства. В случае успеха по истечении некоторого времени расчетов справа от введенного выражения появится его аналитический результат (листинг 4.4). Если же функцию не удается проинтегрировать аналитически, введенное вами выражение будет просто продублировано (листинг 4.5).

### *Примечание*

Помните, что при символьном интегрировании допускается использовать в выражениях, которые вы вводите, различные параметры. Если перед выражением вы нигде не определяли их значения, то (в случае успешных вычислений) символьный процессор Mathcad выдаст аналитическую зависимость результата интегрирования от этих параметров (как в листинге 4.4 от параметра a).

**Листинг 4.4. Аналитическое вычисление неопределенного интеграла**

$$\int \exp\left(-a\cdot x^{2}\right)\,dx \rightarrow \frac{1}{2}\cdot\frac{\pi^{\frac{1}{2}}}{a^{\frac{1}{2}}}\cdot \mathrm{erf}\left(a^{\frac{1}{2}}\cdot x\right)$$

**Листинг 4.5. Аналитическое интегрирование невозможно**

$$\int \exp\left(-a\cdot x^{b}\right)\,dx \rightarrow \int \exp\left(-a\cdot x^{b}\right)\,dx$$

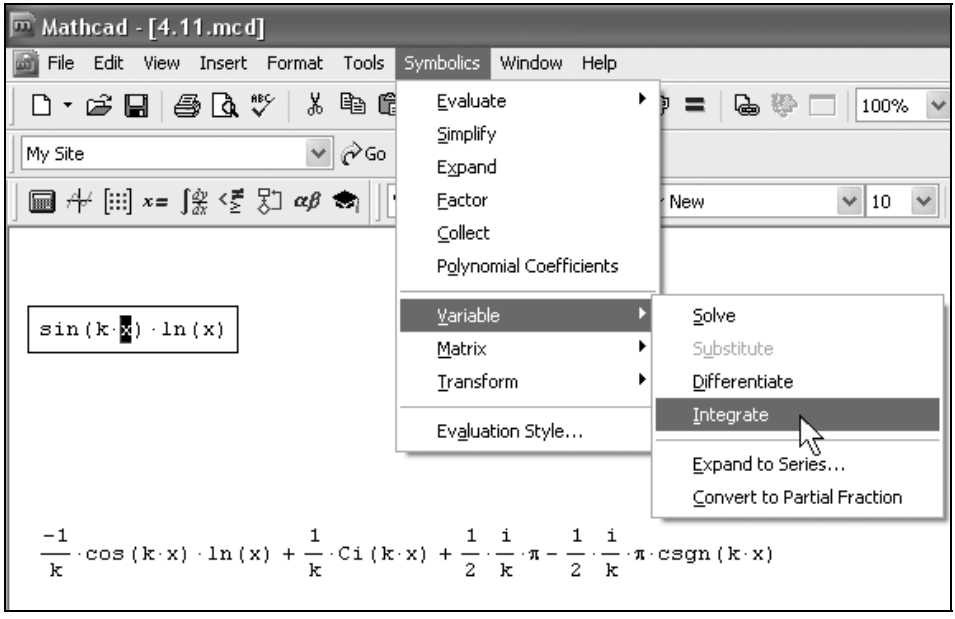



**Рис. 4.6.** Интегрирование выражения по переменной при помощи меню

### 4.2.2. Интегрирование при помощи меню

Для вычисления неопределенного интеграла от некоторого выражения по определенной переменной при помощи меню выделите в выражении переменную и выполните команду **Symbolics** / **Variable** / **Integrate** (Символика / Переменная / Интегрировать) (рис. 4.6). Вычисленное аналитическое представление неопределенного интеграла появится ниже. При этом результат может содержать как встроенные в Mathcad функции, так и другие спецфункции, которые нельзя непосредственно рассчитать в Mathcad, но символьный процессор "умеет" выдавать их в качестве результата некоторых аналитических операций.

## 4.3. Интегралы специального вида

Завершим разговор о приемах интегрирования в среде Mathcad примерами вычислений в некоторых специальных случаях, которые довольно часто встречаются в самых разнообразных областях математики.

### 4.3.1. Интегралы с бесконечными пределами

Как мы уже говорили (см. примечание 1 в *разд. 4.1.1* и листинг 4.2), для того, чтобы вычислить определенный интеграл с одним или обоими бесконечными пределами, достаточно ввести, пользуясь панелью **Calculus** (Вычисления), специально предусмотренный разработчиками символ бесконечности в нужные местозаполнители интервалов интегрирования.

### 4.3.2. Расходящиеся интегралы

Если интеграл расходится (равен бесконечности), то вычислительный процессор Mathcad может выдать сообщение об ошибке, выделив при этом оператор интегрирования, как обычно, красным цветом. Чаще всего ошибка будет иметь тип "Found a number with a magnitude greater than 10^307" (Найдено число, превышающее значение $10^{307}$) или "Can't converge to a solution" (Не сходится к решению). Листинг 4.6 демонстрирует невозможность численного расчета интеграла $\int_0^\infty \frac{1}{\sqrt{x}}\,dx$ (нижняя строка листинга). Тем не менее символьный процессор справляется с этим интегралом, совершенно правильно находя его бесконечное значение (верхняя строка листинга 4.6).

**Листинг 4.6. Вычисление расходящегося интеграла**

$$\int_0^{\infty} \frac{1}{\sqrt{x}}\,dx \to \infty$$

$$\int_0^{\infty} \frac{1}{\sqrt{x}}\,dx = \blacksquare$$

При попытке численного решения задачи листинга 4.6 методом, отличным от алгоритма вычисления интегралов с бесконечными пределами (**Infinite Limit**), получится неверное решение (листинг 4.7). А именно, вместо бесконечности будет выдано большое, но конечное число, немного не дотягивающее до численной бесконечности, являющейся для вычислительного процессора просто большим числом $10^{307}$. Отметим, что Mathcad в режиме автоматического выбора алгоритма (**AutoSelect**) предлагает для интегралов с бесконечным пределом именно алгоритм **Infinite Limit**.

**Листинг 4.7. Плохо выбранный численный алгоритм (в данном случае адаптивный) неверно находит расходящийся интеграл**

$$\int_0^{\infty} \frac{1}{\sqrt{x}}\,dx = 6.325 \times 10^{153}$$

## 4.3.3. Интеграл с переменным пределом

Символьный процессор предоставляет замечательные возможности аналитического вычисления интегралов, в том числе зависящих от параметров. Особую важность имеет вычисление интеграла *с переменным пределом* (верхним или нижним), для которого один из пределов интегрирования является переменной, отличной от переменной интегрирования (листинг 4.8). Нетрудно сообразить, что, с точки зрения символьного процессора, интеграл с переменным пределом является обычным определенным интегралом, зависящим от дополнительного параметра.

**Листинг 4.8. Аналитическое вычисление интеграла с переменным верхним пределом**

$$\int_0^z \frac{1}{\sqrt{x}}\,dx \rightarrow 2\cdot z^{\frac{1}{2}}$$

## 4.3.4. Кратные интегралы

*Кратным* называется интеграл функции многих переменных, берущийся по нескольким переменным. Для того чтобы вычислить кратный интеграл:

1. Введите, как обычно, оператор интегрирования.
2. В соответствующих местозаполнителях введите имя первой переменной интегрирования и пределы интегрирования по этой переменной.
3. На месте ввода подынтегральной функции введите еще один оператор интегрирования (рис. 4.7).
4. Точно так же введите вторую переменную, пределы интегрирования и подынтегральную функцию (если интеграл двукратный) или следующий оператор интегрирования (если более чем двукратный) и т. д., пока выражение с многократным интегралом не будет введено окончательно.

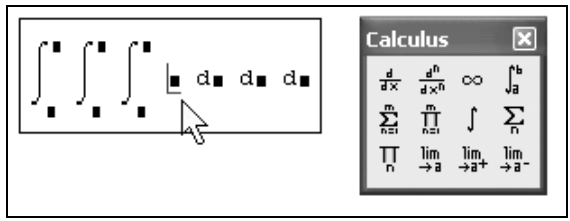



**Рис. 4.7.** Ввод нескольких операторов интегрирования для расчета кратного интеграла

Пример символьного и численного расчета двукратного интеграла в бесконечных пределах приведен в листинге 4.9. Обратите внимание, что символьный процессор "угадывает" точное значение интеграла π, а вычислительный определяет его приближенно и выдает в виде числа `3.142`.

**Листинг 4.9. Символьное и численное вычисления кратного интеграла**

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-\left(x^2+y^2\right)}\,dx\,dy \rightarrow \pi$$

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-\left(x^2+y^2\right)}\,dx\,dy = 3.142$$

***Внимание!***

Аккуратнее вводите в редакторе Mathcad кратные интегралы, если они имеют различные пределы интегрирования по разным переменным. Не перепутайте пределы, относящиеся к разным переменным. Если вы имеете дело с такого рода задачами, обязательно разберитесь с листингом 4.10, в котором символьный процессор вычисляет двукратный интеграл. В первой строке пределы интегрирования `[a, b]` относятся к переменной `y`, а во второй строке — к переменной `x`.

**Листинг 4.10. Символьное вычисление кратных интегралов**

$$\int_{a}^{b}\int_{-1}^{1} x + y^3\,dx\,dy \rightarrow \frac{1}{2}\cdot b^4 - \frac{1}{2}\cdot a^4$$

$$\int_{a}^{b}\int_{-1}^{1} x + y^3\,dy\,dx \rightarrow b^2 - a^2$$

## 4.3.5. Пример: длина дуги кривой

В заключение приведем пример использования вычислительного процессора Mathcad для расчета длины участка кривой, задаваемой некоторой функцией `f(x)` в промежутке между двумя значениями ее аргумента `a` и `b` (рис. 4.8). Решение данной простой задачи математического анализа приведено в листинге 4.11, причем формула, по которой рассчитывается длина дуги, приведена в третьей (последней) строке листинга. Обратите внимание, что для получения результата необходимо применить и операцию численного интегрирования, и дифференцирования.

**Листинг 4.11. Расчет длины дуги кривой**

$$f(x) := x^2 - \frac{x^3}{2}$$

$$a := 0 \qquad b := 2$$

$$\int_a^b \sqrt{1 + \left(\frac{d}{dx} f(x)\right)^2}\, dx = 2.42$$

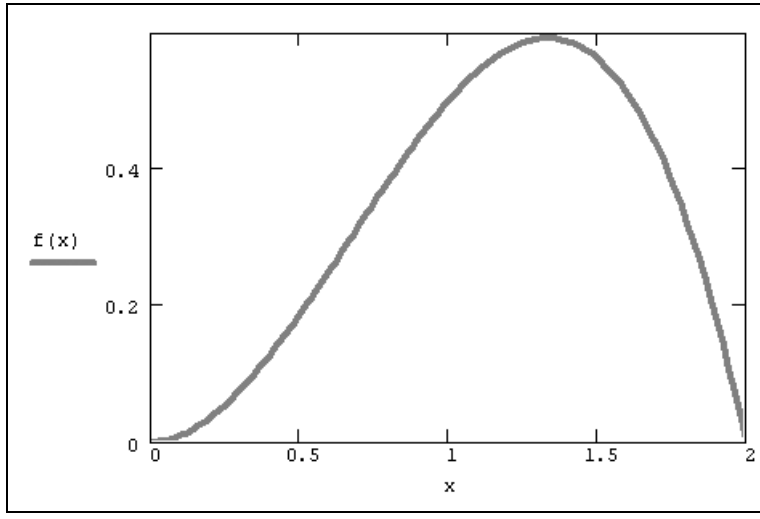



**Рис. 4.8.** График функции определяет дугу некоторой длины (продолжение листинга 4.11)

# 4.4. Интеграл Фурье

Обратимся теперь к характерным проблемам вычислительной математики, связанным с (аналитическим или численным) вычислением интегралов определенного вида. Задачи, о которых мы собираемся рассказать, тесно связаны с алгоритмами обработки данных, поэтому содержание этого раздела будет перекликаться с материалом *главы 14* (*см. разд. 14.1*). Благодаря обширному применению для вычисления таких интегралов разработаны специфические алгоритмы, несравненно более эффективные, чем традиционные (*см. разд. 4.1.4*), причем некоторые из них имеются в арсенале Mathcad

в форме встроенных функций вычислительного процессора и соответствующих операций символьного процессора.

Наиболее широко распространенным интегральным преобразованием является преобразование Фурье, которое представляет функцию f(x) в виде интеграла по гармоническим функциям, называемого *интегралом Фурье*:

$$F(\omega) = \int_{-\infty}^{\infty} f(x) \cdot \exp(-i\omega x)\, dx.$$

Функция F(ω) называется также *преобразованием Фурье*, или *Фурье-спектром* исходной функции f(x). Ее аргумент ω имеет смысл частоты соответствующей гармонической составляющей f(x). Важно отметить, что функция, выражающая преобразование Фурье, комплексна, даже если f(x) является действительной.

## ⓘ 4.4.1. Об интегральных преобразованиях функций

Вообще говоря, интегральные преобразования по определению ставят в соответствие некоторой функции f(x) другую функцию от другого аргумента F(ω). Причем это соответствие f(x)→F(ω) задается интегральной зависимостью. Символьный процессор Mathcad позволяет осуществлять три вида интегральных преобразований функций — преобразование Фурье, Лапласа и Z-преобразование. Наряду с прямыми преобразованиями имеется возможность совершать любое из этих трех обратных преобразований, т. е. F(ω)→f(x).

Аналитически все интегральные преобразования выполняются аналогично символьному интегрированию (*см. разд. 4.2.2*). Для вычисления преобразования выражения выделяется переменная, по которой будет осуществляться преобразование, и затем выбирается соответствующий пункт меню. Преобразования с применением оператора символьного вывода используются с одним из соответствующих ключевых слов, вслед за которым требуется указать имя нужной переменной.

Приведем примеры символьного расчета каждого из трех интегральных преобразований, а также расскажем о численных методах Фурье- и вейвлет-преобразований.

## 4.4.2. Аналитическое преобразование Фурье

Аналитический расчет преобразования Фурье при помощи меню показан на рис. 4.9, для чего используется команда меню **Symbolics / Transform / Fou-**

**rier** (Символика / Преобразование / Фурье). В листинге 4.12 приведены два примера вычисления прямого преобразования Фурье с применением ключевого слова `fourier` и оператора символьного вывода →. Листинг 4.13 иллюстрирует применение обратного преобразования Фурье и последующее прямое преобразование полученного выражения, в результате которого получается исходная функция.

***Внимание!***

Помните о том, что результаты символьных преобразований могут включать спецфункции, которые не являются встроенными функциями Mathcad и поэтому не могут впоследствии использоваться в вычислениях. Их имена представляют собой не более, чем текстовое сообщение.

**Листинг 4.12. Примеры прямого преобразования Фурье**

$$\cos\left(k^2 \cdot x\right) \text{ fourier , x } \rightarrow \pi \cdot \Delta\left(\omega - k^2\right) + \pi \cdot \Delta\left(\omega + k^2\right)$$

$$\cos\left(k^2 \cdot x\right) \text{ fourier , k } \rightarrow \left(\frac{\pi}{|x|}\right)^{\frac{1}{2}} \cdot \exp\left(\frac{-1}{4} \cdot \frac{\omega^2}{|x|}\right)$$

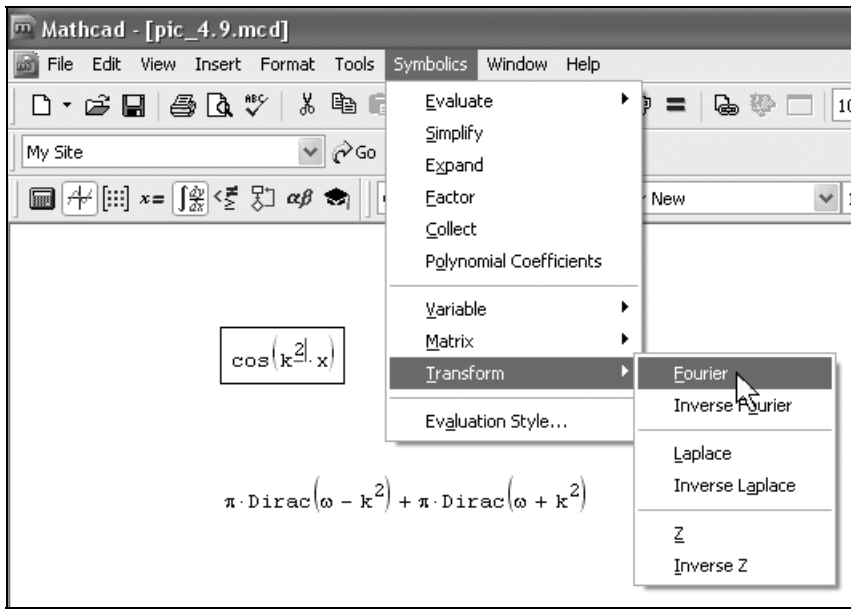



**Рис. 4.9.** Расчет Фурье-преобразования при помощи меню

**Листинг 4.13. Обратное и прямое преобразования Фурье**

$$\frac{1}{\omega}\ \text{invfourier}\ ,\omega\ \rightarrow \frac{1}{2}\cdot i\cdot\left(\Phi(t)-\Phi(-t)\right)$$

$$\frac{-1}{2}\cdot i\cdot\left(-\Phi(t)+\Phi(-t)\right)\ \text{fourier}\ ,t\ \rightarrow \frac{1}{\omega}$$

## 4.4.3. Дискретное преобразование Фурье

В предыдущем разделе рассказывалось о возможностях символьного процессора Mathcad, позволяющего осуществить аналитическое преобразование Фурье-функции, заданной формулой. Между тем огромный пласт задач вычислительной математики связан с расчетом интегралов Фурье для функций, либо заданных таблично (например, представляющих собой результаты какого-либо эксперимента), либо функций, проинтегрировать которые аналитически не удается. В данном случае вместо символьных преобразований приходится применять численные методы интегрирования, связанные с дискретизацией подынтегральной функции и называемые потому *дискретным Фурье-преобразованием*.

В численном процессоре Mathcad дискретное преобразование Фурье реализовано при помощи популярнейшего алгоритма *быстрого преобразования Фурье* (сокращенно БПФ). Этот алгоритм реализован в нескольких встроенных функциях Mathcad, различающихся только нормировками:

- `fft(y)` — вектор прямого преобразования Фурье;
- `FFT(y)` — вектор прямого преобразования Фурье в другой нормировке;
- `ifft(ω)` — вектор обратного преобразования Фурье;
- `IFFT(ω)` — вектор обратного преобразования Фурье в другой нормировке:
  - `y` — вектор действительных данных, взятых через равные промежутки значений аргумента;
  - ω — вектор действительных данных Фурье-спектра, взятых через равные промежутки значений частоты.

***Внимание!***

Аргумент прямого Фурье-преобразования, т. е. вектор `y`, должен иметь ровно $2^n$ элементов (`n` — целое число). Результатом является вектор с $1+2^{n-1}$ элементами. И наоборот, аргумент обратного Фурье-преобразования должен иметь $1+2^{n-1}$ элементов, а его результатом будет вектор из $2^n$ элементов. Если

число данных не совпадает со степенью 2, то необходимо дополнить недостающие элементы нулями.

В листинге 4.14 показан пример расчета Фурье-спектра для модельной функции f(x), представляющей собой сумму двух синусоид разной амплитуды (верхний график на рис. 4.10). Расчет проводится по N=128 точкам, причем предполагается, что интервал дискретизации данных $y_i$ равен h. В предпоследней строке листинга корректно определяются соответствующие значения частот $\Omega_i$, а в последней применяется встроенная функция FFT. Полученный график Фурье-спектра показан на рис. 4.10 (снизу). Обратите внимание, что результаты расчета представляются в виде его модуля, поскольку сам спектр, как уже отмечалось, является комплексным. Очень полезно сравнить полученные амплитуды и местоположение пиков спектра с определением синусоид в начале листинга.

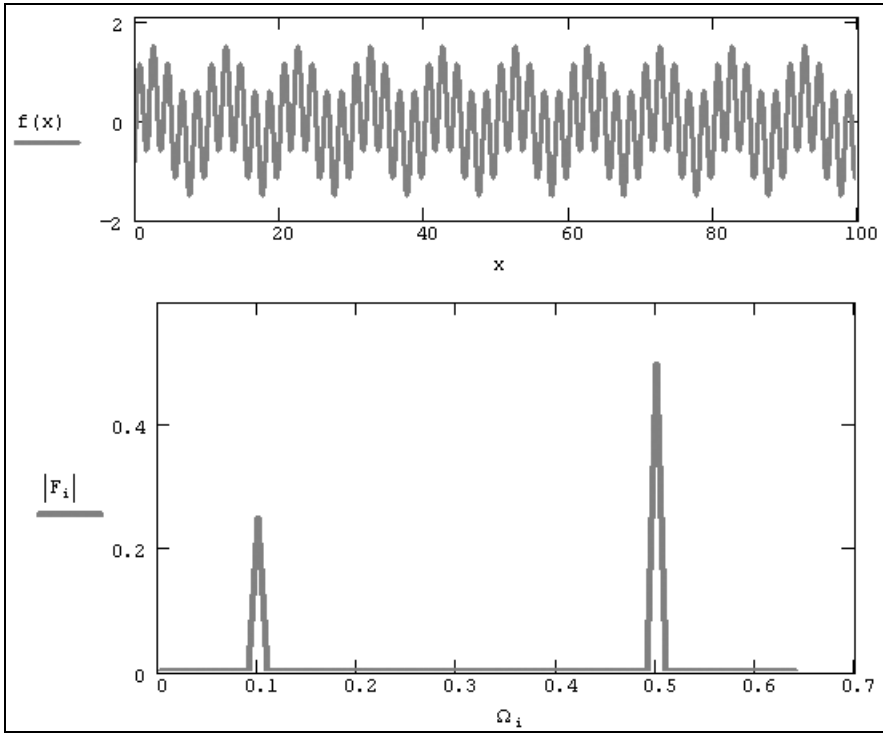



**Рис. 4.10.** Модельная функция и ее преобразование Фурье (продолжение листинга 4.14)

***Примечание***

Более подробную информацию о свойствах и практике применения Фурье-преобразования вы найдете в *главе 14*.

**Листинг 4.14. Дискретное преобразование Фурье (алгоритм БПФ) модельного сигнала**

$$f(x) := 0.5 \cdot \sin(2\pi \cdot 0.1 \cdot x) + 1 \cdot \sin(2\pi \cdot 0.5 \cdot x)$$

$$L := 100 \qquad N := 128 \qquad h := \frac{L}{N}$$

$$i := 0 .. N - 1$$

$$x_i := i \cdot h \qquad y_i := f(x_i) \qquad \Omega_i := \frac{i}{L}$$

$$F := FFT(y)$$

## 4.4.4. Преобразование Фурье комплексных данных

Алгоритм быстрого преобразования Фурье для комплексных данных встроен в соответствующие функции, в название которых входит литера "C":

- `cfft(y)` — вектор прямого комплексного преобразования Фурье;
- `CFFT(y)` — вектор прямого комплексного преобразования Фурье в другой нормировке;
- `icfft(y)` — вектор обратного комплексного преобразования Фурье;
- `ICFFT(ω)` — вектор обратного комплексного преобразования Фурье в другой нормировке:
  - `y` — вектор данных, взятых через равные промежутки значений аргумента;
  - ω — вектор данных Фурье-спектра, взятых через равные промежутки значений частоты.

Функции действительного преобразования Фурье используют тот факт, что в случае действительных данных спектр получается симметричным относительно нуля, и выводят только его половину (*см. разд. 4.4.3*). Поэтому, в частности, по 128 действительным данным получалось всего 65 точек спектра Фурье. Если к тем же данным применить функцию комплексного преобразо-

вания Фурье (рис. 4.11), то получится вектор из 128 элементов. Сравнивая рис. 4.10 и 4.11, можно уяснить соответствие между результатами действительного и комплексного Фурье-преобразования.

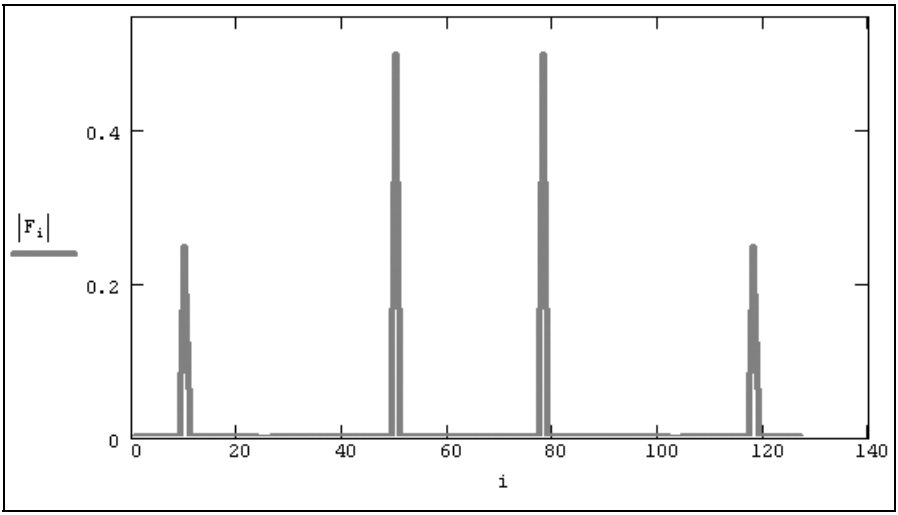



**Рис. 4.11.** Комплексное преобразование Фурье (продолжение листинга 4.14)

## 4.4.5. Двумерное преобразование Фурье

В Mathcad имеется возможность вычислять не только одномерное преобразование Фурье функции `f(x)`, но и двумерное преобразование функции двух переменных `f(x,y)`. Иными словами, допускается применять встроенные функции комплексного дискретного преобразования Фурье не только к одномерным, но и к двумерным массивам, т. е. матрицам. Соответствующий пример приведен в листинге 4.15 и на рис. 4.12 в виде графика поверхности исходных данных (верхний график) и рассчитанного двумерного Фурье-спектра (нижний график).

**Листинг 4.15. Двумерное дискретное преобразование Фурье**

```
N := 32

i := 0 .. N – 1          j := 0 .. N – 1
```

$$Y_{i,j} := \exp\left[-\frac{(i+j)^2}{N}\right]$$

$$F := CFFT(Y)$$

$$F_{i,j} := |F_{i,j}|$$

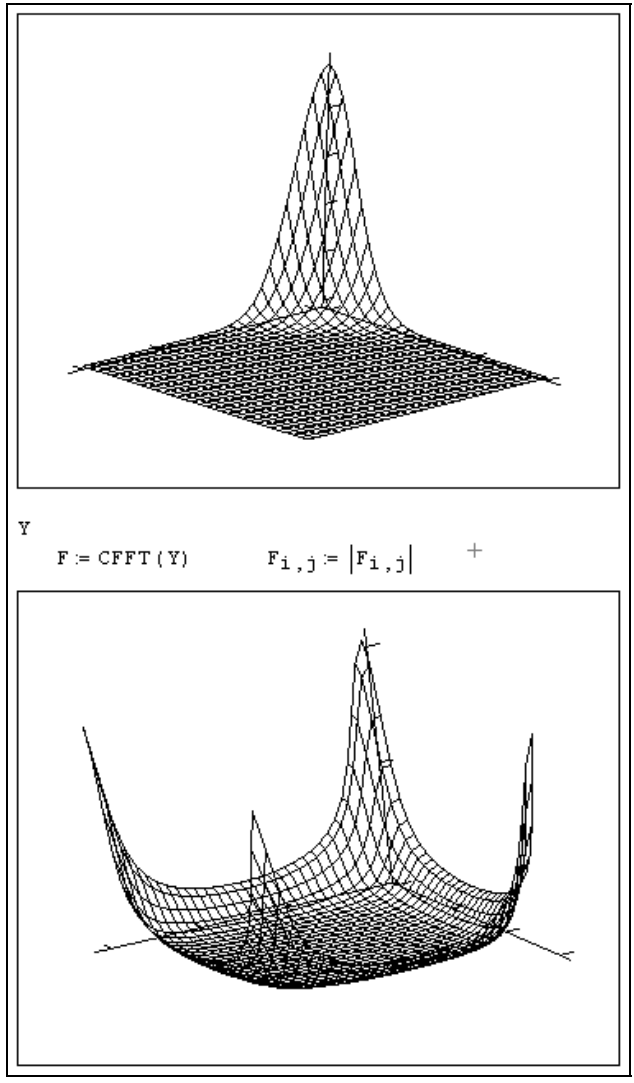



**Рис. 4.12.** Функция двух переменных и ее двумерное преобразование Фурье (продолжение листинга 4.15)

# 4.5. Другие интегральные преобразования

Рассмотрим в завершение главы, посвященной интегрированию, еще три преобразования, которые часто применяются помимо интеграла Фурье. Отметим, что преобразование Лапласа и Z-преобразование встречаются в практических задачах вычислительной математики относительно редко, а вот вейвлет-преобразование, теория которого появилась достаточно недавно, постепенно выходит на лидирующие позиции в проблемах обработки данных.

## 4.5.1. Преобразование Лапласа

Преобразованием Лапласа называют интеграл от `f(x)` следующего вида:

$$F(s) = \int_0^{\infty} f(x) \cdot \exp(-sx)\, dx.$$

Рассчитывается преобразование Лапласа совершенно аналогично Фурье-преобразованию (*см. разд. 4.4*). Примеры преобразования Лапласа приведены в листинге 4.16 и на рис. 4.13.

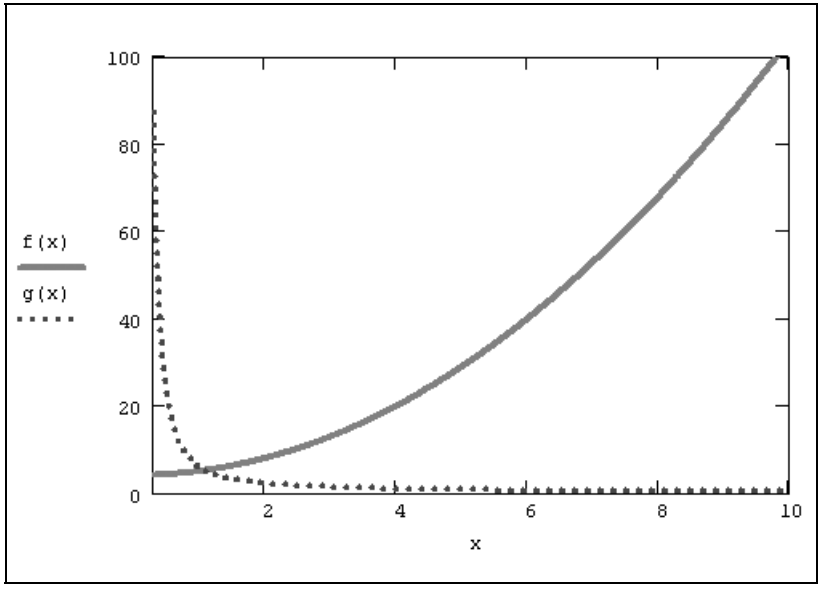



**Рис. 4.13.** Прямое и обратное преобразование Лапласа (продолжение листинга 4.16)

**Листинг 4.16. Двумерное преобразование Лапласа**

$$f(x) := x^2 + 4$$

$$g(s) := f(x) \text{ laplace}, x \rightarrow \frac{2}{s^3} + \frac{4}{s}$$

$$g(s) \text{ invlaplace}, s \rightarrow t^2 + 4$$

## 4.5.2. Z-преобразование

Z-преобразование функции `f(x)` определяется не интегралом, а бесконечной суммой следующего вида:

$$F(z) = \sum_{n=0}^{\infty} f(n) \cdot z^{-n}.$$

Пример Z-преобразования приведен в листинге 4.17, а его результаты — на рис. 4.14.

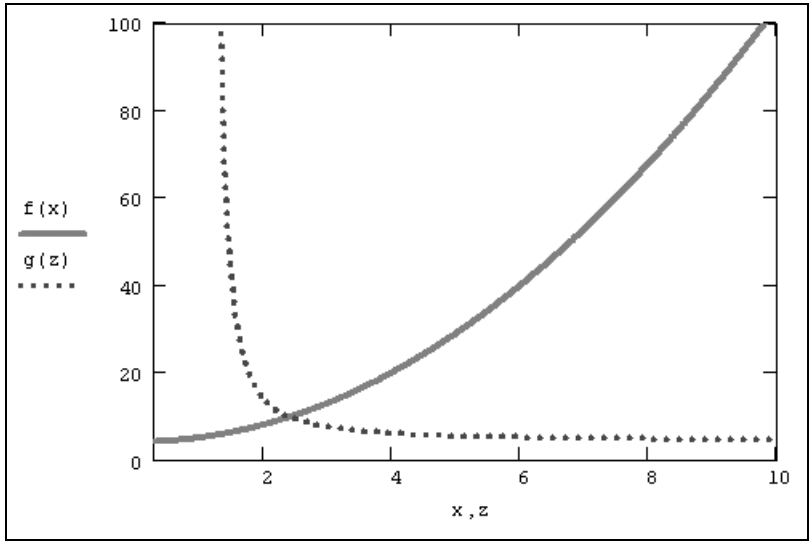



**Рис. 4.14.** Прямое и обратное Z-преобразования (продолжение листинга 4.17)

**Листинг 4.17. Прямое и обратное Z-преобразования**

$$f(x) := x^2 + 4$$

$$g(z) := f(x)\ \text{ztrans}, x \rightarrow z \cdot \frac{\left(-7 \cdot z + 5 + 4 \cdot z^2\right)}{(z-1)^3}$$

$$g(z)\ \text{invztrans}, z \rightarrow 4 + n^2$$

## 4.5.3. Вейвлет-преобразование

В последнее время возрос интерес к другим интегральным преобразованиям, в частности, к *вейвлет-преобразованию* (или *дискретному волновому преобразованию*). Оно применяется, главным образом, для анализа нестационарных сигналов и для многих задач подобного рода оказывается более эффективным, чем преобразование Фурье.

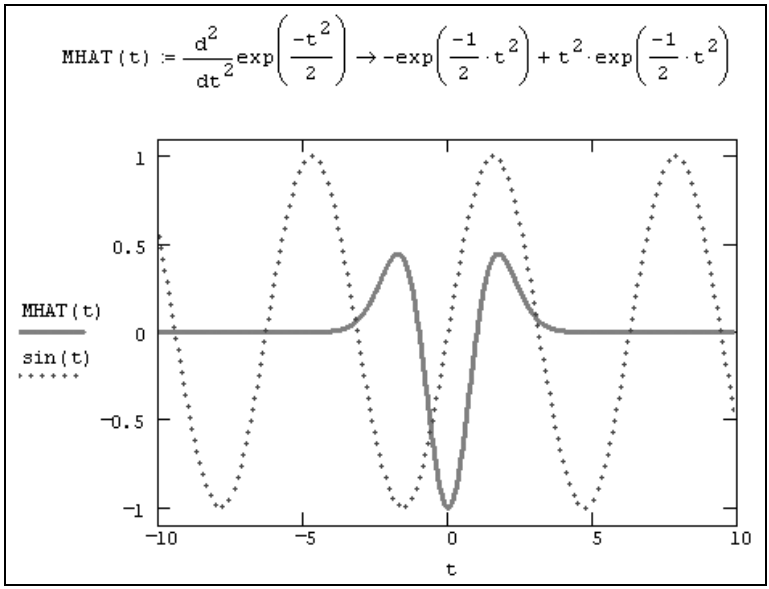



**Рис. 4.15.** Сравнение синусоиды и вейвлетобразующей функции

Основным отличием вейвлет-преобразования является разложение данных не по синусоидам (как для преобразования Фурье), а по другим функциям, на-

зываемым *вейвлетобразующими*. Вейвлетобразующие функции, в противоположность бесконечно осциллирующим синусоидам, локализованы в некоторой ограниченной области своего аргумента, а вдали от нее равны нулю или ничтожно малы. Пример такой функции, называемой "мексиканской шляпой", показан на рис. 4.15.

По своему математическому смыслу вейвлет-спектр имеет не один аргумент, а два. Помимо частоты, вторым аргументом `b` является место локализации вейвлетобразующей функции. Поэтому `b` имеет ту же размерность, что и `x`.

## Встроенная функция вейвлет-преобразования

Mathcad имеет одну встроенную функцию для расчета вейвлет-преобразования на основе вейвлетобразующей функции Добеши:

- `wave(y)` — вектор прямого вейвлет-преобразования Добеши;
- `iwave(v)` — вектор обратного вейвлет-преобразования Добеши:
  - `y` — вектор данных, взятых через равные промежутки значений аргумента;
  - `v` — вектор данных вейвлет-спектра.

Аргумент функции вейвлет-преобразования, т. е. вектор `y`, должен так же, как и в преобразовании Фурье, иметь ровно $2^n$ элементов (`n` — целое число). Результатом функции `wave` является вектор, скомпонованный из нескольких коэффициентов `C` двухпараметрического вейвлет-спектра. Особенности использования функции `wave` иллюстрируются листингом 4.18, где в качестве модельной функции взята сумма двух синусоид, график которой был изображен на рис. 4.10. Результаты вычислений вейвлет-спектра Добеши представлены в виде трех семейств его коэффициентов на рис. 4.16.

**Листинг 4.18. Вычисление вейвлет-спектра Добеши модельного сигнала**

$$f(t) := 0.5 \cdot \sin(2\pi \cdot 0.1 \cdot t) + 1 \cdot \sin(2\pi \cdot 0.5 \cdot t)$$

$$N := 128$$

$$i := 0 .. N - 1$$

$$Y_i := f(i)$$

$$W := wave(Y)$$

$$coeffs(level) := submatrix\left(W, 2^{level}, 2^{level+1} - 1, 0, 0\right)$$

$$Nlevels := \frac{\ln(N)}{\ln(2)} - 1$$

```
Nlevels := 6

k := 1 .. Nlevels

C_{i,k} := coeffs(k)
```

$$\text{floor}\left(\frac{i \cdot 2^{k}}{N}\right)$$

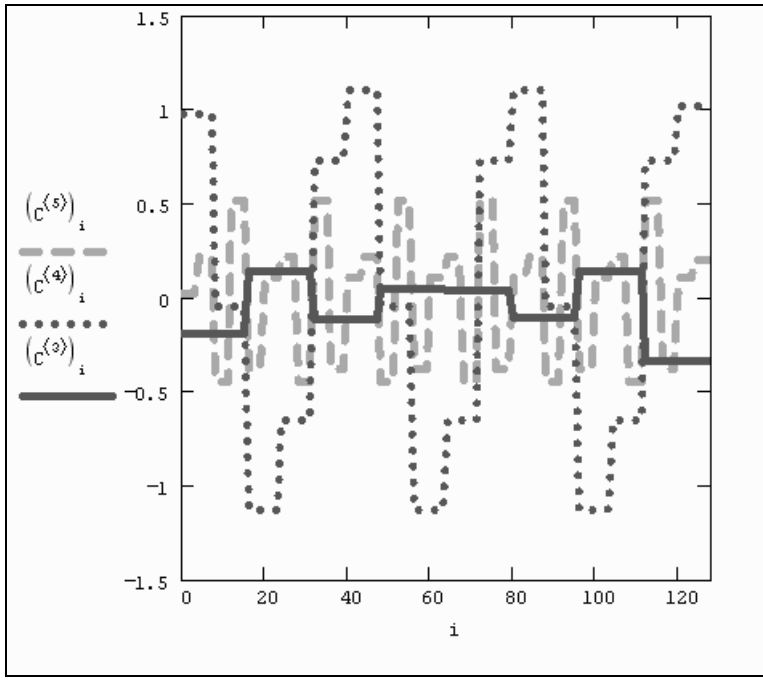



**Рис. 4.16.** Вейвлет-спектр Добеши модельного сигнала (продолжение листинга 4.18)

## Программирование других вейвлет-преобразований

Наряду со встроенной функцией `wave`, профессиональные версии Mathcad 11—13 снабжены пакетом расширения для осуществления вейвлет-анализа. Пакет расширения содержит большое число дополнительных встроенных функций, имеющих отношение к вейвлет-преобразованиям. Обзор пакетов расширения выходит за рамки данной книги, поэтому ограничимся простым упоминанием об этой возможности.

### *Примечание*

Дополнительную информацию об использовании данных встроенных функций можно найти в соответствующей электронной книге, которую можно открыть при помощи меню **Help / E-Books / Wavelet extension pack** (Справка / Электронные книги / Вейвлет-анализ данных).

Помимо встроенной функции вейвлет-спектра Добеши и возможностей пакета расширения, допускается непосредственное программирование алгоритмов пользователя для расчета вейвлет-спектров. Оно сводится к аккуратному численному расчету соответствующих семейств интегралов. Один из примеров такой программы приведен в листинге 4.19. Анализу подвергается функция, составленная из суммы двух синусов, а график двухпараметрического спектра `C(a,b)` выведен на рис. 4.17 в виде привычных для вейвлет-анализа линий уровня на плоскости `(a,b)`.

### *Примечание*

Программа листинга очень проста, но исключительно далека от оптимальной в смысле быстродействия. Каждый интеграл вычисляется независимо, без использования методов ускорения, типа применяемых в алгоритме БПФ. Однако простые приемы программирования вполне доступно раскрывают математический смысл вейвлет-преобразования.

**Листинг 4.19. Вычисление вейвлет-спектра на основе "мексиканской шляпы"**

$$f(t) := 1.0 \cdot \sin\left(2\pi \cdot 0.02 \cdot t\right) + 0.3 \cdot \sin\left(2\pi \cdot 0.1 \cdot t\right)$$

$$MHAT(t) := \frac{d^2}{dt^2} \exp\left(\frac{-t^2}{2}\right)$$

$$N := 256$$

$$W(a,b) := \int_{-N}^{N} f(t) \cdot MHAT\left(\frac{t-b}{a}\right) dt$$

$$b := 0, 1 .. \frac{N}{10}$$

$$i := 0 .. 10$$

$$a_i := \frac{(i+15)^4}{2 \times 10^4}$$

$$C_{i,b} := W\left(a_i, 2 \cdot b - \frac{N}{10}\right)$$

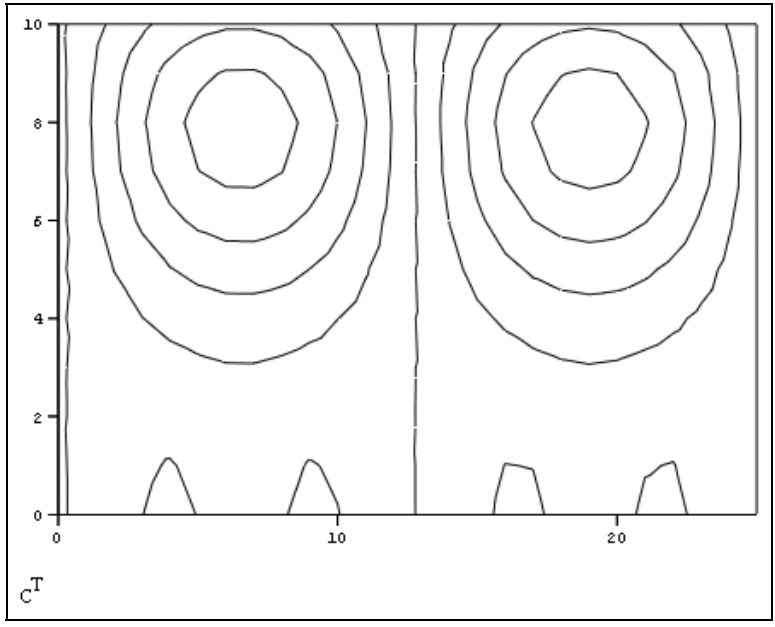



**Рис. 4.17.** Вейвлет-спектр модельного сигнала на основе "мексиканской шляпы" (продолжение листинга 4.19)

# ЧАСТЬ II

## Алгебраические уравнения

# Глава 5

# Нелинейные алгебраические уравнения

Огромное количество задач вычислительной математики связано с решением нелинейных алгебраических уравнений, а также систем таких уравнений. При этом необходимость решения нелинейных уравнений возникает зачастую на промежуточных шагах, при реализации фрагментов более сложных алгоритмов (к примеру, при расчетах дифференциальных уравнений при помощи разностных схем и т. п.).

Постановка задач выглядит следующим образом. Пусть имеется либо одно алгебраическое уравнение с неизвестным $x$:

$$f(x)=0,$$

(где $f(x)$ — некоторая функция), либо система из $N$ алгебраических уравнений:

$$\begin{cases} f_1(x_1,...,x_M)=0 \\ ... \\ f_N(x_1,...,x_M)=0 \end{cases}$$

Требуется найти корни, т. е. все значения $x$ (или, в случае системы все $N$-покомпонентные векторы $x$), которые переводят уравнение (или, соответственно, систему уравнений) в верное равенство (равенства).

> ***Примечание***
>
> Решение систем линейных уравнений, у которых все функции имеют вид $f_i(x)=a_{i1}\cdot x_1+a_{i2}\cdot x_2+...+a_{iN}\cdot x_N$, представляет собой отдельную задачу вычислительной линейной алгебры. Она рассматривается в *главе 8*.

# 5.1. Аналитическое решение уравнений

Относительно небольшое количество задач отыскания корней алгебраических уравнений можно решить аналитически, а на практике почти всегда приходится искать решение при помощи численных методов. Тем не менее мы начнем знакомство с принципами решения алгебраических уравнений в Mathcad именно с описания использования символьного процессора, что позволит, с одной стороны, освоить соответствующий инструментарий Mathcad, а с другой — лучше понять специфику исследуемых задач.

## 5.1.1. Вычислительный блок *Given* / *Find*

Рассмотрим решение системы N нелинейных уравнений с M неизвестными

$$\begin{cases} f_1(x_1, \ldots, x_M) = b_1, \\ \ldots \\ f_N(x_1, \ldots, x_M) = b_N \end{cases} \qquad (5.1)$$

Здесь $f_1(x_1,\ldots,x_M)=b_1,\ldots, f_N(x_1,\ldots,x_M)=b_N$ — некоторые скалярные выражения, зависящие от скалярных переменных $x_1, x_2, \ldots, x_M$ и, возможно, от еще каких-либо переменных. Уравнений может быть как больше, так и меньше числа переменных. Заметим, что систему (5.1) можно формально переписать в виде

$$f(x)=b, \qquad (5.2)$$

где x — вектор, составленный из переменных $x_1, x_2, \ldots, x_M$; b — вектор, составленный из правых частей уравнений, а f(x) — соответствующая векторная функция их левых частей.

Для решения систем в Mathcad применяется специальный *вычислительный блок* Given/Find (Дано/Найти), состоящий из трех частей, идущих последовательно друг за другом:

- Given — ключевое слово;
- система, записанная логическими операторами в виде равенств и, возможно, неравенств;
- Find($x_1,\ldots,x_M$) — встроенная функция для решения системы уравнений относительно переменных $x_1,\ldots,x_M$.

Вставлять логические операторы следует, пользуясь панелью инструментов **Boolean** (Булевы операторы). Если вы предпочитаете ввод с клавиатуры, помните, что логический знак равенства вводится сочетанием клавиш <Ctrl>+<=>. Значение функции Find представляет собой матрицу, составленную из всевозможных решений по каждой переменной, причем количество ее строк в точности равно числу аргументов Find. Структура матрицы решения

станет сразу вам понятной, как только вы бросите взгляд на примеры, приведенные ниже в данном разделе.

### Примечание

При решении уравнений в векторной форме в вычислительном блоке не рекомендуется, а, начиная с версии Mathcad 12, просто запрещено использование одних элементов вектора в качестве неизвестных, а других — в качестве параметров задачи. Соответствующие примеры, иллюстрирующие данное ограничение, вы найдете на компакт-диске.

## 5.1.2. Одно уравнение

Поясним сказанное на примере решения одного (кубического) уравнения с одним неизвестным x (рис. 5.1):

$$3x^3+2x^2-7x=0. \qquad (5.3)$$

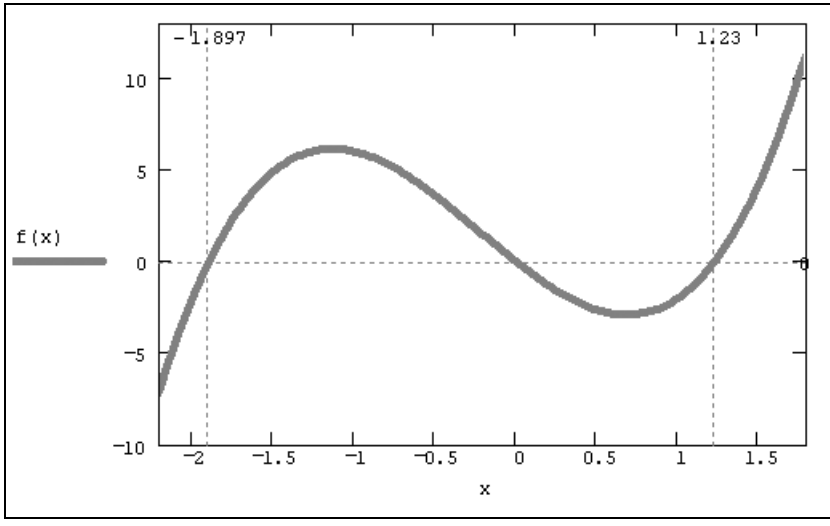



**Рис. 5.1.** График функции $f(x)=3x^3+2x^2-7x$

**Листинг 5.1. Аналитическое решение кубического уравнения**

```
Given

3x^3 + 2x^2 - 7x = 0
```

$$\text{Find}(x) \rightarrow \left( 0 \quad \frac{-1}{3} + \frac{1}{3} \cdot \sqrt{22} \quad \frac{-1}{3} - \frac{1}{3} \cdot \sqrt{22} \right) = (0 \quad 1.23 \quad -1.897)$$

В листинге 5.1 вы видите все три последовательные строки вычислительного блока. Первая строка представляет собой обязательное ключевое слово `Given`, следующая строка является, собственно, записью уравнения (5.3), а в последней строке листинга включается в работу встроенная функция `Find`. Обратите внимание, что после имени функции `Find` находится оператор символьного вывода, справа от которого (по истечении необходимого времени работы символьного процессора) возникает аналитический результат решения уравнения. Существенно, что он является точным решением (записанным в данном случае в трансцендентном виде), а получить числовые значения корней можно, поставив после него символ численного равенства (как это сделано в последней строке листинга 5.1).

Как видно из листинга 5.1, уравнение имеет три различных корня, которые представляются справа от функции соответствующим трехкомпонентным вектором. Таким образом, решение предлагается пользователю в форме матрицы размера `1×3` (одна неизвестная переменная имеет три значения, каждое из которых обращает уравнение в тождество).

Пример, использованный в листинге 5.1, включает уравнение, записанное в традиционной форме равенства. Приведем решение того же самого уравнения, если оно представлено в несколько другой форме, подчеркивающей специфику задачи нахождения корней функции (листинг 5.2). Основное отличие листинга 5.2 от предыдущего связано с другой формой записи исследуемого уравнения через функцию пользователя `f(x)`. Иными словами, подчеркивается специфика поставленной задачи отыскания нулевых значений некоторой функции.

Важно заметить, что точное решение уравнения (непосредственно после оператора символьного вывода результата работы функции `Find`) обращает `f(x)` в тождественный ноль, а пересчитанные числовые значения корней (после знака обычного равенства) обеспечивают лишь ее приближенное равенство нулю (разумеется, это связано с ошибками округления).

> ***Внимание!***
>
> Не забывайте о том, что вводить знаки равенства в уравнение в пределах вычислительного блока `Given/Find` следует при помощи панели **Boolean** (Булевы операторы).

**Листинг 5.2. Аналитический поиск нулей функции `f(x)`**

$$f(x) := 3x^3 + 2x^2 - 7x$$

Given

$$f(x) = 0$$

$$\text{Find}(x) \rightarrow \left( 0 \quad \frac{-1}{3} + \frac{1}{3}\cdot\sqrt{22} \quad \frac{-1}{3} - \frac{1}{3}\cdot\sqrt{22} \right) = (0 \quad 1.23 \quad -1.897)$$

$$f(0) = 0$$

$$f\left(\frac{-1}{3} + \frac{1}{3}\cdot\sqrt{22}\right) = 0 \qquad f\left(\frac{-1}{3} - \frac{1}{3}\cdot\sqrt{22}\right) = 5.329 \times 10^{-15}$$

$$f(1.23) = -1.599 \times 10^{-3} \qquad f(-1.897) = -3.466 \times 10^{-3}$$

В заключение разговора о символьном решении уравнений с одним неизвестным приведем еще два показательных примера, связанных с нахождением нулей функции нескольких аргументов (т. е. зависящих, помимо, собственно, неизвестного, еще и от дополнительных параметров). Листинг 5.3 демонстрирует, как выглядит решение уравнения, включающего четыре различные переменные, по некоторым из этих переменных. Обратите внимание на последний из трех приведенных в листинге 5.3 примеров, иллюстрирующий результат решения уравнения относительно сразу всех входящих в него параметров.

Помните, что для корректной работы символьного процессора вовсе не обязательно задавать конкретные значения переменных, входящих в уравнение, а если такие значения (для некоторых, либо всех, параметров) определены, то это учитывается при выводе результата (что иллюстрируется листингом 5.4).

**Листинг 5.3. Символьное решение уравнения относительно разных переменных**

Given $\qquad a x^3 + b x^2 - c x = 0$

$$\text{Find}(x) \rightarrow \left[ 0 \quad \frac{1}{2\cdot a}\cdot\left[-b + \left(b^2 + 4\cdot a\cdot c\right)^{\frac{1}{2}}\right] \quad \frac{1}{2\cdot a}\cdot\left[-b - \left(b^2 + 4\cdot a\cdot c\right)^{\frac{1}{2}}\right]\right]$$

Given $\qquad a x^3 + b x^2 - c x = 0$

$$\text{Find}(a) \rightarrow \frac{-1}{x^2} \cdot (b \cdot x - c)$$

Given $\quad a x^3 + b x^2 - c x = 0$

$$\text{Find}(a, b, c, x) \rightarrow \begin{pmatrix} a & a \\ b & b \\ c & a \cdot x^2 + b \cdot x \\ 0 & x \end{pmatrix}$$

**Листинг 5.4. Символьное решение уравнения, зависящего от параметров, в случае предварительного задания их числовых значений**

$a := 3 \qquad b := 2$

Given

$a x^3 + b x^2 - c x = 0$

$$\text{Find}(x) \rightarrow \begin{bmatrix} 0 & \frac{-1}{3} + \frac{1}{3} \cdot (1 + 3 \cdot c)^{\frac{1}{2}} & \frac{-1}{3} - \frac{1}{3} \cdot (1 + 3 \cdot c)^{\frac{1}{2}} \end{bmatrix}$$

Если решить уравнение аналитически не удается, то результатом применения оператора символьного вывода после функции `Find` будет либо тривиальное выражение типа `Find(x)→x` (как в листинге 5.5), либо сообщение об ошибке "No symbolic result was found" (Ни один символьный результат не найден). Следует помнить, что символьный процессор Mathcad "умеет" находить не только действительные, но и комплексные корни уравнений. В качестве примера приведем листинг 5.6 с решением кубического уравнения, имеющего три очевидных корня — одного действительного (равного нулю) и двух чисто мнимых (±i, где `i` — мнимая единица).

**Листинг 5.5. Решить уравнение аналитически не удается**

Given

$\ln(\sin(x)) = 0$

$\text{Find}(x) \rightarrow x$

**Листинг 5.6. Символьное решение уравнения, имеющего и действительные, и мнимые корни**

Given

$x \cdot \left(x^2 + 1\right) = 0$

Find(x) → ( 0  i  −i )

## 5.1.3. Системы уравнений

Символьное решение системы алгебраических уравнений отличается от описанного случая одного уравнения только количеством соотношений, задаваемых после ключевого слова `Given`. Соответственно, число неизвестных также может быть любым, причем необязательно равным числу уравнений. Если система уравнений имеет не обособленные решения, а целые семейства решений, то соответствующие результаты выдаются символьным процессором Mathcad в виде выражений, формально зависящих от одной из переменных как от параметра (пример решения одного уравнения с тремя неизвестными был приведен ранее в последних двух строках листинга 5.3).

Решение системы двух нелинейных уравнений иллюстрирует листинг 5.7. Нахождение символьным процессором его обоих корней визуализируется на графике, приведенном на рис. 5.2. На нем каждое из уравнений показывается в виде зависимости `y(x)`: первое — сплошной кривой, а второе — пунктиром. Поскольку первое уравнение является квадратичным, то оно определяет на плоскости `XY` параболу, и поскольку второе уравнение линейное, то оно соответствует на графике прямой линии. Очевидно, что две точки пересечения кривых соответствуют одновременному выполнению обоих уравнений, т. е. их координаты равны искомым действительным корням системы.

**Листинг 5.7. Символьное решение системы двух уравнений**

Given

$-x^2 + y + 1 = 0$

$x = 2y + 1$

$\text{Find}(x, y) \rightarrow \begin{pmatrix} 1 & \frac{-1}{2} \\ 0 & \frac{-3}{4} \end{pmatrix}$

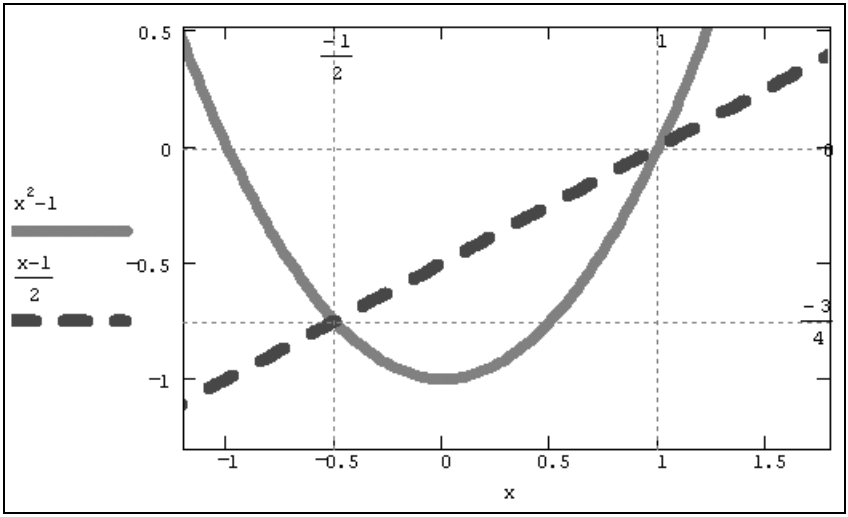



**Рис. 5.2.** Графическая интерпретация решения системы двух уравнений (см. листинг 5.7)

## 5.1.4. Решение уравнений при помощи меню

Аналитическое решение алгебраического уравнения можно получить и при помощи меню. Данный способ удобно применять, когда уравнение записано в форме математического выражения (зависящего, возможно, от различных переменных), и требуется вычислить аналитически значение одной переменной, при котором выражение обращается в ноль. Для этого:

1. Введите выражение.
2. Выделите переменную, относительно которой будет решаться уравнение, приравнивающее выражение нулю.
3. Выберите в меню **Symbolics** (Символика) пункт **Variable / Solve** (Переменная / Решить) (рис. 5.3).

В итоге после выражения появится строка с результатом решения уравнения (т. е. значениями переменной, которые обращают исходное выражение в ноль). Следует иметь в виду, что решение уравнения производится именно по той переменной, которую пользователь выделяет перед вводом команды меню. Разумеется, если выделять различные переменные, то в итоге будут выдаваться совершенно разные решения.

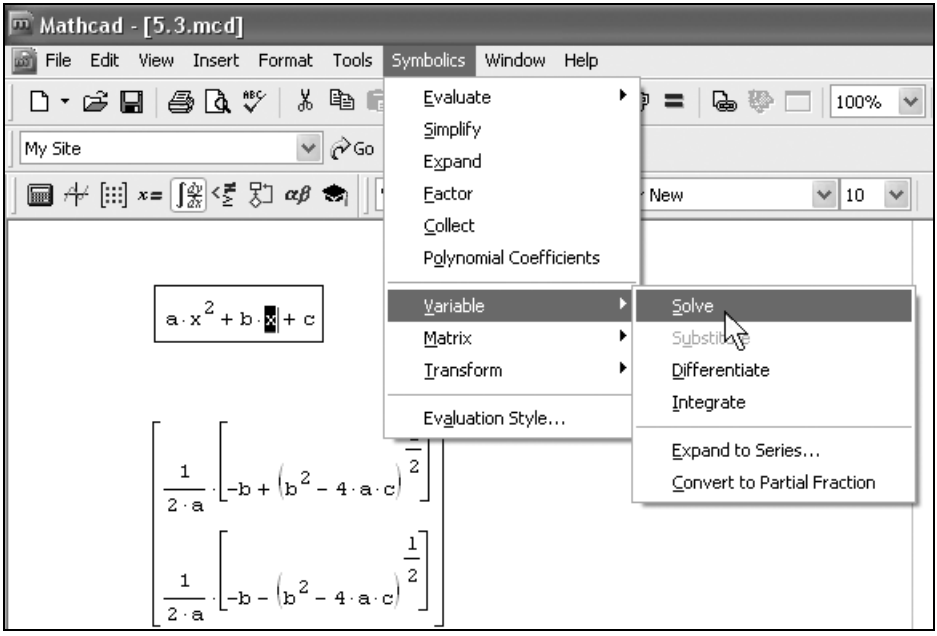



**Рис. 5.3.** Символьное решение уравнения

***Внимание!***

Помните о том, что символьные вычисления, проводимые при помощи меню, не являются "живыми", т. е. не будут меняться впоследствии, если вы будете редактировать исходное выражение.

# 5.2. Численное решение уравнений

Как правило, отыскание корней алгебраического уравнения (или системы уравнений) численными методами связано с двумя задачами:

- локализация корней, т. е. определение их существования в принципе, а также исследование их количества и примерного расположения;
- собственно отыскание корней с заданной погрешностью (в Mathcad ее роль играет системная константа `TOL`).

Последнее означает, что надо найти значения $x_0$, при которых `f(x0)` отличается от нуля не более чем на `TOL`. Почти все встроенные функции системы Mathcad, предназначенные для решения нелинейных алгебраических уравнений (в том числе функция `Find` применительно к численному процессору)

нацелены на решение второй задачи, т. е. предполагают, что корни уже приблизительно локализованы. Иными словами, перед тем, как непосредственно приступать к нахождению корней уравнения, необходимо (хотя бы приблизительно) представлять себе, где они находятся. Имея в виду данную оговорку, мы перейдем к рассмотрению второй из задач, а проблему предварительной локализации корней рассмотрим в конце данного раздела.

## 5.2.1. Системы уравнений: функция *Find*

Рассмотрим решение системы N нелинейных уравнений с M неизвестными

$$\begin{cases} f_1(x_1, \ldots, x_M) = 0, \\ \ldots \\ f_N(x_1, \ldots, x_M) = 0 \end{cases} \tag{5.4}$$

Здесь $f_1(x_1, \ldots, x_M), \ldots, f_N(x_1, \ldots, x_M)$ — некоторые скалярные функции от скалярных переменных $x_1, x_2, \ldots, x_M$ и, возможно, от еще каких-либо переменных. Уравнений может быть как больше, так и меньше числа переменных. Заметим, что систему (5.4) можно формально переписать в виде

$$f(x) = 0, \tag{5.5}$$

где x — вектор, составленный из переменных $x_1, x_2, \ldots, x_M$, а f(x) — соответствующая векторная функция.

### Вычислительный блок *Given / Find*

Для численного решения систем уравнений применяется тот же самый *вычислительный блок*, что и для символьных вычислений (*см. разд. 5.1.1*). Повторимся, что он состоит из ключевого слова Given, самой системы уравнений, записанной при помощи логических операторов панели **Boolean** (Булевы операторы), а также встроенной функции Find. Find($x_1, \ldots, x_M$) — встроенная функция для решения системы алгебраических уравнений и неравенств относительно переменных $x_1, \ldots, x_M$. Значение функции Find представляет собой вектор, составленный из решений по каждой переменной.

#### *Примечание 1*

Встроенная функция Find использует в качестве численного алгоритма один из градиентных методов (*см. разд. 5.3*). Этот факт налагает некоторые ограничения на уравнения системы, которые должны быть достаточно гладкими функциями своих аргументов.

***Примечание 2***

При численных расчетах может быть найден только один из корней уравнения, в отличие от символьных вычислений, которые позволяют определить все имеющиеся корни.

Применение численного нахождения корней отличается от символьного двумя обстоятельствами:

- вместо оператора символьного вывода после функции `Find` следует использовать оператор численного вывода (знак равенства);
- перед вычислительным блоком `Given/Find` должны быть заданы *начальные значения* (guess value) для всех неизвестных, т. е. всем переменным $x_1, \ldots, x_M$, относительно которых решается уравнение, следует предварительно присвоить некоторые численные значения, с которых и будет начинаться поиск корня. Таким образом, присвоение начального значения требует априорной информации о примерном местонахождении корня и связано с проблемой локализации корней, упомянутой в начале *разд. 5.2*.

***Примечание***

Помните также о том, что, в отличие от символьного, для численного процессора необходимо задать числовые значения всех параметров, входящих в уравнение.

## Одно уравнение с одним неизвестным

Рассмотрим в качестве примера (листинг 5.8) одно уравнение с одним неизвестным, которое уже решалось нами аналитически (см. листинги 5.1 и 5.2). Уравнение имеет три корня, как видно из графика, приведенного на рис. 5.4. Обратите внимание, что перед ключевым словом `Given` переменной `x` присваивается некоторое значение `x=1`. В остальном применение функции `Find` для решения уравнения не отличается от символьных расчетов.

**Листинг 5.8. Численное решение кубического уравнения с начальным значением x=1**

```
x := 1

Given

3x^3 + 2x^2 - 7x = 0

Find(x) = 1.23
```

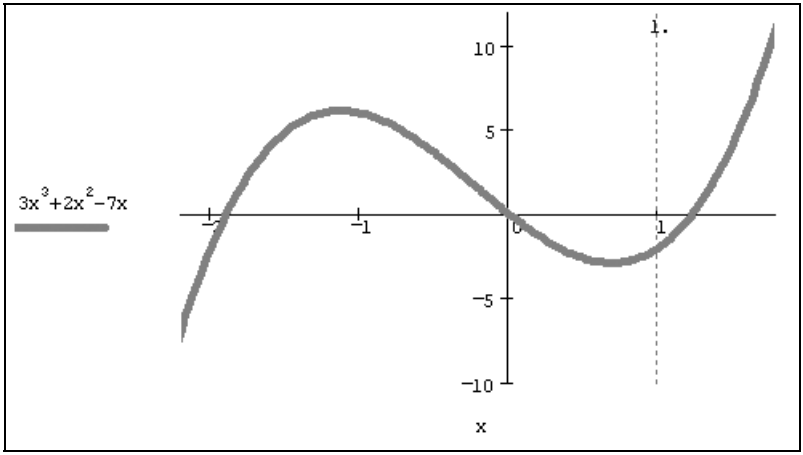



**Рис. 5.4.** Графическая иллюстрация решения кубического уравнения

Как уже отмечалось выше, результатом численного решения алгебраического уравнения является один его корень. Для того чтобы отыскать остальные корни, необходимо повторно решить уравнение, взяв для переменной `x` другие начальные значения. Например, если присвоить ей в начале листинга значение `x=-1`, то численным процессором будет выдан в качестве результата другой корень `x=0` (листинг 5.9). Такая работа программы Mathcad связана с особенностями применяемых численных алгоритмов (*см. разд. 5.3*). Для численного определения всех корней уравнения следует применять специальные приемы, например, сканирование по неизвестным (*см. разд. 5.2.4*).

**Листинг 5.9. Численное решение кубического уравнения с начальным значением `x=-1` сходится к другому корню `x=0`**

```
x := -1
Given
3x^3 + 2x^2 - 7x = 0
Find(x) = 0
```

## Системы уравнений

Приведем еще один пример численного решения алгебраических уравнений, обратившись на этот раз к системе двух уравнений, которая также уже

исследовалась нами при помощи символьного процессора. Система имеет два решения, показанные графически на рис. 5.2 и найденные аналитически в листинге 5.7 (*см. разд 5.1.3*). Листинг 5.10, демонстрирующий численное решение рассматриваемой системы, начинается с присвоения неизвестным начальных значений `x=10`, `y=10`. После этого следует ключевое слово `Given` и два логических оператора, выражающих рассматриваемую систему уравнений. В результате (последняя строка листинга) Mathcad находит один из корней `x=1`, `y=0`, причем первый элемент вектора решения есть первый аргумент функции `Find`, а второй элемент — ее второй аргумент. Поскольку решение производится численным методом, оно выдается с некоторой погрешностью, не превышающей встроенной константы `CTOL`. Если задать в первой строке листинга другие начальные значения, расположенные ближе к другому корню, например, `x=0`, `y=0`, то найден в итоге будет другой корень `x=-0.5`, `y=-0.75`.

### *Примечание 1*

На самом деле в вычислительном блоке используются обе системные константы Mathcad, связанные с заданием погрешности: `TOL` и `CTOL`. Константа `CTOL` ограничивает невязку, т. е. задает точность выполнения уравнений, введенных после ключевого слова `Given`. Например, если `CTOL=0.001`, то уравнение `x=10` будет считаться выполненным и при `x=10.001`, и при `x=9.999`. Другая константа `TOL` определяет условие прекращения итераций численным алгоритмом (*см. разд. 5.4*). Значение `CTOL` может быть задано пользователем так же, как и `TOL`, например, `CTOL:=0.01`. По умолчанию принято, что `CTOL=TOL=0.001`, но вы по желанию можете переопределить их.

### *Примечание 2*

Часто бывает очень полезным проверить точность решения уравнений "вручную", подставив найденные вычислительным процессором корни в исходные уравнения и оценив значение их невязок.

**Листинг 5.10. Численное решение системы алгебраических уравнений**

$x := 10 \qquad y := 10$

Given

$-x^2 + y + 1 = 0$

$x = 2y + 1$

$\text{Find}(x, y) = \begin{pmatrix} 1 \\ -1.133 \times 10^{-5} \end{pmatrix}$

Если предпринять попытку решить несовместную систему, в частности, в рассматриваемом примере добавить еще одно уравнение, то Mathcad выдаст сообщение об ошибке, гласящее, что ни одного решения не найдено, и предложение попробовать поменять начальные значения или значение погрешности.

Особую осторожность следует соблюдать при решении систем с числом неизвестных бóльшим, чем число уравнений. Например, можно удалить одно из двух уравнений из рассмотренной нами задачи, попытавшись решить оставшееся единственное уравнение с двумя неизвестными `x` и `y` (листинг 5.11). В такой постановке задача имеет бесконечное множество корней: для любого `x` и, соответственно, `y=x^2-1` условие, определяющее единственное уравнение, выполнено. Однако даже если корней бесконечно много, численный метод будет производить расчеты только до тех пор, пока логические выражения в вычислительном блоке не будут выполнены (конечно, в пределах погрешности). После этого итерации будут остановлены и выдано решение. В результате будет найдена всего одна пара значений `(x,y)`, обнаруженная первой, как это показано в последней строке листинга 5.11.

***Примечание***

Для того чтобы найти все решения рассматриваемой задачи, можно обратиться к возможностям символьного процессора Mathcad. Достаточно в последней строке листинга заменить знак равенства на оператор символьного вывода, и в качестве ответа будет выдано семейство решений `x` и `x^2-1`.

**Листинг 5.11. Численное решение уравнения, имеющего бесконечное множество корней, приводит к одному из них**

```
x := 10          y := 10

Given

-x^2 + y + 1 = 0

Find(x, y) = ( 3.377  )
             ( 10.402 )
```

## Системы уравнений и неравенств

Пока мы рассматривали примеры систем уравнений, число которых было таким же, как и число неизвестных, что встречается наиболее часто. Но число уравнений и неизвестных может и не совпадать. Более того, в вычислительный блок можно добавить дополнительные условия в виде неравенств.

Например, введение ограничения на поиск только отрицательных значений x в рассмотренный выше листинг 5.10 приведет к нахождению другого решения, как это показано в листинге 5.12.

Обратите внимание, что, несмотря на те же начальные значения, что и в листинге 5.10, в листинге 5.12 мы получили другой корень системы уравнений. Это произошло именно благодаря введению дополнительного неравенства, которое определено в блоке Given в предпоследней строке листинга 5.12.

**Листинг 5.12. Численное решение системы алгебраических уравнений и неравенств**

$x := 10 \qquad y := 10$

Given

$-x^2 + y + 1 = 0$

$x = 2y + 1$

$x < 0$

$\text{Find}(x, y) = \begin{pmatrix} -0.5 \\ -0.75 \end{pmatrix}$

## 5.2.2. Уравнение с одним неизвестным: функция *root*

Для решения уравнения с одним неизвестным в Mathcad, помимо вычислительного блока Given/Find, предусмотрена встроенная функция root, которая, в зависимости от типа задачи, может включать либо два, либо четыре аргумента и, соответственно, использует разные алгоритмы поиска корней.

- root(f(x),x);
- root(f(x),x,a,b):
  - f(x) — скалярная функция, определяющая уравнение f(x)=0;
  - x — имя скалярной переменной, относительно которой решается уравнение;
  - a,b — границы интервала, внутри которого происходит поиск корня.

Первый тип функции root, аналогично встроенной функции Find, требует дополнительного задания начального значения переменной x, для чего нужно

просто перед применением функции root присвоить x некоторое число. Таким образом, присвоение начального значения требует априорной информации о примерной локализации корня, т. к. поиск корня будет производиться вблизи этого числа. Пример работы функции root объясняется листингом 5.13.

**Листинг 5.13. Два варианта решения уравнения методом секущих**

$$\mathrm{root}\left(e^{x} - 1, x, -1, 1\right) = 0$$

$$x := 1$$

$$\mathrm{root}\left(e^{x} - 1, x\right) = 2.21 \times 10^{-5}$$

Как вы можете убедиться (первая строка листинга 5.13), для решения уравнения при помощи функции root(f(x),x,a,b) не требуется задавать начального приближения, а достаточно указать интервал [a,b]. Поиск корня будет осуществлен в промежутке между a и b альтернативным численным методом (Риддера или Брента). Когда root имеет четыре аргумента, следует помнить о двух ее особенностях. Во-первых, внутри интервала не должно находиться более одного корня, иначе будет найден один из них, заранее неизвестно, какой именно. Во-вторых, значения f(a) и f(b) должны иметь разные знаки, иначе будет выдано сообщение об ошибке.

В чем же отличие встроенной функции Find от функции root? Оно состоит в том, что для решения одних и тех же задач используются различные численные алгоритмы (градиентные и метод секущих соответственно). В примерах уравнений с одним неизвестным, которые мы рассматривали до сего момента, выбор метода не влиял на окончательный результат, поскольку фигурировавшие в них функции были "хорошими", т. е. достаточно гладкими для поиска корня одним из градиентных методов, требующих, как известно, вычисления производных. Между тем бывают ситуации, когда применение того или иного метода имеет решающее значение.

Приведем пример простой функции f(x), корни которой удается отыскать только при помощи функции root (листинг 5.14). Она определена в первой строке этого листинга, а ее корень вычислен во второй строке. Из графика, представленного на рис. 5.5, видно, что f(x) имеет особенность в окрестности своего корня, являясь в ней разрывной. В завершающей части листинга 5.14 предпринимается попытка отыскать нулевое значение f(x) посредством вычислительного блока Given/Find, которая оказывается неудачной.

**Листинг 5.14. Пример уравнения, которое удается решить только методом секущих**

```
f(x) := | (Φ(x - 1.01) - 0.5)   if x ≠ 1.01
        | 0   otherwise

root(f(x), x, 0, 2) = 1.01

x := 1

Given

f(x) = 0

Find(x) = ■
```

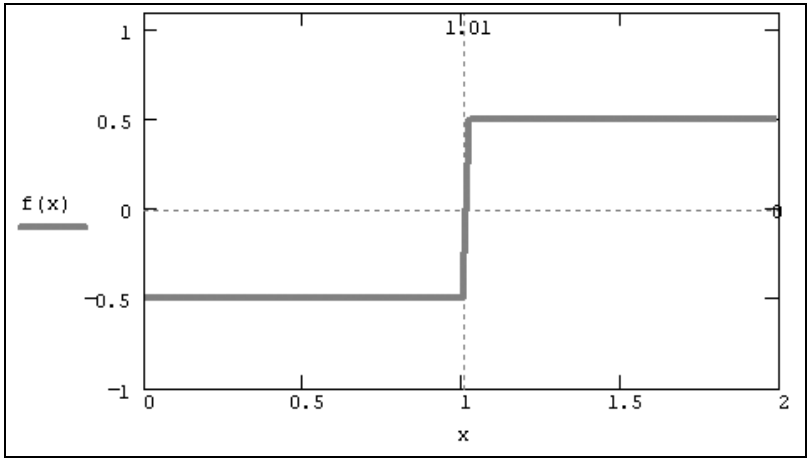



**Рис. 5.5.** Модельная функция `f(x)`
(продолжение листинга 5.14)

Остается добавить, что `f(x)` может быть функцией не только `x`, а любого количества аргументов. Именно поэтому в самой функции `root` необходимо определить, относительно какого из аргументов следует решить уравнение. Эта возможность проиллюстрирована листингом 5.15 на примере функции двух переменных $f(x,y)=x^2-y^2+1$. В нем сначала решается уравнение $f(x,0)=0$ относительно переменной `x`, а потом — другое уравнение $f(0,y)=0$ относительно переменной `y`, причем, благодаря удачному подбору начальных значений, вычисляются все корни данного квадратичного уравнения.

Таким образом, в обоих случаях один из аргументов функции `f(x)` воспринимается как неизвестное, а другой — как параметр. Не забывайте при чис-

ленном решении уравнений относительно одной из переменных предварительно определить значения остальных переменных. Иначе попытка вычислить уравнения приведет к появлению ошибки "This variable or function is not defined above", в данном случае говорящей о том, что другая переменная ранее не определена.

***Примечание***

Для того чтобы отыскать зависимость корней уравнения, вычисленных по одной переменной, от других переменных, разработаны специальные эффективные алгоритмы. Об одной из возможностей читайте в *разд. 5.3.3*.

**Листинг 5.15. Поиск корней уравнения, зависящего от двух переменных**

```
f(x, y) := x^2 - y^2 + 1
x := 1        root(f(x, 0), x) = -i
x := -1       root(f(x, 0), x) = i
y := 0        root(f(0, y), y) = 1
y := -100     root(f(0, y), y) = -1
```

## 5.2.3. Корни полинома: функция *polyroots*

Если функция `f(x)` является *полиномом*, то все его корни можно определить, используя встроенную функцию:

❒ `polyroots(v)`

- где `v` — вектор, составленный из коэффициентов полинома.

Поскольку полином `N`-ой степени имеет ровно `N` корней (некоторые из них могут быть кратными), вектор `v` должен состоять из `N+1` элемента. В основе встроенной функции `polyroots` лежат специфические численные алгоритмы, а результатом ее действия является вектор, составленный из `N` корней рассматриваемого полинома. При этом нет надобности вводить какое-либо начальное приближение, как для функции `root`. Пример поиска корней полинома четвертой степени иллюстрируется листингом 5.16.

Коэффициенты рассматриваемого в примере полинома

$f(x)=(x-3)\cdot(x-1)^3=x^4-6x^3+12x^2-10x+3$

записаны в виде вектора в первой строке листинга. Первым в векторе должен идти свободный член полинома, вторым — коэффициент при $x^1$ и т. д. Соответственно, последним N+1 элементом вектора должен быть коэффициент при старшей степени $x^N$.

### Совет

Иногда исходный полином имеется не в развернутом виде, а, например, как произведение нескольких полиномов. В этом случае определить все его коэффициенты можно, выделив его и выбрав в меню **Symbolics** (Символика) пункт **Expand** (Разложить). В результате символьный процессор Mathcad сам преобразует полином в нужную форму; пользователю надо будет только корректно ввести ее в аргументы функции polyroots.

**Листинг 5.16. Вычисление корней полинома**

$$v := (3 \quad -10 \quad 12 \quad -6 \quad 1)^T$$

$$\text{polyroots}(v) = \begin{pmatrix} 0.992 \\ 1.004 + 7.177i \times 10^{-3} \\ 1.004 - 7.177i \times 10^{-3} \\ 3 \end{pmatrix}$$

Обратим внимание на результат применения функции polyroots, заметив, что численный метод вместо двух из трех действительных единичных корней (иными словами, кратного корня 1) выдает два мнимых числа. Однако малая мнимая часть этих корней находится в пределах погрешности, определяемой константой TOL, и не должна вводить пользователей в заблуждение. Просто нужно помнить, что корни полинома могут быть комплексными, и ошибка вычислений может сказываться как на действительной, так и на комплексной части искомого корня.

Для функции polyroots можно выбрать один из двух численных методов — метод полиномов Лаггера (он установлен по умолчанию) или метод парной матрицы.

Для смены метода:

1. Вызовите контекстное меню, щелкнув правой кнопкой мыши на слове polyroots.
2. В верхней части контекстного меню выберите либо пункт **LaGuerre** (Лаггера), либо **Companion Matrix** (Парная матрица).

3. Щелкните правой кнопкой мыши вне действия функции `polyroots` — если включен режим автоматических вычислений, будет произведен пересчет корней полинома в соответствии с вновь выбранным методом.

Для того чтобы оставить за Mathcad выбор метода решения, установите флажок **AutoSelect** (Автоматический выбор), выбрав одноименный пункт в том же самом контекстном меню.

## 5.2.4. Локализация корней

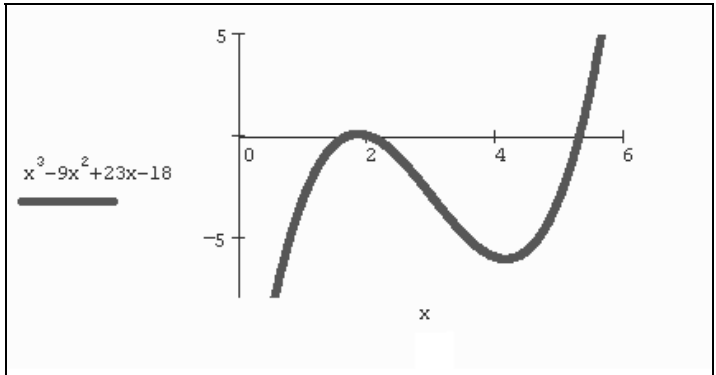
Чтобы решить задачу предварительной (грубой) локализации корней, в самых простых случаях можно использовать графическое представление `f(x)` (см. рис. 5.1, 5.2 и 5.4). Понятно, что в случае многомерных систем такой способ практически неприменим. Если требуется исследовать некоторую область определения переменных уравнения на наличие корней, задав их примерное положение, то обычно применяют весьма расточительный способ, называемый *сканированием*. Оно состоит в последовательном поиске корня, начиная из множества пробных точек, покрывающих расчетную область.

Обычно (вне Mathcad) сканирование организуют следующим образом. Область определения функции разбивается на элементарные области (в случае функции двух переменных чаще всего прямоугольные, в случае трех переменных — кубические и т. д.). Из центра каждой элементарной области запускается численный метод поиска корня, и в случае выхода итераций за ее пределы расчеты прерываются, а в противном случае происходит нахождение корня. Гораздо менее надежной (но зато более экономной) альтернативой является простое вычисление и сравнение между собой невязок системы уравнений в центральных точках элементарных областей. На тех участках области определения, где норма невязки невелика, вероятность локализации корня больше, и именно из локальных минимумов нормы невязки можно запускать градиентный метод для уточнения корня.

Пример организации упрощенного варианта сканирования по одной переменной приведен на рис. 5.6. График функции, корни которой подлежат определению, показан в его верхней части. Затем осуществляется решение уравнения при помощи функции `root` для нескольких последовательно расположенных узлов. Результат выдается в последней строке листинга в виде таблицы, из которой видно, что на рассматриваемом интервале уравнение имеет три корня.

***Примечание***

Конечно, гарантии, что все существующие корни будут найдены, особенно в многомерных случаях, чаще всего нет. Всегда существует вероятность "просмотреть" корень, расположенный между узлами сканирования.




$i := 0 .. 10$

$z_i := root(i^3 - 9i^2 + 23 \cdot i - 18, i)$

$z =$

| | 0 |
|---|---|
| 0 | 1.697 |
| 1 | 1.697 |
| 2 | 2 |
| 3 | 2 |
| 4 | 5.303 |
| 5 | 5.303 |
| 6 | 5.303 |
| 7 | 5.303 |
| 8 | 5.303 |
| 9 | 5.303 |
| 10 | 5.303 |

**Рис. 5.6.** Сканирование по переменной `x`

# ⓘ 5.3. О численных методах

Уделим теперь небольшое внимание численным алгоритмам, которые используются в работе встроенных функций `Find` и `root`, чтобы читатель имел

возможность применять их для решения алгебраических уравнений более осмысленно.

## 5.3.1. Метод секущих: функция *root*

Итерационный алгоритм, реализованный в функции `root`, который называется *методом секущих*, состоит в следующем (рис. 5.7):

1. Начальное приближение принимается за нулевое приближение к корню: `x0=x`.
2. Выбирается шаг `h=TOL·x` и определяется первое приближение к корню `x1=x0+h`. Если `x=0`, то принимается `h=TOL`.
3. Через эти две точки проводится секущая — прямая линия, которая пересекает ось `X` в некоторой точке `x2`. Эта точка принимается за второе приближение.
4. Новая секущая проводится через первую и вторую точки, тем самым определяя третье приближение, и т. д.
5. Если на каком-либо шаге оказывается, что уравнение выполнено, т. е. `|f(x)|<TOL`, то итерационный процесс прерывается и `x` выдается в качестве решения.

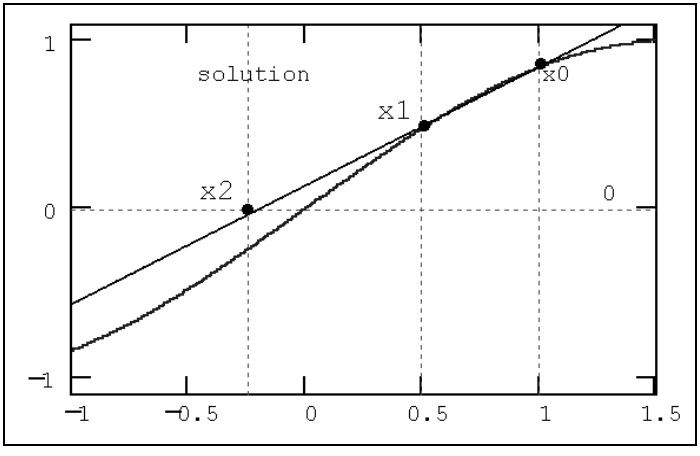



**Рис. 5.7.** Иллюстрация метода секущих

Результат, показанный на рис. 5.7, получен для погрешности вычислений, которой в целях иллюстративности предварительно присвоено значение

TOL=0.5. Поэтому для поиска корня с такой невысокой точностью оказалось достаточно одной итерации. В вычислениях, приведенных в листингах 5.13—5.15 (*см. разд. 5.2.2*), погрешность TOL=0.001 была установлена по умолчанию, и решение, выданное численным методом, лежало намного ближе к истинному положению корня. Иными словами, чем меньше константа TOL, тем ближе к нулю будет значение f(x) в найденном корне, но тем больше времени будет затрачено вычислительным процессором Mathcad на его поиск.

***Примечание***

Соответствующий пример можно найти на компакт-диске, а также в быстрых шпаргалках. Он расположен в разделе "Solving Equations" (Решение уравнений) и называется "Effects of TOL on Solving Equations" (Влияние константы TOL на решение уравнений).

Если уравнение неразрешимо, то при попытке найти его корень будет выдано сообщение об ошибке. Кроме того, к ошибке или выдаче неправильного корня может привести и попытка применить метод секущих в области локального максимума или минимума f(x). В этом случае секущая может иметь направление, близкое к горизонтальному, выводя точку следующего приближения далеко от предполагаемого положения корня. Для решения таких уравнений лучше применять другую встроенную функцию Minerr (*см. разд. 6.2*). Аналогичные проблемы могут возникнуть, если начальное приближение выбрано слишком далеко от настоящего решения или f(x) имеет особенности типа бесконечности.

## 5.3.2. Градиентные методы: функция *Find*

Если вы решаете "хорошие" уравнения, как все те, которые были приведены в предыдущих разделах, то, вообще говоря, можете никогда не задумываться, как именно Find ищет их корни. Однако даже в этом случае полезно представлять, что происходит "за кадром", т. е. какие действия совершаются в промежутке между введением необходимых условий после ключевого слова Given и получением результата после применения функции Find. Это важно хотя бы с позиций выбора начальных значений переменных перед вычислительным блоком. Рассмотрим в данном разделе некоторые особенности численных методов и возможности установки их различных параметров, которые предоставляет Find.

## Принцип действия градиентных алгоритмов

Во встроенной функции `Find` реализовано несколько градиентных численных алгоритмов, один из которых может выбрать либо программа Mathcad, либо сам пользователь. Покажем их основную идею на примере уравнения с одним неизвестным `f(x)=0` для функции $f(x)=x^2+5x+2$, график которой показан на рис. 5.8. Принцип градиентных методов состоит в последовательных приближениях к истинному решению уравнения, которые вычисляются с помощью производной от `f(x)`. Приведем наиболее простую форму алгоритма, называемого методом Ньютона:

1. За нулевую итерацию принимается введенное пользователем начальное значение `x0=x`.
2. В точке `x0` методом конечных разностей вычисляется производная `f'(x0)`.
3. Пользуясь разложением Тейлора, можно заменить `f(x)` в окрестности `x0` касательной — прямой линией $f(x) \approx f(x0)+f'(x0)\cdot(x-x0)$.
4. Определяется точка `x1`, в которой прямая пересекает ось `X` (рис. 5.8).
5. Если `f(x1)<TOL`, то итерации прерываются, и значение `x1` выдается в качестве решения. В противном случае `x1` принимается за новую итерацию, и цикл повторяется: строится касательная к `f(x)` в точке `x1`, определяется `x2` — точка ее пересечения с осью `X` и т. д.

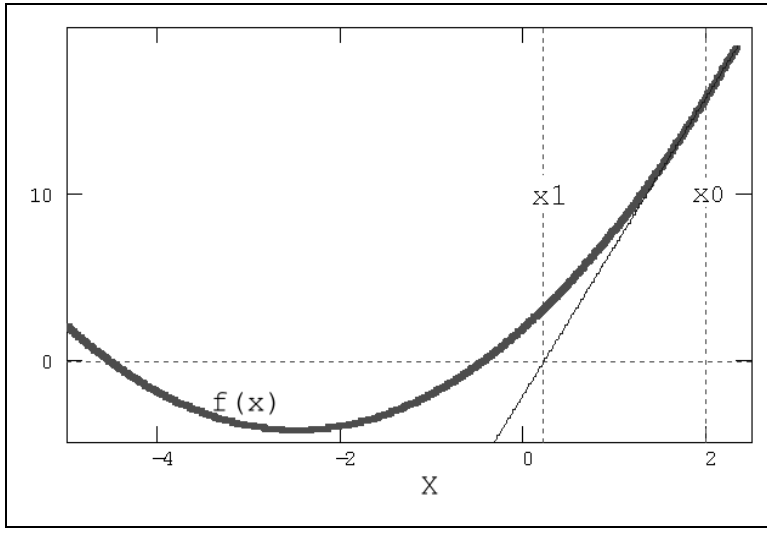



**Рис. 5.8.** Иллюстрация метода Ньютона

Модификация алгоритма Ньютона для решения системы нескольких уравнений заключается в линеаризации соответствующих функций многих переменных, т. е. аппроксимации их линейной зависимостью с помощью частных производных.

Например, для нулевой итерации в случае системы двух уравнений используются выражения типа:

$$f_1(x, y) \approx f_1(x0, y0) + \frac{\partial f_1(x0, y0)}{\partial x}(x - x0) + \frac{\partial f_1(x0, y0)}{\partial y}(y - y0);$$

$$f_2(x, y) \approx f_2(x0, y0) + \frac{\partial f_2(x0, y0)}{\partial x}(x - x0) + \frac{\partial f_2(x0, y0)}{\partial y}(y - y0).$$

Чтобы отыскать точку, соответствующую каждой новой итерации, требуется приравнять оба равенства нулю, т. е. решить на каждом шаге полученную систему линейных уравнений.

## Выбор градиентного алгоритма

Как уже отмечалось, Mathcad предлагает три различных варианта градиентных методов. Чтобы поменять численный метод:

1. Щелкните правой кнопкой мыши на названии функции `Find`.
2. Наведите указатель мыши на пункт **Nonlinear** (Нелинейный) в контекстном меню.
3. В появившемся подменю (рис. 5.9) выберите один из трех методов: **Conjugate Gradient** (Сопряженных градиентов), **Quasi-Newton** (Квазиньютоновский) или **Levenberg-Marquardt** (Левенберга—Маркарда).

Чтобы вернуть автоматический выбор типа численного метода, в контекстном меню надо выбрать пункт **AutoSelect** (Автоматический выбор). Если установлена опция автоматического выбора (о чем говорит флажок, установленный в пункте **AutoSelect**), то текущий тип численного метода можно узнать, вызвав то же самое подменю и посмотрев, который из них отмечен точкой. Два последних метода являются квазиньютоновскими, основная идея которых была рассмотрена выше. Первый из них, метод сопряженных градиентов, является двухшаговым — для поиска очередной итерации он использует как текущую, так и предыдущую итерации. Алгоритм Левенберга подробно описан в справочной системе Mathcad, а подробную информацию о методах Ньютона и сопряженных градиентов можно найти в большинстве книг по численным методам.

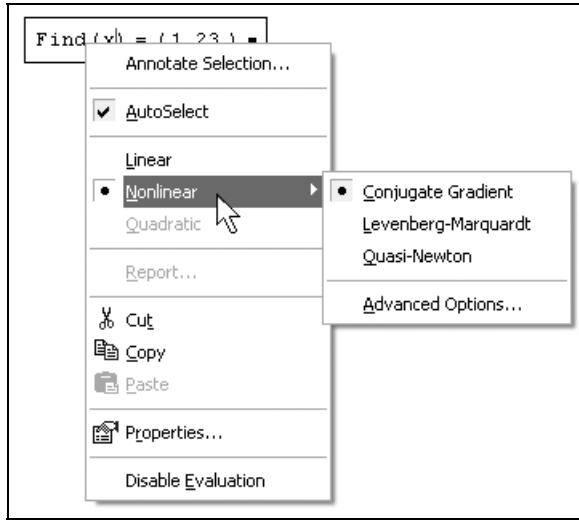



**Рис. 5.9.** Выбор численного алгоритма

## Параметры градиентных алгоритмов

Помимо выбора самих методов имеется возможность устанавливать их некоторые параметры. Для этого нужно вызвать с помощью того же контекстного меню диалоговое окно **Advanced Options** (Дополнительные параметры), выбрав в контекстном меню пункты **Nonlinear / Advanced options** (Нелинейный / Дополнительные параметры). В этом диалоговом окне (рис. 5.10) имеется пять групп переключателей по два в каждой.

В первой строке — **Derivative estimation** (Аппроксимация производной) — определяется метод вычисления производной **Forward** (Вперед) или **Central** (Центральная). Они соответствуют аппроксимации производной либо правой (двухточечная схема "вперед"), либо центральной (трехточечная симметричная схема) конечной разностью.

> ***Примечание***
>
> Обратите внимание, что вычисление производной в градиентных численных методах решения уравнений производится более экономичным способом, нежели при численном дифференцировании (*см. главу 3*).

Во второй строке — **Variable estimation** (Аппроксимация переменных) — можно определить тип аппроксимации рядом Тейлора. Для рассмотренного нами в этом разделе случая аппроксимации касательной прямой линией выберите переключатель **Tangent** (Касательная), для более точной квадратич-

ной аппроксимации (параболой) выберите **Quadratic** (Квадратичная). Следующая группа переключателей — **Linear variable check** (Проверка линейности) — позволяет в специфических задачах сэкономить время вычислений. Если вы уверены, что нелинейности всех функций, входящих в уравнение, мало сказываются на значениях всех их частных производных, то установите переключатель **Yes** (Да). В этом случае производные будут приняты равными константам и не будут вычисляться на каждом шаге.

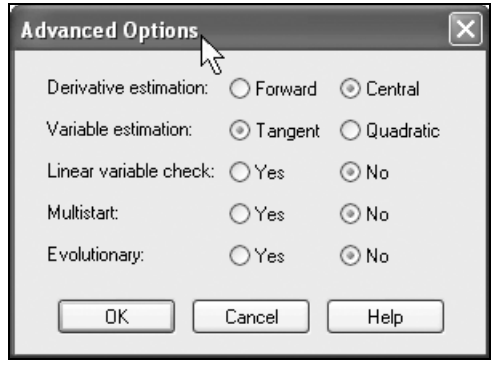



**Рис. 5.10.** Диалоговое окно **Advanced Options**

***Внимание!***

С осторожностью изменяйте параметры численных методов. Пользуйтесь ими, когда решение не находится при выставленных по умолчанию параметрах, или когда расчеты занимают очень продолжительное время.

Пара переключателей **Multistart** (Сканирование) задает опцию поиска глобального или локального минимума или максимума. Если выставлен переключатель **Yes** (Да), Mathcad будет пытаться найти наиболее глубокий экстремум из области, близкой к начальному приближению. Эта опция предназначена, в основном, для настройки (тех же самых градиентных) алгоритмов поиска экстремума, а не для решения алгебраических уравнений (*см. главу 6*).

Наконец, последний переключатель — **Evolutionary** (Эволюционный алгоритм), — если установить его в положение **Yes** (Да), позволяет использовать модификацию численного метода для решения уравнений, определяемых не обязательно гладкими функциями. Как мы убедились в этом разделе, все градиентные методы, реализованные в функции `Find`, требуют многократного

вычисления производных. Если вы работаете с достаточно гладкими функциями, то градиентные методы обеспечивают быстрый и надежный поиск корня. Для поиска корня недостаточно гладких функций одной переменной — следует либо выставить данную опцию функции `Find`, либо использовать метод секущих (функцию `root`). Помните, что правильный выбор численного метода и его параметров может помочь при решении нестандартной задачи, которая при стандартных установках может и не поддаваться решению.

## 5.3.3. Метод продолжения по параметру

Решение "хороших" нелинейных уравнений и систем типа тех, которые были рассмотрены в предыдущих разделах этой главы, представляет собой несложную, с вычислительной точки зрения, задачу. В реальных инженерных и научных расчетах очень распространена более сложная проблема: решение не одного уравнения (или системы), а целой серии уравнений, зависящих от некоторого параметра (или нескольких параметров). Для таких задач существуют очень эффективные методы, которые называются *методами продолжения*. Эти методы непосредственно не встроены в Mathcad, но могут быть легко запрограммированы с помощью уже рассмотренных нами средств. Будем далее говорить об одном уравнении, имея в виду, что всегда возможно обобщение результатов на случай системы уравнений.

Пусть имеется уравнение `f(a,x)=0`, зависящее не только от неизвестного `x`, но и от параметра `a`. Требуется определить зависимость его корня `x` от параметра `a`, т. е. `x(a)`. Простой пример такой задачи был приведен в листинге 5.3 (*см. разд. 5.1.2*). Тогда нам повезло, и решение в общем виде было найдено с помощью символьных вычислений. Рассмотрим еще один, чуть более сложный, пример алгебраического уравнения, зависящего от параметра `a` следующим образом: $\ln(a \cdot x^2)=x$ (рис. 5.11).

**Листинг 5.17. Попытка отыскания зависимости `x(a)` решения уравнения $\ln(a \cdot x^2)=x$**

$i := 0 .. 30$

$a_i := 2 + i$

$x := 1$

$y_i := \mathrm{root}\left(\ln\left(a_i \cdot x^2\right) - x, x\right)$

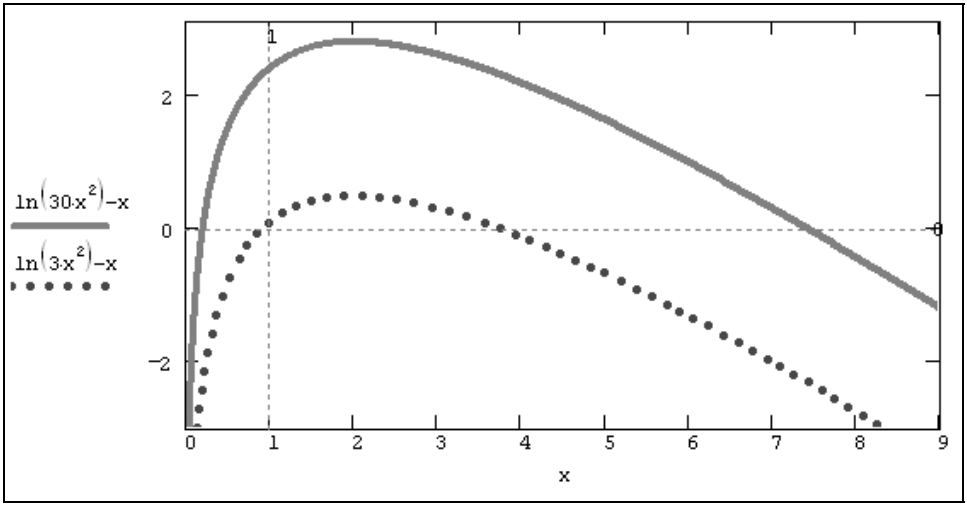



**Рис. 5.11.** График функции `ln(a·x²)-x` (для `a=3` и `a=30`)

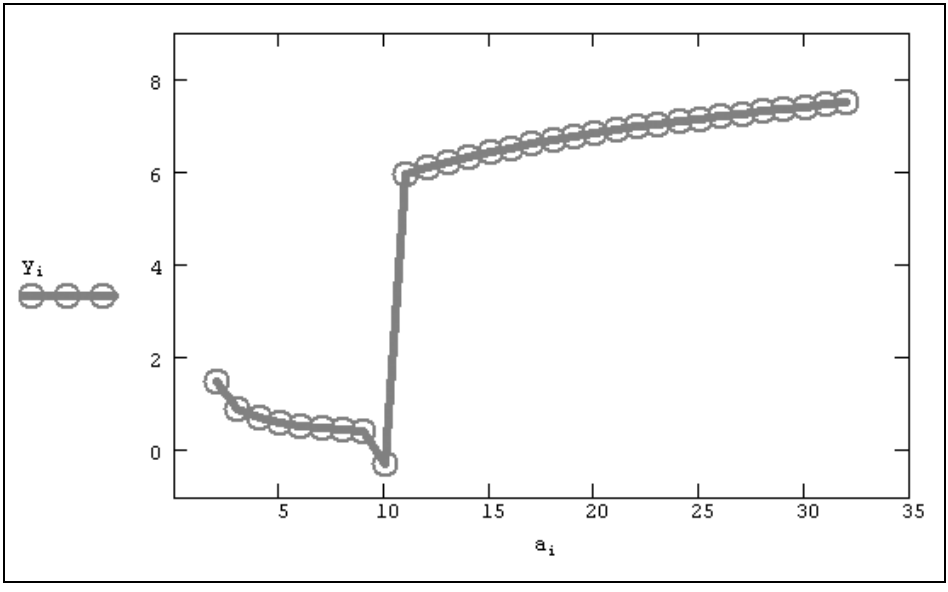



**Рис. 5.12.** Попытка отыскания зависимости `x(a)` решения уравнения `ln(a·x²)=x` (продолжение листинга 5.17)

Решим данное уравнение методом секущих, применяя для этого встроенную функцию `root`. Самый простой, но далеко не лучший, способ иллюстрирует-

ся листингом 5.17. Начинается листинг с вывода графика функции $\ln(a \cdot x^2) - x$, корни которой нам предстоит исследовать (ради определенности, для положительных значений x). Глядя на график, сразу можно сказать, что на рассматриваемом интервале уравнение будет иметь два решения (для каждого значения параметра a, бóльших некоторого порогового значения, ниже которого, видимо, уравнение вовсе не имеет корня).

Для того чтобы получить зависимость решения уравнения от параметра a, в следующих строках листинга создается ранжированная переменная i, с помощью которой определяется вектор значений параметра $a_i$. Его элементы пробегают значения от 3 до 33 с шагом 1 (эти числа взяты ради примера, вы можете поэкспериментировать с другими значениями и убедиться в том, что для значений параметра ниже порога $a \cong 3$ решение уравнения отсутствует).

Последняя строка листинга присваивает элементам еще одного вектора y вычисленные с помощью функции root значения корней уравнения для каждого $a_i$. Но для того чтобы функция root заработала, необходимо предварительно задать начальное приближение к решению, что сделано в предыдущей строке. Ключевой момент метода, примененного в листинге 5.17, заключается в том, что одно и то же начальное значение x=1 использовано для решения уравнения при всех $a_i$.

Результат расчетов $y_i$ показан на рис. 5.12. Обратите внимание, что по мере увеличения a кривая корней уравнения сначала плавно идет по одному (нижнему) семейству решений, а потом (в районе $a \approx 11$) явно срывается, "перепрыгивая" на другое семейство. С вычислительной точки зрения такая ситуация чаще всего крайне неблагоприятна, поскольку хотелось бы отыскать непрерывное семейство решений. Скачки зависимости y(a) могут вводить пользователя в заблуждение, вовсе скрывая от него существование нижнего семейства решений при a>11.

Почему же происходят эти скачки с одного семейства решений на другое? Конечно, причина кроется в выборе начального значения для вычисления каждого из корней. Линия начальных значений x=1 обозначена на графике функции (рис. 5.11) в виде пунктирной вертикальной прямой. Для $a_0=3$, и вообще для нескольких первых $a_i$, начальное значение x=1 находится ближе всего к нижнему семейству решений. Поэтому неудивительно, что численный метод находит именно эти корни. В правой части графика к линии начальных значений ближе второе (верхнее) семейство решений, к ним-то и приводит численный метод.

Приведенные соображения диктуют очень простой рецепт избавления от скачков и нахождения одного из семейств непрерывных решений. Для этого требуется при поиске каждого (i+1)-го корня взять начальное значение, по

возможности близкое к отыскиваемому семейству. Неплохим вариантом будет выбор приближения в виде предыдущего `i`-го корня, который был найден для прошлого значения параметра $a_i$. Возможный вариант воплощения этого метода, называемого *продолжением по параметру*, приведен в листинге 5.18. В нем функция `root` применена внутри функции пользователя `f(x0,a)`, определенной в самом начале листинга с помощью средств программирования. Назначение функции `f(x0,a)` заключается в том, что она выдает значение корня для заданного значения параметра `a` и начального приближения к решению `x0`. В остальном смысл листинга повторяет предыдущий, за исключением того, что осуществляется поиск сразу обоих семейств решений `y` и `z`, причем для каждого сначала явно задается начальное значение только для точки $a_0$. Для всех последующих точек, как следует из последней строки листинга, взято начальное значение, равное предыдущему корню.

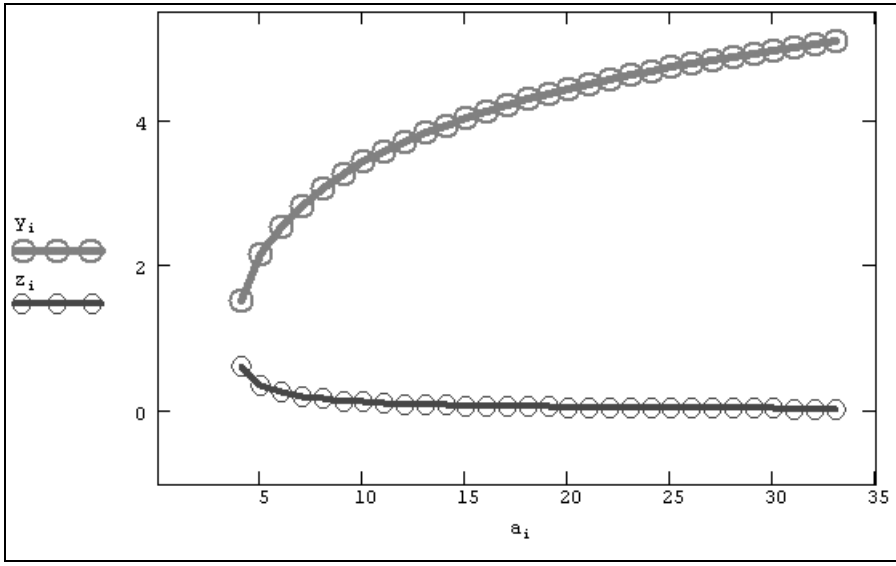



**Рис. 5.13.** Поиск зависимости `x(a)` решения уравнения $\ln(a \cdot x^2) = x$ методом продолжения (продолжение листинга 5.18)

**Листинг 5.18. Поиск зависимости `x(a)` решения уравнения $\ln(a \cdot x^2) = x$ методом продолжения**

```
f(x0, a) := | x ← x0
            | root(ln(a · x) − x, x)
```

```
i := 0 .. 30
a_i := 3 + i
y_0 := f(10 , a_0)          z_0 := f(0.1 , a_0)
y_{i+1} := f(y_i , a_i)     z_{i+1} := f(z_i , a_i)
```

Результаты вычислений, приведенные в виде двух графиков на рис. 5.13, разительно отличаются от предыдущего. Как видно, столь малое изменение идеологии применения численного метода привело к определению непрерывного семейства корней. Отметим, что получить результат рис. 5.12 (без продолжения по параметру) в терминах введенной нами в листинге 5.18 функции `f(x0,a)` можно, изменив ее первый аргумент на константу: `f(1,a_i)`.

> ***Примечание***
>
> С помощью метода продолжения можно решать и соответствующие задачи оптимизации, зависящие от параметра. Идеология в этом случае остается точно такой же, но вместо функций решения нелинейных уравнений `root` или `Find` вам следует применить одну из функций поиска экстремума `Minerr`, `Maximize` или `Minimize` (*см. разд. 6.1 и 6.2*).

Мы привели основную идею и один из возможных способов реализации метода продолжения по параметру. Безусловно, вы можете предложить иные, как математические, так и программистские решения этой проблемы. В частности, для выбора очередного начального приближения к корню можно использовать результат экстраполяции уже найденной зависимости `x(a)`, придумать более сложные алгоритмы для ветвящихся семейств решений и т. д.

# Глава 6

# Оптимизация

В этой главе рассматриваются задачи на поиск экстремума функций и близкие к ним задачи приближенного решения алгебраических нелинейных уравнений и систем. Задачи поиска *экстремума* функции близки к задачам на нахождение ее *максимума* (наибольшего значения) или *минимума* (наименьшего значения) в некоторой области определения ее аргументов. С вычислительной точки зрения две задачи являются практически одинаковыми, т. к., например, задача поиска максимума `f(x)` тождественна проблеме отыскания минимума `-f(x)`. Поэтому ниже будем часто называть задачу поиска экстремума функции задачей минимизации.

Общая проблема поиска экстремума функции включает в себя задачи нахождения *локального* и *глобального* минимума. Последние называют еще *задачами оптимизации*, и решить их, как правило, намного труднее, поскольку они подразумевают локализацию всех минимумов `f(x)` и выбор из них наименьшего. (На рис. 6.1 показаны два локальных минимума функции, из которых левый является глобальным.) Ограничения значений аргументов, задающих область определения `f(x)`, как и прочие дополнительные условия, могут быть определены в виде системы неравенств и (или) уравнений. В таком случае говорят о задаче на *условный экстремум*.

Численные методы, применяемые для минимизации, сходны с методами решения нелинейных уравнений, и поэтому материал этой главы близок по содержанию к предыдущей.

> ***Примечание***
>
> Решение задач минимизации в Mathcad реализовано только при помощи численного алгоритма. Таким образом, непосредственное символьное нахождение минимума невозможно. Однако аналитический поиск экстремума функции несложно запрограммировать, опираясь на соответствующие сведения математического анализа (*см. разд. 6.1.5*).

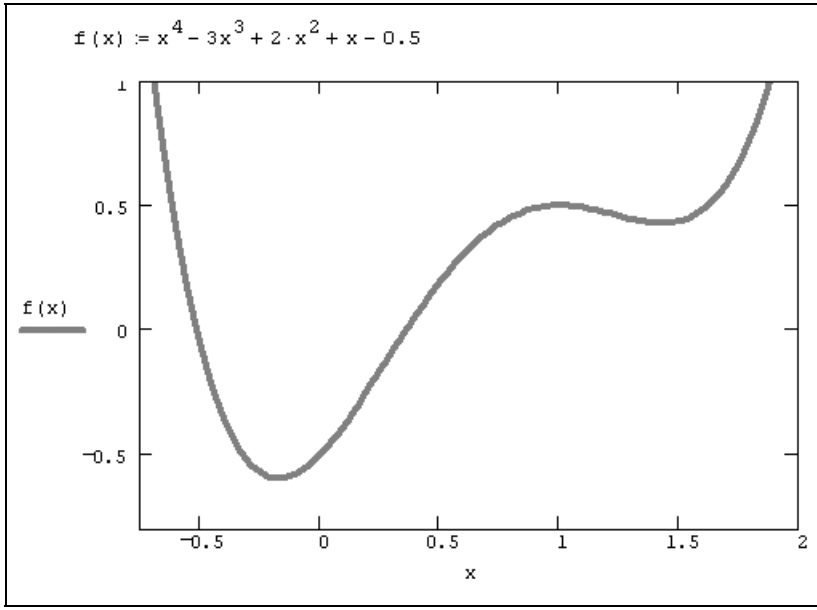



**Рис. 6.1.** К пояснению задач поиска локального и глобального экстремума

# 6.1. Поиск экстремума функции

Для численного решения задач поиска локального максимума и минимума в Mathcad имеются встроенные функции `Minerr`, `Minimize` и `Maximize`. Принцип их действия очень близок к принципу расчетов, заложенных во встроенной функции `Find`, предназначенной для решения алгебраических уравнений (*см. главу 5*). В частности, все встроенные функции минимизации используют те же градиентные численные методы, что и функция `Find`, поэтому допускается "вручную" выбирать численный алгоритм минимизации из уже рассмотренных нами численных методов (*см. разд. 5.3.2*). Кроме того, как и в случае решения уравнений, применение градиентного алгоритма, во-первых, требует задания некоторого начального приближения к точке минимума и, во-вторых, позволяет отыскать лишь один (т. е. локальный) из минимумов функции.

Таким образом, как и в случае решения уравнений (*см. разд. 5.2.4*), чтобы найти глобальный максимум (или минимум), требуется сначала *просканировать* с некоторым шагом рассматриваемую область и вычислить все локальные значения и потом выбрать из них наибольший (наименьший). Другим вариантом будет простое сканирование с вычислением значений функции,

позволяющее выделить из нее подобласть наибольших (наименьших) значений функции и осуществить поиск глобального экстремума, уже находясь в его окрестности. Последний путь таит в себе некоторую опасность уйти в зону другого локального экстремума, но часто может быть предпочтительнее из соображений экономии времени.

## 6.1.1. Локальный экстремум

Для поиска локальных экстремумов имеются две встроенные функции, которые могут применяться как в пределах вычислительного блока, так и автономно:

- `Minimize(f,x1,…,xM)` — вектор значений аргументов, при которых функция `f` достигает минимума;
- `Maximize(f,x1,…,xM)` — вектор значений аргументов, при которых функция `f` достигает максимума:
  - `f(x1,…,xM,…)` — функция;
  - `x1,…,xM` — аргументы, по которым производится минимизация (максимизация).

> ***Примечание***
>
> Вычислительный блок (ключевое слово `Given` со следующими после него логическими выражениями) обычно используется в задачах на условный экстремум (*см. следующий разд.*).

В качестве примера рассмотрим задачу численного поиска экстремумов полинома четвертой степени `f(x)`, график которого был приведен на рис. 6.1. Как известно, парабола четвертой степени имеет три точки экстремума, и все они видны на рис. 6.1.

Всем аргументам функции `f` предварительно следует присвоить некоторые значения, причем для тех переменных, по которым производится минимизация, они будут восприниматься как начальные приближения. Примеры вычисления локальных экстремумов функции одной переменной показаны в листингах 6.1—6.2. Поскольку никаких дополнительных условий в них не вводится, поиск экстремумов выполняется для любых значений `x` от $-\infty$ до $\infty$.

**Листинг 6.1. Поиск минимума функции одной переменной (для трех начальных значений x)**

$f(x) := x^4 - 3x^3 + 2 \cdot x^2 + x - 0.5$

$x := 0$

```
Minimize(f, x) = -0.175

x := 100

Minimize(f, x) = 1.425

x := 1.426

Minimize(f, x) = -0.175
```

**Листинг 6.2. Поиск максимума функции одной переменной**

```
f(x) := x^4 - 3x^3 + 2·x^2 + x - 0.5

x := 5

Maximize(f, x) = 1

x := 10

Maximize(f, x) =
```

Как видно из листингов, существенное влияние на результат оказывает выбор начального приближения, в зависимости от чего в качестве ответа выдаются различные локальные экстремумы. Очень полезно сопоставить результаты минимизации (листинг 6.1) с графиком функции `f(x)` (см. рис. 6.1). Как видно, функция `Minimize` очень уверенно находит глубокий минимум `x=-0.75`. А вот на второй (плохо выраженный) минимум можно набрести лишь случайно, выбирая определенные начальные значения `x`. В последнем из трех примеров демонстрируется, что если взять начальное приближение `x` даже в непосредственной близости от этого локального минимума, численный метод все равно "сваливается" в первый, более глубокий минимум `f(x)`. Попробуйте повторить расчеты, выбирая различные начальные значения, чтобы в этом убедиться.

В листинге 6.2 показаны аналогичные свойства функции `Maximize`. Если начальное приближение выбрать удачно, то итерационный процесс алгоритма сойдется к максимуму функции, а вот если выбрать его вдали от него, на участке `f(x)`, где неограниченно возрастает (при $x \to \pm\infty$), численный метод вообще не справится с задачей, выдавая сообщение об ошибке. Это происходит, поскольку начальное приближение `x=-10` выбрано далеко от области локального максимума, и поиск решения уходит в сторону увеличения `f(x)`, т. е. расходится при $x \to \infty$.

### Примечание

Помните о возможности выбора численного алгоритма минимизации, который осуществляется при помощи контекстного меню (рис. 6.2). Не забывайте также, что, начиная с версии Mathcad 11, имеется возможность управлять параметром **Multistart** (Сканирование), при помощи которого можно попытаться организовать поиск глобального экстремума (*см. разд. 5.3.2*). Однако не слишком полагайтесь на эту опцию и, если перед вами стоит задача поиска глобального экстремума, постарайтесь организовать сканирование вручную.

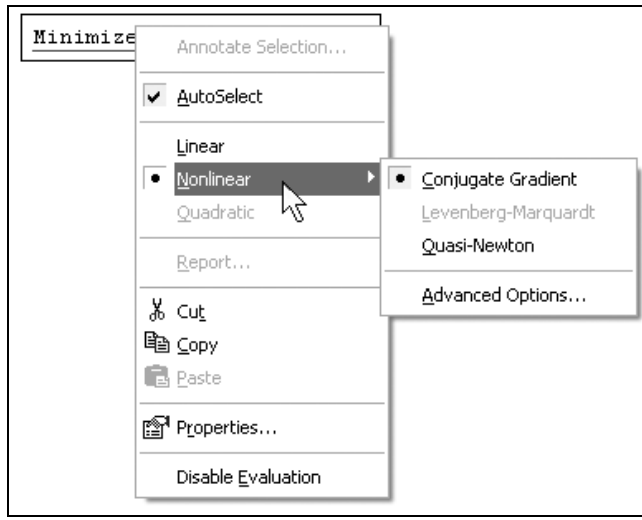



**Рис. 6.2.** Выбор численного метода минимизации

## 6.1.2. Условный экстремум

В задачах на условный экстремум встроенные функции минимизации и максимизации должны быть включены в вычислительный блок, т. е. им должно предшествовать ключевое слово `Given`. В промежутке между `Given` и функцией поиска экстремума с помощью булевых операторов записываются логические выражения (неравенства, уравнения), задающие ограничения на значения аргументов минимизируемой функции. Листинги 6.3 и 6.4 содержат примеры поиска условного экстремума на различных интервалах, определенных неравенствами. У вас не должны возникнуть сложности с записью условий в Mathcad, а вот разобраться в скрытой от глаз вычислительной стороне будет полезно. С этой точки зрения поучительно сравнить результаты работы приведенных листингов с листингами 6.1 и 6.2 соответственно.

**Листинг. 6.3. Поиск условного минимума**

```
f(x) := x^4 - 3x^3 + 2·x^2 + x - 0.5
x := 0
Given
1 < x < ∞
Minimize(f, x) = 1.425
```

**Листинг. 6.4. Поиск условного максимума**

```
f(x) := x^4 - 3x^3 + 2·x^2 + x - 0.5
x := 1
Given
2 < x < 10
Maximize(f, x) = 10
```

Как видно из листинга 6.3, если ограничить значения `x` интервалом, расположенным в окрестности правого локального максимума, с поиском которого мы встретили большие сложности при решении задачи на безусловный экстремум, то этот максимум будет без труда найден численным методом. Следует помнить также об одной особенности, иллюстрируемой листингом 6.4. А именно (в случае максимизации), если на границе интервала `f(x)` достигает большего значения, нежели на локальном максимуме внутри интервала, то в качестве решения, скорее всего, будет выдано наибольшее значение (т. е. граница интервала).

Конечно, если на рассматриваемом интервале `x` расположено несколько локальных максимумов, ответ станет еще менее предсказуемым, поскольку будет напрямую зависеть от выбранного начального приближения. Не забывайте о важности его правильного выбора и в случае задач на условный экстремум. В частности, если вместо начального значения `x=-1` задать `x=1`, то в качестве максимума будет найдена правая граница интервала: `Maximize(f,x)=2`, что неверно, поскольку максимальное значение достигается функцией `f(x)` на левой границе интервала при `x=-3`. Выбор начального приближения `x=-4` решает задачу правильно, выдавая в качестве результата `Maximize(f,x)=-5`.

## 6.1.3. Экстремум функции нескольких переменных

Вычисление экстремума функции многих переменных не несет принципиальных особенностей по сравнению с функциями одной переменной. Поэтому ограничимся примером нахождения максимума и минимума функции, показанной в виде графиков трехмерной поверхности и линий уровня (листинг 6.5).

Привлечем внимание читателя только к тому, как с помощью неравенств, введенных логическими операторами, задается область на плоскости `(X,Y)`.

**Листинг 6.5. Экстремум функции двух переменных**

$$f(x,y) := 2\cdot(x-5.07)^2+(y-10.03)^2-0.2\cdot(x-5.07)^3$$

$$x := 1$$

$$y := 1$$

Given

$$0 < x < 15$$

$$0 < y < 20$$

$$\text{Minimize}(f,x,y) = \begin{pmatrix} 5.07 \\ 10.03 \end{pmatrix}$$

Дополнительные условия могут быть заданы и равенствами. Например, определение после ключевого слова `Given` уравнения `x+y=10` приводит к такому решению задачи на условный экстремум:

$$\text{Minimize}(f,x,y) = \begin{pmatrix} 3.589 \\ 6.411 \end{pmatrix}.$$

Как нетрудно сообразить, новое дополнительное условие означает, что численный метод ищет минимальное значение функции `f(x,y)` не во всей прямоугольной области `(X,Y)`, а лишь вдоль отрезка прямой, показанного на рис. 6.3.

***Примечание***

Поиск минимума можно организовать и с помощью функции `Minerr`. Для этого в листинге 6.5 надо поменять имя функции `Minimize` на `Minerr`, а после ключевого слова `Given` добавить выражение, приравнивающее функции `f(x,y)` значение, заведомо меньшее минимального, например, `f(x,y)=0`.

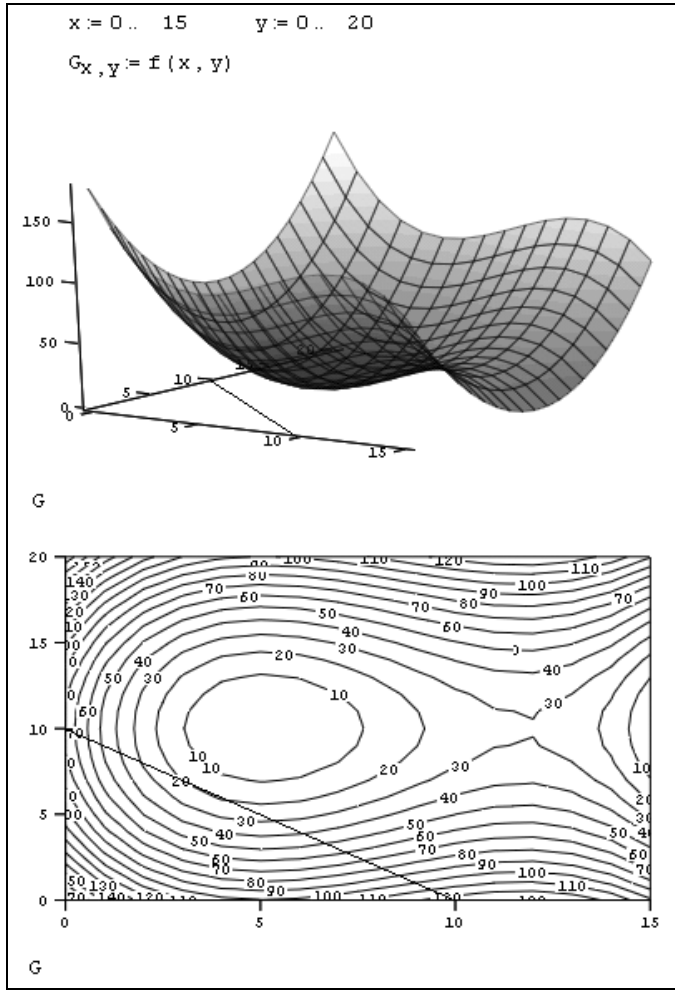



**Рис. 6.3.** Иллюстрация задачи на условный экстремум функции двух переменных: график функции `f(x,y)` и отрезок прямой `x+y=10` (продолжение листинга 6.5)

## 6.1.4. Пример: линейное программирование

Задачи поиска условного экстремума функции многих переменных часто встречаются в экономических расчетах для минимизации издержек, финансовых рисков, максимизации прибыли и т. п. Целый класс экономических задач оптимизации описывается системами линейных уравнений и неравенств. Они называются задачами *линейного программирования*. Приведем

характерный пример так называемой *транспортной задачи*, которая решает одну из проблем оптимальной организации доставки товара потребителям с точки зрения экономии транспортных средств (листинг 6.6).

**Листинг 6.6. Решение задачи линейного программирования**

$$a := \begin{pmatrix} 145 \\ 210 \\ 160 \end{pmatrix} \qquad b := \begin{pmatrix} 237 \\ 278 \end{pmatrix}$$

$$\sum a = 515 \qquad \sum b = 515$$

$$M := \mathrm{rows}(a) \qquad N := \mathrm{rows}(b)$$

$$M = 3 \qquad N = 2$$

$$c := \begin{pmatrix} 11.5 & 7 & 12 \\ 6.2 & 10 & 9.0 \end{pmatrix}$$

$$f(x) := \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} c_{i,j} \cdot x_{i,j}$$

$$x_{N-1,M-1} := 0$$

Given

$$x_{0,0} + x_{0,1} + x_{0,2} = b_0 \quad x_{0,0} + x_{1,0} = a_0 \quad x_{0,0} \geq 0 \quad x_{1,0} \geq 0$$

$$x_{1,0} + x_{1,1} + x_{1,2} = b_1 \quad x_{0,1} + x_{1,1} = a_1 \quad x_{0,1} \geq 0 \quad x_{1,1} \geq 0$$

$$x_{0,2} + x_{1,2} = a_2 \quad x_{0,2} \geq 0 \quad x_{1,2} \geq 0$$

$$sol := \mathrm{Minimize}(f, x)$$

$$sol = \begin{pmatrix} 0 & 210 & 27 \\ 145 & 0 & 133 \end{pmatrix} \qquad f(sol) = 3.89 \times 10^3$$

Модель типичной транспортной задачи следующая. Пусть имеется $N$ предприятий-производителей, выпустивших продукцию в количестве $b_0, \ldots, b_{N-1}$ тонн. Эту продукцию требуется доставить $M$ потребителям в количестве $a_0, \ldots, a_{M-1}$ тонн каждому. Численное определение векторов $a$ и $b$ находится в первой строке листинга. Сумма всех заказов потребителей $a_i$ равна сумме произведенной продукции, т. е. всех $b_i$ (проверка равенства во второй строке). Если известна стоимость перевозки тонны продукции от $i$-го производителя к $j$-му потребителю $c_{ij}$, то решение задачи задает оптимальное распределе-

ние соответствующего товаропотока $x_{ij}$ с точки зрения минимизации суммы транспортных расходов. Матрица `c` и минимизируемая функция `f(x)` матричного аргумента `x` находятся в середине листинга 6.6.

Условия, выражающие неотрицательность товаропотока, и равенства, задающие сумму произведенной каждым предприятием продукции и сумму заказов каждого потребителя, находятся после ключевого слова `Given`. Решение, присвоенное матричной переменной `sol`, выведено в последней строке листинга вместе с соответствующей суммой затрат. В строке, предшествующей ключевому слову `Given`, определяются нулевые начальные значения для `x` простым созданием нулевого элемента матрицы $x_{N-1,M-1}$.

### Примечание 1

Если взять другие начальные значения для `x`, решение, скорее всего, будет другим! Возможно, вы сумеете отыскать другой локальный минимум, который еще больше минимизирует транспортные затраты. Это еще раз доказывает, что задачи на глобальный минимум, к классу которых относится линейное программирование, требуют аккуратного отношения в смысле выбора начальных значений. Часто ничего другого не остается, кроме сканирования всей области начальных значений, чтобы из множества локальных минимумов выбрать наиболее глубокий.

### Примечание 2

Не забывайте о том, что в вычислительном блоке в качестве неизвестных следует использовать все компоненты вектора `x`, как это сделано в предпоследней строке листинга 6.6 (*см. разд. 5.1.1*). Иными словами, нельзя задавать некоторые из компонент `x` в виде параметров.

### Примечание 3

Еще раз напомним, что в новых версиях Mathcad (начиная с 11-й) появилась возможность программного управления глобальной минимизацией. Для этого служит параметр численного алгоритма **Multistart** (Сканирование), который следует установить в положение **Yes** (Да) для поиска глобального экстремума (*см. разд. 5.3.2*). Любопытно, что если при решении нашей задачи поменять только эту опцию функции `Minimize`, то численный метод даст сбой, и вместо решения появится сообщение об ошибке. Однако если вспомнить, что наша транспортная задача — линейная, и выставить в контекстном меню, вызываемом на имени функции `Minimize`, флажок **Linear** (Линейный), задающий линейный вариант алгоритма, то глобальное решение будет найдено.

## 6.1.5. Аналитическое решение задач на экстремум

Несмотря на то, что, как уже говорилось, разработчиками Mathcad символьное решение задач оптимизации не предусмотрено, пользователь все-таки имеет возможность аналитического исследования экстремумов функций, опираясь на базовые сведения математического анализа. Следует лишь вспомнить о том, что (при выполнении соответствующих условий на непрерывность и гладкость функции) точки экстремума `f(x)` характеризуются тем, что в них производная этой функции проходит через нулевое значение. Тип экстремума (максимум или минимум) определяется знаком второй производной в этой точке.

Таким образом, имея в виду данные правила, не представляет особого труда организовать аналитическое решение задачи на экстремум, центральным моментом которого будет решение алгебраического уравнения `f'(x)=0`. Сразу стоит подчеркнуть, что можно использовать гибрид символьных и аналитических расчетов, когда, например, производная `f'(x)` считается аналитически, а уравнение `f'(x)=0` (если символьное решение получить не удается) — численно. В этом случае во всей красе может проявиться мощь Mathcad, предоставляющего пользователю богатый арсенал как аналитических, так и численных методов.

Приведем несложный пример реализации аналитического поиска экстремумов той же функции (полинома 4-й степени), которая использовалась нами при демонстрации возможностей численных методов (*см. разд. 6.1.1*). В листинге 6.7, в первой строке, приводится определение `f(x)`, а затем при помощи символьного процессора осуществляется отыскание корней нелинейного уравнения `f'(x)=0`. Результатом решения являются все три точки экстремума (последняя строка листинга). На рис. 6.4 показан график функций `f(x)` и `f'(x)`, и легко убедиться, что экстремумам исходной функции соответствуют нули ее производной.

**Листинг 6.7. Аналитический поиск экстремумов функции одной переменной**

$$f(x) := x^4 - 3x^3 + 2 \cdot x^2 + x - 0.5$$

Given

$$\frac{d}{dx} f(x) = 0$$

$$\text{Find}(x) \rightarrow \left( 1 \quad \frac{5}{8} + \frac{1}{8}\cdot\sqrt{41} \quad \frac{5}{8} - \frac{1}{8}\cdot\sqrt{41} \right) = (1 \quad 1.425 \quad -0.175)$$

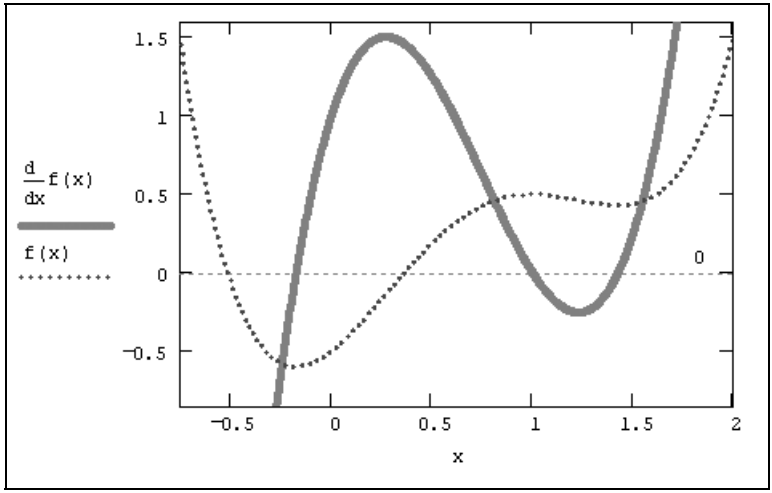



**Рис. 6.4.** Функция `f(x)` и ее производная
(продолжение листинга 6.7)

Чтобы завершить анализ `f(x)` на экстремумы, необходимо определить, какие из найденных точек являются точками минимума, а какие — максимума. Для этого (листинг 6.8) следует просто рассчитать значения второй производной в результатах решения уравнения и определить ее знак.

**Листинг 6.8. Анализ типа точек экстремума (продолжение листинга 6.7)**

$x := 1$ $\quad \text{sign}\left( \frac{d^2}{dx^2} f(x) \right) \cdot |f(x)| = -0.5$

$x := \frac{5}{8} + \frac{1}{8}\cdot\sqrt{41}$ $\quad \text{sign}\left( \frac{d^2}{dx^2} f(x) \right) \cdot |f(x)| = 0.429$

$x := \frac{5}{8} - \frac{1}{8}\cdot\sqrt{41}$ $\quad \text{sign}\left( \frac{d^2}{dx^2} f(x) \right) \cdot |f(x)| = 0.597$

# ⓘ 6.2. Приближенное решение алгебраических уравнений

Градиентные численные методы решения задач отделения корней уравнений и поиска экстремума функций очень близки. Поэтому, в частности, пользователь может тем же самым образом, с помощью контекстного меню, выбирать конкретный метод приближенного решения для функций `Minimize` и `Maximize`. Применительно к нелинейным уравнениям основная идея градиентных алгоритмов была приведена в *разд. 5.3.2*. Основным отличием в случае задач минимизации является критерий правильности решения (прекращения итераций): если при решении уравнений критерием служит близость нулю невязки системы уравнений, то при минимизации критерий заключается в схождении итераций к минимальным значениям исследуемой функции.

Близость рассмотренных задач связана также с тем, что иногда приходится заменять проблему решения нелинейных уравнений задачей поиска экстремума функции многих переменных. Например, когда невозможно найти решение с помощью функции `Find`, можно попытаться потребовать вместо точного выполнения уравнений условий минимизировать их невязку. Чтобы не нужно было реализовывать данную схему вручную (вместо встроенной функции `Find` использовать соответствующую постановку задачи для функции `Minimize`), разработчики Mathcad предусмотрели дополнительную встроенную функцию `Minerr`. Она применяется аналогично функции `Find` (*см. разд. 5.1.1*), в частности, имеет тот же самый набор параметров и должна находиться в пределах вычислительного блока.

1. $x_1:=C_1$ … $x_M:=C_M$ — начальные значения для неизвестных.
2. `Given` — ключевое слово.
3. Система алгебраических уравнений и неравенств, записанная логическими операторами.
4. `Minerr(`$x_1$`,…,`$x_M$`)` — приближенное решение системы относительно переменных $x_1,\ldots,x_M$, минимизирующее невязку системы уравнений.

> ⓘ ***Примечание 1***
>
> В функции `Minerr` реализованы те же самые алгоритмы, что и в функции `Find`, иным является только условие завершения работы численного метода (как в случае функций `Minimize` и `Maximize`).

***Примечание 2***

Согласно своему математическому смыслу, функция Minerr может применяться для построения регрессии серии данных по закону, заданному пользователем (*см. главу 13*).

Пример использования функции Minerr показан в листинге 6.9. Как видно, достаточно заменить в вычислительном блоке имя функции на Minerr, чтобы вместо точного (конечно, с поправкой на погрешность порядка TOL) получить приближенное решение уравнения, заданного после ключевого слова Given.

**Листинг 6.9. Приближенное численное решение уравнения, имеющего корень (x=0, y=0)**

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

$k := 10^6$

Given

$k \cdot x^2 + y^2 = 0$

$MinErr(x, y) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

Листинг 6.9 демонстрирует приближенное решение уравнения $k \cdot x^2 + y^2 = 0$, которое при любом значении коэффициента k имеет единственный точный корень (x=0, y=0). Тем не менее при попытке решить его функцией Find для больших k, порядка принятых в листинге (рис. 6.5), происходит генерация ошибки "No solution was found" (Решение не найдено). Это связано с иным поведением функции $f(x,y) = k \cdot x^2 + y^2$ вблизи ее корня по сравнению с функциями, поиск корней которых был разобран в предыдущей главе (см. рис. 5.1).

В отличие от них, f(x,y) не пересекает плоскость f(x,y)=0, а лишь касается ее (рис. 6.5) в точке (x=0, y=0). Поэтому и найти корень изложенными в предыдущем разделе градиентными методами намного сложнее, поскольку вблизи корня производные f(x,y) близки к нулю, и итерации, строящиеся по градиентному принципу, могут уводить предполагаемое решение далеко от корня.

Ситуация еще более ухудшается, если наряду с корнем типа касания (как на рис. 6.5) имеются (возможно, весьма удаленные) корни типа пересечения. Тогда попытка решить уравнение или систему уравнений с помощью функции Find может приводить к нахождению корня второго типа, даже если на-

чальное приближение было взято очень близко к первому. Поэтому если вы предполагаете, что система уравнений имеет корень типа касания, намного предпочтительнее использовать функцию `Minerr`, тем более что всегда есть возможность проконтролировать невязку уравнений простой подстановкой полученного решения в исходную систему уравнений.

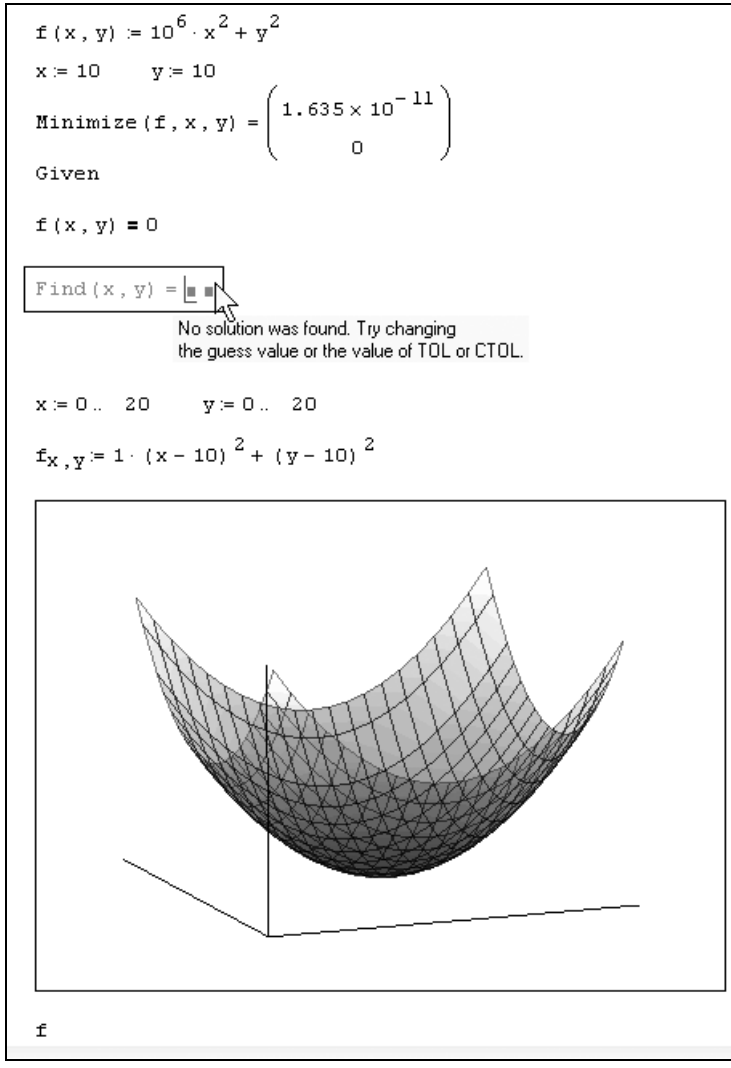



**Рис. 6.5.** Поиск минимума и попытка нахождения корня функции $f(x,y)=k\cdot x^2+y^2$

Пока мы рассматривали пример нахождения существующего решения уравнения. Приведем теперь пример нахождения функцией Minerr приближенного решения не имеющего корней уравнения (листинг 6.10), а также несовместной системы уравнений и неравенств (листинг 6.11). Решение, выдаваемое функцией Minerr, минимизирует невязку данной системы. Как видно из листингов, в качестве результата выдаются значения переменных, наилучшим образом удовлетворяющие уравнению и неравенствам внутри вычислительного блока.

***Внимание!***

Полученное в листинге 6.11 решение не удовлетворяет неравенствам, составляющим задачу. Это и неудивительно, поскольку точного решения системы нет, и в качестве ответа Mathcad выдает значения аргументов, минимизирующих норму общей невязки (не отдавая предпочтения ни уравнению, ни неравенствам).

**Листинг 6.10. Приближенное решение уравнения $x^2+y^2+1=0$**

$x := 10 \qquad y := 10$

Given

$x^2 + y^2 + 1 = 0$

$\text{MinErr}(x, y) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

**Листинг 6.11. Приближенное решение несовместной системы уравнений и неравенств**

$x := 10 \qquad y := 10$

Given

$x^2 + y^2 + 1 = 0$

$x > 1$

$y > 2$

$\text{MinErr}(x, y) = \begin{pmatrix} 0.125 \\ 2 \end{pmatrix}$

Внимательный читатель может обнаружить, что решение, выдаваемое функцией Minerr в рассматриваемом примере, не является единственным, по-

скольку множество пар значений (x,y) в равной степени минимизирует невязку данной системы уравнений и неравенств. Поэтому для различных начальных значений будут получаться разные решения, подобно тому, как разные решения выдаются функцией Find в случае бесконечного множества корней (*см. разд. 5.2.4*). Еще более опасен случай, когда имеются всего несколько локальных минимумов функции невязки. Тогда неудачно выбранное начальное приближение приведет к выдаче именно этого локального минимума, несмотря на то, что другой (глобальный) минимум невязки может удовлетворять системе гораздо лучше.

В завершение раздела сделаем очень важное замечание, связанное с возможностью использования встроенной функции Minerr в символьных расчетах. Как и функция решения алгебраических систем Find, она может применяться без предварительного присвоения каких-либо начальных значений любым переменным, входящим в уравнение, как это проиллюстрировано листингом 6.12, решающим ту же самую задачу аналитически. Замечательно, что в результате получается не одно решение, а все семейство решений, одинаково минимизирующее невязку.

**Листинг 6.12. Аналитическое приближенное решение уравнения $k \cdot x^2 + y^2 + 1 = 0$**

Given

$$k \cdot x^2 + y^2 + 1 = 0$$

$$\text{MinErr}(x, y) \rightarrow \begin{bmatrix} \frac{1}{k} \cdot \left[-k \cdot \left(y^2 + 1\right)\right]^{\frac{1}{2}} & \frac{-1}{k} \cdot \left[-k \cdot \left(y^2 + 1\right)\right]^{\frac{1}{2}} \\ y & y \end{bmatrix}$$

## 6.3. Пример: некорректные задачи

Еще один широко распространенный круг задач на решение систем уравнений, называемых *обратными*, наиболее типичен для современной экспериментальной физики. Значительную часть обратных задач относят к классу некорректно поставленных (или просто *некорректных*), которые, благодаря своей принципиальной неустойчивости, требуют специального подхода. Как правило, общим принципом решения некорректных задач является их регуляризация, заключающаяся в их сведении к задаче минимизации.

Рассмотрим сначала типичную постановку обратных задач, а затем обратимся к обсуждению корректности их постановки. Строго говоря, обратные задачи связаны, как правило, не с отысканием значения корня уравнения (или системы) `f(x)=0`, а с вычислением неизвестной функции `y(x)`, описываемой уравнением

$$S[y(x)]=b(x). \qquad (6.1)$$

Здесь `S[y(x)]` — некоторый *функционал* (т. е. функция от функции), а `b(x)` — известная функция (правая часть уравнения).

Рассмотрим типичную постановку обратных задач на примере так называемой инструментальной задачи. Пусть имеется некоторый сигнал `y(x)`, который подвергается измерению на приборе, условно обозначаемом `S`. Физику-исследователю доступно измерение данного сигнала, которое находится на выходе прибора (дисплее, табло и т. п.). Обозначим это измерение `b(x)`. Поскольку прибор вносит в сигнал, во-первых, искажения (например, в устройствах типа спектрометров часто измеряются некоторые интегральные характеристики сигнала) и, во-вторых, шумовую компоненту. Формально данную физическую модель можно записать равенством (6.1), в котором оператором `S` обозначается *аппаратная функция*, т. е. действие прибора на сигнал `y(x)`.

**Листинг 6.13. Пример моделирования прямой задачи, выражающей линейную схему измерений**

```
y(x) := cos(x)

k := 20        σ := 0.5                    A(x) := exp(-k·x^2)

S(x) := ∫_0^10 A(|x - χ|)·y(χ) dχ + σ·(rnd(1) - 1/2)

x := 0, 0.1 .. 10

b(x) := S(x)
```

Согласно изложенной модели, измерения `b(x)` могут довольно сильно отличаться от исходного сигнала `y(x)`, что иллюстрируется простым примером листинга 6.13 и рис. 6.6. В первой строке листинга выбирается модельный сигнал `y(x)`, а во второй и третьей — определяется оператор `S[y(x)]`. Важно отметить, что использованная структура оператора — интегральная зависимость от сигнала `y(x)` в сумме с шумовой компонентой — наиболее типична для инструментальных обратных задач. Читателю будет очень полезно "по-

играть" с параметрами задачи k и σ (эффективной шириной спектральной характеристики прибора и интенсивностью шума соответственно), чтобы "почувствовать" специфику модели измерений.

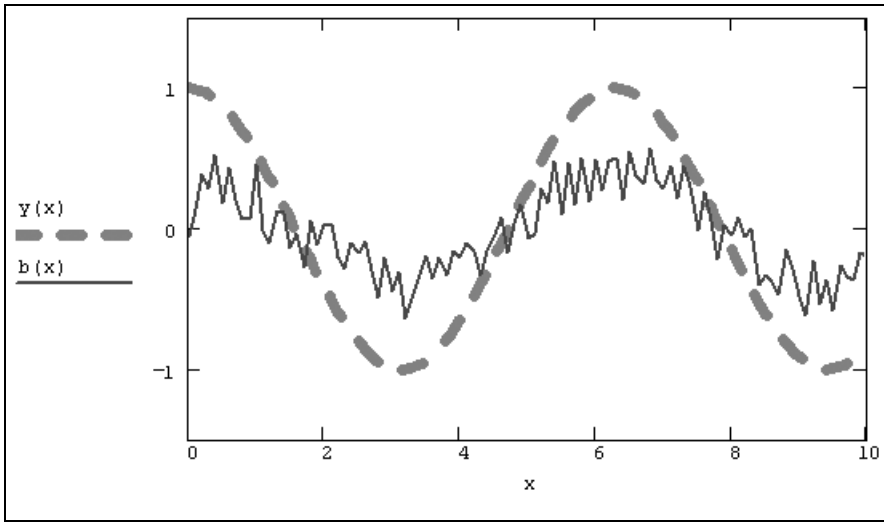



**Рис. 6.6.** Исходный сигнал и показания прибора
(продолжение листинга 6.13)

### *Примечание*

Модель измерений, представленная в третьей строке листинга 6.13, описывает, с математической точки зрения, типичное *интегральное уравнение*, в которое неизвестная функция y(x) входит в виде части подынтегральной функции. Класс обратных задач чаще всего (но не всегда) соответствует как раз интегральным уравнениям.

Из сказанного выше понятно, что в практических приложениях весьма актуальна задача восстановления сигнала y(x) по показаниям прибора b(x) (при наличии определенной дополнительной информации о физике измерения, т. е. об операторе S, выражающем действие прибора). Таким образом, в отличие от прямой задачи, выражающейся равенством (6.1), обратной задачей является восстановление функции y(x) по известной b(x).

Обычно сигнал (и, соответственно, его измерение) может зависеть от времени и/или пространственных координат. Эту зависимость мы обозначили как зависимость от x. При использовании численных методов непрерывные

зависимости от x дискретизируются, заменяясь соответствующими векторами. Таким образом, задача отыскания неизвестной функции y(x) заменяется задачей поиска неизвестного вектора Y и может быть записана в матричном виде

$$S(Y)=B, \tag{6.2}$$

где вектор y неизвестен, а оператор S (в линейном случае $S(Y)=A\cdot Y$, где A — матрица) и вектор правых частей уравнений B известны. Таким образом, в дискретном случае обратная задача для рассматриваемой математической модели состоит в решении системы (в общем случае, нелинейных) уравнений (6.2). Построенная описанным способом дискретная задача представлена в листинге 6.14. Важно отметить, что, поскольку исходная задача линейна, дискретный вариант ее оператора записывается в виде матрицы A, а сама задача сводится к системе линейных алгебраических уравнений $A\cdot Y=B$.

***Примечание***

Рекомендуем читателю обратиться к соответствующему файлу Mathcad на прилагаемом к книге компакт-диске и сравнить графики сигнала и измерений, соответствующие непрерывной (интегральной) и дискретной (матричной) модели. Будет очень полезным осуществить расчеты для разных значений параметров, в том числе количества точек дискретизации N, а также разобраться, какова структура матрицы A.

**Листинг 6.14. Дискретная форма прямой задачи линейной модели измерений (продолжение листинга 6.13)**

$$N := 100 \qquad h := \frac{10}{N}$$

$$i := 0 \,..\, N \qquad j := 0 \,..\, N$$

$$X_i := i \cdot h \qquad A_{i,j} := A\left(\left|X_i - X_j\right|\right) \cdot h$$

$$Y_i := y\left(X_i\right)$$

$$B_i := b\left(X_i\right)$$

При решении обратных задач важную роль играет их устойчивость. Задача устойчива, если малым флуктуациям правых частей, т. е. вектора B, соответствуют малые флуктуации решения Y. Иными словами, *устойчивость по правой части* состоит в требовании, чтобы решения близких задач $A\cdot Y=B$ и $A\cdot Y=B+\delta B$ мало отличались друг от друга. Если задача изначально является не-

устойчивой, то решать ее нет смысла, поскольку погрешности алгоритмов, накапливающиеся в ходе решения численными методами, неизбежно приведут к тому, что будет найдено неверное решение.

Как правило (в том числе в нашем примере), обратные задачи характеризуются наличием шумов, что может быть символически отражено равенством (если опять-таки предположить, что шум σ входит в схему измерений линейно):

$$A \cdot Y + \sigma = B. \qquad (6.3)$$

Наличие шума коренным образом меняет идеологию решения обратных задач. Если сама задача является устойчивой, то существование шума может эту устойчивость нарушать. Попросту говоря, различные (даже очень сильно отличающиеся) сигналы `Y1` и `Y2` могут, будучи искажены шумом, приводить к очень похожим измерениям `B1~B2`. Поэтому встает вопрос, можно ли извлечь из измерений полезную информацию о сигнале, если наличие шума делает задачу неустойчивой? Такие задачи называются задачами, *поставленными некорректно*, и для их решения развиты специальные методы, основанные на привлечении дополнительной априорной информации о решении `Y`, которые будут рассмотрены ниже. Следует также отметить, что класс некорректных задач шире класса обратных (*один из примеров вы найдете в конце разд. 11.1.2*).

Одним из наиболее простых методов решения некорректных обратных задач является концепция поиска их *квазирешения*. Рассмотрим обратную задачу `A·Y=B`, где неизвестный вектор `Y` подлежит определению, а оператор (в линейном случае, матрица) `A` и вектор правых частей уравнений `B` известны. Подчеркнем, что задача может быть и нелинейной, т. е. оператор `A` может описывать сложную зависимость.

> ***Примечание***
>
> Проблемы, возникающие при попытке непосредственного обращения матрицы `A`, связаны с ее плохой обусловленностью. Подход к подобным задачам может быть не обязательно таким, как рассказывается ниже (*см. главу 8*).

Основная идея квазирешения состоит в параметризации неизвестного вектора `Y`, исходя из физических соображений постановки задачи. То есть на основе некоторой имеющейся априорной информации следует заранее задать модельный вид `Y~Y0`, зависящий от ряда параметров `r1,r2,…`. В результате пространство поиска решений значительно сужается — вместо отыскания

всех компонент вектора Y требуется лишь найти значения модельных параметров, решающих (в определенном смысле) задачу.

Квазирешение Y0 находится из решения задачи на минимум:

$$r=\arg\min\{|A\cdot Y0(r)-B|\}, \tag{6.4}$$

где минимизация проводится по вектору s параметров модельной зависимости Y0(r). Следует подчеркнуть, что задача поиска квазирешения является задачей на глобальный экстремум, что важно с позиций выбора вычислительного метода (как уже отмечалось, наиболее популярны градиентные методы поиска минимума в комбинации со сканированием для достижения глобальной минимизации — алгоритмами сплошного или случайного поиска).

Пример отыскания квазирешения обсуждавшейся в предыдущих разделах задачи приведен в листинге 6.15 и на рис. 6.7 (данные о результатах модельных измерений B и матрице A взяты из предыдущего листинга). Понятно, что при сведении некорректной задачи к проблеме отыскания квазирешения решающее значение принадлежит правильно выбранной параметризации неизвестного вектора Y.

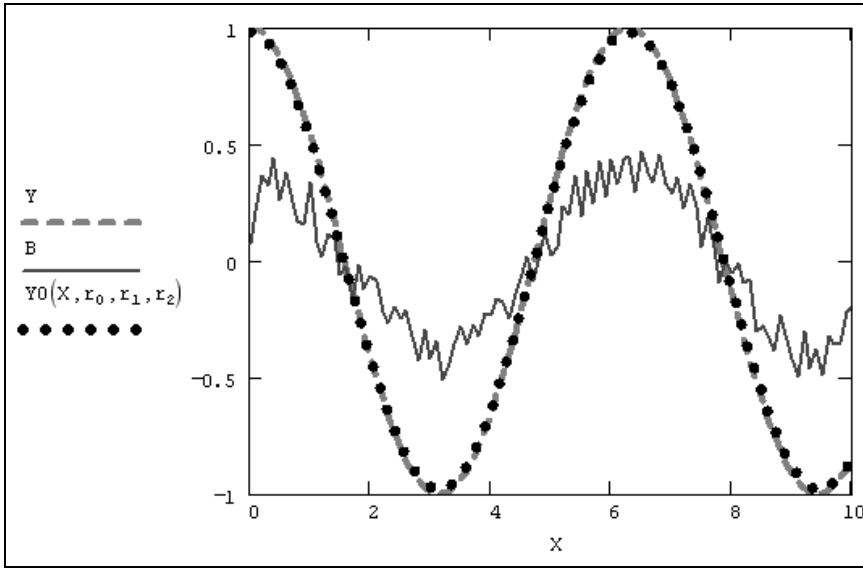



**Рис. 6.7.** Исходный сигнал Y, измерения B и квазирешение Y0
(продолжение листинга 6.15)

**Листинг 6.15. Квазирешение некорректной задачи (продолжение листингов 6.13 и 6.14)**

$$Y0(x, c, k, \phi) := c \cdot \sin(k \cdot x + \phi)$$

$$f(c, k, \phi) := \left| A \cdot Y0(X, c, k, \phi) - B \right|$$

$$c := 1 \qquad k := 0.5 \qquad \phi := 0$$

$$r := \text{Minimize}(f, c, k, \phi)$$

$$r = \begin{pmatrix} -0.981 \\ 0.996 \\ -1.556 \end{pmatrix}$$

# Глава 7

# Линейная алгебра

Задачи линейной алгебры, решаемые в Mathcad, можно условно разделить на два класса. Первый — это простейшие матричные операции, которые сводятся к определенным арифметическим действиям над элементами матрицы. Они реализованы в виде операторов (*см. разд. 7.1 и 7.2*) и нескольких специфических функций, предназначенных для создания, объединения, сортировки, получения основных свойств матриц и т. п. (*см. разд. 7.4*). Второй класс — это более сложные действия, которые реализуют алгоритмы вычислительной линейной алгебры, такие как вычисление определителей и обращение матриц (*см. разд. 7.3*), вычисление собственных векторов и собственных значений, решение систем линейных алгебраических уравнений и различные матричные разложения (*см. главу 8*).

## 7.1. Простейшие матричные операции

Простейшие операции матричной алгебры реализованы в Mathcad в виде операторов, причем их запись максимально приближена к математическому значению. Каждый оператор выражается соответствующим символом. Некоторые операции применимы только к квадратным матрицам `N×N`, некоторые допускаются только для векторов (например, скалярное произведение), а другие, несмотря на одинаковое написание, по-разному действуют на векторы и матрицы.

### 7.1.1. Транспонирование

*Транспонированием* называют операцию, переводящую матрицу размерности `M×N` в матрицу размерности `N×M`, делая столбцы исходной матрицы строками,

а строки — столбцами. Несколько примеров транспонирования приведены в листинге 7.1. Ввод символа транспонирования (transpose) осуществляется с помощью панели инструментов **Matrix** (Матрица) (рис. 7.1) или нажатием клавиш <Ctrl>+<1>. Не забывайте, что для вставки символа транспонирования матрица должна находиться между линиями ввода.

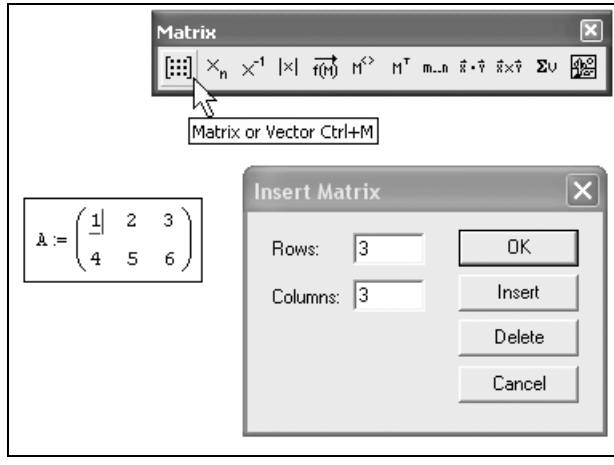



**Рис. 7.1.** Ввод матриц и основные операции над ними осуществляются при помощи панели **Matrix**

### *Примечание*

Все матричные и векторные операторы, о которых пойдет речь, допустимо использовать как в численных, так и в символьных расчетах. Мощь символьных операций заключается в возможности проводить их не только над конкретными числами, но и над переменными. Смело используйте символьный процессор в качестве мощного математического справочника, когда вы хотите вспомнить какое-либо определение из области линейной алгебры. Именно по этой причине мы прибегаем к примерам символьных расчетов в большинстве иллюстраций к данному разделу.

**Листинг 7.1. Транспонирование векторов и матриц**

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}^{T} = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$

$$(1.01 \quad 2.02 \quad 3.03 \quad 4.04)^{T} = \begin{pmatrix} 1.01 \\ 2.02 \\ 3.03 \\ 4.04 \end{pmatrix}$$

$$(x \quad y \quad z)^{T} \rightarrow \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

## 7.1.2. Сложение и вычитание

В Mathcad можно как складывать матрицы, так и вычитать их друг из друга. Для этих операторов применяются стандартные символы "+" или "–" соответственно. Матрицы должны иметь одинаковую размерность, иначе будет выдано сообщение об ошибке. Каждый элемент суммы двух матриц равен сумме соответствующих элементов матриц-слагаемых (листинг 7.2).

Результат унарной операции смены знака матрицы эквивалентен смене знака всех ее элементов. Для того чтобы изменить знак матрицы, достаточно ввести перед ней знак минуса, как перед обычным числом (нижняя строка листинга 7.2).

**Листинг 7.2. Сложение, вычитание и смена знака матриц**

$$\begin{pmatrix} a & b & c \\ d & f & g \end{pmatrix} + \begin{pmatrix} u & v & w \\ x & y & z \end{pmatrix} \rightarrow \begin{pmatrix} a+u & b+v & c+w \\ d+x & f+y & g+z \end{pmatrix}$$

$$\begin{pmatrix} a & b & c \\ d & f & g \end{pmatrix} - \begin{pmatrix} u & v & w \\ x & y & z \end{pmatrix} \rightarrow \begin{pmatrix} a-u & b-v & c-w \\ d-x & f-y & g-z \end{pmatrix}$$

$$-\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} = \begin{pmatrix} -1 & -2 & -3 \\ -4 & -5 & -6 \end{pmatrix}$$

Кроме сложения матриц Mathcad поддерживает операцию сложения матрицы со скаляром (листинг 7.3). Каждый элемент результирующей матрицы равен сумме соответствующего элемента исходной матрицы и скалярной величины.

**Листинг 7.3. Сложение матрицы со скалярной величиной**

$$\begin{pmatrix} a & b & c \\ d & f & g \end{pmatrix} + k \rightarrow \begin{pmatrix} a+k & b+k & c+k \\ d+k & f+k & g+k \end{pmatrix}$$

Иногда бывает нужно вычислить сумму всех элементов вектора или матрицы. Для этого существует вспомогательный оператор (листинг 7.4, первая и вторая строки соответственно), задаваемый кнопкой **Vector Sum** (Суммирование элементов вектора) на панели **Matrix** (Матрица) или сочетанием клавиш <Ctrl>+<4>. Этот оператор чаще оказывается полезным не в матричной алгебре, а при организации циклов с индексированными переменными.

В том же листинге 7.4 (снизу) показано применение операции суммирования диагональных элементов квадратной матрицы. Эту сумму называют *следом* (trace) матрицы. Данная операция организована в виде встроенной функции `tr`:

- `tr(A)` — след квадратной матрицы `A`.

**Листинг 7.4. Суммирование элементов и вычисление следа матрицы**

$$\sum \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \rightarrow a+b+c+d$$

$$\sum \begin{pmatrix} a & b \\ c & d \\ f & g \end{pmatrix} \rightarrow a+b+c+d+f+g$$

$$\mathrm{tr}\left(\begin{pmatrix} a & b & c \\ d & f & g \\ x & y & z \end{pmatrix}\right) \rightarrow a+f+z$$

## 7.1.3. Умножение

При умножении следует помнить, что матрицу размерности `M×N` допустимо умножать только на матрицу размерности `N×P` (`P` может быть любым). В результате получается матрица размерности `M×P`.

Чтобы ввести символ умножения, нужно нажать клавишу со звездочкой <*> или воспользоваться панелью инструментов **Matrix** (Матрица), нажав на ней кнопку **Dot Product** (Умножение). Умножение матриц обозначается по умолчанию точкой, как показано в листинге 7.5.

**Листинг 7.5. Перемножение матриц**

$$\begin{pmatrix} a & b & c \\ d & f & g \end{pmatrix} \cdot \begin{pmatrix} u & x \\ v & y \\ w & z \end{pmatrix} \rightarrow \begin{pmatrix} a \cdot u + b \cdot v + c \cdot w & a \cdot x + b \cdot y + c \cdot z \\ d \cdot u + f \cdot v + g \cdot w & d \cdot x + f \cdot y + g \cdot z \end{pmatrix}$$

$$\begin{pmatrix} a & b & c \\ d & f & g \end{pmatrix} \cdot \begin{pmatrix} u & v & w \\ x & y & z \end{pmatrix} \rightarrow$$

Обратите внимание (нижняя строка листинга 7.5), что попытка перемножить матрицы `A` и `B` несоответствующего (одинакового `2×3`) размера оказалась безрезультатной: после введенного знака равенства находится пустой местозаполнитель, а само выражение в редакторе Mathcad выделяется красным цветом. При установке курсора на это выражение появляется сообщение о несовпадении числа строк первой матрицы с числом столбцов второй матрицы.

Еще один пример, относящийся к умножению вектора на матрицу-строку и, наоборот, строки на вектор, приведен в листинге 7.6.

> ***Внимание!***
>
> Тот же самый оператор умножения действует на два вектора по-другому (*см. разд. 7.2.2*).

**Листинг 7.6. Умножение вектора и строки**

$$(a \quad b \quad c \quad d) \cdot \begin{pmatrix} s \\ x \\ y \\ z \end{pmatrix} \rightarrow a \cdot s + b \cdot x + c \cdot y + d \cdot z$$

$$\begin{pmatrix} s \\ x \\ y \\ z \end{pmatrix} \cdot (a \quad b \quad c \quad d) \rightarrow \begin{pmatrix} a \cdot s & s \cdot b & s \cdot c & s \cdot d \\ a \cdot x & b \cdot x & x \cdot c & d \cdot x \\ y \cdot a & b \cdot y & c \cdot y & y \cdot d \\ z \cdot a & z \cdot b & c \cdot z & d \cdot z \end{pmatrix}$$

Аналогично сложению матриц со скаляром определяется умножение и деление матрицы на скалярную величину (листинг 7.7). Символ умножения вводится так же, как и в случае умножения двух матриц. На скаляр можно умножать матрицу любой размерности.

**Листинг 7.7. Умножение матрицы на скалярную величину**

$$\begin{pmatrix} a & b & c \\ d & f & g \end{pmatrix} \cdot k \rightarrow \begin{pmatrix} a \cdot k & b \cdot k & c \cdot k \\ d \cdot k & f \cdot k & g \cdot k \end{pmatrix}$$

# 7.2. Векторная алгебра

Векторы являются частным случаем матриц размерности N×1, поэтому для них справедливы все те операции, что и для матриц, если ограничения особо не оговорены. Вместе с тем для векторов в линейной алгебре предусмотрен целый ряд специфических операций, и все они реализованы в системе Mathcad.

> ***Внимание!***
>
> Непосредственное проведение многих векторных операций над матрицами-строками, т. е. матрицами 1×N, невозможно; для того, чтобы превратить строку в вектор, ее нужно предварительно транспонировать.

## 7.2.1. Модуль вектора

*Модуль вектора* (vector magnitude) по определению равен квадратному корню из суммы квадратов его элементов (листинг 7.8).

> ***Внимание!***
>
> Не путайте модуль вектора и определитель матрицы, который обозначается тем же символом (*см. разд. 7.3.1*). Это характерный пример оператора, действующего по-разному на векторы и квадратные матрицы.

**Листинг 7.8. Модуль вектора**

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

$$\left\| \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \right\| = 3.742$$

## 7.2.2. Скалярное произведение

*Скалярное произведение векторов* (vector inner product) определяется как скаляр, равный сумме попарных произведений соответствующих элементов. Векторы должны иметь одинаковую размерность, скалярное произведение имеет ту же размерность. Скалярное произведение двух векторов `u` и `v` равно $u \cdot v = |u| \cdot |v| \cdot \cos\theta$, где $\theta$ — угол между векторами. Если векторы ортогональны, то их скалярное произведение равно нулю. Обозначается скалярное произведение тем же символом, что и умножение (листинг 7.9).

> ***Внимание!***
>
> Для обозначения скалярного произведения пользователю позволяется выбирать представление оператора умножения при помощи контекстного меню (подобно умножению скалярных величин). Однако никогда не применяйте для обозначения скалярного произведения символ $\times$, который является общеупотребительным символом векторного произведения (*см. разд. 7.2.3*).

**Листинг 7.9. Скалярное произведение векторов**

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow a \cdot x + b \cdot y + c \cdot z$$

С осторожностью перемножайте несколько (более двух) векторов. Поразному расставленные скобки полностью изменяют результат умножения. Примеры такого умножения приведены в листинге 7.10.

**Листинг 7.10. Скалярное произведение векторов, умноженное на третий вектор**

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix} \cdot \begin{pmatrix} 7 \\ 8 \\ 9 \end{pmatrix} = \begin{pmatrix} 224 \\ 256 \\ 288 \end{pmatrix}$$

$$\left[\begin{pmatrix}1\\2\\3\end{pmatrix}\cdot\begin{pmatrix}4\\5\\6\end{pmatrix}\right]\cdot\begin{pmatrix}7\\8\\9\end{pmatrix}=\begin{pmatrix}224\\256\\288\end{pmatrix}$$

$$\begin{pmatrix}1\\2\\3\end{pmatrix}\cdot\left[\begin{pmatrix}4\\5\\6\end{pmatrix}\cdot\begin{pmatrix}7\\8\\9\end{pmatrix}\right]=\begin{pmatrix}122\\244\\366\end{pmatrix}$$

## 7.2.3. Векторное произведение

*Векторное произведение* (cross product) двух векторов u и v с углом θ между ними равно вектору с модулем |u|·|v|·sinθ, направленным перпендикулярно плоскости векторов u и v. Обозначают векторное произведение символом "×", который можно ввести нажатием кнопки **Cross Product** (Векторное произведение) в панели **Matrix** (Матрица) или сочетанием клавиш <Ctrl>+<8>. Пример приведен в листинге 7.11.

**Листинг 7.11. Векторное произведение двух векторов**

$$\begin{pmatrix}a\\b\\c\end{pmatrix}\times\begin{pmatrix}x\\y\\z\end{pmatrix}\rightarrow\begin{pmatrix}b\cdot z-c\cdot y\\c\cdot x-a\cdot z\\a\cdot y-b\cdot x\end{pmatrix}$$

## 7.2.4. Векторизация массива

Векторная алгебра Mathcad включает несколько необычный оператор, который называется *оператором векторизации* (vectorize operator). Этот оператор предназначен, как правило, для работы с массивами. Он позволяет провести однотипную операцию над всеми элементами массива (т. е. матрицы или вектора), упрощая тем самым программирование циклов. Например, иногда требуется умножить каждый элемент одного вектора на соответствующий элемент другого вектора, чтобы в результате также получился вектор. Непосредственно такой операции в Mathcad нет, но ее легко осуществить с помощью векторизации (листинг 7.12). Для этого:

1. Введите векторное выражение, как показано во второй строчке листинга 7.12 (обратите внимание, что в таком виде символ умножения обозначает оператор скалярного произведения векторов).

2. Переместите курсор таким образом, чтобы линии ввода выделяли все выражение, которое требуется подвергнуть векторизации.
3. Введите оператор векторизации, нажав кнопку **Vectorize** (Векторизация) на панели **Matrix** (Матрица) (листинг 7.12) или сочетанием клавиш <Ctrl>+<->.
4. Введите <=>, чтобы получить результат.

**Листинг 7.12. Использование оператора векторизации для перемножения элементов вектора**

$$v := \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$\left[\overrightarrow{\left[v \cdot \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}\right]}\right] = \begin{pmatrix} 1 \\ 4 \\ 9 \end{pmatrix} \qquad v \cdot \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = 14$$

Большинство неспецифических функций Mathcad не требуют векторизации для проведения одной и той же операции над всеми элементами вектора. Например, аргументом тригонометрических функций по определению является скаляр. Если попытаться вычислить экспоненту от векторной величины, Mathcad осуществит векторизацию по умолчанию, возведя в степень `e` каждый элемент и выдав в качестве результата соответствующий вектор (листинг 7.13).

**Листинг 7.13. Векторизация аргумента необязательна для большинства встроенных функций Mathcad**

$$v := \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$\exp(v) = \begin{pmatrix} 2.718 \\ 7.389 \\ 20.086 \end{pmatrix} \qquad \exp\left(\overrightarrow{v}\right) = \begin{pmatrix} 2.718 \\ 7.389 \\ 20.086 \end{pmatrix}$$

# 7.3. Вычисление определителей и обращение квадратных матриц

Рассмотрим еще несколько исключительно важных действий линейной алгебры, связанных с понятием определителя матрицы. Несмотря на то, что некоторые из них реализованы в Mathcad также в виде операторов, они требуют (при проведении расчетов по численным алгоритмам) несравненно больше внимания, нежели операторы упомянутые в двух предыдущих разделах.

## 7.3.1. Определитель квадратной матрицы

*Определитель* (Determinant) матрицы обозначается стандартным математическим символом. Чтобы ввести оператор нахождения определителя матрицы, можно нажать кнопку **Determinant** (Определитель) на панели инструментов **Matrix** (Матрица) (листинг 7.14) или набрать на клавиатуре <|> (нажав клавиши <Shift>+<\>). В результате любого из этих действий появляется местозаполнитель, в который следует поместить матрицу. Чтобы вычислить определитель уже введенной матрицы:

1. Переместите курсор в документе таким образом, чтобы поместить матрицу между линиями ввода (напоминаем, что линии ввода — это вертикальный и горизонтальный отрезки синего цвета, образующие уголок, указывающий на текущую область редактирования).
2. Введите оператор нахождения определителя матрицы.
3. Введите знак равенства (либо символьного вывода), чтобы вычислить определитель (численно или аналитически соответственно, как это показано в листинге 7.14).

> **Внимание!**
>
> Не путайте операторы вычисления определителя квадратной матрицы и длины вектора. В Mathcad 12 введен принудительный контроль действий пользователя при вводе этих операторов во избежание путаницы (т. к. один и тот же символ используется для этих двух операций). При попытке вычислить определитель матрицы с помощью оператора $|A|$, введенного с панели **Calculator** (Калькулятор), а не **Matrix** (Матрица), будет выдано сообщение об ошибке, а результат вычисления детерминанта появится только после того, как пользователь вызовет контекстное меню и подтвердит в нем, что он собирается вычислить именно определитель матрицы. То же самое касается и длины вектора, если попытаться ввести его не с панели **Calculator** (Калькулятор), а с панели **Matrix** (Матрица).

**Листинг 7.14. Вычисление определителя квадратной матрицы**

$$\left|\begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 0 \\ 5 & 4 & 3.00001 \end{pmatrix}\right| = 16$$

$$\left|\begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 0 \\ 5 & 4 & 3.00001 \end{pmatrix}\right| \rightarrow 16.00002$$

## 7.3.2. Ранг матрицы

*Рангом* (rank) матрицы называют наибольшее натуральное число `k`, для которого существует не равный нулю определитель `k`-го порядка подматрицы, составленной из любого пересечения `k` столбцов и `k` строк матрицы.

Для вычисления ранга в Mathcad предназначена функция `rank` (листинг 7.15).

❒ `rank(A)` — ранг матрицы:

- `A` — матрица.

**Листинг 7.15. Вычисление ранга матрицы**

$$\left|\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}\right| = 0 \qquad \mathrm{rank}\left(\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}\right) = 2$$

$$\left|\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{pmatrix}\right| = 27 \qquad \mathrm{rank}\left(\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{pmatrix}\right) = 3$$

## 7.3.3. Обращение квадратной матрицы

Поиск *обратной матрицы* возможен, если матрица квадратная и ее определитель не равен нулю. Произведение исходной матрицы на обратную по определению является единичной матрицей. Для ввода оператора поиска обратной матрицы нажмите кнопку **Inverse** (Обратная матрица) на панели инструментов **Matrix** (Матрица). В листинге 7.16 приведен пример поиска обратной матрицы и последующая проверка правильности ее вычисления.

**Листинг 7.16. Вычисление обратной матрицы**

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.25 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.25 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.25 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.25 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

## 7.3.4. Возведение квадратной матрицы в степень

К квадратным матрицам можно формально применять операцию возведения в степень `n`. Для этого `n` должно быть целым числом. Результат данной операции приведен в табл. 7.1. Ввести оператор возведения матрицы `M` в степень `n` можно точно так же, как и для скалярной величины: нажав кнопку **Raise to Power** (Возвести в степень) на панели **Calculator** (Калькулятор) или нажав клавишу <^>. После появления местозаполнителя в него следует ввести значение степени `n`.

***Таблица 7.1.*** *Правила возведения матрицы в степень*

| **n** | **M^n** |
|---|---|
| 0 | Единичная матрица размерности матрицы `M` |
| 1 | Сама матрица `M` |
| –1 | $M^{-1}$ — матрица, обратная `M` |
| 2, 3, … | $M \cdot M$, $(M \cdot M) \cdot M$, … |
| –2, –3, … | $M^{-1} \cdot M^{-1}$, $(M^{-1} \cdot M^{-1}) \cdot M^{-1}$, … |

Примеры возведения матрицы в степень приведены в листинге 7.17.

**Листинг 7.17. Возведение квадратной матрицы в целую степень**

$$\begin{pmatrix}1&0&0\\0&2&0\\0&0&4\end{pmatrix}^{-1}=\begin{pmatrix}1&0&0\\0&0.5&0\\0&0&0.25\end{pmatrix}$$

$$\begin{pmatrix}1&0&0\\0&2&0\\0&0&4\end{pmatrix}^{-2}=\begin{pmatrix}1&0&0\\0&0.25&0\\0&0&0.063\end{pmatrix}$$

$$\begin{pmatrix}1&0&0\\0&2&0\\0&0&4\end{pmatrix}^{0}=\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}$$

$$\begin{pmatrix}1&0&0\\0&2&0\\0&0&4\end{pmatrix}^{1}=\begin{pmatrix}1&0&0\\0&2&0\\0&0&4\end{pmatrix}$$

$$\begin{pmatrix}1&0&0\\0&2&0\\0&0&4\end{pmatrix}^{2}=\begin{pmatrix}1&0&0\\0&4&0\\0&0&16\end{pmatrix}$$

## 7.3.5. Матричные нормы

В линейной алгебре используются различные векторные и матричные *нормы* (norm), которые ставят в соответствие матрице некоторую скалярную числовую характеристику. Норма матрицы отражает порядок величины матричных элементов. В разных специфических задачах линейной алгебры применяются различные виды норм. Mathcad имеет четыре встроенных функции для расчета разных норм квадратных матриц:

- `norm1(A)` — норма в пространстве L1;
- `norm2(A)` — норма в пространстве L2;
- `norme(A)` — евклидова норма (euclidean norm);

- `normi(A)` — max-норма, или ∞-норма (infinity norm):
  - `A` — квадратная матрица.

Примеры расчета различных норм двух матриц `A` и `B` с различающимися на два порядка элементами иллюстрирует листинг 7.18.

> ***Совет***
>
> В большинстве задач неважно, какую норму использовать. Как видно, в обычных случаях разные нормы дают примерно одинаковые значения, хорошо отражая порядок величины матричных элементов. Конкретные формулы, определяющие нормы, заинтересованный читатель отыщет в справочниках по линейной алгебре или в ресурсах Mathcad.

**Листинг 7.18. Вычисление различных норм матриц**

$$A := \begin{pmatrix} 1 & 2 & 3 \\ -4 & 5 & -6 \\ -7 & 8 & 9 \end{pmatrix} \qquad B := \begin{pmatrix} 100 & 200 & 300 \\ -400 & 500 & -600 \\ -700 & 800 & 900 \end{pmatrix}$$

$\text{norm1}(A) = 18$ $\qquad \text{norm1}(B) = 1.8 \times 10^3$

$\text{norm2}(A) = 14.213$ $\qquad \text{norm2}(B) = 1.421 \times 10^3$

$\text{normi}(A) = 24$ $\qquad \text{normi}(B) = 2.4 \times 10^3$

$\text{norme}(A) = 16.882$ $\qquad \text{norme}(B) = 1.688 \times 10^3$

$\max(A) = 9$ $\qquad \max(B) = 900$

## 7.3.6. Число обусловленности квадратной матрицы

Еще одной важной характеристикой матрицы является ее *число обусловленности* (condition number). Число обусловленности является мерой чувствительности системы линейных уравнений `A·x=b`, определяемой матрицей `A`, к погрешностям задания вектора `b` правых частей уравнений (*см. главу 8*). Чем больше число обусловленности, тем сильнее это воздействие и тем более неустойчив процесс нахождения решения линейной системы. Число обусловленности связано с нормой матрицы и вычисляется по-разному для каждой из норм:

- `cond1(A)` — число обусловленности в норме L1;
- `cond2(A)` — число обусловленности в норме L2;
- `conde(A)` — число обусловленности в евклидовой норме;

❒ `condi(A)` — число обусловленности в ∞-норме:

- `A` — квадратная матрица.

Расчет чисел обусловленности для двух матриц `A` и `B` показан в листинге 7.19. Обратите внимание, что первая из матриц является *хорошо обусловленной*, а вторая — *плохо обусловленной* (два ее последних столбца очень близки между собой, с точностью до множителя `2`). Вторая строка листинга дает формальное определение числа обусловленности как произведения норм исходной и обратной матриц. В других нормах определение точно такое же.

> ***Примечание***
>
> Как нетрудно понять, матрицы `A` и `B` из листинга 7.18 (*см. предыдущий разд.*) обладают одинаковыми числами обусловленности, т. к. `B=100·A`, и, следовательно, обе матрицы определяют одну и ту же систему уравнений.

**Листинг 7.19. Вычисление чисел обусловленности матриц (в различных нормах)**

$$A := \begin{pmatrix} 1 & 2 & 3 \\ -4 & 5 & -6 \\ -7 & 8 & 9 \end{pmatrix} \qquad B := \begin{pmatrix} 1 & 2 & 4 \\ -4 & 5 & 10 \\ -7 & 8 & 16.1 \end{pmatrix}$$

$$cond1(A) = 12.14 \qquad cond1(B) = 1.535 \times 10^3$$

$$cond2(A) = 6.951 \qquad cond2(B) = 1.017 \times 10^3$$

$$condi(A) = 11.721 \qquad condi(B) = 1.811 \times 10^3$$

$$conde(A) = 8.556 \qquad conde(B) = 1.024 \times 10^3$$

# 7.4. Вспомогательные матричные функции

Перечислим основные встроенные функции, предназначенные для облегчения работы с векторами и матрицами. Они нужны для создания матриц, слияния и выделения части матриц, получения основных свойств матриц и т. п.

## 7.4.1. Автоматическая генерация матриц

Самым наглядным способом создания матрицы или вектора является применение первой кнопки панели инструментов **Matrix** (Матрицы). Однако в

большинстве случаев, в частности, при программировании сложных проектов, удобнее бывает создавать массивы с помощью встроенных функций.

## Создание матриц на основе некоторой функции

Наиболее удобный прием автоматизации создания матриц заключается в предварительном определении функции `f(i,j)`, аргументом которой должны быть индексы элементов матрицы:

- `matrix(M,N,f)` — создание матрицы размера `M×N`, каждый `i,j` элемент которой есть `f(i,j)` (листинг 7.20):
  - `M` — количество строк матрицы;
  - `N` — количество столбцов матрицы;
  - `f(i,j)` — функция.

**Листинг 7.20. Создание матрицы на основе функции пользователя**

$$f(i,j) := i^2 + j^2$$

$$matrix(3,4,f) = \begin{pmatrix} 0 & 1 & 4 & 9 \\ 1 & 2 & 5 & 10 \\ 4 & 5 & 8 & 13 \end{pmatrix}$$

## Создание матриц для построения 3D-графиков

Для создания матриц имеются еще две специфические функции, применяемые, в основном, для быстрого и эффектного представления каких-либо зависимостей в виде трехмерных графиков (типа поверхности или пространственной кривой). Все их аргументы, кроме первого (имени функции), необязательны. Рассмотрим первую из этих встроенных функций.

- `CreateSpace(F(`или `f1,f2,f3),t0,t1,tgrid,fmap)` — создание вложенного массива, представляющего `x`-, `y`- и `z`-координаты параметрической пространственной кривой, заданной функцией `F`:
  - `F(t)` — векторная функция из трех элементов, заданная параметрически относительно единственного аргумента `t`;
  - `f1(t),f2(t),f3(t)` — скалярные функции;
  - `t0` — нижний предел `t` (по умолчанию `-5`);
  - `t1` — верхний предел `t` (по умолчанию `5`);

- `tgrid` — число точек сетки по переменной `t` (по умолчанию `20`);
- `fmap` — векторная функция от трех аргументов, задающая преобразование координат.

Пример использования функции `CreateSpace` показан на рис. 7.2. Заметьте, для построения графика кривой не потребовалось никакого дополнительного кода, кроме определения параметрической зависимости в вектор-функции `F`!

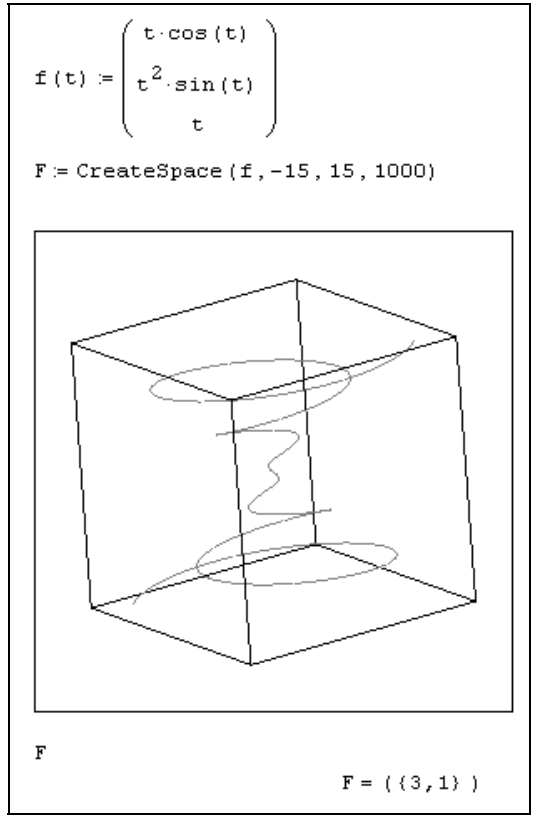



**Рис. 7.2.** Использование функции `CreateSpace` для построения графика трехмерной кривой

Функция создания матрицы для графика трехмерной поверхности устроена совершенно аналогично, за тем исключением, что для определения поверхности требуется не одна, а две переменных.

Пример ее использования иллюстрирует рис. 7.3.

- `CreateMesh(F(`или `g,` или `f1,f2,f3),s0,s1,t0,t1,sgrid,tgrid,fmap)` — создание вложенного массива, представляющего `x`-, `y`- и `z`-координаты параметрической поверхности, заданной функцией `F`:
  - `F(s,t)` — векторная функция из трех элементов, заданная параметрически относительно двух аргументов `s` и `t`;
  - `g(s,t)` — скалярная функция;
  - `f1(s,t),f2(s,t),f3(s,t)` — скалярные функции;
  - `s0,t0` — нижние пределы аргументов `s,t` (по умолчанию `-5`);
  - `s1,t1` — верхние пределы аргументов `s,t` (по умолчанию `5`);

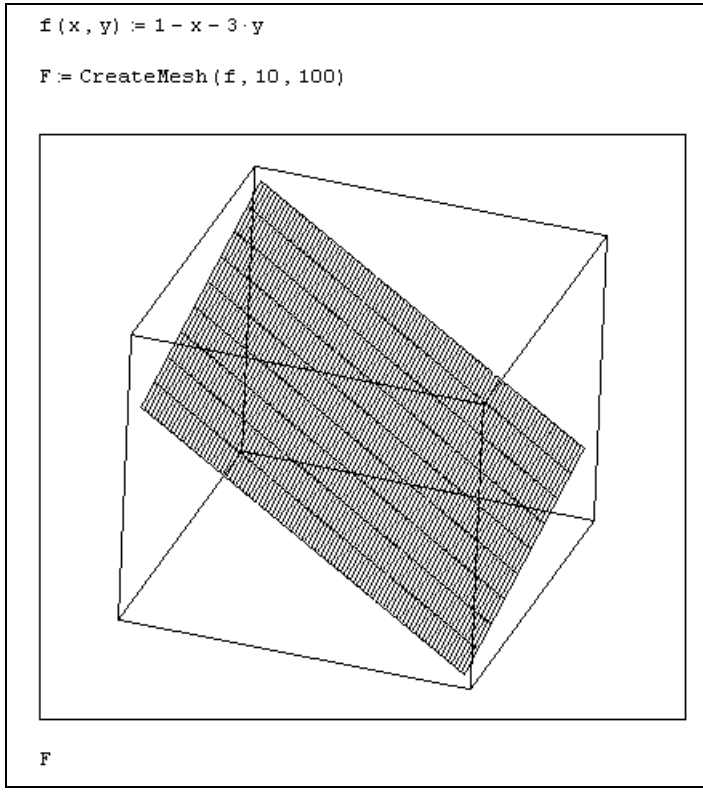



**Рис. 7.3.** Использование функции `CreateMesh` для построения графика трехмерной поверхности

- sgrid, tgrid — число точек сетки по переменным s и t (по умолчанию 20);
- fmap — векторная функция из трех элементов от трех аргументов, задающая преобразование координат.

Результатом обеих рассмотренных функций CreateMesh и CreateSpace является соответствующий вложенный массив, служащий в Mathcad для представления тензора. Каждая матрица из числа трех вложенных матриц, образующих вложенный массив данных, определяет x-, y- и z-координаты точек поверхности или кривой.

## Создание диагональных матриц

В Mathcad легко создать матрицы, имеющие определенное простое строение, с помощью одной из встроенных функций. Примеры использования этих функций приведены в листинге 7.21:

- identity(N) — единичная матрица размера N×N;
- diag(v) — диагональная матрица, на диагонали которой находятся элементы вектора v:
  - N — целое число;
  - v — вектор.

**Листинг 7.21. Создание единичной и диагональной матрицы заданной размерности**

$$\text{identity}(3) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$v := \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \qquad \text{diag}(v) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix}$$

# 7.4.2. Разбиение и слияние матриц

Из матрицы или вектора можно выделить либо подматрицу, либо вектор-столбец, либо отдельный элемент. И обратно, можно "склеить" несколько матриц в одну.

## Выделение подматрицы

Часть матрицы выделяется одним из следующих способов (листинг 7.22):

- для выделения одного элемента предназначен оператор нижнего индекса. Оператор вводится нажатием кнопки **Subscript** (Нижний индекс) со значком $x_n$ на панели **Matrix** (Матрица), либо нажатием клавиши <[> (вторая строка листинга 7.22);
- для выделения из матрицы столбца примените оператор выделения столбца нажатием кнопки **Matrix Column** с изображением угловых скобок <> на панели **Matrix**, либо сочетанием клавиш <Ctrl>+<6> (третья строка листинга 7.22). Этот оператор называют еще, по аналогии с предыдущим, оператором *верхнего индекса*;
- чтобы выделить из матрицы строку, примените тот же оператор <> к транспонированной матрице (конец листинга 7.22);
- для выделения подматрицы используйте встроенную функцию `submatrix(A,ir,jr,ic,jc)`, возвращающую часть матрицы `A`, находящуюся между строками `ir,jr` и столбцами `ic,jc` включительно (листинг 7.23).

**Листинг 7.22. Доступ к отдельным элементам, столбцам и строкам матрицы**

$$A := \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

$A_{1,0} = 5$ $A_{2,1} = 10$

$$A^{\langle 0 \rangle} = \begin{pmatrix} 1 \\ 5 \\ 9 \end{pmatrix} \quad A^{\langle 3 \rangle} = \begin{pmatrix} 4 \\ 8 \\ 12 \end{pmatrix}$$

$$\left(A^T\right)^{\langle 0 \rangle^T} = (1 \quad 2 \quad 3 \quad 4)$$

$$\left(A^T\right)^{\langle 1 \rangle^T} = (5 \quad 6 \quad 7 \quad 8)$$

### *Примечание 1*

Выделить из матрицы один столбец или строку можно и с помощью функции `submatrix` (листинг 7.23, нижняя строка).

***Примечание 2***

Те же операции применимы к матрицам-векторам и матрицам-строкам. Следует помнить только, что размер их составляет `N×1` и `1×N` соответственно.

**Листинг 7.23. Выделение подматрицы**

$$A := \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

$$\text{submatrix}(A, 0, 1, 0, 2) = \begin{pmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \end{pmatrix}$$

$$\text{submatrix}(A, 0, 0, 0, 2) = (1 \quad 2 \quad 3)$$

## Слияние матриц

Для того чтобы составить из двух или более матриц одну, в Mathcad предусмотрена пара матричных функций (листинг 7.24):

- `augment(A,B,C,…)` — матрица, сформированная слиянием матриц-аргументов слева направо;
- `stack(A,B,C,…)` — матрица, сформированная слиянием матриц-аргументов сверху вниз:
  - `A,B,C,…` — векторы или матрицы соответствующего размера.

**Листинг 7.24. Примеры слияния матриц**

$$A := \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix} \qquad B := \begin{pmatrix} 4 & 4 \\ 4 & 4 \end{pmatrix}$$

$$\text{augment}(A, B) = \begin{pmatrix} 2 & 2 & 4 & 4 \\ 2 & 2 & 4 & 4 \end{pmatrix}$$

$$\text{stack}(A, B) = \begin{pmatrix} 2 & 2 \\ 2 & 2 \\ 4 & 4 \\ 4 & 4 \end{pmatrix}$$

$$\text{augment}(A, B, A, B) = \begin{pmatrix} 2 & 2 & 4 & 4 & 2 & 2 & 4 & 4 \\ 2 & 2 & 4 & 4 & 2 & 2 & 4 & 4 \end{pmatrix}$$

## Специфические преобразования матриц

Еще две встроенных функции Mathcad позволяют создавать матрицы на основе некоторой имеющейся матрицы (листинг 7.25):

- `geninv(A)` — создание матрицы, обратной (слева) прямоугольной матрице `A`;
- `rref(A)` — преобразование матрицы или вектора `A` в ступенчатый вид:
  - `A` — матрица, составленная из действительных чисел.

***Примечание***

Размер `N×M` матрицы `A` для функции `geninv` должен быть таким, чтобы `N≥M`.

**Листинг 7.25. Создание матриц на основе другой матрицы**

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

$$\text{geninv}(A) = \begin{pmatrix} -0.55 & -0.225 & 0.1 & 0.425 \\ 0.25 & 0.125 & 0 & -0.125 \end{pmatrix}$$

$$\text{geninv}(A) \cdot A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\text{rref}(A) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \qquad \text{rref}(\text{geninv}(A)) = \begin{pmatrix} 1 & 0 & -1 & -2 \\ 0 & 1 & 2 & 3 \end{pmatrix}$$

## 7.4.3. Сортировка элементов матриц

Часто бывает нужно переставить элементы матрицы или вектора, расположив их в определенной строке или столбце в порядке возрастания или убывания. Для этого имеются несколько встроенных функций, которые позволяют гибко управлять сортировкой матриц:

- `sort(v)` — сортировка элементов вектора в порядке возрастания (листинг 7.26, верхняя строка);

- `reverse(v)` — перестановка элементов вектора в обратном порядке (листинг 7.26, нижняя строка);
- `csort(A,i)` — сортировка строк матрицы выстраиванием элементов `i`-го столбца в порядке возрастания (листинг 7.27, верхняя строка);
- `rsort(A,i)` — сортировка столбцов матрицы выстраиванием элементов `i`-ой строки в порядке возрастания (листинг 7.27, нижняя строка):
  - `v` — вектор;
  - `A` — матрица;
  - `i` — индекс строки или столбца.

***Примечание***

Если элементы матриц или векторов комплексные, то сортировка ведется по действительной части, а мнимая часть игнорируется.

**Листинг 7.26. Сортировка вектора**

$$v := \begin{pmatrix} 3 \\ 1 \\ 4 \\ 2 \end{pmatrix} \qquad \mathrm{sort}(v) = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

$$\mathrm{reverse}(v) = \begin{pmatrix} 2 \\ 4 \\ 1 \\ 3 \end{pmatrix} \qquad \mathrm{reverse}(\mathrm{sort}(v)) = \begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \end{pmatrix}$$

**Листинг 7.27. Сортировка матриц по строке и столбцу**

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

$$\text{csort}(A,0)=\begin{pmatrix}1&7\\2&8\\3&6\\4&5\end{pmatrix}\qquad \text{csort}(A,1)=\begin{pmatrix}4&5\\3&6\\1&7\\2&8\end{pmatrix}$$

$$\text{rsort}(A,0)=\begin{pmatrix}1&7\\3&6\\4&5\\2&8\end{pmatrix}\qquad \text{rsort}(A,1)=\begin{pmatrix}1&7\\3&6\\4&5\\2&8\end{pmatrix}$$

## 7.4.4. Вывод размера матрицы

Для получения сведений о характеристиках матриц или векторов предусмотрены следующие встроенные функции (листинги 7.28 и 7.29 соответственно):

- `rows(A)` — число строк;
- `cols(A)` — число столбцов;
- `length(v)` — число элементов вектора;
- `last(v)` — индекс последнего элемента вектора:
  - `A` — матрица или вектор;
  - `v` — вектор.

***Примечание***

Если матричные индексы нумеруются с `1`, т. е. системная константа `ORIGIN` равна не `0` (по умолчанию), а `1`, то число элементов вектора и индекс его последнего элемента совпадают.

**Листинг 7.28. Размер матриц**

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

$\text{rows}(A)=4$ $\qquad$ $\text{cols}(A)=2$

**Листинг 7.29. Размер векторов**

$$v := \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \qquad w := (1 \quad 2 \quad 3 \quad 4)$$

last(v) = 3 last($w^T$) = 3

length(v) = 4 length($w^T$) = 4

rows(v) = 4 rows(w) = 1

cols(v) = 1 cols(w) = 4

# Глава 8

# Системы линейных уравнений

Одной из центральных проблем вычислительной линейной алгебры является решение систем линейных уравнений (*см. разд. 8.1 и 8.2*), отыскание собственных векторов и собственных значений (*см. разд. 8.4*), а также различные матричные разложения (*см. разд. 8.3*). Все они будут рассмотрены в данной главе, являющейся, фактически, продолжением предыдущей (которая была посвящена простейшим матричным вычислениям).

> ***Примечание***
>
> К системам линейных уравнений сводится множество, если не сказать большинство, задач вычислительной математики. Один из таких примеров подробно рассмотрен в *разд. 10.4*.

Задачу решения *систем линейных алгебраических уравнений* (СЛАУ), т. е. систем `N` уравнений вида

$$a_{i1}\cdot x_1+a_{i2}\cdot x_2+\ldots+a_{iN}\cdot x_N=b_i \qquad (8.1)$$

можно записать в эквивалентной матричной форме:

$$A\cdot x=b, \qquad (8.2)$$

где `A` — матрица коэффициентов СЛАУ размерности `N×N`; `x` — вектор неизвестных; `b` — вектор правых частей уравнений. Из общего курса линейной алгебры известно, что такая СЛАУ имеет единственное решение, если матрица `A` является *невырожденной* или, по-другому, *несингулярной*, т. е. ее определитель не равен нулю. Самый простой способ решения почти всякой несингулярной системы — использование алгоритма Гаусса, реализованного во встроенной функции `lsolve` (*см. разд. 8.1.2*).

> ***Примечание***
>
> На практике, в основном при решении *обратных задач* (inverse problems), часто приходится сталкиваться не только со СЛАУ с квадратной матрицей `A`,

но и с прямоугольной матрицей размера M×N, т. е. системами, в которых число неизвестных не равно числу уравнений (как больше, так и меньше него). Такие системы требуют специфического подхода, который будет рассмотрен в *разд. 8.2 и 8.3*.

# 8.1. Хорошо обусловленные системы с квадратной матрицей

С вычислительной точки зрения, решение СЛАУ с квадратной матрицей A не представляет трудностей, если размерность A не очень велика. С большой матрицей проблем также не возникает, если она не очень плохо обусловлена (конечно, надо учитывать, что объем вычислений возрастает с увеличением размерности матрицы). В данном разделе будут рассмотрены именно такие системы, решение которых реализовано в Mathcad в двух вариантах:

- вычислительный блок Given/Find (приближенный итерационный алгоритм);
- встроенная функция lsolve (точный алгоритм Гаусса).

При этом СЛАУ можно решать как в более наглядной форме (8.1), так и в более компактной матричной форме (8.2). Для первого способа следует использовать блок Given/Find (он может также применяться в случае систем уравнений и неравенств, а также при решении переопределенных СЛАУ), а для второго — как вычислительный блок, так и встроенную функцию lsolve.

## 8.1.1. Вычислительный блок *Given* / *Find*

Для того чтобы численным методом решить СЛАУ при помощи вычислительного блока (он был подробно описан в *главе 5*), следует после ключевого слова Given выписать ее, пользуясь логическими операторами. Рассмотрим в качестве примера систему трех уравнений, приведенную в листинге 8.1 (после ключевого слова Given). Неизвестными являются три компонента вектора x, поэтому именно эта векторная переменная является аргументом встроенной функции Find(x), решающей систему в последней строке листинга. Очень важно, что при использовании вычислительного блока Given/Find всем неизвестным требуется присвоить начальные значения, как это сделано в первой строке листинга 8.1.

***Примечание***

Если матрица СЛАУ является невырожденной (точнее, если ее число обусловленности не слишком велико), то известно, что численное решение системы

уравнений единственно. Поэтому начальные значения могут быть произвольными, т. к. результат работы численного метода все равно сойдется к точному решению.

**Листинг 8.1. Решение СЛАУ с помощью вычислительного блока**

$x_0 := 0 \quad x_1 := 0 \quad x_2 := 0$

Given

$1 \cdot x_0 + 2 \cdot x_1 - 3 \cdot x_2 = 10$

$4 \cdot x_0 + 5 \cdot x_1 + 6 \cdot x_2 = 20$

$7 \cdot x_0 - 8 \cdot x_1 - 9 \cdot x_2 = 30$

$$\text{Find}(x) = \begin{pmatrix} 4.667 \\ 1.333 \\ -0.889 \end{pmatrix}$$

Листинг 8.1 демонстрирует запись каждого уравнения системы (в промежутке между `Given` и `Find`), что очень неудобно, когда система содержит большое число уравнений. В последнем случае намного лучше применить матричную запись СЛАУ, как это показано в листинге 8.2. Первая строка листинга представляет собой определение матрицы СЛАУ `A` и вектора правых частей `b`, а в остальном работа блока `Given/Find` полностью идентична предыдущему листингу.

**Листинг 8.2. Решение СЛАУ, записанной в матричной форме**

$$A := \begin{pmatrix} 1 & 2 & -3 \\ 4 & 5 & 6 \\ 7 & -8 & -9 \end{pmatrix} \quad b := \begin{pmatrix} 10 \\ 20 \\ 30 \end{pmatrix}$$

$$x := \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Given

$A \cdot x = b$

$$\text{Find}(x) = \begin{pmatrix} 4.667 \\ 1.333 \\ -0.889 \end{pmatrix}$$

Проверка правильности решения СЛАУ прямой подстановкой, причем в матричной форме, приведена в листинге 8.3. Обратите внимание на матрицу в первой строке листинга, представляющую рассматриваемую систему уравнений. Во второй строке листинга 8.3 производится вычисление нормы невязки, характеризующей точность полученного решения СЛАУ.

***Примечание***

Такая большая невязка может вызвать совершенно обоснованное удивление читателя. На самом деле точность решения гораздо выше (в данном примере $\sim 10^{-15}$), а полученное значение невязки $\sim 10^{-3}$ объясняется соответствующим представлением вещественных чисел результата на экране (по умолчанию с точностью до 3-го знака).

**Листинг 8.3. Проверка правильности решения СЛАУ**

$$\begin{pmatrix} 1 & 2 & -3 \\ 4 & 5 & 6 \\ 7 & -8 & -9 \end{pmatrix} \cdot \begin{pmatrix} 4.667 \\ 1.333 \\ -0.889 \end{pmatrix} = \begin{pmatrix} 10 \\ 19.999 \\ 30.006 \end{pmatrix}$$

$$\left| \begin{pmatrix} 10 \\ 19.999 \\ 30.006 \end{pmatrix} - \begin{pmatrix} 10 \\ 20 \\ 30 \end{pmatrix} \right| = 6.083 \times 10^{-3}$$

## 8.1.2. Функция *lsolve*

Альтернативой способу решения СЛАУ, введенному в предыдущем разделе, является применение встроенной функции `lsolve`. Для этого система уравнений должна быть записана в матричной форме `A·x=b`:

- `lsolve(A,b)` — вектор решения системы линейных алгебраических уравнений:
  - `A` — матрица коэффициентов системы;
  - `b` — вектор правых частей.

***Примечание***

Начиная с версии Mathcad 13, встроенная функция `lsolve` может использоваться не только для решения линейных систем с квадратной матрицей, но

также и систем с прямоугольной матрицей, т. е. как недоопределенных, так и переопределенных систем уравнений (*см. разд. 8.2*).

### Примечание

В функции `lsolve` запрограммирован численный метод LU-разложения (*см. разд. 8.3.3*), основанный на *алгоритме последовательных исключений Гаусса*. Он состоит в преобразовании матрицы `A` линейной системы к треугольному виду, т. е. к форме, когда все элементы ниже главной диагонали матрицы являются нулевыми (*см. разд. 8.3.1*). Точнее, исходная СЛАУ `Ax=b` заменяется эквивалентной системой с другой матрицей $A^*$ и другим вектором правых частей $b^*$, но имеющей то же решение, что и исходная система. Очень важно заметить, что результат, выдаваемый методом Гаусса, является *точным* (конечно, с поправкой на неизбежно присутствующие ошибки численного округления, которые, в случае хорошо обусловленной матрицы `A`, являются ничтожными). Таким образом, в противоположность применению вычислительного блока `Given/Find` (в основе которого лежит приближенный итерационный алгоритм), функция `lsolve` не нуждается в присвоении начальных значений вектору `x`.

Применение функции `lsolve` показано в листинге 8.4. При этом матрица `A` может быть определена любым из способов, необязательно явно, как во всех примерах этого раздела. В последней строке листинга осуществляется вычисление нормы невязки, которая оказывается равной нулю. Заметим, что встроенную функцию `lsolve` допускается применять и при символьном решении СЛАУ (листинг 8.5). В последнем случае в уравнениях допускается использовать параметры (т. е. имена переменных, которым не присвоены никакие значения).

**Листинг 8.4. Численное решение СЛАУ**

$$A := \begin{pmatrix} 1 & 2 & -3 \\ 4 & 5 & 6 \\ 7 & -8 & -9 \end{pmatrix} \qquad b := \begin{pmatrix} 10 \\ 20 \\ 30 \end{pmatrix}$$

$$\text{lsolve}(A, b) = \begin{pmatrix} 4.667 \\ 1.333 \\ -0.889 \end{pmatrix}$$

$$\left| A \cdot \text{lsolve}(A, b) - b \right| = 0$$

**Листинг 8.5. Символьное решение СЛАУ (продолжение листинга 8.4)**

$$\text{lsolve}(A, b) \rightarrow \begin{pmatrix} \frac{14}{3} \\ \frac{4}{3} \\ \frac{-8}{9} \end{pmatrix}$$

# 8.2. Произвольные системы линейных уравнений

Классические задачи решения СЛАУ, рассмотренные в предыдущем разделе, предполагают равное количество уравнений и неизвестных, т. е. квадратную матрицу A. Именно для таких систем доказано, что решение существует и единственно, если определитель матрицы $|A| \neq 0$. Рассмотрим теперь задачи, в которых матрица A не является квадратной либо плохо обусловлена. Разберемся в этом разделе лишь с основными свойствами и идеологией подхода к таким системам, имея в виду, что наиболее эффективными способами их решения являются матричные разложения (*см. разд. 8.3*).

## 8.2.1. Переопределенные системы

Начнем разговор о "плохих" СЛАУ с переопределенными условиями, в которых число уравнений больше числа неизвестных.

### О постановке задач

Чаще всего СЛАУ с прямоугольной матрицей размера M×N (при M>N, т. е. при числе уравнений большем числа неизвестных) вовсе не имеет решения, т. е. является *несовместной*, или *переопределенной*. Листинг 8.6 демонстрирует, что несовместные системы не могут быть решены посредством вычислительного блока Given/Find.

> **Внимание!**
>
> При помощи встроенной функции lsolve (вторая строка листинга) можно решать системы с прямоугольной матрицей только, начиная с версии Mathcad 13. В более ранних версиях попытка применить ее для расчета решения таких

СЛАУ закончится неудачей, т. е. вместо результата во второй строке листинга будет выведено сообщение об ошибке.

**Листинг 8.6. Попытка решения несовместных СЛАУ**

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \qquad b := \begin{pmatrix} 51 \\ 109 \\ 172 \end{pmatrix}$$

$$\mathrm{lsolve}(A, b) = \begin{pmatrix} 10.333 \\ 19.917 \end{pmatrix}$$

$$x := \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Given

$A \cdot x = b$

Find(x) = ■

Конечно, в редких случаях система с прямоугольной матрицей может оказаться совместной (если выбран соответствующий вектор `b`), как это показано в листинге 8.7.

**Листинг 8.7. Пример нахождения решения СЛАУ с прямоугольной матрицей**

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \qquad b := \begin{pmatrix} 50 \\ 110 \\ 170 \end{pmatrix}$$

$$\mathrm{lsolve}(A, b) = \begin{pmatrix} 10 \\ 20 \end{pmatrix}$$

$$x := \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Given $\qquad A \cdot x = b$

$$\mathrm{Find}(x) = \begin{pmatrix} 10 \\ 20 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \cdot \begin{pmatrix} 10 \\ 20 \end{pmatrix} = \begin{pmatrix} 50 \\ 110 \\ 170 \end{pmatrix}$$

На практике (особенно в последнее время) задачи отыскания решения переопределенных СЛАУ встречаются довольно часто. Благодаря нововведению разработчиков, начиная с версии Mathcad 13, их решение не представляет труда, т. к. оно унифицировано с решением "хороших" СЛАУ. Тем не менее очевидно, что в большинстве случаев "настоящего" решения (такого, которое бы обращало уравнения в тождества) не существует (а в случае недоопределенных уравнений "настоящих" решений, наоборот, бесконечно много). Поэтому посвятим оставшуюся часть *разд. 8.2*, главным образом, пояснению вопросов математической постановки задач на решение СЛАУ с прямоугольной матрицей.

> **Примечание**
>
> Пользователи Mathcad 13 могут расценивать содержание *разд. 8.2* как объяснение алгоритмов работы функции `lsolve`, в то время, как пользователи прежних версий найдут в нем рецепты решения недоопределенных и переопределенных СЛАУ без применения `lsolve`.

## Псевдорешение (метод наименьших квадратов)

Напрашивающийся сам собой путь решения нашего простого примера является хорошей иллюстрацией подхода к общей проблеме переопределенных СЛАУ. А именно, вместо точного решения системы уравнений следует организовать поиск такого вектора `x`, который будет наилучшим образом удовлетворять всем уравнениям, т. е. минимизировать их *невязку* (расхождение между вектором `A·x` и вектором правой части СЛАУ `b`). Поскольку невязка `A·x-b` является векторной величиной, то, исходя из практических соображений, минимизации надо подвергать ее норму (т. е. скаляр) `|A·x-b|`. Этот подход позволит, с одной стороны, получить разумное, с физической точки зрения, решение задачи, а, с другой — использовать полезную информацию, заключенную во всех уравнениях.

Таким образом, при интерпретации переопределенных СЛАУ принято искать не точное решение (которого, как уже отмечалось, при данной постановке задачи просто нет), а *псевдорешение* — вектор, минимизирующий норму невязки системы уравнений. Таким образом, задача решения линейной системы уравнений заменяется задачей отыскания глобального минимума функции `f(x)=|A·x-b|`. Повторимся еще раз, что, в полном соответствии с обозначениями, принятыми в Mathcad, `x` — это неизвестный вектор, а символ модуля (две вертикальные черты) означает операцию вычисления нормы (евклидовой длины вектора). Поскольку эта минимизируемая норма зависит от суммы квадратов компонент неизвестного вектора, то процедура поиска псевдоре-

шения является ни чем иным, как реализацией *метода наименьших квадратов* (МНК).

График функции двух переменных $f(x_0, x_1)$ показан в виде трехмерной поверхности на рис. 8.1, а несколько его сечений в окрестности минимума — на рис. 8.2. Нетрудно сообразить, почему структура функции оказывается именно такой, если вспомнить, что $f(x)^2$ является квадратичной формой относительно неизвестных $x_0$ и $x_1$. Надо отметить, что если бы наша система имела большее число уравнений, то вычислительная задача соответствующим образом усложнилась бы, т. к. минимизацию следовало бы проводить не по двум, а по большему числу переменных.

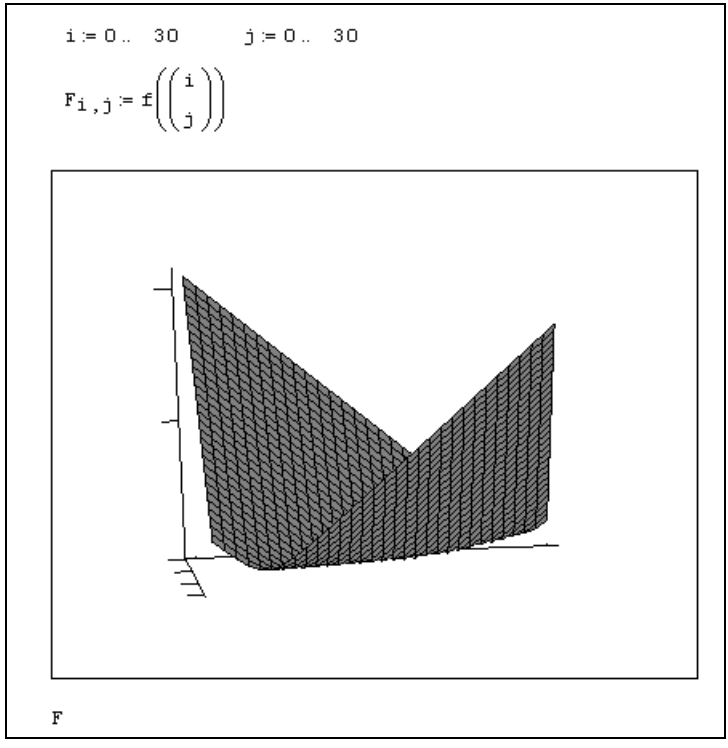



**Рис. 8.1.** График минимизируемой функции невязки $f(x) = |A \cdot x - b|$

Обращаясь к практической стороне дела, следует вспомнить, что для решения задач минимизации невязки системы уравнений (*см. главу 6*) в Mathcad предусмотрены две встроенные функции — `Minerr` и `Minimize`. Применение этих альтернативных средств иллюстрируется листингами 8.8 и 8.9 соответ-

ственно. В первом случае используется ключевое слово `Given` для ввода СЛАУ в матричной форме, а во втором — в явном виде определяется функция `f(x)`, подлежащая минимизации. Обе встроенные функции, как вы помните, применяют итерационные численные алгоритмы, и поэтому требуют ввода начальных значений для всех неизвестных, т. е. нулевой итерации (вторая строка обоих листингов).

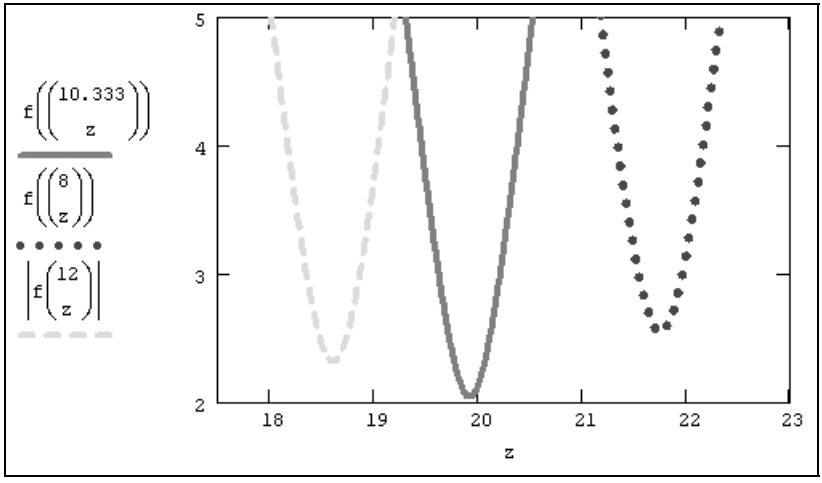



**Рис. 8.2.** Сечения графика невязки `f(x)` для трех фиксированных значений $x_0$

**Листинг 8.8. Поиск псевдорешения при помощи функции `Minerr`**

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \qquad b := \begin{pmatrix} 51 \\ 109 \\ 172 \end{pmatrix}$$

$$x := \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Given

$A \cdot x = b$

$$\text{MinErr}(x) = \begin{pmatrix} 12.364 \\ 17.979 \end{pmatrix}$$

**Листинг 8.9. Поиск псевдорешения при помощи функции `Minimize`**

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \qquad b := \begin{pmatrix} 51 \\ 109 \\ 172 \end{pmatrix}$$

$$x := \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$f(x) := |A \cdot x - b|$$

$$\text{Minimize}(f, x) = \begin{pmatrix} 10.333 \\ 19.917 \end{pmatrix}$$

$$\text{lsolve}(A, b) = \begin{pmatrix} 10.333 \\ 19.917 \end{pmatrix}$$

На первый взгляд, удобнее использовать функцию `Minerr`, поскольку для нее текст Mathcad-программы (листинг 8.8) выглядит более приближенным к изначальной постановке задачи (а именно, найти приближенное решение заданной системы уравнений). Между тем, выбор функции `Minerr` таит в себе, по крайней мере, две опасности.

Во-первых, сравнивая результаты, можно с недоумением обнаружить, что они абсолютно различны, причем более похож на правильный ответ, выдаваемый функцией `Minimize` (об этом можно судить, опираясь на соображение ожидаемой близости псевдорешения к решению системы из листинга 8.7, от которого мы отошли лишь немного). Разгадка заключается в особенностях применения численного алгоритма, заложенного в функцию `Minerr`. По умолчанию решение линейной системы уравнений производится при помощи линейного алгоритма. Чтобы сменить его на нелинейный, необходимо нажатием правой кнопки мыши вызвать из области имени функции контекстное меню и сменить тип метода **Linear** (Линейный) на один из нелинейных методов **Nonlinear** (Нелинейный) (рис. 8.3). Однако даже в этом случае, как показывает опыт, наиболее правильный ответ получается только для одного из трех доступных численных алгоритмов.

Во-вторых, приближенное решение СЛАУ посредством вычислительного блока `Given`/`Minerr` невозможно при наличии дополнительных условий, выражающих вспомогательную априорную информацию о постановке задачи. В результате исходная задача сводится к условной минимизации функции `f(x)`, что в Mathcad-программе соответствует появлению дополнительных неравенств после ключевого слова `Given` (*см. пример ниже, решенный в лис-*

*тинге 8.10*). Однако все уравнения и неравенства, предшествующие функции `Minerr`, будут восприниматься численным процессором Mathcad одинаково, т. е. не в качестве жестких условий, а как выражения, которые должны выполняться лишь с некоторой точностью. В результате найденное решение может вовсе не отвечать условию положительной определенности (этот вопрос обсуждался в *разд. 6.2*, когда речь шла о задачах оптимизации).

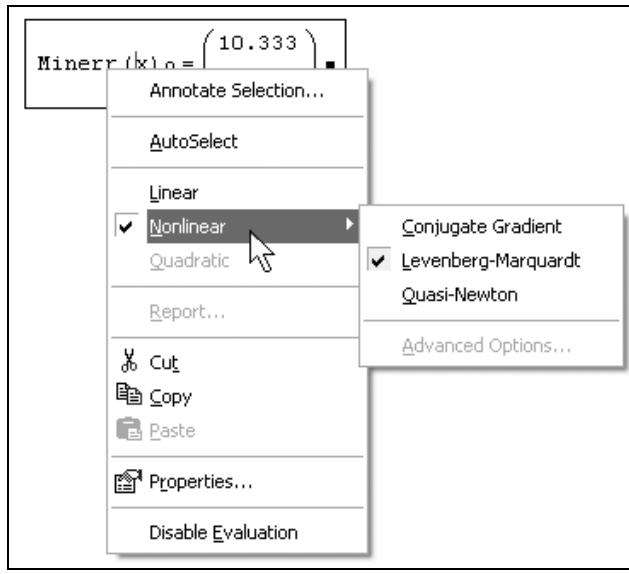



**Рис. 8.3.** Изменение численного алгоритма для функции `Minerr`

В свете перечисленных проблем применение функции `Minimize` для поиска псевдорешения СЛАУ выглядит более обоснованным. К слову заметим, что выбор любого из трех нелинейных численных алгоритмов минимизации приводит к одинаковым результатам (листинг 8.9). Функция `Minimize` годится и для отыскания псевдорешения с дополнительными условиями, выражающими имеющуюся априорную информацию. Соответствующий пример иллюстрирует листинг 8.10, в котором осуществляется поиск псевдорешения СЛАУ, немного отличающейся от рассмотренной выше. Это отличие приводит к тому, что при использовании той же самой методики получается отрицательное решение (третья строка листинга). При добавлении в задачу дополнительных условий результат минимизации оказывается другим (последняя строка листинга 8.10).

***Примечание***

Несколько иной подход к переопределенным системам, для которых имеется дополнительная информация другого типа (а именно, априорная оценка неизвестного вектора x), будет изложен в следующем разделе.

**Листинг 8.10. Поиск псевдорешения при наличии априорной информации**

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \qquad b := \begin{pmatrix} 62 \\ 109 \\ 172 \end{pmatrix}$$

$$f(x) := |A \cdot x - b| \qquad x := \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\text{Minimize}(f, x) = \begin{pmatrix} -4.333 \\ 31.833 \end{pmatrix}$$

Given

$x_0 > 0.1$

$x_1 > 0.1$

$$\text{Minimize}(f, x) = \begin{pmatrix} 0.1 \\ 28.35 \end{pmatrix}$$

## 8.2.2. Недоопределенные системы

Альтернативным рассмотренному в предыдущем разделе классу СЛАУ с прямоугольной матрицей размера M×N (при M<N) является случай, когда количество уравнений меньше количества неизвестных. Как несложно сообразить, такие системы либо имеют бесконечное число решений, либо не имеют решения вовсе. Решение несовместной системы можно искать в смысле минимизации нормы невязки по методу наименьших квадратов (*см. разд. 8.2.1 и 8.2.3*). Ниже в этом разделе будут разобраны задачи, имеющие бесконечное множество решений.

### О постановке задач

Пример аналитического решения системы двух уравнений с тремя неизвестными при помощи символьного процессора приведен в листинге 8.11.

**Листинг 8.11. Аналитический поиск семейства решений недоопределенной СЛАУ**

```
Given
1x + 3y + 5z = 10
2x + 4y + 6z = 10
Find(x, y, z) → ⎛  z − 5  ⎞
                ⎜−2·z + 5 ⎟
                ⎝    z    ⎠
```

Приведенную простую систему двух уравнений нам удалось без труда решить аналитически, однако для решения недоопределенных систем, состоящих из большого числа уравнений, необходимо уметь использовать численные алгоритмы.

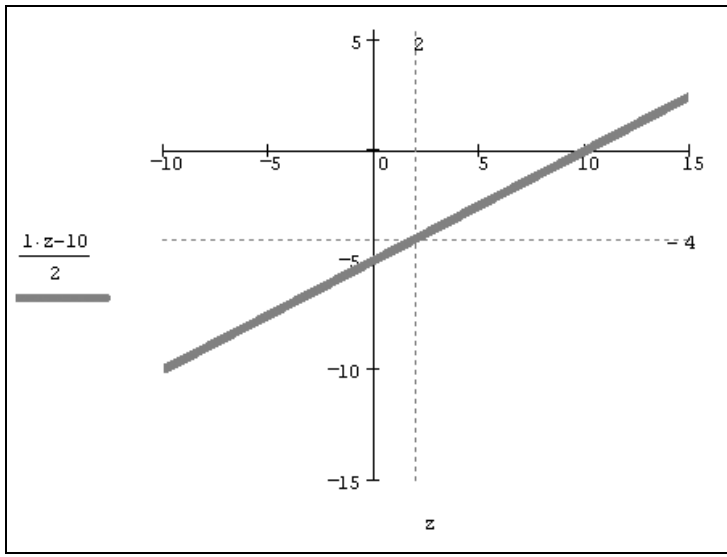



**Рис. 8.4.** График всех решений уравнения $x_0-2x_1=10$

Рассмотрим в качестве еще одного модельного примера единственное уравнение с двумя неизвестными $x_0-2x_1=10$. Хорошей визуализацией, подчеркивающей специфику данной задачи, будет график геометрического места его решений на плоскости $(x_0,x_1)$ (рис. 8.4). Заметим, что на этом и на следую-

щем рисунке мы использовали обозначение (для корректного построения графика) x0≡z. Очевидно, что решений уравнения бесконечно много, и все они находятся на прямой линии x1=(x0-10)/2.

Как и в предыдущем примере (листинг 8.11), нам удалось решить задачу аналитически, и опять число решений получилось бесконечным. Для того чтобы получить возможность осмысленного численного решения подобных (уже не настолько тривиальных) задач, необходимо выделить из бесконечного множества решений одно, оправданное с математической точки зрения, и предложить алгоритм его поиска. Эта проблема решается посредством привлечения понятия *нормального псевдорешения*.

## Нормальное псевдорешение

Способ выбора одного решения из бесконечного множества, изображенного на рис. 8.4, подсказывает, по аналогии с переопределенными СЛАУ (*см. разд. 8.2.1*), сам физический смысл задачи, которую можно интерпретировать как M измерений с N неизвестными (M<N). Для того чтобы получить разумное единственное решение задачи, необходимо "доопределить" ее, добавив некоторые априорные соображения о значении неизвестного вектора x.

Если априорной информации о примерной величине вектора x нет, единственным образом решить СЛАУ невозможно. Однако если о неизвестном векторе хоть что-то можно сказать, данная информация позволит доопределить систему уравнений и получить решение, учитывающее как систему, так и априорную информацию. Иными словами, следует ввести в задачу определенные ожидания о величине вектора x. Математически, не теряя общности, можно полагать ожидаемое значение вектора x нулевым, поскольку перейти от любого x к нуль-вектору можно простым линейным преобразованием переменных, которое изменит только вектор правой части b.

Таким образом, вполне логично объявить решением недоопределенной СЛАУ такое из решений, которое ближе всего находится к нулевому вектору, т. е. обладает минимальной нормой |x|~min. Это решение называют *нормальным псевдорешением* СЛАУ, и искать его следует, минимизируя норму вектора x на предварительно полученном семействе решений СЛАУ. Иными словами, решение недоопределенной СЛАУ сводится к условной минимизации функции |x| (рис. 8.5). Геометрический смысл нормального псевдорешения (в рассматриваемом случае одного уравнения с двумя неизвестными) очевиден: это точка, лежащая на пересечении прямой семейства всех решений и перпендикуляра к этой прямой, восстановленного из начала координат. На рис. 8.4 нормальное псевдорешение выделено пунктирными линиями.

Принимая во внимание введенную технику решения недоопределенных СЛАУ, можно предложить для этих задач простой и понятный алгоритм, опирающийся на встроенную функцию `Minimize`. В листинге 8.12 решена задача численного отыскания псевдорешения рассматриваемого единственного уравнения (рис. 8.5), а в листинге 8.13 — решение недоопределенной системы двух уравнений с тремя неизвестными, которая исследовалась аналитически в листинге 8.11 (*см. предыдущий разд.*). Как уже отмечалось, смысл обоих листингов заключается в минимизации нормы искомого вектора при условии, что выполнена система равенств $A \cdot x = b$. Графики функции $f(x) = |x|$, при условии выполнения СЛАУ из листингов 8.12 и 8.13, изображены на рис. 8.5 и 8.6 соответственно.

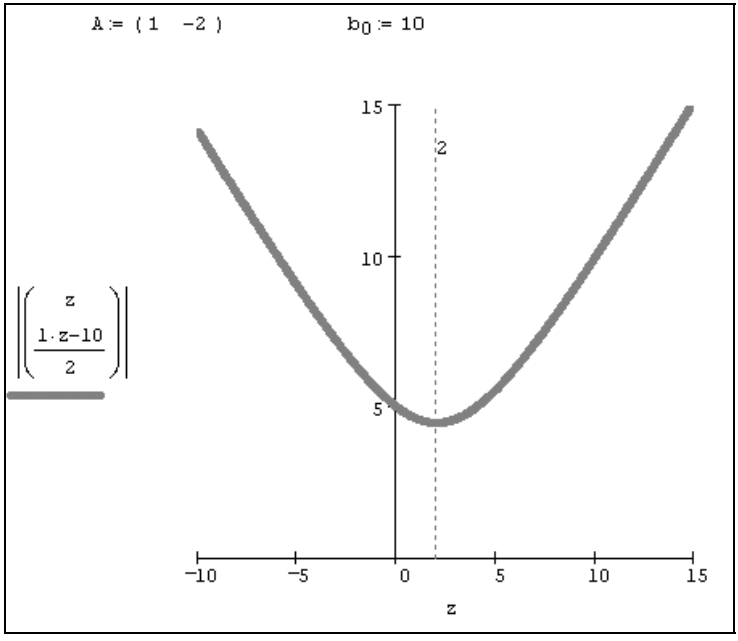



**Рис. 8.5.** График функции $f(x_0) = |x|$ при условии, что $x_0 - 2x_1 = 10$

**Листинг 8.12. Поиск нормального псевдорешения уравнения $x_0 - 2x_1 = 10$**

```
A := ( 1  −2 )

b0 := 10
```

$$\text{lsolve}(A, b) = \begin{pmatrix} 2 \\ -4 \end{pmatrix}$$

$$f(x) := |x|$$

$$x := (0 \quad 0)^T$$

Given

$$A \cdot x = b$$

$$\text{Minimize}(f, x) = \begin{pmatrix} 2 \\ -4 \end{pmatrix}$$

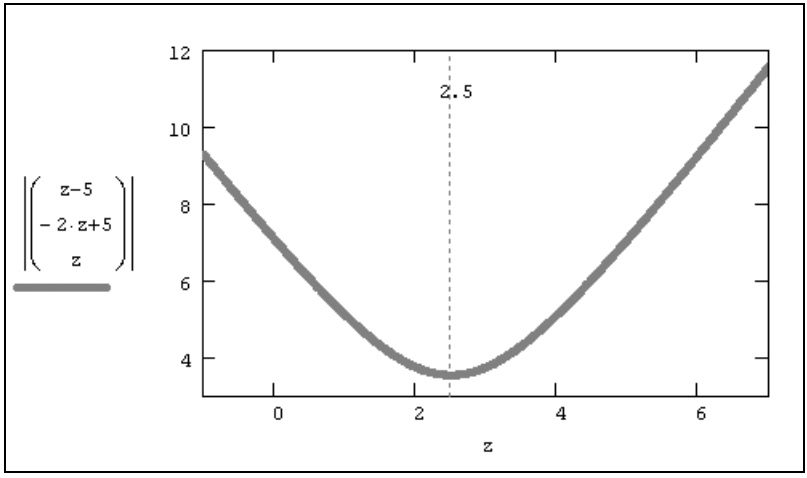



**Рис. 8.6.** График функции $f(x_2) = |x|$
при условии выполнения СЛАУ из листинга 8.13

## Листинг 8.13. Поиск нормального псевдорешения недоопределенной СЛАУ

$$A := \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix} \qquad b := \begin{pmatrix} 10 \\ 10 \end{pmatrix}$$

$$\text{lsolve}(A, b) = \begin{pmatrix} -2.5 \\ -2.887 \times 10^{-15} \\ 2.5 \end{pmatrix}$$

$$f(x) := |x|$$

$$x := \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Given

$$A \cdot x = b$$

$$\text{Minimize}(f, x) = \begin{pmatrix} -2.5 \\ 5.101 \times 10^{-7} \\ 2.5 \end{pmatrix}$$

***Примечание***

Если недоопределенная СЛАУ не имеет бесконечного множества решений, а является несовместной, то способ, предложенный в листинге 8.13, использовать нельзя, т. к. условие заведомо невыполнимо. Подход к решению таких задач заключается в поиске компромиссного решения, минимизирующего как норму невязки, так и норму решения (*см. разд. 8.2.3*).

## 8.2.3. Вырожденные и плохо обусловленные системы

Вернемся вновь к СЛАУ $A \cdot x = b$ с квадратной матрицей $A$ размера $M \times M$, которая, в отличие от рассмотренного выше "хорошего" случая (*см. разд. 8.1*), требует специального подхода. Обратим внимание на два похожих типа СЛАУ:

- вырожденная система (с нулевым определителем $|A| = 0$);
- плохо обусловленная система (определитель $A$ не равен нулю, но число обусловленности очень велико).

Несмотря на то, что эти типы систем уравнений существенно отличаются друг от друга (для первого решение отсутствует, а для второго существует и единственно), с практической точки зрения вычислителя между ними много общего.

### Вырожденные СЛАУ

Вырожденная система — это система, описываемая матрицей с нулевым определителем $|A| = 0$ (*сингулярной* матрицей). Поскольку некоторые уравнения, входящие в такую систему, представляются линейной комбинацией других уравнений, то фактически сама система является недоопределенной. Не-

сложно сообразить, что, в зависимости от конкретного вида вектора правой части `b`, существует либо бесконечное множество решений, либо не существует ни одного. Первый из вариантов сводится к построению нормального псевдорешения (т. е. выбора из бесконечного множества решений такого, которое наиболее близко к определенному, например, нулевому, вектору). Данный случай был подробно рассмотрен в *разд. 8.2.2* (см. листинги 8.11—8.13).

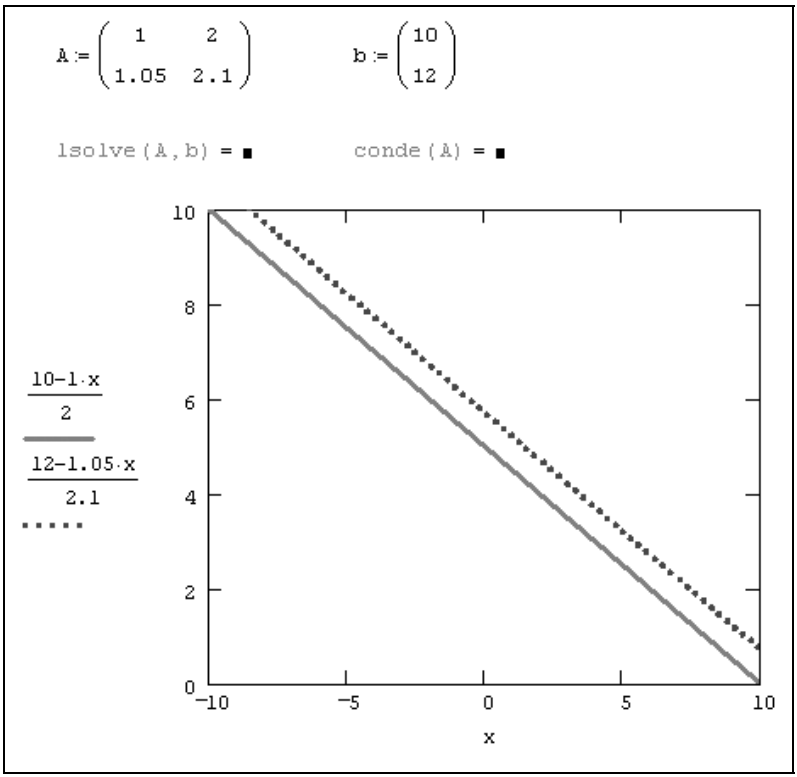



**Рис. 8.7.** Графическое представление несовместной системы двух уравнений с сингулярной матрицей

Рассмотрим второй случай, когда СЛАУ `A·x=b` с сингулярной квадратной матрицей `A` не имеет ни одного решения. Пример такой задачи (для системы двух уравнений) проиллюстрирован рис. 8.7, в верхней части которого вводятся матрица `A` и вектор `b`, а также предпринимается (неудачная, поскольку матрица `A` сингулярная) попытка решить систему при помощи функции `lsolve`. График, занимающий основную часть рисунка, показывает, что два уравнения, задающие систему, определяют на плоскости $(x_0, x_1)$ две парал-

лельные прямые. Прямые не пересекаются ни в одной точке координатной плоскости, и решения системы, соответственно, не существует.

### *Примечание*

Во-первых, заметим, что СЛАУ, заданная несингулярной квадратной матрицей размера 2×2, определяет на плоскости пару пересекающихся прямых (см. рис. 8.9 ниже). Во-вторых, стоит сказать, что, если бы система была совместной, то геометрическим изображением уравнений были бы две совпадающие прямые, описывающие бесконечное число решений.

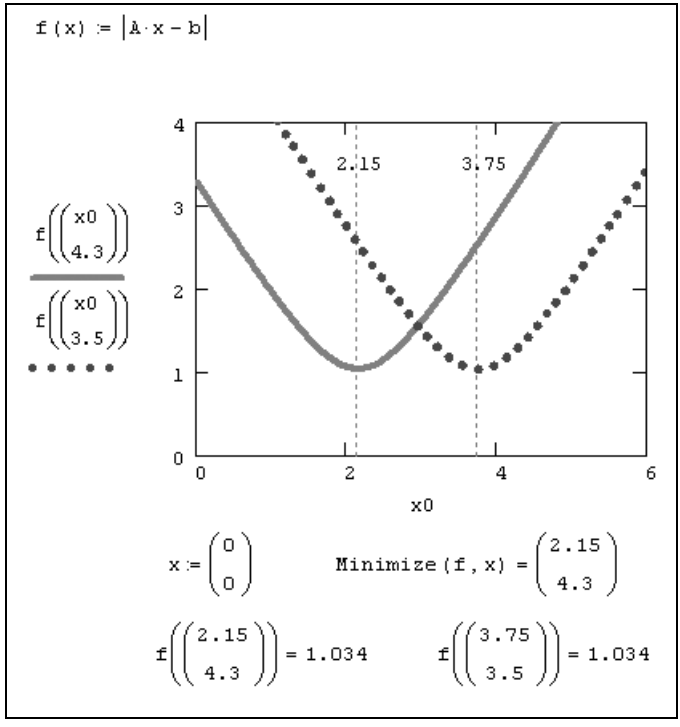



**Рис. 8.8.** График сечений функции невязки f(x)=|A·x–b|

Несложно догадаться, что в рассматриваемом сингулярном случае псевдорешений системы, минимизирующих невязку |A·x–b|, будет бесконечно много, и лежать они будут на третьей прямой, параллельной двум показанным на рис. 8.7 и расположенной в середине между ними. Это иллюстрирует рис. 8.8, на котором представлено несколько сечений функции f(x)=|A·x–b|, которые говорят о наличии семейства минимумов одинаковой глубины. Если попытаться использовать для их отыскания встроенную функцию Minimize,

ее численный метод будет всегда отыскивать какую-либо одну точку упомянутой прямой (в зависимости от начальных условий). Поэтому для определения единственного решения следует выбрать из всего множества псевдорешений то, которое обладает наименьшей нормой. Можно пытаться оформить данную многомерную задачу минимизации в Mathcad при помощи комбинаций встроенных функций `Minimize`, однако более эффективным способом будет использование регуляризации (см. ниже) или ортогональных матричных разложений (*см. разд. 8.3*).

## Плохо обусловленные системы

Плохо обусловленная система — это система, у которой определитель `A` не равен нулю, но число обусловленности $|A^{-1}|\cdot|A|$ очень велико. Несмотря на то, что плохо обусловленные системы имеют единственное решение, на практике искать это решение чаще всего не имеет смысла. Рассмотрим свойства плохо обусловленных СЛАУ на двух конкретных примерах (листинг 8.14).

**Листинг 8.14. Решение двух близких плохо обусловленных СЛАУ**

$$\mathrm{lsolve}\left[\begin{pmatrix}1 & 2\\ 3 & 6.01\end{pmatrix},\begin{pmatrix}3\\5\end{pmatrix}\right]=\begin{pmatrix}803\\-400\end{pmatrix}$$

$$\mathrm{lsolve}\left[\begin{pmatrix}1 & 2.01\\ 3 & 6\end{pmatrix},\begin{pmatrix}3\\5\end{pmatrix}\right]=\begin{pmatrix}-265\\133.333\end{pmatrix}$$

Каждая строка листинга 8.14 содержит решение двух очень близких плохо обусловленных СЛАУ (с одинаковой правой частью `b` и мало отличающимися матрицами `A`). Несмотря на близость, точные решения этих систем оказываются очень далекими друг от друга. Надо заметить, что для системы двух уравнений точное решение получить легко, однако при решении СЛАУ большой размерности незначительные ошибки округления, неминуемо накапливаемые при расчетах (в том числе и "точным" алгоритмом Гаусса), приводят к огромным погрешностям результата. Возникает вопрос: имеет ли смысл искать численное решение, если заранее известно, что, в силу неустойчивости самой задачи, оно может оказаться совершенно неправильным?

Еще одно соображение, которое вынуждает искать специальные методы решения плохо обусловленных СЛАУ (даже приведенной в качестве примера в листинге 8.14 системы двух уравнений), связано с их физической интерпре-

тацией как результатов эксперимента. Если изначально известно, что входные данные получены с некоторой погрешностью, то решать плохо обусловленные системы не имеет вовсе никакого смысла, поскольку малые ошибки модели (матрицы `A` и вектора `b`) приводят к большим ошибкам решения. Задачи, обладающие такими свойствами, называются *некорректными*.

Чтобы лучше понять причину некорректности, полезно сравнить графическую интерпретацию хорошо (рис. 8.9) и плохо (рис. 8.10) обусловленной системы двух уравнений. Решение системы визуализируется точкой пересечения двух прямых линий, изображающих каждое из уравнений.

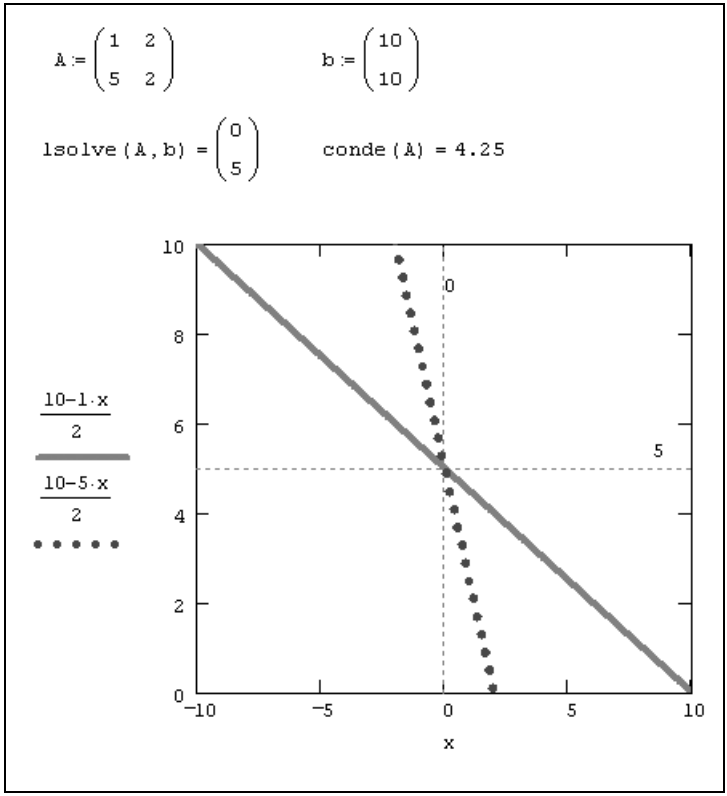



**Рис. 8.9.** График хорошо обусловленной системы двух уравнений

Из рис. 8.10 видно, что прямые, соответствующие плохо обусловленной СЛАУ, располагаются в непосредственной близости друг от друга (почти па-

раллельны). В связи с этим малые ошибки в расположении каждой из прямых могут привести к значительным погрешностям локализации точки их пересечения (решения СЛАУ) в противоположность случаю хорошо обусловленной системы, когда малые погрешности в наклоне прямых мало повлияют на место точки их пересечения (рис. 8.9).

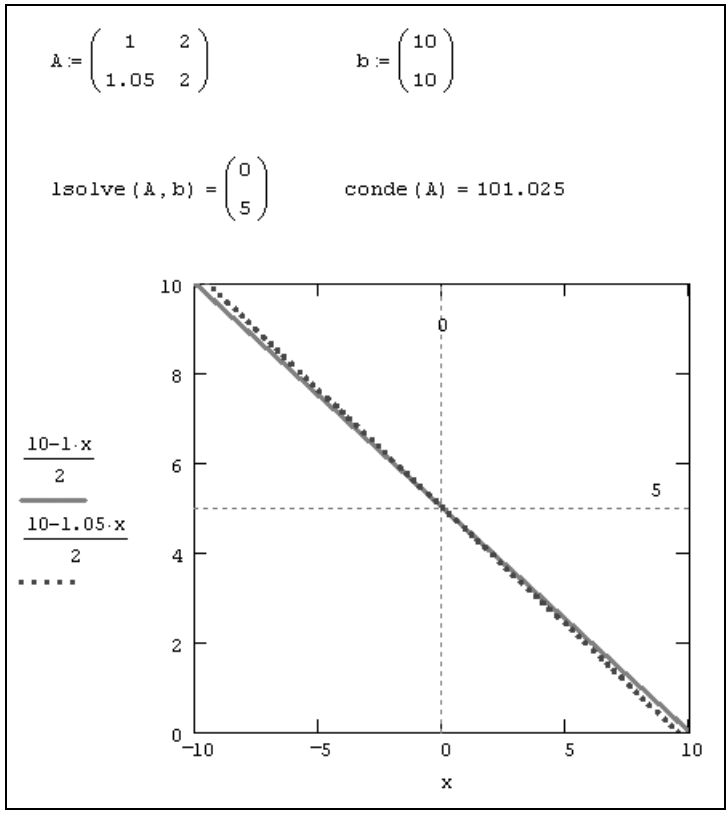



**Рис. 8.10.** График плохо обусловленной системы двух уравнений

### *Примечание*

Плохая обусловленность матрицы типична и при реконструкции экспериментальных данных, задаваемых переопределенными (несовместными) СЛАУ (например, в задачах томографии). Именно такой случай приведен в качестве примера в следующем разделе (см. листинг 8.16 ниже).

## Метод регуляризации

Для решения некорректных задач, в частности, вырожденных и плохо обусловленных СЛАУ, разработан очень эффективный прием, называемый *регуляризацией*. В его основе лежит учет дополнительной априорной информации о структуре решения (векторе априорной оценки `x0`), которая очень часто присутствует в практических случаях. В связи с тем, что регуляризация была подробно рассмотрена в *разд. 6.3.3*, напомним лишь, что задачу решения СЛАУ `A·x=b` можно заменить задачей отыскания минимума *функционала Тихонова*:

$$\Omega(x,\lambda)=|A\cdot x-b|^2+\lambda\cdot|x-x0|^2. \qquad (8.3)$$

Здесь $\lambda$ — малый положительный *параметр регуляризации*, который необходимо подобрать некоторым оптимальным способом. Можно показать, что задачу минимизации функционала Тихонова можно, в свою очередь, свести к решению другой СЛАУ:

$$(A^T\cdot A+\lambda\cdot I)\cdot x=A^T\cdot B+\lambda\cdot x0, \qquad (8.4)$$

которая при $\lambda\to 0$ переходит в исходную плохо обусловленную систему, а при больших $\lambda$, будучи хорошо обусловленной, имеет решение `x0`. Очевидно, оптимальным будет некоторое промежуточное значение $\lambda$, устанавливающее определенный компромисс между приемлемой обусловленностью и близостью к исходной задаче. Отметим, что регуляризационный подход сводит некорректную задачу к условно-корректной (по Тихонову) задаче отыскания решения системы (8.4), которое, в силу линейности задачи, является единственным и устойчивым.

Приведем, без излишних комментариев, регуляризованное решение вырожденной системы, которая была представлена на рис. 8.8. Листинг 8.15 демонстрирует отыскание решения задачи (8.4), а полученная зависимость невязки и самого решения от параметра регуляризации $\lambda$ показана на рис. 8.11 и 8.12 соответственно. Важно подчеркнуть, что решения исходной системы и, следовательно, системы (8.4) при $\lambda=0$ не существует.

**Листинг 8.15. Регуляризация вырожденной СЛАУ**

$$A:=\begin{pmatrix}1 & 2\\ 1.05 & 2.1\end{pmatrix}\qquad b:=\begin{pmatrix}10\\ 10\end{pmatrix}\qquad x0:=\begin{pmatrix}0\\ 0\end{pmatrix}$$

$$g(\lambda):=\mathrm{lsolve}\left(A^T\cdot A+\lambda\cdot \mathrm{identity}(2),\ A^T\cdot b+\lambda\cdot x0\right)$$

Заключительным шагом регуляризации является выбор оптимального $\lambda$. Имеется, по крайней мере, два соображения, исходя из которых, можно выбрать параметр регуляризации, если опираться на зависимость от него невязки. В рассматриваемом примере применим критерий определения $\lambda$, опирающийся на подбор нормы невязки, равной априорной оценке погрешностей задания входных данных: матрицы $A$ и вектора $b$, т. е. величине $|\delta A|+|\delta b|$. Например, можно выбрать норму невязки и, соответственно, параметр $\lambda$ и решение $x(\lambda)$, которые отмечены на рис. 8.11 и 8.12 пунктирами.

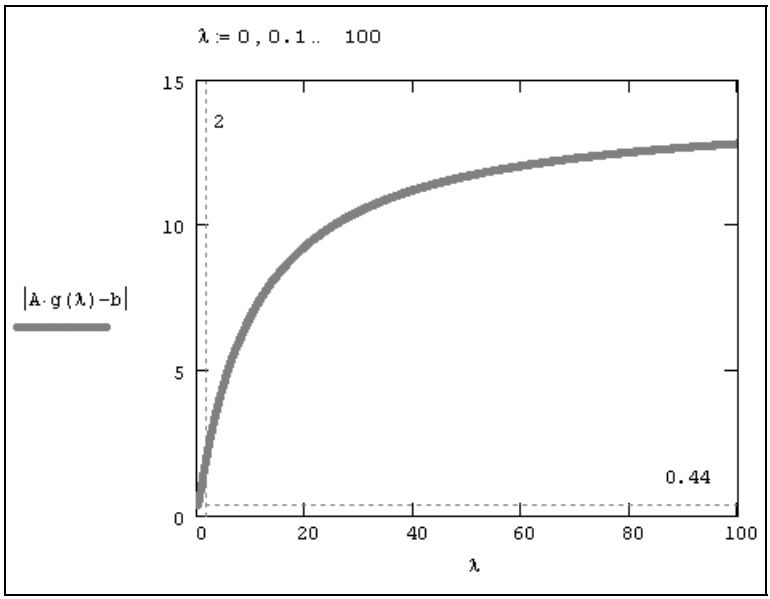



**Рис. 8.11.** Зависимость невязки регуляризованного решения вырожденной СЛАУ от параметра $\lambda$ (продолжение листинга 8.15)

### *Примечание 1*

Другим вариантом выбора $\lambda$, не требующим никаких априорных соображений относительно погрешностей модели, является так называемый *квазиоптимальный* метод, рассмотренный в *разд. 6.3.3*.

### *Примечание 2*

Полезно убедиться в том, что формула (8.4) в случае линейной задачи дает тот же результат, что и общая формула (8.3). Для этого достаточно изменить в листинге 8.15 последнюю строку, выражающую решение СЛАУ (8.4), на код,

реализующий минимизацию численным методом, как это показано в листинге 8.16. Расчеты (которые требуют значительно большего компьютерного времени) дают тот же самый результат, что был приведен на рис. 8.11 и 8.12.

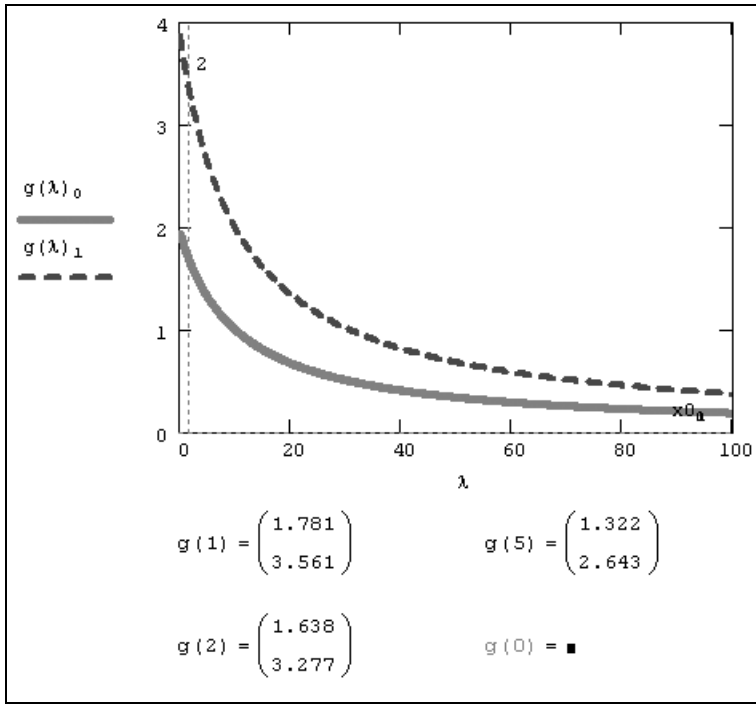



**Рис. 8.12.** Регуляризованное решение в зависимости от $\lambda$ (продолжение листинга 8.15)

### *Примечание 3*

Попробуйте в расчетах листингов 8.15 и 8.16 взять иную, например, более реалистичную, априорную оценку решения (вместо использованного в них нулевого вектора `x0`) и посмотреть, как это повлияет на результат.

### *Примечание 4*

Любопытно также применить вместо формулы (8.3) в качестве функционала Тихонова другую зависимость: $\Omega(x,\lambda)=|A\cdot x-b|+\lambda\cdot|x-x0|$. Соответствующий пример расчетов вы найдете на компакт-диске.

**Листинг 8.16. Регуляризация СЛАУ при помощи алгоритма минимизации λ (продолжение листинга 8.15)**

$$f(x, \lambda) := \left(\left|A \cdot x - b\right|\right)^2 + \lambda \cdot \left(\left|x - x0\right|\right)^2$$

$$x := \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad g(\lambda) := \text{Minimize}(f, x)$$

# 8.3. Матричные разложения

Современная вычислительная линейная алгебра — бурно развивающаяся наука. Главная проблема линейной алгебры — это решение систем линейных уравнений. В настоящее время разработано множество методов, упрощающих эти задачи, которые, в частности, зависят от структуры матрицы СЛАУ. Большинство методов основано на представлении матрицы в виде произведения других матриц специального вида, или *матричных разложениях* (или, по-другому, *факторизациях*). Как правило, после определенного разложения матрицы задача решения СЛАУ существенно упрощается. Кроме того, прибегая к определенным разложениям матриц, можно получить универсальный вид решения, как для "хороших" (*см. разд. 8.1*), так и для "плохих" (*см. разд. 8.2*) СЛАУ.

В Mathcad имеется несколько встроенных функций, реализующих алгоритмы наиболее популярных матричных разложений; и мы рассмотрим их вместе с соответствующими типовыми задачами решения СЛАУ. Однако вначале сделаем некоторые важные замечания о решении систем с треугольной матрицей.

## 8.3.1. СЛАУ с треугольной матрицей

Начнем разговор о решении СЛАУ посредством матричных разложений с простого, но исключительно важного частного случая, а именно, систем с *треугольной матрицей*, т. е. такой матрицей, по одну из сторон от диагонали которой находятся одни нули. В листинге 8.17 приведен пример верхнетреугольной `L` матрицы (первая строка листинга), а также ряд формул, по которым за `N` операций рассчитывается решение соответствующей линейной системы. Из этого примера ясно, что сведение задачи общего вида к СЛАУ с треугольной матрицей практически решает эту задачу, поскольку гарантировано получение результата за минимальное число операций.

### Примечание 1

Подчеркнем, что мы намеренно оформили расчет решения треугольной СЛАУ в виде пользовательской функции `trg(A,b)`, для чего нам пришлось применить элементы программирования. Мы так поступили, чтобы иметь впоследствии возможность применять ее в качестве подпрограммы решения СЛАУ посредством различных матричных разложений, включая листинг 8.17 в последующие листинги. Более того, несложно модифицировать листинг 8.17 для решения нижне-треугольных систем, определив, таким образом, еще одну функцию `ltrg(A,b)` (вы найдете соответствующий листинг на компакт-диске).

### Примечание 2

Алгоритм решения СЛАУ с треугольной матрицей называют иногда *прямым ходом* решения СЛАУ (часто подразумевая, что до него выполнены определенные преобразования, именуемые *обратным ходом*) либо просто *подстановкой*. В частности, решение еще одного часто встречающегося типа систем с трехдиагональной матрицей сводится к прямому и обратному ходу алгоритма, называемого *прогонкой* (*см. разд. 11.2.2*).

**Листинг 8.17. Решение СЛАУ с треугольной матрицей (прямой ход)**

$$A := \begin{pmatrix} 1 & 2 & -3 \\ 0 & 5 & 6 \\ 0 & 0 & -3 \end{pmatrix} \qquad b := \begin{pmatrix} 10 \\ 20 \\ 30 \end{pmatrix}$$

$$\begin{aligned} &trg(A,b) := \left| \begin{array}{l} N \leftarrow cols(A) \\ x_{N-1} \leftarrow \dfrac{b_{N-1}}{A_{N-1,N-1}} \\ \text{for } i \in N-2 \,..\, 0 \\ \quad x_i \leftarrow \dfrac{1}{A_{i,i}} \cdot \left( b_i - \displaystyle\sum_{j=i+1}^{N-1} A_{i,j} \cdot x_j \right) \\ x \end{array} \right. \\ &x := trg(A,b) \end{aligned}$$

$$x = \begin{pmatrix} -52 \\ 16 \\ -10 \end{pmatrix} \qquad A \cdot x - b = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

## 8.3.2. Разложение Холецкого

Разложением Холецкого *симметричной* (т. е. содержащей одинаковые элементы на местах, расположенных симметрично относительно главной диагонали) матрицы `A` является представление вида $A=L\cdot L^T$, где `L` — треугольная матрица. Алгоритм Холецкого реализован во встроенной функции `cholesky`:

❒ `cholesky(A)` — разложение Холецкого:

- `A` — квадратная, положительно определенная симметричная матрица.

Пример разложения Холецкого приведен в листинге 8.18. Обратите внимание, что в результате получается верхняя треугольная матрица (нули сверху от диагонали), а транспонированная матрица является нижней треугольной. В последней строке листинга приведена проверка правильности найденного разложения.

> ***Примечание***
>
> Исходя из математического вида разложения Холецкого, матрицу `L` иногда называют *квадратным корнем* матрицы `A`.

**Листинг 8.18. Разложение Холецкого**

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

$$L:=\text{cholesky}(A)$$

$$L=\begin{pmatrix}2.236 & 0 & 0\\ 0.894 & 2.28 & 0\\ 1.342 & -0.088 & 2.279\end{pmatrix}\qquad L^T=\begin{pmatrix}2.236 & 0.894 & 1.342\\ 0 & 2.28 & -0.088\\ 0 & 0 & 2.279\end{pmatrix}$$

$$L\cdot L^T-A=\begin{pmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{pmatrix}$$

Решение СЛАУ, если разложение Холецкого для него известно, основано на замене исходной системы $A\cdot x=b$ другой системой $L\cdot y=b$ (где $y=L^T\cdot x$), что иллюстрируется листингом 8.19. В первой строке листинга задается вектор правой части `b` и вычисляется стандартным методом Гаусса решение системы. В ос-

тавшейся части листинга СЛАУ решается при помощи разложения Холецкого, проведенного в листинге 8.18. После нахождения (простой подстановкой, т. к. матрица L — треугольная) вектора y опять подстановкой в уравнение $y=L^T \cdot x$ (т. к. $L^T$ — тоже треугольная матрица) отыскивается вектор x.

> ***Внимание!***
>
> В листинге 8.19 использованы пользовательские функции решения треугольных СЛАУ, описанные в предыдущем разделе. Если вы набираете листинг "от руки", их можно заменить универсальной встроенной функцией lsolve.

**Листинг 8.19. Решение СЛАУ при помощи разложения Холецкого (продолжение листингов 8.18 и 8.17)**

$$b := \begin{pmatrix} 10 \\ 20 \\ 30 \end{pmatrix} \qquad lsolve(A, b) = \begin{pmatrix} -2.148 \\ 3.259 \\ 4.741 \end{pmatrix}$$

$$y := ltrg(L, b)$$

$$x := trg\left(L^T, y\right)$$

$$x = \begin{pmatrix} -2.148 \\ 3.259 \\ 4.741 \end{pmatrix}$$

## 8.3.3. *LU*-разложение

LU-разложением матрицы A, или *треугольным* разложением, называется матричное разложение вида $P \cdot A = L \cdot U$, где L и U — нижняя и верхняя треугольные матрицы соответственно, а P — (диагональная) матрица перестановок. P, A, L, U — квадратные матрицы одного порядка:

❒ lu(A) — LU-разложение матрицы:

- A — квадратная матрица.

Результатом работы встроенной функции LU-разложения является матрица, составленная из матриц L и U соответственно. Чтобы выделить сами матрицы LU-разложения, а именно, P, L, U, необходимо применить функцию выделения подматрицы submatrix (листинг 8.20).

**Листинг 8.20. LU-разложение**

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 10 \end{pmatrix}$$

$$B := lu(A)$$

$$B = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 7 & 8 & 10 \\ 1 & 0 & 0 & 0.143 & 1 & 0 & 0 & 0.857 & 1.571 \\ 0 & 1 & 0 & 0.571 & 0.5 & 1 & 0 & 0 & -0.5 \end{pmatrix}$$

$$P := submatrix\left(B, 0, rows(B) - 1, 0, \frac{cols(B) - 1}{3}\right)$$

$$L := submatrix\left(B, 0, rows(B) - 1, \frac{cols(B) + 1}{3}, \frac{cols(B) - 1}{3}2\right)$$

$$U := submatrix\left(B, 0, rows(B) - 1, \frac{cols(B) + 1}{3}2, cols(B) - 1\right)$$

$$P = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad L = \begin{pmatrix} 1 & 0 & 0 \\ 0.143 & 1 & 0 \\ 0.571 & 0.5 & 1 \end{pmatrix} \quad U = \begin{pmatrix} 7 & 8 & 10 \\ 0 & 0.857 & 1.571 \\ 0 & 0 & -0.5 \end{pmatrix}$$

$$|P \cdot A - L \cdot U| = 0$$

Треугольное разложение матрицы системы линейных уравнений производится при ее решении численным методом Гаусса. Фактически именно алгоритм LU-разложения заложен во встроенной функции lsolve, применяемой для "точного" решения хорошо обусловленных систем (*см. разд. 8.1.2*). Таким образом, посредством LU-разложения можно решать СЛАУ с хорошо обусловленной матрицей A.

Почему метод LU-разложения является мощным средством решения СЛАУ и особенно важен при обработке соответствующих экспериментальных данных? Чтобы понять это, заменим исходную систему A·x=b эквивалентной системой P·L·U·x=P·b, а ее, в свою очередь, парой других систем: L·y=P·b и U·x=y. Несложно сообразить, что обе системы решаются прямой подстановкой, т. к. матрицы P·L и U — треугольные. Сначала из первой СЛАУ определяется промежуточный вектор y, а затем (из второй системы) — искомый вектор x.

Главное преимущество метода LU-разложения заключается в том, что явный вид вектора правой части b при решении СЛАУ используется только на за-

ключительном этапе (в формулах прямого хода), а наиболее трудоемкие операции по вычислению самих матриц L и U вовсе не требуют знания вектора b. Таким образом, если решается серия СЛАУ с одной и той же матрицей A, но разными правыми частями b (что весьма характерно в обратных задачах интерпретации эксперимента), очень выгодно единожды вычислить LU-разложение матрицы A, а уже затем быстрой подстановкой решить каждую из конкретных систем. Сказанное иллюстрирует листинг 8.21, в котором решается две СЛАУ с матрицей из предыдущего листинга и различными векторами правых частей b.

**Листинг 8.21. Решение СЛАУ при помощи LU-разложения (продолжение листинга 8.20)**

$$b := \begin{pmatrix} 6 \\ 12 \\ 24 \end{pmatrix} \qquad \text{lsolve}(A, b) = \begin{pmatrix} 4 \\ -8 \\ 6 \end{pmatrix}$$

$$y := \text{ltrg}(L, P \cdot b)$$

$$x := \text{trg}(U, y)$$

$$x = \begin{pmatrix} 4 \\ -8 \\ 6 \end{pmatrix}$$

## 8.3.4. *QR*-разложение

Среди матричных разложений особую роль играют ортогональные преобразования, обладающие свойством сохранения нормы вектора. Напомним из курса линейной алгебры, что матрица Q называется *ортогональной*, если $Q^T \cdot Q = I$, где I — единичная матрица. Свойство сохранения нормы вектора при ортогональных преобразованиях, т. е. $|Q \cdot x| = |x|$, дает рецепт поиска псевдорешения вырожденных СЛАУ, а именно, замену исходной задачи минимизации невязки с "плохой" матрицей $|A \cdot x - b|$ задачей $|Q^T \cdot (A \cdot x - b)|$, в которой матрица уже будет "хорошей" благодаря специальному построению матрицы Q. Таким образом, ортогональные разложения используются при решении любых систем (в том числе с прямоугольной матрицей A, причем как переопределенных, так и недоопределенных).

Одним из важнейших вариантов ортогональных разложений некоторой матрицы `A` является *`QR`-разложение* вида `A=Q·R`, где `Q` — ортогональная матрица, а `R` — верхняя треугольная матрица:

❒ `qr(A)` — `QR`-разложение:

- `A` — вектор или матрица любого размера.

Результатом действия функции `qr(A)` является матрица, составленная из матриц `Q` и `R` соответственно (подобно рассмотренному в предыдущем разделе `LU`-разложению). Выделить матрицы `QR`-разложения несложно при помощи встроенной функции выделения подматрицы `submatrix` (листинг 8.22).

**Листинг 8.22. `QR`-разложение сингулярной матрицы**

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

$B := qr(A)$

$$B = \begin{pmatrix} 0.123 & -0.905 & 0.408 & 8.124 & 9.601 & 11.078 \\ 0.492 & -0.302 & -0.816 & 0 & -0.905 & -1.809 \\ 0.862 & 0.302 & 0.408 & 0 & 0 & 0 \end{pmatrix}$$

$$Q := submatrix\left(B, 0, rows(B) - 1, 0, \frac{cols(B) - 1}{2}\right)$$

$$R := submatrix\left(B, 0, rows(B) - 1, \frac{cols(B) + 1}{2}, cols(B) - 1\right)$$

$$Q = \begin{pmatrix} 0.123 & -0.905 & 0.408 \\ 0.492 & -0.302 & -0.816 \\ 0.862 & 0.302 & 0.408 \end{pmatrix} \qquad R = \begin{pmatrix} 8.124 & 9.601 & 11.078 \\ 0 & -0.905 & -1.809 \\ 0 & 0 & 0 \end{pmatrix}$$

$$norme(A - Q \cdot R) = 4.311 \times 10^{-15}$$

Если бы исходная СЛАУ `A·x=b` не была вырожденной, то можно было сразу записать: $Q^T \cdot Q \cdot R \cdot x = Q^T \cdot b$, откуда следует (благодаря ортогональности матрицы `Q`): $R \cdot x = Q^T \cdot b$. Так как матрица `R` — треугольная, решение данной системы получается по формулам прямого хода. Если использовать нашу пользовательскую функцию решения треугольной системы (*см. разд. 8.3.1*), то результат исходной СЛАУ запишется в виде одной строки кода: $trg(R, Q^T \cdot b)$ (соответствующий пример решения невырожденной СЛАУ вы найдете на компакт-диске).

Обратимся теперь к проблеме решения СЛАУ с сингулярной квадратной N×N или прямоугольной N×M матрицей A. Если ее ранг равен k (он может быть меньше N и M, как в листинге 8.22, где N=M=3, а k=2), то получающаяся треугольная матрица R имеет следующую структуру:

$$R = \begin{pmatrix} R1 & R2 \\ 0 & 0 \end{pmatrix},$$

где R1 — верхняя треугольная матрица, а R2 — просто некоторая матрица, а нули обозначают, в общем случае, нулевые матрицы соответствующих размеров.

Если система вырожденная, то она имеет бесконечное множество псевдорешений (векторов, минимизирующих норму невязки). При помощи QR-разложения можно сразу выписать одно из них (правда, не обладающее минимальной нормой). В нашем примере последняя строка матрицы R содержит одни нули, поэтому последняя компонента вектора псевдорешения x может быть абсолютно любой. Если положить $x_2=0$, то остальные компоненты x определятся из треугольной СЛАУ $R1 \cdot x = Q^T \cdot b$, как это проиллюстрировано листингом 8.23.

**Листинг 8.23. Поиск одного из псевдорешений вырожденной СЛАУ посредством QR-разложения (продолжение листинга 8.22)**

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad b := \begin{pmatrix} 6 \\ 12 \\ 24 \end{pmatrix}$$

R1 := submatrix(R, 0, 1, 0, 1)

R2 := submatrix(R, 0, 1, 2, 2)

b1 := submatrix($Q^T \cdot b$, 0, 1, 0, 0)

$$R1 = \begin{pmatrix} 8.124 & 9.601 \\ 0 & -0.905 \end{pmatrix} \qquad R2 = \begin{pmatrix} 11.078 \\ -1.809 \end{pmatrix} \qquad b1 = \begin{pmatrix} 27.326 \\ -1.809 \end{pmatrix}$$

x := stack(trg(R1, $Q^T \cdot b$), 0)

$$x = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}$$

$|A \cdot x - b| = 2.449$ $|x| = 2.236$

Для того чтобы выбрать из всего множества псевдорешений (минимизирующих невязку исходной СЛАУ) нормальное псевдорешение (т. е. обладающее

минимальной нормой), необходимо решить соответствующую задачу минимизации (*см. разд. 8.2.3*). Если построено QR-разложение, сделать это намного легче. Если произвольную компоненту решения обозначить переменной y: $x_2$=y, то она определится из соответствующей задачи оптимизации (листинг 8.24, 1—3 строки), а остальные составляющие самого решения x — из треугольной СЛАУ $R1 \cdot x = Q^T \cdot b - R2 \cdot y$. В последней строке листинга выводится полученное нормальное псевдорешение, а также его норма и соответствующая норма невязки (которые полезно сравнить с результатом прошлого листинга). Вспомогательный рис. 8.13 помогает понять структуру минимизируемой функции из листинга 8.24.

**Листинг 8.24. Нормальное псевдорешение вырожденной СЛАУ (продолжение листингов 8.22 и 8.23)**

$$f(y) := \left|\text{stack}(\text{trg}(R1, b1 - R2 \cdot y), y)\right|$$

$$y := 0$$

$$y := \text{minimize}(f, y) \qquad y = 0.5$$

$$x := \text{stack}(\text{trg}(R1, b1 - R2 \cdot y), y)$$

$$x = \begin{pmatrix} 1.5 \\ 1 \\ 0.5 \end{pmatrix} \qquad \left|A \cdot x - b\right| = 2.449 \qquad \left|x\right| = 1.871$$

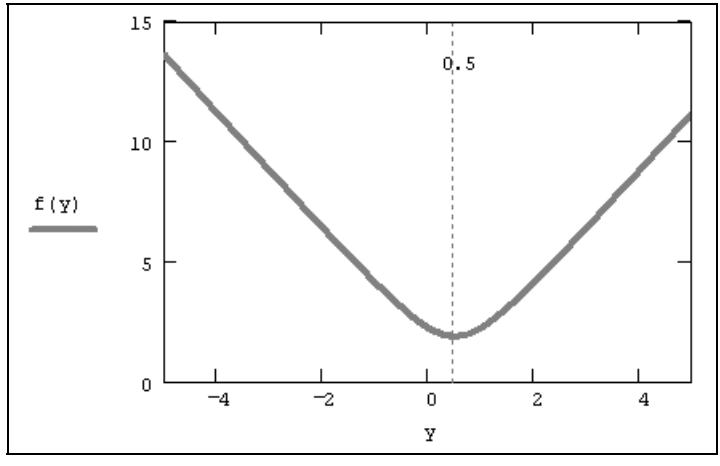



**Рис. 8.13.** Норма псевдорешения в зависимости от y (продолжение листинга 8.24)

Резюмируя практические аспекты применения QR-разложения, надо отметить, что алгоритмы решения СЛАУ на его основе практически одинаковы как для хорошо обусловленных, так и для сингулярных систем.

> **Примечание**
>
> На прилагаемом к книге компакт-диске вы найдете дополнительные примеры построения QR-разложения как для квадратной, так и прямоугольной матрицы A.

## 8.3.5. SVD-(сингулярное) разложение

Наиболее эффективными (но и ресурсоемкими) средствами решения произвольных СЛАУ (с матрицей A размера N×M) по методу наименьших квадратов являются так называемые полные ортогональные разложения, имеющие, по определению, вид $A=U\cdot K\cdot V^T$.

Здесь U и V — ортогональные матрицы размером N×N и M×M соответственно, а K — матрица размера N×M, имеющая следующую структуру:

$$K = \begin{pmatrix} W & 0 \\ 0 & 0 \end{pmatrix}, \tag{8.5}$$

причем W — матрица размера k×k, где k — ранг исходной матрицы A.

Самое известное из ортогональных, SVD- (singular value decomposition) или *сингулярное* разложение — разложение вида $A=U\cdot S\cdot V^T$, где S — диагональная матрица, состоящая из нулей и расположенных на диагонали *сингулярных чисел* матрицы A. Расчет сингулярного разложения в Mathcad осуществляется при помощи пары встроенных функций:

- svds(A) — вектор, состоящий из сингулярных чисел;
- svd(A) — сингулярное разложение:
  - A — действительная матрица.

Пример поиска сингулярного разложения сингулярной матрицы приведен в листинге 8.25, причем его последняя строка представляет собой проверку правильности найденного разложения. Подчеркнем, что вычисленные сингулярные числа находятся на главной диагонали средней матрицы разложения, а ее остальные элементы, по определению, равны нулю. Сравнивая матрицы из листинга 8.25, вы без труда разберетесь, каким образом следует выделять искомые матрицы сингулярного разложения из результата, поставляемого функцией svd.

**Листинг 8.25. Сингулярное разложение сингулярной матрицы**

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

$B := \mathrm{svd}(A)$

$$B = \begin{pmatrix} -0.215 & -0.887 & 0.408 \\ -0.521 & -0.25 & -0.816 \\ -0.826 & 0.388 & 0.408 \\ -0.48 & 0.777 & 0.408 \\ -0.572 & 0.076 & -0.816 \\ -0.665 & -0.625 & 0.408 \end{pmatrix} \qquad \mathrm{svds}(A) = \begin{pmatrix} 16.848 \\ 1.068 \\ 0 \end{pmatrix}$$

$$U := \mathrm{submatrix}\left(B, 0, \frac{\mathrm{rows}(B) - 1}{2}, 0, \mathrm{cols}(B) - 1\right)$$

$$V := \mathrm{submatrix}\left(B, \frac{\mathrm{rows}(B) + 1}{2}, \mathrm{rows}(B) - 1, 0, \mathrm{cols}(B) - 1\right)$$

$S := \mathrm{diag}(\mathrm{svds}(A))$

$$U = \begin{pmatrix} -0.215 & -0.887 & 0.408 \\ -0.521 & -0.25 & -0.816 \\ -0.826 & 0.388 & 0.408 \end{pmatrix} \qquad V = \begin{pmatrix} -0.48 & 0.777 & 0.408 \\ -0.572 & 0.076 & -0.816 \\ -0.665 & -0.625 & 0.408 \end{pmatrix}$$

$$S = \begin{pmatrix} 16.848 & 0 & 0 \\ 0 & 1.068 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\mathrm{norme}\left(U \cdot S \cdot V^{T} - A\right) = 2.878 \times 10^{-15}$$

Венцом современных алгоритмов решения произвольных СЛАУ является SVD-разложение. Прежде чем прокомментировать соответствующий алгоритм, приведенный в листинге 8.26, обратим внимание на матрицу `S`, которая имеет блочную структуру. Согласно формуле (8.5), из матрицы `S` можно выделить подматрицу `W` с ненулевыми диагональными элементами. Дальнейшая последовательность действий по построению нормального псевдорешения выглядит так:

1. Нахождение единственного решения вспомогательной СЛАУ: $W \cdot y = U^{T} \cdot b$ (первые два элемента вектора `y` во второй строке листинга 8.26, которая учитывает диагональность матрицы `S`).

2. Дополнение вектора нулевыми элементами до размера искомого вектора x.

3. Вычисление x простым умножением x=V·y (третья строка листинга).

Полученный результат (искомый вектор x и промежуточный y) выведен в конце листинга 8.26. Как вы можете убедиться, он совпадает с ответом, полученным в листинге 8.24 (*см. предыдущий разд.*) при помощи QR-разложения.

**Листинг 8.26. Решение вырожденной СЛАУ при помощи сингулярного разложения (продолжение листинга 8.25)**

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad b := \begin{pmatrix} 6 \\ 12 \\ 24 \end{pmatrix}$$

$$y := \begin{bmatrix} \dfrac{\left(U^T \cdot b\right)_0}{S_{0,0}} \\ \dfrac{\left(U^T \cdot b\right)_1}{S_{1,1}} \\ 0 \end{bmatrix}$$

$$x := V \cdot y$$

$$x = \begin{pmatrix} 1.5 \\ 1 \\ 0.5 \end{pmatrix} \qquad y = \begin{pmatrix} -1.624 \\ 0.928 \\ 0 \end{pmatrix}$$

# 8.4. Собственные векторы и собственные значения матриц

Завершим главу, посвященную решению СЛАУ, еще одной задачей вычислительной линейной алгебры — задачей отыскания собственных векторов x и собственных значений λ матрицы A, т. е. решения матричного уравнения A·x=λ·x. Такое уравнение имеет решения в виде собственных значений λ1,λ2,… и соответствующих им собственных векторов x1,x2,…. Для решения таких задач на собственные векторы и собственные значения в Mathcad

встроено несколько функций, реализующих довольно сложные вычислительные алгоритмы:

- `eigenvals(A)` — вычисляет вектор, элементами которого являются собственные значения матрицы `A`;
- `eigenvecs(A)` — вычисляет матрицу, содержащую нормированные собственные векторы, соответствующие собственным значениям матрицы `A`. `n`-й столбец вычисляемой матрицы соответствует собственному вектору `n`-го собственного значения, вычисляемого `eigenvals`;
- `eigenvec(A,λ)` — вычисляет собственный вектор для матрицы `A` и заданного собственного значения `λ`:
  - `A` — квадратная матрица.

Применение этих функций иллюстрирует листинг 8.27. В его конце приведена проверка правильности нахождения первого из собственных векторов и собственных значений, причем подстановка результата в выражение `A·x=λ·x` проведена дважды — сначала на числовых значениях `x` и `λ`, а потом путем перемножения соответствующих матричных компонентов.

**Листинг 8.27. Собственные векторы и собственные значения матрицы**

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

$$\text{eigenvals}(A) = \begin{pmatrix} -0.372 \\ 5.372 \end{pmatrix}$$

$$\text{eigenvecs}(A) = \begin{pmatrix} 0.825 & 0.416 \\ -0.566 & 0.909 \end{pmatrix}$$

$$\text{eigenvec}(A, -0.372) = \begin{pmatrix} 0.825 \\ -0.566 \end{pmatrix}$$

$$-0.372 \cdot \begin{pmatrix} 0.825 \\ -0.566 \end{pmatrix} - \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 0.825 \\ -0.566 \end{pmatrix} = \begin{pmatrix} 10 \times 10^{-5} \\ -4.48 \times 10^{-4} \end{pmatrix}$$

$$\text{eigenvals}(A)_0 \cdot \text{eigenvecs}(A)^{\langle 0 \rangle} - A \cdot \text{eigenvecs}(A)^{\langle 0 \rangle} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Помимо рассмотренной проблемы поиска собственных векторов и значений иногда рассматривают более общую задачу, называемую задачей на *обоб-*

*щенные собственные значения*: A·x=λ·B·x. В ее формулировке помимо матрицы A присутствует еще одна квадратная матрица B. Для задачи на обобщенные собственные значения имеются еще две встроенные функции, действие которых аналогично рассмотренным (листинг 8.28):

- genvals(A,B) — вычисляет вектор v собственных значений, каждое из которых удовлетворяет задаче на обобщенные собственные значения;
- genvecs(A,B) — вычисляет матрицу, содержащую нормированные собственные векторы, соответствующие собственным значениям в векторе v, который вычисляется с помощью genvals. В этой матрице i-й столбец является собственным вектором x, удовлетворяющим задаче на обобщенные собственные значения:
  - A,B — квадратные матрицы.

**Листинг 8.28. Поиск обобщенных собственных векторов и собственных значений**

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \qquad B := \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

$$\text{genvals}(A,B) = \begin{pmatrix} 2.686 \\ -0.186 \end{pmatrix}$$

$$\text{genvecs}(A,B) = \begin{pmatrix} 0.416 & 0.825 \\ 0.909 & -0.566 \end{pmatrix}$$

$$2.686 \cdot \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \cdot \begin{pmatrix} 0.416 \\ 0.909 \end{pmatrix} - \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 0.416 \\ 0.909 \end{pmatrix} = \begin{pmatrix} 7.52 \times 10^{-4} \\ -8.52 \times 10^{-4} \end{pmatrix} \blacksquare$$

$$\text{genvals}(A,B)_0 \cdot B \cdot \text{genvecs}(A,B)^{\langle 0 \rangle} - A \cdot \text{genvecs}(A,B)^{\langle 0 \rangle} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

# ЧАСТЬ III

## Дифференциальные уравнения

# Глава 9

# Обыкновенные дифференциальные уравнения: динамические системы

В этой главе рассматриваются численные методы решений задач с начальными условиями (называемых *задачами Коши*) для *обыкновенных дифференциальных уравнений* (далее используется сокращение *ОДУ*). Такие задачи требуют нахождения функции (или нескольких функций) одной переменной, если, во-первых, определено дифференциальное уравнение (или система уравнений), содержащее производную функции, и, во-вторых, необходимое количество дополнительных условий, задающих значение функции в некоторой начальной точке.

Решение задач Коши для ОДУ — давно и детально разработанная технология. С "хорошими" ОДУ вообще никаких вычислительных проблем обычно не возникает (чаще всего они решаются при помощи алгоритма Рунге—Кутты), а для ОДУ особого типа, называемых жесткими, необходимо применять специальные методы. Все эти возможности заложены в Mathcad, причем пользователю позволено выбирать конкретный алгоритм решения ОДУ.

## 9.1. О постановке задач

Начнем разговор об обыкновенных дифференциальных уравнениях с формулировки типичных задач, сопровождая вопросы их постановки конкретными примерами. При этом мы будем, несколько забегая вперед, приводить их решение, не обсуждая особенностей его отыскания средствами Mathcad.

## 9.1.1. Задачи Коши для ОДУ

*Дифференциальные уравнения* — это уравнения, в которых неизвестными являются не переменные (т. е. числа), а функции одной или нескольких переменных. Эти уравнения (или системы) включают соотношения между искомыми функциями и их производными. Если в уравнения входят производные только по одной переменной, то они называются *обыкновенными дифференциальными.* В противном случае говорят об *уравнениях в частных производных* (*см. главу 11*). Таким образом, решить (иногда говорят *проинтегрировать*) дифференциальное уравнение — значит, определить неизвестную функцию на определенном интервале изменения ее переменных.

ОДУ с неизвестной функцией `y(t)`, в которое входят производные этой функции вплоть до $y^{(N)}(t)$, называется *ОДУ N-го порядка.* В частности, уравнение первого порядка может по определению содержать помимо самой искомой функции `y(t)` только ее первую производную `y'(t)`, второго порядка — `y'(t)` и `y''(t)` и т. д. В подавляющем большинстве практических случаев дифференциальное уравнение можно записать в *стандартной форме* (*форме Коши*): `y'(t)=f(y(t),t)`. Уравнение второго порядка может содержать, помимо самой функции, ее первую и вторую производные и т. д. Листинг 9.1 демонстрирует решение простого ОДУ второго порядка, описывающего модель затухающего гармонического осциллятора. Само уравнение приведено во второй строке листинга после задания параметров модели, а вычисленный результат, т. е. искомая функция `y(t)`, показан на рис. 9.1.

***Примечание***

Модель гармонического осциллятора описывает, в частности, колебания маятника: `y(t)` описывает изменения угла его отклонения от вертикали, `y'(t)` — угловую скорость маятника, `y''(t)` — ускорение, а начальные условия, соответственно, начальное отклонение маятника `y(0)=1.0` и начальную скорость `y'(0)=0`. Важно отметить, что модель является линейной, т. е. неизвестная функция (и ее производные) входят в уравнение в первой степени.

**Листинг 9.1. Решение задачи Коши для ОДУ второго порядка (модель затухающего гармонического осциллятора)**

$\omega := 0.5 \qquad \beta := 0.2$

Given $\qquad \omega^2 \cdot \frac{d^2}{dt^2} y(t) + \beta \cdot \frac{d}{dt} y(t) + y(t) = 0$

```
y'(0) = 0
y(0) = 1.0
y := Odesolve(t, 10)
```

Как показывает листинг 9.1, помимо самого уравнения потребовалось определить два начальных условия (третья и четвертая строки листинга) — начальные значения `y(t)` и `y'(t)` при `t=0`. Вообще говоря, ОДУ (или система ОДУ) имеет единственное решение, если помимо уравнения определенным образом заданы *начальные* или *граничные* условия. В соответствующих курсах высшей математики доказываются теоремы о существовании и единственности решения в зависимости от тех или иных условий.

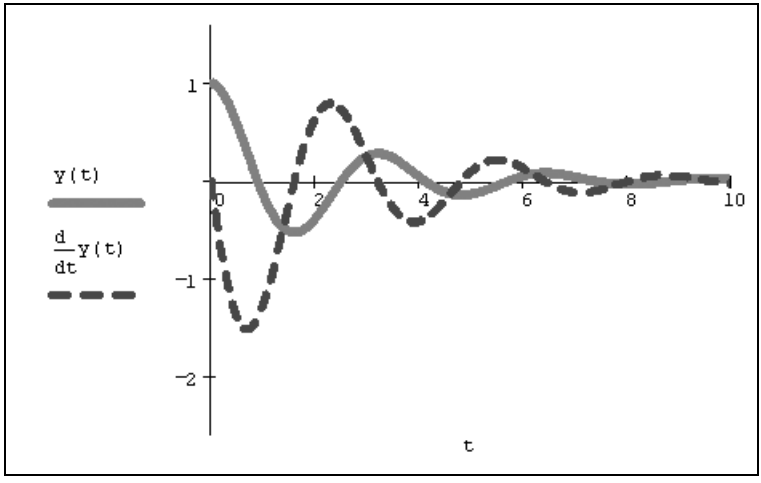



**Рис. 9.1.** Решение уравнения $\omega^2 \cdot y'' + \beta \cdot y' + y = 0$ (продолжение листинга 9.1)

Имеются два типа задач, которые возможно решать с помощью Mathcad:

- *задачи Коши* — для которых определены начальные условия на искомые функции, т. е. заданы значения этих функций в начальной точке интервала интегрирования уравнения (именно им и посвящена данная глава);
- *краевые задачи* — для которых заданы определенные соотношения сразу на обеих границах интервала (они рассматриваются в *главе 10*).

Сказанное относится как к отдельным дифференциальным уравнениям (*см. разд. 9.2*), так и к системам ОДУ (*см. разд. 9.3*). Важно подчеркнуть, что Mathcad умеет решать только такие системы дифференциальных уравнений,

которые могут быть представлены в стандартной форме (форме Коши). Для неизвестных функций $y_0(t), y_1(t), \ldots, y_{N-1}(t)$ система ОДУ должна быть записана в форме:

$$\begin{aligned} y_0'(t) &= f_0(y_0(t), y_1(t), \ldots, y_{N-1}(t), t); \\ y_1'(t) &= f_1(y_0(t), y_1(t), \ldots, y_{N-1}(t), t); \\ &\ldots \\ y_{N-1}'(t) &= f_{N-1}(y_0(t), y_1(t), \ldots, y_{N-1}(t), t), \end{aligned} \qquad (9.1)$$

что эквивалентно следующему векторному представлению:

$$Y'(t) = F(Y(t), t). \qquad (9.2)$$

Здесь `Y` и `Y'` — соответствующие неизвестные векторные функции переменной `t` размера `N×1`, а `F` — векторная функция того же размера и количества переменных `(N+1)` (`N` компонент вектора и, возможно, `t`). Именно векторное представление (9.2) используется для ввода системы ОДУ в среде Mathcad.

Для того чтобы определить задачу Коши для системы из `N` ОДУ первого порядка, следует определить еще ровно `N` начальных условий, задающих значение каждой из функций $y_i(t0)$ в начальной точке интегрирования системы `t0`. В векторной форме они могут быть записаны в виде

$$Y(t0) = B, \qquad (9.3)$$

где `B` — вектор начальных условий размера `N×1`, составленный из $y_i(t0)$. Обратите внимание на необходимость векторной записи как самого уравнения, так и начального условия. В случае одного ОДУ первого порядка соответствующие векторы имеют только один элемент, а в случае системы `N>1` уравнений — `N`.

Стандартные процедуры Mathcad применимы для систем ОДУ первого порядка, записанных в форме (9.2)—(9.3). Тем не менее, если в систему входят и уравнения высших порядков, то ее можно свести к системе большего числа уравнений первого порядка. Рассмотрим в качестве примера уравнение второго порядка модели осциллятора $\omega^2 \cdot y'' + \beta \cdot y' + y = 0$, решенное в листинге 9.1. Если ввести обозначение $y_0(t) \equiv y(t)$, $y_1(t) \equiv y'(t)$, то уравнение сведется к эквивалентной системе:

$$\begin{aligned} y_0(t) &= y_1(t); \\ \omega^2 \cdot y_1' + \beta \cdot y' + y &= 0, \end{aligned}$$

форма которой удовлетворяет форме (9.2) и уравнение может быть решено средствами Mathcad, предназначенными для систем ОДУ. Именно эта система решена ниже в листинге 9.3 (*см. разд. 9.3.1*). Отметим, что на рис. 9.1 показаны как зависимость `y(t)`, так и $y_1(t) = y'(t)$.

## 9.1.2. Фазовый портрет динамической системы

Модели, основанные на задачах Коши для ОДУ, часто называют *динамическими системами*, подчеркивая, что, как правило, они содержат производные по времени `t` и описывают динамику некоторых параметров. Проблемы, связанные с динамическими системами, на самом деле весьма разнообразны и зачастую не сводятся к простому интегрированию ОДУ. Некоторые из них мы обозначим в данном разделе, отметив, что для изучения динамических систем центральным моментом является анализ фазовых портретов, т. е. решений, получающихся при выборе всевозможных начальных условий.

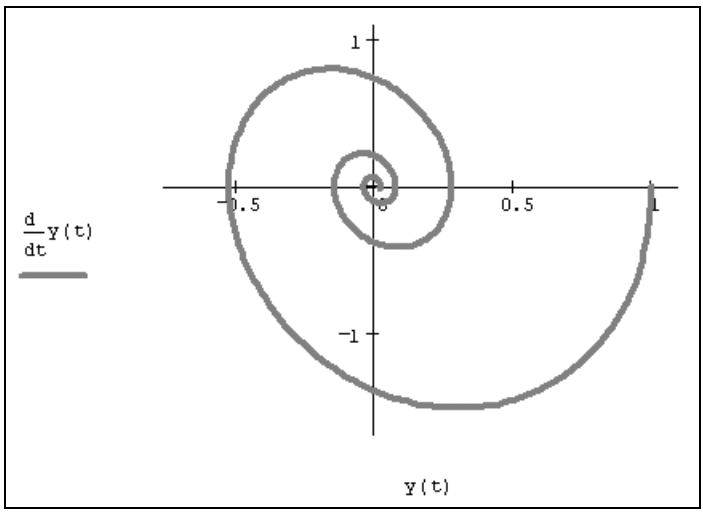



**Рис. 9.2.** Решение уравнения $\omega^2 \cdot y'' + \beta \cdot y' + y = 0$ на фазовой плоскости (продолжение листинга 9.1)

Решение ОДУ часто удобнее изображать не в виде графика $y_0(t), y_1(t), \ldots$, как на рис. 9.1, а в *фазовом пространстве*, по каждой из осей которого откладываются значения каждой из найденных функций. При таком построении графика аргумент `t` будет присутствовать на нем лишь параметрически. В рассматриваемом случае двух ОДУ (мы свели к ним в предыдущем разделе дифференциальное уравнение осциллятора второго порядка) фазовое пространство является координатной плоскостью, а решение представляет собой кривую, или, по-другому, *траекторию*, выходящую из точки, координаты которой равны начальным условиям (рис. 9.2). В общем случае, если система

состоит из N ОДУ, то фазовое пространство является N-мерным. При N>3 наглядность теряется, и для визуализации фазового пространства приходится строить его различные проекции или прибегать к другим специальным приемам (например, отображению Пуанкаре).

Как правило, решение задач Коши для ОДУ и их систем — задача хорошо разработанная и с вычислительной точки зрения довольно простая. На практике чаще встречаются другие, более сложные задачи, в частности, исследование поведения динамической системы в зависимости от начальных условий. При этом в большинстве случаев бывает необходимым изучить только асимптотическое решение ОДУ, т. е. y(t→∞), называемое *аттрактором*. Очень наглядным образом можно визуализировать такую информацию на фазовой плоскости во многом благодаря тому, что существует всего несколько типов аттракторов, и для них можно построить четкую классификацию.

С одной стороны, каждое решение будет выходить из точки, координаты которой являются начальными условиями, но, оказывается, для большинства ОДУ целые семейства траекторий будут заканчиваться в одних и тех же аттракторах (стационарных точках или предельных циклах). Множество решений, вычисленное для всевозможных начальных условий, образует *фазовый портрет* динамической системы. С вычислительной точки зрения задача исследования фазового портрета часто сводится к обычному *сканированию* семейств решений ОДУ при разных начальных условиях.

***Примечание***

Для рассматриваемого примера модели гармонического осциллятора имеется единственная стационарная точка (аттрактор), на которую "накручивается" решение, из каких бы начальных условий оно ни выходило. В теории динамических систем аттрактор такого типа называется *фокусом*.

Дальнейшее усложнение задач анализа фазовых портретов связано с их зависимостью от параметров, входящих в систему ОДУ. В частности, при плавном изменении параметра модели может меняться расположение аттракторов на фазовой плоскости, а также могут возникать новые аттракторы и прекращать свое существование старые. В первом случае, при отсутствии особенностей, будет происходить простое перемещение аттракторов по фазовой плоскости (без изменения их типов и количества), а во втором — фазовый портрет динамической системы будет коренным образом перестраиваться. Критическое сочетание параметров, при которых фазовый портрет системы качественно меняется, называется в теории динамических систем точкой *бифуркации*.

Поясним понятие бифуркации на примере той же модели осциллятора, которая зависит от двух параметров ($\omega$ и $\beta$). При $\beta>0$ существует единственная стационарная точка типа фокуса (рис. 9.2), которая в точке бифуркации $\beta=0$ вырождается в аттрактор типа *центр*, характеризующийся тем, что решения ОДУ представляют собой циклы, совершаемые вокруг этой точки с амплитудой, которая существенно зависит от начальных условий (рис. 9.3). Для надежного исследования фазового портрета практически всегда необходимо решить систему ОДУ большое количество раз с самыми разными начальными условиями (и, возможно, с разным набором параметров модели), чтобы посмотреть, к каким аттракторам сходятся различные траектории.

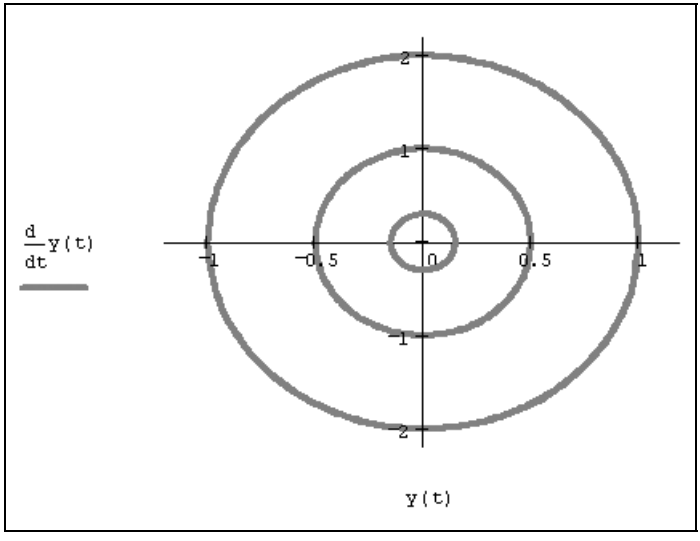



**Рис. 9.3.** Решение уравнения $\omega^2 \cdot y'' + y = 0$ для различных начальных условий (коллаж графиков)

Резюмируя содержание вводного раздела главы, перечислим еще раз типичные постановки задач, характерные для динамических систем:

- решение одной задачи Коши для ОДУ;
- исследование фазового портрета (поиск аттракторов);
- отыскание зависимости положения аттракторов в фазовом пространстве от параметров модели и фиксация бифуркационных значений параметров.

В дальнейших разделах этой главы при рассказе о возможностях Mathcad мы будем в первую очередь описывать решение первой (базовой) задачи, для

которой предусмотрен целый арсенал средств. А именно: вычислительный блок для решения ОДУ (*см. разд. 9.2*), несколько встроенных функций для решения систем ОДУ (*см. разд. 9.3*), в том числе *жестких*, которые не поддаются решению стандартными методами (*см. разд. 9.4*). Приемы, которые автор рекомендует в качестве идиом решения остальных задач, будут рассмотрены эпизодически, на конкретных примерах классических динамических систем вычислительной физики, химии и биологии (*см. разд. 9.5 и 9.4.3*), связанных с динамическими системами. В частности, программа для визуализации фазового портрета рассмотрена в конце главы, на примере модели брюсселятора (*см. разд. 9.5.4*). Сводка алгоритмов с рекомендациями по их применению в зависимости от типа задачи приведена конспективно, без детального разбора (*см. разд. 9.3.4*).

## 9.2. Дифференциальное уравнение *N*-го порядка

Для решения ОДУ порядка `N≥1` в Mathcad предусмотрены две возможности:

- вычислительный блок `Given/Odesolve` (начиная с версии 2000) — в этом случае решение имеет вид функции от `t`;
- встроенные функции решения систем ОДУ, причем уравнения высших порядков необходимо предварительно свести к эквивалентной системе уравнений первого порядка, как об этом рассказано в *разд. 9.1.1*, — в этом случае решение имеет формат вектора.

В данном разделе рассматривается первый из вариантов, а второй будет описан в следующем, применительно к общему случаю системы `N` уравнений. Заметим лишь, что первый путь предпочтительнее из соображений наглядности представления задачи и результатов, а второй дает пользователю больше рычагов воздействия на параметры численного метода.

Вычислительный блок для решения ОДУ, реализующий численный метод Рунге—Кутты, состоит из трех частей:

- `Given` — ключевое слово;
- ОДУ и начальные условия в формате `y(t0)=b`, записанные с помощью логических операторов, которые должны набираться на панели инструментов **Boolean** (Булевы операторы);
- `Odesolve(t,t1)` — встроенная функция для решения ОДУ относительно переменной `t` на интервале `(t0,t1)`, причем `t0<t1`.

### Примечание 1

На запись ОДУ в пределах вычислительного блока накладывается несколько ограничений. Во-первых, ОДУ должно быть линейно относительно старшей производной, т. е. фактически должно быть поставлено в стандартной форме. Во-вторых, начальные условия должны иметь форму `y(t0)=b` или $y^{(N)}(t0)=b$, а не более сложную (как, например, встречающаяся в некоторых математических приложениях форма `y(t0)+y'(t0)=b`).

### Примечание 2

Допустимо, и даже часто предпочтительнее, задание функции `Odesolve(t,t1,step)` с тремя параметрами, где `step` — внутренний параметр численного метода, определяющий количество шагов, в которых метод Рунге—Кутты будет рассчитывать решение дифференциального уравнения. Чем больше `step`, тем с лучшей точностью будет получен результат, но тем больше времени будет затрачено на его поиск. Помните, что подбором этого параметра можно заметно (в несколько раз) ускорить расчеты без существенного ухудшения их точности.

Пример решения задачи Коши для ОДУ второго порядка, являющегося усложнением модели из листинга 9.1 на нелинейный случай (с параметром квадратичной нелинейности $\gamma$), приведен в листинге 9.2, а результат — на рис. 9.4. Учтите, что символ производной допускается вводить как средствами панели **Calculus** (Вычисления), как это сделано в листинге 9.1, так и в виде штриха, набрав его с помощью сочетания клавиш <Ctrl>+<F7> (как в листинге 9.2). Выбирайте тот или иной способ представления производной из соображений наглядности представления результатов — на ход расчетов он не влияет.

Еще раз подчеркнем, что результатом применения блока `Given/Odesolve` является функция `y(t)`, определенная на промежутке `(t0,t1)`. Следует воспользоваться обычными средствами Mathcad, чтобы построить ее график или получить значение функции в какой-либо точке указанного интервала, например: `y(10)=0.048`.

**Листинг 9.2. Решение задачи Коши для ОДУ второго порядка (модель нелинейного осциллятора)**

$\omega := 0.5 \quad \beta := 0.2 \quad \gamma := 0.95$

Given

$\omega^2 \cdot y''(t) + \beta \cdot y'(t) + y(t) + \gamma \cdot y(t)^2 = 0$

$y(0) = 0 \qquad y'(0) = 3$

$y := \text{Odesolve}(t, 20)$

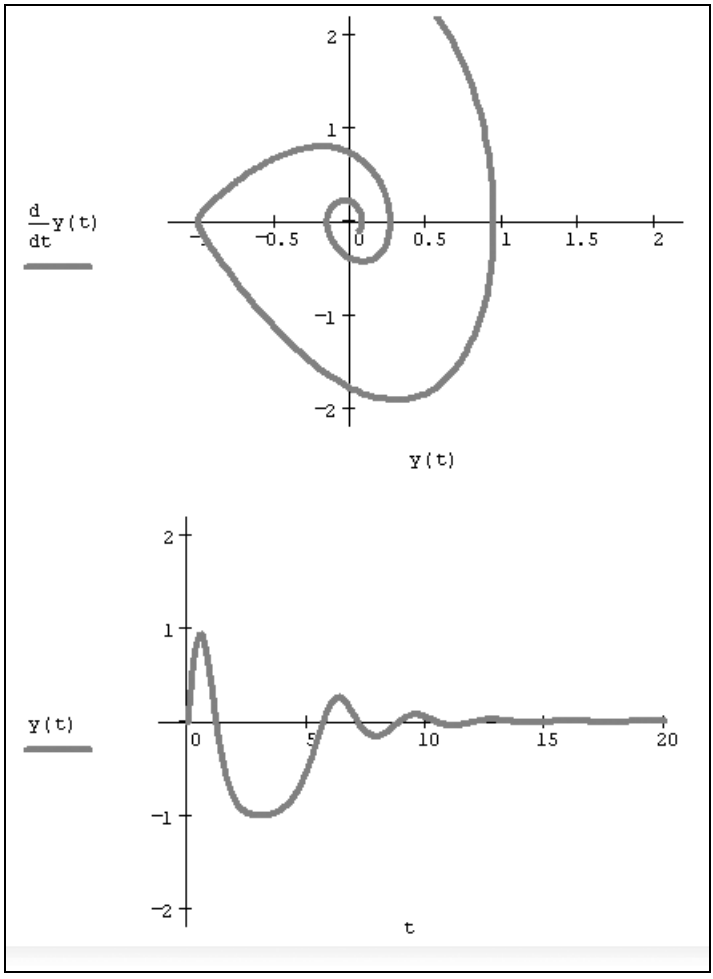



**Рис. 9.4.** Решение уравнения $\omega^2 \cdot y'' + \beta \cdot y' + y + \gamma \cdot y^2 = 0$
(продолжение листинга 9.2)

Пользователь имеет возможность выбирать между двумя модификациями численного метода Рунге—Кутты. Для смены метода необходимо нажатием правой кнопки мыши на области функции `Odesolve` вызвать контекстное меню и выбрать в нем один из трех пунктов: **Fixed** (С фиксированным шагом), **Adaptive** (Адаптивный) или **Stiff** (Для жестких ОДУ). Подробнее о различиях этих методов сказано в *разд. 9.3.4* и *9.4*.

# 9.3. Система *N* дифференциальных уравнений

При помощи Mathcad можно решать системы `N≥1` ОДУ первого порядка, если они записаны в стандартной форме (Коши) в виде векторного соотношения: `Y'(t)=F(Y(t),t)` (*см. разд. 9.1.1*).

## 9.3.1. Встроенные функции для решения систем ОДУ

В Mathcad имеется несколько встроенных функций, которые позволяют решать задачу Коши различными численными методами. Для "хороших" нежестких систем ОДУ применяются следующие функции:

- `rkfixed(y0,t0,t1,M,D)` — метод Рунге—Кутты с фиксированным шагом;
- `Rkadapt(y0,t0,t1,M,D)` — метод Рунге—Кутты с переменным шагом;
- `Bulstoer(y0,t0,t1,M,D)` — метод Булирша—Штера:
  - `y0` — вектор начальных значений в точке `t0` размера `N×1`;
  - `t0` — начальная точка расчета;
  - `t1` — конечная точка расчета;
  - `M` — число шагов, на которых численный метод находит решение;
  - `D` — векторная функция размера `N×1` двух аргументов — скалярного `t` и векторного `y`. При этом `y` — искомая векторная функция аргумента `t` того же размера `N×1`.

> ***Внимание!***
>
> Соблюдайте регистр первой буквы рассматриваемых функций, поскольку это влияет на выбор алгоритма счета, в отличие от многих других встроенных функций Mathcad, например, `Find≡find` (*см. разд. 9.3.2*).

Каждая из приведенных функций выдает решение в виде матрицы размера `(M+1)×(N+1)`. В ее левом столбце находятся значения аргумента `t`, делящие интервал на равномерные шаги, а в остальных `N` столбцах — значения искомых функций $y_0(t), y_1(t), \ldots, y_{N-1}(t)$, рассчитанные для этих значений аргумента (рис. 9.5). Поскольку всего точек (помимо начальной) `M`, то строк в матрице решения будет всего `M+1`.

В листинге 9.3 приведен пример решения той же самой системы ОДУ осциллятора с затуханием (*см. разд. 9.1 и 9.2*) при помощи первой из функций `rkfixed`. Результат расчетов представлен на рис. 9.5 как содержимое матри-

цы и на рис. 9.6 в виде графика. Точки, в которых получено решение, отмечены на графике рис. 9.6 кружками. Чтобы использовать другой численный алгоритм, достаточно поменять имя функции `rkfixed` в последней строке листинга на другое (на практике как раз более эффективны функции `Rkadapt` и `Bulstoer`).

**Листинг 9.3. Решение системы двух ОДУ (модель осциллятора)**

$\omega := 0.5 \qquad \beta := 0.2$

$y0 := \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

$M := 50$

$D(t, y) := \begin{pmatrix} y1 \\ -\omega \cdot y0 - \beta \cdot y1 \end{pmatrix}$

$u := rkfixed(y0, 0, 40, M, D)$

u =

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 1 | 0.8 | 0.852 | -0.35 |
| 2 | 1.6 | 0.482 | -0.548 |
| 3 | 2.4 | 0.027 | -0.559 |
| 4 | 3.2 | -0.368 | -0.408 |
| 5 | 4 | -0.599 | -0.162 |
| 6 | 4.8 | -0.624 | 0.095 |
| 7 | 5.6 | -0.466 | 0.286 |
| 8 | 6.4 | -0.197 | 0.367 |
| 9 | 7.2 | 0.089 | 0.33 |
| 10 | 8 | 0.307 | 0.204 |
| 11 | 8.8 | 0.404 | 0.038 |
| 12 | 9.6 | 0.371 | -0.114 |
| 13 | 10.4 | 0.236 | -0.211 |
| 14 | 11.2 | 0.054 | -0.233 |
| 15 | 12 | -0.118 | -0.185 |

**Рис. 9.5.** Результат, выдаваемый встроенной функцией в качестве решения системы ОДУ (продолжение листинга 9.3)

Первая строка листинга представляет задание параметров модели, вторая — начального условия задачи Коши, а в третьей строке листинга определено число шагов, на которых рассчитывается решение. Самая важная — это предпоследняя строка листинга, в которой, собственно, определяется система ОДУ. Последняя строка присваивает матричной переменной `u` результат действия функции `rkfixed`. Решение системы ОДУ будет осуществлено на промежутке `(0,40)`.

Сравните рассматриваемую систему (*см. разд. 9.1.1*), записанную в стандартной форме с формальной ее записью в Mathcad, чтобы не делать впоследствии ошибок. Во-первых, функция `D`, входящая в число параметров встроенных функций для решения ОДУ, должна быть функцией обязательно двух аргументов. Во-вторых, второй ее аргумент должен быть вектором того же размера, что и сама функция `D`. В-третьих, точно такой же размер должен быть и у вектора начальных значений `y0`. Не забывайте, что векторную функцию `D(t,y)` следует определять через компоненты вектора `y` с помощью кнопки **Subscript** (Нижний индекс) с наборной панели **Calculator** (Калькулятор) или нажатием клавиши <[>.

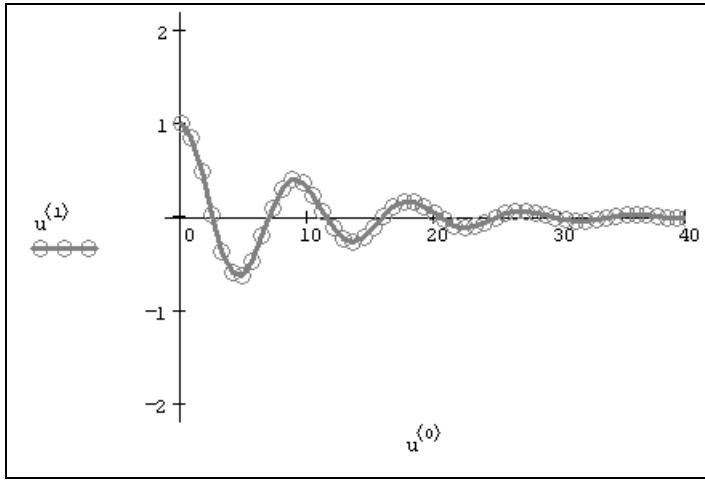



**Рис. 9.6.** График решения системы ОДУ осциллятора (продолжение листинга 9.3)

Матрица, представляющая решение, показана на рис. 9.5. Размер полученной матрицы будет равен `(M+1)×(N+1)`, т. е. `51×3`. Просмотреть все компоненты матрицы `u`, которые не помещаются на экране, можно с помощью вертикальной полосы прокрутки. Как нетрудно сообразить, на этом рисунке отмечены

выделением расчетные значения искомых векторов на 10-м шаге. Это соответствует, с математической точки зрения, найденным значениям $y_0(8.0)=0.307$ и $y_1(8.0)=0.204$. Для вычисления элементов решения в последней точке интервала используйте вывод элемента $u_{M,1}$. Для построения графика решения надо отложить соответствующие компоненты матрицы решения по координатным осям: значения аргумента $u^{<0>}$ — вдоль оси X, а $u^{<1>}$ и $u^{<2>}$ — вдоль оси Y (рис. 9.6).

> ***Внимание!***
>
> Обратите внимание на некоторое разночтение в обозначении индексов вектора начальных условий и матрицы решения. В ее первом столбце собраны значения нулевой компоненты искомого вектора, во втором столбце — первой компоненты и т. д.

## 9.3.2. Решение одного уравнения (*N*=1)

Метод решения ОДУ при помощи встроенных функций `rkfixed`, `Rkadapt` или `Bulstoer` (в противоположность вычислительному блоку `Given/Odesolve`) сохранился с прежних версий Mathcad (до 2000-й). В большинстве случаев лучше использовать вычислительный блок `Given/Odesolve`, который выигрывает в простоте и в наглядности, однако иногда предпочтительнее решать ОДУ первого порядка с помощью второго способа, например, при следующих обстоятельствах:

- вы работаете одновременно с более старыми (до 2001-й включительно) версиями Mathcad и хотите, чтобы ваши документы воспринимались каждой из них корректно;
- одно ОДУ решается в контексте решения более сложных задач, в которые входят системы дифференциальных уравнений (для которых вычислительный блок неприменим) — в этом случае может потребоваться единый стиль программирования;
- ответ предпочтительнее получить в виде вектора, а не функции;
- вы привыкли к записи ОДУ в старых версиях Mathcad, у вас много документов, созданных с их помощью, и т. п.

Поскольку решение вторым способом одного ОДУ не отличается от решения систем ОДУ (*см. предыдущий разд.*), приведем пример его использования (листинг 9.4) практически без комментариев. Отметим лишь в случае одного ОДУ, что как само уравнение, так и начальное условие можно задавать не в векторной, а в скалярной форме. Результат выдается в виде матрицы размерности `M×2`, которая состоит из двух столбцов: в одном находятся значения

аргумента t (от t0 до t1 включительно), а в другом соответствующие значения искомой функции y(t).

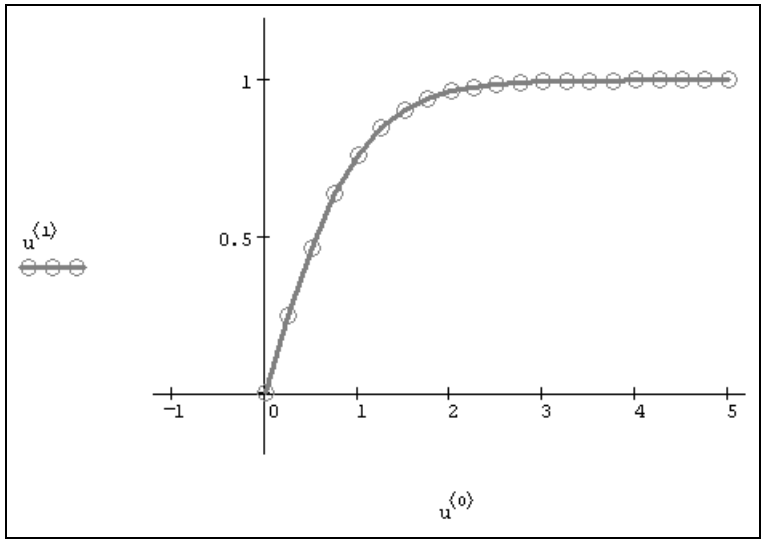



**Рис. 9.7.** Решение уравнения $y'=1-y^2$ (продолжение листинга 9.4)

Построение графика (рис. 9.7) осуществляется так же, как и в рассмотренном в предыдущем разделе случае N уравнений, при помощи выделения столбцов из матрицы решения посредством оператора <>.

**Листинг 9.4. Решение задачи Коши для ОДУ первого порядка**

```
y0 := 0
M := 20
D(t, y) := 1 - y^2
u := rkfixed(y0, 0, 5, M, D)
```

## 9.3.3. Решение систем ОДУ в одной заданной точке

Зачастую при решении дифференциальных уравнений требуется определить значения искомых функций не на всем интервале (t0,t1), а только в одной

его последней точке. Например, весьма распространены задачи поиска аттракторов динамических систем. Известно, что для широкого класса ОДУ одна и та же система при разных (или даже любых, как рассмотренный выше пример осциллятора с затуханием) начальных условиях при $t \to \infty$ приходит в одну и ту же точку (аттрактор). Поэтому часто нужно определить именно эту точку.

Такая задача требует меньше ресурсов компьютера, чем решение системы ОДУ на всем интервале, поэтому в Mathcad имеются модификации встроенных функций `Rkadapt` и `Bulstoer`. Они имеют несколько другой набор параметров и работают значительно быстрее своих аналогов:

- `rkadapt(y0,t0,t1,acc,D,k,s)` — метод Рунге—Кутты с переменным шагом;
- `bulstoer(y0,t0,t1,acc,D,k,s)` — метод Булирша—Штера:
  - `y0` — вектор начальных значений в точке `t0`;
  - `t0`,`t1` — начальная и конечная точки расчета;
  - `acc` — погрешность вычисления (чем она меньше, тем с лучшей точностью будет найдено решение; рекомендуется выбирать значения погрешности в районе `0.001`);
  - `D` — векторная функция, задающая систему ОДУ;
  - `k` — максимальное число шагов, на которых численный метод будет находить решение;
  - `s` — минимально допустимая величина шага.

Как легко заметить, вместо числа шагов на интервале интегрирования ОДУ в этих функциях необходимо задать точность расчета численным методом значения функций в последней точке. В этом смысле параметр `acc` похож на константу `TOL`, которая влияет на большинство встроенных численных алгоритмов Mathcad. Количество шагов и их расположение определяются численным методом автоматически, чтобы обеспечить эту точность. Два последних параметра нужны для того, чтобы пользователь мог искусственно повлиять на разбиение интервала на шаги. Параметр `k` служит для того, чтобы шагов не было чрезмерно много, причем нельзя сделать `k>1000`. Параметр `s` — для того, чтобы ни один шаг не был слишком малым для появления больших погрешностей при разностной аппроксимации дифференциальных уравнений внутри алгоритма. Эти параметры следует задавать явно, исходя из свойств конкретной системы ОДУ. Как правило, проведя ряд тестовых расчетов, можно подобрать их оптимальный набор для каждого конкретного случая.

Пример использования функции bulstoer для той же модели линейного осциллятора приведен в листинге 9.5. В его первых двух строках, как обычно, определяется система уравнений и начальные условия; в следующей строке матрице u присваивается решение, полученное с помощью bulstoer. Структура этой матрицы в точности такая же, как и в случае решения системы ОДУ посредством уже рассмотренных нами ранее встроенных функций (*см. разд. 9.3.1*). Однако в данном случае нам интересна только последняя точка интервала. Поскольку сделанное численным методом количество шагов, т. е. размер матрицы u, заранее неизвестно, то его необходимо предварительно определить. Это сделано в четвертой строке листинга, присваивающей это число переменной M, в этой же строке оно выведено на экран.

В предпоследней строке листинга осуществлен вывод решения системы ОДУ на конце интервала, т. е. в точке t=50 в виде вектора. В последней строке для примера еще раз выводится искомое значение первой функции из системы ОДУ (сравните его с соответствующим местом вектора из предыдущей строки). Чтобы попробовать альтернативный численный метод, достаточно в листинге 9.5 заменить имя функции bulstoer на rkadapt.

**Листинг 9.5. Поиск аттрактора системы двух ОДУ модели осциллятора**

$$y0 := \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$M := 50$$

$$D(t, y) := \begin{pmatrix} y_1 \\ -0.5 \cdot y_0 - 0.2 \cdot y_1 \end{pmatrix}$$

$$u := \text{bulstoer}\left(y0, 0, 300, 10^{-5}, D, M, 0.01\right)$$

$$N := \text{rows}(u) - 1$$

$$\left(u^{T}\right)^{\langle N \rangle} = \begin{pmatrix} 300 \\ -7.645 \times 10^{-14} \\ -3.128 \times 10^{-14} \end{pmatrix}$$

***Внимание!***

Функции bulstoer и rkadapt (те, что пишутся со строчной буквы) не предназначены для нахождения решения в промежуточных точках интервала, хотя они и выдают их в матрице-результате.

## ⓘ 9.3.4. О численных методах

В завершение раздела сделаем несколько важных замечаний относительно выбора численного алгоритма решения ОДУ и задания его параметров. Они не претендуют на общность, но, надеемся, будут весьма полезны читателю, особенно в случае возникновения проблем.

### ⓘ Рекомендации по выбору численного алгоритма

Все численные методы решения ОДУ основаны на аппроксимации дифференциальных уравнений разностными аналогами. В зависимости от конкретной формы аппроксимации получаются алгоритмы различной точности и быстродействия. В Mathcad использован наиболее популярный алгоритм Рунге—Кутты четвертого порядка, описанный в большинстве книг по методам вычислений. Он обеспечивает малую погрешность для широкого класса систем ОДУ за исключением жестких систем. Поэтому в большинстве случаев стоит применять функцию `rkfixed`. Если по различным причинам время расчетов становится критичным или точность неудовлетворительна, стоит попробовать вместо `rkfixed` другие функции, тем более, что сделать это очень просто благодаря одинаковому набору параметров. Для этого нужно только поменять имя функции в программе.

Функция `Rkadapt` может быть полезна в случае, когда известно, что решение на рассматриваемом интервале меняется слабо либо существуют участки медленных и быстрых его изменений. Метод Рунге—Кутты с переменным шагом разбивает интервал не на равномерные шаги, а более оптимальным способом. Там, где решение меняется слабо, шаги выбираются более редкими, а в областях его сильных изменений — частыми. В результате для достижения одинаковой точности требуется меньшее число шагов, чем для `rkfixed`. Метод Булирша—Штера `Bulstoer` часто оказывается более эффективным для поиска гладких решений.

Имея в виду сделанные замечания, приведем краткую сводку алгоритмов решения задач Коши для ОДУ, отмечая, какие из встроенных функций следует использовать в конкретных случаях.

- Для решения единственного уравнения (любого порядка) используйте вычислительный блок `Given/Odesolve`.
- Для стандартных нежестких систем используйте алгоритм Булирша—Штера `Bulstoer`.

- Для систем с участками быстро и медленно меняющихся решений используйте адаптивный алгоритм Рунге—Кутты `Rkadapt`.
- В учебных целях и для решения несложных задач можно использовать алгоритм Рунге—Кутты с фиксированным шагом `Rkfixed`.
- Для получения решения в одной конечной точке интервала используйте (в зависимости от перечисленных классов задач) одну из встроенных функций с именем, начинающимся со строчной буквы.
- Для жестких систем (*см. разд. 9.4*) используйте функции `Radau` `Stiffb` или `Stiffr`, причем если вы затрудняетесь с вводом якобиана, применяйте первую из них).

Мы не будем в этой книге приводить описание численных алгоритмов, поскольку им посвящено огромное количество книг, и читатель, несомненно, найдет соответствующую литературу с подробным объяснением их работы. Тем не менее разберем несколько важных моментов, касающихся выбора алгоритма и его параметров.

## ⓘ Число шагов

Все встроенные функции требуют задания для численного метода количества шагов, разбивающих интервал интегрирования ОДУ. Это очень важный параметр, непосредственно влияющий на погрешность результата и скорость расчетов. При решении систем ОДУ многие проблемы могут быть устранены при помощи простой попытки увеличить число шагов численного метода. В частности, сделайте так при неожиданном возникновении ошибки "Found a number with a magnitude greater than 10^307" (Найдено число, превышающее значение $10^{307}$). Данная ошибка не обязательно говорит о том, что решение в действительности расходится, а возникает из-за недостатка шагов для корректной работы численного алгоритма.

В качестве иллюстрации приведем результаты простых расчетов (листинг 9.6), в которых определяется пользовательская функция `u(M)`, представляющая решение ОДУ в зависимости от числа шагов `M`. На рис. 9.8 показано несколько графиков решения системы ОДУ затухающего линейного осциллятора для нескольких значений `M`.

Как можно заметить, несмотря на, в целом, правильное решение ОДУ в узлах (оно показано кружками и квадратами), профиль решения при малых `M` получается неверным. Еще лучше это видно на рис. 9.9, представляющем фазовый портрет системы с разным числом шагов. Видно, что для `M=200`

решение получается более гладким (конечно, при этом увеличивается и время расчетов).

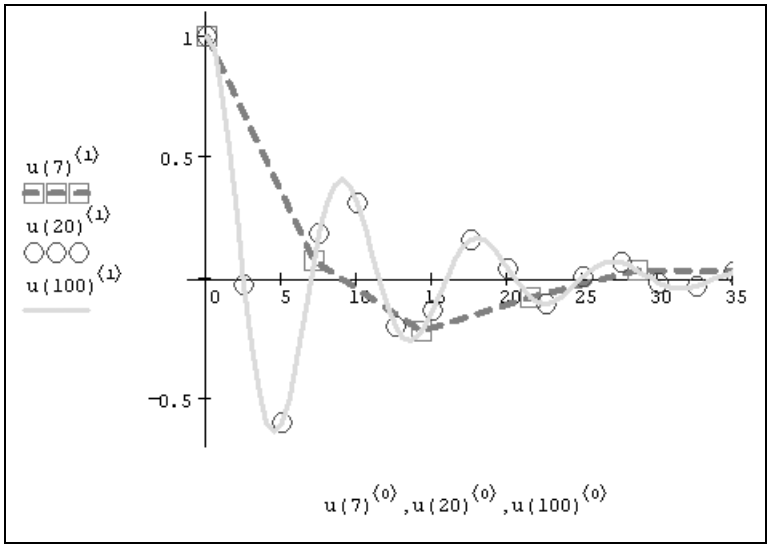



**Рис. 9.8.** Решение системы ОДУ в зависимости от числа узлов `M` (продолжение листинга 9.6)

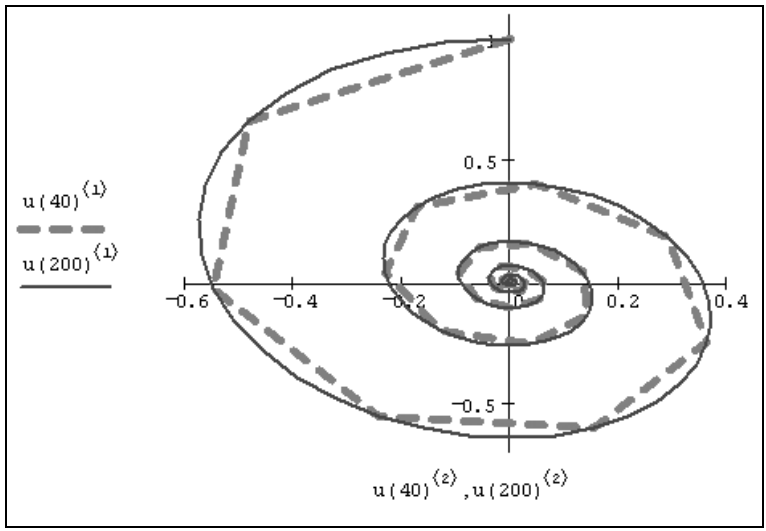



**Рис. 9.9.** Фазовый портрет решения системы ОДУ при `M=40` и `M=200` (продолжение листинга 9.6)

**Листинг 9.6. Решение системы ОДУ как функция числа шагов**

```
D(t, y) := [ y1
             -0.5·(y0) - 0.2·y1 ]

u(M) := Bulstoer[ (1
                   0 ), 0, 50, M, D ]
```

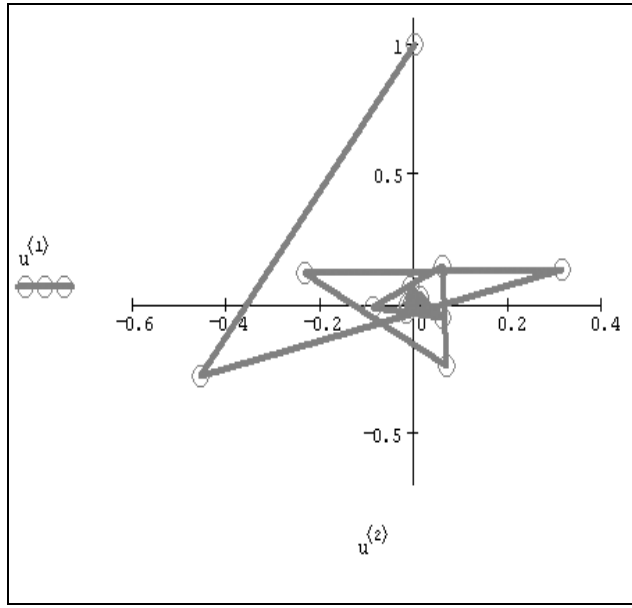
## Погрешность алгоритмов решения ОДУ в точке

Обратимся теперь к функциям `bulstoer` и `rkadapt` (тем, что пишутся со строчной буквы), которые предназначены для нахождения решения в определенной точке интервала интегрирования системы ОДУ. В первую очередь, подчеркнем еще раз, что их нельзя применять для получения решения в промежуточных точках интервала, несмотря на их вывод в матрице-результате. На рис. 9.10 показан фазовый портрет системы ОДУ динамической модели осциллятора, полученный при помощи функции `bulstoer` в листинге 9.5 (*см. разд. 9.3.3*) и с помощью `rkadapt` (при соответствующей замене третьей строки листинга 9.5). Видно, что, несмотря на высокую точность, равную $10^{-5}$, и верный результат на конце интервала, график, полученный при помощи функции `bulstoer` (рис. 9.10, *а*), мало напоминает правильный фазовый портрет (см. рис. 9.9), начиная быть приемлемым только при предельно допустимом для обсуждаемых функций значении `acc`=$10^{-16}$. В данном случае конкретной системы ОДУ функция `rkadapt` дает качественно верное решение (рис. 9.10, *б*), однако его необходимая точность обеспечивается только в последней точке временного интервала.

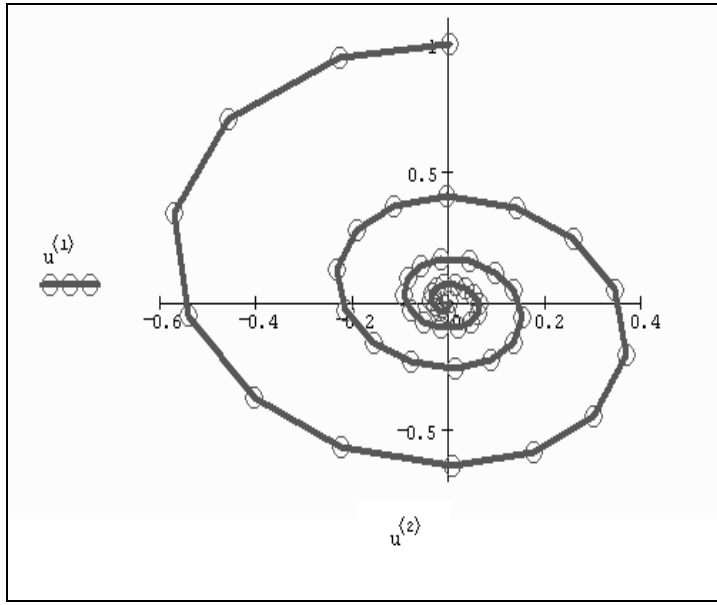
В заключение остановимся на влиянии выбора параметра `acc` на расчеты. Как правило, разные численные методы работают несколько по-разному. Алгоритм Рунге—Кутты дает результат тем ближе к истинному, чем меньше выбирается параметр `acc`, а Булирша—Штера часто демонстрирует менее естественную зависимость `y(ε)`: даже при относительно больших ε реальная погрешность остается хорошей (намного лучше метода Рунге—Кутты).

Поэтому для экономии времени расчетов в функции `bulstoer` можно выбирать и большие `acc`.

Чтобы обеспечить заданную точность, данные алгоритмы могут изменять как количество шагов, разбивающих интервал `(t0,t1)`, так и их расположение вдоль интервала. Чтобы выяснить, на сколько шагов разбивался интервал при расчетах, воспользуемся простой программой, представленной в листинге 9.7.




*а*




*б*

**Рис. 9.10.** Фазовый портрет, полученный `bulstoer` (*а*) и `rkadapt` (*б*) (продолжение листинга 9.5)

В ней определены пользовательские функции `R(ε)` и `B(ε)`, вычисляющие для конкретной задачи (методов Рунге—Кутты и Булирша—Штера соответственно) зависимость числа шагов (т. е. числа строк получающейся матрицы) от параметра ε=`acc`. Графики функций `R(ε)` и `B(ε)` показаны на рис. 9.11. Как видно, методу Рунге—Кутты при увеличении требуемой точности требуется все возрастающее количество шагов (и, соответственно, времени на расчеты), в противоположность методу Булирша—Штера, демонстрирующему бóльшую экономичность.

### *Примечание*

Максимальное количество шагов алгоритма ограничивается еще одним параметром `k` встроенных функций `bulstoer` и `rkadapt` (*см. разд. 9.3.3*).

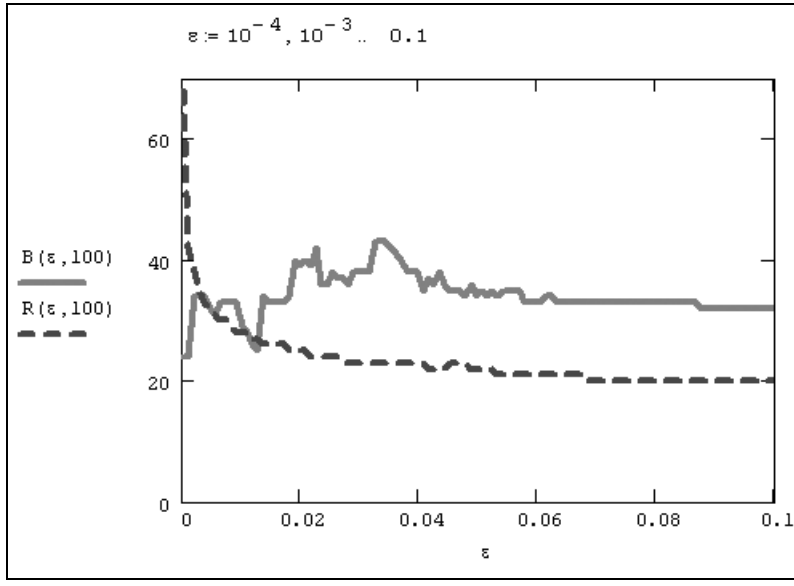



**Рис. 9.11.** Зависимость числа шагов от параметра `acc` численных методов (продолжение листинга 9.7)

**Листинг 9.7. Число шагов адаптивного численного алгоритма в зависимости от погрешности `acc`**

$$D(t, y) := \begin{pmatrix} y_1 \\ -y_0 - 0.1 \cdot y_1 \end{pmatrix}$$

$$k := 300$$

$$B(\varepsilon) := \text{rows}\left[\text{bulstoer}\left[\begin{pmatrix}0.1\\0\end{pmatrix}, 0, 50, \varepsilon, D, k, 0.001\right]\right]$$

$$R(\varepsilon) := \text{rows}\left[\text{rkadapt}\left[\begin{pmatrix}0.1\\0\end{pmatrix}, 0, 50, \varepsilon, D, k, 0.001\right]\right]$$

Таким образом, проводя тестовые расчеты для различных задач и подбирая наилучший набор параметров, можно существенно сэкономить ресурсы компьютера. Конечно, проводить подобный анализ стоит в случаях, когда время расчетов для вас критично.

# 9.4. Жесткие системы ОДУ

До сих пор мы имели дело с "хорошими" уравнениями, которые надежно решались численными методами Рунге—Кутты. Однако имеется класс так называемых *жестких* (stiff) систем ОДУ, для которых стандартные методы практически неприменимы, поскольку их решение требует исключительно малого значения шага численного метода. Некоторые из специальных алгоритмов, разработанных для этих систем, реализованы в Mathcad.

## 9.4.1. Что такое жесткие ОДУ?

Исторически интерес к жестким системам возник в середине XX в. при изучении уравнений химической кинетики с одновременным присутствием очень медленно и очень быстро протекающих химических реакций (оно будет рассмотрено в *разд. 9.4.3*). Тогда неожиданно оказалось, что считавшиеся исключительно надежными методы Рунге—Кутты стали давать сбой при расчете этих задач.

Строгого общепринятого математического определения жестких ОДУ нет. В рамках этой книги будем считать, что жесткие системы — это те уравнения, решение которых получить намного проще с помощью определенных неявных методов, чем с помощью явных методов (типа тех, что были рассмотрены в предыдущих разделах). Примерно такое определение было предложено в 1950-х гг. классиками в этой области Кертиссом и Хиршфельдером. Начнем разговор о жестких ОДУ с примера нежесткого уравнения (листинг 9.8), решение которого удается получить численным методом Рунге—Кутты (рис. 9.12).

**Листинг 9.8. Решение нежесткого ОДУ**

```
Given

d/dt y(t) = -10·(y(t) - cos(t))

y(0) = 1

y := Odesolve(t, 1)
```

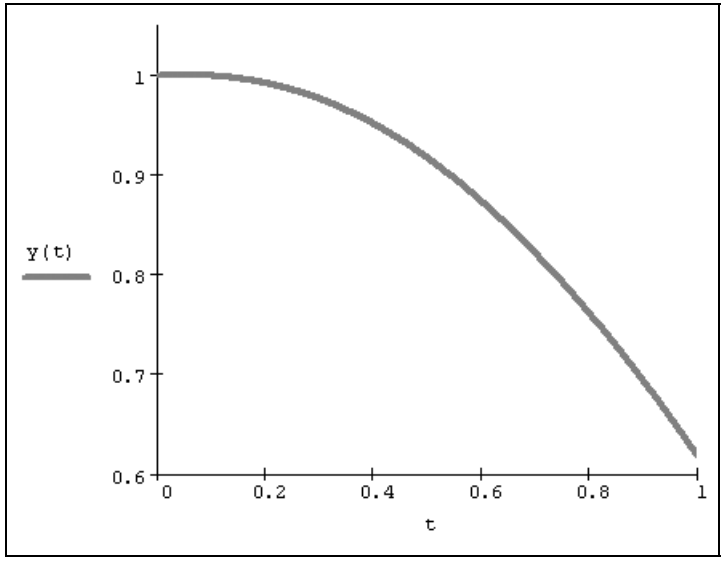



**Рис. 9.12.** Решение нежесткого ОДУ методом Рунге—Кутты (продолжение листинга 9.8)

Обратите внимание на значение коэффициента `-10` во второй строке листинга. Если изменить его на `-50`, то решение меняется до неузнаваемости (рис. 9.13). Вас, несомненно, должен насторожить результат, выданный Mathcad. Характерная "разболтка" решения говорит о неустойчивости алгоритма. Первое, что можно сделать, — увеличить количество шагов в методе Рунге—Кутты. Для этого достаточно добавить третий параметр `step` в функцию `Odesolve(t,1,step)`. После нескольких экспериментов можно подобрать такое значение `step`, которое будет обеспечивать устойчивость решения. Читатель может самостоятельно убедиться, что при `step>50` "разболтка" пропадает, и решение принимает вид графика, показанного на рис. 9.14.

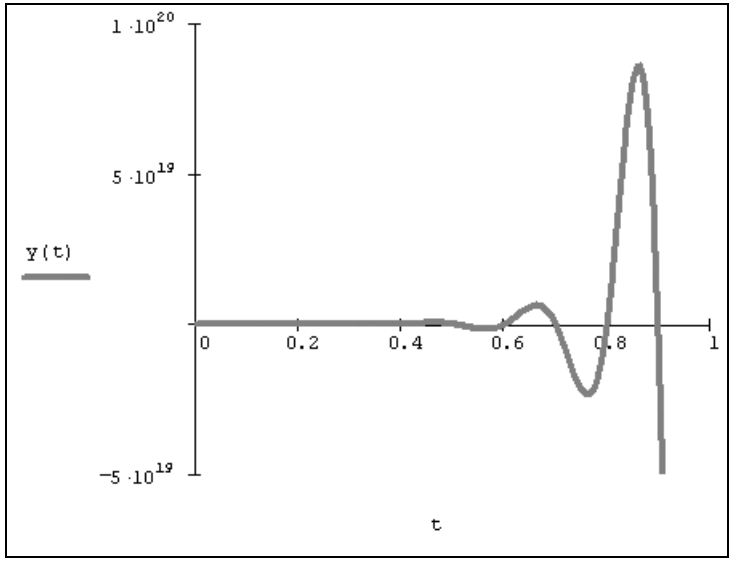



**Рис. 9.13.** Неверное решение более жесткого ОДУ методом Рунге—Кутты (листин 9.8 с коэффициентом `-50`)

Таким образом, во-первых, мы выяснили, что одни и те же уравнения с разными параметрами могут быть как жесткими, так и нежесткими. Во-вторых, чем жестче уравнение, тем больше шагов в обычных численных методах требуется для его устойчивого решения. С классическим примером ОДУ из листинга 9.8 все получилось хорошо, т. к. оно было не очень жестким, и небольшое увеличение числа шагов разрешило все проблемы. Для решения обычными методами более жестких уравнений требуются миллионы, миллиарды и даже большее число шагов.

***Примечание***

Некоторые ученые замечают, что в последние годы методы Рунге—Кутты, считавшиеся ранее незыблемыми, стали уступать свое главенствующее положение среди алгоритмов решения ОДУ методам, способным решать жесткие задачи.

## 9.4.2. Функции для решения жестких ОДУ

Решение жестких систем дифференциальных уравнений можно осуществить только с помощью встроенных функций, аналогичных по действию семейству рассмотренных выше функций для обычных ОДУ:

- `Radau(y0,t0,t1,M,F)` — алгоритм RADAUS для жестких систем ОДУ;

- `Stiffb(y0,t0,t1,M,F,J)` — алгоритм Булирша—Штера для жестких систем ОДУ;
- `Stiffr(y0,t0,t1,M,F,J)` — алгоритм Розенброка для жестких систем ОДУ:
  - `y0` — вектор начальных значений в точке `t0`;
  - `t0,t1` — начальная и конечная точки расчета;
  - `M` — число шагов численного метода;
  - `F` — векторная функция `F(t,y)` размера `1×N`, задающая систему ОДУ;
  - `J` — матричная функция `J(t,y)` размера `(N+1)×N`, составленная из вектора производных функции `F(t,y)` по `t` (правый столбец) и ее якобиана (`N` левых столбцов).

Как вы можете заметить, для двух последних функций серьезным отличием от функций, решающих нежесткие системы, является добавление к стандартному набору параметров дополнительной матричной функции, задающей якобиан системы ОДУ. Решение выдается в виде матрицы, по форме идентичной аналогичным функциям решения нежестких задач Коши.

> ***Примечание***
>
> Встроенная функция `Radau`, которая не требует явного задания якобиана системы уравнений, появилась в версии Mathcad 2001i, а остальные две — в Mathcad 2001.

Решение жесткой задачи из предыдущего раздела при помощи функции `Radau` приведено в листинге 9.9. Результат показан в виде графика на рис. 9.14 вместе с графиком решения менее жесткой задачи (для которого применялся листинг 9.8). Как вы видите, хватило всего пяти точек разбиения интервала интегрирования жесткого ОДУ, чтобы метод с ним справился. Специфика применения других встроенных функций, требующих дополнительного задания якобиана, будет рассмотрена в следующем разделе на примере уравнения химической кинетики.

**Листинг 9.9. Решение жесткого ОДУ алгоритмом RADAUS**

```
F(t, y) := -100 · (y - cos(t))

y0 := 1        M := 5

u := Radau(y0, 0, 1, M, F)
```

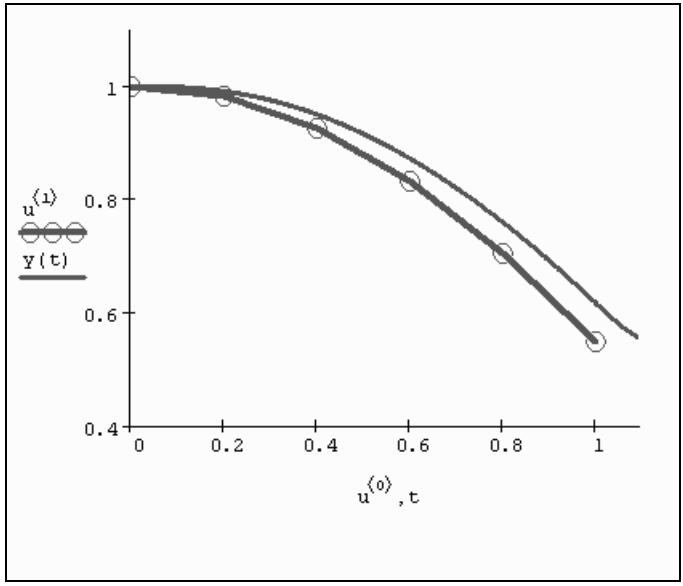



**Рис. 9.14.** Решение жесткого ОДУ методом RADAUS (продолжение листингов 9.8 и 9.9)

В заключение приведем соответствующие встроенные функции, которые применяются для решения жестких систем ОДУ не на всем интервале, а только в одной заданной точке `t1`.

- `radau(y0,t0,t1,acc,F,k,s)` — метод RADAUS.
- `stiffb(y0,t0,t1,acc,F,J,k,s)` — метод Булирша—Штера.
- `stiffr(y0,t0,t1,acc,F,J,k,s)` — метод Розенброка.

Имена этих функций пишутся со строчной буквы, а их действие и набор параметров полностью аналогичны рассмотренным нами ранее для функций, относящихся к решению в заданной точке нежестких систем (*см. разд. 9.3.2*). Отличие заключается в специфике применяемого алгоритма и необходимости задания матричной функции якобиана `J(t,y)` (для двух последних функций).

## 9.4.3. Пример: химическая кинетика

Рассмотрим классическую модель химической кинетики (Робертсон, 1966 г.), которая как нельзя лучше передает смысл понятия жесткости ОДУ.

## Попытка решения стандартными методами

Рассмотрим составную схему химического взаимодействия трех веществ. Пусть вещество "0" медленно превращается в "1": "0"→"1" (со скоростью 0.1), вещество "1" при каталитическом воздействии самого себя превращается очень быстро в вещество "2": "1"+"1"→"2"+"1" ($10^3$). И, наконец, подобным образом (но со средней скоростью) реагируют вещества "2" и "1": "1"+"2"→"0"+"2" ($10^2$). Система ОДУ, описывающая динамику концентрации реагентов, с попыткой решения методом Рунге—Кутты, приведена в символической форме в первой строке листинга 9.10.

**Листинг 9.10. Жесткая система ОДУ химической кинетики**

$$F(t, y) := \begin{pmatrix} -0.1 \cdot y_0 + 10^2 \cdot y_1 \cdot y_2 \\ 0.1 \cdot y_0 - 10^2 \cdot y_1 \cdot y_2 - 10^3 \cdot y_1 \\ 10^3 \cdot y_1 \end{pmatrix}$$

$$y0 := \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$D := \text{rkfixed}(y0, 0, 50, 20000, F)$$

Бросается в глаза сильно различающийся порядок коэффициентов при разных слагаемых. Именно степень этого различия чаще всего и определяет жесткость системы ОДУ. В качестве соответствующей характеристики выбирают матрицу Якоби (якобиан) векторной функции F(t,y), т. е. функциональную матрицу, составленную из производных F(t,y) (*см. разд. 3.4.3*). Чем сильнее вырождена матрица Якоби, тем жестче система уравнений. В приведенном примере определитель якобиана и вовсе равен нулю при любых значениях $y_0$, $y_1$ и $y_2$ (листинг 9.11, вторая строка). В первой строке листинга 9.11 приведено напоминание способа вычисления якобиана средствами Mathcad на примере определения элементов его первой строки.

**Листинг 9.11. Якобиан рассматриваемой системы ОДУ химической кинетики**

$$\frac{\partial}{\partial x} F\left[t, \begin{pmatrix} x \\ y_1 \\ y_2 \end{pmatrix}\right]_0 \rightarrow -.1 \qquad \frac{\partial}{\partial x} F\left[t, \begin{pmatrix} y_0 \\ x \\ y_2 \end{pmatrix}\right]_0 \rightarrow 100 \cdot y_2 \qquad \frac{\partial}{\partial x} F\left[t, \begin{pmatrix} y_0 \\ y_1 \\ x \end{pmatrix}\right]_0 \rightarrow 100 \cdot y_1$$

$$\left\| \begin{pmatrix} -0.1 & 10^2 \cdot y_2 & 10^2 \cdot y_1 \\ 0.1 & -10^2 \cdot y_2 - 10^3 & -10^2 \cdot y_1 \\ 0 & 10^3 & 0 \end{pmatrix} \right\| \rightarrow 0$$

Для примера, приведенного в листинге 9.10, стандартным методом Рунге—Кутты все-таки удается найти решение (оно показано на рис. 9.15). Однако для этого требуется очень большое число шагов, `M=20000`, что делает расчеты очень медленными. При меньшем числе шагов численному алгоритму не удается найти решение. В процессе работы алгоритма оно расходится, и Mathcad вместо результата выдает ошибку о превышении предельно большого числа.

Еще один факт, на который стоит обратить внимание, — это различие в порядке величины получающегося решения. Как видно из рис. 9.15, концентрация первого реагента $y_1$ существенно (в тысячи раз) превышает концентрацию остальных. Это свойство также очень характерно для жестких систем.

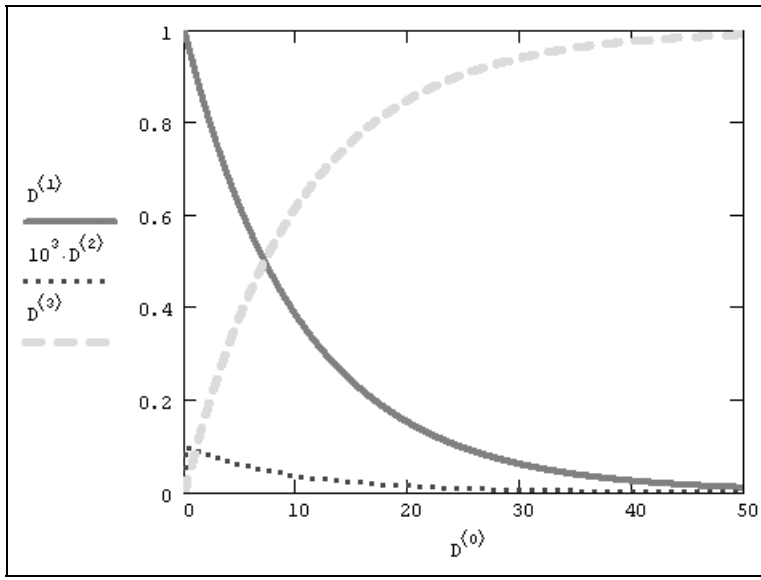



**Рис. 9.15.** Решение жесткой системы ОДУ химической кинетики методом Рунге—Кутты (продолжение листинга 9.10)

### *Примечание*

В принципе, можно было бы снизить жесткость системы "вручную", применяя масштабирование. Для этого нужно искусственно уменьшить искомую функцию

$y_1$, к примеру, в тысячу раз, разделив все слагаемые в системе ОДУ, содержащие $y_1$, на 1000. После масштабирования для решения полученной системы методом Рунге—Кутты будет достаточно взять всего M=20 шагов. Соответствующий пример вы найдете на компакт-диске.

## Применение специфических алгоритмов

Покажем действие этих алгоритмов на том же примере жесткой системы ОДУ химической кинетики (листинг 9.12). Обратите внимание, как следует представлять в данном случае якобиан, сравнив задание матричной функции в предпоследней строке листинга 9.12 с заданием якобиана из листинга 9.11.

**Листинг 9.12. Решение жесткой системы ОДУ химической кинетики**

$$y0 := \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$F(t, y) := \begin{pmatrix} -0.1 \cdot y_0 + 10^2 \cdot y_1 \cdot y_2 \\ 0.1 \cdot y_0 - 10^2 \cdot y_1 \cdot y_2 - 10^3 \cdot y_1 \\ 10^3 \cdot y_1 \end{pmatrix}$$

$$J(t, y) := \begin{pmatrix} 0 & -0.1 & 10^2 \cdot y_2 & 10^2 \cdot y_1 \\ 0 & 0.1 & -10^2 \cdot y_2 - 10^3 & -10^2 \cdot y_1 \\ 0 & 0 & 10^3 & 0 \end{pmatrix}$$

$$D := Stiffb(y0, 0, 50, 20, F, J)$$

Расчеты показывают, что для получения того же результата (который был изображен на рис. 9.15) оказалось достаточно в тысячу раз меньшего количества шагов численного алгоритма, чем для стандартного метода Рунге—Кутты! Примерно во столько же раз требуется меньше компьютерного времени на проведение расчетов. Стоит ли говорить, что, если вы имеете дело с жесткими (в той или иной степени) системами, применение описанных специальных алгоритмов просто необходимо.

Важно заметить, что до сих пор мы имели дело с примером не очень жесткой системы. Попробуйте вместо скоростей упомянутых химических реакций 0.1, $10^3$ и $10^2$ взять другие числа, например, 0.05, $10^4$ и $10^7$ соответственно. Заметим, что такое соотношение скоростей часто встречается в прикладных

задачах химической кинетики и определяет куда более жесткую систему ОДУ. Ее уже никак не удается решить стандартными методами, поскольку число шагов численного метода должно быть просто гигантским. А между тем алгоритмы для жестких ОДУ справляются с этой задачей с легкостью (рис. 9.16), причем практически при тех же значениях шага, что были взяты в листинге 9.12. Обратите внимание, что порядки величины решений для концентраций различных веществ на рис. 9.16 различаются еще сильнее, чем в предыдущем (менее жестком) примере.

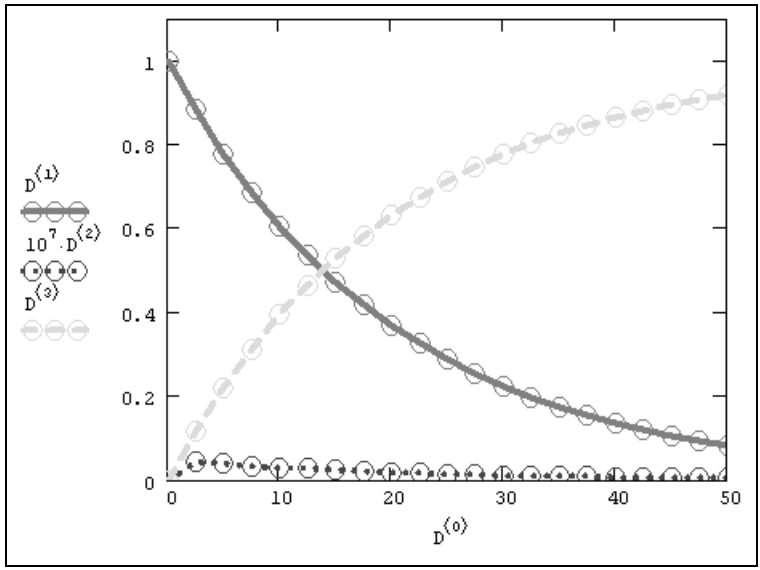



**Рис. 9.16.** Решение более жесткой системы ОДУ химической кинетики методом Розенброка

Для решения очень жестких систем особенно подходит функция `Radau`, которая соблазнительна еще и тем, что избавляет от необходимости предварительного расчета якобиана. В случае очень жесткой задачи химической кинетики результат, выдаваемый функцией `Radau`, практически тот же, что и в случае использования алгоритма Розенброка (рис. 9.16). Любопытно, что решение менее жесткой системы (из предыдущего листинга) при помощи функции `Radau` удается получить только для первой половины интервала, примерно до `t1=20`. Для больших интервалов она дает сбой, выводя вместо решения сообщение об ошибке.

## 9.5. Примеры: классические динамические системы

В предыдущих разделах было использовано в качестве примера в основном *линейное* уравнение осциллятора (оно содержало только первую степень неизвестных функций и их производных). Между тем многие *нелинейные* уравнения демонстрируют совершенно удивительные свойства, причем решение большинства из них можно получить лишь численно.

Рассмотрим несколько наиболее известных классических примеров динамических систем, имея в виду, что читателю они могут пригодиться как в познавательных, так и в практических целях. Это модели динамики популяций (Вольтерры), генератора автоколебаний (Ван дер Поля), турбулентной конвекции (Лоренца) и химической реакции с диффузией (Пригожина). Как уже было сказано, для изучения динамических систем разработана специальная теория, центральным моментом которой является анализ фазовых портретов, т. е. решений, получающихся при выборе всевозможных начальных условий.

> ***Примечание***
>
> В большинстве примеров, изложенных ниже, для построения схемы фазового портрета рассчитывается несколько решений для разных начальных условий. О том, как проделать такие расчеты в Mathcad, будет рассказано ниже на примере модели брюсселятора (*см. разд. 9.5.4*).

Ограничимся в дальнейшем минимальными комментариями и приведем листинги и графики решений без подробного обсуждения.

### 9.5.1. Модели динамики биологических популяций

Модель взаимодействия "хищник—жертва" независимо предложили в 1925—1927 гг. Лотка и Вольтерра. Два дифференциальных уравнения (листинг 9.13) моделируют временную динамику численности двух биологических популяций жертвы $y_0$ и хищника $y_1$. Предполагается, что жертвы размножаются с постоянной скоростью `C`, а их численность убывает вследствие поедания хищниками. Хищники же размножаются со скоростью, пропорциональной количеству пищи (с коэффициентом `r`), и умирают естественным образом (смертность определяется константой `D`). В листинге рассчитываются три решения `D`, `G`, `P` для разных начальных условий.

**Листинг 9.13. Модель "хищник—жертва"**

```
C := 0.1          D := 1          r := 0.1

F(t, y) := ( C·y0 − r·y0·y1 )
           ( −D·y1 + r·y0·y1 )

M := 500

t0 := 0           t1 := 100

D := rkfixed[ ( 10 ), t0, t1, M, F ]
             (  8 )

G := rkfixed[ ( 10 ), t0, t1, M, F ]
             (  4 )

P := rkfixed[ ( 10  ), t0, t1, M, F ]
             ( 1.5 )
```

Модель замечательна тем, что в такой системе наблюдаются циклическое увеличение и уменьшение численности и хищника (рис. 9.17), и жертвы, так часто наблюдаемое в природе. Фазовый портрет системы представляет собой концентрические замкнутые кривые, окружающие одну стационарную точку, называемую *центром*. Как видно, модельные колебания численности обеих популяций существенно зависят от начальных условий — после каждого периода колебаний система возвращается в ту же точку. Динамические системы с таким поведением называют *негрубыми*.

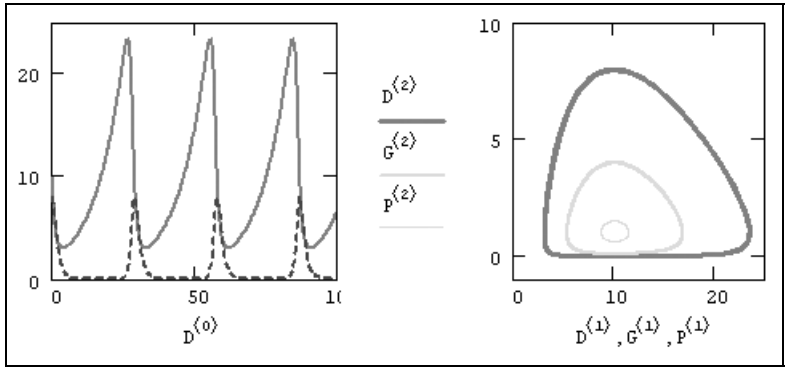



**Рис. 9.17.** График решения (слева) и фазовый портрет (справа) системы "хищник—жертва" (продолжение листинга 9.13)

### *Примечание*

Пример негрубой системы (модель осциллятора без затухания) с особой точкой типа "центр" был нами рассмотрен ранее (*см. разд. 9.1*).

Рассмотренную модель динамики двух популяций легко можно модифицировать, изменив тип взаимодействия "хищник—жертва" на тип конкуренции. Для этого надо учесть, что рост численности каждой популяции тормозит, во-первых, межвидовая, и, во-вторых, внутривидовая конкуренция.

В результате система (во второй строке листинга) запишется в виде:

$$F(t, y) := \begin{bmatrix} C \cdot y_0 - r_{0,1} \cdot y_0 \cdot y_1 - r_{0,0} \cdot (y_0)^2 \\ D \cdot y_1 - r_{1,0} \cdot y_0 \cdot y_1 - r_{1,1} \cdot (y_1)^2 \end{bmatrix},$$

где матрица `r` задает коэффициенты убывания численности вследствие конкурентной борьбы (диагональные элементы соответствуют внутри-, а недиагональные — межвидовой конкуренции).

График решения (для разных начальных условий) и фазовый портрет для описанной системы ОДУ показаны на рис. 9.18. Как видно, конкурентная борьба приводит к установлению некоторого стационарного состояния, выражающего равновесие видов. Особая точка, к которой стремится решение системы ОДУ подобным образом, называется *узлом*.

### *Примечание*

Соответствующий файл с программой вы найдете на компакт-диске вместе с более простой моделью динамики одной популяции с внутривидовой конкуренцией (называемой *логистической* моделью).

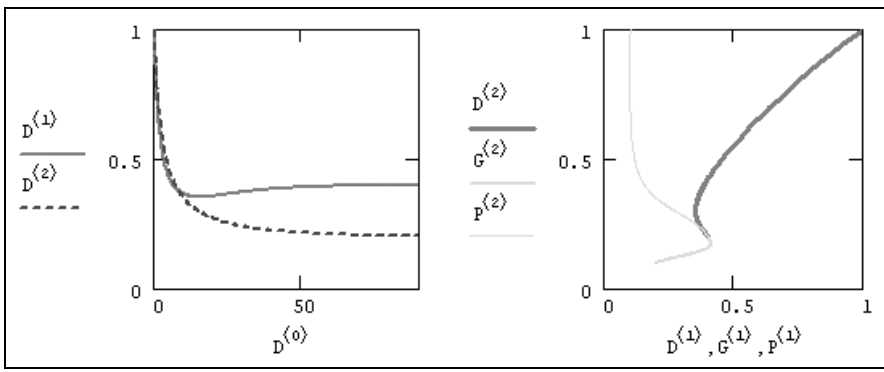



**Рис. 9.18.** График решения (слева) и фазовый портрет (справа) модели конкуренции популяций

## 9.5.2. Автоколебания

Рассмотрим решение уравнения Ван дер Поля, описывающего электрические колебания в замкнутом контуре, состоящем из соединенных последовательно конденсатора, индуктивности, нелинейного сопротивления и элементов, обеспечивающих подкачку энергии извне (листинг 9.14). Неизвестная функция времени `y(t)` имеет смысл электрического тока, а в параметре `μ` заложены количественные соотношения между составляющими электрической цепи, в том числе и нелинейной компонентой сопротивления.

**Листинг 9.14. Модель Ван дер Поля (μ=1)**

```
μ := 1
Given
d^2/dt^2 y(t) − μ·(1 − y(t)^2)·d/dt y(t) + y(t) = 0
y(0) = 0.01
y'(0) = 0
y := Odesolve(t, 30)
```

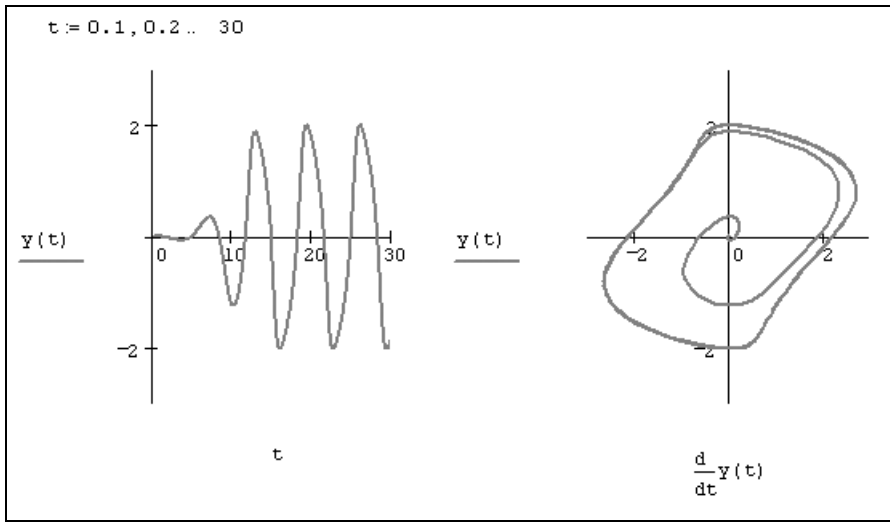



**Рис. 9.19.** График решения (слева) и фазовый портрет (справа) уравнения Ван дер Поля (продолжение листинга 9.14)

Решением уравнения Ван дер Поля являются колебания, вид которых для μ=1 показан на рис. 9.19. Они называются *автоколебаниями* и принципиально отличаются от рассмотренных нами ранее (например, колебаний маятника в модели осциллятора или численности популяций в модели Вольтерра) тем, что их характеристики (амплитуда, частота, спектр) не зависят от начальных условий, а определяются исключительно свойствами самой динамической системы. Через некоторое время расчетов после выхода из начальной точки решение выходит на один и тот же цикл колебаний, называемый *предельным циклом*. Аттрактор типа предельного цикла является замкнутой кривой на фазовой плоскости. К нему асимптотически притягиваются все окрестные траектории, выходящие из различных начальных точек, как изнутри (рис. 9.19), так и снаружи (рис. 9.20) предельного цикла.

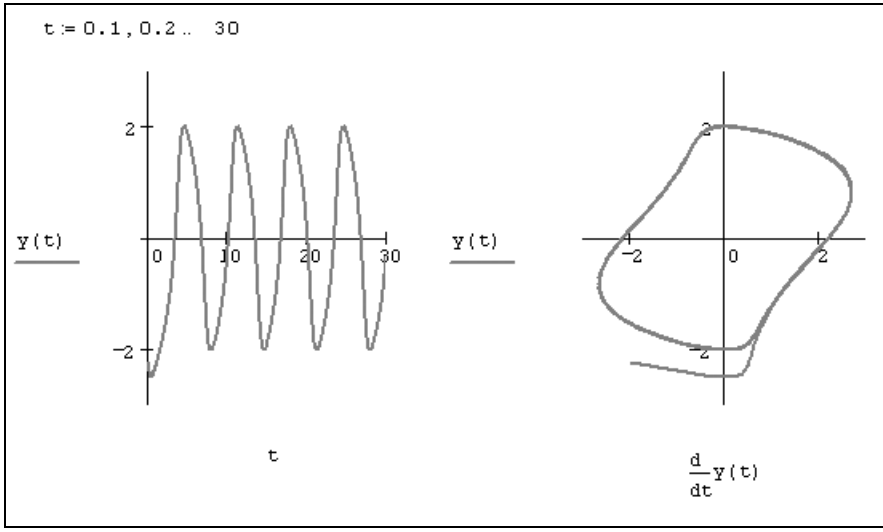



**Рис. 9.20.** Решение уравнения Ван дер Поля при других начальных условиях y=-2, y'=-3

### *Примечание 1*

Если компьютер у вас не самый производительный, то расчет в Mathcad фазового портрета с рис. 9.19, 9.20 может занять относительно продолжительное время, что связано с численным определением сначала решения y(t), а потом его производной. Время расчетов можно было бы существенно сократить, если использовать вместо вычислительного блока Given/Odesolve одну из встроенных функций, которые выдают решение в виде матрицы, например, rkfixed.

***Примечание 2***

По мере возрастания параметра $\mu$ модель Ван дер Поля становится все более жесткой. Например, при $\mu=5000$ решение уже придется искать при помощи специфических методов решения жестких задач (*см. разд. 9.3*). Это еще раз доказывает, что одна и та же система ОДУ с различными коэффициентами может быть жесткой в разной степени.

## 9.5.3. Странный аттрактор

Одна из самых знаменитых динамических систем предложена в 1963 г. Лоренцем в качестве упрощенной модели конвективных турбулентных движений жидкости в нагреваемом сосуде тороидальной формы. Система состоит из трех ОДУ и имеет три параметра модели (листинг 9.15). Поскольку неизвестных функций три, то фазовый портрет системы должен определяться не на плоскости, а в трехмерном пространстве.

**Листинг 9.15. Модель Лоренца**

$\sigma := 10 \qquad r := 27 \qquad b := \frac{8}{3}$

$$y0 := \begin{pmatrix} 10 \\ 10 \\ 10 \end{pmatrix}$$

$$F(t, y) := \begin{pmatrix} \sigma \cdot y_1 - \sigma \cdot y_0 \\ -y_0 \cdot y_2 + r \cdot y_0 - y_1 \\ y_1 \cdot y_0 - b \cdot y_2 \end{pmatrix}$$

$t0 := 0 \qquad t1 := 30$

$N := 1000$

$D := rkfixed(y0, t0, t1, N, F)$

Решением системы Лоренца при определенном сочетании параметров (рис. 9.21 и 9.22) является *странный аттрактор* (или *аттрактор Лоренца*) — притягивающее множество траекторий на фазовом пространстве, которое по виду идентично случайному процессу. В некотором смысле аттрактор Лоренца является стохастическими автоколебаниями, которые поддерживаются в динамической системе за счет внешнего источника.

Решение в виде странного аттрактора появляется только при некоторых сочетаниях параметров. В качестве примера на рис. 9.23 приведен результат для $r=10$ и тех же значений остальных параметров.

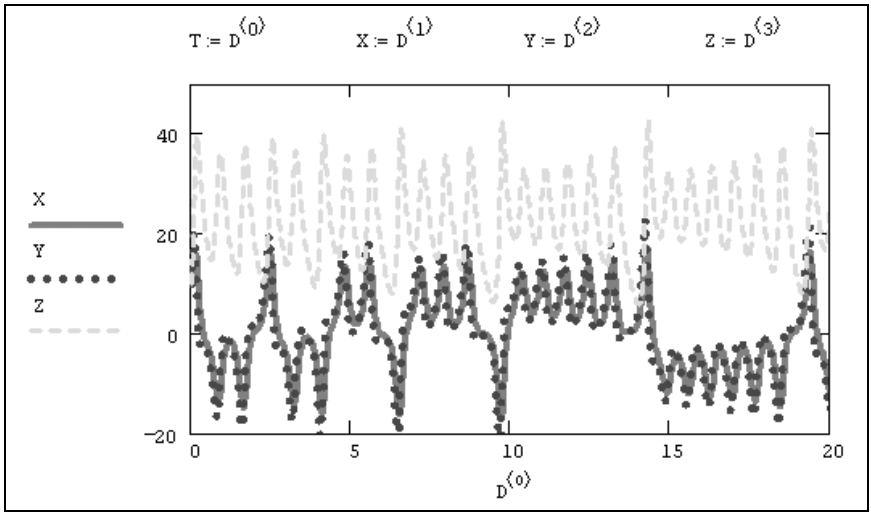



**Рис. 9.21.** Решение в виде аттрактора Лоренца (продолжение листинга 9.15)

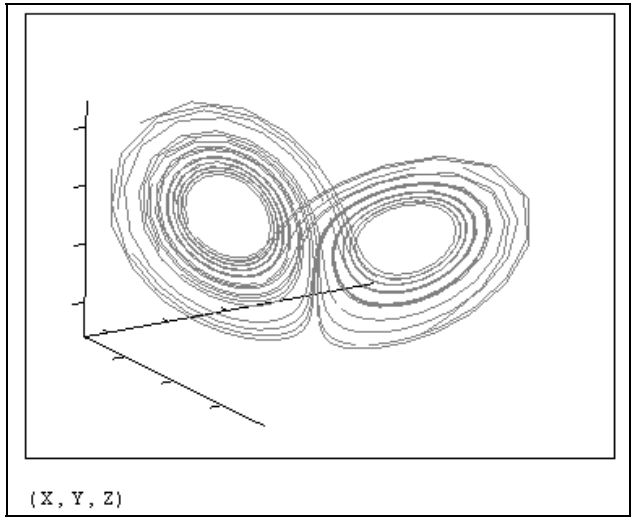



**Рис. 9.22.** Аттрактор Лоренца на фазовой плоскости (продолжение листинга 9.15)

Как видно, аттрактором в этом случае является фокус. Перестройка типа фазового портрета происходит в области промежуточных `r`. Критическое сочетание параметров, при которых фазовый портрет системы качественно меняется, называется в теории динамических систем точкой *бифуркации*.

Физический смысл бифуркации в модели Лоренца, согласно современным представлениям, описывает переход ламинарного движения жидкости к турбулентному.

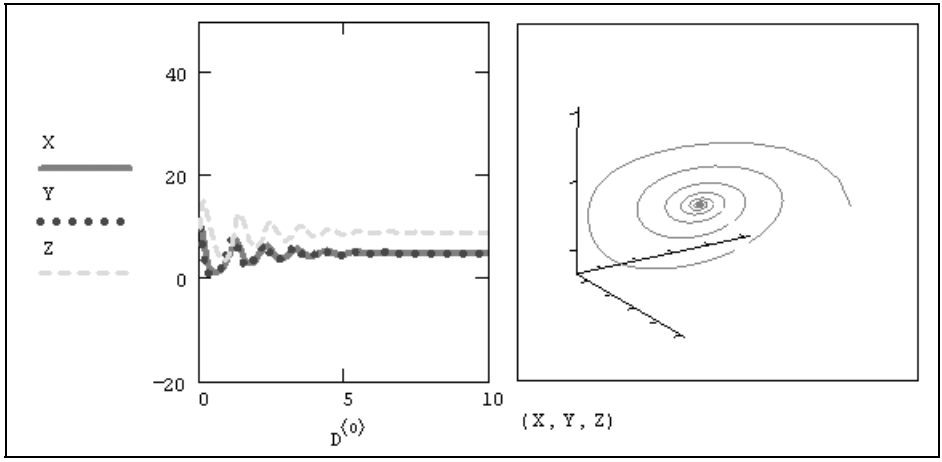



**Рис. 9.23.** Решение системы Лоренца с измененным параметром `r=10`

Замечательно, что решение подобных нелинейных динамических систем можно получить только численно, поэтому их изучение стало бурно развиваться с ростом возможностей вычислительной техники в последние полвека.

### *Примечание 1*

На компакт-диске вы найдете решение в виде странного аттрактора еще одной классической динамической системы (*палеомагнетизм Рикитаке*), моделирующей глобальное движение магнитных полюсов Земли.

### *Примечание 2*

Также на компакт-диске находится программа с реализацией алгоритма поиска *отображения Пуанкаре* (на примере модели Лоренца) — эффективного приема теории динамических систем.

## 9.5.4. Брюсселятор

До сих пор в этой главе в качестве примеров расчета динамических систем мы приводили графики 1—3 траекторий на фазовой плоскости. Однако для

надежного исследования фазового портрета необходимо решить систему ОДУ большое количество раз с самыми разными начальными условиями (и, возможно, с разным набором параметров модели), чтобы посмотреть, к каким аттракторам сходятся различные траектории. В Mathcad можно реализовать эту задачу, например, в форме алгоритма, приведенного в листинге 9.16 для решения системы уравнений автокаталитической химической реакции с диффузией. Эта модель, называемая моделью *брюсселятора*, предложена в 1968 г. Лефевром и Пригожиным. Неизвестные функции отражают динамику концентрации промежуточных продуктов некоторой реальной химической реакции. Параметр модели `B` равен исходной концентрации катализатора.

**Листинг 9.16. Построение фазового портрета для модели брюсселятора**

```
v := ( 0    0    2.5   1.5   0.5   1   1   1.5   0.1   0.5 )
     ( 0.5  1.5   0     0     1    0   1   2     0.1   0.2 )

B := 0.5

D(t, y) := [ -(B + 1)·y0 + (y0)^2·y1 + 1 ]
           [          B·y0 - (y0)^2·y1   ]

t0 := 0        t1 := 10

M := 100

U := | y ← v^⟨0⟩
     | Z ← rkfixed(y, t0, t1, M, D)
     | Z1^⟨0⟩ ← Z^⟨0⟩
     | Z1^⟨1⟩ ← Z^⟨1⟩
     | Z1^⟨2⟩ ← Z^⟨2⟩
     | for k ∈ 1.. last[(v^T)^⟨1⟩]
     |     | y ← v^⟨k⟩
     |     | Z ← rkfixed(y, t0, t1, M, D)
     |     | Z2^⟨0⟩ ← Z^⟨0⟩
     |     | Z2^⟨1⟩ ← Z^⟨1⟩
     |     | Z2^⟨2⟩ ← Z^⟨2⟩
     |     | Z1 ← stack(Z1, Z2)
     | Z1
```

Предложенный алгоритм формирует из отдельных матриц решений системы ОДУ с разными начальными условиями объединенную матрицу `U`. Пары начальных условий задаются в первой строке листинга в виде матрицы `v` размера `2×10`. Последнее означает построение десяти траекторий. Для того чтобы поменять количество траекторий, измените соответствующим образом размер этой матрицы. Затем (рис. 9.24) элементы матрицы `U` выводятся на график в виде отдельных точек. Отсутствие соединения точек линиями является недостатком алгоритма, но это плата за возможность представить в Mathcad несложным образом сразу большое количество траекторий на фазовой плоскости.

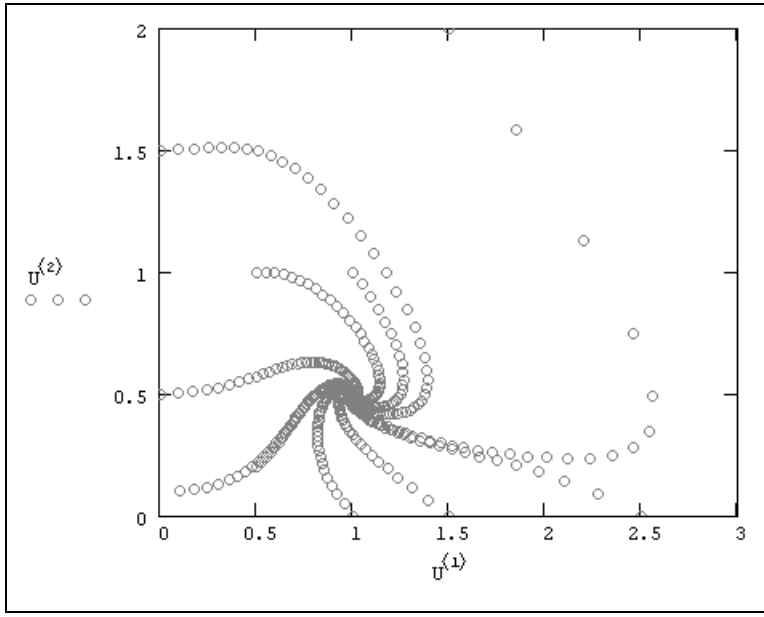



**Рис. 9.24.** Фазовый портрет брюсселятора при `B=0.5` (продолжение листинга 9.16)

Как видно из рис. 9.24, все траектории, вышедшие из разных точек, асимптотически стремятся к одному и тому же аттрактору `(1,0.5)`. Из теории динамических систем нам известно, что такой аттрактор называется *узлом* (с узлом мы уже встречались *в примере разд. 9.4.1*). Конечно, в общем случае при анализе фазового портрета желательно "прощупать" большее число траекторий, задавая более широкий диапазон начальных условий. Не исключено, что в других областях фазовой плоскости траектории будут сходиться к другим аттракторам.

Эволюцию фазового портрета брюсселятора можно наблюдать, проводя расчеты с различным параметром B. При его увеличении узел будет сначала постепенно смещаться в точку с координатами (1,B), пока не достигнет бифуркационного значения B=2. В этой точке происходит качественная перестройка портрета, выражающаяся в рождении предельного цикла. При дальнейшем увеличении B происходит лишь количественное изменение параметров этого цикла. Решение, полученное при B=2.5, показано на рис. 9.25.

### *Примечание*

Чтобы найти аттракторы динамической системы, как известно, нужно решить систему алгебраических уравнений, получающуюся из системы ОДУ заменой нулями их левых частей. Эти задачи также удобно решать средствами Mathcad (*см. главу 5*). В частности, исследование зависимости фазового портрета от параметров системы ОДУ и поиск бифуркаций можно проводить методами продолжения (*см. разд. 5.3.3*).

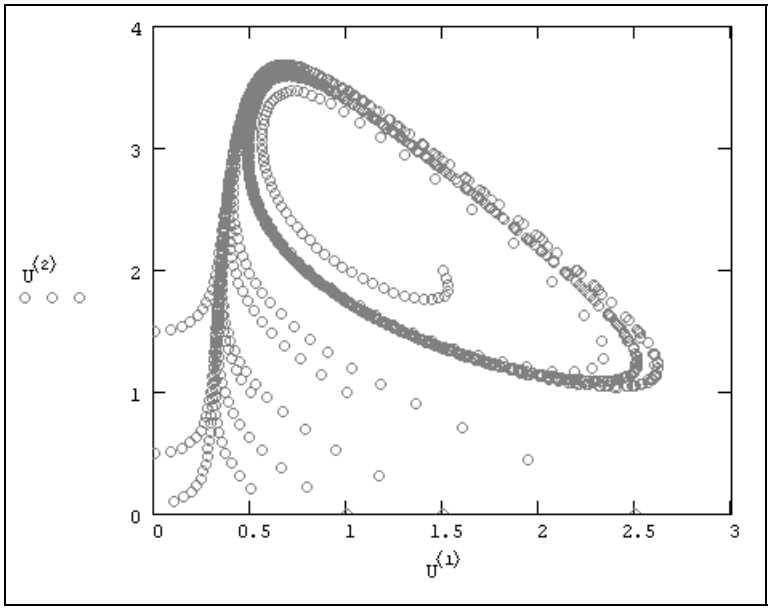



**Рис. 9.25.** Фазовый портрет брюсселятора при B=2.5

Читатели, сталкивающиеся с расчетом динамических систем, несомненно, оценят возможности Mathcad по построению фазовых портретов и исследованию бифуркаций. Возможно также, что они найдут лучшие программные решения этой задачи, чем алгоритм, предложенный в данном разделе автором.

# Глава 10

# Обыкновенные дифференциальные уравнения: краевые задачи

В этой главе рассматриваются краевые задачи для обыкновенных дифференциальных уравнений (ОДУ). Средства Mathcad, реализующие алгоритм стрельбы (*см. разд. 10.2*), позволяют решать краевые задачи для систем ОДУ, в которых часть граничных условий поставлена в начальной точке интервала, а остальная часть — в его конечной точке. Также возможно решать уравнения с граничными условиями, поставленными в некоторой внутренней точке.

Краевые задачи во множестве практических приложений часто зависят от некоторого числового параметра. При этом решение существует не для всех его значений, а лишь для счетного их числа. Такие задачи называют задачами на собственные значения (*см. разд. 10.3*).

Несмотря на то, что, в отличие от задач Коши для ОДУ, в Mathcad не предусмотрены встроенные функции для решения жестких краевых задач, с ними все-таки можно справиться, применив программирование разностных схем, подходящих для решения задач этого класса (*см. разд. 10.4*). О подходе к решению нелинейных краевых задач написано в конце главы (*см. разд. 10.5*).

## 10.1. О постановке задач

Постановка краевых задач для ОДУ отличается от задач Коши, рассмотренных в *главе 9*, тем, что граничные условия для них ставятся не в одной начальной точке, а на обеих границах расчетного интервала. Если имеется система `N` обыкновенных дифференциальных уравнений первого порядка, то

часть из N условий может быть поставлена на одной границе интервала, а оставшиеся условия — на противоположной границе.

### *Примечание*

Дифференциальные уравнения высших порядков можно свести к эквивалентной системе ОДУ первого порядка (*см. главу 9*).

Чтобы лучше понять, что из себя представляют краевые задачи, рассмотрим их постановочную часть на конкретном физическом примере модели взаимодействия встречных световых пучков. Предположим, что надо определить распределение интенсивности оптического излучения в пространстве между источником (лазером) и зеркалом, заполненном некоторой средой (рис. 10.1). Будем считать, что от зеркала отражается R-я часть падающего излучения (т. е. его коэффициент отражения равен R), а среда как поглощает излучение с коэффициентом ослабления a(x), так и рассеивает его. Причем коэффициент рассеяния назад равен r(x). В этом случае закон изменения интенсивности $y_0(x)$ излучения, распространяющегося вправо, и интенсивности $y_1(x)$ излучения влево определяется системой двух ОДУ первого порядка:

$$\begin{aligned} \frac{dy_0(x)}{dx} &= -a(x)\cdot y_0(x) + r(x)\cdot y_1(x);\\ \frac{dy_1(x)}{dx} &= a(x)\cdot y_1(x) - r(x)\cdot y_0(x). \end{aligned} \qquad (10.1)$$

Для правильной постановки задачи требуется, помимо уравнений, задать такое же количество граничных условий. Одно из них будет выражать известную интенсивность излучения I0, падающего с левой границы x=0, а второе — закон отражения на его правой границе x=1:

$$\begin{aligned} y_0(0) &= I0;\\ y_1(1) &= R\times y_0(1). \end{aligned} \qquad (10.2)$$

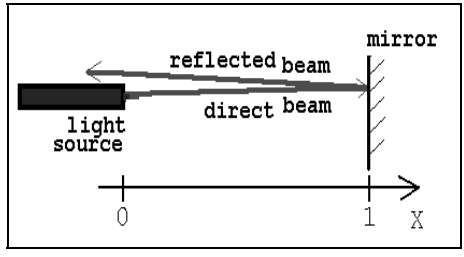



**Рис. 10.1.** Модель краевой задачи

Полученную задачу называют *краевой* (boundary value problem), поскольку условия поставлены не на одной, а на обеих границах интервала `(0,1)`. И, в связи с этим, их не решить методами предыдущей главы, предназначенными для задач с начальными условиями. Далее для показа возможностей Mathcad будем использовать этот пример с `R=1` и конкретным видом `a(x)=const=1` и `r(x)=const=0.1`, описывающим случай *изотропного* (не зависящего от координаты `x`) рассеяния.

***Примечание 1***

Модель, представленная на рис. 10.1, привела к краевой задаче для системы *линейных* ОДУ. Она имеет аналитическое решение в виде комбинации экспонент. Более сложные, *нелинейные*, задачи возможно решить только численно. Нетрудно сообразить, что модель станет нелинейной, если сделать коэффициенты ослабления и рассеяния зависящими от интенсивности излучения. Физически это будет соответствовать изменению оптических свойств среды под действием мощного излучения.

***Примечание 2***

Модель встречных световых пучков привела нас к системе уравнений (10.1), в которые входят производные только по одной переменной `x`. Если бы мы стали рассматривать более сложные эффекты рассеяния в стороны (а не только вперед и назад), то в уравнениях появились бы частные производные по другим пространственным переменным `y` и `z`. В этом случае получилась бы краевая задача для уравнений в частных производных, решение которой во много раз сложнее ОДУ.

# 10.2. Решение краевых задач средствами Mathcad

Для решения краевых задач в Mathcad реализован наиболее популярный алгоритм, называемый методом *стрельбы* или *пристрелки* (shooting method). Он, по сути, сводит решение краевой задачи к решению серии задач Коши с различными начальными условиями. Рассмотрим сначала основной принцип алгоритма стрельбы на примере модели световых пучков (см. рис. 10.1), а встроенные функции, реализующие этот алгоритм, приведем в следующем разделе.

## 10.2.1. Алгоритм стрельбы

Суть метода стрельбы заключается в пробном задании недостающих граничных условий на левой границе интервала и решении затем полученной задачи

Коши хорошо известными методами (*см. главу 11*). В нашем примере не хватает начального условия для $y_1(0)$, поэтому сначала зададим ему произвольное значение, например, $y_1(0)=10$. Конечно, такой выбор не совсем случаен, поскольку из физических соображений ясно, что, во-первых, интенсивность излучения — величина заведомо положительная, и, во-вторых, отраженное излучение должно быть намного меньше падающего. Решение задачи Коши с помощью функции `rkfixed` приведено в листинге 10.1, причем в его последней строке выведены расчетные значения $y_0$ и $y_1$ на правой крайней точке интервала `x=1`. Разумеется, они не совпадают, т. е. правое краевое условие не выполнено, из чего следует, что полученный результат не является решением поставленной краевой задачи.

**Листинг 10.1. Решение пробной задачи Коши для модели (10.1)**

```
r(x) := .1    a(x) := 1

R := 1

D(x, y) := ( -a(x)·y0 + r(x)·y1 )
           ( a(x)·y1 - r(x)·y0  )

I0 := 100

y := ( I0 )
     ( 10 )

M := 10

S := rkfixed(y, 0, 1, M, D)

S_M,1 = 37.558        S_M,2 = 15.372
```

График полученных решений показан на рис. 10.2, слева. Из него также видно, что взятое наугад второе начальное условие не обеспечило выполнение граничного условия при `x=1`. В целях лучшего выполнения этого граничного условия следует взять большее значение $y_1(0)$, например, $y_1(0)=15$, и вновь решить задачу Коши. Соответствующий результат показан на том же рис. 10.2, в центре. Граничное условие выполняется с лучшей точностью, но опять оказалось недостаточным. Для еще одного значения $y_1(0)=20$ получается решение, показанное на рис. 10.2, справа. Из сравнения двух правых графиков легко заключить, что недостающее начальное условие больше `15`, но меньше `20`. Продолжая подобным образом "пристрелку" по недостающему начальному условию, возможно отыскать правильное решение краевой задачи.

В этом и состоит принцип алгоритма стрельбы. Выбирая пробные начальные условия (проводя пристрелку) и решая соответствующую серию задач Коши,

можно найти то решение системы ОДУ, которое (с заданной точностью) удовлетворит граничному условию (или, в общем случае, условиям) на другой границе расчетного интервала.

Конечно, описанный алгоритм несложно запрограммировать самому, оформив его как решение системы заданных алгоритмически уравнений, выражающих граничные условия на второй границе, относительно неизвестных пристрелочных начальных условий. Но делать этого нет необходимости, поскольку он оформлен в Mathcad в виде встроенных функций.

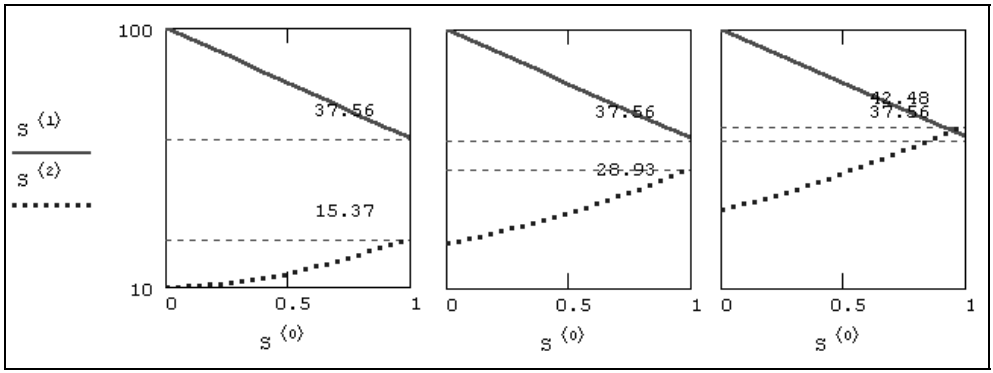



**Рис. 10.2.** Иллюстрация метода стрельбы (продолжение листинга 10.1)

## 10.2.2. Двухточечные краевые задачи

Решение краевых задач для систем ОДУ методом стрельбы в Mathcad достигается применением двух встроенных функций. Первая предназначена для двухточечных задач с краевыми условиями, заданными на концах интервала:

- `sbval(z,x0,x1,D,load,score)` — поиск вектора недостающих `L` начальных условий для двухточечной краевой задачи для системы `N` ОДУ:
  - `z` — вектор размера `L×1`, присваивающий недостающим начальным условиям (на левой границе интервала) начальные значения;
  - `x0` — левая граница расчетного интервала;
  - `x1` — правая граница расчетного интервала;
  - `load(x0,z)` — векторная функция размера `N×1` левых граничных условий, причем недостающие начальные условия поименовываются соответствующими компонентами векторного аргумента `z`;

- `score(x1,y)` — векторная функция размера `L×1`, выражающая `L` правых граничных условий для векторной функции `y` в точке `x1`;
- `D(x,y)` — векторная функция, описывающая систему `N` ОДУ, размера `N×1` и двух аргументов — скалярного `x` и векторного `y`. При этом `y` — это неизвестная векторная функция аргумента `x` того же размера `N×1`.

> ***Внимание!***
>
> Решение краевых задач в Mathcad, в отличие от большинства остальных операций, реализовано не совсем очевидным образом. В частности, помните, что число элементов векторов `D` и `load` равно количеству уравнений `N`, а векторов `z`, `score` и результата действия функции `sbval` — количеству правых граничных условий `L`. Соответственно, левых граничных условий в задаче должно быть `(N-L)`.

Как видно, функция `sbval` предназначена не для поиска собственно решения, т. е. неизвестных функций $y_i(x)$, а для определения недостающих начальных условий в первой точке интервала, т. е. $y_i(x0)$. Чтобы вычислить $y_i(x)$ на всем интервале, требуется дополнительно решить задачу Коши.

Разберем особенности использования функции `sbval` на конкретном примере (листинг 10.2), описанном выше (*см. разд. 10.1.1*). Краевая задача состоит из системы двух уравнений (`N=2`), одного левого (`L=1`) и одного правого (`N-L=2--1=1`) граничного условия.

**Листинг 10.2. Решение краевой задачи**

```
r(x) := .1        a(x) := 1

R := 1            I0 := 100

D(x, y) := ( -a(x)·y0 + r(x)·y1 )
           ( a(x)·y1 - r(x)·y0  )

z0 := 10

load(x0, z) := ( 100 )
               ( z0  )

score(x1, y) := R·y0 - y1

I1 := sbval(z, 0, 1, D, load, score)

I1 = ( 18.555 )

S := rkfixed[ ( I0  ), 0, 1, 10, D ]
            ( I10 )
```

Первые три строки листинга задают необходимые параметры задачи и саму систему ОДУ. В четвертой строке определяется вектор `z`. Поскольку правое граничное условие всего одно, то недостающее начальное условие тоже одно, соответственно, и вектор `z` имеет только один элемент $z_0$. Ему необходимо присвоить начальное значение (мы приняли $z_0$=10, как в листинге 10.1), чтобы запустить алгоритм стрельбы (*см. разд. 10.1.2*).

> ***Примечание***
>
> Начальное значение фактически является параметром численного метода и поэтому может решающим образом повлиять на решение краевой задачи.

В следующей строке листинга векторной функции `load(x,z)` присваиваются левые граничные условия. Эта функция аналогична векторной переменной, определяющей начальные условия для встроенных функций, решающих задачи Коши. Отличие заключается в записи недостающих условий. Вместо конкретных чисел на соответствующих местах пишутся имена искомых элементов вектора `z`. В нашем случае вместо второго начального условия стоит аргумент $z_0$ функции `load`. Первый аргумент функции `load` — это точка, в которой ставится левое граничное условие. Ее конкретное значение определяется непосредственно в списке аргументов функции `sbval`.

Следующая строка листинга определяет правое граничное условие, для введения которого используется функция `score(x,y)`. Оно записывается точно так же, как система уравнений в функции `D`. Аргумент `x` функции `score` аналогичен функции `load` и нужен для тех случаев, когда граничное условие явно зависит от координаты `x`. Вектор `score` должен состоять из такого же числа элементов, что и вектор `z`.

Реализованный в функции `sbval` алгоритм стрельбы ищет недостающие начальные условия таким образом, чтобы решение полученной задачи Коши делало функцию `score(x,y)` как можно ближе к нулю. Как видно из листинга, результат применения `sbval` для интервала `(0,1)` присваивается векторной переменной `I1`. Этот вектор похож на вектор `z`, только в нем содержатся искомые начальные условия вместо приближенных начальных значений, заданных в `z`. Вектор `I1` содержит, как и `z`, всего один элемент $I1_0$. С его помощью можно определить решение краевой задачи `y(x)` (последняя строка листинга). Тем самым функция `sbval` сводит решение краевых задач к задачам Коши. График решения краевой задачи показан на рис. 10.3.

На рис. 10.4 показано решение той же самой краевой задачи, но с другим правым граничным условием, соответствующим `R=0`, т. е. без зеркала на правой границе. В этом случае слабый обратный пучок света образуется исклю-

чительно за счет обратного рассеяния излучения от лазера. Конечно, многие из читателей уже обратили внимание, что реальная физическая среда не может создавать такого большого рассеяния назад. Иными словами, более реальны значения `r(x)<<a(x)`. Однако когда коэффициенты в системе ОДУ при разных $y_i$ очень сильно (на порядки) различаются, система ОДУ становится жесткой, и функция `sbval` не может найти решения, выдавая вместо него сообщение об ошибке "Could not find a solution" (Невозможно найти решение).

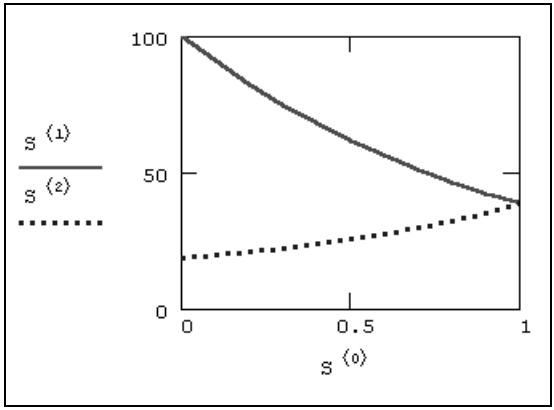



**Рис. 10.3.** Решение краевой задачи для `R=1` (продолжение листинга 10.2)

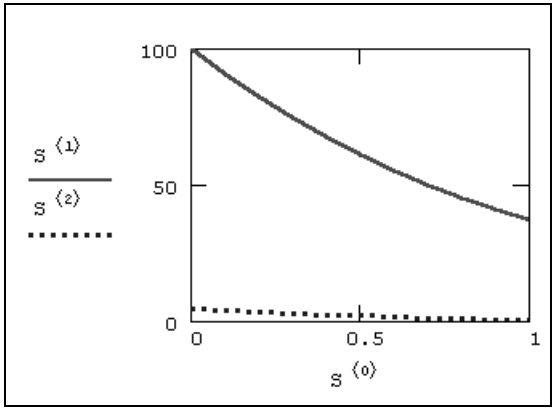



**Рис. 10.4.** Решение краевой задачи для `R=0` (продолжение листинга 10.2 при `R=0`)

***Внимание!***

Метод стрельбы не годится для решения жестких краевых задач. Поэтому алгоритмы решения жестких ОДУ в Mathcad приходится программировать самому (*см. разд. 10.4*).

## 10.2.3. Краевые задачи с условием во внутренней точке

Иногда дифференциальные уравнения определяются с граничными условиями не только на концах интервала, но и с дополнительным условием в некоторой промежуточной точке расчетного интервала. Чаще всего такие задачи содержат данные о негладких в некоторой внутренней точке интервала решениях. Для них имеется встроенная функция `bvalfit`, также реализующая алгоритм стрельбы.

- `bvalfit(z1,z2,x0,x1,xf,D,load1,load2,score)` — поиск вектора недостающих граничных условий для краевой задачи с дополнительным условием в промежуточной точке для системы `N` ОДУ:
  - `z1` — вектор, присваивающий недостающим начальным условиям на левой границе интервала начальные значения;
  - `z2` — вектор того же размера, присваивающий недостающим начальным условиям на правой границе интервала начальные значения;
  - `x0` — левая граница расчетного интервала;
  - `x1` — правая граница расчетного интервала;
  - `xf` — точка внутри интервала;
  - `D(x,y)` — векторная функция, описывающая систему `N` ОДУ размера `N×1` и двух аргументов — скалярного `x` и векторного `y`. При этом `y` — это неизвестная векторная функция аргумента `x` того же размера `N×1`;
  - `load1(x0,z)` — векторная функция размера `N×1` левых граничных условий, причем недостающие начальные условия поименовываются соответствующими компонентами векторного аргумента `z`;
  - `load2(x1,z)` — векторная функция размера `N×1` левых граничных условий, причем недостающие начальные условия поименовываются соответствующими компонентами векторного аргумента `z`;
  - `score(xf,y)` — векторная функция размера `N×1`, выражающая внутреннее условие для векторной функции `y` в точке `xf`.

Обычно функция `bvalfit` применяется для задач, в которых производная решения имеет разрыв во внутренней точке `xf`. Некоторые из таких задач не удается решить обычным методом пристрелки, поэтому приходится вести пристрелку сразу из обеих граничных точек. В этом случае дополнительное внутреннее условие в точке `xf` является просто условием сшивки в ней левого и правого решений. Поэтому для данной постановки задачи оно записывается в форме `score(xf,y):=y`.

Рассмотрим действие функции `bvalfit` на знакомом примере модели взаимодействия пучков света (см. рис. 10.1), предположив, что в промежутке между `xf=0.5` и `x1=1.0` находится другая, оптически более плотная среда с другим коэффициентом ослабления излучения `a(x)=3`. Соответствующая краевая задача решена в листинге 10.3, причем разрывный показатель ослабления определяется в его второй строке.

**Листинг 10.3. Краевая задача с граничным условием в промежуточной точке**

```
I0 := 100    r(x) := .1    R := 1

a(x) := | 1  if x < 0.5
        | 3  otherwise

D(x, y) := ( -a(x)·y0 + r(x)·y1 )
           (  a(x)·y1 - r(x)·y0 )

z1_0 := 20

load1(x0, z1) := ( 100  )
                 ( z1_0 )

z2_0 := 30

load2(x1, z2) := (  z2_0  )
                 ( R·z2_0 )

score(xf, y) := y

I1 := bvalfit(z1, z2, 0, 1, 0.5, D, load1, load2, score)^T

I1 = ( 5.618  )
     ( 13.801 )
```

Система уравнений и левое краевое условие вводятся так же, как и в предыдущем листинге для функции `sbval`. Обратите внимание, что таким же образом записано и правое краевое условие. Для того чтобы ввести условие отра-

жения на правой границе, пришлось определить еще один неизвестный пристрелочный параметр $z2_0$. Строка листинга, в которой определена функция score, задает условие стрельбы — сшивку двух решений в точке xf. В самой последней строке листинга выдан ответ — определенные численным методом значения обоих пристрелочных параметров, которые объединены в вектор I1 (мы применили в предпоследней строке операцию транспонирования, чтобы результат получился в форме вектора, а не матрицы-строки).

Для корректного построения графика решения лучше составить его из двух частей — решения задачи Коши на интервале (x0,xf) и другой задачи Коши для интервала (xf,x1). Реализация этого способа приведена в листинге 10.4, который является продолжением листинга 10.3. В последней строке листинга 10.4 выведено значение второй искомой функции на правой границе интервала. Всегда полезно проконтролировать, что оно совпадает с соответствующим пристрелочным параметром (выведенным в последней строке листинга 10.3).

**Листинг 10.4. Решение краевой задачи (продолжение листинга 10.3)**

$$M := 10$$

$$S0 := \mathrm{rkfixed}\left[\begin{pmatrix} I0 \\ I1_0 \end{pmatrix}, 0, 0.5, M, D\right]$$

$$S1 := \mathrm{rkfixed}\left[\begin{pmatrix} S0_{M,1} \\ S0_{M,2} \end{pmatrix}, 0.5, 1, M, D\right]$$

$$S1_{M,2} = 13.801$$

Решение краевой задачи приведено на рис. 10.5. С физической точки зрения естественно, что интенсивность света уменьшается быстрее по мере распространения в более плотной среде в правой половине расчетного интервала. В средней точке xf=0.5, как и ожидалось, производные обоих решений имеют разрыв.

> ***Примечание***
>
> Еще один пример решения краевых задач с разрывными коэффициентами ОДУ приведен в справочной системе Mathcad.

Ради справедливости необходимо заметить, что разобранную краевую задачу легко решить и с помощью функции sbval, заменив в листинге 10.2 зависимость a(x) на третью строку листинга 10.3. В этом случае листинг 10.2 даст в

точности тот же ответ, что показан на рис. 10.5. Однако в определенных случаях (в том числе из соображений быстроты расчетов) удобнее использовать функцию `bvalfit`, т. е. вести пристрелку с обеих границ интервала.

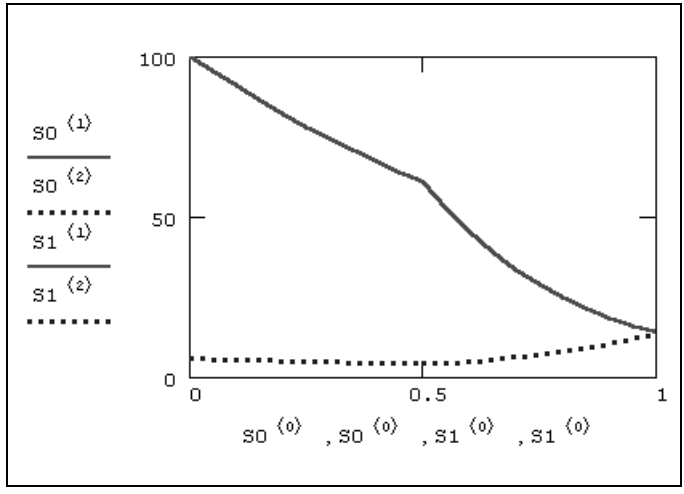



**Рис. 10.5.** Решение краевой задачи с разрывом в `xf=0.5` (продолжение листингов 10.3–10.4)

***Совет***

Если вы имеете дело с подобными уравнениями, попробуйте сначала решить их как обычную краевую задачу с помощью более надежной и легкой в применении функции `sbval`.

# 10.3. Задачи на собственные значения для ОДУ

Задачи на собственные значения — это краевые задачи для системы ОДУ, в которой правые части зависят от одного или нескольких параметров $\lambda$. Значения этих параметров неизвестны, а решение краевой задачи существует только при определенных $\lambda_k$, которые называются *собственными значениями* (eigenvalues) задачи. Решения, соответствующие этим $\lambda_k$, называют *собственными функциями* (eigenfunctions) задачи. Правильная постановка таких задач требует формулировки количества граничных условий, равного сумме числа уравнений и числа собственных значений. Физическими примерами

задач на собственные значения являются, например, уравнение колебаний струны, уравнение Шредингера в квантовой механике, уравнения волн в резонаторах и многие другие.

С вычислительной точки зрения, задачи на собственные значения очень похожи на рассмотренные выше краевые задачи. В частности, для многих из них также применим метод стрельбы (*см. разд. 10.1.2*). Отличие заключается в пристрелке не только по недостающим левым граничным условиям, но еще и по искомым собственным значениям. В Mathcad для решения задач на собственные значения используются те же функции `sbval` и `bvalfit`. В их первый аргумент, т. е. вектор, присваивающий начальные значения недостающим начальным условиям, следует включить и начальное приближение для собственного значения.

Рассмотрим методику решения на конкретном примере определения собственных упругих колебаний струны. Профиль колебаний струны `y(x)` описывается линейным дифференциальным уравнением второго порядка

$$\frac{d}{dx}\left[p(x)\cdot\frac{dy(x)}{dx}\right]+\lambda\cdot q(x)\cdot y(x)=0, \qquad (10.3)$$

где `p(x)` и `q(x)` — жесткость и плотность, которые, вообще говоря, могут меняться вдоль струны. Если струна закреплена на обоих концах, то граничные условия задаются в виде `y(0)=y(1)=0`. Сформулированная задача является частным случаем *задачи Штурма—Лиувилля*. Поскольку решается система двух ОДУ, содержащая одно собственное значение $\lambda$, то по идее задача требует задания трех (2+1) условий. Однако, как легко убедиться, уравнение колебаний струны — линейное и однородное, поэтому в любом случае решение `y(x)` будет определено с точностью до множителя. Это означает, что производную решения можно задать произвольно, например, `y'(0)=1`, что и будет третьим условием. Тогда краевую задачу можно решать как задачу Коши, а пристрелку вести только по одному параметру — собственному значению.

Процедура поиска первого собственного значения представлена в листинге 10.5.

**Листинг 10.5. Решение задачи о собственных колебаниях струны**

```
p(x) := 1    q(x) := 1    p'(x) := 0

a := 0    b := 1

λ0 := 0.5
```

$$D(x, y) := \begin{bmatrix} y_1 \\ -\frac{1}{p(x)} \cdot \left(p'(x) \cdot y_1 + y_2 \cdot q(x) \cdot y_0\right) \\ 0 \end{bmatrix}$$

$$load(a, \lambda) := \begin{pmatrix} 0 \\ 1 \\ \lambda_0 \end{pmatrix}$$

$$score(b, y) := y_0$$

$$\Lambda := sbval(\lambda, a, b, D, load, score)$$

$$\Lambda = (\ 9.87\ ) \qquad 1^2 \cdot \pi^2 = 9.87$$

В первых двух строках листинга определяются функции, входящие в задачу, в том числе `p'(x):=0`, и границы расчетного интервала `(0,1)`. В третьей строке дается начальное приближение к собственному значению $\lambda_0$, в четвертой вводится система ОДУ. Обратите внимание, что она состоит не из двух, а из трех уравнений. Первые два из них определяют эквивалентную (10.3) систему ОДУ первого порядка, а третье необходимо для задания собственного значения в виде еще одного компонента $y_2$ искомого вектора `y`. Поскольку по определению собственное значение постоянно при всех `x`, то его производная должна быть приравнена нулю, что отражено в последнем уравнении. Важно также, что во втором из уравнений собственное значение записано как $y_2$, поскольку является одним из неизвестных.

В следующих двух строках листинга задается левое граничное условие, включающее и недостающее условие на собственное значение для третьего уравнения, и правое граничное условие $y_0=0$. В предпоследней строке листинга обычным образом применяется функция `sbval`, а в последней выводится результат ее работы вместе с известным аналитически собственным значением $n^2 \cdot \pi^2$. Как легко убедиться, мы нашли первое собственное значение для `n=1`, а чтобы найти другие собственные значения, необходимо задать другие начальные приближения к ним (в третьей строке листинга 10.5). Например, выбор $\lambda_0=50$ приводит ко второму собственному значению $2^2 \cdot \pi^2$, а $\lambda_0=100$ — к третьему $3^2 \cdot \pi^2$.

Чтобы построить график соответствующей собственной функции, надо добавить в листинг строку, программирующую решение задачи Коши, например, такую: `U:=rkfixed(load(a,Λ),a,b,100,D)`. Полученные кривые показаны на рис. 10.6 в виде коллажа трех графиков, рассчитанных для трех собственных значений.

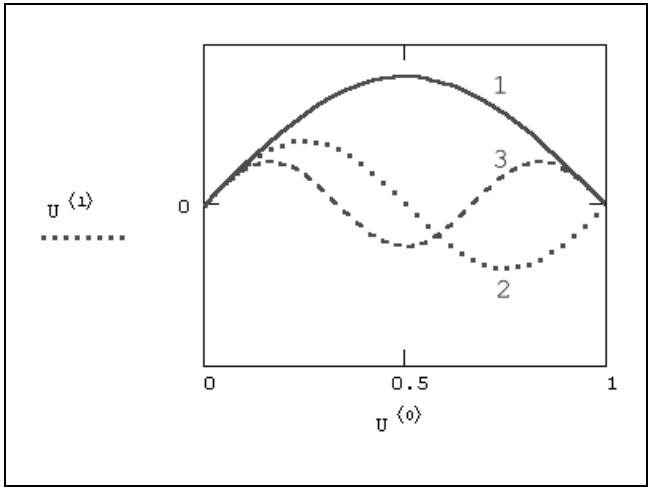



**Рис. 10.6.** Первые три собственные функции задачи колебаний струны (коллаж трех графиков, рассчитанных листингом 10.5 для разных $\lambda 0$)

***Примечание***

Примеры решения нескольких задач на собственные значения можно найти в различных ресурсах Mathcad.

# 10.4. Разностные схемы для ОДУ

Многие краевые задачи не поддаются решению методом стрельбы. Однако в Mathcad 12 других встроенных алгоритмов нет. Тем не менее это не означает, что по-другому решать краевые задачи невозможно, ведь другие численные алгоритмы несложно запрограммировать самому пользователю. Рассмотрим возможную реализацию наглядного метода, называемого разностным, которым можно решать краевые задачи как для ОДУ, так и для дифференциальных уравнений в частных производных.

## 10.4.1. О разностном методе

Разберем идею разностного метода решения краевых задач на примере взаимодействия световых пучков (см. рис. 10.1), переобозначив в системе (10.1) интенсивность излучения вправо на $Y$, а интенсивность излучения влево на $y$ (просто в целях удобства, чтобы не писать в дальнейшем индекс). Суть метода заключается в покрытии расчетного интервала сеткой из $N$ точек. Тем са-

мым определяются (N-1) шагов (рис. 10.7). Затем надо заменить дифференциальные уравнения исходной краевой задачи аппроксимирующими их уравнениями в конечных разностях, выписав соответствующие разностные уравнения для каждого i-го шага. В нашем случае достаточно просто заменить первые производные из (10.1) их разностными аналогами (такой метод называется еще *методом Эйлера*):

$$\begin{aligned} \frac{Y_{i+1} - Y_i}{\Delta} &= -a(x_i) \cdot Y_i + r(x_i) \cdot y_i; \\ \frac{y_{i+1} - y_i}{\Delta} &= a(x_i) \cdot y_i - r(x_i) \cdot Y_i. \end{aligned} \qquad (10.4)$$

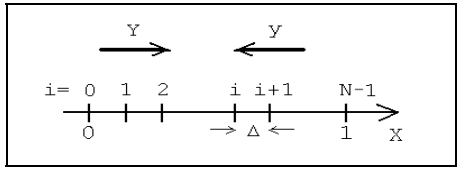



**Рис. 10.7.** Сетка, покрывающая расчетный интервал

### *Примечание*

Существует множество способов аппроксимации дифференциальных уравнений разностными. От выбора конкретного варианта зависит не только простота, быстрота и удобство вычислений, но и сама возможность получения правильного ответа.

Получилась система (по числу шагов) 2·(N-1) разностных линейных алгебраических уравнений с 2·N неизвестными $Y_i$ и $y_i$. Для того чтобы она имела единственное решение, надо дополнить число уравнений до 2·N.

Это можно сделать, записав в разностном виде оба граничных условия:

$$Y_0 = I0, \quad y_N = R \cdot Y_N. \qquad (10.5)$$

Сформированная полная система алгебраических уравнений называется *разностной схемой*, аппроксимирующей исходную краевую задачу. Обратите внимание, что правые части разностных уравнений системы (10.4) на каждом шаге записаны для левой границы шага. Такие разностные схемы называют *явными*, т. к. все значения $Y_{i+1}$ и $y_{i+1}$ находятся в левой части уравнений. Полученную явную разностную схему легко записать в матричной форме

$$A \cdot z = B, \qquad (10.6)$$

где z — неизвестный вектор, получающийся объединением векторов Y и y. Решив систему (10.6), мы получим решение краевой задачи.

***Примечание***

На самом деле все несколько сложнее, поскольку, вообще говоря, необходимо еще доказать, что, во-первых, разностная схема действительно аппроксимирует дифференциальные уравнения и, во-вторых, при $N\to\infty$ разностное решение действительно сходится к дифференциальному.

Процесс решения системы разностных уравнений называют также *реализацией* разностной схемы. Программа, которая решает рассматриваемую краевую задачу разностным методом, приведена в листинге 10.6.

**Листинг 10.6. Реализация явной разностной схемы**

```
a(x) := 1    r(x) := 0.01    I0 := 100    R := 1

N := 5                Δ := 1/(N − 1)

i := 1 .. N − 1

A_{i,i} := a(i·Δ − Δ)·Δ − 1    A_{i,i+1} := 1    A_{i,i+N} := −Δ·r(i·Δ − Δ)

i := N + 1 .. 2·N − 1

A_{i,i} := −a(i·Δ − Δ)·Δ − 1    A_{i,i+1} := 1    A_{i,i−N} := Δ·r(i·Δ − Δ)

A_{0,1} := 1

A_{N,N·2} := 1    A_{N,N} := −R

i := 1 .. 2·N − 1

B_i := 0

B_0 := I0

A := submatrix(A, 0, 2·N − 1, 1, 2·N)

U := lsolve(A, B)

U_N = 13.453        U_{N−1} = 31.78
```

Дадим минимальные комментарии, надеясь, что заинтересовавшийся читатель с карандашом в руках разберется в порядке индексов и соответствии матричных элементов, а возможно, составит и более удачную программу.

В первой строке листинга определяются функции и константы, входящие в модель, во второй задается число точек сетки `N=5` и ее равномерный шаг. Следующие две строки определяют матричные коэффициенты, аппроксимирующие уравнения для $Y_i$, а пятая и шестая — для $y_i$. Седьмая и восьмая строки листинга задают соответственно левое и правое граничное условие, а строки с девятой по одиннадцатую — правые части системы (10.6). В сле-

дующей строке завершается построение матрицы A вырезанием из нее левого нулевого столбца. В предпоследней строке листинга применена встроенная функция lsolve для решения системы (10.6), а в последней выведены рассчитанные ею неизвестные граничные значения. Графики решения приведены на рис. 10.8, причем первые N элементов итогового вектора есть вычисленное излучение вперед, а последние N элементов — излучение назад.

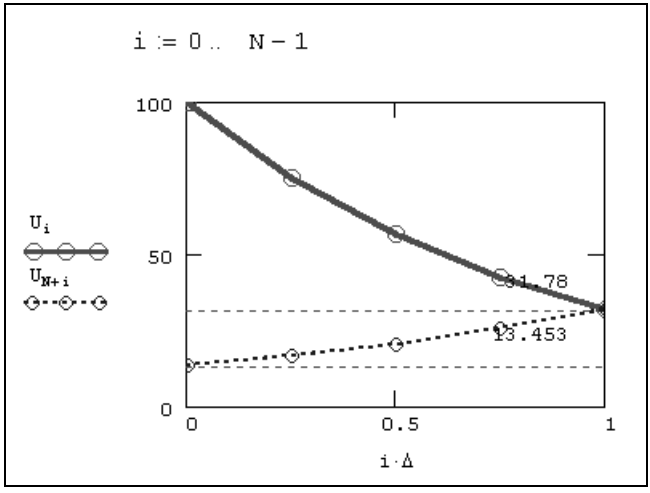



**Рис. 10.8.** Решение краевой задачи разностным методом (продолжение листинга 10.6)

Как мы увидели, реализация в Mathcad разностных схем вполне возможна и не слишком трудоемка — предложенная программа состоит всего из двух десятков математических выражений. Конечно, для их написания требуется и время, и часто кропотливые расчеты, но, собственно, в этом и состоит работа математика. Кстати говоря, при небольшом числе шагов расчеты по разностным схемам не требуют существенного времени (программа, приведенная в листинге 10.6, работает быстрее, чем метод стрельбы, встроенный в функцию sbval). Существуют, кроме того, весьма очевидные для многих читателей пути ускорения расчетов, связанные с применением более подходящих методов решения систем линейных уравнений с разреженной матрицей.

## 10.4.2. Жесткие краевые задачи

Один из случаев, когда применение разностных схем может быть очень полезным, связан с решением жестких краевых задач (подробнее о жестких

ОДУ читайте *в главе 11*). В частности, рассматриваемая задача о встречных световых пучках становится жесткой при увеличении коэффициента ослабления `a(x)` в несколько десятков раз. Например, при попытке решить ее с `a(x):=100` с помощью листинга 10.2 вместо ответа выдается сообщение об ошибке "Can't converge to a solution. Encountered too many integration steps" (Не сходится к решению. Слишком много шагов интегрирования). Это и неудивительно, поскольку жесткие системы характерны тем, что требуют исключительно малого значения шага в стандартных алгоритмах.

Для жестких задач неприменимы и явные разностные схемы, о которых рассказывалось в предыдущем разделе (*см. разд. 10.3.1*). Результат расчетов по программе листинга 10.6, например с `a(x):=20` (рис. 10.9), дает характерную для неустойчивых разностных схем "разболтку" — колебания нарастающей амплитуды, не имеющие ничего общего с реальным решением.

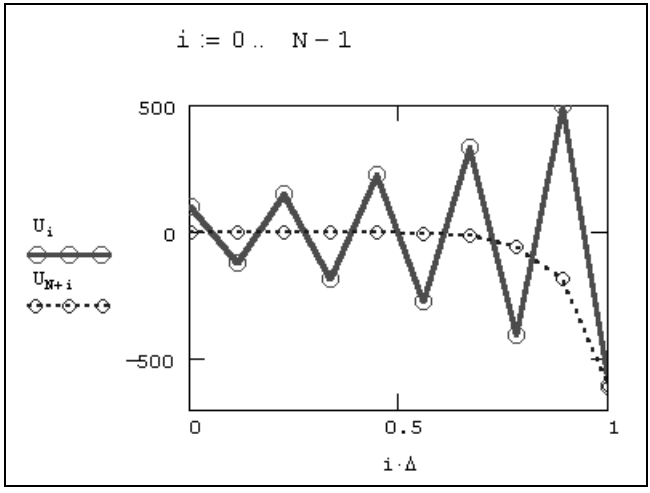



**Рис. 10.9.** Неверное решение жесткой краевой задачи по неустойчивой явной разностной схеме

Выходом из положения будет использование *неявных* разностных схем. Применительно к нашей задаче достаточно заменить правые части уравнений (10.1) значениями не на левой, а на правой границе каждого шага:

$$\begin{aligned} \frac{Y_{i+1} - Y_i}{\Delta} &= -a(x_{i+1}) \cdot Y_{i+1} + r(x_{i+1}) \cdot y_{i+1}; \\ \frac{y_{i+1} - y_i}{\Delta} &= a(x_{i+1}) \cdot y_{i+1} - r(x_{i+1}) \cdot Y_{i+1}. \end{aligned} \qquad (10.7)$$

Граничные условия, конечно, можно оставить в том же виде (10.5). Поскольку мы имеем дело с линейными дифференциальными уравнениями, то и схему (10.7) легко будет записать в виде матричного равенства (10.6), перегруппировывая соответствующим образом выражение (10.7) и приводя подобные слагаемые. Разумеется, полученная матрица `A` будет иной, нежели матрица `A` для явной схемы (10.4). Поэтому и решение (реализация неявной схемы) может отличаться от изображенного на рис. 10.9 результата расчетов по явной схеме. Программа, составленная для решения системы (10.7), приведена в листинге 10.7.

**Листинг 10.7. Реализация неявной разностной схемы для жесткой краевой задачи**

```
a(x) := 20    r(x) := 0.01    I0 := 100    R := 1

N := 20        Δ := 1/(N − 1)

i := 1 .. N − 1

A_{i,i} := 0 − 1    A_{i,i+N+1} := −Δ·r(i·Δ − Δ)    A_{i,i+1} := a(i·Δ − Δ)·Δ + 1

i := N + 1 .. 2·N − 1

A_{i,i} := 0 − 1    A_{i,i+1} := 1 − a(i·Δ − Δ)·Δ    A_{i,i−N+1} := Δ·r(i·Δ − Δ)

A_{0,1} := 1

A_{N,N·2} := 1    A_{N,N} := −R

i := 1 .. 2·N − 1

B_i := 0

B_0 := I0

A := submatrix(A, 0, 2·N − 1, 1, 2·N)

U := lsolve(A, B)

U_N = 0.017        U_{N−1} = 1.523 × 10^−6
```

Не будем специально останавливаться на обсуждении листинга 10.7, поскольку он почти в точности повторяет предыдущий листинг. Отличие заключается лишь в формировании матрицы `A` другим способом, согласно неявной схеме. Решение, показанное на рис. 10.10, демонстрирует, что произошло "небольшое чудо": "разболтка" исчезла, а распределение интенсивностей стало физически предсказуемым. Обратите внимание, что (из-за взятого нами слишком большого коэффициента ослабления излучения) отра-

женный пучок света имеет очень маленькую интенсивность, и ее пришлось построить на графике с увеличением в тысячу раз.

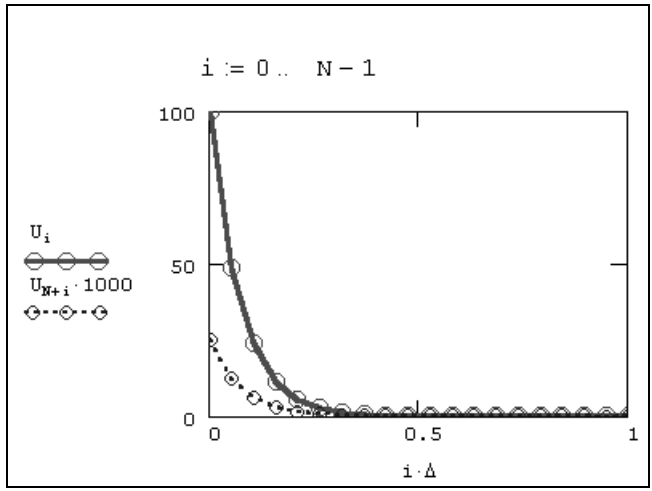



**Рис. 10.10.** Решение краевой задачи по неявной разностной схеме (продолжение листинга 10.7)

## 10.5. Нелинейные краевые задачи

Все примеры, приведенные пока в этом разделе, связаны с краевыми задачами для линейных ОДУ. Между тем на практике часто приходится исследовать именно нелинейные задачи, которые несравненно сложнее с вычислительной точки зрения. Рассмотрим оба подхода к решению нелинейных задач (алгоритм стрельбы и разностный метод), не забывая о том, что нелинейные ОДУ также могут быть в той или иной степени жесткими.

### 10.5.1. О постановке задачи

На примере решения нелинейной задачи взаимодействия световых пучков при помощи алгоритма стрельбы разберемся с физикой явления, которую описывает нелинейность. Рассмотрим простейшее усложнение модели (10.1) (*см. разд. 10.1*), добавив в уравнения малые слагаемые, зависящие квадратично от интенсивности света. Чтобы быть ближе к физической сущности явления, мы введем эту зависимость в формулу коэффициента поглощения

среды a(x), сделав a(x) из (10.1) функцией не только координаты, но и суммарной интенсивности:

$$a(x,Y,y)=a0(x)-\varepsilon(x)\cdot(Y+y), \tag{10.8}$$

где a0(x) — постоянная (не зависящая от интенсивности света) составляющая коэффициента поглощения, а ε(x) — малый коэффициент пропорциональности. В этом случае модель (10.1) перепишется в виде:

$$\begin{aligned}\frac{dY(x)}{dx} &= -a0(x)\cdot Y(x)+r(x)\cdot y(x)+\varepsilon(x)\cdot\left(Y^2(x)+Y(x)\cdot y(x)\right);\\ \frac{dy(x)}{dx} &= a0(x)\cdot y(x)-r(x)\cdot Y(x)-\varepsilon(x)\cdot\left(y^2(x)+Y(x)\cdot y(x)\right).\end{aligned} \tag{10.9}$$

Таким образом, физическая подоплека квадратичной нелинейности задачи (10.9) связана с зависимостью коэффициента поглощения среды от Y+y. Иными словами, чем мощнее свет в некоторой точке среды, тем меньше локальный коэффициент поглощения среды в этой точке.

Механизм этого явления будет хорошо понятен, если предположить, что средой является воздух, насыщенный парами жидкости (попросту говоря, туман). Чем сильнее в некоторой точке туман, тем больше локальное поглощение. Если свет, проходящий через туман, имеет малую интенсивность, то он не оказывает на среду никакого (или почти никакого) воздействия. Однако если пучок света будет очень мощным, за счет его поглощения пары воды будут разогреваться и, следовательно, испаряться. По мере испарения капель воды туман станет рассеиваться, и среда просветлеет (т. е. ее коэффициент поглощения уменьшится). Таким образом, чем мощнее свет, тем активнее пойдет процесс испарения, тем существеннее уменьшится коэффициент поглощения. Именно такой эффект, совершенно понятный с физической точки зрения, и описывает формула (10.8) и, соответственно, модель (10.9).

Подчеркнем еще раз и физический смысл коэффициентов a0(x) и ε(x). Первый описывает постоянный (не зависящий от интенсивности света) коэффициент поглощения среды, а второй — его зависимость от эффекта разогрева среды светом. Чем больше коэффициент a0(x), тем более жесткой является краевая задача; а чем больше ε(x), тем сильнее ее нелинейность. В предельном случае ε(x)→0 задача (10.9) переходит в задачу (10.1), становясь линейной.

## 10.5.2. Метод стрельбы

Встроенная функция sbval, реализующая в Mathcad алгоритм стрельбы (*см. разд. 10.2*), может справляться и с нелинейными задачами. Приведем пример решения краевой задачи (10.9) (листинг 10.8 и рис. 10.11) с теми же

граничными условиями, что были поставлены для модели (10.1). Интерес представляет решение y, описывающее рост интенсивности отраженного пучка по мере его распространения справа налево (обратите внимание, что на рис. 10.11 оно отложено с десятикратным увеличением).

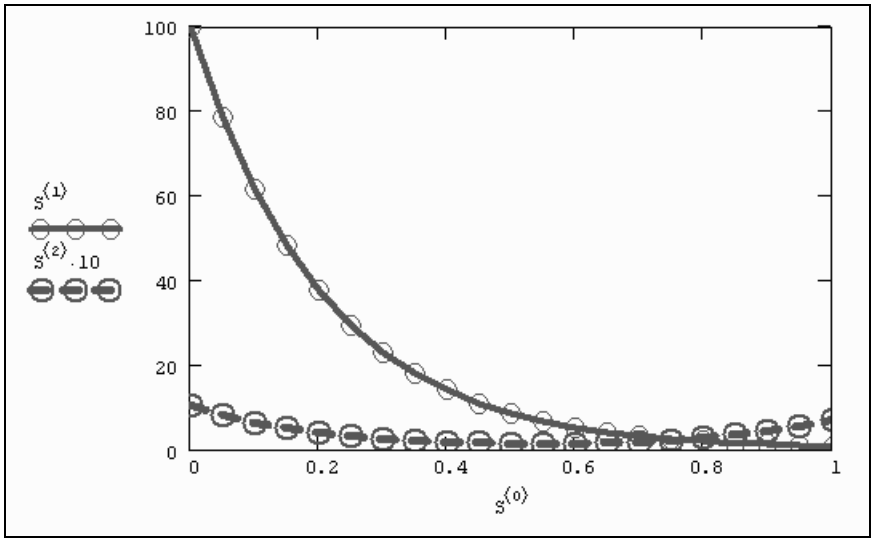



**Рис. 10.11.** Решение нелинейной краевой задачи (продолжение листинга 10.8)

**Листинг 10.8. Решение нелинейной краевой задачи методом стрельбы**

```
I0 := 100                    r(x) := .1

ε(x) := 0.002                a(x) := 5

R := 1

D(x, y) := [ -a(x)·y0 + r(x)·y1 + ε(x)·(y0 + y1)·y0 ]
           [  a(x)·y1 - r(x)·y0 - ε(x)·(y0 + y1)·y1 ]

z0 := 10

load(x0, z) := [ 100 ]
               [ z0  ]

score(x1, y) := R·y0 - y1

I1 := sbval(z, 0, 1, D, load, score)
```

```
I1 = ( 1.033 )
S := rkfixed[( I0 ), 0, 1, 20, D]
             ( I1_0 )
```

Следует помнить, что чем сильнее нелинейность и чем жестче краевая задача, тем более сильные требования предъявляются к начальному значению алгоритма, ввод которого осуществляется посредством функции `load`. Попробуйте повторить расчеты листинга 10.8 с иным сочетанием параметров `a0(x)` и `ε(x)`, и вы увидите, что с немного более жесткими задачами алгоритм стрельбы уже не справляется.

> **Примечание**
>
> На компакт-диске, прилагаемом к книге, вы найдете еще более интересное решение для другой задачи, с нелинейностью типа $(y+Y)^2$.

## 10.5.3. Разностные схемы

Решение краевых задач при помощи разностных схем связано с необходимостью разработки собственного алгоритма для каждой конкретной задачи. Как вы помните (*см. разд. 10.4*), в случае линейных уравнений в результате построения разностной схемы система алгебраических сеточных уравнений также получалась линейной. Это автоматически означало, что она имеет единственное решение, которое в большинстве случаев могло быть найдено стандартными численными методами. В ситуации, когда исходные дифференциальные уравнения нелинейны, система разностных уравнений тоже является нелинейной, и их решение существенно усложняется, хотя бы потому, что оно не является единственным. Поэтому подход к построению разностных схем нелинейных уравнений должен быть специфическим, но наградой за него станет решение задач, с которыми не справляется алгоритм стрельбы (например, жестких).

Подробная разработка алгоритмов и соответствующих программ Mathcad выходит далеко за пределы данной книги, поэтому мы конспективно представим один из приемов решения нелинейных краевых задач, сводящийся к их *линеаризации*. В общих чертах подход заключается в следующем. Предположим, что некоторое приближение (обозначим его `Y(x)` и `y(x)`) к решению нелинейной задачи (10.9) нам известно, и можно считать, что `Y→Y+Z` и `y→y+z`, где `Z` и `z` — близкие к нулю функции `x`. Тогда, пользуясь их малостью (по

сравнению с Y0 и y0), можно разложить нелинейные слагаемые в уравнениях (10.9) в ряд Тейлора по Z и z. Получим:

$$\begin{aligned} Y'+Z' &= -aY + ry + \varepsilon(Y^2 + Yy) - aZ + rz + 2\varepsilon Y(Z + z); \\ y'+z' &= \phantom{-}ay - rY - \varepsilon(y^2 + Yy) + az - rZ - 2\varepsilon y(Z + z). \end{aligned} \qquad (10.10)$$

Теперь, поскольку Y(x) и y(x) являются приближением к решению исходной задачи, то можно считать, что они (приблизительно) удовлетворяют и уравнению, и граничным условиям. Тогда, вычитая (10.9) из (10.10), получим краевую задачу для Z(x) и z(x):

$$\begin{aligned} Z' &= -aZ + rz + 2\varepsilon Y(Z + z); \\ z' &= \phantom{-}az - rZ - 2\varepsilon y(Z + z); \\ Z(0) &= z(0) = Z(1) = z(1) = 0. \end{aligned} \qquad (10.11)$$

Это и есть та самая новая задача для "маленьких" функций Z(x) и z(x), которую надо решить, и которая, благодаря малости Z и z, является линейной. Вся беда в том, что мы не знаем "больших" функций Y(x) и y(x), а они, увы, входят в задачу (10.11). Тем не менее рецепт получения этих функций напрашивается сам собой: если нелинейность исходных ОДУ не слишком сильная, то в качестве Y и y можно взять решение линейной краевой задачи, т. е. задачи (10.9) с ε(x)=0 (*см. разд. 10.4.1*).

Сказанное иллюстрируют листинги 10.9 и 10.10, первый из которых решает линейную краевую задачу (являясь, фактически, небольшой модификацией листинга 10.7 из *разд. 10.4.2*), а второй решает линеаризованную задачу (10.11), учитывая результат листинга 10.9. В листинге 10.9 матрица D и вектор правых частей B являются разностной аппроксимацией ОДУ (ее первые N строк аппроксимируют первое уравнение, а оставшиеся N строк — второе). Такой же смысл и точно такую же структуру имеют матрица C и вектор правых частей G для второй (линеаризованной) задачи (10.11). Для решения сеточных уравнений Dy=B и Cz=G используется (конечно, весьма неэкономично) встроенная функция lsolve, реализующая алгоритм Гаусса.

Важно привлечь внимание читателя к последним строкам листинга 10.9. В них осуществляется интерполяция полученного решения системы сеточных уравнений для того, чтобы в нелинейной задаче (в листинге 10.10) можно было использовать непрерывные "большие" функции из линейной задачи. В последней строке листинга 10.10 осуществляется сложение "больших" и "маленьких" функций (результатов листингов 10.9 и 10.10) для получения полного решения нелинейной задачи (10.9), которое изображено на рис. 10.12.

Не будем давать дополнительных комментариев к Mathcad-программам, надеясь, что читатель, заинтересовавшийся нелинейным примером со световыми пучками и эффектом разогрева светом среды, сам разберется в листингах,

тем более что техника разработки разностных схем была нами детально разобрана раньше (*см. разд. 10.4*).

**Листинг 10.9. Решение линейной (приближенной) краевой задачи**

$I0 := 100$ $R := 1$ $\varepsilon(x) := 0.001$

$a(x) := 5$ $r(x) := .1$

$N := 20$ $\Delta := \frac{1}{N-1}$

$D_{0,1} := 1$ $D_{N,N\cdot 2} := 1$ $D_{N,N} := -R$

$i := 1 .. N-1$

$D_{i,i} := a(i\cdot\Delta - \Delta)\cdot\frac{\Delta}{2} - 1$ $D_{i,i+1} := a(i\cdot\Delta - \Delta)\cdot\frac{\Delta}{2} + 1$

$D_{i,i+N+1} := \frac{-\Delta}{2}\cdot r(i\cdot\Delta - \Delta)$ $D_{i,i+N} := \frac{-\Delta}{2}\cdot r(i\cdot\Delta - \Delta)$

$i := N+1 .. 2\cdot N-1$

$D_{i,i} := -a(i\cdot\Delta - \Delta)\cdot\frac{\Delta}{2} - 1$ $D_{i,i+1} := 1 - a(i\cdot\Delta - \Delta)\cdot\frac{\Delta}{2}$

$D_{i,i-N} := \frac{\Delta}{2}\cdot r(i\cdot\Delta - \Delta)$ $D_{i,i-N+1} := \frac{\Delta}{2}\cdot r(i\cdot\Delta - \Delta)$

$B_0 := I0$

$i := 1 .. 2\cdot N-1$

$B_i := 0$

$D := \text{submatrix}(D, 0, 2\cdot N-1, 1, 2\cdot N)$

$I := \text{lsolve}(D, B)$

$i := 0 .. N-1$

$Z1_i := i\cdot\Delta$ $Z2_i := I_i$

$Z3_i := I_{i+N}$ $Z4_i := I_i + I_{i+N}$

$KS1 := \text{cspline}(Z1, Z2)$ $I1(z) := \text{interp}(KS1, Z1, Z2, z)$

$KS2 := \text{cspline}(Z1, Z3)$ $I2(z) := \text{interp}(KS2, Z1, Z3, z)$

$KS3 := \text{cspline}(Z1, Z4)$ $S(z) := \text{interp}(KS3, Z1, Z4, z)$

$DI1(x) := \frac{d}{dx}I1(x)$ $DI2(x) := \frac{d}{dx}I2(x)$

**Листинг 10.10. Решение линеаризованной задачи (продолжение листинга 10.9)**

$C_{0,1} := 1 \qquad C_{N,N\cdot 2} := 1 \qquad C_{N,N} := -R$

$G_0 := 0 \qquad G_N := R\cdot I1(1) - I2(1)$

$i := 1 \mathinner{..} N-1$

$C_{i,i} := 0 - 1$

$C_{i,i+1} := a(i\cdot\Delta)\cdot\Delta - 2\cdot\varepsilon(i\cdot\Delta)\cdot S(i\cdot\Delta)\cdot\Delta + 1$

$C_{i,i+N+1} := -\Delta\cdot r(i\cdot\Delta) - 2\cdot\varepsilon(i\cdot\Delta)\cdot S(i\cdot\Delta)\cdot\Delta$

$$G_i := \Delta\cdot\left(-1\cdot a(i\cdot\Delta)\cdot I1(i\cdot\Delta) + 1\cdot r(i\cdot\Delta)\cdot I2(i\cdot\Delta) + \varepsilon(i\cdot\Delta)\cdot S(i\cdot\Delta)^2\right) +$$

$$+ -1\cdot\Delta\cdot DI1(i\cdot\Delta)$$

$i := N+1 \mathinner{..} 2\cdot N-1$

$C_{i,i} := 0 - 1$

$C_{i,i+1} := 1 - a(i\cdot\Delta)\cdot\Delta + 2\cdot\varepsilon(i\cdot\Delta)\cdot S(i\cdot\Delta - N\cdot\Delta)\cdot\Delta$

$C_{i,i-N+1} := \Delta\cdot r(i\cdot\Delta) + 2\cdot\varepsilon(i\cdot\Delta)\cdot S(i\cdot\Delta - N\cdot\Delta)\cdot\Delta$

$$G_i := \Delta\cdot\left(a(i\cdot\Delta)\cdot I2(i\cdot\Delta - N\cdot\Delta)\cdot 1 - 1\cdot r(i\cdot\Delta)\cdot I1(i\cdot\Delta - N\cdot\Delta) -\right.$$

$$\left. - \varepsilon(i\cdot\Delta)\cdot S(i\cdot\Delta - N\cdot\Delta)^2\right) + -1\cdot\Delta\cdot DI2(i\cdot\Delta - N\cdot\Delta)$$

$C := \mathrm{submatrix}(C, 0, 2\cdot N-1, 1, 2\cdot N)$

$JK := \mathrm{lsolve}(C, G)$

$i := 0 \mathinner{..} N-1$

$J_i := JK_i + I1(i\cdot\Delta)$

$J_{N+i} := JK_{N+i}\cdot 1 + I2(i\cdot\Delta)$

Последний важный момент, который следует обозначить, связан с решением задач, обладающих значительной нелинейностью. Решение, приведенное в листингах 10.9, 10.10 и на рис. 10.12, согласно самой постановке, должно не сильно отличаться от решения линейной краевой задачи, поскольку функции `Z(x)` и `z(x)` малы по сравнению с `Y(x)` и `y(x)`. Если же нелинейность сильная, то решение может существенно отличаться от `Y` и `y`, и линеаризация (10.11) будет просто неправильной. В этом случае следует слегка усложнить алгоритм решения нелинейной краевой задачи.

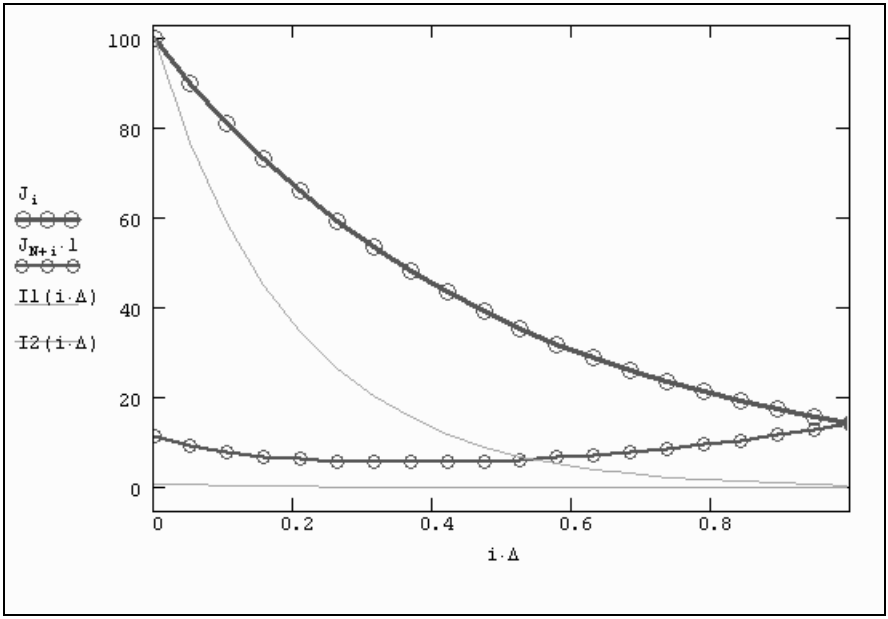



**Рис. 10.12.** Решение нелинейной краевой задачи разностным методом (продолжение листингов 10.9 и 10.10)

Обозначим полученное в результате решение, как и в листинге 10.10, вектором `J(ε)`, подчеркивая тем самым его зависимость от параметра нелинейности. Очевидно, что `J(0)` есть решение линейной задачи. Для того чтобы решить задачу с сильной нелинейностью, т. е. довольно большим `ε=ε1`, можно организовать продолжение по ε как по параметру. Иными словами, используя в качестве начального приближения `J(0)`, можно решить задачу для другого, малого `ε=Δε`, получив `J(Δε)`, затем, взяв это `J(Δε)` в качестве приближенного решения, получить `J(2Δε)` и т. д. малыми шагами добраться до желаемого `ε1`.

### *Примечание*

Упрощенную реализацию этого алгоритма вы найдете на компакт-диске, прилагаемом к книге. Она связана с выводом во внешний файл данных результата задачи из листинга 10.10 и считыванием из него же этих данных в качестве входной информации для следующей итерации. В качестве нулевой итерации используется решение линейной задачи, выводимое предварительно в файл из усовершенствованного листинга 10.9.

# Глава 11

# Дифференциальные уравнения в частных производных

Дифференциальные уравнения в частных производных представляют собой одну из наиболее сложных и одновременно интересных задач вычислительной математики. Эти уравнения характеризуются тем, что для их решения не существует единого универсального алгоритма, и большинство задач требует своего собственного особого подхода. Уравнениями в частных производных описывается множество разнообразных физических явлений, и с их помощью можно с успехом моделировать самые сложные явления и процессы (диффузия, гидродинамика, квантовая механика, экология и т. д.).

Дифференциальные уравнения в частных производных требуют нахождения функции не одной, как для ОДУ, а нескольких переменных, например, `f(x,y)` или `f(x,t)`. Постановка задач (*см. разд. 11.1*) включает в себя само уравнение (или систему уравнений), содержащее производные неизвестной функции по различным переменным (частные производные), а также определенное количество краевых условий на границах расчетной области.

Несмотря на то, что Mathcad обладает довольно ограниченными возможностями по отношению к уравнениям в частных производных, в нем имеется несколько встроенных функций (*см. разд. 11.3*). Решать уравнения в частных производных можно и путем непосредственного программирования пользовательских алгоритмов (*см. разд. 11.2*). Автор совершенно сознательно сначала рассматривает численные методы для решения уравнений в частных производных, а уже затем описывает предназначенные для этого встроенные функции, чтобы читатель ясно осознавал, каким образом Mathcad производит расчеты. "Слепое" использование встроенных функций для решения уравнений в частных производных не всегда бывает успешным, и ответственность за верный выбор их параметров часто ложится на пользователя, которому

необходимо четко представлять основные принципы функционирования численных алгоритмов, примененных во встроенных функциях.

## 11.1. О постановке задач

Остановимся сначала на общей классификации уравнений в частных производных, принятой в математике, а затем более детально поговорим о постановочной части соответствующих задач на примере различных вариаций уравнения диффузии тепла, которое допускает наглядную физическую интерпретацию.

### 11.1.1. Классификация уравнений в частных производных

Постановка задач для уравнений в частных производных включает определение самого уравнения (или системы нескольких уравнений), а также необходимого количества краевых условий (число и характер задания которых определяются спецификой уравнения). По своему названию уравнения должны содержать частные производные неизвестной функции `u` (или нескольких функций, если уравнений несколько) по различным аргументам, например, пространственной переменной `x` и времени `t`. Соответственно, для решения задачи требуется вычислить функцию нескольких переменных, например, `u(x,t)` в некоторой области определения аргументов `0≤x≤L` и `0≤t≤T`. Граничные условия определяются как заданные временные зависимости функции `u`, или производных этой функции, на границах расчетной области `0` и `L`, а начальные — как заданная `u(x,0)`.

Сами уравнения в частных производных (несколько условно) можно разделить на три основных типа:

- *параболические* — содержащие первую производную по одной переменной и вторую — по другой, причем все эти производные входят в уравнение с одинаковым знаком;
- *гиперболические* — содержащие первую производную по одной переменной и вторую — по другой, входящие в уравнение с разными знаками;
- *эллиптические* — содержащие только вторые производные, причем одного знака.

Некоторые более сложные уравнения нельзя однозначно подогнать под приведенную классификацию, тогда говорят о гибридных типах уравнений.

## 11.1.2. Пример: уравнение диффузии тепла

На протяжении всей главы мы будем использовать в качестве примера очень наглядное и имеющее различные, от очевидных до самых неожиданных, решения уравнение теплопроводности.

### Двумерное динамическое уравнение

Рассмотрим следующее параболическое уравнение в частных производных, зависящее от трех переменных — двух пространственных `x` и `y`, а также от времени `t`:

$$\frac{\partial u(x, y, t)}{\partial t} = D\left(\frac{\partial^2 u(x, y, t)}{\partial x^2} + \frac{\partial^2 u(x, y, t)}{\partial y^2}\right) + \phi(x, y, t, u). \qquad (11.1)$$

***Примечание***

Выражение в скобках в правой части уравнения (сумму вторых пространственных производных функции `u`) часто, ради краткости, обозначают при помощи *оператора Лапласа*: $\Delta u$.

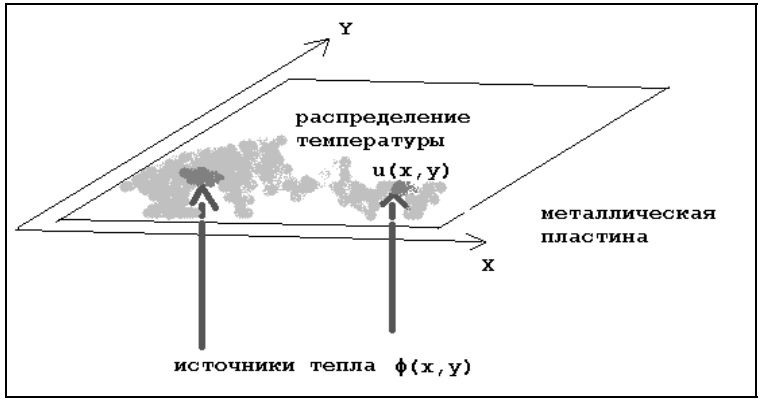



**Рис. 11.1.** Физическая модель, описываемая двумерным уравнением теплопроводности

Это уравнение называется *двумерным уравнением теплопроводности* или, по-другому, *уравнением диффузии тепла*. Оно описывает динамику распределения температуры `u(x,y,t)` на плоской поверхности (например, на металлической пластине) в зависимости от времени (рис. 11.1). Физический смысл коэффициента `D`, который, вообще говоря, может быть функцией как коорди-

нат, так и самой температуры заключается в задании скорости перетекания тепла от более нагретых областей в менее нагретые. Функция $\phi(x,y,t,u)$ описывает приток тепла извне, т. е. источники тепла, которые также могут зависеть как от пространственных координат (что задает локализацию источников), так и от времени и от температуры $u$.

Для того чтобы правильно поставить краевую задачу для двумерного уравнения теплопроводности, следует определить следующие дополнительные условия:

- граничные условия, т. е. динамику функции $u(x,y,t)$ и (или) ее производных на границах расчетной области;
- начальное условие, т. е. функцию $u(x,y,t)$.

> ***Примечание***
>
> Если рассматривается не одно уравнение в частных производных, а система уравнений, то соответствующие начальные и граничные условия должны быть поставлены для каждой из неизвестных функций.

## Стационарное двумерное уравнение

Частный случай уравнения теплопроводности определяет стационарную, т. е. не зависящую от времени, задачу. Стационарное уравнение описывает физическую картину распределения температуры по пластине, не изменяющуюся с течением времени. Такая картина может возникнуть при условии, что стационарный источник тепла действует довольно продолжительное время, и переходные процессы, вызванные его включением, прекратились. Пример численного решения такого уравнения показан на рис. 11.2 в виде поверхности $u(x,y)$.

Как несложно сообразить, если искомая функция не зависит от времени, то частная производная по времени в левой части уравнения равна нулю, и само уравнение можно переписать (переобозначив ради упрощения $\phi \leftarrow \phi/D$) следующим образом:

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = -\phi(x,y,u). \qquad (11.2)$$

Полученное уравнение, согласно классификации предыдущего раздела, является эллиптическим. Его называют *уравнением Пуассона*, а для его решения в Mathcad предусмотрены две встроенные функции. Если, к тому же, источники равны нулю, то уравнение (11.2), принимающее вид $\Delta u=0$, называют *уравнением Лапласа*.

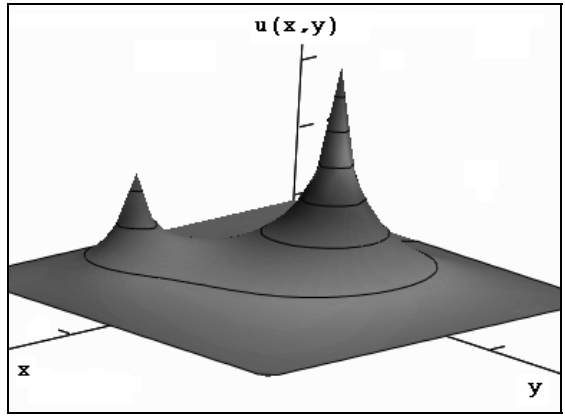



**Рис. 11.2.** Решение стационарного двумерного уравнения теплопроводности (см. листинг 11.7 ниже)

## Одномерное динамическое уравнение

Предположим, что мы рассматриваем задачу распределения тепла не по плоской поверхности, а по удлиненному телу типа металлического стержня (рис. 11.3). В этом случае зависимость от координаты `y` в общем уравнении теплопроводности пропадает, и получается одномерное уравнение:

$$\frac{\partial u(x,t)}{\partial t} = D\frac{\partial^2 u(x,t)}{\partial x^2} + \phi(x,t,u). \qquad (11.3)$$

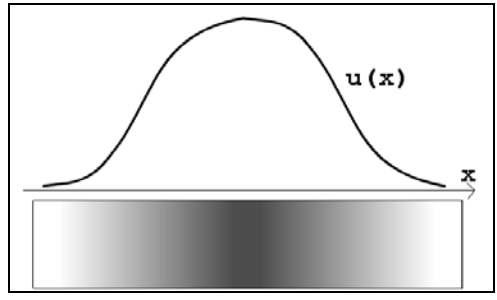



**Рис. 11.3.** Физическая модель одномерного уравнения теплопроводности

Одномерное уравнение намного проще двумерного, поскольку объем вычислений для реализации алгоритма его численного решения не так велик. Ти-

пичное решение одномерного уравнения диффузии тепла с коэффициентом диффузии D=2, нулевым источником ϕ=0 и начальным распределением температуры в форме нагретой центральной области стержня показано (в виде графика поверхности) на рис. 11.4.

Начиная с версии Mathcad 11, для решения одномерных параболических и гиперболических уравнений можно применять встроенную функцию pdesolve.

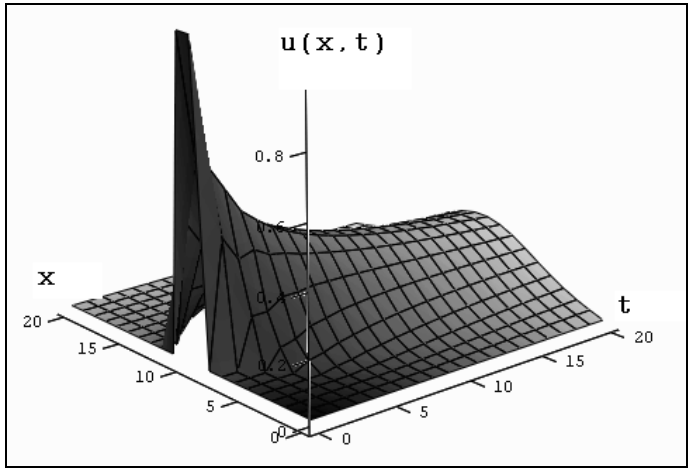



**Рис. 11.4.** Решение одномерного уравнения теплопроводности (см. листинг 11.1 ниже)

## Линейное и нелинейное уравнения

Если присмотреться к уравнению диффузии тепла внимательнее, то можно условно разделить практические случаи его использования на два типа.

- Линейная задача — если коэффициент диффузии D не зависит от температуры u и, кроме того, если источник тепла ϕ либо также не зависит от u, либо зависит от u линейно. В этом случае неизвестная функция u(x,t) и все ее производные входят в уравнение только в первой степени (линейно).
- Нелинейная задача — если уравнение имеет нелинейную зависимость от u(x,t), т. е. или коэффициент диффузии зависит от u, и (или) источник тепла нелинейно зависит от u.

Решения линейных уравнений в частных производных, как правило, получаются вполне предсказуемыми, и их часто можно получить аналитически

(этим проблемам посвящены соответствующие разделы науки, называемой математической физикой). В случае уравнения теплопроводности линейная задача описывает физически ожидаемое решение, выражающее остывание пластины или стержня в форме перетекания тепла от нагретого центра к холодной периферии.

Нелинейные уравнения, напротив, могут демонстрировать самые неожиданные решения, причем в подавляющем большинстве практических задач их можно получить только численно, а никак не аналитически.

***Примечание***

Различные линейные и нелинейные варианты рассматриваемого уравнения теплопроводности описывают различные модели физических сред, которые характеризуются определенными зависимостями `D(u)` и $\phi$`(u)`. В частности, для металлов в большинстве случаев можно считать, что `D=const`, в то время как для плазмы имеется специфическая зависимость коэффициента диффузии от температуры.

## Обратное уравнение теплопроводности

Замечательными свойствами обладает так называемое обратное уравнение диффузии тепла, которое получается путем замены в исходном (прямом) уравнении переменной `t` на `-t`. Согласно постановке задачи, обратное уравнение теплопроводности описывает реконструкцию динамики профиля температуры остывающего стержня, если известно начальное условие в виде профиля температуры в некоторый момент времени после начала остывания. Таким образом, требуется определить, как происходило остывание стержня. Мы ограничимся самым простым линейным уравнением с `D=const` без источников тепла:

$$\frac{\partial u(x,t)}{\partial t} = -D\frac{\partial^2 u(x,t)}{\partial x^2}. \qquad (11.4)$$

Это уравнение гиперболического типа и оно, несмотря на кажущуюся близость к рассмотренным вариантам уравнения теплопроводности, обладает замечательными свойствами.

Если попробовать осуществить расчет обратного уравнения диффузии тепла по тем же самым алгоритмам, что и для обычных уравнений (для этого достаточно в листинге 11.1 или 11.2 заменить значение коэффициента диффузии на отрицательное число, например, `D=-1`), то мы получим заведомо нефизичное решение. Оно показано на рис. 11.5 в виде профилей распределения температуры для нескольких последовательных моментов времени. Как видно, решение выражается в появлении все более быстрых пространственных ос-

цилляций профиля температуры для каждого нового момента времени. Очень существенно, что такое решение является не проявлением неустойчивости численного алгоритма (как для ситуации, рассмотренной в *разд. "Устойчивость" этой главы*), а определяется спецификой самой задачи.

Оказывается, что обратное уравнение теплопроводности принадлежит к довольно широкому классу задач, называемых некорректными. Некорректные задачи нельзя решать стандартными методами, а для того, чтобы с ними справиться (т. е., чтобы получить осмысленное физическое решение), приходится несколько менять саму их постановку, вводя в нее дополнительную априорную информацию о строении решения.

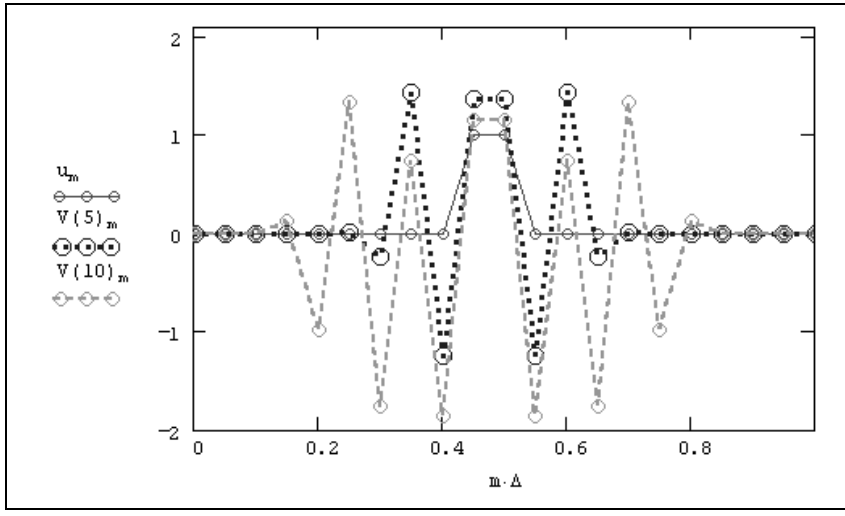



**Рис. 11.5.** Численное решение обратного уравнения теплопроводности дает совершенно нефизичную картину динамики температуры (см. листинг 11.2 ниже с параметром `D=-1`)

# ⓘ 🛠 11.2. Разностные схемы

Рассмотрим одномерное уравнение теплопроводности (11.3) и на его примере разберем наиболее часто использующийся для численного решения уравнений в частных производных *метод сеток*. Выпишем еще раз само уравнение

$$\frac{\partial u(x,t)}{\partial t} = D\,\frac{\partial^2 u(x,t)}{\partial x^2} + \phi(x,t,u), \qquad (11.5)$$

а также и начальное

$$u(x,0) = u_0(x) \tag{11.6}$$

и граничные условия

$$u(0, t) = f_0(t), \quad u(L, t) = f_1(t), \tag{11.7}$$

которые необходимы для правильной с математической точки зрения постановки задачи.

Основная идея численного решения уравнений в частных производных очень похожа на метод решения краевых задач для ОДУ, рассмотренный нами в предыдущей главе. Основным отличием от ОДУ является необходимость дискретизации уравнения не по одной, а по нескольким переменным (в зависимости от размерности задачи).

Таким образом, сначала следует покрыть расчетную область $(x,t)$ *сеткой* и использовать затем узлы этой сетки для разностной аппроксимации уравнения. В результате, вместо поиска непрерывных зависимостей $u(x,t)$ достаточно будет отыскать значения функции в узлах сетки (а ее поведение в промежутках между узлами может быть получено при помощи построения какой-либо интерполяции). По этой причине дискретное представление функции $u$ часто называют *сеточной функцией*.

Поскольку уравнения в частных производных по определению зависят от производных неизвестных функций по нескольким переменным, то способов дискретизации этих уравнений может быть, как правило, несколько. Конфигурацию узлов, используемую для разностной записи уравнений в частных производных на сетке, называют *шаблоном*, а полученную систему разностных уравнений — *разностной схемой*. О принципах построения разностных схем и, в частности, о классах явных и неявных схем мы уже подробно говорили на примере краевых задач для ОДУ (*см. разд. 10.4.1*), поэтому, излишне не повторяясь, перейдем к рассмотрению типичных особенностей уравнений в частных производных, которые возникают при разработке и реализации разностных схем.

## 11.2.1. Явная схема Эйлера

Рассмотрим сначала математические аспекты построения разностной схемы для уравнения диффузии тепла, а затем приведем примеры работы разработанного алгоритма применительно к линейному и нелинейному уравнениям.

## Построение разностной схемы

Используем для решения уравнения теплопроводности шаблон, изображенный на рис. 11.6. Для дискретизации второй производной по пространственной координате необходимо использовать три последовательных узла, в то время как для разностной записи первой производной по времени достаточно двух узлов. Записывая на основании данного шаблона дискретное представление для `(i,k)`-го узла, получим разностное уравнение:

$$\frac{u_{i,k+1} - u_{i,k}}{\Delta_t} = D\,\frac{u_{i-1,k} - 2\,u_{i,k} + u_{i+1,k}}{\Delta_x^2} + \phi_{i,k}. \qquad (11.8)$$

Приведем в разностной схеме (11.8) подобные слагаемые, перенеся в правую часть значения сеточной функции с индексом `k` (как часто говорят, с предыдущего *слоя* по времени), а в левую — с индексом `k+1` (т. е. со следующего временного слоя). Кроме этого, введем коэффициент `C`, который будет характеризовать отношение шагов разностной схемы по времени и пространству:

$$C_{i,k} = \frac{2 \cdot \Delta_t \cdot D_{i,k}}{\Delta_x^2}.$$

Несколько забегая вперед, заметим, что значение параметра `C`, называемого *коэффициентом Куранта*, имеет большое значение для анализа устойчивости разностной схемы. С учетом этих замечаний, разностная схема (11.8) запишется в виде:

$$u_{i,k+1} = \frac{C_{i,k}}{2}\,u_{i-1,k} + \left(1 - C_{i,k}\right)u_{i,k} + \frac{C_{i,k}}{2}\,u_{i+1,k} + \phi_{i,k}. \qquad (11.9)$$

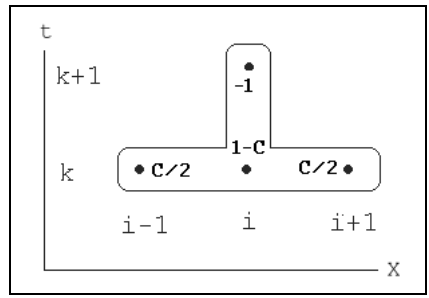



**Рис. 11.6.** Шаблон аппроксимации явной схемы для уравнения теплопроводности

Множители для каждого из значений сеточной функции в узлах шаблона, соответствующие разностному уравнению (11.9), приведены рядом с каждой

точкой шаблона на рис. 11.6. Фактически геометрия шаблона и эти множители задают построенную нами разностную схему.

Несложно убедиться в том, что для получения замкнутой системы разностных алгебраических уравнений систему (11.9) необходимо дополнить дискретным представлением начального и граничных условий (11.6) и (11.7). Тогда число неизвестных будет в точности равно числу уравнений, и процесс формирования разностной схемы будет окончательно завершен.

> ***Примечание***
>
> Важно подчеркнуть, что возможная нелинейность полученной системы алгебраических уравнений определяется зависимостями от температуры функций `D(u)` и $\phi$`(u)`, т. е. как коэффициент диффузии, так и источник тепла могут быть функциями сеточной функции $u_{ik}$.

Если присмотреться к разностным уравнениям (11.9) повнимательнее, то можно сразу предложить несложный алгоритм реализации этой разностной схемы. Действительно, каждое неизвестное значение сеточной функции со следующего временного слоя, т. е. левая часть соотношения (11.9) явно выражается через три ее значения с предыдущего слоя (правая часть), которые уже известны. Таким образом, в случае уравнения теплопроводности нам очень повезло. Для расчета первого слоя по времени следует попросту подставить в (11.9) начальное условие (известные значения `u` с нулевого слоя в узлах сетки), для расчета второго слоя достаточно использовать вычисленный таким образом набор `u` с первого слоя и т. д. Из-за того что разностная схема сводится к такой явной подстановке, ее и называют *явной*, а благодаря пересчету значений с текущего слоя через ранее вычисленные слои — *схемой бегущего счета*.

## Линейное уравнение

Сделанные замечания относительно реализации явной схемы для уравнения диффузии тепла сразу определяют алгоритм ее программирования в Mathcad. Для решения задачи нужно аккуратно ввести в листинг соответствующие формулы при помощи элементов программирования.

Решение системы разностных уравнений (11.9) для модели без источников тепла, т. е. $\phi$`(x,T,t)=0` и постоянного коэффициента диффузии `D=const` приведено в листинге 11.1. В его первых трех строках заданы шаги по временной и пространственной переменным $\tau$ и $\Delta$, а также коэффициент диффузии `D`, равный единице. В следующих двух строках заданы начальные (нагретый центр области) и граничные (постоянная температура на краях) условия соответственно. Затем приводится возможное программное решение разност-

ной схемы, причем функция пользователя `V(t)` задает вектор распределения искомой температуры в каждый момент времени (иными словами, на каждом слое), задаваемый целым числом `t`.

**Листинг 11.1. Явная схема для линейного уравнения теплопроводности**

$\tau := 0.0005 \qquad M := 20$

$\Delta := \frac{1}{M}$

$D(u) := 1$

$\phi(x, u) := 0$

$Init(x) := \Phi(x - 0.45) - \Phi(x - 0.55)$

$Border(t) := 0$

$m := 0 .. M$

$u_m := Init(m \cdot \Delta)$

$$F(v) := \left| \begin{array}{l} v1_0 \leftarrow Border(\tau \cdot T) + 0. \cdot (v_0 + v_1) \\ v1_M \leftarrow Border(\tau \cdot T) + 0. \cdot (v_M + v_{M-1}) \\ \text{for } m \in 1 .. M - 1 \\ \quad v1_m \leftarrow \phi(m \cdot \Delta, v_m) \cdot \tau + v_{m-1} \cdot \frac{D(v_{m-1}) \cdot \tau}{\Delta^2} + v_m \cdot \left(1 - \frac{2 \cdot D(v_m) \cdot \tau}{\Delta^2}\right) + v_{m+1} \cdot \frac{D(v_{m+1}) \cdot \tau}{\Delta^2} \\ v1 \end{array} \right.$$

$$V(t) := \left| \begin{array}{ll} u & \text{if } t = 0 \\ F(V(t-1)) & \text{otherwise} \end{array} \right.$$

Начальное распределение температуры вдоль расчетной области и решение для двух моментов времени показано на рис. 11.7 сплошной, пунктирной и штриховой линиями соответственно. Физически такое поведение вполне естественно — с течением времени тепло из более нагретой области перетекает в менее нагретую, а зона изначально высокой температуры остывает и размывается.

### *Примечание*

Несколько забегая вперед, заметим, что показанное на рис. 11.7 решение и соответствует коэффициенту Куранта `C=0.4`. Попробуйте осуществить расчет с увеличенным временным шагом, чтобы коэффициент `C` был больше 1, и посмотрите, что из этого получится (такой расчет и его объяснение приведены ниже *в разд. "Устойчивость"*).

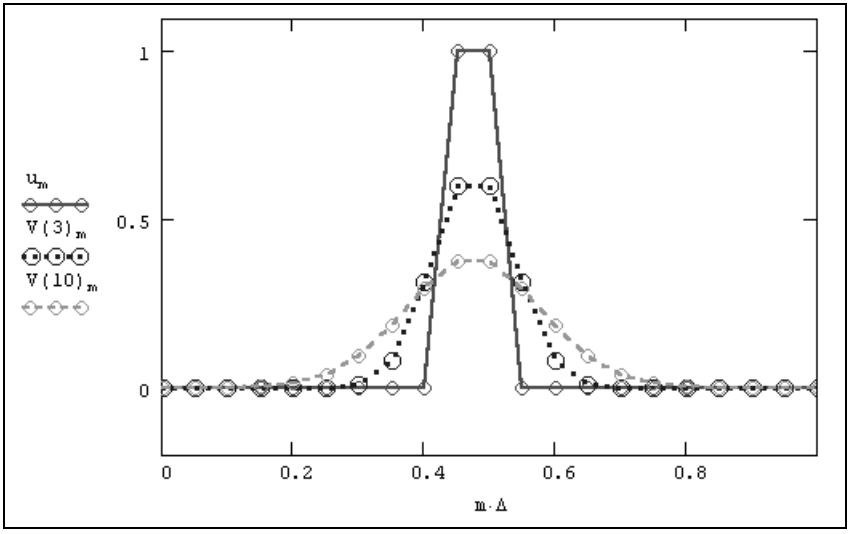



**Рис. 11.7.** Решение линейного уравнения теплопроводности (продолжение листинга 11.1)

## Нелинейное уравнение

Намного более интересные решения можно получить для нелинейного уравнения теплопроводности, например, с нелинейным источником тепла $\phi(u)=10^3\cdot(u-u^3)$. Заметим, что в листинге 11.1 мы предусмотрительно определили коэффициент диффузии и источник тепла в виде пользовательских функций, зависящих от аргумента `u`, т. е. от температуры. Если бы мы собирались моделировать явную зависимость их от координат, то следовало бы ввести в пользовательскую функцию в качестве аргумента переменную `x`, как это сделано для источника тепла $\phi$. Поэтому нет ничего проще замены определения этих функций с констант `D(u)=1` и $\phi(x,u)=0$ на новые функции, которые станут описывать другие модели диффузии тепла. Начнем с того, что поменяем четвертую строку листинга 11.1 на $\phi(x,u)=10^3\cdot(u-u^3)$, не изменяя пока постоянного значения коэффициента диффузии.

### *Примечание*

С физической точки зрения, зависимость коэффициента диффузии и функции источника тепла от температуры означает, что эти параметры будут меняться от точки к точке среды, определяясь локальными значениями текущей температуры в этих точках. Ввод ненулевого источника тепла означает, что среда получает определенное количество тепла, тем большее, чем больше локальная температура. Можно догадаться, что введение такой зависимости может моделировать, в частности, горение среды.

Если осуществить расчеты с упомянутым источником (имеющим кубическую нелинейность), то получится очень интересное решение уравнения теплопроводности, имеющее профиль тепловых фронтов. С течением времени граница раздела высокой и низкой температуры распространяется в обе стороны от зоны первичного нагрева, оставаясь весьма четко выделенной (рис. 11.8).

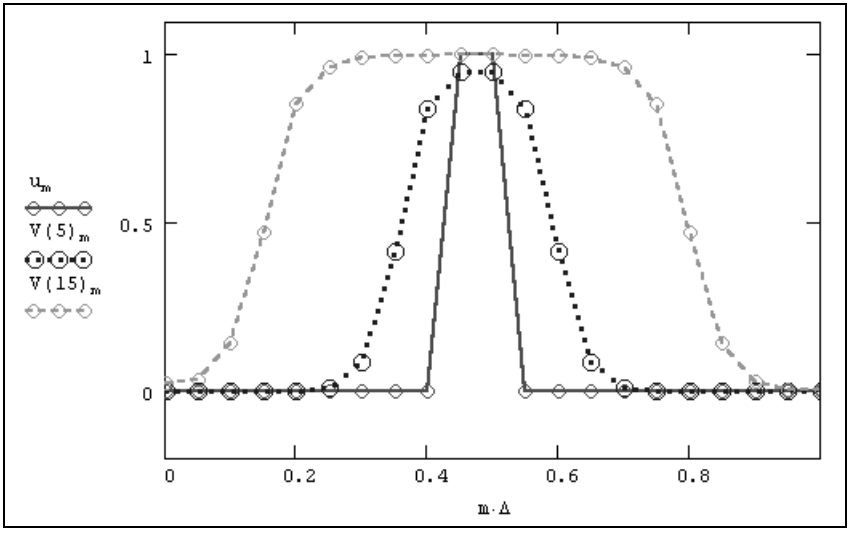



**Рис. 11.8.** Решение уравнения теплопроводности с нелинейным источником (тепловой фронт)

Еще более неожиданные решения возможны при нелинейности также и коэффициента диффузии. Например, если взять квадратичный коэффициент диффузии $D(x,u)=u^2$ (что с учетом его умножения на неизвестные функции создаст кубическую нелинейность уравнения), а также $\phi(x,u)=10^3\cdot u^{3.5}$, то вы сможете наблюдать совсем иной режим горения среды. В отличие от рассмотренного эффекта распространения тепловых фронтов, горение оказывается локализованным в области первичного нагрева среды, причем температура в центре нагрева со временем возрастает до бесконечной величины (рис. 11.9). Такое решение описывает так называемый режим горения "с обострением".

Читателю предлагается поэкспериментировать с этим и другими нелинейными вариантами уравнения теплопроводности. Существенно, что такие интересные результаты удается получить лишь численно, а в Mathcad только с применением элементов программирования.

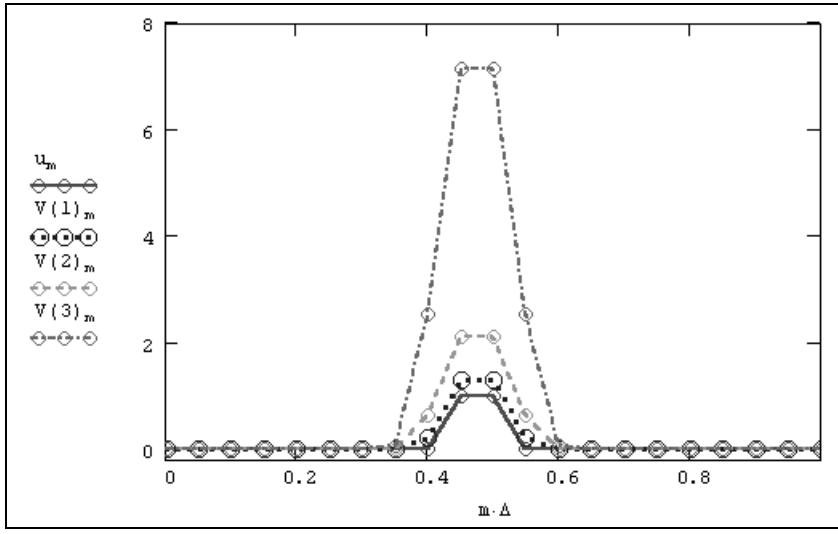



**Рис. 11.9.** Решение уравнения теплопроводности с нелинейным источником и коэффициентом диффузии (режим локализации горения)

## Устойчивость

Как мы убедились, явная разностная схема Эйлера дает вполне разумные результаты и вполне может использоваться для практического моделирования задач, связанных с решением уравнений в частных производных. Однако теперь пришло время сказать об очень важной характеристике разностных схем, которая называется их *устойчивостью*. Не вдаваясь в детали, заметим, что производить расчеты можно только при помощи устойчивых разностных схем, а чтобы пояснить это понятие, обратимся вновь к листингу 11.1, реализующему явную схему для линейного уравнения диффузии.

Слегка изменим соотношение шагов по времени и пространственной координате, произведя расчеты сначала с $\tau$=0.0005 (эти результаты показаны на рис. 11.7 выше), а также с $\tau$=0.0010 и $\tau$=0.0015. Результат применения разностной схемы Эйлера для $\tau$=0.0010 примерно тот же, что и для меньшего значения $\tau$, приведенного на рис. 11.7. А вот следующее (казалось бы, незначительное) увеличение шага по времени приводит к катастрофе (рис. 11.10).

Вместо ожидаемого решения получаются совершенно неожиданные профили температуры, которые быстро осциллируют вдоль пространственной координаты, причем амплитуда и число пиков этих осцилляций быстро увеличи-

ваются от шага к шагу. Совершенно ясно, что полученное решение не имеет ничего общего с физикой моделируемого явления, а является следствием внутренних свойств самой разностной схемы, которые до этого были для нас скрыты.

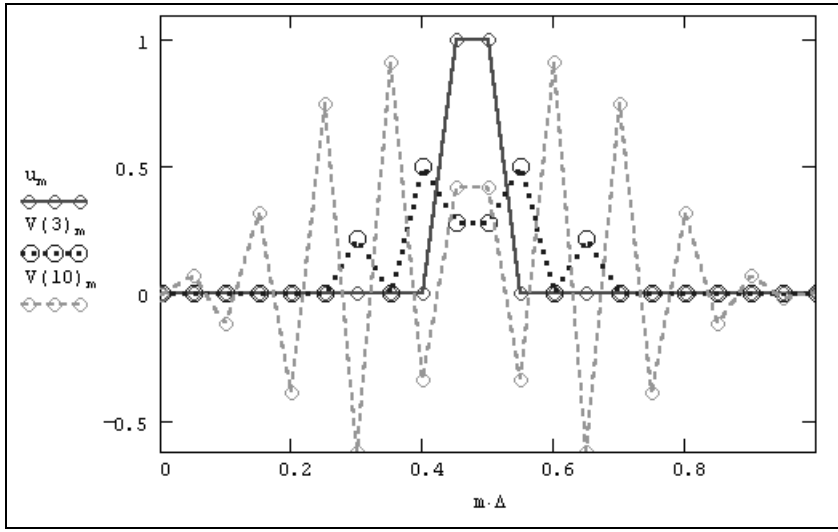



**Рис. 11.10.** Численное решение уравнения теплопроводности при помощи явной схемы Эйлера (см. листинг 11.1 ниже с временным шагом $\tau$=0.0015)

Характерная "разболтка" решения как раз и является проявлением неустойчивости явной схемы Эйлера для выбранного соотношения шагов по времени и пространству. В теории численных методов показывается, что явная схема Эйлера для уравнения теплопроводности устойчива при значениях коэффициента Куранта, меньших 1, и неустойчива в противоположном случае. Иными словами, существует ограничение для выбора соотношения шагов, заключающееся в том, что для расчета на более частых пространственных сетках необходимо использовать также и малые шаги по времени.

### *Примечание*

Как несложно убедиться, для $\tau$=0.0005 коэффициент Куранта C=0.4, для $\tau$=0.0010 он все еще меньше единицы: C=0.8, а для $\tau$=0.0015 решение уже больше единицы: C=1.2, в связи с чем схема становится неустойчивой (см. рис. 11.10).

## 11.2.2. Неявная схема Эйлера

В отличие от явной схемы Эйлера, неявная является безусловно-устойчивой (т. е. не выдающей "разболтки" ни при каких значениях коэффициента Куранта). Однако ценой устойчивости является необходимость решения на каждом шаге по времени системы алгебраических уравнений.

### Построение неявной разностной схемы

Чтобы построить неявную разностную схему для уравнения диффузии, используем шаблон, изображенный на рис. 11.11, т. е. для дискретизации пространственной производной будем брать значения сеточной функции с верхнего (неизвестного) слоя по времени. Таким образом, разностное уравнение для `(i,k)`-го узла будет отличаться от уравнения для явной схемы (11.8) только индексами по временной координате в правой части:

$$\frac{u_{i,k+1} - u_{i,k}}{\Delta_t} = D\frac{u_{i-1,k+1} - 2u_{i,k+1} + u_{i+1,k+1}}{\Delta_x^2} + \phi_{i,k}. \qquad (11.10)$$

Если привести подобные слагаемые, то получится система уравнений, связывающая для каждого `i`-го узла три неизвестных значения сеточной функции (в самом этом узле и в соседних с ним слева и справа узлах). Множители при неизвестных значениях сеточной функции в узах шаблона показаны на рис. 11.11 в виде подписей, подобно тому, как это было сделано для явной схемы (см. рис. 11.6).

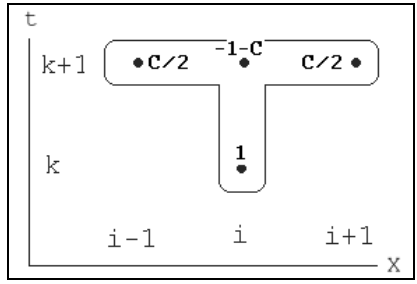



**Рис. 11.11.** Шаблон неявной схемы для уравнения теплопроводности

Очень важно, что если само уравнение теплопроводности линейно, то `c` в левой части разностного уравнения является константой, а $\phi$ в его правой части

может зависеть только от первой степени u. Поэтому система уравнений (11.10) для всех пространственных узлов i=1, ..., M-1 является линейной системой, что существенно упрощает ее решение (поскольку известно, что для линейных систем с ненулевым определителем решение существует и является единственным). Напомним, что для получения замкнутой системы линейных уравнений необходимо дополнить данный набор разностных уравнений граничными условиями, т. е. известными значениями сеточной функции для i=0 и i=M.

***Примечание***

Если рассматривать нелинейный случай, то на каждом шаге по времени пришлось бы решать систему нелинейных уравнений, число решений которых могло бы быть большим, и среди них требовалось бы отыскать нужное, а не паразитное решение.

Для реализации неявной схемы, таким образом, можно использовать комбинацию средств программирования Mathcad и встроенной функции решения системы линейных уравнений lsolve. Один из возможных способов решения предложен в листинге 11.2. Бóльшая часть этого листинга является вводом параметров задачи (шагов, начальных и граничных условий), и только в последней его строке определяется функция пользователя, вычисляющая сеточную функцию на каждом временном слое (при помощи встроенной функции решения системы линейных уравнений lsolve). В нескольких предыдущих строках листинга (после расчета коэффициента Куранта Cou) формируется матрица системы уравнений A, которая записывается в подходящем для Mathcad виде, как это сделано в листинге 11.2. Как несложно убедиться, столбец правых частей разностных уравнений выражается вычисленными значениями сеточной функции с предыдущего слоя.

Результаты расчетов по неявной схеме показаны на рис. 11.12 и, как видно, они дают примерно те же результаты, что и в случае применения явной схемы (см. рис. 11.7). Обратите внимание, что решение устойчиво при любых значениях коэффициента Куранта (в том числе и бóльших 1), поскольку, как следует из соответствующих положений теории численных методов, неявная схема является безусловно-устойчивой.

**Листинг 11.2. Неявная схема для линейного уравнения теплопроводности**

```
D := 1                                   ϕ(x, u) := 0

Border(t) := 0

Init(x) := Φ(x - 0.45) - Φ(x - 0.55)
```

$\tau := 0.005$

$M := 20$

$\Delta := \frac{1}{M}$ $\Delta = 0.05$

$m := 0 .. M$

$u_m := Init(m \cdot \Delta)$

$Cou := \frac{2 \cdot D \cdot \tau}{\Delta^2}$ $Cou = 4$

$m := 1 .. M - 1$

$A_{m,m} := -Cou - 1$

$A_{m,m-1} := \frac{Cou}{2}$ $A_{m,m+1} := \frac{Cou}{2}$

$A_{0,0} := 1$ $A_{M,M} := 1$

$$V(n) := \begin{cases} u & \text{if } n = 0 \\ \text{lsolve}(A, -V(n-1)) & \text{otherwise} \end{cases}$$

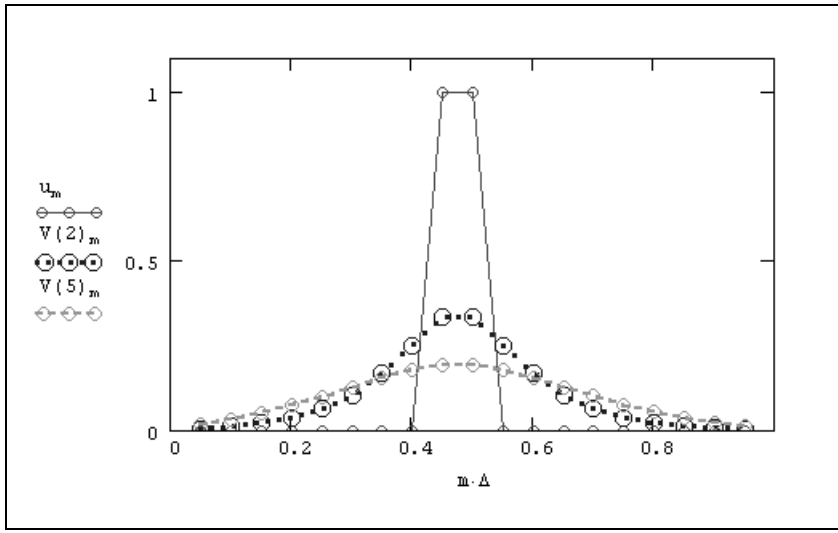



**Рис. 11.12.** Решение линейного уравнения теплопроводности при помощи неявной схемы на первом слое по времени (листинг 11.2)

## Алгоритм прогонки

Приведем в данном разделе описание чрезвычайно популярного алгоритма реализации неявных разностных схем, который называется методом *прогонки*. Этот алгоритм имел историческое значение для становления технологий расчетов уравнений в частных производных, и мы просто не можем не упомянуть о нем в этой книге.

> ***Примечание***
>
> Сразу оговоримся, что его применение для решения уравнений в частных производных в среде Mathcad может быть оправдано, только если вы работаете с очень частыми сетками, которые приводят к системам разностных уравнений большой размерности и, соответственно, очень большому времени вычислений.

Основным вычислительным ядром программы, реализующей на Mathcad неявную разностную схему, было решение (на каждом временном слое) системы линейных алгебраических уравнений, задаваемых матрицей `A`. Заметим, что эта матрица, как говорят, имеет *диагональное преобладание*, а точнее, является *трехдиагональной* (рис. 11.13). Все ее элементы, кроме элементов на главной диагонали и двух соседних диагоналях, равны нулю. С точки зрения оптимизации быстродействия алгоритма применение встроенной функции `lsolve` является весьма расточительным, поскольку основной объем арифметических операций, выполняемых компьютером (а он составляет, как нетрудно убедиться, величину порядка $M^2$), сводится к непроизводительному перемножению нулей.

A =

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0.5 | -2 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0.5 | -2 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0.5 | -2 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0.5 | -2 | 0.5 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0.5 | -2 | 0.5 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0.5 | -2 | 0.5 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0.5 | -2 | 0.5 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.5 | -2 | 0.5 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.5 | -2 | 0.5 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**Рис. 11.13.** Матрица системы линейных разностных уравнений для неявной схемы (листинг 11.2 для `M=10`)

Для отыскания решения линейных систем алгебраических уравнений имеется чрезвычайно эффективный алгоритм, называемый прогонкой, который позволяет уменьшить число арифметических операций на целый порядок, т. е. до значения порядка M. Это означает, что при использовании пространственных сеток с 1000 узлами выигрыш во времени вычислений составит величину порядка $10^3$! Реализация данного алгоритма приведена в листинге 11.3, который является продолжением листинга 11.2, используя определенные в нем коэффициенты матрицы A, а также начальное условие.

Программа листинга 11.3 осуществляет пересчет одного шага по времени, т. е. заменяет содержимое столбца u с предыдущего временного слоя вычисленными значениями неизвестной функции со следующего слоя. Первые пять строк листинга 11.3 представляют так называемый *обратный* ход прогонки, а последние две строки — ее *прямой* ход. Заинтересовавшемуся читателю предлагается самому оформить представленный алгоритм прогонки в виде программы решения разностных уравнений для вычисления произвольного временного слоя по примеру листингов 11.1 и 11.2. Заметим, что описание этого знаменитого алгоритма можно отыскать практически в любом современном учебнике по численным методам.

**Листинг 11.3. Алгоритм прогонки (продолжение листинга 11.2)**

$\alpha_{M-1} := 0$ $\beta_{M-1} := 0$

$m := M - 1 .. 1$

$$\gamma_m := -\frac{1}{A_{m,m} + A_{m,m+1} \cdot \alpha_m}$$

$\alpha_{m-1} := \gamma_m \cdot A_{m,m-1}$

$\beta_{m-1} := \gamma_m \cdot (A_{m,m+1} \cdot \beta_m - b_m)$

$m := 0 .. M - 1$

$u_{m+1} := \alpha_m \cdot u_m + \beta_m$

## 11.2.3. О возможности решения многомерных уравнений

Все, что было сказано до сих пор, касалось исключительно способов решения одномерных (в смысле пространственных координат) уравнений. И алгоритмы разностных схем, и встроенные функции, включая появившиеся в 11-й

версии (*см. следующий разд.*), относились к уравнениям, зависящим от одной пространственной координаты.

Можно ли при помощи Mathcad решать двумерные или трехмерные (пространственные) уравнения? С точки зрения программирования пользователем численных алгоритмов типа метода сеток принципиальных ограничений нет. Разумеется, если сначала аккуратно выписать разностную схему соответствующего многомерного дифференциального уравнения, то вполне возможно запрограммировать ее при помощи описанных нами средств. Самым главным противодействием будет существенное увеличение времени расчетов. Простая оценка необходимого количества операций показывает, что ввод зависимости уравнения от второй пространственной координаты многократно увеличивает число разностных уравнений, которые должны решаться при реализации каждого шага по времени. Например, если используется пространственная сетка из 100 узлов по каждой координате, то вместо $10^2$ разностных уравнений на каждом шаге придется решать уже $10^4$ уравнений, т. е. объем вычислений сразу же возрастает в 100 раз. Вообще говоря, пакет Mathcad не является экономичной средой вычислений, и бороться с их сильно возрастающим объемом следует пользователю еще на этапе разработки алгоритма. Хорошим примером такой борьбы может служить применение специфических алгоритмов, типа метода прогонки (*см. разд. "Алгоритм прогонки" этой главы*).

Приведем некоторые дополнительные замечания, связанные с возможностью осуществить редукцию громоздких (в смысле организации вычислений) двумерных задач к более простым. Рассмотрим ради примера двумерное уравнение теплопроводности (11.1) без источника с нулевыми граничными условиями и некоторым начальным двумерным распределением температуры по расчетной поверхности:

$$\frac{\partial u(x, y, t)}{\partial t} = D\left(\frac{\partial^2 u(x, y, t)}{\partial x^2} + \frac{\partial^2 u(x, y, t)}{\partial y^2}\right).$$

Произведем дискретизацию данного уравнения по временной координате, заменяя первую производную ее разностным аналогом и несколько перегруппировывая слагаемые и множители:

$$\frac{\partial^2 u_{i+1}}{\partial x^2} + \frac{\partial^2 u_{i+1}}{\partial y^2} = \frac{u_{i+1} - u_i}{D\tau}.$$

Как вы видите, мы используем неявную разностную схему, заранее заботясь о том, чтобы разностная задача была более устойчивой. Здесь $u_i(x,y)$ — известная с предыдущего шага по времени функция двух пространственных

переменных, а $u_{i+1}(x,y)$ — функция, подлежащая определению при реализации каждого шага по времени.

Посмотрим на полученную задачу с несколько другой стороны — а именно как на дифференциальное уравнение относительно неизвестной функции двух переменных $u_{i+1}(x,y)$. Подчеркнем, что такое уравнение получается для каждого шага по времени, т. е. для реализации всей разностной схемы требуется решить большое число таких уравнений.

С предложенной точки зрения, на каждом временном шаге необходимо решить некоторое двумерное эллиптическое линейное уравнение, причем его граничные условия определяются граничными условиями исходной задачи. Это уравнение очень похоже на уравнение Пуассона с той лишь разницей, что в его правую часть, описывающую источник, входит неизвестная функция (к счастью, линейно). Таким образом, зависимость от найденного на предыдущем шаге по времени решения определяется зависимостью от него источника, т. е. правой части.

Суммируя сказанное, можно констатировать, что если вы имеете запрограммированный алгоритм решения выписанного эллиптического уравнения, чуть более сложного, чем уравнение Пуассона, то его с легкостью можно использовать в качестве подпрограммы реализации разностной схемы двумерного уравнения теплопроводности. Забегая вперед, приходится отметить, что, к сожалению, встроенные функции Mathcad для решения уравнения Пуассона в данном случае не годятся в качестве такой подпрограммы, поскольку предполагают независимость источника от самой неизвестной функции, и могут справляться лишь с правой частью, зависящей только от пространственных координат (*см. разд. 11.3.2*).

Не будем далее останавливаться на способах решения многомерных уравнений, ограничившись этими замечаниями относительно путей оптимизации алгоритмов их решения.

## 11.3. Встроенные функции для решения уравнений в частных производных

Как видно из предыдущего раздела, с уравнениями в частных производных вполне можно справиться и не прибегая к специфическим средствам Mathcad. Между тем в версиях Mathcad 11 (и выше) имеется несколько встроенных функций, при помощи которых можно автоматизировать процесс решения дифференциальных уравнений в частных производных. Рассмотрим

в данном разделе основные аспекты их применения, отмечая не только инструкции по их применению, описанные разработчиками Mathcad, но также и некоторые "скрытые" возможности этих функций.

## 11.3.1. Параболические и гиперболические уравнения

Разработчики впервые применили дополнительные встроенные функции для решения параболических и гиперболических уравнений в частных производных в версии Mathcad 11, отлично осознавая значимость этих задач для современного исследователя и инженера. Предусмотрены два варианта решения: при помощи вычислительного блока `Given/pdesolve`, а также при помощи встроенной функции `numol`. Первый путь проще в применении и нагляднее, зато второй позволяет автоматизировать процесс решения уравнений в частных производных, например, если нужно включить его в качестве составного шага в более сложную Mathcad-программу.

### Вычислительный блок *Given / pdesolve*

Встроенная функция `pdesolve` применяется в рамках вычислительного блока, начинающегося ключевым словом `Given`, и пригодна для решения различных гиперболических и параболических уравнений. Она предназначена для решения одномерного уравнения (или системы уравнений) в частных производных (того, которое определит пользователь в рамках вычислительного блока `Given`), зависящего от времени `t` и пространственной координаты `x`, имеет целый набор различных аргументов и работает следующим образом:

- `pdesolve(u,x,xrange,t,trange,[xpts],[tpts])` — возвращает скалярную (для единственного исходного уравнения) или векторную (для системы уравнений) функцию двух аргументов `(x,t)`, являющуюся решением дифференциального уравнения (или системы уравнений) в частных производных. Результирующая функция получается интерполяцией сеточной функции, вычисляемой согласно разностной схеме:
  - `u` — явно заданный вектор имен функций (без указания имен аргументов), подлежащих вычислению. Эти функции, а также граничные условия (в форме Дирихле или Неймана) должны быть определены пользователем перед применением функции `pdesolve` в вычислительном блоке после ключевого слова `Given`. Если решается не система уравнений в частных производных, а единственное уравнение, то, соответственно, вектор `u` должен содержать только одно имя функции и вырождается в скаляр;

- x — пространственная координата (имя аргумента неизвестной функции);
- xrange — пространственный интервал, т. е. вектор значений аргумента x для граничных условий. Этот вектор должен состоять из двух действительных чисел (представляющих левую и правую границу расчетного интервала);
- t — время (имя аргумента неизвестной функции);
- trange — расчетная временная область: вектор значений аргумента t, который должен состоять из двух действительных чисел (представляющих левую и правую границу расчетного интервала по времени);
- xpts — количество пространственных точек дискретизации (может не указываться явно, в таком случае будет подобрано программой автоматически);
- tpts — количество временных слоев, т. е. интервалов дискретизации по времени (также может не указываться пользователем явно).

В качестве примера использования функции pdesolve (листинг 11.4) решим то же самое одномерное уравнение теплопроводности (11.5) с граничными и начальными условиями (11.6) и (11.7).

**Листинг 11.4. Решение одномерного уравнения теплопроводности**

$$D := 0.1$$

$$L := 1 \qquad T := 10$$

Given

$$u_t(x, t) = D \cdot u_{xx}(x, t)$$

$$u(x, 0) = \Phi(x - 0.45) - \Phi(x - 0.55)$$

$$u(0, t) = 0 \qquad u(L, t) = 0$$

$$u := \text{Pdesolve}\left[u, x, \begin{pmatrix} 0 \\ L \end{pmatrix}, t, \begin{pmatrix} 0 \\ T \end{pmatrix}, 100, 10\right]$$

Для корректного использования функции pdesolve предварительно, после ключевого слова Given, следует записать само уравнение и граничные условия при помощи логических операторов (для их ввода в Mathcad существует специальная панель). Обратите внимание, что уравнение должно содержать имя неизвестной функции u(x,t) вместе с именами аргументов (а не так, как она записывается в пределах встроенной функции pdesolve). Для идентификации частных производных в пределах вычислительного блока следует ис-

пользовать нижние индексы, например, $u_{xx}(t)$, для обозначения второй производной функции u по пространственной координате x.

Как видно из рис. 11.14, на котором изображены результаты расчетов по листингу 11.4, встроенная функция с успехом справляется с уравнением диффузии, отыскивая уже хорошо знакомое нам решение. Заметим, что использование встроенной функции pdesolve связано с довольно громоздкими вычислениями, которые могут отнимать существенное время.

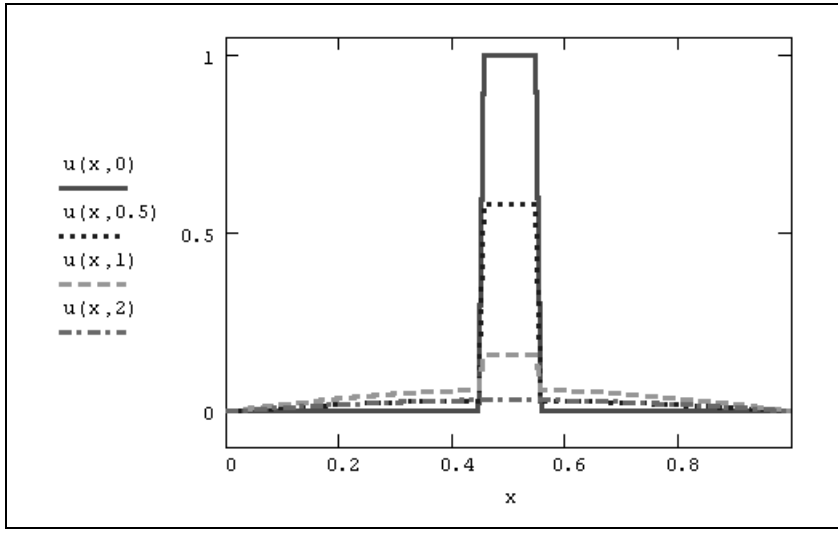



**Рис. 11.14.** Решение уравнения диффузии тепла при помощи встроенной функции pdesolve (листинг 11.4)

### *Примечание*

Как вы можете заметить, выбирать величину шага по пространственной и временной переменным может как сам алгоритм, так и пользователь (неявным образом, через число узлов сетки). Читателю предлагается повторить вычисления листинга 11.4 для различных комбинаций параметров (главным образом, числа узлов сетки), чтобы проверить, в каких случаях алгоритм встроенной функции справляется с задачей, выдавая верное решение, а в каких дает сбой.

## Пример: волновое уравнение

Приведем еще один пример применения функции pdesolve для решения уравнений в частных производных. Рассмотрим одномерное волновое урав-

нение, которое описывает, например, свободные колебания струны музыкального инструмента:

$$\frac{\partial^2 u(x,t)}{\partial t^2} = c^2 \frac{\partial^2 u(x,t)}{\partial x^2}. \qquad (11.11)$$

Здесь неизвестная функция `u(x,t)` описывает динамику смещения профиля струны относительно невозмущенного (прямолинейного) положения, а параметр `c` характеризует материал, из которого изготовлена струна.

Как вы видите, уравнение (11.11) содержит производные второго порядка, как по пространственной координате, так и по времени. Для того чтобы можно было использовать встроенную функцию `pdesolve`, необходимо переписать волновое уравнение в виде системы двух уравнений в частных производных, введя вторую неизвестную функцию $v=u_t$. Программа для решения волнового уравнения приведена в листинге 11.5, а результат — на рис. 11.15.

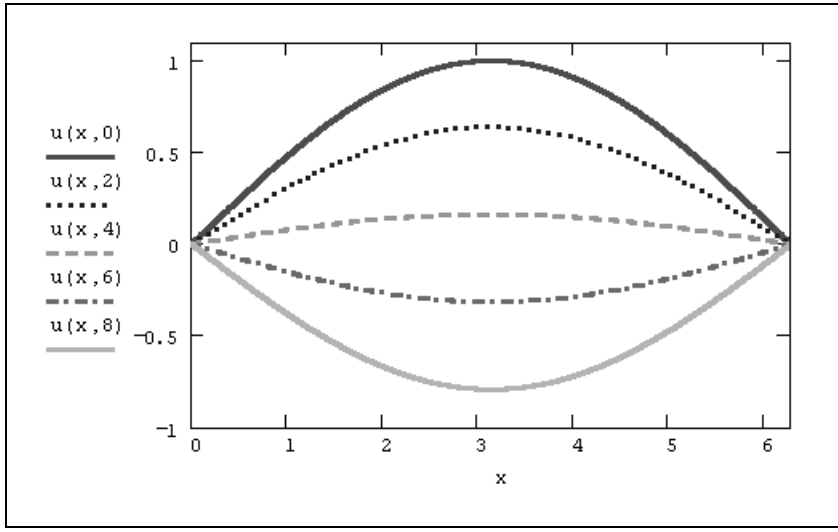



**Рис. 11.15.** Решение волнового уравнения (продолжение листинга 11.5)

**Листинг 11.5. Решение волнового уравнения**

```
L := 2·π                                T := 1

Given
```

$$v_t(x,t) = c^2 \cdot u_{xx}(x,t)$$

$$u_t(x,t) = v(x,t)$$

$$u(x,0) = \sin\left(\frac{\pi \cdot x}{L}\right) \qquad v(x,0) = 0$$

$$u(0,t) = 0 \qquad u(L,t) = 0$$

$$\begin{pmatrix} u \\ v \end{pmatrix} := \text{Pdesolve}\left[\begin{pmatrix} u \\ v \end{pmatrix}, x, \begin{pmatrix} 0 \\ L \end{pmatrix}, t, \begin{pmatrix} 0 \\ T \end{pmatrix}\right]$$

## Встроенная функция *numol*

Альтернативный вариант решения дифференциальных уравнений в частных производных заключается в применении еще одной встроенной функции `numol`, которая реализует тот же самый алгоритм сеток, позволяя вручную задать большинство его параметров:

- `numol(xrange,xpts,trange,tpts,Npde,Nae,rhs,init,bc)` — возвращает матрицу решения дифференциального уравнения в частных производных, представляющую искомую сеточную функцию в каждой точке по пространственной (по строкам) и временной координате (по столбцам). Если решается не одно уравнение, а система уравнений, то результатом является составная матрица, образованная путем слияния (слева-направо) матриц со значениями каждой искомой сеточной функции:
  - `Npde` — общее количество дифференциальных уравнений в частных производных в системе;
  - `Nae` — общее количество дополнительных алгебраических уравнений, которые также могут входить в систему;
  - `rhs` — векторная функция, определяющая систему дифференциальных и алгебраических уравнений (формат этого и двух следующих матричных параметров объяснен в листинге 11.9);
  - `init` — векторная функция, определяющая начальные условия для каждой неизвестной функции;
  - `bc` — функциональная матрица граничных условий;
  - `xrange` — пространственный интервал, т. е. вектор значений аргумента `x` для граничных условий. Этот вектор должен состоять из двух действительных чисел (представляющих левую и правую границу расчетного интервала);

- `xpts` — количество пространственных точек дискретизации (может не указываться явно, в таком случае будет подобрано программой автоматически);
- `trange` — расчетная временная область: вектор значений аргумента `t`, который должен состоять из двух действительных чисел (представляющих левую и правую границу расчетного интервала по времени);
- `tpts` — количество временных слоев, т. е. интервалов дискретизации по времени (также может не указываться пользователем явно).

Пример решения волнового уравнения при помощи функции `numol` приведен в листинге 11.6, особое внимание в котором мы призываем уделить формату представления векторов `rhs`, `init` и `bc`, а также принципу извлечения отдельных сеточных решений из матрицы-результата. График решения, показанный на рис. 11.16, полезно сравнить с результатом применения вычислительного блока из предыдущего раздела (см. листинг 11.5 и рис. 11.15).

**Листинг 11.6. Решение волнового уравнения при помощи функции `numol`**

```
c := 1
L := 2·π                    Nx := 20
T := 10                     Nt := 15
num_pde := 2                num_pae := 0
rhs(x, t, u, u_x, u_xx) := ( u_1 )
                           ( c^2·u_xx_0 )
init(x) := ( sin(π·x / L) )
           (      0      )
bc_func(t) := ( init(0)_0   init(L)_0   "D" )
              (   "NA"        "NA"      "D" )
sol := numol[ (0), Nx, (0), Nt, num_pde, num_pae, rhs, init, bc_func ]
       (L)     (T)
rows(sol) = 20              Nx = 20
cols(sol) = 30              Nt·2 = 30
SOL := submatrix(sol, 0, Nt − 1, 0, Nx − 1)
```

Как вы видите, функция `numol` имеет еще большее число аргументов, нежели `pdesolve`, и позволяет автоматизировать применение метода сеток. Однако пользоваться ею намного сложнее, чем вычислительным блоком, поскольку и уравнения, и начальные и граничные условия должны быть записаны в специальном формате. Применение функции `numol` оправданно, когда необходимо включить решение уравнений в частных производных в более сложные вычисления в качестве подпрограммы, организовать серию расчетов с меняющимся параметром, подготовить анимацию графиков решения и т. п.

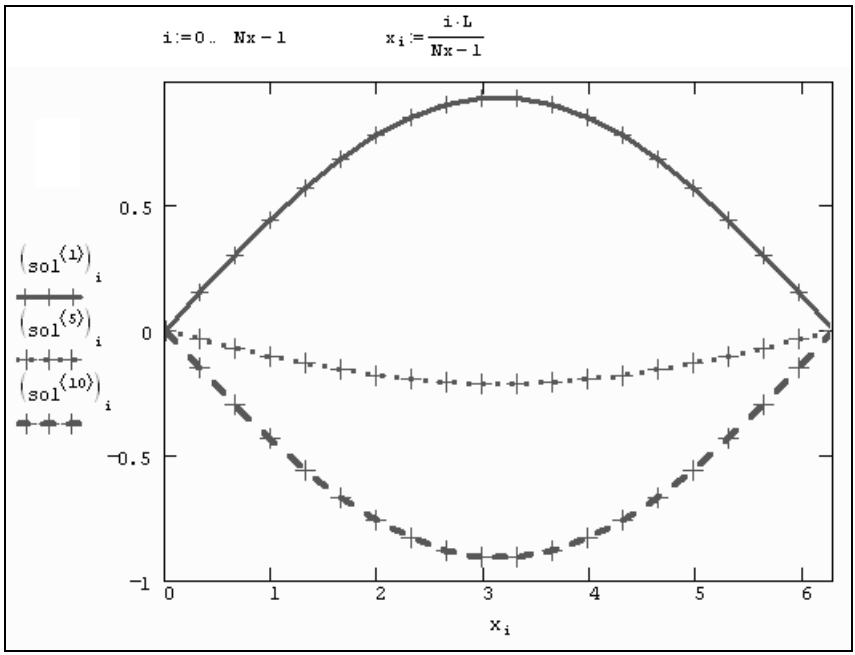



**Рис. 11.16.** Решение волнового уравнения (продолжение листинга 11.6)

Именно в целях визуализации решения параболических и гиперболических уравнений в частных производных использование функции `numol` наиболее полезно. График решения динамических уравнений (зависящих от времени `t`) выглядит намного эффектнее и воспринимается несравненно лучше, если он оформлен в виде анимации. Для создания анимационных роликов расчетное время следует выразить через константу `FRAME` и затем применить команду **View / Animate** (Вид / Анимация) (*см. разд. 13.3.2*).

## 11.3.2. Эллиптические уравнения

Решение эллиптических уравнений в частных производных реализовано только для единственного типа задач — двумерного уравнения Пуассона. Это уравнение содержит вторые производные функции `u(x,y)` по двум пространственным переменным:

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = -f(x,y). \qquad (11.12)$$

Уравнение Пуассона описывает, например, распределение электростатического поля `u(x,y)` в двумерной области с плотностью заряда `f(x,y)`, или (*см. разд. 11.1.2*) стационарное распределение температуры `u(x,y)` на плоскости, в которой имеются источники (или поглотители) тепла с интенсивностью `f(x,y)`.

> ***Примечание***
>
> Несмотря на то, что применение встроенных функций, описанных в данном разделе, анонсировано разработчиками Mathcad только для уравнения Пуассона, их можно применять и для решения других уравнений, даже необязательно эллиптического типа. О том, как осуществить такие расчеты, написано в конце данного раздела.

### Уравнение Пуассона с нулевыми граничными условиями

Корректная постановка краевой задачи для уравнения Пуассона требует задания граничных условий. В Mathcad решение ищется на плоской квадратной области, состоящей из `(M+1)×(M+1)` точек. Поэтому граничные условия должны быть определены пользователем для всех четырех сторон упомянутого квадрата. Самый простой вариант — это нулевые граничные условия, т. е. постоянная температура по всему периметру расчетной области. В таком случае можно использовать встроенную функцию `multigrid`:

- `multigrid(F,ncycle)` — матрица решения уравнения Пуассона размера `(M+1)×(M+1)` на квадратной области с нулевыми граничными условиями:
  - `F` — матрица размера `(M+1)×(M+1)`, задающая правую часть уравнения Пуассона;
  - `ncycle` — параметр численного алгоритма (количество циклов в пределах каждой итерации).

***Внимание!***

Сторона квадрата расчетной области должна включать точно $M=2^n$ шагов, т. е. $2^n+1$ узлов, где $n$ — целое число.

Параметр численного метода `ncycle` в большинстве случаев достаточно взять равным `2`. Листинг 11.7 содержит пример использования функции `multigrid` для расчета краевой задачи на области `33×33` точки и точечным источником тепла в месте, задаваемом координатами `(15,20)` внутри этой области.

**Листинг 11.7. Решение уравнения Пуассона с нулевыми граничными условиями**

```
M := 32
F_{M,M} := 0
F_{15,20} := 10^4
G := multigrid (-F, 2)
```

В первой строке листинга задается значение `M=32`, в двух следующих строках создается матрица правой части уравнения Пуассона, состоящая из всех нулевых элементов, за исключением одного, задающего расположение источника.

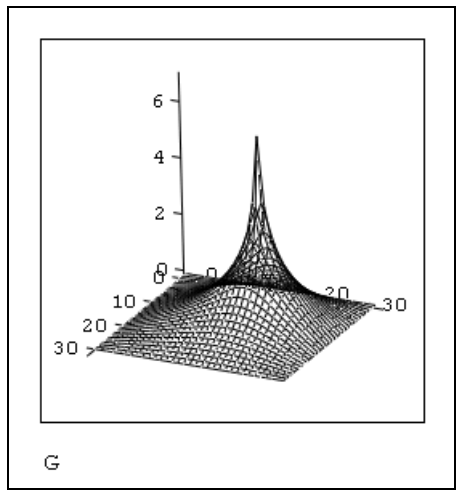



**Рис. 11.17.** График поверхности решения уравнения Пуассона (продолжение листинга 11.7)

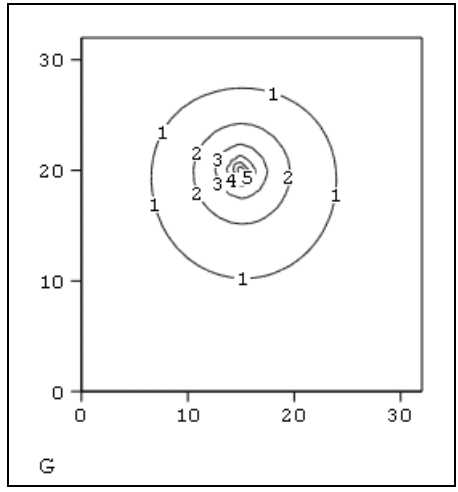



**Рис. 11.18.** График линий уровня решения уравнения Пуассона (продолжение листинга 11.7)

В последней строке матрице `G` присваивается результат действия функции `multigrid`. Обратите внимание, первый ее аргумент сопровождается знаком "минус", что соответствует записи правой части уравнения Пуассона (11.11). Графики решения показаны на рис. 11.17 и 11.18 в виде трехмерной поверхности и линий уровня соответственно.

## Уравнение Пуассона с произвольными граничными условиями

В более сложных случаях, например, для решения краевой задачи с ненулевыми условиями на границах, следует использовать другую встроенную функцию `relax`, имеющуюся в Mathcad:

- `relax(a,b,c,d,e,F,v,rjac)` — матрица решения дифференциального уравнения в частных производных на квадратной области, полученного с помощью алгоритма релаксации для метода сеток:
  - `a,b,c,d,e` — квадратные матрицы коэффициентов разностной схемы, аппроксимирующей дифференциальное уравнение;
  - `F` — квадратная матрица, задающая правую часть дифференциального уравнения;
  - `v` — квадратная матрица граничных условий и начального приближения к решению;

- `rjac` — параметр численного алгоритма (спектральный радиус итераций Якоби).

Параметр численного алгоритма характеризует скорость сходимости итераций. Он должен быть числом от `0` до `1`. В матрице граничных условий `v` необходимо задать только граничные элементы, исходя из значения краевых условий по периметру расчетной области. Прочие (внутренние) элементы этой матрицы служат для задания начального приближения к решению. Суть алгоритма релаксации сводится к тому, что в ходе итераций происходит проверка уравнений и соответствующая коррекция значений искомой функции в каждой точке. Если начальное приближение выбрано удачно, то можно надеяться, что алгоритм сойдется ("релаксирует") к правильному решению.

> ***Внимание!***
>
> Все матрицы, задающие как коэффициенты разностной схемы `a`, `b`, `c`, `d`, `e`, граничные условия `v`, так и само решение `F`, должны иметь одинаковый размер `(M+1)×(M+1)`, соответствующий размеру расчетной области. При этом целое число `M` обязательно должно быть степенью двойки: $M=2^n$.

Решение уравнения Пуассона с тремя источниками разной интенсивности при помощи функции `relax` приведено в листинге 11.8.

**Листинг 11.8. Решение уравнения Пуассона с помощью функции `relax`**

```
M := 32

F[M,M] := 0

F[15,20] := 10    F[25,10] := 5    F[10,10] := -5

i := 0 .. M       k := 0 .. M

a[i,k] := 1

b := a

c := a

d := a

e := -4·a

v[i,k] := 0

G := relax(a,b,c,d,e,-F,v,.95)
```

Первые три строки имеют тот же смысл, что и в предыдущем листинге. Только вместо одного источника тепла взято их другое распределение —

один сильный источник, один более слабый и один сток тепла. В следующих шести строках задаются коэффициенты разностной схемы. Отложим их обсуждение до последнего раздела этой главы, ограничившись утверждением, что для решения уравнения Пуассона коэффициенты должны быть взяты именно такими, как показано в листинге 11.8. В предпоследней строке задана матрица нулевых граничных условий и нулевых начальных приближений, а в последней матрице `G` присваивается результат действия функции `relax`. График полученного решения в виде линий уровня показан на рис. 11.19.

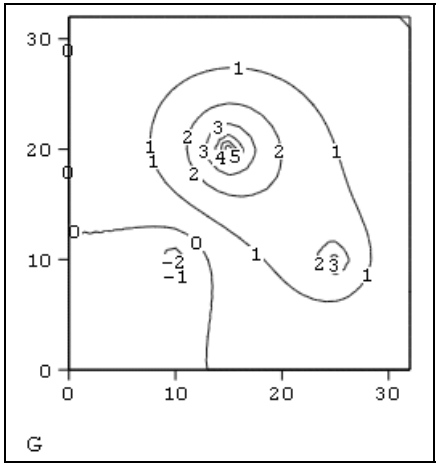



**Рис. 11.19.** Решение уравнения Пуассона с помощью функции `relax` (продолжение листинга 11.8)

## Разностная схема для решения уравнения Пуассона

Несмотря на отсутствие сведений в справочной системе Mathcad о решении других линейных дифференциальных уравнений в частных производных, кроме уравнения Пуассона, сделать это возможно с помощью той же функции `relax` (*см. предыдущий разд.*). Для этого нужно правильным образом задать коэффициенты разностной схемы.

Начнем с пояснения выбора этих коэффициентов (см. листинг 11.8) для уравнения Пуассона. Согласно основным идеям метода сеток (*см. разд. 11.2*), для дискретизации обеих пространственных производных в уравнении (11.12)

следует использовать по три соседних узла вдоль каждой из координат. Поэтому уравнение Пуассона (11.12) может быть записано в разностной форме при помощи шаблона типа "крест" (рис. 11.20). В этом случае, после приведения подобных слагаемых в разностных уравнениях коэффициенты разностной схемы будут такими, как показано возле узлов шаблона на этом рисунке (аналогичные коэффициенты для явной и неявных схем решения уравнения теплопроводности см. на рис. 11.6 и 11.11 соответственно).

Теперь, если вы сравните полученные числа с константами, которые присвоены элементам матриц-аргументов функции `relax` (см. листинг 11.8), то увидите, что они как раз и описывают вычисленные нами только что коэффициенты разностной схемы "крест". Таким образом, нетрудно сообразить, что с помощью встроенной функции `relax` можно решать и другие линейные дифференциальные уравнения в частных производных, которые можно аппроксимировать схемой типа "крест" или схемой, являющейся ее составной частью. Конечно, для того чтобы использовать эту встроенную функцию для другого уравнения, необходимо будет составить соответствующую разностную схему.

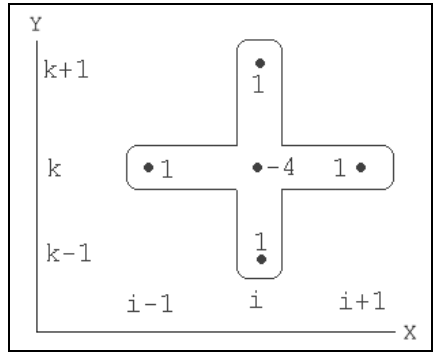



**Рис. 11.20.** Шаблон аппроксимации уравнения Пуассона "крест"

## Решение уравнения диффузии тепла при помощи функции *relax*

Приведем пример применения встроенной функции `relax` для решения другого уравнения в частных производных (т. е. не уравнения Пуассона, для которого она изначально предназначена). Вычислим при помощи этой функции

решение уже хорошо нам знакомого однородного линейного уравнения теплопроводности (*см. разд. 11.1.2*). Будем использовать явную разностную схему, шаблон которой изображен на рис. 11.6. Для того чтобы "приспособить" для явной схемы функцию `relax`, требуется только задать ее аргументы в соответствии с коэффициентами, показанными на шаблоне (см. рис. 11.6). Программа, реализующая таким способом явную схему, представлена на листинге 11.9. Число Куранта в этом листинге обозначено переменной `C`, как и положено явной разностной схеме, она выдает устойчивое решение только для `C<1`.

**Листинг 11.9. Решение уравнения теплопроводности при помощи функции `relax`**

```
M := 16

C := 0.9

i := 0 .. M                          k := 0 .. M

a[i,k] := C/2

b[i,k] := C/2

c[i,k] := 0

d[i,k] := 1 - C

e[i,k] := -1

F[i,k] := 0

v[i,k] := 0                                  v[0,5] := 1

G := relax(a, b, c, d, e, F, v, .5)
```

Результат действия программы листинга 11.9 показан на рис. 11.21 в виде трехмерной поверхности. Если сравнить рис. 11.21 с рис. 11.4, полученным при расчетах по запрограммированной разностной схеме, то в графиках рис. 11.4 нетрудно узнать сечения этой поверхности плоскостями `t=const`. Еще раз подчеркнем, что использовать встроенную функцию можно только для тех уравнений, которые допускают построение разностной схемы типа "крест" (см. рис. 11.17) или составного фрагмента этой схемы.

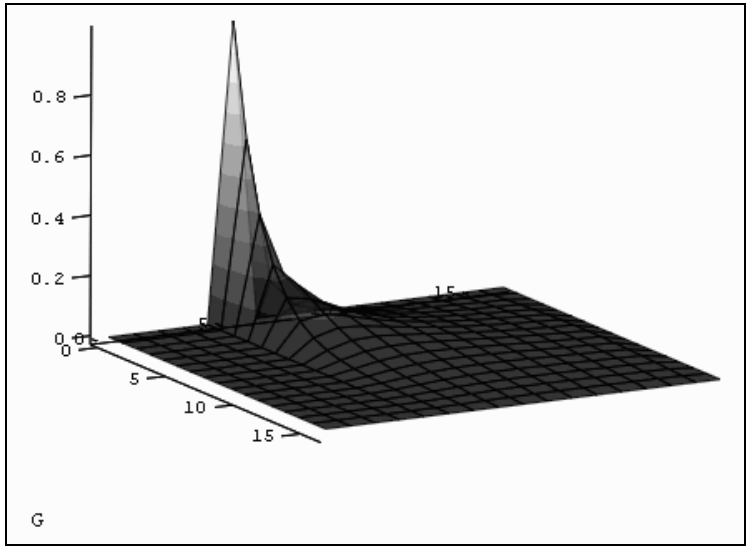



**Рис. 11.21.** Решение уравнения теплопроводности с помощью функции `relax` (продолжение листинга 11.9)

# ЧАСТЬ IV

## Обработка данных

# Глава 12

# Статистика

Mathcad имеет развитый аппарат работы с задачами математической статистики и обработки эксперимента.

Во-первых, имеется большое количество встроенных специальных функций, позволяющих рассчитывать плотности вероятности и другие основные характеристики основных законов распределения случайных величин (*см. разд. 12.1*). Наряду с этим, в Mathcad запрограммировано соответствующее количество генераторов псевдослучайных чисел для каждого закона распределения (*см. разд. 12.1*), что позволяет эффективно проводить моделирование методами Монте-Карло.

Во-вторых, предусмотрена возможность построения гистограмм и расчета статистических характеристик выборок случайных чисел и случайных процессов, таких как средние, дисперсии, корреляции и т. п. (*см. разд. 12.2*). При этом случайные последовательности могут как создаваться генераторами случайных чисел (методы Монте-Карло, *см. разд. 12.3*), так и вводиться пользователем из файлов.

В-третьих, имеется целый арсенал средств, направленных на интерполяцию/экстраполяцию данных, построение регрессии по методу наименьших квадратов, фильтрацию сигналов. Наконец, реализован ряд численных алгоритмов, осуществляющих расчет различных интегральных преобразований, что позволяет организовать спектральный анализ различного типа.

## 12.1. Статистические распределения

Фундаментальным понятием математической статистики является понятие *случайного числа* (*случайной величины*). Согласно определению, случайная величина принимает то или иное значение, но какое конкретно — зависит от принципиально непредсказуемых обстоятельств опыта и заранее точно пред-

сказано быть не может. Можно лишь говорить о *вероятности* $P(x_K)$ принятия *случайной дискретной* величиной того или иного значения $x_K$ или о вероятности попадания *непрерывной* случайной величины в тот или иной числовой интервал $(x, x+\Delta x)$. Вероятность $P(x_K)$ или $P(x)\cdot(\Delta x)$ соответственно может принимать значения от 0 (такое значение случайной величины совершенно невероятно) до 1 (случайная величина заведомо примет значение от $x$ до $x+\Delta x$). Соотношение $P(x_K)$ называют *законом распределения* случайной величины, а зависимость $P(x)$ между возможными значениями непрерывной случайной величины и вероятностями попадания в их окрестность называется ее *плотностью вероятности* (probability density).

## 12.1.1. Статистические функции

В Mathcad имеется ряд встроенных функций, задающих используемые в математической статистике законы распределения. Они вычисляют как значение плотности вероятности различных распределений по значению случайной величины $x$, так и некоторые сопутствующие функции. Все они, по сути, являются либо встроенными аналитическими зависимостями, либо специальными функциями. Большой интерес представляет наличие генераторов случайных чисел, создающих *выборку* псевдослучайных данных с соответствующим законом распределения, что является основой методов Монте-Карло (*см. разд. 12.2*).

В Mathcad заложена информация о большом количестве разнообразных статистических распределений, включающая, с одной стороны, табулированные функции вероятности, и, с другой, возможность генерации последовательности случайных чисел с соответствующим законом распределения. Для реализации этих возможностей имеются четыре основных категории встроенных функций. Их названия являются составными и устроены одинаковым образом: первая литера идентифицирует определенный закон распределения, а оставшаяся часть (ниже в списке функций она условно обозначена звездочкой) задает смысловую часть встроенной функции:

- `d*(x,par)` — плотность вероятности;
- `p*(x,par)` — функция распределения;
- `q*(P,par)` — квантиль распределения;
- `r*(M,par)` — вектор `M` независимых случайных чисел, каждое из которых имеет соответствующее распределение:
  - `x` — значение случайной величины (аргумент функции);
  - `P` — значение вероятности;
  - `par` — список параметров распределения.

Чтобы получить функции, относящиеся, например, к равномерному распределению, вместо * надо поставить unif и ввести соответствующий список параметров par. Он будет состоять в данном случае из двух чисел: a,b — границ интервала распределения случайной величины.

Перечислим все типы распределения, реализованные в Mathcad, вместе с их параметрами, на этот раз обозначив звездочкой * недостающую первую букву встроенных функций. Некоторые из плотностей вероятности показаны на рис. 12.1.

- *beta(x,s1,s2) — бета-распределение (s1,s2>0 — параметры, 0<x<1).
- *binom(k,n,p) — биномиальное распределение (n — целый параметр, 0≤k≤n и 0≤p≤1 — параметр, равный вероятности успеха единичного испытания).

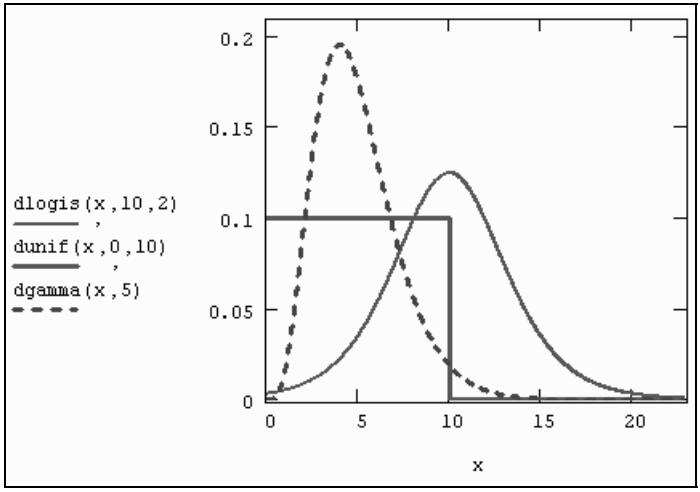



**Рис. 12.1.** Плотность вероятности некоторых распределений

- *cauchy(x,l,s) — распределение Коши (l — параметр разложения, s>0 — параметр масштаба).
- *chisq(x,d) — $\chi^2$ ("хи-квадрат") распределение (d>0 — число степеней свободы).
- *exp(x,r) — экспоненциальное распределение (r>0 — показатель экспоненты).

- `*F(x,d1,d2)` — распределение Фишера (`d1,d2>0` — числа степеней свободы).
- `*gamma(x,s)` — гамма-распределение (`s>0` — параметр формы).
- `*geom(k,p)` — геометрическое распределение (`0≤p≤1` — параметр, равный вероятности успеха единичного испытания).
- `*hypergeom(k,a,b,n)` — гипергеометрическое распределение (`a,b,n` — целые параметры).
- `*lnorm(x,μ,σ)` — логарифмически нормальное распределение (μ — натуральный логарифм математического ожидания, `σ>0` — натуральный логарифм среднеквадратичного отклонения).
- `*logis(x,l,s)` — логистическое распределение (`l` — математическое ожидание, `s>0` — параметр масштаба).
- `*nbinom(k,n,p)` — отрицательное биномиальное распределение (`n>0` — целый параметр, `0<p≤1`).
- `*norm(x,μ,σ)` — нормальное распределение (μ — среднее значение, `σ>0` — среднеквадратичное отклонение).
- `*pois(k,λ)` — распределение Пуассона (`λ>0` — параметр).
- `*t(x,d)` — распределение Стьюдента (`d>0` — число степеней свободы).
- `*unif(x,a,b)` — равномерное распределение (`a<b` — границы интервала).
- `*weibull(x,s)` — распределение Вейбулла (`s>0` — параметр).

> ***Примечание***
>
> Математический смысл каждой из четырех типов функций будет объяснен в следующем разделе на примере распределения Гаусса.

Вставку рассмотренных статистических функций в программы удобно осуществлять с помощью диалогового окна **Insert Function** (Вставка функции). Для этого необходимо выполнить следующие действия:

1. Установите курсор на место вставки функции в документе.
2. Вызовите диалоговое окно **Insert Function** нажатием кнопки ***f(x)*** на стандартной панели инструментов, или командой меню **Insert / Function** (Вставка / Функция), или нажатием клавиш <Ctrl>+<E>.
3. В списке **Function Category** (Категория функции) (рис. 12.2) выберите одну из категорий статистических функций. Категория **Probability Density** (Плотность вероятности) содержит встроенные функции для плотно-

сти вероятности, **Probability Distribution** (Функция распределения) — для вставки функций или квантилей распределения, **Random Numbers** (Случайные числа) — для вставки функции генерации случайных чисел.

4. В списке **Function Name** (Имя функции) выберите функцию в зависимости от требующегося закона распределения. При выборе того или иного элемента списка в текстовых полях в нижней части окна будет появляться информация о назначении выбранной функции.
5. Нажмите кнопку **OK** для вставки функции в документ.

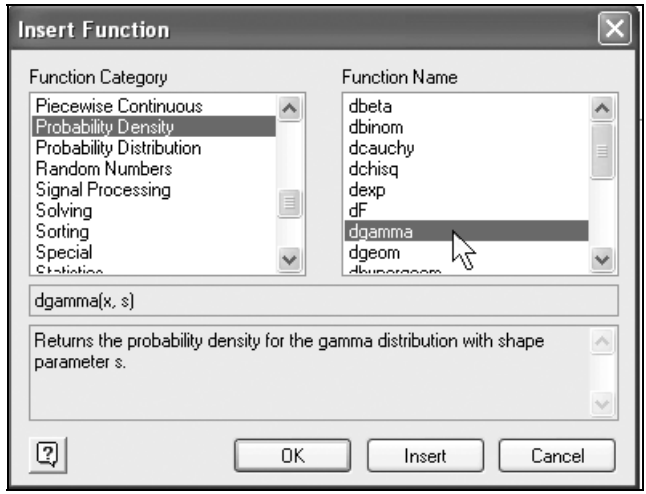



**Рис. 12.2.** Диалоговое окно **Insert Function**

## 12.1.2. Пример: нормальное (Гауссово) распределение

В теории вероятности доказано, что сумма различных независимых случайных слагаемых (независимо от их закона распределения) оказывается случайной величиной, распределенной согласно нормальному закону (так называемая *центральная предельная теорема*). Поэтому нормальное распределение хорошо моделирует самый широкий круг явлений, для которых известно, что на них влияют несколько независимых случайных факторов.

Перечислим еще раз встроенные функции, имеющиеся в Mathcad для описания нормального распределения вероятностей:

- `dnorm(x,μ,σ)` — плотность вероятности нормального распределения;
- `pnorm(x,μ,σ)` — функция нормального распределения;
- `cnorm(x)` — функция нормального распределения для `μ=0,σ=1`;
- `qnorm(P,μ,σ)` — обратная функция нормального распределения;
- `rnorm(M,μ,σ)` — вектор `M` независимых случайных чисел, каждое из которых имеет нормальное распределение:
  - `x` — значение случайной величины;
  - `P` — значение вероятности;
  - `μ` — математическое ожидание;
  - `σ` — среднеквадратичное отклонение.

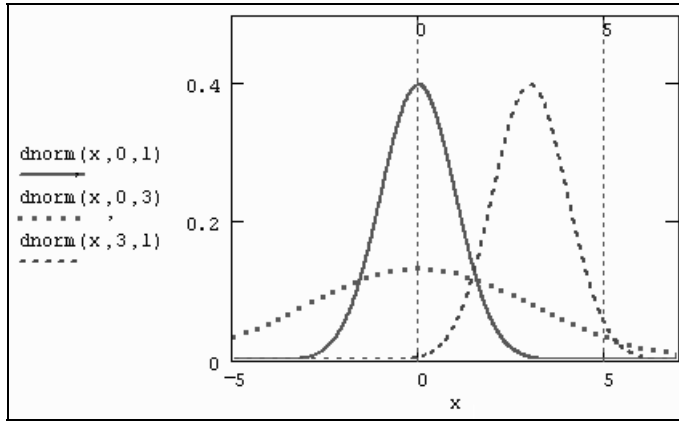



**Рис. 12.3.** Плотность вероятности нормальных распределений

Математическое ожидание и дисперсия являются, по сути, параметрами распределения. Плотность распределения для трех пар значений параметров показана на рис. 12.3. Напомним, что плотность распределения `dnorm` задает вероятность попадания случайной величины `x` в малый интервал от `x` до `x+Δx`. Таким образом, например, для первого графика (сплошная линия) вероятность того, что случайная величина `x` примет значение в окрестности нуля, приблизительно в три раза больше, чем вероятность того, что она примет

значение в окрестности x=2. А значения случайной величины большие 5 и меньшие –5 и вовсе очень маловероятны.

*Функция распределения* F(x) (cumulative probability) — это вероятность того, что случайная величина примет значение, меньшее или равное x. Как следует из математического смысла, она является интегралом от плотности вероятности в пределах от -∞ до x. Функции распределения для упомянутых нормальных законов изображены на рис. 12.4. Функция, обратная F(x) (inverse cumulative probability), называемая еще *квантилем распределения*, позволяет по заданному аргументу p определить значение x, причем случайная величина будет меньше или равна x с вероятностью p.

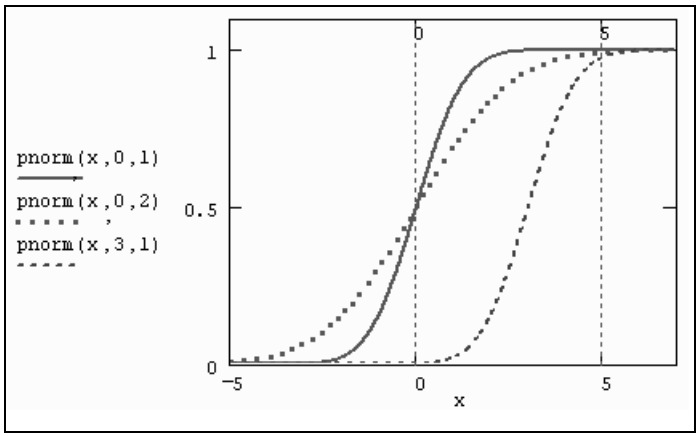



**Рис. 12.4.** Нормальные функции распределения

### *Примечание*

Здесь и далее графики различных статистических функций, показанные на рисунках, получены с помощью Mathcad без каких-либо дополнительных выражений в рабочей области.

Приведем несколько примеров, позволяющих почувствовать математический смысл рассмотренных функций на примере случайной величины x, распределенной по нормальному закону с μ=0 и σ=1 (листинги 12.1—12.5).

**Листинг 12.1. Вероятность того, что x будет меньше 1.881**

```
pnorm (1.881 , 0 , 1) = 0.97
```

**Листинг 12.2. 97%-ный квантиль нормального распределения**

```
qnorm (0.97 , 0 , 1) = 1.881
```

**Листинг 12.3. Вероятность того, что x будет больше 2**

```
1 – pnorm (2 , 0 , 1) = 0.02275
```

**Листинг 12.4. Вероятность того, что x будет находиться в интервале (2,3)**

```
pnorm (3 , 0 , 1) – pnorm (2 , 0 , 1) = 0.021
```

$$\frac{1}{2}\cdot\left(\mathrm{erf}\left(\frac{3}{\sqrt{2}}\right)-\mathrm{erf}\left(\frac{2}{\sqrt{2}}\right)\right)=0.021$$

**Листинг 12.5. Вероятность того, что $|x|<2$**

```
pnorm (2 , 0 , 1) – pnorm (–2 , 0 , 1) = 0.954
```

$$\mathrm{erf}\left(\frac{2}{\sqrt{2}}\right)=0.954$$

Обратите внимание, что задачи двух последних листингов решаются двумя разными способами. Второй из них связан с еще одной встроенной функцией `erf`, называемой *функцией ошибок* (или *интегралом вероятности*, или *функцией Крампа*).

- `erf(x)` — функция ошибок.
- `erfc(x)≡1-erf(x)`.

Математический смысл функции ошибок ясен из листинга 12.5. Интеграл вероятности имеет всего один аргумент, в отличие от функции нормального распределения. Исторически последняя пересчитывалась через табулированный интеграл вероятности по формулам, приведенным в листинге 12.6 для произвольных значений параметров $\mu$ и $\sigma$.

**Листинг 12.6. Вероятность того, что x будет в интервале (2,3)**

$\mu := 5 \qquad \sigma := 2$

```
pnorm (3 , μ , σ) – pnorm (2 , μ , σ) = 0.092
```

$$\frac{1}{2}\cdot\left(\mathrm{erf}\left(\frac{3-\mu}{\sigma\cdot\sqrt{2}}\right)-\mathrm{erf}\left(\frac{2-\mu}{\sigma\cdot\sqrt{2}}\right)\right)=0.092$$

Если вы имеете дело с моделированием методами Монте-Карло, то в качестве генератора случайных чисел с нормальным законом распределения применяйте встроенную функцию `rnorm`. В листинге 12.7 ее действие показано на примере создания двух векторов по `M=500` элементов в каждом с независимыми псевдослучайными числами $x1_i$ и $x2_i$, распределенными согласно нормальному закону. О характере распределения случайных элементов векторов можно судить по рис. 12.5. В дальнейшем мы будем часто сталкиваться с генерацией случайных чисел и расчетом их различных средних характеристик.

**Листинг 12.7. Генерация двух векторов с нормальным законом распределения**

$\sigma := 1 \qquad \mu := 0$

$M := 500$

$x1 := \mathrm{rnorm}(M, \mu, \sigma)$

$x2 := \mathrm{rnorm}(M, \mu, \sigma)$

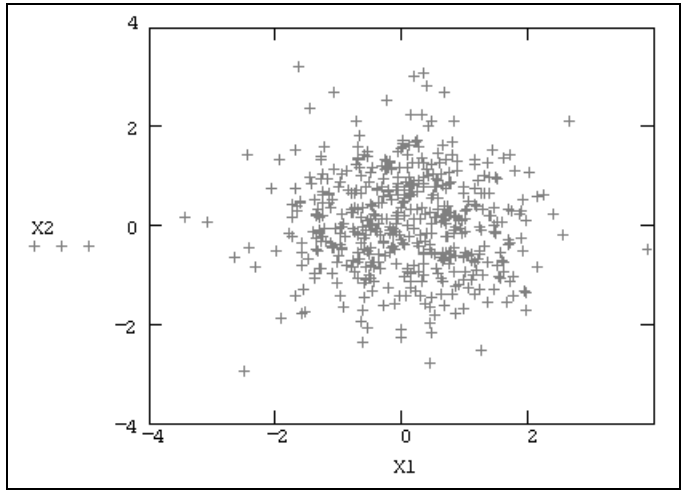



**Рис. 12.5.** Псевдослучайные числа с нормальным законом распределения (продолжение листинга 12.7)

# 12.2. Выборочные статистические характеристики

В большинстве статистических расчетов вы имеете дело с *выборками*: либо со случайными данными, полученными в ходе какого-либо эксперимента (которые выводятся из файла или печатаются непосредственно в документе), либо с результатами генерации случайных чисел, рассмотренными в предыдущих разделах встроенными функциями, моделирующими то или иное явление методом Монте-Карло (*см. разд. 12.3*). Рассмотрим возможности Mathcad по оценке функций распределения и расчету числовых характеристик случайных данных.

## 12.2.1. Гистограммы

*Гистограммой* называется график, аппроксимирующий по случайным данным плотность их распределения. При построении гистограммы область значений случайной величины `(a,b)` разбивается на некоторое количество `bin` сегментов, а затем подсчитывается процент попадания данных в каждый сегмент. Для построения гистограмм в Mathcad имеется несколько встроенных функций. Рассмотрим их, начиная с самой сложной по применению, чтобы лучше разобраться в возможностях каждой из функций.

### Гистограммы с произвольными интервалами

- `hist(intvls,x)` — вектор частоты попадания данных в интервалы гистограммы:
  - `intvls` — вектор, элементы которого задают сегменты построения гистограммы в порядке возрастания $a \le intvls_i < b$;
  - `x` — вектор случайных данных.

Если вектор `intvls` имеет `bin` элементов, то и результат `hist` имеет столько же элементов. Построение гистограммы иллюстрируется листингом 12.8 и рис. 12.6.

**Листинг 12.8. Построение гистограммы**

```
N := 1000
bin := 30
x := rnorm(N, 0, 1)
```

$$lower := floor(min(x))$$

$$upper := ceil(max(x))$$

$$h := \frac{upper - lower}{bin}$$

$$j := 0 .. bin$$

$$int_j := lower + h \cdot j$$

$$f := \frac{1}{N \cdot h} \cdot hist(int, x)$$

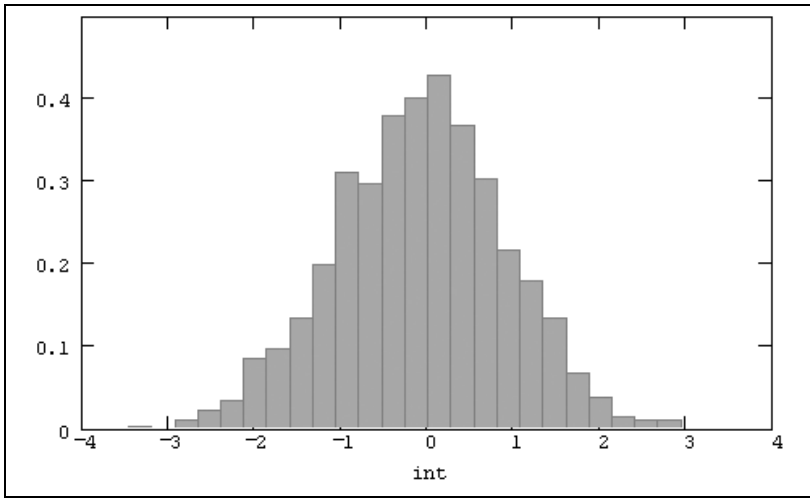



**Рис. 12.6.** Построение гистограммы
(продолжение листинга 12.8)

Для анализа взято `N=1000` данных с нормальным законом распределения, созданных генератором случайных чисел (третья строка листинга). Далее определяются границы интервала `(upper,lower)`, содержащего внутри себя все случайные значения, и осуществляется его разбиение на количество `(bin)` одинаковых сегментов, начальные точки которых записываются в вектор `int` (предпоследняя строка листинга).

### *Примечание*

В векторе `int` можно задать произвольные границы сегментов разбиения так, чтобы они имели разную ширину.

Обратите внимание, что в последней строке листинга осуществлена нормировка значений гистограммы, с тем чтобы она правильно аппроксимировала плотность вероятности, также показанную на графике.

## Гистограммы с равными интервалами

Если нет необходимости задавать сегменты гистограммы разной ширины, то удобнее воспользоваться упрощенным вариантом функции `hist`:

- `hist(bin,x)` — вектор частоты попадания данных в интервалы гистограммы:
  - `bin` — количество сегментов построения гистограммы;
  - `x` — вектор случайных данных.

Для того чтобы использовать этот вариант функции `hist` вместо предыдущего, достаточно заменить первый из ее аргументов в листинге 12.8 следующим образом:

$$f := \frac{1}{N \cdot h} \cdot \mathrm{hist}(\mathrm{int}, x).$$

Недостаток упрощенной формы функции `hist` в том, что по-прежнему необходимо дополнительно определять вектор сегментов построения гистограммы.

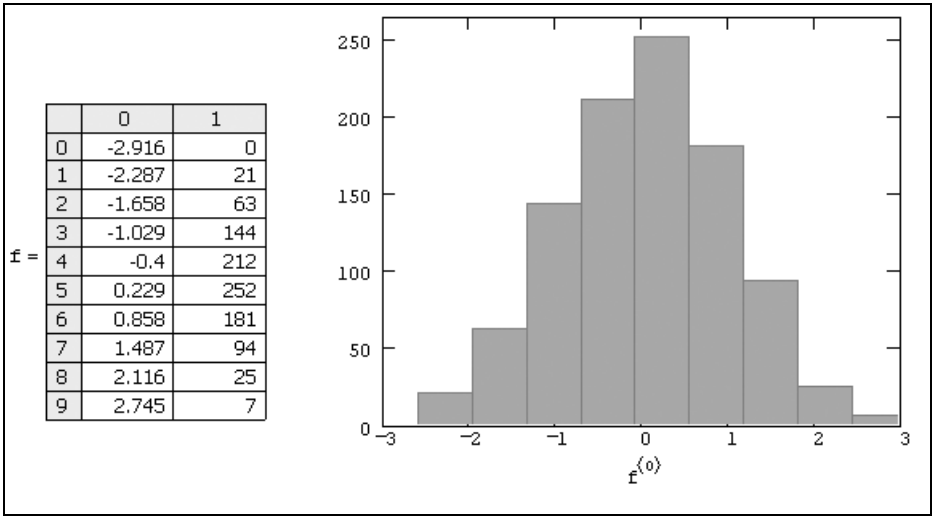
От этого недостатка свободна функция `histogram`:

- `histogram(bin,x)` — матрица гистограммы размера `bin×2`, состоящая из столбца сегментов разбиения и столбца частоты попадания в них данных:
  - `bin` — количество сегментов построения гистограммы;
  - `x` — вектор случайных данных.

Примеры использования функции `histogram` приведены в листинге 12.9 и на рис. 12.7. Сравнение с предыдущим листингом подчеркивает простоту построения гистограммы этим способом (стоит отметить, что в листинге 12.9, в отличие от предыдущего, мы не нормировали гистограмму).

**Листинг 12.9. Упрощенный вариант построения гистограммы**

```
N := 1000
bin := 10
x := rnorm(N, 0, 1)
f := histogram(bin, x)
```



f =

| | 0 | 1 |
|---|---|---|
| 0 | -2.916 | 0 |
| 1 | -2.287 | 21 |
| 2 | -1.658 | 63 |
| 3 | -1.029 | 144 |
| 4 | -0.4 | 212 |
| 5 | 0.229 | 252 |
| 6 | 0.858 | 181 |
| 7 | 1.487 | 94 |
| 8 | 2.116 | 25 |
| 9 | 2.745 | 7 |



**Рис. 12.7.** График и матрица гистограммы
(продолжение листинга 12.9)

***Примечание***

Для того чтобы назначить двумерному графику тип гистограммы, в диалоговом окне **Formatting Currently Selected Graph** (Форматирование) установите на вкладке **Traces** (Графики) тип списка **bar** (Столбцы) или **solidbar** (Гистограмма). На рис. 12.6 и 12.7 применены установки второго типа: закрашенными столбиками (**solidbar**).

## 12.2.2. Среднее и дисперсия

В Mathcad имеется ряд встроенных функций для расчетов числовых статистических характеристик рядов случайных данных:

- `mean(x)` — выборочное среднее значение;
- `median(x)` — выборочная *медиана* (median) — значение аргумента, которое делит гистограмму плотности вероятностей на две равные части;
- `var(x)` — выборочная *дисперсия* (variance);
- `stdev(x)` — *среднеквадратичное* (или *стандартное*) *отклонение* (standard deviation);
- `max(x),min(x)` — максимальное и минимальное значения выборки;
- `mode(x)` — наиболее часто встречающееся значение выборки;

- `Var(x),Stdev(x)` — выборочная дисперсия и среднеквадратичное отклонение в другой нормировке:
  - `x` — вектор (или матрица) с выборкой случайных данных.

Пример использования первых четырех функций приведен в листинге 12.10.

**Листинг 12.10. Расчет числовых характеристик случайного вектора**

```
N := 1000
x := runif(N, 0, 1)
mean(x) = 0.497
median(x) = 0.501
var(x) = 0.082
√Var(x) = 0.287   stdev(x) = 0.287
```

Определение статистических характеристик случайных величин приведено в листинге 12.11 на еще одном примере обработки выборки малого объема (по пяти данным). В том же листинге иллюстрируется применение еще двух функций, которые имеют смысл дисперсии и стандартного отклонения в несколько другой нормировке. Сравнивая различные выражения, вы без труда освоите связь между встроенными функциями.

> ***Внимание!***
>
> Осторожно относитесь к написанию первой литеры в этих функциях, особенно при обработке малых выборок (листинг 12.11).

**Листинг 12.11. К определению статистических характеристик**

```
x := ( 5  2  14  3  2 )^T
N := length(x)          N = 5
(1/N) · Σ_{i=0}^{N-1} x_i = 5.2
                        m := mean(x)     m = 5.2
median(x) = 3           mode(x) = 2      max(x) = 14   min(x) = 2
(1/N) · Σ_{i=0}^{N-1} (x_i − m)^2 = 20.56
```

$$\mathrm{var}(x) = 20.56 \qquad \mathrm{Var}(x)\cdot\frac{N-1}{N} = 20.56$$

$$\sqrt{\mathrm{var}(x)} = 4.534 \qquad \mathrm{stdev}(x) = 4.534$$

$$\mathrm{Stdev}(x)\cdot\sqrt{\frac{N-1}{N}} = 4.534$$

$$\frac{1}{N-1}\cdot\sum_{i=0}^{N-1}\left(x_i - m\right)^2 = 25.7$$

$$\mathrm{Var}(x) = 25.7 \qquad \mathrm{var}(x)\cdot\frac{N}{N-1} = 25.7$$

$$\sqrt{\mathrm{Var}(x)} = 5.07 \qquad \mathrm{Stdev}(x) = 5.07$$

$$\mathrm{stdev}(x)\cdot\sqrt{\frac{N}{N-1}} = 5.07$$

Иногда в статистике встречаются и иные функции, например, помимо арифметического среднего, применяются другие средние значения:

- `gmean(x)` — геометрическое среднее выборки случайных чисел;
- `hmean(x)` — гармоническое среднее выборки случайных чисел.

Математическое определение этих функций и пример их использования в Mathcad приведены в листинге 12.12.

**Листинг 12.12. Вычисление различных средних значений**

```
N := 10
x := runif(N, 0, 1)
```

$$\frac{1}{N}\cdot\sum_{i=0}^{N-1} x_i = 0.338 \qquad \mathrm{mean}(x) = 0.338$$

$$\left(\frac{1}{N}\cdot\sum_{i=0}^{N-1}\frac{1}{x_i}\right)^{-1} = 0.012 \qquad \mathrm{hmean}(x) = 0.012$$

$$\sqrt[N]{\prod_{i=0}^{N-1} x_i} = 0.171 \qquad \mathrm{gmean}(x) = 0.171$$

## 12.2.3. Примеры: выборочная оценка дисперсии и среднего нормальной случайной величины

Типовые задачи математической статистики связаны с получением тех или иных интервальных и точечных оценок различных параметров случайной выборки. Приведем пример двух задач, иллюстрирующих назначение и принципы применения введенных в предыдущих разделах статистических функций.

### Интервальная оценка дисперсии

Требуется определить числовой интервал `(L,U)`, внутри которого будет лежать с вероятностью `1-α=75%` дисперсия нормальной случайной величины, исходя из объема выборки в `N` чисел. Эта задача решается в статистике с помощью $\chi^2$-распределения (листинг 12.13).

**Листинг 12.13. Интервальное оценивание дисперсии**

```
N := 50
x := rnorm ( N , 0 , 1 )
α := 0.25                                   1 − α = 0.75
χ2 0 := qchisq( α/2 , N − 1 )               χ2 0 = 37.901
χ2 1 := qchisq( 1 − α/2 , N − 1 )           χ2 1 = 60.53
L := (N − 1) · Stdev (x) ^ 2 / χ2 1         U := (N − 1) · Stdev (x) ^ 2 / χ2 0
L = 0.662                                   U = 1.057
```

Указанный интервал называется `(1-α)` *доверительным интервалом*. Обратите внимание на использование при решении данной задачи функции `Stdev` (с прописной буквы) для расчета выборочного стандартного отклонения. В статистике часто встречаются выражения, которые более удобно записывать через функции в такой нормировке, именно для этого они и появились в Mathcad.

## Проверка статистических гипотез

В статистике рассматривается огромное число задач, связанных с проверкой тех или иных гипотез H. Разберем пример простой гипотезы. Пусть имеется выборка N чисел с нормальным законом распределения и неизвестными дисперсией и математическим ожиданием. Требуется принять или отвергнуть гипотезу H о том, что математическое ожидание закона распределения равно некоторому числу μ0=0.2.

Задачи проверки гипотез требуют задания *уровня критерия* проверки гипотезы α, который описывает вероятность ошибочного отклонения истинной H. Если взять α очень малым, то гипотеза, даже если она ложная, будет почти всегда приниматься; если, напротив, взять α близким к 1, то критерий будет очень строгим, и гипотеза, даже верная, скорее всего, будет отклонена.

В нашем случае гипотеза состоит в том, что μ0=0.2, а альтернатива — что μ0≠0.2. Оценка математического ожидания, как следует из курса классической статистики, решается с помощью распределения Стьюдента с параметром N-1 (этот параметр называется *степенью свободы распределения*).

Для проверки гипотезы (листинг 12.14) рассчитывается (α/2)-квантиль распределения Стьюдента T, который служит критическим значением для принятия или отклонения гипотезы. Если соответствующее выборочное значение t по модулю меньше T, то гипотеза принимается (считается верной). В противном случае гипотезу следует отвергнуть.

**Листинг 12.14. Проверка гипотезы о математическом ожидании при неизвестной дисперсии**

```
N := 50
x := rnorm (N , 0 , 1)
α := 0.1                                  1 - α = 0.9
μ0 := 0.2
t := mean(x) - μ0 / (Stdev(x) / √N)       t = -1.883
T := qt(1 - α/2 , N - 1)                  T = 1.677
|t| < T = 0
```

В последней строке листинга вычисляется истинность или ложность условия, выражающего решение задачи. Поскольку условие оказалось ложным (равным не 1, а 0), то гипотезу необходимо отвергнуть.

На рис. 12.8 показано распределение Стьюдента с `N-1` степенью свободы, вместе с критическими значениями, определяющими соответствующий интервал. Если `t` (оно тоже показано на графике) попадает в него, то гипотеза принимается; если не попадает (как произошло в данном случае) — отвергается. Если увеличить $\alpha$, ужесточив критерий, то границы интервала будут сужаться по сравнению с показанными на рисунке.

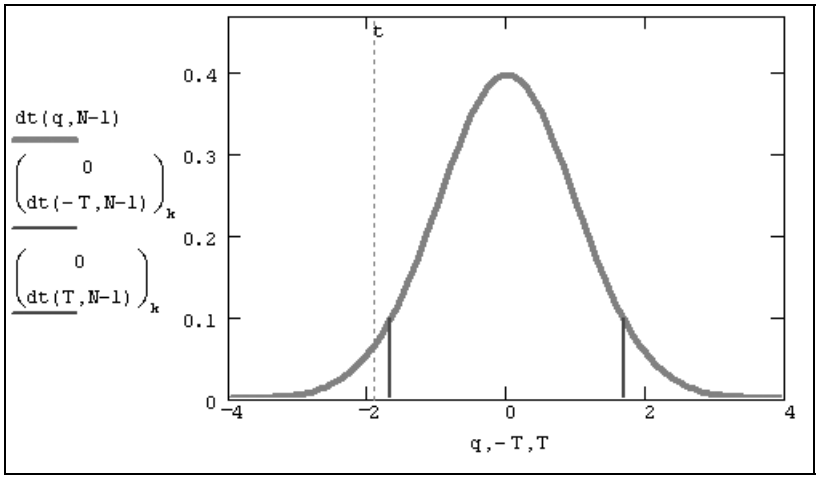



**Рис. 12.8.** К задаче проверки статистических гипотез (продолжение листинга 12.14)

В листинге 12.15 приводится альтернативный способ проверки той же самой гипотезы, связанный с вычислением значения не квантиля, а самого распределения Стьюдента.

**Листинг 12.15. Другой вариант проверки гипотезы (продолжение листинга 12.14)**

```
pt(t,N-1) = 0.033
```

$$\frac{\alpha}{2} < \mathrm{pt}(t, N-1) < 1 - \frac{\alpha}{2} = 0$$

Мы разобрали только два характерных примера статистических вычислений. Однако с помощью Mathcad легко решаются самые разнообразные задачи математической статистики.

***Примечание***

Большое количество задач разобрано в ресурсах Mathcad в рубрике **Statistics** (Статистика) быстрых шпаргалок.

## 12.2.4. Корреляция

Функции, устанавливающие связь между парами двух случайных векторов, называются *ковариацией* и корреляцией (или, по-другому, *коэффициентом корреляции*). Они различаются нормировкой, как следует из их определения (листинг 12.16):

- `corr(x)` — коэффициент корреляции двух выборок;
- `cvar(x)` — ковариация двух выборок:
  - `x1,x2` — векторы (или матрицы) одинакового размера с выборками случайных данных.

**Листинг 12.16. Расчет ковариации и корреляции**

$N := 100 \qquad \sigma := 1$

$x1 := rnorm(N, 0, \sigma) \qquad x2 := rnorm(N, 0, \sigma)$

$m1 := mean(x1) \qquad m2 := mean(x2)$

$\sigma 1 := stdev(x1) \qquad \sigma 2 := stdev(x2)$

$$\frac{1}{N} \cdot \sum_{i=0}^{N-1} \left[ \left( x1_i - m1 \right) \cdot \left( x2_i - m2 \right) \right] = 5.255 \times 10^{-3}$$

$cvar(x1, x2) = 5.255 \times 10^{-3}$

$$\frac{cvar(x1, x2)}{\sigma 1 \cdot \sigma 2} = 5.337 \times 10^{-3}$$

$corr(x1, x2) = 5.337 \times 10^{-3}$

***Примечание***

Как видно, полученное значение корреляции близко к нулю, т. е. псевдослучайные числа генерируются практически независимо. О том, как создать массив коррелированных чисел, написано ниже (*см. разд. 12.3.2*).

## 12.2.5. Новые функции корреляционного анализа сигналов

В Mathcad 12 появились две функции, связанные с корреляционной обработкой сигналов и изображений:

- `correl(x,y)` — вектор, представляющий значения коэффициента ковариации двух векторов;
- `correl2d(A,K)` — матрица, равная ковариации матрицы-аргумента и матрицы-окна:
  - `x,y` — векторы;
  - `A` — матрица-прототип;
  - `K` — матрица-окно (размера, меньшего чем `A`).

Смысл действия одномерной функции `correl` заключается в последовательном сдвиге одного вектора относительно другого, перемножении и суммировании их элементов, стоящих в таком положении друг напротив друга. Это не совсем ковариация в терминах математической статистики, поскольку не осуществляется нормировки (деления полученной суммы на число перемноженных элементов). Работа двумерной функции `correl2d` состоит в последовательном позиционировании "маленькой" матрицы (окна) на фоне "крупной" (прототипа), перемножении их элементов, находящихся друг над другом, и суммировании их. В результате получается элемент матрицы корреляции, соответствующий центрам наложения матрицы-прототипа и матрицы-окна. Обе эти функции играют определенную роль в задачах обработки сигналов.

## 12.2.6. Коэффициенты асимметрии и эксцесса

*Коэффициент асимметрии* задает степень асимметричности плотности вероятности относительно оси, проходящей через ее центр тяжести. Коэффициент асимметрии определяется третьим центральным моментом распределения. В любом симметричном распределении с нулевым математическим ожиданием, например, нормальным, все нечетные моменты, в том числе и третий, равны нулю, поэтому коэффициент асимметрии тоже равен нулю.

Степень сглаженности плотности вероятности в окрестности главного максимума задается еще одной величиной — *коэффициентом эксцесса*. Он показывает, насколько острую вершину имеет плотность вероятности по сравнению с нормальным распределением. Если коэффициент эксцесса больше нуля, то распределение имеет более острую вершину, чем распределение Гаусса, если меньше нуля, то более плоскую.

Для расчета коэффициентов асимметрии и эксцесса в Mathcad имеются две встроенные функции:

- `kurt(x)` — коэффициент эксцесса (*kurtosis*) выборки случайных данных `x`;
- `skew(x)` — коэффициент асимметрии (*skewness*) выборки случайных данных `x`.

Примеры расчета коэффициентов асимметрии и эксцесса для распределения Вейбулла приведены в листинге 12.17.

**Листинг 12.17. Расчет выборочных коэффициентов асимметрии и эксцесса**

```
x := rweibull (1000 , 1.5)
skew (x) = 1.216
kurt (x) = 1.89
```

## 12.2.7. Статистические функции матричного аргумента

Все рассмотренные примеры работы статистических функций относились к векторам, элементы которых были случайными числами. Но точно так же все эти функции применяются и по отношению к выборкам случайных данных, сгруппированных в матрицы. При этом статистические характеристики рассчитываются для совокупности всех элементов матрицы, без разделения ее на строки и столбцы. Например, если матрица имеет размерность `M×N`, то и объем выборки будет равен `M·N`.

Соответствующий пример вычисления среднего значения приведен в листинге 12.18. В его первой строке определяется матрица данных `x` размера `4×2`. Действие встроенной функции `mean` матричного аргумента (последняя строка листинга) иллюстрируется явным суммированием элементов матрицы `x` (предпоследняя строка). Действие прочих встроенных функций на матрицы совершенно аналогично действию их на векторы (листинг 12.19).

**Листинг 12.18. Вычисление среднего значения элементов матрицы**

$$x := \begin{pmatrix} 1.0 & 4 \\ 1.5 & 7 \\ 0.9 & 1.2 \\ 1.2 & 12 \end{pmatrix}$$

$M := rows(x)$ $M = 4$

$N := cols(x)$ $N = 2$

$$\frac{1}{M \cdot N} \cdot \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} x_{i,j} = 3.6$$

$mean(x) = 3.6$

**Листинг 12.19. Действие различных статистических функций на матрицу**

$median(x) = 1.35$

$mode(x) = 1.2$

$var(x) = 14.033$

$Var(x) = 16.037$

$stdev(x) = 3.746$

$Stdev(x) = 4.005$

***Примечание***

Некоторые статистические функции (например, вычисления ковариации) имеют два аргумента. Они также могут быть матрицами, но, в соответствии со смыслом функции, должны иметь одинаковую размерность.

Большинству статистических функций позволяется иметь в качестве аргументов даже не одну матрицу, а любое количество матриц, векторов и скаляров. Числовые характеристики будут рассчитаны для всей совокупности значений аргументов функции. Соответствующий пример приведен в листинге 12.20.

**Листинг 12.20. Статистические функции нескольких аргументов**

$x := (1 \quad -1 \quad 4)$

$$y := \begin{pmatrix} 3 \\ 8 \end{pmatrix}$$

$$z := \begin{pmatrix} 3 & 7 \\ 11 & 4 \end{pmatrix}$$

$mean(x, y, z) = 4.444$

```
stdev(x, 5, 77) = 29.976
median(2) = 2
mode(y, z) = 3
```

# 12.3. Методы Монте-Карло

Для моделирования различных физических, экономических и прочих эффектов широко распространены методы, называемые *методами Монте-Карло*, обязанные своим названием европейскому центру азартных игр, основанных на случайных событиях. Основная идея этих методов состоит в создании определенной последовательности *случайных* чисел, моделирующей тот или иной эффект, например, шум в физическом эксперименте, случайную динамику биржевых индексов и т. п. Фактически все содержание предыдущего раздела (*см. разд. 12.2*) являлось примером реализации методов Монте-Карло, поскольку содержало расчет тех или иных статистических характеристик имитационных данных, полученных при помощи генератора случайных чисел.

Рассмотрим еще несколько типовых задач, относящихся к методам Монте-Карло, попутно изучая соответствующие возможности, заложенные в Mathcad.

## 12.3.1. Генерация псевдослучайных чисел

Как уже отмечалось (*см. разд. 12.1.1*), для генерации `M`-компонентного вектора независимых псевдослучайных чисел имеется ряд встроенных функций, реализующих различные типы статистических распределений и имеющих вид `r*(M,par)`, где `*` — идентификатор, а `par` — список параметров конкретного распределения. В частности, генератор нормальных псевдослучайных чисел был рассмотрен ранее (*см. разд. 12.1.2*), а для равномерного распределения предусмотрено две встроенных функции:

- `runif(M,a,b)` — вектор `M` независимых случайных чисел, каждое из которых имеет равномерное распределение;
- `rnd(x)` — случайное число, имеющее равномерную плотность распределения на интервале `(0,x)`:
  - `x` — значение случайной величины;
  - `P` — значение вероятности;
  - `(a,b)` — интервал, на котором случайная величина распределена равномерно.

Чаще всего в несложных программах применяется последняя функция, которая приводит к генерации одного псевдослучайного числа. Наличие такой встроенной функции в Mathcad — дань традиции, применяемой в большинстве сред программирования. Возможно, именно скалярный тип этой функции определяет простоту и привычность ее использования в расчетах моделей типа Монте-Карло.

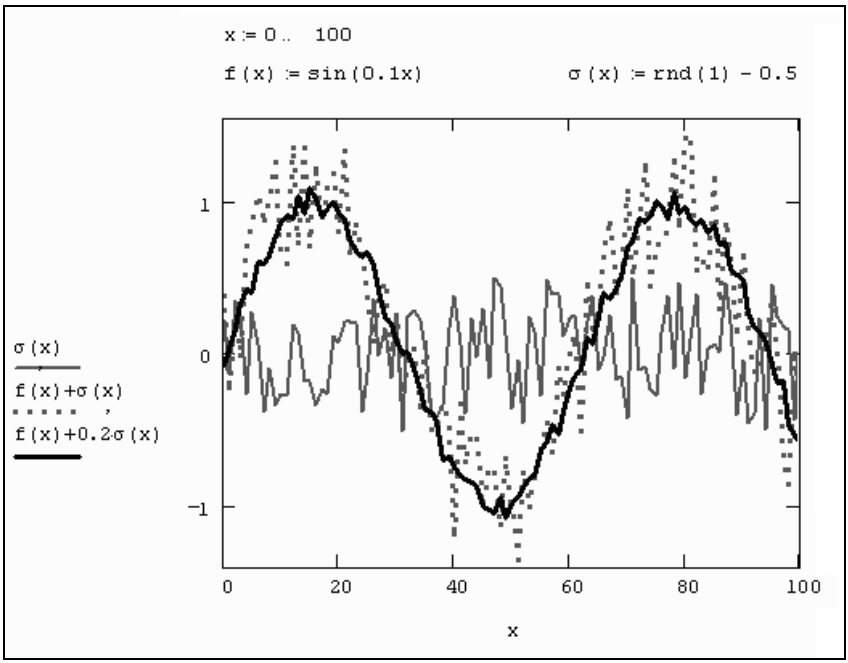



**Рис. 12.9.** Модель сигнал/шум с равномерным законом распределения

Рисунок 12.9 демонстрирует широко распространенный прием моделирования, основанный на смеси некоторого полезного сигнала `f(x)` и шумовой компоненты σ, что характерно для типичного физического эксперимента. В качестве шума используется серия равномерно распределенных псевдослучайных чисел. Различные соотношения интенсивностей сигнала и шума определяют различные условия модельной задачи и позволяют эффективно протестировать алгоритмы, которые разработаны для ее решения.

### *Примечание*

Более сложные статистические расчеты для модели сигнал/шум приведены в листинге 12.24 (*см. разд. 12.3.4*).

## Параметры генераторов псевдослучайных чисел

В Mathcad применяются типичные алгоритмы генерации последовательностей псевдослучайных чисел, которые используют в качестве "отправной точки" некоторое *начальное* (или *отправное*) *значение* (seed value). Это начальное значение используется для того, чтобы совершить над ним определенные математические действия (к примеру, взять остаток от деления на некоторое другое число) и получить в итоге первое псевдослучайное число последовательности. Затем те же математические операции совершаются с первым числом для получения второго, и т. д.

Несложно догадаться, что если использовать все время одно и то же начальное значение генератора псевдослучайных чисел, то, открывая всякий раз новый документ со встроенной функцией получения тех или иных псевдослучайных чисел, будет выдаваться в точности одна и та же их последовательность. Сами числа внутри последовательности будут "почти" случайными (значимость этого "почти" будет зависеть только от качества алгоритма генерации), но вот сама последовательность при каждом открытии документа будет одной и той же.

> ***Примечание***
>
> Вы можете сами убедиться в идентичности выдаваемых последовательностей псевдослучайных чисел, открывая и закрывая несколько раз документ с рис. 12.9 или любой документ с иным датчиком случайных чисел, описанных в данном разделе. Любопытным также будет эксперимент по последовательному применению нескольких встроенных функций генерации псевдослучайных чисел в пределах одного документа.

Для того чтобы иметь возможность поменять саму последовательность сгенерированных случайным датчиком чисел, в Mathcad 11 (и выше) предусмотрена новая возможность явного определения используемого им начального значения. Для этого вызовите командой меню **Tools / Worksheet Options** (Сервис / Опции документа) диалог **Worksheet Options** (Опции документа) и на вкладке **Built-In Variables** (Встроенные переменные) введите желаемое (целое) число в поле ввода **Seed value for random numbers** (Начальное значение для генератора псевдослучайных чисел).

Альтернативный способ заключается в использовании соответствующей встроенной функции непосредственно в документе:

- `seed(x)` — функция установки нового начального значения для генератора псевдослучайных чисел:
  - `x` — новое начальное значение для генератора псевдослучайных чисел (целое число от `1` до `2147483647`).

## 12.3.2. Генерация коррелированных выборок

До сих пор мы рассматривали наиболее простой случай применения генераторов независимых случайных чисел. В методах Монте-Карло часто требуется создавать случайные числа с определенной *корреляцией*. Приведем пример программы, создающей два вектора `x1` и `x2` одинакового размера и одним и тем же распределением, случайные элементы которых попарно коррелированны с коэффициентом корреляции `R` (листинг 12.21).

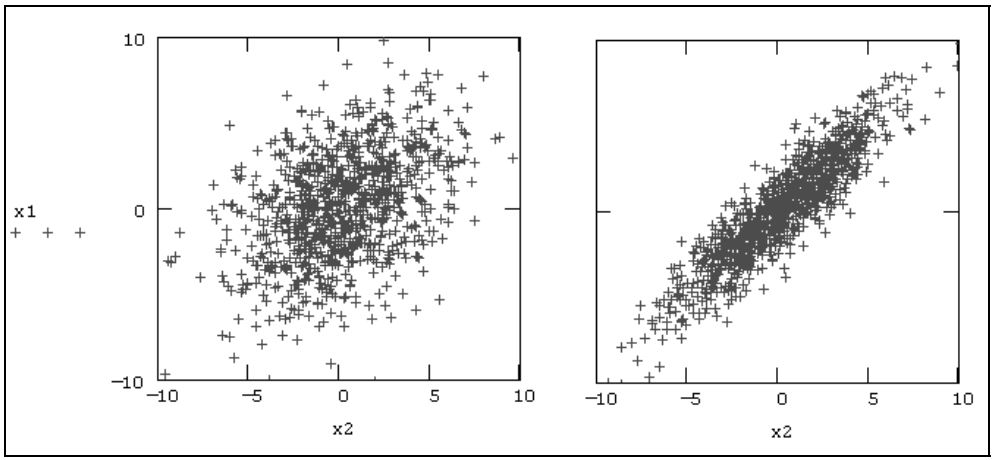



**Рис. 12.10.** Псевдослучайные числа с корреляцией `R=0.4` (продолжение листинга 12.21) и `R=0.9`

**Листинг 12.21. Генерация попарно коррелированных случайных чисел**

$\sigma := 3$ $N := 1000$

$R := 0.4$

$x1 := \mathrm{rnorm}(N, 0, \sigma)$

$x2 := R \cdot x1 + \sqrt{1 - R^2} \cdot \mathrm{rnorm}(N, 0, \sigma)$

$\mathrm{corr}(x1, x2) = 0.415$

Результат действия программы для `R=0.4` показан на рис. 12.10, слева. Сравните полученную выборку с правым графиком, полученным для высокой корреляции (`R=0.9`) и с рис. 12.5 (*см. разд. 12.1.2*) для независимых данных, т. е. `R=0`.

## 12.3.3. Моделирование случайного процесса

Встроенные функции для генерации случайных чисел создают выборку из случайных данных $A_i$. Часто требуется создать непрерывную или дискретную случайную функцию A(t) одной или нескольких переменных (*случайный процесс* или *случайное поле*), значения которой будут упорядочены относительно своих переменных. Создать псевдослучайный процесс можно способом, представленным в листинге 12.22.

**Листинг 12.22. Генерация псевдослучайного процесса**

```
N := 20

τ := 0.5

Tmax := (N − 1) · τ

j := 0 .. N − 1

Tj := j · τ

x := rnorm(N, 0, 1)

KS1 := cspline(T, x)

A(t) := interp(KS1, T, x, t)
```

В первых строках листинга 12.22 определено количество N независимых случайных чисел, которые будут впоследствии сгенерированы, и радиус временной корреляции τ. В следующих трех строках определяются моменты времени $T_j$, которым будут отвечать случайные значения $A(t_j)$. Создание нормального случайного процесса сводится к генерации обычным способом вектора независимых случайных чисел x и построению интерполяционной зависимости в промежутках между ними. В листинге 12.22 используется сплайн-интерполяция (*см. главу 13*).

В результате получается случайный процесс A(t), радиус корреляции которого определяется расстоянием τ между точками, для которых строится интерполяция. График случайного процесса A(t) вместе с исходными случайными числами показан на рис. 12.11. Случайное поле можно создать несколько более сложным способом с помощью многомерной интерполяции.

***Примечание***

Простой пример генерации двумерного случайного поля вы найдете на компакт-диске, прилагаемом к книге.

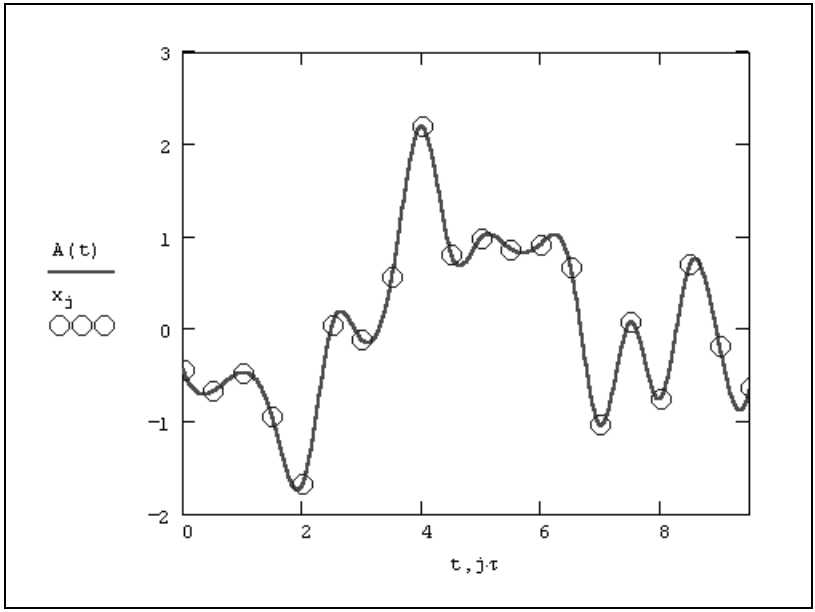



**Рис. 12.11.** Псевдослучайный процесс
(продолжение листинга 12.22)

К случайным процессам, сгенерированным таким способом, как и к данным эксперимента, применяются любые статистические методы обработки, например, корреляционный или спектральный анализ. Приведем в качестве примера листинг 12.23, показывающий, как организовать расчет корреляционной функции случайного процесса.

**Листинг 12.23. Дискретизация случайного процесса и вычисление корреляционной функции (продолжение листинга 12.19)**

$\Delta := 0.02$

$M := 20$

$n := \text{floor}\left(\frac{Tmax}{\Delta}\right)$ $\quad n = 475$

$j := 0 .. \ n$

$Y_j := A(\Delta \cdot j)$

$m := \text{mean}(Y)$

$$D := var(Y) \qquad D = 0.785$$

$$R(j) := \frac{1}{(n - 2 \cdot M)} \cdot \sum_{i = M}^{n-M} \frac{(Y_{i+j} - m) \cdot (Y_i - m)}{D}$$

$$R(0) = 1.025$$

Дискретизация интервала `(0,Tmax)` для случайного процесса `A(t)` произведена с различным элементарным интервалом $\Delta$ (первая строка листинга). В зависимости от значения $\Delta$ получается различный объем `n` выборки случайных чисел $Y_i$, являющихся значениями случайной функции `A(t)` в точках дискретизации. В последних четырех строках определяются различные характеристики случайной величины `Y`, являющиеся, по сути, характеристиками случайного процесса `A(t)`. График рассчитанной в `2·M+1` точках корреляционной функции `R(j)` показан на рис. 12.12.

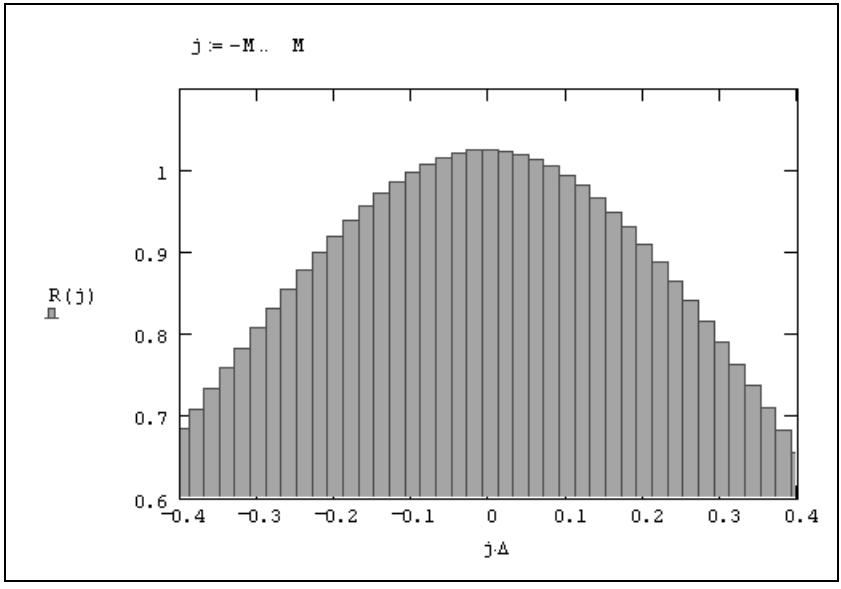



**Рис. 12.12.** Корреляционная функция
(продолжение листингов 12.22—12.23)

***Примечание***

Внимательному читателю предлагается самостоятельно ответить на вопрос: почему при таком расчете корреляционной функции ее значение `R(0)` не равно 1, как должно быть по определению?

## 12.3.4. Пример: огибающая и фаза нормального случайного процесса

Завершим разговор о моделировании случайных процессов примером, часто встречающимся в задачах статистической радиофизики. Рассмотрим модель, представляющую собой сумму гармонической функции и нормально распределенной шумовой компоненты, которая хорошо описывает передачу сигнала в электронных устройствах в условиях помех (листинги 12.24—12.25) и называется *узкополосным нормальным процессом*. Как известно, узкополосный процесс представим в виде `E(t)·exp(i·ϕ(t))`, где случайные функции `E(t)` и `ϕ(t)` называются, соответственно, его *огибающей* и *фазой*. Мы приведем пример нулевого сигнала (т. е. расчет огибающей и фазы чистого случайного процесса), хотя минимальное изменение листинга 12.25 даст вам возможность промоделировать и ненулевые значения сигнала.

Первая половина листинга 12.24 представляет собой подготовительный этап, заключающийся в генерации двух векторов с нормальным распределением вероятности. Из курса математической статистики известно, что узкополосный гауссов процесс можно представить в виде, приведенном в последней строке листинга 12.24, причем случайные функции `A(t)` и `C(t)` называются *квадратурными составляющими* нормального случайного процесса. Графики `A(t)` и `C(t)` показаны на рис. 12.13. Листинг 12.25 содержит суммирование полученных шумовых компонент с составляющими гармонического сигнала и выдает в качестве результата функции `E(t)` и `ϕ(t)` (они показаны на рис. 12.14).

**Листинг 12.24. Квадратурные составляющие нормального случайного процесса**

```
N := 200                τ := 1

j := 0 .. N − 1

Tj := j·τ

Tmax := N·τ             Tmax = 200
```

$\sigma := 1 \qquad \mu := 0$

$X1 := \mathrm{rnorm}(N, \mu, \sigma)$ $\qquad$ $X2 := \mathrm{rnorm}(N, \mu, \sigma)$

$KS1 := \mathrm{cspline}(T, X1)$ $\qquad$ $KS2 := \mathrm{cspline}(T, X2)$

$A(t) := \mathrm{interp}(KS1, T, X1, t)$ $\qquad$ $C(t) := \mathrm{interp}(KS2, T, X2, t)$

$\Omega := \pi \cdot 1$

$\xi(t) := A(t) \cdot \cos(\Omega \cdot t) + C(t) \cdot \sin(\Omega \cdot t)$

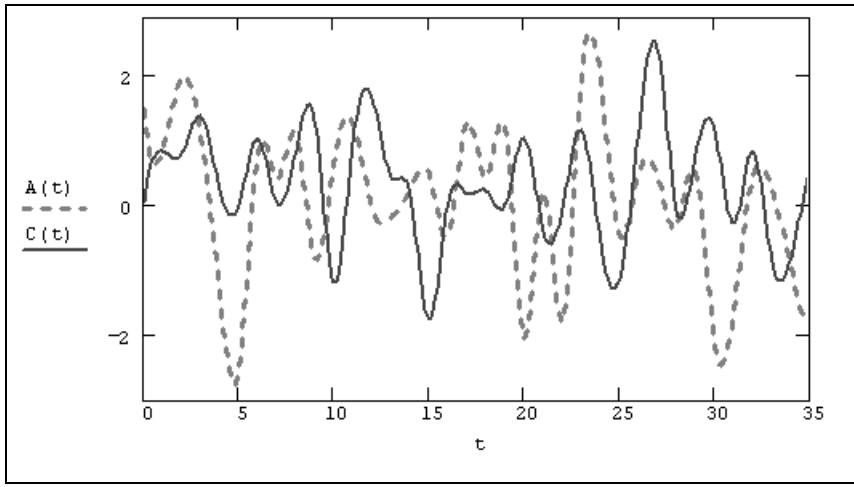



**Рис. 12.13.** Квадратурные составляющие случайного процесса (продолжение листинга 12.24)

## Листинг 12.25. Огибающая и фаза нормального случайного процесса (продолжение листинга 12.24)

$Amax := 0$

$u(t) := Amax \cdot \cos(\Omega \cdot t)$

$v(t) := Amax \cdot \sin(\Omega \cdot t)$

$S(t) := u(t) + v(t)$

$$E(t) := \sqrt{(A(t) + u(t))^2 + (C(t) + v(t))^2}$$

$$\phi(t) := \mathrm{atan}\left(\frac{C(t) + v(t)}{A(t) + u(t)}\right)$$

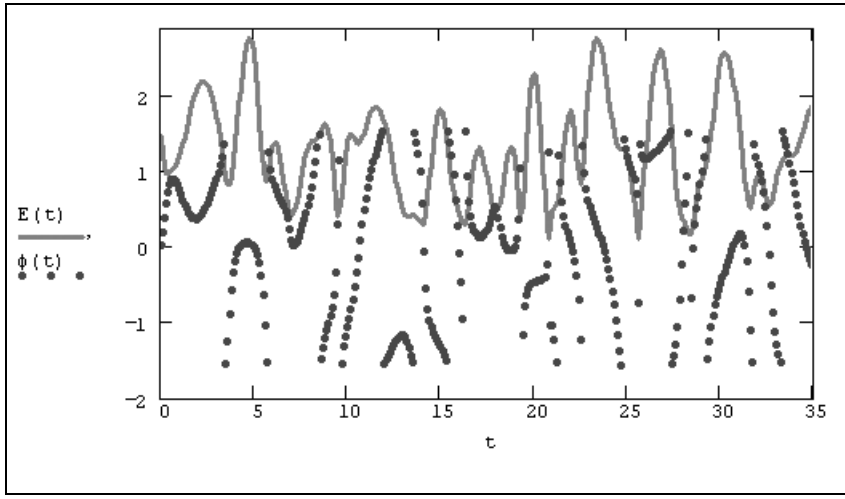



**Рис. 12.14.** Огибающая и фаза нормального случайного процесса (продолжение листингов 12.24—12.25)

# Глава 13

# Интерполяция и регрессия

Посвятим данную главу самым простым методам обработки данных — интерполяции/экстраполяции и регрессии. Будем считать, что основным объектом исследования будет выборка экспериментальных данных, которые, чаще всего, представляются в виде массива, состоящего из пар чисел $(x_i, y_i)$ (проблеме ввода/вывода числовых данных во внешние файлы посвящен заключительный раздел этой главы). В связи с этим возникает задача аппроксимации дискретной зависимости $y(x_i)$ непрерывной функцией $f(x)$. Функция $f(x)$, в зависимости от специфики задачи, может отвечать различным требованиям:

- $f(x)$ должна проходить через точки $(x_i, y_i)$, т. е. $f(x_i)=y_i, i=1...n$. В этом случае (*см. разд. 13.1*) говорят об *интерполяции* данных функцией $f(x)$ во внутренних точках между $x_i$ или *экстраполяции* за пределами интервала, содержащего все $x_i$;
- $f(x)$ должна некоторым образом (например, в виде определенной аналитической зависимости) приближать $y(x_i)$, не обязательно проходя через точки $(x_i, y_i)$. Такова постановка задачи *регрессии* (*см. разд. 13.2*), которую во многих случаях также можно назвать *сглаживанием* данных. Вообще говоря, сглаживание является частным случаем *фильтрации* данных (*см. главу 14*), основанной на уменьшении шумовой компоненты измерений.

Различные виды построения аппроксимирующей зависимости $f(x)$ иллюстрирует рис. 13.1. На нем исходные данные обозначены кружками, интерполяция отрезками прямых линий — пунктиром, линейная регрессия — наклонной прямой линией, а фильтрация — жирной гладкой кривой. Эти зависимости приведены в качестве примера и отражают лишь малую часть возможностей Mathcad по обработке данных. Вообще говоря, в Mathcad имеется целый арсенал встроенных функций, позволяющий осуществлять самую различную регрессию, интерполяцию и экстраполяцию.

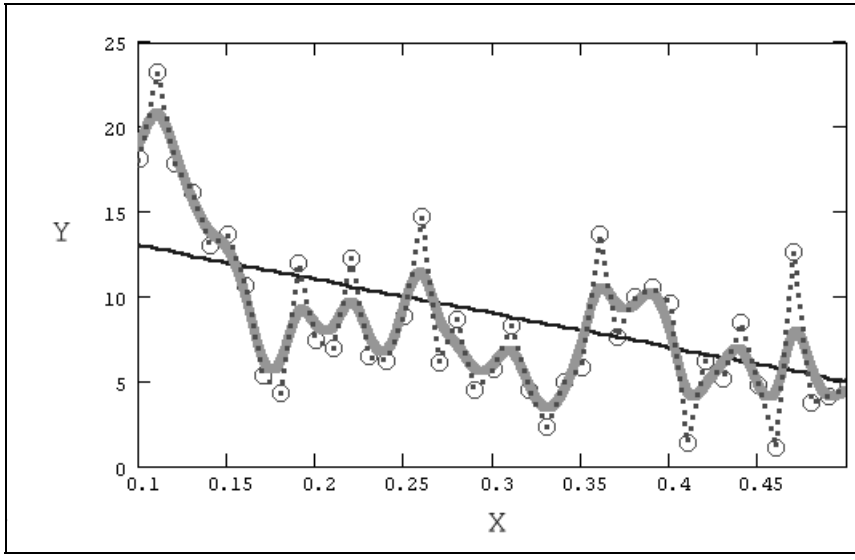



**Рис. 13.1.** Разные задачи аппроксимации данных

# 13.1. Интерполяция

Для построения интерполяции-экстраполяции в Mathcad имеется несколько встроенных функций, позволяющих "соединить" точки выборки данных $(x_i, y_i)$ кривой разной степени гладкости. По определению интерполяция означает построение функции $A(x)$, аппроксимирующей зависимость $y(x)$ в промежуточных точках (между $x_i$). Поэтому интерполяцию еще по-другому называют *аппроксимацией*. В точках $x_i$ значения интерполяционной функции должны совпадать с исходными данными, т. е. $A(x_i)=y(x_i)$.

***Примечание***

Везде в этом разделе, при рассказе о различных типах интерполяции будем использовать вместо обозначения $A(x)$ другое имя ее аргумента $A(t)$, чтобы не путать вектор данных $x$ и скалярную переменную $t$.

## 13.1.1. Линейная интерполяция

Самый простой вид интерполяции — линейная, которая представляет искомую зависимость $A(x)$ в виде ломаной линии. Интерполирующая функция $A(x)$ состоит из отрезков прямых, соединяющих точки (рис. 13.2).

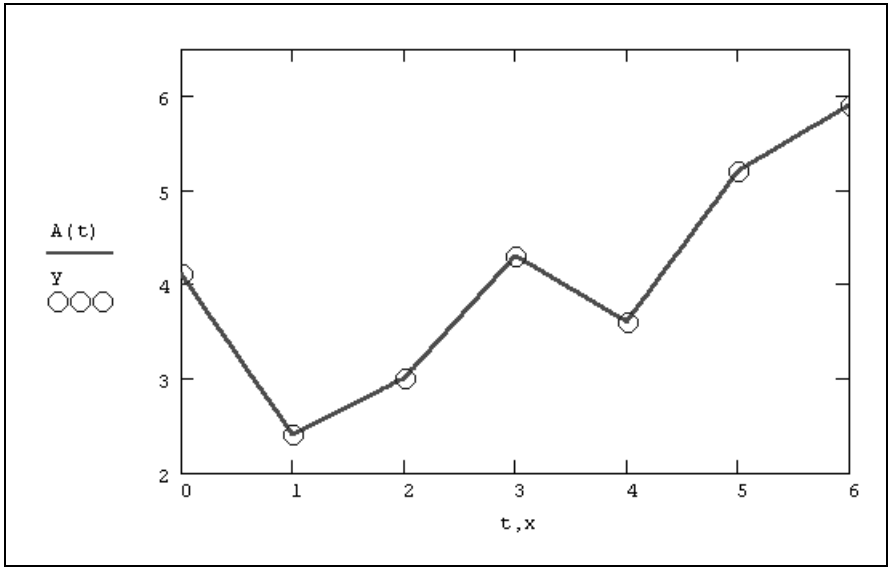



**Рис. 13.2.** Линейная интерполяция (продолжение листинга 13.1)

Для построения линейной интерполяции служит встроенная функция `linterp` (листинг 13.1):

- `linterp(x,y,t)` — функция, аппроксимирующая данные векторов `x` и `y` кусочно-линейной зависимостью:
  - `x` — вектор действительных данных аргумента;
  - `y` — вектор действительных данных значений того же размера;
  - `t` — значение аргумента, при котором вычисляется интерполирующая функция.

  ***Внимание!***

  Элементы вектора `x` должны быть определены в порядке возрастания, т. е. $x_1<x_2<x_3<\ldots<x_N$.

**Листинг 13.1. Линейная интерполяция**

```
x := ( 0  1  2  3  4  5  6 )^T

y := ( 4.1  2.4  3  4.3  3.6  5.2  5.9 )^T

A(t) := linterp(x, y, t)
```

Как видно из листинга, чтобы осуществить линейную интерполяцию, надо выполнить следующие действия:

1. Ввести векторы данных `x` и `y` (первые две строки листинга).
2. Определить функцию `linterp(x,y,t)`.
3. Вычислить значения этой функции в требуемых точках, например, `linterp(x,y,2.4)=3.52`, или `linterp(x,y,6)=5.9`, или постройте ее график, как показано на рис. 13.3.

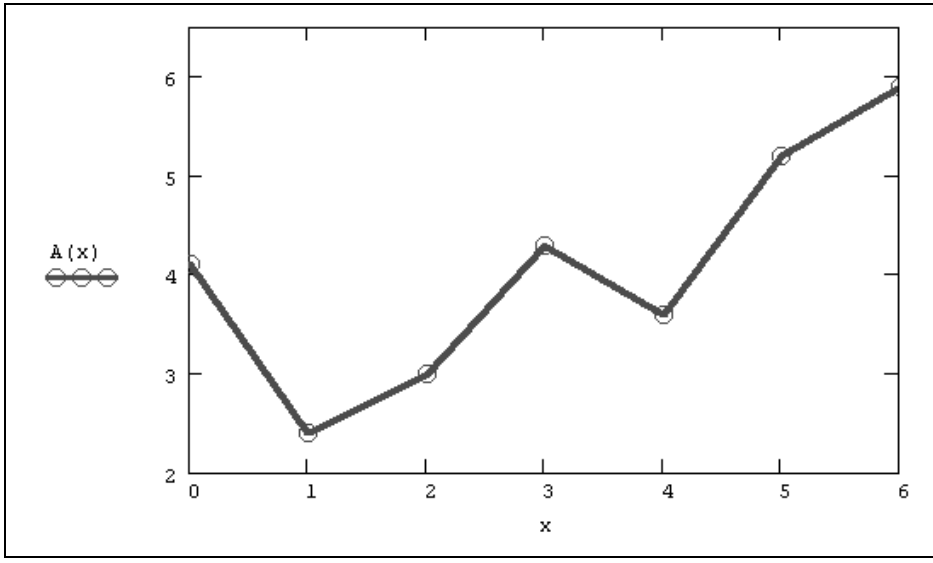



**Рис. 13.3.** Обычное построение графика функции от векторной переменной `x` (продолжение листинга 13.1)

### *Примечание*

Обратите внимание, что функция `A(t)` на графике имеет аргумент `t`, а не `x`. Это означает, что функция `A(t)` вычисляется не только при значениях аргумента (т. е. в семи точках), а при гораздо большем числе аргументов в интервале `(0,6)`, что автоматически обеспечивает Mathcad. Просто в данном случае эти различия незаметны, т. к. при обычном построении графика функции `A(x)` от векторного аргумента `x` (рис. 13.3) Mathcad по умолчанию соединяет точки графика прямыми линиями (т. е. скрытым образом осуществляет их линейную интерполяцию).

## 13.1.2. Кубическая сплайн-интерполяция

В большинстве практических приложений желательно соединить экспериментальные точки не ломаной линией, а гладкой кривой. Лучше всего для этих целей подходит интерполяция кубическими сплайнами, т. е. отрезками кубических парабол (рис. 13.4):

- `interp(s,x,y,t)` — функция, аппроксимирующая данные векторов `x` и `y` кубическими сплайнами:
  - `s` — вектор вторых производных, созданный одной из сопутствующих функций `cspline`, `pspline` или `lspline`;
  - `x` — вектор действительных данных аргумента, элементы которого расположены в порядке возрастания;
  - `y` — вектор действительных данных значений того же размера;
  - `t` — значение аргумента, при котором вычисляется интерполирующая функция.

Сплайн-интерполяция в Mathcad реализована чуть сложнее линейной. Перед применением функции `interp` необходимо предварительно определить первый из ее аргументов — векторную переменную `s`. Делается это при помощи одной из трех встроенных функций тех же аргументов `(x,y)`:

- `lspline(x,y)` — вектор значений коэффициентов линейного сплайна;
- `pspline(x,y)` — вектор значений коэффициентов квадратичного сплайна;
- `cspline(x,y)` — вектор значений коэффициентов кубического сплайна:
  - `x,y` — векторы данных.

Выбор конкретной функции сплайновых коэффициентов влияет на интерполяцию вблизи конечных точек интервала. Пример сплайн-интерполяции приведен в листинге 13.2.

**Листинг 13.2. Кубическая сплайн-интерполяция**

```
x := (0  1  2  3  4  5  6)^T

y := (4.1  2.4  3  4.3  3.6  5.2  5.9)^T

s := cspline(x, y)

A(t) := interp(s, x, y, t)
```

Смысл сплайн-интерполяции заключается в том, что в промежутках между точками осуществляется аппроксимация в виде зависимости $A(t)=a\cdot t^3+b\cdot t^2+c\cdot t+d$. Коэффициенты a, b, c, d рассчитываются независимо для каждого промежутка, исходя из значений $y_i$ в соседних точках. Этот процесс скрыт от пользователя, поскольку смысл задачи интерполяции состоит в выдаче значения A(t) в любой точке t (рис. 13.4).

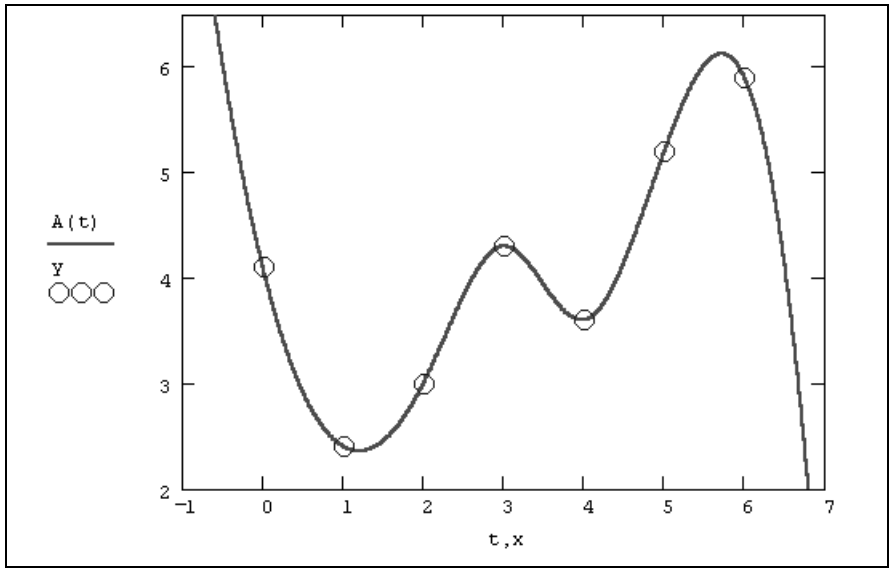



**Рис. 13.4.** Сплайн-интерполяция
(продолжение листинга 13.2)

Чтобы подчеркнуть различия, соответствующие разным вспомогательным функциям cspline, pspline, lspline, покажем результат действия листинга 13.2 при замене функции cspline в предпоследней строке на линейную lspline (рис. 13.5). Как видно, выбор вспомогательных функций существенно влияет на поведение A(t) вблизи граничных точек рассматриваемого интервала (0,6) и особенно разительно меняет результат экстраполяции данных за его пределами.

В заключение остановимся на уже упоминавшейся в предыдущем разделе распространенной ошибке при построении графиков интерполирующей функции (см. рис. 13.3). Если на графике, например, являющемся продолжением листинга 13.2, задать построение функции A(x) вместо A(t), то будет получено просто соединение исходных точек ломаной (рис. 13.6).

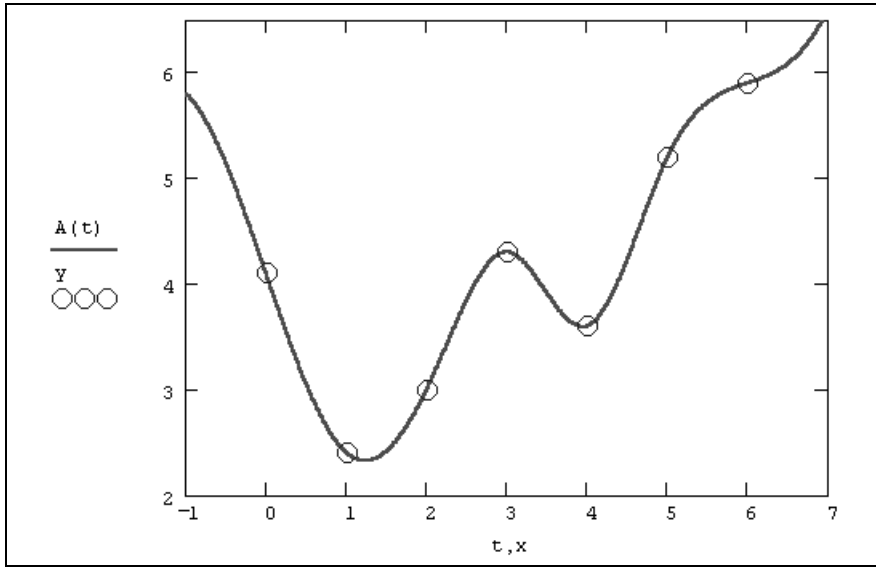



**Рис. 13.5.** Сплайн-интерполяция
с выбором коэффициентов линейного сплайна `lspline`

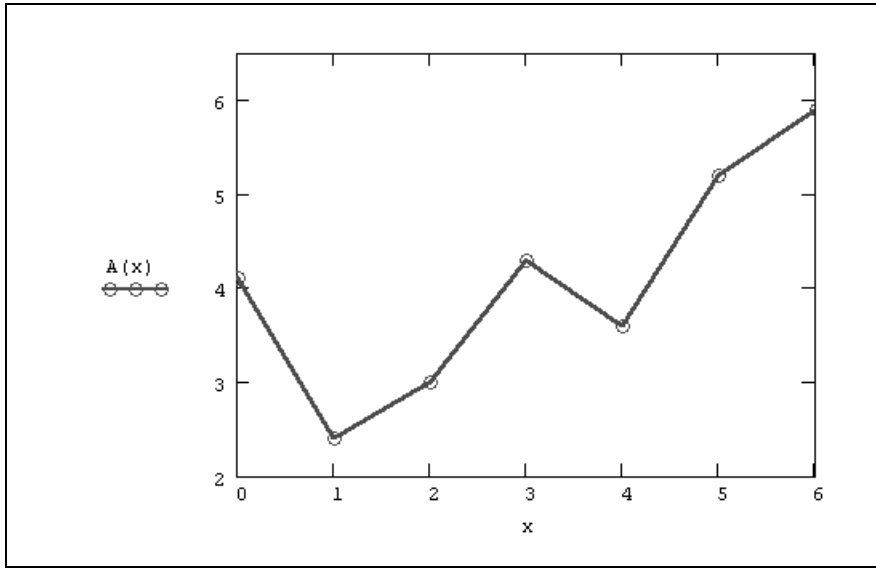



**Рис. 13.6.** Ошибочное построение графика сплайн-интерполяции
(продолжение листинга 13.2)

Так происходит потому, что в промежутках между точками вычисления интерполирующей функции не производятся.

## 13.1.3. Полиномиальная сплайн-интерполяция

Более сложный тип интерполяции — так называемая *интерполяция B-сплайнами*. В отличие от обычной сплайн-интерполяции (*см. разд. 13.1.2*), сшивка элементарных B-сплайнов производится не в точках $x_i$, а в других точках $u_i$, координаты которых предлагается ввести пользователю. Сплайны могут быть полиномами 1, 2 или 3 степени (линейные, квадратичные или кубические). Применяется интерполяция B-сплайнами точно так же, как и обычная сплайн-интерполяция, различие состоит только в определении вспомогательной функции коэффициентов сплайна.

- `interp(s,x,y,t)` — функция, аппроксимирующая данные векторов `x` и `y` с помощью B-сплайнов.
- `bspline(x,y,u,n)` — вектор значений коэффициентов B-сплайна:
  - `s` — вектор вторых производных, созданный функцией `bspline`;
  - `x` — вектор действительных данных аргумента, элементы которого расположены в порядке возрастания;
  - `y` — вектор действительных данных значений того же размера;
  - `t` — значение аргумента, при котором вычисляется интерполирующая функция;
  - `u` — вектор значений аргумента, в которых производится сшивка B-сплайнов;
  - `n` — порядок полиномов сплайновой интерполяции (`1`, `2` или `3`).

  ***Примечание***

  Размерность вектора `u` должна быть на `1`, `2` или `3` меньше размерности векторов `x` и `y`. Первый элемент вектора `u` должен быть меньше или равен первому элементу вектора `x`, а последний элемент `u` — больше или равен последнему элементу `x`.

Интерполяция B-сплайнами иллюстрируется листингом 13.3 и рис. 13.7.

**Листинг 13.3. Интерполяция B-сплайнами**

```
x := ( 0  1  2  3  4  5  6 )^T

y := ( 4.1  2.4  3  4.3  3.6  5.2  5.9 )^T
```

```
u := (-0.5  2.2  3.3  4.1  5.5  7)^T
s := bspline(x, y, u, 2)
A(t) := interp(s, x, y, t)
```

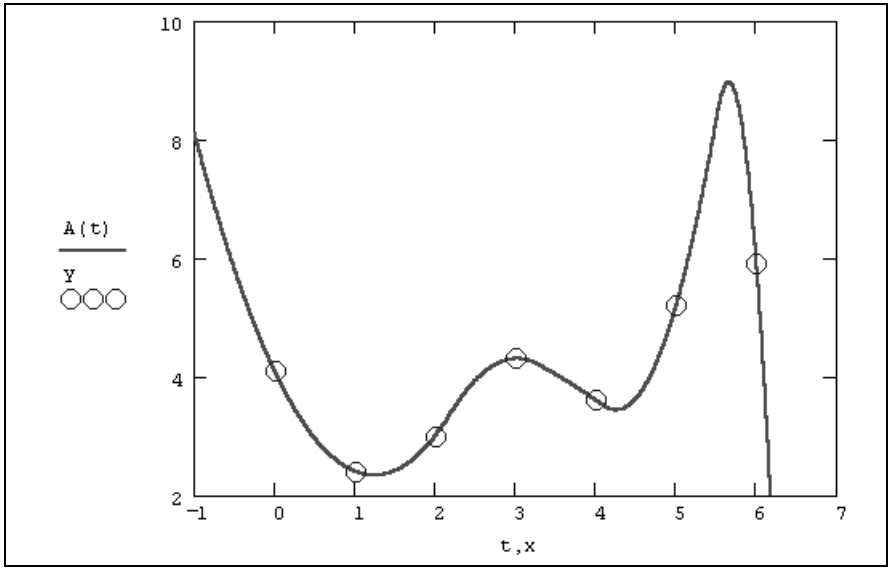



**Рис. 13.7.** B-сплайн-интерполяция
(продолжение листинга 13.3)

## 13.1.4. Сплайн-экстраполяция

Все описанные в предыдущих разделах типы интерполяции работают также и как функции экстраполяции данных. Для вычисления экстраполяции достаточно просто указать соответствующее значение аргумента, которое лежит за границами рассматриваемого интервала. С этой точки зрения разницы в применении в Mathcad между интерполяцией и экстраполяцией нет.

На практике при построении экстраполяции следует соблюдать известную осторожность, не забывая о том, что ее успех определяется значимостью ближайших к границе интервала точек. Чем дальше от них вы будете пытаться экстраполировать зависимость, заданную экспериментальными точками, тем сомнительнее будет результат. Сказанное иллюстрируется рис. 13.8 и 13.9, на которых изображена линейная (пунктир на обоих графиках) и сплайн- (сплошные кривые) экстраполяция.

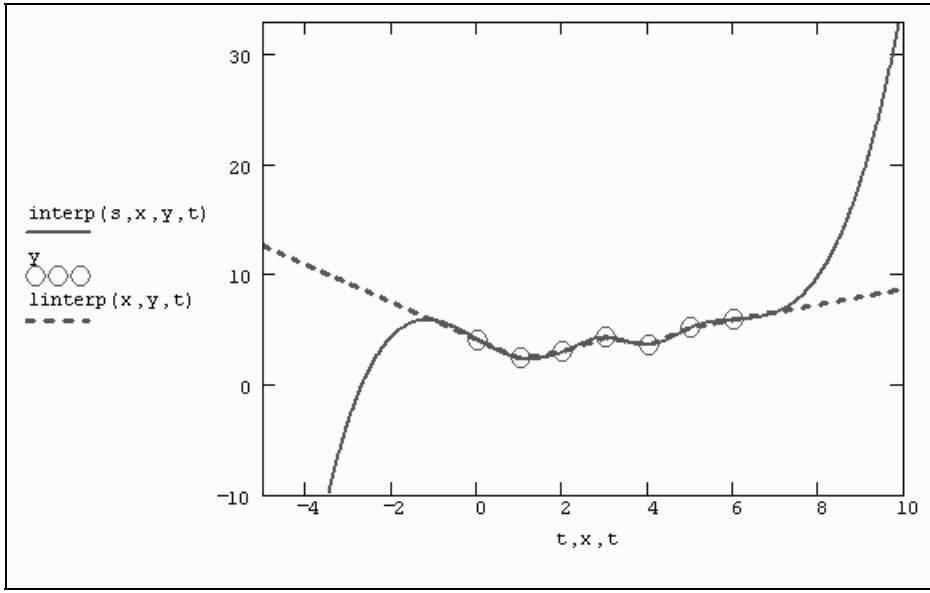



**Рис. 13.8.** Линейная сплайн-экстраполяция

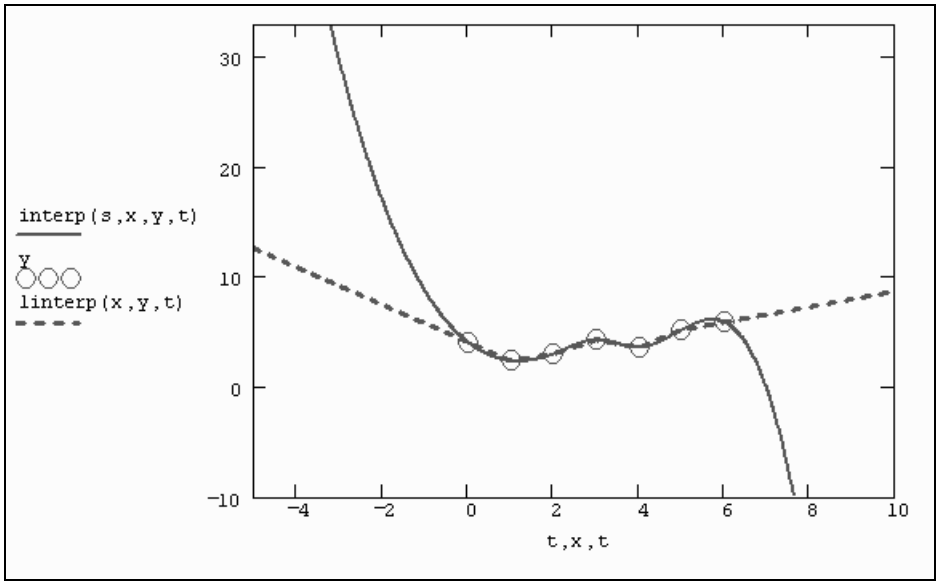



**Рис. 13.9.** Квадратичная сплайн-экстраполяция
(продолжение листинга 13.2)

На рис. 13.8 используется линейная сплайн-экстраполяция при помощи функции `lspline` (см. рис. 13.5 в качестве примера интерполяции), а на рис. 13.9 — функции кубического сплайна `cspline` (что соответствует листингу 13.2 и рис. 13.4). Видно, что вдали от рассматриваемого интервала результаты экстраполяции совершенно различны, что, конечно, объясняется тем, что она является ни чем иным, как параболической зависимостью.

## 13.1.5. Экстраполяция функцией предсказания

Как мы увидели (*см. разд. 13.1.4*), стандартные функции интерполяции-экстраполяции стоит применять только в непосредственной близости границ интервала данных. В Mathcad имеется более развитый инструмент экстраполяции, который учитывает распределение данных вдоль всего интервала. В функцию `predict` встроен линейный алгоритм предсказания поведения функции, основанный на анализе, в том числе осцилляций:

- `predict(y,m,n)` — функция предсказания вектора, экстраполирующего выборку данных:
  - `y` — вектор действительных значений, взятых через равные промежутки значений аргумента;
  - `m` — количество последовательных элементов вектора `y`, согласно которым строится экстраполяция;
  - `n` — количество элементов вектора предсказаний.

Пример использования функции предсказания на примере экстраполяции осциллирующих данных $y_j$ с меняющейся амплитудой приведен в листинге 13.4. Полученный график экстраполяции, наряду с самой функцией, показан на рис. 13.10. Аргументы и принцип действия функции `predict` отличаются от рассмотренных выше встроенных функций интерполяции-экстраполяции. Значений аргумента для данных не требуется, поскольку по определению функция действует на данные, идущие друг за другом с равномерным шагом. Обратите внимание, что результат функции `predict` вставляется "в хвост" исходных данных.

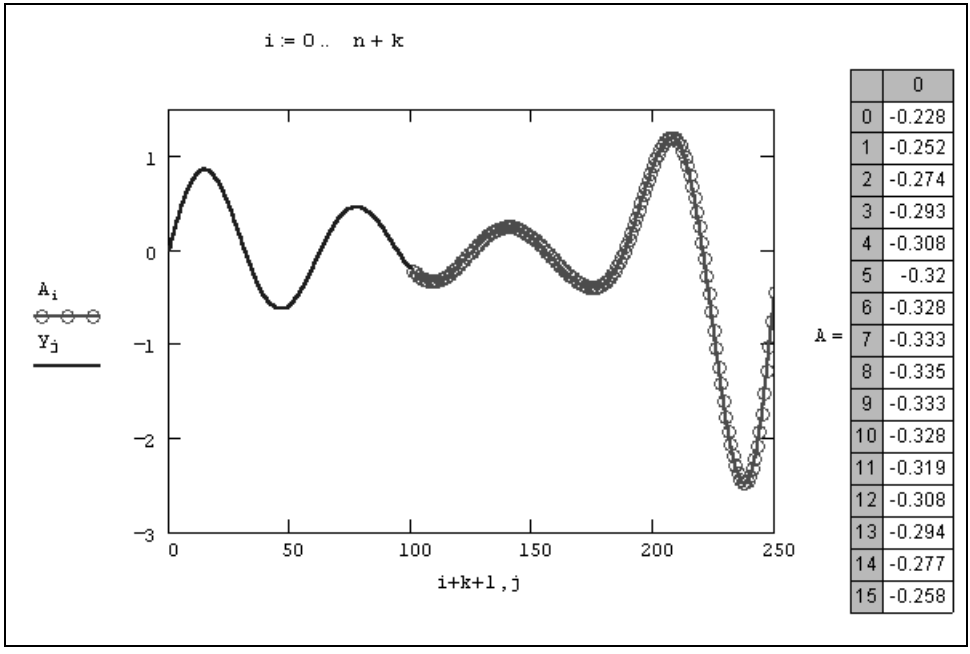
**Листинг 13.4. Экстраполяция при помощи функции предсказания**

$k := 100$ $\qquad$ $j := 0 .. k$

$$y_j := e^{\frac{-j}{100}} \cdot \sin\left(\frac{j}{10}\right)$$

```
m := 50                    n := 150
A := predict(y, m, n)
```

Как видно из рис. 13.11, функция предсказания может быть полезна при экстраполяции данных на небольшие расстояния. Вдали от исходных данных результат часто бывает неудовлетворительным. Кроме того, функция `predict` хорошо работает в задачах анализа подробных данных с четко прослеживающейся закономерностью (типа рис. 13.10), в основном осциллирующего характера.




| | 0 |
|---|---|
| 0 | -0.228 |
| 1 | -0.252 |
| 2 | -0.274 |
| 3 | -0.293 |
| 4 | -0.308 |
| 5 | -0.32 |
| 6 | -0.328 |
| 7 | -0.333 |
| 8 | -0.335 |
| 9 | -0.333 |
| 10 | -0.328 |
| 11 | -0.319 |
| 12 | -0.308 |
| 13 | -0.294 |
| 14 | -0.277 |
| 15 | -0.258 |

**Рис. 13.10.** Экстраполяция при помощи функции предсказания (продолжение листинга 13.4)

Если данных мало, то предсказание может оказаться бесполезным. В листинге 13.5 приведена экстраполяция небольшой выборки данных (из примеров, рассмотренных в предыдущих разделах). Соответствующий результат показан на рис. 13.11 для различных крайних точек массива исходных данных, для которых строится экстраполяция.

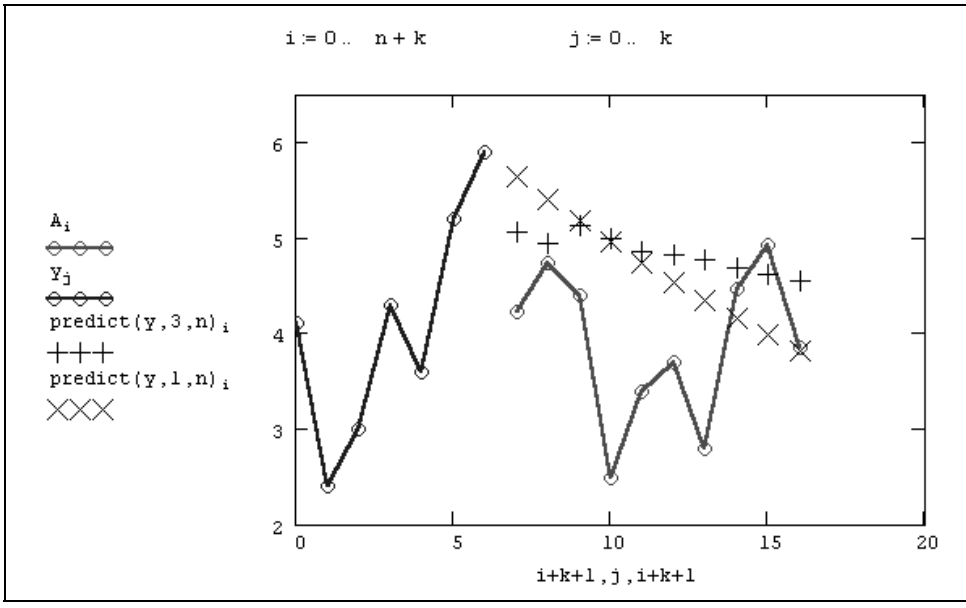



**Рис. 13.11.** Работа функции предсказания в случае малого количества данных (продолжение листинга 13.5)

**Листинг 13.5. Экстраполяция при помощи функции предсказания**

```
y := ( 4.1  2.4  3  4.3  3.6  5.2  5.9 )^T
k := rows(y) - 1
m := 5                                  n := 10
A := predict(y, m, n)
```

## 13.1.6. Многомерная интерполяция

Двумерная сплайн-интерполяция приводит к построению поверхности `z(x,y)`, проходящей через массив точек, описывающий сетку на координатной плоскости `(x,y)`. Поверхность создается участками двумерных кубических сплайнов, являющихся функциями `(x,y)` и имеющих непрерывные первые и вторые производные по обеим координатам.

Многомерная интерполяция строится с помощью тех же встроенных функций, что и одномерная (*см. разд. 13.1.2*), но имеет в качестве аргументов не векторы,

а соответствующие матрицы. Существует одно важное ограничение, связанное с возможностью интерполяции только квадратных N×N массивов данных:

- `interp(S,X,Z,V)` — скалярная функция, аппроксимирующая данные выборки двумерного поля по координатам `x` и `y` кубическими сплайнами:
  - `S` — вектор вторых производных, созданный одной из сопутствующих функций `cspline`, `pspline` или `lspline`;
  - `X` — матрица размерности `N×2`, определяющая диагональ сетки значений аргумента (элементы обоих столбцов соответствуют меткам `x` и `y` и расположены в порядке возрастания);
  - `Z` — матрица действительных данных размерности `N×N`;
  - `V` — вектор из двух элементов, содержащий значения аргументов `x` и `y`, для которых вычисляется интерполяция.

***Примечание***

Вспомогательные функции построения вторых производных имеют те же матричные аргументы, что и `interp`: `lspline(X,Y)`, `pspline(X,Y)`, `cspline(X,Y)`.

Пример исходных данных приведен на рис. 13.12 в виде графика линий уровня, программная реализация двумерной интерполяции показана в листинге 13.6, а ее результат — на рис. 13.13.

**Листинг 13.6. Двумерная интерполяция**

$$X := \begin{pmatrix} 0 & 0 \\ 1 & 10 \\ 2 & 20 \\ 3 & 30 \\ 4 & 40 \end{pmatrix} \qquad Y := \begin{pmatrix} 1 & 1 & 0 & 1.1 & 1.2 \\ 1 & 2 & 3 & 2.1 & 1.5 \\ 1.3 & 3.3 & 5 & 1.7 & 2 \\ 1.3 & 3 & 3.7 & 2.1 & 2.9 \\ 1.5 & 2 & 2.5 & 2.8 & 4 \end{pmatrix}$$

$$S := \mathrm{cspline}(X, Y)$$

$$V := \begin{pmatrix} 3.7 \\ 2.2 \end{pmatrix} \qquad \mathrm{interp}(S, X, Y, V) = 1.636$$

$$k := 30$$

$$i := 0 .. k \qquad j := 0 .. k$$

$$A_{i,j} := \mathrm{interp}\left[ S, X, Y, \begin{pmatrix} \frac{i}{k} \cdot 4 \\ \frac{j}{k} \cdot 40 \end{pmatrix} \right]$$

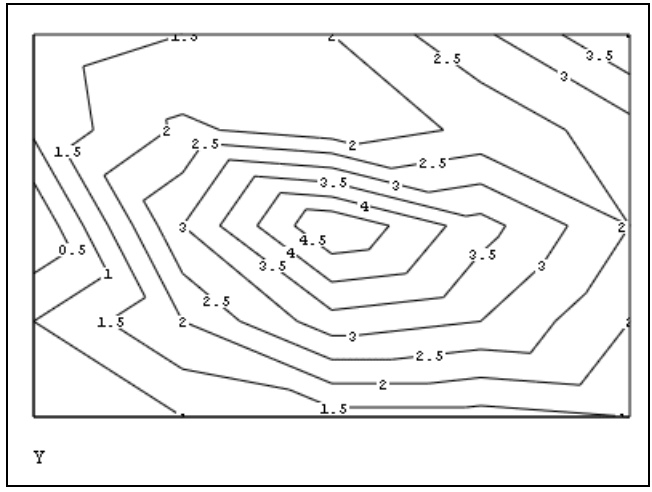



**Рис. 13.12.** Исходное двумерное поле данных
(продолжение листинга 13.6)

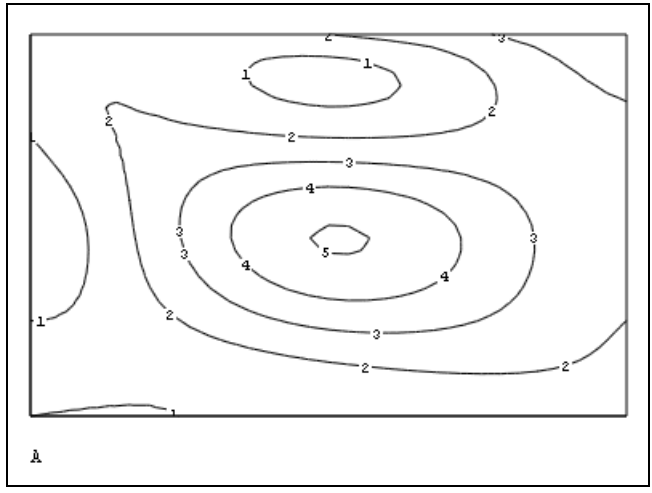



**Рис. 13.13.** Результат двумерной интерполяции
(продолжение листинга 13.6)

# 13.2. Регрессия

Задачи математической регрессии имеют смысл приближения выборки данных `(xi,yi)` некоторой функцией `f(x)`, определенным образом минимизи-

рующей совокупность ошибок $|f(x_i)-y_i|$. Регрессия сводится к подбору неизвестных коэффициентов, определяющих аналитическую зависимость $f(x)$. В силу производимого действия большинство задач регрессии являются частным случаем более общей проблемы сглаживания данных.

Как правило, регрессия очень эффективна, когда заранее известен (или, по крайней мере, хорошо угадывается) закон распределения данных $(x_i, y_i)$.

## 13.2.1. Линейная регрессия

Самый простой и наиболее часто используемый вид регрессии — линейная. Приближение данных $(x_i, y_i)$ осуществляется линейной функцией $y(x)=b+a\cdot x$. На координатной плоскости $(x, y)$ линейная функция, как известно, представляется прямой линией (рис. 13.14). Еще линейную регрессию часто называют *методом наименьших квадратов*, поскольку коэффициенты $a$ и $b$ вычисляются из условия минимизации суммы квадратов ошибок $|b+a\cdot x_i-y_i|$.

> ***Примечание 1***
>
> Чаще всего такое же условие ставится и в других задачах регрессии, т. е. приближения массива данных $(x_i, y_i)$ другими зависимостями $y(x)$. Исключение рассмотрено в листинге 13.9.

> ***Примечание 2***
>
> Различным расчетным аспектам реализации метода наименьших квадратов, в большинстве случаев сводящимся к решению систем алгебраических линейных уравнений, была посвящена значительная часть *главы 8*.

Для расчета линейной регрессии в Mathcad имеются два дублирующих друг друга способа. Правила их применения представлены в листингах 13.7 и 13.8. Результат обоих листингов получается одинаковым (рис. 13.14):

- `line(x,y)` — вектор из двух элементов `(b,a)` коэффициентов линейной регрессии $b+a\cdot x$;
- `intercept(x,y)` — коэффициент `b` линейной регрессии;
- `slope(x,y)` — коэффициент `a` линейной регрессии:
  - `x` — вектор действительных данных аргумента;
  - `y` — вектор действительных данных значений того же размера.

**Листинг 13.7. Линейная регрессия**

$x := (0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6)^T$

$y := (4.1 \quad 2.4 \quad 3 \quad 4.3 \quad 3.6 \quad 5.2 \quad 5.9)^T$

$$line(x, y) = \begin{pmatrix} 2.829 \\ 0.414 \end{pmatrix}$$

$f(t) := line(x, y)_0 + line(x, y)_1 \cdot t$

**Листинг 13.8. Другая форма записи линейной регрессии**

$x := (0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6)^T$

$y := (4.1 \quad 2.4 \quad 3 \quad 4.3 \quad 3.6 \quad 5.2 \quad 5.9)^T$

$intercept(x, y) = 2.829$

$slope(x, y) = 0.414$

$f(t) := intercept(x, y) + slope(x, y) \cdot t$

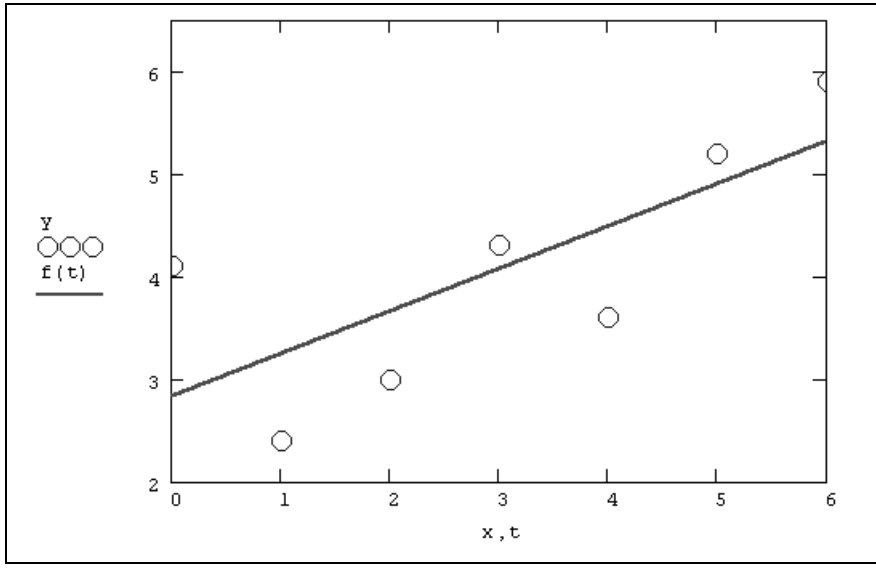



**Рис. 13.14.** Линейная регрессия
(продолжение листинга 13.7 или 13.8)

В Mathcad имеется альтернативный алгоритм, реализующий не минимизацию суммы квадратов ошибок, а медиан-медианную линейную регрессию для расчета коэффициентов a и b (листинг 13.9):

- `medfit(x,y)` — вектор из двух элементов `(b,a)` коэффициентов линейной медиан-медианной регрессии `b+a·x`:
  - `x,y` — векторы действительных данных одинакового размера.

**Листинг 13.9. Построение линейной регрессии двумя разными методами (продолжение листинга 13.7)**

$$\text{medfit}(x,y) = \begin{pmatrix} 2.517 \\ 0.55 \end{pmatrix}$$

$$g(t) := \text{medfit}(x,y)_0 + \text{medfit}(x,y)_1 \cdot t$$

Различие результатов среднеквадратичной и медиан-медианной регрессии иллюстрируется на рис. 13.15.

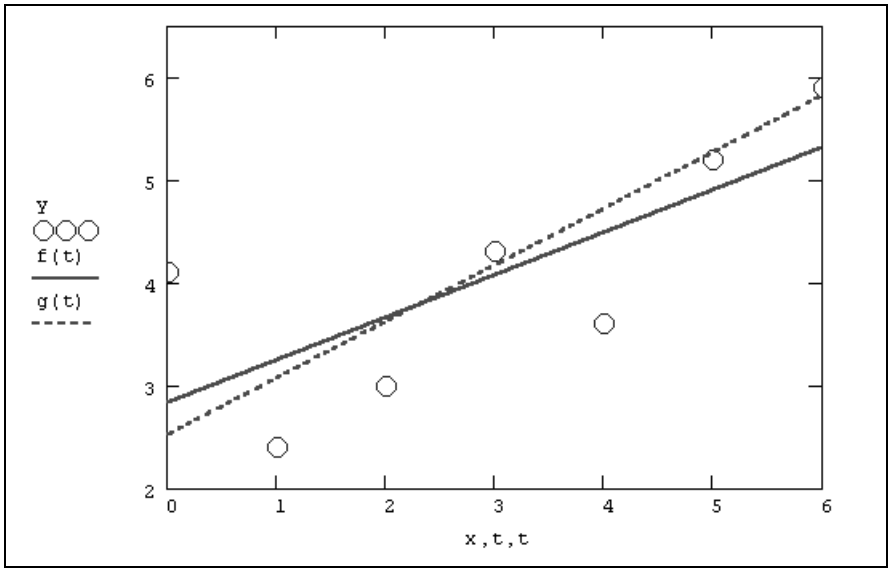



**Рис. 13.15.** Линейная регрессия по методу наименьших квадратов и методу медиан (продолжение листингов 13.7 и 13.9)

## 13.2.2. Полиномиальная регрессия

В Mathcad реализована регрессия одним полиномом, отрезками нескольких полиномов, а также двумерная регрессия массива данных.

### Полиномиальная регрессия

Полиномиальная регрессия означает приближение данных $(x_i, y_i)$ полиномом $k$-й степени $A(x)=a+b\cdot x+c\cdot x^2+d\cdot x^3+\ldots+h\cdot x^k$ (рис. 13.16). При $k=1$ полином является прямой линией, при $k=2$ — параболой, при $k=3$ — кубической параболой и т. д. Как правило, на практике применяются $k<5$.

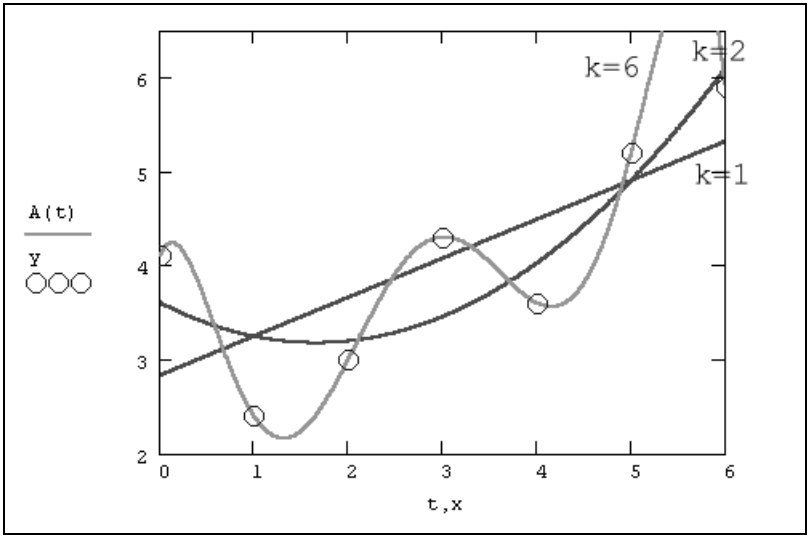



**Рис. 13.16.** Регрессия полиномами разной степени (коллаж результатов листинга 13.10 для разных $k$)

#### *Примечание*

Для построения регрессии полиномом $k$-ой степени необходимо наличие, по крайней мере, $(k+1)$ точек данных.

В Mathcad полиномиальная регрессия осуществляется комбинацией встроенной функции `regress` и полиномиальной интерполяции (*см. разд. 13.1.2*):

- `regress(x,y,k)` — вектор коэффициентов для построения полиномиальной регрессии данных;

- `interp(s,x,y,t)` — результат полиномиальной регрессии:
  - `s=regress(x,y,k)`;
  - `x` — вектор действительных данных аргумента, элементы которого расположены в порядке возрастания;
  - `y` — вектор действительных данных значений того же размера;
  - `k` — степень полинома регрессии (целое положительное число);
  - `t` — значение аргумента полинома регрессии.

> ***Внимание!***
>
> Для построения полиномиальной регрессии после функции `regress` вы обязаны использовать функцию `interp`.

Пример полиномиальной регрессии квадратичной параболой приведен в листинге 13.10.

**Листинг 13.10. Полиномиальная регрессия**

```
x := (0  1  2  3  4  5  6)^T

y := (4.1  2.4  3  4.3  3.6  5.2  5.9)^T

k := 2

s := regress(x, y, k)

A(t) := interp(s, x, y, t)
```

## Регрессия отрезками полиномов

Помимо приближения массива данных одним полиномом имеется возможность осуществить регрессию сшивкой отрезков (точнее говоря, участков, т. к. они имеют криволинейную форму) нескольких полиномов. Для этого имеется встроенная функция `loess`, применение которой аналогично функции `regress` (листинг 13.11 и рис. 13.17):

- `loess(x,y,span)` — вектор коэффициентов для построения регрессии данных отрезками полиномов;
- `interp(s,x,y,t)` — результат полиномиальной регрессии:
  - `s=loess(x,y,span)`;
  - `x` — вектор действительных данных аргумента, элементы которого расположены в порядке возрастания;

- `y` — вектор действительных данных значений того же размера;
- `span` — параметр, определяющий размер отрезков полиномов (положительное число, хорошие результаты дает значение порядка `span=0.75`).

Параметр `span` задает степень сглаженности данных. При больших значениях `span` регрессия практически не отличается от регрессии одним полиномом (например, `span=2` дает почти тот же результат, что и приближение точек параболой).

**Листинг 13.11. Регрессия отрезками полиномов**

```
x := (0  1  2  3  4  5  6)^T

y := (4.1  2.4  3  4.3  3.6  5.2  5.9)^T

s := loess(x, y, 0.9)

A(t) := interp(s, x, y, t)
```

### *Совет*

Регрессия одним полиномом эффективна, когда множество точек выглядит как полином, а регрессия отрезками полиномов оказывается полезной в противоположном случае.

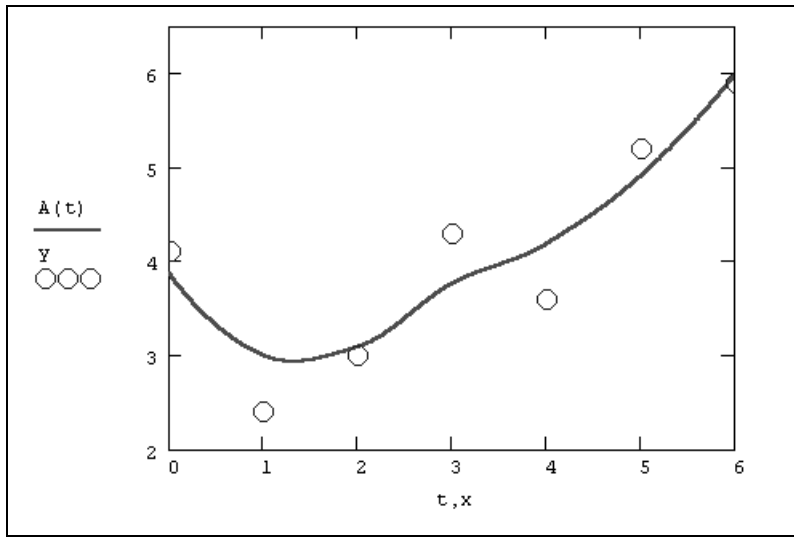



**Рис. 13.17.** Регрессия отрезками полиномов (продолжение листинга 13.11)

## Двумерная полиномиальная регрессия

По аналогии с одномерной полиномиальной регрессией и двумерной интерполяцией (*см. разд. 13.1.5*), Mathcad позволяет приблизить множество точек $z_{i,j}(x_i, y_j)$ поверхностью, которая определяется многомерной полиномиальной зависимостью. В качестве аргументов встроенных функций для построения полиномиальной регрессии должны стоять в этом случае не векторы, а соответствующие матрицы.

- `regress(X,Z,k)` — вектор коэффициентов для построения полиномиальной регрессии данных.
- `loess(X,Z,span)` — вектор коэффициентов для построения регрессии данных отрезками полиномов.
- `interp(S,X,Z,V)` — скалярная функция, аппроксимирующая данные выборки двумерного поля по координатам `x` и `y` кубическими сплайнами:
  - `S` — вектор вторых производных, созданный одной из сопутствующих функций `loess` или `regress`;
  - `X` — матрица размерности `N×2`, определяющая пары значений аргумента (столбцы соответствуют меткам `x` и `y`);
  - `Z` — вектор действительных данных размерности `N`;
  - `span` — параметр, определяющий размер отрезков полиномов;
  - `k` — степень полинома регрессии (целое положительное число);
  - `V` — вектор из двух элементов, содержащий значения аргументов `x` и `y`, для которых вычисляется интерполяция.

> ***Внимание!***
>
> Для построения регрессии не предполагается никакого предварительного упорядочивания данных (как, например, для двумерной интерполяции, которая требует их представления в виде матрицы `N×N`). В связи с этим данные представляются как вектор.

Двумерная полиномиальная регрессия иллюстрируется листингом 13.12 и рис. 13.18. Сравните стиль представления данных для двумерной регрессии с представлением тех же данных для двумерной сплайн-интерполяции (см. листинг 13.6) и ее результаты с исходными данными (см. рис. 13.12) и их сплайн-интерполяцией (см. рис. 13.13).

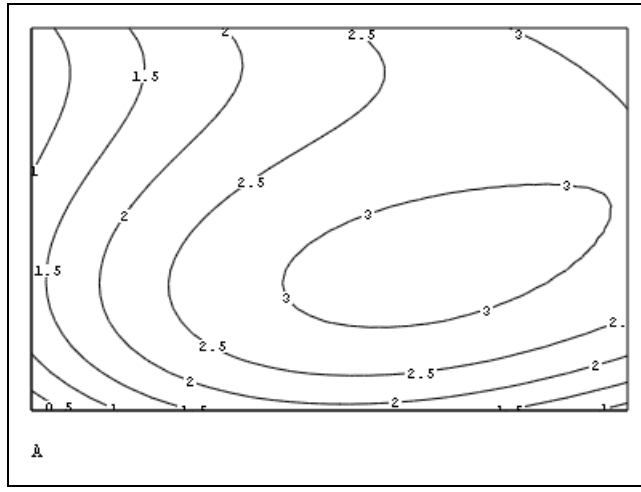



**Рис. 13.18.** Двумерная полиномиальная регрессия (продолжение листинга 13.12)

**Листинг 13.12. Двумерная полиномиальная регрессия**

$$X:=\begin{pmatrix}0&0&0&0&0&1&1&1&1&1&2&2&2&2&2&3&3&3&3&3&4&4&4&4&4\\0&10&20&30&40&0&10&20&30&40&0&10&20&30&40&0&10&20&30&40&0&10&20&30&40\end{pmatrix}^{T}$$

$$Z:=(1\ 1\ 0\ 1.1\ 1.2\ 1\ 2\ 3\ 2.1\ 1.5\ 1.3\ 3.3\ 5\ 1.7\ 2\ 1.3\ 3.7\ 2.1\ 2.9\ 1.5\ 2\ 2.5\ 2.8\ 4)^{T}$$

$$S := \mathrm{regress}(X, Z, 3)$$

$$k := 30$$

$$i := 0 .. k \qquad j := 0 .. k$$

$$A_{i,j} := \mathrm{interp}\left[S, X, Z, \begin{pmatrix}\frac{i}{k}\cdot 4\\ \frac{j}{k}\cdot 40\end{pmatrix}\right]$$

### *Примечание*

Обратите внимание на знаки транспонирования в листинге. Они применены для корректного представления аргументов (например, `Z`, в качестве вектора, а не строки).

## 13.2.3. Другие типы регрессии

Кроме рассмотренных, в Mathcad встроено еще несколько видов трехпараметрической регрессии. Их реализация несколько отличается от приведенных выше вариантов регрессии тем, что для них, помимо массива данных, требуется задать некоторые начальные значения коэффициентов `a`, `b`, `c`. Используйте соответствующий вид регрессии, если хорошо представляете себе, какой зависимостью описывается ваш массив данных. Когда тип регрессии плохо отражает последовательность данных, то ее результат часто бывает неудовлетворительным и даже сильно различающимся в зависимости от выбора начальных значений. Каждая из функций выдает вектор уточненных параметров `a`, `b`, `c`.

- `expfit(x,y,g)` — регрессия экспонентой $f(x)=a\cdot e^{bx}+c$.
- `lgsfit(x,y,g)` — регрессия логистической функцией $f(x)=a/(1+b\cdot e^{-cx})$.
- `sinfit(x,y,g)` — регрессия синусоидой $f(x)=a\cdot\sin(x+b)+c$.
- `pwfit(x,y,g)` — регрессия степенной функцией $f(x)=a\cdot x^{b}+c$.
- `logfit(x,y,g)` — регрессия логарифмической функцией $f(x)=a\cdot\ln(x+b)+c$.
- `lnfit(x,y)` — регрессия двухпараметрической логарифмической функцией $f(x)=a\cdot\ln(x)+b$:
  - `x` — вектор действительных данных аргумента;
  - `y` — вектор действительных значений того же размера;
  - `g` — вектор из трех элементов, задающий начальные значения `a`, `b`, `c`.

> ***Примечание***
>
> Правильность выбора начальных значений можно оценить по результату регрессии — если функция, выданная Mathcad, хорошо приближает зависимость `y(x)`, значит, они были подобраны удачно.

Пример расчета одного из видов трехпараметрической регрессии (экспоненциальной) приведен в листинге 13.13 и на рис. 13.19. В предпоследней строке листинга выведены в виде вектора вычисленные коэффициенты `a`, `b`, `c`, а в последней строке через эти коэффициенты определена искомая функция `f(x)`.

**Листинг 13.13. Экспоненциальная регрессия**

```
x := (0  1  2  3  4  5  6)^T

y := (4.1  2.4  3  4.3  3.6  5.2  5.9)^T
```

$$g := \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$C := expfit(x, y, g)$

$$C = \begin{pmatrix} 0.111 \\ 0.544 \\ 3.099 \end{pmatrix}$$

$f(t) := C_0 \cdot \exp(C_1 \cdot t) + C_2$

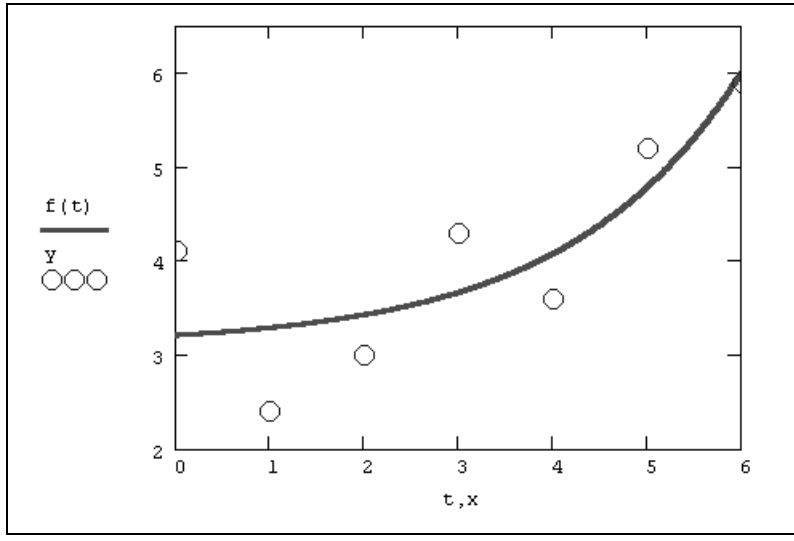



**Рис. 13.19.** Экспоненциальная регрессия
(продолжение листинга 13.13)

### *Примечание*

Многие задачи регрессии данных различными двухпараметрическими зависимостями $y(x)$ можно свести к более надежной, с вычислительной точки зрения, линейной регрессии. Делается это с помощью соответствующей замены переменных.

## 13.2.4. Регрессия общего вида

В Mathcad можно осуществить регрессию в виде линейной комбинации $C_1f_1(x)+C_2f_2(x)+...$, где $f_i(x)$ — любые функции пользователя, а $C_i$ — подле-

жащие определению коэффициенты. Кроме того, имеется путь проведения регрессии более общего вида, когда комбинацию функций и искомых коэффициентов задает сам пользователь.

Приведем встроенные функции для регрессии общего вида и примеры их использования (листинги 13.14 и 13.15), надеясь, что читатель при необходимости найдет более подробную информацию об этих специальных возможностях в справочной системе и ресурсах Mathcad.

- `linfit(x,y,F)` — вектор параметров линейной комбинации функций пользователя, осуществляющей регрессию данных.
- `genfit(x,y,G,g)` — вектор параметров, реализующих регрессию данных с помощью функций пользователя общего вида:
  - `x` — вектор действительных данных аргумента, элементы которого расположены в порядке возрастания;
  - `y` — вектор действительных значений того же размера;
  - `F(x)` — пользовательская векторная функция скалярного аргумента;
  - `G` — вектор начальных значений параметров регрессии размерности `N`;
  - `g(x,C)` — функция пользователя, на основе которой строится регрессия. Если вы пользуетесь Mathcad ранних версий (вплоть до 12-й включительно), то вид `g(x,C)` немного более сложный. Она должна быть не скалярной, а векторной функцией размерности `N+1`, составленной из функции пользователя и ее `N` частных производных по каждому из параметров `C` (листинг 13.16).

### ⑫ ⑬ *Примечание*

Число данных (количество элементов в векторах `x` и `y`) должно быть не меньше, чем `N`. Это менее жесткое требование появилось в Mathcad 12, а до этого для функции регрессии общего вида `genfit` было необходимо задание не менее `N+1` данных.

### ⑬ *Примечание*

Как вы можете заметить, сравнивая листинги 13.15 и 13.16, начиная с Mathcad 13, для функции регрессии общего вида `genfit` достаточно определить (в качестве последнего аргумента) только саму функцию регрессии, не тратя время на ввод ее производных.

## Листинг 13.14. Регрессия линейной комбинацией функций пользователя

$$x := (0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6)^T$$

$$y := (4.1 \quad 2.4 \quad 3 \quad 4.3 \quad 3.6 \quad 5.2 \quad 5.9)^T$$

$$F(x) := \begin{pmatrix} \frac{1}{x+1} \\ x \\ e^x \end{pmatrix}$$

$$C := \text{linfit}(x, y, F)$$

$$C = \begin{pmatrix} 3.957 \\ 0.854 \\ 5.605 \times 10^{-4} \end{pmatrix}$$

$$f(x) := \frac{C_0}{x+1} + C_1 \cdot x + C_2 \cdot e^x$$

## 13 Листинг 13.15. Регрессия общего вида

$$x := (0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6)^T$$

$$y := (4.1 \quad 2.4 \quad 3 \quad 4.3 \quad 3.6 \quad 5.2 \quad 5.9)^T$$

$$f(x, C) := C_0 \cdot e^{C_1 \cdot x}$$

$$C0 := \begin{pmatrix} .1 \\ .2 \end{pmatrix}$$

$$C := \text{genfit}(x, y, C0, f)$$

$$C = \begin{pmatrix} 2.817 \\ 0.113 \end{pmatrix}$$

## Листинг 13.16. Регрессия общего вида (в версиях Mathcad 2001—12)

$$x := (0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6)^T$$

$$y := (4.1 \quad 2.4 \quad 3 \quad 4.3 \quad 3.6 \quad 5.2 \quad 5.9)^T$$

$$F(x, C) := \begin{pmatrix} C_0 \cdot e^{C_1 \cdot x} \\ e^{C_1 \cdot x} \\ C_0 \cdot x \cdot e^{C_1 \cdot x} \end{pmatrix} \qquad C0 := \begin{pmatrix} .1 \\ .2 \end{pmatrix}$$

$C := genfit(x, y, C0, F)$

$$C = \begin{pmatrix} 2.817 \\ 0.113 \end{pmatrix}$$

# 13.3. Ввод/вывод данных

Завершим главу, посвященную интерполяции и регрессии, реализацией в Mathcad функции ввода/вывода во внешние файлы, поскольку, как правило, анализ данных чаще всего связан с их импортом из внешних источников (например, результатов эксперимента из текстовых или графических файлов) и экспортом на внешние носители. В большинстве случаев ввод внешних данных в документы Mathcad применяется чаще вывода, поскольку Mathcad имеет гораздо лучшие возможности представления результатов расчетов, чем многие пользовательские программы. Для общения с внешними файлами данных в Mathcad имеет несколько разных способов, причем, начиная с 12-ой версии добавлены новые, намного более удобные опции импорта.

## 13.3.1. Ввод/вывод в текстовые файлы

Перечислим встроенные функции для работы с текстовыми файлами, которые имеются в Mathcad 2001—13.

> **(12) *Примечание***
>
> В Mathcad 12 и выше имеется дополнительная универсальная встроенная функция `READFILE`, значительно облегчающая процесс импорта данных.

- `READPRN("file")` — чтение данных в матрицу из текстового файла.
- `WRITEPRN("file")` — запись данных в текстовый файл.
- `APPENDPRN("file")` — дозапись данных в существующий текстовый файл:
  - `file` — путь к файлу.

***Примечание***

Можно задавать как полный путь к файлу, например, C:\Мои документы, так и относительный, имея в виду, что он будет отсчитываться от папки, в которой находится файл с документом Mathcad. Если вы задаете в качестве аргумента просто имя файла (как в листингах 13.16—13.17), то файл будет записан или прочитан из той папки, в которой находится сам документ Mathcad.

Примеры использования встроенных функций иллюстрируются листингами 13.17—13.18. Результат их действия можно понять, просмотрев получающиеся текстовые файлы, например, с помощью Блокнота Windows (рис. 13.20 и 13.21 соответственно).

**Листинг 13.17. Запись матрицы в текстовый файл и чтение данных из него**

$$Z := \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 99 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

```
WRITEPRN("datafile.prn") := Z
C := READPRN("datafile.prn")
```

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 99 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

**Листинг 13.18. Дозапись вектора `k` в существующий текстовый файл**

```
k := (1  2  3  4  5)
APPENDPRN("datafile.prn") := k
```

Обратите внимание, что, если вы выводите данные в файл, пользуясь встроенной функцией `WRITEPRN`, то в любом случае создается новый текстовый файл. Если даже до записи данных файл с таким именем существовал, то его содержимое будет уничтожено, заменившись новыми данными. Если вы хотите сохранить прежнее содержимое текстового файла с данными, пользуй-

тесь функцией APPENDPRN. Эта встроенная функция может применяться и для создания нового файла. Иными словами, если файла с заданным именем не существовало, то он будет создан и наполнен теми данными, которые вами определены в документе.

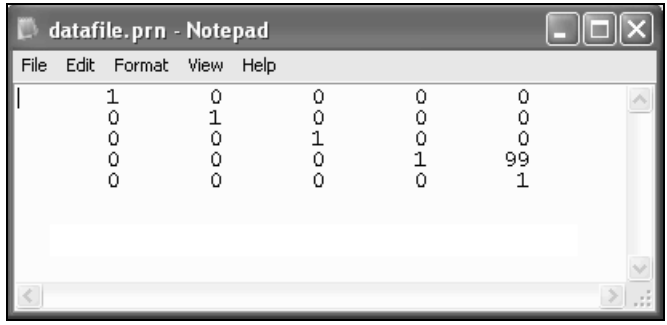



**Рис. 13.20.** Файл, созданный листингом 13.17

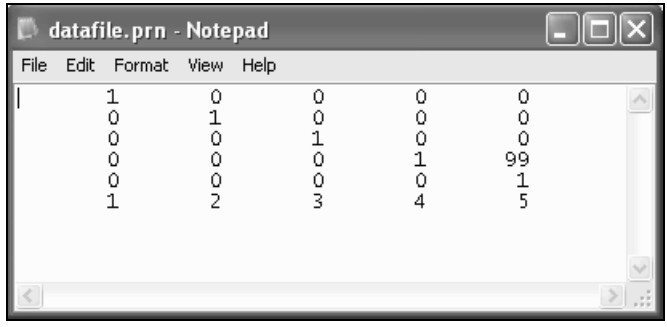



**Рис. 13.21.** Файл, созданный листингами 13.17 и 13.18

### *Примечание*

Создание нового файла путем использования функции APPENDPRN добавлено разработчиками только начиная с версии Mathcad 11. В прежних версиях программы попытка добавить данные к несуществующему файлу при помощи этой функции вызовет сообщение об ошибке.

## 13.3.2. Ввод/вывод в файлы других типов

Подобно вводу/выводу в текстовые файлы можно организовать чтение и запись данных в графические звуковые и файлы.

## Графические файлы

При записи и чтении числовой информации в файлы различных графических форматов данные отождествляются с интенсивностью того или иного цвета пиксела изображения, находящегося в файле. Перечислим основные встроенные функции, предназначенные для графического ввода/вывода:

- `READRGB("file")` — чтение цветного изображения;
- `READBMP("file")` — чтение изображения в оттенках серого;
- `WRITERGB("file")` — запись цветного изображения;
- `WRITEBMP("file")` — запись изображения в оттенках серого:
  - `file` — путь к файлу.

> ***Примечание***
>
> Имеется также большое количество функций специального доступа к графическим файлам, например, чтение интенсивности цветов в других цветовых моделях (яркость-насыщенность-оттенок), а также чтение только одного из основных цветов и т. п. Вы без труда найдете информацию об этих функциях в справочной системе Mathcad, а их применение полностью эквивалентно описанным встроенным функциям.

Действие функций доступа к графическим файлам иллюстрируется листингами 13.19—13.21. Заметим, что для создания изображения используется встроенная функция `identity`, создающая единичную матрицу. Изображение, созданное листингом 13.19, приведено на рис. 13.22.

**Листинг 13.19. Запись матрицы `I` в графический файл**

```
I := identity(100) · 100
I3,9 := 500
WRITEBMP("data.bmp") := I
```

**Листинг 13.20. Чтение из графического файла**

```
C := READBMP("data.bmp")
```

**Листинг 13.21. Запись в цветной графический файл**

```
R := identity(100) · 100
G := identity(100)
```

```
B := identity(100)

WRITERGB("color.bmp") := augment(R, G, B)
```

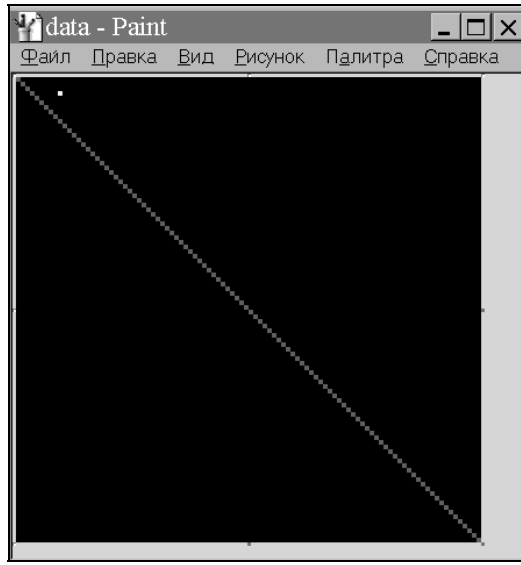



**Рис. 13.22.** Файл, созданный листингом 13.19

## Звуковые файлы

Начиная с версии Mathcad 2001, появилась возможность записывать и считывать амплитуду акустических сигналов в звуковые файлы с расширением wav.

- `READWAV("file")` — чтение звукового файла в матрицу.
- `WRITEWAV("file",s,b)` — запись данных в звуковой файл.
- `GETWAVINFO("file")` — создает вектор из четырех элементов с информацией о звуковом файле:
  - `file` — путь к файлу;
  - `s` — скорость следования сэмплов, задаваемых матрицей;
  - `b` — разрешение звука в битах.

Использование этих встроенных функций позволяет организовать обработку звука.

## Анимация

Во многих случаях самый зрелищный способ представления результатов математических расчетов — это анимация. В Mathcad очень просто создавать анимационные ролики и сохранять их в видеофайлах.

Основной принцип анимации в Mathcad — покадровая анимация. Ролик анимации — это просто последовательность кадров, представляющих собой некоторый участок документа, который выделяется пользователем. Расчеты производятся обособленно для каждого кадра, причем формулы и графики, которые в нем содержатся, должны быть функцией от номера кадра. Номер кадра задается системной переменной `FRAME`, которая может принимать лишь натуральные значения. По умолчанию, если не включен режим подготовки анимации, `FRAME=0`.

Рассмотрим последовательность действий для создания ролика анимации, например, демонстрирующего перемещение гармонической бегущей волны. При этом каждый момент времени будет задаваться переменной `FRAME`.

1. Введите в документ необходимые выражения и графики, в которых участвует переменная номера кадра `FRAME`. Подготовьте часть документа, которую вы желаете сделать анимацией, таким образом, чтобы она находилась в поле вашего зрения на экране. В нашем примере подготовка сводится к определению функции `f(x,t):=sin(x-t)` и созданию ее декартова графика `y(x,FRAME)`.
2. Выполните команду **Tools / Animation / Record** (Сервис / Анимация / Запись).
3. В диалоговом окне **Record Animation** (Анимация) задайте номер первого кадра в поле **From** (От), номер последнего кадра в поле **To** (До) и скорость анимации в поле **At** (Скорость) в кадрах в секунду (рис. 13.23).
4. Выделите протаскиванием указателя мыши при нажатой левой кнопке мыши область в документе, которая станет роликом анимации.
5. В диалоговом окне **Record Animation** (Анимация) нажмите кнопку **Animate** (Анимация). После этого в окошке диалогового окна **Record Animation** (Анимация) будут появляться результаты расчетов выделенной области, сопровождающиеся выводом текущего значения переменной `FRAME`. По окончании этого процесса на экране появится окно проигрывателя анимации (рис. 13.24).
6. Запустите просмотр анимации в проигрывателе нажатием кнопки воспроизведения в левом нижнем углу окна проигрывателя.

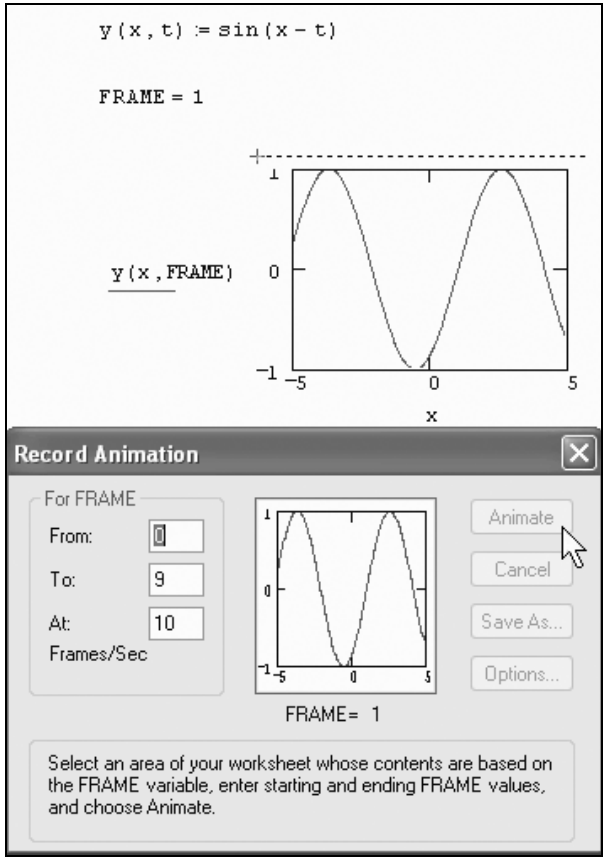



**Рис. 13.23.** Начало создания анимации

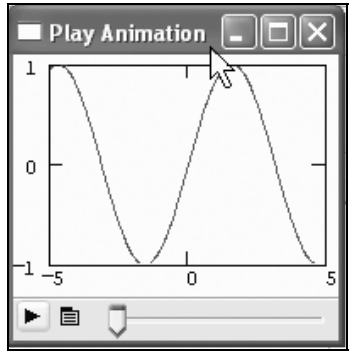



**Рис. 13.24.** Просмотр созданного ролика анимации

7. В случае если вид анимации вас устраивает, сохраните ее в виде видеофайла, нажав кнопку **Save As** (Сохранить как) в диалоговом окне **Record Animation** (Анимация). В появившемся диалоговом окне **Save Animation** (Сохранить анимацию) обычным для Windows способом укажите имя файла и его расположение на диске.
8. Закройте диалог **Record Animation** (Анимация) нажатием кнопки **Cancel** (Отмена) или кнопки управления его окном.

После того как вы сохранили видеофайл, его можно использовать за пределами Mathcad. Скорее всего, если вы, находясь в обозревателе Windows, дважды щелкнете на имени этого файла, он будет загружен в проигрыватель видеофайлов Windows, и вы увидите его на экране компьютера. Таким образом, запуская видеофайлы в обычном проигрывателе, вы имеете возможность устроить красочную презентацию результатов вашей работы как на своем, так и на другом компьютере.

***Примечание***

При создании файлов анимации допускается выбирать программу видеосжатия (кодек) и качество компрессии. Делается это с помощью кнопки **Options** (Опции) в диалоговом окне **Record Animation** (Анимация).

## ⑫ 13.3.3. Мастер импорта данных и функция

В Mathcad 12 появились две новых, более универсальных, возможности для импорта данных из внешнего файла. Они связаны с появлением мастера импорта данных, позволяющего осуществить импорт в нужном формате в диалоговом режиме с подсказками, а также новой встроенной функции `READFILE`, призванной унифицировать процесс импорта. Первый путь позволяет импортировать данные "вручную", проследив процесс считывания данных последовательно, шага за шагом, а второй — автоматизировать его, не путаясь в других многочисленных функциях импорта.

***Примечание***

Оба способа подразумевают возможность импорта файлов данных самых разных форматов: текстовых с разнообразными символами-разделителями, а также файлы формата XLS (Microsoft Excel).

Рассмотрим реализацию первой возможности на примере считывания данных из файла, представленного ранее на рис. 13.20 и 13.21 и в листингах 13.16 и 13.17 (*см. разд. 13.3.1*):

1. Введите команду меню **Insert / Component** (Вставка / Компонент), а затем выберите в списке тип компонента **Data Import Wizard** (Мастер импорта данных). В результате появится окно мастера, которое в пошаговом диалоговом режиме позволит осуществить считывание нужной информации (рис. 13.25).

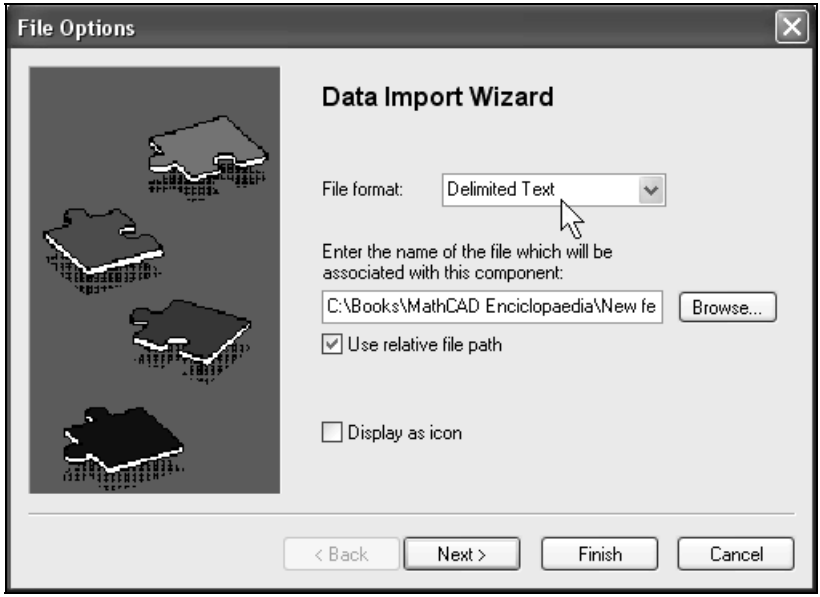



**Рис. 13.25.** Стартовая страница окна **Data Import Wizard**

2. Выберите в раскрывающемся списке **File format** (Формат файла) желаемый формат файла, из которого вы осуществляете импорт (рис. 13.26). Если вы затрудняетесь с его точной идентификацией, лучшим решением будет задание типа **Delimited Text** (Текст с разделителями), что позволит возложить распознавание типа данных и формат их записи на Mathcad.
3. Нажмите кнопку **Browse** (Пролистать) и отыщите в открывшемся диалоговом окне местоположение нужного вам файла.

4. Если вы уверены (например, основываясь на накопленном опыте работы с мастером импорта), что данные будут считаны правильно, то можете сразу нажать кнопку **Finish** (Завершить). Если вы хотите просмотреть и, при необходимости, изменить те или иные опции импорта, нажмите кнопку **Next** (Вперед). В последнем случае в диалоговом режиме еще на двух страницах мастера (подобных рис. 13.26, но с новыми параметрами) можно будет отредактировать многочисленные параметры импорта (такие как тип разделителя между данными, интервалы импорта и т. д.).

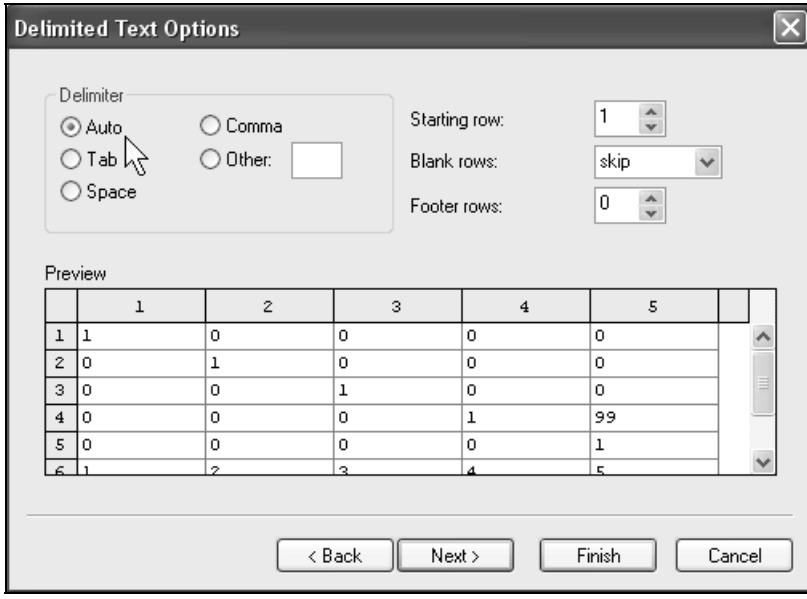
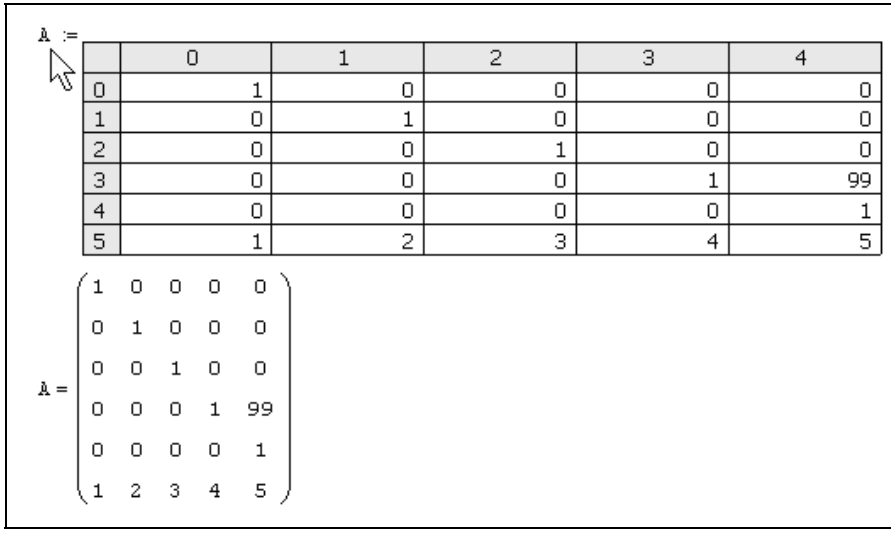



**Рис. 13.26.** Следующая страница окна **Data Import Wizard**

5. После нажатия кнопки **Finish** (Завершить) в диалоге **Data Import Wizard** (Мастер импорта данных) и возвращения на рабочую область документа Mathcad введите в местозаполнитель, появившийся слева от таблицы импортированных данных, желаемое имя переменной. В дальнейших расчетах ее можно будет использовать как обычную матрицу.

Итог работы мастера показан на рис. 13.27. Его первая строка является результатом описанных шагов по считыванию данных в матрицу, а вторая строка показывает вывод этой матрицы в стандартной для Mathcad форме.



A :=

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 99 |
| 4 | 0 | 0 | 0 | 0 | 1 |
| 5 | 1 | 2 | 3 | 4 | 5 |

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 99 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

**Рис. 13.27.** Результат импорта данных из файла

Новая функция `READFILE` облегчает процесс "программного" считывания данных из файла (листинг 13.22):

- `READFILE ("file","type",[colwidth,rows,cols,emptyfill])` — возвращает матрицу с элементами, считанными из внешнего файла данных:
  - `"file"` — название файла (включая путь к нему на диске);
  - `"type"` — тип файла (`"delimited"` или `"Excel"`);
  - `colwidth` — ширина столбца данных, считываемого из файла в случае выбора в качестве предыдущего параметра типа `"fixed"`, т. е. с фиксированной шириной данных;
  - `rows` — начальная строка импорта данных или двухкомпонентный вектор, задающий интервал импорта строк;
  - `cols` — начальный столбец импорта данных или двухкомпонентный вектор, задающий интервал импорта столбцов;
  - `emptyfill` — значение, которое будет использовано для замены отсутствующих данных (пустот в файле). Для него можно использовать значение НеЧисло (NaN) (*см. разд. 1.2.5*).

**Листинг 13.22. Импорт данных при помощи универсальной функции `READFILE`**

$$A := \text{READFILE}(\text{"datafile.txt"}, \text{"delimited"})$$

$$B := \text{READFILE}\left[\text{"datafile.txt"}, \text{"delimited"}, \begin{pmatrix}4\\6\end{pmatrix}, \begin{pmatrix}4\\5\end{pmatrix}\right]$$

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 99 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 99 \\ 0 & 1 \\ 4 & 5 \end{pmatrix}$$

# Глава 14

# Спектральный анализ

Мощным инструментом обработки данных, определенных дискретной зависимостью $y(x_i)$ или непрерывной функцией $f(x)$ (полученной, например, посредством интерполяции или регрессии, как об этом рассказано в *главе 13*), является спектральный анализ, имеющий в своей основе различные интегральные преобразования. Спектральный анализ используется как в целях подавления шума, так и для решения других проблем обработки данных. *Спектром* совокупности данных $y(x)$ называют некоторую функцию другой координаты (или координат) $F(\omega)$, полученную в соответствии с определенным алгоритмом. Примерами спектров являются преобразование Фурье (*см. разд. 14.1*) и вейвлет-преобразование (*см. разд. 14.2*). Напомним, что некоторые преобразования, например, Фурье и Лапласа, можно осуществить в режиме символьных вычислений (*см. главу 5*). Каждое из интегральных преобразований эффективно для решения своего круга задач анализа данных.

Задачами, непосредственно связанными со спектральным анализом, являются проблемы *сглаживания* и *фильтрации* данных (*см. разд. 14.3*). Они заключаются в построении для исходной экспериментальной зависимости $y(x_i)$ некоторой (непрерывной или дискретной) зависимости $f(x)$, которая должна приближать ее, учитывая к тому же, что данные $(x_i, y_i)$ получены с некоторой погрешностью, выражающей шумовую компоненту измерений. При этом функция $f(x)$ с помощью того или иного алгоритма уменьшает погрешность, присутствующую в данных $(x_i, y_i)$. Такого типа задачи называют *задачами фильтрации*. Сглаживание путем построения регрессии данных (*см. разд. 13.2*) — это частный случай фильтрации.

### *Примечание*

Здесь нельзя не отметить, что в качестве дополнений к Mathcad поставляются три пакета расширения, включающие большое количество дополнительных встроенных функций для обработки данных. Названия пакетов расширения говорят сами за себя: **Wavelet extension pack** (Вейвлет-анализ данных), **Signal**

**processing** (Анализ сигналов) и **Image processing** (Анализ изображений). Работа с пакетами расширения не слишком отличается от обычных приемов работы с Mathcad — следует только установить их определенным образом, как описано в руководстве пользователя Mathcad. Рассмотрение их возможностей выходит далеко за пределы данной книги, поэтому лишь упомянем о том, что описание встроенных функций и примеры их применения рассмотрены в трех электронных книгах, представляющих, соответственно, три упомянутых пакета расширения. Получить доступ к нужной книге можно, наводя указатель мыши на пункт **E-Books** (Электронные книги) в меню **Help** (Справка) и выбирая в открывающемся подменю имя нужного пакета расширения.

# 14.1. Фурье-спектр

Интегральные преобразования массива сигнала `y(x)` ставят в соответствие всей совокупности данных `y(x)` некоторую функцию другой координаты `F(v)`. Рассмотрим встроенные функции для расчета интегральных преобразований, реализованных в Mathcad.

Математический смысл преобразования Фурье состоит в представлении сигнала `y(x)` в виде бесконечной суммы синусоид вида `F(v)·sin(v·x)`. Функция `F(v)` называется *преобразованием Фурье*, или *интегралом Фурье*, или *Фурье-спектром* сигнала. Ее аргумент `v` имеет смысл частоты соответствующей составляющей сигнала. Обратное преобразование Фурье переводит спектр `F(v)` в исходный сигнал `y(x)`. Согласно определению,

$$F(v) = \int_{-\infty}^{\infty} y(x) \cdot \exp(-ivx)\,dx .$$

Как видно, преобразование Фурье является комплексной величиной, даже если сигнал действительный.

## 14.1.1. Фурье-спектр действительных данных

Преобразование Фурье имеет огромное значение для различных математических приложений, и для него разработан очень эффективный алгоритм, называемый *БПФ* (быстрое преобразование Фурье). Рассмотрим сначала наиболее типичную для физического эксперимента ситуацию расчета Фурье-спектра действительного сигнала, для которой алгоритм БПФ реализован в нескольких встроенных функциях Mathcad, различающихся нормировками:

- `fft(y)` — вектор прямого преобразования Фурье;
- `FFT(y)` — вектор прямого преобразования Фурье в другой нормировке:
  - `y` — вектор действительных данных, взятых через равные промежутки значений аргумента.

***Внимание!***

Аргумент прямого Фурье-преобразования, т. е. вектор `y`, должен иметь ровно $2^n$ элементов (`n` — целое число). Результатом является вектор с $1+2^{n-1}$ элементами. Если число данных не совпадает со степенью `2`, то необходимо дополнить недостающие элементы нулями, иначе вместо решения появится сообщение об ошибке.

Чтобы смысл преобразования Фурье был более понятен, используем в качестве модельных данных дискретизацию детерминированного сигнала, равного сумме трех синусоид (рис. 14.1). Листинг 14.1 демонстрирует расчет Фурье-спектра по `N=128` точкам, причем предполагается, что интервал дискретизации данных $y_i$ равен $\Delta$. В середине листинга применяется встроенная функция `fft`, а его оставшаяся часть предназначена для корректного пересчета соответствующих значений частот $\Omega_i$ (они вычисляются в последней строке листинга). Обратите внимание, что результаты расчета представляются в виде модуля Фурье-спектра (рис. 14.2), поскольку сам спектр является комплексным. Очень полезно сравнить полученные амплитуды и местоположение пиков спектра (рис. 14.3) с определением синусоид в листинге 14.1.

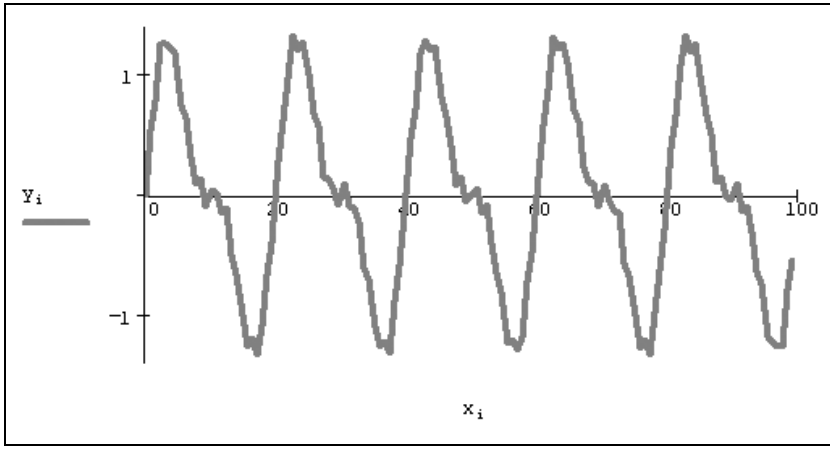



**Рис. 14.1.** Исходные модельные данные (продолжение листинга 14.1)

**Листинг 14.1. Быстрое преобразование Фурье**

$N := 128$ $\quad xMAX := 100$

$\Delta := \frac{xMAX}{N}$ $\quad i := 0 .. N - 1$

$$x_i := i \cdot \Delta$$

$$y_i := \sin\left(2 \cdot \pi \cdot 0.05 \cdot x_i\right) + 0.5 \cdot \sin\left(2 \cdot \pi \cdot 0.1 \cdot x_i\right) + 0.1 \cdot \sin\left(2 \cdot \pi \cdot 0.5 \cdot x_i\right)$$

$$z := \mathrm{fft}(y)$$

$$i := 0 \,..\, \frac{N}{2}$$

$$a_i := \left|z_i\right|$$

$$\Omega 0 := \frac{1}{xMAX} \qquad \Omega_N := \frac{N}{2 \cdot xMAX}$$

$$\Omega D_i := (i + 1) \cdot \Omega 0$$

z =

| | 0 |
|---|---|
| 0 | 0 |
| 1 | -1.547i·10^-13 |
| 2 | 1.144·10^-14-4.623i·10^-14 |
| 3 | -1.79·10^-14-3.246i·10^-13 |
| 4 | 0 |
| 5 | 5.657i |
| 6 | -3.344·10^-14-6.674i·10^-14 |
| 7 | 1.567i·10^-13 |
| 8 | 1.09i·10^-15 |
| 9 | 6.657i·10^-14 |
| 10 | 2.828i |
| 11 | -1.772·10^-14-4.261i·10^-15 |
| 12 | 1.022·10^-15 |
| 13 | 3.624·10^-14+2.097i·10^-13 |
| 14 | 3.433·10^-13-9.737i·10^-14 |
| 15 | 1.988i·10^-14 |

a =

| | 0 |
|---|---|
| 0 | 0 |
| 1 | 1.547·10^-13 |
| 2 | 4.762·10^-14 |
| 3 | 3.251·10^-13 |
| 4 | 0 |
| 5 | 5.657 |
| 6 | 7.465·10^-14 |
| 7 | 1.567·10^-13 |
| 8 | 1.419·10^-15 |
| 9 | 6.658·10^-14 |
| 10 | 2.828 |
| 11 | 1.822·10^-14 |
| 12 | 1.139·10^-15 |
| 13 | 2.128·10^-13 |
| 14 | 3.568·10^-13 |
| 15 | 1.988·10^-14 |

**Рис. 14.2.** Матрица-результат вычисления Фурье-спектра данных (продолжение листинга 14.1)

Исключительно важными представляются два параметра, заданные в предпоследней строке листинга 14.1, называемые соответственно *граничной частотой* и *частотой Найквиста*. Граничная частота $\Omega 0$ определяет нижнюю, а частота Найквиста $\Omega_N$ — верхнюю границу аргумента вычисленного спектра, как показано маркерами на рис. 14.3. Кроме того, важно, что интервал дискретизации Фурье-спектра также равен $\Omega 0$, а общее число вычисляемых точек спектра составляет `N/2` (в нашем примере `N/2=64`). Последние утверждения иллюстрируются маркерами на рис. 14.4, изображающем график Фурье-спектра вблизи нижней границы частот.

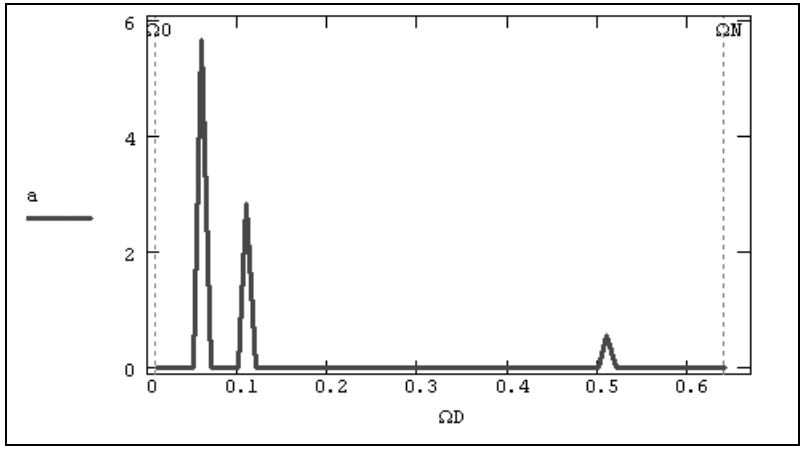



**Рис. 14.3.** График Фурье-спектра данных
(продолжение листинга 14.1)

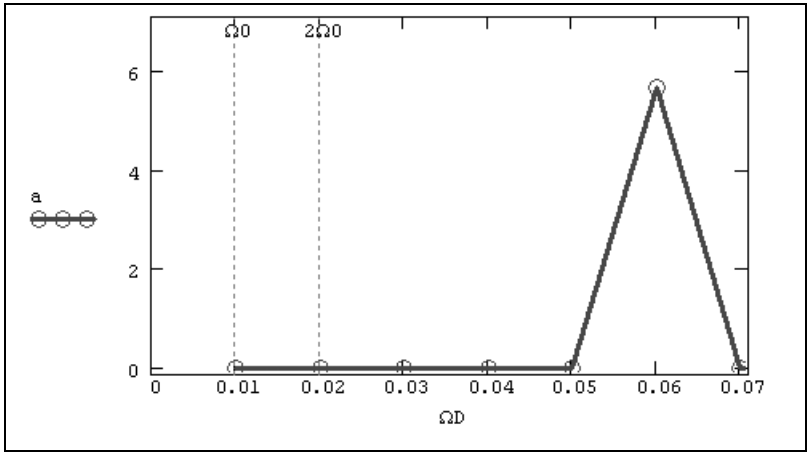



**Рис. 14.4.** Низкочастотная область Фурье-спектра
(продолжение листинга 14.1)

## 14.1.2. Обратное преобразование Фурье

Для расчета обратного Фурье-преобразования (восстановления сигнала по имеющемуся действительному спектру) следует использовать следующие встроенные функции (они также реализуют алгоритм БПФ):

- `ifft(v)` — вектор обратного действительного преобразования Фурье;

- `IFFT(v)` — вектор обратного действительного преобразования Фурье в другой нормировке:
  - `v` — вектор данных Фурье-спектра, взятых через равные промежутки значений частоты.

***Примечание***

Аргумент (вектор `v`) функций, реализующих обратное преобразование Фурье, может быть как действительным, так и комплексным. А вот результат их работы является вектором, составленным из действительных чисел. Если аргумент является `N`-компонентным вектором, где $N=1+2^n$, то в результате получается в два раза больший вектор из $2(N-1)=2^{n+1}$ компонент.

Результат обратного преобразования Фурье спектра, представленного на рис. 14.2 и 14.3, показан в виде кружков на рис. 14.5 вместе с исходными данными.

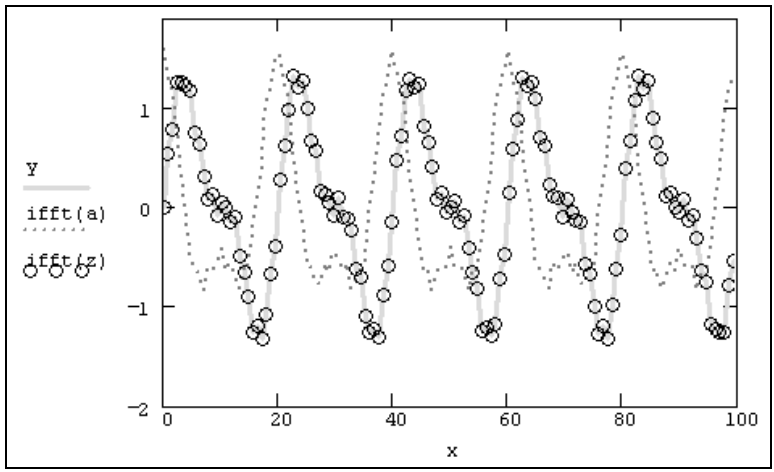



**Рис. 14.5.** Обратное преобразование Фурье (продолжение листинга 14.1)

Видно, что в рассматриваемом случае сигнал `y(x)` восстановлен с большой точностью, что характерно для плавного изменения сигнала. Если же в качестве аргумента функции `ifft` использовать модуль Фурье-спектра (мы присвоили этому вектору в листинге 14.1 имя `a`), то профиль исходного сигнала будет реконструирован правильно, но окажется сдвинутым на определенное

расстояние вдоль оси x. Так происходит из-за того, что взятие абсолютной величины комплексного спектра уничтожает информацию об относительной фазе отсчетов данных.

## 14.1.3. Преобразование Фурье комплексных данных

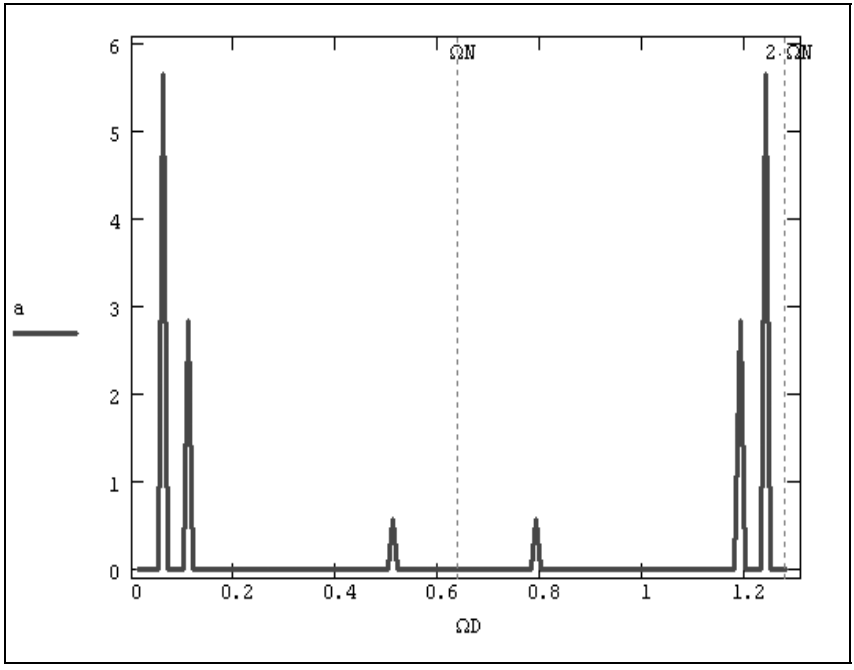
Алгоритм быстрого преобразования Фурье для комплексных данных встроен в соответствующие функции, в имя которых входит литера "C":

- cfft(y) — вектор прямого комплексного преобразования Фурье;
- CFFT(y) — вектор прямого комплексного преобразования Фурье в другой нормировке;
- icfft(y) — вектор обратного комплексного преобразования Фурье;
- ICFFT(v) — вектор обратного комплексного преобразования Фурье в другой нормировке:
  - y — вектор данных, взятых через равные промежутки значений аргумента;
  - v — вектор данных Фурье-спектра, взятых через равные промежутки значений частоты.

Функции действительного преобразования Фурье используют тот факт, что в случае действительных данных спектр получается симметричным относительно нуля, и выводят только его половину (*см. разд. 14.1.1*). Поэтому, в частности, по 128 действительным данным получалось всего 65 точек спектра Фурье. Если к тем же данным применить функцию комплексного преобразования Фурье (рис. 14.6), то получится вектор из 128 элементов. Сравнивая рис. 14.3 и 14.6, можно уяснить соответствие между результатами действительного и комплексного Фурье-преобразования.

**Листинг 14.2. Комплексное быстрое преобразование Фурье (продолжение листинга 14.1)**

```
z := cfft(y)
i := 0 .. N - 1
a_i := |z_i|                ΩD_i := (i + 1)·Ω0
```




z =

| | 0 |
|---|---|
| 112 | 0 |
| 113 | 0 |
| 114 | 0 |
| 115 | 0 |
| 116 | $1.023 \cdot 10^{-15}$ |
| 117 | 0 |
| 118 | -2.828i |
| 119 | $1.033 \cdot 10^{-15}$ |
| 120 | $-1.093i \cdot 10^{-15}$ |
| 121 | 0 |
| 122 | 0 |
| 123 | -5.657i |
| 124 | 0 |
| 125 | 0 |
| 126 | 0 |
| 127 | 0 |

**Рис. 14.6.** Комплексное преобразование Фурье (продолжение листинга 14.2)

## 14.1.4. Пример: артефакты дискретного Фурье-преобразования

При численном нахождении преобразования Фурье следует очень внимательно относиться к таким важнейшим параметрам, как объем выборки (в терминах листинга 14.1, xMAX) и интервал дискретизации ($\Delta$). Соотношение этих двух величин определяет диапазон частот $(\Omega 0, \Omega N)$, для которых возможно вычисление значений Фурье-спектра. В этой связи хотелось бы обратить внимание на три типичные опасности, которые могут подстерегать неподготовленного исследователя при расчете дискретного Фурье-преобразования и быть для него весьма неожиданными.

### Влияние конечности интервала выборки

Во-первых, следует обратить внимание на само определение преобразования Фурье как интеграла с бесконечными пределами. Его численное отыскание подразумевает принципиальную ограниченность интервала интегрирования (просто в силу конечности объема выборки). Поэтому самым очевидным несоответствием будет поиск, вообще говоря, другого интеграла, отличного от интеграла Фурье. Влияние конечности интервала выборки проявляется главным образом на искажении его низкочастотной области. В качестве примера приведем Фурье-спектр гармонической функции с частотой 0.015. Для его расчета достаточно заменить в листинге 14.1 четвертую строку на равенство $y_i := \sin(2 \cdot \pi \cdot 0.015 \cdot x_i)$. Соответствующий Фурье-спектр изображен на рис. 14.7 (сверху — в обычном, а снизу — в более крупном масштабе) и демонстрирует не совсем правильное поведение в низкочастотной области. Как видно, спектр содержит довольно широкий максимум вместо одиночного пика, как было в случае средних частот сигнала на рис. 14.3.

#### *Примечание 1*

Если быть точным, вместо спектра некоторой функции f(x) дискретное преобразование Фурье подразумевает вычисление спектра другой функции $f(x) \cdot \Phi(x)$, где $\Phi(x)$ — это функция-ступенька, равная единице в пределах расчетного интервала и нулю за его пределами. В частотной области это соответствует операции *свертки* означенных двух функций, что, конечно, искажает (неизвестный) точный спектр f(x). Для борьбы с проявлением конечности интервала выборки используются специальные методы, основанные на применении техники *спектральных окон*.

#### *Примечание 2*

Из сказанного ясно, почему исследователя не должна смущать необходимость дополнения массива исходных данных нулями до размера $2^n$ (чтобы можно было использовать алгоритм БПФ). По самому определению дискретного Фурье-

преобразования, исходная функция и так предполагается равной нулю за пределами расчетного интервала, что приводит к неминуемым искажениям.

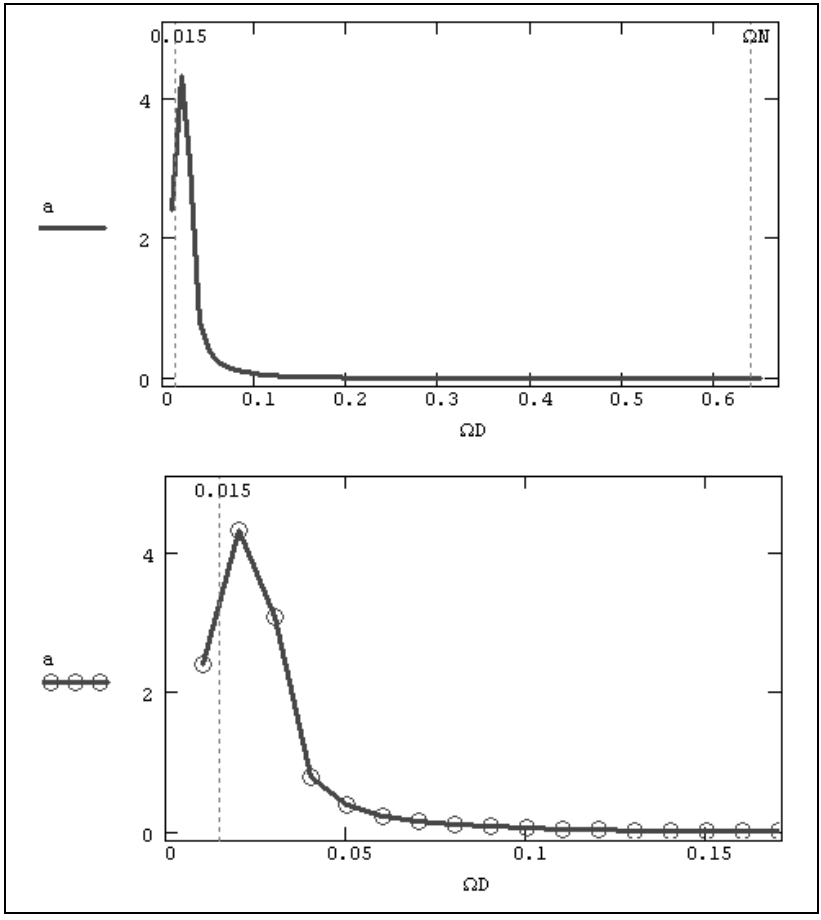



**Рис. 14.7.** Иллюстрация влияния конечности выборки на расчет низкочастотной части Фурье-спектра

## Сдвиг ноль-линии

Еще одним, наиболее ярким, проявлением вредного влияния конечности интервала выборки может служить расчет Фурье-преобразования суммы гармонического сигнала и константы (рис. 14.8). Для того чтобы получить данный рисунок, достаточно еще слегка (по сравнению с рис. 14.7) модифицировать строку листинга, касающуюся определения компонент вектора `y`, добавив к нему `1`: $y_i := \sin(2 \cdot \pi \cdot 0.015 \cdot x_i) + 1$.

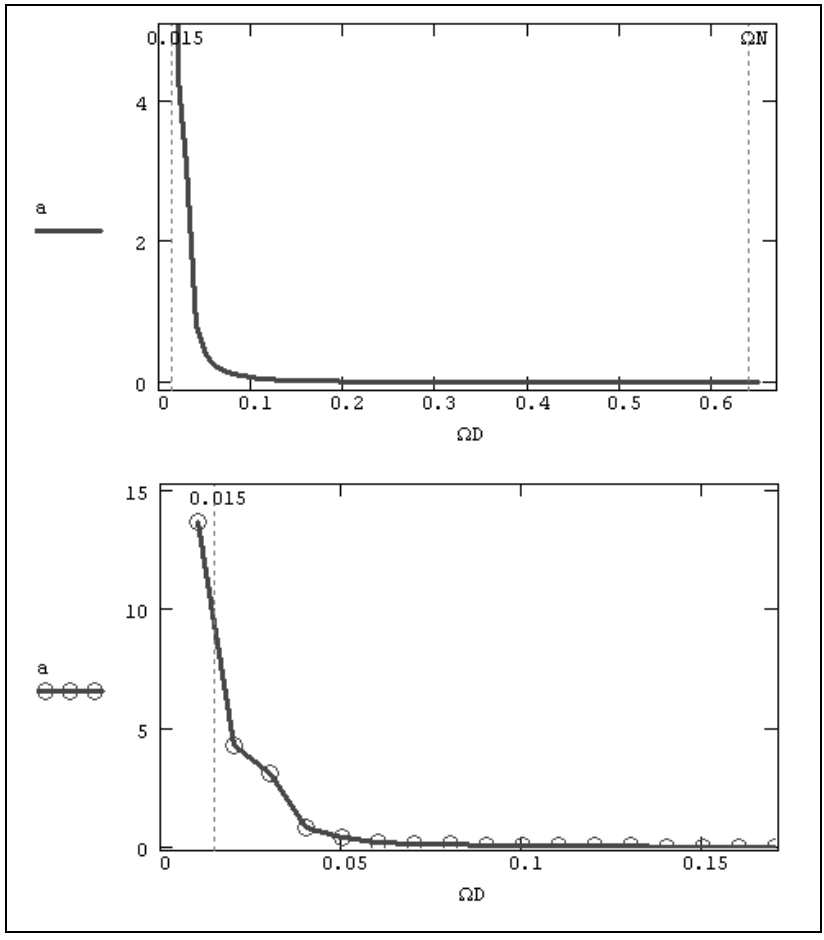



**Рис. 14.8.** Фурье-спектр суммы гармонического сигнала и константы (влияние конечности выборки)

Сравнивая рис. 14.7 и 14.8, несложно догадаться, почему так разительно изменился вид спектра в низкочастотной области. Пугающий рост спектра на левом крае частотного интервала объясняется совокупностью двух факторов: конечности выборки и добавлением к сигналу ненулевой постоянной составляющей (так называемым сдвигом *ноль-линии*). Сумма сигнала и константы определяет соответствующее влияние на вычисленный спектр, который также оказывается (просто в силу линейности операции интегрирования) суммой спектров сигнала и ступенчатой функцией (равной той самой константе внутри расчетного интервала и нулю за его пределами).

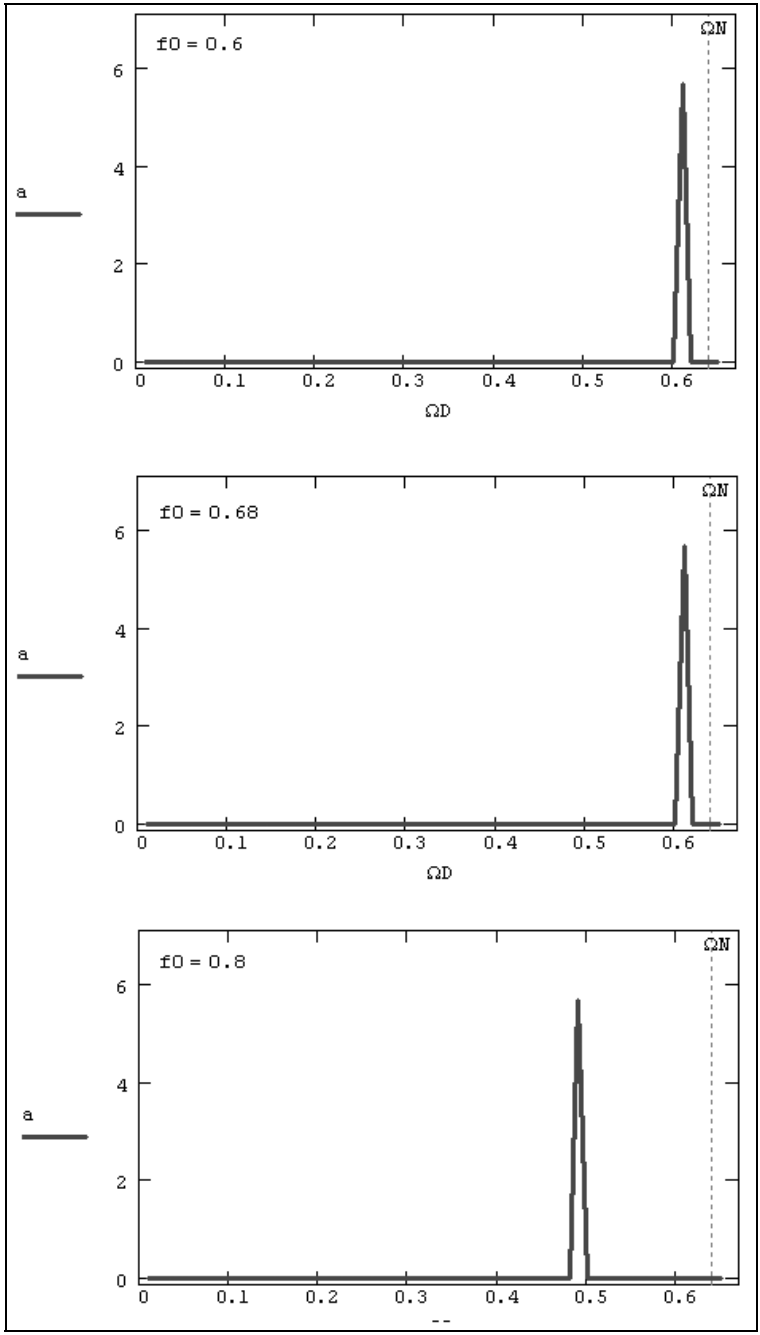



**Рис. 14.9.** Расчеты Фурье-спектров гармонических сигналов с разной частотой ("маскировка частот")

Избавиться от искажений, вызванных сдвигом ноль-линии, довольно просто. Достаточно (до Фурье-преобразования) вычислить среднее значение выборки и затем вычесть его из каждого элемента выборки. Если после этой операции вычислить Фурье-спектр, то он окажется примерно таким, как показано на рис. 14.7.

### Маскировка частот

Еще один классический пример ошибочного расчета Фурье-спектра связан с возможным присутствием в сигнале гармоник с частотой, превышающей частоту Найквиста, в данном примере `ΩN=0.64` (*см. разд. 14.1.1*). Иллюстрация эффекта, называемого "маскировкой частот", приведена на рис. 14.9, который содержит расчет спектров трех различных синусоидальных сигналов с разной частотой `f0`. Первый спектр сигнала с частотой, меньшей частоты Найквиста, вычислен верно, а вот два остальных спектра показывают, что в случае превышения частоты Найквиста в спектре начинают присутствовать "лишние" пики. Появление артефактов спектра связано с тем, что дискретных отсчетов начинает не хватать для того, чтобы прописать высокочастотные гармоники с достаточной информативностью.

> ***Примечание***
>
> Напоминаем, что все листинги, имеющиеся в книге, а также документы Mathcad с расчетами *всех* рисунков, вы найдете на прилагаемом к книге компакт-диске.

## 14.1.5. Пример: спектр модели сигнал/шум

Пока мы использовали в качестве примера детерминированный сигнал, представляющий собой сумму трех синусоид. Несмотря на единство термина "дискретное преобразование Фурье", прикладное применение спектрального анализа можно довольно четко разделить на две категории.

- Сигнал, подвергающийся спектральному анализу, получен в условиях пренебрежимо малой погрешности, т. е. его можно, фактически, считать детерминированным. Такая ситуация характерна для экспериментальной оптики и (разного рода) спектроскопии. В этом случае для большинства задач анализа сигналов бывает вполне достаточно использовать простые спектры Фурье, рассмотренные выше (*см. разд. 14.1.1—14.1.3*).
- Сигнал, полученный в присутствии значительной шумовой компоненты, которая существенно искажает его структуру. В этом случае следует гово-

рить о смеси (к счастью, чаще всего аддитивной) "полезный сигнал + шум", причем в большинстве случаев заранее известна некоторая информация о статистике шумовой компоненты. Данная ситуация очень часто встречается в экспериментальной геофизике и радиофизике. В этом случае подходить к интерпретации спектров следует с вероятностной точки зрения. Как раз этому вопросу мы и посвятим данный пример.

## Фурье-спектр

Внесем минимальное добавление в расчеты листинга 14.1, а именно добавим к его четвертой строке (в которой определяется `yi`) еще одно (четвертое) слагаемое: псевдослучайную величину `σ·rnd(1)`, где значение `1/σ` характеризует отношение сигнал/шум. Явный вид изменений, которые следует внести в листинг 14.1, приведен на рис. 14.10, наряду с графиком сигнала `y(x)`. Расчет Фурье-спектра данного сигнала (в соответствии с алгоритмом, представленным выше, см. листинг 14.1) показан на рис. 14.11. Как видно, присутствие шумовой компоненты может значительно искажать спектр сигнала и затруднять его интерпретацию.

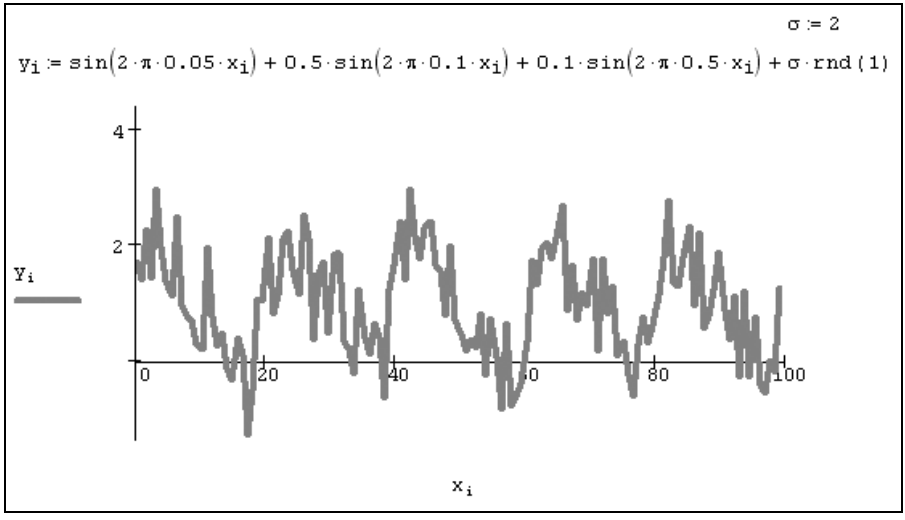



**Рис. 14.10.** Модель сигнал/шум

### *Примечание*

Максимальное значение спектра на левом крае частотного интервала является ни чем иным, как проявлением искажающего влияния конечности выборки и

сдвига ноль-линии (*см. разд. 14.1.3*), произошедшим из-за внесения шума с математическим ожиданием, равным `0.5`.

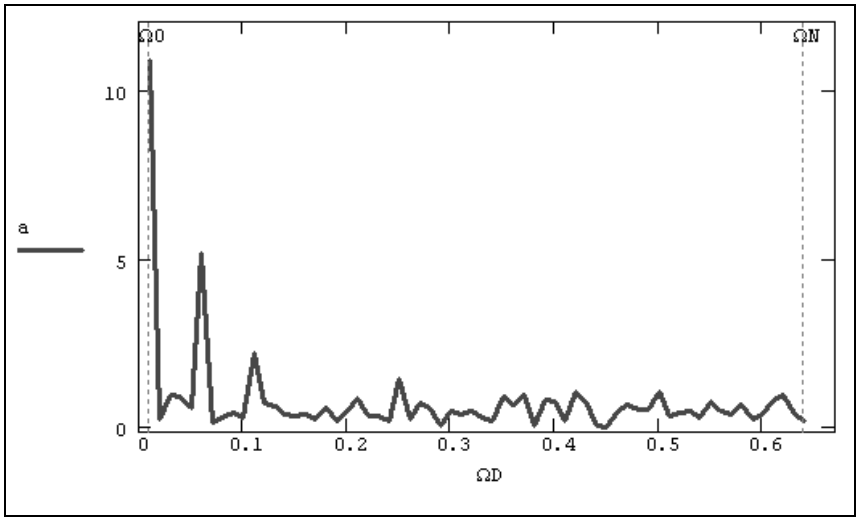



**Рис. 14.11.** График Фурье-спектра данных

## Спектр мощности

В силу стохастичности исходных данных, представляющих сумму полезного сигнала и шума, сами вычисленные значения спектра Фурье носят также случайный характер. В этой связи необходимо знать, с какой погрешностью они рассчитываются. Однако из курса математической статистики известно, что для обычного Фурье-преобразования случайного сигнала (в частности, нормального) не найдено оценок для погрешности. Это слабое место Фурье-спектров делает их практически неприменимыми для анализа случайных сигналов, а вместо них надо применять так называемые *спектры мощности* (или, по-другому, *энергетические* спектры), для которых указанные оценки существуют.

Не углубляясь в теорию математической статистики, приведем пример вычисления спектра мощности сигнала (рис. 14.10), основанный его определении. Как известно, спектром мощности сигнала называют Фурье-преобразование его корреляционной функции. Таким образом, алгоритм расчета спектра мощности сводится к следующему: во-первых, вычислению автокорреляционной функции (рис. 14.12); во-вторых, ее прореживанию и (или)

сглаживанию (в целях уменьшения влияния конечности выборки); и, наконец, в-третьих, расчету ее Фурье-преобразования. Результат вычисления спектра мощности (листинг 14.3) в соответствии с приведенным сценарием показан на рис. 14.13.

### *Примечание 1*

Аналогичным образом, через Фурье-преобразование взаимной корреляционной функции, определяются *взаимные спектры мощности* двух выборок.

### *Примечание 2*

Еще один способ вычисления спектров мощности, не требующий расчета функции корреляции, приведен ниже (*см. разд. 14.3.6*).

### *Примечание 3*

Методика расчета в Mathcad корреляционной функции случайного процесса обсуждалась *в главе 12* (*см. разд. 12.3.3*).

**Листинг 14.3. Расчет спектра мощности для модели сигнал/шум**

$N := 128 \qquad i := 0 .. N - 1$

$x_i := i$

$\sigma := 2$

$Y_i := \sin\left(2 \cdot \pi \cdot 0.05 \cdot x_i\right) + 0.5 \cdot \sin\left(2 \cdot \pi \cdot 0.1 \cdot x_i\right) + 0.1 \cdot \sin\left(2 \cdot \pi \cdot 0.5 \cdot x_i\right) + \sigma \cdot \mathrm{rnd}(1)$

$m := \mathrm{mean}(Y) \qquad D := \mathrm{var}(Y)$

$M := 2^5 + 1 \qquad M = 33$

$$R(j) := \frac{1}{N - 2 \cdot M} \cdot \sum_{i = M}^{N-M} \frac{\left(Y_{i+j} - m\right) \cdot \left(Y_i - m\right)}{D}$$

$j := -M + 2 .. M - 1$

$r_{j+M-2} := R(j)$

$z := \mathrm{fft}(r)$

$i := 0 .. \mathrm{length}(z) - 1$

$a_i := \left|z_i\right| \qquad \Omega D_i := \frac{i + 1}{2 \cdot M}$

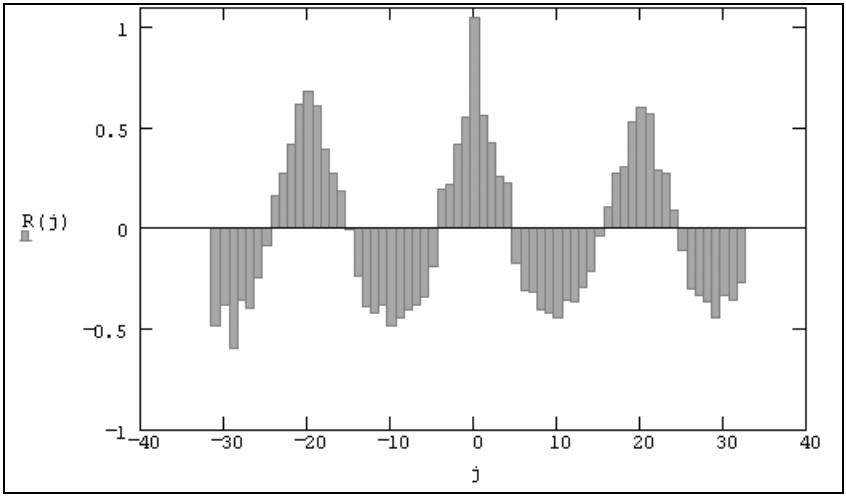



**Рис. 14.12.** Автокорреляционная функция модельной зависимости сигнал/шум (продолжение листинга 14.3)

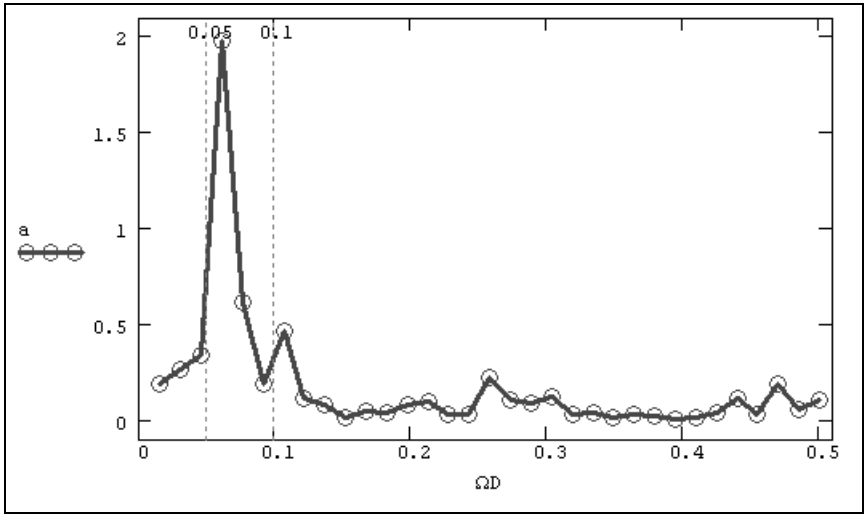



**Рис. 14.13.** График спектра мощности данных модельной зависимости сигнал/шум (продолжение листинга 14.3)

## 14.1.6. Двумерный спектр Фурье

В Mathcad имеется возможность применять встроенные функции комплексного преобразования Фурье не только к одномерным, но и к двумерным мас-

сивам, т. е. матрицам. Соответствующий пример приведен в листинге 14.4 и на рис. 14.14 в виде графика линий уровня исходных данных и рассчитанного Фурье-спектра.

**Листинг 14.4. Двумерное преобразование Фурье**

```
N := 64
i := 0 .. N – 1          j := 0 .. N – 1
y_{i,j} := sin((i + j)/10) + sin((i – j)/10)
F := CFFT(y)
F := submatrix(F, 7, N – 7, 7, N – 7)
```

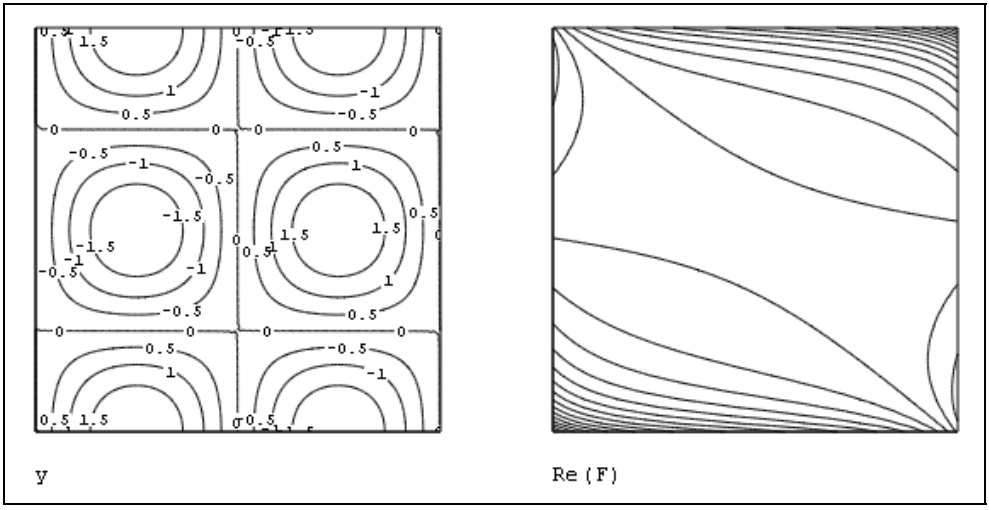



**Рис. 14.14.** Данные (слева) и их Фурье-спектр (справа)
(продолжение листинга 14.4)

# 14.2. Вейвлет-спектры

В последнее время возрос интерес к другим интегральным преобразованиям, в частности, *вейвлет-преобразованию* (или *дискретному волновому*). Оно применяется, главным образом, для анализа нестационарных сигналов и для многих задач подобного рода оказывается более эффективным, чем преоб-

разование Фурье. Основным отличием вейвлет-преобразования является разложение данных не по синусоидам (как для преобразования Фурье), а по другим функциям, называемым *вейвлетобразующими*. Вейвлетобразующие функции, в противоположность бесконечно осциллирующим синусоидам, локализованы в некоторой ограниченной области своего аргумента, а вдали от нее равны нулю или ничтожно малы. Пример такой функции, называемой "мексиканской шляпой", показан на рис. 14.15.

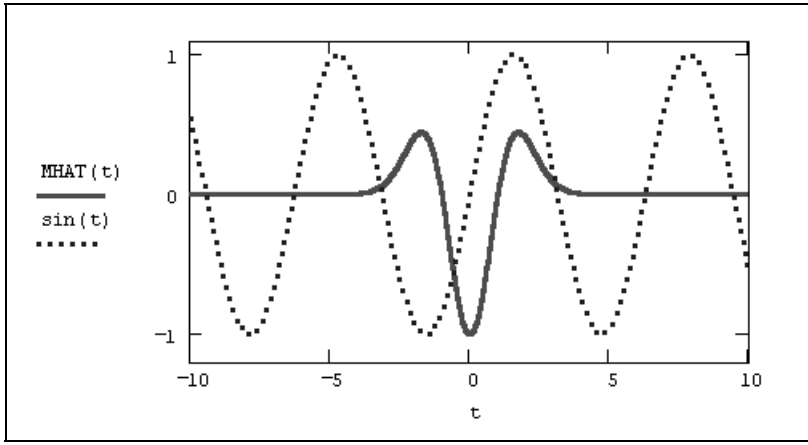



**Рис. 14.15.** Сравнение синусоиды и вейвлетобразующей функции

Из-за своего математического смысла вейвлет-спектр имеет не один аргумент, а два. Помимо частоты, вторым аргументом `b` является место локализации вейвлетобразующей функции. Поэтому `b` имеет ту же размерность, что и `x`.

## 14.2.1. Встроенная функция вейвлет-преобразования

Mathcad имеет одну встроенную функцию для расчета вейвлет-преобразования на основе вейвлетобразующей функции Добеши:

- `wave(y)` — вектор прямого вейвлет-преобразования;
- `iwave(v)` — вектор обратного вейвлет-преобразования:
  - `y` — вектор данных, взятых через равные промежутки значений аргумента;
  - `v` — вектор данных вейвлет-спектра.

Аргумент функции вейвлет-преобразования, т. е. вектор `y`, должен так же, как и в преобразовании Фурье, иметь ровно $2^n$ элементов (`n` — целое число). Результатом функции `wave` является вектор, скомпонованный из нескольких коэффициентов `C` двухпараметрического вейвлет-спектра. Использование функции `wave` объясняется на примере анализа суммы двух синусоид в листинге 14.5, а три семейства коэффициентов вычисленного вейвлет-спектра показаны на рис. 14.16.

**Листинг 14.5. Поиск вейвлет-спектра Добеши**

$$f(t) := \sin\left(2\cdot\pi\cdot\frac{t}{50}\right) + 0.3\cdot\sin\left(2\cdot\pi\cdot\frac{t}{10}\right)$$

$Nmax := 256 \qquad i := 0 .. Nmax - 1$

$y_i := f(i)$

$W := wave(y)$

$$Nlevels := \frac{\ln(Nmax)}{\ln(2)} - 1 \qquad Nlevels = 7$$

$k := 1, 2 .. Nlevels$

$coeffs(level) := submatrix\left(W, 2^{level}, 2^{level+1} - 1, 0, 0\right)$

$$C_{i,k} := coeffs(k)_{floor\left[\frac{i}{\left(\frac{Nmax}{2^k}\right)}\right]}$$

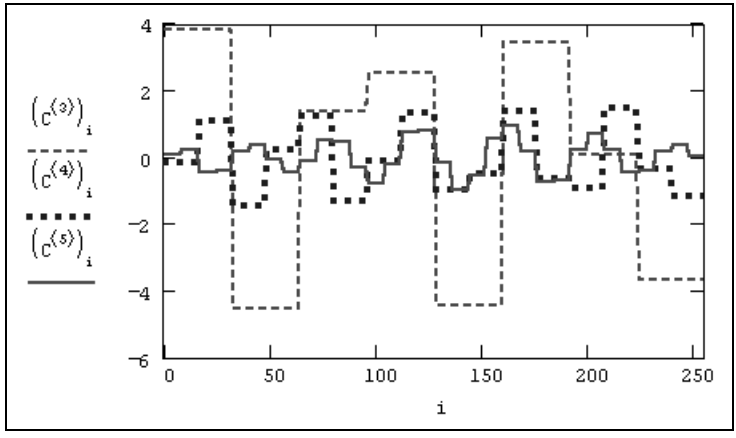



**Рис. 14.16.** Вейвлет-спектр на основе функции Добеши (продолжение листинга 14.5)

## 14.2.2. Программирование вейвлет-преобразований

Наряду со встроенной функцией `wave` Mathcad снабжен пакетом расширения для осуществления вейвлет-анализа. Пакет расширения содержит большое число дополнительных встроенных функций, имеющих отношение к вейвлет-преобразованиям. Обзор пакетов расширения выходит за рамки данной книги, поэтому ограничимся простым упоминанием об этой возможности. Напомним, что дополнительную информацию об использовании данных встроенных функций можно найти в соответствующей электронной книге, которую можно открыть при помощи меню **Help / E-Books / Wavelet extension pack** (Справка / Электронные книги / Вейвлет-анализ данных).

Помимо встроенной функции вейвлет-спектра Добеши и возможностей пакета расширения Mathcad, возможно непосредственное программирование алгоритмов пользователя для расчета вейвлет-спектров. Оно сводится к аккуратному расчету соответствующих семейств интегралов. Один из примеров такой программы приведен в листинге 14.6, а ее результат на рис. 14.17. Анализу подвергается та же функция, составленная из суммы двух гармонических функций, сильно различающихся по частоте. Сам график двухпараметрического вейвлет-спектра `C(a,b)` на плоскости `(a,b)` выведен в виде привычных для вейвлет-анализа линий уровня.

**Листинг 14.6. Поиск вейвлет-спектра на основе "мексиканской шляпы"**

$$f(t) := \sin\left(2\cdot\pi\cdot\frac{t}{50}\right) + 0.3\cdot\sin\left(2\cdot\pi\cdot\frac{t}{10}\right)$$

$$MHAT(t) := \frac{d^2}{dt^2}\exp\left(\frac{-t^2}{2}\right)$$

$$Nmax := 256$$

$$\psi(a,b,t) := MHAT\left(\frac{t-b}{a}\right)$$

$$W(a,b) := \int_{-Nmax}^{Nmax} \psi(a,b,t)\cdot f(t)\ dt$$

$$i := 0 .. 10 \qquad b := 0, 1 .. \frac{Nmax}{10}$$

$$a_i := \frac{(i + 15)^4}{2 \times 10^4}$$

$$N_{i,b} := W\left[\left(a_i\right), 2 \cdot b - \frac{Nmax}{10}\right]$$

***Примечание***

Программа листинга очень проста, но исключительно далека от оптимальной в смысле быстродействия. Каждый интеграл вычисляется независимо, без использования методов ускорения, типа применяемых в алгоритме БПФ. Однако простые приемы программирования вполне доступно раскрывают математический смысл вейвлет-преобразования.

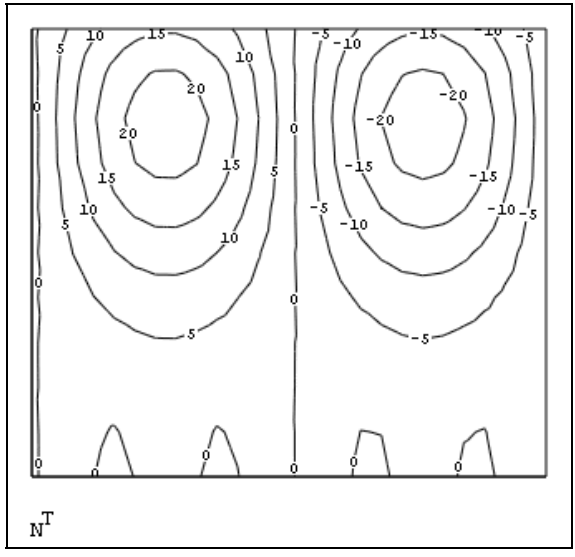



**Рис. 14.17.** Вейвлет-спектр на основе "мексиканской шляпы" (продолжение листинга 14.6)

## 14.3. Сглаживание и фильтрация

При анализе данных часто возникает задача их фильтрации, заключающаяся в устранении одной из составляющих зависимости $y(x_i)$. Наиболее часто целью фильтрации является подавление быстрых вариаций $y(x_i)$, которые чаще всего обусловлены шумом. В результате из быстроосциллирующей

зависимости $y(x_i)$ получается другая, сглаженная зависимость, в которой доминирует более низкочастотная составляющая. В связи с этим считают, что сглаживание является частным случаем более общей задачи фильтрации сигнала, которая связана с подавлением определенных диапазонов частот спектра.

Наиболее простым и эффективным рецептом сглаживания (smoothing) можно считать регрессию различного вида, разобранную в предыдущей главе (*см. разд. 13.2*). Однако регрессия часто уничтожает информативную составляющую данных, оставляя лишь наперед заданную пользователем зависимость.

Часто рассматривают противоположную задачу фильтрации — устранение медленно меняющихся вариаций в целях исследования высокочастотной составляющей. В этом случае говорят о задаче устранения тренда. Иногда интерес представляют смешанные задачи выделения среднемасштабных вариаций путем подавления как более быстрых, так и более медленных вариаций. Одна из возможностей решения связана с применением полосовой фильтрации.

Несколько примеров программной реализации различных вариантов фильтрации приведены в данном разделе.

## 14.3.1. Встроенные функции для сглаживания: ВЧ-фильтр

В Mathcad имеется несколько встроенных функций, реализующих различные алгоритмы сглаживания данных:

- `medsmooth(y,b)` — сглаживание алгоритмом "бегущих медиан";
- `ksmooth(x,y,b)` — сглаживание на основе функции Гаусса;
- `supsmooth(x,y)` — локальное сглаживание адаптивным алгоритмом, основанное на анализе ближайших соседей каждой пары данных:
  - `x` — вектор действительных данных аргумента (для `supsmooth` его элементы должны быть расположены в порядке возрастания);
  - `y` — вектор действительных значений того же размера, что и `x`;
  - `b` — ширина окна сглаживания.

Все функции имеют в качестве аргумента векторы, составленные из массива данных, и выдают в качестве результата вектор сглаженных данных того же размера. Функция `medsmooth` предполагает, что данные расположены равномерно.

### *Примечание*

Подробную информацию об алгоритмах, заложенных в функции сглаживания, вы найдете в справочной системе Mathcad в статье **Smoothing** (Сглаживание), находящейся в разделе **Statistics** (Статистика). Очень полезные сведения о разных типах фильтрации можно отыскать в быстрых шпаргалках.

Часто бывает полезным совместить сглаживание с последующей интерполяцией или регрессией. Соответствующий пример приведен в листинге 14.7 для функции `supsmooth`. Результат работы листинга показан на рис. 14.18 (кружки обозначают исходные данные, крестики — сглаженные, пунктирная кривая — результат сплайн-интерполяции). Сглаживание тех же данных при помощи "бегущих медиан" и функции Гаусса с разным значением ширины окна пропускания показаны на рис. 14.19 и 14.20 соответственно.

**Листинг 14.7. Сглаживание с последующей сплайн-интерполяцией**

```
x := (0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16)^T

y := (4.1  2.4  3  4.3  3.6  5.2  5.9  5  4.7  4  3.5  3.9  3  2.7  3.7  4.8  5.4)^T

z := supsmooth(x, y)

s := cspline(x, z)

A(t) := interp(s, x, z, t)
```

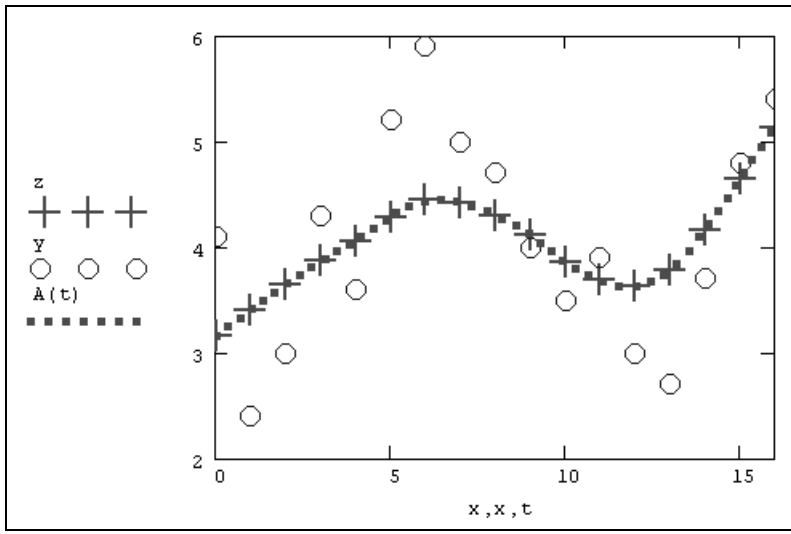



**Рис. 14.18.** Адаптивное сглаживание (продолжение листинга 14.7)

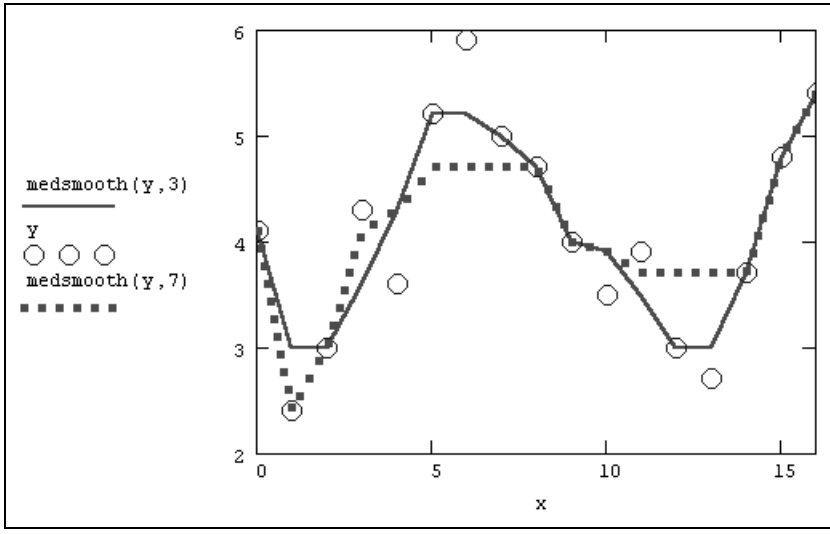



**Рис. 14.19.** Сглаживание
"бегущими медианами"

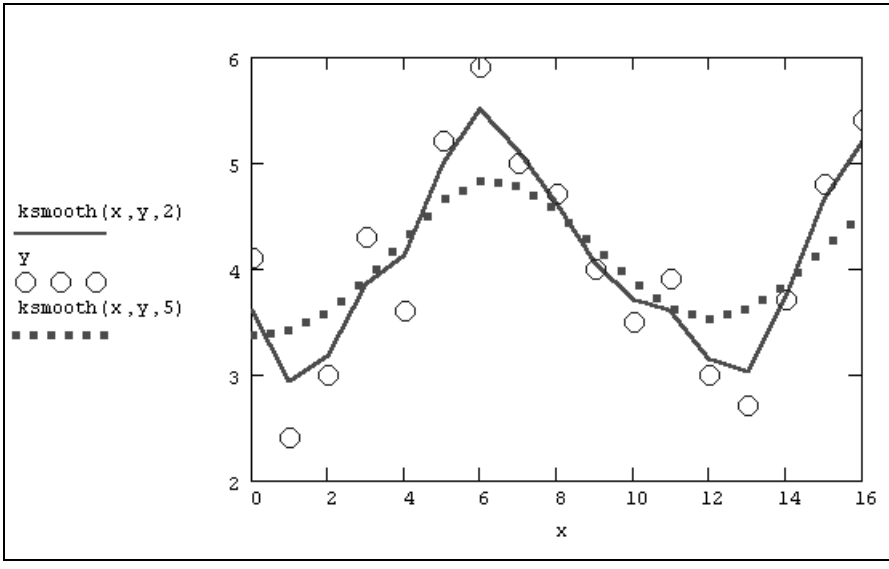



**Рис. 14.20.** Сглаживание
при помощи функции `ksmooth`

## 14.3.2. Скользящее усреднение: ВЧ-фильтр

Помимо встроенных в Mathcad, существует несколько популярных алгоритмов сглаживания, на одном из которых хочется остановиться особо. Самый простой и очень эффективный метод — это скользящее усреднение. Его суть состоит в расчете для каждого значения аргумента среднего значения по соседним `w` данным. Число `w` называют *окном* скользящего усреднения; чем оно больше, тем больше данных участвуют в расчете среднего, тем более сглаженная кривая получается. На рис. 14.21 показан результат скользящего усреднения одних и тех же данных (кружки) с разным окном `w=3` (пунктир), `w=5` (штрихованная кривая) и `w=15` (сплошная кривая). Видно, что при малых `w` сглаженные кривые практически повторяют ход изменения данных, а при больших `w` — отражают лишь закономерность их медленных вариаций.

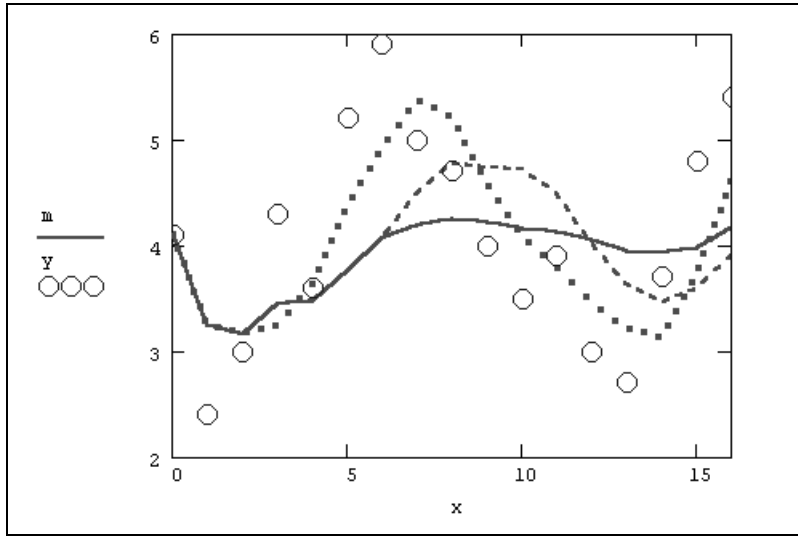



**Рис. 14.21.** Скользящее усреднение с разными `w=3, 5, 15` (листинг 14.8, коллаж трех графиков)

Чтобы реализовать в Mathcad скользящее усреднение, достаточно очень простой программы, приведенной в листинге 14.8. Она использует только значения `y`, оформленные в виде вектора, неявно предполагая, что они соответствуют значениям аргумента `x`, расположенным через одинаковые промежутки. Вектор `x` применялся лишь для построения графика результата (рис. 14.21).

**Листинг 14.8. Сглаживание скользящим усреднением**

$$x := (0 \;\; 1 \;\; 2 \;\; 3 \;\; 4 \;\; 5 \;\; 6 \;\; 7 \;\; 8 \;\; 9 \;\; 10 \;\; 11 \;\; 12 \;\; 13 \;\; 14 \;\; 15 \;\; 16)^T$$

$$y := (4.1 \;\; 2.4 \;\; 3 \;\; 4.3 \;\; 3.6 \;\; 5.2 \;\; 5.9 \;\; 5 \;\; 4.7 \;\; 4 \;\; 3.5 \;\; 3.9 \;\; 3 \;\; 2.7 \;\; 3.7 \;\; 4.8 \;\; 5.4)^T$$

$$w := 15$$

$$N := \mathrm{rows}(y) \qquad N = 17$$

$$i := 0 \,..\, N - 1$$

$$m_i := \mathrm{if}\left(i < w, \frac{\sum_{j=0}^{i} y_j}{i+1}, \frac{\sum_{j=i-w+1}^{i} y_j}{w}\right)$$

***Примечание***

Приведенная программная реализация скользящего усреднения самая простая, но не самая лучшая. Возможно, вы обратили внимание, что все кривые скользящего среднего на рис. 14.21 слегка "обгоняют" исходные данные. Почему так происходит, понятно: согласно алгоритму, заложенному в последнюю строку листинга 14.8, скользящее среднее для каждой точки вычисляется путем усреднения значений предыдущих `w` точек. Чтобы результат скользящего усреднения был более адекватным, лучше применить центрированный алгоритм расчета по `w/2` предыдущим и `w/2` последующим значениям. Он будет немного сложнее, поскольку придется учитывать недостаток точек не только в начале (как это сделано в программе с помощью функции условия `if`), но и в конце массива исходных данных.

## 14.3.3. Устранение тренда: НЧ-фильтр

Еще одна типичная задача возникает, когда интерес исследований заключается не в анализе медленных (или *низкочастотных*) вариаций сигнала `y(x)` (для чего применяется сглаживание данных), а в анализе быстрых его изменений. Часто бывает, что быстрые (или *высокочастотные*) вариации накладываются определенным образом на медленные, которые обычно называют *трендом*. Часто тренд имеет заранее предсказуемый вид, например, линейный. Чтобы устранить тренд, можно предложить последовательность действий, реализованную в листинге 14.9.

1. Вычислить регрессию `f(x)`, например, линейную, исходя из априорной информации о тренде (предпоследняя строка листинга).
2. Вычесть из данных `y(x)` тренд `f(x)` (последняя строка листинга).

**Листинг 14.9. Устранение тренда**

```
x := (0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16)^T

y := (5.1 4.4 5.7 5.3 5.6 5.2 5.9 7.7 6.7 7  6.5 5.9 8.1 8.7 9.7 9.8 10.4)^T

N := rows(y)                    N = 17

i := 0 .. N - 1

f(t) := line(x, y)_0 + line(x, y)_1 · t

m_i := y_i - f(x_i)
```

На рис. 14.22 показаны исходные данные (кружками), выделенный с помощью регрессии линейный тренд (сплошной прямой линией) и результат устранения тренда (пунктир, соединяющий крестики).

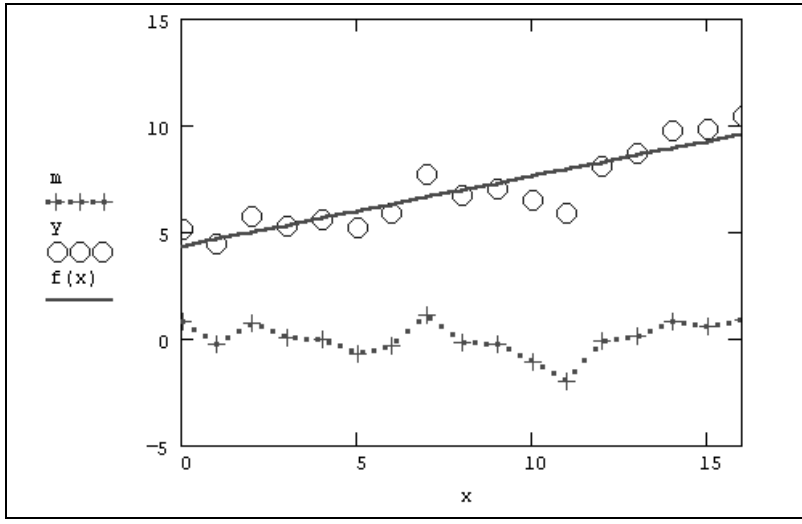



**Рис. 14.22.** Устранение тренда (продолжение листинга 14.9)

## 14.3.4. Полосовая фильтрация

В предыдущих разделах была рассмотрена фильтрация быстрых вариаций сигнала (сглаживание) и его медленных вариаций (снятие тренда). Иногда требуется выделить среднемасштабную составляющую сигнала, уменьшив как более быстрые, так и более медленные его компоненты. Одна из возмож-

ностей решения этой задачи связана с применением полосовой фильтрации на основе последовательного скользящего усреднения.

Алгоритм полосовой фильтрации приведен в листинге 14.10, а результат его применения показан на рис. 14.23 сплошной кривой. Алгоритм реализует такую последовательность операций:

1. Выставление ноль-линии, т. е. приведение массива данных `y` к нулевому среднему значению путем его вычитания из каждого элемента `y` (третья и четвертая строки листинга).
2. Устранение из сигнала `y` высокочастотной составляющей, имеющее целью получить сглаженный сигнал `middle`, например, с помощью скользящего усреднения с малым окном `w` (в листинге 14.10 `w=3`).
3. Выделение из сигнала `middle` низкочастотной составляющей `slow`, например, путем скользящего усреднения с большим окном `w` (в листинге 14.9 `w=7`), либо с помощью снятия тренда (*см. разд. 14.3.3*).
4. Вычесть из сигнала `middle` тренд `slow` (последняя строка листинга), тем самым выделяя среднемасштабную составляющую исходного сигнала `y`.

**Листинг 14.10. Полосовая фильтрация**

```
x := (0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16)^T

y:=(5.1 4.4 5  5.3 5.6 5.2 5.9 7.7 8.7 6.7 7.5 7.9 8.1 8.7 9.7 8.8 10.4)^T

meanY := mean(y)

y := y - meanY

N := rows(y)                    N = 17

i := 0 .. N - 1

w := 3

middle_i := if(i < w, (sum_{j=0}^{i} y_j)/(i+1), (sum_{j=i-w+1}^{i} y_j)/w)

w := 7

slow_i := if(i < w, (sum_{j=0}^{i} middle_j)/(i+1), (sum_{j=i-w+1}^{i} middle_j)/w)

m := middle - slow
```

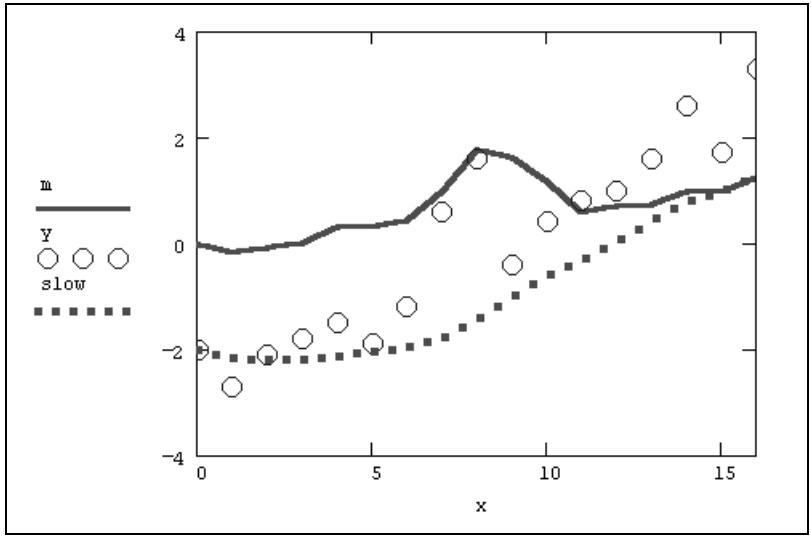



**Рис. 14.23.** Результат полосовой фильтрации (продолжение листинга 14.10)

## 14.3.5. Спектральная фильтрация

Альтернативой всем представленным до сих пор алгоритмам, в частности, методу полосовой фильтрации (*см. предыдущий разд.*) является фильтрация на основе интеграла Фурье. Пока мы использовали для подавления в сигнале тех или иных частотных диапазонов определенные процедуры, основанные на арифметических преобразованиях. Между тем, для той же цели (правда, с большими вычислительными затратами) можно применять методы Фурье-анализа. Действительно, если вычислить спектр сигнала, удалить из него (или существенно уменьшить) определенные частоты, а затем выполнить обратное преобразование Фурье, то результатом будет фильтрованный сигнал.

Пример фильтрации на основе преобразования Фурье приведен в листинге 14.11. В качестве модельного сигнала использовалась смесь двух гармонических сигналов и равномерно распределенного шума (рис. 14.24). Фурье-спектр данных `z`, вычисленный при помощи встроенной функции `fft`, показан на рис. 14.25. Листинг 14.11 отличается от листинга 14.1 (*см. разд. 14.1.1*), главным образом, последними двумя строками, в которых, собственно, и определяется явный вид спектрального фильтра `W(Ω)`, или, по-другому, *спектральное окно*. Обратное Фурье-преобразование спектра произведения спектрального окна `W(Ω)` и Фурье-спектра сигнала `z`,

представляющее собой результат фильтрации, показано на рис. 14.24 сплошной кривой.

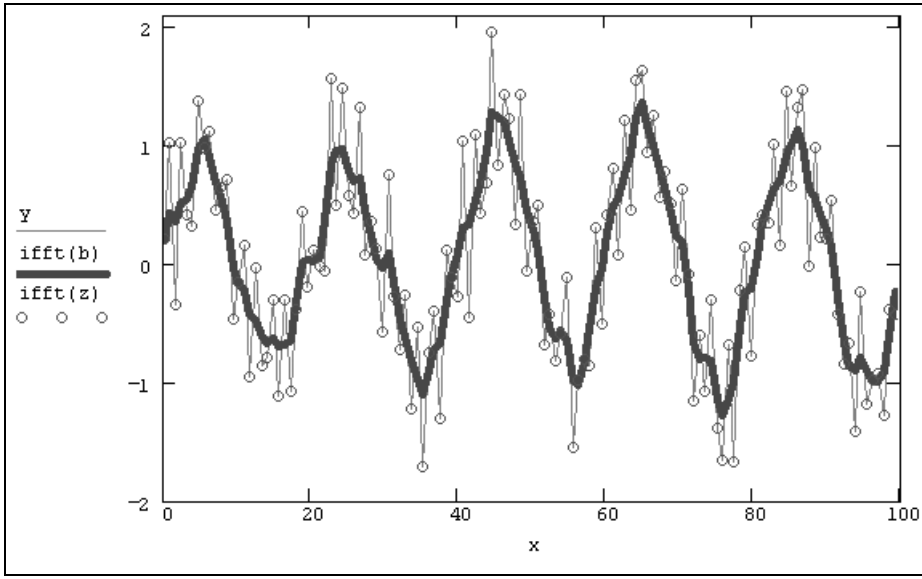



**Рис. 14.24.** Исходный модельный сигнал (кружки) и результат его фильтрации на основе Фурье-преобразования (продолжение листинга 14.11)

**Листинг 14.11. Фильтрация на основе преобразования Фурье**

$N := 128$ $xMAX := 100$

$$\Delta := \frac{xMAX}{N}$$

$i := 0 .. N - 1$

$x_i := i \cdot \Delta$

$y_i := \sin\left(2 \cdot \pi \cdot 0.05 \cdot x_i\right) + 0.5 \cdot \sin\left(2 \cdot \pi \cdot 0.5 \cdot x_i\right) + (rnd(1) - .5)$

$z := fft(y)$

$$\Omega 0 := \frac{1}{xMAX} \qquad \Omega N := \frac{N}{2 \cdot xMAX}$$

$$i := 0 .. \frac{N}{2} \qquad \Omega D_i := (i + 1) \cdot \Omega 0$$

$$W(\Omega) := \exp\left(-\frac{\Omega^2}{0.1}\right)$$

$$b_i := z_i \cdot W\left(\Omega D_i\right)$$

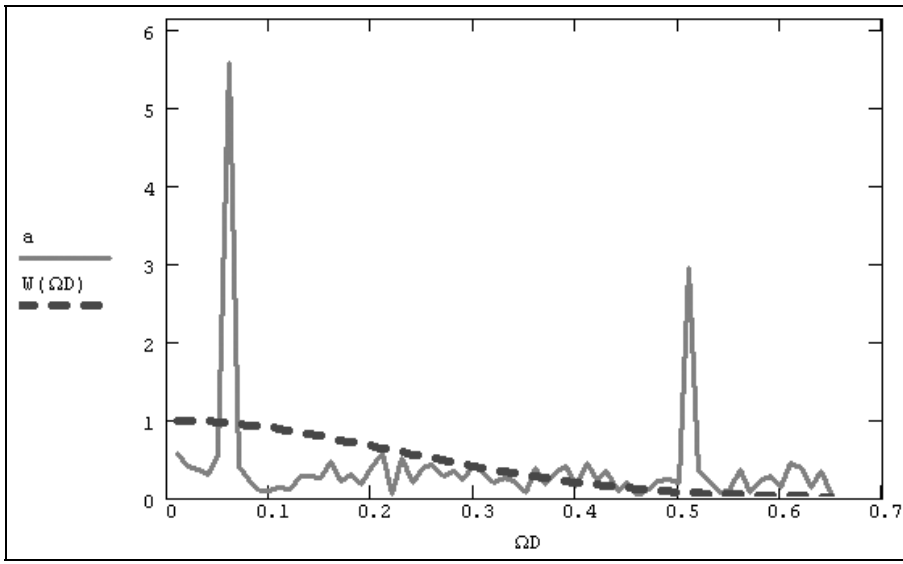



**Рис. 14.25.** Фурье-преобразование модельного сигнала и спектральное окно-фильтр (продолжение листинга 14.11)

## 14.3.6. Пример: вычисление спектра мощности

Завершим главу, посвященную спектральному анализу, еще одним примером вычисления спектра мощности (его определение приведено в *разд. 14.1.5*) модельного сигнала, связанного с использованием его Фурье-спектра. В листинге 14.12 сначала выполняется преобразование Фурье суммы гармонического сигнала и шума (распределенного равномерно), а затем (в трех последних строках листинга) производится его сглаживание путем скользящего усреднения с окном, равным 3. Из курса математической статистики известно, что квадратная степень сглаженного преобразования Фурье может считаться оценкой спектра мощности, и для вычисленного таким образом спектра уже можно применять вероятностные оценки погрешностей его отсчетов. Результаты работы листинга 14.12 (Фурье- и энергетический спектр) показаны на рис. 14.26 (кружками и сплошной кривой соответственно).

## Листинг 14.12. Вычисление спектра мощности

$f0 := 0.1$ $\sigma := 3$

$N := 120$ $T0 := 100$

$$\Delta t := \frac{T0}{N}$$

$i := 0 .. N - 1$

$t_i := i \cdot \Delta t$

$x_i := \sin\left(2 \cdot \pi \cdot f0 \cdot t_i\right) + \sigma \cdot (rnd(1) - 0.5)$

$y := cfft(x)$

$a_i := |y_i|$

$$\Omega D_i := \frac{i + 1}{T0}$$

$i := 1 .. N - 2$

$b_i := 0.23 \cdot a_{i-1} + 0.54 \cdot a_i + 0.23 \cdot a_{i+1}$

$b_0 := 0.54 \cdot a_0 + 0.46 \cdot a_1$

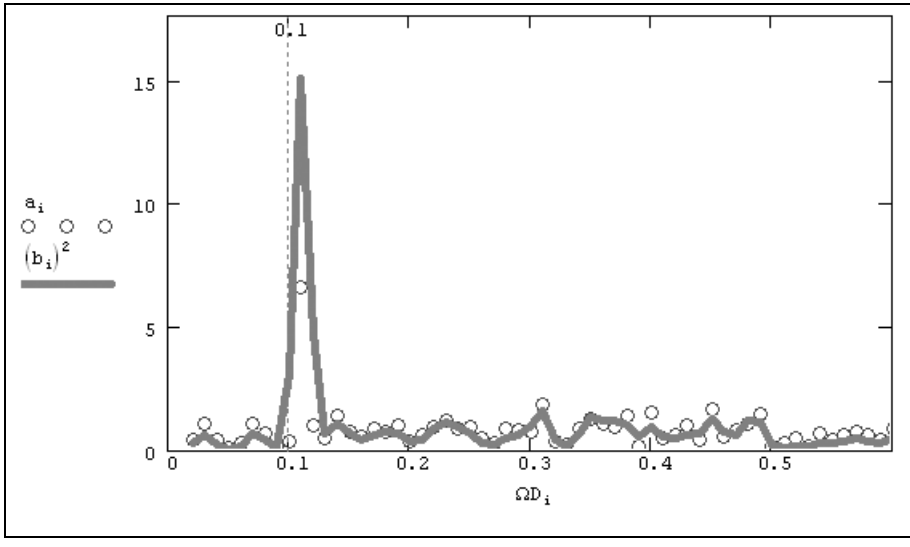
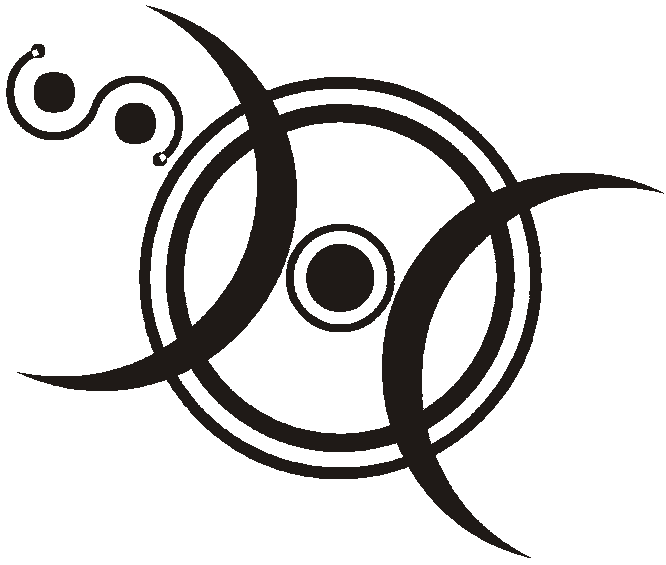



**Рис. 14.26.** Фурье-преобразование модельного сигнала и его спектр мощности (продолжение листинга 14.12)



# Приложения

# Приложение 1

# Пользователям предыдущих версий Mathcad

## ⑬ Новые возможности Mathcad 13

- Автоматическое (фоновое) сохранение документов (*см. разд. 1.6.3*).
- Поддержка новых размерностей: температурные величины и пользовательские размерности (*см. разд. 1.2.7*).
- Отладчик программ (*см. разд. 1.1.2 и 1.5.2*).
- Модернизированный аппарат построения двумерных графиков (*см. разд. 1.4.6*).
- Явные символьные вычисления (*см. разд. 2.3.12*).
- Улучшенный аппарат расчета параметрической регрессии (*см. разд. 13.2.4*).
- Дополнительные алгоритмы линейной алгебры (*см. разд. 8.2*).
- Лучшая совместимость с прежними версиями благодаря появившейся опции переключения быстродействия (*см. разд. 1.6.4*).
- Сообщения об ошибках содержат информацию о размерностях и типах переменных (*см. разд. 1.5.1*).
- Дополнительные опции управления метаданными и примечаниями (*см. разд. 1.5.4*).
- Новая редакция справочной документации (*см. разд. 1.1.4*).

# ⑫ Новые возможности Mathcad 12

- Новый более быстрый математический процессор (*см. разд. 1.2.1*).
- Модернизированный аппарат работы с размерными перемеменными, статическая проверка размерности, новые единицы измерения (ангстрем и т. д.) (*см. разд. 1.2.7*).
- Новый формат файлов на основе XML-разметки (*см. разд. 1.1.1*).
- Новые опции комментирования документов (примечания и метаданные) (*см. разд. 1.3.4*).
- Новые удобные опции импорта данных из внешних файлов (*см. разд. 13.3.3*).
- Возможность построения второй оси `Y` на двумерных графиках (*см. разд. 1.4.6*).
- Новые встроенные функции генерации массивов для повышения удобства построения графиков в логарифмическом масштабе (*см. разд. 1.4.6*).
- Новый тип данных NaN (НеЧисло) (*см. разд. 1.2.5*).
- Новые функции корреляционного анализа сигналов (*см. разд. 12.2.5*).
- Контроль правильности ввода оператора определителя матрицы и длины вектора (*см. разд. 7.3.1*).
- Улучшенная работа функций округления (*см. разд. 2.2.2*).
- Восстановление функции `until` (*см. разд. 2.2.2*).
- Упразднение функции (символа) `δ` (*см. разд. 2.2.2*).
- Новая функция `time` для хронометрирования вычислений (*см. разд. 2.2.3*).
- Новые масштабированные функции Эйри (*см. разд. 2.2.4*).
- Усовершенствование алгоритма регрессии общего вида (*см. разд. 13.2.4*).
- Ограничение на форму записи неизвестных в пределах вычислительного блока (*см. разд. 5.1.1*).
- Запрещение рекуррентного задания функций (*см. разд. 1.2.4*).
- Возможность переопределения встроенных и пользовательских функций и переменных при помощи нового именного оператора (*см. разд. 1.2.4*).

- Изменения в работе строковых функций `str2num` и `vec2str` (*см. разд. 1.2.5*).
- Константа `ORIGIN` теперь может относиться и к строковым переменным (*см. разд. 1.2.5*).

## ⑪ Новые возможности Mathcad 11

- Расширенные возможности импорта и экспорта данных в текстовом формате и формате Microsoft® Excel, введены функции ввода/вывода в двоичные файлы (READBIN и WRITEBIN).
- Улучшенный редактор, повышенная эффективность команд **Undo** и **Redo**, немного модернизированный математический процессор, улучшенный RTF-экспорт и возможность обычного открытия файлов с удаленных серверов по протоколу HTTP.
- Расширенный класс поддающихся решению дифференциальных уравнений в частных производных (введена дополнительная встроенная функция для решения параболических и гиперболических уравнений).
- Расчет ряда специальных функций с комплексными аргументами, таких как функции округления, интегралы ошибок, а также функции Бесселя и Ганкеля.
- Новая функция `sinc` с улучшенным вычислением значений `sin(x)/x` при `x`, стремящемся к `0`.
- Доступ к начальным значениям, использующимся в генераторах последовательностей псевдослучайных чисел.
- Изменения некоторых меню, панелей и диалогов, в частности, вместо меню **Math** (Математика) введено меню **Tools** (Сервис).
- Руководство пользователя в формате PDF.

## Новые возможности Mathcad 2001 и 2001i

- Улучшенный процессор, позволяющий проводить высокоскоростные расчеты, т. н. *режим ускоренных вычислений* (higher speed calculation). Оптимизация вычислений, улучшенное вычисление неопределенных выражений типа `0·ln(0)` и предварительная проверка матриц на сингулярность.

- Повышенные возможности организации гиперссылок, в том числе из одного региона на другой, что осуществляется расстановкой тегов.
- Новые встроенные функции и возможности:
  - функции преобразования декартовых, сферических, цилиндрических координат;
  - новые функции регрессии, например, логарифмической и линейной специального вида;
  - семейство `lookup`-функций для выборки значений из матриц;
  - функции оцифровки звука из звуковых файлов;
  - новый тип графика для подготовки классических гистограмм;
  - возможность представления числа в виде простой дроби.
- Улучшенная связь с другими приложениями благодаря вставке компонентов.
- Новые мощные средства для вставки и редактирования рисунков, в том числе панель **Picture** (Рисунок).
- В Mathcad 2001i введена новая встроенная функция для решения жестких систем обыкновенных дифференциальных уравнений `Radau`, не требующая явного ввода в качестве аргумента якобиана системы.

# Приложение 2

# Команды меню и панели инструментов

Команды меню Mathcad и панели инструментов описаны в табл. П2.1—П2.9.

***Таблица П2.1.*** *Команды меню*

| Меню | Команда | Перевод | Горячая клавиша или комбинация клавиш | Описание |
|---|---|---|---|---|
| **File** (Файл) | **New** | Создать | <Ctrl>+<N> | Создать новый документ |
| | **Open** | Открыть | <Ctrl>+<O> | Открыть существующий документ |
| | **Close** | Закрыть | <Ctrl>+<W> | Закрыть активный документ |
| | **Save** | Сохранить | <Ctrl>+<S> | Сохранить активный документ |
| | **Save As** | Сохранить как | | Сохранить активный документ в другом файле |
| | **Save As Web Page** | Сохранить как Web-страницу | | Сохранить копию активного документа в файле формата HTML |
| | **Page Setup** | Параметры страницы | | Опции вывода активного документа на печать |

***Таблица П2.1*** *(продолжение)*

| Меню | Команда | Перевод | Горячая клавиша или комбинация клавиш | Описание |
|---|---|---|---|---|
| | **Print Preview** | Просмотр | | Предварительный просмотр на экране вывода на печать активного документа |
| | **Print** | Печать | <Ctrl>+<P> | Распечатать активный документ |
| | **Send** | Отправить | | Отправить активный документ по электронной почте |
| | **Properties** | Свойства | | Свойства документа (примечания, опции XML и т. п.) |
| | **Exit** | Выход | | Завершение работы с Mathcad |
| **Edit** (Правка) | **Undo** | Отменить | <Ctrl>+<Z> | Отменить последнее действие |
| | **Redo** | Повторить | <Ctrl>+<Y> | Повторить последнее отмененное действие |
| | **Cut** | Вырезать | <Ctrl>+<X> | Вырезать выбранное выражение в буфер |
| | **Copy** | Копировать | <Ctrl>+<C> | Копировать выбранное выражение в буфер |
| | **Paste** | Вставить | <Ctrl>+<V> | Вставить выражение из буфера |
| | **Paste Special** | Специальная вставка | | Вставить объект специального формата, находящийся в буфере |
| | **Delete** | Удалить | <Ctrl>+<D> | Удалить выбранный регион |
| | **Select All** | Выделить все | <Ctrl>+<A> | Выделить всю рабочую область |
| | **Find** | Найти | <Ctrl>+<F> | Поиск текста |

***Таблица П2.1*** *(продолжение)*

| Меню | Команда | Перевод | Горячая клавиша или комбинация клавиш | Описание |
|---|---|---|---|---|
| | **Replace** | Заменить | <Ctrl>+<H> | Замена искомого текста другим |
| | **Go to Page** | Перейти к странице | | Переход к другой странице |
| | **Links** | Ссылки | | Управление связями OLE с другими приложениями |
| | **Object** | Объект | | Активизировать вставленный объект OLE |
| **View** (Вид) | **Toolbars** | Панели инструментов | | Показать или скрыть панели инструментов |
| | **Ruler** | Линейка | | Показать или скрыть линейку |
| | **Status Bar** | Строка состояния | | Показать или скрыть строку состояния |
| | **Trace window** | Окно трассировки | | Показать или скрыть окно трассировки |
| | **Header and Footer** | Верхний и нижний колонтитулы | | Изменить колонтитулы для распечатываемых страниц |
| | **Regions** | Регионы | | Показать или скрыть границы регионов |
| | **Annotations** | Примечания | | Показать или скрыть примечания |
| | **Refresh** | Обновить | <Ctrl>+<R> | Обновить документ |
| | **Zoom** | Масштаб | | Изменить масштаб отображения документа |
| **Insert** (Вставка) | **Graph** | График | | Вставить график (с выбором типа графика из подменю) |
| | **Matrix** | Матрица | <Ctrl>+<M> | Вставить матрицу или вектор |

***Таблица П2.1*** *(продолжение)*

| Меню | Команда | Перевод | Горячая клавиша или комбинация клавиш | Описание |
|---|---|---|---|---|
| | **Function** | Функция | <Ctrl>+<E> | Вставить встроенную функцию |
| | **Unit** | Единицы | <Ctrl>+<U> | Вставить единицы измерения размерной величины |
| | **Picture** | Рисунок | <Ctrl>+<T> | Создать рисунок для отображения матрицы |
| | **Area** | Область | | Создать зону |
| | **Page Break** | Разрыв страницы | | Начать новую страницу |
| | **Math Region** | Математическая область | <Ctrl>+ +<Shift>+<A> | Создать математическую область в тексте |
| | **Text Region** | Текстовая область | <"> | Создать текстовую область в документе |
| | **Component** | Компонент | | Вставить компонент другого приложения |
| | **Data** | Данные | | Вставить данные в различных форматах |
| | **Control** | Элемент управления | | Вставить элемент управления |
| | **Object** | Объект | | Внедрение объекта |
| | **Reference** | Ссылка | | Вставить ссылку на другой документ |
| | **Hyperlink** | Гиперссылка | <Ctrl>+<K> | Вставить гиперссылку |
| **Format** (Формат) | **Equation** | Формула | | Форматирование формул |
| | **Result** | Результат | | Форматирование вывода результатов вычислений |

***Таблица П2.1*** *(продолжение)*

| Меню | Команда | Перевод | Горячая клавиша или комбинация клавиш | Описание |
|---|---|---|---|---|
| | **Text** | Текст | | Форматирование текста |
| | **Paragraph** | Абзац | | Изменение разметки абзаца |
| | **Tabs** | Табуляции | | Установить табуляцию для документа или выделенного участка текста |
| | **Style** | Стиль текста | | Определить или применить стиль — комбинацию настроек текстового формата |
| | **Properties** | Свойства | | Изменение свойств области |
| | **Graph** | График | | Изменения в графиках |
| | **Color** | Цвет | | Настройка цвета |
| | **Area** | Область | | Работа с зоной |
| | **Separate Regions** | Разделить регионы | | Разделить перекрывающиеся регионы в документе |
| | **Align Regions** | Выровнять регионы | | Выравнивание региона по горизонтали или вертикали |
| | **Repaginate Now** | Разбить на страницы | | Разбиение документа на страницы |
| **Tools** (Сервис) | **Spelling** | Проверка орфографии | | Проверка орфографии текстовых регионов |
| | **Animation** | Анимация | | Создать или воспроизвести анимацию |
| | **Protect Worksheet** | Запереть документ | | Защита документа от редактирования |

***Таблица П2.1*** *(продолжение)*

| Меню | Команда | Перевод | Горячая клавиша или комбинация клавиш | Описание |
|---|---|---|---|---|
| | **Calculate** | Пересчитать | <F9> | Управление вычислением формул |
| | **Optimize** | Оптимизировать | | Управление режимом оптимизации расчетов |
| | **Debug** | Отладка | | Управление отладкой программ |
| | **Disable Evaluation** | Отключить вычисления | | Включить/выключить вычисление формулы |
| | **Trace Error** | Трассировка ошибки | | Трассировка источника сообщения об ошибке |
| | **Worksheet Options** | Опции документа | | Установка опций математики |
| | **License** | Лицензия | | Параметры лицензирования копии Mathcad |
| | **Preferences** | Настройки | | Изменить основные настройки |
| **Symbolics** (Символика) | **Evaluate** | Вычислить | | Вычислить выражение в виде числа, если это возможно |
| | **Simplify** | Упростить | | Упростить выражение |
| | **Expand** | Разложить | | Представить выражение в более развернутом виде |
| | **Factor** | Разложить на множители | | Разложить полином или целое число на простые множители |
| | **Collect** | Привести подобные | | Привести подобные слагаемые |
| | **Polynomial Coefficients** | Коэффициенты полинома | | Вычислить полиномиальные коэффициенты |

***Таблица П2.1*** *(продолжение)*

| Меню | Команда | Перевод | Горячая клавиша или комбинация клавиш | Описание |
|---|---|---|---|---|
| | **Variable** | Переменная | | Символьные действия с выделенной переменной |
| | **Matrix** | Матрица | | Символьные действия с матрицей |
| | **Transform** | Преобразование | | Символьные интегральные преобразования |
| | **Evaluation Style** | Стиль вычислений | | Изменить показ символьных ответов |
| **Window** (Окно) | **Cascade** | Каскад | | Расположить окна документов каскадом |
| | **Tile Horizontal** | По горизонтали | | Расположить окна документов по горизонтали |
| | **Tile Vertical** | По вертикали | | Расположить окна документов по вертикали |
| | * | (Имя документа) | | Активизировать окно |
| **Help** (Справка) | **Mathcad Help** | Справка | <F1> | Получение справочной информации |
| | **What's This?** | Что это такое? | | Быстрая интерактивная справка об элементах интерфейса |
| | **Developer's Reference** | Справка для разработчиков | | Дополнительная справка для разработчиков |
| | **Author's Reference** | Справка для авторов | | Дополнительная справка для авторов электронных книг |
| | **Tutorials** | Учебники | | Доступ к электронным книгам учебников |

***Таблица П2.1*** *(окончание)*

| Меню | Команда | Перевод | Горячая клавиша или комбинация клавиш | Описание |
|---|---|---|---|---|
| | **QuickSheets** | Быстрые шпаргалки | | Доступ к электронным книгам Быстрых шпаргалок |
| | **Reference Tables** | Справочный стол | | Доступ к электронной книге со справочными таблицами |
| | **E-Books** | Электронные книги | | Открыть существующую в виде файла электронную книгу или пакет расширения |
| | **User's Forum** | Форум | | Перейти на форум пользователей Mathcad |
| | **Mathcad.com** | Mathcad.com | | Перейти на сайт Mathcad |
| | **Mathcad Update** | Обновление Mathcad | | Проверить сайт компании MathSoft на наличие обновлений версии Mathcad 13 |
| | **About Mathcad** | О программе | | Информация о текущей версии Mathcad |
| | **Register Mathcad** | Регистрация | | Онлайн-регистрация копии Mathcad |

***Таблица П2.2.*** *Панель* ***Math*** *(Математика)*

| Панель | Перевод |
|---|---|
| **Calculator** | Калькулятор |
| **Graph** | График |
| **Matrix** | Матрица |
| **Evaluation** | Выражения |

***Таблица П2.2*** *(окончание)*

| Панель | Перевод |
|---|---|
| **Calculus** | Вычисления |
| **Boolean** | Булевы операторы |
| **Programming** | Программирование |
| **Greek Symbols** | Греческие символы |
| **Symbolic Keyword** | Символика |

***Таблица П2.3.*** *Панель* ***Calculator*** *(Калькулятор)*

| Кнопка панели инструментов | Перевод | Горячая клавиша или комбинация клавиш |
|---|---|---|
| **sin** | Синус | |
| **cos** | Косинус | |
| **tan** | Тангенс | |
| **ln** | Натуральный логарифм | |
| **log** | Десятичный логарифм | |
| **n!** | Факториал | <!> |
| **i** | Ввод мнимой единицы | <1>, <i > |
| **\|x\|** | Модуль | <Shift>+<\> |
| $\sqrt{\ }$ | Квадратный корень | <\> |
| $\sqrt[n]{\ }$ | Корень *n*-ой степени | <Ctrl>+<\> |
| $e^x$ | Экспонента в *n*-ой степени | |
| **1/x** | Обратная величина | |
| **( )** | Скобки | <'> |
| $x^2$ | Возведение в квадрат | |
| $x^y$ | Возведение в степень *y* | <^> |
| $\pi$ | Ввод числа $\pi$ | <Ctrl>+<Shift>+<P> |
| **/** | Деление | </> |
| $\times$ | Умножение | <*> |
| $\div$ | Деление в одну строку | <Ctrl>+</> |
| **+** | Сложение | <+> |

***Таблица П2.3*** *(окончание)*

| Кнопка панели инструментов | Перевод | Горячая клавиша или комбинация клавиш |
|---|---|---|
| **:=** | Присваивание | <:> |
| **.** | Десятичная точка | <.> |
| **0, 1, 2, …, 9** | Числа 0—9 | <0>,<1>, <2>, …, <9> |
| **–** | Вычитание ("минус") | <-> |
| **=** | Вычислить численно ("равно") | <=> |

***Таблица П2.4.*** *Панель* ***Graph*** *(График)*

| Кнопка панели инструментов | Перевод | Горячая клавиша или комбинация клавиш |
|---|---|---|
| **XY Plot** | XY (декартовый) график | <Shift>+<2> |
| **Zoom** | Масштаб графика | |
| **Trace** | Трассировка графика | |
| **Polar Plot** | Полярный график | <Ctrl>+<7> |
| **Surface Plot** | График трехмерной поверхности | <Ctrl>+<2> |
| **Contour Plot** | График линий уровня | <Ctrl>+<5> |
| **3D Bar Plot** | Трехмерная гистограмма | |
| **Vector Field Plot** | Векторное поле | |
| **3D Scatter Plot** | Трехмерное множество точек | |

***Таблица П2.5.*** *Панель* ***Matrix*** *(Матрица)*

| Кнопка панели инструментов | Перевод | Горячая клавиша или комбинация клавиш |
|---|---|---|
| **Matrix or Vector** | Матрица или вектор | <Ctrl>+<M> |
| **Subscript** | Нижний индекс | <[> |
| **Inverse** | Обратная матрица | |
| **Determinant** | Определитель | <\|><Shift>+<\> |
| **Vectorize** | Векторизовать | <Ctrl>+<-> |
| **Matrix Column** | Выделение столбца | <Ctrl>+<6> |

***Таблица П2.5*** *(окончание)*

| Кнопка панели инструментов | Перевод | Горячая клавиша или комбинация клавиш |
|---|---|---|
| **Matrix Transpose** | Транспонирование | <Ctrl>+<1> |
| **Range Variable** | Ранжированная переменная | <;> |
| **Cross Product** | Векторное произведение | <Ctrl>+<8> |
| **Dot Product** | Умножение | <*> |
| **Vector Sum** | Сумма вектора | <Ctrl>+<4> |
| **Picture** | Рисунок | <Ctrl>+<T> |

***Таблица П2.6.*** *Панель* ***Evaluation*** *(Выражения)*

| Оператор и кнопка панели инструментов | Перевод | Горячая клавиша или комбинация клавиш |
|---|---|---|
| **Evaluate Numerically =** | Вычислить численно ("равно") | <=> |
| **Definition :=** | Присваивание | <:> |
| **Global Definition ≡** | Глобальное присваивание | <~> |
| **Evaluate Symbolically →** | Вычислить символьно | <Ctrl>+<.> |
| **Symbolic Keyword Evaluation • →** | Символьное вычисление с ключевым словом | <Ctrl>+<Shift>+<.> |
| **Prefix Operator fx** | Оператор "перед" | |
| **Postfix Operator xf** | Оператор "после" | |
| **Infix Operator xfy** | Оператор "внутри" | |
| **Tree Operator $x^{f}_{y}$** | Оператор "дерево" | |

***Таблица П2.7.*** *Панель* ***Calculus*** *(Вычисления)*

| Оператор | Перевод | Горячая клавиша или комбинация клавиш |
|---|---|---|
| **Derivative** | Производная | <?> |
| **Nth Derivative** | *n*-я производная | <Ctrl>+<?> |

***Таблица П2.7*** *(окончание)*

| Оператор | Перевод | Горячая клавиша или комбинация клавиш |
|---|---|---|
| **Infinity** | Символ бесконечности | <Ctrl>+<Shift>+<Z> |
| **Definite Integral** | Определенный интеграл | <&> |
| **Summation** | Сумма | <Ctrl>+<Shift>+<4> |
| **Iterated product** | Произведение | <Ctrl>+<Shift>+<3> |
| **Indefinite Integral** | Неопределенный интеграл | <Ctrl>+<I> |
| **Summation with range variables** | Сумма ранжированной переменной | <Ctrl>+<4> |
| **Iterated product with range variables** | Произведение ранжированной переменной | <Ctrl>+<3> |
| **Two-sided limit** | Предел | <Ctrl>+<L> |
| **Left-sided limit** | Левый предел | <Ctrl>+<A> |
| **Right-sided limit** | Правый предел | <Ctrl>+<B> |

***Таблица П2.8.*** *Панель* ***Boolean*** *(Булевы операторы)*

| Кнопка панели инструментов и оператор | Перевод | Горячая клавиша или комбинация клавиш |
|---|---|---|
| **= equal** | Равно | <Ctrl>+<=> |
| **< less than** | Меньше | <<> |
| **> greater than** | Больше | <>> |
| **≤ less than or equal** | Меньше или равно | <Ctrl>+<9> |
| **≥ greater than or equal** | Больше или равно | <Ctrl>+<0> |
| **≠ not equal to** | Не равно | <Ctrl>+<3> |
| **¬ Not** | Не | <Ctrl>+<Shift>+<1> |
| **∧ And** | И | <Ctrl>+<Shift>+<7> |
| **∨ Or** | ИЛИ | <Ctrl>+<Shift>+<6> |
| **⊕ Exclusive or** | Исключающее ИЛИ | <Ctrl>+<Shift>+<5> |

***Таблица П2.9.*** *Панель* ***Controls*** *(Элементы управления)*

| Элемент управления | Перевод |
|---|---|
| **Check Box** | Флажок проверки |
| **Radio Button** | Переключатель |
| **Push Button** | Кнопка |
| **Slider** | Ползунковый регулятор |
| **Text Box** | Поле текстового ввода |
| **List Box** | Список |

# Приложение 3

# Встроенные операторы и функции

Встроенные операторы и функции описаны в табл. П3.1—П3.4.

***Таблица П3.1.*** *Арифметические операторы*

| Оператор | Клавиша | Скаляр | Вектор | Матрица |
|---|---|---|---|---|
| := | <:> | Присваивание | | |
| ≡ | <~> | Глобальное присваивание | | |
| = | <=> | Численный вывод | | |
| → | <Ctrl>+<=> | Символьный вывод | | |
| + | <+> | Сложение | | |
| − | <-> | Вычитание или отрицание (унарная операция) | | |
| . | <*> | Умножение | Матричное умножение, умножение на скаляр | |
| | | | Скалярное произведение | |
| × | <Ctrl>+<8> | | Векторное произведение | |
| / либо ÷ | </> либо <Ctrl>+</> | Деление | | |
| ! | <!> | Факториал | | |
| ‾ | <"> | Комплексное сопряжение | | |
| √‾ | <\> | Квадратный корень | | |
| ∜‾ | <Ctrl>+<\> | Корень степени *m* | | |
| (■) | <'> | Скобки (изменение приоритета) | | |
| ■■ | <[> | | Нижний индекс | |
| ■T | <Ctrl>+<1> | | Транспонирование | |

***Таблица П3.1*** *(окончание)*

| Оператор | Клавиша | Скаляр | Вектор | Матрица |
|---|---|---|---|---|
| $\lvert\blacksquare\rvert$ | <Shift>+<\> | Модуль | Модуль вектора | Определитель |
| $\sum\blacksquare$ | <Ctrl>+<4> | | Сумма элементов | |
| $\blacksquare^{-1}$ | | Обратная величина | | Обратная матрица |
| $\blacksquare^{n}$ | <^>+<*n*> | Возведение в степень *n* | | Возведение матрицы в степень *n* |
| $\rightarrow$ | <Ctrl>+<-> | | Векторизация | |
| $\blacksquare^{<\blacksquare>}$ | <Ctrl>+<6> | | Выделение столбца | |

### *Примечание*

Скалярные операции над векторами и матрицами, если это не оговорено особо, производятся независимо над их каждым элементом, как над скаляром.

***Таблица П3.2.*** *Вычислительные операторы*

| Оператор | Клавиша | Описание |
|---|---|---|
| $\int_{\blacksquare}^{\blacksquare}\blacksquare\,d\blacksquare$ | <Shift>+<7> | Определенный интеграл |
| $\int\blacksquare\,d\blacksquare$ | <Ctrl>+<I> | Неопределенный интеграл |
| $\frac{d}{d\blacksquare}\blacksquare$ | <?> | Дифференцирование |
| $\frac{d^{n}}{d\blacksquare^{n}}\blacksquare$ | <Ctrl>+<?> | Вычисление *n*-ой производной |
| $\sum_{\blacksquare=\blacksquare}^{\blacksquare}\blacksquare$ | <Ctrl>+<Shift >+<4> | Сумма |

***Таблица П3.2*** *(окончание)*

| Оператор | Клавиша | Описание |
|---|---|---|
| $\sum_{\blacksquare}^{\blacksquare}\blacksquare$ | <Ctrl>+<4> | Сумма ранжированной переменной |
| $\prod_{\blacksquare=\blacksquare}^{\blacksquare}\blacksquare$ | <Ctrl>+<Shift>+<3> | Произведение |
| $\prod_{\blacksquare}^{\blacksquare}\blacksquare$ | <Ctrl>+<3> | Произведение ранжированной переменной |
| $\lim_{\blacksquare\to\blacksquare}\blacksquare$ | <Ctrl>+<L> | Предел |
| $\lim_{\blacksquare\to\blacksquare^{-}}\blacksquare$ | <Ctrl>+<A> | Левый предел |
| $\lim_{\blacksquare\to\blacksquare^{+}}\blacksquare$ | <Ctrl>+<B> | Правый предел |

***Таблица П3.3.*** *Встроенные функции*

| Функция | Аргументы | Описание |
|---|---|---|
| `a*(z)` | `z` — аргумент | Обратная тригонометрическая или гиперболическая функция |
| `Ai(x)` | `x` — аргумент | Функция Эйри первого рода |
| `angle(x,y)` | `x,y` — координнаты точки | Угол между точкой и осью `OX` |
| `APPENDPRN (file)` | `file` — строковое представление пути к файлу | Дозапись данных в существующий текстовый файл |
| `arg(z)` | `z` — аргумент функции | Аргумент комплексного числа |
| `atan2(x,y)` | `x,y` — координнаты точки | Угол, отсчитываемый от оси `OX` до точки `(x,y)` |
| `augment (A,B,C,…)` | `A,B,C,…` — векторы или матрицы | Слияние матриц слева направо |
| `bei(n,x)` `ber(n,x)` | `n` — порядок<br>`x` — аргумент | Мнимая и действительная части функции Бесселя—Кельвина |
| `Bi(x)` | `x` — аргумент | Функция Эйри второго рода |

***Таблица П3.3*** *(продолжение)*

| Функция | Аргументы | Описание |
|---|---|---|
| `Bspline (x,y,u,n)` | `x,y` — векторы данных<br>`u` — вектор значений сшивок B-сплайнов<br>`n` — порядок полиномов | Вектор коэффициентов B-сплайна |
| `Bulstoer (y0,t0,t1, M,D)` | См. `rkfixed` | Возвращает матрицу с решением задачи Коши для системы ОДУ методом Булирша—Штера |
| `bulstoer (y0,t0,t1, acc,D,k,s)` | См. `rkadapt` | Возвращает матрицу с решением задачи Коши для системы ОДУ методом Булирша—Штера (для определения только последней точки интервала) |
| `Bvalfit (z1,z2,x0,x1, xf,D,load1, load2,score)` | `z1,z2` — вектор начальных значений для недостающих левых и правых граничных условий<br>`x0` — левая граница<br>`x1` — правая граница<br>`xf` — внутренняя точка<br>`D(x,y)` — векторная функция, задающая систему ОДУ<br>`load1(x0,z)`, `load2(x1,z)` — векторные функции, задающие левые и правые граничные условия<br>`score(xf,y)` — векторная функция, задающая сшивку решений в `xf` | Возвращает вектор недостающих граничных условий у краевой задачи для системы `N` ОДУ с дополнительным условием в промежуточной точке |
| `ceil(x)` | `x` — аргумент | Наименьшее целое, не меньшее `x` |
| `cfft(y),`<br>`CFFT(y)` | `y` — вектор данных | Вектор прямого комплексного преобразования Фурье (в разных нормировках) |
| `cholesky(A)` | `A` — квадратная, определенная матрица | Разложение Холецкого |
| `cols(A)` | `A` — матрица или вектор | Число столбцов |

***Таблица П3.3*** *(продолжение)*

| Функция | Аргументы | Описание |
|---|---|---|
| `concat(S1,S2,…)` | `S1`, `S2`,… — строки | Объединение строковых переменных |
| `cond1(A)`, `cond2(A)`, `conde(A)`, `condi(A)` | `A` — квадратная матрица | Числа обусловленности в разных нормах (`L1`, `L2`, евклидова, ∞) |
| `cos(z)` | `z` — аргумент | Косинус |
| `cosh(z)` | `z` — аргумент | Гиперболический косинус |
| `cot(z)` | `z` — аргумент | Котангенс |
| `coth(z)` | `z` — аргумент | Гиперболический котангенс |
| `csort(A,i)` | `A` — матрица<br>`i` — индекс столбца | Сортировка строк матрицы по элементам `i`-го столбца |
| `CreateMesh(F,s0,s1,t0,t1,sgr,tgr,fmap)` | `F(s,t)` — векторная функция из трех элементов<br>`t0,t1` — пределы `t`<br>`s0,s1` — пределы `s`<br>`tgr,sgr` — число точек сетки по `t` и `s`<br>`fmap` — функция преобразования координат | Создание вложенного массива, представляющего `x`-, `y`- и `z`-координаты параметрической поверхности, заданной функцией `F` |
| `CreateSpace(F[,t0,t1,tgr,fmap])` | `F(t)` — векторная функция из трех элементов<br>`t0,t1` — пределы `t`<br>`tgr` — число точек сетки по `t`<br>`fmap` — функция преобразования координат | Создание вложенного массива, представляющего `x`-, `y`- и `z`-координаты параметрической пространственной кривой, заданной функцией `F` |
| `csc(z)` | `z` — аргумент | Косеканс |
| `csch(z)` | `z` — аргумент | Гиперболический косеканс |
| `csgn(z)` | `z` — аргумент | Комплексный знак числа |
| `cspline(x,y)` | `x,y` — векторы данных | Вектор коэффициентов кубического сплайна |

***Таблица П3.3*** *(продолжение)*

| Функция | Аргументы | Описание |
|---|---|---|
| `cyl2xyz (r,θ,z)` | `r,θ,z` — цилиндрические координаты | Преобразование цилиндрических координат в прямоугольные |
| `D* (x,par)` | `x` — значение случайной величины<br>`par` — список параметров распределения | Плотность вероятности со статистикой распределения |
| `diag(v)` | `v` — вектор | Диагональная матрица, на диагонали которой находятся элементы вектора |
| `eigenvals(A)` | `A` — квадратная матрица | Собственные значения матрицы |
| `eigenvec(A,λ)` | `A` — квадратная матрица<br>`λ` — собственное значение | Собственный вектор матрицы, соответствующий заданному собственному значению |
| `eigenvecs(A)` | `A` — квадратная матрица | Собственные векторы матрицы |
| `erf(x)` | `x` — аргумент | Функция ошибок |
| `erfc(x)` | `x` — аргумент | Обратная функция ошибок |
| `error(S)` | `S` — строка | Возвращает строку `S` как сообщение об ошибке |
| `exp(z)` | `z` — аргумент | Экспонента в степени `z` |
| `expfit(x,y,g)` | `x,y` — векторы данных<br>`g` — вектор начальных значений `a`, `b`, `c` | Регрессия экспонентой $a \cdot e^{bx}+c$ |
| `fft(y)`,<br>`FFT(y)` | `y` — вектор данных | Вектор прямого преобразования Фурье (в разных нормировках) |
| `fhyper (a,b,c,x)` | `a,b,c` — параметры<br>`x` — аргумент, `-1<x<1` | Гауссова гипергеометрическая функция |
| `Find(x1,x2,…)` | `x1,x2,…` — переменные | Возвращает корень алгебраического уравнения (скаляр) или системы (вектор), определенных в блоке с ключевым словом `Given` |
| `floor(x)` | `x` — аргумент | Наибольшее целое число, меньшее или равное `x` |

***Таблица П3.3*** *(продолжение)*

| Функция | Аргументы | Описание |
|---|---|---|
| `Gamma(x)`,<br>`Gamma(a,x)` | `x` — аргумент | Гамма-функция Эйлера или неполная гамма-функция порядка `a` |
| `Genfit (x,y,g,G)` | `x,y` — векторы данных<br>`g` — вектор начальных значений параметров регрессии<br>`G(x,C)` — векторная функция, составленная из функции пользователя и ее частных производных по каждому параметру | Вектор коэффициентов регрессии функциями пользователя общего вида |
| `geninv(A)` | `A` — матрица | Создание обратной матрицы |
| `genvals(A,B)` | `A,B` — квадратные матрицы | Расчет обобщенных собственных значений |
| `genvecs(A,B)` | `A,B` — квадратные матрицы | Расчет обобщенных собственных векторов |
| `Given` | | Ключевое слово для ввода систем уравнений, неравенств и т. п. |
| `heaviside step(x)` | `x` — аргумент | Функция Хевисайда |
| `Her(n,x)` | `x` — аргумент<br>`n` — порядок | Полином Эрмита |
| `I0(x)`,<br>`I1(x)`,<br>`In(m,x)` | `x` — аргумент | Модифицированная функция Бесселя первого рода нулевого, первого и *m*-го порядка |
| `ibeta(a,x,y)` | `x,y` — аргументы<br>`a` — параметр | Неполная бета-функция |
| `identity(N)` | `N` — размер матрицы | Создание единичной матрицы |
| `icfft(v)`,<br>`ICFFT(v)` | `v` — вектор частотных данных Фурье-спектра | Вектор комплексного обратного преобразования Фурье (в разных нормировках) |
| `if(cond,x,y)` | `cond` — логическое условие<br>`x,y` — значения, возвращаемые, если условие верно (ложно) | Функция условия |

***Таблица П3.3*** *(продолжение)*

| Функция | Аргументы | Описание |
|---|---|---|
| `ifft(v)`, `IFFT(v)` | `v` — вектор частотных данных Фурье-спектра | Вектор обратного преобразования Фурье (в разных нормировках) |
| `isNaN(x)` | `x` — аргумент | Возвращает `1`, если `x=NaN` и `0` в остальных случаях |
| `lsString(x)` | `x` — аргумент | Возвращает `1`, если `x` — строка, и `0` в остальных случаях |
| `iwave(v)` | `v` — вектор частотных данных вейвлет-спектра | Вектор обратного вейвлет-преобразования |
| `Im(z)` | `z` — аргумент | Мнимая часть комплексного числа |
| `interp(s,x,y,t)` | `s` — вектор вторых производных<br>`x,y` — векторы данных<br>`t` — аргумент | Сплайн-интерполяция |
| `intercept(x,y)` | `x,y` — векторы данных | Коэффициент `b` линейной регрессии `b+a·x` |
| `J0(x)`, `J1(x)`, `Jn(m,x)` | `x` — аргумент | Функция Бесселя первого рода нулевого, первого и *m*-го порядка |
| `Jac(n,a,b,x)` | `x` — аргумент<br>`a,b` — параметры<br>`n` — порядок | Полином Якоби |
| `js(n,x)` | `n` — порядок<br>`x` — аргумент | Сферическая функция Бесселя первого рода |
| `K0(x)`, `K1(x)`, `Kn(m,x)` | `x` — аргумент | Модифицированная функция Бесселя второго рода нулевого, первого и *m*-го порядка |
| `Kronecker delta(x,y)` | `x,y` — аргументы | Дельта-символ Кронекера |
| `ksmooth(x,y,b)` | `x,y` — векторы данных<br>`b` — ширина окна сглаживания | Сглаживание с помощью функции Гаусса |

***Таблица П3.3*** *(продолжение)*

| Функция | Аргументы | Описание |
|---|---|---|
| `Lag(n,x)` | `x` — аргумент<br>`n` — порядок | Полином Лагерра |
| `last(v)` | `v` — вектор | Индекс последнего элемента вектора |
| `Leg(n,x)` | `x` — аргумент<br>`n` — порядок | Полином Лежандра |
| `length(v)` | `v` — вектор | Число элементов вектора |
| `line(x,y)` | `x,y` — векторы данных | Вектор из коэффициентов линейной регрессии `b+a·x` |
| `linfit(x,y,F)` | `x,y` — векторы данных<br>`F(x)` — векторная функция пользователя | Вектор коэффициентов регрессии функцией пользователя |
| `linterp(x,y,t)` | `x,y` — векторы данных<br>`t` — аргумент | Кусочно-линейная интерполяция |
| `lgsfit(x,y,g)` | `x,y` — векторы данных<br>`g` — вектор начальных значений `a`, `b`, `c` | Регрессия логистической функцией $a/(1+b\cdot e^{-cx})$ |
| `ln(z)` | `z` — аргумент | Натуральный логарифм |
| `lnfit(x,y)` | `x,y` — векторы данных | Регрессия логарифмической функцией `a·ln(x)+b` |
| `Loess (x,y,span)` | `x,y` — векторы данных<br>`span` — параметр размера полиномов | Вектор коэффициентов для регрессии отрезками полиномов (применяется вместе с `interp`) |
| `log(z)` | `z` — аргумент | Десятичный логарифм |
| `log(z, b)` | `z` — аргумент | Логарифм `z` по основанию `b` |
| `logfit(x,y,g)` | `x,y` — векторы данных<br>`g` — вектор начальных значений `a`, `b`, `c` | Регрессия логарифмической функцией `a·ln(x+b)+c` |
| `Logpts (min,dec,N)` | `min` — показатель начала интервала,<br>`dec` — количество декад<br>`N` — число точек в пределах каждой декады | Возвращает вектор из чисел, расположенных линейно-равномерно в пределах каждой логарифмической декады |

***Таблица П3.3*** *(продолжение)*

| Функция | Аргументы | Описание |
|---|---|---|
| `Logspace (min,max,N)` | `min,max` — границы интервала<br>`N` — число точек | Возвращает вектор из чисел, расположенных равномерно (в логарифмическом масштабе) на интервале `(min,max)` |
| `lsolve(A,b)` | `A` — матрица СЛАУ<br>`b` — вектор правых частей | Решение системы линейных уравнений (СЛАУ) |
| `lspline(x,y)` | `x,y` — векторы данных | Вектор коэффициентов линейного сплайна |
| `lu(A)` | `A` — квадратная матрица | LU-разложение |
| `matrix(M,N,f)` | `M` — количество строк<br>`N` — количество столбцов<br>`f(i,j)` — функция | Создание матрицы с элементами `f(i,j)` |
| `Maximize (f,x1,…)` | `f(x1,…)` — функция<br>`x1,…` — аргументы, по которым производится максимизация | Вектор значений аргументов, при которых функция `f` достигает максимума (возможно задание дополнительных условий в блоке с ключевым словом `Given`) |
| `mhyper(a,b,x)` | `x` — аргумент<br>`a,b` — параметры | Конфлюэнтная гипергеометрическая функция |
| `Minerr (x1,x2,…)` | `x1,x2,…` — переменные | Возвращает вектор приближенного решения системы уравнений и неравенств, определенных в блоке с ключевым словом `Given` |
| `Minimize (f,x1,…)` | `f(x1,…)` — функция<br>`x1,…` — аргументы, по которым производится минимизация | Вектор значений аргументов, при которых функция `f` достигает минимума (возможно задание дополнительных условий в блоке с ключевым словом `Given`) |
| `Medsmooth (y,b)` | `y` — вектор данных<br>`b` — ширина окна сглаживания | Сглаживание методом "бегущих медиан" |
| `Multigrid (F,ncycle)` | `F` — матрица правой части уравнения Пуассона<br>`ncycle` — параметр алгоритма сеток | Матрица решения уравнения Пуассона на квадратной области с нулевыми граничными условиями |

***Таблица П3.3*** *(продолжение)*

| Функция | Аргументы | Описание |
|---|---|---|
| `n*(M,par)` | `M` — размерность вектора<br>`x` — значение случайной величины<br>`par` — список параметров распределения | Вектор случайных чисел со статистикой |
| `norm1(A)`,<br>`norm2(A)`,<br>`norme(A)`,<br>`normi(A)` | `A` — квадратная матрица | Нормы матриц (`L1`, `L2`, евклидова, ∞) |
| `num2str(z)` | `z` — число | Возвращает строку, чьи знаки соответствуют десятичному значению числа `z` |
| `Odesolve(t,t1 [,step])` | `t` — переменная интегрирования ОДУ<br>`t1` — конечная точка интервала интегрирования<br>`step` — число шагов интегрирования ОДУ | Возвращает матрицу с решением задачи Коши для одного ОДУ, определенного в блоке с ключевым словом `Given` и начальными условиями в точке `t0` |
| `p*(x,par)` | `x` — значение случайной величины<br>`par` — список параметров распределения | Функция распределения со статистикой |
| `pdesolve(u,x, xrange,t, trange, [xpts], [tpts]))` | `u` — вектор имен функций<br>`x` — пространственная переменная<br>`xrange` — интервал интегрирования по пространству<br>`t` — временная переменная<br>`trange` — интервал интегрирования по времени<br>`xpts` — число пространственных узлов сетки<br>`tpts` — число временных шагов сетки | Возвращает скалярную функцию двух аргументов (`x`,`t`), являющуюся решением дифференциального уравнения (или системы уравнений) в частных производных |

***Таблица П3.3*** *(продолжение)*

| Функция | Аргументы | Описание |
|---|---|---|
| `pol2xy(r,θ)` | `r,θ` — полярные координаты | Преобразование полярных координат в прямоугольные |
| `polyroots(v)` | `v` — вектор, составленный из коэффициентов полинома | Возвращает вектор всех корней полинома |
| `Predict (y,m,n)` | `y` — исходный вектор<br>`m` — число элементов `y`, по которым строится экстраполяция<br>`n` — количество предсказываемых элементов | Функция предсказания, экстраполирующая вектор |
| `pspline(x,y)` | `x,y` — векторы данных | Вектор коэффициентов квадратичного сплайна |
| `pwfit(x,y,g)` | `x,y` — векторы данных<br>`g` — вектор начальных значений `a,b,c` | Регрессия степенной функцией $a \cdot x^b + c$ |
| `q*(p,par)` | `p` — значение вероятности<br>`par` — список параметров распределения* | Квантиль (функция, обратная функции распределения) со статистикой* |
| `qr(A)` | `A` — вектор или матрица | QR-разложение |
| `Radau(y0,t0,t1, M,D)` | См. `rkfixed` | Возвращает матрицу с решением задачи Коши для жесткой системы ОДУ методом RADAUS |
| `radau(y0,t0,t1, M,D)` | См. `rkfixed` | Возвращает матрицу с решением задачи Коши для жесткой системы ОДУ методом RADAUS (для определения только последней точки интервала) |
| `rank(A)` | `A` — матрица | Ранг матрицы |
| `Re(z)` | `z` — аргумент | Действительная часть комплексного числа |
| `READ*(file)` | `file` — строковое представление пути к файлу | Запись данных в файл типа |
| `Regress (x,y,k)` | `x,y` — векторы данных<br>`k` — степень полинома | Вектор коэффициентов для полиномиальной регрессии (применяется вместе с `interp`) |

***Таблица П3.3*** *(продолжение)*

| Функция | Аргументы | Описание |
|---|---|---|
| `Relax (a,b,c,d,e,F, v,rjac)` | `a,b,c,d,e` — матрицы коэффициентов разностной схемы<br>`F` — матрица правой части уравнения<br>`v` — матрица граничных условий<br>`rjac` — параметр алгоритма (0…1) | Матрица решения методом сеток дифференциального уравнения в частных производных на квадратной области |
| `reverse(v)` | `v` — вектор | Перестановка элементов вектора в обратном порядке |
| `Rkadapt (y0,t0,t1, acc,D,k,s)` | `y0` — вектор начальных условий<br>`(t0,t1)` — интервал интегрирования<br>`acc` — погрешность вычисления<br>`D(t,y)` — векторная функция, задающая систему ОДУ<br>`k` — максимальное число шагов интегрирования<br>`s` — минимальный шаг интегрирования | Возвращает матрицу с решением задачи Коши для системы ОДУ методом Рунге—Кутты с переменным шагом и заданной точностью (для определения только последней точки интервала) |
| `Rkadapt (y0,t0,t1, M,D)` | См. `rkfixed` | Возвращает матрицу с решением задачи Коши для системы ОДУ методом Рунге—Кутты с переменным шагом |
| `rkfixed (y0,t0,t1, M,D)` | `y0` — вектор начальных условий<br>`(t0,t1)` — интервал интегрирования<br>`M` — число шагов интегрирования<br>`D(t,y)` — векторная функция, задающая систему ОДУ | Возвращает матрицу с решением задачи Коши для системы ОДУ методом Рунге—Кутты с фиксированным шагом |

***Таблица П3.3*** *(продолжение)*

| Функция | Аргументы | Описание |
|---|---|---|
| `root (f(x,…), x[a,b])` | `f(x,…)` — функция<br>`x` — переменная<br>`(a,b)` — интервал поиска корня | Возвращает корень функции |
| `round(x, n)` | `x` — аргумент<br>`n` — число знаков округления после десятичной точки | Округление |
| `rows(A)` | `A` — матрица или вектор | Число строк |
| `rref(A)` | `A` — матрица или вектор | Преобразование матрицы в ступенчатый вид |
| `rsort(A,i)` | `A` — матрица<br>`i` — индекс строки | Сортировка матрицы по элементам `i`-й строки |
| `sbval(z,x0, x1,D,load, score)` | `z` — вектор начальных приближений для недостающих начальных условий<br>`x0` — левая граница<br>`x1` — правая граница<br>`D(x,y)` — векторная функция, задающая систему ОДУ<br>`load(x0,z)` — векторная функция с начальными условиями<br>`score(x1,y)` — векторная функция, задающая правые граничные условия | Возвращает вектор недостающих начальных условий для двухточечной краевой задачи для системы ОДУ |
| `Search (S,Subs,m)` | `S` — строка<br>`Sub` — подстрока<br>`m` — стартовая позиция поиска | Стартовая позиция подстроки в строке |
| `sec(z)` | `z` — аргумент | Секанс |
| `sech(z)` | `z` — аргумент | Гиперболический секанс |
| `sign(x)` | `x` — аргумент | Знак числа |

***Таблица П3.3*** *(продолжение)*

| Функция | Аргументы | Описание |
|---|---|---|
| `signum(z)` | `z` — аргумент | Комплексный знак числа `z/\|z\|` |
| `sin(z)` | `z` — аргумент | Синус |
| `sinh(z)` | `z` — аргумент | Гиперболический синус |
| `sinfit(x,y,g)` | `y` — векторы данных<br>`g` — вектор начальных значений `a,b,c` | Регрессия синусоидой `f(x)=a·sin(x+b)+c` |
| `sinc(z)` | `z` — аргумент | Sinc-функция |
| `SIUnitsOf(x)` | `x` — аргумент | Возвращает единицу измерения в системе СИ |
| `slope(x,y)` | `x,y` — векторы данных | Коэффициент `a` линейной регрессии `b+a·x` |
| `sort(v)` | `v` — вектор | Сортировка элементов вектора |
| `sph2xyz(r,θ,ϕ)` | `r,θ,ϕ` — сферические координаты | Преобразование сферических координат в прямоугольные |
| `Stack (A,B,C,…)` | `A,B,C,…` — векторы или матрицы | Слияние матриц сверху вниз |
| `Stiffb(y0,t0, t1,M,D,J)` | См. `rkfixed`<br>`J(t,y)` — матричная функция Якоби для `D(t,y)` | Возвращает матрицу с решением задачи Коши для жесткой системы ОДУ методом Булирша—Штера |
| `stiffb(y0,t0, t1,acc,D,J, k,s)` | См. `rkadapt`<br>`J(t,y)` — матричная функция Якоби для `D(t,y)` | Возвращает матрицу с решением задачи Коши для жесткой системы ОДУ методом Булирша—Штера (для определения только последней точки интервала) |
| `Stiffr(y0,t0, t1,M,D,J)` | См. `Stiffb` | Возвращает матрицу с решением задачи Коши для жесткой системы ОДУ методом Розенброка |
| `stiffr(y0,t0, t1,acc,D,J, k,s)` | См. `stiffb` | Возвращает матрицу с решением задачи Коши для жесткой системы ОДУ методом Розенброка (для определения только последней точки интервала) |
| `str2num(S)` | `S` — строка | Преобразование строкового представления в действительное число |

***Таблица П3.3*** *(продолжение)*

| Функция | Аргументы | Описание |
|---|---|---|
| `str2vec(S)` | `S` — строка | Преобразование строкового представления в вектор ASCII-кодов |
| `strlen(S)` | `S` — строка | Количество знаков в строке |
| `submatrix(A,ir,jr,ic,jc)` | `A` — матрица<br>`ir,jr` — строки<br>`ic,jc` — столбцы | Возвращает часть матрицы, находящуюся между `ir,jr`-строками и `ic,jc`-столбцами |
| `substr(S,m,n)` | `S` — строка | Подстрока, полученная из строки `S` выделением `n` знаков, начиная с позиции `m` в строке `S` |
| `Supsmooth (x,y)` | `x,y` — векторы данных | Сглаживание с помощью адаптивного алгоритма |
| `svd(A)` | `A` — действительная матрица | Сингулярное разложение |
| `svds(A)` | `A` — действительная матрица | Вектор, состоящий из сингулярных чисел |
| `tan(z)` | `z` — аргумент | Тангенс |
| `tanh(z)` | `z` — аргумент | Гиперболический тангенс |
| `Tcheb(n,x)` | `x` — аргумент<br>`n` — порядок | Полином Чебышева первого рода |
| `time(x)` | `x` — аргумент | Значение системной константы текущего времени |
| `tr(A)` | `A` — квадратная матрица | След матрицы |
| `trunc(x)` | `x` — аргумент | Целая часть числа |
| `Ucheb(n,x)` | `x` — аргумент<br>`n` — порядок | Полином Чебышева второго рода |
| `vec2str(v)` | `v` — вектор ASCII-кодов | Строковое представление элементов вектора `v` |
| `wave(y)` | `y` — вектор данных | Вектор прямого вейвлет-преобразования |
| `WRITE*(file)` | `file` — строковое представление пути к файлу | Запись данных в файл типа |
| `xy2pol(x,y)` | `x,y` — прямоугольные координаты на плоскости | Преобразование прямоугольных координат в полярные |

***Таблица П3.3*** *(окончание)*

| Функция | Аргументы | Описание |
|---|---|---|
| `xyz2cyl (x,y,z)` | `x,y,z` — прямоугольные координаты | Преобразование прямоугольных координат в цилиндрические |
| `xyz2sph (x,y,z)` | `x,y,z` — прямоугольные координаты | Преобразование прямоугольных координат в сферические |
| `Y0(x)`, `Y1(x)`, `Yn(m,x)` | `x` — аргумент, `x>0` | Функция Бесселя второго рода нулевого, первого и *m*-го порядка |
| `ys(n,x)` | `n` — порядок<br>`x` — аргумент | Сферическая функция Бесселя второго рода |

### *Примечание*

Некоторые функции, составляющие семейства типовых функций, приведены в сокращенном виде с недостающей частью имени в виде звездочки "*". Например, различные статистические функции, описывающие различные распределения или функции вывода в файлы. Подробные сведения содержатся в разделе, на который указывает соответствующая ссылка. Специальных функций комплексного аргумента с измененной нормировкой, а также финансовых функций (см. табл. П3.4) в данном списке нет.

***Таблица П3.4.*** *Встроенные функции финансового анализа*

| Функция | Аргументы | Описание |
|---|---|---|
| `cnper(rate, pv,fv)` | `rate` — фиксированный процент по вкладу, `rate>-1`<br>`pv` — текущее значение вклада, `pv>0`<br>`fv` — будущее значение вклада, `fv>0` | Отвечает числу составных периодов, необходимых для получения будущего значения вклада при заданных текущем значении вклада и проценте начислений |
| `crate(nper, pv,fv)` | `nper` — целое число составных периодов, `nper≥1`<br>`pv` — текущее значение вклада, `pv>0`<br>`fv` — будущее значение вклада, `fv>0` | Отвечает фиксированному проценту начислений по вкладу на период, необходимый для прироста от текущего значения вклада до будущего значения при заданном числе составных периодов |

***Таблица П3.4*** *(продолжение)*

| Функция | Аргументы | Описание |
|---|---|---|
| `cumint(rate, nper,pv, start,end, [type])` | `rate` — фиксированный процент по вкладу, должен быть действительным скаляром, `rate≥0`<br>`nper` — общее число составных периодов, должно быть положительным целым числом<br>`pv` — текущее значение займа, `pv>0`<br>`start` — начальный период накопления; должен быть положительным целым числом<br>`end` — конечный период накопления, должен быть положительным целым числом, `start≤end`<br>`type=0` для платежа, сделанного в конце периода, или `1` для платежа, сделанного в начале периода | Отвечает совокупному проценту по займу между начальным и конечным периодами при фиксированном проценте, общем числе составных периодов и текущем значении займа |
| `cumprn(rate, nper,pv, start,end, [type])` | См. `cumint` | Отвечает совокупной сумме по займу между начальным и конечным периодами при фиксированном проценте, общем числе составных периодов и текущем значении займа |
| `eff(rate, nper)` | `rate` — номинальная процентная ставка, должна быть действительным скаляром | Отвечает эффективной ежегодной процентной ставке при данной номинальной ежегодной процентной ставке и числе составных периодов в год |
| `fv(rate,nper, pmt,[[pv], [type]])` | `rate` — фиксированная процентная ставка за период, должна быть действительным скаляром, `rate>0`<br>`nper` — общее число составных периодов в год, `nper >0` | Соответствует будущему значению вклада или займа через особое число составных периодов, установленных периодически, при постоянных платежах и фиксированной процентной ставке |

***Таблица П3.4*** *(продолжение)*

| Функция | Аргументы | Описание |
|---|---|---|
| `cnper(rate, pv,fv)` | `rate` — фиксированный процент по вкладу, должен быть действительным скаляром, `rate>-1`<br>`pv` — текущее значение вклада, `pv>0`<br>`fv` — будущее значение вклада, `fv>0` | Отвечает числу составных периодов, необходимых для получения будущего значения вклада при заданных текущем значении вклада и проценте начислений |
| `crate(nper, pv,fv)` | `nper` — число составных периодов, должно быть целым числом, `nper≥1`<br>`pv` — текущее значение вклада, `pv>0`<br>`fv` — будущее значение вклада, `fv>0` | Отвечает фиксированному проценту начислений по вкладу на период, необходимый для прироста от текущего значения вклада до будущего значения при заданном числе составных периодов |
| `cumint(rate, nper,pv, start,end, [type])` | `rate` — фиксированный процент по вкладу, должен быть действительным скаляром, `rate≥0`<br>`nper` — общее число составных периодов, должно быть положительным целым числом<br>`pv` — текущее значение займа, `pv>0`<br>`start` — начальный период накопления, должен быть положительным целым числом<br>`end` — конечный период накопления, должен быть положительным целым числом, `start≤end`<br>`type=0` для платежа, сделанного в конце периода, или `1` для платежа, сделанного в начале периода | Отвечает совокупному проценту по займу между начальным и конечным периодами при фиксированном проценте, общем числе составных периодов и текущем значении займа |
| `cumprn(rate, nper,pv, start,end, [type])` | См. `cumint` | Отвечает совокупной сумме по займу между начальным и конечным периодами при фиксированном проценте, общем числе составных периодов и текущем значении займа |

***Таблица П3.4*** *(продолжение)*

| Функция | Аргументы | Описание |
|---|---|---|
| `eff(rate, nper)` | `rate` — номинальная процентная ставка, должна быть действительным скаляром<br>`nper` — общее число составных периодов в год, `nper >0` | Отвечает эффективной ежегодной процентной ставке при данной номинальной ежегодной процентной ставке и числе составных периодов в год |
| `fv(rate,nper, pmt,[[pv], [type]])` | `rate` — фиксированная процентная ставка за период; должна быть действительным скаляром, `rate>0`<br>`nper` — общее число составных периодов в год, `nper>0`<br>`pv` — текущее значение займа<br>`type=0` для платежа, сделанного в конце периода, или `1` для платежа, сделанного в начале периода | Соответствует будущему значению вклада или займа через особое число составных периодов, установленных периодически, при постоянных платежах и фиксированной процентной ставке |
| `fvadj(prin,v)` | `prin` — ежегодная общая сумма<br>`v` — вектор процентных ставок, каждая из которых применяется с той же самой основной суммой и процентами с нее за период времени | Соответствует будущему значению ежегодной общей суммы капитала, на который начисляются проценты, при применении серии составных процентных ставок |
| `fvc(rate,v)` | `rate` — фиксированная процентная ставка за период; должна быть действительным скаляром<br>`v` — вектор регулярных денежных потоков | Соответствует будущему значению серии денежных потоков, происходящих с регулярными интервалами, и приносящими специальную процентную ставку |
| `ipmt(rate, per,nper,pv, [[fv],type]])` | `rate` — фиксированная процентная ставка за период, `rate≥0`<br>`per` — период, за который вы хотите найти ставку, должен быть положительным целым числом<br>`nper` — общее число составных периодов, `per≤nper`<br>`pv` — текущее значение<br>`fv` — будущее значение<br>`type=0` для платежа, сделанного в конце периода, или `1` для платежа, сделанного в начале периода | Соответствует процентному платежу по вкладу или займу за данный период, основанному на периодичности, постоянных платежах через данное число составных периодов, использующих фиксированную процентную ставку и особое текущее значение |

***Таблица П3.4*** *(продолжение)*

| Функция | Аргументы | Описание |
|---|---|---|
| `Irr (v,[guess])` | `v` — вектор денежных потоков, определяемых за регулярные интервалы, должен состоять, по крайней мере, из одного положительного и отрицательного числа<br><br>`guess` — численное значение, которым вы предполагаете аппроксимировать ответ; если им пренебрегается, то `guess=0.1(10%)` | Отвечает внутренней ставке возврата для серии денежных потоков, происходящих с регулярными интервалами |
| `Mirr (v,fin_rate, rein_rate)` | `v` — вектор денежных потоков, определяемых за регулярные интервалы, он должен состоять, по крайней мере, из одного положительного и отрицательного числа<br><br>`fin_rate` — финансовая ставка платежа по заимствованным денежным потокам<br><br>`rein_rate` — ставка реинвестирования | 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 вы повторно инвестируете |
| `nom(rate, nper)` | `rate` — эффективная ежегодная процентная ставка, должна быть действительным скаляром `rate>-1`,<br><br>`nper` — общее число составных периодов за год, `nper>0` | Соответствует номинальной процентной ставке, включающей эффективную ежегодную процентную ставку и число составных периодов за год |
| `npv(rate,v)` | `rate` — фиксированная процентная ставка, с которой вклад зарабатывает процент за период, должна быть действительным скаляром<br><br>`v` — вектор регулярных денежных потоков | Вычисляет чистое текущее значение вклада, включающее скидки и регулярные денежные потоки |

***Таблица П3.4*** *(продолжение)*

| Функция | Аргументы | Описание |
|---|---|---|
| `nper(rate, pmt,pv,[[fv], [type]])` | `rate` — фиксированная процентная ставка<br>`per` — период<br>`nper` — общее число составных периодов за год, должно быть положительным целым числом<br>`pmt` — платеж, делаемый каждый период<br>`pv` — текущее значение вклада<br>`fv` — будущее значение вклада<br>`type=0` для платежа, сделанного в конце периода, или `1` для платежа, сделанного в начале периода<br>`guess` — численное значение, которым вы предполагаете аппроксимировать ответ, если им пренебрегается, то `guess=0.01(10%)` | Отвечает числу периодов для вклада или займа, основанных на периодичности, постоянных платежах, использующих фиксированную процентную ставку, и особое текущее значение |
| `pmt(rate, nper,pv, [[fv], [type]])` | См. `nper` | Соответствует платежу по вкладу или займу, основанному на периодичности, постоянных платежах через данное число составных периодов, использующих фиксированную процентную ставку, и особое текущее значение |
| `ppmt(rate, per,nper,pv, [[fv], [type]])` | См. `nper` | Соответствует платежу по общей сумме вклада или займа, основанному на периодичности, постоянных платежах через данное число составных периодов, использующих фиксированную процентную ставку, и особое будущее значение |

***Таблица П3.4*** *(окончание)*

| Функция | Аргументы | Описание |
|---|---|---|
| `pv(rate,nper, pmt,[[fv], [type]])` | См. `nper` | Соответствует текущему значению вклада или займа, основанному на периодичности, постоянных платежах через данное число составных периодов, использующих фиксированную процентную ставку, и особый взнос |
| `rate(nper, pmt,pv,[[fv], [type], [guess]])` | См. `nper` | Соответствует процентной ставке на период вклада или займа при особом числе периодических составных периодов, постоянных платежах и особом текущем значении |

# Приложение 4

# Сообщения об ошибках

***Таблица П4.1.*** *Сообщения об ошибках*

| Ошибка | Перевод | Вероятная причина | Возможные пути устранения |
|---|---|---|---|
| **Сообщения об ошибках в численных вычислениях** | | | |
| A "Find" or "Minerr" must be preceded by a matching "Given" | `Find` или `Minerr` должны предваряться ключевым словом `Given` | Эта ошибка выделяет функцию `Find` или `Minerr` при их несогласованности с `Given` | Каждый вычислительный блок, который заканчивается функцией `Find` или `Minerr`, должен начинаться с ключевого слова `Given` |
| All evaluations resulted in either an error or a complex result | Вычисления приводят к ошибке или комплексному результату | Mathcad не может начертить некоторые точки, потому что не существует действительных значений для их нанесения на график | Это сообщение может появиться, если имеется ошибка или все значения комплексные |
| Arguments in function definitions must be names | Аргументы в определениях функции должны быть именами | Выделенное определение функции содержит неправильный перечень аргументов | В списке аргументов должны быть правильно поименованы переменные или список имен необходимо отделить запятыми |
| At least one limit must be infinity | По крайней мере один предел должен быть бесконечным | Когда для интегрирования выбран алгоритм бесконечного предела, то по крайней мере один из пределов интеграла должен быть бесконечным | Тип бесконечности вводится нажатием сочетания клавиш <Ctrl>+<Shift>+<Z>.<br><br>Для изменения алгоритма, использующего бесконечный предел, или для вычисления какого-либо другого интеграла щелкните на интеграле правой кнопкой мыши и измените алгоритм с помощью контекстного меню |

***Таблица П4.1*** *(продолжение)*

| Ошибка | Перевод | Вероятная причина | Возможные пути устранения |
|---|---|---|---|
| **Сообщения об ошибках в численных вычислениях** | | | |
| Can only evaluate an nth order derivative when n = 0,1..5. | Можно вычислить `n`-й порядок производной, только когда `n=0,1..5` | Порядок производной должен быть одним из следующих чисел: 0, 1, 2,... 5. | Если вы хотите посчитать производную более высокого порядка, то делайте это с помощью символьного дифференцирования |
| Can't evaluate this function when its argument less than or equal to zero | Невозможно вычислить эту функцию, когда ее аргумент меньше или равен нулю | Ошибка может заключаться в использовании неположительных данных на графиках, построенных в логарифмическом масштабе | Отрицательные числа и ноль не могут быть расположены на логарифмических осях. Смените тип осей графика или постройте его для других значений |
| Can't converge to a solution | Не сходится к решению | Численный метод расходится (не может найти решения) | Убедитесь, что операция не применяется к функции в области непосредственной близости точки ее сингулярности (деления на ноль).<br>Попробуйте поменять параметры численного метода (например, начальное приближение).<br>Попробуйте увеличить константу `TOL`, т. е. осуществить поиск решения с худшей погрешностью.<br>Попробуйте поменять численный алгоритм, если это возможно (вызвав контекстное меню нажатием на месте ошибки правой кнопки мыши) |
| Can't define the same variable more than once in the same expression | Невозможно определить ту же самую переменную более одного раза в одном и том же выражении | Вы пытаетесь вычислить одну и ту же переменную дважды в одном выражении | Пример подобной ошибки: если вы создаете вектор с левой стороной `a:=` и используете это же имя справа, то получите ошибку |

***Таблица П4.1*** *(продолжение)*

| Ошибка | Перевод | Вероятная причина | Возможные пути устранения |
|---|---|---|---|
| **Сообщения об ошибках в численных вычислениях** | | | |
| Can't determine what units the result of this operation should have | Невозможно определить, в каких единицах следует быть результату этой операции | Вы возвели выражение, содержащее единицы измерения, в степень, являющуюся переменной в неких пределах или вектором. В результате невозможно определить размерность результата | Если выражение включает в себя единицы измерений, то можно возводить его только в действительную фиксированную степень |
| Can't divide by zero | Деление на ноль невозможно | Где-то в программе или внутри численного метода возникло деление на ноль | Найдите место деления на ноль и устраните его.<br>Попробуйте поменять параметры численного метода, константы точности или сам численный алгоритм |
| Could not find a solution | Невозможно найти решение | Численный метод расходится (не может найти решения) | *См. "Can't converge to a solution"* |
| Can't find the data file you're trying to use | Невозможно найти файл, который вы пытаетесь использовать | Невозможно найти файл данных или другой тип файла, к которому вы обращаетесь | Удостоверьтесь, что такой файл существует в указанном месте |
| Can't have anything with units or dimensions here | Здесь нет ничего в единицах измерений или в размерностях | Это выражение использует единицы измерений где-то, где они не разрешены | Единицы измерений не разрешены:<br>• в аргументах большинства функций;<br>• в экспонентах;<br>• в верхних и нижних индексах.<br>Для того чтобы использовать выражения с единицами измерений, вначале переведите это выражение в UnitsOf |

***Таблица П4.1*** *(продолжение)*

| Ошибка | Перевод | Вероятная причина | Возможные пути устранения |
|---|---|---|---|
| **Сообщения об ошибках в численных вычислениях** | | | |
| Can't have more than one array in a contour plot | Нельзя иметь более одного массива в контурном графике | Вы вводите более одного массива в местозаполнитель контурного или поверхностного графика | Можно иметь только один массив в данном местозаполнителе, т. к. графики могут выдавать лишь одну поверхность в один момент времени |
| Can't perform this operation on the entire array at once. Try using "vectorize" to perform it element by element. | Невозможно представить эту операцию в целом массиве сразу. Попытайтесь использовать векторизацию, чтобы представить элемент за элементом | Например, можно увидеть это сообщение при попытке разделить один вектор на другой | Для того чтобы применять функцию или оператор к каждому элементу вектора или матрицы, используйте оператор векторизации |
| Can't plot this many points | Невозможно начертить график с таким большим количеством точек | Попытка построения графика с числом точек, превосходящим возможное | Попробуйте сделать число точек меньше, чем 150 000 |
| Can't put a ":=" inside a solve block. | Нельзя помещать ":=" внутрь вычислительного блока | Внутри вычислительного блока не должно быть формулировки присваивания. Он должен содержать только булевы выражения | Используйте панель с булевыми операторами |
| Can't raise an expression having units to a complex power | Нельзя возводить в комплексную степень выражение, имеющее единицы измерений | Это выражение содержит единицы измерений, а вы возводите его в комплексную степень | Выражение с единицами измерений можно возводить только в действительную степень. Для того чтобы возводить в комплексную степень выражение с единицами измерений, вначале переведите это выражение в SIUnitsOf — единицы измерений будут отменены |

***Таблица П4.1*** *(продолжение)*

| Ошибка | Перевод | Вероятная причина | Возможные пути устранения |
|---|---|---|---|
| **Сообщения об ошибках в численных вычислениях** | | | |
| Can't solve a system having this many equations | Невозможно решить систему, имеющую так много уравнений | Mathcad не способен решить систему | См. определение термина "вычислительный блок" (*разд. 5.1*) |
| Can't understand something in this data file | Невозможно что-то понять в файле данных | Файл, к которому вы пытаетесь получить доступ при помощи `READ` или `READ*`, имеет дефект | Файл должен быть ASCII-текстом.<br>Все строки файла должны иметь тот же номер значений, что используется в `READ*`.<br>Если файл имеет требуемый формат, а это сообщение продолжает появляться, попробуйте удалить любую часть текста из файла |
| Can't understand the name of this function | Невозможно понять имя этой функции | Такое сообщение может появиться, если в качестве имени функции используется, например, число `6(x)` | Выражение должно соответствовать требованиям, предъявляемым в Mathcad к написанию имен функций |
| Can't understand the way this range variable is defined | Невозможно понять определение ранжированной переменной | Определение ранжированной переменной неверно | Вводя область определения ранжированной переменной, необходимо использовать один из следующих видов:<br>• `Rvar := n1 ..n2`<br>• `Rvar := n1,n2 ..n3` |
| Can't understand this number | Невозможно понять это число | Это выражение содержит символ или десятичную точку там, где это непозволительно | Вы увидите эту ошибку, например, если случайно запишете число так: ".452" |
| Can't use a range variable in a solve block. | Невозможно использовать ранжированную переменную в вычислительном блоке | Эта ошибка появится, если использовать область определения переменной в неподходящем месте | Придумайте алгоритм, не допускающий применение ранжированной переменной в вычислительном блоке |

***Таблица П4.1*** *(продолжение)*

| Ошибка | Перевод | Вероятная причина | Возможные пути устранения |
|---|---|---|---|
| **Сообщения об ошибках в численных вычислениях** | | | |
| Cannot evaluate this accurately at one or more of the values you specified | Невозможно точно вычислить одно или более значений | Эта ошибка появляется, если попытаться вычислить функцию для аргумента, находящегося за пределами точной области определения функции | Проверьте область определения функции |
| Cross product is defined only for vectors having exactly three elements | Векторное произведение определяется только для векторов, имеющих точно три элемента | Возможно, предпринята попытка вычислить векторное произведение для векторов, имеющих не три, а другое количество элементов | См. определение векторного произведения (*разд. 7.2.3*) |
| Can't evaluate this expression. It may have resulted in an overflow or an infinite loop | Невозможно вычислить это выражение. Это может быть результатом переполнения или бесконечных циклов | Это функциональное определение может содержать слишком много вложенных функций. Или функция может быть константой в бесконечных циклах | Проверьте несколько итераций цикла |
| Degree of the polynomial must be between 1 and 99. | Степень полинома должна находиться в пределах между 1 и 99 | Вектор, пропущенный через функцию поиска корней полинома, должен содержать, по крайней мере, 2 и не более 99 элементов | |
| Dimensions must be > 4 | Размерность должна быть > 4 | Эта матрица должна иметь, по крайней мере, 4 ряда и 4 столбца | |

***Таблица П4.1*** *(продолжение)*

| Ошибка | Перевод | Вероятная причина | Возможные пути устранения |
|---|---|---|---|
| **Сообщения об ошибках в численных вычислениях** | | | |
| End of file | Конец файла | Вы пытаетесь прочитать больше значений в файле данных, чем там имеется | Например, если файл данных имеет 10 значений, а записано выражение `i:=1…100` `xi:=READ*(file)`, то появится это сообщение |
| End points cannot be the same | Конечные точки не могут быть одинаковыми | Это сообщение появляется при некорректном решении дифференциальных уравнений | Конечные точки интервала, на котором будет вычисляться решение, должны быть различны |
| Equation too large | Уравнение слишком большое | Это выражение слишком сложное для вычисления | Разбейте выражение на несколько простых |
| Floating point error | Ошибка вычислений с плавающей точкой | Функция вычисляется в точке, в которой это запрещено | |
| Found a singularity while evaluating this expression. You may be dividing by zero | Найдена сингулярность при вычислении этого выражения. Возможно, вы делите на ноль | Вычисляется функция или выполняется операция с недопустимым значением | Например, это сообщение возникнет при попытке деления на ноль или обращения сингулярной матрицы; разберитесь, где это происходит |
| Found a number with a magnitude greater than 10^307 | Найдено число, превышающее значение $10^{307}$ | | Попробуйте поменять параметры численного алгоритма или сам алгоритм |
| Illegal context. Press <F1> for help | Недопустимый контекст. Нажмите <F1>, чтобы получить помощь | Часто встречается при синтаксических ошибках | Проверьте синтаксис и порядок расположения формул в документе |
| Illegal dimensions | Недопустимые размерности | Матрица, на которую вы ссылаетесь, не имеет достаточно строк или столбцов | Введите имя матрицы с клавиатуры и нажмите знак "=", чтобы проверить число ее строк и столбцов |

***Таблица П4.1*** *(продолжение)*

| Ошибка | Перевод | Вероятная причина | Возможные пути устранения |
|---|---|---|---|
| **Сообщения об ошибках в численных вычислениях** | | | |
| Integer too large/ Integer too small | Целое число слишком большое/слишком маленькое | Это число слишком велико/мало для работы с ним | Если вы работаете со встроенными функциями, то щелкните левой кнопкой мыши на имени функции и вызовите подсказку с помощью клавиши <F1> |
| Invalid format | Недопустимый формат | Аргументы этой функции могут быть некорректны | Если вы работаете со встроенными функциями, то щелкните левой кнопкой мыши на имени функции и вызовите подсказку с помощью клавиши <F1> |
| Live symbolics not available | Символьные вычисления неприменимы | | |
| Must be <= <=10 000 | Значение должно быть <=10 000 | | |
| Must be >= >=10^-16 | Значение должно быть $>=10^{-16}$ | | |
| Must be string | Функция или оператор должен быть строковым аргументом | | *См. разд. 1.2.5* |
| Must be function | Аргумент должен быть функцией | | |
| Must be increasing | Значения вектора должны быть возрастающими | | Введите с клавиатуры имя вектора и знак "=", чтобы проверить его значения |
| Must be less than the number of data points | Должен быть меньше, чем число точек данных | Этот аргумент должен быть меньше, чем число точек имеющихся данных | |

***Таблица П4.1*** *(продолжение)*

| Ошибка | Перевод | Вероятная причина | Возможные пути устранения |
|---|---|---|---|
| **Сообщения об ошибках в численных вычислениях** | | | |
| Must be positive | Должен быть положительным | Невозможно вычислить функцию, когда ее значения меньше или равны нулю | Это сообщение может касаться построения X-Y или полярных графиков с логарифмическими осями. Отрицательные числа или ноль не могут располагаться на логарифмических осях |
| Must be real | Должно быть действительным | Это значение должно быть действительным. Его мнимая часть должна быть нулем | Примером таких выражений могут служить нижний и верхний индексы, решения дифференциальных уравнений, углы |
| Must be real scalar | Должен быть действительным скаляром | Это значение не должно быть комплексным или мнимым | |
| Must be real vector | Должно быть действительным вектором | Этот вектор не может иметь комплексные или мнимые элементы. Он должен также быть вектором-столбцом, а не строкой | |
| Must be square | Должна быть квадратной | Эта ошибка выделяет неквадратную матрицу в той операции или функции, в которой ей следует быть квадратной | Например, матрица должна быть квадратной при обращении, возведении ее в степень, или в функциях `eigenvals` и `eigenvec` |
| No solution found | Не найдено решение | | Если вы используете встроенные функции, то щелкните левой кнопкой мыши на имени функции и нажмите клавишу <F1> для того, чтобы быть уверенным в корректности использования функции. Однако решение может просто не существовать.<br>*См. также "Can't converge to a solution"* |

***Таблица П4.1*** *(продолжение)*

| Ошибка | Перевод | Вероятная причина | Возможные пути устранения |
|---|---|---|---|
| **Сообщения об ошибках в численных вычислениях** | | | |
| Not enough memory for this operation | Для этой операции недостаточно памяти | Не хватает памяти, чтобы завершить это вычисление | Попытайтесь освободить немного памяти путем уменьшения массива или матрицы (Mathcad тратит около 8 байт памяти на каждый элемент матрицы) или удаления каких-либо больших побитовых отображений, массивов, матриц |
| Singular matrix | Сингулярная матрица | Эта матрица не может быть ни сингулярной, ни близкой к сингулярности | Матрица называется сингулярной, если ее определитель равен нулю. Матрица близка к сингулярной, если она имеет высокое число обусловленности *(см. главу 9)* |
| The expression to the left of the equal sign cannot be defined | Выражение слева от знака равенства не может быть определено | В левой части находится что-то, что не является допустимым определяемым выражением | В левой части можно разместить одно из следующих определений:<br>• имя переменной;<br>• имя переменной с верхним или нижним индексом;<br>• явный вектор или матрицу;<br>• имя функции с аргументами `f(x,y)`.<br>Любые другие выражение недопустимы |
| The number of rows and/or columns in these arrays do not match | Число рядов и/или столбцов в этих массивах не согласовано | Попытка произвести матричные или векторные операции над массивами, размеры которых не совпадают | Например, сложение двух матриц разного размера недопустимо. Матричное умножение требует, чтобы число столбцов первой матрицы совпадало с числом строк второй *(см. разд. 9.1)* |

***Таблица П4.1*** *(продолжение)*

| Ошибка | Перевод | Вероятная причина | Возможные пути устранения |
|---|---|---|---|
| **Сообщения об ошибках в численных вычислениях** | | | |
| The units in this expression do not match | Размерности в этом выражении не согласованы | Это сообщение появится, если складываются два элемента разной размерности, либо создана матрица, элементы которой имеют разную размерность, либо вы пытаетесь решить систему уравнений для неизвестных переменных разной размерности | Проверьте использование размерных переменных |
| There is an extra comma in this expression | В выражении лишняя запятая | | Запятые должны использоваться для того, чтобы отделять:<br>• аргументы в функции;<br>• первые два элемента области в определении интервала;<br>• выражения в графике;<br>• элементы во входной таблице;<br>• нижние индексы в матрице.<br>Любые другие применения запятой приводят к ошибке. Например, запись 4,000 неправильная, а запись 4 000 — правильная |
| This expression is incomplete. You must fill in the placeholders | Выражение неполное. Необходимо добавить содержимое в местозаполнители | Не заполнены указанные местозаполнители | Необходимо дописать числа или выражения в указанные местозаполнители |

***Таблица П4.1*** *(продолжение)*

| Ошибка | Перевод | Вероятная причина | Возможные пути устранения |
|---|---|---|---|
| **Сообщения об ошибках в численных вычислениях** | | | |
| This expression is incomplete. You must provide an operator | Выражение неполное. Необходимо вставить оператор | Не заполнены местозаполнители оператора или пустое пространство между двумя операндами | Это могло произойти при удалении оператора; проверьте правильность ввода выражения |
| This function has too many arguments | Функция имеет слишком много аргументов | Выделенное выражение содержит функцию с числом аргументов большим, нежели требуется | Проверьте правильность применения функции |
| This function is undefined at one or more of the points you specified | Функция не определена для одной или более точек | Попытка вычисления оператора или функции с неподходящими значениями | Например, вычисление `-3!` и `ln(0)` — приведет к ошибке, т. к. факториал не определен для отрицательного числа, а логарифм для нуля |
| This function needs more arguments | Функции не хватает аргументов | Выделенное выражение содержит функцию с меньшим, нежели требуется, числом аргументов | Для встроенных функций щелкните левой кнопкой мыши на имени функции и воспользуйтесь подсказкой <F1>, чтобы проверить правильность числа и типа аргументов. Для функции пользователя проверьте ее определение |
| This operation can only be performed on a function | Операция может применяться только для функций | Этот аргумент должен быть функцией | Для встроенных функций щелкните левой кнопкой мыши на имени функции и воспользуйтесь подсказкой <F1> |
| This operation can only be performed on an array. It can't be performed on a number | Операция может применяться только для массивов. Она не может быть использована для чисел | | Например, это сообщение появится, если переменная верхнего индекса определена как скаляр. Поскольку переменная верхнего индекса представляет собой столбец матрицы, то ее следует определять как вектор.<br>Для поверхностных или контурных графиков массив данных должен иметь, по крайней мере, два ряда и два столбца |

***Таблица П4.1*** *(продолжение)*

| Ошибка | Перевод | Вероятная причина | Возможные пути устранения |
|---|---|---|---|
| **Сообщения об ошибках в численных вычислениях** | | | |
| This operation can only be performed on a number or an array | Операция может применяться только для чисел или массивов | Используемая функция или оператор требуют представления в виде константы, матрицы или вектора | |
| This operation can only be performed on a string | Эта операция может применяться только для строк | Используемая функция или оператор требуют представления в виде строки. Например, строковые функции обычно требуют, по крайней мере, одного строкового аргумента | |
| This subscript is too large | Нижний индекс слишком велик | Попытка использовать верхний или нижний индекс, который превышает ограничения | |
| This value must be a matrix | Значение должно быть матрицей | Попытка произвести матричную операцию не над матрицей | |
| This value must be a vector. It can be neither a matrix nor a scalar | Значение должно быть вектором. Оно не может быть ни матрицей, ни скаляром | Это сообщение маркирует матрицу или скаляр в операциях, которые требуют вектора (одностолбцового массива). Например, суммирование элементов вектора | |
| This value must be an integer greater than 1 | Значение должно быть целым числом, превосходящим единицу | Значение должно быть $\geq 1$ | При использовании встроенных функций щелкните левой кнопкой мыши на имени функции и нажмите клавишу <F1> |

***Таблица П4.1*** *(продолжение)*

| Ошибка | Перевод | Вероятная причина | Возможные пути устранения |
|---|---|---|---|
| **Сообщения об ошибках в численных вычислениях** | | | |
| This variable or function is not defined above | Переменная или функция не определена выше | Имя неопределенной функции будет помечено красным цветом | Удостоверьтесь, что эта функция или переменная определена выше. Это сообщение появится, если переменная некорректно используется в глобальном определении. Эта ошибка часто свидетельствует о том, что другое уравнение выше в документе является ошибкой. В этом случае все выражения, использующие выражение с ошибкой, будут помечены красным цветом |
| Underflow | Потеря значимости (исчезновение значащих разрядов) | Из-за ограничений, присущих представлению чисел на компьютере, числа, которые слишком малы, не могут быть представлены. Это сообщение появляется, когда выражение включает такое число. Иногда, особенно в сложных вычислениях, промежуточный результат будет слишком мал, и вся разрядная сетка заполнится нулями | |
| Value of subscript or superscript is too big (or too small) for this array | Значение нижнего или верхнего индекса слишком велико (или слишком мало) для этого массива | Это выражение использует нижний или верхний индекс, который относится к несуществующему элементу массива | |

***Таблица П4.1*** *(продолжение)*

| Ошибка | Перевод | Вероятная причина | Возможные пути устранения |
|---|---|---|---|
| **Сообщения об ошибках в численных вычислениях** | | | |
| This is not a scalar. Press <F1> for help | Это не скаляр. Нажмите клавишу <F1>, чтобы получить помощь | Использован вектор или выражение с интервалами, или какой-то другой тип выражения, где требуется применение скаляра | |
| You have one solve block inside another. Every "Given" must have a matching "Find" or "Minerr" | Один вычислительный блок содержится внутри другого. Каждому ключевому слову `Given` должно сопоставляться `Find` или `Minerr` | Указаны два ключевых слова `Given` подряд без `Find` или `Minerr` посередине. Вычислительный блок не может иметь внутри себя другой вычислительный блок | В качестве альтернативы можно задать функцию в терминах одного вычислительного блока и использовать ее внутри другого вычислительного блока. Во многих случаях это дает тот же самый эффект |
| You interrupted calculation. To resume, click here and choose "Calculate" from the "Math" menu | Вычисления прерваны. Для того чтобы продолжить, щелкните здесь и выберите пункт **Calculate** меню **Math** | | Вычисления прерваны нажатием клавиши <Esc>. Для того чтобы пересчитать выделенное уравнение, наведите на него курсор и воспользуйтесь меню **Math / Calculate** (Математика / Вычислить) |
| **Сообщения об ошибках в символьных вычислениях** | | | |
| Argument too large (Integer too large in context, Object too large) | Аргумент слишком велик | Обычно это результат вычисления выражения с плавающей точкой со значением большим, чем около 10×10 миллиардов | |
| Discarding large result | Сброс большого результата | Ответ слишком велик для отображения его в отформатированной математической области | Можно разместить ответ в буфере обмена |

***Таблица П4.1*** *(продолжение)*

| Ошибка | Перевод | Вероятная причина | Возможные пути устранения |
|---|---|---|---|
| **Сообщения об ошибках в символьных вычислениях** | | | |
| Expecting array or list | Ожидается массив или список | Операторы в упрощаемом или вычисляемом выражении требуют векторных или матричных операндов | |
| Expression contains non-symbolic operators | Выражение содержит несимвольные операторы | Применена символьная операция к выражению, содержащему местозаполнители оператора или переменной | |
| Floats not handled | С плавающей запятой не поддерживается | Команда `Factor` была применена к выражению с десятичным числом | |
| Illegal function syntax | Недопустимый синтаксис функции | Символьный процессор не может интерпретировать выражение, подобное `(f)(x)` | |
| Invalid arguments | Недопустимые аргументы | Символьный процессор не может выполнить требуемую операцию для данных аргументов | Это сообщение появится, если, например, применить скалярную функцию к массиву без использования оператора векторизации и выбрать команду **Symbolics / Simplify** (Символика / Упростить) |
| Invalid range | Недопустимый интервал | Для поиска численного решения уравнения символьный процессор пытается вычислить одну из своих встроенных функций за пределами области ее определения | |

***Таблица П4.1*** *(окончание)*

| Ошибка | Перевод | Вероятная причина | Возможные пути устранения |
|---|---|---|---|
| **Сообщения об ошибках в символьных вычислениях** | | | |
| No answer found; stack limit reached | Ответа не найдено | Символьный процессор достиг предела своих возможностей без вычисления или упрощения, которое затребовал пользователь | |
| No answer found | Ответа не найдено | Символьный процессор не смог найти точного решения уравнения | |
| No closed form found for | Не найдено замкнутой формы для | Символьный процессор не смог найти интеграл, или сумму, или произведение в замкнутой форме | |
| Syntax error | Синтаксическая ошибка | Обычно результат применения символьной операции в неподходящих или некорректных выражениях. Символьные вычисления выражений с размерностями также приведут к появлению этого сообщения | |

# Приложение 5

# Ресурсы Mathcad

Ресурсы Mathcad (Mathcad Resource) — это библиотека электронных книг, поставляемая вместе с Mathcad 12. Она содержит обширную справочную информацию и обладает всеми свойствами электронных книг, подключаемых к Mathcad. Ресурсы представляют собой сборник примеров решения различных математических, физических и инженерных задач и содержат справочную информацию о возможностях Mathcad. Ресурсы содержат очень большое количество информации, пополняемое от одной версии Mathcad к другой. Практически по любому разделу математики и любому методу решения той или иной задачи в Mathcad здесь можно найти справочные сведения. Приведем краткий перечень глав электронных книг ресурсов.

***Таблица П5.1. Tutorials*** *(Учебники)*

| Название учебника | Описание |
|---|---|
| **Overview and Quick Tour** (Обзор Mathcad и Быстрый старт) | Учебник для тех, кто делает первые шаги в Mathcad и абсолютно с ним не знаком, но хочет быстро начать собственные расчеты. Содержит вводные замечания о том, что можно делать при помощи Mathcad, и рассказ о его основных возможностях |
| **New Features in Mathcad 12** (Новые возможности) | Учебник, адресованный давним пользователям Mathcad, уже имевшим дело с его прежними версиями и которым будет интересно познакомиться с реальными примерами использования новых возможностей версии 12 |
| **Getting Started Primers** (Учебник для начинающих) | Очень полезный для новых пользователей интерактивный учебник, который шаг за шагом продемонстрирует пользователю все возможности Mathcad, без знания которых трудно проводить какие бы то ни было расчеты |

***Таблица П5.1*** *(окончание)*

| Название учебника | Описание |
|---|---|
| **Features in Depth** (Возможности) | Учебник для пользователей, имеющих опыт работы с интерфейсом, нацеленный на раскрытие основных возможностей Mathcad и приемов решения конкретных задач |
| **Where to Get More Help** (Как получить дополнительную справку) | Информация о способах получения дополнительной справки |

***Таблица П5.2. QuickSheets*** *(Быстрые шпаргалки)*

| Название шпаргалки | Описание |
|---|---|
| **About QuickSheets** (О шпаргалках) | Справочная информация по использованию шпаргалок |
| **Vectors and Matrices** (Векторы и матрицы) | *См. главы 7, 8* |
| **Solving Equations** (Решение уравнений) | *См. главы 5, 6* |
| **Graphing and Visualization** (Построение графиков и визуализация данных) | *См. разд. 1.4* |
| **Calculus and Differential Equations** (Вычисления и дифференциальные уравнения) | *См. главы 9—11* |
| **Engineering applications** (Инженерные приложения) | Примеры применения Mathcad для решения инженерных и научных задач |
| **Mathcad Techniques** (Технические приемы) | *См. главы 1, 2* |
| **Data Analysis** (Анализ данных) | *См. главы 13—14* |
| **Statistics** (Статистика) | *См. главу 12* |
| **Using Mathcad with Other Applications** (Mathcad и другие приложения) | Примеры подключения к расчетам Mathcad других приложений |
| **Symbolic Math** (Символьная математика) | *См. главы 1, 2* |
| **Programming** (Программирование) | *См. разд. 1.3.5* |
| **Extra Math Symbols** (Дополнительные математические символы) | Сводка математических символов |
| **Reference Tables** (Таблицы) | Электронная книга с самой разной справочной информацией из области математики, физики, химии |
| **User's guide** (Руководство пользователя) | Справочное руководство |

# Приложение 6

# Описание компакт-диска

## Содержимое диска

На компакт-диске, прилагаемом к книге, вы найдете (таблица П6.1):

- полное содержание почти всех примеров и листингов в формате Mathcad (по возможности 2001), объединенных в электронную книгу Mathcad;
- HTML-учебник автора по вычислительной математике (зеркало сайта **www.keldysh.ru/comma**) с примерами расчетов в Mathcad;
- демо-версию мультимедийного курса по вычислительной математике (виртуальные видеолекции).

***Таблица П6.1.*** *Содержание компакт-диска*

| | |
|---|---|
| **index.html** | Web-страница с приглашением и справочной информацией об использовании диска |
| **kirianov.hbk** | Стартовый файл электронной книги с примерами листингов и рисунков |
| **kirianov** | Папка с документами Mathcad (листинги и рисунки из книги) |
| **comma** | Папка с электронным учебником по вычислительной математике |
| **html** | Папка со служебными файлами |

## Как пользоваться расчетами Mathcad

Диск содержит большинство листингов и рисунков к книге в формате документов Mathcad. Вы можете просматривать их и использовать фрагменты для копирования в ваши собственные расчеты. Наиболее удобный вариант рабо-

ты с документами — это просмотр в виде электронной книги Mathcad. Такой способ, во-первых, позволяет быстро переходить от одного документа к другому (не разыскивая и не открывая каждый раз нужный файл) и, во-вторых, допускает возможность одновременно держать открытым в основном окне Mathcad ваш собственный документ.

Для того чтобы открыть примеры в формате электронной книги Mathcad, выполните следующее:

1. Запустите Mathcad.
2. Выберите в меню команду **Help / Open Book** (Справка / Открыть книгу).
3. В диалоговом окне **Open Book** (Открыть книгу) перейдите к корневой папке компакт-диска и откройте файл kirianov.hbk.
4. На появившейся странице-приглашении (рис. П6.1) нажмите кнопку **GO!**. Имейте в виду, что электронная книга открывается в новом окне (независимом от основного окна Mathcad), снабженном специальной панелью инструментов навигации.

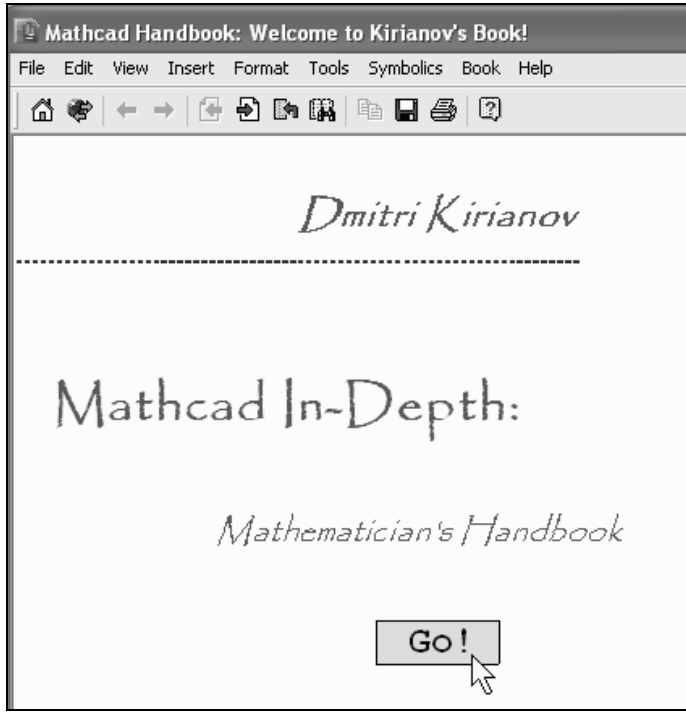



**Рис. П6.1.** Стартовая страница-заставка

### *Примечание*

Во избежание недоразумений, связанных с воспроизведением кириллицы, электронная книга имеет англоязычный интерфейс.

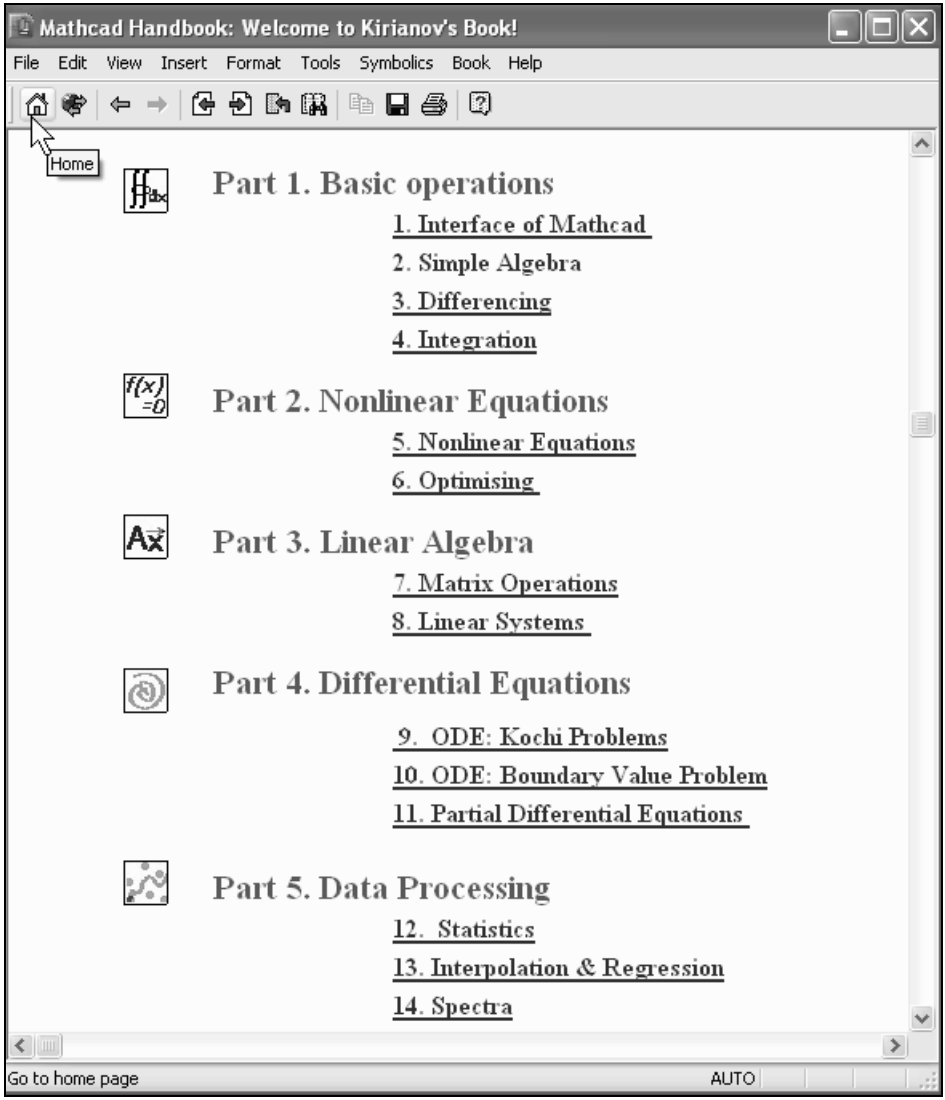



**Рис. П6.2.** Домашняя страница
с общим оглавлением электронной книги

Просмотр документов Mathcad, входящих в электронную книгу, организован следующим образом.

1. Со страницы-оглавления (рис. П6.2) можно перейти к содержанию отдельных глав.
2. Для того чтобы возвратиться к оглавлению из любого места электронной книги, достаточно нажать кнопку **Home** (Домой) на панели инструментов навигации (на рис. П6.2 на нее наведен указатель мыши).
3. Просматривайте все страницы электронной книги (по порядку) при помощи кнопок **Next topic** и **Previous topic** на панели инструментов (рис. П6.3).
4. Чтобы просмотреть избранные расчеты, пользуйтесь гиперссылками в оглавлениях.
5. Если вы хотите использовать фрагменты страниц книги в ваших собственных расчетах, выделите их и нажмите клавиши <Ctrl>+<C>. Затем вернитесь в свой документ и вставьте фрагмент нажатием клавиш <Ctrl>+<V>.

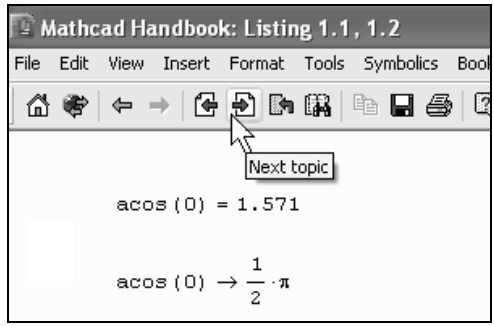



**Рис. П6.3.** Просматривать документы книги по порядку можно при помощи кнопок **Next topic** и **Previous topic**

Если вы по каким-либо причинам не хотите использовать документы Mathcad в форме электронной книги, то можно просматривать их как обычные документы. Для этого:

1. Выберите в Mathcad команду **File / Open** (Файл / Открыть) и перейдите к папке kirianov, находящейся на компакт-диске. В ней откройте вложенную папку с номером главы книги.
2. Откройте нужный файл с расширением mcd или xmcd (Mathcad 12). Например, для просмотра листинга 8.1 откройте файл 8_1.mcd, рисунка 3.4 — файл pic_3_4.mcd и т. д.

# Как пользоваться учебником по вычислительной математике

На компакт-диске, помимо примеров Mathcad, вы найдете электронный учебник по вычислительной математике (для студентов вузов), созданный автором этой книги. Для того чтобы открыть стартовую страницу учебника, перейдите к ней по гиперссылке с главной Web-страницы (index.html) в браузере. Домашняя страница учебника (рис. П6.4) находится на компакт-диске, путь к ней выглядит следующим образом: cd:/comma/index.html. Учебник не требует дополнительной установки и работает исключительно в браузере.

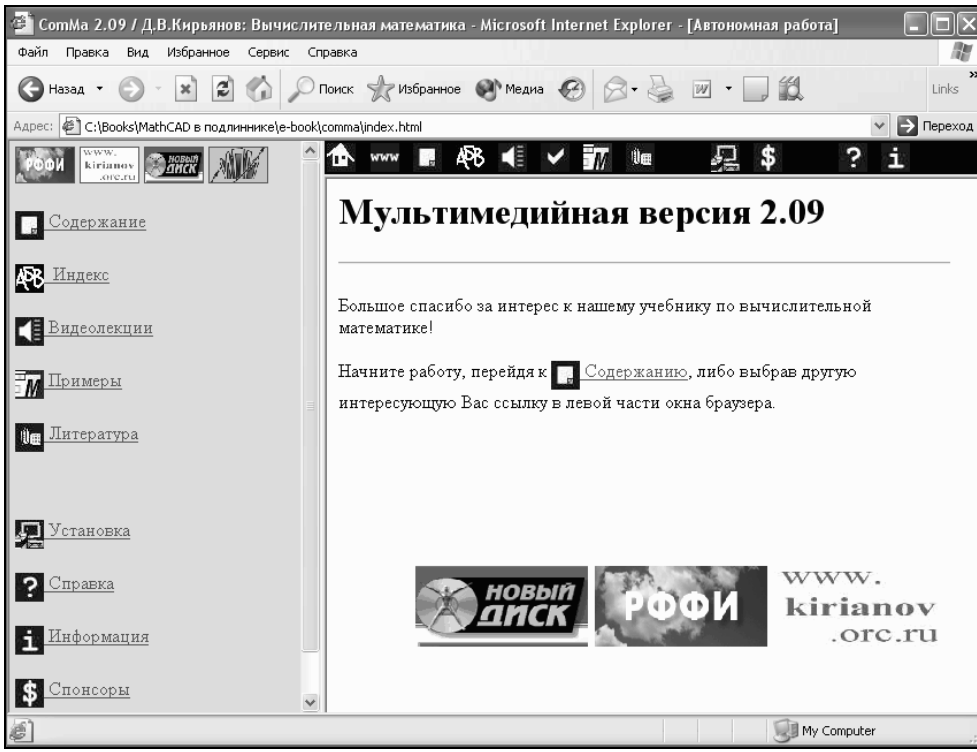



**Рис. П6.4.** Электронный учебник по вычислительной математике открывается в браузере

Электронный учебник выполнен в виде комбинации аудио-, видео- и информационных HTML-файлов, снабженных в качестве иллюстраций Java-

апплетами, примерами в формате Mathcad и контрольными тестами. Перечень составляющих учебника сведен в таблицу П6.2. Основу учебника составляет авторский курс лекций Д. В. Кирьянова по вычислительной физике, прочитанный им на физическом факультете МГУ. Лекции записаны в формате видео + звук посредством аппаратных средств компьютера, а также представлены в виде текста. Обратите внимание, что данный диск содержит полную HTML- и Mathcad-версии учебника, а также демо-версию его мультимедийной части.

***Таблица П6.2.*** *Курс по вычислительной математике*

| | |
|---|---|
| HTML-учебник | Обычный учебник с множеством гиперссылок, представляющий собой сжатое изложение материала лекционного курса. Для его просмотра не требуется никаких дополнительных средств, помимо обычного браузера |
| Индекс | Алфавитный указатель терминов с выходом как на гипертекстовую, так и мультимедийную части курса |
| Видеолекции | Лекционный курс, содержащий видео и звук. Видеолекции могут запускаться как обычные видеофайлы (компакт-диск содержит только несколько демонстрационных лекций, полная редакция выходит на компакт-диске в издательстве "Новый Диск") |
| Примеры | Большое количество практических примеров, дополняющих курс. Для их полноценного просмотра требуется только браузер, а если вы хотите воспользоваться расчетами для своих задач, откройте соответствующие файлы в Mathcad |
| Практикум | Несколько примеров, оформленных как Java-апплеты, требуют поддержки Java |

Общее строение электронного учебника следующее. Наряду с видеолекциями в медиа-формате имеется соответствующее количество гипертекстовых документов, а также иллюстраций в виде Mathcad-примеров. Эти иллюстрации дают простое, но понятное представление об изучаемом предмете (т. е. о работе вычислительных методов). Главными отличительными особенностями Mathcad-примеров являются, во-первых, интерактивность (т. е. возможность пользователя изменять параметры каждого метода и наблюдать соответствующий результат) и, во-вторых, возможность применения данных Mathcad-фрагментов в собственных математических расчетах пользователя. На рис. П6.5 изображена открытая в браузере страница учебника с содержанием Mathcad-примеров. Для просмотра нужного документа с расчетами (также в браузере) достаточно щелкнуть на соответствующей гиперссылке.

Переход от одного стиля освоения материала к другому максимально упрощен благодаря единой гипертекстовой компоновке материала и возможности просмотра средствами браузеров. Таким образом, для работы с электронным учебником пользователю нужен лишь компьютер, оснащенный средствами мультимедиа, и браузер. Для дополнительной информации о работе с электронным учебником обратитесь к его разделу "Справка".

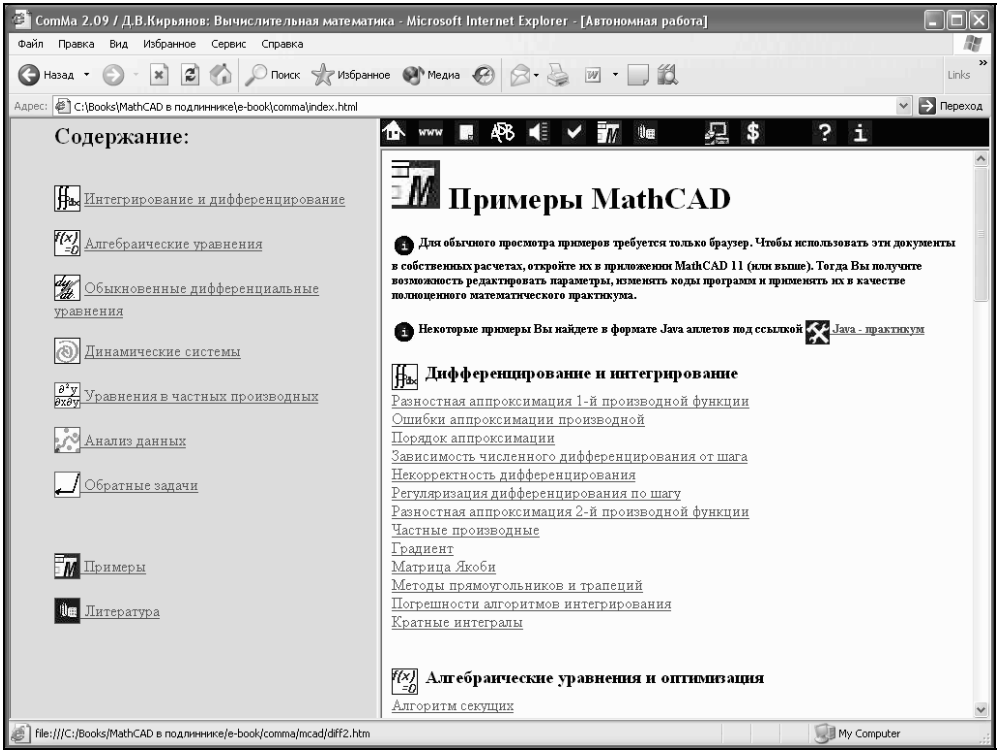



**Рис. П6.5.** Страница учебника
с оглавлением расчетов Mathcad (в формате HTML)

При работе с виртуальными видеолекциями учебника имейте в виду, что для прослушивания доступны только первые из мультимедийных лекций основных разделов. Полную версию электронного курса вы можете приобрести у дистрибьюторов CD-ROM-издательства "Новый Диск"(**www.nd.ru**).

### *Примечание 1*

Электронный учебник по вычислительной математике (© авторы — Д. В. Кирьянов и Е. Н. Кирьянова) создан при финансовой поддержке Российского фонда фундаментальных исследований (грант № 01-07-90135).

### *Примечание 2*

Полная версия мультимедийного учебника (со всеми мультимедийными видеолекциями) выпущена в 2005 г. CD-ROM-издательством "Новый Диск".

### *Примечание 3*

Интернет-версия учебника доступна на сервере ИПМ им. Келдыша РАН: **www.keldysh.ru/comma**. Обновления и дополнения к содержанию компакт-диска будут публиковаться на сервере автора этой книги **www.kirianov.orc.ru**.

# Список литературы

## I. Книги и мультимедийные учебники автора


1. Кирьянов Д. В., Кирьянова Е. Н. Лекции по вычислительной физике. — М.: Полибук Мультимедиа, 2005.

   Современный курс "Computational science", разбитый на две части: моделирование и обработка эксперимента. Это "живая книга" с мультимедийным учебником (курсом виртуальных видеолекций) и интерактивными расчетами всех иллюстраций, выполненными в формате Mathcad. Примеры оформлены в формате электронной книги Mathcad, что облегчает их использование в собственных расчетах читателя и позволяет "поиграть" параметрами.

2. Кирьянов Д. В. Самоучитель Mathcad 13. — СПб.: БХВ-Петербург, 2006.

   В книге подробно описана работа в среде Mathcad 13. Рассмотрены все основные возможности этой программы. Рассматриваются типичные математические задачи и способы их решения с помощью Mathcad. Подробно излагаются сведения, касающиеся профессионального оформления расчетов в Mathcad и методы эффективной работы для опытных пользователей.

3. Кирьянов Д. В. Вычислительная математика. Мультимедийный обучающий CD. — М.: Новый Диск, 2005.

   Мультимедийный учебник (курс виртуальных видеолекций), созданный на основе лекционного курса по вычислительной физике, прочитанного автором в МГУ им. Ломоносова.

4. **www.keldysh.ru/comma** — обучающий и справочный ресурс по вычислительной математике (облегченная версия мультимедийного учебника, содержащая всю гипертекстовую составляющую и часть медиа-контента в формате Flash).

# II. Литература по математике и методам вычислений

1. Анищенко В. С. Сложные колебания в простых системах. — М.: Наука, 1990.
2. Бат М. Спектральный анализ в геофизике. — М.: Наука, 1980.
3. Гласко В. Б. Обратные задачи математической физики. — М.: МГУ, 1984.
4. Дьяконов В. П. Справочник по алгоритмам и программам на языке Бейсик для ПЭВМ. — М.: Наука, 1987.
5. Калиткин Н. Н. Численные методы. — М.: Наука, 1978.
6. Каханер Д., Моулер К., Нэш С. Численные методы и программное обеспечение. — М.: Мир, 2001.
7. Кунин С. Вычислительная физика: Пер. с англ. — М.: Мир, 1992.
8. Пытьев Ю. П., Шишмарёв И. А. Курс теории вероятности и математической статистики для физиков. — М.: МГУ, 1983.
9. Рябенький В. С. Введение в вычислительную математику. — М.: Наука, 1994.
10. Свирежев Ю. М., Логофет Д. О. Устойчивость биологических сообществ. — М.: Наука, 1978.
11. Странные аттракторы (сборник). — М.: Мир, 1981.
12. Федоренко Р. П. Введение в вычислительную физику. — М.: МФТИ, 1984.

# Предметный указатель

## И



## К



## Л



## М

## Н



## О



## П



## Р

## С



## Т



## У

## Ф



## Ц



## Ч



## Ш



## Э



## Я