В.Дьяконов

MAPLE 7: УЧЕБНЫЙ КУРС

СПб.: Питер, 2002. — 672 с.

Известный специалист по системам компьютерной математики профессор В. Дьяконов предлагает в этой книге обширный учебный курс по новейшей версии одной из самых мощных и популярных систем компьютерной алгебры — Maple 7. Эта система позволяет решать в диалоговом режиме огромное число математических задач, от простых расчетов и задач численного моделирования до сложнейших аналитических преобразований и вычислений. В книге описан интерфейс программы, ее обширные возможности по выполнению самых разнообразных вычислений, мощные графические средства визуализации полученных результатов, удобный язык для задания команд в интерактивном и отложенном режимах, а также многочисленные пакеты, расширяющие и без того богатые возможности системы. Впервые описана поддержка языков MathML и XML, широко используемых в Интернете, и целый ряд новых пакетов. Особое внимание уделено визуализации результатов вычислений, а также полноте описания работы с программой.

Краткое содержание

Содержание

Алфавитный указатель

Предисловие

Автор данной книги, как и многие почитатели компьютерных вычислений, прошел долгий путь их реализации: от программируемых микрокалькуляторов до работы на малых и персональных ЭВМ, использующих универсальные языки программирования высокого уровня. Это нашло отражение в его ранних книгах [1–3].

Совсем недавно пользователь ЭВМ, решая даже простые численные задачи, был вынужден осваивать основы программирования и готовить кустарные программы, вряд ли нужные кому-либо еще, кроме их создателя. Между тем возможности компьютеров постоянно росли. Сейчас персональный компьютер (ПК) с микропроцессором класса Pentium II, III или 4 намного превосходит по своим возможностям первые ЭВМ, занимавшие целые комнаты и залы. А скорость вычислений нынешних ПК в сотни раз превосходит скорость вычислений легендарных IBM PC XT и AT (первых ПК) и вплотную приближается к скорости вычислений суперЭВМ недавнего прошлого.

В связи с этим стал меняться взгляд на назначение компьютера. На первое место вышло применение их для работы с текстовыми процессорами (например, Microsoft Word) и прикладными программными системами для автоматизации офисной деятельности. Увы, при этом многие пользователи стали забывать о том, что ЭВМ изначально создавались для вычислений, а вовсе не для замены ими популярной, но ставшей неудобной пишущей машинки. Развитие мультимедиа привело к бурному применению компьютеров в роли игровых автоматов. В результате главный стимул развития «электронного помощника» создается отнюдь не высокоинтеллектуальными задачами.

Однако времена меняются и вечные ценности, к коим принадлежат разум и образование, вновь возвращаются. В последние годы во всем мире существенно возрос интерес к серьезному применению ПК, в том числе в области математических расчетов. Этому в большой степени способствовала разработка специальных компьютерных математических программных систем, резко снизивших потребность в написании собственных программ при решении математических задач. Первое поколение таких систем [4–10] было ориентировано на операционную систему MS-DOS и появилось, казалось бы, совсем недавно — в начале 90-х гг.

Так или иначе, но компьютерный мир вновь заговорил об «искусственном интеллекте», понимая под этим способность электронной машины выдавать нетриви-

альные решения и обучаться решению новых задач. Интерес к компьютерному моделированию в самых широких областях заметно возрос после шахматных баталий между суперкомпьютером фирмы IBM и бывшим чемпионом мира по шахматам Гарри Каспаровым. Как известно, они завершились триумфальной победой машины — или, точнее говоря, коллективного разума тех, кто создал ее и ее программное обеспечение.

В последние годы показателем интеллектуальной мощи компьютеров, в том числе и персональных, стали уже не программы для игры в шахматы, а новейшие программные системы символьной математики или компьютерной алгебры [17–38]. Созданные для проведения символьных преобразований математических выражений, эти системы были доведены до уровня, позволяющего резко облегчить, а подчас и заменить, труд самой почитаемой научной элиты мира — математиков: теоретиков и аналитиков. Уже появились открытия, сделанные с помощью таких систем — но не ими самими! Вряд ли есть хоть один действительно серьезный научный проект, связанный с математикой, где они не применялись бы в деле.

Системы символьной математики долгое время были ориентированы на большие компьютеры. С появлением ПК класса IBM PC и Macintosh и с ростом их возможностей эти системы были переработаны под них и доведены до уровня массовых серийных программных систем. Сейчас системы символьной математики (или компьютерной алгебры) выпускаются самого разного «калибра» — от рассчитанной «на всех» системы Mathcad [11–21], поразительно компактной, быстрой и удобной для простых символьных вычислений системы Derive [22–24] и до компьютерных монстров Mathematica [26–28], MATLAB [29–31] и Maple [32–38], имеющих тысячи встроенных и библиотечных функций и изумительные возможности графической визуализации вычислений.

Все эти системы работают на персональных компьютерах, оснащенных популярными операционными системами класса Windows 95/98/NT/2000. Но не только на них — есть версии под операционные системы Linux, Unix, Mac и др. Они давно знакомы пользователям больших компьютеров и даже суперкомпьютеров.

К среднему уровню таких систем относятся интенсивно развиваемые системы класса Mathcad, имеющие (в дополнение к прекрасным средствам числовых вычислений) приобретенное по лицензии у фирмы Waterloo Maple Inc. (создателя систем Maple) ядро символьных вычислений. Ядро системы Maple используется и в другой маститой системе — MATLAB, придавая ей необычные для нее возможности символьной математики.

В данной книге впервые дается достаточно полное описание одной из самых мощных и интеллектуальных систем компьютерной алгебры — Maple под Windows, ее последней реализации — Maple 7. Эта система была создана группой ученых, занимающихся символьными вычислениями (The Symbolic Group), организованной Кейтом Геддом (Keith Geddes) и Гастоном Гонэ (Gaston Gonnet) в 1980 году в университете Waterloo, Канада. Вначале она была реализована на больших компьютерах и прошла долгий путь апробации, вобрав в свое ядро и библиотеки большую часть математических функций и правил их преобразований, выработанных математикой за столетия развития. Есть реализации программы на платформах ПК Macintosh, Unix, Sun и др.

Системам класса Maple посвящены сотни книг. Отметим лишь некоторые из них [39–56], изданные за рубежом. Достаточно полный список (около 400 наименований) книг по системам Maple можно найти на сайте разработчика этой системы — компании Waterloo Maple Software (www.maplesoft.com). Однако книг по системе Maple 7 (за исключением фирменных руководств по ней) на момент сдачи рукописи данной книги в этом списке не было.

Вряд ли эта мощная математическая система, разделяющая претензии на мировое лидерство с системами Mathematica фирмы Wolfram Research Inc., нужна секретарше или даже директору небольшой коммерческой фирмы. Но, несомненно, любая серьезная научная лаборатория или кафедра вуза должны располагать подобной системой, если они всерьез заинтересованы в автоматизации выполнения математических расчетов любой степени сложности. Несмотря на свою направленность на серьезные математические вычисления, системы класса Maple необходимы довольно широкой категории пользователей: студентам и преподавателям вузов, инженерам, аспирантам, научным работникам и даже учащимся математических классов общеобразовательных и специальных школ. Все они найдут в Maple многочисленные достойные возможности для применения.

По мнению автора данной книги, сравнение системы Maple 7 с лидером среди систем компьютерной математики — системой Mathematica 4.1 — непродуктивно. У каждой программы есть свои достоинства и недостатки. А главное — у них есть свои приверженцы, которых бесполезно убеждать, что иная система в чем-то лучше. Это все равно, что сравнивать великих исполнителей джазовой и рок-музыки Луи Амстронга и Би Би Кинга. Все, кто всерьез применяют системы компьютерной математики, должны работать с несколькими системами, ибо только это гарантирует высокий уровень надежности сложных вычислений.

И все же надо отметить, что интерфейс Maple 7 более интуитивно понятен, чем у строгой Mathematica 4.1. Maple 7 на первый взгляд имеет несколько менее мощную графику, но простота управления параметрами и легкость подготовки графических процедур часто позволяет визуализировать решения математических задач с меньшими усилиями, чем при использовании системы Mathematica 4.1. Обе системы в последних реализациях сделали качественный скачок в направлении эффективности решения задач в численном виде, в частности за счет повышения скорости выполнения матричных операций.

Особенно эффективно использование Maple при обучении математике. Высочайший «интеллект» этой системы символьной математики объединяется в ней с прекрасными средствами математического численного моделирования и просто потрясающими возможностями графической визуализации решений. Применение таких систем, как Maple, возможно при преподавании и самообразовании от самых основ до вершин математики.

Практика (да и личный опыт автора) показывает, что самым трудным является первый этап освоения системы. Первое знакомство с программой многих пользователей просто подавляет — убедившись в невероятном множестве возможностей системы и не имея ее систематизированного описания (а оно поставляется в

виде трех книг приличного размера, включая книги учебного характера [39, 40]), многие пользователи помещают систему в архив, где она «пылится» без дела.

Эта книга впервые основательно знакомит читателя с новейшей версией системы Maple — Maple 7 в форме подробного учебного курса. Книга написана на основе ранее изданного учебного курса по системе Maple 6 [37], существенно переработанного и дополненного. Среди новых материалов, появившихся в книге, следует отметить:

- полное описание инсталляции системы Maple 7;
- улучшенное описание интерфейса;
- обновление материала по информационной поддержке системы, в том числе через Интернет;
- адаптация примеров и рисунков к новой версии Maple 7;
- устранение замеченных в предыдущем издании [37] неточностей и опечаток;
- впервые представленное описание целого ряда новых встроенных математических пакетов системы (CurveFitting, LinearFunctionalSystems, PolynomialTools, OrthogonalSeries, RandomTools и др.);
- описание расширенной поддержки Интернета и встроенного пакета XMLTools, который ее обеспечивает;
- впервые приведенное описание новейших средств поддержки стандарта MathML и пакета MathML;
- описание многочисленных новинок системы Maple 7 в области реализации вычислений и их графической визуализации;
- десятки новых примеров применения Maple 7.

Основное внимание в книге уделено обучению основам и приемам эффективной работы с системой Maple 7. На роль всеобъемлющего справочного руководства книга не претендует, хотя во многих случаях способна выполнять роль справочника или руководства пользователя Maple 7. Хотя эта книга одна из самых обширных среди русскоязычных книг, посвященных системе Maple, автор был вынужден ограничить описание ряда важных возможностей программы. Основной акцент в книге сделан на описание возможностей системы в области математических расчетов и интерфейса пользователя. Менее подробно, а порой просто не описаны средства расширения системы и пакеты узкоспециального назначения (электронная база данных по системе Maple 7 дает полное представление о всех свыше 3000 операторов и функций этой системы, большинство из которых используется крайне редко).

Хотя книга посвящена версии Maple 7, большинство материала может использоваться и пользователями предшествующей версии Maple 6. В частности, это достигнуто выделением новых возможностей Maple 7 в отдельные разделы книги. Остальные разделы книги без каких-либо ограничений могут быть отнесены к наиболее распространенной версии Maple — Maple 6.

Структура книги

Книга содержит 17 уроков и составлена так, что эти уроки постепенно знакомят читателя с возможностями системы. Уже после прохождения урока 1 вы сможете начать осмысленно и плодотворно работать с системой. Этот урок является как бы ознакомительным курсом по работе с системой Maple 7. Он может быть полезен как для быстрого самостоятельного освоения системы не слишком требовательным пользователем, так и как основа вводного курса по системе в вузах и школах, где для основательного изучения Maple 7 не предусмотрено достаточного количества учебных часов. Кроме того, этот урок знакомит читателя с основами интерфейса пользователя и правилами работы с панелями инструментов и форматирования выражений. По сравнению с учебным курсом по системе Maple 6 [37] этот урок существенно переработан и дополнен. В частности, подробно описана инсталляция системы Maple 7 и аппаратные требования для работы с ней.

Последующие уроки расширяют заведенное знакомство и постепенно готовят читателя к серьезной самостоятельной работе практически без применения какой-либо иной документации, кроме встроенной в систему справочной базы данных. Урок 2 посвящен знакомству с мощной справочной базой данных системы Maple 7 и информационной поддержкой этой системы в Интернете. Данные разделы намеренно вынесены в начало книги, поскольку успех освоения системы Maple 7 до профессионального уровня требует обучения работе со справочной базой данных.

В уроке 3 описаны основные приемы работы с файлами документов, которые готовятся в Maple 7. Урок 4 дает систематизированное описание интерфейса, хотя и без многих подробностей, известных всем пользователям операционных систем класса Windows 95/98/NT/2000.

Остальные уроки посвящены базовым математическим возможностям системы Maple 7 и основам практической работы в ней. В уроке 5 описаны основные типы данных системы, а в уроке 6 — основные виды встроенных операторов и функций. Урок 7 посвящен основам программирования в среде Maple 7. Читатель должен понимать, что все описанные и в других главах команды системы Maple 7 являются одновременно и командами ее языка программирования. Именно это позволяет считать язык программирования Maple 7 языком программирования сверхвысокого уровня, проблемно ориентированным на математические расчеты.

Урок 8 является одним из наиболее важных. Он посвящен решению типовых задач математического анализа, таких как вычисление сумм и произведений последовательностей, производных и интегралов, разложений функций в ряд и т. д. При этом особое внимание уделяется технике аналитических вычислений, где возможности системы Maple 7 вызывают особый интерес. Но и техника численных расчетов рассмотрена достаточно детально, в частности арифметика высокой точности.

Работа с функциями и степенными многочленами (полиномами) описана в уроке 9. Урок 10 посвящен изучению главной «козырной карты» системы Maple 7 — ее возможностям в области символьной математики. Здесь описано много тонкостей работы с математическими выражениями и другими объектами системы, позволяющими выполнять множество математических преобразований и подстановок.

Два больших урока — уроки 11 и 12 — посвящены графическим возможностям системы. При этом урок 11 описывает обычные графические средства, а в уроке 12 дается описание расширенных средств, позволяющих эффективно решать задачи визуализации решений математических проблем — вплоть до подготовки графиков с элементами анимации.

Учитывая огромную роль дифференциальных уравнений в решении ряда математических, физических и технических задач, работе с ними посвящен отдельный урок 13. Наряду с решением одиночных дифференциальных уравнений первого и второго порядка рассматривается решение систем линейных и нелинейных дифференциальных уравнений как аналитическими, так и численными методами. Большое внимание уделено графической визуализации решений и построению наглядных фазовых портретов решения. В уроке отражены новые возможности Maple 7 в решении дифференциальных уравнений.

В уроке 14 рассмотрены важнейшие пакеты системы Maple 7 математической направленности. Эти пакеты поставляются вместе с системой, и применение функций из них столь же важно, как и применение средств ядра системы. Описанные в уроке 14 пакеты рассмотрены достаточно полно.

В отдельный урок 15 вынесены широко используемые на практике средства решения задач линейной алгебры. Это операции с векторами и матрицами, различные их преобразования и техника решения систем линейных уравнений. Здесь описаны такие важные пакеты, как linalg (стандартные средства линейной алгебры) и LinearAlgebra. В последний пакет входят новые средства линейной алгебры повышенной эффективности на основе алгоритмов, заимствованных из знакомого математикам пакета программ NAG (Numbering Algorithms Group). Впервые описан новый пакет анализа линейных функциональных систем LinearFunctionalSystems, появившийся в версии Maple 7.

Остальные пакеты, относящиеся к сравнительно узким областям математики и представляющие ограниченный интерес для большинства читателей, рассмотрены обзорно или в виде аннотации в уроке 16. К сожалению, материал по всем пакетам расширения Maple 7 настолько обширен, что его невозможно отразить в одной книге (тем более в форме учебного курса). Тем не менее в отличие от учебного курса по системе Maple 6 [37] этот урок существенно расширен и в нем впервые описан ряд новых пакетов системы Maple, в частности пакеты PolymomialTools, OrthogonalSeries, RandomTools, MathML и XMLTools.

Последний урок 17 описывает законченное решение ряда конкретных и интересных задач из области математики, физики и радиоэлектроники. Таким образом, читатель получает возможность познакомиться с широким спектром применения системы Maple 7 — от примеров простых расчетов и вычислений (таких в книге тысячи) до решения конкретных научных и технических проблем.

Материал книги иллюстрируется многими сотнями копий экрана как в виде отдельных рисунков, так и фрагментов вычислений и программных процедур в тексте книги. Они дают наглядное представление о реальном диалоге с системой и о форматах ввода и вывода.

Большинство примеров в книге оригинальны и отражают взгляд автора на методологию изучения системы Maple. В то же время в книге использованы и лучшие

(и наиболее поучительные) примеры, которые приведены в обширной библиотеке процедур, составленных пользователями систем Maple разных реализаций со всего мира, и примеры из ряда учебников по системе — в том числе новейших электронных (таких, как Power Tools), размещенных в Интернете. Все заимствованные примеры также специально адаптированы применительно к новейшей версии системе Maple 7, описанной в книге.

Некоторые замечания

Перед вами — уже третья книга автора по системам Maple, которая содержит результаты многолетней работы с одной из самых мощных математических систем компьютерной алгебры — Maple, начиная с ее первой реализации и кончая новейшей Maple 7. Подобные программы, по мнению автора, являются куда более интересными и поучительными объектами для изучения, чем обычные языки программирования (по которым, кстати, защищены сотни диссертаций во всем мире).

Однако пусть научный характер приложений систем класса Maple не пугает читателей. Книга, по мере возможностей и пристрастий автора, написана достаточно простым языком и вполне может претендовать на роль не только серьезного учебного пособия, но и руководства пользователя по системам Maple, причем вне зависимости от того, на какой компьютерной платформе эта система реализована и кто является ее пользователем. Хотя основное внимание уделялось, естественно, тем реализациям, которые работают в среде наиболее массовой операционной системы Windows 95/98/NT. Тем не менее автор приносит извинения тем пользователям, которые не найдут в книге описания отдельных тонкостей интерфейса Maple для не Windows-платформ.

Автор далек от мысли, что данная книга лишена недостатков и погрешностей. Она готовилась в спешном порядке, что имеет свои достоинства и недостатки. Именно благодаря ускоренной подготовке книги как автором, так и издательством, она попадет в руки читателей до того, как на рынок выйдет очередная версия системы Maple 8 или Maple 9. Автор и издательство постарались сделать все возможное, чтобы последствия этой спешки не сказались на качестве данной книги. Тем не менее автор считает своим долгом заведомо извиниться перед читателями, которые могут не обнаружить в книге нужные именно им сведения или если отдельные из них окажутся не вполне точными.

Эта книга ориентирована на определенный, но достаточно широкий круг читателей — это прежде всего научные работники, преподаватели и учащиеся университетов и вузов, аспиранты, инженеры и все пользователи персональных компьютеров, применяющие или осваивающие математические методы решения задач в любой области науки, техники и образования.

Книга является учебным курсом только по стилю изложения материала и рубрикации книги. По своей полноте она вполне может претендовать на справочное пособие по системе Maple 7, хотя исчерпывающего справочного материала книга не содержит, да и не может содержать, поскольку его объем многократно превосходит объем данной, отнюдь не тонкой книги. Поэтому автор приносит извине-

ния всем тем читателям, кто остался неудовлетворенным полнотой описаний некоторых функций или команд системы. Как уже отмечалось, полное описание всех функций и команд есть только в объемном электронном справочнике по системе.

В этой книге намеренно подобраны достаточно простые примеры для иллюстрации возможностей системы Maple. Если среди читателей найдутся те, кого волнуют большие математические формулы, то им можно посоветовать найти корни простого многочлена третьей или четвертой степени в аналитическом виде. Надо полагать, что эти простые примеры сразу дадут понять таким «максималистам», что система Maple способна порождать при решении некоторых задач сложнейшие и практически необозримые формулы. Задача при написании книги была не в том, чтобы напичкать ее громоздкими формулами (что можно было сделать в два счета и чего уж и так хватает), а именно в подборе простых и наглядных примеров.

Благодарности и адреса

Эта книга, как и ряд последних книг автора, подготовлена в инициативном порядке в рамках работ кафедры физической и информационной электроники Смоленского государственного педагогического университета. В нее вошли материалы работ по гранту Минобразования РФ, полученному автором для выполнения работ по применению компьютерных математических систем в решении задач естествознания. Гранты Соросовского профессора по математике 1999 и 2001 гг. и прямая поддержка корпорации Waterloo Maple и ее представителя в России фирмы SoftLine стали стимулом для подготовки очередной книги по системам Maple.

Автор выражает особую благодарность сотрудникам вычислительного центра Заочного университета города Хагена (Германия), обратившим внимание автора на систему Maple V еще в самом начале ее появления на персональных компьютерах и предоставившим ее первую реализацию.

Разумеется, эта книга не могла бы появиться без «курочки», снесшей такое «золотое яичко», каким является система Maple 7. Поэтому высшей благодарности заслуживает фирма Waterloo Maple Inc. — создатель этого уникального, мощного и прекрасно реализованного программного продукта. Автор выражает благодарность директору Российской фирмы SoftLine И. П. Боровикову, предоставившему автору новейшую версию Maple 7 сразу после ее появления на рынке.

Автор выражает горячую признательность дружному коллективу компьютерной редакции издательства «Питер», в особенности заведующему редакцией Илье Корнееву и редактору Дмитрию Лещеву, благодаря оперативной работе которых книга увидит свет в значительно улучшенном, причем в кратчайшие сроки, качестве.

Автор благодарен главному конкуренту фирмы Waterloo Maple Software — компании Wolfram Research Inc. (США), создавшей систему Mathematica, за предоставленную возможность длительной научной стажировки в этой фирме. Во время этой стажировки автор по-новому взглянул на современные системы символьной математики. И пришел к твердому убеждению, что разработчики подобных систем не столько конкуренты, сколько партнеры по большому и важному делу

внедрения систем компьютерной математики в образование, науку и технологии всех стран мира. Предоставленные автору условия были настолько хороши, что позволили, в частности благодаря подключению к скоростному Интернету, получить массу новых материалов по всем системам компьютерной математики, в том числе как по Mathematica 4/4.1, так и по Maple 6/7.

Наконец, автор не может не высказать признательности генеральному директору фирмы «Телепорт» (г. Смоленск) Г. И. Рухамину за оперативное подключение домашнего компьютера автора к глобальной сети Интернет, что позволило включить в книгу самые свежие материалы по системе Maple и поддерживать оперативную связь с разработчиком этой системы. Автор благодарит своих коллег И. В. Абраменкову, Р. Е. Кристаллинского и А. Г. Лучаинова за обсуждение ряда разделов книги и наброски отдельных примеров, использованных при подготовке последнего урока в книге.

Эта книга входит в серию книг с пометкой «Учебный курс». Это отражает заинтересованность автора во внедрении современных систем компьютерной математики в российское образование. Автор положительно относится к критическим и позитивным откликам и пожеланиям по данным книгам. Их можно отправлять как на адрес издательства, так и по месту работы автора — 21400, Смоленск, ул. Пржевальского 14, СГПУ, В. П. Дьяконову. Вы можете также направлять свои отзывы по электронной почте автора (dyak@keytown.com) или издательства «Питер», выпустившего данную книгу.

К фирме Waterloo Maple Inc. можно обращаться по адресу: Waterloo Maple Inc., 57 Erb Street West, Waterloo, ON, Canada N2L 6C2. E-mail: support@maplesoft.com. Phone: (519) 747-2505. Fax: (519) 747-5284. Web-сервер фирмы: www.maplesoft.com.

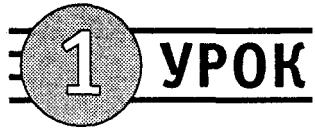
С фирмой SoftLine — представителем интересов Waterloo Maple в России — можно связаться по телефону (095) 232-0023, по факсу (005) 126-9065 или по электронной почте (info@softline.ru). С этой фирмой можно ознакомиться и на ее web-сервере: www.softline.ru.

От издательства

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

Мы будем рады узнать ваше мнение!

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



Первое знакомство с системой Maple 7

- ❒ Краткая характеристика систем класса Maple
- ❒ Об ошибках в символьных вычислениях
- ❒ Ядро и пакеты расширения Maple 7
- ❒ Языки системы
- ❒ Ориентация систем Maple
- ❒ Загрузка Maple и ее составляющие
- ❒ Интерфейс системы и работа в диалоговом режиме
- ❒ Меню системы Maple 7
- ❒ Управление формой представления документа
- ❒ Повышение эффективности работы с системой
- ❒ Доступ к справке и примерам

Краткая характеристика систем класса Maple

Назначение и место систем Maple

Maple — система компьютерной математики, рассчитанная на широкий круг пользователей. До недавнего времени ее называли системой компьютерной алгебры, что указывало на особую роль символьных вычислений и преобразований, которые способна осуществлять эта система. Но такое название сужает сферу применения системы. На самом деле она уже способна выполнять быстро и эффективно не только символьные, но и численные расчеты, причем сочетает это с превосходными средствами графической визуализации и подготовки электронных документов.

Казалось бы, нелепо называть такую мощную систему, как Maple 7 математической системой «для всех». Однако по мере ее распространения она становится полезной для многих пользователей ПК, вынужденных в силу обстоятельств (работа, учеба, хобби) заниматься математическими вычислениями и всем, что с ними связано. А все это простирается от решения учебных задач в вузах до моделирования сложных физических объектов, систем и устройств, и даже создания художественной графики (например, фракталов).

Для наших читателей (в том числе и для математиков-профессионалов) возможности систем символьной математики, реализованных на массовых ПК класса IBM PC, порой являются полной неожиданностью и вызывают вполне заслуженное удивление и восхищение, но иногда и резкое отрицание. Впрочем, последнее характерно скорее для тех, кто с системой Maple просто не работал и относится к ней, как дама из анекдота о паровозе — увидев паровоз впервые, она воскликнула: «Не может быть, что он едет без лошадей!»

Maple — тщательно и всесторонне продуманная система компьютерной математики. Она с равным успехом может использоваться как для простых, так и для самых сложных вычислений и выкладок. Заслуженной популярностью системы Maple (всех версий) пользуются в университетах — свыше 300 самых крупных университетов мира (включая и наш МГУ) взяли эту систему на вооружение. А число только зарегистрированных пользователей системы уже давно превысило один миллион. Ядро системы Maple используется в ряде других математических систем, например в MATLAB и Mathcad, для реализации в них символьных вычислений.

Добавьте к этому куда большее число незарегистрированных пользователей — ведь система записана на многих компакт-дисках, лихо продаваемых в России

по вполне доступным ценам. Если учесть все это, то оказывается, что популярность системы Maple ничуть не ниже, а то и выше, чем у гораздо более простых систем, таких как Derive и Mathcad. Вот и решайте, какая из систем и впрямь рассчитана на всех!

Maple — типичная интегрированная система. Она объединяет в себе:

- мощный язык программирования (он же язык для интерактивного общения с системой);
- редактор для подготовки и редактирования документов и программ;
- современный многооконный пользовательский интерфейс с возможностью работы в диалоговом режиме;
- мощную справочную систему со многими тысячами примеров;
- ядро алгоритмов и правил преобразования математических выражений;
- численный и символьный процессоры;
- систему диагностики;
- библиотеки встроенных и дополнительных функций;
- пакеты функций сторонних производителей и поддержку некоторых других языков программирования и программ.

Ко всем этим средствам имеется полный доступ прямо из программы. Maple — одна из самых мощных и «разумных» интегрированных систем символьной математики, созданная фирмой Waterloo Maple, Inc. (Канада).

Во многих обзорах систем компьютерной алгебры Maple справедливо считается одним из первых кандидатов на роль лидера среди них. Это лидерство она завоевывает в честной конкурентной борьбе с другой замечательной математической системой — Mathematica 4.1. Каждая из данных двух систем имеет свои особенности, но в целом эти две лидирующие системы практически равноценны. Однако надо отметить, что появление новейшей версии Maple 7 означает очередной виток в соревновании этих систем за место лидера мирового рынка. Причем виток на этот раз раньше сделала система Maple 7.

Система Maple прошла долгий путь развития и апробации. Она реализована на больших ЭВМ, рабочих станциях Sun, ПК, работающих с операционной системой Unix, ПК класса IBM PC, Macintosh и др. Все это самым положительным образом повлияло на ее отработку и надежность (в смысле высокой вероятности правильности решений и отсутствия сбоев в работе). Не случайно ядро системы Maple V используется целым рядом других мощных систем компьютерной математики, например системами класса Mathcad и MATLAB. А совсем недавно упрощенная версия Maple для операционной системы Windows CE стала использоваться в миниатюрных компьютерах фирмы Casio — Cassiopeia.

Версии систем класса Maple

Известен ряд версий системы Maple, называемых реализациями. Одной из самых известных реализаций является реализация Maple V R5. В ней появилась

возможность работы с электронными таблицами, несколько улучшен интерфейс пользователя (введены палитры для ввода математических символов и расширены возможности управления мышью), стала возможной запись файлов в формате HTML и введена возможность обмена объектами между документами методом перетаскивания (Drag and Drop).

Основное достоинство предшествующей версии Maple 6 — это существенное ускорение вычислений с большими матрицами, достигнутое применением алгоритмов матричных вычислений известного пакета NAG (Numbering Algorithms Group). Хотя данная книга посвящена новейшей реализации системы Maple 7, ее основной материал будет полезен и пользователям реализации Maple 6.

Новейшая версия систем Maple — Maple 7 появилась 21 июня 2001 г. Корпорация Waterloo Maple оценивает ее появление как новый виток в борьбе за мировое лидерство в области автоматизации математических вычислений — как численных, так и, в особенности, символьных. Являясь одними из лучших и надежных систем компьютерной математики, Maple 6 и Maple 7 становятся мировым стандартом в области математических вычислений.

Об ошибках в символьных вычислениях

На многих пользователей систем символьной математики удручающее впечатление может произвести наличие хотя и редких, но ошибочных решений. В самом деле, мы немедленно стерли бы с жесткого диска табличный процессор, давший ошибку в бухгалтерских расчетах, и перестали бы доверять системе проверки орфографии, дающей ошибки при проверке. Впрочем, последнее случается сплошь и рядом — пока нет таких систем, которые корректно проверяли бы орфографию и грамматику. Тот же текстовый процессор Word 97/2000 постоянно ошибается при проверке орфографии текстов, в чем автор не раз убеждался, готовя с его помощью большие книги.

У систем компьютерной алгебры нет проблем с обработкой естественного языка — математика полностью формализованная наука. Однако в них много своих условностей и неоднозначностей, которые здесь как бы заранее запрограммированы. К примеру, что считать более простым выражением: $\tan(x)$ или $\sin(x)/\cos(x)$? Система Derive полагает более простым выражением $\tan(x)$ и преобразует к нему выражение $\sin(x)/\cos(x)$. А вот система Maple V ничуть не менее справедливо считает, что функции $\sin(x)$ и $\cos(x)$ математически более простые, чем $\tan(x)$, и вообще — $\tan(x)$, по сути, не самостоятельная функция, а $\sin(x)/\cos(x)$. Поэтому Maple V везде вместо $\tan(x)$ будет выводить $\sin(x)/\cos(x)$.

Представьте себе, что таких условностей десятки и вы ничего об этом не знаете. Поэтому не стоит удивляться, что символьное значение какой-либо производной или интеграла может заметно отличаться по виду от приводимого в том справочнике, из которого взято исходное выражение для проверки правильности работы системы. Часто, чтобы получить результат в необходимом виде, необходимо приложить определенные усилия либо дать конкретные указания системе о типе преобразований в ходе вычислений. Указания реализуются в виде параметров к командам и функциям системы.

По образному выражению автора обзора [40], решение задач в символьном виде напоминает переход через поле, густо напичканное минами. Удивительно не то, что системы символьной математики могут ошибаться и «взрываться», а то, что число этих ошибок мало и уже на нынешнем этапе развития таких систем это не мешает их серьезному практическому применению. Стоит еще раз подчеркнуть, что Maple в этом отношении является одной из лучших систем, реализованных на ПК класса IBM PC и Macintosh с достаточно умеренными техническими характеристиками. Кстати говоря, для ПК Macintosh последней реализацией пока что является Maple V R4.

Один знакомый автор любил говорить, что компьютеры делают умных людей умнее, а глупых — глупее. Пожалуй, это более чем справедливо для людей, сидящих у ПК с установленной на нем системой символьной математики. Лишь те, кто понимают суть математических вычислений и имеют должную математическую интуицию и подготовку, могут получить от таких систем самые серьезные и даже новые результаты. Те же, кто думает, что системы символьной математики заменят им математические знания, глубоко ошибаются и могут получить красочно выглядящие, но абсолютно неверные и даже псевдонаучные результаты!

Однако вряд ли следует утрировать вероятность выдачи системами символьной математики ошибочных результатов — даже самые опытные математики-аналитики тоже могут ошибаться в своих вычислениях. В разработке таких систем, как Maple или Mathematica принимают участие крупные математические школы всего мира! Эти системы — кладезь математических понятий, сведений и знаний. Они способны заменить самые серьезные справочники по математическим вычислениям в любой области науки, техники и образования. Кроме того, они имеют множество средств для проверки корректности выполняемых вычислений, например путем подстановки полученных результатов в исходные выражения.

Кстати, одно из самых действенных приемов проверки таких средств — решение задачи одновременно на нескольких системах символьной математики. Не случайно уже сейчас можно заметить тенденцию к объединению математических систем. Эта новая и безусловно прогрессивная тенденция в ближайшее время приведет к созданию автоматизированных рабочих мест математиков и ученых других близких специальностей. Разработки таких рабочих мест (разумеется, компьютер на них — главный инструмент), в том числе с использованием систем Maple, уже появились и о них немного говорится в заключении.

В добавление к сказанному надо отметить, что Maple 7 — одна из самых надежных систем компьютерной математики. Надежных прежде всего в смысле высокой достоверности получения правильных результатов при сложных символьных вычислениях. Эта первая система компьютерной математики, успешно прошедшая полное тестирование на задачах повышенной сложности, предлагаемых для оценки качества подобных систем.

Ядро и пакеты Maple 7

Основой для работы с символьными преобразованиями в Maple является ядро системы. Оно содержит сотни базовых функций и алгоритмов символьных пре-

образований. В новых реализациях объем ядра достигает 6–7 Мбайт. Имеется также основная библиотека операторов, команд и функций. Многие встроенные в нее функции, как и функции ядра, могут использоваться без какого-либо объявления, другие нуждаются в объявлении. Кроме того, имеется ряд подключаемых пакетов (packages).

Дополнительные функции из пакетов могут применяться после объявления подключения пакета с помощью команды `with(name)`, где `name` — имя применяемого пакета. Общее число функций, с учетом встроенных в ядро и размещенных в пакетах в системе Maple V R4 приближается к 2500, в реализации R5 — к 2700, а в Maple 6 и в Maple 7 оно уже превышает 3000. Это означает, что большинство задач может решаться в режиме прямого диалога с системой без использования каких-либо средств программирования.

Языки системы Maple 7

Maple способна решить огромное число задач вообще без какого-либо программирования в общепринятом смысле этого понятия. Достаточно лишь описать алгоритм решения задачи и разбить его на отдельные вопросы, на которые система Maple способна дать ответы. Более того, есть тысячи задач, алгоритмы решения которых уже реализованы в виде функций и команд системы. Тем не менее это вовсе не означает, что в Maple нельзя программировать. На самом деле Maple поддерживает три собственных языка: входной, реализации и программирования.

Maple имеет входной язык сверхвысокого уровня, ориентированный на решение математических задач практически любой сложности. Он служит для задания системе вопросов или, говоря иначе, задания входных данных для последующей их обработки. Это язык интерпретирующего типа и по своей идеологии напоминает добрый старый Бейсик. И такое сходство вовсе не недостаток, а огромное достоинство — ведь именно с Бейсика начался подлинный диалог пользователя напрямую с компьютером! Входной язык имеет большое число заранее определенных математических и графических функций, а также обширную библиотеку, подключаемую по мере необходимости.

Имеет Maple и свой язык процедурного программирования — Maple-язык. Этот язык имеет вполне традиционные средства структурирования программ: операторы циклов, операторы условных и безусловных переходов, операторы сравнения, логические операторы, команды управления внешними устройствами, функции пользователя, процедуры и т. д. Он также включает в себя все команды и функции входного языка, ему доступны все специальные операторы и функции. Многие из них являются весьма серьезными программами, например символьное дифференцирование, интегрирование, разложение в ряд Тейлора, построение сложных трехмерных графиков и т. д.

Не следует путать входной язык и язык программирования системы (Maple-язык) с языком ее реализации. Им является один из самых лучших и мощных универсальных языков программирования — Си. На нем написано ядро системы, содержащее тщательно оптимизированные процедуры. Большинство же

функций, которые содержатся в пакетах, написаны на Maple-языке, благодаря чему их можно модифицировать и даже писать свои собственные библиотеки. По разным оценкам, лишь от 5 до 10 % средств Maple создано на языке реализации — все остальное написано на Maple-языке. Таким образом, система имеет развитые возможности для расширения и адаптации к задачам пользователя.

Для подготовки программ на языке Maple могут использоваться внешние редакторы, но система имеет и свой встроенный редактор, вполне удовлетворяющий требованиям большинства пользователей. Он открывается командами New и Open в меню File. Этот редактор можно использовать для редактирования файлов программ или математических выражений. Версии Maple для MS-DOS имеют свой редактор программ и отладчик с функциями проверки синтаксиса. После версии Maple V для Windows необходимость в этих средствах практически отпала.

Синтаксис структурных операторов языка Maple напоминает смесь Бейсика и Паскаля. Это облегчает знакомство с ним тем, кто имеет хотя бы начальный опыт программирования на этих языках. По близким к Бейсику правилам (и при помощи общепринятых математических сокращений) выполняется и ввод математических выражений в диалоговом режиме работы с системой.

Ориентация систем Maple

Вообще говоря, системы Maple ориентированы на решение сложных задач, хотя и решение в них простых задач вполне возможно и уместно. Возможно, для решения таких задач вполне подойдет весьма простая, быстрая и надежная система Derive или система Mathcad, в которую (начиная с версии 3.0 под Windows) включен приобретенный по лицензии фирмы Waterloo Maple упрощенный символьный процессор Maple. Однако по числу доступных пользователю математических функций эти скромные системы не идут ни в какое сравнение с патриархом символьной математики — системой Maple.

Система Maple 7 может с успехом применяться для решения самых серьезных математических задач аэродинамики, теории поля, теплопроводности и диффузии, теоретической механики и др. Решение таких задач нередко является многолетним трудом элитных научных коллективов.

Впрочем, поскольку система может быть установлена на любом современном ПК, ее можно (да и нужно) применять как можно чаще и по любому поводу. Это способствует как приобретению практических навыков работы с Maple, так и росту математических познаний тех, кто с ней работает.

Возможности предшествующей версии Maple 6

Перечислим основные возможности предшествующей версии системы Maple 6.

Интерфейс:

- работа со многими окнами;
- вывод графиков в отдельных окнах или в окне документа;

- представление выходных и входных данных в виде естественных математических формул;
- задание текстовых комментариев различными шрифтами;
- возможность использования гиперссылок и подготовки электронных документов;
- удобное управление с помощью клавиатуры через главное меню и инструментальную панель;
- управление с помощью мыши.

Символьные и численные вычисления:

- дифференцирование функций;
- численное и аналитическое интегрирование;
- вычисление пределов функций;
- разложение функций в ряды;
- вычисление сумм и произведений;
- интегральные преобразования Лапласа, Фурье и др.;
- дискретные Z-преобразования;
- прямое и обратное быстрое преобразование Фурье;
- работа с кусочно-заданными функциями.

Работа с уравнениями в численном и символьном виде:

- решение систем линейных и нелинейных уравнений;
- решение систем дифференциальных уравнений;
- символьное вычисление рядов;
- работа с рекуррентными функциями;
- решение трансцендентных уравнений;
- решение систем с неравенствами.

Работа с функциями:

- вычисление значений всех элементарных функций;
- вычисление значений большинства специальных математических функций;
- пересчет координат точек между различными координатными системами;
- задание функций пользователя.

Линейная алгебра:

- свыше ста операций с векторами и матрицами;
- решение систем линейных уравнений;
- формирование специальных матриц и их преобразования;
- вычисление собственных значений и собственных векторов матриц;
- поддержка быстрых векторных и матричных алгоритмов пакета программ NAG.

Графическая визуализация результатов вычислений:

- построение графиков многих функций;
- различные типы осей (с линейным и логарифмическим масштабом);
- графики функций в декартовой и полярной системах координат;
- специальные виды графиков (точки массивов, векторные графики, диаграммы уровней и др.);
- системы координат, определяемые пользователем;
- графики, представляющие решения дифференциальных уравнений;
- графики трехмерных поверхностей с функциональной закраской;
- построение пересекающихся в пространстве объектов;
- задание пользователем окраски графиков;
- импорт графиков из других пакетов и программных систем;
- анимация графиков;
- создание и проигрывание анимационных файлов.

Программирование:

- встроенный язык процедурного программирования;
- простой и типичный синтаксис языка программирования;
- обширный набор типов данных;
- типы данных, задаваемых пользователем;
- средства отладки программ;
- мощные библиотеки функций;
- задание внешних функций и процедур;
- поддержка языков программирования C и Fortran;
- возможность записи формул в формате LaTeX.

Новые возможности системы Maple 7

Система Maple 7 приобрела ряд новых возможностей. Кратко отметим их:

- расширенная поддержка численных алгоритмов пакета программ NAG, в том числе при решении численных задач математического анализа (например, вычисление определенных интегралов в Maple 7 ускорено в 20–40 раз в сравнении с Maple 6) и при решении дифференциальных уравнений;
- новый обучающий курс User's Tour, встроенный в ее справку;
- существенно переработанные и обновленные пакеты функций;
- ускоренные алгоритмы целочисленных вычислений (например, факториал числа 25000 вычисляется более чем на порядок быстрее, чем системой Maple 6);

- обширный набор новых алгоритмов решения дифференциальных уравнений, обеспечивающий дополнительную эффективность решения задач в области моделирования физических явлений и устройств;
- выполненное впервые 100% успешное испытание при решении специальных тестовых задач, что является высшим достижением на рынке средств компьютерной математики;
- усовершенствованные и новые алгоритмы реализации многих численных методов решения задач;
- новые встроенные пакеты аппроксимации кривых CurveFitting, внешних вычислений ExternalCalling, решения линейных функциональных систем LinearFunctionalSystem, ортогональных рядов OrthogonalSeries, работы с полиномами PolynomialTools, решения уравнений SolveTools и поддержки вычислений с размерными величинами Units;
- новый пакет для поддержки языка XML;
- поддержка новейшего стандарта записи математической информации — языка MathML 2.0;
- улучшение пользовательского интерфейса, в частности введение новой палитры ввода шаблонов векторов;
- поддержка протокола TCP/IP, обеспечивающего динамический удаленный доступ к данным, например, для финансового анализа в реальном масштабе времени или данных о погоде;
- дополнительные пакеты (Maple PowerTools™), доступные через Интернет, поддерживающие анализ методом конечных элементов (FEM), нелинейную оптимизацию и статистику, а также три новых пакета: вычисления для новичков, теоретическая физика и программирование;
- возможность работы с курсом университетского математического образования, загружаемого через Интернет.

В сочетании с сохраненными возможностями предшествующей версии системы это дает новой версии Maple 7 обширные возможности в эффективном решении широкого класса математических и научно-технических задач, а также задач в области образования.

Установка системы Maple 7 на ПК

Аппаратные требования

В этой книге рассматривается реализация системы Maple 7, требующая:

- процессор — Pentium 150 МГц и выше;
- оперативная память — не менее 16 Мбайт (рекомендуется 32 Мбайта и выше);
- место на жестком диске — около 80 Мбайт (120 Мбайт для сетевой версии).

Обширная библиотека пользователя, поставляемая с предшествующими версиями системы Maple V, с новыми версиями Maple 6 и Maple 7 пока не поставляется, поскольку требует серьезной переработки. Достаточно отметить, что использование в новых версиях системы оператора % для вызова последнего результата (вместо применяемого ранее оператора кавычек) делает неработоспособными большинство программ этой библиотеки. Также в новой версии изменен синтаксис некоторых функций.

Установка системы Maple 7

Установка системы производится обычно с компакт-диска. Обычно компакт-диск с инсталляционной версией Maple 7 стартует автоматически. При этом появляется окно Мастера установки со шкалой выполнения предварительной подготовки, показанное на рис. 1.1.

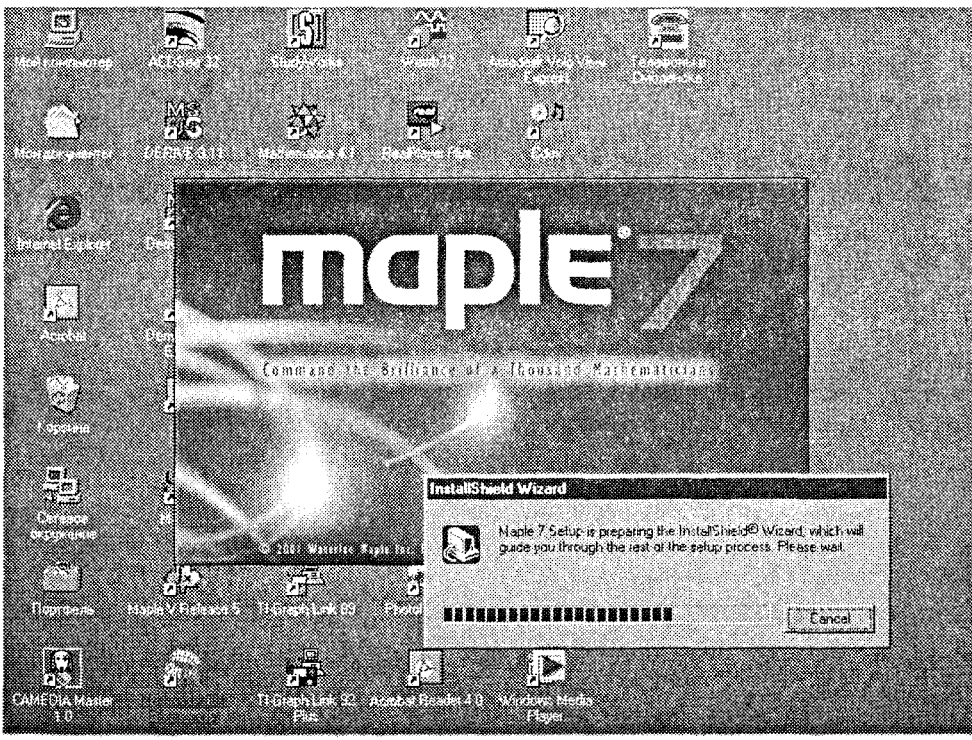


Рис. 1.1. Окно Мастера установки Maple 7

Затем Мастер автоматически переходит к окну, в котором просит ввести серийный номер программного продукта, который имеется в документации к системе или указан на конверте с компакт-диском.

Для продолжения установки следует щелкнуть на кнопке Next. Это приводит к появлению следующего окна и т. д. В каждом окне после указания необходи-

мых данных нужно нажать кнопку Next или Yes. Работа с остальными окнами мастера установки описана ниже.

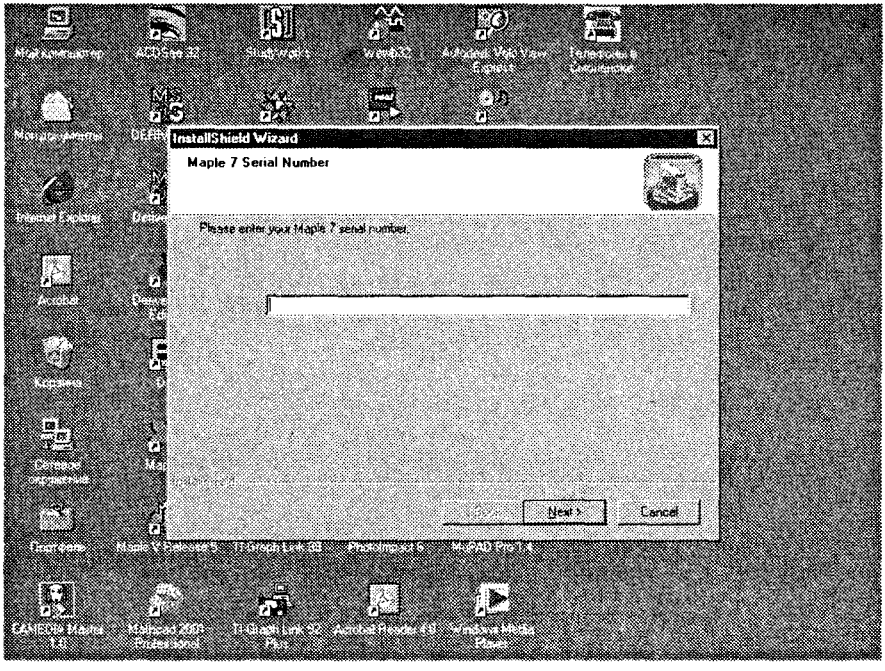


Рис. 1.2. Окно с приглашением к установке системы Maple 7

На рис. 1.3 показано следующее окно Мастера, которое служит для задания папки, в которой будут установлены файлы Maple 7. Вы можете оставить предлагаемую по умолчанию папку, а можете ввести любую другую.

Если папка, заданная по умолчанию в поле Destination Folder вас не устраивает, то вы можете определить другую. Для этого щелкните на кнопке Browser. Появится стандартное окно Choose Folder, показанное на рис. 1.4. В этом окне можно задать желаемую папку, либо явно указав ее имя в поле Path, либо выбрав существующую в дереве Directories.

Система Maple 7 может быть установлена как для индивидуального пользователя, так и для многих пользователей (сетевой вариант). Данный выбор можно осуществить в следующем окне Мастера установки, показанном на рис. 1.5. В этом окне надо задать вариант установки, щелкнув в соответствующем положении переключателя. Здесь же можно уточнить директорию (по умолчанию Users) для хранения файлов пользователей.

Следующий этап установки — определение имени ярлыка, активизацией которого в дальнейшем будет запускаться система. Для этого служит окно Мастера установки, показанное на рис. 1.6. В этом окне можно оставить имя ярлыка Maple 7, предлагаемое по умолчанию, ввести вместо него другое или выбрать имя из списка, представленного в нижней части окна. Разумно согласиться с предложенным Мастером именем.

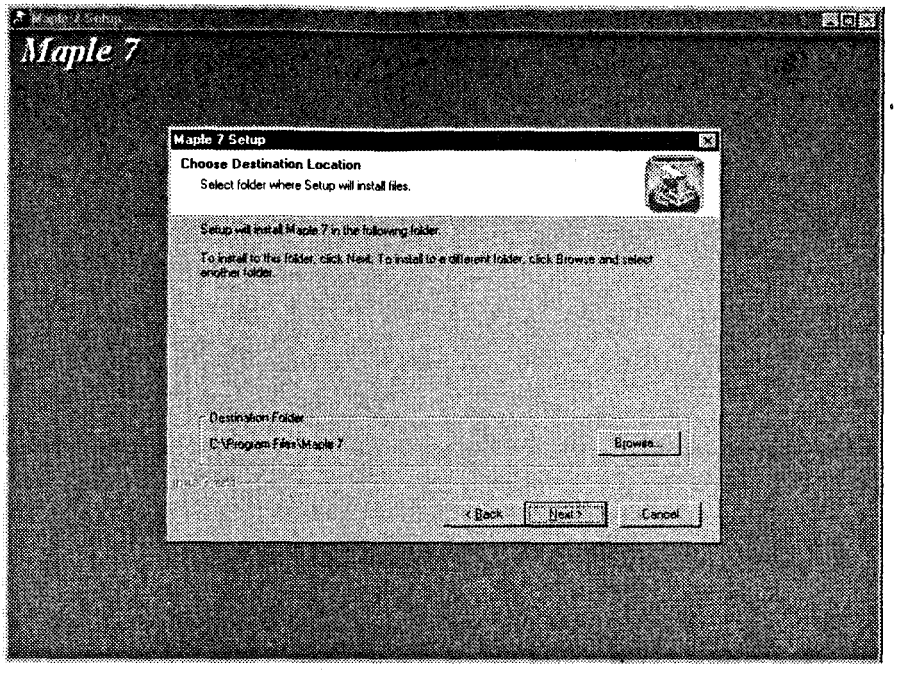


Рис. 1.3. Окно задания папки для размещения файлов системы Maple 7

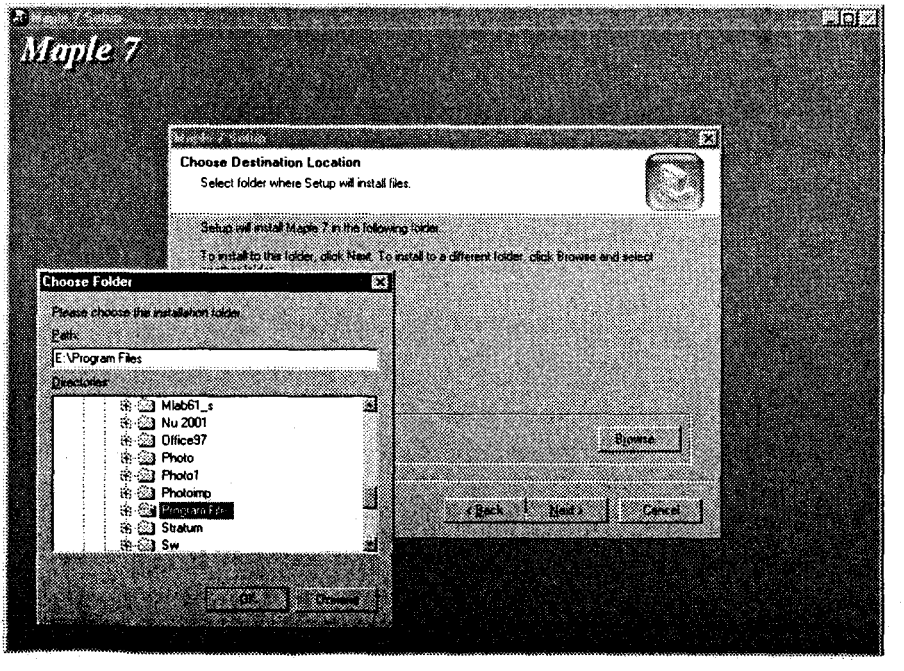


Рис. 1.4. Задание имени папки для размещения файлов системы Maple 7

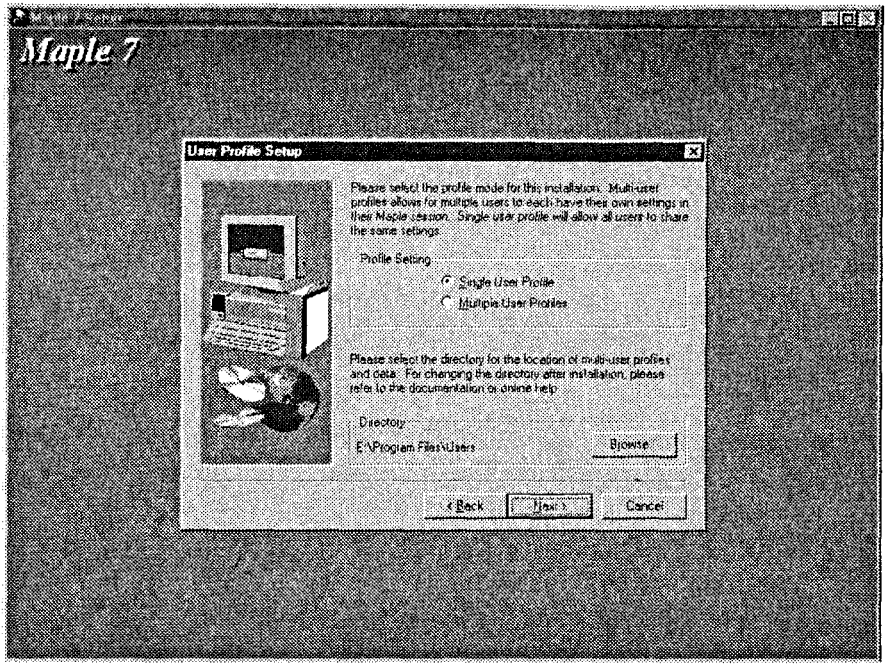


Рис. 1.5. Выбор установки системы Maple 7 для одного пользователя или для многих пользователей

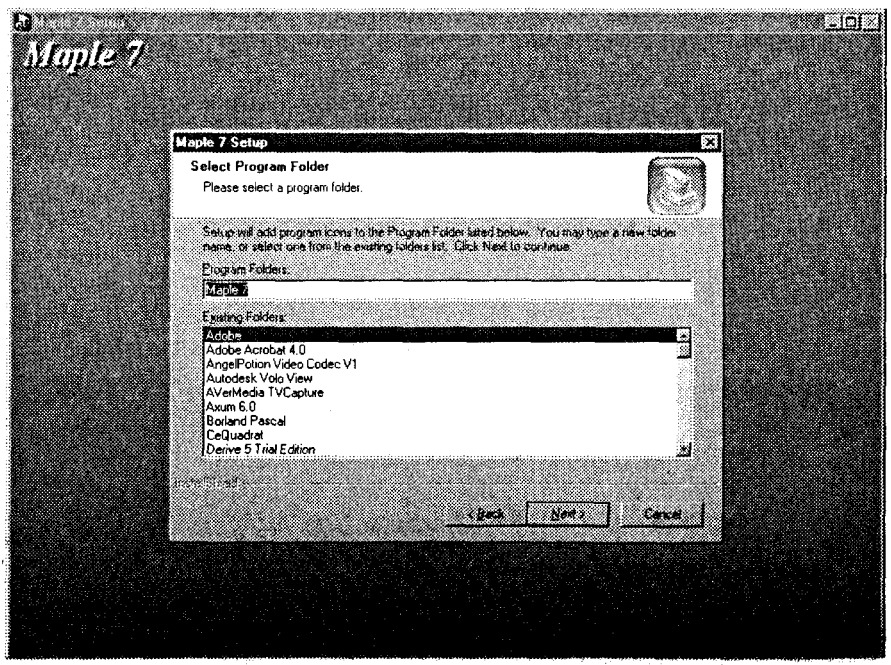


Рис. 1.6. Выбор ярлыка для запуска системы Maple 7

На этом предварительная часть установки завершается, и нажатием кнопки Next в окне рис. 1.6 можно перейти к этапу копирования файлов в намеченную папку (учтите, что часть файлов копируется и в папки с операционной системой Windows, установленной на вашем ПК). В процессе копирования файлов окно Мастера установки сменяется информационными окнами, которые сообщают о возможностях Maple 7. Одно из таких окон представлено на рис. 1.7. Кроме того, на экране будет отображаться окно с индикатором процесса копирования файлов, который позволяет судить о том, какая часть работы уже проделана, а также какие файлы копируются в данный момент.

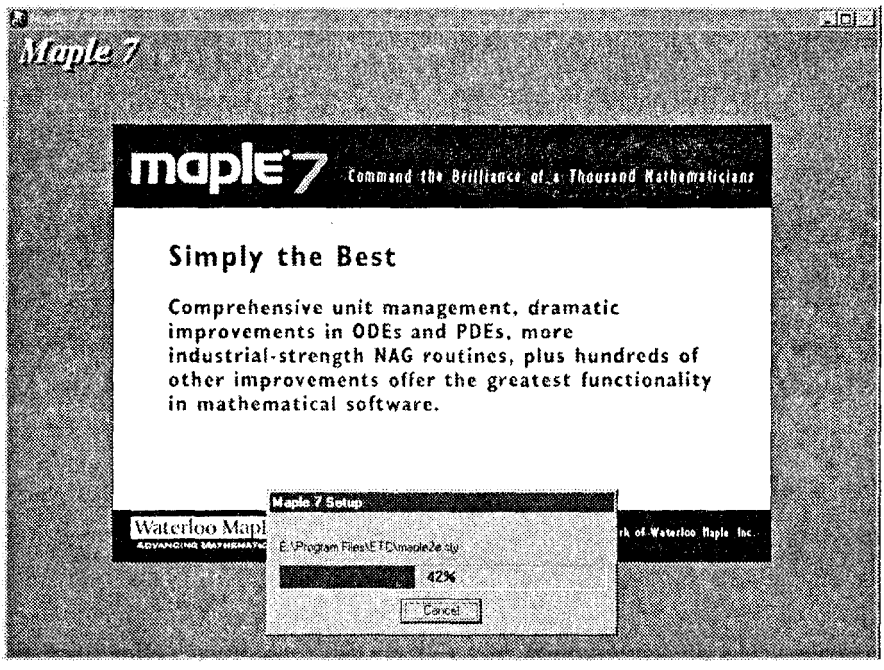


Рис. 1.7. Информационное окно и индикатор процесса копирования файлов

В процессе копирования файлов могут быть ситуации, когда встречается файл с таким же именем, что и копируемый. В этом случае Мастер установки сообщит вам об этом с помощью соответствующего сообщения. Чаще всего такая ситуация встречается, если Maple 7 устанавливается поверх прежней версии системы Maple. Если пользователь твердо намерен установить новую версию системы, то рекомендуется предварительно удалить предшествующую версию (с помощью соответствующих программ и средств, разумеется). В противном случае придется решать, какой файл будет занимать место на диске — останется старый или его заменит новый (естественно, рекомендуется установить новый файл).

По завершении копирования файлов появляется окно Мастера с запросом о том, надо ли помещать ярлык запуска системы на рабочий стол, изображенное на рис. 1.8.

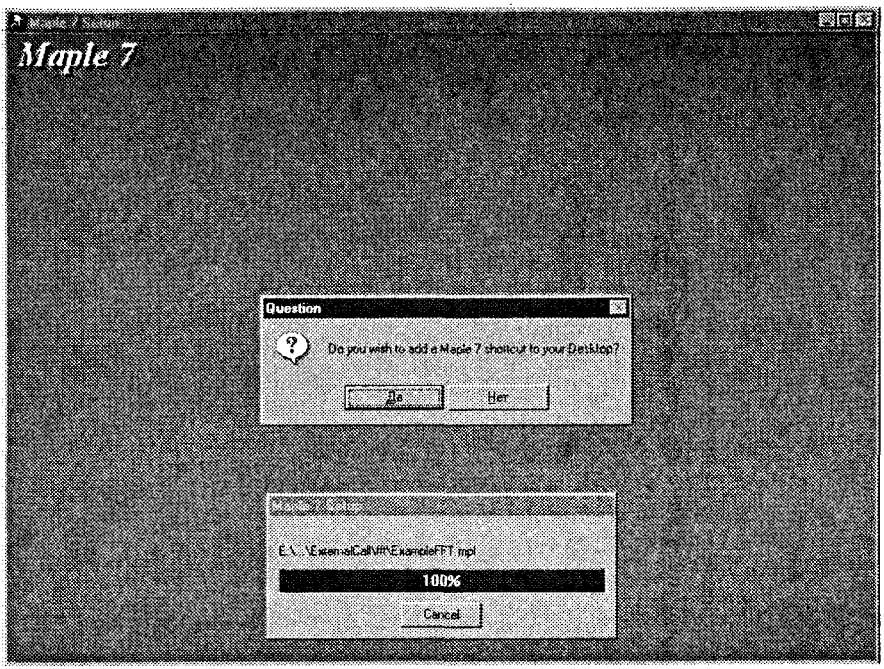


Рис. 1.8. Запрос о размещении ярлыка запуска Maple 7 на рабочий стол

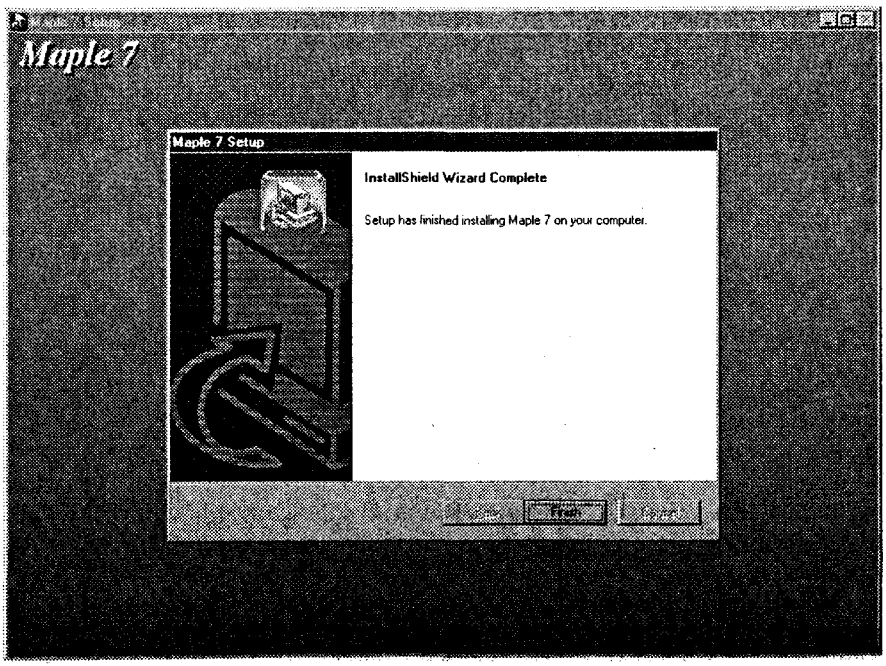


Рис. 1.9. Окно Мастера установки с сообщением о ее завершении

Вы можете отказаться от такого размещения и запускать Maple 7 из меню Пуск или из Проводника. Но, если вы намерены работать с Maple 7 часто, то рекомендуется поместить ярлык на рабочий стол Windows.

Установка завершается выводом окна Мастера установки, в котором сообщается о ее успешном завершении. Данное окно представлено на рис. 1.9.

В этом окне достаточно нажать кнопку Finish для завершения установки. Maple 7 будет установлена на вашем компьютере и готова к работе.

Проблемы, связанные с установкой и возможными конфликтами, описаны в файле readme.txt, который рекомендуется просмотреть перед инсталляцией системы. К сожалению, он подготовлен на английском языке, как и вся встроенная в Maple 7 документация.

Запуск системы

Запуск Maple 7 производится, как обычно, из меню Пуск (рис. 1.10). Найдя строчку Maple 7, необходимо открыть подменю и щелкнуть на команде Maple 7 (рис. 1.10).

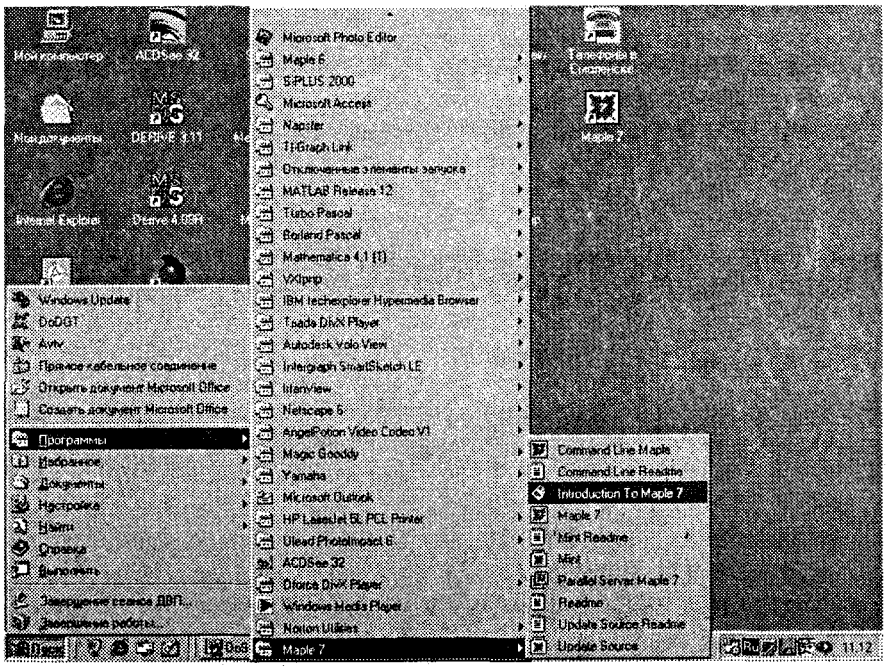


Рис. 1.10. Запуск Maple 7 из меню Пуск

Возможен также запуск с помощью ярлыка, помещенного на рабочий стол. В любом случае, вначале на некоторое время появляется красочное окно-заставка, показанное в центре рис. 1.1. После этого появляется рабочее окно системы, изображенное на рис. 1.11.

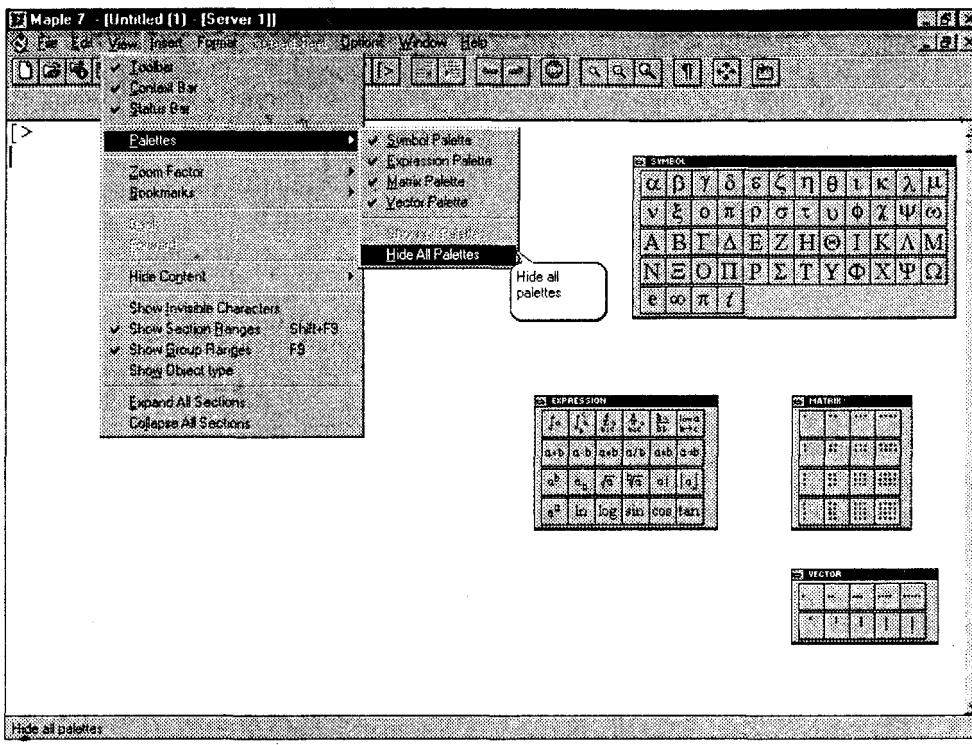


Рис. 1.11. Окно системы Maple 7

Интерфейс системы Maple 7

Обзор интерфейса Maple 7

Долгое время системы Maple имели довольно архаичный (хотя и неплохой) пользовательский интерфейс, ориентированный на операционную систему MS-DOS. Но затем версия Maple V R3, далее Maple V R4, Maple V R5 и, наконец, Maple 6 и Maple 7 приобрели вполне современный графический пользовательский интерфейс, характерный для приложений операционных систем Windows. Это намного повысило удобство работы с Maple и обеспечило интеграцию со многими другими программными продуктами.

Как у всех приложений под Windows интерфейс Maple 7 имеет ряд характерных элементов, видимых на рис. 1.11 и перечисленных ниже:

- строка заголовка (сверху);
- строка главного меню;
- главная панель инструментов;

- контекстная панель инструментов, вид которой зависит от режима работы с Maple 7;
- окно ввода и редактирования документов;
- строка состояния (в самом низу окна).

Пользовательский интерфейс Maple 7 позволяет готовить документы, содержащие одновременно текстовые комментарии, команды входного языка (с возможным преобразованием их в естественную математическую форму), результаты вычислений в виде обычных математических формул и графические данные. Это обеспечивает понятное представление исходных данных и результатов вычислений, а также удобство их повторного использования.

В основе пользовательского интерфейса Maple 7 лежит графический многооконный интерфейс операционной системы Windows. Управление системой Maple 7 возможно с помощью главного меню, панелей инструментов и палитр, а также «горячих» клавиш. Поддерживаются также многие возможности мыши, присущие приложениям под Windows.

Важно отметить и прекрасно реализованную справочную систему Maple 7. Преодолев первые трудности общения с системой, пользователь быстро осваивает систему справки, которая позволяет без какой-либо бумажной документации получить исчерпывающую информацию о любом операторе, функции или пакете (разумеется, на английском языке). На каждой странице справки находится по несколько примеров, причем их можно скопировать и перенести в редактор или в окно исполняемых документов системы. Это способствует быстрому обучению пользователя. Подробное описание справочной системы Maple 7 дано в уроке 2.

Пользователь Maple 7 (как и ряда других математических систем) работает с документами, которые являются одновременно описаниями алгоритмов решения задач, программами и результатами их исполнения. Все данные команды и результаты размещаются в соответствующих ячейках. Графические построения выполняются как в ячейках документа, так и в отдельных окнах, и имеют свои меню для оперативного управления параметрами.

Меню системы Maple 7

Наиболее полные возможности управления предоставляет главное меню системы Maple 7. Оно, как обычно, расположено непосредственно под строкой заголовка. Меню предоставляет доступ к основным операциям и параметрам пользовательского интерфейса системы. Ниже дан перечень меню, доступных при наличии открытого документа:

- **File** — работа с файлами и печатью документов;
- **Edit** — команды редактирование документа и операции с буфером обмена;
- **View** — управление видом пользовательского интерфейса;
- **Insert** — операции вставки;
- **Format** — операции задания форматов;

- Spreadsheet — операции задания таблиц;
- Options — задание параметров;
- Window — управление окнами;
- Help — работа со справочной системой.

Главное меню Maple 7 является контекстно-зависимым. Это означает, что его вид может меняться в зависимости от текущего состояния (контекста) системы. Например, если все документы закрыты, то главное меню содержит только три заголовка меню: File, Options и Help. При этом место для окон документов пусто и окрашено в серый цвет. Вид меню также меняется в зависимости от того, какие объекты в документе выделены. Более подробно работу с меню системы Maple 7 мы рассмотрим позже, начиная с урока 3.

Палитры ввода математических символов

Полезно сразу обратить внимание на возможность модификации интерфейса системы Maple 7 с помощью команд меню View. В этом меню (оно показано на рис. 1.11 в открытом состоянии) можно увидеть список палитр Palettes, предназначенных для ввода математических знаков. Установив флажки соответствующих палитр, можно вывести их на экран и переместить в любое место. Все четыре палитры математических символов представлены на рис. 1.11. При этом палитра VECTOR введена в Maple 7 впервые.

Назначение знаков в палитрах очевидно из их названий:

- SYMBOL — ввод отдельных символов (греческих букв и некоторых математических знаков);
- EXPRESSION — ввод шаблонов математических операторов и операций;
- MATRIX — ввод шаблонов матриц разных размеров;
- VECTOR — ввод шаблонов векторов разных размеров и типов (векторы-столбцы или векторы-строки).

Следует отметить, что не всегда введенный на палитре символ буквально повторяет представленный на кнопке. Например, вместо символа α может быть введено слово alpha. Так происходит, если установлен действующий по умолчанию Maple-режим представления символов. Палитра ввода векторов введена в Maple 7 впервые, что сделало ввод векторов более удобным.

Всплывающие подсказки

Еще один важный и полезный элемент интерфейса — всплывающие подсказки. Они появляются, если навести курсор мыши на тот или иной элемент интерфейса. На рис. 1.11 показана одна из всплывающих подсказок. Подсказки имеют вид облачка, которое вытекает из указанного элемента интерфейса. Особенно удобны подсказки для пояснения назначения кнопок палитр и панелей инструментов. В дальнейшем мы будем неоднократно приводить примеры всплывающих подсказок при работе с интерфейсом.

Основы работы с Maple 7 в диалоговом режиме

Начальные навыки работы

После загрузки и запуска системы можно начать диалог с ней, используя ее операторы и функции (с параметрами) для создания и вычисления математических выражений.

ВНИМАНИЕ Во избежании грубых ошибок при исполнении того или иного примера рекомендуется перед этим исполнить команду restart, которая снимает определения со всех использованных ранее переменных и позволяет начать вычисления «с чистого листа».

На рис. 1.12 представлен реальный диалог с системой (в виде копии экрана) при решении простейших арифметических задач и построении графика функции $\sin(x)/x$.

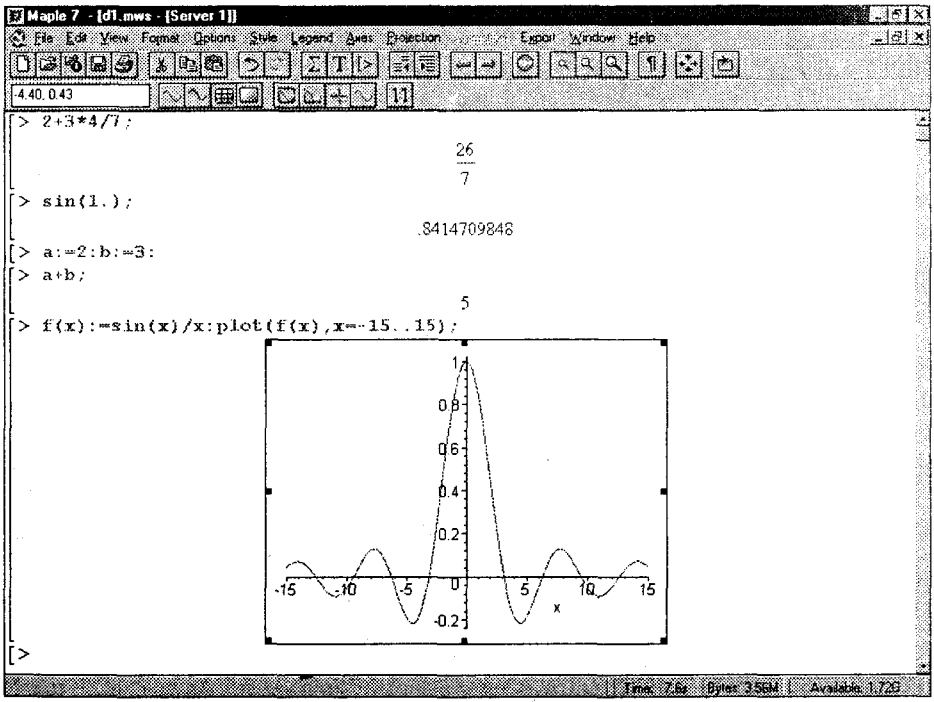


Рис. 1.12. Начало диалога с системой Maple 7

Уже из этого простого примера видны особенности диалога с Maple и синтаксиса ее входного языка, то есть языка, на котором системе задаются вопросы. Диалог

идет в стиле: «задал вопрос, получил ответ». Вопросы и ответы занимают отдельные блоки, выделяемые в левой части квадратными скобками. Длина квадратных скобок зависит от размера выражений — исходных (вопроса) и результатов вычислений (ответов на вопросы). Знак > является знаком приглашения к заданию вопроса. Мигающая вертикальная черта | — маркер ввода (курсор).

Ввод выражений (вопросов) задается по правилам, давно принятым для строчных редакторов. Они хорошо известны, и мы не будем на них останавливаться подробно. Отметим лишь, что клавиша Ins позволяет задавать два основных режима ввода — замены и вставки. В режиме замены вводимый символ заменяет существующий символ, который отмечен маркером ввода. А в режиме вставки новый символ вставляется в текст, не уничтожая имеющиеся символы.

Перемещение маркера ввода осуществляется клавишами перемещения курсора ← и →. Клавиша Backspace стирает символ слева от маркера ввода, а клавиша Del — справа от маркера ввода. Для ввода любого символа надо нажать соответствующую клавишу. Клавиша Shift включает верхний регистр для ввода заглавных (прописных) букв, а клавиша Caps Lock переключает верхний и нижний регистры клавиш с буквами (они меняются местами).

Знак фиксации конца выражения ; (точка с запятой) указывает, что результат его вычисления должен быть выведен на экран, а знак : (двоеточие) отменяет вывод и может использоваться как знак разделителя при записи нескольких выражений в одной строке. Клавиши перемещения курсора позволяют передвигаться по ранее введенным строкам на экране.

Понятие о функциях и операторах

Важным понятием системы Maple 7 (да и математики вообще) является понятие функции. Функция возвращает результат некоторого преобразования исходных данных — параметров функции. Maple 7 имеет множество встроенных функций, включенных в его ядро и в пакеты.

Функция в выражениях задается вводом ее имени и списка параметров функции (одного или нескольких), заключенного в круглые скобки, например sqrt(2) задает функцию вычисления квадратного корня с параметром 2 (численной константой). Основным признаком функции является возврат значения в ответ на обращение к ней по имени (идентификатору) с указанием списка параметров функции. Например:

```
> 2*sin(1.);
1.682941970
> 2*sin(1);
2 sin(1)
```

Обратите внимание на особую роль десятичной точки — здесь она служит указанием к выполнению вычисления значения sin(*1.0*) (или, что то же самое, sin(*1.*)). А вот синус целочисленного аргумента 1 не вычисляется — считается, что вычисленное значение менее ценно, чем точное значение sin(*1*).[1]

[1] В данном случае надо понимать, что Maple предпочитает иметь с дело точными значениями функций. Логично, что точное значение синуса от 1 записывается в виде sin(*1*). — *Прим. ред.*

Ради единства терминологии мы будем пользоваться расширительным понятием функции, относя к нему и те объекты, которые в некоторых языках программирования именуют процедурами или командами. Например, слова `plot` и `plot3d` мы также будем называть функциями, которые возвращают графики аргументов. Под командами же мы будем подразумевать прежде всего команды, содержащиеся в меню.

Помимо функций в математических системах для записи математических выражений используются специальные знаки — операторы. К примеру, вычисление квадратного корня часто записывается с помощью его специального знака — √. Достаточно хорошо известны операторы сложения +, вычитания –, умножения *, деления / и некоторые другие. Операторы обычно используются с операндами в виде констант или переменных, например в записи `2*(3+4)` числа 2, 3 и 4 — это операнды, а знаки * и + — операторы. Скобки используются для изменения порядка выполнения операций. Так, без них `2*3+4=10`, тогда как `2*(3+4)=14`, поскольку вначале вычисляется выражение в скобках.

Пожалуй, самым распространенным оператором является оператор присваивания :=. Он используется для задания переменным конкретных значений, например:

```
> x:=y;
x := y
> y:=z;
y := z
> z:=2;
z := 2
> x;
2
> y;
2
```

Этот простой пример наглядно иллюстрирует эволюцию переменных и особую роль оператора присваивания в системе Maple. В частности, в этом примере переменные `x`, `y` и `z` взаимосвязаны с помощью операций присваивания. Поэтому задание значения 2 переменной `z` приводит к тому, что и переменные `y` и `x` принимают то же значение.

Другой распространенный оператор — оператор равенства = — используется для задания равенств и логических условий (например, `a=b`), указания областей изменения переменных (например, `i=1..5` означает формирование диапазона изменения `i` от 1 до 5) и определения значений параметров в функциях и командах (например, `color=black` для задания черного цвета у линий графиков).

Операторы сами по себе результат не возвращают. Но они, наряду с функциями и своими параметрами (операндами), позволяют конструировать математические выражения, которые при их вычислении также возвращают результат. В силу этого математические выражения, содержащие операторы и операнды, могут быть параметрами функций. Выражения в Maple бывают очень простыми (например, имена переменных `x` и `y` или константы 1 и 2), а могут содержать многие тысячи знаков.

С позиции канонов символьной математики квадратный корень из двух уже является основным результатом вычислений. Поэтому такая функция обычно не вычисляется в численном виде, а выводится в естественном виде с применением знака квадратного корня √. Для вычисления в привычном виде (в виде десятичного числа с мантиссой и порядком) надо воспользоваться функцией

evalf(sqrt(2)) — эта функция обеспечивает вычисление символьного выражения, заданного ее параметром (числом 2).

Обработка и индикация ошибок

При работе с системой Maple 7 надо строго придерживаться правил корректного ввода выражений и иных объектов Maple-языка, называемых синтаксисом языка. Однако, как гласит русская пословица, не ошибается только тот, кто ничего не делает. Даже у опытного пользователя возможны ошибки в ходе ввода выражений и задания алгоритмов вычислений.

Алгоритмические, но синтаксически корректные ошибки часто могут не распознаваться системой. Например, если в выражении a*sin(x) вы вместо аргумента x взяли аргумент b, то есть записали a*sin(b), то такую ошибку Maple 7 распознать не может, ибо синтаксически как a*sin(x), так и a*sin(b) абсолютно корректны. Если вы перепутаете синус с косинусом и запишете a*cos(x), то такая ошибка также не будет распознана.

ПРИМЕЧАНИЕ Ошибки в записи выражений, описывающих те или иные алгоритмы вычислений, не нарушающие синтаксическую корректность, системой Maple 7 не распознаются. Контроль за такими ошибками целиком лежит на пользователе.

Приведем еще один характерный пример ошибки, которую Maple 7 не может распознать. Вводя выражение X/Y*Z, мы можем предположить, что это означает X/(Y*Z). Однако в Maple 7 приоритет операций деления и умножения одинаков. Поэтому Maple 7 вначале делит X на Y, а затем полученный результат умножает на Z:

```
> X/Y*Z:
> %;
```

$$\frac{XZ}{Y}$$

Ошибки такого рода называют семантическими. Если бы мы не проверили вывод с помощью оператора вычисления предыдущего выражения (%), то эта ошибка осталась бы нераспознанной. Выявление и устранение семантических ошибок выполняется на этапе отладки вычислений и программ.

ПРИМЕЧАНИЕ Используйте блокировку вычислений с помощью знака двоеточия только в том случае, когда вы абсолютно уверены в правильности записи выражения. Иначе вы можете не заметить вкравшейся в вычисления серьезной ошибки.

В нашем примере мы получили бы верный результат, заключив выражение Y*X в круглые скобки. Как обычно, они предназначены для задания явного приоритета выполнения операций — в нашем случае вначале будет вычислено выражение в скобках (Y*Z), а затем уже X будет поделено на это выражение:

```
> X/(Y*Z);
```

$$\frac{X}{YZ}$$

Но вернемся к ситуации с синтаксическими ошибками, которые система Maple 7 распознает с помощью встроенного в нее синтаксического анализатора. Например, если вы задали неправильное имя функции, то это будет опознано синтаксическим анализатором и вычисления не будут выполняться. Maple 7 просто повторит выражение в строке вывода:

```
> son(1.0);
 son(1.0)
```

В этом примере вместо имени функции sin введено ошибочное имя son. Maple воспринимает его как некую введенную пользователем функциональную зависимость и потому просто повторяет запись и не выводит сообщение об ошибке. А вот другая ситуация — имя функции sin введено верно, но вместо десятичной точки при задании вещественного числа 1.0 использована запятая:

```
> sin(1,0);
Error, (in sin) expecting 1 argument, got 2
```

В данном случае Maple 7 «знает», что работа идет с его встроенной функцией синуса, которая должна иметь единственный аргумент. Задание (1,0) означает, что растяпа-пользователь ввел вместо вещественного числа два целочисленных числа, разделенных запятой. Этого синтаксический анализатор Maple 7 стерпеть уже не смог, и он отреагировал выдачей сообщения об ошибке (на экране дисплея оно имеет малиновый цвет). Исправьте ошибку, и синус единицы будет благополучно вычислен:

```
> sin(1.0);
 .8414709848
```

А вот еще одна типичная ситуация — в последовательности выражений опущен знак-разделитель (двоеточие или точка с запятой):

```
> X:=2: Y:=3| Z:=4:
Error, missing operator or `;`
```

Тут Maple 7 не только реагирует на ошибку, но и пытается подсказать, что именно пропущено. Более того, маркер ввода в виде мигающей вертикальной черточки будет помещен на место ошибки и вы сможете тут же устранить ошибку. Правда, подсказки не всегда точны — в нашем случае явно пропущен разделитель в виде двоеточия, а Maple 7 сообщает о пропуске точки с запятой. Впрочем, откуда системе знать, хотим мы вывести результат операции Y:=4 сразу (для этого нужен разделитель в виде точки с запятой) или откладываем на потом (с помощью символа двоеточия).

Вот еще один пример характерной ошибки — три знака * подряд:

```
> 2**|*3*sin(1.);
Error, `*` unexpected
```

Здесь Maple 7 подсказывает, что один оператор * надо убрать — два знака * подряд означают вполне законный вариант оператора — возведение в степень. При этом маркер ввода вновь указывает место ошибки. Проанализируйте следующие простые примеры:

```
> 2**3*sin(1.);
 6.731767878
```

```
> 2^3*sin(1.0);
 6.731767878
> 2^(3*sin(1.0));
 5.753392735
```

В этом примере Maple 7 вначале вычисляет функцию синуса, затем производит возведение в степень и лишь потом операцию умножения. Впрочем, такой приоритет операций принят практически во всех системах компьютерной математики и в языках программирования.

Некоторые пользователи версии системы Maple V R5 наблюдали, что русская буква «я» в конце программного комментария (он вводится после символа #) вела к зависанию программы, что требовало ее перезагрузки. В Maple 7 этот недостаток устранен:

```
> #Буква "я" в конце комментария уже не вызывает зависания
>
```

Позже, при описании программирования в Maple 7, мы опишем более развитые средства контроля над допускаемыми пользователем ошибками. Пока же ограничимся приведенными выше сведениями, полезными уже в начале диалога с системой.

Управление с помощью мыши

Для управления состоянием ячеек можно использовать контекстное меню, появляющееся при нажатии правой кнопки мыши. Если установить указатель мыши на входной ячейке, то это меню будет содержать три команды:

- Standard Math — включает и выключает показ входных выражений в естественной математической форме;
- Maple Input — управляет видом ячейки ввода (математический/текстовый);
- Execute — включает выполнение ячейки.

Также в зависимости от состояния буфера обмена и наличия выделения в контекстном меню могут присутствовать команды Cut, Copy и Paste.

Левая кнопка мыши может использоваться для передачи фокуса управления или переноса маркера ввода, а также выделения частей документа.

Примеры задания функции пользователя и построения ее графика

На рис. 1.12 показано, как задается функция пользователя `f(x)` с одним параметром `x`. Нетрудно заметить, что параметр указывается в скобках после имени функции а для записи выражения функции используется знак присваивания `:=` (двоеточие со знаком равенства). Для построения графика функции `f(x)` используется функция `plot` в форме:

```
plot(f(x), x = -15..15);
```

Нетрудно заметить, что при наличии нескольких параметров функции (в нашем случае их два) они разделяются запятыми. Выражение `x=-15..15` задает, во-первых, указание, относительно какой переменной строится график, а во-вторых, говорит, в какой области значений меняются значения этой переменной — в нашем случае от −15 до +15. Шаг изменения переменной выбирается автоматически в зависимости от размеров и вида графика.

Пример построения трехмерного графика поверхности

Столь же просто, как и график обычной функции в декартовой системе координат, можно построить график трехмерной поверхности. Это показано на примере рис. 1.13. В данном случае задана функция двух переменных `z(x,y):=sin(x*y)` и ее график строится с использованием графической функции `plot3d`. Правила задания пределов изменения переменных `x` и `y` соответствуют описанным выше.

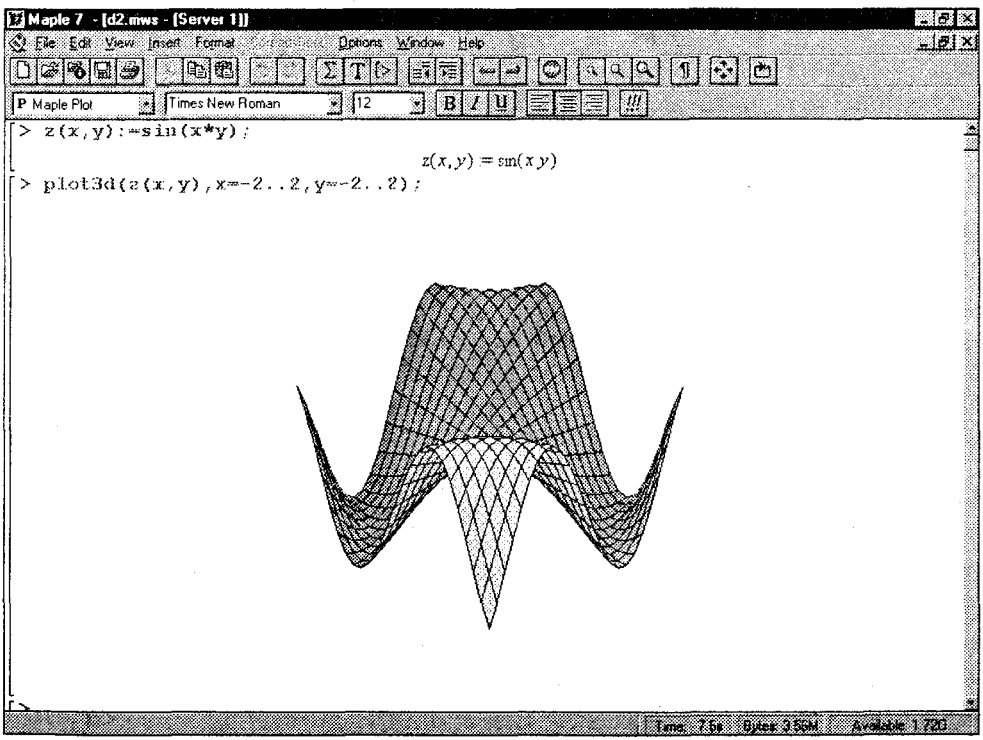


Рис. 1.13. Построение графика трехмерной поверхности

Возможно, многих читателей вполне удовлетворят уже описанные возможности, но сила системы Maple 7 прежде всего в возможности выполнения аналитических (символьных) вычислений. Поэтому мы перейдем к обсуждению некоторых из них.

Управление формой представления документа

Форматы математических выражений

Приведенные выше примеры реализуют обычную форму представления документа. В нем имеются текстовые комментарии (для их ввода надо нажать клавишу F5), сформулированные на Maple-языке задания на вычисления, результаты вычислений в виде обычных математических формул и, там где это указано, графики.

Эта выстраданная форма представления документов является компромиссом между наглядностью и простотой ввода исходных данных. Может показаться, что в этом отношении намного дальше продвинулись системы класса Mathcad — у них исходные данные и описание алгоритмов вычислений давно задаются в виде естественных математических символов и формул. За исключением, правда, функций символьных вычислений, пока не имеющих общепринятых специальных математических символов и вводимых путем указания их имен.

Однако это достоинство кажется явным лишь на первый взгляд. Ввод сложных формул довольно трудоемок и требует специфических навыков, отсутствующих даже у самых опытных пользователей. В Mathcad эту проблему решили созданием панелей (палитр) с полным набором всех математических символов и шаблонов для представления сложных формул, таких как интегралы, суммы и произведения рядов, производные и т. д. Однако, хотя при этом их ввод и становится более простым, легким его не назовешь, а монотонность операций нервирует многих пользователей.

В Maple 7 ввод исходных данных производится привычными для языков программирования средствами — с помощью функций и операторов, задаваемых в командной строке. Зато результаты вычислений получаются по умолчанию в виде обычных формул (хотя есть возможность их представления в другом виде, например принятом в редакторе LaTeX или языках программирования Fortran и C).

Тем не менее вид документа с таким специфическим заданием формул может озадачить математика и любого пользователя, не слишком знакомого с основами программирования. В целом он отрицательно сказывается на восприятии документов.

Представление входных выражений в математической форме

Для устранения подобного недостатка (а скорее, противоречия) Maple 7 предлагает ряд средств. Во-первых, это текстовые комментарии, в которые можно вводить формулы. Во-вторых, это инертные функции, которые не вычисляются, но дают вывод на экран в естественной математической форме (рис. 1.14). И в-третьих, это возможность быстрого преобразования строковых выражений ввода в естественные математические формулы.

Об инертных функциях мы поговорим позже более подробно. Отметим лишь, что имена таких функций начинаются с большой буквы и функции выводят

математическое выражение в естественной математической нотации. С помощью ряда функций, например evalf, можно вычислить математическое выражение, полученное инертной функцией. На рис. 1.14 внизу дан пример такого вычисления для предела функции $\sin(x)/x$.

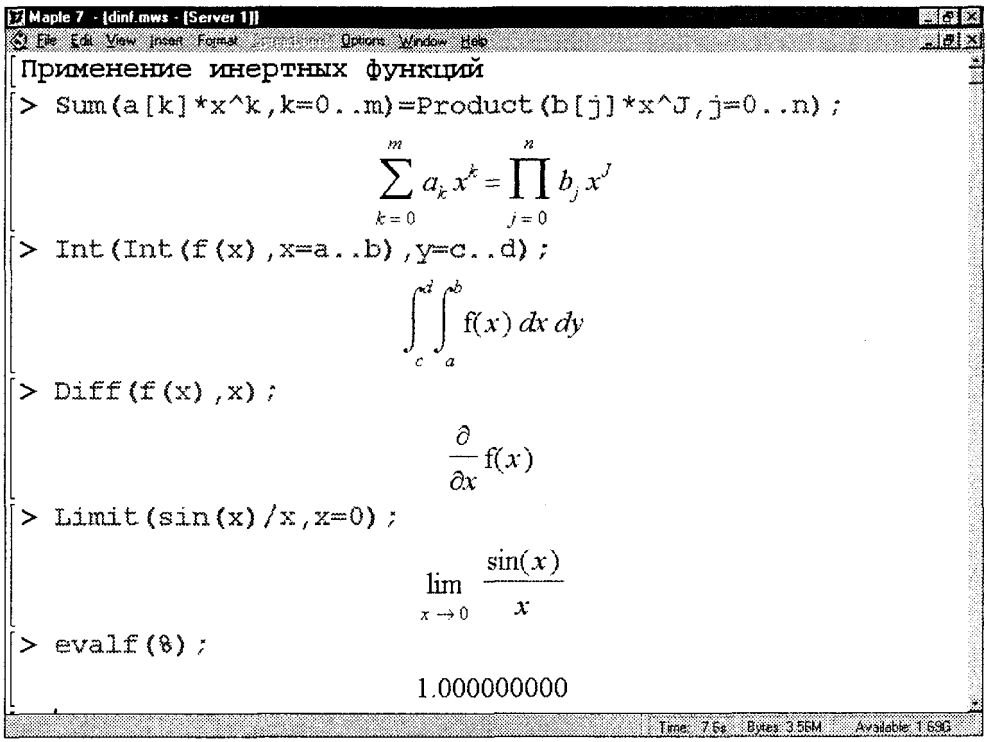


Рис. 1.14. Примеры применения инертных функций

Теперь остановимся на преобразовании исполняемых выражений ввода на Maple-языке в обычные математические формулы. Для этого достаточно, выделив входное выражение, нажать первую кнопку контекстной панели — соответствующее выражение тут же приобретет вид обычной математической формулы. На рис. 1.15 показаны примеры вычислений интеграла при его задании в строках ввода в виде текстового выражения и в обычной математической нотации.

Таким образом, всегда можно получить формульное представление входных выражений. Более того, другой кнопкой их можно превратить в инертную форму, тогда выражение перестает вычисляться и становится, по существу, обычным комментарием.

Следует, однако, учитывать, что представление входных выражений в виде формул обычно занимает заметно больше места на экране и в документе, чем описание выражения на Maple-языке, поэтому оно используется довольно редко. Кроме того, далеко не всякое входное выражение может быть представлено в виде математической формулы — многие функции ядра и библиотек Maple 7

попросту не имеют общепринятых обозначений в виде специальных математических знаков.

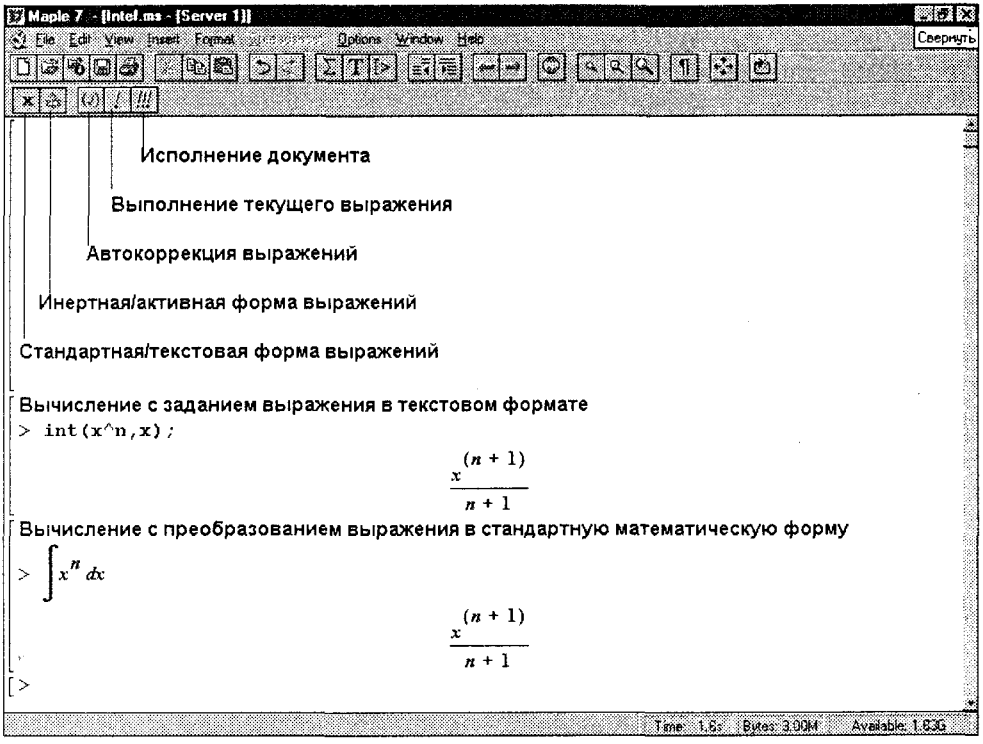


Рис. 1.15. Примеры вычислений интеграла при его задании в текстовой и математической нотации

Символьные вычисления

Простой пример символьных вычислений

Maple 7 открывает обширные возможности выполнения символьных (аналитических) вычислений. Начнем с простого примера — требуется найти сопротивление трех параллельно включенных резисторов *R1*, *R2* и *R3* произвольной величины. Из курса электротехники известно, что можно задать следующее равенство, определяющее суммарное сопротивление *R0*:

```
> eq:=1/R0=1/R1+1/R2+1/R3;
```

$$eq := \frac{1}{R0} = \frac{1}{R1} + \frac{1}{R2} + \frac{1}{R3}$$

Теперь достаточно использовать функцию решения уравнений `solve`, чтобы найти значение *R0* в общей аналитической форме:

```
> R0:=solve(eq,R0);
```

$$R0 := \frac{R1\ R2\ R3}{R2\ R3 + R1\ R3 + R1\ R2}$$

Нетрудно проверить, что результат может быть получен и в численном виде для конкретных значений *R1*, *R2* и *R3*:

```
> R1:=1:R2:=2:R3:=3:R0;
```

$$\frac{6}{11}$$

```
> evalf(%);
```

$$.5454545455$$

Типовые символьные вычисления

На рис. 1.16 показано несколько примеров выполнения символьных вычислений математического характера: преобразование тригонометрического выражения с помощью функции упрощения `simplify`, вычисление суммы ряда функцией `sum` и вычисление неопределенного интеграла функцией `int`.

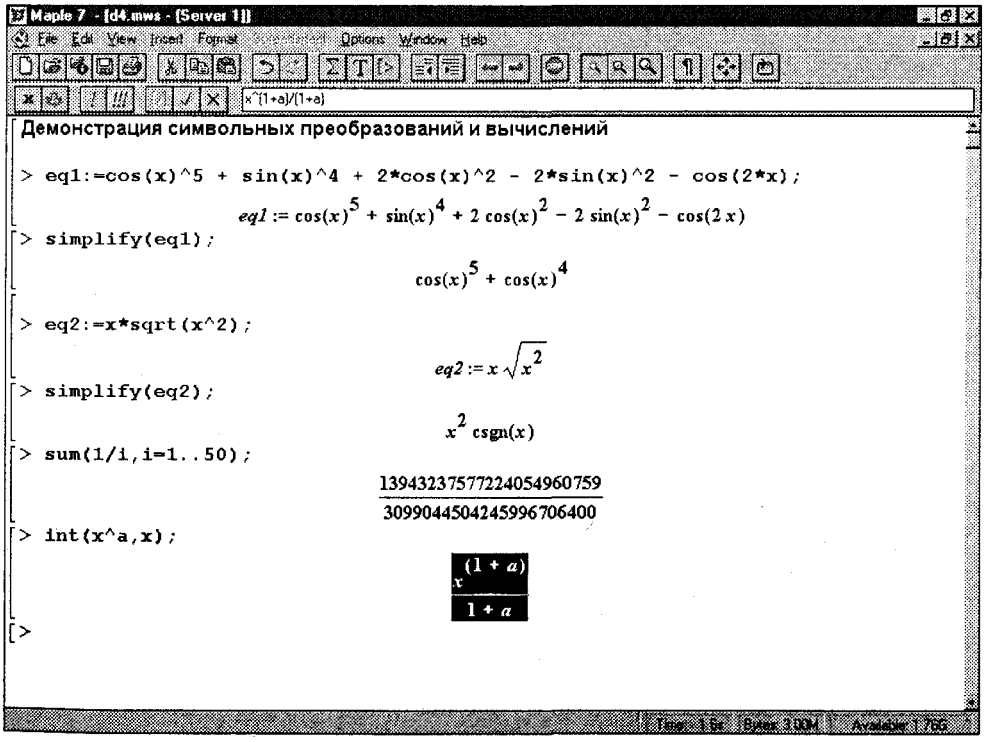


Рис. 1.16. Примеры символьных вычислений

Обратите внимание на результат выполнения последнего примера. Он выделен. Выделение можно осуществить протаскиванием указателя мыши с нажатой левой кнопкой.

Вычисления производных и интегралов в символьном виде, пожалуй, являются наиболее характерными областями применения систем символьной математики. На рис. 1.17 показаны примеры таких вычислений с применением функции `dif` для вычисления производной и `int` для вычисления определенных интегралов.

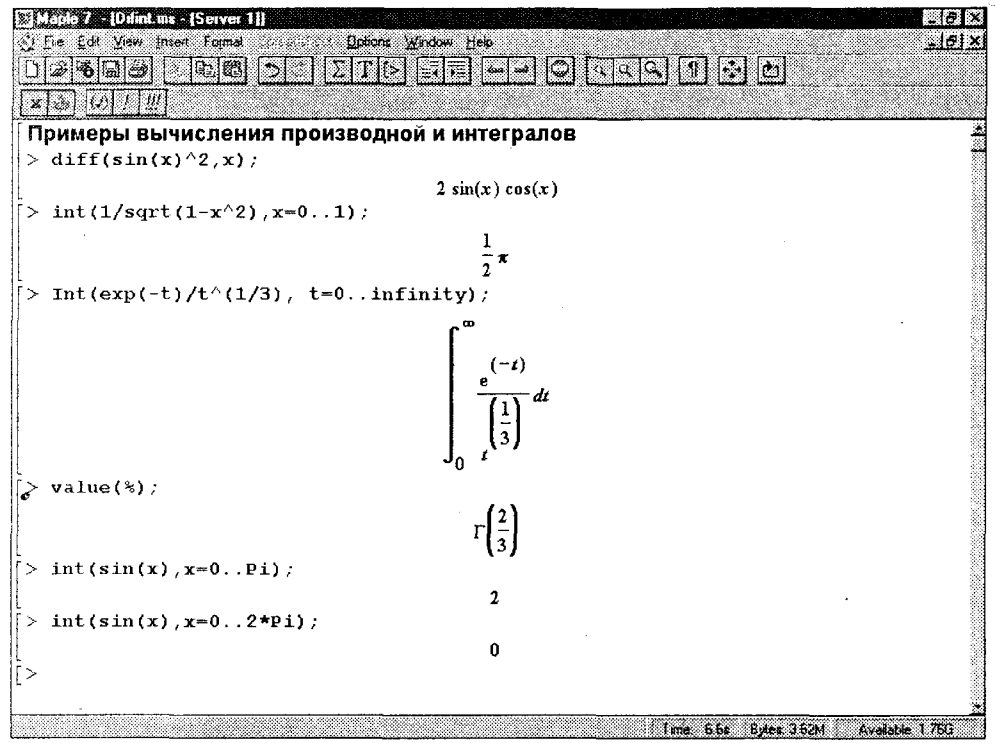


Рис. 1.17. Примеры вычисления производной и интегралов

Обратите внимание на функцию `Int` — инертную форму функции `int`. Как уже отмечалось, инертная форма служит для вывода записи интеграла в естественной математической форме, но с отложенным «на потом» выводом результата вычислений. Как отмечалось, это один из путей наглядного представления входных выражений. Все инертные функции имеют имена, начинающиеся с большой буквы, тогда как обычные функции имеют имена, начинающиеся с маленькой буквы.

На другом рисунке (рис. 1.18) показано вычисление интеграла, который не имеет представления через функции системы Maple 7, но может быть вычислен ею в численном виде.

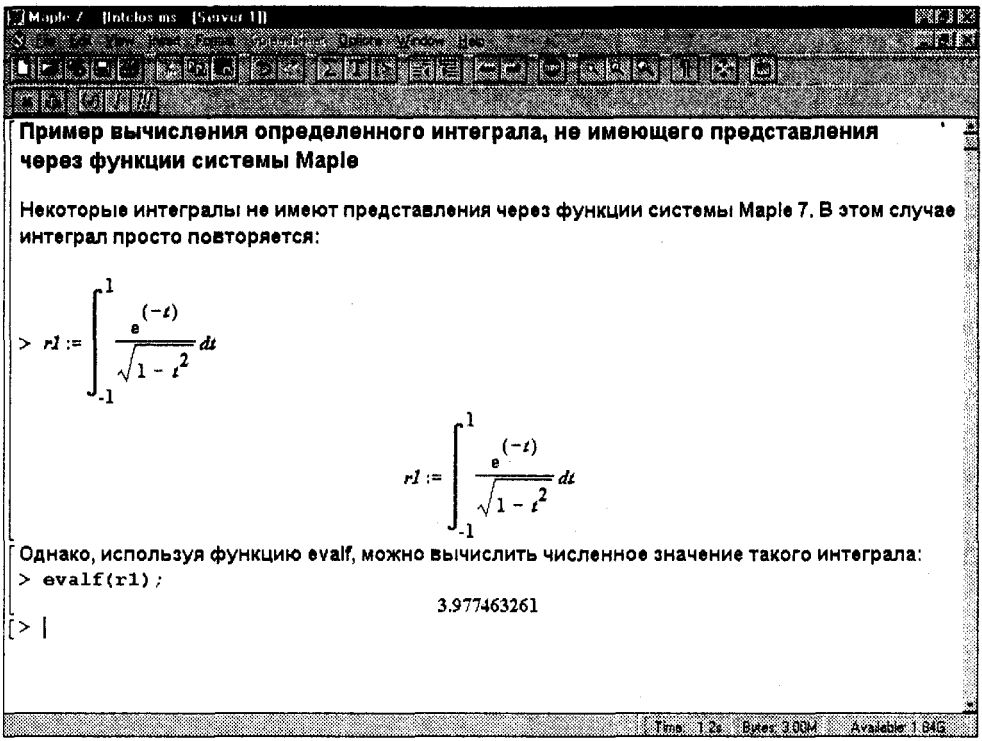


Рис. 1.18. Численное вычисление значения интеграла, не имеющего аналитического представления

Разбухание результатов символьных вычислений

Одной из проблем систем компьютерной алгебры является «разбухание» результатов — как оконечных, так и промежуточных. Связано это с тем, что аналитическое представление порою может оказаться весьма громоздким даже для простых задач — пожалуй, это главная причина прохладного отношения к аналитическим вычислениям со стороны инженеров, особенно практиков. К примеру, численное решение кубического уравнения не вызовет трудностей даже на калькуляторе [1], тогда как системы символьной математики выдают его в виде формул, едва помещающихся на экране. Это и иллюстрирует рис. 1.19, на котором показано решение квадратного уравнения (его знает каждый мало-мальски преуспевающий в учебе школьник) и решение кубического уравнения (оно вызывает бурный восторг или легкий шок — в зависимости от отношения учащегося к математике).

Щепетильность системы в ее стремлении выдать полный и математически предельно точный результат, безусловно, очень важна для математиков. Но для многих прикладных задач, с которыми имеют дело инженеры и техники, она оборачивается неудобствами. Инженеры часто прекрасно знают, какие из членов математических формул можно преспокойно отбросить, тогда как для мате-

матика-теоретика или аналитика такое действо — типичное кощунство. Порою системы компьютерной алгебры выдают настолько «заумный» и огромный результат, что его упрощение может занять куда больше времени, чем получение более простого результата с заранее выполненными упрощениями. Впрочем, каждому свое! И Maple имеет множество функций, обеспечивающих преобразование результатов в ту или иную форму.

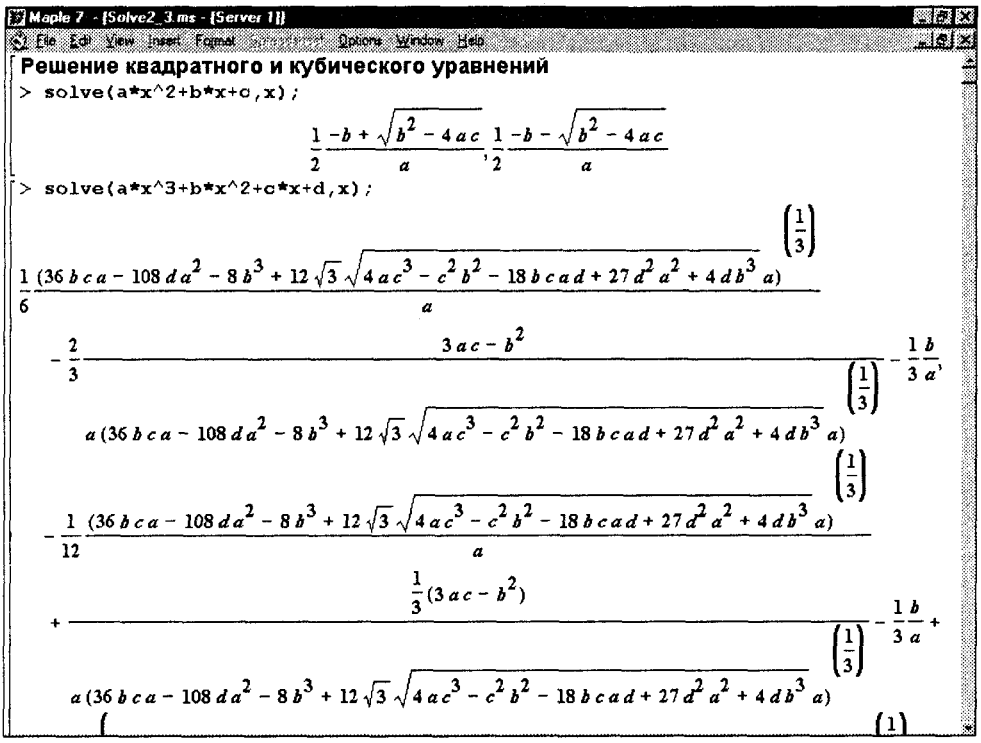


Рис. 1.19. Решение квадратного и кубического уравнений в символьной форме

Пример решения системы линейных уравнений

Приведем еще один характерный пример — решение системы линейных уравнений с помощью функции `solve` (рис. 1.20). Обратите внимание на форму задания уравнений и выдачи результатов и поразительную естественность решения задачи. Значение переменной `z` на рис. 1.20 выделено, где видно, что Maple отображает его поле под панелью инструментов.

Слова `solve`, `diff` и `int` с их аргументами являются именами встроенных в систему функций, возвращающих символьные значения результатов. Нормальному пользователю может стать дурно, если вспомнить, что таких функций с их вариантами система Maple 7 имеет около трех тысяч! Да к тому же многие

функции (та же solve для решения уравнений) подчас могут применяться во многих случаях и имеют массу параметров и директив для уточнения направлений решения и расширения областей применения.

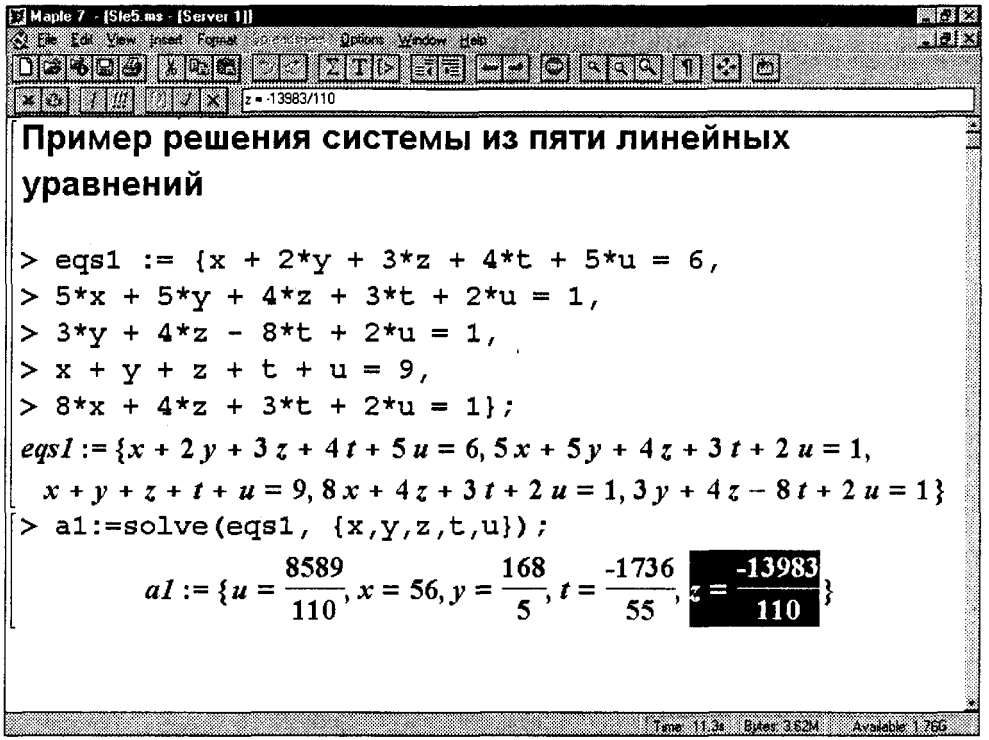


Рис. 1.20. Решение системы из пяти линейных уравнений

В утешение можно отметить три важных обстоятельства:

- мало кто на практике использует из всей этой массы функций более чем несколько десятков;
- названия и формы представления многих функций интуитивно предсказуемы;
- наконец, система имеет превосходную справочную базу данных, с помощью которой при определенном терпении (и непременном желании) можно разобраться с синтаксисом любой функции.

Необходимые функции и правила их преобразования система черпает в библиотеке размером около 40 Мбайт (она содержит файлы maple.hdb, maple.lib, maple.ind и maple.cmd). Это иногда занимает заметное время, особенно при первом использовании определенной группы операторов (например, тригонометрических). При повторном использовании этой группы система заметно убыстряется, так как использует уже загруженные средства.

Повышение эффективности работы с системой

Работа с панелью инструментов

Пока что мы при проведении вычислений пользовались лишь простейшими средствами управления системой — вводом выражений и текстовых надписей с клавиатуры. Теперь пора расширить представления о работе с Maple. Прежде чем начать работать с ее меню, надо отметить, что для многих (особенно начинающих) пользователей оказывается удобнее использовать кнопки, расположенные на панелях инструментов, которые находятся прямо под строкой меню.

На рис. 1.21 показано назначение кнопок панели инструментов (Tool Bar). Эти кнопки дублируют наиболее важные операции главного меню и имеют наглядные и типовые для Windows-приложений обозначения. Назначение кнопок и других деталей интерфейса также показаны на рис. 1.21.

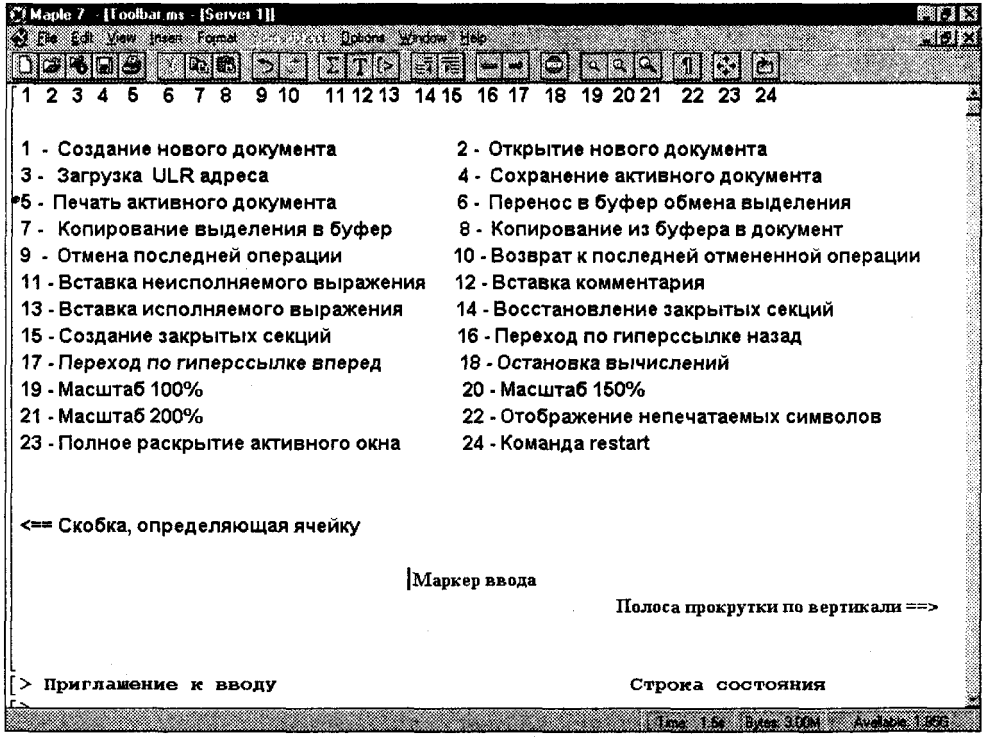


Рис. 1.21. Панель инструментов

При необходимости панели инструментов можно убрать с помощью команд меню View (см. рис. 1.11). Если графика выводится в отдельное окно, там имеется своя панель инструментов, которая будет описана ниже.

Работа с контекстной панелью

Другое полезное средство для облегчения работы по форматированию текстов, заданию параметров входных математических выражений и графиков — контекстная панель инструментов. Как следует из названия, контекстная панель Context Bar является контекстно-зависимой — ее содержание зависит от текущего положения маркера ввода или выделения.

Контекстная панель содержит следующие элементы при вводе текста комментария (рис. 1.22):

- списки для задания стиля, шрифта и размера символов, кнопки для придания полужирного (Bold), наклонного (Italic) и подчеркнутого (Underline) начертания;
- кнопки для выравнивания текста;
- кнопку команды исполнения всего документа.

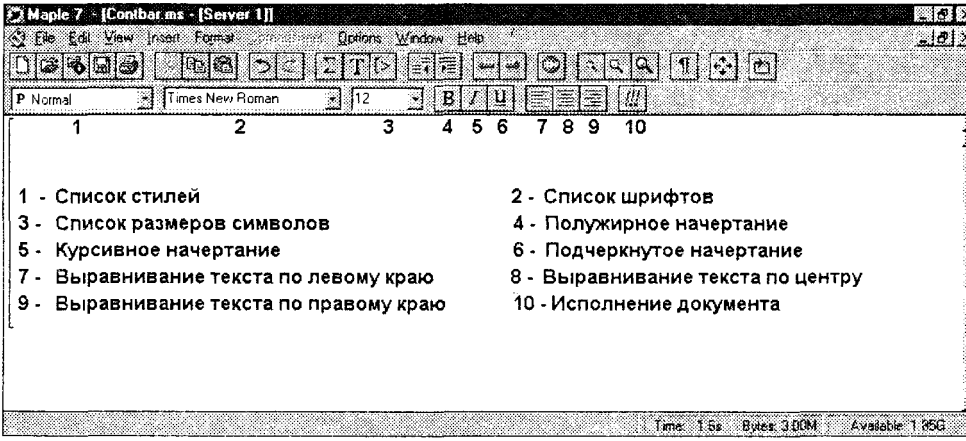


Рис. 1.22. Контекстная панель инструментов для текста комментария

На рис. 1.15 была показана контекстная панель в случае, когда маркер ввода находится в строке ввода. Там же поясняется назначение кнопок панели для данного случая. Особо остановимся на кнопке со значком (√), которая обеспечивает проверку синтаксиса вводимого выражения до того, как оно завершено символами двоеточия или точки с запятой.

Поясним на примере. Допустим, мы ввели ошибочное выражение для интеграла, забыв указать показатель степени в подынтегральном выражении x^n:

```
> int(x^,x)
```

Если нажать кнопку автокоррекции, оно примет вид:

```
> int(x^ %? ,x)
```

Maple явно указывает на необходимость ввода показателя степени — в нашем случае переменной n.

А вот другой пример — мы забыли ввести закрывающую скобку в выражение:

```
> int(x^n ,x
```

Если теперь нажать кнопку автокоррекции, то вставка скобки произойдет автоматически:

```
> int(x^n ,x)
```

Таким образом, данная кнопка может быть полезна для оперативного контроля синтаксиса и исправления грубых ошибок при вводе выражений в формате **Maple Input**. Однако необходимо делать это до их исполнения.

Заметим, что пока формула является входным выражением в математической форме, она может редактироваться — но не сама по себе, а в виде текстового выражения, отображающегося в поле редактирования на контекстной панели. При этом изменение записи выражения в поле редактирования немедленно влечет соответствующее изменение вводимой формулы. Это тоже довольно удобное средство, имеющее свои преимущества перед прямым редактированием формулы — в MathCAD, например, прямое редактирование формул требует определенных навыков и усваивается довольно туго.

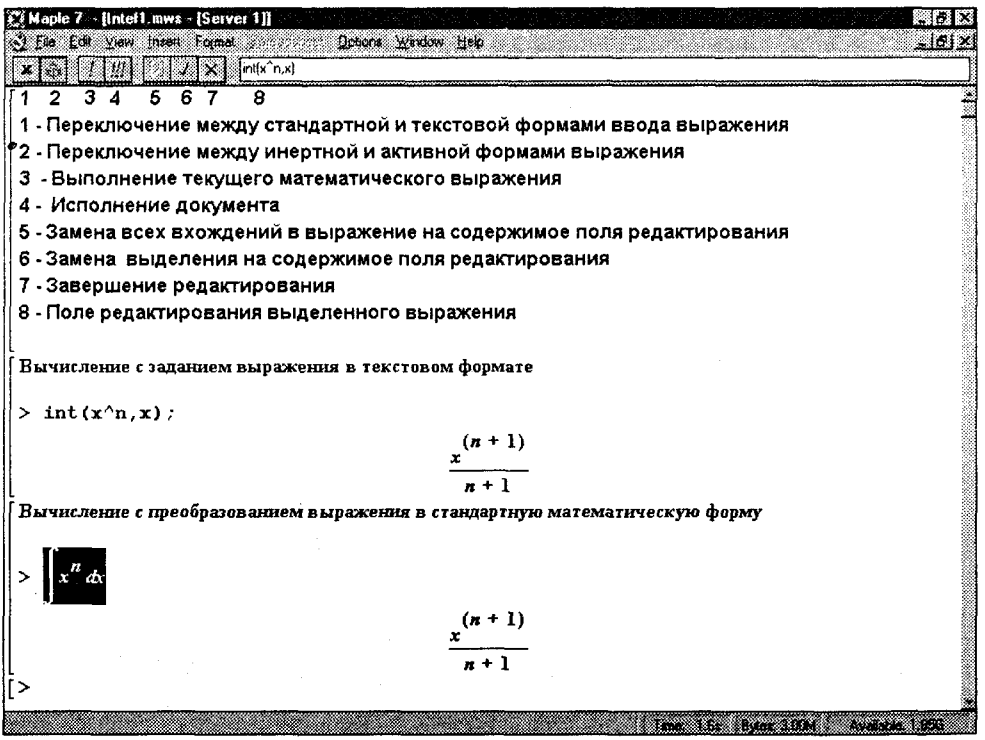


Рис. 1.23. Контекстная панель инструментов для выражений, представленных в математической нотации

Рисунок 1.23 показывает назначение кнопок контекстной панели при установке маркера ввода в строку, содержащую входное выражение в математической нотации. Нетрудно заметить, что в этом случае контекстная панель несколько изменяется — в частности, в ней появляется поле ввода, в котором выведено выделенное выражение на Maple-языке. Такой же вид контекстной панели будет, если выделено выражение или его часть в строке вывода.

Контекстная панель инструментов для двумерных графиков

Двумерные графики строятся с заданием ряда параметров, определяющих общий стиль графика. Эти параметры задают цвет и стиль линий графика, вывод координатных осей и т. д. Все параметры имеют значение по умолчанию — они и определяют вид графика, при формировании которого параметры не указаны.

Однако ряд параметров можно изменять, щелкая на соответствующих кнопках контекстной панели. На рис. 1.24 показано назначение кнопок контекстной панели инструментов для редактирования параметров двумерных графиков. Такая панель появляется, если двумерный график выделен или на нем находится маркер ввода.

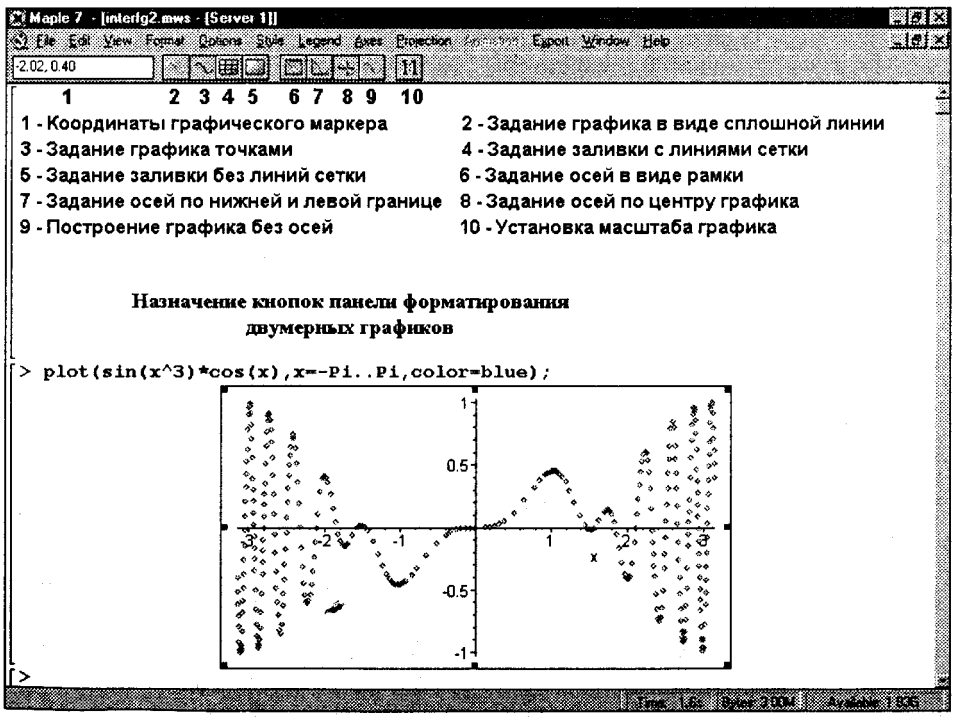


Рис. 1.24. Контекстная панель инструментов для двумерного графика

Действие большинства кнопок этой формы контекстной панели достаточно очевидно, и вы легко сможете опробовать эти кнопки в работе. Так, график на рис. 1.18 построен точками при нажатии кнопки, задающей стиль Point style. Кроме того, в функции `plot` построения графика явно использована опция `color=blue`, которая задает синий цвет точек графика.

Полезно отметить, что в левой части контекстной панели есть поле с координатами текущей выделенной точки графика. Чтобы выделить точку, надо подвести к ней указатель мыши и щелкнуть левой ее кнопкой.

При построении контурных графиков и графиков плотности имеется возможность заливки их областей между линиями уровня с выводом линии сетки, на которой рассчитываются линии уровня, и без вывода линий сетки.

Контекстная панель инструментов для трехмерных графиков

Свой вид контекстной панели имеют и трехмерные графики. Назначение ее элементов представлено на рис. 1.25.

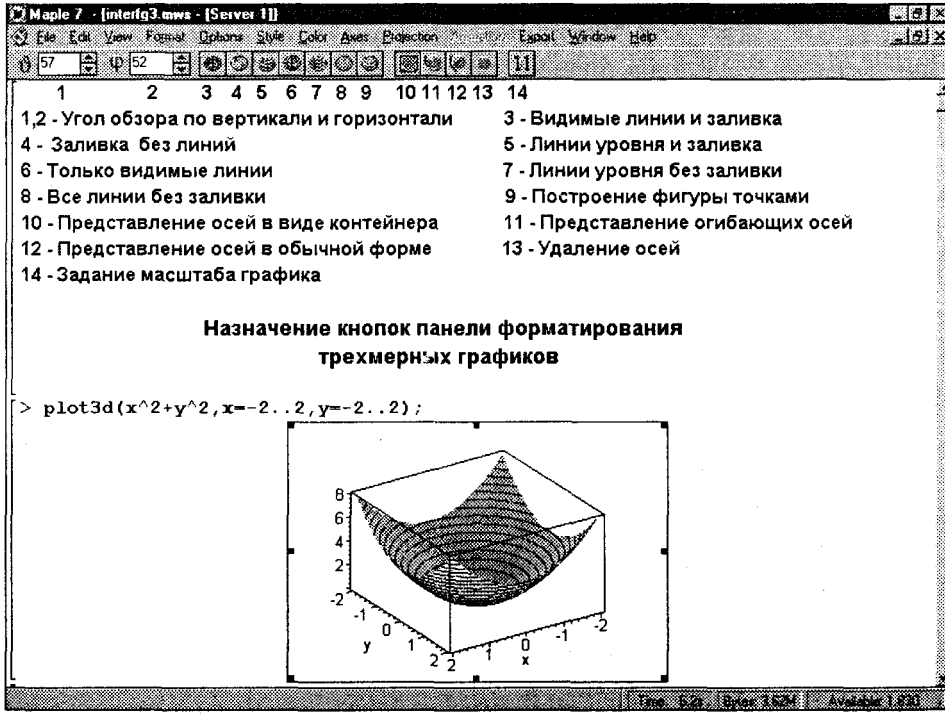


Рис. 1.25. Контекстная панель инструментов для трехмерных графиков

С помощью контекстной панели трехмерных графиков можно задать 7 стилей построения трехмерных графиков и 4 стиля вывода координатных осей. Воз-

можны следующие стили трехмерных графиков (группа из семи кнопок в середине панели): функциональная окраска с видимыми линиями каркаса, функциональная окраска без линий каркаса, функциональная окраска с контурными линиями, цветной каркас с видимыми линиями, цветные контурные линии, цветной каркас со всеми (в том числе невидимыми) линиями и поверхность, построенная точками.

Два расположенных слева счетчика позволяют задавать нужный угол обзора, причем Maple сразу же отражает заданный поворот построенной фигуры. Ее также можно вращать мышью, поместив указатель в область графика и держа нажатой левую кнопку. При этом счетчики будут отображать изменяющиеся при перемещении мыши углы обзора. Это очень удобное средство для наблюдения за деталями трехмерных поверхностей и фигур, которые строит функция `plot3d`.

Строка состояния

При работе с Maple полезно следить за информацией в строке состояния системы, расположенной внизу экрана. В ней появляются надписи, поясняющие исполняемые операции. Кроме того, в полях на правой части выводится текущее время, объем используемой оперативной памяти и объем доступной памяти на жестком диске, на котором установлена система.

Хотя панели инструментов и строка состояния удобны для быстрого управления системой, они обладают одним существенным недостатком — занимают часть полезной площади экрана. Однако в меню View имеются команды, позволяющие убрать с экрана любые из этих элементов. Действие этих команд вы можете заметить, если внимательно присмотритесь к предыдущим рисункам.

Горячие клавиши системы

Для открытия меню можно использовать одновременное нажатие клавиши Alt и клавиши, которая подчеркнута в названии меню. Вызов команды при одновременном нажатии нескольких клавиш получило название «горячих» клавиш управления (Hot Keys).

Таблица 1.1. Горячие клавиши для открытия меню

Меню	Горячие клавиши
Edit	Alt+E
File	Alt+F
Format	Alt+R
Help	Alt+H
Insert	Alt+I
Options	Alt+O
View	Alt+V
Spreadsheet	Alt+S
Window	Alt+W

«Горячие» клавиши присвоены и многим другим, наиболее распространенным операциям. Они облегчают и ускоряют (наряду с кнопками панелей инструментов) быстрое управление программой. Ниже в таблицах приведен список наиболее используемых «горячих» клавиш, разбитых на ряд категорий.

Таблица 1.2. Горячие клавиши для задания начертания символов и режимов ввода текста

Действие	Горячие клавиши
Полужирное начертание	Ctrl+B
Курсивное начертание	Ctrl+I
Подчеркнутое начертание	Ctrl+U
Включение/выключение ввода текста	F5
Установка режима ввода (Input Mode)	Ctrl+M
Задание режима ввода текста (Text Input Mode)	Ctrl+T

Таблица 1.3. Горячие клавиши для операций выделения

Действие	Горячие клавиши
Выделение символа слева	Shift+¬
Выделение символа справа	Shift+®
Выделение строки вверх	Shift+-
Выделение строки вниз	Shift+?
Выделение от маркера ввода до начала строки	Shift+Home
Выделение от маркера ввода до конца строки	Shift+End
Выделение от маркера ввода и до начала документа	Shift+PgUp
Выделение от маркера ввода и до конца документа	Shift+PgDn

Таблица 1.4. Горячие клавиши операций удаления, копирования и вставки

Операция	Горячие клавиши
Выделить все	Ctrl+A
Копирование выделения в буфер	Ctrl+C
Перенос выделения из документа в буфер	Ctrl+X
Вставка содержимого буфера в документ	Ctrl+V
Удаление строки ввода (параграфа)	Ctrl+Delete
Закрытие выделенной секции	Ctrl+.
Поиск	Ctrl+F
Вставка параграфа после маркера	Shift+Ctrl+J
Вставка параграфа до маркера	Shift+Ctrl+K
Восстановление закрытых секций	Ctrl+,
Вставка исполняемой группы после маркера	Ctrl+J
Вставка исполняемой группы до маркера	Ctrl+K
Вставка конца страницы	Ctrl+Enter
Вставка выражения в стандартной математической форме	Ctrl+R
Вставка выражения в форме Standart Math Input	Ctrl+G

Таблица 1.5. Горячие клавиши переходов по документу

Действие	Горячие клавиши
Переход к началу строки	Home
Переход в конец документа	Ctrl+End
Переход к концу строки	End
Переход в начало документа	Ctrl+Home
Создание новой строки	Shift+Enter
Переход к следующей строке ввода	Tab
Переход к предыдущей строке ввода	Shift+Tab
Переход к справке по контексту	Ctrl+F1 или F1
Переход на предшествующий уровень вложенности секций	Ctrl+UpArrow

Таблица 1.6. Горячие клавиши команд для работы с файлами

Команда	Горячие клавиши
Создание нового документа	Ctrl+N
Открытие документа	Ctrl+O
Сохранение документа	Ctrl+S
Печать документа	Ctrl+P
Закрытие активного окна (документа)	Ctrl+F4
Завершение работы с Maple	Alt+F4

Таблица 1.7. Горячие клавиши команд просмотра документа

Команда	Горячие клавиши
Перерисовка экрана (Redraw Screen)	Ctrl+L
Просмотр групп ячеек (Show Group Ranges)	F9
Показ секций (Show Section Ranges)	Shift+F9
Разделение строки на две части (Split Group)	F3
Объединение смежных строк (Join Group)	F4
Разделение секции на две части (Split Section)	Shift F3
Объединение смежных секций (Join Section)	Shift F4
Отмена предшествующей операции (Undo)	Ctrl+Z

Таблица 1.8. Горячие клавиши установки масштаба (Zoom Factor)

Масштаб	Горячие клавиши
50%	Ctrl+1
100%	Ctrl+2
150%	Ctrl+3
200%	Ctrl+4
300%	Ctrl+5
400%	Ctrl+6

Разумеется, можно успешно работать с системой, вообще ничего не зная про «горячие» клавиши. Многие так и делают! Однако быстрая и профессиональная работа в Maple 7 невозможна без использования этих клавиш, поскольку они существенно экономят время пользователя.

При работе с русифицированной версией Windows латинские буквы в обозначениях «горячих» клавиш нередко заменяются русскими буквами, не несущими никакой смысловой связи с выполняемой операцией. В этом случае полезно знать, что соответствующая латинская буква (или иной знак) и указанная в меню русская находятся на одной клавише (например, комбинация клавиш для сохранения документа (Ctrl+S) может быть показана как Ctrl+ы).

Доступ к справкам и примерам

В меню Help системы Maple 7 сосредоточены средства доступа к справке по всем функциям системы. Детально работу со справочной базой данных Maple 7 мы рассмотрим в следующей главе. Здесь лишь отметим, что справку по любой функции можно получить, просто установив на ее имени маркер ввода и нажав клавишу F1. На рис. 1.26 показано окно с началом справки по функции $\cos(x)$.

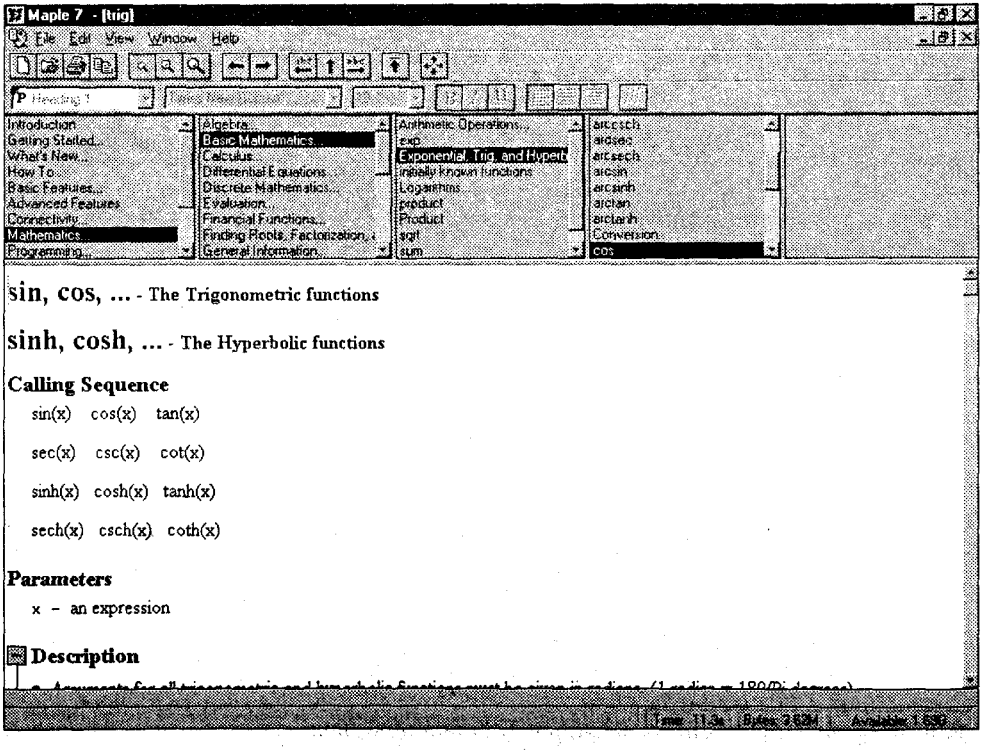


Рис. 1.26. Пример справки по функции косинуса

Как видно из рис. 1.26, окно справки содержит пятиступенчатый контекстный указатель, позволяющий последовательно отыскивать нужный раздел справки. Когда справка запрашивается по конкретной теме (функции), то сразу появляется посвященный ей раздел, точнее начало этого раздела.

Примеры из справочной системы можно модифицировать, для этого их нужно скопировать в буфер и перенести в окно документа Maple 7 (см. следующую главу). Также в Maple 7 есть специальный раздел справки, дающий доступ к примерам и без их копирования. Для осуществления такого доступа в окне справки достаточно выбрать тему ExampleWorksheets в первом же разделе контекстного указателя. Откроется окно (теперь уже документа) с индексным каталогом примеров (рис. 1.27).

Каталог примеров дает доступ к огромному числу примеров применения Maple 7. Просмотр одного из них (вычисление эллиптических интегралов) показан на рис. 1.27 справа. Рисунок 1.27 иллюстрирует также технику работы с двумя окнами документов.

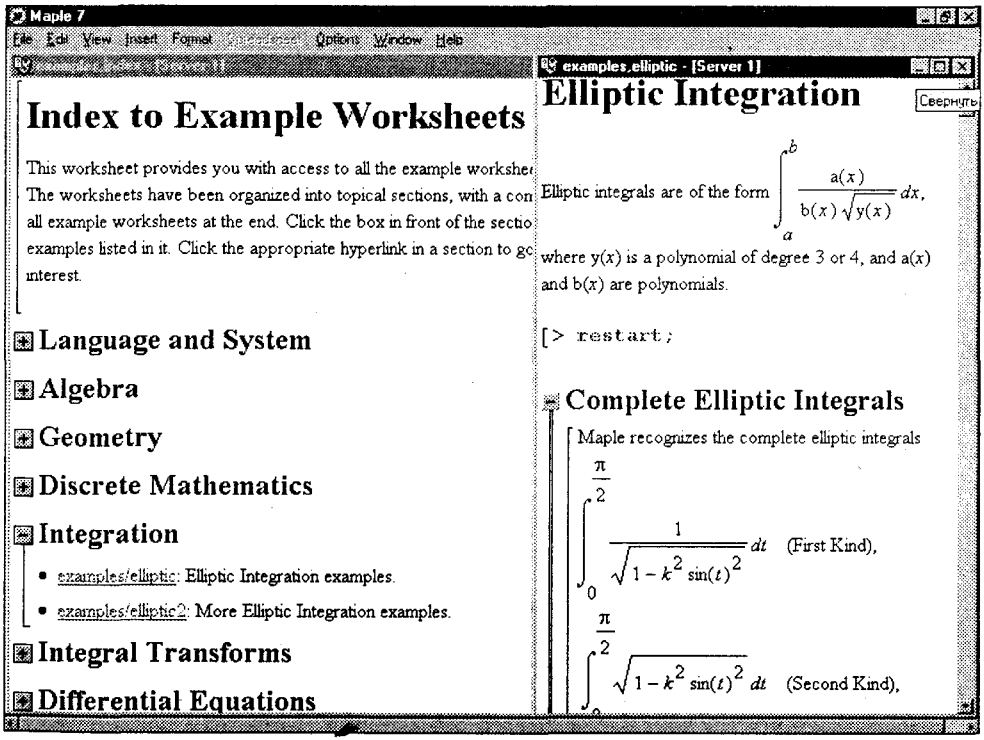


Рис. 1.27. Работа с каталогом примеров

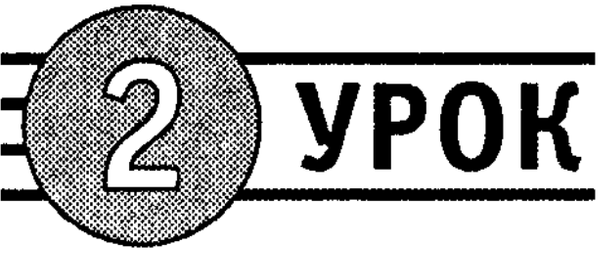
В этой книге использованы некоторые наиболее интересные примеры из справочной базы данных системы Maple 7. Однако большинство примеров книги не повторяет помещенные в справку Maple, носит оригинальный характер и отражает результаты исследования возможностей системы Maple 7 автора. В связи

с этим читателю рекомендуется просмотреть не только примеры в данной книге, но и примеры, представленные в справочной системе по интересующим читателя функциям.

Что нового мы узнали?

В этом уроке мы научились:

- Устанавливать Maple 7.
- Отличать языки системы — входной, реализации и программирования.
- Загружать систему и работать с ней в режиме диалога.
- Выполнять простые вычисления.
- Строить простые графики.
- Пользоваться кнопками быстрого управления системой.
- Обращаться к справкам и примерам применения системы Maple 7.



Информационная поддержка Maple

- Работа со справочной системой Maple 7
- Отражение систем класса Maple в Интернете
- Информационная поддержка Maple в Интернете
- Поддержка MathML
- Maple на Российских Интернет-сайтах
- Maple в карманном компьютере фирмы Casio

Работа со справочной системой

Меню Help

Справочной системе Maple 7 принадлежит исключительная роль — только в ней можно найти полную информацию обо всех почти трех тысячах функций Maple 7. Использование англоязычной справочной системы может быть полезно и для тех, кто и «двух слов по-английски связать не может», поскольку в ней приведен синтаксис функций и операторов, а также многочисленные примеры их применения — по самым скромным подсчетам их свыше десяти тысяч.

К сожалению, справочная система Maple 7 очень громоздка. Но это нельзя считать недостатком справочной системы, поскольку просто велик объем входящего в нее материала. В справочной системе имеются все присущие современным базам данных возможности для быстрого поиска нужной информации и даже для ее структурирования и пополнения.

Основные команды по работе со справочной системой Maple 7 сосредоточены в меню Help, показанном на рис. 2.1.

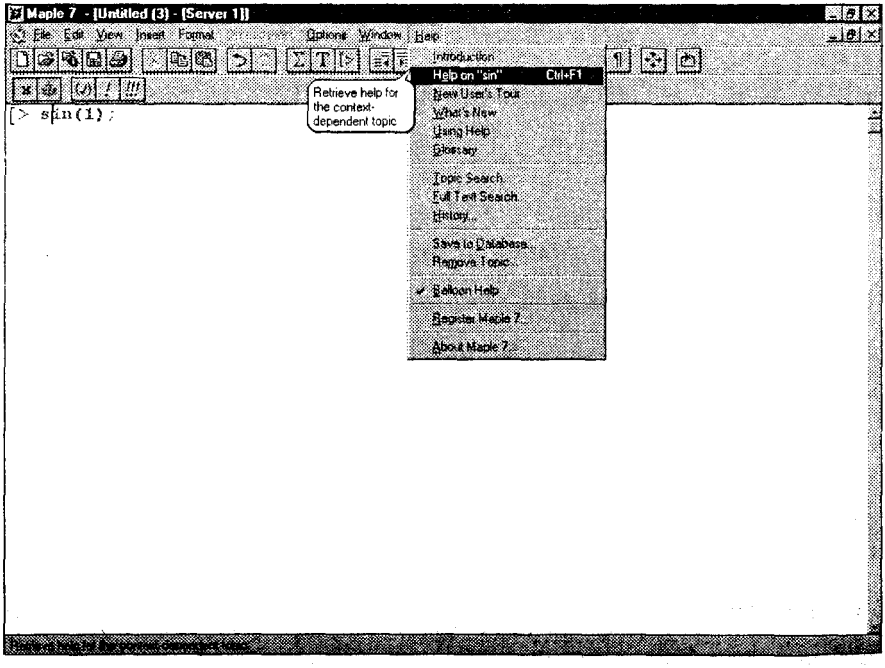


Рис. 2.1. Меню Help справочной системы Maple 7

Оно содержит команды, объединенные в несколько групп. В первую группу входят следующие команды:

- Introduction — показ начального раздела справки (введения);
- Help on Context — вывод оперативной справки по контексту;
- New User's Tours — запуск обучающей системы;
- What's New — описание новых возможностей системы;
- Using Help — описание правил использования справочной системы;
- Glossary — вывод указателя терминов.

Второй раздел меню содержит команды:

- Topic Search — предметный поиск по заданному образцу;
- Full Text Search — предметный поиск с полным обзором текста справки;
- History — вывод истории поиска.

В третьем разделе имеются две команды для работы с базой данных:

- Save to Database — запись данных в базу данных;
- Remove Topic — восстановление базы данных предметного поиска путем удаления дополнительных данных;

Остальные разделы представлены следующими командами:

- Balloon Help — включение всплывающих подсказок;
- Register Maple 7 — регистрация Maple 7;
- About Maple 7 — вывод окна с информацией о Maple 7.

Рассмотрим детально работу справочной системы Maple 7. Следует отметить, что ценность справочной системы для наших читателей намного снижается из-за того, что она написана на английском языке. Учитывая громоздкость справочной системы и необходимость в наличии компьютера для ее использования, для знакомства с системой Maple 7 более подходят обычные книги, тогда как справочную систему следует применять при необходимости ознакомиться с тонкими деталями применения тех или иных операторов, функций и иных средств Maple 7.

Просмотр введения

Команда Introduction в меню справки запускает справочную систему на странице введения (рис. 2.2).

В введении определено назначение Maple 7 как системы компьютерной алгебры и дается ссылка на сайт фирмы — разработчика системы (www.maplesoft.com). Щелкнув на гиперссылке, вы перейдете на начальную страницу web-сервера фирмы. На странице введения имеются также гиперссылки на обучающий курс (New User's Tour), на страницы с обзором новых возможностей Maple 7 и справки по различным элементам интерфейса.

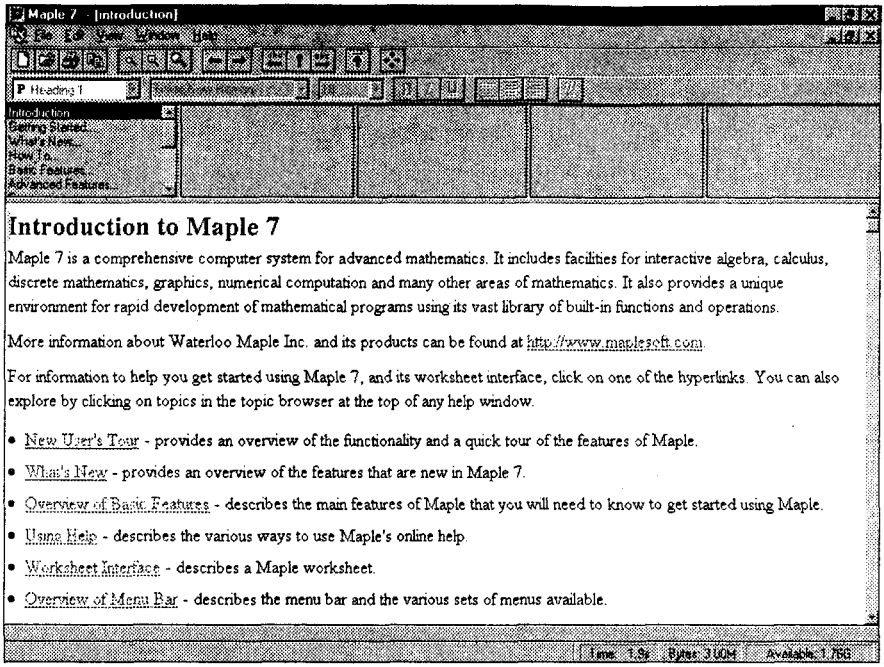


Рис. 2.2. Окно справочной системы с введением

Обратите внимание на оригинальный подход к представлению пути нужной справки. Для его определения служат 5 окон-списков. На рис. 2.2 используется только одно окно, а остальные 4 пока пусты. Выбирая последовательно по элементу из каждого списка, вы сможете достигнуть требуемой справочной информации. Подробнее навигация по справочной системе будет описана позже.

Оперативная справка по контексту

Оперативная справка по контексту — сравнительно новая возможность справочных систем. Она особенно удобна при разборе примеров, содержащих незнакомые пользователю функции и иные объекты системы Maple. Полезна она и в том случае, когда пользователь знаком с применяемым объектом, но хотел бы уточнить его свойства и синтаксис.

Для получения оперативной справки по контексту достаточно установить курсор на соответствующий объект, например на имя какой-либо функции, и открыть меню **Help**. В нем можно обнаружить, что операция **Help on Context** модифицируется и приобретает вид **Help on "..."**, где на месте многоточия стоит слово, на котором остановился курсор. На рис. 2.1 таким словом является имя функции, вычисляющей синус, — **sin**.

Допустим, что в тексте документа в строке ввода есть функция $\sin(x)$. Если теперь выполнить команду **Help on "..."** — тут же появится окно со справкой о функ-

ции синуса. Существуют и горячие клавиши для этой команды — Ctrl+F1 (или просто F1). Пример справки по функции sin представлен на рис. 2.3.

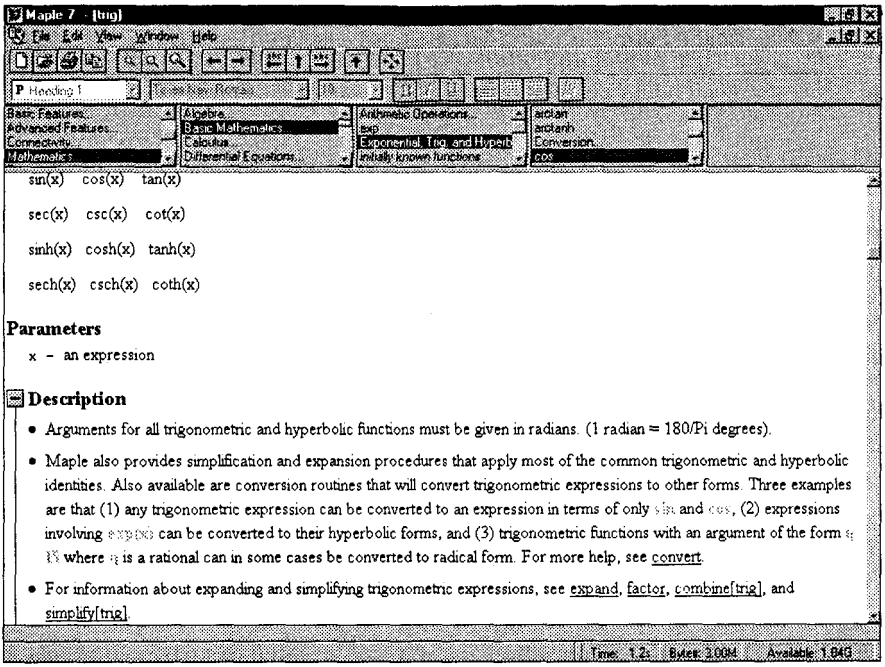


Рис. 2.3. Пример справки по функции sin

Как видно из рис. 2.3, справка по контексту позволяет судить о назначении функции, синтаксических правилах ее задания и примерах применения. Реализована она по единым правилам справочной системы и содержит открываемые разделы и гипертекстовые ссылки.

В этом примере уже хорошо видна техника использования окон в верхней части справки для уточнения необходимого раздела. Так, обнаруженная функция cos (кстати не sin, это любопытное свойство справки Maple — наличие «союзных» разделов) находится уже в четвертном окне. Последовательность доступа к ней следующая: Mathematics-Basic Mathematics-Exponential, Trig and Hyperbolic-cos. Очевидно, что функция cos найдена потому, что она выступает в паре с функцией sin, — на странице справки приведены примеры и к той, и к другой функции. Учитывая огромное число функций системы Maple 7 и соответственно разделов справки, подобный способ поиска информации представляется очень удобным. Он, кстати, стал использоваться и в Mathematica 3/4 — ближайшем конкуренте Maple.

Обучающий курс New User's Tour

Команда New User's Tour открывает окно курса по обучению основам пользования Maple, показанное на рис. 2.4.

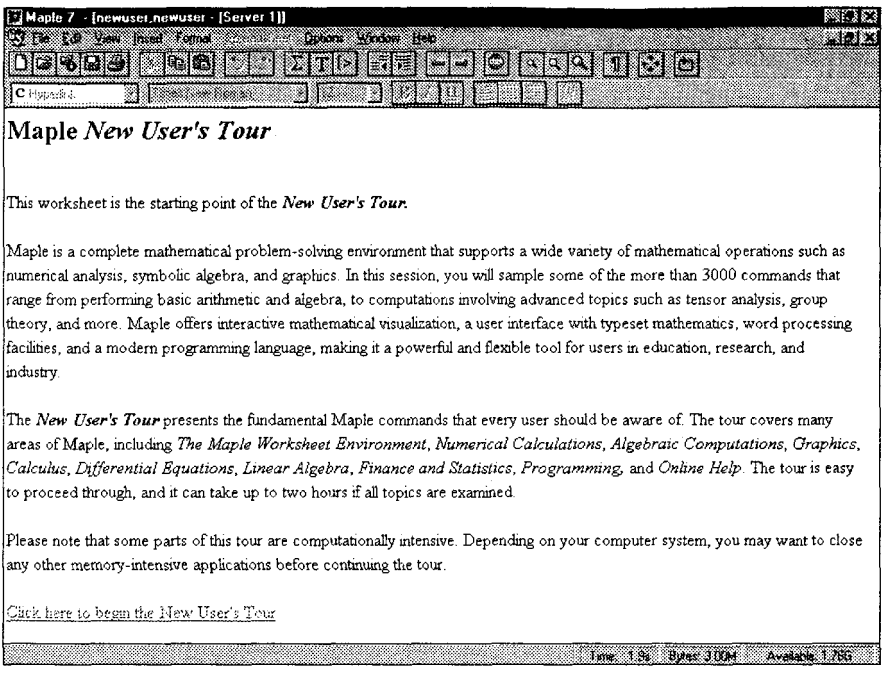


Рис. 2.4. Окно курса обучения основам Maple

В конце страницы (рис. 2.4) имеется гиперссылка Click here to begin the New User's Tour. Она открывает окно обучающего курса по Maple с перечнем ее разделов, представленное на рис. 2.5. Наименования разделов являются гиперссылками.

Как видно из рис. 2.5, обучающий курс имеет следующие разделы:

- (1) Working Through the New User's Tour — обучение работе с курсом;
- (2) The Worksheet Environment — создание документов;
- (3) Numerical Calculations — численные вычисления;
- (4) Algebraic Computations — алгебраические преобразования;
- (5) Graphics — графики;
- (6) Calculus — вычисления;
- (7) Differential Equations — дифференциальные уравнения;
- (8) Linear Algebra — линейная алгебра;
- (9) Finance and Statistics — финансы и статистика;
- (10) Programming — программирование;
- (11) Online Help — помощь через Интернет;
- (12) Summary — заключение.

Активизация любой из этих гиперссылок приводит к выводу соответствующего раздела обучающего курса. На рис. 2.6 представлено начало раздела 3, посвященного численным вычислениям.

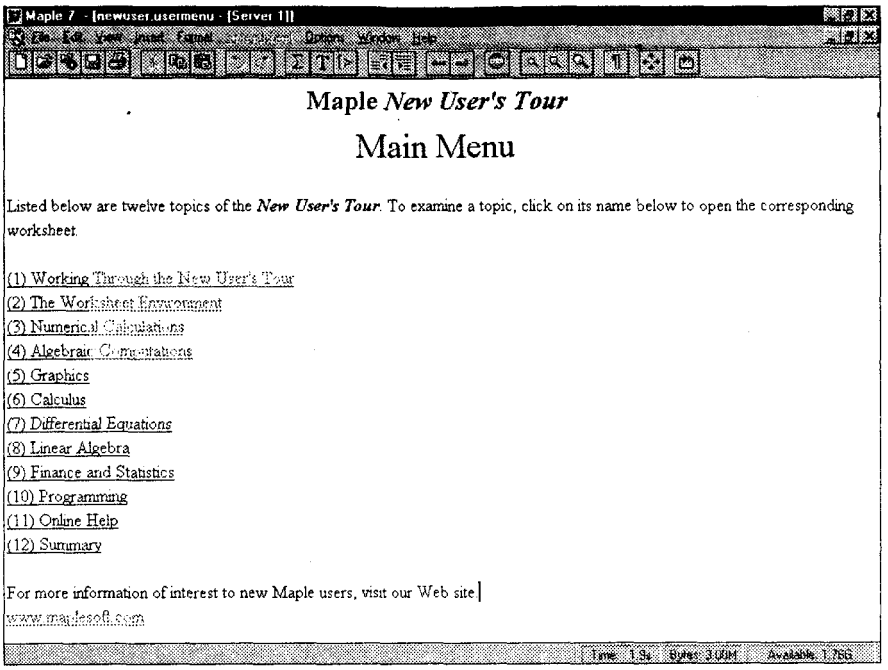


Рис. 2.5. Окно с перечнем разделов обучающего курса

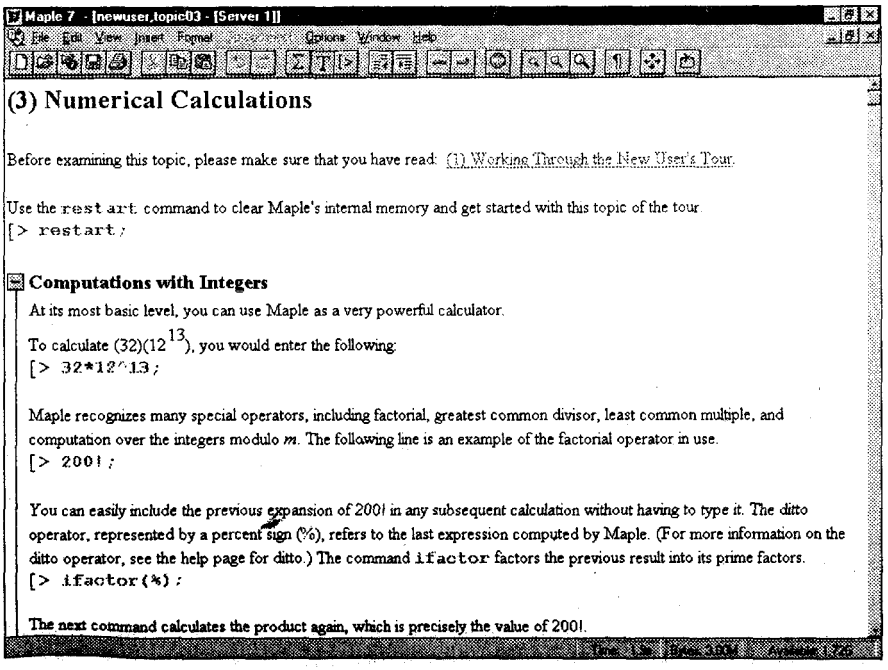


Рис. 2.6. Начало раздела обучающего курса по численным вычислениям

Основные достоинства обучающего курса в том, что он (в отличие от справочной базы данных) дает обычное описание работы с системой Maple 7 с «живыми» примерами, которые не надо копировать в документы. Фактически обучающая система является просто набором документов системы. Вначале примеры даны без ячеек вывода, которые появляются после исполнения команды Edit ▸ Execute ▸ Worksheet. Это иллюстрирует рис. 2.7.

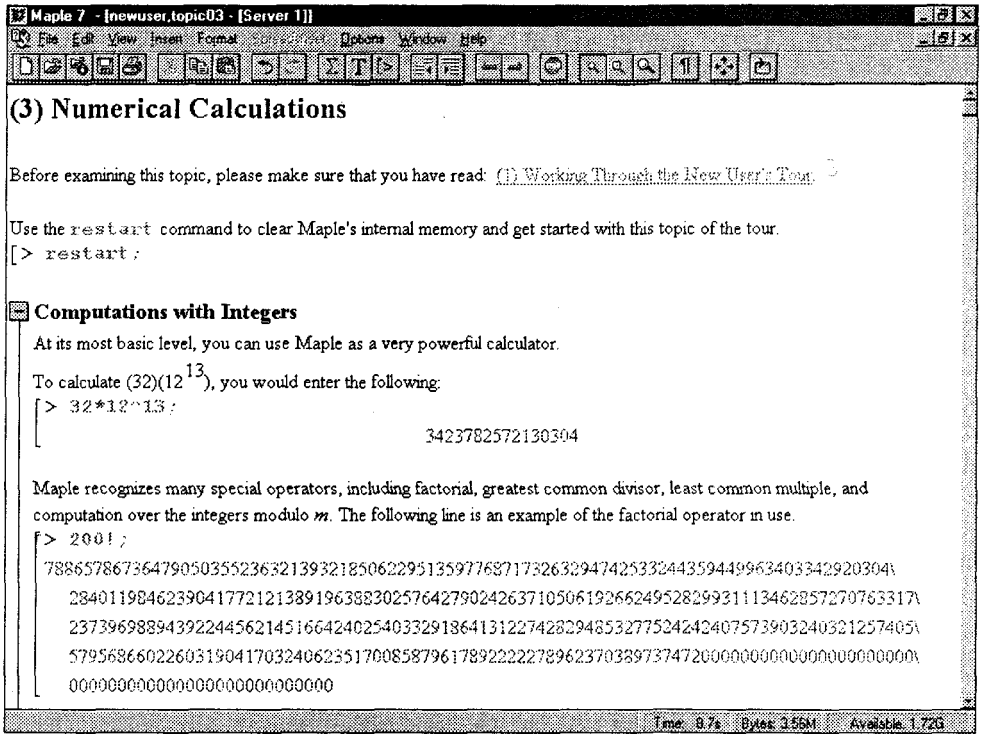


Рис. 2.7. Начало раздела обучающей системы по численным вычислениям после исполнения команды Edit ▸ Execute ▸ Worksheet

Пользователь может модифицировать любой пример и немедленно получить новые результаты (попробуйте, например, заменить вычисление 200! на 100! или 500!). К сожалению, написан обучающий курс на английском языке и в отличие от обычной книги для работы с ним нужен компьютер.

Новые возможности Maple 7

Пользователи, знакомые с предшествующими версиями системы Maple, обычно хотят узнать, что нового введено в последней версии. Однако, как у нас говорят, «лучшее — враг хорошего» — при большом числе новых возможностей (и особенно при модификации старых возможностей) появляется несовместимость между документами для старых и новых версий системы. Несмотря на меры, предпринятые по предотвращению такой несовместимости, ее проявление вполне возможно,

в частности документы, созданные в Maple 7, уже нельзя использовать в предшествующих версиях из-за различий в их внутренних форматах.

Выполнение команды What's New открывает окно с описанием новых возможностей Maple 7. Оно показано на рис. 2.8.

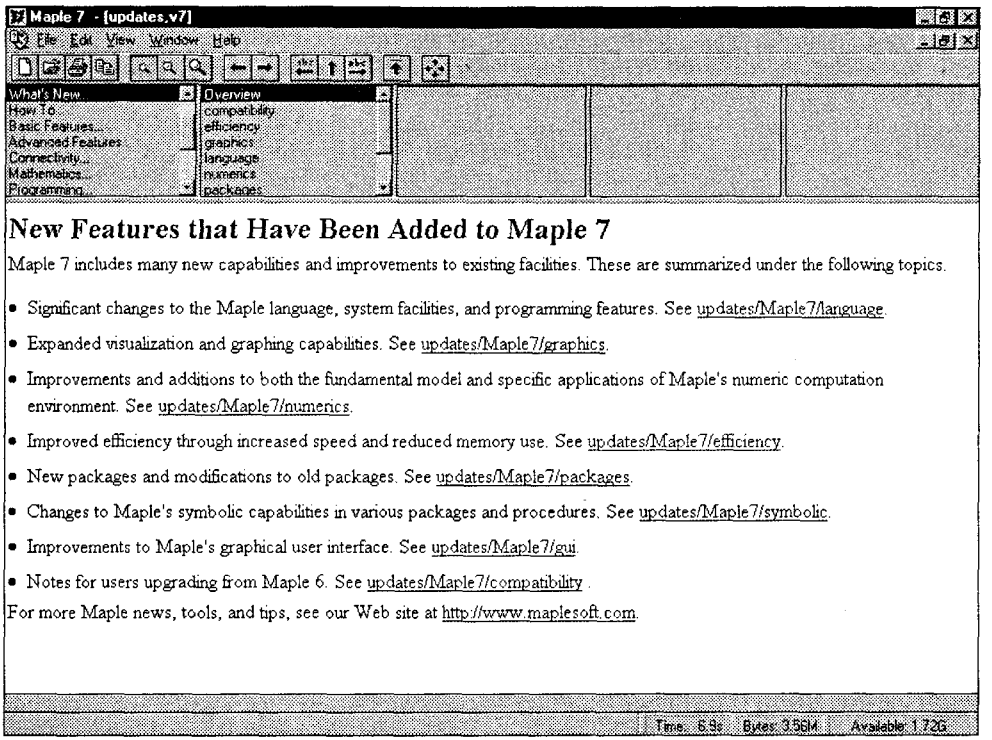


Рис. 2.8. *Окно справки с описанием новых возможностей системы Maple 7*

С помощью гиперссылок на этой странице можно получить достаточно подробное описание всех появившихся в Maple 7 возможностей. Мы уже отмечали их в уроке 1.

Правила работы со справочной системой

Справочная система Maple 7, по существу, является мощной базой данных с обширными возможностями поиска нужной информации и многочисленными примерами применения Maple 7. Работа с такой системой может вызвать затруднения у начинающих пользователей, поэтому в состав справочной системы включено описание правил ее использования. При исполнении команды Using Help появляется страница с перечнем разделов описания справочной системы (рис. 2.9).

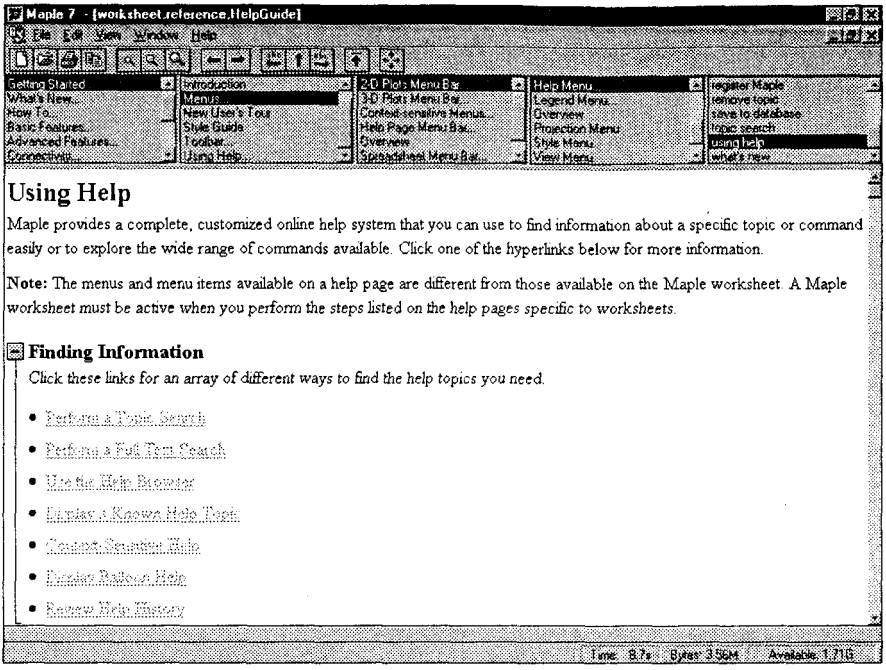


Рис. 2.9. Окно с перечнем разделов описания справочной системы

Мы не будем подробно описывать содержание страницы и гиперссылок, поскольку весь материал данного урока и является, по существу, таким описанием.

Предметный поиск

Команда Topic Search (предметный поиск) — одна из самых мощных функций справки. Она выводит окно поиска (рис. 2.10), содержащее в верхней части поле для ввода образца. Образцом может быть слово (например, имя функции) или даже часть слова. В окне под этим полем появляется список всех объектов Maple 7, в индекс которых входит заданный образец (рис. 2.10).

Теперь остается из заданного списка слов выбрать нужное, что приведет к появлению окна справки с информацией по данному слову. Иногда индекс по данному слову будет иметь несколько ссылок, на появившейся странице вам придется уточнить, справку по какому объекту вы хотите получить. Окно предметного поиска в правой части имеет четыре кнопки со следующим назначением:

- Search — поиск по образцу;
- Apply — вывод окна выбранного раздела справки при сохранении окна поиска;
- OK — окончание поиска (выводится окно с выбранным разделом и исчезает окно поиска);
- Cancel — закрытие окна поиска.

Кроме того, имеется возможность задания двух параметров:

- Same Window — вывод в то же окно (если окно не открыто на полный экран);
- Auto-Search — автоматический поиск по образцу по мере ввода последнего.

Заметим, что если установлен флажок Auto-Search, то кнопка Search становится недоступна, поскольку отпадает необходимость в ней.

Предметный поиск с полным обзором текста справки

Предметный поиск с полным обзором текста справки (Full Text Search) — еще один эффективный метод получения справочной информации. Он напоминает ранее рассмотренный, но вхождение заданного образца ищется во всем тексте справочной системы, а не только по индексу, составленному создателями Maple. При этом выводится окно, подобное показанному на рис. 2.11. Обратите внимание на то, что в большом поле этого окна указано существенно больше разделов справки, чем при поиске по индексу — очевидно, что большинство из них будет случайным упоминанием данного образца.

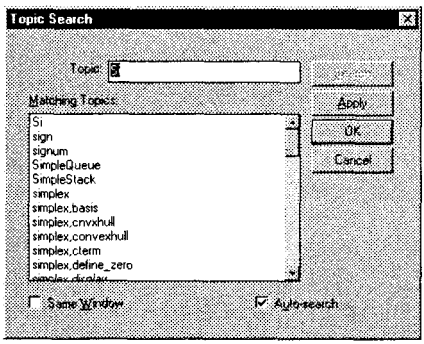


Рис. 2.10. Окно предметного поиска

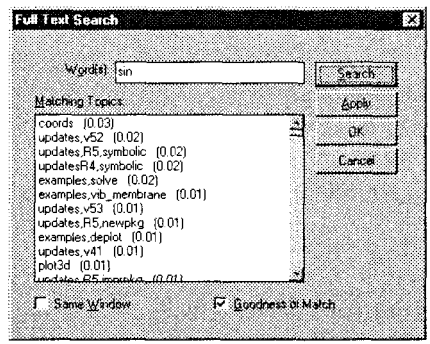


Рис. 2.11. Окно предметного поиска с полным обзором текста справки

В этом окне используются те же кнопки управления, что и для окна поиска справки по образцу. Флажок Goodness of Match включает вывод статистики повторяемости образца в текстах разделов справочной системы.

История работы со справкой

Было замечено, что пользователь, занятый решением определенного класса задач, обычно неоднократно возвращается к ранее просмотренным разделам справочной системы. Чтобы не искать их всякий раз заново, справочная система хранит список разделов, просмотренных в данном сеансе работы. Он выводится операцией History (рис. 2.12).

Достаточно найти в этом окне нужный раздел справки и нажать кнопку OK, чтобы вывести его на экран. Кнопка Apply позволяет просматривать каждый новый раздел в своем окне.

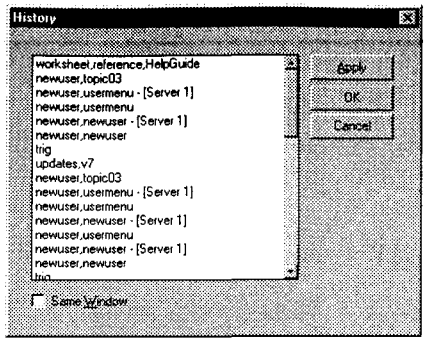


Рис. 2.12. Список ранее просмотренных разделов справки

Модернизация справочной базы данных

В справочной базе данных предусмотрена возможность ее расширения путем записи текущего документа, составленного пользователем, в указанный раздел. При выполнении операции Save to Database выводится специальное окно, в котором надо указать соответствующие данные о модернизируемом разделе справки. Вид окна представлен на рис. 2.13.

Ограничимся этим указанием, учитывая, что для нашего пользователя модернизация англоязычной справочной базы данных явно отдает экзотикой.

Удаление разделов базы данных

Для удаления разделов базы данных служит команда Remove Topic. Она выводит окно, показанное на рис. 2.14.

Обратите внимание, что модернизации в обоих случаях подвергается один и тот же файл базы данных maple.hdb.

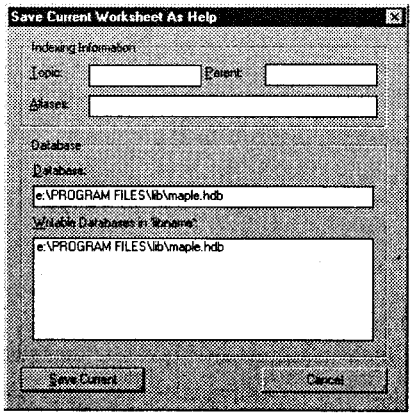


Рис. 2.13. Окно дополнения базы данных

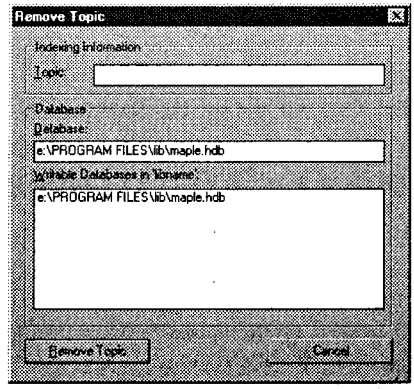


Рис. 2.14. Окно удаления разделов справочной базы данных

Включение всплывающих подсказок

На первом этапе знакомства с пользовательским интерфейсом системы Maple 7 удобно использовать всплывающие подсказки. Они вводятся при установке флажка Balloon Help. Для получения подсказки по какому-либо объекту достаточно задержать на пару секунд указатель мыши на этом объекте. Пример вывода всплывающей подсказки показан на рис. 2.1 — она указывает на назначение команды меню Help — Help on "sin".

Регистрация системы

Команда Register Maple 7 меню справочной системы открывает окно регистрации системы Maple 7. Это окно позволяет вызвать браузер Интернета, с помощью которого выполняется регистрация.

Вывод окна с информацией о системе

Последняя команда меню Help — About Maple 7 — выводит окно с информацией о версии Maple 7 (рис. 2.15). В этом окне содержатся данные, необходимые для регистрации системы Maple 7 (номера лицензии и самого продукта), а также данные о времени выпуска системы.

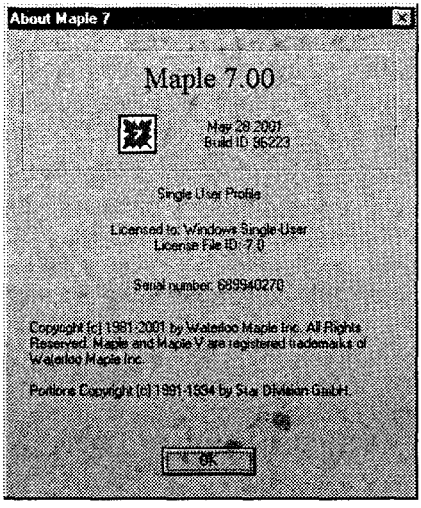


Рис. 2.15. Окно с данными о системе Maple 7

Обратите внимание, что данный продукт датирован концом мая 2001 г. Как уже отмечалось в уроке 1, официально он выпущен на рынок 21 июня 2001 г.

Информационная поддержка Maple 7 в Интернете

Значение Интернета в информационной поддержке

В последние годы все разработчики систем компьютерной математики уделяют огромное внимание информационной поддержке таких систем в глобальной общемировой сети. Это обусловлено следующими обстоятельствами:

- современные системы компьютерной математики — очень сложные и быстро развивающиеся программные средства, нуждающиеся в оперативной рекламе и, главное, в оперативной коррекции их возможностей и устранении обнаруженных недостатков;
- справочные базы систем компьютерной математики очень обширны и нуждаются в постоянном обновлении;
- для практического изучения систем компьютерной математики желательна постоянная модернизация примеров их применения, для чего Интернет предоставляет обширные возможности;
- при решении сложных научных задач и при разработке серьезных проектов в образовании (данная книга наглядный пример этого) необходима связь пользователей с разработчиками программных средств и со своими коллегами во всем мире;
- современные технологии программирования, такие как Java-апплеты и ActiveX, новые языки передачи структурированной информации, такие как XML и MathML, широко применяются как в Интернете, так и в системах компьютерной математики.

Для использования возможностей Интернета компьютер должен быть оснащен модемом или подключен к локальной сети, имеющей вход в Интернет, и располагать необходимым программным обеспечением — прежде всего браузером и программой электронной почты. Необходимое программное обеспечение часто включается в состав операционной системы, хотя его можно приобрести и отдельно. Например, в Windows стандартаным браузером является Internet Explorer, а почтовой программой — Outlook Express фирмы Microsoft. Есть множество подобных программ и у других фирм. Как правило, доступ к ресурсам Всемирной паутины предоставляет провайдер — фирма, через сервер которой вы будете подключены к Интернету.

Подключение к интернет-серверу фирмы Waterloo Maple

Если все перечисленное выше есть, то вход в Интернет осуществляется прямо из вводного раздела справочной системы путем щелчка на гиперссылке с адресом начальной страницы фирмы Waterloo Maple — http://www.maplesoft.com. Это

приведет к автоматической загрузке используемого по умолчанию браузера — в дальнейшем предполагается, что это Internet Explorer 5.0/5.5. Начальный этап входа в Интернет показан на рис. 2.16.

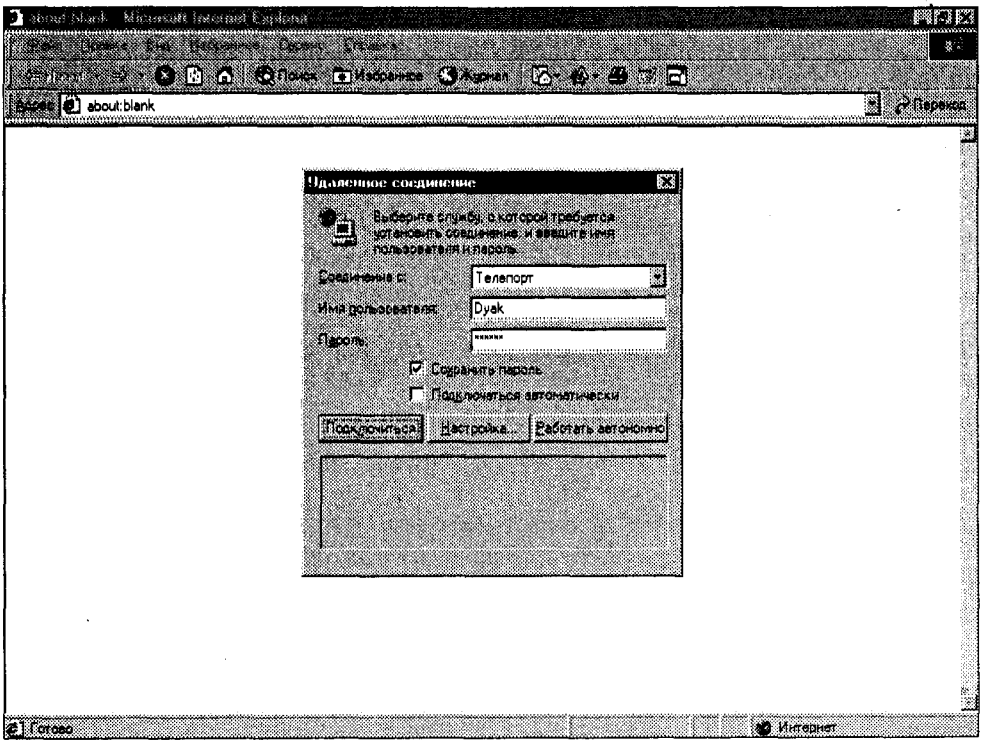


Рис. 2.16. Начало входа в Интернет

Как видно из рис. 2.16, активизация указанной ссылки приводит к появлению окна браузера и окна подключения к Интернету. В этом окне должны быть заданы имя соединения, имя и пароль пользователя (при вводе пароля каждый знак заменяется звездочкой, чтобы пароль не могли прочесть посторонние). Окно подключения на рис. 2.16 дано не более как пример — каждый пользователь должен занести в него свои данные.

Начальная страница корпорации Waterloo Maple

После нажатия кнопки Подключиться можно наблюдать подключение к Интернету — начало работы модема и ряд сообщений о ходе подключения. Обычно через несколько секунд происходит подключение компьютера через модем к узлу провайдера, после чего загружается начальная страница корпорации Waterloo Maple, показанная на рис. 2.17.

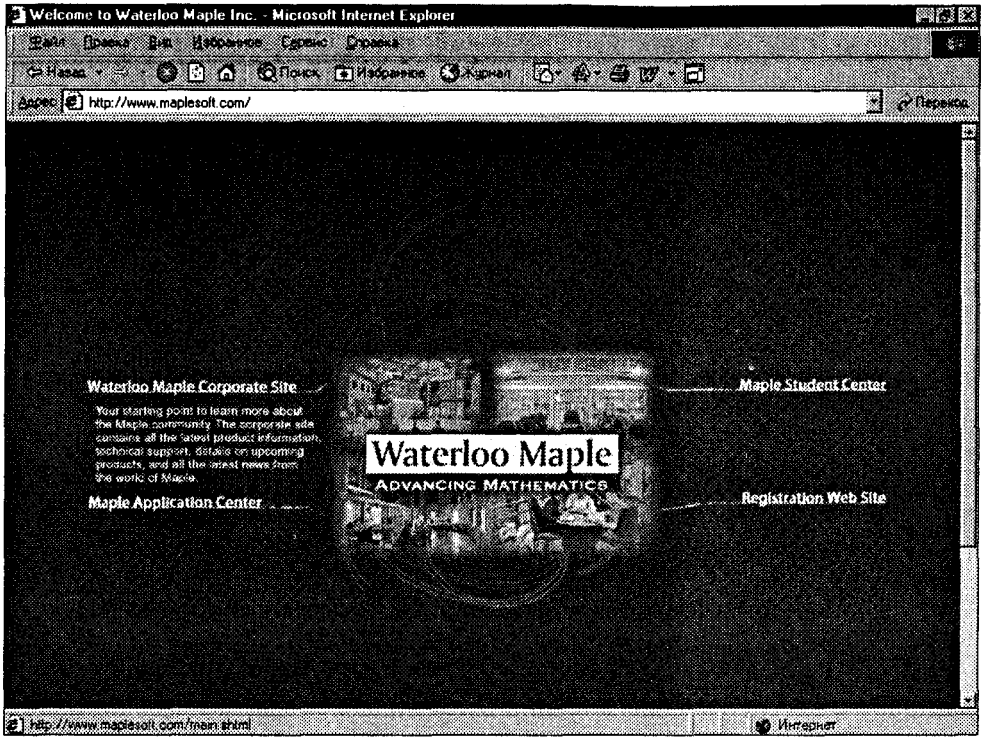


Рис. 2.17. Начальная страница фирмы-разработчика системы Maple 7 — корпорации Waterloo Maple

Начальная страница не содержит никакой конкретной информации. Она представлена лишь красочным рисунком и четырьмя ссылками:

- Waterloo Maple Corporate Site — выход на главную страницу корпорации Waterloo Maple;
- Maple Application Center — выход на страницу Центра применений системы Maple;
- Maple Student Center — выход на страницу Студенческого центра;
- Registration Web Site — выход на web-страницу регистрации.

Обратите внимание на то, что наведение курсора мыши в браузере на соответствующую гиперссылку вызывает появление всплывающей подсказки по ней — на рис. 2.17 такая подсказка выведена для ссылки на начальную страницу корпорации Waterloo Maple.

Главная страница корпорации Waterloo Maple

Щелкнув на гиперссылке Waterloo Maple Corporate Site, можно выйти на главную страницу корпорации Waterloo Maple. Она показана на рис. 2.18.

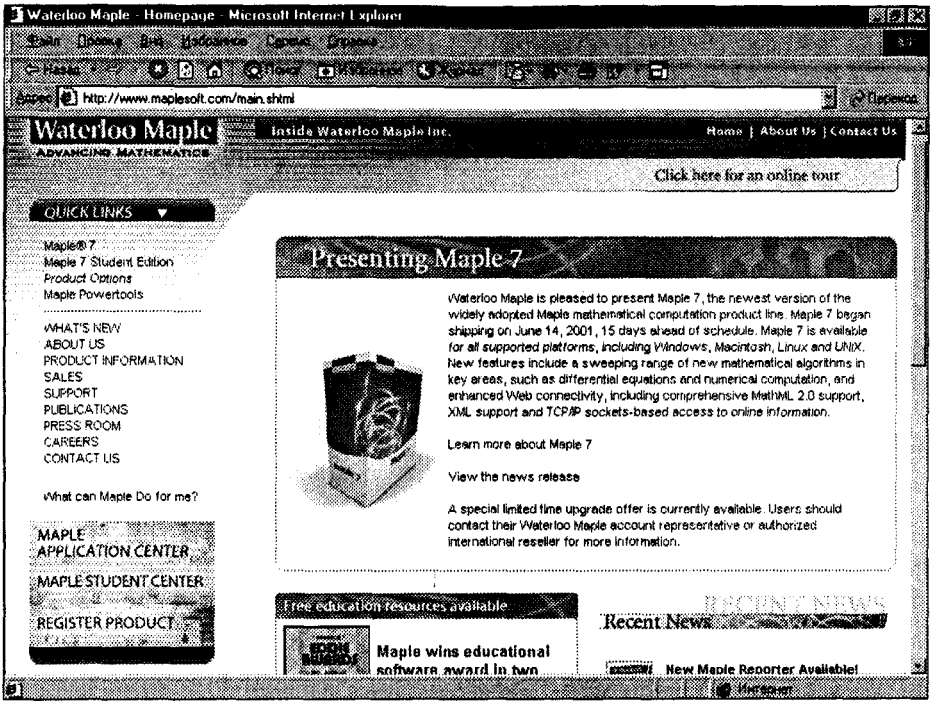


Рис. 2.18. Главная страница фирмы-разработчика системы Maple 7 — корпорации Waterloo Maple

На этой странице можно найти ряд гиперссылок, поясненных ниже:

- Maple 7 — информация об основной версии Maple 7;
- Maple 7 Student Edition — информация о студенческой версии Maple 7;
- Products Options — информация о продукции фирмы;
- Maple Powertools — мощный инструментальный пакет для системы Maple;
- What's New — последние новости;
- About As — информация о Waterloo Maple;
- Product Information — информация о продуктах;
- Sales — информация о продаже программных продуктов и ценах;
- Support — поддержка программных продуктов;
- Publications — информация о публикациях (статьях и книгах);
- Press Room — пресс-конференция;
- Careers — сведения о вакансиях в фирме;
- Contact Us — контактные данные.

Имеется и еще одна группа гиперссылок:

- Application Center — доступ к Центру применений Maple;

- Maple Student Center — доступ в Студенческий центр;
- Register Product — доступ к web-странице регистрации.

Щелчок на той или иной гиперссылке приводит к переходу на соответствующую страницу сайта фирмы Waterloo Maple.

ВНИМАНИЕ

Вид web-страниц время от времени меняется — от косметической доработки до кардинального изменения имиджа и содержания. Поэтому для приведенных в этом уроке страниц нет никакой гарантии, что их вид не изменится — даже к моменту опубликования данной книги. Тем не менее, как бы не выглядели web-страницы, большинство указанных разделов на них сохранится.

Информация о продукции

Главная страница сайта фирмы Waterloo Maple выводит информацию о продукции фирмы. Помимо стандартной версии Maple 7 выпускается упрощенная студенческая версия и демонстрационная (trial) версия с ограниченным временем использования (30 дней). На сайте можно найти также информацию о предшествующих версиях системы Maple. Фирма выпускает также множество пакетов-библиотек функций к Maple 7.

Вид страницы, специально посвященной системе Maple 7, приведен на рис. 2.19.

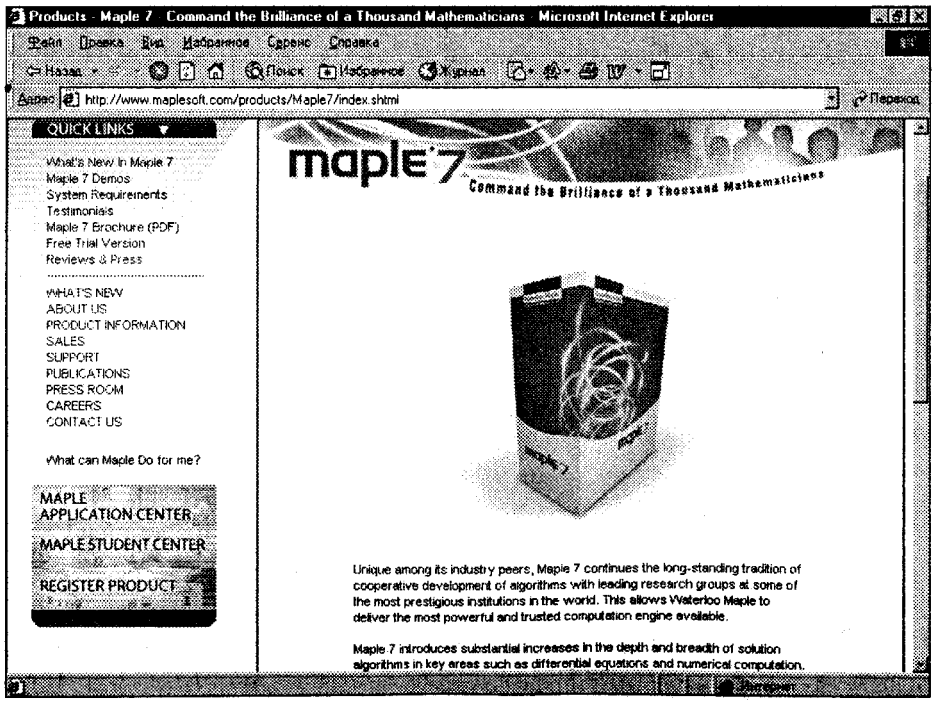


Рис. 2.19. Страница, посвященная Maple 7

Здесь можно найти информацию о получении демо-версии системы (с ограниченными возможностями, но бесплатно), просмотреть примеры демонстрации возможностей Maple 7, получить брошюру по Maple 7 в формате PDF, ознакомиться с отзывами прессы по этой системе и, наконец, получить доступ к уже отмеченным другим информационным ресурсам.

Информация о системных требованиях для работы с Maple 7 приведена на web-странице, изображенной на рис. 2.20. Maple 7 — сложная программа, и для успешной работы с ней нужен достаточно «приличный» компьютер — если и не самый новейший, то по крайней мере с процессором не хуже, чем Pentium.

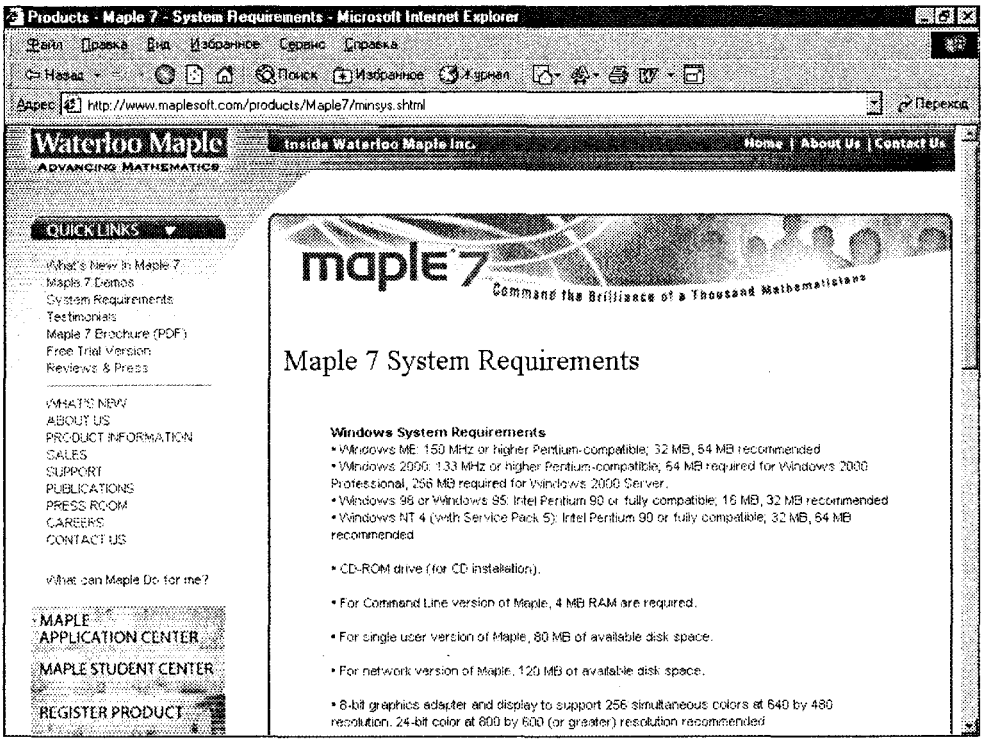


Рис. 2.20. Web-страница с данными о системных требованиях к компьютеру для работы с Maple 7

В уроке 1 уже приводились системные требования для версии Maple 7, предназначенной для компьютера с операционной системой Windows 95/98. На указанной странице можно найти такие требования и для других операционных систем, например Windows 2000, Unix, Linux и др.

Информация о покупке Maple 7

Информация о покупке программных продуктов содержится на страницах, к которым можно перейти, щелкнув на гиперссылке **Sales**. Вид первой страницы этого раздела сайта показан на рис. 2.21.

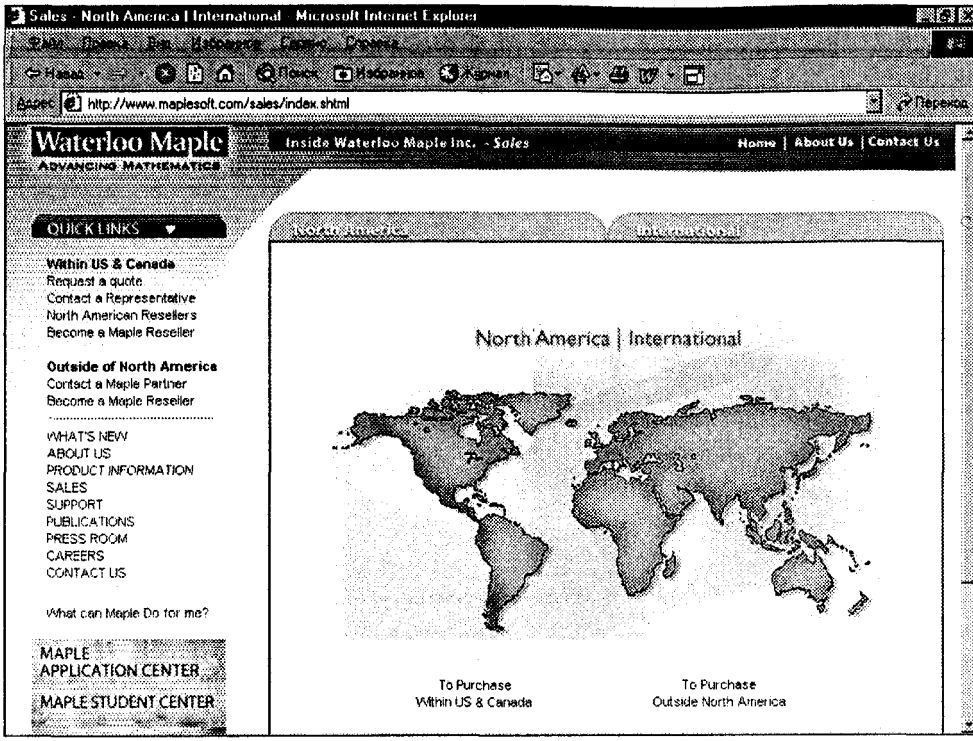


Рис. 2.21. Первая страница раздела сайта, предоставляющая информацию о покупке Maple 7

Карта на этой странице символизирует, что «щупальца» корпорации Waterloo Maple и ее представителя Maple Software охватывают весь Земной шар. Однако надо помнить, что во многих странах Maple 7 можно приобрести только через региональных представителей (или покупкой «левого» компакт-диска). В России «приличные» пользователи могут заказать Maple 7 через Российскую фирму SoftLine — www.softline.ru.

Информация о поддержке программных продуктов

Maple 7, как и предшествующие версии Maple, относится к весьма сложным программным продуктам, требующим обновления и грамотного применения. Поэтому важное значение приобретает поддержка таких продуктов в ходе их эксплуатации со стороны фирмы Waterloo Maple. Этому и посвящена страница поддержки, показанная на рис. 2.22.

Поддержка обеспечивается только для легально приобретенных продуктов, номера и лицензии которых занесены в базу данных фирмы. Поэтому пользователи нелегальных копий системы такой поддержки лишены, и обращаться за ней через Интернет бесполезно — вы получите вежливое сообщение о том, что ваш продукт не зарегистрирован.

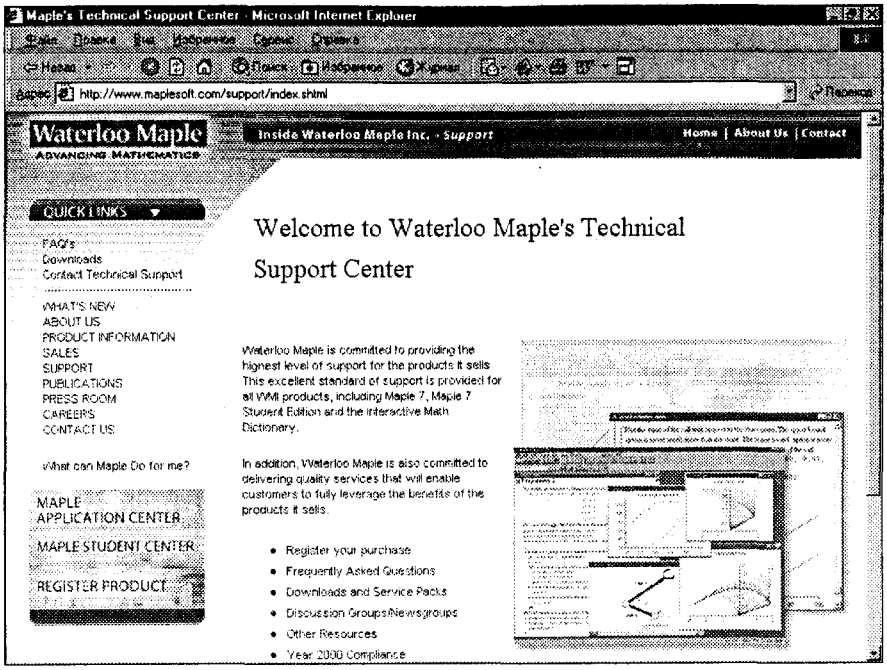


Рис. 2.22. Страница, посвященная Центру информационной и технической поддержки систем Maple

Информация о публикациях

На основной странице публикаций сайта можно найти информацию о многих сотнях книг и тысячах статей, посвященных системе Maple (рис. 2.23).

Как видно из рис. 2.23, эта страница содержит гиперссылки на книги и статьи по системе Maple и специальное электронное издание по ней — «The Maple Reporter».

Особый интерес представляет раздел публикаций, посвященный книгам, поскольку только в довольно большой книге можно описать достаточно серьезно и подробно такую сложную программу, как Maple. Щелкнув на гиперссылке Maple Books, можно открыть web-страницу, посвященную книгам по системам Maple, изданным во всем мире. На рис. 2.24 показана часть этой страницы со списком книг, изданных на русском языке. Названия книг представлены в виде гипертекстовых ссылок, щелчок на которых позволяет получить более подробную информацию о каждой книге.

Обратите внимание на выпадающие списки для выбора категории книги и языка (список по языкам на рис. 2.24 открыт). На момент подготовки данной книги среди отраженных на сайте публикаций русскоязычных книг по системе Maple 7 не было. Однако большинство книг по прежним версиям уже нашло отражение на сайте (рис. 2.25).

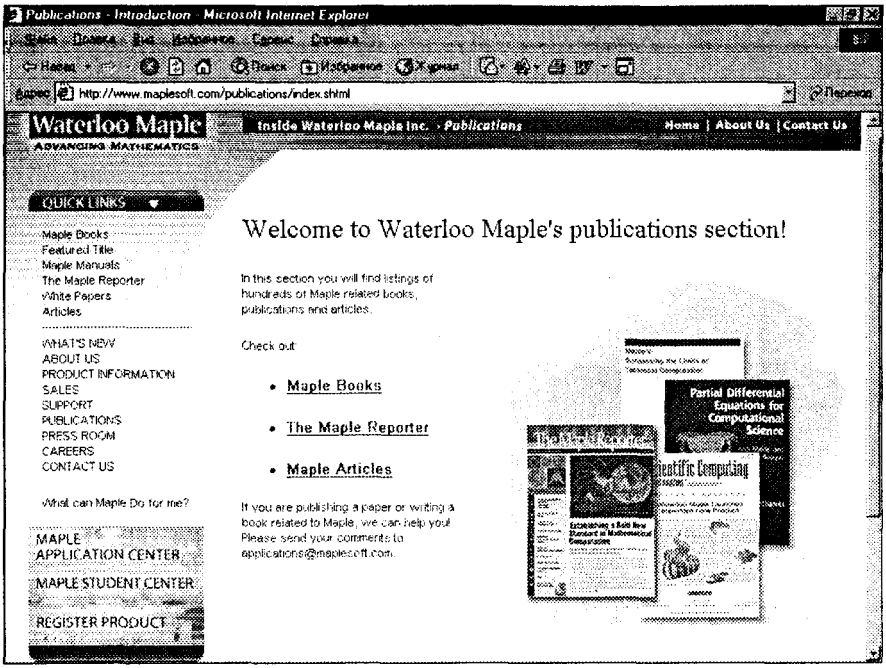


Рис. 2.23. Основная страница, посвященная публикациям по системам Maple

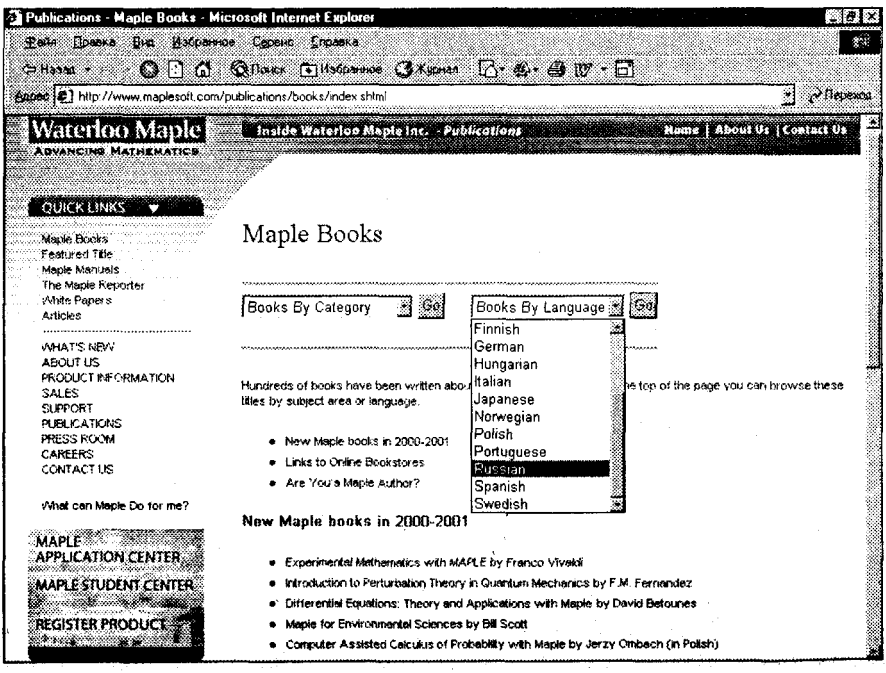


Рис. 2.24. Web-страница, посвященная книгам по системе Maple на всех языках мира

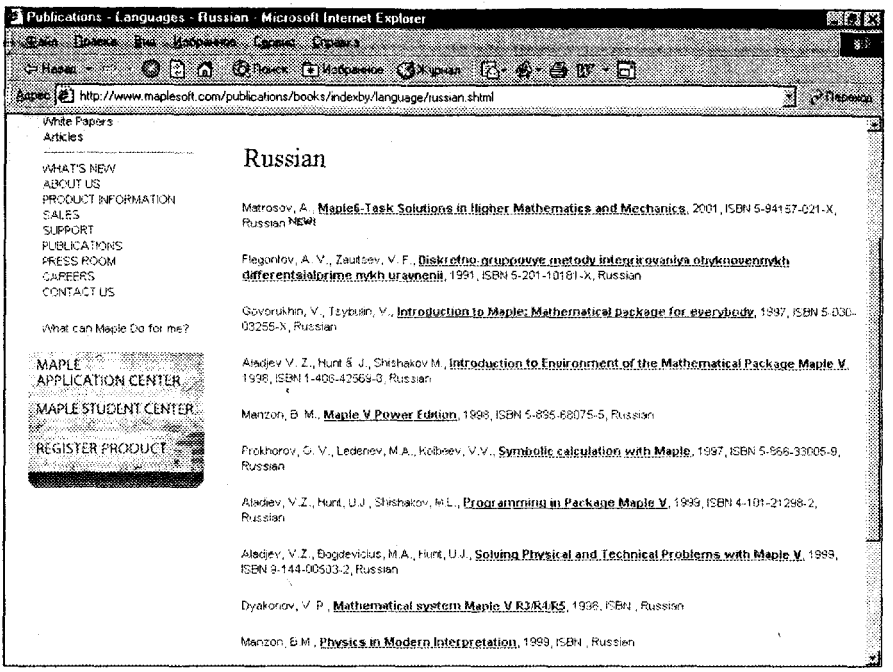


Рис. 2.25. Web-страница, посвященная русскоязычным книгам по системе Maple

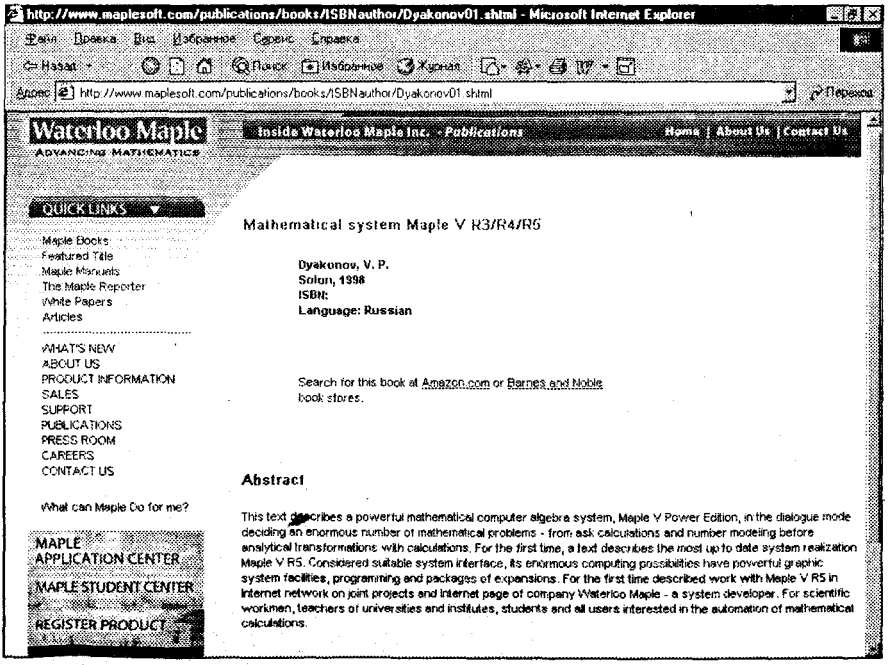


Рис. 2.26. Информация об одной из книг по Maple V R3/R4/R5

В качестве примера информации о книгах на рис. 2.26 представлена информация об одной из книг автора, посвященная Maple V R3/R4/R5. Нетрудно заметить, что приведены только аннотационные данные.

Необходимо отметить, что русскоязычных книг по системам Maple (только предшествующих версий) представлено довольно много, что свидетельствует о высокой популярности данной программы у наших пользователей.

Центр применений системы Maple

Основная страница Центра применений Maple

Центр применений Maple — это большой интерактивный раздел сайта компании Waterloo Maple, содержащий массу примеров применения Maple во всех областях науки, техники и образования. На рис. 2.27 представлено начало страницы The Maple Application Center.

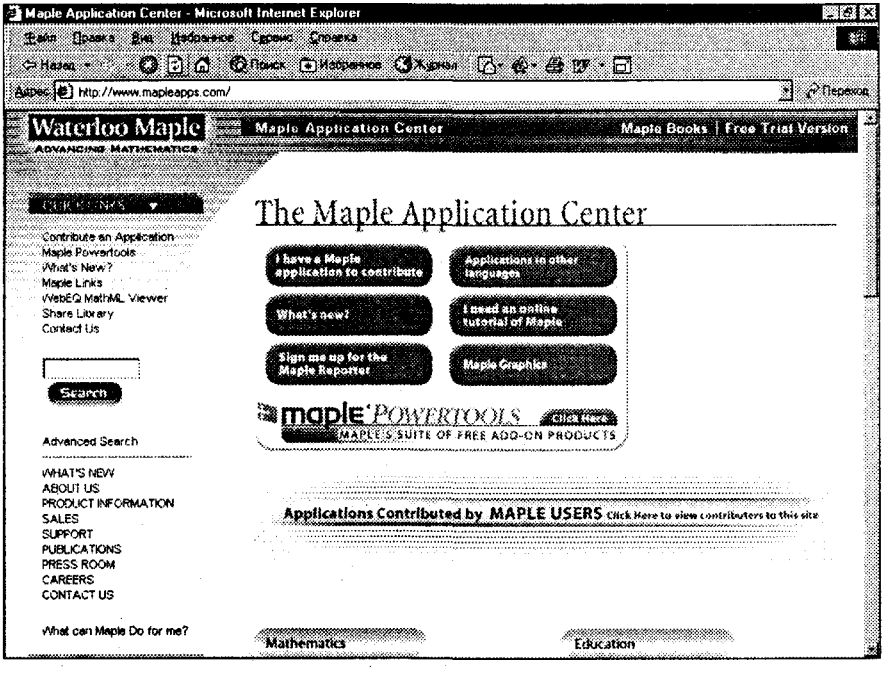


Рис. 2.27. Страница Центра применений Maple (начало)

Следует отметить, что большинство материалов этого раздела содержат примеры применения предшествующих версий системы Maple, но уже есть и много примеров для новейшей версии Maple 7, описанной в данной книге. На рис. 2.28 показано окончание страницы The Maple Application Center, где представлен обширный список тем, по которым имеются примеры в этом Центре.

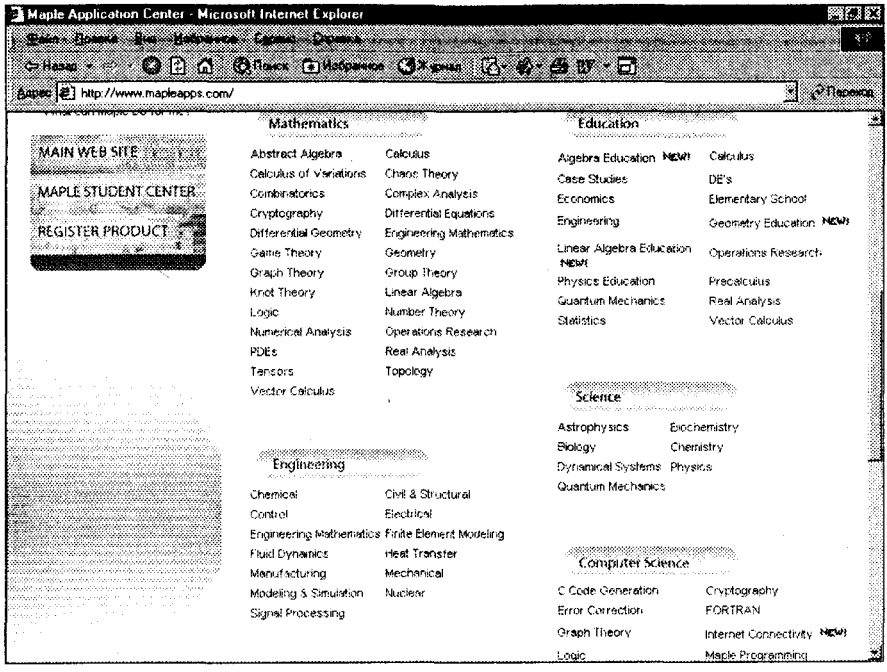


Рис. 2.28. Страница Центра применений системы Maple (окончание)

Приведенные в разделе примеры постоянно обновляются. В центре применений есть множество материалов образовательного характера — от отдельных демонстрационных примеров применения Maple в курсах математики, физики, химии и в других науках, до полноценных обучающих комплексов.

Информация о примерах

Рассмотрим работу с примерами из Центра применений Maple. Прежде всего надо выбрать раздел, в котором содержится интересующий вас пример. На рис. 2.29 показан раздел, посвященный решению дифференциальных уравнений.

В этом разделе предусмотрено 4 варианта работы с примерами:

- Details — просмотр примера в деталях;
- View Documents — просмотр примера прямо в Интернете (с помощью броузера);
- Download Worksheet — загрузка примера в среду Maple с возможностью его исполнения уже в ней;
- Download Code — загрузка кодов примеров (возможна в исключительных случаях, например для модификации Maple подключением новых функций).

Щелкнув на гиперссылке Details, можно открыть окно с данными о примере. Оно показано на рис. 2.30 слева для выбранного примера Laplace's equation in a cylinder. В окне сообщаются данные об авторе примера, дате создания, аннотации, при-

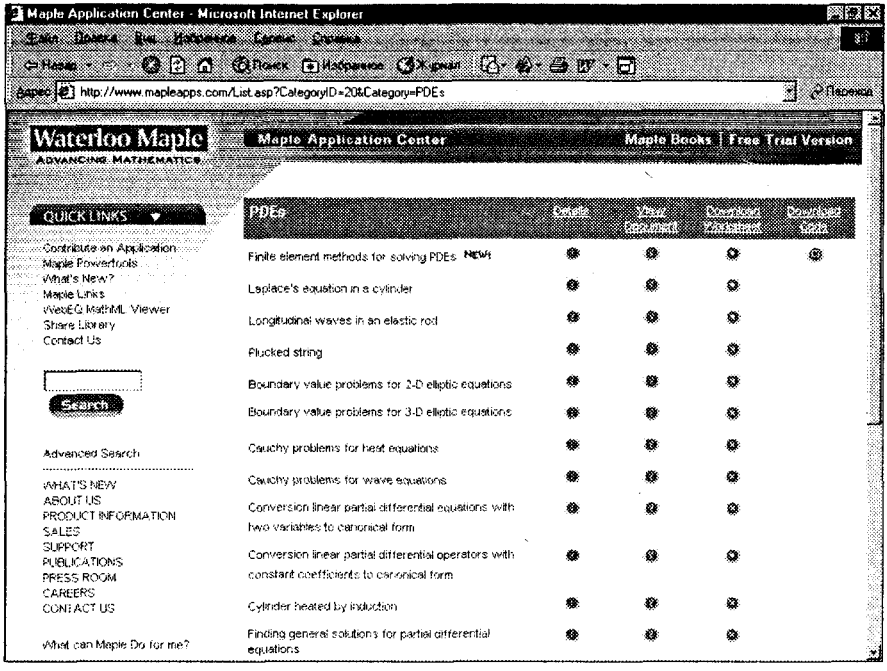


Рис. 2.29. Перечень примеров на решение дифференциальных уравнений

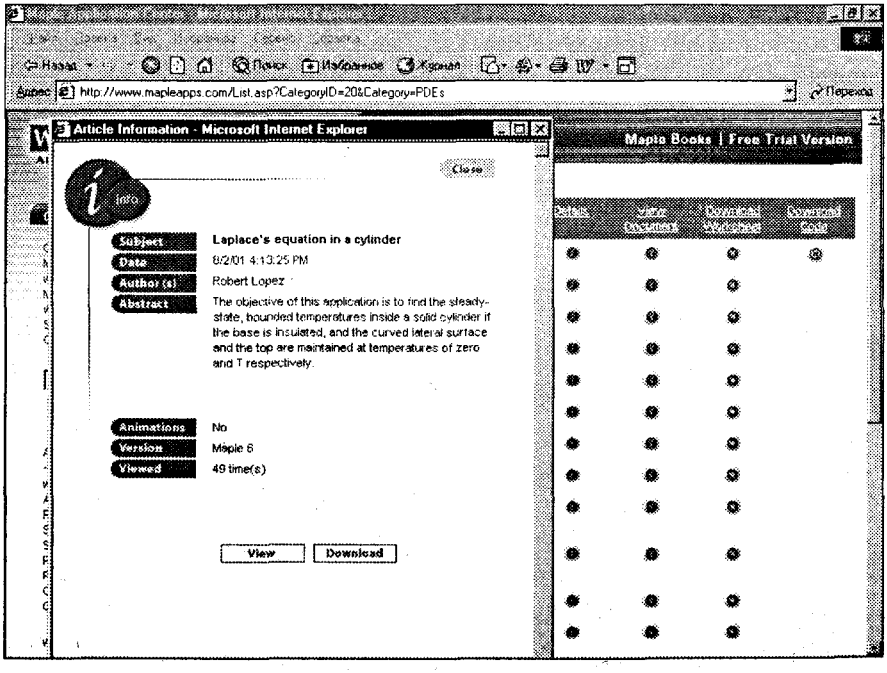


Рис. 2.30. Информация об одном из примеров

менении средств анимации и др. Кнопки View и Download позволяют начать просмотр или загрузку примера в среду Maple прямо из этого окна.

Просмотр примеров с помощью броузера

Просмотр примеров возможен прямо из Интернета с помощью браузера. Желательно подключить к нему plug-in (дополнительный модуль) языка Java и MathML. Рисунок 2.31 демонстрирует просмотр выбранного примера в том его месте, где имеются математические формулы. Нетрудно заметить, что прекрасно отображаются даже самые сложные формулы.

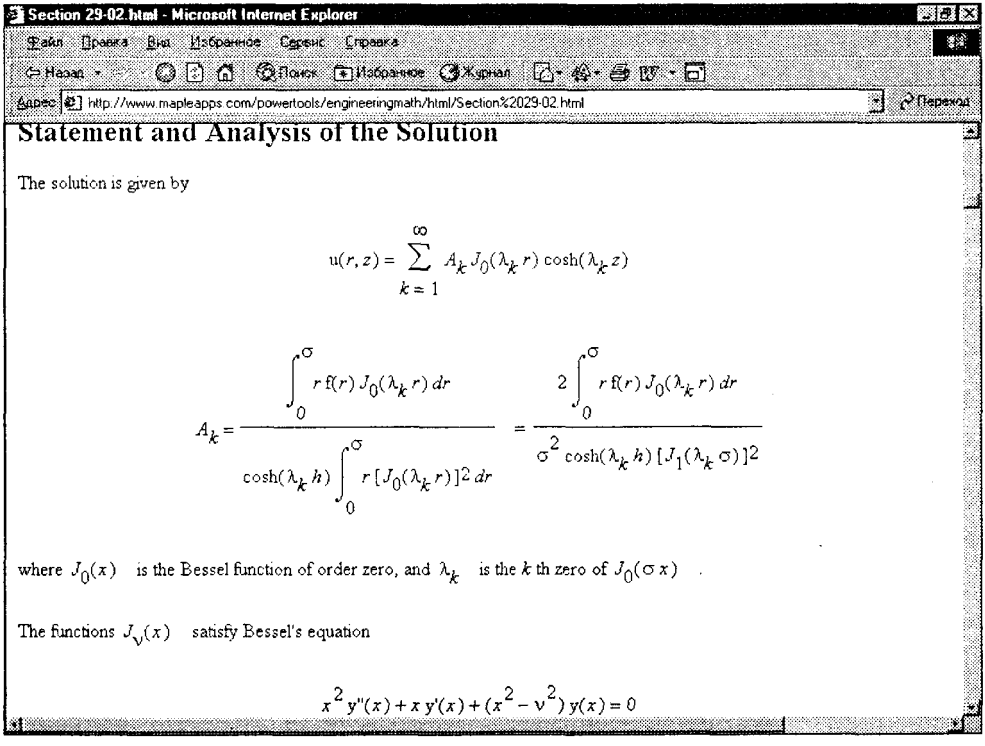
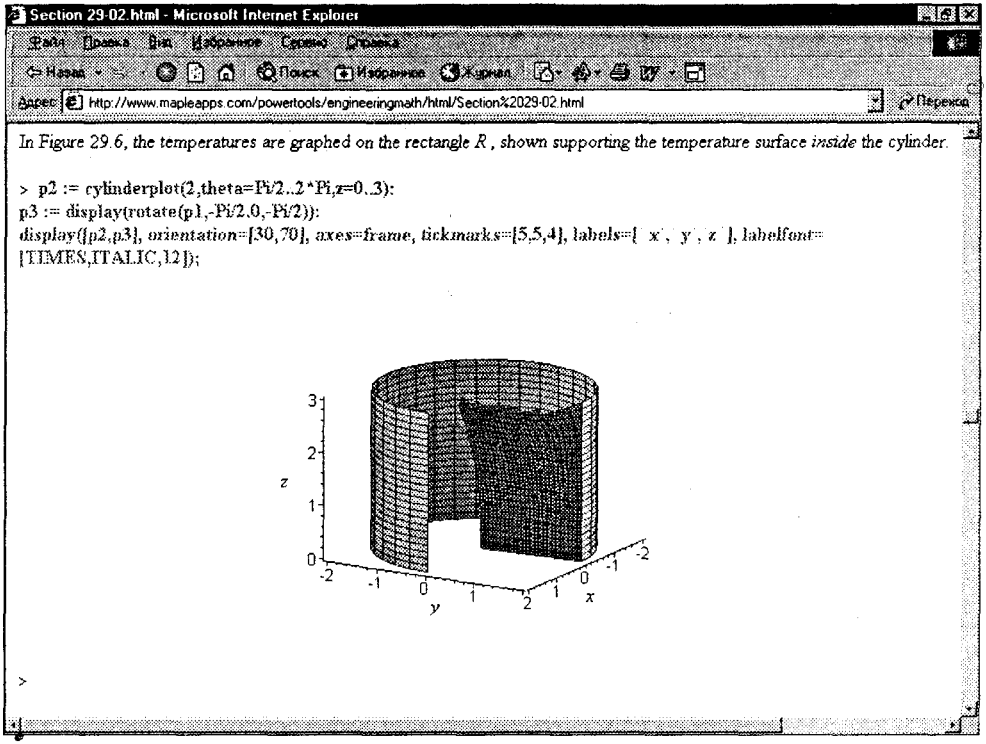


Рис. 2.31. Просмотр примера прямо Интернета (часть примера с математическими формулами)

На другом рисунке (рис. 2.32) показан просмотр той части примера, в которой имеется графика. Нетрудно заметить, что как графика, так и строки ввода воспроизводятся достаточно хорошо. Однако надо помнить, что примеры в этом случае можно лишь просматривать, но их нельзя корректировать и исполнять. Таким образом, просмотр имеет чисто ознакомительное значение.



Загрузка примеров на диск

Чтобы полноценно работать с примерами, нужно загрузить их в среду Maple. Это можно сделать двумя способами:

- скачать пример на жесткий диск и затем открыть его в Maple (с помощью команды Open);
- загрузить пример прямо из Интернета в среду Maple без предварительной записи на диск.

Подготовку к записи примера на диск иллюстрирует рис. 2.33. Окно загрузки файла, представленное на нем, появляется при щелчке на гиперссылке с именем примера.

- Переключатель в этом окне имеет два положения:
- Открыть этот файл из текущего места;

Сохранить этот файл на диске.

После установки его во второе положение и щелчка на кнопке OK появится окно с запросом папки, в которой будет помещен файл выбранного примера. Это окно представлено на рис. 2.34. В данном случае файл будет записываться в папку с именем Maple, внутри которой есть папка с именем 1.

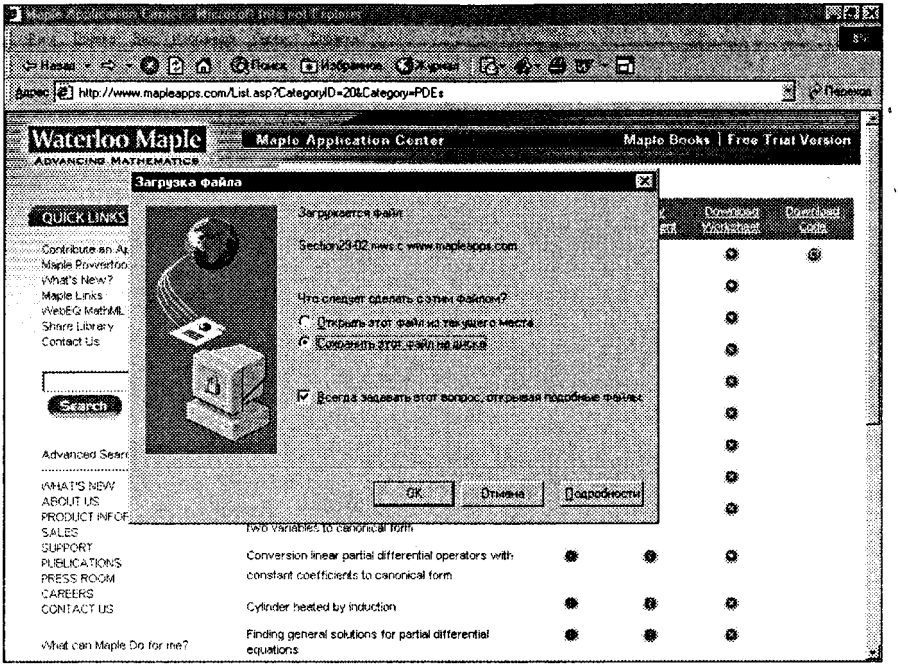


Рис. 2.33. Подготовка к загрузке примера на жесткий диск

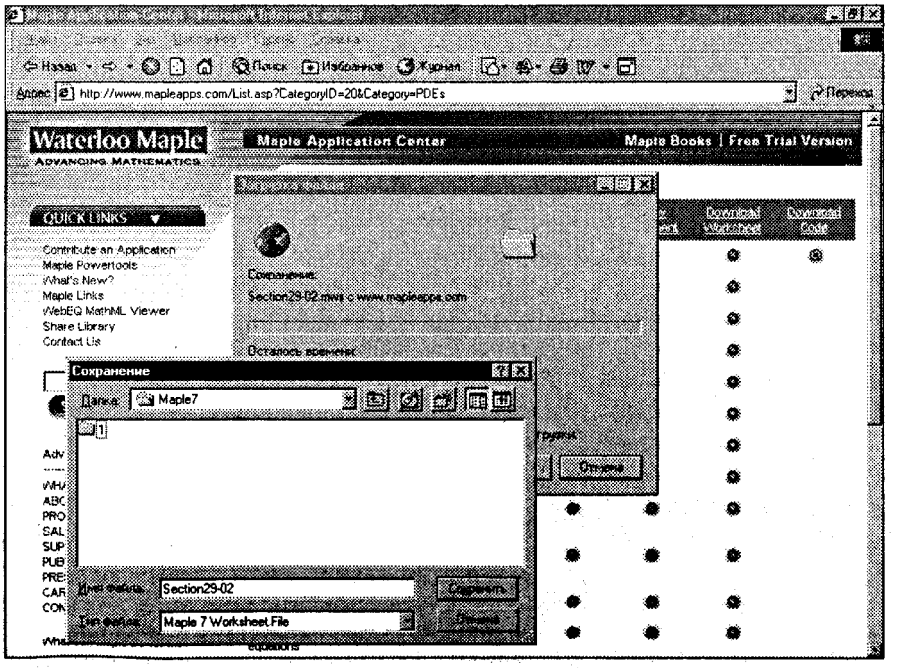


Рис. 2.34. Выбор папки для сохранения примера

После выбора папки достаточно нажать кнопку Сохранить и можно будет наблюдать процесс копирования файла в выбранную папку (рис. 2.35). На экран будет выведено окно загрузки с линейным индикатором хода процесса. В этом окне приводятся данные о скорости загрузки, имени файла и пути к папке, в которую загружается файл. Окно исчезает после завершения загрузки, если установлен флажок Закрыть диалоговое окно после загрузки.

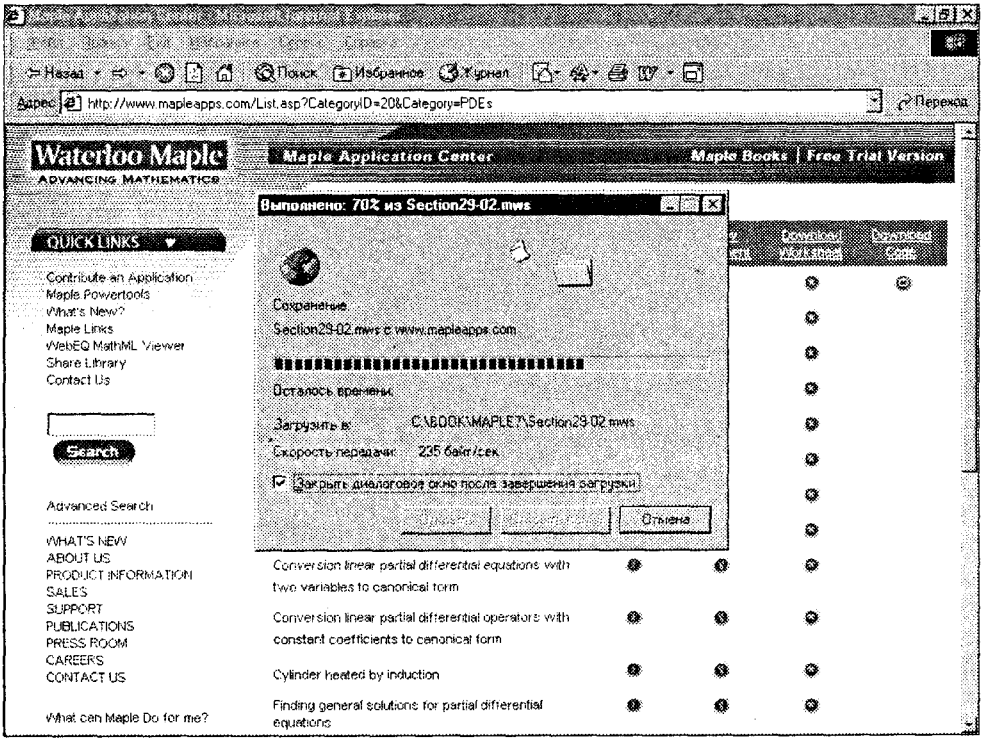


Рис. 2.35. Процесс загрузки примера из Интернета на жесткий диск

Просмотр примеров в среде Maple

После загрузки примера на диск его можно открыть в Maple и запустить на исполнение. При этом пример можно модифицировать, исследуя, как скажутся изменения на его работе. Для открытия файлов служит команда Open.

Однако в Maple 7 предусмотрена возможность и прямой загрузки примеров из Интернета, минуя промежуточную запись файла на диск. Для этого у выбранного примера надо щелкнуть на гиперссылке Download Worksheet. По окончании загрузки примера появляется окно Maple 7 с открытым рабочим листом — рис. 2.36.

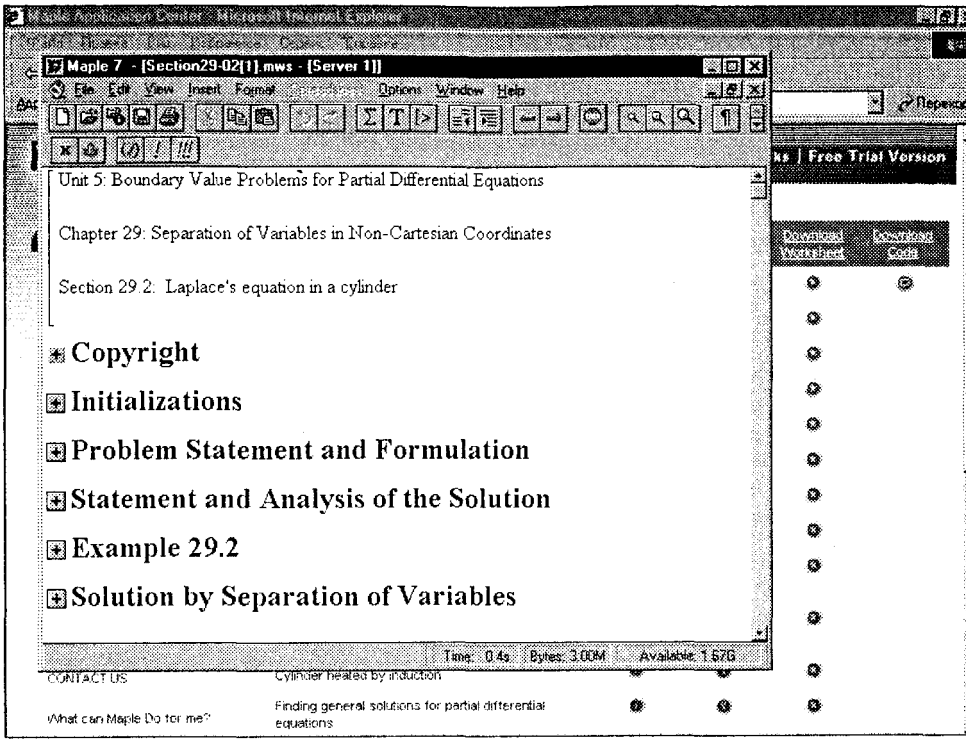


Рис. 2.36. Просмотр примера, напрямую загруженного из Интернета

Новые инструменты Maple Powertools

В конце 2000 г. на сайте фирмы Waterloo Maple появился новый раздел, сразу привлекший внимание пользователей системы Maple, — инструментальный пакет **Maple Powertools**. Это бесплатно распространяемый пакет инструментальных средств, облегчающий использование Maple и дополняющий программу новыми возможностями. Страница сайта, посвященная этому пакету, представлена на рис. 2.37.

На рис. 2.38 показана часть средств пакета Maple Powertools. Здесь видны обучающие компоненты и примеры применений системы Maple. Здесь обращает на себя внимание мощный пакет вычислений Calculus, содержащий три части, и пакет операций с векторами Vector Calculus.

Нижняя часть страницы представлена на рис. 2.39. Здесь помещены пакеты по инженерной математике и матричной алгебре, пакет-презентация по Maple, пакет для математического класса и для продолженного образования. Пакеты, представленные на этой web-странице (рис. 2.28 и 2.29), образуют большое хранилище интересных и поучительных примеров применения и ознакомительных курсов по Maple. Но, увы, написанных на английском языке.

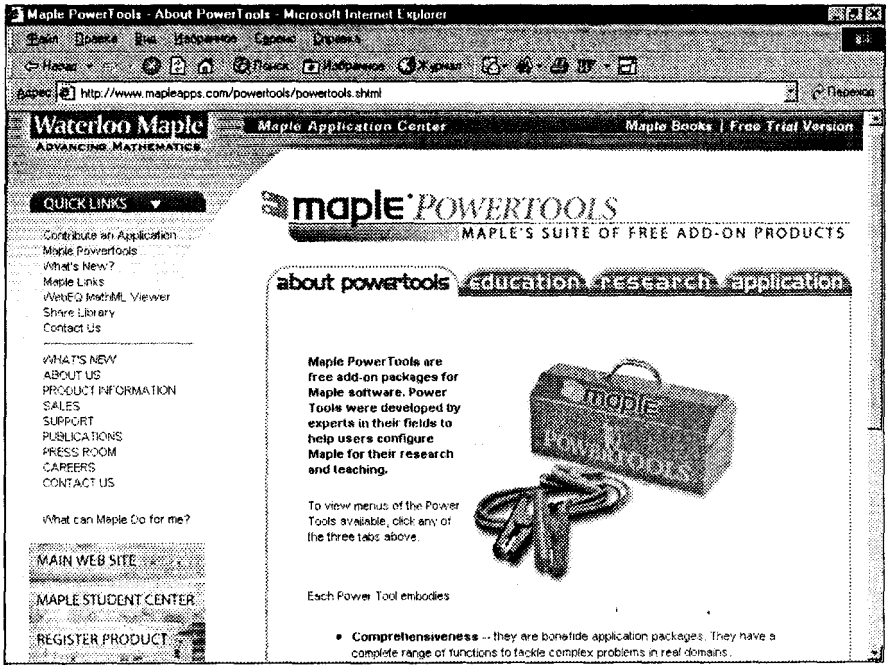


Рис. 2.37. Страница, посвященная инструментальному пакету Maple Powertools

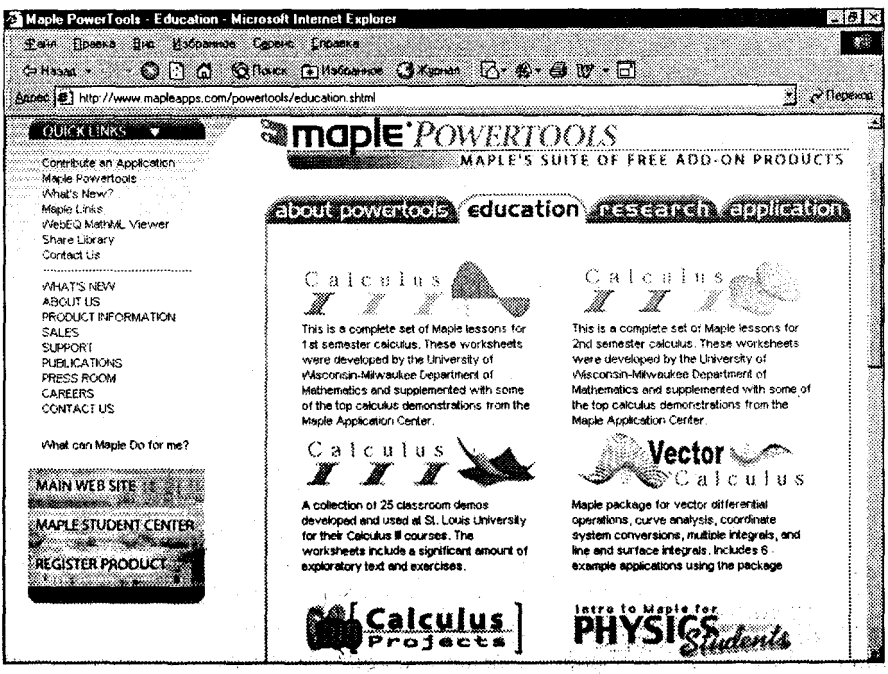


Рис. 2.38. Страница, посвященная образованию

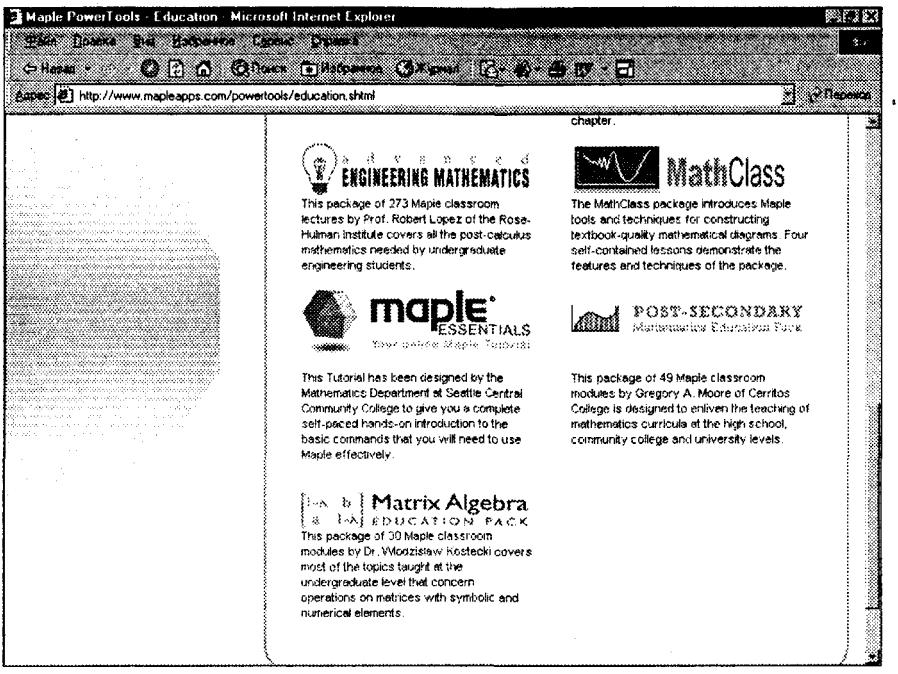


Рис. 2.39. Окончание страница Education

На других вкладках Maple Powertools (Research и Applications) также можно найти подборки примеров. Некоторые повторяют уже рассмотренные. Из новых наборов примеров можно отметить пакеты по конечным элементам Finite Elements, нелинейному программированию Nonlinear Programming, применению статистики Statistics Supplement, динамике Dynaflex и расчету электрических и электронных цепей Syrap.

Студенческий центр

Для доступа в Студенческий центр надо щелкнуть на гиперссылке Maple Student Center на начальной странице сайта корпорации Waterloo Maple (рис. 2.17). Появится начальная страница этого центра, представленная на рис. 2.40.

Щелкнув на гиперссылке Enter в правом нижнем углу страницы (рис. 2.40), можно получить доступ к следующей странице, изображенной на рис. 2.41 и содержащей ссылки на компоненты, находящиеся в Студенческом центре. Наряду с уже упомянутыми пакетами примеров тут можно найти и новые. Список обучающих пакетов в открытом виде представлен на странице рис. 2.41. Выбрав тот или иной пакет, его можно загрузить по описанным выше правилам.

В Студенческом центре представлено особенно много пакетов примеров работы с Maple. И это отражает особое внимание, уделяемое разработчиками этой системы развитию образования.

Рис. 2.40. Начальная страница Студенческого центра

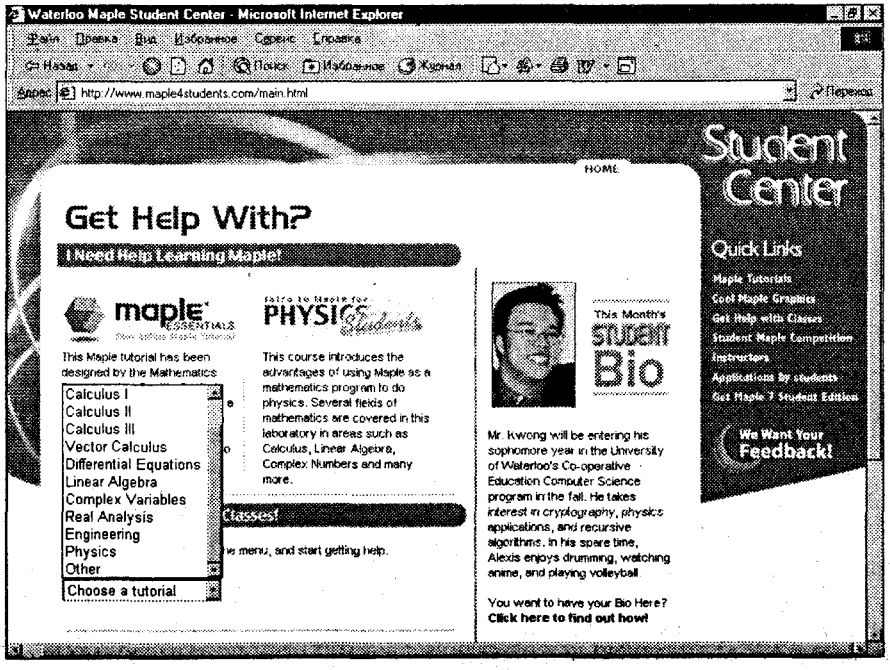


Рис. 2.41. Страница Студенческого центра с перечнем доступных обучающих пакетов

Дополнительные информационные ресурсы в Интернете

Регистрация Maple 7

Политика корпорации Waterloo ориентирована на предельно открытую информационную поддержку системы. Это видно из обзора информационных материалов по системе, представленного выше. Тем не менее для легальных пользователей системы предусмотрены многочисленные блага — рассылка новых материалов по Maple, обновление программы, приглашение на конференции, круглосуточные консультации по телефону и т. д. Но для этого надо зарегистрировать полученный программный продукт.

Регистрация системы Maple 7 выполняется либо по обычной почте, либо по Интернету. В последнем случае достаточно выйти на страницу регистрации, представленную на рис. 2.42.

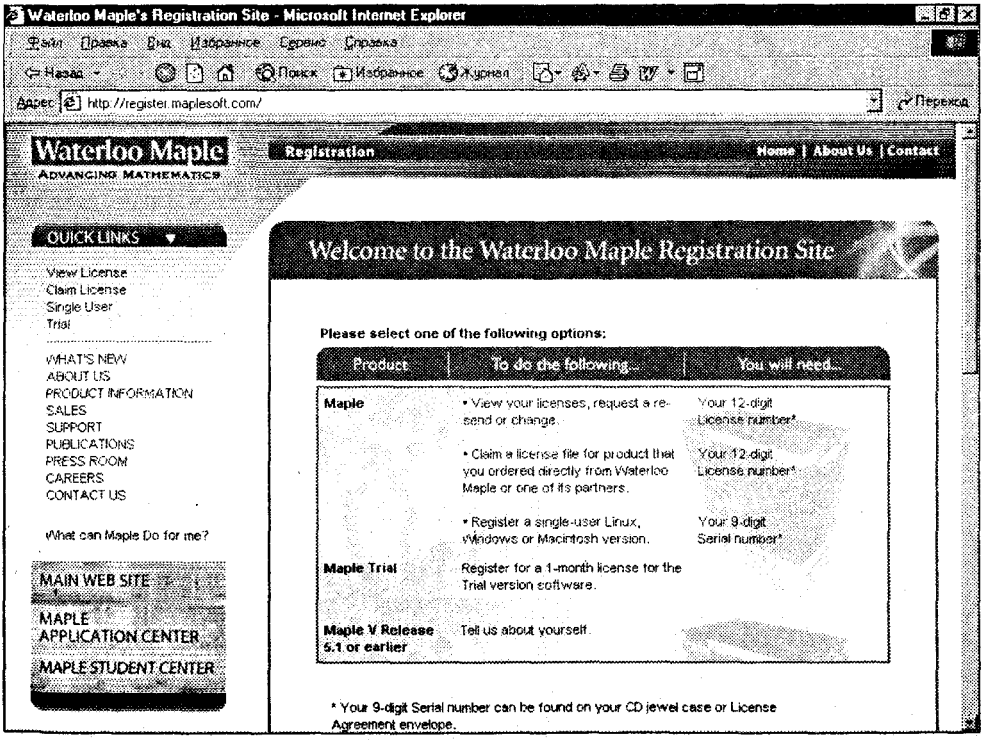


Рис. 2.42. Страница регистрации системы Maple 7

В ходе регистрации надо дать определенные данные о себе (имя, фамилия, обычный адрес, телефоны и др.), сообщить номер лицензии и серийный номер при-

обретенного пакета и указать другие принятые для такого рода операции данные. После регистрации пользователю будет послано по электронной почте подтверждение о ней, и пользователь будет занесен в базу данных корпорации Waterloo Maple.

Контактные адреса корпорации Waterloo Maple

Если возникает необходимость обратиться прямо в корпорацию Waterloo Maple (например, по почте обычной или электронной либо по телефону), то можно получить необходимые контактные данные, просмотрев страницу корпорации, представленную на рис. 2.43.

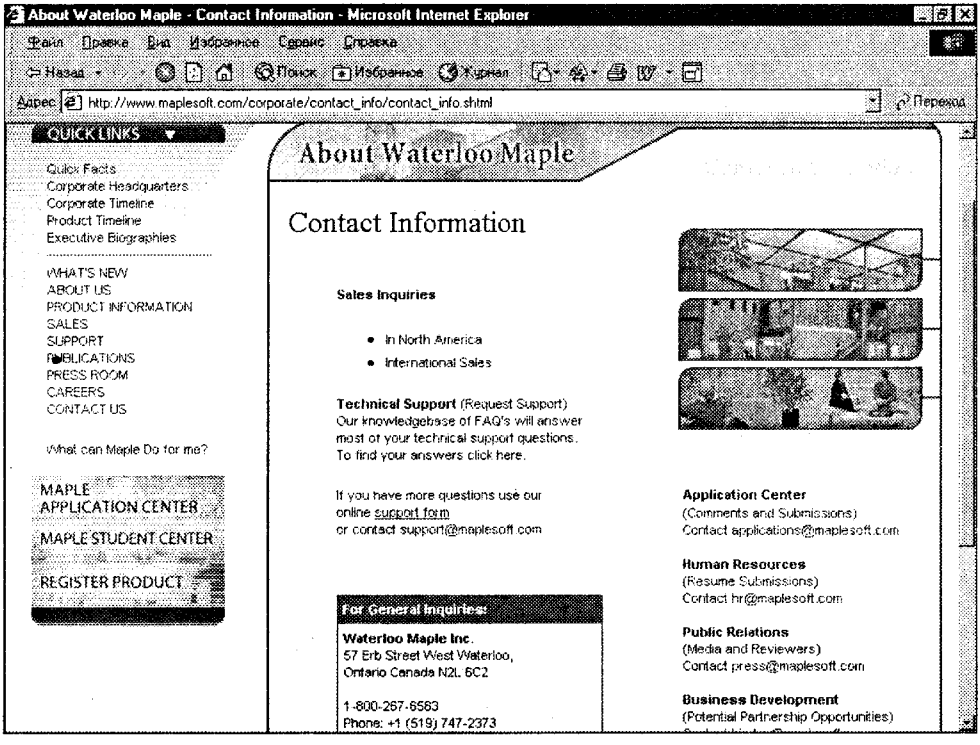


Рис. 2.43. Web-страница корпорации Waterloo Maple с контактными данными

Обзор источников информационной поддержки

Наиболее полный обзор источников информационной поддержки Maple можно получить с web-страницы, представленной на рис. 2.44. Она загружается щелчком на гиперссылке **Maple Links**, присутствующей на многих web-страницах корпорации Waterloo Maple.

С этой страницы можно попасть на страницы с различными источниками информационных ресурсов для Maple и web-сайты, им посвященные. К примеру,

в сентябре 2001 г. на этой странице присутствовала ссылка на новый российский сайт Донецкого государственного университета, посвященный системе Maple (начало ссылки можно заметить внизу страницы на рис. 2.44, если перейти в ее конец с помощью полосы прокрутки).

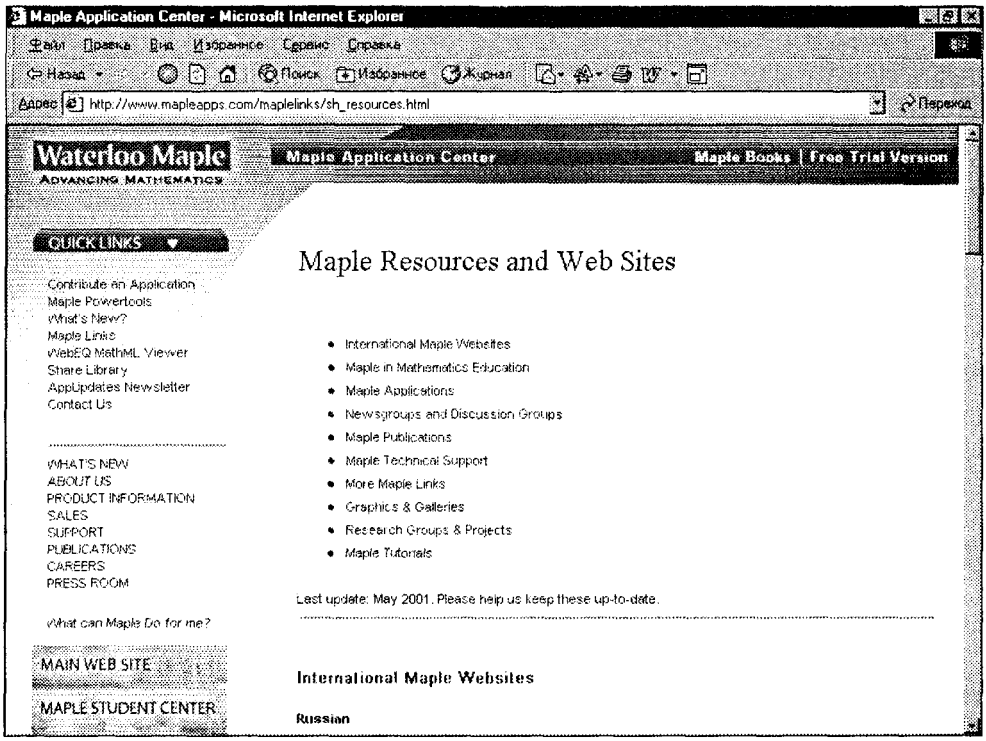


Рис. 2.44. Страница с обзором источников информационной поддержки Maple

Модернизация системы

Хотя Maple 7 — тщательно проработанная система, у нее могут быть отдельные неточности в работе — как говорится, «и на Солнце есть пятна». Эти недоработки устраняются с помощью специальных patch-файлов, которые можно найти в разделе Support сайта фирмы (рис. 2.45). На этой странице необходимо заполнить небольшую анкету, в которой следует указать ваши контактные данные.

Загрузка patch-файлов осуществляется аналогично описанной выше загрузке пакетов-библиотек. Patch-файлы обычно помещают в основную директорию системы Maple и запускают — все необходимые для модернизации системы действия выполняются автоматически. Примерно так же можно загружать файлы библиотек Maple 7. Это позволяет в течение долгого времени поддерживать качество работы программы на самом высоком уровне, не покупая новых версий.

Рис. 2.45. Страница модернизации системы Maple 7

Галерея графики

Даже предшествующие версии системы Maple обладали обширными графическими возможностями. Новая версия Maple 7 имеет просто потрясающие воображение возможности по созданию графических изображений. Для доступа к галереям графики (а их множество как в фирме Waterloo Maple, так и в ряде университетов) надо щелкнуть на гиперссылке Graphics&Galleries на web-странице, изображенной на рис. 2.44. Появится страница со списком галерей графики, на которой можно выбрать подходящую.

Ниже представлена страница галереи графики корпорации Waterloo Maple (рис. 2.46). Она содержит множество красочных картинок цветных графиков, полученных средствами Maple. На рис. 2.46 представлено лишь начало галереи, вам стоит просмотреть эту страницу и дальше.

Важно отметить, что разработчики сайта приводят не только сами графики (их созерцание многим доставляет удовольствие), но и возможность загрузить коды (программы) на языке Maple 7, выполнение которых создает выбранный график. Изучение таких программ позволяет учащимся разобраться в ряде весьма тонких алгоритмов современной машинной графики, описание которых (из-за большого объема) трудно найти в учебниках.

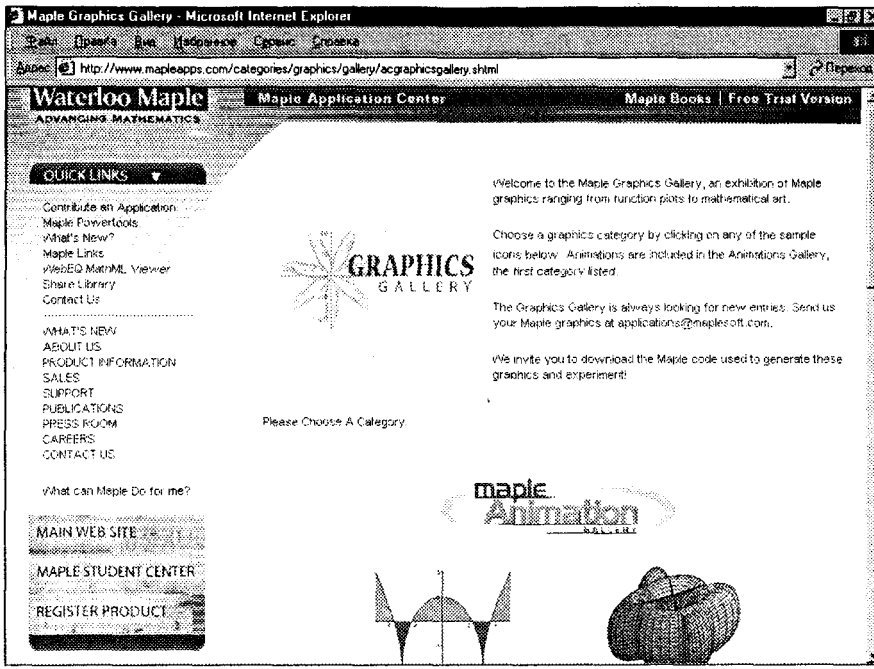


Рис. 2.46. Галерея графики корпорации Waterloo Maple

Библиотека Share Library

В развитии систем класса Maple принимают участие не только специалисты корпорации Waterloo Maple и ее партнеры, но и специалисты со всех стран мира. Это научные работки, преподаватели и студенты университетов и вузов разного профиля и просто любители математики и различных сфер ее приложения. Их усилиями создана уникальная библиотека Share Library — подлинный кладезь интересных примеров применения систем класса Maple.

Доступ к библиотеке открывает гиперссылка Share Library, имеющаяся на многих web-страницах, приведенных выше, — например, на рис. 2.44. Начальная страница библиотеки представлена на рис. 2.47.

Предусмотрены две возможности использования библиотеки: установить ее целиком на жесткий диск или перейти в режим использования только некоторых примеров. Библиотека занимает сотни Мбайт памяти на диске и содержит множество примеров для уже устаревших версий систем Maple. Поэтому переписывать библиотеку целиком едва ли целесообразно. Страница с началом перечня разделов библиотеки (тематический каталог) представлена на рис. 2.48.

Помимо показанных на странице рис. 2.48 разделов по применению Maple и по математическим расчетам имеются и другие разделы, в том числе по примерам в области образования.

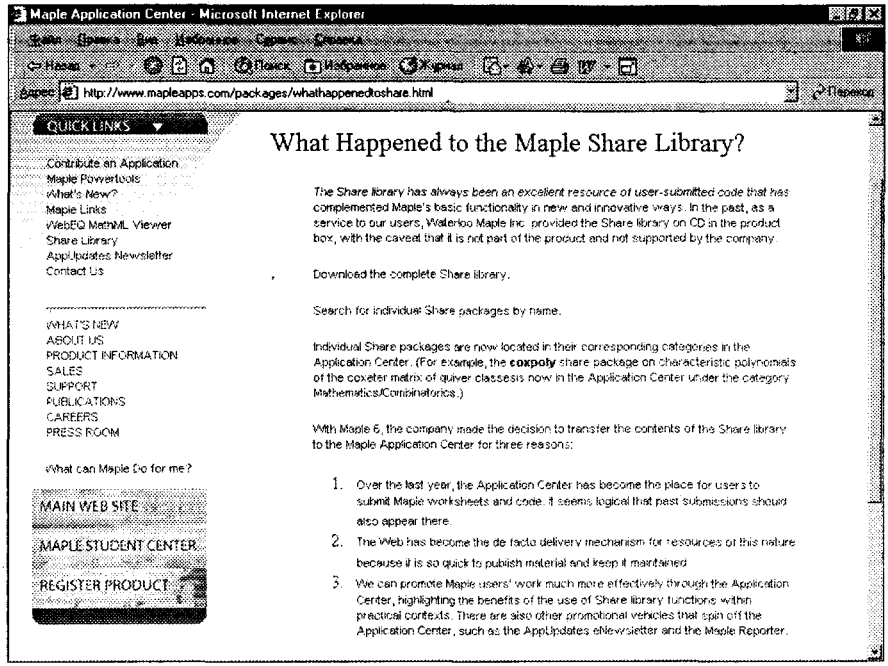


Рис. 2.47. Начальная страница библиотеки Share Library

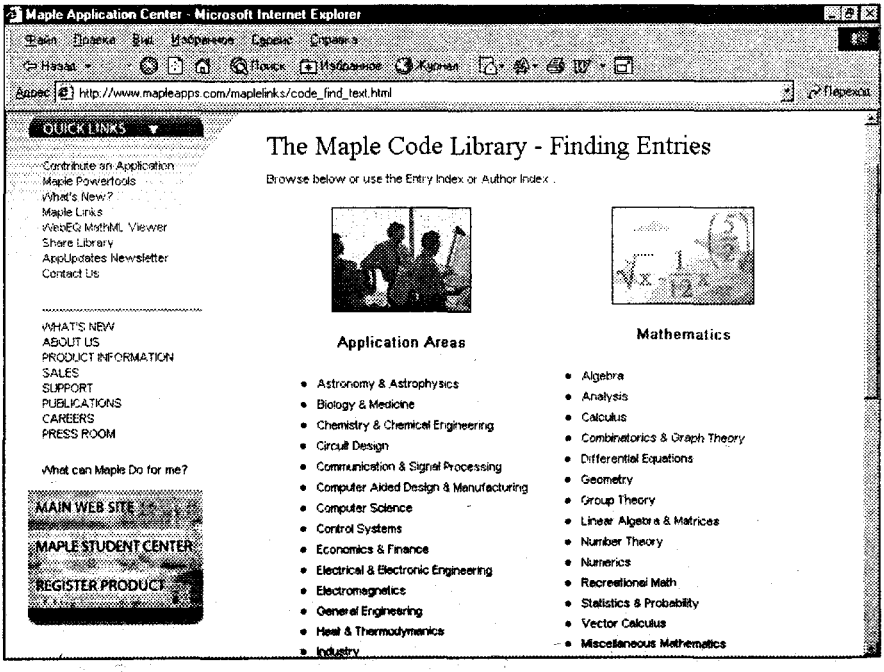


Рис. 2.48. Страница с тематическим каталогом библиотеки Share Library

Поддержка MathML 2.0

Выход на web-страницу поддержки MathML

MathML — это специальный язык представления математической информации. Основное достоинство данного языка — то, что он является мировым стандартом и поддерживается большинством разработчиков математических систем (Maple, Mathcad, Mathematica и др.) и браузеров. В результате появляется возможность записи сложных математических выражений и графиков в очень небольшом по объему текстовом формате. По сути, он похож на язык XML, адаптированный для представления математической информации.

Корпорация Waterloo Maple дает возможность подключить к браузерам и Maple 7 средства поддержки языка MathML. Для этого достаточно щелкнуть на гиперссылке WebEQ MathML Viewer (см. рис. 2.44, например). Откроется специальная страница с подробным описанием возможностей MathML (рис. 2.49).

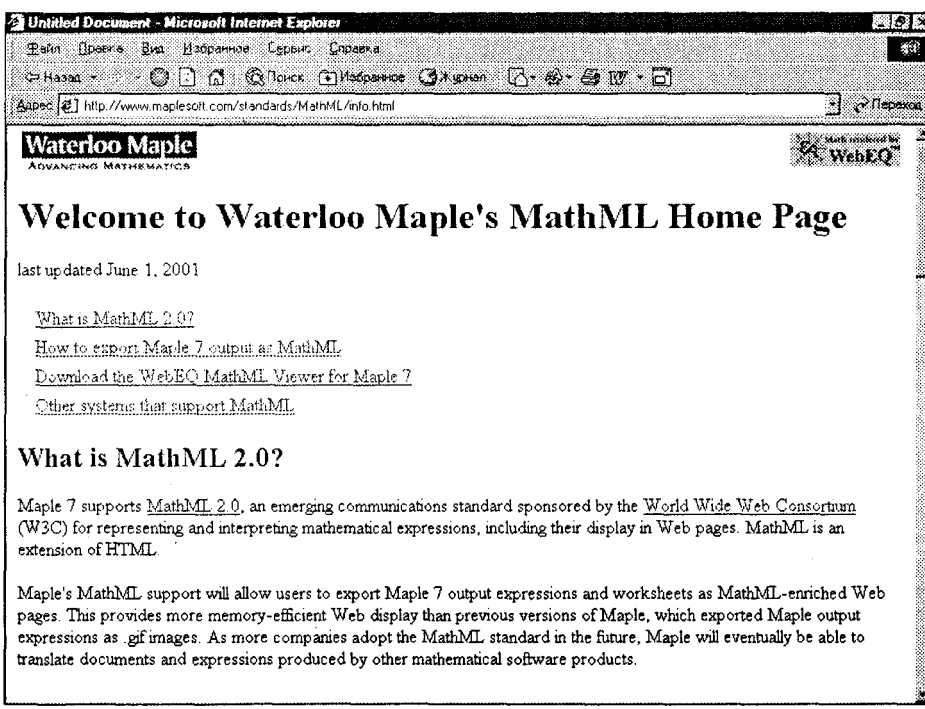


Рис. 2.49. Страница поддержки языка MathML

MathML лишь начинает внедряться, поэтому мы ограничимся здесь лишь описанием средств поддержки этого языка. С помощью страницы, представленной на рис. 2.48, можно ознакомиться с MathML более подробно.

Загрузка средств поддержки MathML

Для загрузки средств поддержки MathML достаточно щелкнуть на гиперссылке Download WebEQ MathML Viewer for Maple 7, которая видна на рис. 2.49. Появится начальная страница Мастера установки WebEQ MathML Viewer, показанная на рис. 2.50.

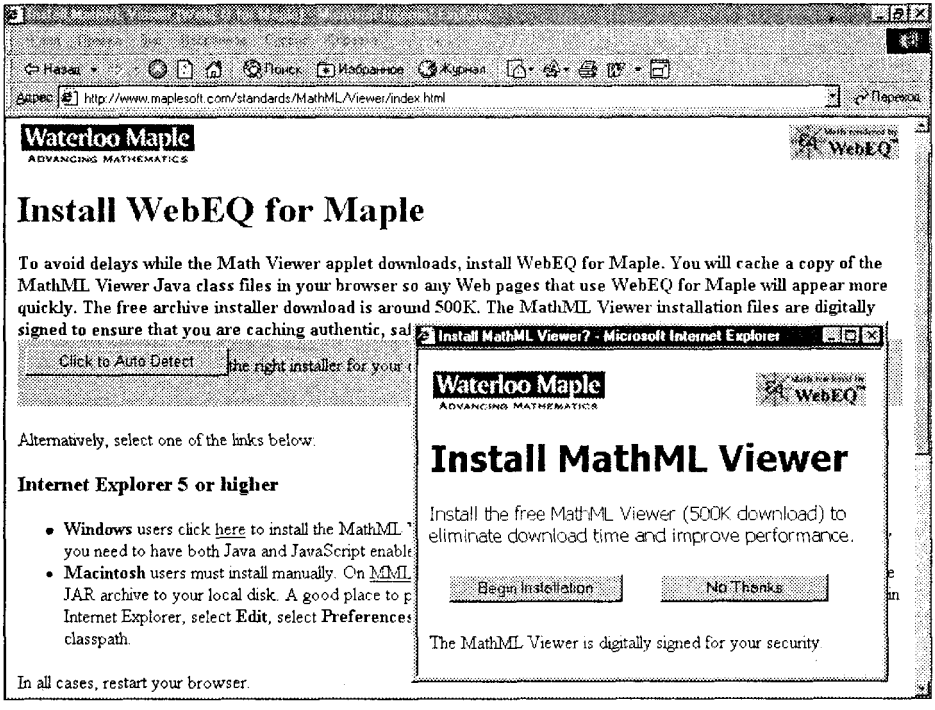


Рис. 2.50. Страница Мастера установки WebEQ MathML Viewer

Обратите внимание на кнопку Click to Autodetect, которая позволяет проверить, не установлена ли уже на вашем компьютере поддержка MathML. Это весьма маловероятно, но стоит проверить — как говорится, «чем черт не шутит». Если такой поддержки нет, то появится окно установки, показанное на рис. 2.50 справа, и вы сможете начать установку MathML Viewer, нажав кнопку Begin Installation, или отказаться от нее, щелкнув на No Thanks.

Если вы рискнули начать установку модулей поддержки MathML (это стоит сделать, если вы всерьез намерены передавать математическую информацию через Интернет), то нажмите кнопку Begin Installation. Через некоторое время (до нескольких минут) появится окно с предупреждением системы безопасности (рис. 2.51).

Нажав кнопку Да, вы подтвердите свое доверие к корпорации Waterloo Maple и начнете процесс установки. Ход процесса установки отражается на линейном индикаторе в окне Мастера, что показано на рис. 2.52.

После завершения процесса установки окошко с линейным индикатором сменится информационным окошком с сообщением о завершении установки (рис. 2.53).

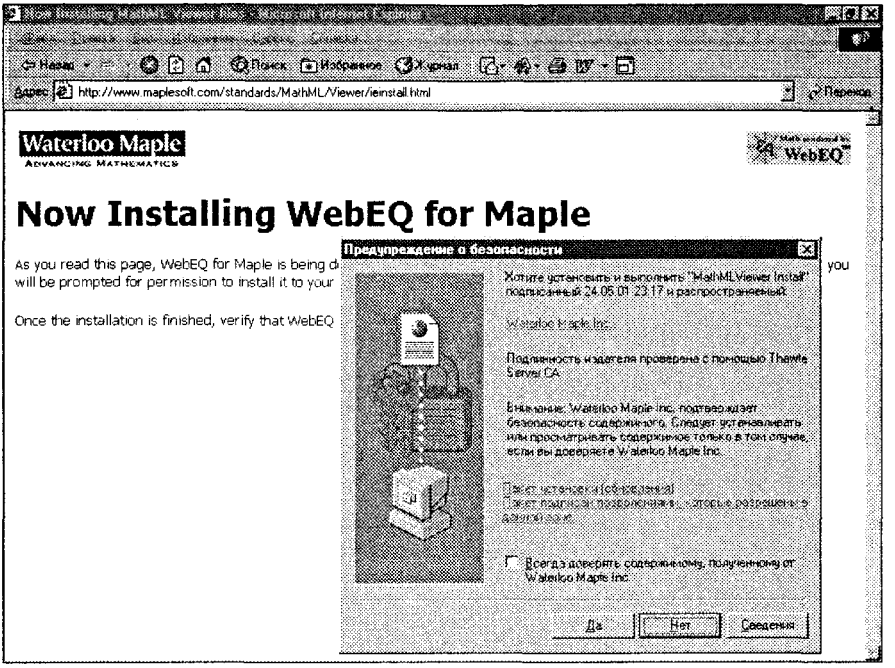


Рис. 2.51. Предупреждение системы безопасности о начале установки MathML Viewer

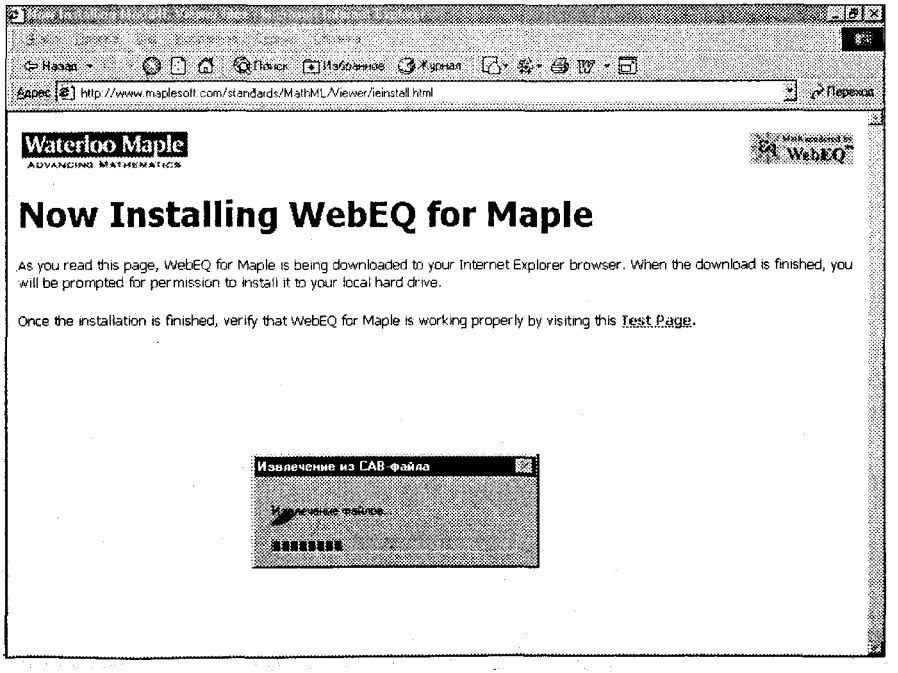


Рис. 2.52. Процесс инсталляции MathML Viewer

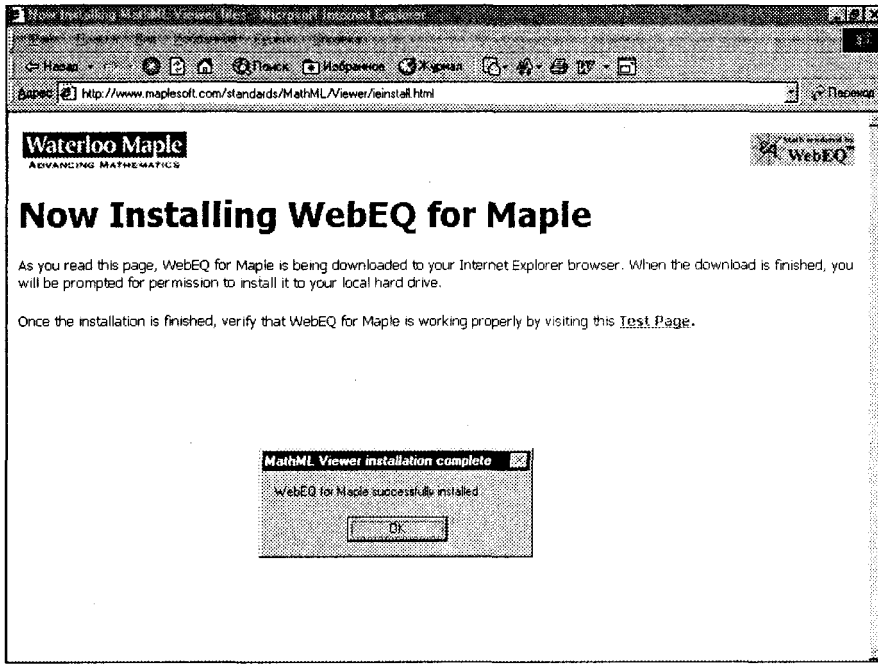


Рис. 2.53. Завершение установки MathML Viewer

Тестирование MathML Viewer

После завершения установки рекомендуется завершить работу браузера и даже перезапустить компьютер (у автора без этого MathML Viewer не заработал). После этого надо вернуться на страницу поддержки средств MathML и провести тестирование, щелкнув на гиперссылке Test Page. Если все в порядке, то в конце этой короткой страницы появится математическая формула, представленная на рис. 2.54.

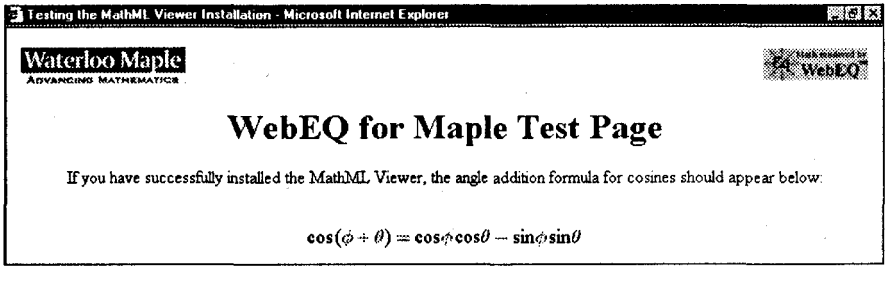


Рис. 2.54. Страница тестирования MathML Viewer

Если установка прошла неудачно, то вместо контрольной формулы будет виден серый прямоугольник. В этом случае надо проанализировать причины неудачи и попробовать повторить установку снова. Надо прямо сказать, что большинство

пользователей не слишком много потеряют, если откажутся от установки MathML Viewer или потерпят при ней неудачу.

Использование средств MathML

Детальное описание использования средств MathML выходит за рамки данной книги. Поэтому ограничимся ссылкой на страницу How to Export Maple 7 output to MathML (Как экспортировать документы Maple 7 в формате MathML). Начало этой страницы представлено на рис. 2.55.

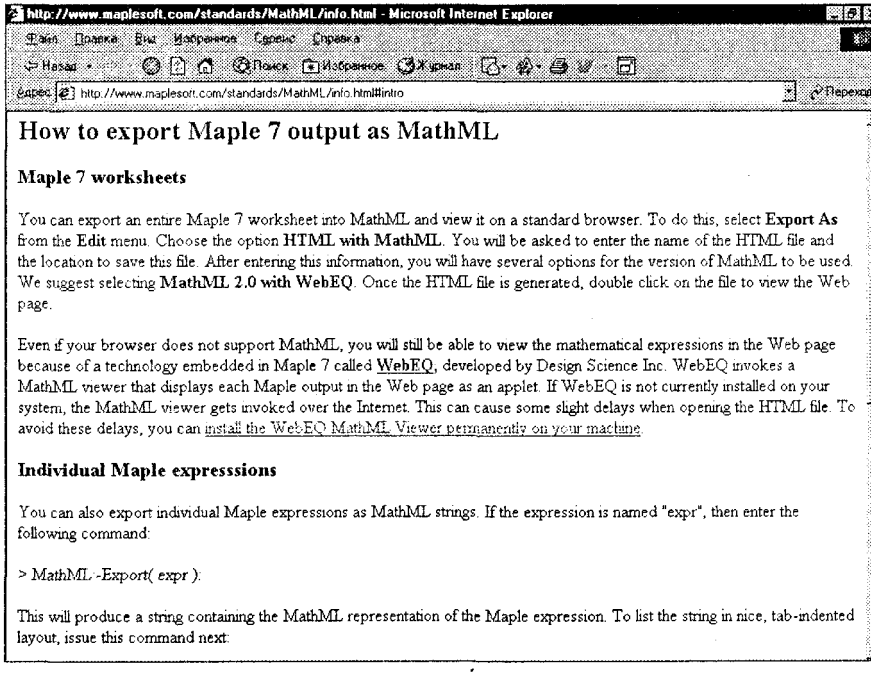


Рис. 2.55. Страница с описанием экспорта в формат MathML

На этой странице дан пример того, как обеспечить экспорт документов Maple 7 в формате MathML. В уроке 16 вы найдете описание пакета MathML, в котором содержатся дополнительные средства поддержки MathML.

Maple на российских сайтах

Maple на сайте exponenta.ru

В последнее время появилось множество сайтов, посвященных системам компьютерной математики, и в частности Maple. К сожалению, большинство из них не

русскоязычные. Но есть и сайты, ориентированные на наших пользователей. Один из них — www.exponenta.ru.

Этот сайт посвящен применению систем компьютерной математики в образовании. На нем есть достаточно большой раздел, посвященный Maple (рис. 2.56).

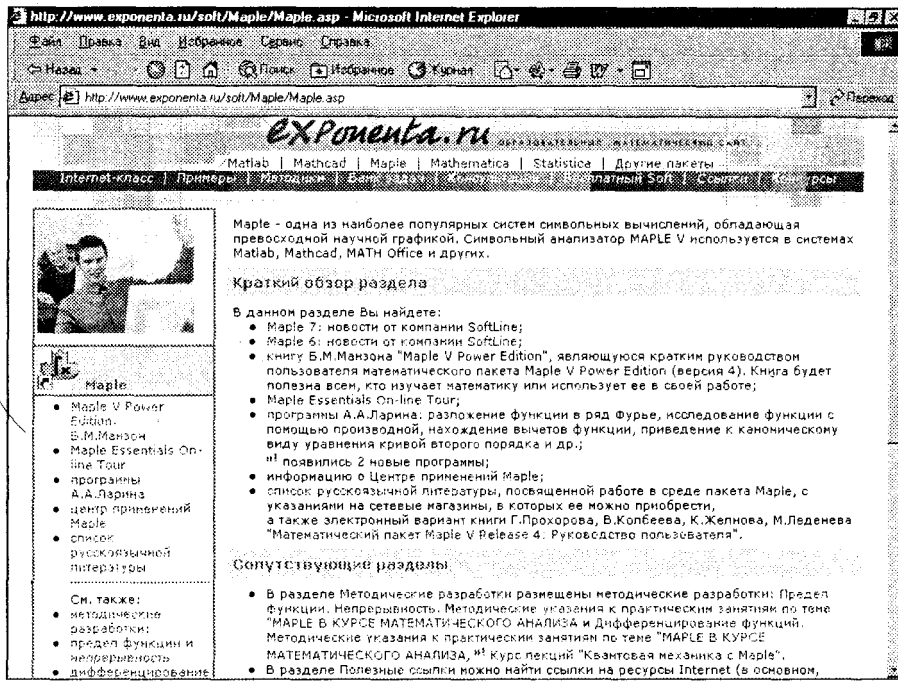


Рис. 2.56. Начальная страница сайта www.exponenta.ru

На этой странице, в частности, имеется подборка литературы по данным системам, включая учебные пособия и примеры применения (рис. 2.57). Нетрудно заметить, что список этой литературы достаточно велик, что свидетельствует об интересе читателей к этому программному продукту.

Среди упомянутой литературы можно найти и упоминание книги автора, посвященной предшествующей версии системы — Maple 6 (рис. 2.58). Эта книга издана издательством «Питер» в 2001 г.

Российский сайт Донецкого университета

Прекрасно оформленный сайт, посвященный Maple, создан Донецким университетом. Его начальная страница по стилю копирует начальную страницу корпорации Waterloo Maple, но с русскоязычными гиперссылками (рис. 2.59).

Этот сайт дает достаточно полное представление об информационной поддержке Maple. Это видно уже из главной страницы, посвященной описанию Центра применений систем Maple (рис. 2.60). На ней имеются ссылки на десятки разделов, посвященных таким применениям.

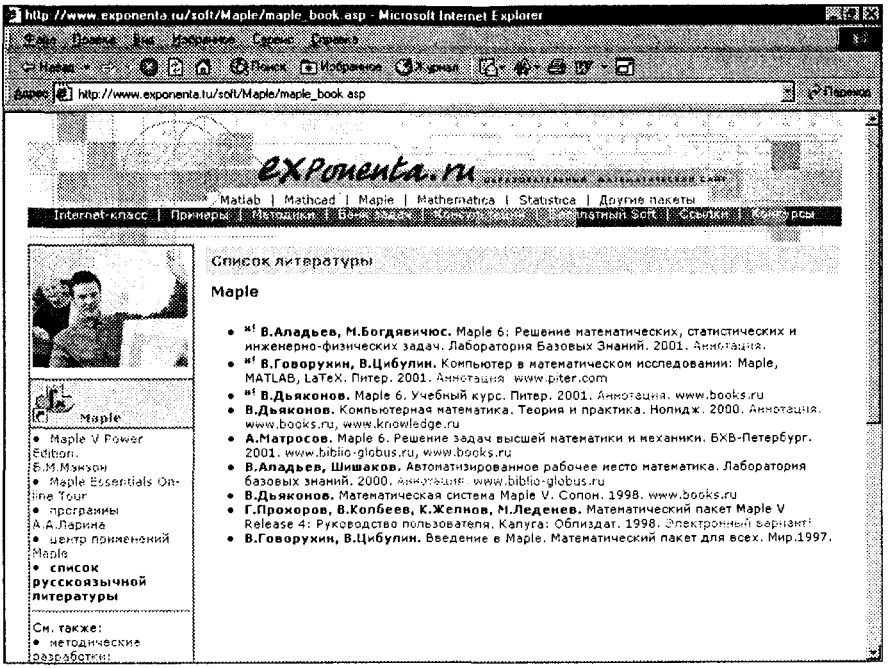


Рис. 2.57. Раздел сайта www.exponenta.ru, посвященный русскоязычной литературе по системам Maple

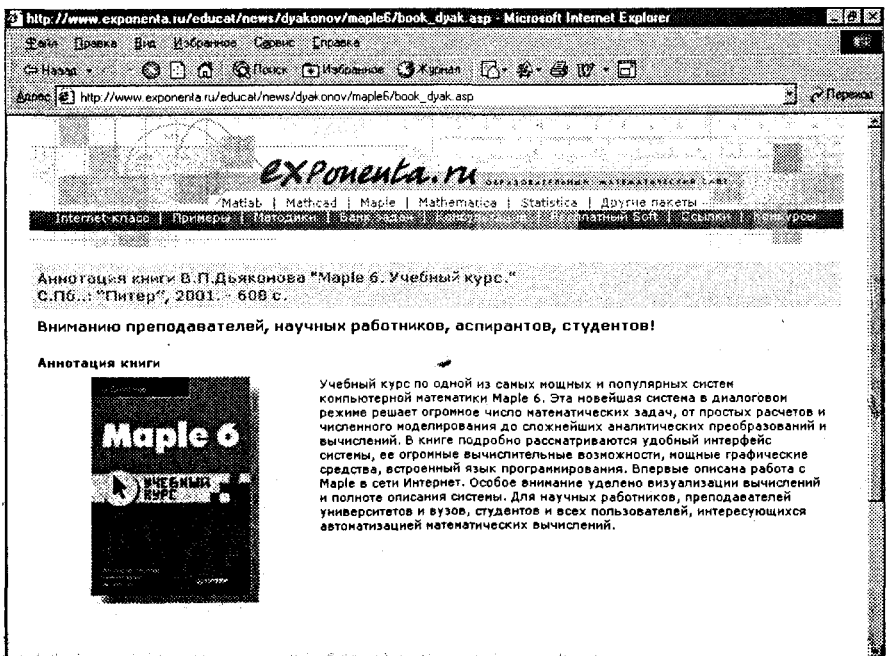


Рис. 2.58. Книга по системе Maple 6 на сайте www.exponenta.ru

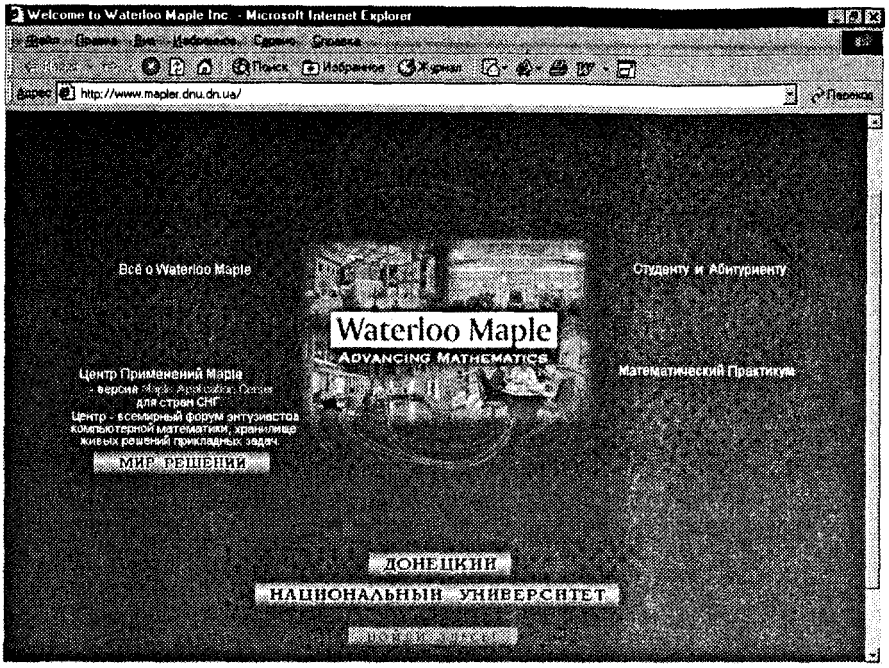


Рис. 2.59. Начальная страница сайта Донецкого госуниверситета, посвященная системам Maple

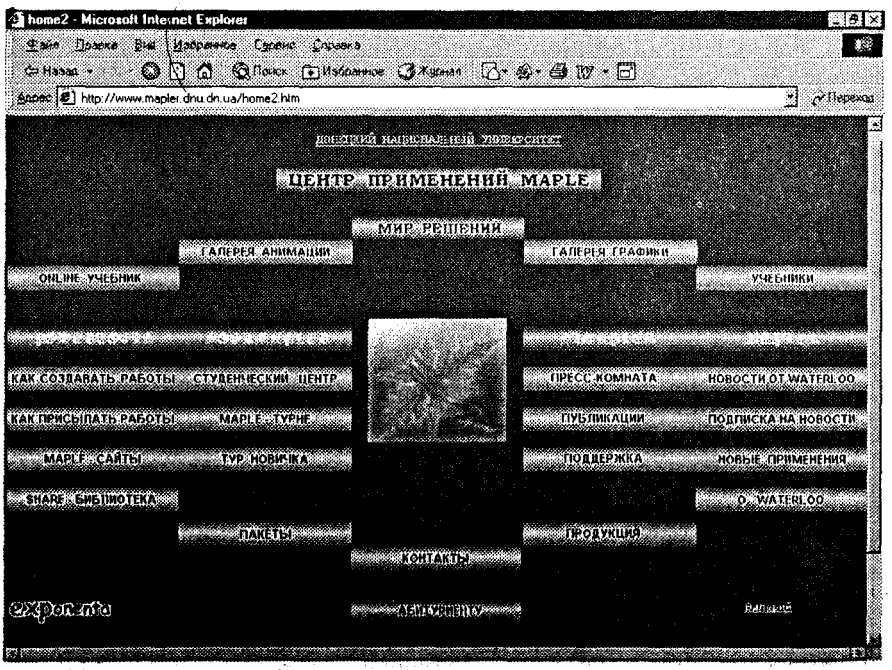


Рис. 2.60. Главная страница сайта Донецкого госуниверситета, посвященная Центру применений систем Maple

Объективности ради надо отметить, что ссылка на многие разделы попросту отправляет пользователя к просмотру соответствующих разделов сайта корпорации Waterloo Maple. Но есть немало разделов, содержащих интересную и важную информацию по системам Maple и их применению. Важно, что создатели этого сайта не ограничились описанием только работ Донецкого госуниверситета, но и дали объективную информацию по применению систем класса Maple как за рубежом, так и в России. В качестве примера можно привести отражение на этом сайте книг по системам Maple, опубликованным на русском языке (рис. 2.61).

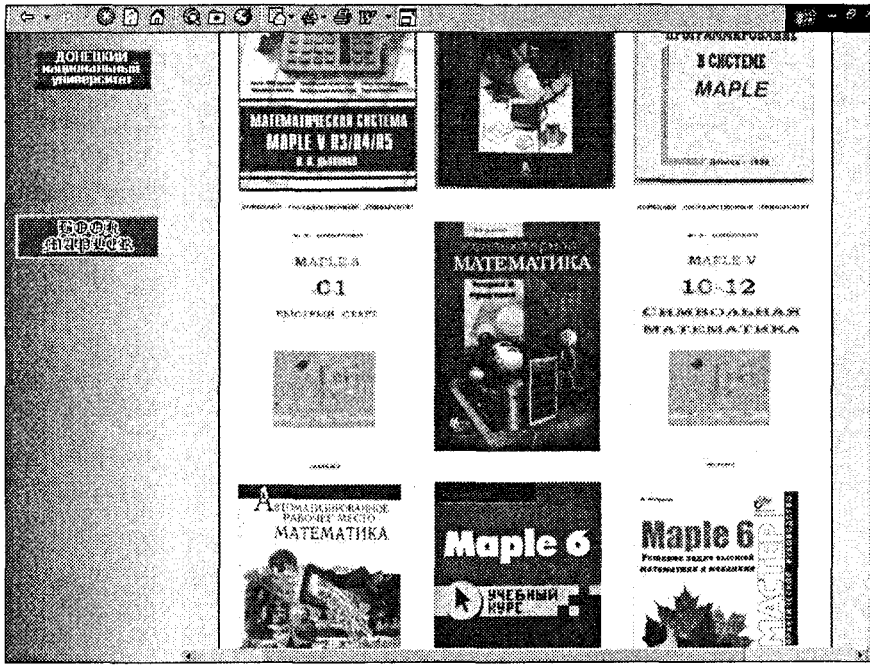


Рис. 2.61. Страница сайта Донецкого госуниверситета, посвященная русскоязычным книгам по Maple

Maple в карманном компьютере

Maple — довольно громоздкие программы, хотя им далеко до MATLAB, уже занимающих на жестком диске компьютера объем около 1–1,5 Гбайт. Поэтому особый интерес вызвало сообщение о том, что несколько урезанная версия системы Maple размещена в ПЗУ карманного компьютера Cassiopeia A22T известной своими миниатюрными изделиями фирмы Casio (рис. 2.62).

Этот помещающийся на ладони компьютер работает под управлением операционной системы Windows CE. Благодаря размещенной в ПЗУ программе Maple компьютер может выполнять сложные математические расчеты с удобствами, которым позавидует любой пользователь обычного компьютера. Ведь эту малышку можно взять и на дачу, и на пляж, не беспокоясь о сети электропитания и даже о зарядке аккумуляторных батарей — питается компьютер от обычных миниатюр-

ных гальванических элементов, причем одной их зарядки хватает на многие десятки часов работы.

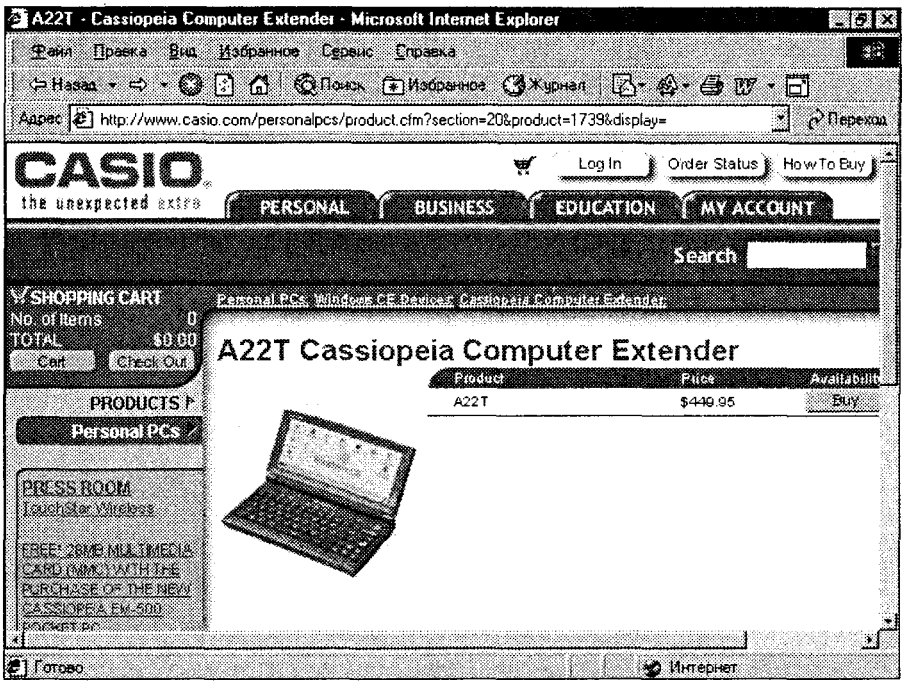


Рис. 2.62. Web-страница фирмы Casio с рекламой карманного компьютера Cassiopeia A22T со встроенной системой Maple

Вид экрана компьютера при работе с системой Maple представлен на рис. 2.63. Нетрудно заметить, что, как и в случае с «большими» «братьями» Maple, работа происходит с использованием традиционных для Windows окон и выпадающих меню. Естественно, что, учитывая малые размеры экрана у миниатюрного компьютера, полного совпадения в интерфейсе пользователя все же нет.

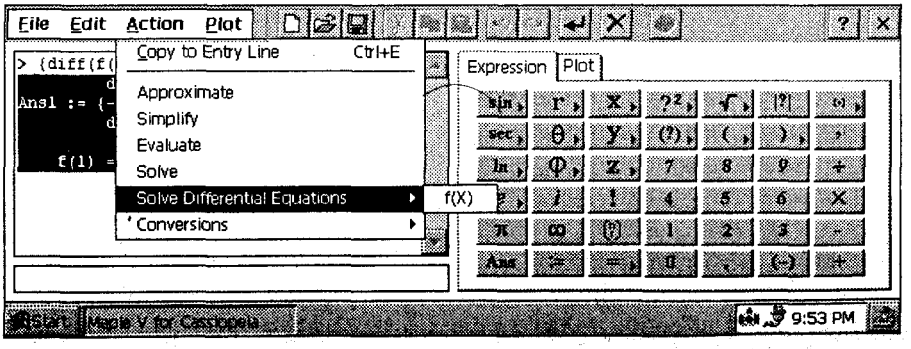


Рис. 2.63. Вид экрана компьютера Cassiopeia A22T при работе с системой Maple

Рисунок 2.64 показывает, как этот компьютер лихо справляется с аналитическим решением дифференциального уравнения. Увы, его экран настолько мал, что на нем не помещается само уравнение — видно лишь его решение. Это служит, пожалуй, главным утешением пользователям обычных компьютеров — на больших экранах их мониторов при работе с Maple можно наблюдать куда больше информации и в цвете.

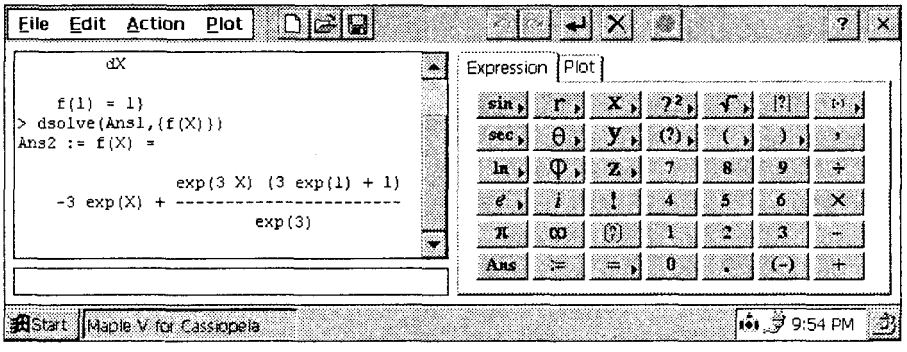


Рис. 2.64. Пример работы с аналитическими вычислениями

Есть у этого компьютера и средства для работы с графикой — увы, тоже не цветной. Рисунок 2.65 показывает построение трехмерного графика функции двух переменных.

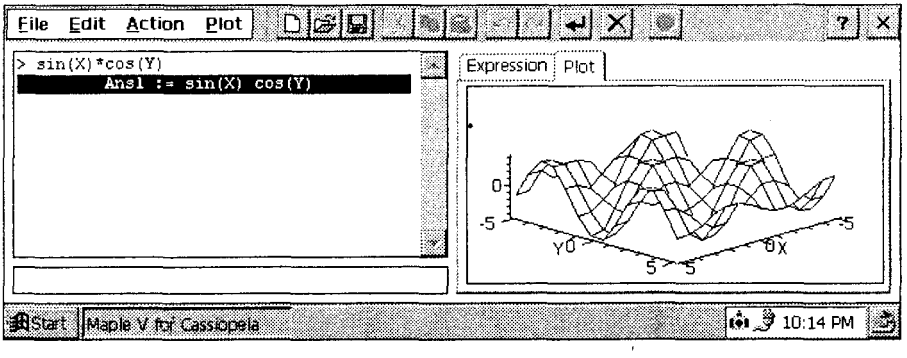


Рис. 2.65. Пример построения графика функции двух переменных

Самым удивительным является стоимость такого компьютера. Она (правда, не для наших студентов и преподавателей вузов) сравнительно невелика и составляет около $450 (см. рис. 2.62). Несомненно, что такой компьютер — прекрасный помощник студенту, преподавателю и научному работнику, всерьез занимающемуся повседневными вычислениями. Полученные при этом результаты можно перенести на обычный компьютер, что снимает некоторые ограничения на объем и сложность решаемых задач, обусловленные меньшей памятью миниатюрного компьютера. Остается лишь отметить, что в какой-то степени альтернативой (увы,

заметно более дорогой) является установка вполне полноценной версии системы Maple на портативный компьютер класса notebook. Это может быть и новейшая версия Maple 7, описанная в данной книге.

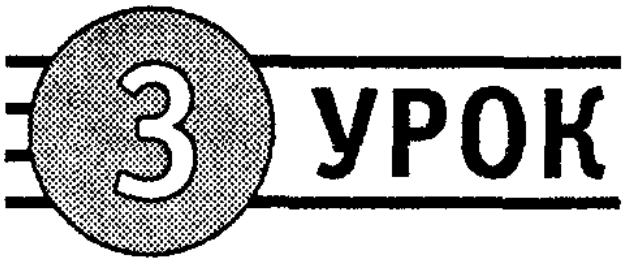
Что нового мы узнали?

В этом уроке мы научились:

- Использовать контекстную справку.
- Работать со справочной системой.
- Входить в Интернет.
- Загружать из Интернета средства поддержки MathML.

А также познакомились с:

- Web-сайтом фирмы Waterloo Maple Software.
- Информационными ресурсами Интернета.
- Российскими сайтами, посвященными Maple.



Работа с файлами и документами

- ❒ Операции с файлами
- ❒ Печать документов
- ❒ Редактирование документов
- ❒ Операции вставки
- ❒ Операции форматирования
- ❒ Операция внедрения в секцию
- ❒ Операция выведения из секции
- ❒ Работа с объектами

Операции с файлами

Система Maple работает с документами в стиле notebooks («блокноты» или «записные книжки»). Как было показано в уроке 1, документы содержат текстовые и формульные блоки, результаты вычислений, графики разного типа и другие компоненты. Документы могут готовиться с нуля или существовать в готовом виде — подготовленные кем-то ранее. Хранятся документы на внешних устройствах памяти в виде файлов. Файлом называют имеющую имя упорядоченную совокупность данных, размещенную на том или ином носителе — обычно на жестком, гибком или компакт-диске.

В Maple 7 используются файлы различных форматов, который указывается расширением файла (знак * означает произвольное имя файла):

- *.ms — файлы документов для систем с графическим интерфейсом (Windows/Macintosh);
- *.msw — файлы документов (Worksheets);
- *.txt — текстовые файлы (включая формат Maple-текст);
- *.tex — файлы в формате LaTeX;
- *.ind и *.lib — файлы библиотек;
- *.m — файлы внутреннего Maple-языка.

Файлы документов содержат все необходимые данные для правильного отображения содержимого документа в окне редактирования с указаниями координат расположения блоков, фактического содержания и характера выполняемых операций, форматов предоставления информации и т. д. Таким образом, файл содержит кроме текста, отображаемого на рабочем листе, специальные команды, адресованные Maple, аналогично файлам HTML, имеющим теги, предназначенные для интерпретации браузером.

Предусмотрена возможность записи документов и в особом формате LaTeX, предназначенном для создания книг и статей по математике. Текстовые файлы (с расширением .txt) можно просматривать и редактировать текстовыми редакторами, работающими с ASCII-кодировкой.

Важно отметить, что даже при записи документов со сложными рисунками используется не прямая запись их растровой или векторной копии, а сохранение данных для построения графиков. Поэтому размеры файлов Maple 7 невелики и их легко передавать по современным средствам телекоммуникаций, например по сети Интернет. Они требуют небольшого свободного пространства на дисках для записи. Тем не менее, чем сложнее график, содержащийся в документе, тем больше объем памяти, необходимой для хранения файла. Помимо обычных операций

по работе с файлами (запись на диск и загрузка с диска) предусмотрены возможности распечатки документов принтерами различного типа.

Меню File

Меню File содержит основные операции для работы с файлами документов (рис. 3.1).

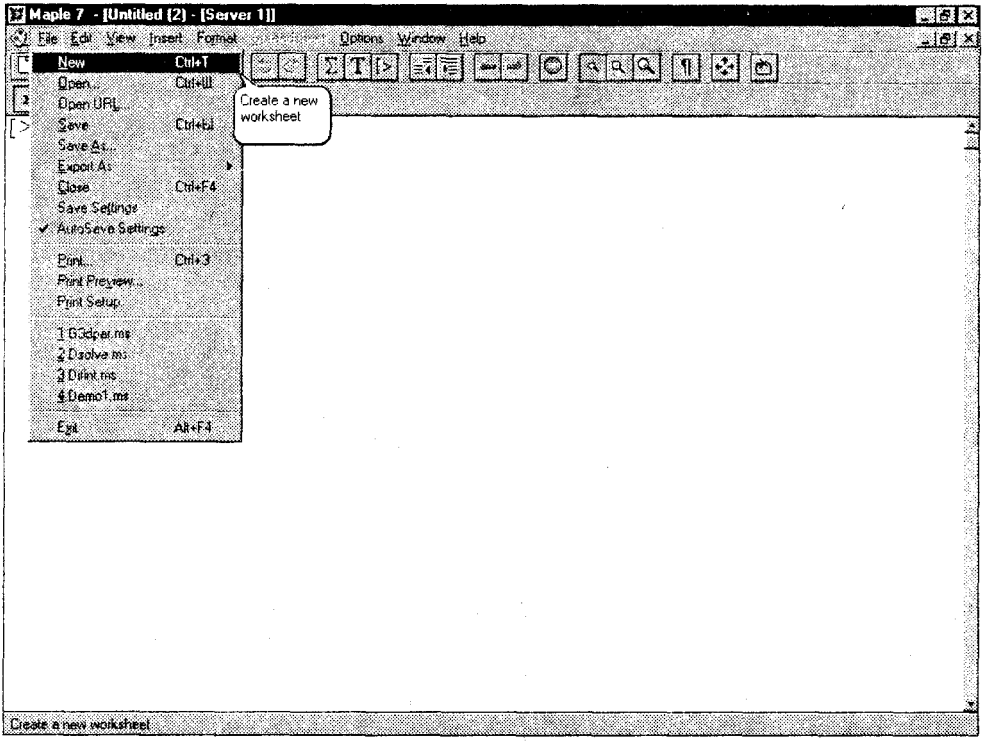


Рис. 3.1. Меню File и пустое окно нового документа

Меню содержит ряд операций, разбитых на группы. В первую группу входят следующие операции по работе с документами (в скобках приведены горячие клавиши):

- New (Ctrl+N) — создать новый документ;
- Open (Ctrl+O) — открыть существующий документ;
- Open URL — открыть URL-адрес;
- Save (Ctrl+S) — сохранить активный документ;
- Save As — сохранить активный документ под новым именем;
- Export As — экспортировать файл;

- Close (Ctrl+F4) — закрыть окно активного документа;
- Save Settings — запись конфигурации (установок) Maple;
- AutoSave Settings — автоматическая запись конфигурации.

Вторая группа команд относится к печати документов:

- Print Preview — предварительный просмотр документа перед печатью;
- Print (Ctrl+P) — печать документа;
- Printer Setup — установка параметров принтера.

Вторая из этих команд позволяет распечатать весь текст документа с комментариями, математическими формулами, таблицами и графиками. Печать производится принтерами в графическом режиме, поэтому и не очень быстро. Зато печатаются все шрифты и математические спецзнаки. Можно напечатать и отдельные части документа.

После этой группы команд имеется список документов (файлов с расширением .ms), которые были загружены в систему в предшествующие сеансы работы. Выбрав в этом списке название одного из файлов, можно быстро загрузить его, не тратя времени на открытие файла через команду Open.

Последняя группа представлена единственной командой:

- Exit (Alt+F4) — выйти из Maple 7.

ПРИМЕЧАНИЕ Если вы работаете в русифицированной версии Windows, то горячие клавиши могут содержать вместо латинских букв русские эквиваленты. Например, команда New имеет горячие клавиши Ctrl+N, тогда как в меню они могут быть обозначены как Ctrl+Т (см. рис. 3.1). На самом деле, как обозначать клавишу: русской буквой или латинской — дело вкуса.

Создание нового документа

Вернемся к широко используемым операциям первой группы и рассмотрим их более подробно. Заметим, что к некоторым из них можно быстро обратиться с помощью «горячих» клавиш (они указаны в скобках после названия команды).

Команда New используется для создания нового документа. Она открывает новое пустое окно редактирования и переводит Maple в режим редактирования (рис. 3.1).

При создании нового документа в его начале появляется ячейка со знаком приглашения >, после которой виден мигающий маркер ввода в виде вертикальной черты |. Ячейка ввода обрамляется открывающей квадратной скобкой. Созданный документ приобретает имя Untitled (N) (в вольном переводе — «Безымянный под номером N», где N — целое число). Следуя приглашению программы, можно приступать к работе в Maple.

Операция New дублируется кнопкой со значком в виде чистого листа, размещенной на панели инструментов. Разумеется, нажать ее вы сможете, только если панель инструментов выведена на экран.

Открытие документа

Команда Open служит для открытия созданного ранее документа. Вначале открывается диалоговое окно выбора файла (рис. 3.2). Для быстрого доступа к команде служит значок в виде открывающейся папки на панели инструментов.

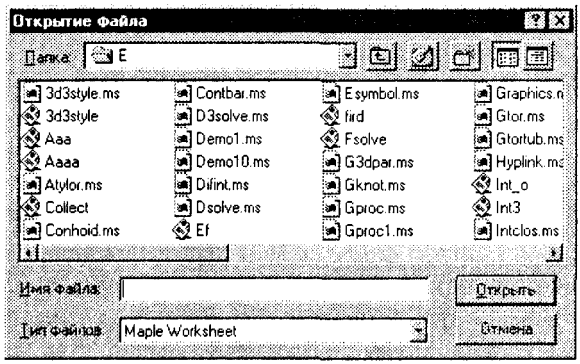


Рис. 3.2. Окно открытия документа

В окне Открытие файла вам нужно найти нужный файл, а затем дважды щелкнуть на его имени или на кнопке OK. Данное окно — пример единых диалоговых окон для всех приложений Windows.

Maple 7 — далеко не первая из версий системы компьютерной алгебры. Каждая очередная версия может иметь модифицированные элементы Maple-языка. Поэтому иногда возникает проблема частичной несовместимости документов, подготовленных в разных версиях системы Maple. Разработчики системы по мере возможностей учли это и предусмотрели автоматическое определение версии Maple, в которой был создан документ. Если выбранный документ был сохранен в старой версии программы, то перед его загрузкой появляется окно, представленное на рис. 3.3.

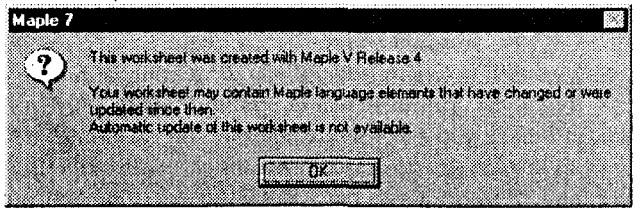


Рис. 3.3. Окно с предупреждением о загрузке документа другой версии Maple

Нажмите кнопку OK, и документ будет открыт. Не следует думать, что загруженный документ старой версии Maple непременно будет полностью работоспособен. К сожалению, это не так — некоторые документы требуют коррекции, прежде чем Maple 7 сможет их исполнить.

ПРИМЕЧАНИЕ Вас не должно удивлять появление в отдельных случаях элементов интерфейса с русскоязычными надписями — например, окно открытия файла (рис. 3.2). Они обусловлены тем, что Maple использует некоторые типовые элементы интерфейса операционной системы Windows. И если последняя русифицирована, то вполне возможно появление элементов интерфейса с русскоязычными надписями.

После загрузки документа его содержимое появляется в новом окне (см., например, рис. 1.16 или 1.17) и можно приступать к работе с ним.

Во многих системах открытие нового окна командой New или Open отменяет все предыдущие действия (значения переменных, функции и т. д.). Но в Maple 7 это не так — поскольку система предполагает совместную работу в нескольких окнах, каждое новое окно будет «знать» о происходившем в других окнах. Если же вы хотите начать с нуля в новом окне — исполните в нем команду restart.

Maple 7 позволяет работать и с документами, представленными в HTML формате и имеющими URL-адрес. Для загрузки таких документов служит команда Open URL. Она открывает простое окно с полем для ввода URL-адреса. Работа с ним очевидна.

Сохранение документа

Команда Save записывает содержимое активного в данный момент окна в виде файла на диск с использованием его текущего имени. Исключением будут документы, созданные командой New и не переименованные, тогда действие команды будет аналогично выполнению команды Save as, обсуждаемой ниже.

Следует с осторожностью пользоваться командой Save в том случае, когда вы модернизируете какой-либо документ, но желаете сохранить оригинал в неизмененном виде, ведь содержимое модернизированного файла будет записано «поверх» оригинала. Чтобы этого не произошло, для сохранения файла следует воспользоваться командой Save as, описанной в следующем разделе.

При подготовке сложных документов рекомендуется периодически (в некоторых ситуациях довольно часто) давать команду Save, сохраняя сделанные изменения. Это позволяет избежать потери хотя бы части проделанной работы в случае сбоя компьютера. (Не важно, чем будет вызвано выключение или зависание компьютера — халтурной работой электрика или программиста или шалостью вашего ребенка — если вы лишний раз сохраните файл, вам придется меньше сил потратить на его восстановление.) Выполнение команды Save не приводит к выдаче сообщений и окон (кроме уже упомянутого исключения), и поэтому ее не обременительно дать лишний раз, особенно если вы запомните «горячие» клавиши для нее — Ctrl+S.

Запись документа на диск с переименованием

Команда Save As отличается от предыдущей тем, что перед записью файла на диск в появившемся диалоговом окне вы можете изменить имя файла (рис. 3.4).

Таким образом можно сохранить доработанный документ и в то же время оставить неизменным оригинал.

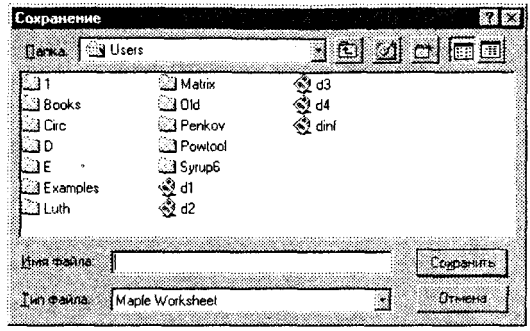


Рис. 3.4. *Диалоговое окно для записи файла с указанием его имени*

В списке **Папка** нужно найти папку, в которую вы хотите поместить файл, а в поле Имя файла вам нужно указать новое имя. Впрочем, не обязательно новое — если вы выберете другую папку, то можете сохранить и со старым именем — оригинал не пострадает.

Maple позволяет сохранить файл в следующих форматах (список внизу окна):

- Maple Worksheet (*.mws, *.ms) — файлы формата Maple 7;
- Maple Text (*.txt) — файлы в формате текста Maple;
- HTML Source (*.html) — файлы в формате HTML;
- Text (*.txt) — файлы в текстовом формате;
- LaTeX Source (*.tex) — файлы в формате LaTeX.

Формат HTML — новый для Maple формат, позволяющий записывать файлы документов в виде web-страниц.

Команда Save as особенно полезна при доработке и модификации файлов, например входящих в комплект поставки системы, когда надо сохранить оригинальные файлы в неприкосновенности. Для этого достаточно записать измененные файлы под новыми именами.

Экспорт файлов

Maple 7 имеет возможность экспорта файлов в различные форматы. Команда Export As открывает подменю, содержащее форматы, запись в которых поддерживает Maple После выбора нужного формата в подменю появляется окно, аналогичное окну для сохранения файла.

Maple экспортирует файлы в следующие форматы:

- Maple Text (*.txt);
- HTML (*.html);

- HTML with MathML (*.html);
- Plain Text (*.txt);
- Rich Text Format (*.rtf);
- LaTeX (*.tex).

Графики Maple может записывать в следующих форматах: DFX, EPS, GIF, JPEG, POV, WMF и BMP.

Закрытие документа

Команда Close закрывает окно вместе с текущим документом, и система переходит к работе со следующим окном (либо к пустому серому окну, если был закрыт последний документ).

Если закрываемый документ подвергался модификации, то система спросит, надо ли сохранять изменения (рис. 3.5).

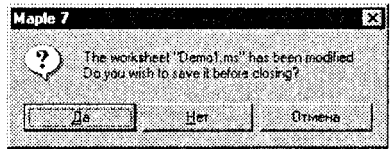


Рис. 3.5. Диалоговое окно, появляющееся при закрытии модифицированного документа

Следует помнить, что каждое окно, будучи сложным графическим объектом, занимает определенный и вовсе не малый объем памяти. Поэтому команда Close является эффективным средством освобождения оперативной памяти, особенно когда закрывается большой документ. Однако надо помнить, что бывшие в нем определения (например, значения переменных, введенные функции пользователя и т. д.) сохраняются в памяти, даже когда документ закрыт, естественно, пока вы не дадите команду restart.

Быстро закрыть документ можно, нажав клавиши Ctrl+F4.

Запись настроек программы

Для записи сделанных настроек Maple в меню File предусмотрены две команды: Save Settings и Auto Save Settings. Последняя команда — это флажок, при установке которого новая настройки Maple будут записываться автоматически при завершении работы.

Выход из системы

Команда Exit служит для выхода из Maple. Тогда при использовании операции Exit можно наблюдать последовательное исчезновение окон документов. Если

пользователь забыл записать какой-либо документ на диск, система сообщит об этом, выдав запрос. Нужно ответить Yes (Да), если документ нужно сохранить, и No (Нет), если сохранение не требуется. Однако стоит сохранить документы, подвергавшиеся редактированию и модификации, заранее — вдруг вы по ошибке нажмете не ту кнопку.

Не следует применять команду Exit, если вам необходимо временно переключиться в окно другой программы, так как повторная загрузка Maple занимает много времени.

Печать документов

Команда Print

Команда Print служит для печати документа. Она имеет кнопку с изображением принтера на панели инструментов для быстрого доступа.

После того как отдана команда Print, появляется диалоговое окно, показанное на рис. 3.6, — стандартное окно печати в Windows.

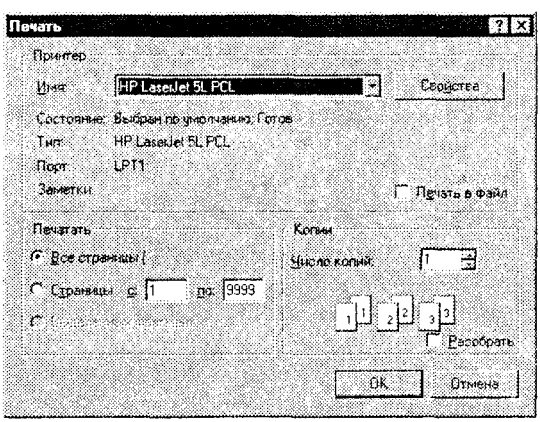


Рис. 3.6. Окно печати

Нажатие кнопки Свойства в окне печати открывает окно свойств выбранного принтера. Вид окна зависит от типа принтера, функции его элементов достаточно очевидны, поэтому мы воздержимся от обсуждения работы с ним.

Таким образом, в списке вверху окна печати прежде всего следует установить принтер. Можно просто оставить установленный по умолчанию. В диалоговом окне также можно указать диапазон страниц для печати и число копий, а также установить флажок для разбора страниц по копиям. В целом диалог с компьютером при осуществлении операции печати прост и нагляден.

Предварительный просмотр страниц

Даже одна страница документа может не поместиться на экране монитора. Поэтому перед печатью полезно просмотреть расположение элементов документа на странице. Для этого служит команда Print Preview, которая выводит окно с изображением текущей страницы (рис. 3.7).

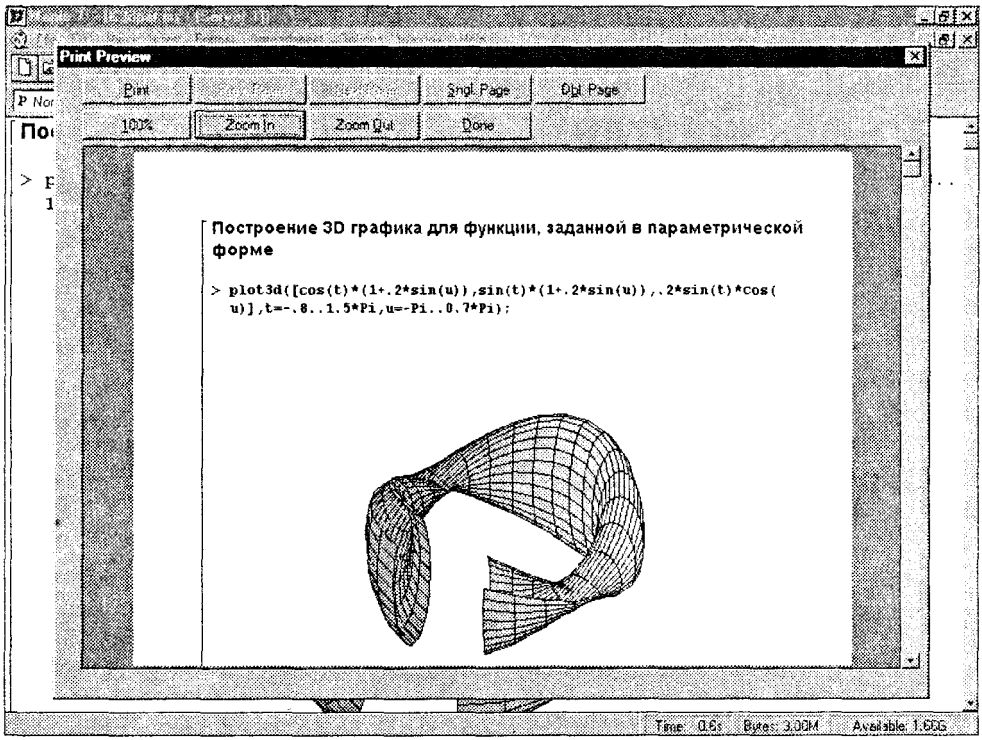


Рис. 3.7. Предварительный просмотр страницы, предназначенной для печати

У окна просмотра имеется ряд элементов управления. Прежде всего это полосы прокрутки для перемещения изображения документа в окне просмотра. Кроме того, имеется ряд кнопок, назначение которых указано ниже:

- Print — *печать просматриваемого документа*;
- Prev. Page — просмотр предыдущей страницы многостраничного документа;
- Next Page — просмотр следующей страницы многостраничного документа;
- Sngl. Page — просмотр одной полной страницы;
- Dbl. Page — просмотр двух полных страниц;
- 100% — установка масштаба в 100 %;
- Zoom In и Zoom Out — увеличение и уменьшение масштаба;
- Done — завершение работы с окном предварительного просмотра.

Использование команды Print Preview может сэкономить не один лист чистой бумаги и картридж краски.

Установка параметров принтера

Печать документов — одна из основных функций любой среды подготовки документов. В общем случае она предполагает установку параметров принтера. Для этого служит команда Printer Setup, приводящая к открытию диалогового окна, показанного на рис. 3.8.

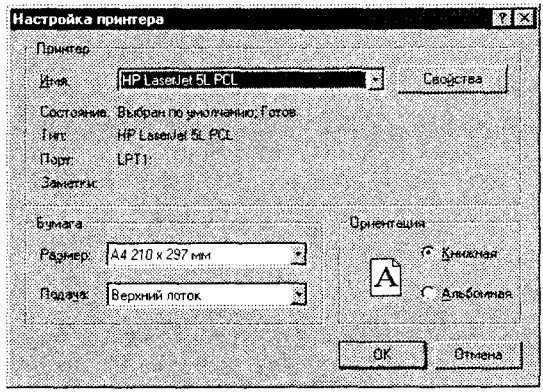


Рис. 3.8. Диалоговое окно настройки параметров принтера

Следует отметить, что вид этого окна будет варьироваться в зависимости от принтера и может существенно отличаться от приведенного на рисунке. Однако большинство параметров, которые требуют пользовательского вмешательства, очевидны.

Редактирование документов

Меню Edit

Как видно из рис. 3.9, меню Edit содержит различные операции редактирования. Они делятся на ряд групп. Первая группа содержит следующие операции:

- Undo (Ctrl+Z) — отменить последнюю операцию редактирования;
- Redo (Ctrl+Y) — восстановить последнюю отмененную операцию;
- Cut (Ctrl+X) — переместить выделенный фрагмент в буфер обмена;
- Copy (Ctrl+C) — скопировать выделенный фрагмент в буфер обмена;
- Copy As Maple Text — копирование выделения в буфер обмена в формате Maple-текста;

- Paste (Ctrl+V) — вставить содержимое буфера обмена в документ;
- Paste Maple Text — вставить данные из буфера обмена в формате Maple-текста;
- Delete Paragraph (Ctrl+Del) — удаление параграфа (строки);
- Select All (Ctrl+A) — выделение всех объектов документа.

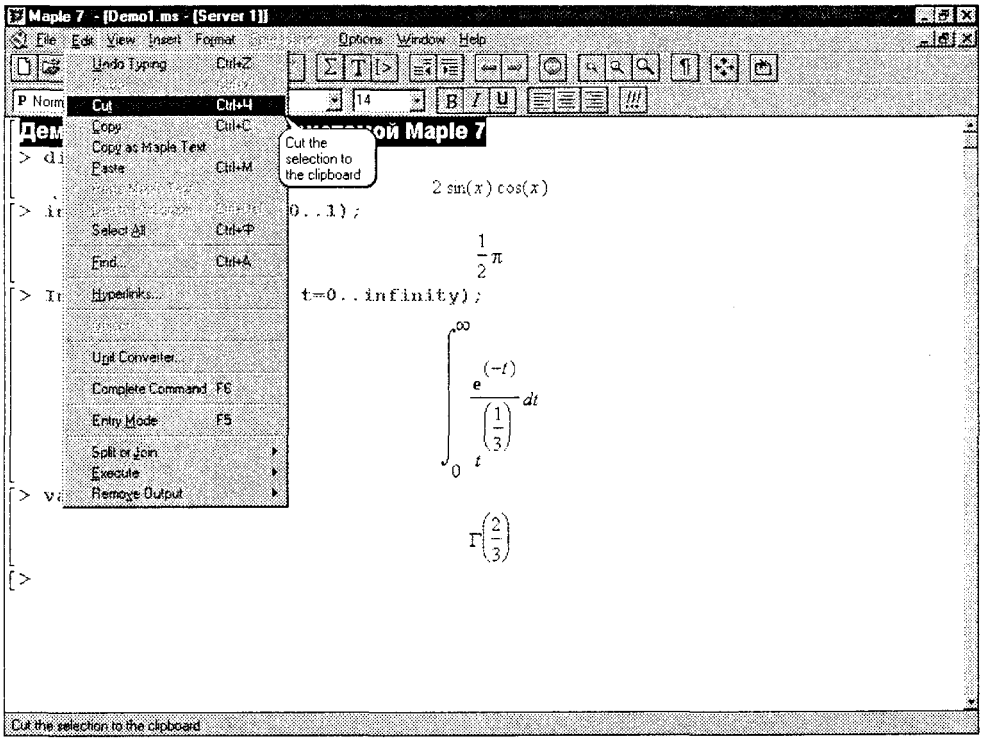


Рис. 3.9. Меню Edit

Операции первой группы используют буфер обмена (Clipboard). Так называется специально организованная и динамически изменяющаяся область памяти в операционной системе Windows. В нее могут помещаться различные (обычно предварительно выделенные) объекты, документы и даже файлы. Буфер можно использовать для обмена объектами как в пределах текущего документа, так и между различными документами и даже приложениями.

ПРИМЕЧАНИЕ

Очень часто некоторые операции меню Edit будут недоступны — они отображены серым шрифтом. Это означает, что в данный момент не существует объекта, к которому может быть применена команда. Однако в демо-версии Maple команды копирования и вставки будут недоступны всегда — таково ограничение создателей.

Несколько следующих групп представлены одной операцией:

- Find (Ctrl+F5) — выводит окно поиска заданной строки и ее замены на другую строку;

- Hyperlinks — редактирование гиперссылок;
- Object — редактирование объекта;
- Unit Converter — перевод между различными единицами измерения;
- Complete Command — подсказка для завершения текущей незаконченной команды Maple-языка;
- Entry Mode (F5) — переключение режима ввода.

Последняя команда позволяет менять режим строк ввода — они могут содержать математические выражения или неисполняемые текстовые комментарии. Именно благодаря текстовым комментариям документы Maple 7 приобретают достаточно наглядный вид. Наглядность документов дополнительно повышается благодаря возможности представления результатов вычислений (а иногда и вводимых выражений) в естественной математической форме.

Восьмая группа команд открывает подменю, содержащие команды с ячейками и секциями документа:

- Split or Join — разделение или объединение объектов;
- Execute — исполнение выделенных или всех строк документа;
- Remove Output — удаление вывода для выделенных или всех строк документа.

Команды подменю Split or Join позволяют легко модифицировать вид документов путем разделения и объединения строк и секций.

Отмена последней операции

Команда Undo Delete служит для отмены последней операции редактирования. Она позволяет удалить до 5 последних операций. Если вы случайно удалили нужный текст — вам придет на помощь команда Undo. В меню, а также во всплывающей подсказке указывается название последней операции редактирования — той, которую вы собрались отменить.

Восстановление отмененной операции

Команда Redo позволяет вернуть отмененную операцию, если в этом возникает необходимость. Действие этой команды, естественно, тоже распространяется не более чем на пять операций. Команды Undo и Redo имеют кнопки (с изображением стрелок) на панели инструментов для быстрого доступа к себе.

Перенос объекта в буфер обмена

Команда Cut копирует выделенный объект в буфер обмена и удаляет его из документа. Выделить объект или группу объектов можно, обведя его мышью при нажатой левой кнопке. Можно также использовать клавиши перемещения курсора при нажатой клавише Shift. Выделенный фрагмент отмечается черным фоном и инверсией цвета входящих в него символов (на рис. 3.9 выделена первая строка).

Графические объекты не инвертируют свои цвета при выделении, к тому же их можно вырезать и копировать, когда на них находится маркер ввода.

Вырезанный командой Cut объект поступает в буфер обмена, а его изображение в окне редактирования исчезает. Эта команда выполняется также комбинациями клавиш Shift+Del и Ctrl+X (в зависимости от настроек операционной системы). На рис. 3.10 показан результат выполнения команды Cut — видно исчезновение выделенного на предыдущем рисунке фрагмента в документе.

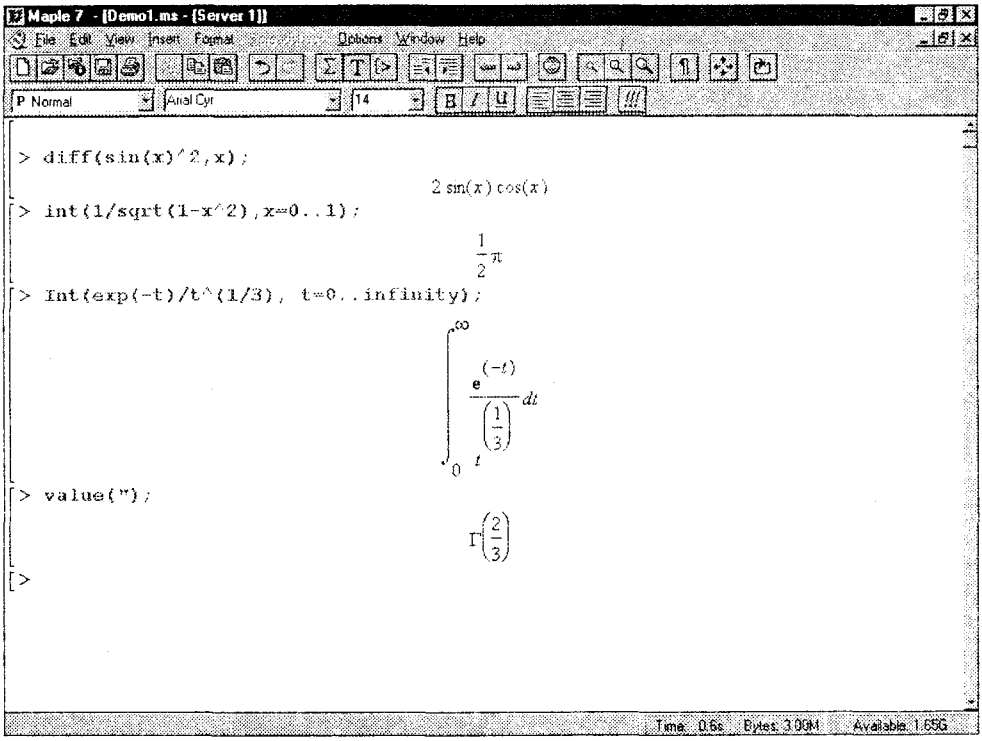


Рис. 3.10. Документ с рис. 3.9 после выполнения команды Cut

Щелчок левой кнопкой мыши за пределами выделенных объектов снимает все выделения.

Копирование объекта в буфер

Комнада Copy аналогична Cut, но с одним существенным отличием — выделенный объект (или блок объектов) не удаляется в окне редактирования. Эта команда обычно используется в том случае, когда нужно перенести заданный объект в другое место, сохранив при этом оригинал, — продублировать объект.

Особенно полезна операция Copy при составлении документа из частей других документов, в том числе страниц справочной системы Maple (рис. 3.11). Пере-

ключаясь между окнами с помощью команд Copy и Paste, описанных ниже, можно с легкостью составить достаточно большой документ, не написав ни строки.

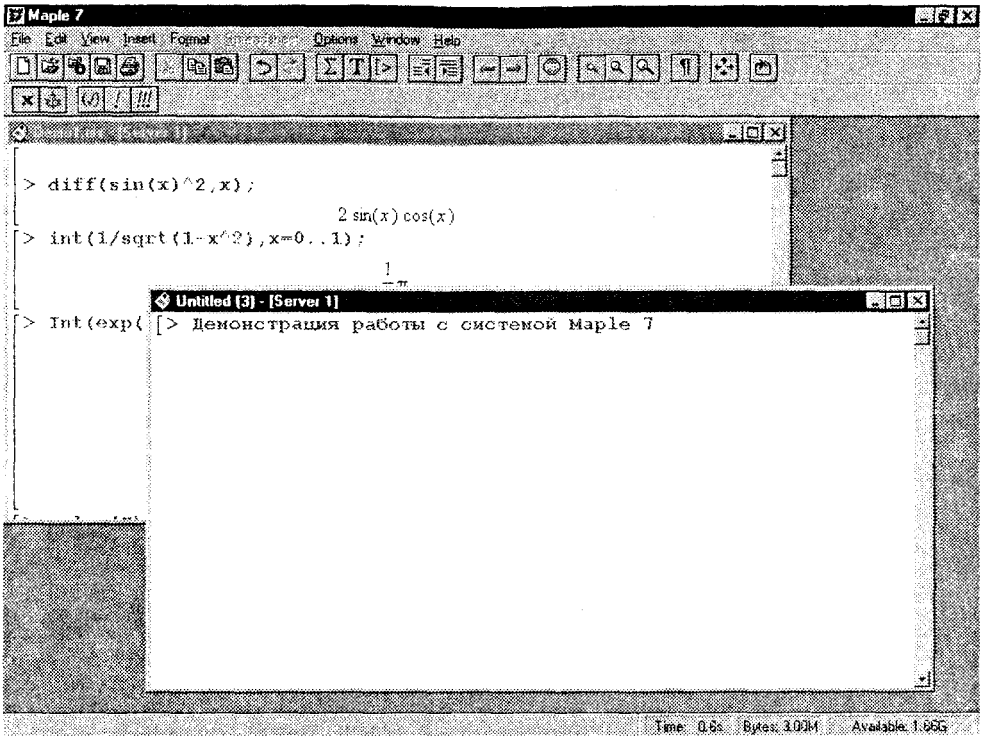


Рис. 3.11. Пример переноса выделенной в одном документе строки в окно другого документа

В данном случае титульная надпись, удаленная командой Cut, перенесена в другой, пустой, документ.

Перенос и копирование объектов перетаскиванием

В Maple 7 имеется возможность переноса объектов из одного окна в другое методом перетаскивания (Drag and Drop). Для этого на группу выделенных объектов надо навести указатель мыши и при нажатой левой кнопке начать перенос объектов в новое место или новое окно. Отпустив левую кнопку мыши, можно наблюдать перенос объектов на новое место. Если все время держать нажатой клавишу Ctrl, то переносимый блок объектов будет сохранен и на старом месте.

Копирование в буфер обмена в формате Maple-текста

Команда Copy As Maple Text используется в тех случаях, когда необходимо, чтобы скопированная в буфер информация была представлена в текстовом формате. При этом в буфер копируются выражения как из строк ввода, так и строк выво-

да — все в текстовом формате. Используя в этом случае команду Paste можно вывести из буфера все выражения как из входных, так и из выходных строк в текстовом формате (рис. 3.12).

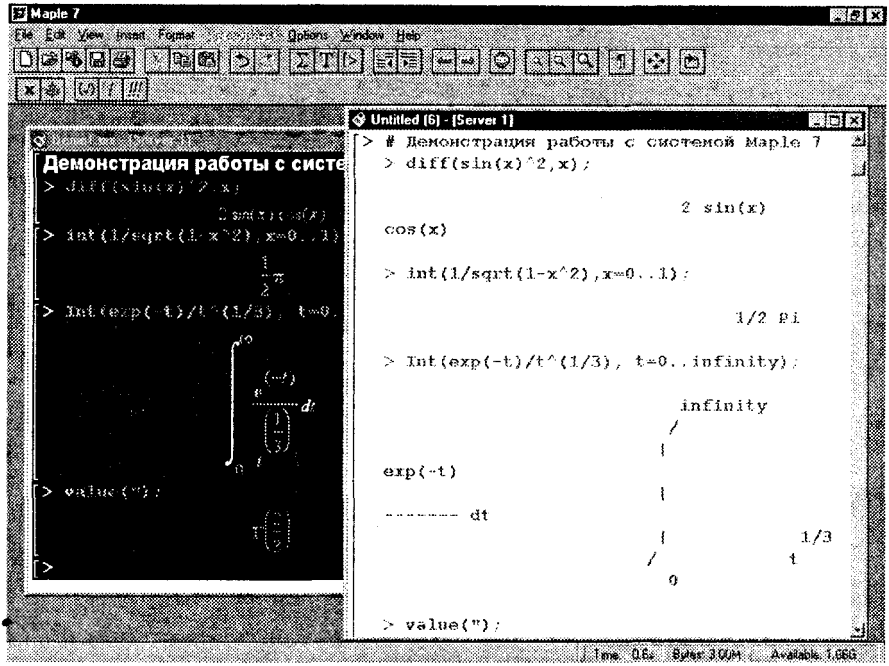


Рис. 3.12. Пример использования операций Copy as Maple Text и Paste

Обратите внимание, что графическое представление интеграла в строке вывода превратилось в его отображение в текстовом формате. То же имеет место в отношении представления производных, сумм, произведений и т. д. Обратите также внимание на то, что текстовый комментарий в данном случае выводится стандартным шрифтом. Атрибуты (признаки) стиля, цвета и размера символов теряются.

Вставка из буфера обмена в документ

Команда Paste копирует содержимое буфера обмена, помещенное туда командами Copy или Cut, в место, указанное маркером ввода. При этом сохраняются форматы всех объектов документа, если они были скопированы (рис. 3.13).

Как видно из рис. 3.13, возможно применение этой операции не только в пределах окна одного документа, но и при переносе данных из одного окна в другое.

Надо отметить, что при копировании в буфер обмена математической формулы из ячейки вывода и вставке ее в строку ввода формат формулы меняется — она автоматически превращается в текстовое выражение. Это иллюстрирует рис. 3.14.

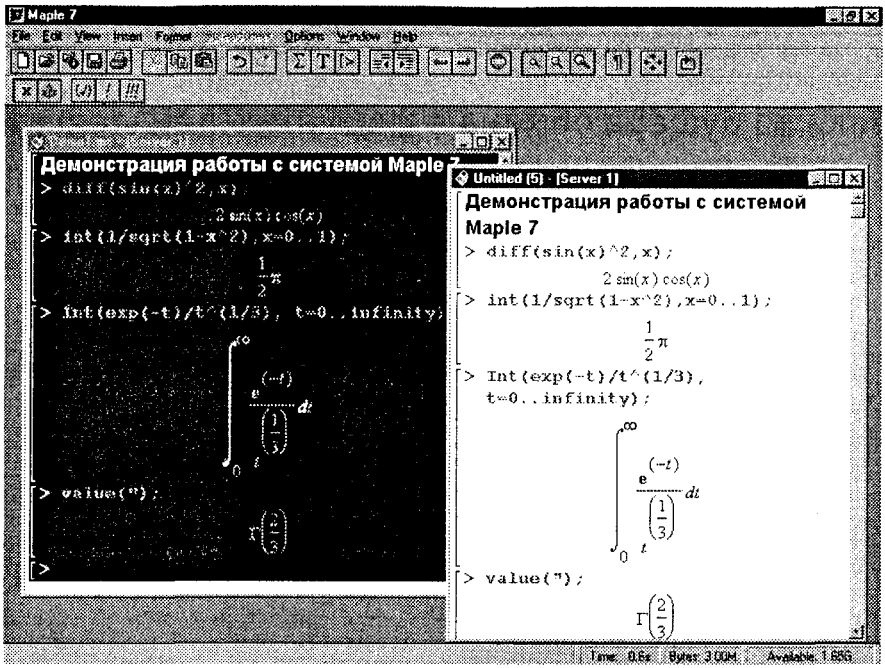


Рис. 3.13. Пример использования операций Copy и Paste

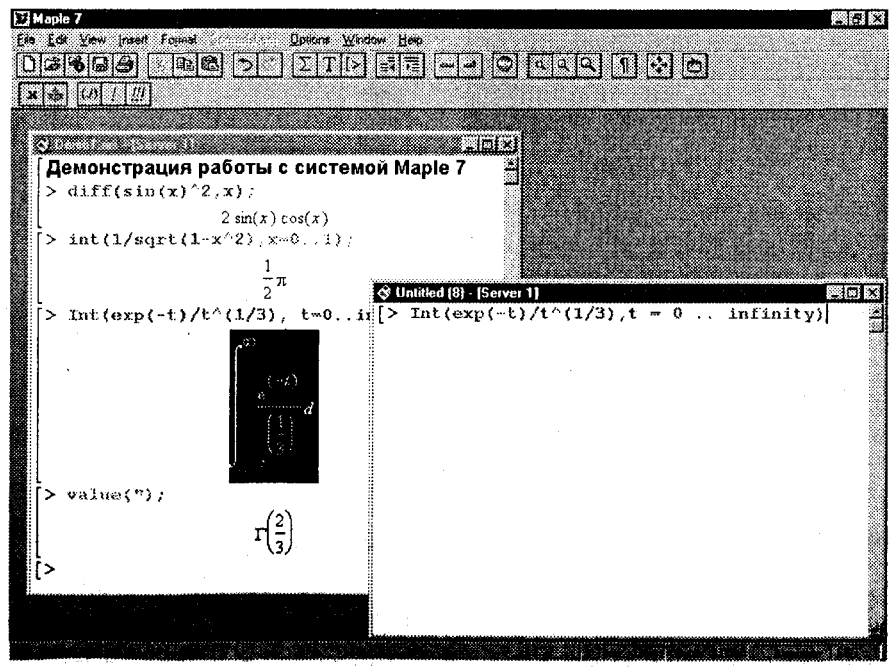


Рис. 3.14. Пример копирования формулы из ячейки вывода одного документа в ячейку ввода другого документа

Различные варианты преобразования форматов при использовании операций копирования и вставки надо учитывать при подготовке сложных документов.

Вставка из буфера обмена в формате Maple-текста

Операция Paste As Maple Text служит для вставки данных из буфера обмена с одновременным их преобразованием в текстовый формат.

Если данные в буфер обмена поступили в результате выполнения операции Copy as Maple Text, то при операции Paste Maple Text в документ будут скопированы только данные из строк ввода в формате Maple-текста. Это показано на рис. 3.15.

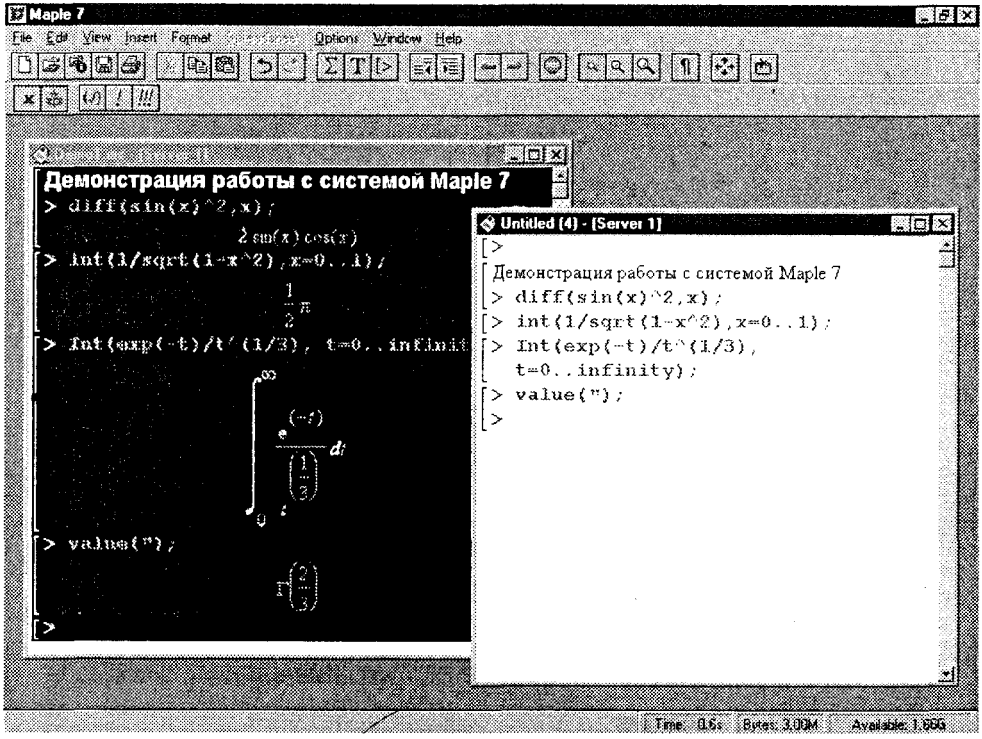


Рис. 3.15. Пример использования операций Copy as Maple Text и Paste Maple Text

Любопытно сравнить две пары операций, действие которых демонстрируют рис. 3.12 и 3.15.

Уничтожение выделенного абзаца

Команда Delete Paragraph служит для уничтожения блока ввода, на котором расположен курсор. При этом в буфер обмена удаленный текст не заносится. В отличие от операции вырезания Cut ее применение не загружает буфер и предотвращает нехватку оперативной памяти. Если вы случайно удалите нужную строку — есть время спохватиться — дайте команду Undo.

Выделение всех объектов

Команда Select All выделяет все объекты документа. Это полезно, например, для вставки всего документа в другой документ.

Поиск подстроки и ее замена

Команда Find служит для поиска указанного фрагмента в тексте документа. Она открывает окно Find, в котором можно указать искомую фрагмент-подстроку (рис. 3.16). При нажатии кнопки Next будет производиться поиск следующего вхождения (ниже по тексту), а при использовании Previous — предыдущего. Если искомое слово не найдется, выводится окно с сообщением (слева на рис. 3.17).

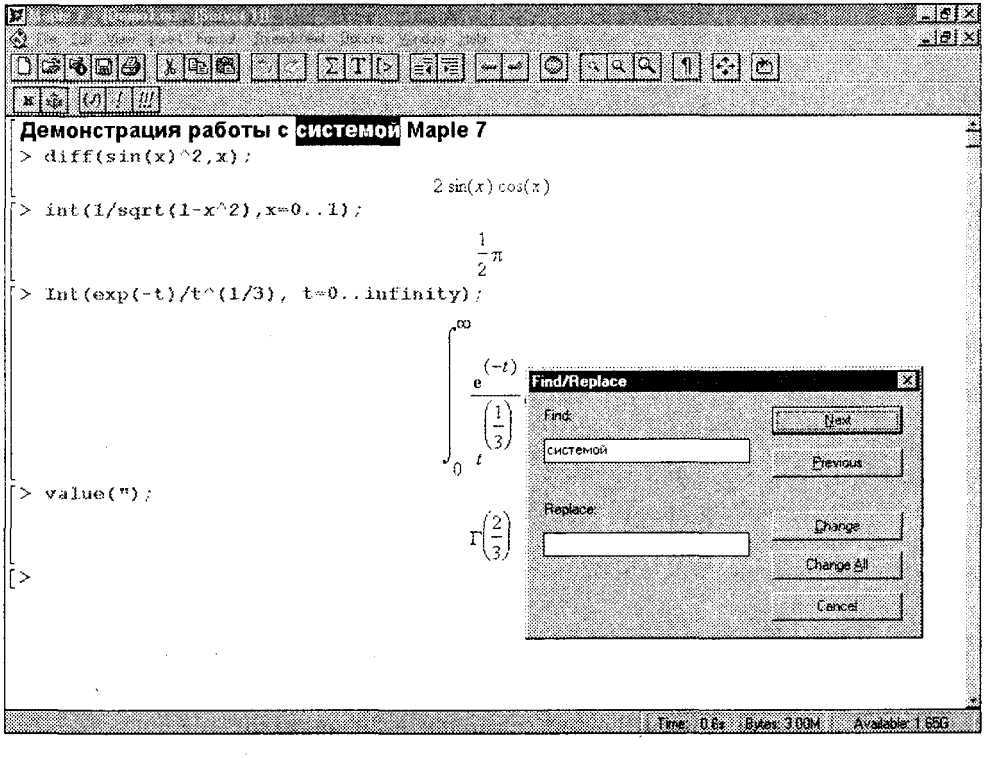


Рис. 3.16. Поиск заданного фрагмента текста

При обнаружении указанного фрагмента он выделяется, и маркер ввода устанавливается в начало найденной подстроки. После этого можно приступить к ее редактированию.

С помощью команды Find возможна также замена подстроки. Для этого надо указать текст замены в нижнем поле окна. Чтобы заменить найденную подстроку, надо нажать кнопку Change, чтобы заменить все подстроки, — Change All.

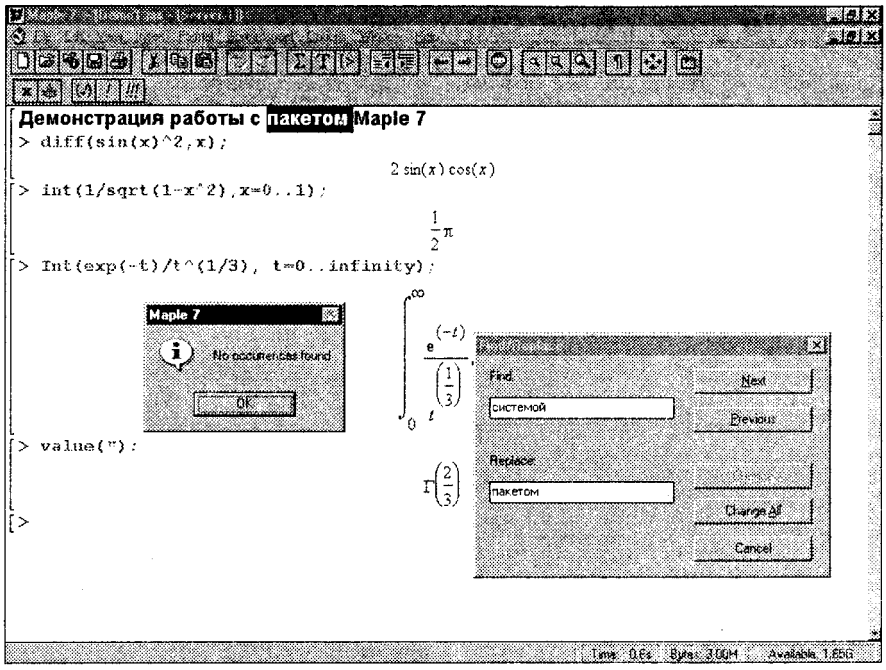


Рис. 3.17. Пример замены одной подстроки другой

Включение и выключение режима ввода текста

Maple 7 позволяет вводить в ячейки текст комментариев и исполняемые математические выражения (для того чтобы ввести в одну ячейку комментарий и функцию Maple, надо воспользоваться командами меню Insert). Операция Input mode служит для переключения между режимами ввода. Горячая клавиша команды — F5. Если режим ввода текста комментария включен, приглашение в виде знака > исчезает. Если этот режим отменен, то можно вводить исполняемые математические выражения, придерживаясь синтаксиса языка Maple.

Операции разделения и объединения объектов

Команда Split or Join служит для разделения или объединения объектов документа, она открывает подменю со следующими операциями:

- Split Execution Group (F3) — разделение строки на две;
- Join Execution Group (F4) — соединение смежных строк;
- Split Section (Shift+F3) — разделение секции на две;
- Join Section (Shift+F4) — объединение смежных секций.

Исполнение выделенных ячеек или всего документа

Команда Execute служит для запуска вычислений во всех выделенных ячейках или во всех ячейках документа. Соответственно она имеет подменю с двумя командами:

- Selection — исполнение выделенных ячеек;
- Worksheet — исполнение ячеек по всему документу.

Заметим, что альтернативой является нажатие клавиши Enter для каждой исполняемой строки документа, что при больших документах довольно нудное занятие.

Удаление ячеек вывода

Команда Remove Output служит для удаления из документа всех ячеек вывода. Это полезно для редактирования ячеек ввода, поскольку объем документа при этом заметно сокращается. Она открывает подменю с двумя командами:

- From Selection — удаление вывода только для выделенных ячеек;
- From Worksheet — удаление вывода для всего документа.

Рассмотрим, к примеру, документ, который был представлен на рис. 3.16. После выполнения для него команды Remove Output From Worksheet документ принимает вид, показанный на рис. 3.18.

Заметим, что документ с рис. 3.18 можно вернуть к исходному виду, выполнив команду Execute Worksheet, описанную выше. Однако в данном конкретном случае исполнение команды `value(")` даст сообщение об ошибке, поскольку в версиях Maple 6 и 7 данная команда должна быть записана в виде `value(%)`.

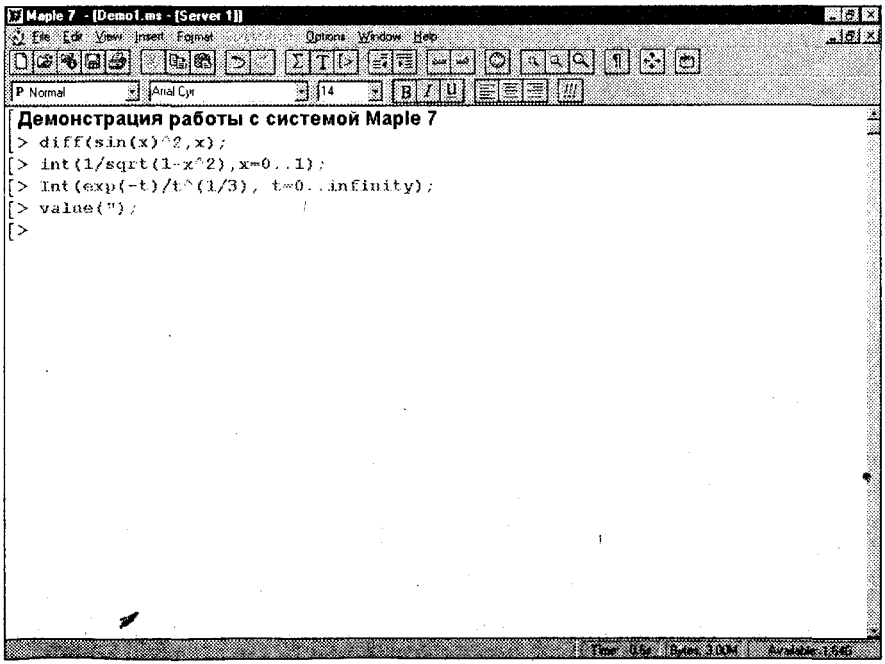


Рис. 3.18. *Документ после удаления всех ячеек вывода*

Операции вставки

Меню Insert

Документ системы Maple 7 состоит из различных объектов — текстовых областей, областей ввода, ячеек, абзацев, секций, подсекций и т. д. Некоторые из них формируются автоматически по мере ввода и исполнения документа. Но, кроме того, в меню Insert (рис. 3.19) сосредоточены основные команды по обеспечению вставки различных объектов в редактируемый документ. Это позволяет быстро модифицировать документ и добиться его большей выразительности.

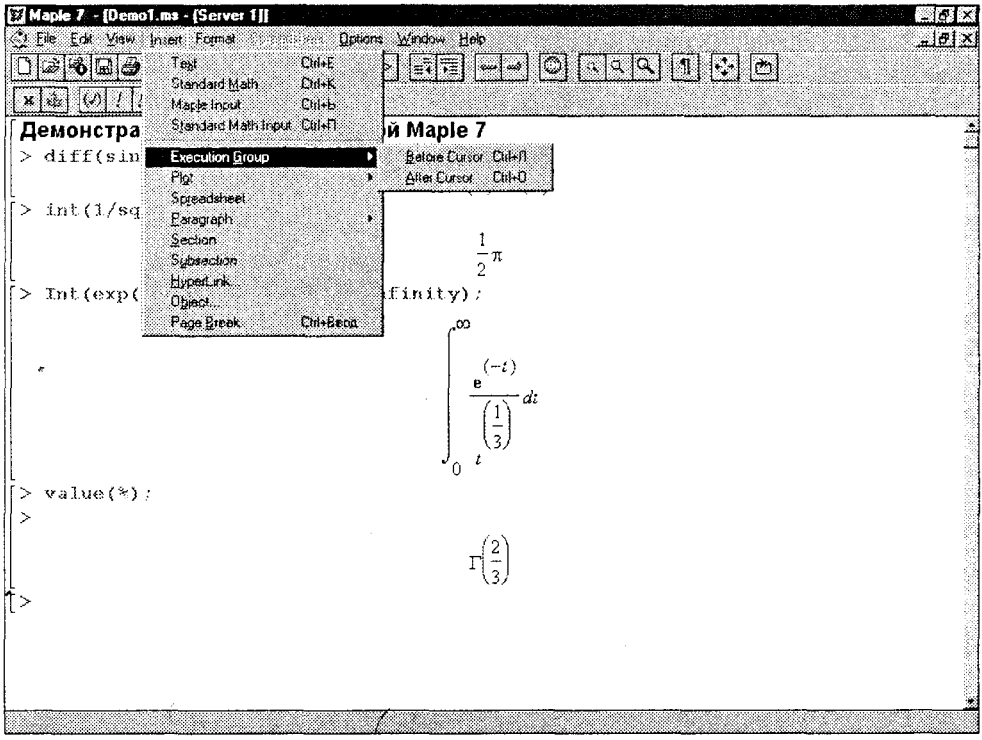


Рис. 3.19. Меню Insert

Следует отметить, что меню Insert является контекстно-зависимым. К примеру, оно исчезает, если выделен график.

Меню вставки разделено на две группы. В первой группе содержатся следующие команды:

- Text (Ctrl+T) — ввод текста;
- Standard Math (Ctrl+R) — ввод неисполняемых математических выражений;
- Maple Input (Ctrl+M) — ввод исполняемых выражений в Maple-формате;

- Standard Math Input (Ctrl+G) — ввод выражений в математической форме в строку ввода.

Во второй группе содержатся следующие команды:

- Execution Group — вставка исполняемой ячейки до или после маркера ввода;
- Plot — вставка пустого шаблона двумерного или трехмерного графика;
- Spreadsheet — вставка электронной таблицы;
- Paragraph — вставка текстовой области (абзаца);
- Section — вставка кнопки секции;
- Subsection — вставка кнопки подсекции;
- HyperLink — вставка гиперссылки;
- Object — вставка связанного или внедренного объекта;
- Page Break — вставка разрыва страниц.

Перейдем к более подробному рассмотрению этих команд.

Ввод текста

Команда Text приводит к исчезновению знака приглашения >, после чего можно сразу же начинать ввод текста комментария. Данная команда позволяет формировать ячейку, содержащую текст комментария и исполняемые функции.

Ввод выражений в стандартной форме

Команда Standard Math выводит в строке ввода вопросительный знак. После этого в поле ввода строки форматирования можно начинать ввод неисполняемого математического выражения (рис. 3.20).

Рис. 3.20. Ввод математического выражения по команде Standard Math

По завершении ввода надо нажать клавишу Enter, и выражение появится в строке ввода (рис. 3.21). При этом выражение будет выделено.

Рис. 3.21. Завершение ввода по команде Standard Math

Ввод выражений

Команда Maple Input в меню Insert превращает текущую строку в строку ввода исполняемых математических выражений. В начале строки появляется приглашение ко вводу в виде значка >, после чего можно начинать ввод выражения. Кнопка со знаком x в начале контекстной панели позволяет представить вводимое выражение в естественной математической форме, если таковая возможна. В таком случае ввод выражения осуществляется в поле на контекстной панели инструментов.

Ввод математических выражений

Команда Standard Math Input выводит новую строку ввода со знаком вопроса в ней. После этого ввод начинается в поле ввода строки форматирования. По завершении ввода нажимается клавиша Enter и введенное выражение появляется в строке ввода (обычно с выделенной последней частью).

Вставка исполняемых ячеек до и после курсора

Команда Execution Group обеспечивает вывод подменю с двумя командами:

- Before Cursor (Ctrl+K) — вставка исполняемых ячеек ввода до курсора;
- After Cursor (Ctrl+J) — вставка исполняемых ячеек ввода после курсора.

Напоминаем, что признаком исполняемых ячеек является знак приглашения >. Данные команды позволяют ввести в любом месте документа новые входные ячейки, что часто бывает нужно при модификации документов.

Электронные таблицы

Вставка электронных таблиц

Электронные таблицы были впервые введены в реализацию Maple V R5. В системе Maple 7 для вставки электронных таблиц используется команда Insert Spreadsheet. Она выводит шаблон пустой таблицы, показанный на рис. 3.22.

Как видно из рис. 3.22, электронная таблица представляет собой двумерный массив ячеек, имеющих адресацию по строкам и столбцам. Номера строк задаются цифрами, а номера столбцов — латинскими буквами. Верхняя левая ячейка имеет адрес A1, где A — номер столбца и 1 — номер строки. Если одиночные буквы в номерах столбцов заканчиваются, происходит переход на двухбуквенные адреса (AA, AB, AC и т. д.). Такая адресация используется в функциях обработки табличных данных.

По команде Insert Spreadsheet вставляется пустая электронная таблица, во всех ячейках которой нет никаких данных. Однако помимо заполнения таблицы с по-

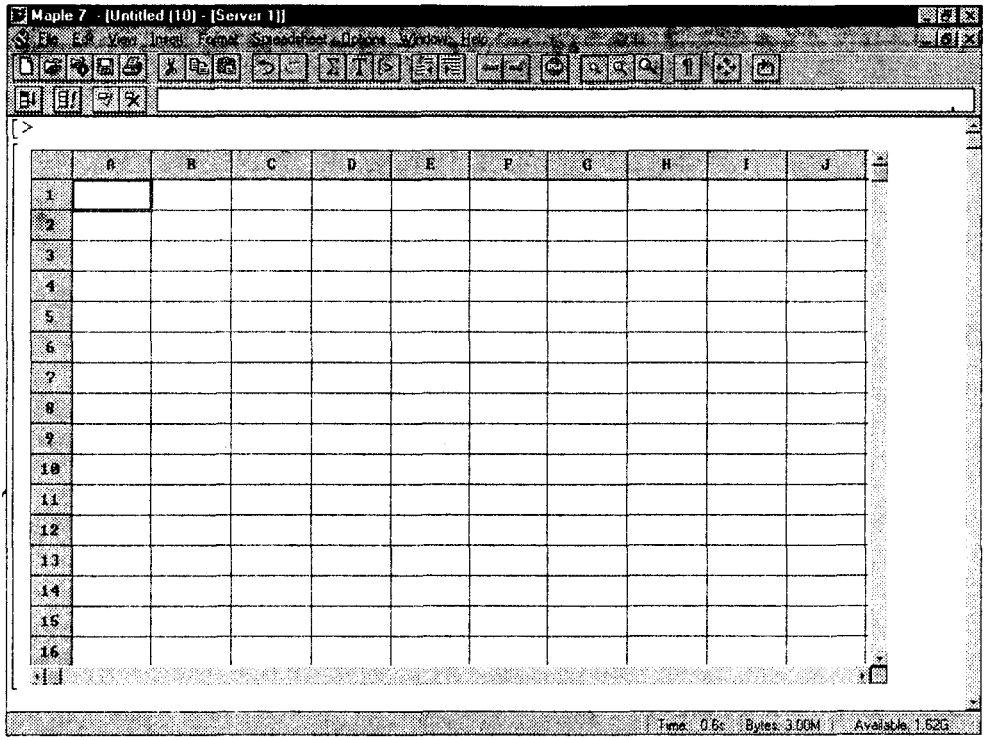


Рис. 3.22. Вставка шаблона электронной таблицы

мощью соответствующих операций можно провести заполнение вручную. Для этого достаточно мышью выделить ячейку, щелкнув в ней левой кнопкой. Ячейка обводится черным контуром (ячейка A1 на рис. 3.22), и появляется контекстное меню с полем для ввода выражения. Во время ввода выражения ячейка покрывается серой сеточкой. Если после набора выражения нажать клавишу Enter, то числовое значение выражения будет помещено в ячейку A1 таблицы. К примеру, на рис. 3.23 показано введенное выражение 2+3. Однако в поле редактирования сохраняется исходное выражение (рис. 3.23).

Если маркер ввода находится в одной из ячеек электронной таблицы, становится доступным меню Spreadsheet (рис. 3.23).

Меню Spreadsheet

Для работы с таблицами в Maple 7 появилось отдельное меню Spreadsheet (рис. 3.23). Оно содержит набор команд, обеспечивающих работу с табличными данными:

- Evaluate Selection — вычисление выражения в выделенной ячейке;
- Evaluate Spreadsheet — вычисление выражений по всем ячейкам таблицы;

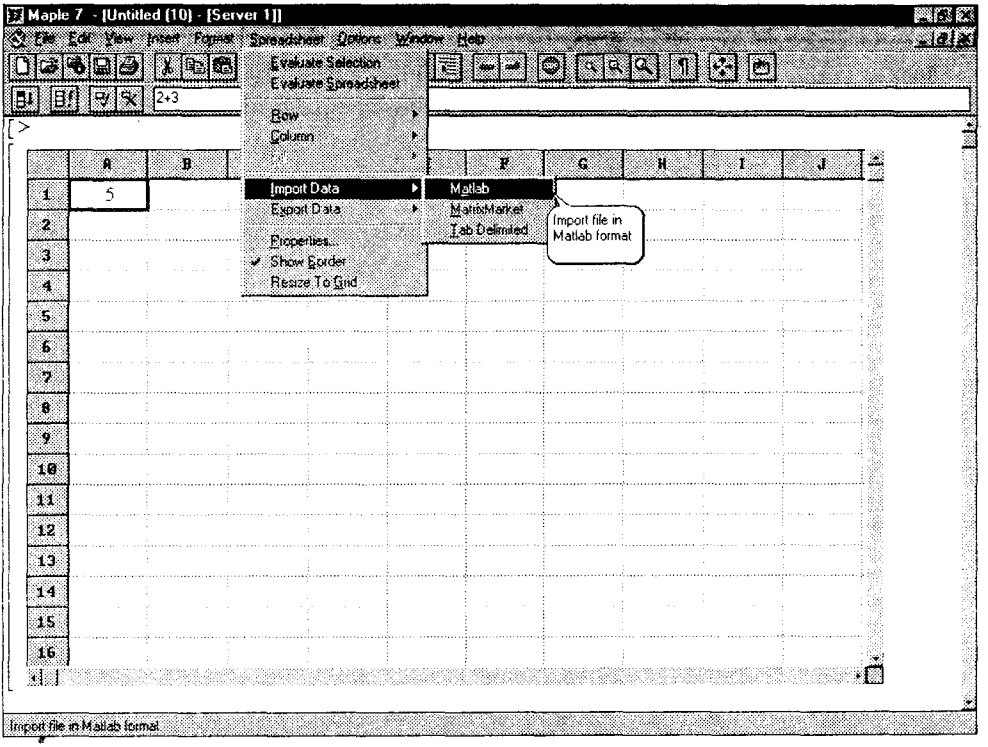


Рис. 3.23. Шаблон электронной таблицы и меню Spreadsheet

- Row — работа со строками (вставка, удаление и т. д.);
- Column — работа со столбцами (вставка, удаление и т. д.);
- Fill — автоматическое заполнение ячеек;
- Import data — импорт данных из других программ (например, из MATLAB);
- Export data — экспорт данных в другие программы;
- Properties — просмотр свойств ячеек;
- Show Border — управление показом обрамления таблицы;
- Resize to Grid — изменение размеров видимой таблицы с обрезанными ячейками до размера таблицы с целыми ячейками (дополнение идет по линиям раздела ячеек).

Если таблица активизирована (курсор находится внутри таблицы), то нажатие правой клавиши мыши вызовет появление контекстного меню. Его вид аналогичен виду описанного выше меню Spreadsheet.

Возможности Maple 7 в обработке табличных данных намного превосходят возможности обычных табличных процессоров. В частности, наряду с текстовыми и численными данными электронные таблицы Maple 7 могут работать с символьными данными — формулами.

Работа с электронными таблицами

Для ввода данных в ячейку таблицы достаточно щелкнуть на ней мышью. После этого можно вводить нужные данные в поле ввода контекстной панели (она видна на рис. 3.22 и 3.23 под панелью инструментов). Контекстная панель в режиме редактирования таблиц имеет четыре кнопки. Их назначение (слева направо) следующее:

- Fill a range of cells — автоматическое заполнение ячеек таблицы;
- Evaluate all stale cells in the spreadsheet — исполнение всех ячеек таблицы;
- Accept the input and evaluate it — ввод напечатанных данных и их исполнение;
- Restore input to the previous value — восстановление предшествующего значения ячейки.

Основным способом ввода данных является активизация ячейки таблицы мышью и ввод данных (объектов) в поле ввода контекстной панели. Нажатие третьей кнопки (Accept the input and evaluate it) или нажатие клавиши Enter приводит к вводу данных в ячейку и их исполнению.

Имеется ряд возможностей для автоматического заполнения ячеек таблицы. Например, можно заполнить ряд ячеек, примыкающих к заданной ячейке, предварительно наметив направление заполнения. Для этого курсор помещается в заданную ячейку, а затем мышь перемещается в нужном направлении при нажатой левой кнопке. На рис. 3.24 показан случай, когда в заданную ячейку A1 помещено число 2 и затем мышью выделены первые ячейки столбца A.

Теперь, нажав первую кнопку (Fill a range of cells) на контекстной панели форматирования (или исполнив команду Spreadsheet Fill Down в меню), можно вывести окно автоматического заполнения ячеек таблицы — Fill. Это окно также показано на рис. 3.24. В этом окне можно задать направление заполнения (обычно по умолчанию задано уже направление заполнения при выделении ячеек) и указать шаг изменения аргумента и значение, которого он не должен превышать. Например, на рис. 3.24 заданы шаг 3 и конечное значение 21. Нажав кнопку OK, можно увидеть автоматическое заполнение таблицы. При этом она принимает вид, показанный на рис. 3.25.

В ячейки таблиц можно вносить различные математические формулы в соответствии с синтаксисом языка Maple 7. При этом возможно сослаться на любую другую ячейку. Такая ссылка указывается значком тильда (~) перед адресом ячейки. Так, обозначение ~A1 означает, что будут подставлены данные из ячейки A1.

В качестве примера составим таблицу значений n, интеграла int(x^n,x) и производной diff(x^n,x) для n = 1..9. В готовом виде эта таблица представлена на рис. 3.26.

Подготовка такой таблицы проходит в три этапа. Вначале формируется первый столбец вводом в ячейку A1 имени переменной n, а в ячейку A2 — значения 1. После этого выделяются ячейки от A2 до A10 и с применением автоматического заполнения они заполняются числами от 1 до 9.

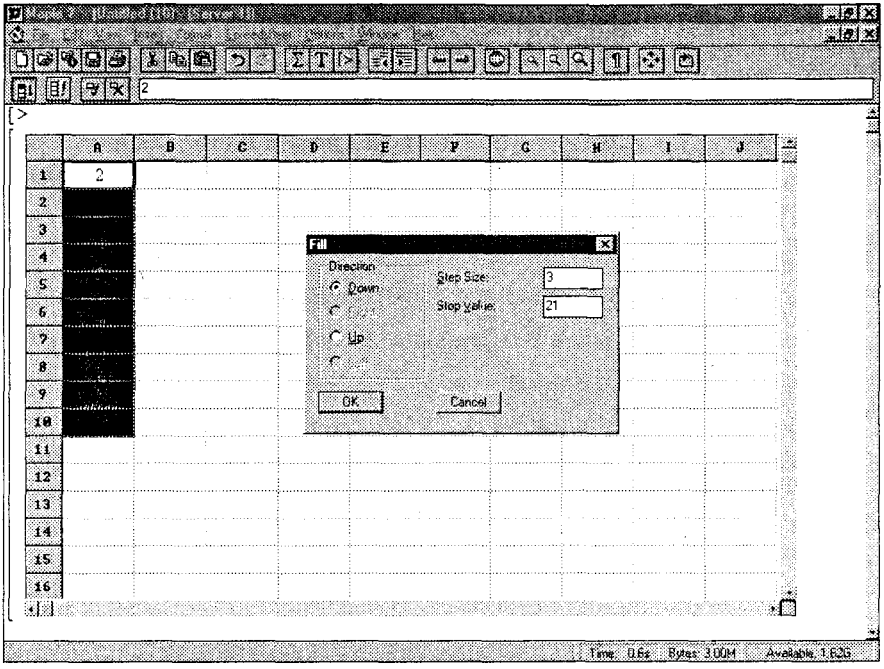


Рис. 3.24. Подготовка к автоматическому заполнению ячеек под заданной ячейкой

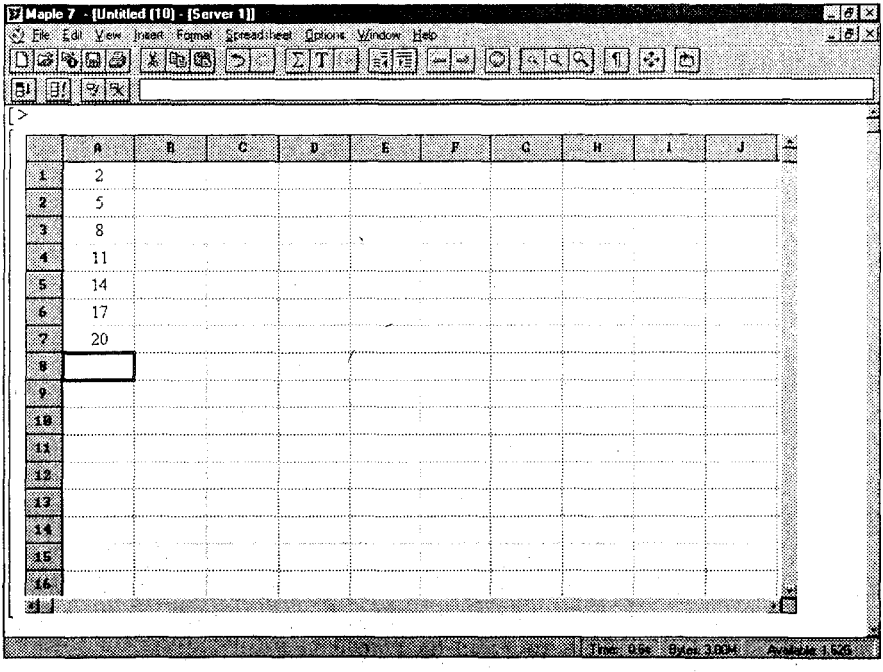
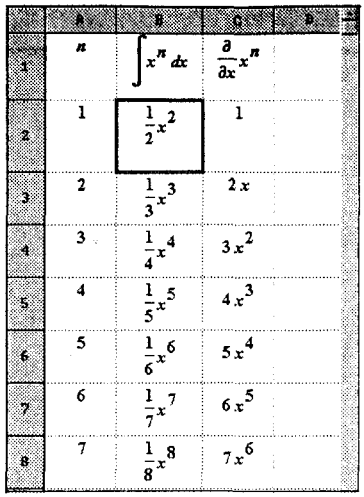


Рис. 3.25. Электронная таблица с рис. 3.24 после автоматического заполнения ячеек первого столбца



	A	B	C	D
1	n	$\int x^n dx$	$\frac{\partial}{\partial x} x^n$	
2	1	$\frac{1}{2}x^2$	1	
3	2	$\frac{1}{3}x^3$	$2x$	
4	3	$\frac{1}{4}x^4$	$3x^2$	
5	4	$\frac{1}{5}x^5$	$4x^3$	
6	5	$\frac{1}{6}x^6$	$5x^4$	
7	6	$\frac{1}{7}x^7$	$6x^5$	
8	7	$\frac{1}{8}x^8$	$7x^6$	

Рис. 3.26. Электронная таблица с символьными данными

Затем во втором столбце в ячейку B1 вводится инертная формула `Int(x^~A1,x)`, а в ячейку B2 — исполняемая формула `int(x^~A2,x)`. После этого выделяются ячейки от B2 до B10 и исполняется команда Spreadsheet Fill Down. В результате формируется столбец с символьными значениями интегралов. Аналогично (третий этап) задается формирование столбца с символьными значениями производной от `x^n` (рекомендуем сделать это самостоятельно для закрепления навыков работы с электронными таблицами в среде Maple 7).

ПРИМЕЧАНИЕ Выше описаны лишь основы работы с электронными таблицами в среде Maple 7. Для более полного знакомства с техникой применения электронных таблиц нужно обратиться к справочной системе Maple 7.

Вставка текстовой области

Для вставки строки текстовой области служит операция Paragraph. Она создает строку без приглашения >, в которую можно вводить текст. Единственным отличием этой команды от команды Text является то, что она вставляет новую строку, не меняя статуса имеющихся строк. При вводе длинных текстов число строк ввода автоматически увеличивается.

Вставка кнопки секции

Команда Section служит для установки кнопки, указывающей начало секции (см. рис. 2.6 с такой кнопкой) и служащей для открытия/закрытия секции. Секция может состоять из различных объектов: текстовых комментариев, строк ввода, строк вывода, графиков и других секций (подсекций).

Как и в операционной системе Windows, значок «+» указывает на закрытую секцию, значок «–» — на открытую. Секции предоставляют дополнительную свободу управления документом.

Вставка кнопки подсекции

Подсекцией называют секцию, размещенную внутри другой секции. Создавая подсекции, можно строить документы со сложной древообразной структурой, напоминающей разделы книги с хорошей рубрикацией. Это может оказать большую помощь в создании электронных вариантов книг и обучающих программ в среде Maple 7.

Команда Subsection создает кнопку секции внутри уже созданной секции. Все, сказанное о секциях, распространяется и на подсекции.

Вставка гиперссылки

Еще одна возможность сделать документы более удобными в работе заключается в создании гиперссылок. Гиперссылка — это текстовая надпись, подчеркнутая снизу, при щелчке на которой Maple перейдет к сопоставленному с ней объекту. Гиперссылку можно связать со следующими объектами:

- с файлом любого документа (Worksheet:);
- с заданной страницей справочной системы (Help Topic:);
- со страницей в Интернете (URL:).

Для создания гиперссылки надо установить на место будущей ссылки маркер ввода и выполнить операцию HyperLink. При этом появится окно связывания гиперссылки с объектом, показанное на рис. 3.27.

В окне надо задать заголовок гиперссылки (в виде короткой текстовой надписи) и выбрать одно из трех положений переключателя, перечисленных выше. Если вы намерены сослаться на документ, то следует использовать кнопку просмотра Browse для поиска нужного файла. Появляющееся окно показано на рис. 3.27 справа.

Как уже говорилось, гиперссылка выглядит как надпись, подчеркнутая снизу. Активизируя ее, можно вызвать объект, связанный с ней, — в нашем случае другой документ, файл которого находится на жестком диске. Использование гиперссылки на другой документ в Maple 7 реализовано не очень удачно. Дело в том, что в этом случае исходный документ по умолчанию закрывается. Если последние изменения в нем не были сохранены, то появляется окно с предупреждением об этом. Оно показано на рис. 3.28.

Можно избежать закрытия документа, щелкнув на кнопке New Window; Maple предоставит открывающемуся документу новое окно. Это иллюстрирует рис. 3.29. В нем также показано контекстное меню гиперссылки. Это меню содержит три команды:

- Properties — вывод окна свойств гиперссылки (рис. 3.27, слева);
- Follow link — смена документа с гиперссылкой на указанный в ней документ;
- Open link — открытие отдельного окна для открываемого документа (рис. 3.29).

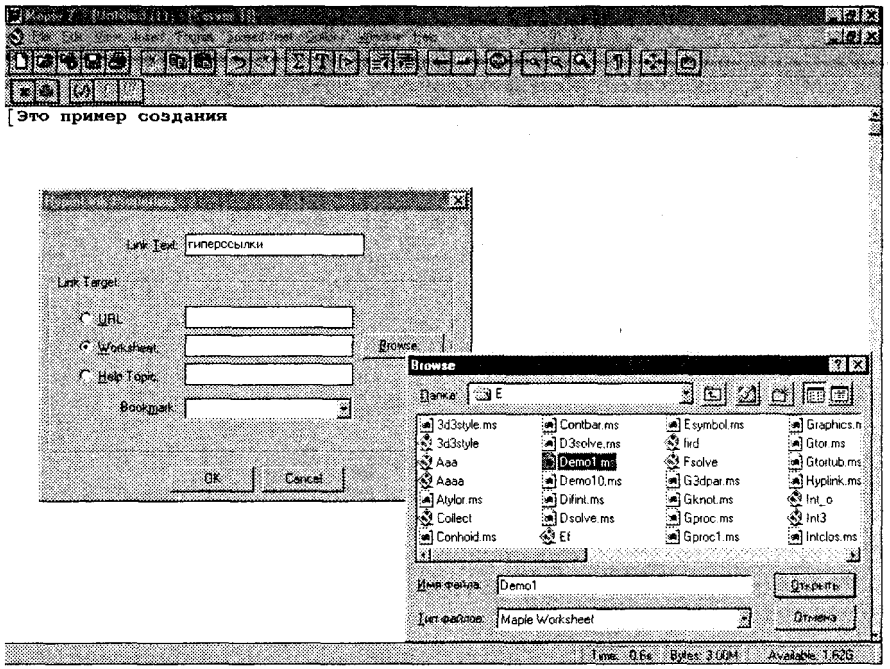


Рис. 3.27. Пример создания гиперссылки

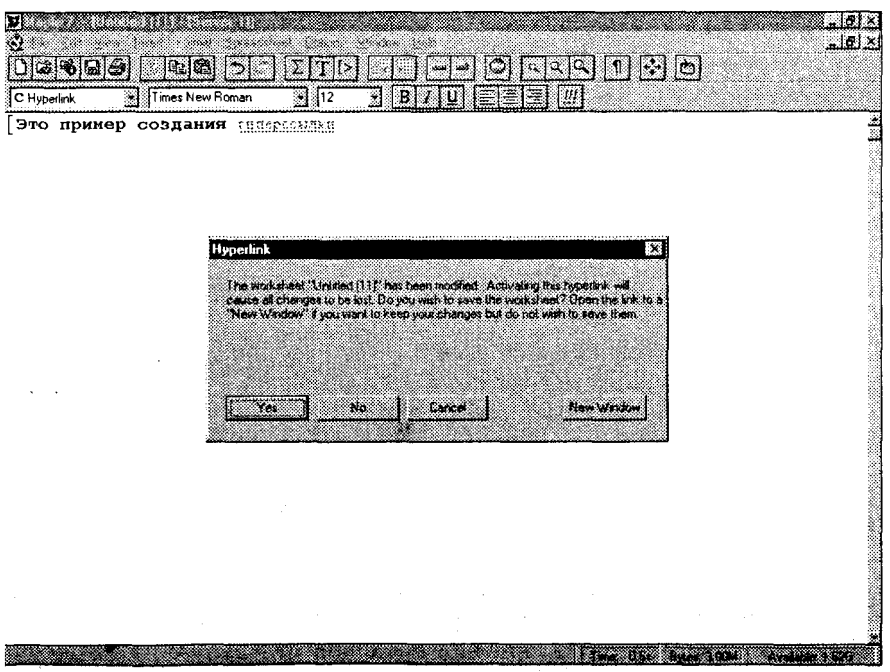


Рис. 3.28. Окно с предупреждением о закрытии документа при переходе по гиперссылке и необходимости сохранения изменений

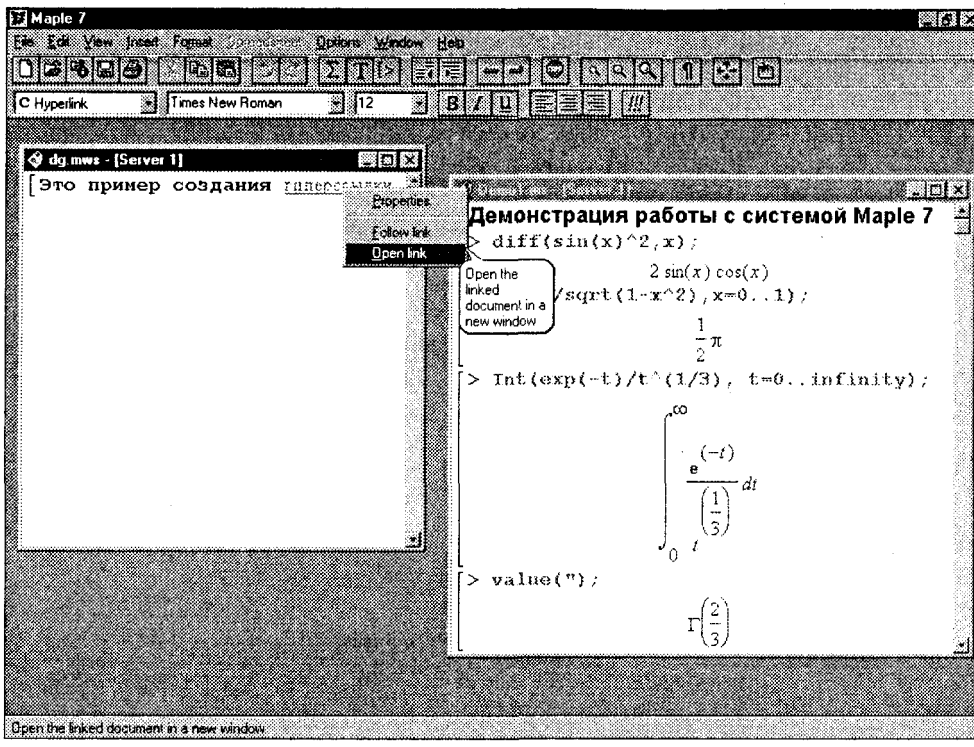


Рис. 3.29. Результат активизации гиперссылки

Гиперссылки позволяют создавать сложные структуры документов, содержащие множество объектов, вызываемых в произвольном порядке. Например, для возврата в исходный документ можно организовать обратную гиперссылку в вызываемом документе. Лучшим примером такого применения гиперссылок является справочная система Maple 7. Гиперссылки также широко используются при создании web-страниц. Maple 7 позволяет сохранять документы в виде web-страниц (формат HTML), которые без каких-либо преобразований можно публиковать в сети Интернет.

Весьма привлекательной кажется возможность организации гиперссылок на разделы справочной системы. Она позволяет создавать учебные программы со ссылками на справки системы Maple 7.

Операции форматирования

Обзор операций меню Format

Операции форматирования служат для придания отдельным объектам и документу в целом определенного стиля путем изменения как общего вида объектов, так и ряда их частных характеристик, например цвета и размера надписей,

выбранного набора шрифтов и т. д. При этом возможна подстройка под вкус любого пользователя и подготовка документов высокого полиграфического качества. *Стиль* является центральным понятием для современных документов, будь то документы текстового процессора класса Word или системы символьной математики Maple 7.

Команды форматирования в меню Format разбиты на шесть подгрупп. В первой подгруппе содержатся две команды:

- Styles — установка стилей для всех объектов;
- Page Numbers — задание параметров нумерации страниц.

Во второй подгруппе имеются три наиболее распространенные команды изменения начертания надписей:

- Italic (Ctrl+I) — задание курсивного начертания;
- Bold (Ctrl+B) — задание полужирного начертания;
- Underline (Ctrl+U) — задание подчеркнутого начертания.

Заметим, что все они дублируются кнопками на панели инструментов и горячими клавишами. При редактировании документов использование кнопок и горячих клавиш более удобно.

В третью группу попали команды выравнивания текста:

- Left Justify — по левому краю;
- Center — по центру;
- Right Justify — по правому краю.

В четвертой группе находятся следующие команды:

- Paragraph — форматирование абзаца;
- Character — форматирование символов.

И наконец, в пятой группе имеются еще две команды:

- Indent (Ctrl+.) — внедрение текущей строки в секцию;
- Outdent (Ctrl+,) — выведение текущей строки из секции.

В последнюю группу попала еще одна команда, не имеющая непосредственного отношения к форматированию символов:

- Convert to — перевод из одной метрической системы в другую.

Рассмотрим применение команд форматирования.

Установка стилей

Команда Styles является основной, поскольку позволяет задать стиль текста — определенный набор значений доступных параметров: размещение на странице, выравнивание, шрифт, начертание, цвет, размер и т. д. Единство стилей документов важно при включении их в отчеты, курсовые и дипломные проекты, диссертации и иные документы. Придерживаться определенных стилей — это

правило хорошего тона при работе как с текстовыми процессорами, так и с математическими системами. Именно поэтому ориентация на определенный стиль подготовки документов стала неотъемлемой частью пользовательского интерфейса системы Maple 7.

Операция Styles выводит диалоговое окно Style Management, в котором можно как изменить уже существующий стиль, так и создать новый (рис. 3.30).

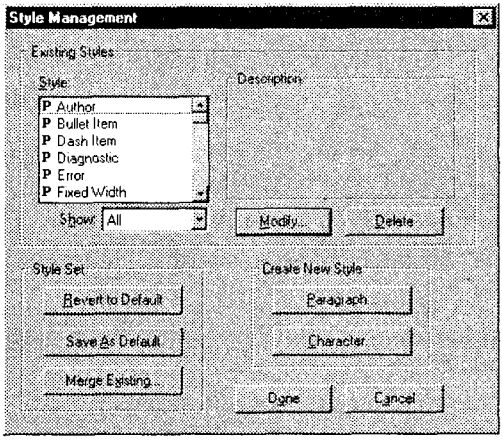


Рис. 3.30. Окно задания и модернизации стилей

В большинстве случаев пользователя Maple 7 вполне удовлетворят стили, заданные по умолчанию. Более того, надо помнить, что задание своего стиля неизбежно означает повышение риска несовместимости при обмене документами Maple 7. Поэтому без особой на то необходимости изменять стандартный стиль не рекомендуется.

Однако бывают принципиальные обстоятельства, когда изменение стиля необходимо. Например, при выводе титульных надписей для графиков Maple 7 используют стандартный англоязычный шрифт Courier New. При вводе русскоязычных надписей Maple 7 воспринимает их вывод как ошибку, поскольку набор символов при вводе не соответствует набору символов, заданному в стиле. Таким образом, для организации вывода русскоязычных надписей необходимо сменить набор символов (шрифт), то есть изменить стиль.

Покажем, как это делается. Вначале в списке стилей надо выбрать наименование Title, после чего нажать кнопку Modify. Появится окно Paragraph Style с параметрами стиля Title (рис. 3.31).

В этом окне надо нажать кнопку Font. Появится новое окно (оно также показано на рис. 3.31) для выбора шрифта. В нашем случае достаточно заменить шрифт по умолчанию Courier New на шрифт Courier New Cyr. Он содержит символы кириллицы, то есть буквы русского языка. После этого задание титульных надписей для графиков на русском языке перестает быть проблемой.

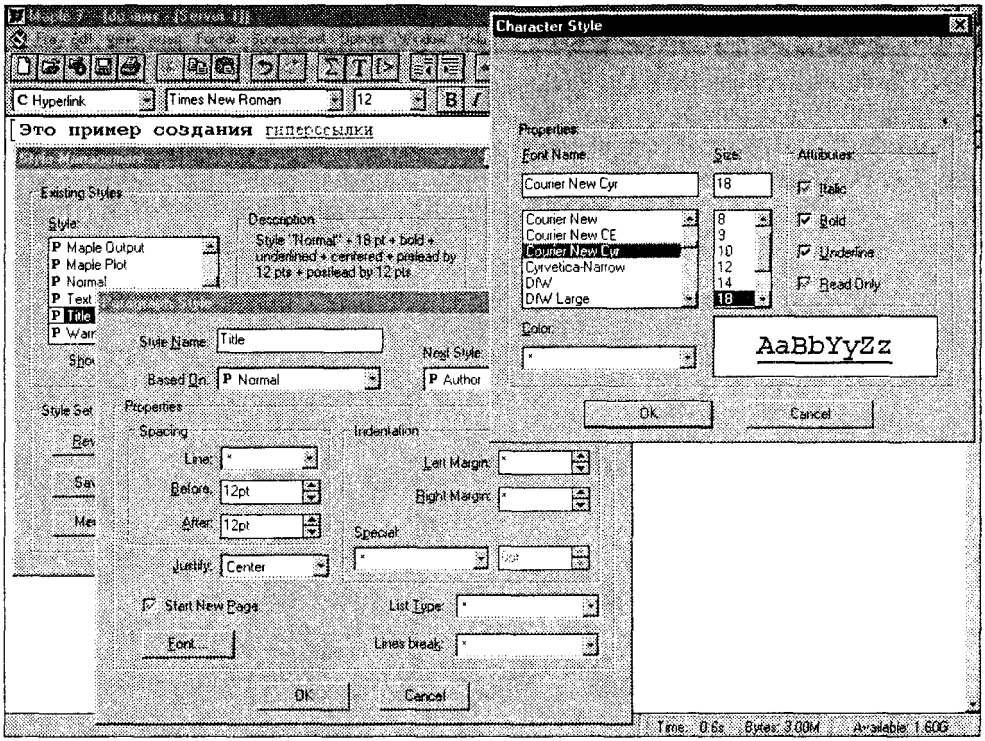


Рис. 3.31. Окна установки параметров абзацного стиля для абзаца и для выбора шрифта

Разумеется, в понятие стиля входит не только шрифт, но и размер, начертание, цвет и т. д. Все это можно настраивать, используя описанные выше окна. При необходимости изменения стиля можно записать его в специальный файл, используя кнопку Save As Default. При этом измененный стиль становится принятым по умолчанию. Кнопка Revert As Default позволяет вернуться к типовым стилям, которые заданы разработчиками системы. Если стили подверглись небольшим изменениям, то можно сохранить их с помощью кнопки Merge Existing.

Форматирование абзацев

Для форматирования абзацев служит команда Paragraph. Она сразу выводит окно установки параметров абзаца. Это окно было показано на рис. 3.31. Правила работы с этим окном уже были описаны выше, так что нет смысла их повторять.

Форматирование символов

Команда Character открывает окно задания стиля символов: шрифта, размера, начертания и цвета.

Операция внедрения ячеек в секцию

Ранее описывалась операция вставки кнопки секции. При выполнения вставки сама секция еще пуста и ее надо заполнять. Команда Indent (или комбинация клавиш Ctrl+.) позволяет оформить в виде секции уже введенные ячейки. На рис. 3.32 показано такое оформление для ячейки документа с вычислением интеграла. Кнопка при этом имеет знак «минус», что указывает на ее открытое состояние.

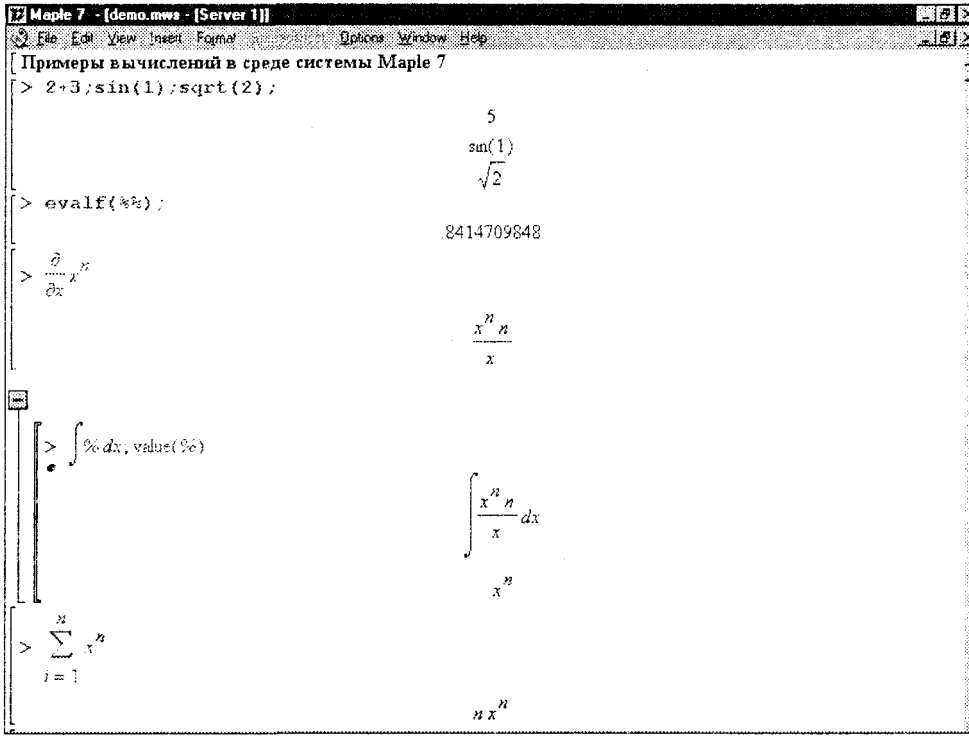


Рис. 3.32. Применение операции Indent для одной из ячеек документа

Если щелкнуть на кнопке со знаком «минус», то секция свернется и вместо нее останется только кнопка со знаком «плюс» (рис. 3.33).

Операция выведения ячеек из секции

Команда Outdent (или комбинация клавиш Ctrl+,) отменяет оформление ячейки в виде секции. Она действует в том случае, если маркер ввода стоит внутри секции.

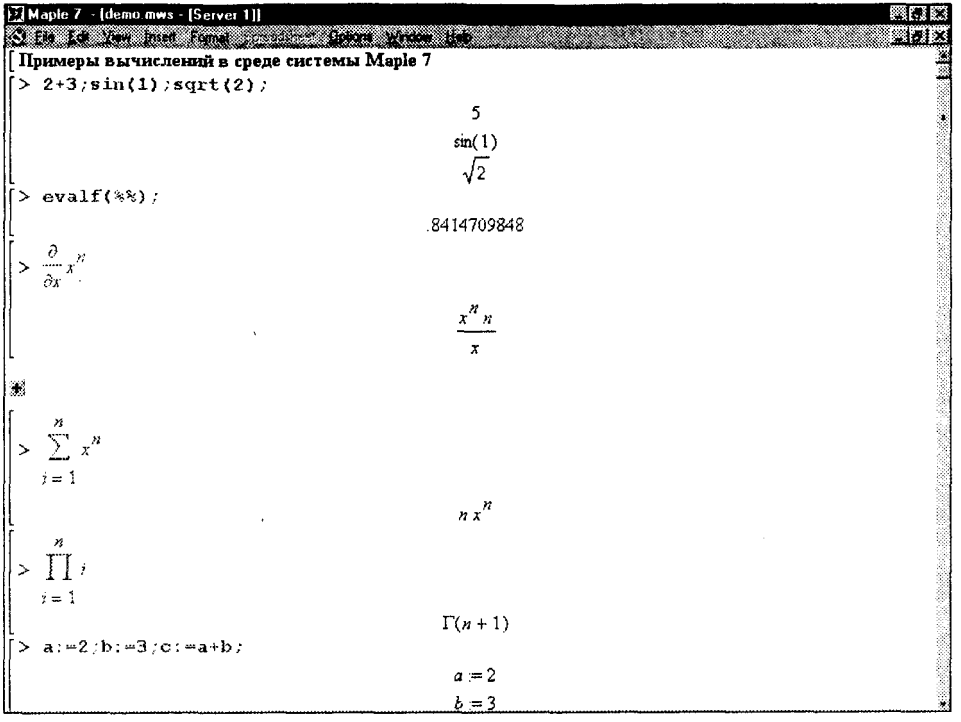


Рис. 3.33. *Пример документа с закрытой секцией*

Работа с объектами

Хотя данная книга посвящена одной Maple 7, надо особо отметить, что эта программа способна взаимодействовать с рядом других программ, например с текстовым процессором Word, табличным процессорам Excel и даже с другими системами компьютерной математики, например, MATLAB. Это может быть копирование через буфер обмена или связь с применением механизма OLE.

В роли объектов могут выступать ячейки из данного или других документов, текстовые фрагменты и файлы, рисунки в различных форматах и т. д. Работа с объектами существенно расширяет возможности пользователя по части создания полноценных и удобных в работе документов.

Вставка объектов

Для организации вставки объекта используется команда Insert ▸ Object. Она выводит окно со списком тех приложений, с которыми возможна связь с применением механизма OLE. Это окно показано на рис. 3.34. Maple 7 использует стан-

дартное окно вставки объектов из операционной системы Windows, поэтому если она русифицирована, то окно имеет русскоязычные надписи.

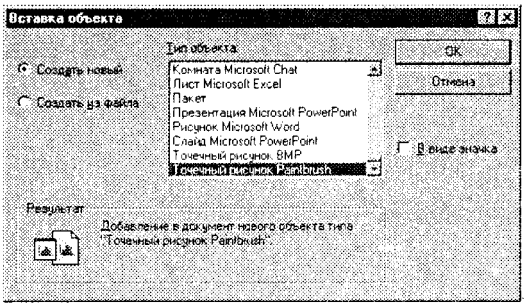


Рис. 3.34. Окно вставки объекта

Если оставить переключатель в первом положении, то объект может быть создан заново с помощью подходящего приложения, например редактора Paint. Этот процесс показан на рис. 3.35.

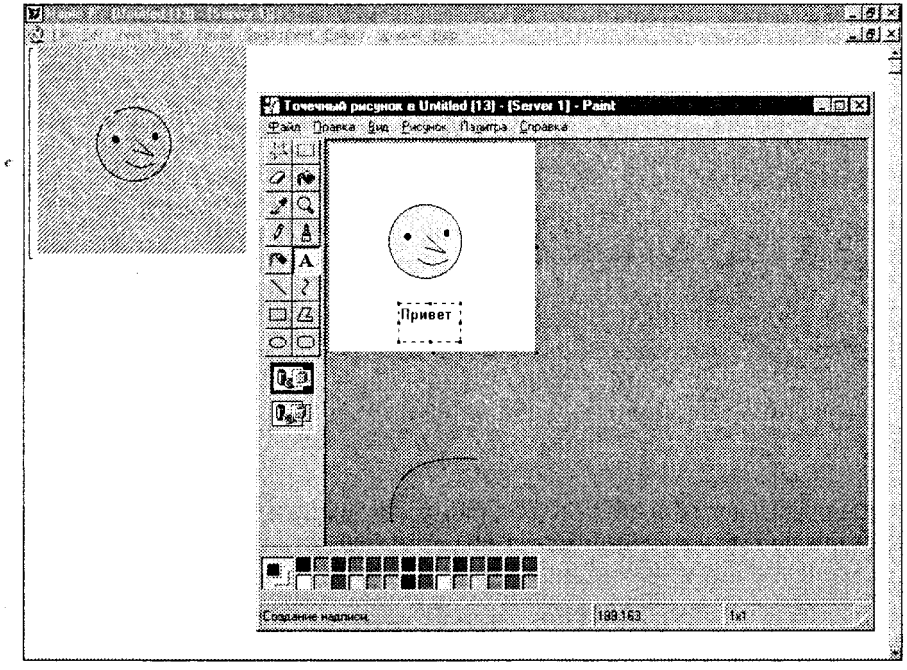


Рис. 3.35. Подготовка объекта в графическом редакторе Paint

После того как объект готов, достаточно выйти из приложения, в котором он создавался, — в нашем случае из редактора Paint. Для этого в меню редактора имеется команда File ▸ Exit. Окно Paint исчезнет, а сам объект появится в той

ячейке Maple 7 (рис. 3.36), в которой был установлен маркер ввода в момент дачи команды вставки объекта.

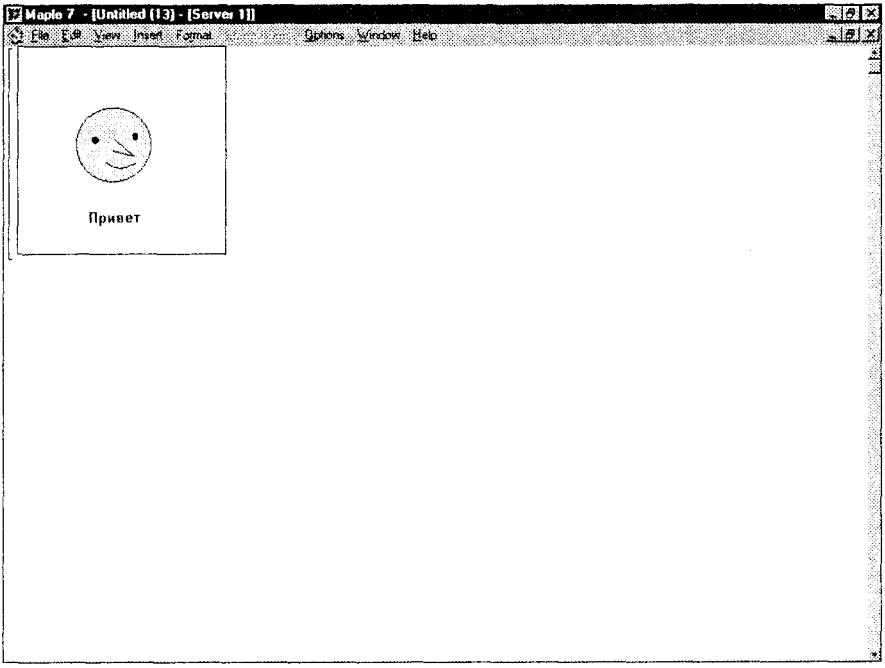


Рис. 3.36. *Пример документа со вставленным объектом-рисунком*

Вместо рисунка (или другого объекта) можно поместить в документе Maple 7 значок — гипермедиа-ссылку. Для этого в окне на рис. 3.34 надо установить флажок В виде значка. Щелчок на вставленном в документ значке вызовет появление объекта. Этот способ удобен, когда объекты, например рисунки, имеют большой размер и их постоянное присутствие на экране нецелесообразно.

Необязательно каждый раз создавать объект с нуля — можно загрузить его из файла, в этом случае следует установить положение переключателя Создать из файла. Для поиска нужного файла пригодится кнопка Обзор. После выбора файла надо решить, какой вид будет иметь значок связи с файлом — вид стандартного значка или уменьшенной копии изображения.

Редактирование вставленного объекта

Если объект выделен (как обычно, щелчком мыши), то команда Edit ▸ Object становится активной и может даже модифицироваться в зависимости от вида объекта, с которым установлена связь. Например, если объектом является .pcx-файл, то эта строка меню открывает подменю с тремя командами:

- Edit — редактировать объект;

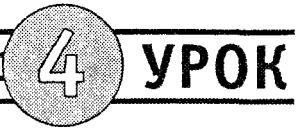
- Open — открыть приложение, связанное с объектом;
- Print — распечатать объект.

Редактирование объекта производится в среде того приложения, с которым объект связан. Естественно, что изменение файла повлечет за собой отражение в документе его измененного варианта. Следует отметить, что в некоторых копиях программы Maple 7 операция Object не работает. Однако сохраняется возможность редактирования объекта в создавшей его программе после двойного щелчка на нем.

Что нового мы узнали?

В этом уроке мы научились:

- Создавать новый документ.
- Загружать ранее созданные документы.
- Работать с файлами документов.
- Редактировать документы.
- Печатать документы.
- Осуществлять операции вставки.
- Выполнять операции форматирования.
- Вставлять в Maple 7 объекты, созданные другими программами.
- Работать с объектами.



Управление интерфейсом пользователя

- Управление видом интерфейса и документа
- Управление показом областей секций
- Работа с параметрами Maple
- Работа с окнами

Управление видом интерфейса и документа

Меню View

Для управления видом интерфейса и документа служит меню View. Оно содержит ряд флажков и несколько команд управления общим видом программы (рис. 4.1). Установленные флажки, управляющие показом элементов интерфейса, распространяют свое действие на все открытые документы. При выходе из системы (командой Exit) все установки сохраняются, так что при новом запуске системы внешний вид интерфейса будет определяться именно ими.

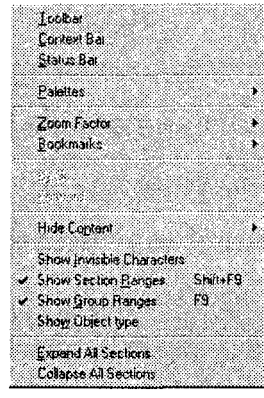


Рис. 4.1. Меню View

Команды меню View разбиты на несколько групп. Первая группа содержит флажки, относящиеся к управлению показом главной и контекстной панелей инструментов, а также строки состояния системы:

- Toolbar — управление показом панели инструментов;
- Context Bar — управление показом контекстной панели;
- Status Line — управление показом строки состояния.

Вторая группа задает показ палитр и представлена одной командой, открывающей подменю управления показом палитр Palettes, содержащей флажки:

- Symbol Palette — палитра символов;
- Expression Palette — палитра выражений;
- Matrix Palette — палитра шаблонов матриц;
- Vector Palette — палитра шаблонов векторов.

И команды:

- Show All Palettes — показать все палитры;
- Hide All Palettes — скрыть все палитры.

Третья группа в меню View задает масштаб отображения документа на экране и установку закладок (bookmarks):

- Zoom Factor — задание масштаба просмотра документа;
- Bookmarks — установка закладок.

Четвертая группа обеспечивает переходы по гиперссылкам:

- Back — переход обратно по последней пройденной гиперссылке;
- Forward — команда, обратная команде Back.

Данные команды аналогичны командам браузера и управляются историей переходов по гиперссылкам. Поэтому, если вы не использовали ни одной гиперссылки, данные команды будут недоступны. В основном данные команды и соответствующие кнопки панели инструментов необходимы при работе со справкой Maple.

Пятая группа представлена командой Hide content, открывающей подменю скрытия элементов документа:

- Hide Spreadsheets — скрыть электронные таблицы;
- Hide Input — скрыть ввод;
- Hide Output — скрыть вывод;
- Hide Graphics — скрыть графические объекты

Шестая группа параметров управляет показом некоторых объектов документа:

- Show Invisible Characters — показ непечатаемых символов;
- Show Section Ranges (Shift+F9) — показ областей секций;
- Show Group Ranges — показ областей групп;
- Show OLE type — показ объектов OLE.

В седьмой группе содержатся следующие команды:

- Expand All Sections — раскрыть все секции;
- Collapse All Sections — свернуть все секции.

Действие всех перечисленных команд более подробно описано ниже. Рекомендуется поэкспериментировать с командами — когда вы уясните их действие, вы сможете настроить интерфейс Maple на свой вкус.

Управление показом панели инструментов (Toolbar)

Панель инструментов (Toolbar) служит для быстрого управления системой без обращения к командам меню. Она позволяет вызывать наиболее часто исполь-

зуемые команды нажатием кнопки. Назначение всех кнопок этой панели было описано выше и представлено на рис. 1.21.

Несмотря на удобства, предоставляемые панелью инструментов, в ряде случаев она не нужна. Например, для ввода исходных данных и их редактирования достаточно иметь на экране только контекстную панель. Панель инструментов полезна при общей отладке документов, открытии нового документа, загрузке имеющегося документа, записи документа на диск, печати документа и т. д. В ряде случаев она просто занимает часть места, нужного для лучшего обзора документа.

Флажок Toolbar управляет показом панели инструментов.

Управление показом контекстной панели

Панель Context Bar служит для размещения кнопок быстрого доступа к операциям с текущим объектом.

Флажок Context Bar задает отображение контекстной панели на экране. Поскольку панель форматирования весьма удобна для оперативной работы, единственным мотивом временного ее удаления является необходимость высвободить больше места на экране монитора для работы с документом.

Управление показом строки состояния

Строка состояния внизу экрана, как отмечалось, выводит контекстно-зависимую информацию о состоянии программы в данный момент времени. В ряде случаев ее можно считать элементом контекстно-зависимой справки. Флажок Status Bar служит для управления показом строки состояния.

Вывод палитр математических символов

Палитры математических символов, впервые введенные в версии Maple V R5, являются очень удобным средством для облегчения набора математических выражений начинающим пользователям. Они выводятся на экран командой Palettes и показаны на рис. 1.11. Всего имеются четыре палитры:

- Symbol — палитра ввода греческих символов и констант;
- Expression — палитра ввода операторов и выражений;
- Matrix — палитра ввода матриц;
- Vector – новая (для Maple 7) палитра ввода векторов.

Ввод осуществляется на место маркера ввода. Щелчок на кнопке палитры — символ или его аналог окажется в точке ввода. Параметры, которые необходимо указать, при математической нотации выражений имеют вид вопросительного знака, а при Maple-нотации — %?.

Итак, в строке ввода введенный оператор или символ не обязательно повторит оператор или символ палитры. Это хорошо видно из рис. 4.2, на котором показан ввод с помощью палитры выражений шаблона определенного интеграла и его последовательное заполнение. Для наглядности заполнение каждого параметра дано в новой строке ввода.

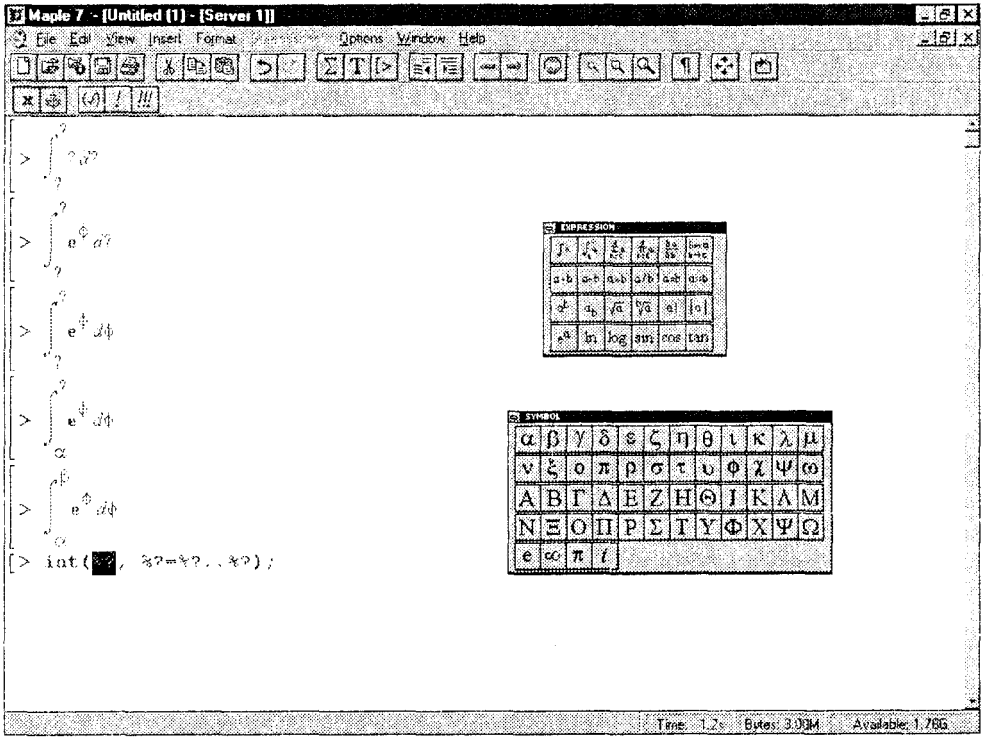


Рис. 4.2. Пример ввода шаблона определенного интеграла из палитры выражений

Установка масштаба отображения документа

Команда Zoom Factor задает масштаб отображения текущего документа. Он задается в процентах относительно масштаба просмотра, заданного по умолчанию, — он принят за 100 %. Команда выводит подменю с перечнем возможных масштабов просмотра. На рис. 4.3 показано, как будет выглядеть документ при установке масштаба просмотра в 150 %. На этом рисунке показано подменю выбора масштаба в открытом виде.

Нетрудно заметить, что все символы документа увеличились при этом в размерах в полтора раза. Эта возможность, безусловно, полезна для пользователей с ослабленным зрением или для учителей, желающих показать примеры решения задач в среде Maple крупным планом для группы учащихся.

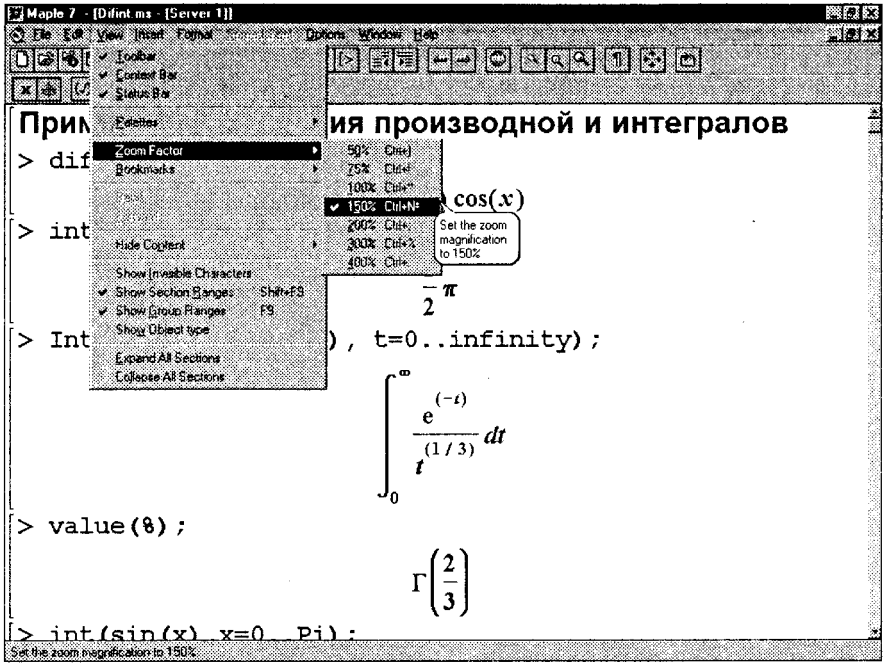


Рис. 4.3. Вид документа при масштабе просмотра в 150 %

Установка закладок

При работе с большими документами, как и при чтении книг, полезно устанавливать специальные маркеры — закладки. Для установки такой закладки достаточно поместить в нужное место документа маркер ввода и выполнить команду View ▸ Bookmarks ▸ Edit Bookmarks. Она выводит окно добавления и модификации закладки (рис. 4.4). Кнопка OK вводит новую закладку, а кнопка Cancel позволяет отказаться от выполнения данной операции.

Если теперь вновь посмотреть подменю Bookmarks, в нем появится строка с созданной закладкой (рис. 4.5).

В нашем случае в списке представлены две ранее созданные закладки с именами «Это определенный интеграл» и «А это значение определенного интеграла».

При переходе по закладке строка, в которой установлена закладка, размещается в верхней части окна редактирования и маркер ввода устанавливается в место, которое определено закладкой. Таким образом, закладки — эффективное средство для быстрого перехода в отмеченные места документа. Однако надо помнить, что в отличие от закладок в книгах закладки в документах Maple 7 не видны — они имеются лишь в списке закладок, открываемом командой Bookmarks (рис. 4.5). Удалить созданную закладку не так-то просто: вам нужно перейти по закладке, дать команду View ▸ Bookmarks ▸ Edit Bookmarks и удалить текст в поле имени. Теперь нажатие кнопки OK приведет к окну с предупреждением об удалении закладки. Нажмите OK в обоих окнах — и закладка исчезнет.

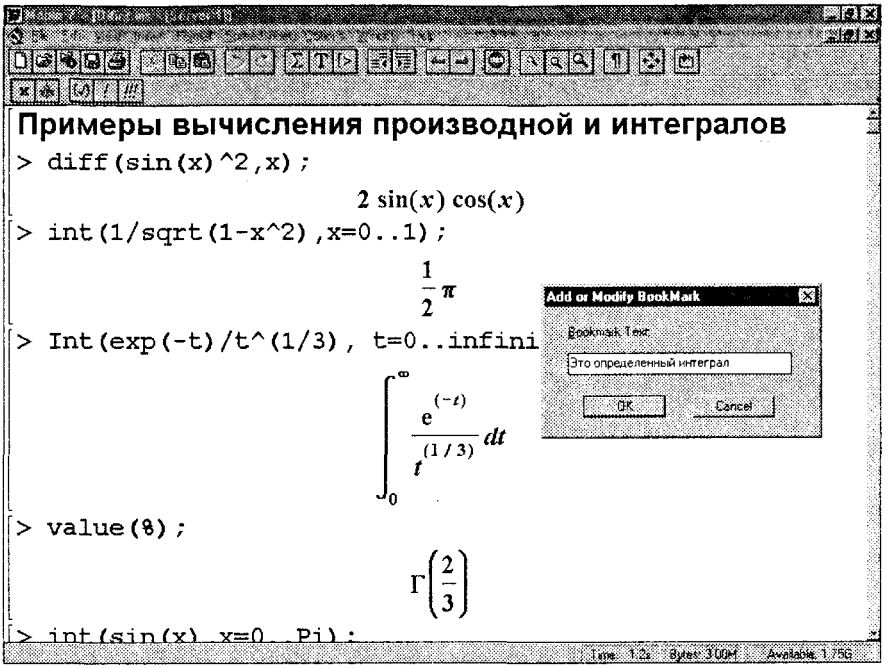


Рис. 4.4. Пример создания и редактирования закладки

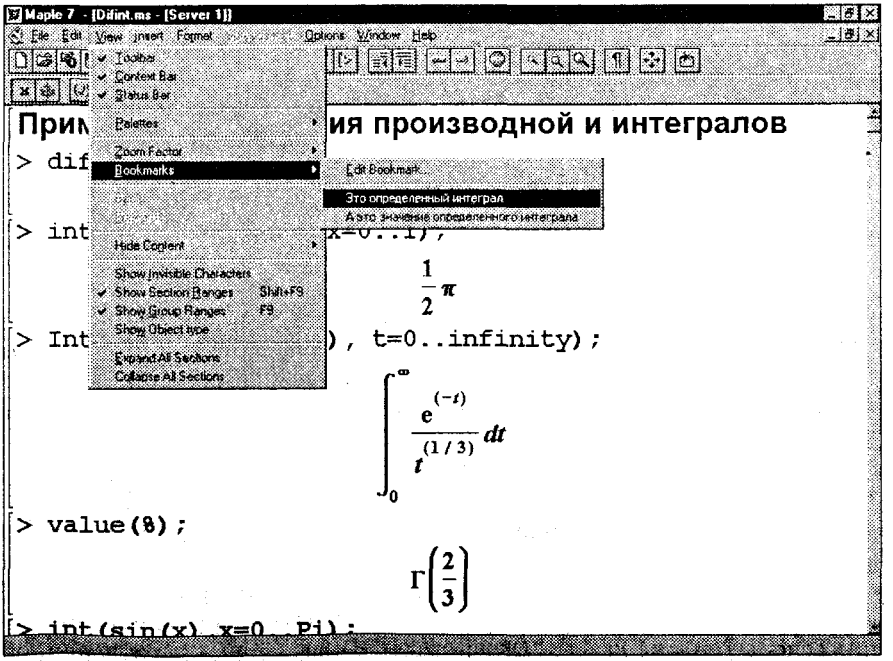


Рис. 4.5. Просмотр закладок

Управление показом компонентов документа

Документы Maple 7 состоят из ряда основных элементов. Это ячейки (строки) ввода и вывода, графики и электронные таблицы. При подготовке различных электронных книг, учебников и статей возникает необходимость скрыть тот или иной компонент документа. Например, педагог может захотеть скрыть от учащихся строки ввода, чтобы они могли мысленно представить функции, отображаемые показанными в документе графиками. Или, наоборот, скрыть графики, чтобы учащиеся назвали их особенности исходя из записи функций. Все эти возможности и обеспечивает команда View ▸ Hide Content.

Для примера на рис. 4.6 показан документ, у которого скрыты все строки ввода и оставлены только строки вывода.

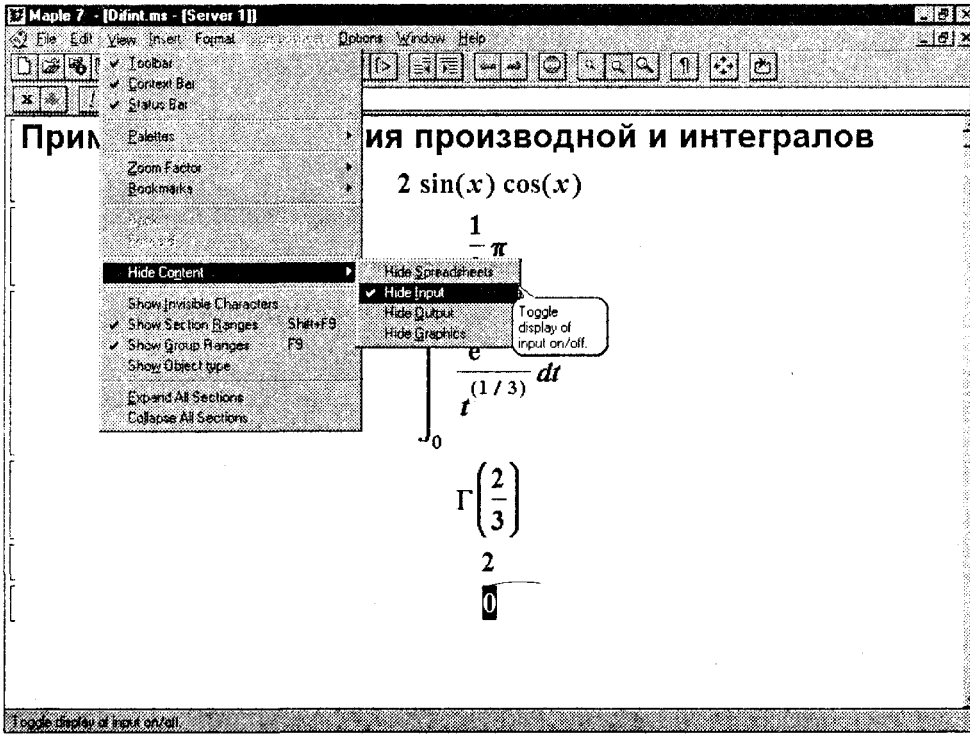


Рис. 4.6. Документ со скрытыми ячейками ввода

Разумеется, можно вернуть показ строк ввода, просто повторно дав эту команду Hide Input в подменю команды View ▸ Hide Content ▸ Hide Input (рис. 4.6). Со скрытием других компонентов документов читатель может разобраться самостоятельно.

Управление показом непечатаемых символов

В любом документе незримо присутствуют различные непечатаемые символы, например управляющие символы перевода строки или пробелы. Это хорошо известно читателям, работающим с текстовым процессором Word. Иногда полезно вывести эти символы — например, если вам неясно, сколько пробелов стоит между какими-то словами.

Для вывода непечатаемых символов служит команда-флажок Show Invisible Characters (рис. 4.7). Ее можно также дать, нажав кнопку на панели инструментов с изображением управляющего символа перевода строки — «¶». Этот символ, кстати, и является одним из наиболее часто встречающихся управляющих непечатаемых символов.

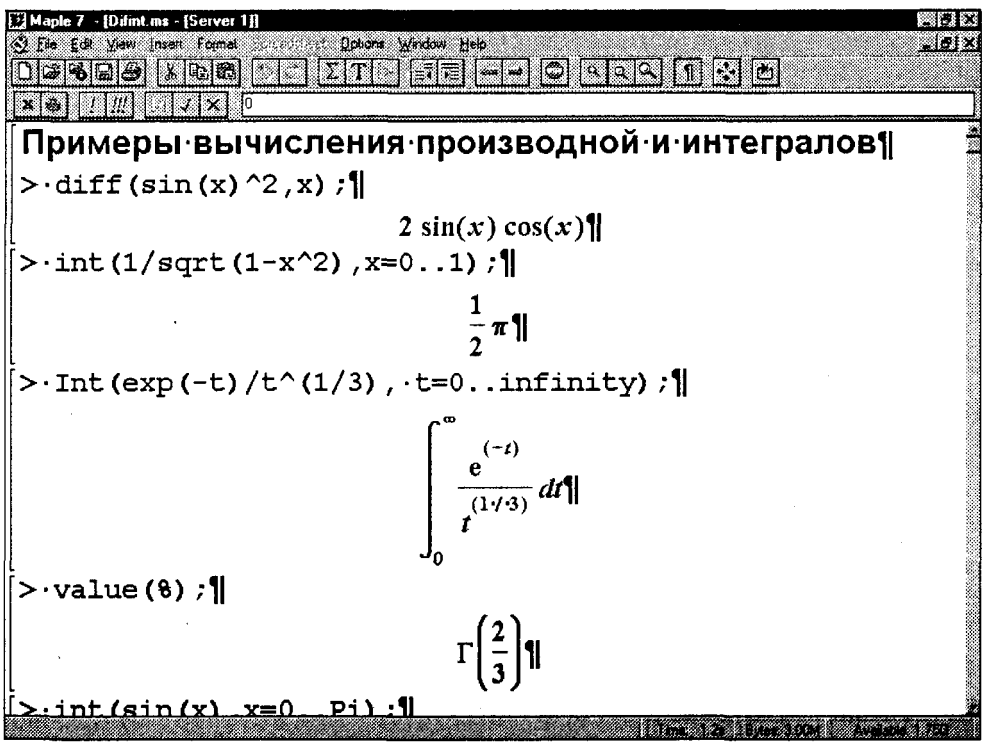


Рис. 4.7. Вид документа с выводом непечатаемых управляющих символов

Помимо символа перевода строки вы увидите множество символов пробела в виде точки на середине строки. По числу точек между словами можно судить о том, сколько пробелов установлено между ними. Проявятся также табуляции и другие управляющие операции, встречающиеся в документах Maple 7.

Управление показом областей секций

Понятие о секциях и подсекциях

Как уже отмечалось, документ Maple 7 состоит из отдельных ячеек (в оригинале — групп, groups). Они выделяются слева длинными тонкими квадратными скобками. Есть еще один способ выделения ячеек — объединение их в секции. Секция начинается с кнопки со знаком «плюс» или «минус» (рис. 4.8), управляющей ее состоянием: открытым или закрытым.

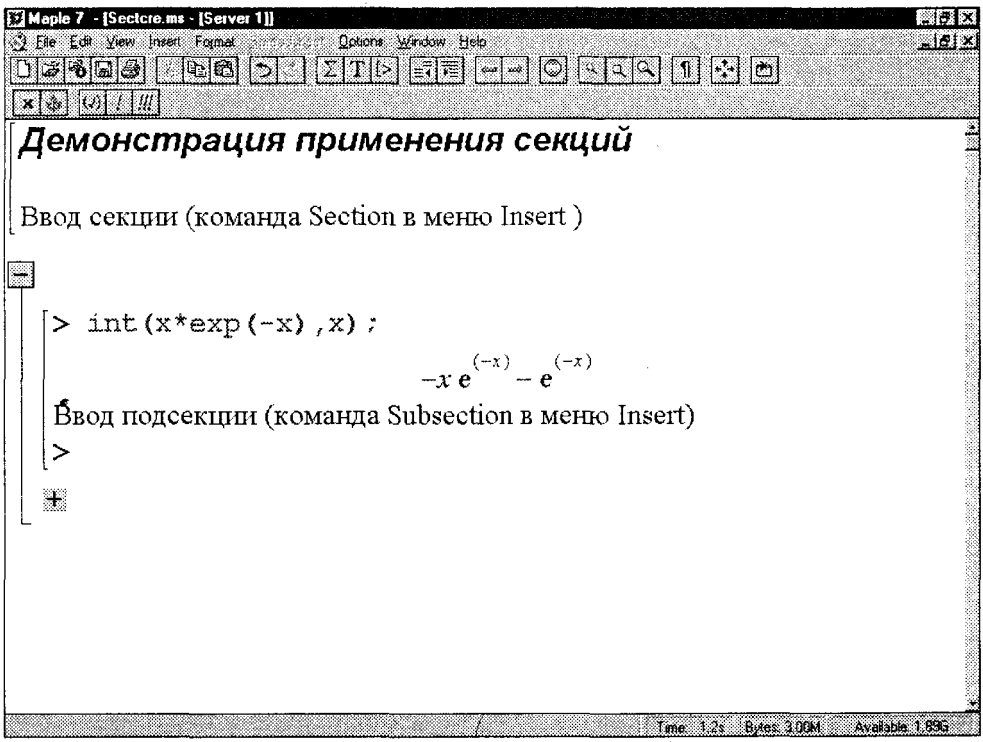


Рис. 4.8. Пример документа с открытой секцией и закрытой подсекцией

Для создания секции или подсекции необходимо дать команду Insert ▸ Section (или Insert ▸ Subsection), что указано на рис. 4.8. Секция выделяется вертикальной одиночной линией, а подсекция — двойной линией.

Секции и подсекции выгодно отличаются от ячеек тем, что они могут быть при необходимости закрыты и представлены только кнопкой со знаком «плюс». Таким образом, в секции удобно помещать различные вспомогательные вычисления, которые не стоит обозревать постоянно. Хотите посмотреть на них — нажмите кнопку, и содержимое секции появится под ней с выделением вертикальной чертой (рис. 4.9).

Секции весьма удобно применять и для подготовки документов в форме электронных книг. В этом случае секции снабжаются текстовыми заголовками и, по существу, являются главами книги, а подсекции — параграфами. Этот способ представления документов нашел широкое применение в сети Интернет, где он (наряду с гипертекстовыми и гипермедиа-ссылками) используется при создании web-страниц. Версия Maple 7 позволяет готовить и сохранять на диске документы в формате HTML, что позволяет напрямую готовить web-страницы.

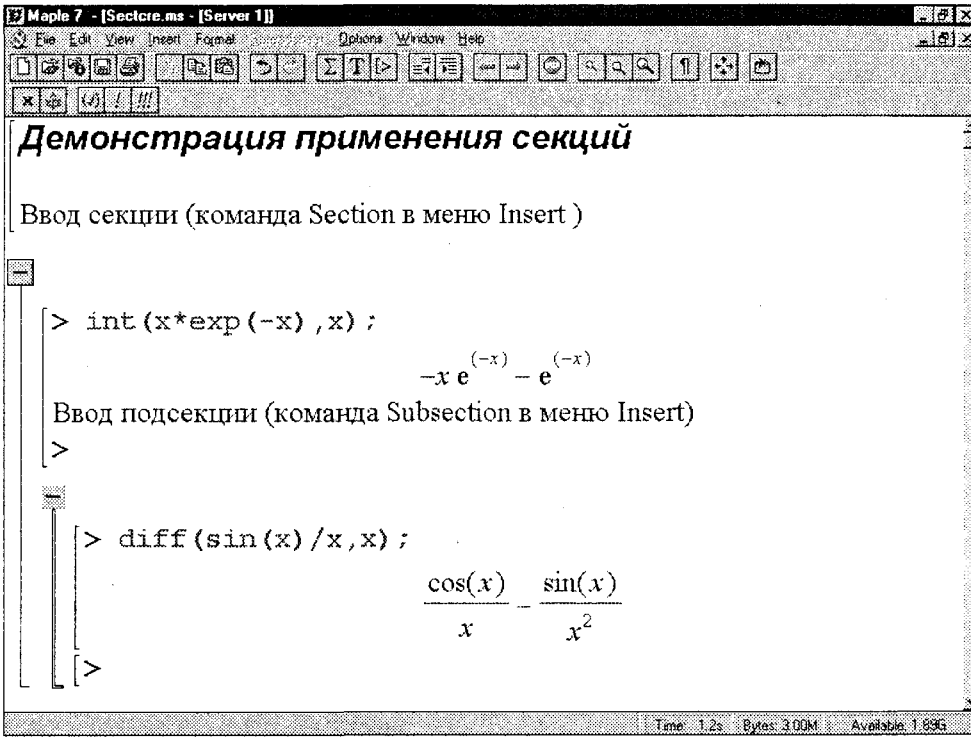


Рис. 4.9. Пример документа с рис. 4.8 с открытой подсекцией

Управление показом областей секций

Итак, обычно секции и подсекции выделяются вертикальными линиями, заканчивающимися короткими горизонтальными штрихами. Это позволяет судить о размерах области экрана, представляющей секцию, особенно если секция большая и целиком не помещается на экране.

Тем не менее в окончательно отлаженном документе необходимость в применении линии показа области секции или подсекции отпадает. С помощью флажка **Show Section Ranges** можно управлять показом линий выделения областей секций и подсекций. Если этот флажок установлен, то линии показа областей секций и подсекций видны, как на рис. 4.9. Если его снять, то эти линии исчезнут (рис. 4.10).

Обратите внимание, что линии выделения групп (ячеек) остаются.

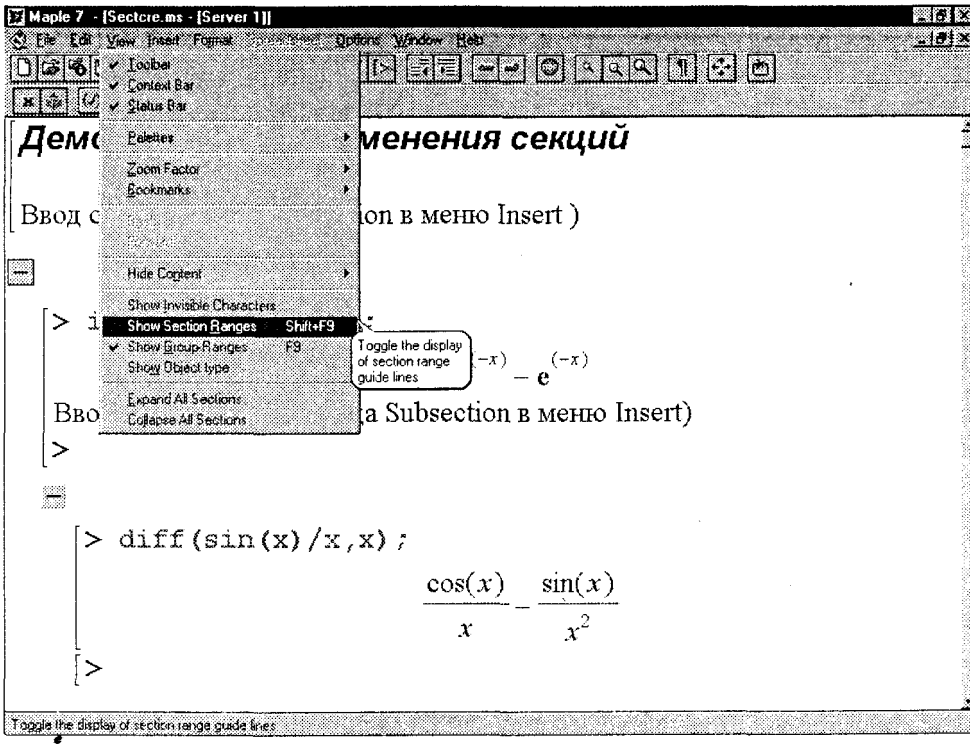


Рис. 4.10. Вид документа с рис. 4.9 при снятии показа областей секции и подсекции

Управление показом областей ячеек (Show Group Ranges)

Команда Show Group Ranges служит для управления показом областей ячеек (групп), то есть длинных квадратных скобок, обрамляющих ячейки слева. Если флажок установлен, то линии показа областей ячеек видны. Если его снять, то эти линии исчезают, как показано на рис. 4.11, где убраны линии выделения как секции и подсекции, так и ячеек.

Убирать линии показа областей ячеек целесообразно, если документ уже отлажен. В таком документе необходимость показа областей выделения ячеек, секций и подсекций отсутствует, хотя иногда их разумно оставлять, так как они способствуют систематизации содержимого документов.

Закрытие всех секций

Все секции и подсекции документа можно закрыть командой Collapse All Sections. К примеру, если применить ее к документу, показанному на рис. 4.9, то документ примет вид, показанный на рис. 4.12.

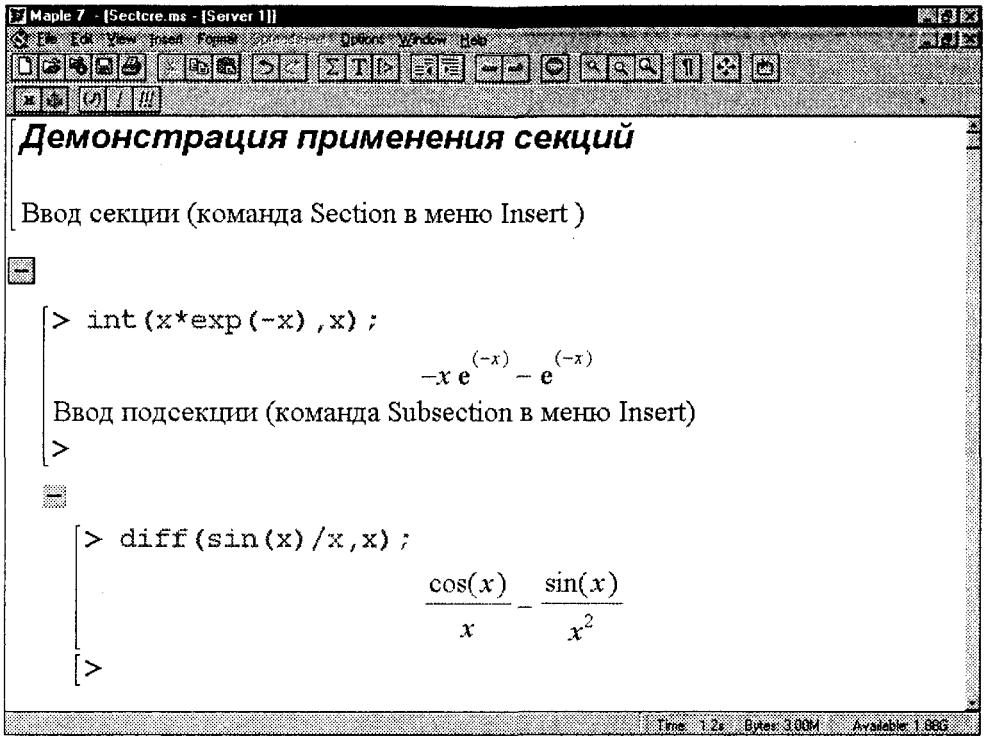


Рис. 4.11. Документ с рис. 4.10 после удаления линий показа областей ячеек

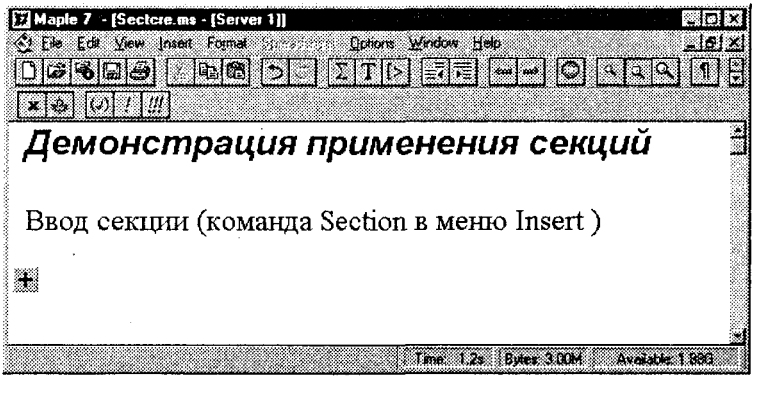


Рис. 4.12. Документ с рис. 4.9 при закрытии всех секций

Документ с закрытыми секциями (и подсекциями) занимает на экране (и при распечатке принтером) минимальное место. При этом, естественно, содержимого секций и подсекций не видно.

Раскрытие всех секций

Для раскрытия всех секций служит команда Expand All Sections. Она открывает и все подсекции. К примеру, если применить эту операцию к документу, показанному на рис. 4.12, то он вернется к исходному виду (рис. 4.9).

Таким образом, Maple 7 имеет достаточно простые, но в то же время полные средства по управлению видом пользовательского интерфейса (окна) системы и видом имеющегося в нем документа. Это позволяет пользователю настраивать интерфейс и вид документа в соответствии со своими привычками, обеспечивая комфортную работу с программой.

Работа с параметрами Maple 7

Меню Options

Помимо ряда уже рассмотренных команд в меню Options сосредоточены средства для установки некоторых глобальных параметров ввода и вывода документов. Их число в Maple 7 существенно увеличено.

Первая группа команд содержит две команды:

- Replace Output — управляет характером вывода;
- Insert Mode — устанавливает режим вставки при вводе.

Во второй группе всего одна команда Browser — задание броузера для перехода по URL.

Третья группа также представлена одной командой:

- Export — параметры экспорта документов.

Четвертая группа содержит 6 команд:

- Input Display — управление показом выражений в строке ввода;
- Output Display — управление показом результатов вычислений;
- Assumed Variables — контроль за предполагаемыми переменными;
- Plot Display — управление отображением графиков;
- Display 2D-legends — управление показом подписей обозначений двумерной графики;
- Print Quality — управление качеством печати.

Пятая и шестая группы содержат по одной команде:

- Palette Size — управление размером палитр;
- AutoSave — управление автоматическим сохранением документа.

С помощью этих параметров можно настроить систему на наиболее приемлемые формы вывода результатов вычислений без задания специальных команд в документе. Однако последние могут отменять параметры, заданные с помощью меню.

Управление выводом

Команда Replace Output задает вывод результатов вычислений, заданных в ячейке, в одно и тоже место. Это означает, что если входные данные меняются, то при установленном флажке каждый последующий результат будет замещать предыдущий. Если же флажок снят, то каждый новый результат будет помещаться в новое место, то есть в документе будут выведены подряд (сверху вниз) все результаты вычислений.

Поясним на примере. Допустим, в какой-то строке ввода мы задаем вычисляемое выражение 2+3:

```
> 2+3;
5
```

Результат (в данном случае в Maple-нотации) появляется снизу. Теперь в той же строке ввода вычислим 3+4, а затем 4+5. Получим:

```
> 4+5;
9
```

Мы заметим, что результат появится на месте прежнего, и в конечном счете мы будем иметь в качестве результата число 9. Теперь снимем флажок Replace Output и проделаем все те же вычисления в новой строке ввода. Получим следующее:

```
> 4+5;
5

7

9
```

Здесь видны уже три ячейки вывода. Не следует следует устанавливать флажок Replace Output в том случае, когда желательно знать суть промежуточных преобразований и самих исходных выражений, поскольку они (а возможно, и предшествующие результаты преобразований) при ее использовании исчезают.

Установка режима вставки новой ячейки

Этот флажок, будучи установленным, обеспечивает при нажатии клавиши Enter ввод новой пустой ячейки. Если флажок снят, то такая вставка не осуществляется.

Если работа с системой происходит в форме простейшего диалога, по типу «задать вопрос — получить ответ», то рекомендуется установить режим вставки новой ячейки. При этом по окончании вычислений в последней ячейке тут же появляется новая пустая ячейка для последующих вычислений.

Задание браузера

Для открытия документов web-страниц (по URL-адресу) Maple 7 использует какой-либо из установленных на компьютере браузеров Интернета. Для этого необходимо указать путь к браузеру с помощью команды Browser. Она открывает окно, показанное на рис. 4.13.

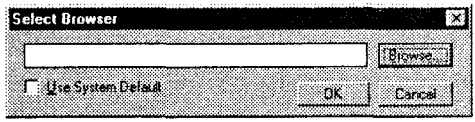


Рис. 4.13. Окно задания браузера Интернета

В этом окне можно установить флажок Use System Default (использовать браузер, заданный в операционной системе по умолчанию). В противном случае путь к браузеру надо указать явно, введя его в поле или отыскав его с помощью кнопки Browse.

Параметры экспорта документов

Команда Export открывает очень простое окно установки параметров экспорта, показанное на рис. 4.14.

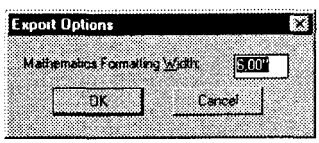


Рис. 4.14. Окно команды Export

Установка параметров представления строк ввода

Команда Input Display выводит подменю, позволяющее выбрать режим представления выражений в строке ввода:

- Maple Notation — ввод выражений в Maple-нотации (в строку);
- Standard Math Notation — ввод выражений в обычном математическом виде (его признаком является появление вопросительного знака в строке ввода).

В качестве примера ниже даны две формы задания ввода двойного интеграла с помощью палитры выражений:

```
> int(int(%?, %?=%?..%?), %?=%?..%?);
```

$$> \int_{?}^{?}\int_{?}^{?} ?\, d?\, d?$$

Верхняя строка соответствует Maple-нотации, а нижняя — стандартной математической нотации.

Установка параметров вывода

Команда Output Display раскрывает подменю, имеющее четыре команды, влияющие на вид результатов вычислений — вывода:

- Maple Notation — вывод в одну строку (как в Maple-языке);
- Character Notation — вывод в виде формулы, набранной из знаков на разных строках;
- Typeset Notation — вывод в печатной форме без возможности редактирования;
- Standard Math Notation — вывод в виде обычной математической формулы.

Рисунок 4.15 наглядно иллюстрирует эти формы вывода. Последняя форма наиболее наглядна и задается по умолчанию.

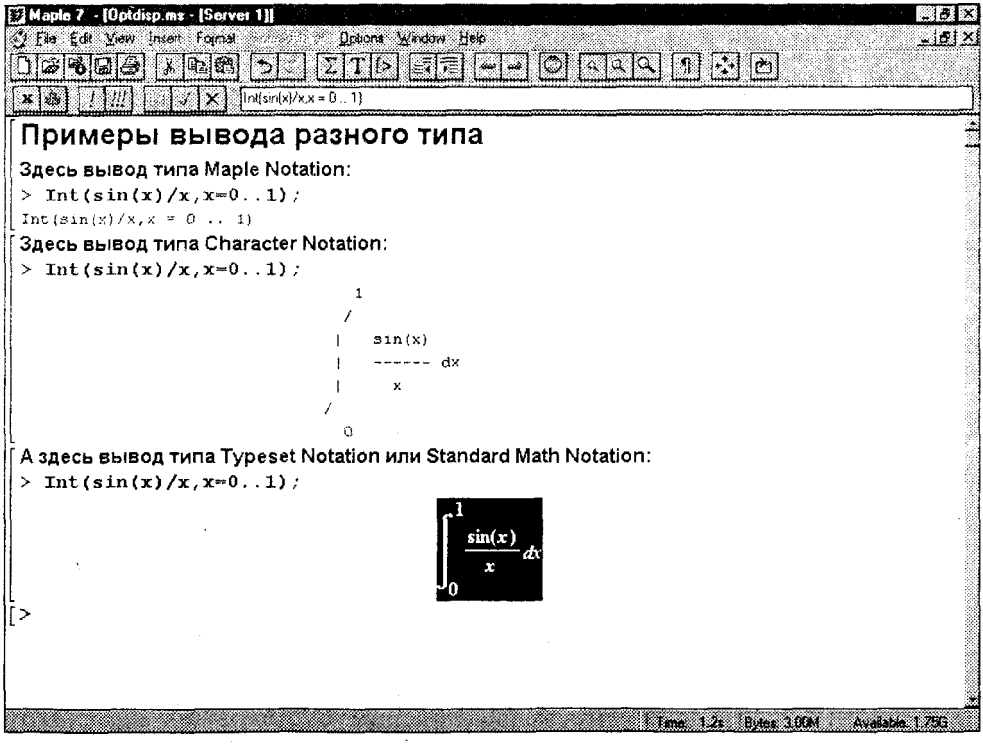


Рис. 4.15. Демонстрация вывода в различных формах

Первая форма наиболее компактна, но менее всего наглядна. Вторая форма имитирует построение формул с помощью отдельных знаков, расположенных на разных уровнях (строках). Эти две формы позволяют использовать Maple

даже в том случае, когда имеются устройства вывода (дисплеи и принтеры), работающие только в текстовых режимах. Третья форма дает вывод в виде обычных математических формул, но без возможности их редактирования.

Четвертая форма позволяет представить результат вывода, там, где это возможно, *в виде обычных математических формул с применением типовых математических знаков* — интегралов, производных, сумм, произведений, квадратных корней и т. д. В большинстве случаев именно эта форма вывода является наиболее наглядной. Она подобна третьей форме, но допускает редактирование выражений. Как отмечалось, нажатием кнопки с символом «x» в панели инструментов документа можно попытаться и вводимое выражение представить в виде обычной математической формулы. Однако это не всегда удается, поскольку далеко не все однострочные функции системы Maple 7 можно представить таким образом.

Контроль за предполагаемыми переменными (Assumed Variables)

На переменные в Maple 7 могут быть наложены различные условия. Для этого используется специальная функция `assume`. Например, если переменная `x` может принимать только положительные значения, то для этого достаточно исполнить команду `assume(x>0)`. Будем называть такие переменные *предполагаемыми*, поскольку предполагается, что они имеют какие-то дополнительные ограничения, помимо накладываемых на них типом.

В подменю Assume Variables меню Options имеются три команды, управляющие *контролем признаков предполагаемых переменных*:

- Trailing Tildes — включает маркировку предполагаемых переменных знаком тильды (~);
- No Annotation — включает параметр «без аннотаций», то есть запрещает вывод аннотации;
- Phrase — включает параметр вывода комментариев для предполагаемых переменных.

Предполагаемые переменные при выводе обычно обозначаются значком тильды (~) после их имени. Этот знак отображается при установки флажка Trailing Tildes (по умолчанию она включена). Однако с помощью флажка No Annotation можно отключить как это обозначение, так и короткий текстовый комментарий, который сопровождает предполагаемые переменные. Наконец, флажок Phrase включает вывод текстовых комментариев. Рисунок 4.16 наглядно иллюстрирует применение этих параметров.

Благодаря применению предполагаемых переменных облегчается реализация ряда алгоритмов, критичных к выбору переменных. Например, если использовать вычисление квадратного корня без привлечения понятия о комплексных числах, то

на численные значения переменных надо наложить условие их положительности. Контроль за статусом таких переменных и дают описанные параметры.

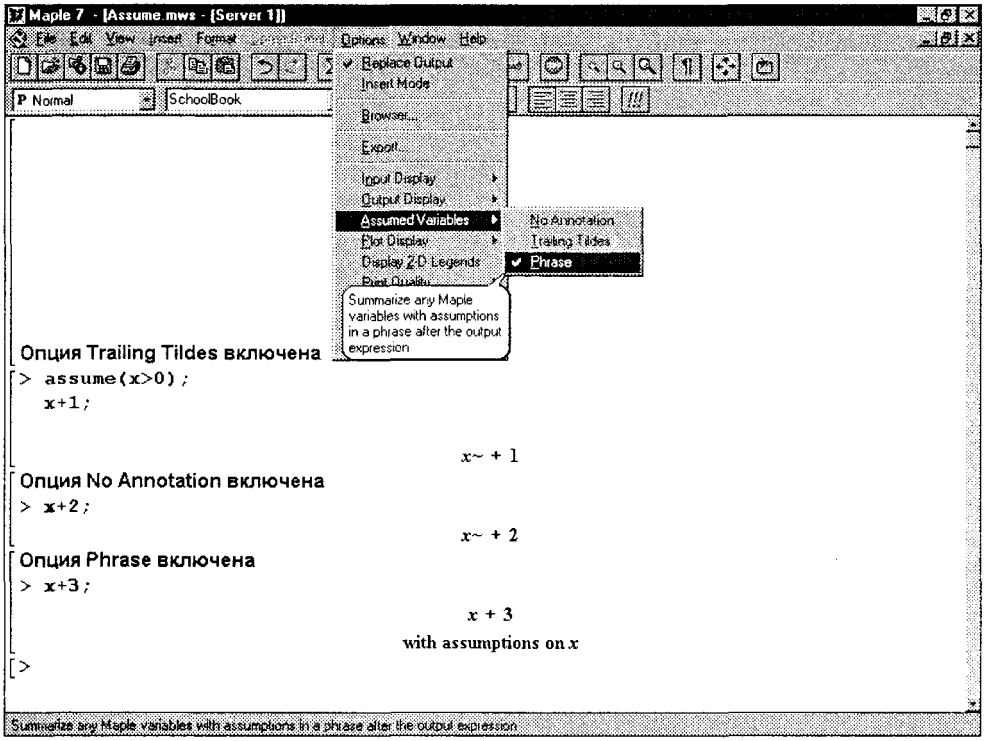


Рис. 4.16. Применение параметров контроля предполагаемых переменных

Управление показом графиков

Графические результаты могут быть представлены прямо в документе в ячейках вывода или в отдельных окнах. Это обеспечивается двумя командами подменю Plot Display:

- Inline — вывод графиков в ячейках вывода;
- Window — вывод графиков в отдельных окнах.

На рис. 4.17 показан пример вывода двух графиков — один выводится с применением Inline в ячейку документа (сразу после ввода), а другой с применением Window в отдельное окно. Какой из этих двух вариантов предпочтительнее, зависит от привычек пользователя.

Следует отметить, что из всех окон (документов или графиков) в данный момент активным может быть только одно окно. Если это окно графическое, то для него выводится своя контекстная панель инструментов, позволяющая ме-

нять вид графика и некоторые параметры, используемые при его построении. Кстати говоря, запись заданного документа на диск возможна только при активном окне этого документа.

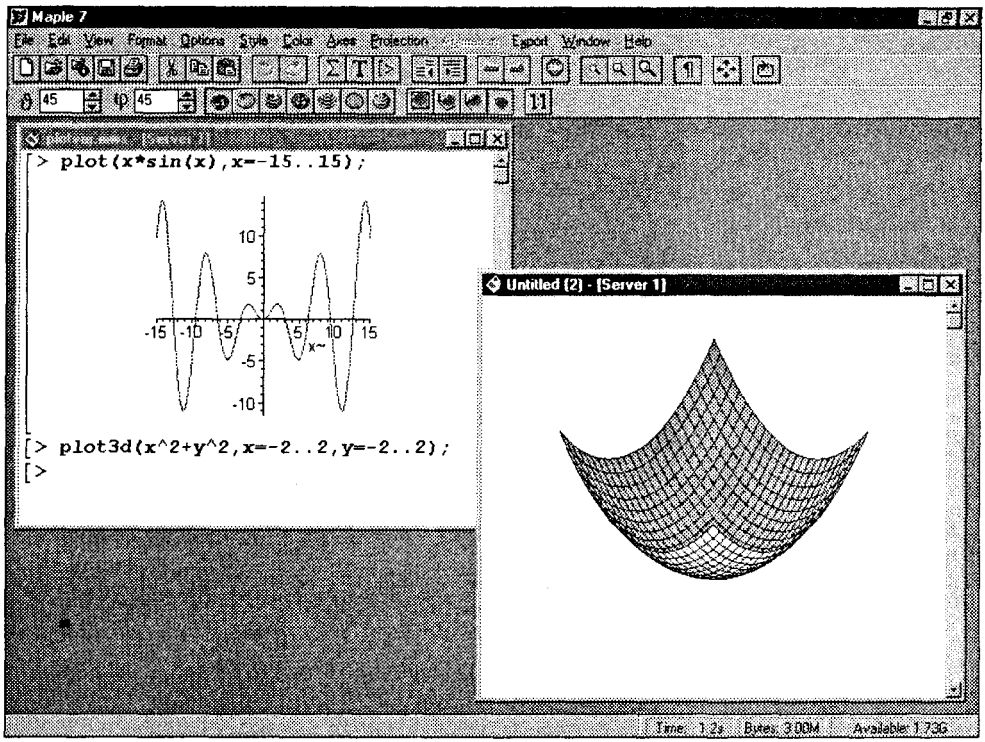


Рис. 4.17. Пример построения двух графиков с выводом одного из них в отдельное окно

Следует отметить, что из всех окон (документов или графиков) в данный момент активным может быть только одно окно. Если это окно графическое, то для него выводится своя контекстная панель инструментов, позволяющая менять вид графика и некоторые параметры, используемые при его построении. Кстати говоря, запись заданного документа на диск возможна только при активном окне этого документа.

Управление построением двумерных графиков

Двумерные графики обычно строятся с применением функции `plot`. На рис. 4.18 (и на рис. 1.24 с графиком в документе) представлено назначение кнопок контекстной панели инструментов двумерной графики. Кнопки обозначены номерами под ними, а в окне документа дано их краткое назначение.

Панель инструментов графиков появляется только при активном окне графики или при выделении графика в ячейке документа.

Рис.4.18. Окно двумерного графика и его панель инструментов

Управление построением трехмерных графиков

Трехмерные графики имеют свою контекстную панель инструментов (рис. 4.19; см. также рис. 1.25 с графиком в документе), с кнопками задания наиболее распространенных параметров для построения трехмерных графиков.

Следует отметить, что панели инструментов графических окон дают доступ лишь к части параметров графиков. Более подробно состав и назначение параметров будут рассмотрены в дальнейшем при описании средств создания графиков. Читателю настоятельно рекомендуется опробовать действие кнопок управления различными форматами графиков, что позволит быстрее освоить огромные возможности Maple 7 в создании цветных и монохромных графиков.

Работа с окнами

Меню Window

При серьезной работе в среде Maple 7 пользователь нередко вынужден работать одновременно с несколькими документами. Удобства такой работы зависят от

того, как окна расположены в пределах экрана. Maple 7 дает возможность расположить их любым стандартным способом.

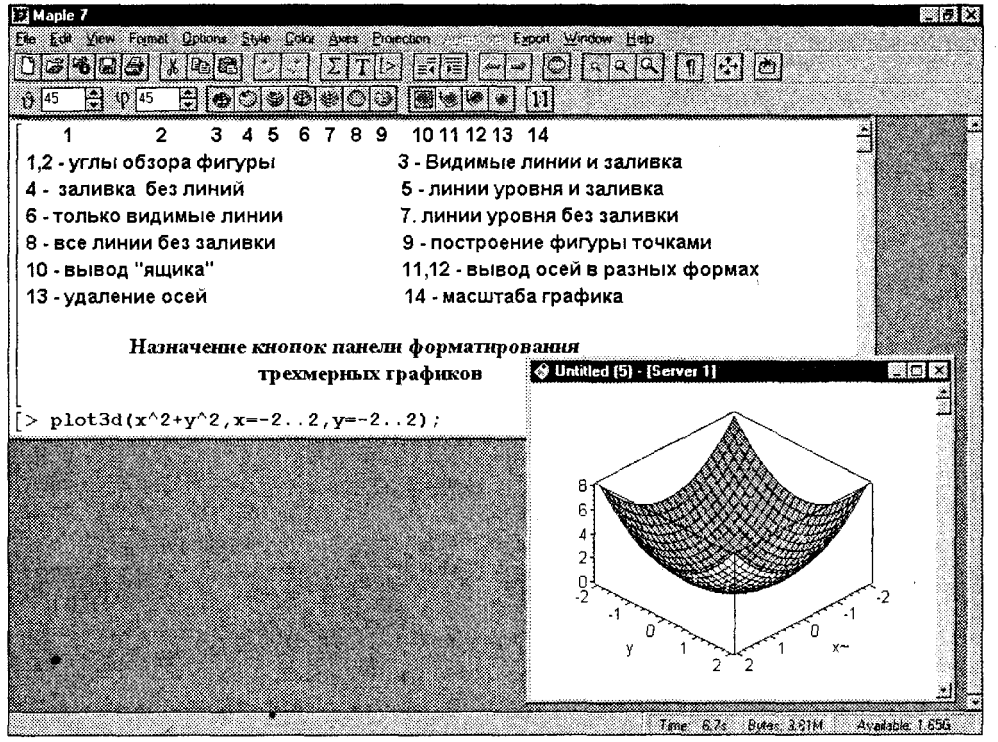


Рис. 4.19. Окно трехмерного графика и назначение кнопок панели инструментов

Основные команды по установке расположения окон сосредоточены в меню Window (рис. 4.20).

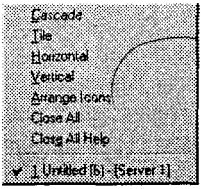


Рис. 4.20. Меню Window

Это меню содержит следующие команды:

- **Cascade** — каскадное расположение окон;
- **Tile** — расположение окон мозаикой;
- **Horizontal** — расположение окон по горизонтали;
- **Vertical** — расположение окон по вертикали;

- Arrange Icons — упорядочение расположения икон;
- Close All — закрытие всех окон документов;
- Close All Help — закрытие всех окон справочной системы.

Каскадное расположение окон

Каскадное расположении окон напоминает колоду карт, сдвинутых так, чтобы были видны их титульные строки. Такое расположение окон показано на рис. 4.21.

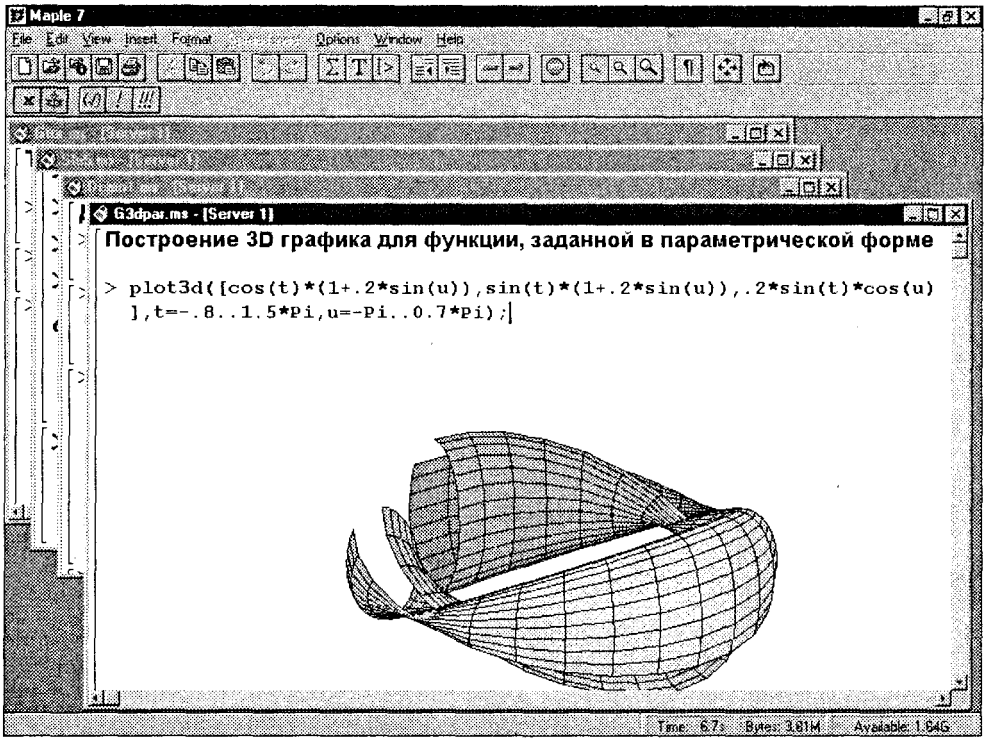


Рис. 4.21. *Каскадное расположение окон документов*

Расположение окон мозаикой

При выполнении команды Tile устанавливается расположение окон мозаикой, показанное на рис. 4.22. При этом окна не перекрываются, имеют примерно одинаковый размер. К сожалению, при большом числе окон область просмотра оказывается настолько мала, что работать с документами при таком расположении окон становится неудобно.

Расположение окон мозаикой достаточно удобно при работе с двумя (или на большом экране тремя) окнами. Оно может быть полезно, например, при переносе содержимого отдельных ячеек с одного документа в другой путем перетас-

кивания. Можно также копировать объекты в одном окне, а затем, переключившись в другое окно, вставлять содержимое буфера обмена в нужную ячейку, используя команду Paste.

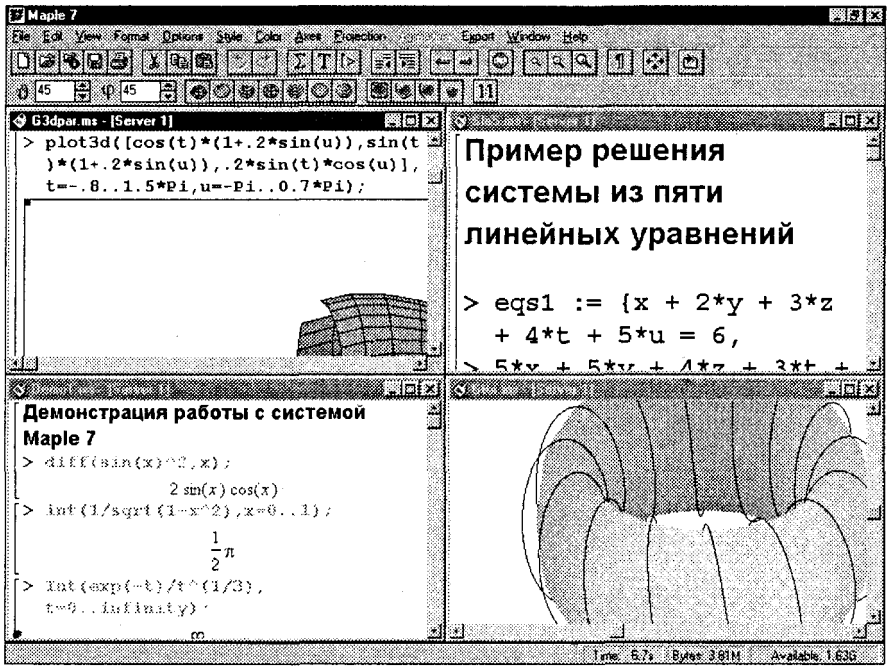


Рис. 4.22. Расположение окон мозаикой

Горизонтальное расположение окон

При использовании команды Horizontal окна оказываются расположеными на экране в виде горизонтальных полос (рис. 4.23). Этот вариант расположения окон, как и каскадное расположение, дает обзор заголовков документов, если область просмотра каждого документа расположена сверху окна. Последнее условие может нарушаться при перемещении окон документов.

Вертикальное расположение окон (Vertical)

Команда Vertical задает расположение окон в виде вертикальных полос (рис. 4.24). Такое расположение окон удобно, если содержимое ячеек документов представлено короткими выражениями.

Следует отметить, что описанным командам подчиняются только развернутые в данный момент окна — расположение вновь открытого или развернутого окна не зависит от данных до этого команд.

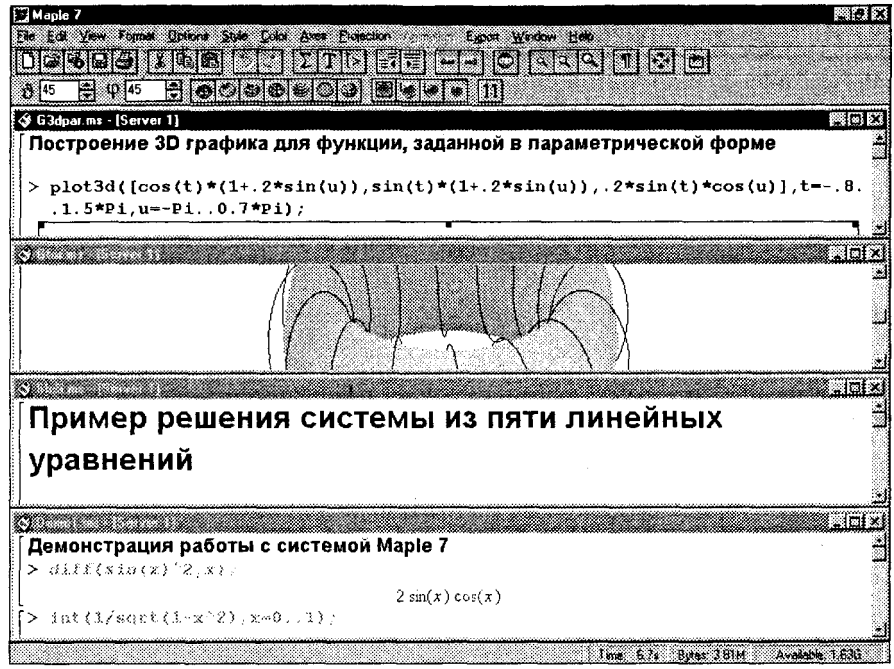


Рис. 4.23. Горизонтальное расположение окон

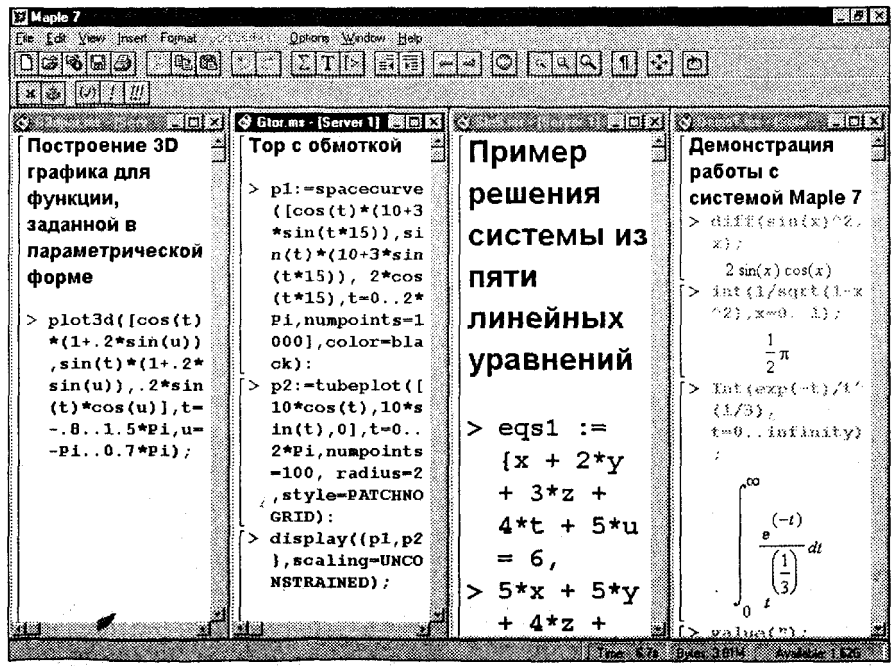


Рис. 4.24. Вертикальное расположение окон

Приведение в порядок значков свернутых окон

Свернутые окна представлены значками. Они могут перемещаться мышью по всему пространству экрана, поэтому иногда значки оказываются хаотично разбросанными (рис. 4.25).

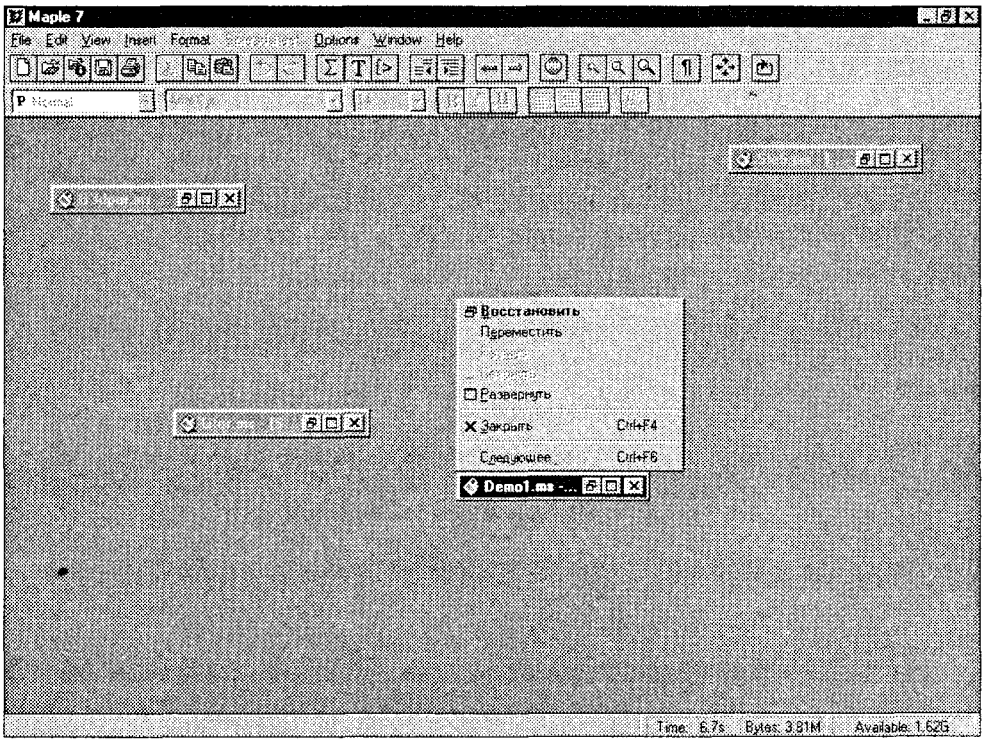


Рис. 4.25. Хаотическое расположение значков свернутых окон

Для наведения порядка с расположением значков можно, разумеется, переместить каждый из них в удобное место. Но это довольно утомительное занятие. Лучше воспользоваться специальной командой Arrange Icons, которая автоматически наводит порядок в расположении значков свернутых окон, аккуратно располагая их снизу экрана. Действие этой команды показано на рис. 4.26.

Как и окно, каждый значок имеет строку заголовка с четырьмя маленькими кнопками. Первая из них открывает меню управления, а три другие используются соответственно для восстановления размера, развертывания на весь экран и закрытия окна.

Закрытие всех окон одновременно

Команда Close All служит для закрытия всех окон одновременно. Эта команда будет выполнена сразу только в том случае, когда все документы не модифицировались или были записаны на диск после внесения исправлений. Если какой-либо

документ был изменен, но еще не сохранен, то в центре экрана появится предупреждающее сообщение об этом (рис. 4.27).

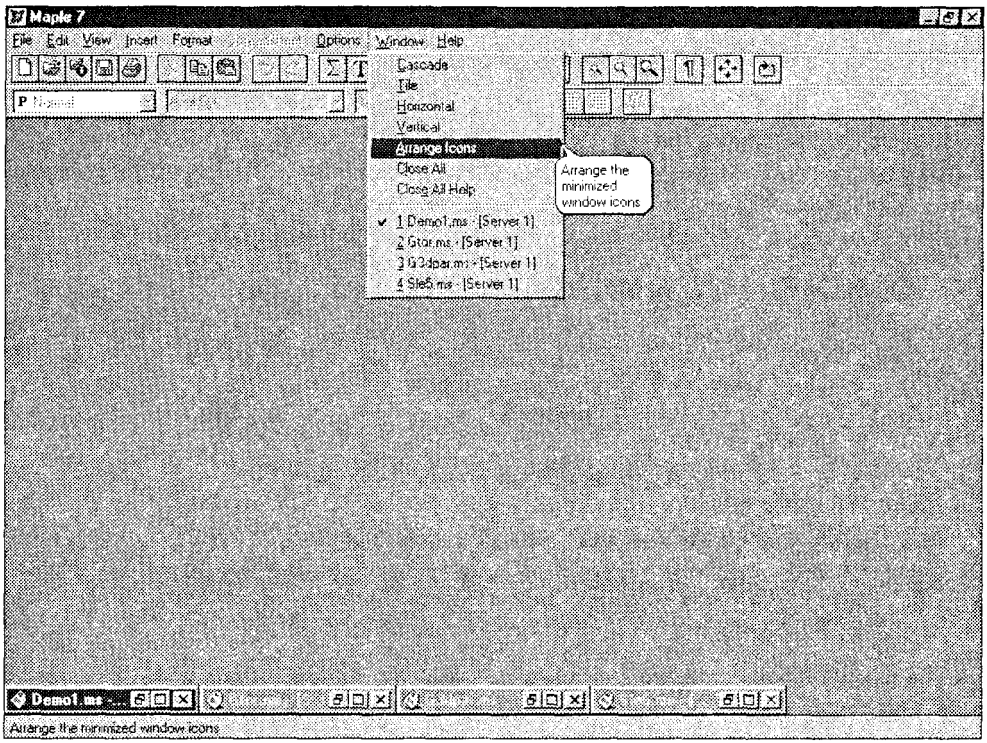


Рис. 4.26. Вид экрана после выполнения команды Arrange Icons

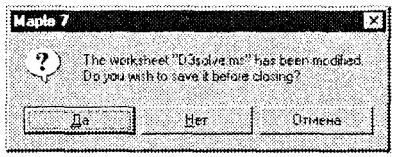


Рис. 4.27. Окно, предупреждающие о том, что закрываемый документ был изменен

Разумеется, такое окно будет появляться столько раз, сколько имеется модифицированных документов.

Закрытие всех окон справочной системы

В Maple 7 использована многооконная справочная система с гипертекстовыми ссылками. Поэтому при работе с ней экран системы довольно быстро оказывается забитым окнами справочной системы. Разумеется, их можно последовательно закрывать одно за другим, но это очень неудобно и долго. Команда Close All Help

обеспечивает закрытие разом всех окон справочной системы, что позволяет продолжить работу с текущим документом. На окна загруженных документов операция Close All Help влияния не оказывает.

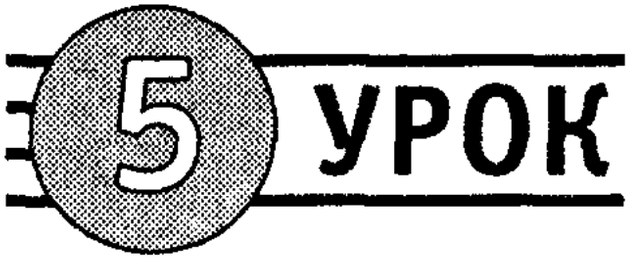
Список открытых документов

Заканчивая рассмотрение меню Window, надо отметить, что оно завершается списком всех открытых в Maple 7 документов и соответственно окон (рис. 4.26). Щелкнув на той или иной строке этого списка, можно открыть соответствующее окно и сделать его активным.

Что нового мы узнали?

В этом уроке мы научились:

- Изменять вид интерфейса и документа.
- Выводить и скрывать палитры математических знаков.
- Управлять показом непечатаемых символов.
- Изменять статус ячеек документа.
- Устанавливать параметры Maple 7.
- Работать с окнами документов.
- Просматривать список открытых документов.



Типы данных системы Maple 7

- Maple-язык и его синтаксис
- Выражения и основы работы с ними
- Простые типы данных
- Данные множественного типа
- Строки и комментарии
- Переменные
- Константы

Maple-язык и его синтаксис

Знаки алфавита

Язык Maple (или Maple-язык) является одновременно входным языком общения с Maple 7 и языком ее программирования. Входящие в него средства (прежде всего операторы и функции) подобраны настолько полно и удачно, что при решении подавляющего большинства типовых математических задач от пользователя не требуется знаний даже основ программирования. Для решения нужной задачи обычно достаточно составить алгоритм и подобрать набор нужных для его реализации функций и иных средств Maple-языка.

В то же время Maple-язык — один из самых мощных языков программирования математических задач, содержащий почти 3000 операторов, команд и функций, входящих в ядро, основную библиотеку и пакеты функций Maple 7. При этом относящаяся к традиционному программированию часть Maple-языка реализована с помощью довольно скромного набора специальных знаков и зарезервированных слов.

Большинство функций Maple 7 (в частности, все, входящие в пакеты) написаны на этом языке. Поэтому знание этого языка является определяющим в серьезном изучении Maple. Ниже Maple-язык описывается как типичный язык программирования.

Алфавит Maple-языка содержит 26 малых латинских букв (от `a` до `z`), 26 больших латинских букв (от `A` до `Z`), 10 арабских цифр (от `0` до `9`) и 32 специальных символа (арифметические операторы `+`, `-`, `*`, `/`, знак возведения в степень `^` и др.). Все они будут рассмотрены в данной главе. Имеется пять пар альтернативных символов (означающих одно и тоже):

`^` и `**` `[` и `(|` `]` и `|)` `{` и `(*` `}` и `*)`

К специальным одиночным и составным знакам относятся элементы синтаксиса языка:

- `%` — системная переменная, хранящая результат предшествующей операции;
- `:` — фиксатор выражения, предотвращающий вывод результата вычисления в ячейку вывода;
- `;` — фиксатор выражения, дающий вывод результата вычисления в ячейку вывода;
- `#` — указатель программного комментария;
- `` ` `` — ограничитель строки (например, `` `string` ``);

- := — оператор присваивания (например, x:=5);
- ; ; — пустой оператор;
- :: — указатель типа переменной (например, n::integer или z::complex);
- \ — знак обратного деления, который имеет множественные значения в зависимости от контекста (см. справку по этому знаку – backslash).

Комментарии в программе, не выводимые в ячейки вывода, задаются после символа #. В них допустимо использовать все символы кодовых таблиц, что важно при вводе русскоязычных комментариев, использующих символы кириллицы. Применение последних для идентификаторов (имен) объектов недопустимо.

Зарезервированные слова

Зарезервированные слова используются для создания условных выражений, циклов, процедур и управляющих команд. Список 42 зарезервированных слов Maple 7 дан ниже.

and	break	by	catch	description
do	done	elif	else	end
error	export	fi	finally	for
from	global	if	in	intersect
local	minus	mod	module	next
not	od	option	options	or
proc	quit	read	return	save
stop	then	to	try	union
use	while			

Совокупность правил, по которым записываются определения всех объектов Maple-языка, называется его синтаксисом. Некоторые особенности синтаксиса полезно знать уже в начале освоения Maple. Например, то что знак - (минус) имеет двойное значение. Применительно к одному числу, переменной или выражению он меняет их знак. Однако два знака минус подряд (например, в записи --3) задавать нельзя. Другое назначение знака минус — создание операции вычитания, например 5-2 или a-b. Соответственно двойное назначение имеет и знак +, причем число без знака считается положительным, так что +5=5.

При вводе действительных чисел с порядком для указания порядка используется символ ^ (например, 2*10^100 или 2*10^-100). Для возведения числа в степень наряду с оператором ^ можно использовать и составной оператор ** (две звездочки подряд). Для изменения общепринятого приоритета вычислений используются круглые скобки, в них же задаются параметры функций и процедур. Более подробно синтаксис Maple-языка рассматривается ниже.

Некоторые операторы представлены двумя символами — например, оператор присваивания переменным их значения := содержит двоеточие и знак равенства. В таких операторах между символами недопустим знак пробела. Однако его можно использовать между отдельными частями выражений — так, (a+b)/c эквивалентно (a + b) / c.

По набору операторов и функций Maple-язык намного превосходит любой универсальный язык программирования. Это позволяет наряду с обычными программными конструкциями задавать множество специальных конструкций, подчас резко упрощающих запись математических выражений. К примеру, возможна работа со списками имен функций. Язык Maple имеет множество операций над символьными выражениями и гибкий аппарат создания и преобразования типов данных и результатов вычислений.

Для большинства пользователей возможности языка Maple кажутся явно избыточными, и большинство наиболее распространенных операций в нем реализуется несколькими способами. Однако каждый пользователь волен выбирать из множества возможностей именно те, которые ему необходимы в конкретной предметной области. Поскольку таких областей превеликое множество, то обширные возможности Maple лишними не являются.

Выражения и основы работы с ними

Выражения и их ввод

Фактически Maple 7 — это система для манипулирования математическими выражениями.

Выражение в системе Maple — это объект, вполне соответствующий сути обычного математического выражения. Оно может содержать операторы, операнды и функции с параметрами. В этом уроке выражения записываются на Maple-языке без использования специальных средств для их представления в естественном математическом виде. Благодаря этому запись выражений и приводимых примеров одинаково пригодна для любой реализации системы Maple — даже под MS-DOS. Такая запись получается наиболее короткой, ее можно выводить и распечатывать без применения графических средств. Кроме того, она соответствует виду, принятому в справочной системе Maple.

Однако пользовательский интерфейс системы Maple 7 для Windows позволяет представлять как вводимые, так и выводимые выражения в самых различных формах, в том числе в естественном математическом виде — примеры этого многократно приводились и будут приводиться в дальнейшем. Maple 7 имеет многочисленные функции преобразования форматов, позволяющие менять форму представления данных.

Выражения в Maple могут оцениваться и изменяться в соответствии с заданными математическими законами и правилами преобразований. Например, функция упрощения выражений `simplify` способна упрощать многие математические выражения, записанные в качестве ее параметра (в круглых скобках):

```
> simplify(sin(x)^2+cos(x)^2);
 1
> simplify((x^2-2*x*a+a^2)/(x-a));
```

$$x - a$$

Символьные преобразования и вычисления математических выражений более подробно будут рассмотрены в следующем уроке.

Для выполнения любых математических операций необходимо обеспечить ввод в систему исходных данных — в общем случае математических выражений. Для ввода их и текстовых комментариев служат два соответствующих типа строк ввода. Переключение типа текущей строки ввода осуществляется клавишей F5. Строка ввода математических выражений имеет отличительный символ >, а строка ввода текстов такого признака не имеет.

В строке ввода может располагаться несколько выражений. Фиксаторами (указанием, что выражение окончено) их могут быть символы ; (точка с запятой) и : (двоеточие). Символ «;» фиксирует выражение и задает вывод результатов его вычисления. А символ «:» фиксирует выражение и блокирует вывод результатов его вычисления. Фиксаторы выполняют также функцию разделителей выражений, если в одной строке их несколько.

Ввод выражения оканчивается нажатием клавиши Enter. При этом маркер ввода (жирная мигающая вертикальная черта) может быть в любой позиции строки. Если надо перенести ввод на новую строку, следует нажимать клавиши Shift и Enter совместно. С помощью одного, двух или трех знаков % (в реализациях до Maple V R5 это был знак прямых кавычек ") можно вызывать первое, второе или третье выражение с конца сессии:

```
> a:b:c:
> %;
```

c

```
> a:b:c:
> %%;
```

b

```
> a:b:c:
> %%%;
```

a

```
> 2+3:
> %;
```

5

```
> %%+5;
```

10

Особая роль при вводе выражений принадлежит знакам прямого апострофа (одиночного ' или двойного ''). Заключенное в такие знаки выражение освобождается от одной пары (закрывающего и открывающего знаков '):

```
> ''factor(a^2+2*a*b^2+b^2)'';
```

factor $(a^2 + 2\,a\,b^2 + b^2)$'

```
> %;
```

facto r $(a^2 + 2\,a\,b^2 + b^2)$'

```
> factor(a^2+2*a*b+b^2);
```

$(a + b)^2$

Некоторые другие возможности обрамления выражений апострофами мы рассмотрим позже. Наиболее важная из них — временная отмена выполненного ранее присваивания переменным конкретных значений.

Для завершения работы с текущим документом достаточно исполнить команду `quit`, `done` или `stop`, набранную в строке ввода (со знаком `:` в конце).

Оценивание выражений

Встречая выражение, Maple 7 оценивает его, то есть устанавливает возможность его вычисления. Если выражение — скалярная переменная, то ее значение будет выведено в ячейке вывода. Для переменных более сложных типов выводится не их значение, а просто повторяется имя переменной. Просто повторяются также имена неопределенных переменных.

Для оценивания выражений различного типа существует группа функций, основные из которых перечислены ниже:

- `eval(array)` — возвращает вычисленное содержимое массива `array`;
- `evalf(expr, n)` — вычисляет `expr` и возвращает вычисленное значение в форме числа с плавающей точкой, имеющего `n` цифр после десятичной точки;
- `evalhf(expr)` — вычисляет `expr` и возвращает вычисленное значение с точностью, присущей оборудованию данного компьютера;
- `evalf(int(f, x=a..b))` — оценивает и возвращает значение определенного интеграла `int(f,x=a..b)`;
- `evalf(Int(f, x=a..b))` — оценивает и возвращает значение определенного интеграла, заданного инертной функцией `Int(f,x=a..b)`;
- `evalf(Int(f, x=a..b, digits, flag))` — аналогично предыдущему, но возвращает значение интеграла с заданным параметром `digits` числом цифр после десятичной точки и со спецификацией метода вычислений `flag`;
- `evalm(mexpr)` — вычисляет значение матричного выражения `mexpr` и возвращает его;
- `evalb(bexpr)` — вычисляет и возвращает значения логических условий;
- `evalc(cexpr)` — вычисляет значение комплексного выражения;
- `evalr(expr, ampl)` — оценивает и возвращает значения интервальных выражений (функция должна вызываться из библиотеки);
- `shake(expr, ampl)` — вычисляет интервальное выражение.

Для функции `evalf` параметр `n` является необязательным, при его отсутствии полагается `n=10`, то есть вещественные числа выводятся с мантиссой, имеющей десять цифр после десятичной запятой.

В выражении `expr` могут использоваться константы, например `Pi`, `exp(1)`, и функции, такие как `exp`, `ln`, `arctan`, `cosh`, `GAMMA` и `erf`. В матричном выражении `mexpr` для функции `evalm` могут использоваться операнды в виде матриц и матричные операторы `&*`, `+`, `-` и `^`. В комплексных выражениях `cexpr` наряду с комплексными операндами вида `(a + I*b)` могут использоваться многие обычные математические функции:

sin	cos	tan	csc	sec	cot
sinh	cosh	tanh	csch	sech	coth
arcsin	arccos	arctan	arccsc	arcsec	arccot
arcsinh	arccosh	arctanh	arccsch	arcsech	arccoth
exp	ln	sqrt	^	abs	conjugate
polar	argument	signum	csgn	Re	Im
Ei	LambertW	dilog	surd		

Примеры применения функций оценивания даны ниже:

```
> A:=[[1,2],[3,4]];
```

$$A := [[1, 2], [3, 4]]$$

```
> eval(A);
```

$$[[1, 2], [3, 4]]$$

```
> evalf(sin(1));
```

$$.8414709848$$

```
> evalf(sin(2)^2+cos(2)^2,20);
```

$$1.0000000000000000000$$

```
> evalhf(sin(1));
```

$$.841470984807896505$$

```
> evalm(20*A+1);
```

$$\begin{bmatrix} 21 & 40 \\ 60 & 81 \end{bmatrix}$$

```
> 1<3;
```

$$1 < 3$$

```
> evalb(1<3);
```

true

```
> readlib(shake):
evalr(min(2,sqrt(3)));
```

$$\sqrt{3}$$

```
> evalr(abs(x));
```

$$\text{INTERVAL}(\text{INTERVAL}(\ , 0 \ldots \infty), -\text{INTERVAL}(\ , -\infty \ldots 0))$$

```
> shake(Pi,3);
```

$$\text{INTERVAL}(3.1102 \ldots 3.1730)$$

В дальнейшем мы многократно будем применять функции оценивания для демонстрации тех или иных вычислений.

Последовательности выражений

Maple 7 может работать не только с одиночными выражениями, но и с последовательностями выражений. Последовательность выражений — это ряд выражений, разделенных запятыми и завершенных фиксатором:

```
> a,y+z,12.3,cos(1.0);
```

$$a, y + z, 12.3, .5403023059$$

Для автоматического формирования последовательности выражений применим специальный оператор $, после которого можно указать число выражений или задать диапазон формирования выражений:

```
> f$5;
```

f, f, f, f, f

```
> $1..5;
```

1, 2, 3, 4, 5

```
> (n^2)$5;
```

n^2, n^2, n^2, n^2, n^2

```
> (n^2)$n=0..5;
```

0, 1, 4, 9, 16, 25

```
> Vl[i]$i=1..5;
```

$Vl_1, Vl_2, Vl_3, Vl_4, Vl_5$

Для создания последовательностей выражений можно использовать также функцию seq:

```
> seq(sin(x),x=0..5);
```

0, sin(1), sin(2), sin(3), sin(4), sin(5)

```
> seq(sin(x*1.),x=0..5);
```

0., .8414709848, .9092974268, .1411200081, -.7568024953, -.9589242747

```
> seq(fl(1.),fl=[sin,cos,tan]);
```

.8414709848 . .5403023059, 1.557407725

```
> sin(1.0),cos(1.0),tan(1.0);
```

.8414709848, .5403023059, 1.557407725

Вывод выражений

При выполнении порой даже простых операций результаты получаются чрезвычайно громоздкими. Для повышения наглядности выражений Maple 7 выводит их с выделением общих частей выражений и с присваиванием им соответствующих меток. Метки представлены символами %N, где N — номер метки.

Помимо меток при выводе результатов вычислений могут появляться и другие специальные объекты вывода, например корни RootOf, члены вида $O(x^n)$, учитывающие погрешность при разложении функций в ряд, и обозначения различных специальных функций, таких как интегральный синус, гамма-функция и др. Примеры такого вывода приведены ниже:

```
> solve(x^7-x^2-1,x);
```

$$\frac{1}{2}+\frac{1}{2}I\sqrt{3}, \frac{1}{2}-\frac{1}{2}I\sqrt{3}, \text{RootOf}(\_Z^5+\_Z^4-\_Z^2-\_Z-1, index=1),$$

$$\text{RootOf}(\_Z^5+\_Z^4-\_Z^2-\_Z-1, index=2),$$

$$\text{RootOf}(\_Z^5+\_Z^4-\_Z^2-\_Z-1, index=3),$$

$$\text{RootOf}(\_Z^5+\_Z^4-\_Z^2-\_Z-1, index=4),$$

$$\text{RootOf}(\_Z^5+\_Z^4-\_Z^2-\_Z-1, index=5)$$

```
> taylor(sin(x),x,5);
```

$$x-\frac{1}{6}x^3+O(x^5)$$

```
> int(sin(x)/x,x); simplify(N!);
```

$$\mathrm{Si}(x)$$

$$\Gamma(N+1)$$

Часто встречаются также знаки ~ для отметки предполагаемых переменных, постоянные интегрирования и другие специальные обозначения. По мере упоминания в тексте таких объектов вывода они будут описаны.

Простые типы данных

Числа и числовые константы

Maple 7 работает с числами следующего типа: целыми (0, 1, 123, -456 и т. д.), рациональными в виде отношения целых чисел (7/9, -123/127 и т. д.), вещественными с мантиссой и порядком (1.23E5, 123.4567E-10). Признаком вещественного числа является десятичная точка (запятая). Примеры простых операций с числами приведены ниже:

```
> 12+34/47;m
```

$$\frac{598}{47}$$

```
> -12+34*47;
```

$$1586$$

```
> 1/3;
```

$$\frac{1}{3}$$

```
> 1./3;
```

$$.3333333333$$

```
> 12*10^(-15)*3;
```

$$\frac{9}{250000000000000}$$

```
> 12.*10^(-15)*3;
```

$$.3600000000\ 10^{-13}$$

Как видно из этих примеров, ввод и вывод чисел имеет следующие особенности:

- для отделения целой части мантиссы от дробной используется разделительная точка;
- нулевая мантисса не отображается (число начинается с разделительной точки);
- мантисса отделятся от порядка пробелом, который рассматривается как знак умножения;

- мнимая часть комплексных чисел задается умножением ее на символ мнимой единицы I (квадратный корень из −1).

Десятичная точка в числах имеет особый статус — указание ее в любом месте числа, даже в конце, делает число вещественным и ведет к переводу вычислений в режим работы с вещественными числами. При этом количеством выводимых после десятичной точки цифр можно управлять, задавая значение системной переменной окружения Digits:

```
> Digits:=3:1./3;
```

.333

```
> Digits:=10:exp(1.);
```

Digits := 10

2.718281828

```
> Digits:=40:evalf(Pi);
```

3.1415926535897932384626433832795028841 97

Для работы с числами Maple 7 имеет множество функций. Они будут рассмотрены в дальнейшем. На комплексной плоскости числа задаются координатами точек (x, y) (рис. 5.1).

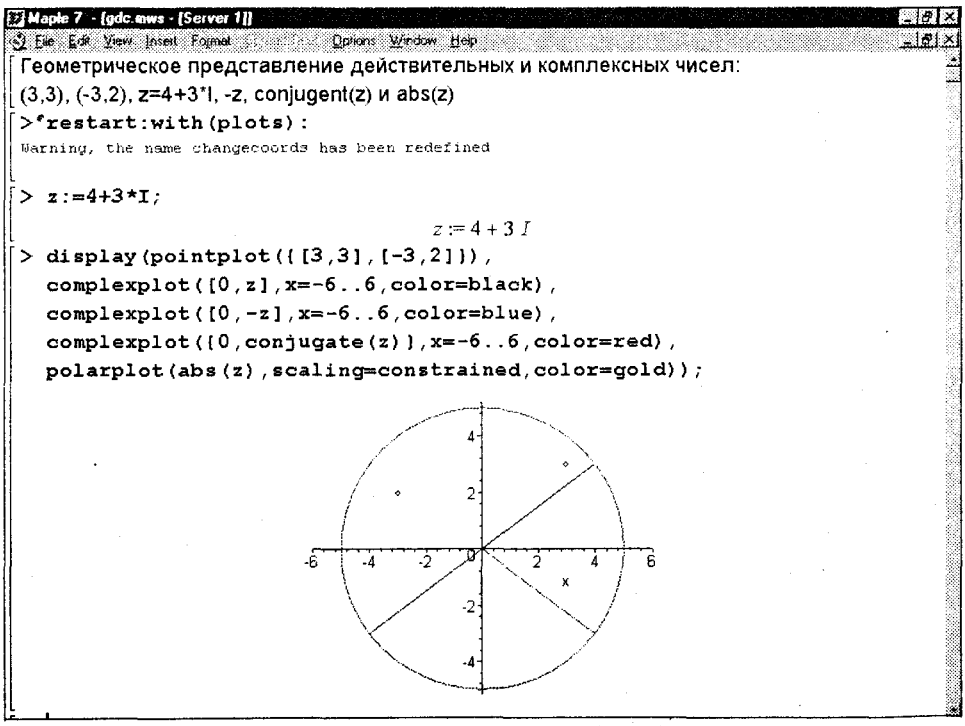


Рис. 5.1. Представление обычных и комплексных чисел на плоскости

Для представления чисел на рис. 5.1 используется функция pointplot(list), где list — список координат точек. Эта функция становится доступной при подключении пакета plots командой with(plots). Кроме того, использована функция вывода ряда графических объектов на один график — display (см. далее описание представления комплексных чисел).

С помощью функции convert Maple 7 может преобразовывать числа с различным основанием (от 2 до 36, в том числе бинарные и шестнадцатеричные) в десятичные числа:

```
> convert("11001111", decimal, binary);
 207
> convert("1AF.C", decimal, hex);
 431.7500000
> convert("Maple", decimal, 36);
 37451282
```

При символьных вычислениях Maple 7 реализует точную арифметику. Это значит, что результат может быть получен с любым числом цифр. Однако надо помнить, что идеально точные численные вычисления выполняются только в случае целочисленных операций, например таких, как приведены ниже:

```
> 201!;
```

```
158520231340322891214025006000369197521322331515121825922243249182116\
    2494426440719269811610864089094047396196399228474048817862008387705\
    0581174472516155188161538060343114234267647167947777284115699117845\
    5122910606916646903767130872655087823727239223545513045727385214935\
    3007054324127382351365326918726180319633667807143644833723187200000\
                    00000000000000000000000000000000000000000000000
```

```
> 201!-(201!+123);
 -123
```

Комплексные числа

Maple 7, естественно, может работать с комплексными числами. Мнимая единица в комплексном числе (корень квадратный из −1) обозначается как I. Функции Re(x) и Im(x) возвращают действительную и мнимую части комплексных чисел. Примеры задания комплексного числа и вывода его действительной и мнимой частей представлены ниже:

```
> 1.25+Pi*I;
 1.25 + I π
> Re(1.25+Pi*I);
 1.25
> Im(1.25+Pi*I);
 π
```

Комплексные числа обычно представляют на так называемой комплексной плоскости, у точек которой координата x задает действительную часть комплексного числа, а y (мнимая ось) показывает мнимую часть такого числа. На рис. 5.1 показано задание в виде радиус-векторов комплексного числа z = 4+3I, -z и комплексно-сопряженного числа 4-3I.

Окружность радиуса $\mathtt{abs(z)}=\sqrt{a^2+b^2}$ представляет абсолютное значение комплексного числа z=a+b*I. Она является геометрическим множеством комплексных чисел, образованных концом вращающегося радиус-вектора числа z вокруг его начала в точке (0, 0) комплексной плоскости. Позже мы рассмотрим ряд функций для работы с комплексными числами.

Контроль за числами

Числа могут служить объектами ввода, вывода и константами, входящими в математические выражения. Функция type(x, numeric) позволяет выяснить, является ли x числом. Если является, то она возвращает логическое значение true (истина), а если нет, то false (ложь). Например:

```
> type(2,numeric);
 true
> type(2.6,numeric);
 true
> type(Pi,numeric);
 false
> type(I,numeric);
 false
> type(3/7,numeric);
 true
> type(3^7,numeric);
 true
> type(x^2,numeric);
 false
```

Функции type(x, integer), type(x, rational) и type(x, fraction) можно использовать для проверки того, имеет ли x значение соответственно целого числа, рационального числа или простой дроби:

```
> type(123,integer);
 true
> type(123.,integer);
 false
> type(123/456,rational);
 true
> type(1./3,rational);
 false
> type(1/2,fraction);
 true
```

```
> type(0.5,fraction);
```
false

Преобразования чисел с разным основанием

В Maple возможна работа с числами, имеющими различное основание (base), в частности с двоичными числами (основание 2 — binary), восьмеричными (основание 8 — octal) и шестнадцатеричными (основание 16 — hex). Функция convert позволяет легко преобразовывать форматы чисел:

```
> convert(12345,binary);
```
11000000111001
```
> convert(%,decimal,binary);
```
12345
```
> convert(12345,octal);
```
30071
```
> convert(123456,hex);
```
1E240
```
> convert(%,decimal,hex);
```
123456

Помимо приведенных вариантов функция convert имеет еще ряд других форм. С ними можно познакомиться с помощью справки по этой мощной функции. В дальнейшем будет приведен ряд других применений этой функции.

Данные множественного типа

Наборы (множества)

Любые выражения могут включаться также в наборы. Такие наборы в виде множеств создаются с помощью фигурных скобок { }:

```
> {a,b,a,a,b,d,e,c,d};
```
$\{a, b, c, e, d\}$
```
> {10,2+3,4+4,8,5,1};
```
$\{1, 5, 8, 10\}$
```
> {`Hello`,`my`,`friend`};
```
$\{friend, Hello, my\}$

Отличительная черта множеств — автоматическое устранение из них повторяющихся по значению элементов. Кроме того, Maple 7 расставляет элементы множеств в определенном порядке — числа в порядке увеличения значения, а символы и строки в алфавитном порядке. Для множеств нет строгого математического определения, и мы будем считать их наборами, удовлетворяющими перечисленным выше признакам.

Списки выражений

Для создания упорядоченных наборов — списков — служат квадратные скобки [ ]:

```
> [10,2+3,4+4,8,5,1];
```

$[10, 5, 8, 8, 5, 1]$

Как нетрудно заметить, элементы списков преобразуются и выводятся строго в том порядке, в каком они были заданы. Списки широко применяются для задания векторов и матриц.

В ряде случаев, например при подготовке данных для двумерных графиков, возникает необходимость в подготовке парных списков — скажем, координат точек (x, y) графика. Для этого можно использовать функцию `zip(f, u, v)` или `zip(f, u, v, d)`. Здесь `f` — бинарная функция, `u`, `v` — списки или векторы, `d` — необязательное значение. Примеры применения функции `zip` даны ниже:

```
> X:=[1,2,3,4,5]:Y:=[3,2,1,1.5,2.5]:
> pare:=(X,Y)->[X,Y];
```

$pare := (X, Y) \rightarrow [X, Y]$

```
> CoordXY:=zip(pare,X,Y,2);
```

$CoordXY := [[1, 3], [2, 2], [3, 1], [4, 1.5], [5, 2.5]]$

```
> zip((X,Y)->X+Y,X,Y);
```

$[4, 4, 4, 5.5, 7.5]$

Рисунок 5.2 показывает применение этих средств для построения точек, представляющих множество действительных чисел на плоскости. Для этого использована функция `pointplot` из пакета `plots`.

Массивы, векторы и матрицы

Как отмечалось, важным типом данных являются списки (lists). Они создаются с помощью квадратных скобок, например:

- `[1,2,3,4]` — список из четырех целых чисел;
- `[1.,2.34,5]` — список из двух вещественных и одного целого числа;
- `[a,b,`Привет`]` — список из двух символов (переменных) и строковой константы;
- `[sin(x), 2*cos(x),a^2-b]` — список из трех математических выражений.

Для создания векторов (одномерных массивов) и матриц (двумерных массивов) служит функция `array`. Обычно она используется в следующих формах:

- `array[a..b,s1]` — возвращает вектор с индексами от `a` до `b` и значениями в одномерном списке `s1`;
- `array[a..b,c..d,s2]` — возвращает матрицу с номерами строк от `a` до `b`, номерами столбцов от `c` до `d` и значениями в двумерном списке `s2`.

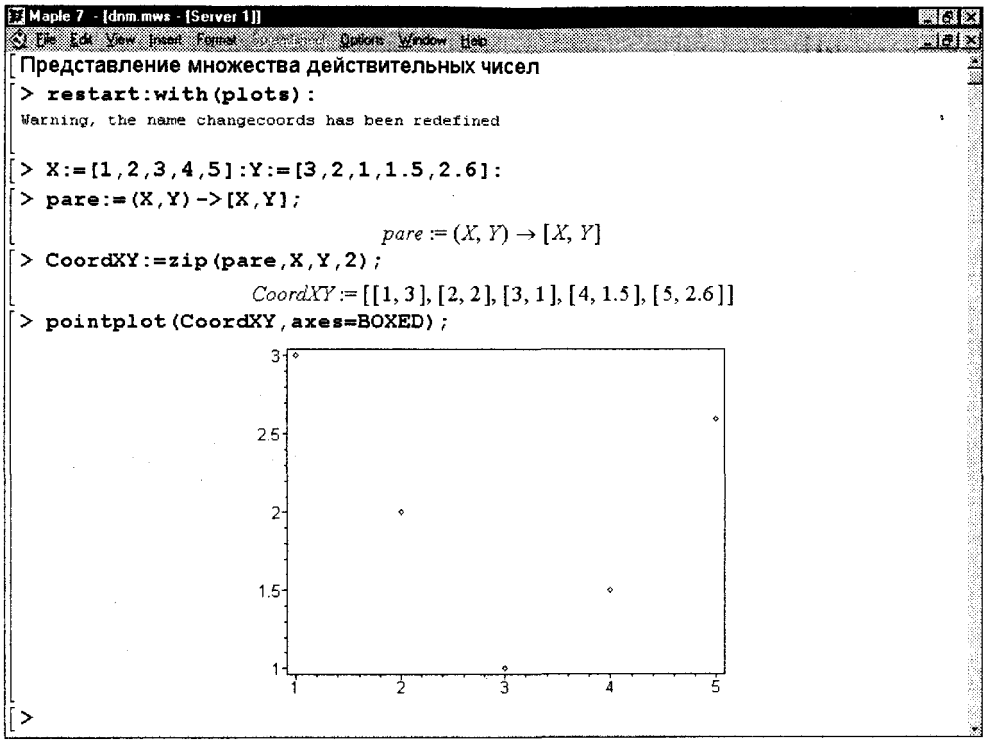


Рис. 5.2. Представление множества чисел на плоскости

Примеры задания вектора и матрицы представлены ниже:

- `array(1..3,[x,y,x+y])` — создает вектор с элементами x, y и $x + y$;
- `array(1..2,1..2,[[a,b],[c,d]])` — квадратная матрица $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$.

Двумерные списки часто путают с матрицами. Следует помнить, что векторы и матрицы создаются с помощью функции `array` и являются отдельным типом данных. Элементами векторов и массивов могут быть константы, переменные, выражения, списки и иные объекты. Эти элементы являются индексированными переменными и их положение указывается индексами.

Имеется множество функций для работы со списками, массивами и матрицами. Они будут рассмотрены в дальнейшем. В принципе, размерность массивов, создаваемых списками, не ограничена и массивы могут быть многомерными.

Таблицы

Еще одним важным типом множественных данных являются таблицы. Они задают данные с произвольной индексацией. Для создания таблиц служит функция `table`, которая при вызове в простейшем виде `table[]` создает шаблон пустой таблицы:

```
> table[];
```
$table_{[\,]}$

Пустая таблица резервирует память под данные. Когда параметром функции table является список выражений, он выводится в естественном порядке расположения элементов таблицы, но с произвольным порядком индексации:

```
> table[];
```
$table_{[\,]}$
```
> T:=table({1,2,Pi,`string`});
```
$T := \mathrm{table}([1 = 1, 2 = 2, 3 = \pi, 4 = string])$
```
> T[3];
```
π
```
> S:=table([(one)=1,(two)=2,(three)=3]);
```
$S := \mathrm{table}([one = 1, three = 3, two = 2])$
```
> S[1];
```
S_1
```
> S[two];
```
2
```
> S[three];
```
3
```
> entries(S);
```
[1], [3], [2]
```
> indices(S);
```
[*one*], [*three*], [*two*]

В конце приведенных примеров показано, как можно выделить отдельные компоненты таблицы и вывести значения и индексы таблицы с помощью функций entries и indices. Следующие примеры показывают, что таблицу можно использовать для выполнения математических преобразований:

```
> F := table([sin=cos,cos=-sin]):
op(op(F));
```
[cos = −sin, sin = cos]
```
> F[cos](Pi/2);
```
-1
```
> F[sin](0);
```
1
```
> evalf(cos(Pi/2));
```
0.
```
> evalf(sin(0));
```
0.

Следует внимательно присмотреться к этим примерам — они демонстрируют замену функции косинуса на отрицательный синус и синуса на косинус.

Строки и комментарии

Строковые данные

Строки как тип данных — это просто цепочки символов. Они обычно используются для создания текстовых комментариев. Строки должны каким-либо образом выделяться, чтобы Maple не отождествляла их с именами констант и переменных. Для этого строки-комментарии имеют внутренний разделительный признак, который устанавливается при их вводе (нажатием клавиши F5, которое приводит к исчезновению знака >).

В других случаях последовательность символов рассматривается как строка, если она заключена в обратные апострофы, то есть в знаки `. Два апострофа подряд формируют апостроф как знак символьной строки, например `abc``def` дает строку abc`def. Любое математическое выражение может входить в строку, разумеется, оно при этом не выполняется:

```
> `2+2 не всегда ``четыре```;
2+2 не всегда `четыре`
```

Неисполняемые программные комментарии

Часто возникает необходимость в задании программных комментариев. Любой текст после знака # рассматривается как невыводимый (неисполняемый) программный комментарий — даже если это математическое выражение. При этом он не вычисляется. Например:

```
> 2+3;#Это пример. А это выражение не вычисляется: 4+5
 5
```

Комментарии полезны в программах на Maple-языке и обычно используются для объяснения особенностей реализованных алгоритмов.

Константы

Числовые константы

Константы — это простейшие именованные объекты, несущие заранее предопределенные значения. Их имена (идентификаторы) также заранее определены и не могут меняться. Подробную информацию о константах можно найти, исполнив команду ?constant.

Обычные числовые константы не имеют имени и представлены просто числами, типы которых были указаны выше. Можно считать, что именем такой константы является само ее значение. Например, в выражении 2*sin(1.25) числа 2 и 1.25 являются числовыми константами. При этом указание десятичной точки делает

константу действительным числом — например, 2 — это целочисленная константа, а 2., 2.0 или 1.25 — это уже действительные константы.

Строковые константы

Строковыми константами являются произвольные цепочки символов, заключенные в обратные апострофы, например `Hello`, `Привет`, `My number` и т. д. Числа, заключенные в апострофы, например `123456`, также становятся строковыми константами, которые нельзя использовать в арифметических выражениях. Строковые константы представляют значения строковых переменных. В них можно использовать символы кириллицы при условии, что соответствующий шрифт имеется.

Встроенные в ядро константы

Есть также ряд констант, которые правильнее считать заведомо определенными глобальными переменными:

- false — логическое значение «ложно»;
- gamma — константа Эйлера, равная 0.5772156649...;
- infinity — положительная бесконечность (отрицательная задается как infinity);
- true — логическое значение «истинно»;
- Catalan — константа Каталана, равная 0.915965594...;
- FAIL — специальная константа (см. справку, выдаваемую по команде ?FAIL);
- I — мнимая единица (квадратный корень из −1);
- Pi — представляет константу $\pi = 3.141...$

Любопытно, что в этот список не входит основание натурального логарифма — число *e*. В качестве этой константы рекомендуется использовать exp(1). Она отображается как жирная прямая буква E. А exp(1.0) выводит 2.71828... (что и следовало ожидать).

Идентификация констант

Функции type(x, constant) и type(x, realcons) возвращают логическое значение true, если x представляет целочисленную или вещественную константу, и false, если x не является константой. Таким образом, эти функции можно использовать для идентификации констант, например:

```
> type(Pi,constant);
 true
> type(2,constant);
 true
> type(1/2,constant);
 true
```

```
> type(.5,constant);
 true
> type(x/y,constant);
 false
> type(ln(-Pi),constant);
 true
> type(infinity,constant);
 true
> type(1.234,realcons);
 true
> type(x*y,realcons);
 false
> type(2+3*I,realcons);
 false
```

Защита идентификаторов констант

Имена встроенных констант (как и имена функций) защищены специальным атрибутом protected. Поэтому (без его снятия) константам нельзя присваивать какие-либо значения:

```
> Pi;
 π
> Pi:=1;
```

Error, attempting to assign to `Pi` which is protected

```
> gamma;
 γ
> gamma:=10;
```

Error, attempting to assign to `gamma` which is protected

Стоит упомянуть о такой экзотической возможности, как задание в Maple 7 собственных констант путем описания алгоритма генерации входящих в константу цифр (это позволяет получать в представлении константы любое число цифр). Большинство пользователей довольствуется применением вместо таких констант обычных переменных подходящего типа. Зато истинные математики соревнуются друг с другом в создании все новых и новых констант и алгоритмов их вычислений. Примеры этого творчества можно найти на сайте фирмы Wateloo Maple.

Переменные

Типы переменных

Как следует из самого названия, переменные — это объекты, значения которых могут меняться по ходу выполнения документа. Пока мы рассматриваем лишь

глобальные переменные, доступные для модификации значений в любом месте документа. Тип переменной в системе Maple 7 определяется присвоенным ей значением — это могут быть целочисленные (`integer`), рациональные (`rational`), вещественные (`real`), комплексные (`complex`) или строчные (`string`) переменные и т. д. Переменные могут также быть символьного типа (их значением является математическое выражение) или типа списка (см. далее). Для явного указания типа переменных используется конструкция:

```
name::type
```

где `name` — имя (идентификатор) переменной, `type` — тип переменной, например целочисленный (`integer`), вещественный с плавающей точкой (`float`), с неотрицательным значением (`nonneg`), комплексный (`complex`) и т. д.

Идентификаторы (имена) переменных

Переменные задаются своим именем — идентификатором, который должен начинаться с буквы и быть уникальным. Это значит, что ключевые слова языка Maple нельзя использовать в качестве имен переменных. Хотя имена ряда команд и функций можно использовать в качестве идентификаторов переменных, делать это крайне нежелательно. Ограничений на длину идентификатора практически нет — точнее, она не должна превышать 524 275 символов! Так что сложностей с подбором идентификаторов для переменных у вас не будет.

Имена переменных могут содержать одну букву (например, `x`, `Y` или `Z`) либо ряд букв (`Xmin` или `Xmax`). *В любом случае имя переменной надо начинать с буквы.* Некоторые символы, например знак `_`, могут использоваться в именах (например, `Var_1`, `Var_2`). Нельзя, однако, вводить в имена переменных знаки, обозначающие операторы, — например, `a/b` или `a-b` будет истолковано как деление `a` на `b` или вычитание из переменной `a` переменной `b`.

Имена могут задаваться в обратных апострофах. При этом они просто тождественны именам без апострофов:

```
> var1:=123;var2:=`Hello`;
var1 := 123
```

var2 := Hello

```
> `var1`;`var2`;
123
```

Hello

Строчные и прописные буквы в идентификаторах различаются, так что `Var1` и `var1` — это разные переменные.

Для проверки предполагаемого имени на уникальность достаточно выполнить команду `?name`, где `name` — выбранное имя. Если при этом откроется окно справки с этим именем, значит, оно уже использовано в Maple. Лучше воздержаться от его применения, так как связанная с этим именем команда или функция перестает работать, как только это имя закрепляется за какой-либо переменной.

Присваивание переменным значений

Поскольку Maple 7 прежде всего система символьной математики, то по умолчанию любые переменные рассматриваются как объекты символьного типа. Благодаря этому такие переменные могут фигурировать в математических выражениях (таких, как $\sin(x)/x$) без их предварительного объявления. В отличие от обычных языков программирования такое использование переменных не влечет за собой появления сообщений об ошибках и является более естественным.

Для присваивания переменным конкретных значений используется комбинированный символ присваивания «:=», например:

- `n:=1` — переменной `n` присваивается целочисленное значение 1;
- `x:=123.456` — переменной `x` присваивается вещественное значение 123.456;
- `y:=17/19` — переменной `y` присваивается рациональное значение 17/18;
- `name:=`Piter`` — переменной `name` присваивается строковое значение `Piter`;
- `expr:=2*Pi/3` — переменной `expr` присваивается значение выражения $2\pi/3$;
- `V:=[1,2,3]` — переменной `V` присваивается значение списка чисел [1,2,3];
- `M:=[[1,2,3],[4,5,6]]` — переменной M присваивается значение двумерного массива;
- `f:=x->x^2` — переменной `f` присваивается значение функции пользователя `f(x)=x^2`.

Правая часть выражения присваивания определяет тип переменной. Например, она может быть целочисленной, действительной, строковой, индексированной (элемент массива) и т. д.

Отмена операции присваивания и команда restart

Переменная, имеющая какое-либо значение, занимает в памяти намного больше места, чем неопределенная переменная. У последней место в памяти занимают только символы идентификатора. Поэтому нередко целесообразно отменить присваивание у тех переменных, которые в дальнейшем можно не использовать. Это может понадобиться и в том случае, когда какую-либо переменную с численным или иным значением нужно использовать просто как неопределенную переменную.

Рассмотрим следующий пример:

```
> x:=10;
```

$x := 10$

```
> x;
```

10

```
> int(x^2,x);
```

Error, (in int) wrong number (or type) of arguments

Здесь не удалось вычислить интеграл с подынтегральной функцией x^2 из-за того, что переменная x уже определена ранее как целочисленная переменная со значе-

нием 10, тогда как для вычисления интеграла она должна быть необъявленной или строковой (убедитесь в этом сами).

Для отмены присваивания надо использовать следующее выражение:

```
> x:='x';
```
$$x := x$$

Итак, заключение имени переменной в прямые апострофы ликвидирует присваивание. Так что запись `x:='x'` означает, что переменной `x` возвращается статус неопределенной переменной. Теперь можно вычислить интеграл:

```
> int(x^2,x);
```
$$\frac{1}{3}x^3$$

Можно сделать переменную `x` неопределенной и с помощью выражения вида `x:=eval(x)`. Это поясняет следующий пример:

```
> x:=123;
```
$$x := 123$$
```
> x:=eval(x);
```
$$x := x$$
```
> int(x^n,x);
```
$$\frac{x^{(n+1)}}{n+1}$$

Для отмены присваивания значений разом всем переменным (и введенным функциям пользователя) можно использовать команду `restart`. Следующий пример поясняет ее применение:

```
> x:=5;
```
$$x := 5$$
```
> x^2;
```
$$25$$
```
> restart;
> x;
```
$$x$$
```
> x^2;
```
$$x^2$$

Следует отметить, что команда `restart` отменяет все предшествующие определения, что иногда чревато осложнениями. Применяйте ее только тогда, когда вы уверены, что предшествующая заданной часть документа (или даже ряда документов) действительно не важна.

Важно отметить, что Maple сохраняет в памяти все определения и присваивания, которые были сделаны во всех загруженных в систему документах. Поэтому результаты вычислений в текущем документе могут зависеть от определений в других документах. Команда `restart` позволяет исключить эту зависимость.

Придание переменным статуса предполагаемых

В большинстве расчетов пользователей Maple вполне удовлетворяет статус переменных, соответствующий присвоенным им значениям. Однако серьезные расчеты предполагают, что переменные могут иметь определенные ограничения — например, они не должны принимать отрицательных значений при обычном вычислении квадратного корня или логарифма числа.

Для придания переменным статуса предполагаемых используется функция assume:

```
assume(x,prop);
```

где x — переменная, имя или выражение, prop — свойство.

Следующие примеры показывают применение функции assume:

```
> restart;
> assume(x,positive);
> x;
```

$x\sim$

```
> s:=x->sqrt(x);
```

$s := \mathrm{sqrt}$

```
> s(2);
```

$\sqrt{2}$

```
> s(2.);
```

1.414213562

```
> s(-2);
```

$I\sqrt{2}$

```
> is(x,positive);
```

true

```
> is(x,negative);
```

false

```
> about(x);
```

Originally x, renamed x~:
  is assumed to be: RealRange(Open(0),infinity)

Обратите внимание, что в этом примере переменная x помечена как положительная и при выводе сопровождается знаком тильды ~, как бы предупреждающим нас о ее особом статусе. Это не означает, что она не может принять отрицательное значение. Однако с помощью функции is можно убедиться в ее особом статусе и при необходимости программным путем исключить вычисления для $x<0$. Кроме того, о свойствах переменной можно узнать с помощью функции about(name).

Иногда к уже имеющимся признакам надо добавить новые. Для этого используется функция additionally:

```
> assume(a,nonnegative);
> additionally(a<=0);
> about(a);
```

Originally a, renamed a~:
  is assumed to be: 0

В этом примере переменной a вначале задан признак положительности, а затем a<=0. Оба признака удовлетворяются только при $a = 0$, что и подтверждает вывод информации о статусе этой переменной функцией about(a).

Предполагаемую переменную можно также изменить путем присваивания ей нового значения, противоречащего ее статусу:

```
> a:=123;
```

$$a := 123$$

```
> about(a);
```

123:

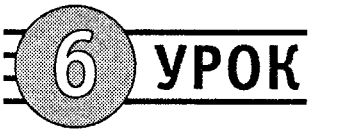
All numeric values are properties as well as objects.
Their location in the property lattice is obvious,
in this case integer.

Для отмены переменным статуса предполагаемых используются те же приемы, что и при отмене присвоенного значения. Например, запись x:='x' отменяет статус предполагаемой для переменной x.

Что нового мы узнали?

В этом уроке мы научились:

- Использовать Maple-язык и его синтаксис.
- Работать с выражениями.
- Задавать простые типы данных.
- Задавать данные множественного типа.
- Задавать данные строкового типа.
- Писать программные комментарии.
- Задавать переменные и определять их тип.
- Использовать константы различного типа.



Встроенные операторы и функции

- Операторы и операнды
- Математические функции
- Операторы и функции для работы с векторами и матрицами
- Функции для работы со строковыми данными

Операторы и операнды

Виды операторов

Операторы во входном языке и языке программирования Maple служат для конструирования выражений. Формально операторы представлены своими идентификаторами в виде специальных математических знаков, слов и иных имен. Операторы, как это вытекает из их названия, обеспечивают определенные операции над данными, представленными операндами.

Имеется пять основных типов операторов:

- binary — бинарные операторы (двумя операндами);
- unary — унарные операторы (с одним операндом);
- nullary — нульарные операторы (без операнда — это одна, две и три пары кавычек);
- precedence — операторы старшинства (включая логические операторы);
- functional — функциональные операторы.

Для просмотра операторов и их свойств можно использовать следующие команды:

```
> ?operators[binary];
> ?operators[unary];
> ?operators[nullary];
> ?operators[precedence];
> ?operators[functional];
```

А для изучения примеров применения операторов нужно задать и исполнить команду:

```
> ?operators[examples];
```

Команда:

```
> ?define;
```

позволяет ознакомиться с функций `define`. С ее помощью можно определять новые операторы.

Бинарные (инфиксные) операторы

Бинарные (инфиксные) операторы используются с двумя операндами, обычно размещаемыми по обе стороны от оператора. В ядро Maple 7 включено около трех десятков бинарных операторов. Основные из них перечислены в табл. 6.1.

Таблица 6.1. Бинарные операторы

Обозначение	Оператор
+	Сложение
-	Вычитание
*	Умножение
/	Деление
** или ^	Возведение в степень
mod	Остаток от деления
$	Оператор последовательности
.	Разделительная точка
@	Оператор композиции
@@	Повторение композиции
,	Разделитель выражений
:=	Присваивание
..	Задание интервала
,	Разделитель выражений
&*	Некоммутативное умножение
&<string>	Нейтральный оператор
||	Конкатенация (объединение)

Примеры использования бинарных операторов:

```
> 2+3-(-4);
```

$$9$$

```
> [2^3,2**3];
```

$$[8, 8]$$

```
> 7 mod 5;
```

$$2$$

```
> [3@2,3@@2];
```

$$[3, 3^{(2)}]$$

```
> [x@x,x@@x];
```

$$[x^{(2)}, x^{(x)}]$$

```
> [x$3,x$4];
```

$$[x, x, x, x, x, x, x]$$

```
> int(x^2,x=1..4);
```

$$21$$

```
> S:=`Hello`||` my `||`friend!`;
```

$$S := Hello\ my\ friend!$$

Оператор композиции @@ может использоваться для создания сложных функций, содержащих цепные дроби:

```
> f:=a->1/(1+a);(f@@3)(a);
```

$$f := a \rightarrow \frac{1}{a+1}$$

$$\frac{1}{\frac{1}{\frac{1}{a+1}+1}+1}$$

```
> f(5);
```

$$\frac{1}{6}$$

```
> g:=a->1/(1+a^2);(g@@3)(a);
```

$$g := a \to \frac{1}{a^2+1}$$

$$\frac{1}{\frac{1}{\left(\frac{1}{(a^2+1)^2}+1\right)^2}+1}$$

```
> g(2);
```

$$\frac{1}{5}$$

А вот еще один пример применения этого оператора для составления цепного радикала и вычисления ряда таких цепочек в цикле:

```
> f := x -> sqrt( 1 + x);
```

$$f := x \to \sqrt{1+x}$$

```
> f(f(0));
```

$$\sqrt{2}$$

```
> f(f(f(0)));
```

$$\sqrt{1+\sqrt{2}}$$

```
> (f@@10)(x);
```

$$\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+x}}}}}}}}}$$

```
> for k from 1 to 10 do (f@@k)(0) = evalf((f@@k)(0)); od;
```

$$1 = 1.$$

$$\sqrt{2} = 1.414213562$$

$$\sqrt{1+\sqrt{2}} = 1.553773974$$

$$\sqrt{1+\sqrt{1+\sqrt{2}}} = 1.598053182$$

$$\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{2}}}} = 1.611847754$$

$$\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{2}}}}} = 1.616121206$$

$$\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{2}}}}}} = 1.617442798$$

$$\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{2}}}}}}} = 1.617851290$$

$$\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{2}}}}}}}} = 1.617977531$$

$$\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{2}}}}}}}}} = 1.618016542$$

Операторы объединения, пересечения и исключения для множеств

Для данных типа «множество» определены следующие бинарные операторы:

- `union` — включает первый операнд (множество) во второй;
- `intersect` — создает множество, содержащее общие для операндов элементы;
- `minus` — исключает из первого операнда элементы второго операнда.

В любом случае, в результирующем множестве устраняются повторяющиеся элементы. Действие этих операторов поясняют следующие примеры:

```
> {a,a,b,c,c,d} union {e,e,f,g};
```

$\{f, g, a, b, e, c, d\}$

```
> {a,a,b,c,c,d} intersect {a,c,e,e,f,g};
```

$\{a, c\}$

```
> {a,a,b,c,c,d} minus {a,d};
```

$\{b, c\}$

Напоминаем, что эти операторы заданы ключевыми словами. Обратите внимание на то, что в Maple 7 результат применения оператора `union` представлен членами, расположенными в довольно произвольном порядке.

Унарные арифметические операторы

Унарные операторы используются с одним операндом. Они могут быть префиксными, если оператор стоит перед операндом, и постфиксными, если он стоит после операнда. К унарным относятся семь операторов, приведенных в табл. 6.2.

Таблица 6.2. Унарные операторы

Обозначение	Оператор
+	Унарный плюс (префикс)
-	Унарный минус (префикс)
!	Факториал (постфикс)
.	Десятичная точка (префикс или постфикс)
$	Последовательность (префикс)

продолжение

Таблица 6.2. (*продолжение*)

Обозначение	Оператор
not	Логическое отрицание (префикс)
&string	Метка (префикс)

Примеры применения унарных операторов:

```
> [-x,x+(-x),x+(+x)];
```

$$[-x, 0, 2x]$$

```
> 100!;
>
> .234;
```

$$933262154439441526816992388562667004907159682643816214685929638952175\backslash$$
$$99993229915608941463976156518286253697920827223758251185210916864 0\backslash$$
$$0000000000000000000000$$

```
.234
> 2.34;
 2.34
> 2*%;
 4.68
> a$3;
```

$$a, a, a$$

Оператор % и команда history

Мы уже неоднократно отмечали, что оператор `%` обеспечивает подстановку в строку ввода (или в выражение) последнего результата операции, `%%` — предпоследнего и `%%%` — третьего с конца. Есть еще одна иногда полезная возможность проследить за ходом частных вычислений внутри документа — применение команды-функции `history`. В Maple V R5 это была библиотечная функция, которая требовала вызова из библиотеки. В Maple 7 такого вызова уже не требуется.

Функция `history(expr)` или `history()` создает список переменных вида `0i`, где индекс $i = 1, 2, 3....$ Этим переменным можно присваивать значения в диалоговом режиме и отслеживать результаты вычислений. Команда `off;`, вводимая после использования данной функции, завершает работу с ней. Ниже представлен диалог с применением функции `history`:

```
> history();
01 := 2;
 2
02 := sin(1.);
 .8414709848
03 := 01*02;
 1.682941970
04 := off;
```

```
> %;
 history
>
```

К сожалению, полученный результат и значения глобальных переменных Oi после завершения работы с данной функцией становятся недоступными, так что практической пользы от ее применения не так уж много. Разумеется, внутри блока history вы можете присвоить результат другой переменной и он сохранится (попробуйте это сделать сами). При каждом очередном применении функции history нумерация переменных Oi начинается с начала, так что какой-либо преемственности при использовании этой функции нет.

Функция history может применяться в качестве средства начальной отладки вычислений. Внутри фрагмента программы, заданного функцией history, можно задавать построения графиков. Например, при исполнении фрагмента программы:

```
> history();

O1 := sin(x);
 sin(x)
O2 := plot(O1,x=0..20);
O3 := off;
```

будет построен график синусоиды. В целом работа с функцией history отличается не слишком высокой стабильностью, так что возможности этой функции пока остаются не слишком востребованными.

Логические операторы

Логические (или булевы) операторы указывают на логическую связь величин (или выражений). Прежде всего они представлены рядом бинарных операторов, приведенных в табл. 6.3.

Таблица 6.3. Бинарные логические операторы

Обозначение	Оператор
<	Меньше
<=	Меньше или равно
>	Больше
>=	Большее или равно
=	Равно
<>	Не равно
and	Логическое «и»
or	Логическое «или»

Конструкции с этими операторами, такие как x=y, возвращают логическое значение — константу true, если условие выполняется, и false, если оно не выполняется. Кроме того, к логическим операторам относится унарный оператор not — он представляет логическое «нет». Для возврата логических значений выражений с этими операторами используется функция evalb(условие), например:

```
> 5<2;
```

$$5 < 2$$

```
> evalb(%);
```

false

```
> evalb(4=2+2);
```

true

```
> evalb(3<>3);
```

false

```
> evalb(not(%));
```

true

```
> evalb(3=3 and 4>2);
```

true

```
> evalb(3=3 or 2<0);
```

true

```
> evalb(x*y=y*x);
```

true

Логические операторы часто используются в управляющих структурах программ, составленных на языке программирования Maple. Такое их применение мы рассмотрим позже.

Специальные типы операторов

Операторы в Maple описывают операции по преобразованию данных, в частности выражений. Последние, в свою очередь, можно отнести к данным абстрактного типа. Могут быть описаны следующие типы операторов:

- неопределенные (`f`);
- нейтральные (`&`);
- процедурные;
- функциональные;
- композиционные (`@`).

Оператор относится к неопределенным, если он не был заранее определен. Такой оператор не выполняет никаких действий и просто повторяется в строке вывода:

```
> restart:f(1,2,a);
```

$$f(1, 2, a)$$

Композиционные операторы (на базе знака `@`) мы уже применяли. Другие типы операторов рассмотрены ниже.

Функциональные операторы

Функциональные операторы Maple-языка являются альтернативами функций и записываются в двух формах.

Нотация	Запись оператора
«arrow» (стрелочная)	vars -> result
«angle bracket» (в угловых скобках)	<result | vars>

Данные операторы могут использоваться для реализации подстановок. Например, запись `x -> x^2` означает подстановку `x^2` на место переменной `x`. Возможны и такие подстановки в множественной форме:

```
(x,y) -> x^2 + y^2
x -> (2*x, 3*x^4)
(x,y,z) -> (x*y, y*z)
```

Функциональный оператор в Maple 7 часто используется для задания функций пользователя, которое будет рассмотрено несколько позднее.

Нейтральные операторы, определяемые пользователем

Для создания нейтральных (задаваемых пользователем и в момент задания неисполняемых) операторов, определяемых пользователем, служит знак амперсанда — &. Синтаксис нейтрального оператора следующий:

```
&name
```

Имя оператора строится по правилам задания допустимых идентификаторов. Также в качестве имени может быть использована последовательность (один и более) специальных символов. В последовательности специальных символов не должно быть букв, цифр, подчеркивания, а также следующих символов:

```
&  |  (  )  {  } [  ]  :  ;  '  `  #   <перевод строки>  <пробел>
```

Максимальная длина имени — 495 символов. Нейтральные операторы могут быть унарными и бинарными. Примеры задания бинарного нейтрального оператора приведены ниже:

```
> x&/y;
```

$$x \&/ y$$

```
> z+x&/y;
```

$$z + (x \&/ y)$$

```
> &/(x,y);
```

$$x \&/ y$$

```
> x&/y-&/(x,y);
```

$$0$$

Определение операторов с помощью оператора define

Большие возможности для создания операторов с заданными свойствами предоставляет специальный оператор `define`. Он записывается в следующей форме:

```
define(oper, property1, property2, …)
```

Здесь oper — имя определяемого оператора, property1, property2 и т. д. — наименования свойств. В принципе, оператор define позволяет создавать операторы с новыми свойствами, которые отсутствуют у операторов и функций, встроенных в систему.

Могут быть указаны следующие свойства операторов:

- unary — унарный оператор;
- binary — бинарный оператор;
- diff — дифференциальный оператор;
- linear — линейный оператор;
- multilinear — *множественный линейный оператор*;
- flat — ассоциативный оператор, для которого $f(x,f(y,z)) = f(f(x,y),z) = f(x,y,z)$;
- orderless — коммутативный симметричный оператор, такой что $f(x,y) = f(y,x)$;
- antisymmetric — асимметричный оператор, такой что $f(x,y) = -f(y,x)$;
- zero — нулевой оператор (например, V:=Vector(5,shape=zero) задает вектор с 5 нулевыми элементами);
- identity — единичный оператор (например, M:=Matrix(3,3,shape=identity) задает единичную матрицу).

Следующий пример задает линейный оператор L:

```
> define(L,linear);
> L(a*x+b*x^2+c*x^3);
```

$$L(a\,x) + L(b\,x^2) + L(c\,x^3)$$

Для задания некоторых свойств операторов можно использовать уравнения и соотношения вида f(x)=value. Чтобы свойство выполнялось для всех аргументов (или некоторого класса аргументов), используется описание forall. Так, приведенный ниже пример задает оператор F, который вычисляет n-е число Фибоначчи ($n > 2$):

```
> restart;
> define(fib,fib(0)=1,fib(1)=1,fib(n::posint)=fib(n-1)+fib(n-2));
> fib(6);
```

```
 13
```

```
> fib(10);
```

```
 89
```

```
> fib(20);
```

```
 10946
```

```
> time(fib(20));
```

```
 3.414
```

Обратите внимание на то, что соотношения fib(0)=1 и fib(1)=1 задают начальные значения целочисленного массива чисел Фибоначчи, которые нужны для реализации обычного итерационного алгоритма их нахождения, — напоминаем, что очередное число Фибоначчи равно сумме двух предшествующих чисел Фибоначчи.

Последний пример иллюстрирует применение системной функции time для определения времени, затраченного на вычисление значения функции fib(20). Это время задается в секундах. Нетрудно заметить, что даже для ПК с процессором Pentium II 350 МГц это время оказалось довольно значительным (более 3 с), поскольку каждое новое число Фибоначчи вычисляется заново.

Математические функции

Понятие о встроенных функциях

Maple 7 имеет полный набор элементарных математических функций. Все они, кроме арктангенса двух аргументов, имеют один аргумент x, например sin(x). Он может быть целым, рациональным, дробно-рациональным, вещественным или комплексным числом. В ответ на обращение к ним элементарные функции возвращают соответствующее значение. Поэтому они могут быть включены в математические выражения. Все описанные здесь функции называются встроенными, поскольку они реализованы в ядре системы.

Как правило, если аргументом функции является фундаментальная константа, целое или рациональное число, то функция выводится с таким аргументом без получения результата в форме действительного числа с плавающей точкой. Например:

```
> sin(Pi);
```

$$0$$

```
> sin(1);
```

$$\sin(1)$$

```
> exp(1);
```

$$e$$

```
> ln(2);
```

$$\ln(2)$$

```
> ln(Pi);
```

$$\ln(\pi)$$

```
> arcsin(1/2);
```

$$\frac{1}{6}\pi$$

```
> arcsin(1/3);
```

$$\arcsin\left(\frac{1}{3}\right)$$

Нетрудно заметить, что есть и исключения из этого правила — например, на экране монитора exp(1) будет выведено как константа e, а значение функции arcsin(1/2) все же вычислено и результат получен как 1/6 от константы Pi. Вообще говоря, если результат выражается через фундаментальную математическую константу, то он будет вычислен и представлен ею. В противном случае функция с целочисленным и рациональным аргументом или с константой просто повторяется в строке вывода в установленном для этой строки формате.

Для получения подробной информации о некоторой произвольной функции <f> достаточно задать команду:

```
?<f>
```

Ввиду общеизвестности элементарных функций мы не будем обсуждать ни их свойства, ни допустимые для них пределы изменения аргумента.

Некоторые целочисленные функции и факториал

Ниже представлены наиболее распространенные целочисленные функции Maple 7, используемые в теории чисел:

- `factorial(n)` — функция вычисления факториала (альтернатива — оператор `!`);
- `iquo(a,b)` — целочисленное деление `a` на `b`;
- `irem(a,b)` — остаток от деления `a` на `b`;
- `igcd(a b)` — наибольший общий делитель;
- `lcm(a,b)` — наименьшее общее кратное.

Примеры применения:

```
> [factorial(10),10!];
```

$$[3628800, 3628800]$$

```
> iquo(234,5);
```

$$46$$

```
> irem(234,5);
```

$$4$$

```
> lcm(124,3);
```

$$372$$

```
> [3!!,(3!)!];
```

$$[720, 720]$$

В последних двух примерах применения оператора факториала полезно обратить внимание, что запись `n!!` означает лишь `(n!)!`, а не $n!! = 2*4*6*...$, то есть произведение четных целых чисел. Действие других функций очевидно.

Тригонометрические функции

В ядре Maple определены следующие тригонометрические функции:

- `sin` — синус;
- `cos` — косинус;
- `tan` — тангенс;
- `sec` — секанс;
- `csc` — косеканс;
- `cot` — котангенс.

Все эти функции являются периодическими (с периодом 2π, кроме тангенса и котангенса, у которых период равен π) и определены для действительного и комплексного аргументов. Примеры вычислений:

```
> [sin(1),sin(1.)];
```
$$[\sin(1), .8414709848]$$
```
> sin(x)^2+cos(x)^2;
```
$$\sin(x)^2 + \cos(x)^2$$
```
> simplify(%);
```
$$1$$
```
> simplify(tan(x)*cos(x));
```
$$\sin(x)$$
```
> sec(2+3*I);
```
$$\sec(2 + 3\,I)$$
```
> sec(2.+3*I);
```
$$-.04167496441 + .09061113720\,I$$
```
> cot(I);
```
$$-I\coth(1)$$
```
> csc(I);
```
$$-I\,\mathrm{csch}(1)$$

Многие свойства тригонометрических функций можно оценить, рассматривая их графики. Для построения таких графиков можно использовать функцию plot. На рис. 6.1 сверху показаны графики ряда тригонометрических функций.

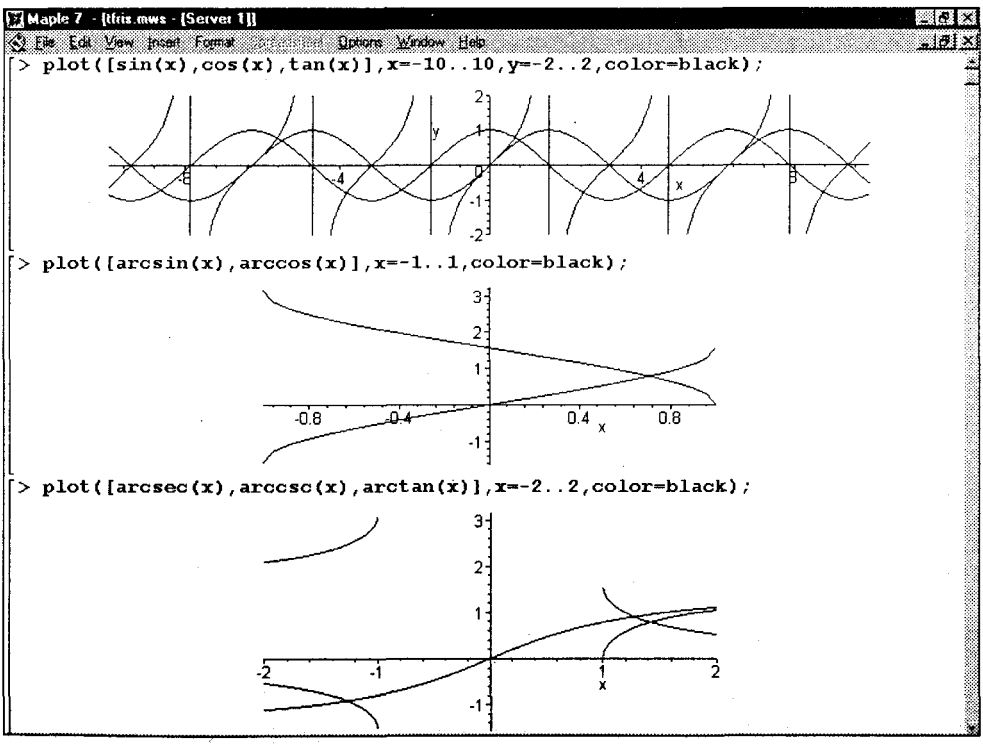
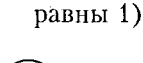


Рис. 6.1. Графики ряда тригонометрических и обратных тригонометрических функций

Из графиков тригонометрических функций (рис. 6.1, сверху) хорошо видна их периодичность. Функция тангенса имеет разрывы, и ее значение в этих точках в пределе равно бесконечности. Поэтому для наглядного ее представления вместе с функциями синуса и косинуса (их экстремальные значения по модулю равны 1) приходится вводить ограничения на масштаб графика по оси y.



ПРИМЕЧАНИЕ Обратите внимание на параметр color=black в функции построения графиков plot. Он задает построение всех графиков черным цветом, что сделано для более четкой печати их в книге. Если убрать этот параметр, то графики разных функций будут строиться с использованием разных цветов, что облегчит их различение. Другие способы выделения отдельных кривых будут описаны в дальнейшем при описании графических возможностей системы Maple 7.

Обратные тригонометрические функции

К обратным тригонометрическим относятся следующие функции:

- `arcsin` — арксинус;
- `arccos` — арккосинус;
- `arctan` — арктангенс;
- `arcsec` — арксеканс;
- `arccsc` — арккосеканс;
- `arccot` — арккотангенс.

Примеры вычислений:

```
> arcsin(.2);
```

.2013579208

```
> arcsin(2.);
```

$1.570796327 - 1.316957897\,I$

```
> evalc(arcsin(5));
```

$$\frac{1}{2}\pi - I\ln(5+2\sqrt{6})$$

```
> arccos(1/2);
```

$$\frac{1}{3}\pi$$

```
> arctan(1);
```

$$\frac{1}{4}\pi$$

```
> arccot(0);
```

$$\frac{1}{2}\pi$$

К этому классу функций принадлежит еще одна полезная функция:

```
arctan(y,x) = argument(x+I*y)
```

Она возвращает угол радиус-вектора в интервале от `-Pi` до `Pi` при координатах конца радиус-вектора `x` и `y` (см. пример ниже):

```
> arctan(2.,3);
 .5880026035
```

Графики ряда обратных тригонометрических функций показаны на рис. 6.1.

Гиперболические функции

Гиперболические функции представлены следующим набором:

- sinh — гиперболический синус;
- cosh — гиперболический косинус;
- tanh — гиперболический тангенс;
- sech — гиперболический секанс;
- csch — гиперболический косеканс;
- coth — гиперболический котангенс.

Примеры применения гиперболических функций представлены ниже:

```
> [sinh(1.),cosh(1.),tanh(1.)];
[1.175201194, 1.543080635, .7615941560]
> [sech(1.),csch(1.),coth(1.)];
[.6480542737, .8509181282, 1.313035286]
```

На рис. 6.2 сверху представлены графики гиперболического синуса, косинуса и тангенса. По ним можно судить о поведении этих функций.

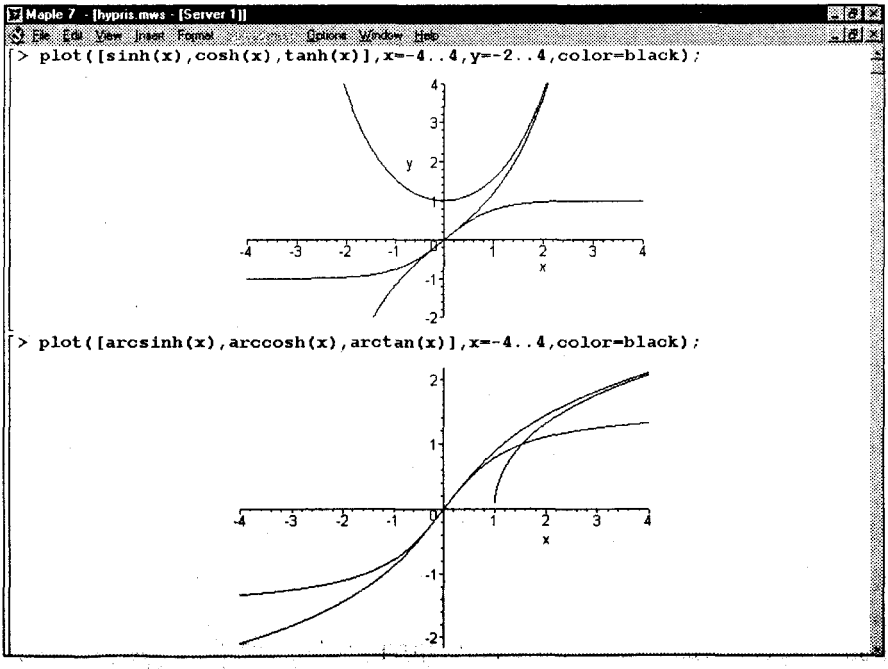


Рис. 6.2. Графики основных гиперболических и обратных гиперболических функций

ПРИМЕЧАНИЕ В отличие от тригонометрических гиперболические функции не являются периодическими. Функция гиперболического тангенса имеет симметричную кривую с характерными ограничениями. Поэтому она широко используется для моделирования передаточных характеристик нелинейных систем с ограничением выходного параметра при больших значениях входного параметра.

Обратные гиперболические функции

Как и тригонометрические функции, гиперболические имеют свои обратные функции:

- `arcsinh` — гиперболический арксинус;
- `arccosh` — гиперболический арккосинус;
- `arctanh` — гиперболический арктангенс;
- `arcsech` — гиперболический арксеканс:
- `arccsch` — гиперболический арккосеканс:
- `arccoth` — гиперболический арккотангенс.

Примеры применения:

```
> [arcsinh(1.),arccosh(1.),arctanh(1.)];
```

[.8813735870, 0., Float(∞) + Float(undefined) *I*]

Графики обратных гиперболических синуса, косинуса и тангенса представлены на рис. 6.2 снизу.

Степенные и логарифмические функции

К степенным и логарифмическим относятся следующие функции системы Maple 7:

- `exp` — экспоненциальная функция;
- `ilog10` — целочисленный логарифм по основанию 10 (возвращает целую часть от логарифма по основанию 10);
- `ilog` — целочисленный логарифм (библиотечная функция, возвращающая целую часть от натурального логарифма);
- `ln` — натуральный логарифм;
- `log` — логарифм по заданному основанию (библиотечная функция);
- `log10` — логарифм по основанию 10;
- `sqrt` — квадратный корень.

Примеры применения:

```
> x:=2;
```

$x := 2$

```
> [exp(x),ln(x),log(x),log10(x)];
```

$$\left[e^2, \ln(2), \ln(2), \frac{\ln(2)}{\ln(10)}\right]$$

```
> x:=2.0;
```

$$x := 2.0$$

```
> [exp(x),ln(x),log(x),log10(x)];
```

$$[7.389056099, .6931471806, .6931471806, .3010299957]$$

```
> ilog[2](100);
```

$$6$$

```
> readlib(log10);
```

$$\text{proc}(x) \ \ldots \ \text{end proc}$$

```
> log10(10000.);
```

$$4.000000000$$

```
> evalc(sqrt(2+3*I));
```

$$\frac{1}{2}\sqrt{4+2\sqrt{13}} + \frac{1}{2}I\sqrt{-4+2\sqrt{13}}$$

```
> sqrt(99+1);
```

$$10$$

Графики ряда алгебраических функций показаны на рис. 6.3.

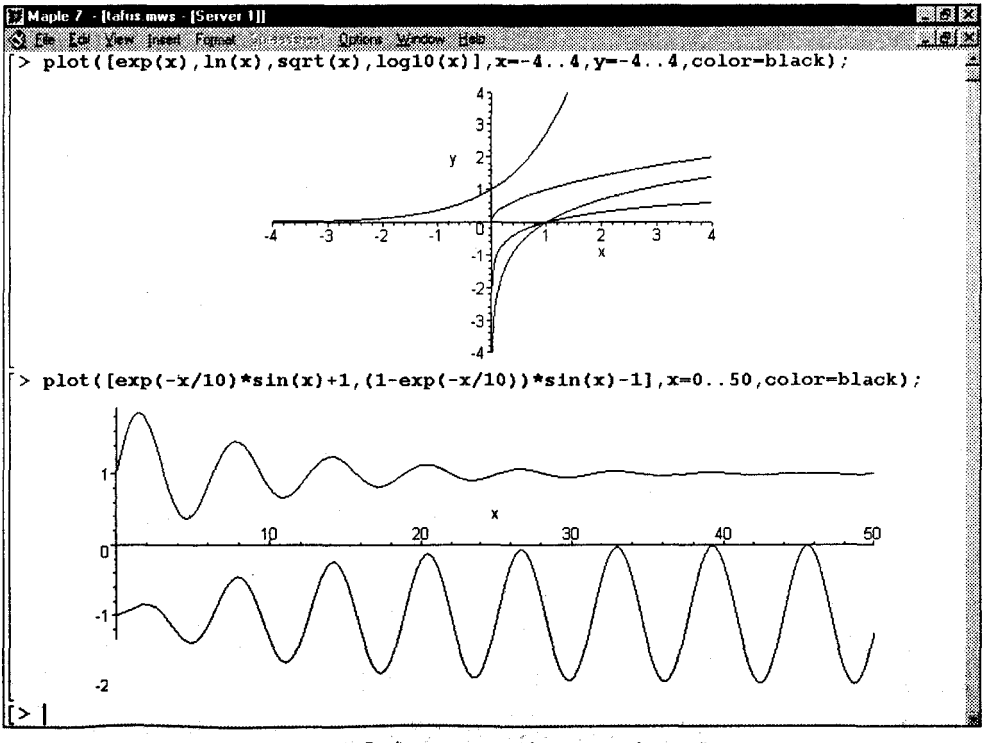


Рис. 6.3. Графики ряда алгебраических функций

На рис. 6.3 показаны также графики синусоиды с экспоненциально падающей и нарастающей амплитудой. Читателю рекомендуется попробовать свои силы в построении графиков комбинаций различных функций.

Функции с элементами сравнения

В алгоритме вычисления ряда функций заложено сравнение результата с некоторым опорным значением. К таким функциям относятся:

- `abs` — абсолютное значение числа;
- `ceil` — наименьшее целое, большее или равное аргументу;
- `floor` — наибольшее целое, меньшее или равное аргументу;
- `frac` — дробная часть числа;
- `trunc` — целое, округленное в направлении нуля;
- `round` — округленное значение числа;
- `signum (x)` — знак x (-1 при $x < 0$, 0 при $x = 0$ и $+1$ при $x > 0$).

Для комплексного аргумента x эти функции определяются следующим образом:

- $\mathrm{trunc}(x) = \mathrm{trunc}(\mathrm{Re}(x)) + I^*\mathrm{trunc}(\mathrm{Im}(x))$;
- $\mathrm{round}(x) = \mathrm{round}(\mathrm{Re}(x)) + I^*\mathrm{round}(\mathrm{Im}(x))$;
- $\mathrm{frac}(x) = \mathrm{frac}(\mathrm{Re}(x)) + I^*\mathrm{frac}(\mathrm{Im}(x))$.

Для введения определения значения `floor(x)` от комплексного аргумента прежде всего запишем $a = \mathrm{Re}(x) - \mathrm{floor}(\mathrm{Re}(x))$ и $b = \mathrm{Im}(x) - \mathrm{floor}(\mathrm{Im}(x))$. Тогда $\mathrm{floor}(x) = \mathrm{floor}(\mathrm{Re}(x)) + I^*\mathrm{floor}(\mathrm{Im}(x)) + X$, где

$$X = \begin{cases} 0, & a+b<1, \\ 1, & a+b\geq 1 \text{ и } a\geq b, \\ I, & a+b\geq 1 \text{ и } a<b. \end{cases}$$

Наконец, функция `ceil` для комплексного аргумента определяется следующим образом:

```
ceil(x) = -floor(-x)
```

Примеры применения:

```
> [ceil(Pi),trunc(Pi),floor(Pi),frac(Pi),round(Pi)];
[4, 3, 3, π - 3, 3]
> frac(evalf(Pi));
.141592654
> [ceil(-Pi),trunc(-Pi),floor(-Pi),round(-Pi)];
[-3, -3, -4, -3]
> trunc(2.6+3.4*I);
2 + 3 I
> [signum(-Pi),signum(0),signum(Pi)];
[-1, 0, 1]
```

Функции комплексного аргумента

Для комплексных чисел и данных, помимо упомянутых в предшествующем разделе, определен следующий ряд базовых функций:

- `argument` — аргумент комплексного числа;
- `conjugate` — комплексно-сопряженное число;
- `Im` — мнимая часть комплексного числа;
- `Re` — действительная часть комплексного числа;
- `polar` — полярное представление комплексного числа (библиотечная функция).

Примеры применения:

```
> z:=2+3*I;
```

$$z := 2 + 3\,I$$

```
> [Re(z),Im(z),abs(z)];
```

$$[2, 3, \sqrt{13}]$$

```
> [argument(z),conjugate(z)];
```

$$\left[\arctan\left(\frac{3}{2}\right), 2 - 3I\right]$$

```
> readlib(polar);
```

proc(*r*::*algebraic*, *th*::*algebraic*) ... end proc

```
> polar(z);
```

$$\text{polar}\left(\sqrt{13}, \arctan\left(\frac{3}{2}\right)\right)$$

```
> polar(-3.,Pi/2);
```

$$\text{polar}\left(-3., \frac{1}{2}\pi\right)$$

Специальные математические функции

Специальные математические функции обычно являются решениями линейных дифференциальных уравнений различного типа и выражаются в виде интегралов, не представимых через элементарные функции. Maple 7 имеет практически полный набор таких функций. Их представления можно найти в справочной литературе, а также в справочной базе данных Maple. В связи с этим ограничимся приведением названий наиболее важных специальных функций:

- `AiryAi (Bi)` — функции Эйри;
- `AngerJ` — функция Ангера;
- `bernoulli` — числа и полиномы Бернулли;
- `BesselI (J, K, Y)` — функции Бесселя разного рода;
- `Beta` — бета-функция;

- `binomial` — биноминальные коэффициенты;
- `Chi` — интегральный гиперболический косинус;
- `Ci` — интегральный косинус;
- `csgn` — комплексная сигнум-функция;
- `dilog` — дилогарифм;
- `Dirac` — дельта-функция Дирака;
- `Ei` — экспоненциальный интеграл;
- `EllipticCE (CK, CPi, E, F, K, Modulus, Nome, Pi)` — эллиптические интегралы;
- `erf` — функция ошибок;
- `erfc` — дополнительная функция ошибок;
- `euler` — числа и полиномы Эйлера;
- `FresnelC (f, g, S)` — интегралы Френеля;
- `GAMMA` — гамма-функция;
- `GaussAGM` — арифметико-геометрическое среднее Гаусса;
- `HankelH1 (H2)` — функции Ганкеля;
- `harmonic` — частичная сумма серии гармоник;
- `Heaviside` — функция Хевисайда;
- `JacobiAM (CN, CD, CS, DN, DC, DS, NC, ND, NS, SC, SD, SN)` — эллиптические функции Якоби;
- `JacobiTheta1 (2, 3, 4)` — дзета-функции Якоби;
- `JacobiZeta` — зет-функция Якоби;
- `KelvinBer (Bei, Her, Hei, Ker, Kei)` — функции Кельвина;
- `Li` — логарифмический интеграл;
- `lnGAMMA` — логарифмическая гамма-функция;
- `MeijerG` — G-функция Мейджера;
- `pochhammer` — символ Похгамера;
- `polylog` — полилогарифмическая функция;
- `Psi` — дигамма-функция;
- `Shi` — интегральный гиперболический синус;
- `Si` — интегральный синус;
- `Ssi` — синусный интеграл смещения;
- `StruveH (L)` — функции Струве;
- `surd` — неглавная корневая функция;
- `LambertW` — W-функция Ламберта;

- `WeberE` — E-функция Вебера;
- `WeierstrassP` — P-функция Вейерштрасса;
- `WeierstrassPPrime` — производная P-функции Вейерштрасса;
- `WeierstrassZeta` — зета-функция Вейерштрасса;
- `WeierstrassSigma` — сигма-функция Вейерштрасса;
- `Zeta` — зета-функция Римана и Гурвица.

Ввиду большого числа специальных функций и наличия множества примеров их вычисления в справочной системе Maple 7 ограничимся несколькими примерами вычисления наиболее распространенных специальных функций. По их подобию читатель может опробовать в работе и другие специальные функции.

На рис. 6.4 даны примеры применения ряда специальных функций. Обратите особое внимание на первый пример. Он показывает, как средствами системы Maple 7 задается определение функций Бесселя. Показано, что функции Бесселя являются решениями заданного на рис. 6.4 дифференциального уравнения второго порядка. Maple 7 способна вычислять производные и интегралы от специальных функций.

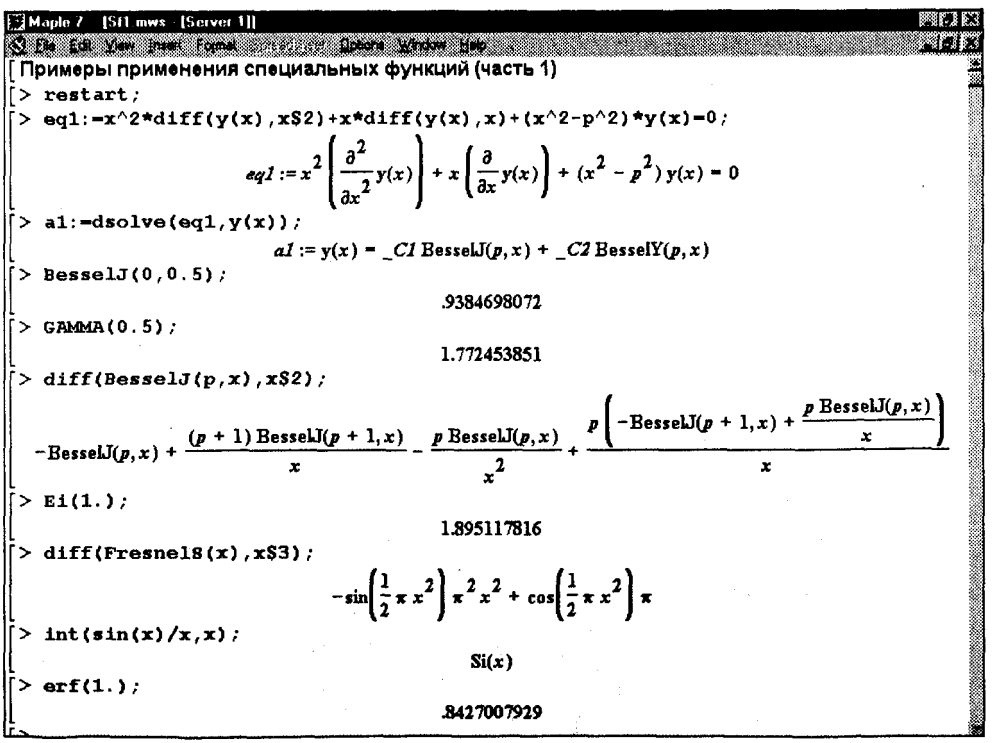


Рис. 6.4. Примеры применения специальных функций

Еще несколько примеров работы со специальными функциями представлены на рис.6.5. Как видно из приведенных примеров, на экране монитора можно получить математически ориентированное представление специальных функций, обычно более предпочтительное, чем представление на Maple-языке или в текстовом формате. Записи функций при этом выглядят как в обычной математической литературе.

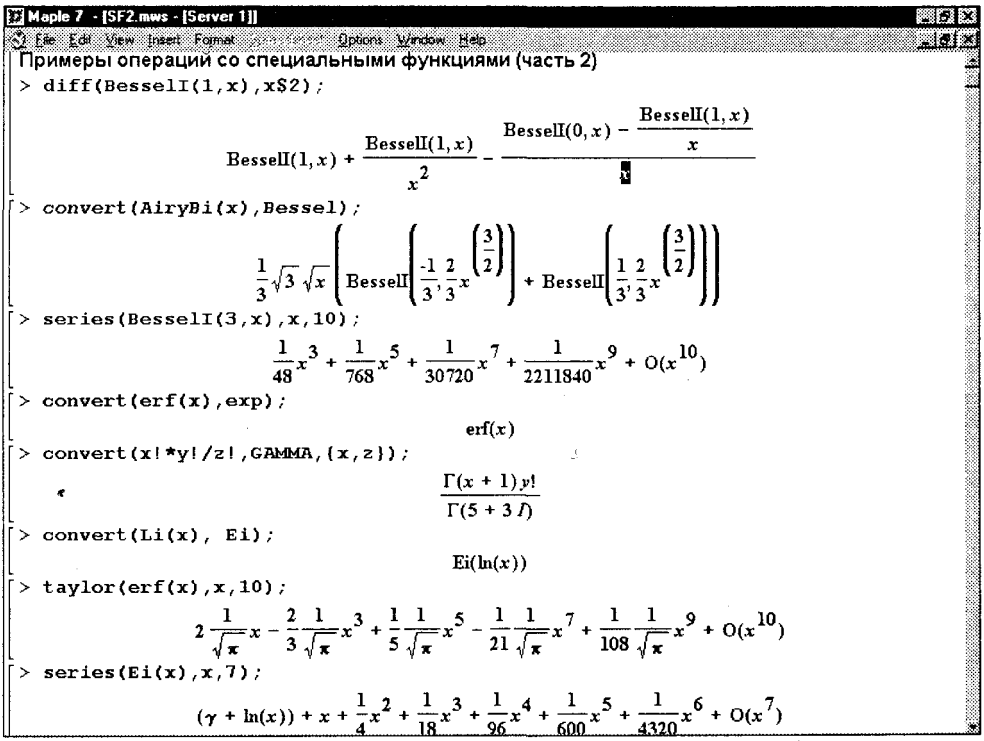


Рис. 6.5. Примеры работы со специальными математическими функциями

На рис. 6.5 показаны примеры разложения специальных функций в ряды и применения функции convert для их преобразования.

Много информации о поведении специальных функций дает построение их графиков. На рис. 6.6 показано построение семейства графиков функций Бесселя BesselJ разного порядка и гамма-функции. Эти функции относятся к числу наиболее известных. Если читателя интересуют те или иные специальные функции, следует прежде всего построить и изучить их графики.

Подробное описание специальных функций можно найти в справочниках [43–45] и в справочной базе данных Maple 7.

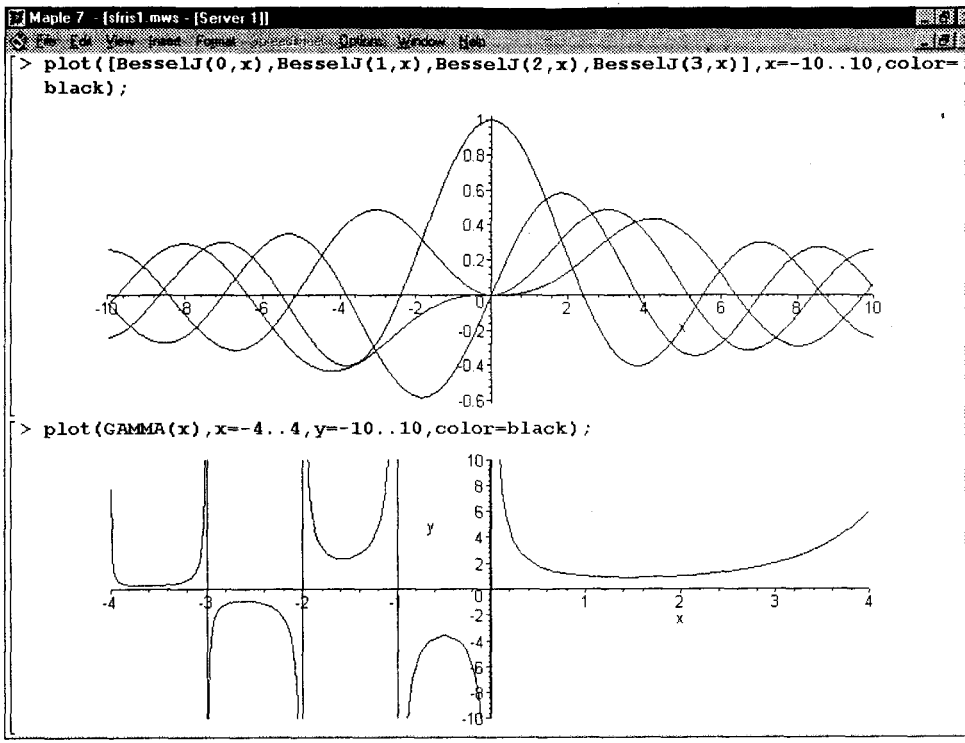


Рис. 6.6. Графики функций Бесселя и гамма-функции

Функции для работы с векторами и матрицами

Элементы векторов и матриц

Элементы векторов и матриц являются *индексированными переменными*, то есть место каждого элемента вектора определяется его индексом, а у матрицы — двумя индексами. Обычно их обобщенно обозначают как i (номер строки матрицы или порядковый номер элемента вектора) и j (номер столбца матрицы). Допустимы операции вызова нужного элемента и присваивания ему нового значения:

- `V[i]` — вызов i-го элемента вектора `V`;
- `M[i,j]` — вызов элемента матрицы `M`, расположенного на i-й строке в j-м столбце;
- `V[i]:=x` — присваивание нового значения `x` i-му элементу вектора `V`;
- `M[i,j]:=x` — присваивание нового значения `x` элементу матрицы `M`.

Преобразование списков в векторы и матрицы

Прежде всего надо обратить внимание на то, что векторы и матрицы хотя и похожи на списки, но не полностью отождествляются с ними. В этом можно убедиться с помощью следующих примеров, в которых функция type используется для контроля типов множественных объектов (векторов и матриц):

```
> M1:=[1,2,3,4];
```

$$M1 := [1, 2, 3, 4]$$

```
> type(M1,vector);
```

false

```
> V:=convert(M1,vector);
```

$$V := [1, 2, 3, 4]$$

```
> type(V,vector);
```

true

```
> M2:=[[1,2],[3,4]];
```

$$M2 := [[1, 2], [3, 4]]$$

```
> type(M2,matrix);
```

false

```
> M:=convert(M2,matrix);
```

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

```
> type(M,matrix);
```

true

Однако, используя функцию преобразования данных convert, можно преобразовывать одномерные списки в векторы, а двумерные — в матрицы. Функция type используется в следующих формах:

- type(V,vector) — тестирует аргумент V и возвращает true, если V — вектор, и false в ином случае;
- type(M,matrix) — тестирует аргумент M и возвращает true, если M — матрица, и false в ином случае.

Здесь параметры vector и matrix используются для указания того, какой тип объекта проверяется.

ПРИМЕЧАНИЕ Обратите внимание на то, что матрицы отображаются иначе, чем двумерные списки, — без двойных квадратных скобок. Отображение вектора подобно отображению одномерного списка, поэтому здесь особенно важен контроль типов данных.

Операции с векторами

Важное достоинство систем компьютерной алгебры, к которым относится и Maple 7, заключается в возможности выполнения аналитических (символьных) операций над векторами и матрицами.

Приведем примеры операций над векторами:

```
> V:=array(1..4,[1,2,3,4]);
```

$$V := [1, 2, 3, 4]$$

```
> [V[1],V[2],V[4]];
```

$$[1, 2, 4]$$

```
> V[1]:=a:V[3]:=b:
> evalm(V);
```

$$[a, 2, b, 4]$$

```
> evalm(V+2);
```

$$[a+2, 4, b+2, 6]$$

```
> evalm(2*V);
```

$$[2\,a, 4, 2\,b, 8]$$

```
> evalm(V**V);
```

$$[a, 2, b, 4]^V$$

```
> evalm(a*V);
```

$$[a^2, 2\,a, a\,b, 4\,a]$$

В этих примерах используется функция `evalm(M)`, осуществляющая вычисление матрицы или вектора `M`.

ПРИМЕЧАНИЕ

Рекомендуется перед проведением символьных операций с векторами и матрицами очистить память от предшествующих определений с помощью команды restart. Если какие-то элементы векторов или матриц были ранее определены, это может привести к очень сильным искажениям вида конечных результатов. Очистка памяти устраняет возможность ошибок такого рода.

Операции над матрицами с численными элементами

Над матрицами с численными элементами можно выполнять разнообразные операции. Ниже приведены основные из них:

```
> M:=array(1..2,1..2,[[1,2],[3,4]]);
```

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

```
> evalm(2*M);
```

$$\begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$$

```
> evalm(2+M);
```

$$\begin{bmatrix} 3 & 2 \\ 3 & 6 \end{bmatrix}$$

```
> evalm(M^2);
```

$$\begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$$

```
> evalm(M^(-1));
```

$$\begin{bmatrix} -2 & 1 \\ \frac{3}{2} & \frac{-1}{2} \end{bmatrix}$$

```
> evalm(M-M);
```

$$0$$

```
> evalm(M+M);
```

$$\begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$$

```
> evalm(M*M);
```

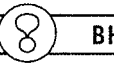
$$\begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$$

```
> evalm(M/M);
```

$$1$$

```
> evalm(M^0);
```

$$1$$



ВНИМАНИЕ Рекомендуется внимательно изучить эти примеры и попробовать свои силы в реализации простых матричных операций. Мы вернемся к гораздо более серьезному описанию матричных операций и функций в дальнейшем.

Символьные операции с матрицами

Ниже представлены примеры символьных операций, осуществляемых над квадратными матрицами одного размера:

```
> M1:=array(1..2,1..2,[[a1,b1],[c1,d1]]);
```

$$M1 := \begin{bmatrix} a1 & b1 \\ c1 & d1 \end{bmatrix}$$

```
> M2:=array(1..2,1..2,[[a2,b2],[c2,d2]]);
```

$$M2 := \begin{bmatrix} a2 & b2 \\ c2 & d2 \end{bmatrix}$$

```
> evalm(M1+M2);
```

$$\begin{bmatrix} a1 + a2 & b1 + b2 \\ c1 + c2 & d1 + d2 \end{bmatrix}$$

```
> evalm(M1-M2);
```

$$\begin{bmatrix} a1 - a2 & b1 - b2 \\ c1 - c2 & d1 - d2 \end{bmatrix}$$

```
> evalm(M1&*M2);
```

$$\begin{bmatrix} a1\,a2 + b1\,c2 & a1\,b2 + b1\,d2 \\ c1\,a2 + d1\,c2 & c1\,b2 + d1\,d2 \end{bmatrix}$$

```
> evalm(M1/M2);
```

$$\begin{bmatrix} \dfrac{a1\,d2}{a2\,d2 - b2\,c2} - \dfrac{b1\,c2}{a2\,d2 - b2\,c2} & -\dfrac{a1\,b2}{a2\,d2 - b2\,c2} + \dfrac{b1\,a2}{a2\,d2 - b2\,c2} \\ \dfrac{c1\,d2}{a2\,d2 - b2\,c2} - \dfrac{d1\,c2}{a2\,d2 - b2\,c2} & -\dfrac{c1\,b2}{a2\,d2 - b2\,c2} + \dfrac{d1\,a2}{a2\,d2 - b2\,c2} \end{bmatrix}$$

```
> evalm(M1&/M2);
```

$$\begin{bmatrix} a1 \,\&/ \begin{bmatrix} a2 & b2 \\ c2 & d2 \end{bmatrix} & b1 \,\&/ \begin{bmatrix} a2 & b2 \\ c2 & d2 \end{bmatrix} \\ c1 \,\&/ \begin{bmatrix} a2 & b2 \\ c2 & d2 \end{bmatrix} & d1 \,\&/ \begin{bmatrix} a2 & b2 \\ c2 & d2 \end{bmatrix} \end{bmatrix}$$

Приведем еще ряд примеров выполнения символьных операций с одной матрицей:

```
> evalm(M1^2);
```

$$\begin{bmatrix} a1^2 + b1\,c1 & a1\,b1 + b1\,d1 \\ c1\,a1 + d1\,c1 & b1\,c1 + d1^2 \end{bmatrix}$$

```
> evalm(sin(M1));
```

$$\begin{bmatrix} \sin(a1) & \sin(b1) \\ \sin(c1) & \sin(d1) \end{bmatrix}$$

```
> evalm(M1*z);
```

$$\begin{bmatrix} z\,a1 & z\,b1 \\ z\,c1 & z\,d1 \end{bmatrix}$$

```
> evalm(M1/z);
```

$$\begin{bmatrix} \frac{a1}{z} & \frac{b1}{z} \\ \frac{c1}{z} & \frac{d1}{z} \end{bmatrix}$$

```
> evalm(M1+z);
```

$$\begin{bmatrix} a1 + z & b1 \\ c1 & d1 + z \end{bmatrix}$$

```
> evalm(M1-z);
```

$$\begin{bmatrix} a1 - z & b1 \\ c1 & d1 - z \end{bmatrix}$$

Среди других функций для работы с матрицами полезно обратить внимание на функцию `map`, которая применяет заданную операцию (например, функции дифференцирования `diff` и интегрирования `int`) к каждому элементу матрицы. Примеры такого рода даны ниже:

```
> M:=array(1..2,1..2,[[x,x^2],[x^3,x^4]]);
```

$$M := \begin{bmatrix} x & x^2 \\ x^3 & x^4 \end{bmatrix}$$

```
> map(diff,M,x);
```

$$\begin{bmatrix} 1 & 2\,x \\ 3\,x^2 & 4\,x^3 \end{bmatrix}$$

```
> map(int,%,x);
```

$$\begin{bmatrix} x & x^2 \\ x^3 & x^4 \end{bmatrix}$$

```
> map(sin,M);
```

$$\begin{bmatrix} \sin(x) & \sin(x^2) \\ \sin(x^3) & \sin(x^4) \end{bmatrix}$$

В результате возвращаются матрицы, каждый элемент которых представлен производной или интегралом. Аналогично можно выполнять над матрицами и другие достаточно сложные преобразования.

В дальнейшем мы продолжим изучение матричных функций и операций, включенных в пакеты Maple 7.

Функции для работы со строковыми данными

Контроль типа строковых данных

Напоминаем, что строковые данные представляются совокупностью любых символов в обратных апострофах, например `Привет` или `2+2`. Для контроля объектов на принадлежность к строковым данным служит функция type с параметром string:

```
> str:=`Hello!`;
```

str := Hello!

```
> type(Hello,string);
```

false

```
> type(str,string);
```

false

```
> type(2+3,string);
```

false

```
> type(`2+3`,string);
```

false

```
> char:=a;
```

char := a

```
> char:=`a`;
```

char := a

Из приведенных примеров видно, что контроль строкового типа осуществляется не очень строго, — в частности, единичные символы рассматриваются как строковые и без заключения их в апострофы. В строках могут быть символы кириллицы, но гарантии в правильности обработки таких символов нет — надо мириться с тем, что Maple — англоязычная программа и ее возможности в поддержке других языков ограниченны.

Интерактивный ввод строк

Для интерактивного ввода строк можно использовать функцию `readline(filename)`, задав в качестве имени файла `terminal` или опустив имя файла. В этом слунае ввод строки осуществляется с клавиатуры компьютера:

```
> s:=readline();
> Привет мой друг!
s:="Привет мой друг!"
```

ПРИМЕЧАНИЕ Полезно обратить внимание на то, что запрос в ходе интерактивного ввода может быть сделан на русском языке (если установленный для запросов шрифт имеет символы кириллицы). Нужно также, чтобы и шрифт строки вывода содержал кириллицу, иначе в строке вывода будет типичная «абракадабра» — смесь непонятных символов.

Обработка строк

Имеется ряд функций для работы со строками. Из них наиболее важны следующие:

- `lenght(str)` — возвращает число символов, содержащихся в строке `str`;
- `substring(str,a..b)` — возвращает подстроку строки `str` от `a`-го символа до `b`-го;
- `cat(str1,str2,...)` — возвращает строку, полученную объединением строк `str1`, `str2`,... (альтернатива — оператор конкатенации в виде точки .);
- `SearchText(s,str)` — производит поиск подстроки `s` в строке `str` и при его успехе возвращает номер позиции `s` в строке `str` (при отсутствии `s` в `str` функция возвращает 0).

Примеры применения этих функций (в виде продолжения ранее приведенных примеров) представлены ниже:

```
> length(str);
 6
> substring(str,1..3);
 Hel
> substring(str,4..6);
 lo!
> s:=cat(`Hello`,` my`,` friend!`);
 s := Hello my friend!
> SearchText(my,s);
 7
> ss:=`Hello `||`my friend!`;
>
 ss := Hello my friend!
> seq(Name||i,i=1..4);
 Name1, Name2, Name3, Name4
```

Эти функции дают достаточно средств для обработки данных строкового типа, которые можно применять не только для создания текстовых комментариев, но и для управления вычислительным процессом в программах.

Преобразование строки в математическое выражение

Часто возникает необходимость в интерактивном вводе математических выражений. Для ввода с запросом выражения используется функция readstat(promt), где promt — строка с текстовым комментарием. Пример ее применения дан ниже:

```
> y:=readstat(`Введите выражение `);
Введите выражение a*x^2+b;
```

$$y := a\,x^2 + b$$

Альтернативой может стать ввод строкового выражения с последующим преобразованием его в математическое выражение с помощью функции parse:

```
> s:=`2+3*5`;
```

$$s := 2+3*5$$

```
> evaln(s);
```

$$s$$

```
> parse(%);
```

17

Обратите внимание на то, что функция evaln не смогла вычислить строковое выражение `2+3`, поскольку оно не является числовым типом данных. Однако функция parse преобразовала это выражение в числовое, что и привело к его вычислению.

Что нового мы узнали?

В этом уроке мы научились:

- Использовать операторы и операнды.
- Применять различные математические функции.
- Использовать операторы и функции для работы с векторами и матрицами.
- Использовать функции для работы со строковыми данными.

Типовые средства программирования

- ❒ Функции пользователя
- ❒ Условные операторы
- ❒ Циклы for, while и do
- ❒ Операторы пропуска и прерывания
- ❒ Процедуры и процедуры-функции
- ❒ Средства контроля и отладки процедур
- ❒ Работа с отладчиком программ
- ❒ Операции ввода–вывода
- ❒ Вывод в специальных форматах
- ❒ Дополнительные возможности Maple-языка

Функции пользователя

Упрощенные функции пользователя

Хотя ядро Maple 7, библиотека и пакеты содержат свыше 3000 функций, всегда может оказаться, что именно нужной пользователю (и порою довольно простой) функции все же нет. Тогда возникает необходимость в создании собственной функции, именуемой функцией пользователя. Начнем описание с обычных функций пользователя, задающих некоторую зависимость от одной или ряда переменных в явном виде.

Основным средством расширения Maple-языка являются модули — процедуры. Однако на первый взгляд они довольно сложны. Есть и более простые способы задания функций пользователя. Один из таких способов заключается просто в присваивании введенной функции (в виде выражения) некоторой переменной:

```
Name:=Выражение
```

Этот прием фактически означает просто операцию присваивания. Следующие примеры иллюстрируют технику работы с такими функциями:

```
> m:=sqrt(x^2+y^2);
```

$$m := \sqrt{x^2 + y^2}$$

```
> x:=3:y:=4:m;
```

$$\sqrt{25}$$

```
> evalf(m);
```

5.000000000

Заданный таким образом объект все же не является полноценной функцией пользователя, и прежде всего потому, что в нем используются только глобальные переменные (x и y) и нет объявленного списка параметров, от которых зависит значение функции. При этом значения переменных функции приходится заведомо задавать отдельно, используя операции присваивания. Подобные конструкции нельзя ввести в библиотеки Maple 7.

Основной способ задания функции пользователя

Более гибкий способ задания полноценных функций пользователя базируется на применении функционального оператора. При этом используется следующая конструкция:

```
name:=(x,y,…)->expr
```

После этого вызов функции осуществляется в виде name(x,y,...), где (x,y,...) — список формальных параметров функции пользователя с именем name. Переменные, указанные в списке формальных параметров, являются локальными. При подстановке на их место фактических параметров они сохраняют их значения только в теле функции (expr). За пределами этой функции переменные с этими именами оказываются либо неопределенными, либо сохраняют ранее присвоенные им значения.

Следующие примеры иллюстрируют сказанное:

```
> restart;
> x:=0;y:=0;
```

$$x := 0$$

$$y := 0$$

```
> m:=(x,y)->sqrt(x^2+y^2);
```

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

```
> m(3,4);
```

$$5$$

```
> m(3.,4);
```

$$5.000000000$$

```
> [x,y];
```

$$[0, 0]$$

Нетрудно заметить, что при вычислении функции m(x,y) переменные x и y имели значения 3 и 4, однако за пределами функции они сохраняют нулевые значения, заданные им перед введением определения функции пользователя.

Еще один способ задания функции пользователя базируется на применении функции unapply:

```
name:=unapply(expr,var1,var2,...)
```

Ниже даны примеры такого задания функции пользователя:

```
> restart;
> fm:=unapply(sqrt(x^2+y^2),x,y);
```

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

```
> fm(4.,3.);
```

$$5.000000000$$

```
> fe:=unapply(x^2+y^2,x,y);
```

$$fe := (x, y) \rightarrow x^2 + y^2$$

```
> fe(sin(x),cos(x));
```

$$\sin(x)^2 + \cos(x)^2$$

```
> simplify(fe(sin(x),cos(x)));
```

$$1$$

Последний пример показывает возможность проведения символьных операций с функцией пользователя.

Графическая визуализация результатов выполнения функций пользователя

В ряде случаев весьма желательна визуализация результатов выполнения функций пользователя. Порой она может давать неожиданный результат. На рис. 7.1 представлены примеры задания двух функций пользователя от двух переменных и построение их графиков с помощью функции plot3d.

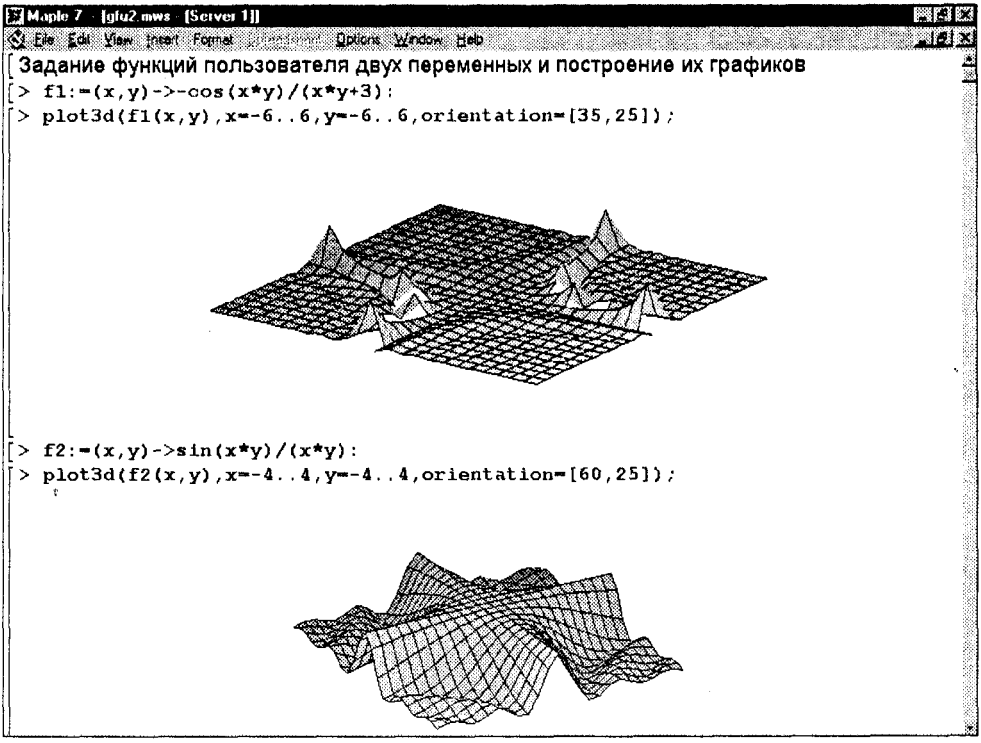


Рис. 7.1. Примеры задания функций пользователя двух переменных с построением их графиков

ВНИМАНИЕ При задании функций пользователя рекомендуется просмотреть их графики в нужном диапазоне изменения аргументов. К сожалению, наглядными являются только графики функций одной и двух переменных.

Импликативные функции

Другой важный класс функций, которые нередко приходится задавать, — импликативные функции, в которых связь между переменными задана неявно, в виде какого-либо выражения. Самый характерный пример такой функции — это выражение для задания окружности радиуса r: $x^2 + y^2 = r^2$.

Итак, импликативные функции записываются как уравнения. Соответственно их можно решать с помощью функции solve. Следующие примеры иллюстрируют задание уравнения окружности в общем и в частном (численном) виде:

```
> impf:=x^2+y^2=r^2;
```

$$impf := x^2 + y^2 = r^2$$

```
> subs(x=a,impf);
```

$$a^2 + y^2 = r^2$$

```
> solve(%);
```

$$\{a = \sqrt{-y^2 + r^2}, y = y, r = r\}, \{y = y, r = r, a = -\sqrt{-y^2 + r^2}\}$$

```
> impf1:=x^2+y^2=25;
```

$$impf1 := x^2 + y^2 = 25$$

```
> subs(x=4,impf1);
```

$$16 + y^2 = 25$$

```
> solve(%);
```

$$3, -3$$

Для графической визуализации импликативных функций служит функция implicitplot пакета plots. На рис. 7.2 представлено задание двух импликативных функций и построение их графиков.

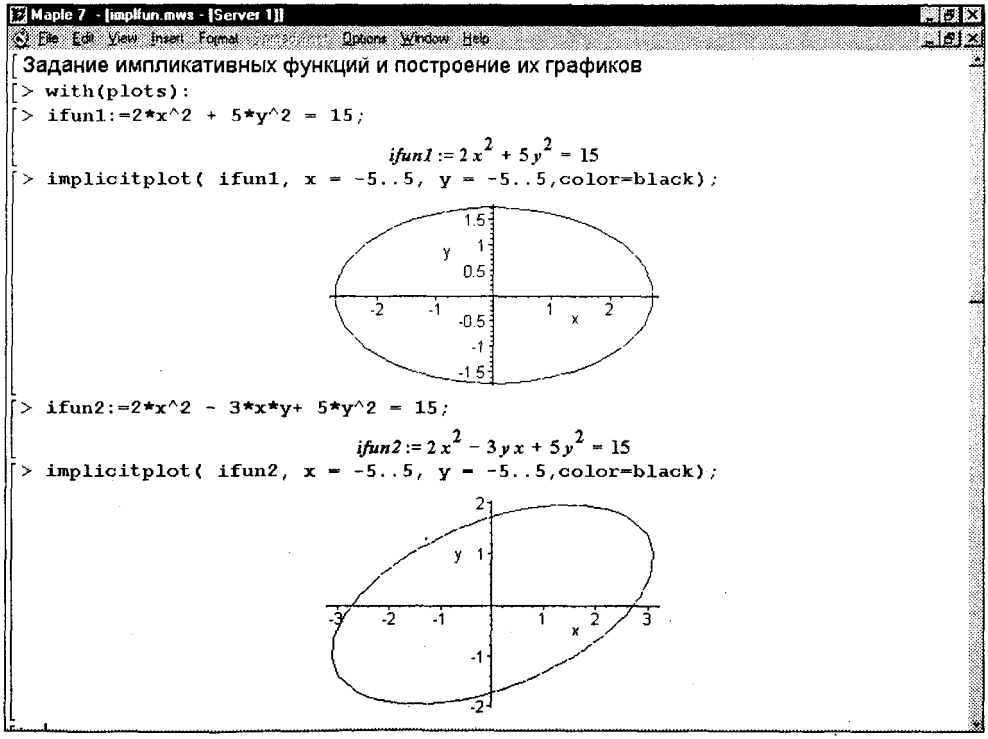


Рис. 7.2. Задание двух импликативных функций и построение их графиков

В данном случае задано построение двух эллипсов. Верхний — это окружность, сплюснутая по вертикали, а второй — наклонный эллипс.

Условные выражения

Для подготовки разветвляющихся программ в Maple-язык программирования включен оператор `if`, позволяющий создавать следующую конструкцию:

```
if <Условие сравнения> then <Элементы>
|elif <Условие сравнения> then <Элементы>|
|else <Элементы>|
fi;
```

В вертикальных чертах | | указаны необязательные элементы данной конструкции. Следующих два вида условных выражений чаще всего используются на практике:

- `if <Условие> then <Элементы 1> fi` — если Условие выполняется, то исполняются Элементы 1, иначе ничего не выполняется;
- `if <Условие> then <Элементы 1> else <Элементы 2> fi` — если Условие выполняется, то исполняются Элементы 1, иначе исполняются Элементы 2.

В задании условий используются любые логические конструкции со знаками сравнения (<, <=, >, >=, =, <>) и логические операторы `and`, `or` и `not`, конструкции с которыми возвращают логические значения `true` и `false`.

Рассмотрим следующий простой пример:

```
> x:=-5:
> if x<0 then print(`Negative`) fi;
```

Negative

```
> x:=1:
> if x<0 then print(`Negative`) fi;
```

В этом примере анализируется значение x. Если оно отрицательно, то с помощью функции вывода `print` на экран выводится сообщение «Negative». А вот если x неотрицательно, то не выводится никакого сообщения. В другом примере если x неотрицательно, то выводится сообщение «Positive»:

```
> x:=-5:
> if x<0 then print(`Negative`) else print(`Positive`) fi;
```

Negative

```
> x:=1:
> if x<0 then print(`Negative`) else print(`Positive`) fi;
```

Positive

Приведем еще один пример, показывающий особую форму задания конструкции `if-then-else-fi`:

```
> x:=-5:
> `if` (x<0, print(`Negative`),print(`Positive`));
```

Negative

```
> x:=1:
> `if` (x<0, print(`Negative`),print(`Positive`));
```

Positive

В этой конструкции вида

```
`if `(Условие, Выражение1, Выражение2)
```

если Условие выполнятся, то будет исполнено Выражение1, в противном случае будет исполнено Выражение2. Ввиду компактности записи такая форма условного выражения нередко бывает предпочтительна, хотя она и менее наглядна. На рис. 7.3 представлено применение данной конструкции для моделирования трех типов сигналов.

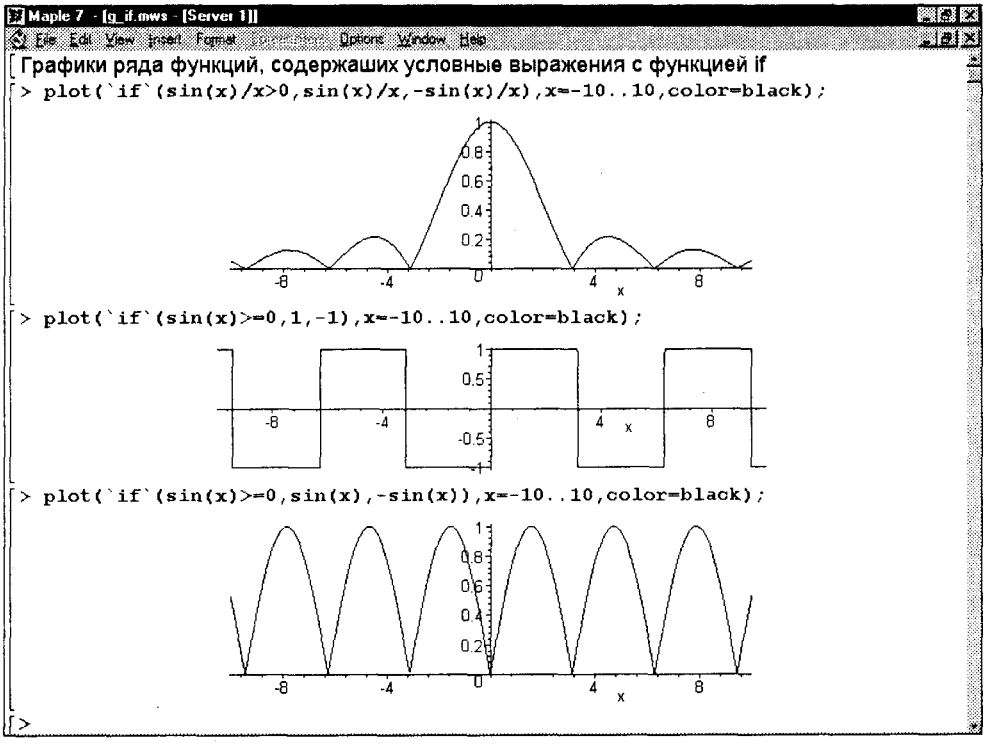


Рис. 7.3. Применение конструкции с функцией if для моделирования сигналов

К сожалению, функции на базе конструкции `if` не всегда корректно обрабатываются функциями символьной математики. Поэтому надо тщательно контролировать полученные в этом случае результаты.

Циклы for и while

Зачастую необходимо циклическое повторение выполнения выражения заданное число раз или до тех пор, пока выполняется определенное условие. Maple 7 имеет обобщенную конструкцию цикла, которая задается следующим образом:

```
|for <name>| |from <expr1>| |to <expr3>| |by <expr2>| |while <expr4>|
do <statement sequence> od;
```

Здесь name — имя управляющей переменной цикла, expr1, expr2 и expr3 — выражения, задающие начальное значение, конечное значение и шаг изменения переменной name, expr4 — выражение, задающее условие, пока цикл (набор объектов между словами do и od) будет выполняться.

В ходе выполнения цикла управляющая переменная меняется от значения expr1 до значения expr2 с шагом, заданным expr3. Если блок by <expr2> отсутствует, то управляющая переменная будет меняться с шагом +1 при expr1<expr2. Это наглядно поясняет следующий пример:

```
> for i from 1 to 5 do print(i) od;
```

```
1
2
3
4
5
```

В нем выводятся значения переменной i в ходе выполнения цикла. Нетрудно заметить, что она и впрямь меняется от значения 1 до значения 5 с шагом +1. Следующий пример показывает, что границы изменения управляющей переменной можно задать арифметическими выражениями:

```
> for i from 7/(2+5) to 2+3 do print(i) od;
```

```
1
2
3
4
5
```

А еще один пример показывает задание цикла, у которого переменная цикла меняется от значения 1 до 10 с шагом 2:

```
> for i from 1 to 10 by 2 do print(i) od;
```

```
1
3
5
7
9
```

В этом случае выводятся нечетные числа от 1 до 9. Шаг может быть и отрицательным:

```
> for i from 9 to 1 by -2 do print(i) od;
```

```
9
7
5
3
1
```

Следует отметить, что если expr1>expr2 задать заведомо невыполнимое условие, например, expr1>expr2 и положительное значение шага, то цикл выполняться не будет. Цикл можно прервать с помощью дополнительного блока while <expr4>. Цикл с таким блоком выполняется до конца или до тех пор, пока условие expr4 истинно.

```
> for i from 1 to 10 by 2 while i<6 do print(i) od;
```

```
1
3
5
```

Таким образом, конструкция цикла в Maple-языке программирования вобрала в себя основные конструкции циклов for и while. Есть еще одна, более специфическая конструкция цикла:

```
|for <name>| |in <expr1>| |while <expr2>| do <statement sequence> od;
```

Здесь expr1 задает список значений, которые будет принимать управляющая переменная name. Цикл будет выполняться, пока не будет исчерпан список и пока выполняется условие, заданное выражением expr2. Следующий пример иллюстрирует сказанное:

```
> for i in [1,2,5,-1,7,12] do print(i) od;
```

```
1
2
5
-1
7
12
```

```
> for i in [1,2,5,-1,7,12] while i>0 do print(i) od;
```

```
1
2
5
```

В цикле этого вида управляющая переменная может меняться произвольно.

Циклы могут быть вложенными. Это иллюстрирует следующий пример, создающий единичную матрицу на базе заданного массива M:

```
> M:=array(1..3,1..3);
```

$$M := \text{array}(1 .. 3, 1 .. 3, [\ ])$$

```
> for i to 3 do for j to 3 do M[i,j]:=0;if i=j then M[i,j]:=1 fi; od od;
> evalm(M);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Этот пример имеет не более чем познавательное значение, поскольку для создания такой матрицы Maple 7 имеет функцию identity, с помощью которой функция array позволяет сразу создать единичную матрицу:

```
> array(1..3,1..3,identity);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

В заключение отметим, что возможна упрощенная частная конструкция цикла типа while:

```
while expr do statseq od;
```

Здесь выражения statseq выполняются, пока выполняется логическое условие expr. Пример такого цикла:

```
> n:=1;
```

$$n := 1$$

```
> while n<16 do n:=2*n od;
```

$$n := 2$$

$$n := 4$$

$$n := 8$$

$$n := 16$$

В этом примере идет удвоение числа n с начальным значением $n = 1$ до тех пор, пока значение n меньше 16.

Операторы пропуска и прерывания

Иногда бывает нужным пропустить определенное значение переменной цикла. Для этого используется оператор next (следующий). Приведенный ниже пример иллюстрирует применение оператора next в составе выражения if-fi для исключения вывода значения $i = -2$:

```
> for i in [1,2,3,-2,4] do if i=-2 then next else print(i) fi od;
```

1

2

3

4

Другой оператор — break — прерывает выполнение фрагмента программы (или цикла). Его действие поясняет слегка модифицированный предшествующий пример:

```
> for i in [1,2,3,-2,4] do if i=-2 then break else print(i) fi od;
```

1

2

3

В данном случае при значении $i = -2$ произошло полное прекращение выполнения цикла. Поэтому следующее значение 4 переменной i присвоено не было и это значение на печать не попало.

Любой из операторов `quit`, `done` или `stop` обеспечивает также прерывание выполнения текущей программы (в частности, цикла), но при этом окно текущего документа закрывается.

Процедуры и процедуры-функции

Простейшие процедуры

Процедурой называют модуль программы, имеющий самостоятельное значение и выполняющий одну или несколько операций, обычно достаточно сложных и отличных от операций, выполняемых встроенными операторами и функциями.

Процедуры являются важнейшим элементом структурного программирования и служат средством расширения возможностей системы Maple 7 пользователем. Каждая процедура имеет свое уникальное имя и список параметров (он может быть пустым). Процедуры вызываются так же, как встроенные функции, — указанием их имени со списком фактических параметров. При этом *просто процедуры* обычно не возвращают каких-либо значений после своего исполнения, хотя могут присваивать значения входящим в них переменным. *Процедуры-функции* в ответ на обращение к ним возвращают некоторое значение. Они как бы являются новыми функциями, задаваемыми пользователем. Описанные ранее функции пользователя фактически являются процедурами-функциями с несколько упрощенной структурой.

Простейшая форма задания процедуры следующая:

```
name := proc(Параметры)
          Тело процедуры
end;
```

Параметры процедуры задаются перечислением имен переменных, например `proc(x)` или `proc(x,y,z)`. С помощью знака `::` после имени переменной можно определить ее тип, например в объявлении `prog(n::integer)` объявляется, что переменная `n` является целочисленной.

При вызове процедуры выражением вида:

```
name(Фактические_параметры)
```

фактические параметры подставляются на место формальных. Несоответствие фактических параметров типу заданных переменных ведет к сообщению об ошибке и к отказу от выполнения процедуры.

В качестве примера ниже приведена процедура вычисления модуля комплексного числа `z` — в данном случае это единственный параметр процедуры:

```
> modc:=proc(z)
> evalf(sqrt(Re(z)^2+Im(z)^2))
> end;
```

$$modc := \mathbf{proc}(z)\ \mathbf{evalf}(\mathbf{sqrt}(\Re(z)\text{^}2 + \Im(z)\text{^}2))\ \mathbf{end\ proc}$$

ПРИМЕЧАНИЕ После ввода заголовка процедуры под строкой ввода появляется сообщение: «Warning, premature end of input». Оно указывает на то, что ввод листинга процедуры не закончен и должен быть продолжен до тех пор, пока не будет введено завершающее слово end листинга процедуры. Если после этого слова поставить символ точки с запятой, то листинг процедуры будет выведен на экран дисплея.

Теперь для вычисления модуля достаточно задать обращение к процедуре modc(z), указав вместо z конкретное комплексное число:

```
> modc(3.+I*4.);
```

5.000000000

Нетрудно заметить, что при знаке ; после завершающего слова end текст процедуры повторяется в строке вывода (в общем случае в несколько ином виде). Если это повторение не нужно, после слова end надо поставить знак двоеточия. Обратите также внимание на то, что для обозначения действительной и мнимой частей процедуры в ее тексте появились готические буквы.

Оператор возврата значения RETURN

Как отмечалось, процедуры, которые возвращают значение результата в ответ на обращение к ним, во многом тождественны функциям. Будем называть их процедурами-функциями. Обычно процедура возвращает значение последнего выражения в ее теле или выражения, намеченного к возврату специальным оператором RETURN:

```
> modc:=proc(z)
> evalf(sqrt(Re(z)^2+Im(z)^2)):
> RETURN(%)
>end;
```

modc := proc (*z*) evalf(sqrt(ℜ(*z*)^2 + ℑ(*z*)^2)); RETURN(%) end proc

```
> modc(3.+I*4.);
```

5.000000000

Параметром оператора RETURN может быть любое выражение. В Maple не принято выделять процедуры-функции в какой-то отдельный класс. Действует правило — если не использован оператор RETURN, процедура возвращает значение последнего выражения в ее теле. Для устранения выдачи значений выражений внутри процедуры-функции после них просто надо установить знак двоеточия.

Статус переменных в процедурах и циклах

Переменные, которые указываются в списке параметров (например, z, в нашем случае), внутри процедуры являются *локальными*. Это означает, что изменение их значений происходит лишь в теле процедуры, то есть локально. За пределами тела процедуры эти переменные имеют то значение, которое у них было до использования процедуры. Это хорошо поясняет следующий пример:

```
> restart:z:=1;
```

z := 1

```
> modc:=proc(z)
> evalf(sqrt(Re(z)^2+Im(z)^2));
> end;
```

modc := proc (*z*) evalf(sqrt(ℜ(*z*)^2 + ℑ(*z*)^2)) end proc

```
> modc(3.+I*4.);
```

5.000000000

```
> z;
```

1

Нетрудно заметить, что внутри процедуры $z = 3 + I*4$, тогда как вне ее значение $z = 1$. Таким образом, имена переменных в списке параметров процедуры могут совпадать с именами глобальных переменных, используемых за пределами процедуры.

Переменные, которым впервые присваивается значение в процедуре, также относятся к локальным. Кроме того, переменные, применяемые для организации циклов, являются локальными. Все остальные переменные — глобальные.

Объявления переменных локальными с помощью оператора local

Если в теле процедуры имеются операции присваивания для ранее определенных (глобальных) переменных, то изменение их значений в ходе выполнения процедуры создает так называемый побочный эффект. Он способен существенно изменить алгоритм решения сложных задач и, как правило, недопустим.

Поэтому Maple-язык программирования имеет встроенные средства для исключения побочных эффектов. Встречая такие операции присваивания, Maple-язык корректирует текст процедуры и вставляет в нее объявление переменных локальными с помощью ключевого слова `local` и выдает предупреждающую надпись о подобном применении:

```
> restart:m:=0;
```

m := 0

```
> modc:=proc(z)
> m:=evalf(sqrt(Re(z)^2+Im(z)^2)):RETURN(m)
> end;
```

Warning, `m` is implicitly declared local to procedure `modc`

modc := proc (*z*) local *m*; *m* := evalf(sqrt(ℜ(*z*)^2 + ℑ(*z*)^2)); RETURN(*m*) end proc

```
> modc(3.+I*4.);
```

5.000000000

```
> m;
```

0

Обратите внимание на то, что в тело процедуры было автоматически вставлено определение `local m`, задающее локальный статус переменной `m`. Оператором `print` можно вывести текст процедуры:

```
> print(modc);
```

proc (*z*) local *m*; *m* := evalf(sqrt(ℜ(*z*)^2 + ℑ(*z*)^2)); RETURN(*m*) end proc

Объявления переменных глобальными с помощью слова global

Говорят, что запретный плод сладок! Что бы ни говорили о нежелательности работы с глобальными переменными, бывает, что их применение желательно или даже необходимо. Чтобы сделать переменные внутри процедуры *глобальными*, достаточно объявить их с помощью ключевого слова `global`, после которого перечисляются идентификаторы переменных.

Следующий пример поясняет применение оператора `global` в процедуре:

```
> a:=1;b:=1;
```

$a := 1$

$b := 1$

```
> fg:=proc(x,y)
> global a,b;
> a:=x^2:b:=y^2:
> RETURN(sqrt(a+b));
> end;
```

$fg := \mathrm{proc}(x, y)\ \mathrm{global}\ a, b;\ a := x\text{^}2;\ b := y\text{^}2;\ \mathrm{RETURN}(\mathrm{sqrt}(a + b))\ \mathrm{end\ proc}$

```
> fg(3,4);
```

5

```
> [a,b];
```

[9, 16]

В примере переменным `a` и `b` вначале присвоены значения 1. Поскольку они в процедуре объявлены глобальными, то внутри процедуры они принимают новые значения x^2 и y^2. В результате при выходе из процедуры они имеют уже новые значения. Это и есть побочный эффект при исполнении данной процедуры. Если пользователь не знает (или не помнит), что та или иная процедура имеет побочный эффект, то он рискует получить самые неожиданные (и неверные) результаты своих вычислений.

ПРИМЕЧАНИЕ Следует отметить, что нельзя делать глобальными переменные, указанные в списке параметров процедуры, поскольку они уже фактически объявлены локальными. Такая попытка приведет к появлению сообщения об ошибке следующего вида «Error, argument and global `x` have the same name». При этом соответствующие переменные останутся локальными.

Функция вывода сообщений об ошибках ERROR

При профессиональной подготовке процедур пользователь должен предусмотреть их поведение при возможных ошибках. Например, если он готовит процедуру или функцию, вычисляющую квадратный корень из действительных чисел, то надо учесть, что такой корень нельзя извлекать из отрицательных чисел (будем, исключительно в учебных целях, считать, что комплексные числа в данном примере недопустимы).

Для контроля за типом данных обычно используются различные функции оценки и тестирования. При выявлении ими ошибки, как правило, предусматривается вывод соответствующего сообщения. Для этого используется функция ERROR:

```
ERROR(expr_1, expr_2, ...)
```

где expr_1, ... — ряд выражений (возможно, пустой). Наиболее часто ERROR выводит просто строковое сообщение об ошибке, например ERROR(`strings`). Полное сообщение об ошибке имеет вид:

```
Error, (in name) string, ...
```

Приведем пример процедуры, в которой предусмотрен вывод сообщения об ошибке при задании переменной $x < 0$:

```
> f := proc (x) if x<0 then error "invalid variable x: %1", x else x^(1/2) end if end proc;
```

f := proc (x) if $x < 0$ then error "invalid variable x: %1", x else sqrt(x) end if end proc

```
> f(3.);
```

1.732050808

```
> f(-3.);
```

Error, (in f) invalid variable x: -3.

```
> lasterror;
```

"invalid variable x: %1", -3.

```
> lastexception;
```

f, "invalid variable x: %1", -3.

Эта процедура вычисляет квадратный корень из числа x. При $x < 0$ выводится заданное сообщение об ошибке. Еще раз обращаем внимание читателя на учебный характер данного примера, поскольку вычисление квадратного корня (в том числе из комплексных и отрицательных действительных чисел) реализовано встроенной функцией sqrt.

Ключи в процедурах

В *объявление процедуры можно включить ключевые слова, вводимые словом*

```
options opseq
```

Иногда их называют *расширяющими ключами*. Предусмотрены следующие ключи:

- arrow — определят процедуру-оператор в нотации ->;
- builtin — *определяет функцию как встроенную;*
- call_external — задает обращение к внешним программным модулям;
- copyright — *защищает процедуру от копирования;*
- inline — определяет процедуру как подчиненную (возможно, не для всех процедур – см. справку);
- load=memberName — загружает нужный для определений процедуры модуль (см. также опцию unload и детали в справке);
- operator — объявляет процедуру — функциональный оператор;

- `system` — определяет процедуру как системную,
- `remember` — определяет таблицу памяти для процедуры;
- `trace` — задает трассировку процедуры;
- `unload=memberName` — выгружает нужный для определения процедуры модуль (см. опцию `load`).

Ключ remember

Ключ `remember` обеспечивает занесение результатов обращений к процедуре в таблицу памяти, которая используется при исполнении процедуры. Функция `op` позволяет вывести таблицу:

```
> f:=proc(x) options remember; x^3 end:
> f(2);
```

8

```
> f(3);
```

27

```
> op(4,eval(f));
```

table([2 = 8, 3 = 27])

Ключ `remember` особенно полезен при реализации итерационных процедур. К примеру, в приведенной ниже процедуре (без использования ключа `remember`) время вычисления n-го числа Фибоначчи растет пропорционально квадрату n:

```
> f:=proc(n) if n<2 then n else f(n-1)+f(n-2) fi end;
```

$f := \mathbf{proc}(n)\ \mathbf{if}\ n < 2\ \mathbf{then}\ n\ \mathbf{else}\ f(n-1) + f(n-2)\ \mathbf{end\ if}\ \mathbf{end\ proc}$

```
> time(f(30));
```

27.400

```
> f(30);
```

832040

Вычисление `f(30)` по этой процедуре на ПК с процессором Pentium II 350 МГц занимает около 30 с — см. контроль этого времени с помощью функции `time` (результат в секундах).

Стоит добавить в процедуру ключ `remember`, и время вычислений резко уменьшится:

```
> restart;
> fe:=proc(n) options remember; if n<2 then n else fe(n-1)+fe(n-2) fi end:
> fe(30);
```

832040

```
> time(fe(30));
```

0.

При этом вычисление `fe(30)` происходит практически мгновенно, так как все промежуточные результаты в первом случае вычисляются заново, а во втором они берутся из таблицы. Однако это справедливо лишь тогда, когда к процедуре было хотя бы однократное обращение. Обратите внимание на то, что данные процедуры являются *рекурсивными* — в их теле имеется обращение к самим себе.

Ключ builtin

Ключ builtin придает процедуре статус встроенной. Он должен использоваться всегда первым. С помощью функции eval(name) можно проверить, является ли функция с именем name встроенной:

```
> eval(type);
        proc() option builtin; 268 end proc
> eval(print);
        proc() option builtin; 229 end proc
```

Числа в теле процедур указывают системные номера функций. Следует отметить, что в новой версии Maple 7 они существенно отличаются от принятых в предшествующих версиях.

Ключ system

Этот ключ придает процедуре статус системной. У таких процедур таблица памяти может быть удалена. У обычных процедур таблица памяти не удаляется и входит в так называемый «мусорный ящик» (garbage collector).

Ключи operator и arrow

Эта пара ключей задает процедуре статус оператора в «стрелочной» нотации (->). Это достаточно пояснить следующими примерами:

```
> o:=proc(x,y) option operator , arrow; x-sqrt(y) end;
```

$$o := (x, y) \rightarrow x - \sqrt{y}$$

```
> o(4,2);
```

$$4 - \sqrt{2}$$

```
> o(4,2.);
```

2.585786438

Ключ trace

Ключ trace задает вывод отладочной информации:

```
> o:=proc(x,y) option trace, arrow; x-sqrt(y) end;
```

$o :=$ proc(x, y) option *trace*, *arrow*; $x -$ sqrt(y) end proc

```
> o(4,2.);
{--> enter o, args = 4, 2.
```

2.585786438

```
<-- exit o (now at top level) = 2.585786438}
```

2.585786438

Ключ copyright

Этот ключ защищает тело процедуры от просмотра. Это поясняют следующие два примера:

```
> o:=proc(x,y) x-sqrt(y) end;
```

$o :=$ proc(x, y) $x -$ sqrt(y) end proc

```
> oo:=proc(x,y) option Copyright; x-sqrt(y) end;
```

$oo := \mathbf{proc}(x, y)$ ... **end proc**

```
> oo(4,2.);
```

2.585786438

Нетрудно заметить, что во втором примере тело процедуры уже не просматривается. Для отмены защиты от просмотра можно использовать оператор `interface(verboseproc=2)`.

Общая форма задания процедуры

Выше мы рассмотрели основные частные формы задания процедур. Все они могут быть объединены в общую форму задания процедуры:

```
name:=proc(<argseq>)              # объявление процедуры
        local<nseq>;                    # объявление локальных переменных
        global<nseq>;                   # объявление глобальных переменных
        options<nseq>;                  # объявление расширяющих ключей
        description<stringseq>;         # объявление комментарий
        <stateq>                        # выражения - тело процедуры
end; (или end:)                   # объявление конца процедуры
```

Эта форма охватывает все описанные выше частные формы и позволяет готовить самые сложные и надежно работающие процедуры.

Средства контроля и отладки процедур

Большая часть функций и операторов системы Maple 7 реализована в виде процедур, написанных на Maple-языке программирования. Благодаря возможности их просмотра пользователь получает неисчерпаемый источник примеров программирования на этом языке. Кроме того, пользователь может создавать свои собственные процедуры.

Для контроля и отладки процедур прежде всего надо уметь вывести их текст. Для этого служит функция:

```
print(name);
```

где `name` — имя процедуры.

Однако перед тем, как использовать эту функцию, надо исполнить команду:

```
> interface(verboseproc=2,prettyprint=1,version);
Maple Worksheet Interface, Maple 7, IBM INTEL NT, May 28 2001 Build ID 96223
```

Ее смысл будет пояснен ниже. Пока же отметим, что эта команда обеспечивает полный вывод текста процедур библиотеки. Встроенные в ядро процедуры, написанные не на Maple-языке, в полном тексте не представляются. Поясним это следующими примерами:

```
> print(evalf);
    proc() option builtin, remember; 167 end proc
> print(erf);
    proc(x::algebraic)
```

```
local Re_x, Im_x, sr, si, xr, xi;
option `Copyright (c) 1994 by the University of Waterloo. All rights reserved.`;
    if nargs <> 1 then error "expecting 1 argument, got %1",
    nargs
    elif type(x, 'complex(float)') then evalf('erf'(x))
    elif x = 0 then 0
    elif type(x, 'infinity') then
            if type(x, {'cx_infinity', 'undefined'}) thenundefined + undefined*I
            elif type(Re(x), 'infinity') then CopySign(1, Re(x))
            elif type(x, 'imaginary') then x
            else infinity + infinity*I
            end if
    elif type(x, 'undefined') then x*undefined
    elif type(x, 'complex(numeric)') then
            if csgn(x) = -1 then -erf(-x) else 'erf'(x) end if
    elif type(x, `*`) and type(op(1, x), 'complex(numeric)')
    and csgn(op(1, x)) = -1 then -erf(-x)
    elif type(x, `+`) and traperror(sign(x)) = -1 then -
    erf(-x)
    else erf(x) := 'erf'(x)
    end if
end proc
```

Здесь вначале выполнен вывод сокращенного листинга встроенной в ядро процедуры `evalf`, а затем выведен полный листинг процедуры вычисления функции ошибок `erf`. Эта функция имеет довольно короткую процедуру — многие важные функции и операторы задаются гораздо более сложными и большими процедурами.

Но вернемся к функции `interface`. Эта функция служит для управления выводом и задается в виде:

```
interface( arg1, arg2, … )
```

где аргументы задаются в виде равенств вида `name=value` и слов-указателей:

ansi	autoassign	echo	errorbreak	errorcursor
imaginaryunit	indentamount	labelling	labelwidth	latexwidth
longdelim	patchlevel	plotdevice	plotoptions	plotoutput
postplot	preplot	prettyprint	prompt	quiet
rtablesize	screenheight	screenwidth	showassumed	verboseproc
version	warnlevel			

К сожалению, объем и характер данной книги не позволяют остановиться на всех вариантах использования этой очень мощной функции, тем более что в ней может использоваться множество аргументов. Мы рассмотрим только некоторые, наиболее важные возможности.

Указание `verboseproc=n` задает степень детальности вывода листинга процедур. При `n=0` текст не выводится, при `n=1` выводится текст только заданных пользователем процедур, а при `n=2` — всех процедур на Maple-языке. Пример этого был дан выше. Указание `prettyprint=0` или 1 управляет выводом стандартных сообщений. Указание `plotdevice=string` управляет выводом графики, например `plotdevice=gif` указывает на то, что запись графиков в виде файлов будет происходить в формате .gif.

Одним из основных средств отладки процедур является функция трассировки trace(name). Детальность ее работы задается системной переменной printlevel (уровень вывода). При printlevel:=n (значение n = 1 по умолчанию) выводится результат только непосредственно исполняемой функции или оператора. Для вывода информации о выполнении k-го уровня вложенности надо использовать значение этой переменной от $5*k$ до $5*(k+1)$. Так, при n от 1 до 5 выводятся результаты трассировки первого уровня, при n от 6 до 10 — второго и т. д. Максимальное значение n = 100 обеспечивает трассировку по всем уровням вложенности процедуры name.

Следующий пример показывает осуществление трассировки для функции int(x^n,x):

```
> printlevel:=5;
```

$$printlevel := 5$$

```
> trace(int);
{--> enter trace, args = int
<-- exit trace (now at top level) = int}
```

$$int$$

```
> int(x^n,x);
{--> enter int, args = x^n, x
```

$$answer := \frac{x^{(n+1)}}{n+1}$$

$$\frac{x^{(n+1)}}{n+1}$$

```
<-- exit int (now at top level) = x^(n+1)/(n+1)}
```

$$\frac{x^{(n+1)}}{n+1}$$

Действие функции трассировки отменяется командой untrace:

```
> untrace(int);
{--> enter untrace, args = int
<-- exit untrace (now at top level) = }
> int(x^n,x);
{--> enter int, args = x^n, x
<-- exit int (now at top level) = x^(n+1)/(n+1)}
```

$$\frac{x^{(n+1)}}{n+1}$$

```
> printlevel:=1;
```

$$printlevel := 1$$

```
> int(x^n,x);
```

$$\frac{x^{(n+1)}}{n+1}$$

При отладке алгоритмов выполнения вычислений надо тщательно следить за сообщениями об ошибках. Для этого в Maple предусмотрены функция traceerr

и системная переменная lasterr, в которой сохраняется последнее сообщение об ошибке. При каждом обращении к tracerr переменная lasterr очищается:

```
> 2/0;
```
Error, numeric exeption:division by zero
```
> 2/4;
```
1/2
```
> 2/.3;
```
6.666666667
```
> lasterror;
```
"division by zero"
```
> traperror(3/4);
```
3/4
```
> lasterror;
```
lasterror
```
> traperror(5/0);
```
Error, numeric exeption:division by zero
```
> lasterror;
```
"numeric exeption:division by zero"

Этот пример показывает, как может быть проведено отслеживание ошибок в ходе вычислений. Вообще говоря, пользователь системы Maple 7 редко привлекает описанные средства, поскольку проще отладить вычислительный алгоритм прежде, чем на его основе будет составлена процедура. При правильном построении процедур ошибочные ситуации заведомо предусматриваются и должным образом обрабатываются.

Работа с отладчиком программ

В большинстве случаев составители программ (процедур) редко прибегают к пошаговой их отладке. Средства общей диагностики Maple 7 развиты настолько хорошо, что позволяют выявлять грубые ошибки в процедурах при их выполнении. Иногда, правда, для этого приходится неоднократно «прогонять» процедуру, пока она не начнет работать как задумано. Тем не менее для отладки процедур служит специальный интерактивный отладчик (debugger). Опишем, как его запустить и как с ним работать.

Допустим, мы составили некоторую процедуру demo, вычисляющую сумму квадратов чисел (1^2+2^2+...+n^2):

```
> demo:=proc(n::integer) local y,i:
>   y:=0:
>   for i to n do y:=y+i^2 od
> end;
```
demo := **proc**(*n*::*integer*) **local** *y*, *i*; *y* := 0; **for** *i* **to** *n* **do** *y* := *y* + *i*^2 **end do** **end proc**
```
> demo(3);
```
14

Чтобы включить отладчик в работу, надо исполнить команду stopat:

```
> stopat(demo);
 [demo]
> demo(3);
demo:
    1*  y := 0;
DBG>
```

Признаком, указывающим на работу отладчика, является изменение приглашения к вводу со знака > на DBG> (как нетрудно догадаться, DBG означает *debugger*). Теперь, подавая команды next (следующий), step (шаг) и stop (остановка), можно проследить выполнение процедуры:

```
DBG> next
0
demo:
    2       for i to n do
              ...
            end do

DBG> step
0
demo:
    3       y := y+i^2

DBG> step
1
demo:
    3       y := y+i^2

DBG> step
5
demo:
    3       y := y+i^2

DBG> step
 14
```

В последнем случае процедура по шагам дошла до конца вычислений; на этом работа отладчика завершается сама собой.

Можно также вывести листинг процедуры с помощью команды showstat:

```
> showstat(demo);
demo := proc(n::integer)
local y, i;
    1*      y := 0;
    2       for i to n do
    3       y := y+i^2
            end do
end proc
```

Обратите внимание, что в этом листинге строки вычисляемых элементов пронумерованы. Это сделано для облегчения разбора работы процедуры.

В общем случае отладчик выключается при выполнении команд stopat, stopwhen или stoperr. Если используется команда stopat, то вывод на экран соответствует исполнению последней выполненной команды. Для отмены этой команды используется команда unstopat.

Команда stopwhen позволяет установить точку наблюдения за указанной в команде переменной. Отменить ее можно командой unstopwhen. Команда stoperror позволяет задать остановку при появлении определенной ошибки. Для отмены этой команды используется команда unstoperror.

Команда cont используется для продолжения работы до следующей точки прерывания, установленной указанными выше командами, или до конца процедуры. Для прерывания отладки можно использовать команду quit. После команды stop можно вычислить любое Maple-выражение.

В действительности команд отладчика намного больше и их функции более развиты, чем это описано выше. Пользователи, заинтересованные в серьезной работе с отладчиком (скорее всего, их немного), могут просмотреть его подробное описание. Для этого в разделе справочной системы Context найдите раздел Programming, а в нем — раздел Debugging.

Операции ввода и вывода

Считывание и запись программных модулей

В уроке 2 рассматривалась работа с файлами документов. Вводимые в текущий документ программные модули хранятся вместе с ним, так что при отказе от загрузки какого-либо документа все его программные блоки не могут использоваться в других документах. Кроме того, порой неудобно загружать объемный документ ради использования одного или нескольких модулей, например процедур. Поэтому в Maple 7 введены средства, позволяющие записывать нужные модули (в том числе результаты вычислений) на диск и считывать их в случае необходимости.

Для записи на диск используется оператор save:

- save filename — запись всех определений текущего файла под именем filename;
- save name_1, name_2, ..., name_k, filename — запись избранных модулей с именами name_1, name_2, ..., name_k под именем filename.

Считывание имеющегося на диске файла filename осуществляется оператором read:

```
read <filename>
```

При считывании все имеющиеся в файле определения становятся доступными для рабочих документов Maple. При записи файлов отдельных определений используется специальный внутренний Maple-формат файлов. Для загрузки файлов типа *.m из стандартной библиотеки используется функция readlib. А для записи файлов в качестве библиотечных достаточно в имени filename оператора

save указать расширение .m. Разумеется, можно считывать такие файлы и оператором read, указав в имени файла расширение .m:

```
> save my_proc,`my_lib.m`: # запись файла my_proc и
> # библиотечного файла my_lib.m;
> load `my_lib.m`: # считывание библиотечного файла
> # my_lib.m.
```

Создание своей библиотеки процедур

Если приведенные выше примеры составления процедур кажутся вам простыми, значит, вы неплохо знаете программирование и, скорее всего, уже имеете несколько полезных процедур, которые вы хотели бы сохранить — если не для потомков, то хотя бы для своей повседневной работы. Сделать это в Maple 7 довольно просто.

Прежде всего надо определить имя своей библиотеки, например mylib, и создать для нее на диске каталог (папку) с заданным именем. Процедуры в Maple 7 ассоциируются с таблицами. Поэтому вначале надо задать таблицу-пустышку под будущие процедуры:

```
> restart;
> mylib:=table();
```

$$mylib := \text{table}([])$$

Теперь надо ввести свои библиотечные процедуры. Они задаются с двойным именем — вначале указывается имя библиотеки, а затем в квадратных скобках имя процедуры. Для примера зададим три простые процедуры с именами f1, f2 и f3:

```
> mylib[f1]:=proc(x::anything) sin(x)+cos(x) end:
> mylib[f2]:=proc(x::anything) sin(x)^2+cos(x)^2 end:
> mylib[f3]:=proc(x::anything) if x=0 then 1 else sin(x)/x fi end:
```

Рекомендуется тщательно проверить работу процедур, прежде чем записывать их на диск. Ограничимся, скажем, такими контрольными примерами:

```
> mylib[f1](x);
```

$$\sin(x) + \cos(x)$$

```
> mylib[f1](1.);
```

$$1.381773291$$

```
> mylib[f2](x);
```

$$\sin(x)^2 + \cos(x)^2$$

```
> simplify(mylib[f2](x));
```

$$1$$

```
> evalf(mylib[f3](x));
```

$$\frac{\sin(x)}{x}$$

```
> sin(0)/0;
   Error, division by zero

> mylib[f3](0);
```

$$1$$

```
> evalf(mylib[f3](.5));
```

.9588510772

Можно построить графики введенных процедур-функций. Они представлены на рис. 7.4.

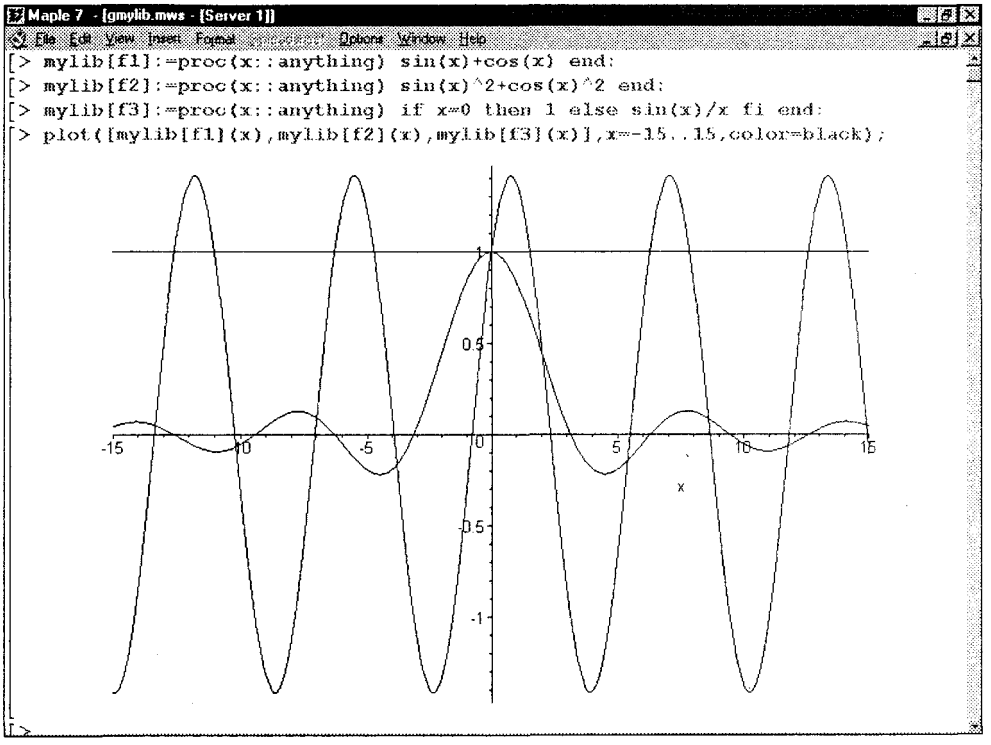


Рис. 7.4. Построение графиков процедур-функций f1, f2 и f3

С помощью функции `with` можно убедиться, что библиотека `mylib` действительно содержит только что введенные в нее процедуры. Их список должен появиться при обращении `with(mylib)`:

```
> with(mylib);
```

[*f1*, *f2*, *f3*]

Теперь надо записать эту библиотеку под своим именем на диск с помощью команды `save`:

```
> save(mylib,`c:/mylib.m`);
```

Обратите особое внимание на правильное задание полного имени файла. Обычно применяемый для указания пути знак \ в строках Maple-языка используется как знак продолжения строки. Поэтому надо использовать либо двойной знак \\, либо знак /. В нашем примере файл записан в корень диска C. Лучше поместить библиотечный файл в другую папку (например, в библиотеку, уже имеющуюся в составе системы), указав полный путь до нее.

После всего этого надо убедиться в том, что библиотечный файл записан. После этого можно сразу и считать его. Для этого вначале следует командой restart устранить ранее введенные определения процедур:

```
> restart;
```

С помощью команды with можно убедиться в том, что этих определений уже нет:

```
> with(mylib);
    Error, (in pacman:-pexports) mylib is not a package
```

После этого командой read надо загрузить библиотечный файл:

```
> read(`c:/mylib.m`);
```

Имя файла надо указывать по правилам, указанным для команды save. Если все выполнено пунктуально, то команда with должна показать наличие в вашей библиотеке списка процедур f1, f2 и f3:

```
> with(mylib);
[f1, f2, f3]
```

И наконец, можно вновь опробовать работу процедур, которые теперь введены из загруженной библиотеки:

```
> f1(x);
```

$$\sin(x) + \cos(x)$$

```
> simplify(f2(y));
```

$$1$$

```
> f3(0);
```

$$1$$

```
> f3(1.);
```

$$.8414709848$$

Описанный выше способ создания своей библиотеки вполне устроит большинство пользователей. Однако есть более сложный и более «продвинутый» способ ввода своей библиотеки в состав уже имеющейся. Для реализации этого Maple 7 имеет следующие операции записи в библиотеку процедур s1, s2, ... и считывания их из файлов file1, file2, ...:

```
savelib(s1, s2, ..., sn, filename)
readlib(f, file1, file2, ...)
```

С помощью специального оператора makehelp можно задать стандартное справочное описание новых процедур:

```
makehelp(n,f,b).
```

где n — название темы, f — имя текстового файла, содержащего текст справки (файл готовится как документ Maple) и b — имя библиотеки. Системная переменная libname хранит имя директории библиотечных файлов. Для регистрации созданной справки надо исполнить команду вида:

```
libname:=libname, `/mylib`;
```

С деталями применения этих операторов можно ознакомиться в справочной системе.

К созданию своих библиотечных процедур надо относиться достаточно осторожно. Их применение лишает ваши Maple-программы совместимости со стандартной версией Maple 7. Если вы используете одну-две процедуры, проще поместить их в те документы, в которых они действительно нужны. Иначе вы будете вынуждены к каждой своей программе прикладывать еще и библиотеку процедур. Она нередко оказывается большей по размеру, чем файл самого документа. Не всегда практично прицеплять маленький файл документа к большой библиотеке, большинство процедур которой, скорее всего, для данного документа попросту не нужны. Особенно рискованно изменять стандартную библиотеку Maple 7.

Впрочем, идти на это или нет — дело каждого пользователя. Разумеется, если вы готовы создать серьезную библиотеку своих процедур, то ее надо записать и тщательно хранить. С Maple 7 поставляется множество библиотек полезных процедур, составленных пользователями со всего мира, так что и вы можете пополнить ее своими творениями (см. урок 14).

Запись и считывание данных

Обширные возможности Maple 7 делают привлекательным применение этой программы для автоматической обработки данных, поступающих от каких-либо экспериментальных установок. Для этого установки снабжаются интерфейсными платами (например, аналого-цифровыми преобразователями) и необходимым программным обеспечением. Возможна и передача данных, полученных с помощью Maple 7, в экспериментальные установки.

Обмен информацией между Maple 7 и внешней средой (к ней, кстати, относятся и другие программы) чаще всего осуществляется через файлы текстового формата, поскольку именно с такими файлами могут работать практически все программы. Для записи данных в файл служит оператор `writedata`:

```
writedata[APPEND](fileID, data)
writedata[APPEND](fileID, data, format)
writedata[APPEND](fileID, data, format, default)
```

Здесь `fileID` — имя или дескриптор файла данных, `data` — список, вектор или матрица данных, `format` — спецификация формата данных (`integer`, `float` или `string`), `default` — процедура, задающая запись нечисловых данных, например:

```
writedata(F,A,float,proc(f,x) fprintf(f,`CMPLX(%g,%g)`,Re(x),Im(x)) end);
```

Необязательный указатель `APPEND` используется, если данные должны дописываться в уже созданный файл.

Считывание из файла `filename` обеспечивает функция `readdata`:

```
readdata(fileID, n)
readdata(fileID, format, n)
readdata(fileID, format)
```

Здесь `n` — целое положительное число, задающее число считываемых столбцов. Ниже представлены примеры этих операций:

```
> data:=array([[1,2,3],[4,5,6],[7,8,9]]);
```

$$data := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
> writedata(`C:\\mydata.txt`,data);
> restart;
> data;
```

data

```
> data:=readdata(`C:\\mydata.txt`,3);
```

data := [[1., 2., 3.], [4., 5., 6.], [7., 8., 9.]]

Maple 7 имеет также типичные файловые операции:

- `writeto` — запись в файл;
- `appendto` — добавление к файлу;
- `open` — открытие файла;
- `close` — закрытие файла;
- `write` — запись в открытый файл;
- `save` — запись выражений в файл;
- `read` — считывание из файла.

Их реализация, однако, зависит от платформы, на которой установлена система, и от ее настройки.

Вывод в специальных форматах

Вывод в формате LaTeX

Maple 7 имеет ряд средств для общения с другими программами. Часть из них, в основном относящаяся к обмену через файлы, уже была описана. Однако Maple 7 способна генерировать коды для прямого их включения в такие программы, причем не только математические.

Для подготовки математических статей и книг широкое распространение получили редакторы TeX и LaTeX. Для подготовки выражений или файлов в их формате служит функция:

```
latex(expr,filename)
```

Параметр `filename` не обязателен, если достаточно получить нужное выражение в ячейке вывода Maple 7:

```
> latex(a*x^2+b*x+c);
a{x}^{2}+bx+c
> latex(diff(x^n,x$2));
{\frac {{x}^{n}{n}^{2}}{{x}^{2}}}-{\frac {{x}^{n}n}{{x}^{2}}}
```

Генерация кодов на языке Фортран

Язык Фортран вот уже многие десятилетия используется для программирования вычислительных задач. Накоплены обширные библиотеки решения таких задач на Фортране. Почитателей этого языка Maple 7 порадует тем, что она позволяет готовить коды для программ на Фортране. Для этого вначале надо загрузить библиотечную функцию:

```
> with(codegen,fortran);
```

[*fortran*]

После этого может использоваться функция fortran:

```
fortran(expr,filename=str,optimized)
```

Два последних параметра не обязательны при выводе выражения expr в форме, присущей языку Фортран:

```
> fortran(a*x^2+b*x+c);
    t0 = a*x**2+b*x+c
> fortran(diff(x^n,x$2));
    t0 = x**n*n**2/x**2-x**n*n/x**2
```

Параметр optimize позволяет генерировать оптимизированные коды:

```
> fortran(a*x^2+b*x+c,optimized);
    t1 = x**2
    t4 = a*t1+b*x+c
```

При этом вычислительный процесс строится так, чтобы минимизировать число арифметических операций.

Генерация кодов на языке C

Язык C (Си) также широко используется для решения вычислительных задач. Достаточно отметить, что сама система Maple 7 создана на языке C.

Для генерации кодов на языке C вначале надо подключить соответствующую функцию:

```
> with(codegen,C);
```

[*C*]

Затем можно использовать функцию C:

```
C(expr,folename=str,optimized)
```

Эта функция используется по аналогии с функцией fortran, что и показывают приведенные ниже примеры:

```
> C(diff(x^b,x$2));
    t0 = pow(x,1.0*b)*b*b/(x*x)-pow(x,1.0*b)*b/(x*x);
> C(diff(x^b,x$2),optimized);
    t1 = pow(x,1.0*b);
    t2 = b*b;
```

```
t4 = x*x;
t5 = 1/t4;
t9 = t1*t2*t5-t1*b*t5;
```

Обширные возможности преобразования выражений в различные формы предоставляет функция convert. А функция interface позволяет управлять выводом. К сожалению, объем книги не позволяет рассмотреть все многочисленные варианты применения этих функций.

Дополнительные возможности Maple-языка

Переназначение определений

В большинстве случаев Maple-язык использует достаточно длинные идентификаторы для своих определений, например функций. Однако с помощью функции alias можно изменить любое определение на другое, если оно кажется пользователю более удобным. Функция alias записывается в виде:

```
alias(e1, e2, ..., eN)
```

где e1, e2, ..., eN — ноль или более равенств.

Эта функция возвращает список переназначений и осуществляет сами переназначения. Например, для замены имени функции BesselJ на более короткое имя BJ достаточно параметром функции alias записать BJ=BesselJ:

```
> alias(BJ=BesselJ);
```

$$BJ, Fx$$

```
> [BJ(0,1.),BesselJ(0,1.)];
```

$$[.7651976866, .7651976866]$$

Можно также переназначить функцию пользователя:

```
> alias(Fx=F(x));
```

$$BJ, Fx$$

```
> diff(F(x),x);
```

$$\frac{\partial}{\partial x} Fx$$

```
> int(F(x),x=a..b);
```

$$\int_a^b Fx\,dx$$

Для отмены переназначения, например BJ, используется та же функция alias с повтором переназначения:

```
> alias(BJ=BJ);
```

$$Fx$$

```
> BJ(0,1.);
```
$$BJ(0, 1.)$$

Обратите внимание на то, что BJ исчезло из списка переназначений и функция BJ(0,1.) уже не вычисляется, поскольку ее больше нет.

Модули

Модули придают языку программирования Maple 7 некоторые свойства языков объектно-ориентированного программирования. Они служат для реализации абстрактного типа данных на основе инкапсуляции — объединения данных и процедур их обработки. Модули задаются ключевым словом module с пустыми скобками () и завершаются словами end module или просто end:

```
name := module()
                        export eseq;  local lseq;   global gseq;
                        option optseq; description desc;
                        Тело модуля
end module (или просто end)
```

Хотя структура модуля во многом напоминает структуру процедуры, включая объявление локальных и глобальных переменных, параметров и описаний, между ними есть существенная разница:

- модуль не имеет списка входных параметров;
- в модуле могут размещаться данные;
- модули могут использоваться для создания пакетов процедур, доступ к которым обеспечивается командой with;
- модули имеют свойства в виде локальных переменных и методы в виде процедур интерфейса модулей;
- реализация абстактных типов данных с помощью модулей скрыта от пользователя;
- модули могут содержать оператор export eseq, объявляющий экспортируемые переменные модуля;
- для доступа к экспортируемым переменным модуля может использоваться специальный оператор «:-» (двоеточие и минус);
- модули и процедуры могут вкладываться друг в друга без ограничения уровня вложенности;
- модули могут иметь специальные конструкторы объектов.

Следующий пример демонстрирует создание модуля pt, в котором заданы две операции (сложения plus и умножения times) и показан доступ к ним:

```
> pt:= module()
                              export plus, times;
                              plus := (a,b) -> a + b;
                              times := (a,b) -> a * b;
end module;
```

```
pt := module () export plus, times; end module
   > pt:-plus(3,5);
 8
   > pt:-times(3,7);
 21
```

Детальную информацию о модулях и о конструкторах объектов можно найти в справках по ним. Некоторые пакеты Maple 7 (в основном сравнительно новые) реализованы уже не в виде процедур, а в виде модулей (например, в виде модуля сделан пакет LinearAlgebra). В простейшем виде модули могут использоваться всеми пользователями системы Maple 7, но их серьезное применение (например, с целью создания полноценных пакетов Maple 7) требует серьезного знакомства с техникой объектно-ориентированного программирования. Такое знакомство выходит за рамки данной книги.

Макросы

Макрос — это макрокоманда, короткая запись длинных определений. По сравнению с переназначениями макросы более гибки и могут использоваться для сокращения операций загрузки новых определений из библиотеки и пакетов. Макросы создаются с помощью функции `macro`:

```
macro(e1, e2, ..., en)
```

где `e1`, `e2`, ..., `en` — ноль или более равенств.

В следующем примере функция `numbperm` с помощью макроса заменена на `np`:

```
> numbperm([1,2,3,4]);
 24
> macro(np=numbperm(V));
 np
> V:=[1,2,3,4];
 V := [1, 2, 3, 4]
> np(V);
 24
```

Макросы могут быть использованы для конструирования выражений из их макроопределений.

Внешние вызовы

Maple 7 имеет команду `system(string)`, с помощью которой можно исполнить любую команду MS-DOS, записанную в виде строки `string`. Например, для форматирования гибкого диска из среды Maple 7 можно использовать стандартную команду MS-DOS:

```
> system(`format a:`);
```

На экране появится окно MS-DOS с начальным диалогом форматирования диска A. Это окно показано на рис. 7.5.

При работе в операционной системе Windows эта возможность практически бесполезна, поскольку форматирование диска с большими удобствами можно выполнить средствами Windows.

Внешние вызовы командой `system` куда более полезны для MS-DOS-реализаций Maple, которые кое-где используются и по сей день. Но поскольку данная книга посвящена самым современным Windows-реализациям системы Maple 7, более подробное рассмотрение операций внешних вызовов не имеет особого смысла.

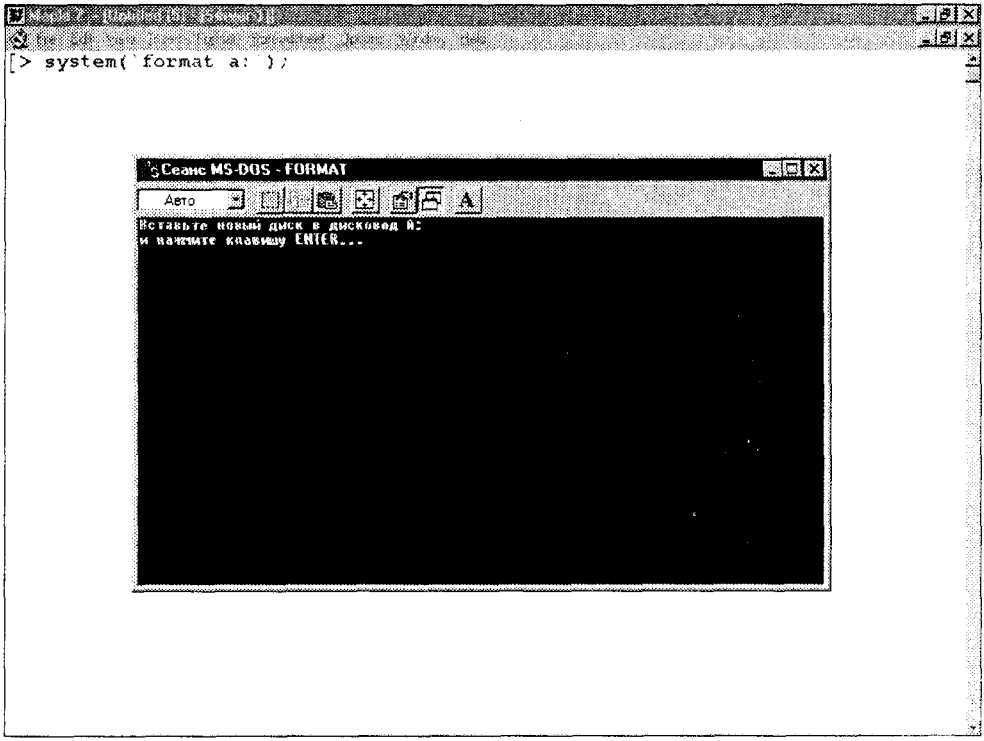


Рис. 7.5. Результат выполнения команды форматирования гибкого диска

Вызов внешних процедур, написанных на языке C

Maple 7 имеет средства для вызова внешних откомпилированных процедур, написанных на языке C. Такая необходимость для подавляющего числа пользователей Maple 7 вызывает большие сомнения в силу следующих причин:

- вся идеология системы Maple 7 основана на максимальном исключении программирования на других языках, помимо Maple-языка;
- язык C сложен для большинства пользователей Maple 7, которых трудно отнести к «путным» программистам;

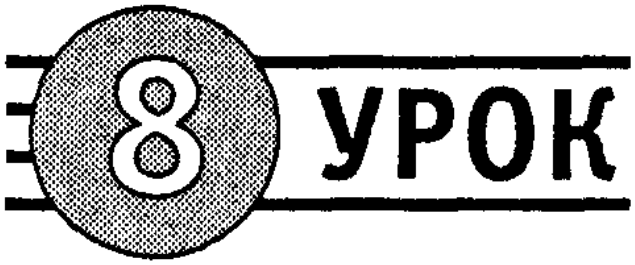
- отладка комплекса Maple 7 + компилятор C (например, фирмы Microsoft) вряд ли под силу обычным пользователям, тем более что на практике такой комплекс реально не работает без кропотливой отладки.

Учитывая сказанное, мы отметим лишь, что для использования внешних процедур (например, остро нужных пользователю или более быстрых, чем аналогичные процедуры Maple) используется специальная команды `define_external`, которая генерирует две интерфейсные программы — на языке C и на языке Maple соответственно. Программа на языке C компилируется вместе с внешней процедурой, которая будет использоваться. Результирующий код создает динамически подключаемую бублиотеку DLL. В свою очередь, программа на языке Maple служит как интерфейсная для организации взаимодействия с вызываемой внешней процедурой. Более подробное описание возможностей работы с внешними процедурами можно найти в справке по ним.

Что нового мы узнали?

В этом уроке мы научились:

- Задавать функции пользователя.
- Использовать условные операторы.
- Применять циклы `for` и `while`.
- Использовать операторы пропуска и прерывания.
- Задавать процедуры и процедуры-функции.
- Использовать средства контроля и отладки процедур.
- Работать с отладчиком программ.
- Применять операции ввода и вывода.
- Осуществлять вывод в специальных форматах.
- Использовать дополнительные возможности Maple-языка.



Математический анализ

- Вычисление сумм последовательностей
- Вычисление произведений членов последовательностей
- Вычисление производных
- Вычисление интегралов
- Разложение функций в ряд
- Решение уравнений и неравенств

Вычисление сумм последовательностей

Основные формулы для вычисления сумм последовательностей

Применение систем символьной математики особенно эффективно при решении задач математического анализа. Maple 7 обладает богатейшей базой данных по формулам математического анализа и может полноценно заменять тома книг со справочными данными. При этом важно, что Maple не только «знает» многие формулы, но и может успешно использовать их при решении достаточно сложных задач в аналитическом (символьном) виде.

Начнем рассмотрение таких операций с вычисления сумм последовательностей. Вычисление суммы членов некоторой последовательности $f(k)$ при изменении целочисленного индекса k от значения m до значения n с шагом +1, то есть выражения:

$$\sum_{k=m}^{n} f(k) = f(m) + f(m+1) + \ldots + f(n-1) + f(n) .$$

является достаточно распространенной операцией математического анализа. Для вычисляемой и инертной форм сумм последовательностей служат следующие функции:

```
sum(f,k);          sum(f,k=m..n);          sum(f,k=alpha);
Sum(f,k);          Sum(f,k=m..n);          Sum(f,k=alpha);
```

Здесь `f` — функция, задающая члены суммируемого ряда, `k` — индекс суммирования, `m` и `n` — целочисленные пределы изменения `k`, `alpha` — `RootOf`-выражение. Значение `n` может быть равно бесконечности. В этом случае для `n` используется обозначение `?` или `infinity`.

Допустимо (а зачастую рекомендуется с целью исключения преждевременной оценки суммы) заключение `f` и `k` в прямые кавычки, например `sum('f', 'k'=m..n)`. Это сделано во всех примерах справочной системы Maple 7, относящихся к функции `sum`. Мы, однако, отказываемся от этого в тех случаях, когда результат идентичен при заключении `f` и `k` в кавычки и без такового. Во избежание путаницы, связанной с этой тонкостью синтаксиса функции `sum`, рекомендуется все примеры проверять после команды `restart`, убирающей предыдущие определения `f` и `k`.

Последовательности с заданным числом членов

Простейшими являются суммы последовательностей с фиксированным числом членов. Ниже даны примеры применения этих функций:

```
> restart;k:=2;
```

$$k := 2$$

```
> Sum(k^2,k=1..4);
```

$$\sum_{2=1}^{4} 4$$

```
> sum(k^2,k=1..4);
    Error, (in sum) summation variable previously assigned, second argument evaluates to 2 = 1 .. 4

> sum('k^2','k'=1..4);
```

$$30$$

```
> sum(1/i,i=1..100);
```

$$\frac{14466636279520351160221518043104131447711}{2788815009188499086581352357412492142272}$$

Обратите внимание, что во втором примере система отказалась от вычисления, а в третьем даже выдала сообщение об ошибке, связанной с тем, что переменной k перед вычислением сумм было присвоено численное значение 2. После заключения выражения и переменной индекса k в прямые кавычки ошибка исчезла, поскольку такая операция означает, что переменной придается неопределенное значение.

Суммы с заданным пределом

Особый класс образуют последовательности, у которых предел задается в общем виде значением переменной. Ниже представлен ряд последовательностей, у которых предел задается как 0..n или 1..n:

```
> restart;
> sum(k,k=1..n);
```

$$\frac{1}{2}(n+1)^2 - \frac{1}{2}n - \frac{1}{2}$$

```
> sum(i/(i+1),i=0..n);
```

$$n + 1 - \Psi(n+2) - \gamma$$

```
> sum(k*binomial(n,k),k=0..n);
```

$$\frac{1}{2}2^n n$$

Такого рода последовательности, как видно из приведенных примеров, нередко имеют аналитические выражения для своего значения. Его вычисление намного проще, чем формирование заданной последовательности с прямым суммированием ее членов. Некоторые из таких сумм выражаются через специальные математические функции.

Суммы бесконечных последовательностей

Многие суммы бесконечных последовательностей сходятся к определенным численным или символьным значениям, и система Maple 7 способна их вычислять. Это поясняют следующие примеры:

```
> restart;
> sum(-exp(-k),k);
```

$$\frac{e}{(-1+e)e^{k}}$$

```
> sum(k*a^k,k);
```

$$\frac{a^{k}(ka-k-a)}{(a-1)^{2}}$$

```
> sum(1/k!,k=0..infinity);
```

$$e$$

```
> Sum(1/i^2,i=1..infinity)=sum(1/i^2,i=1..infinity);
```

$$\sum_{i=1}^{\infty}\frac{1}{i^{2}}=\frac{1}{6}\pi^{2}$$

```
> Sum(1/n!,n=1..infinity)=sum(1/n!,n=1..infinity);
```

$$\sum_{n=1}^{\infty}\frac{1}{n!}=e\left(1-e^{(-1)}\right)$$

```
> evalf(%);
```

$$1.718281828 = 1.718281828$$

```
> Sum(1/i^2,i)=sum(1/i^2,i);
```

$$\sum_{i}\frac{1}{i^{2}}=-\Psi(1,i)$$

```
> Sum(1/i^2,i=1..infinity)=sum(1/i^2,i=1..infinity);
```

$$\sum_{i=1}^{\infty}\frac{1}{i^{2}}=\frac{1}{6}\pi^{2}$$

```
> Sum(1/n!,n=1..infinity)=sum(1/n!,n=1..infinity);
```

$$\sum_{n=1}^{\infty}\frac{1}{n!}=e\left(1-e^{(-1)}\right)$$

```
> evalf(%);
```

$$1.718281828 = 1.718281828$$

```
> Sum(1/i^2,i)=sum(1/i^2,i);
```

$$\sum_{i}\frac{1}{i^{2}}=-\Psi(1,i)$$

Сумма от перемены мест слагаемых меняется!

Даже школьники хорошо знают, что от перестановки слагаемых сумма не изменяется. Однако Maple 7 (кстати, как и большинство других систем компьютерной математики) при вычислении сумм, увы, этому правилу не следует. Приведенные ниже примеры наглядно показывают этот просчет системы:

```
> restart;
> Sum(k,k=1..5)=sum(k,k=1..5);
```

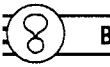
$$\sum_{k=1}^{5} k = 15$$

```
> Sum(k,k=5..1)=sum(k,k=5..1);
```

$$\sum_{k=5}^{1} k = -9$$

```
> 5+4+3+2+1;
```

15



ВНИМАНИЕ При вычислении сумм последовательностей надо строго соблюдать прямой (нарастающий) порядок задания значений индексной переменной суммы. Нарушение этого порядка чревато грубыми ошибками.

Двойные суммы

Могут встречаться множественные суммы по типу «сумма в сумме». Ограничимся приведением примера двойной суммы, имеющей аналитическое значение:

```
> Sum( Sum(k^2, k = 1..m), m = 1..N);   factor( simplify( value(%)));
```

$$\sum_{m=1}^{N}\left(\sum_{k=1}^{m} k^2\right),$$

$$\frac{1}{12} N (N+2) (N+1)^2,$$

При конкретном значении N такую сумму нетрудно вычислить подстановкой:

```
> subs( N = 100, %);
```

8670850

Как видно из приведенных примеров, средства вычисления сумм последовательностей Maple 7 позволяют получать как численные, так и аналитические значения сумм, в том числе представляемые специальными математическими функциями.

Вычисление произведений членов последовательностей

Основные формулы для произведения членов последовательностей

Аналогичным образом для произведений членов $f(i)$ некоторой последовательности, например вида:

$$\prod_{i=m}^{n} f(i) = f(m) f(m+1) \cdots f(n-1) f(n)$$

используются следующие функции:

```
product(f,k);          product(f,k=m..n);          product(f,k=alpha);
Product(f,k);          Product(f,k=m..n);          Product(f,k=alpha).
```

Обозначения параметров этих функций и их назначение соответствуют приведенным для функций вычисления сумм. Это относится, в частности, и к применению одиночных кавычек для f и k.

Примеры вычисления произведений членов последовательностей

Примеры применения функций вычисления произведений даны ниже:

```
> restart;
> Product(k^2,k=1..5)=product(k^2,k=1..5);
```

$$\prod_{k=1}^{5} k^2 = 14400$$

```
> Product(k^2,k)=product(k^2,k);
```

$$\prod_{k} k^2 = \Gamma(k)^2$$

```
> product(a[k],k=1..5);
```

$$a_1 a_2 a_3 a_4 a_5$$

```
> f:=[1,2,3,4,5];
```

$$f := [1, 2, 3, 4, 5]$$

```
> product(f[k],k=1..4);
```

$$24$$

```
> product(n+k,k=1..4);
```

$$(n+1)(n+2)(n+3)(n+4)$$

```
> Product(n+k,k=1..m)=product(n+k,k=1..m);
```

$$\prod_{k=1}^{m} (n+k) = \frac{\Gamma(n+m+1)}{\Gamma(n+1)}$$

```
> product(k,k=RootOf(x^3-9));
```

$$9$$

Как и в случае вычисления сумм, вычисление произведений возможно как в численной, так и в аналитической форме — разумеется, если таковая существует. Это показывает следующий пример:

```
> Product(2/i,i=1..infinity)=product(2/i,i=1..infinity);
```

$$\prod_{i=1}^{\infty} \left(2\frac{1}{i}\right) = 0$$

Нетрудно понять, что при i, стремящемся к бесконечности, перемножаемые члены последовательности стремятся к нулю, а потому к нулю стремится и их произведение. Вопросы доказательства подобных утверждений находятся за рамками данного учебного курса, ибо он посвящен не математике как таковой, а конкретной программе для математики — Maple 7.

От перемены места сомножителей произведение меняется!

Хотя произведение не зависит от порядка расположения сомножителей, их перестановка в Maple 7 недопустима. Это иллюстрируют следующие примеры:

```
> Product(k^2,k=1..4)=product(k^2,k=1..4);
```

$$\prod_{k=1}^{4} k^2 = 576$$

```
> Product(k^2,k=4..1)=product(k^2,k=4..1);
```

$$\prod_{k=4}^{1} k^2 = \frac{1}{36}$$

```
> 4^2*3^2*2^2*1^2;
```

576

ВНИМАНИЕ При вычислении произведений надо строго соблюдать прямой (нарастающий) порядок задания значений индексной переменной произведения. Нарушение этого порядка чревато грубыми ошибками.

Вычисление производных

Функции дифференцирования выражений diff и Diff

Вычисление производных функций $f^n(x) = \mathrm{d}f^n(x)/\mathrm{d}x^n$ n-го порядка — одна из самых распространенных задач математического анализа. Для ее реализации Maple 7 имеет следующие основные функции:

```
diff(a, x1, x2, ..., xn)        diff(a, [x1, x2, ..., xn])
Diff(a, x1, x2, ..., xn)        Diff(a, [x1, x2, ..., xn])
```

Здесь `a` — дифференцируемое алгебраическое выражение, в частности функция `f(x1, x2, ..., xn)` ряда переменных, по которым производится дифференцирование. Функция `Diff` является инертной формой вычисляемой функции `diff` и может использоваться для естественного воспроизведения производных в документах.

Первая из этих функций (в вычисляемой и в инертной форме) вычисляет частные производные для выражения `a` по переменным `x1, x2, ..., xn`. В простейшем случае `diff(f(x),x)` вычисляет первую производную функции $f(x)$ по переменной x. При n, большем 1, вычисления производных выполняются рекурсивно, например `diff(f(x), x, y)` эквивалентно `diff(diff (f(x), x), y)`. Оператор `$` можно использовать для вычисления производных высокого порядка. Для этого после имени соответствующей переменной ставится этот оператор и указывается порядок производной. Например, выражение `diff(f(x),x$4)` вычисляет производную 4-го порядка и эквивалентно записи `diff(f(x),x,x,x,x)`. А `diff(g(x,y),x$2,y$3)` эквивалентно `diff(g(x,y),x,x,y,y,y)`.

Примеры вычисления производных:

```
> restart;
> Diff(a*x^n,x)=diff(a*x^n,x);
```

$$\frac{\partial}{\partial x} a\,x^n = \frac{a\,x^n\,n}{x}$$

```
> Diff(a*sin(b*x),x)=diff(a*sin(b*x),x);
```

$$\frac{\partial}{\partial x} a\,\sin(b\,x) = a\,\cos(b\,x)\,b$$

```
> Diff([sin(x),x^n,exp(a*x)],x)=diff([sin(x),x^n,exp(a*x)],x);
```

$$\frac{\partial}{\partial x}[\sin(x), x^n, \mathrm{e}^{(a\,x)}] = \left[\cos(x), \frac{x^n\,n}{x}, a\,\mathrm{e}^{(a\,x)}\right]$$

```
> Diff(a*x^n,x$3)=diff(a*x^n,x$3);
```

$$\frac{\partial^3}{\partial x^3} a\,x^n = \frac{a\,x^n\,n^3}{x^3} - \frac{3\,a\,x^n\,n^2}{x^3} + \frac{2\,a\,x^n\,n}{x^3}$$

```
> Diff([x^2,x^3,x^n],x)=diff([x^2,x^3,x^n],x);
```

$$\frac{\partial}{\partial x}[x^2, x^3, x^n] = \left[2\,x, 3\,x^2, \frac{x^n\,n}{x}\right]$$

```
> simplify(%);
```

$$\frac{\partial}{\partial x}[x^2, x^3, x^n] = [2\,x, 3\,x^2, x^{(n-1)}\,n]$$

Как видно из приведенных примеров, функции вычисления производных могут использоваться с параметрами, заданными списками. Приведенные ниже примеры показывают эти возможности и иллюстрируют дифференцирование функции пользователя для двух переменных:

```
> restart;
> f(x,y):=cos(x)*y^3;
```

$$f(x,y) := \cos(x)\,y^3$$

```
> Diff(f(x,y),x)=diff(f(x,y),x);
```

$$\frac{\partial}{\partial x}\cos(x)\,y^3 = -\sin(x)\,y^3$$

```
> Diff(f(x,y),y)=diff(f(x,y),y);
```

$$\frac{\partial}{\partial y}\cos(x)\,y^3 = 3\,\cos(x)\,y^2$$

```
> Diff(f(x,y),x,y)=diff(f(x,y),x,y);
```

$$\frac{\partial^2}{\partial y\,\partial x}\cos(x)\,y^3 = -3\,\sin(x)\,y^2$$

```
> Diff(f(x,y),x$4)=diff(f(x,y),x$4);
```

$$\frac{\partial^4}{\partial x^4}\cos(x)\,y^3 = \cos(x)\,y^3$$

```
> Diff(f(x,y),y$2)=diff(f(x,y),y$2);
```

$$\frac{\partial^2}{\partial y^2}\cos(x)\,y^3 = 6\,\cos(x)\,y$$

```
> Diff(f(x,y),x$4,y$4)=diff(f(x,y),x$3,y$2);
```

$$\frac{\partial^8}{\partial y^4\,\partial x^4}\cos(x)\,y^3 = 6\,\sin(x)\,y$$

Получаемые в результате дифференцирования выражения могут входить в другие выражения. Можно задавать их как функции пользователя и строить графики производных.

Дифференциальный оператор D

Для создания функций с производными может также использоваться дифференциальный оператор D. Порою он позволяет создавать более компактные выражения, чем функции diff и Diff. Дифференциальный оператор можно записывать в следующих формах: D(f) или D[i](f), где параметр f — выражение или имя функции, i — положительное целое число, выражение или последовательность. Оператор D(f) просто вычисляет имя производной от f, поскольку в этой форме он эквивалентен unnaply(diff(f(x),x),x). В форме D(f)(x) этот оператор подобен diff(f(x),x).

Приведем примеры дифференцирования функций, заданных только именами, и функций с одним параметром:

```
> restart;
> D(cos^2);
```

$-2\,\sin\,\cos$

```
> D(exp^2+cos^2+tan+GAMMA);
```

$2\,\exp^2 - 2\,\sin\,\cos + 1 + \tan^2 + \Psi\,\Gamma$

```
> D(ln);
```

$$a \to \frac{1}{a}$$

```
> D(sin)(x)=diff(sin(x),x);
```

$\cos(x) = \cos(x)$

Следующий пример показывает дифференцирование функции пользователя fun с применением дифференциального оператора D и функции diff:

```
> fun:=(x)->sin(x^2);
```

$fun := x \to \sin(x^2)$

```
> D(fun)=diff(fun(x),x);
```

$(x \to 2\,\cos(x^2)\,x) = 2\,\cos(x^2)\,x$

Дифференциальный оператор можно применять и для дифференцирования функций нескольких переменных по заданной переменной:

```
> f:=(x,y,z)->x*exp(y)+ln(z);
```

$f := (x, y, z) \to x\,e^y + \ln(z)$

```
> D[1](f);
```

$(x, y, z) \to e^y$

```
> D[2](f);
```

$(x, y, z) \to x\,e^y$

```
> D[3](f);
```

$$(x, y, z) \to \frac{1}{z}$$

Пример применения дифференциального оператора для функции f, заданной программным объектом-процедурой, представлен ниже:

```
> restart;
> f:=proc(x,b,n) local i,d,s;
> s:=0;
> for i from n by -1 to 0 do s:=s*x+b[i] od;
> s
> end:
> D[1](f);
```

proc (x, b, n)
local i, s, sx;
$sx := 0$;
$s := 0$;
for i from n by -1 to 0 do $sx := sx \times x + s$; $s := s \times x + b[i]$ end do ;
sx
end proc

```
> D[1](sin*cos);
```

$\cos^2 - \sin^2$

Этот пример показывает реализацию схемы Горнера для полинома b степени n от переменной x. При этом применение оператора дифференцирования возвращает процедуру. Ряд интересных возможностей по вычислению производных предоставляет пакет расширения student.

Вычисление интегралов

Вычисление неопределенных интегралов

Вычисление неопределенного интеграла обычно заключается в нахождении первообразной функции. Это одна из широко распространенных операций математического анализа.

Для вычисления неопределенных и определенных интегралов Maple V предоставляет следующие функции:

```
int(f,x);          int(f,x=a..b);          int(f,x=a..b,continuous);
Int(f,x);          Int(f,x=a..b);          Int(f,x=a..b,continuous);
```

Здесь f — подынтегральная функция, x — переменная, по которой выполняются вычисления, a и b — нижний и верхний пределы интегрирования, continuous — необязательное дополнительное условие.

Maple 7 старается найти аналитическое значение интеграла с заданной подынтегральной функцией. Если это не удается (например, для «не берущихся» ин-

тегралов), то возвращается исходная запись интеграла. Для вычисления определенного интеграла надо использовать функцию `evalf(int(f,x=a..b))`. Ниже приведены примеры вычисления интегралов:

```
> Int(a*x^n,x)=int(a*x^n,x);
```

$$\int a\,x^n\,dx = \frac{a\,x^{(n+1)}}{n+1}$$

```
> Int(sin(x)/x,x)=int(sin(x)/x,x);
```

$$\int \frac{\sin(x)}{x}\,dx = \mathrm{Si}(x)$$

```
> Int(ln(x)^3,x);
```

$$\int \ln(x)^3\,dx$$

```
> value(%);
```

$$\ln(x)^3\,x - 3\,x\,\ln(x)^2 + 6\,x\,\ln(x) - 6\,x$$

```
> Int(x^5*exp(-x),x);
```

$$\int x^5\,\mathrm{e}^{(-x)}\,dx$$

```
> value(%);
```

$$-x^5\,\mathrm{e}^{(-x)} - 5\,x^4\,\mathrm{e}^{(-x)} - 20\,x^3\,\mathrm{e}^{(-x)} - 60\,x^2\,\mathrm{e}^{(-x)} - 120\,x\,\mathrm{e}^{(-x)} - 120\,\mathrm{e}^{(-x)}$$

```
> Int(1/x,x)=int(1/x,x);
```

$$\int \frac{1}{x}\,dx = \ln(x)$$

Обратите внимание, что в аналитическом представлении неопределенных интегралов отсутствует произвольная постоянная C. Не следует забывать о ее существовании. Для вычисления кратных интегралов (двойных, тройных и т. д.) следует применять функцию int (или Int) внутри такой же функции, делая это столько раз, сколько нужно. В отличие от функции дифференцирования для функции интегрирования нельзя задавать подынтегральные функции в виде списка или множества.

Возможно вычисление сумм интегралов и интегралов сумм, а также интегралов от полиномов:

```
> Sum(Int(x^i,x),i=1..5);
```

$$\sum_{i=1}^{5} \int x^i\,dx$$

```
> value(%);
```

$$\frac{1}{2}x^2 + \frac{1}{3}x^3 + \frac{1}{4}x^4 + \frac{1}{5}x^5 + \frac{1}{6}x^6$$

```
> Int(Sum(x^i,i=1..5),x);
```

$$\int \sum_{i=1}^{5} x^i\,dx$$

```
> value(%);
```

$$\frac{1}{2}x^2+\frac{1}{3}x^3+\frac{1}{4}x^4+\frac{1}{5}x^5+\frac{1}{6}x^6$$

```
> P(x):=a*x^3+b*x^2+c*x+d;
```

$$P(x) := a\,x^3 + b\,x^2 + c\,x + d$$

```
> int(P(x),x);
```

$$\frac{1}{4}a\,x^4+\frac{1}{3}b\,x^3+\frac{1}{2}c\,x^2+d\,x$$

ПРИМЕЧАНИЕ Maple 7 успешно берет большинство справочных интегралов. Но не всегда форма представления интеграла совпадает с приведенной в справочнике. Иногда требуется доводка ее до нужной формы, а иногда Maple 7 упорно дает иное выражение (в большинстве случаев правильное). Тем не менее следует помнить, что всегда может найтись интеграл, который окажется «не по зубам» и Maple 7.

Конвертирование и преобразование интегралов

В некоторых случаях Maple 7 не может вычислить интеграл. Тогда она просто повторяет его. С помощью функций `taylor` и `convert` можно попытаться получить аналитическое решение в виде полинома умеренной степени, что демонстрирует следующий характерный пример:

```
> int(exp(sin(x)),x);
```

$$\int e^{\sin(x)}\,dx$$

```
> convert(taylor(%,x=0,8),polynom);
```

$$x+\frac{1}{2}x^2+\frac{1}{6}x^3-\frac{1}{40}x^5-\frac{1}{90}x^6-\frac{1}{1680}x^7+\frac{1}{720}x^8$$

Естественно, что в этом случае решение является приближенным, но оно все же есть и с ним можно работать, например построить график функции, представляющей данный интеграл.

Система Maple непрерывно совершенствуется. Например, в Maple V R4 интеграл с подынтегральной функцией `exp(x^4)` не брался, а система Maple 7 с легкостью берет его:

```
> Int(exp(x^4),x)=int(exp(x^4),x);
```

$$\int e^{(x^4)}\,dx = -\frac{1}{4}(-1)^{(3/4)}\left(\frac{x\,(-1)^{(1/4)}\,\pi\sqrt{2}}{\Gamma\left(\frac{3}{4}\right)(-x^4)^{(1/4)}}-\frac{x\,(-1)^{(1/4)}\,\Gamma\left(\frac{1}{4},-x^4\right)}{(-x^4)^{(1/4)}}\right)$$

Хотя полученный результат, выраженный через гамма-функцию, нельзя назвать очень простым, но он существует и с ним также можно работать. Например, можно попытаться несколько упростить его, используя функцию `simplify`:

```
> simplify(%);
```

$$\int e^{(x^4)}\,dx = \frac{1}{4}\,\frac{x\left(-\Gamma\left(\frac{1}{4},-x^4\right)\Gamma\left(\frac{3}{4}\right)+\pi\sqrt{2}\right)}{(-x^4)^{(1/4)}\,\Gamma\left(\frac{3}{4}\right)}$$

Разумеется, существует также множество иных возможностей и приемов для выполнения операции интегрирования. В дальнейшем мы неоднократно будем рассматривать и другие, более специфические функции для осуществления интегрирования и вычисления интегральных преобразований. В частности, ряд средств вычисления интегралов реализован в пакете student.

Вычисление определенных интегралов

Другой важной операцией является нахождение в аналитической или численной форме определенного интеграла:

$$\int_a^b f(x)dx\,.$$

Определенный интеграл удобно трактовать как площадь, ограниченную кривой $f(x)$, осью абсцисс и вертикалями с координатами, равными a и b. При этом площадь ниже оси абсцисс считается отрицательной. Таким образом, значение определенного интеграла — это число или вычисляемое выражение.

Для вычисления определенных интегралов используются те же функции int и Int, в которых надо указать пределы интегрирования, например x=a..b, если интегрируется функция переменной x. Это поясняется приведенными ниже примерами:

```
> Int(sin(x)/x,x=a..b)=int(sin(x)/x,x=a..b);
```

$$\int_a^b \frac{\sin(x)}{x}\,dx = \mathrm{Si}(b) - \mathrm{Si}(a)$$

```
> Int(sin(x)/x,x=0..1.)=int(sin(x)/x,x=0..1.);
```

$$\int_0^{1.} \frac{\sin(x)}{x}\,dx = .9460830704$$

```
> Int(x*ln(x),x=0..1)=int(x*ln(x),x=0..1);
```

$$\int_0^1 x\ln(x)\,dx = \frac{-1}{4}$$

```
> Int(x*exp(-x),x=0..infinity)=int(x*exp(-x),x=0..infinity);
```

$$\int_0^\infty x\,e^{(-x)}\,dx = 1$$

```
> Int(1/(x^2+6*x+12),x=-infinity..infinity);
```

$$\int_{-\infty}^{\infty} \frac{1}{x^2+6x+12}\,dx$$

```
> value(%);
```

$$\frac{1}{3}\pi\sqrt{3}$$

Как видно из этих примеров, среди значений пределов может быть бесконечность, обозначаемая как `infinity`.

Каверзные интегралы и визуализация результатов интегрирования

Выше мы уже сталкивались с примерами вычисления «каверзных» интегралов. Немного продолжим эту важную тему и заодно рассмотрим приемы визуализации вычислений, облегчающие понимание их сущности.

В Соросовском образовательном журнале (№ 6, 2000, с. 110) приводятся не совсем удачные примеры вычислений определенного интеграла с применением системы Mathematica, при которых якобы встречаются настолько большие трудности, что они не под силу любому калькулятору или компьютеру. При некоторых попытках вычисления этого интеграла он давал нулевое значение.

Но Maple 7 (кстати, как и Mathematica 4) с легкостью берет этот интеграл и позволяет сразу и без какой-либо настройки вычислить для него как точное, так и приближенное значение:

```
> Int(x^20*exp(-x),x=0..1)=int(x^20*exp(-x),x=0..1);
```

$$\int_0^1 x^{20} e^{(-x)}\,dx = -6613313319248080001\, e^{(-1)} + 2432902008176640000$$

```
> evalf(%,30);
```

$$.018350467697256206326144754231 7 = .01835046770$$

Хотя первое из решений является самым кратким и, скорее всего, единственным точным решением, оно может и должно насторожить опытного пользователя. Дело в том, что в полученном выражении фигурируют большие числа, и потому для правильного приближенного решения (в виде вещественного числа в научной нотации) нужно заведомо использовать аппарат точной арифметики и ни в коем случае не полагаться на погрешность, заданную по умолчанию, — вот в чем основная ошибка в упомянутой статье.

Именно поэтому левая и правая части приближенного решения (выполненного с точностью до 30 цифр) заметно различаются. Знак равенства между ними вызывает чувство протеста у истинных математиков. На самом деле, не надо забывать, что знак равенства здесь был введен просто как текстовый комментарий, — вы можете попробовать сами заменить его на более приемлемый здесь

знак приближенного равенства. Любопытно, что предшествующая версия Maple 6 при задании погрешности по умолчанию вычисляла значение этого интеграла также как 0, тогда как Maple 7 «поумнела» уже настолько, что дает значение 0.01835046770 даже в этом случае.

При таких условиях многие читатели могут сомневаться в корректности конечного результата. Между тем Maple 7 позволяет наглядно проиллюстрировать характер промежуточных вычислений подобных интегралов. Например, для этого можно вычислить неопределенный интеграл подобного вида:

```
> int(x^20*exp(-x),x);
```

$$-x^{20}e^{(-x)} - 20x^{19}e^{(-x)} - 380x^{18}e^{(-x)} - 6840x^{17}e^{(-x)} - 116280x^{16}e^{(-x)}$$
$$- 1860480x^{15}e^{(-x)} - 27907200x^{14}e^{(-x)} - 390700800x^{13}e^{(-x)}$$
$$- 5079110400x^{12}e^{(-x)} - 60949324800x^{11}e^{(-x)} - 670442572800x^{10}e^{(-x)}$$
$$- 6704425728000x^{9}e^{(-x)} - 60339831552000x^{8}e^{(-x)} - 482718652416000x^{7}e^{(-x)}$$
$$- 3379030566912000x^{6}e^{(-x)} - 20274183401472000x^{5}e^{(-x)}$$
$$- 101370917007360000x^{4}e^{(-x)} - 405483668029440000x^{3}e^{(-x)}$$
$$- 1216451004088320000x^{2}e^{(-x)} - 2432902008176640000xe^{(-x)}$$
$$- 2432902008176640000e^{(-x)}$$

Нетрудно заметить, что решение распадается на множество слагаемых, соответствующих общеизвестному интегрированию по частям. В каждом слагаемом имеются большие числа, и потому принципиально необходимо применение арифметики высокой точности (или разрядности). Maple 7 такими средствами, причем превосходными, обладает.

Продолжим изучение данного «каверзного» интеграла. Опробуем силы Maple 7 на интеграле более общего вида, где конкретный показатель степени заменен на обобщенный — n. Здесь нас ожидает приятный сюрприз — Maple 7 с легкостью выдает аналитическое решение для данного определенного интеграла:

```
> y:=(n)->int(x^n*exp(-x),x=0..1);
```

$$y := n \to \int_0^1 x^n e^{(-x)}\,dx$$

```
> y(n);
```

$$\frac{e^{(-1/2)}\,\text{WhittakerM}\left(\frac{1}{2}n, \frac{1}{2}n+\frac{1}{2}, 1\right)}{n+1}$$

```
> y(20);
```

$$-6613133319248080001e^{(-1)} + 2432902008176640000$$

```
> evalf(%,30);
```

.01835046770

```
> y(20.);
```

0.

Однако радоваться несколько преждевременно. Многие ли математики знают, что это за специальная функция — WhittakerM? Студенты, любящие подшучивать над своим профессором, могут попробовать спросить у него об этом. Скорее всего, профессор стушуется, а потом будет долго копаться в литературе, прежде чем найдет ее определение и сможет разъяснить, что это такое.

Но хуже другое — Maple 7 при конкретном $n = 20$ дает грубо неверное решение — 0 (почему — уже объяснялось). Забавно, что при этом сама по себе функция WhittakerM вычисляется для $n = 20$ без проблем:

```
> WhittakerM(10,10.5,1);
```

.6353509348

А теперь присмотритесь к новому результату вычисления злополучного интеграла. Оказывается, он уже не содержит больших чисел, свойственных прямому решению! Зная значение WhittakerM с погрешностью по умолчанию, можно уверенно вычислить приближенное численное значение интеграла с той же погрешностью, уже не прибегая к арифметике высокой точности:

```
> (exp(-.5)*WhittakerM(10,10.5,1))/21;
```

.01835046770

Итак, мы вычислили нужный интеграл несколькими разными способами. В этом и проявляется могущество современной математики, достойно представленной такими системами, как Maple 7. Заинтересованный читатель может попытаться найти еще ряд методов решения данного интеграла и преуспеть в этом! Мы же как торжество Maple 7 приведем график зависимости значений данного интеграла от показателя степени n при его изменении от 0 до 50 (рис. 8.1). Надо ли говорить о том, что полученный результат имеет куда более важное значение, чем вычисление нашего злополучного интеграла при конкретном $n = 20$? А плавный ход графика показывает, что в вычислении данного интеграла нет никаких признаков неустойчивости решения при изменении n, если соблюдать правило выбора погрешности вычислений.

Наличие у функции особых (сингулярных) точек нередко затрудняет выполнение с ней ряда операций, таких как численное интегрирование. В этом случае могут помочь соответствующие параметры. Например, вычисление следующего интеграла дает явно неудобное выражение в виде набора значений, разных для разных интервалов изменения a:

```
> int(1/(x+a)^2,x=0..2);
```

$$\frac{2+2\begin{pmatrix}0 & a\le -2\\ \infty & a<0\\ 0 & 0\le a\end{pmatrix}a+\begin{pmatrix}0 & a\le -2\\ \infty & a<0\\ 0 & 0\le a\end{pmatrix}a^2}{(2+a)\,a}$$

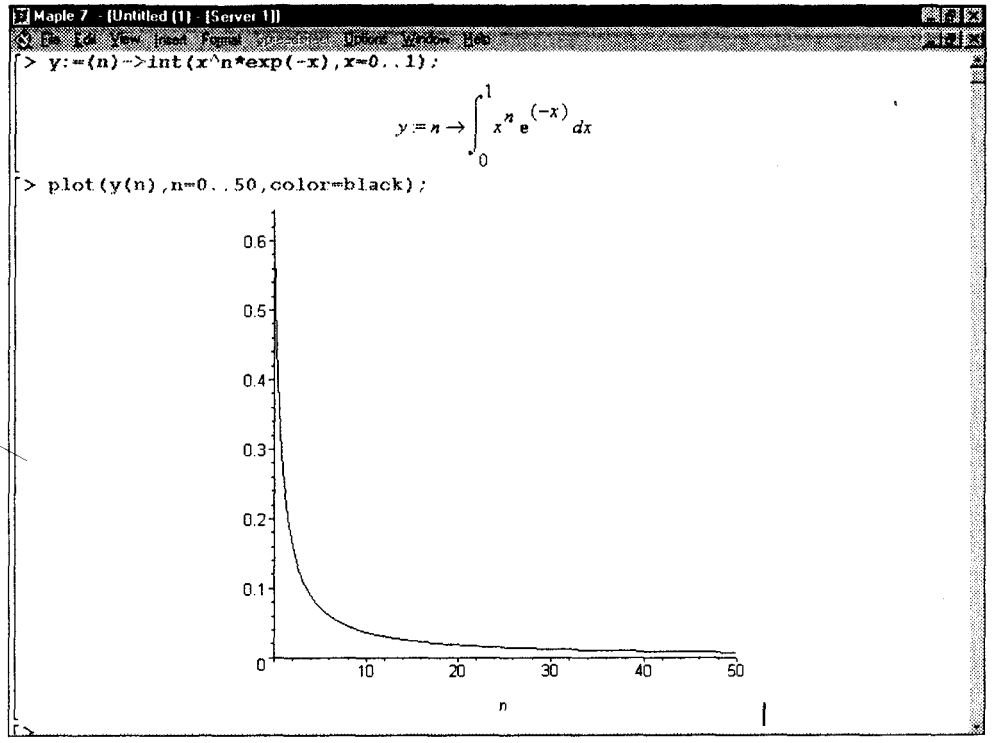


Рис. 8.1. Значение интеграла от x^n*exp(-x) как функция *n*

Увы, попытка вычислить по этому выражению значение интеграла не всегда дает корректный результат. Например, при x от -2 до 0 получаются бесконечные значения. Да и график зависимости значения интеграла от параметра a имеет подозрительный вид (рис. 8.2). Это как раз тот случай, когда с ходу доверяться результатам Maple 7 рискованно.

В данном случае приходится констатировать давно известный факт — системы компьютерной математики (и Maple 7 в их числе) не всесильны и всегда можно найти интегралы даже с обманчиво простым внешним видом, которые поставят систему в тупик или дадут неверные результаты в той или иной области изменения аргументов. Особенно опасны интегралы от кусочных функций с разрывами и интегралы, представляемые такими функциями. Именно к ним и относится обсуждаемый сейчас интеграл. Не меньше проблем вызывают интегралы от функций, области определения которых заданы некорректно или просто не изучены.

Между тем ситуация вовсе не является безнадежной. Надо просто знать, что предпринять, чтобы подсказать системе правильный путь решения. Например, в нашем случае, применив параметр continuous (в апострофах), можно получить куда более простое выражение:

```
> int(1/(x+a)^2,x=0..2,`continuous`);
```

$$2\frac{1}{(2+a)a}$$

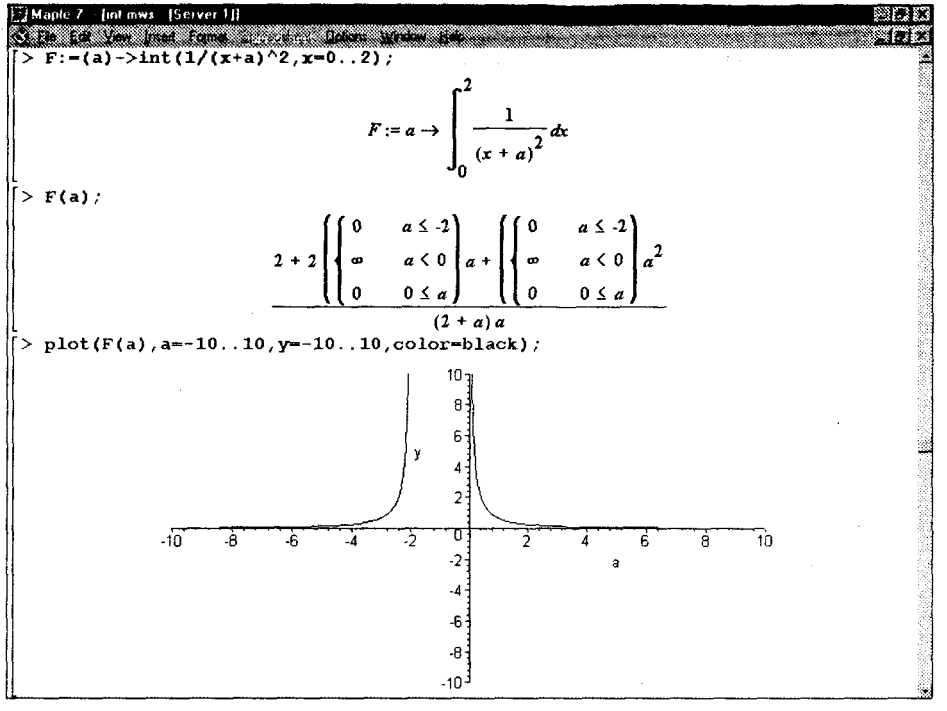


Рис. 8.2. Построение графика зависимости значений интеграла с подынтегральной функцией 1/(x+a)^2 от параметра *a*

Рисунок 8.3 показывает это решение с двумя важными дополнениями — оно представляется функцией пользователя, а ее график строится при изменении *a* от –10 до 10.

Приведем еще один пример «каверзного» интеграла довольно простого вида:

```
> int(1/x^3,x=-1..2);
```

undefined

Этот интеграл вообще не берется функцией `int` без указания параметров (в строке вывода сообщается об этом). Но введение параметра `CauchyPrincipalValue` позволяет получить значение интеграла:

```
> int(1/x^3,x=-1..2,`CauchyPrincipalValue`);
```

$$\frac{3}{8}$$

Возьмем еще один наглядный пример — вычисление интеграла от синусоидальной функции при произвольно больших пределах, но кратных 2π! Очевидно, что при этом (учитывая равность площадей положительной и отрицательной полуволн синусоиды) значение интеграла будет равно 0. Например:

```
> int(sin(x),x=-1000*pi..1000*pi);
```

0

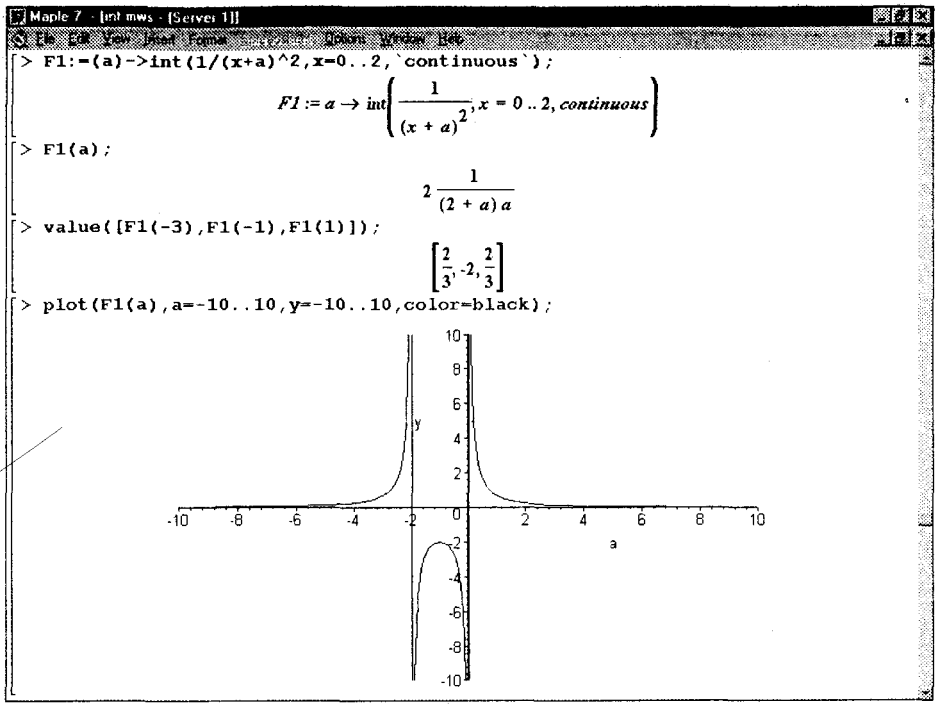


Рис. 8.3. Зависимость значения интеграла с подынтегральной функцией 1/(x+a)^2 и пределами от 0 до 2 от параметра *a*

Однако распространение этого правила на бесконечные пределы интегрирования является грубейшей ошибкой. Интеграл такого рода уже не берется (или говорят, что он не сходится), и Maple 7 дает соответствующий результат:

```
> int(sin(x),x=-infinity..infinity);
```

undefined

Во многих областях техники часто употребляются выражения «затухающая синусоида» или «нарастающая синусоида». Иногда говорят и о «синусоиде с уменьшающейся или возрастающей амплитудой». Бесполезно утверждать, что эти названия *принципиально ошибочны — в рамках допущений, принятых в технических расчетах*, такие утверждения весьма наглядны и эта, в частных случаях вполне оправданная, наглядность с позиций математики идет в ущерб точности фундаментальных определений.

Возьмем, к примеру, широко распространенную функцию: $y(t) = \exp(-t)\sin(2\pi t)$. Построим ее график и вычислим определенный интеграл от этой функции с пределами от 0 до ∞ (рис. 8.4).

С первого взгляда на график видно, что каждая положительная полуволна функции (затухающей «синусоиды») явно больше последующей отрицательной полуволны. К тому же осцилляции функции быстро затухают и через десяток-другой периодов значение функции становится исчезающе малым. Вот почему Maple 7 уверенно вычисляет интеграл с такой подынтегральной функцией. Ее свойство — неопределенность при t→∞ исчезает.

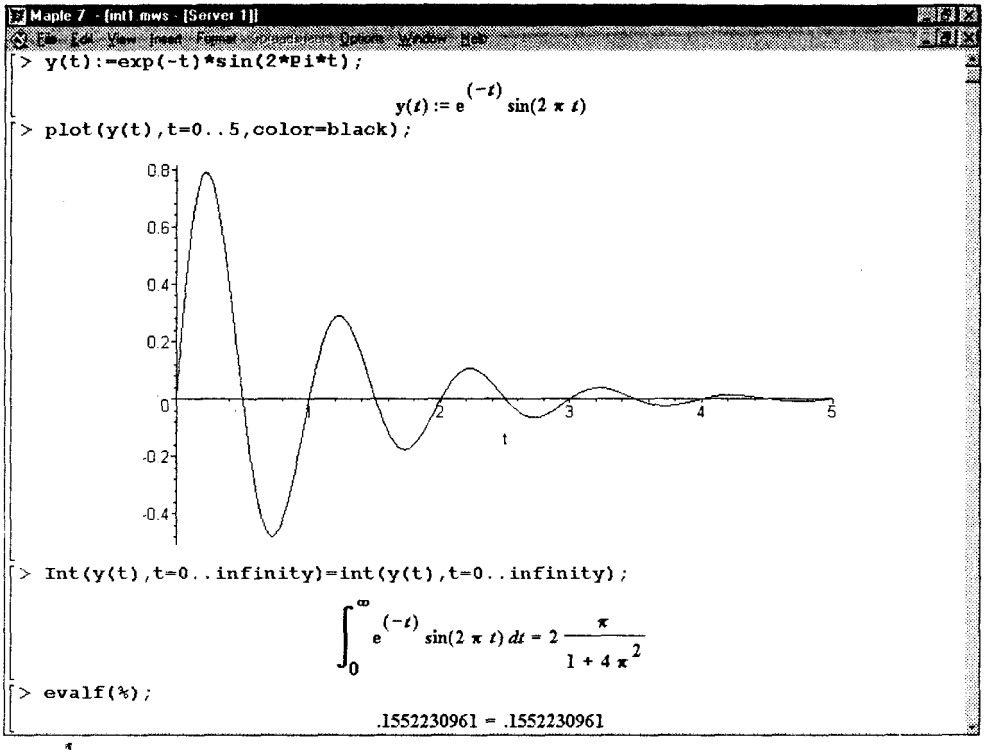


Рис. 8.4. График «затухающей синусоиды» и интеграл от нее с пределами от 0 до ∞

Однако называть такую функцию «затухающей синусоидой», безусловно, неточно. Умножение $\sin(2\pi t)$ на множитель, зависящий от времени t, лишает функцию главного свойства синусоиды — ее строгой симметрии. Так что $\exp(-t)\sin(2\pi t)$ — это совсем новая функция со своими отличительными свойствами. Главные из них — несимметрия при малых t и исчезающе малые значения при больших t. Ни тем, ни другим свойством обычная синусоида не обладает.

А теперь возьмем антипод этой функции — «синусоиду с экспоненциально нарастающей до стационарного значения 1 амплитудой». Такая функция записывается следующим образом:

```
Y(t) = (1 - exp(-t)) sin(2πt).
```

Ее график и попытки вычисления интеграла с такой подынтегральной функцией приведены на рис. 8.5.

Обратите внимание на то, что здесь прямое вычисление интеграла к успеху не привело, хотя из графика функции видно, что каждая положительная полуволна в близкой к $t = 0$ области явно больше по амплитуде, чем последующая отрицательная полуволна. Однако в отличие от предыдущей функции при больших значениях аргумента данная функция вырождается в обычную синусоиду с неизменной (и равной 1) амплитудой. Вот почему трудяга Maple 7 честно отказывается вычислять интеграл от такой коварной функции.

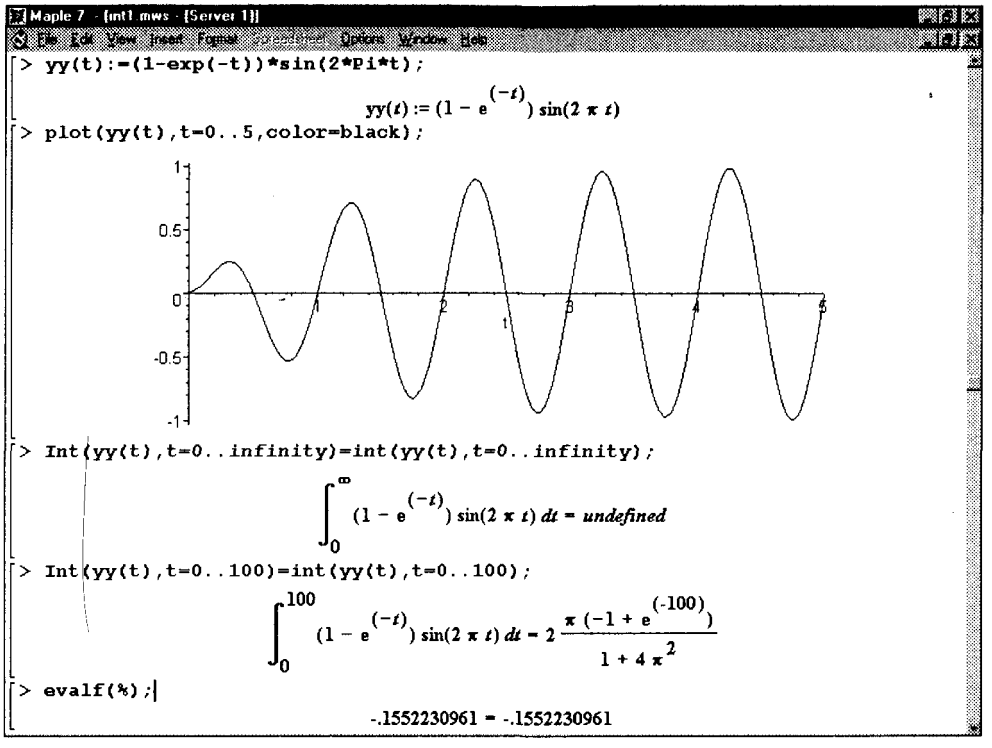


Рис. 8.5. График «экспоненциально нарастающей синусоиды» и интеграл от нее с пределами от 0 до ∞

На этом примере очень четко отслеживается разница в мышлении инженера и математика. Инженер скажет, что интеграл с такой функций должен быть, поскольку вначале положительные площади явно меньше отрицательных, а в дальней области они выравниваются, и потому площадь каждого «периода» функции становится примерно нулевой. По-своему инженер прав — если его не интересует точное определение подынтегральной области в заоблачных высотах бесконечности, то мы должны получить то же значение интеграла, что в предшествующем примере, но со знаком «минус». И в самом деле (см. рис. 8.5), интегрируя в пределах от 0 до 100π, мы получаем именно это значение (опять-таки в пределах погрешности по умолчанию).

И все же прав здесь математик — переход от интегрирования с конечным (да еще и кратным 2π) пределом к интегрированию с бесконечным пределом — далеко не простая операция. Она требует учета поведения функции при значении аргумента, стремящегося к бесконечности, а тут говорить о нулевой алгебраической площади синусоиды некорректно, ибо никакой кратности величине 2π у бесконечности нет! Остается лишь радоваться тому, что система Maple 7 может примирить математиков и инженеров, дав им в руки средства, позволяющие решать подобные задачи с приближениями, приемлемыми для тех или иных категорий пользователей.

Мы подробно рассмотрели этот класс задач потому, что многие важные интегральные преобразования (например, преобразование Фурье) оперируют с по-

добными подынтегральными функциями и надо тщательно разбираться в областях их применения.

ПРИМЕЧАНИЕ

Приведенные выше примеры показывают, что интегрирование является гораздо более тонким делом, чем это кажется на первый взгляд. Тут уместно напомнить, что и студент вуза, и профессор математики университета должны очень внимательно исследовать возможности вычисления интегралов того или иного типа разными математическими системами. Иными словами, применять системы компьютерной математики должны только пользователи, обладающие не столько учеными званиями и степенями, сколько культурой выполнения математических вычислений.

Интегралы с переменными пределами интегрирования

К интересному классу интегралов относятся определенные интегралы с переменными пределами интегрирования. Если обычный определенный интеграл представлен числом (или площадью в геометрической интерпретации), то интегралы с переменными пределами являются функциями этих пределов.

На рис. 8.6 показано два примера задания простых определенных интегралов с переменным верхним пределом (сверху) и обоими пределами интегрирования (снизу).

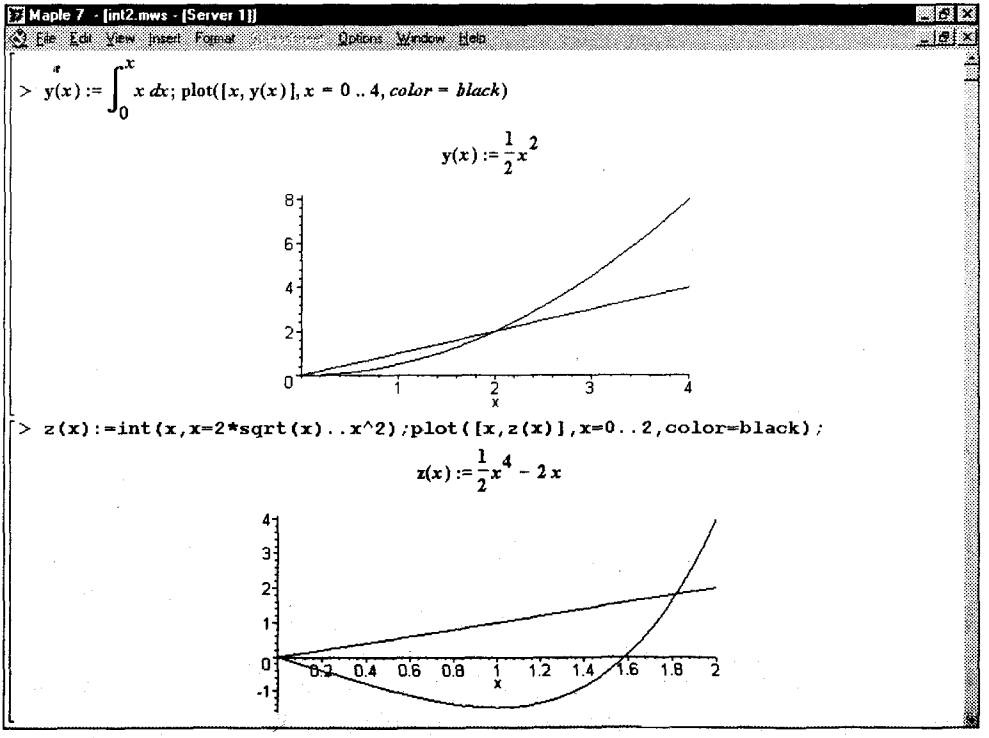


Рис. 8.6. Примеры интегралов с переменными пределами интегрирования

На этом рисунке построены также графики подынтегральной функции (это наклонная прямая) и функции, которую задает интеграл.

Вычисление кратных интегралов

Функции int и Int могут использоваться для вычисления кратных интегралов, например двойных и тройных. Для этого функции записываются многократно:

```
> restart;
> Int(Int(1/(x*y),x=4.0..4.4),y=2.0..2.6);
```

$$\int_{2.0}^{2.6}\int_{4.0}^{4.4}\frac{1}{x\,y}\,dx\,dy$$

```
> value(%);
```

$$.02500598527$$

```
> Int(Int(Int((x^2+y^2)*z, x=0..a), y=0..a), z=0..a);
```

$$\int_0^a\int_0^a\int_0^a (x^2+y^2)\,z\,dx\,dy\,dz$$

```
> value(%);
```

$$\frac{1}{3}a^6$$

```
> Int(Int(2-x-y, x=sqrt(y)..y^2), y=0..1);
```

$$\int_0^1\int_{\sqrt{y}}^{y^2} 2-x-y\,dx\,dy$$

```
> value(%);
```

$$\frac{-11}{30}$$

```
> II := Int(Int(Int(p^2*sin(w), p=0..4*cos(w)), w=0..Pi/4), q=0..Pi/2);
```

$$II := \int_0^{\frac{\pi}{2}}\int_0^{\frac{\pi}{4}}\int_0^{4\cos(w)} p^2\sin(w)\,dp\,dw\,dq$$

$$II := -\frac{8}{3}\cos\left(\frac{1}{4}\pi\right)^4\pi+\frac{8}{3}\pi$$

```
> evalf(I1);
```

$$-2.666666667\cos(.2500000000\,\pi)^4\,\pi+2.666666667\,\pi$$

Обратите внимание на нечеткую работу функции evalf в последнем примере. Эта функция уверенно выдает значение evalf(Pi) в форме вещественного числа с плавающей точкой, но отказывается вычислить значение интеграла, в которое входит число Pi. Этот пример говорит о том, что отдельные недостатки у Maple 7 все же есть, как и поводы для ее дальнейшего совершенствования.

Описанная возможность вычисления кратных интегралов функциями Int и int не является вполне законной. В пакете расширения student имеются дополнительные функции интегрирования, которые дополняют уже описанные возможности. В частности, в этом пакете есть функции для вычисления двойных и тройных интегралов.

Вычисление пределов функций

Для вычисления пределов функции f в точке $x = a$ используются следующие функции:

```
limit(f,x=a);                    limit(f,x=a,dir);
Limit(f,x=a);                    Limit(f,x=a,dir);
```

Здесь f — алгебраическое выражение, x — имя переменной, dir — параметр, указывающий на направление поиска предела (left — слева, right — справа, real — в области вещественных значений, complex — в области комплексных значений). Значением a может быть бесконечность (как положительная, так и отрицательная). Примеры применения этих функций для вычисления пределов в точке приведены ниже:

```
> Limit(f(x),x=a);
```

$$\lim_{x \to a} f(x)$$

```
> Limit(sin(x)/x,x=0)=limit(sin(x)/x,x=0);
```

$$\lim_{x \to 0} \frac{\sin(x)}{x} = 1$$

```
> Limit(1-exp(-x),x=infinity)=limit(1-exp(-x),x=infinity);
```

$$\lim_{x \to \infty} 1 - e^{(-x)} = 1$$

```
> Limit(exp(x),x=infinity)=limit(exp(x),x=infinity);
```

$$\lim_{x \to \infty} e^{x} = \infty$$

```
> Limit(exp(-x),x=infinity)=limit(exp(-x),x=infinity);
```

$$\lim_{x \to \infty} e^{(-x)} = 0$$

```
> Limit((x-sin(x))/x^3,x=0)=limit((x-sin(x))/x^3,x=0);
```

$$\lim_{x \to 0} \frac{x - \sin(x)}{x^3} = \frac{1}{6}$$

```
> Limit((Pi-2*x)*tan(x),x=Pi/2)=limit(tan(x)*(Pi-2*x),x=Pi/2);
```

$$\lim_{x \to (1/2\,\pi)} (\pi - 2x)\tan(x) = 2$$

Обратите внимание на то, что в первом примере фактически дано обозначение предела в самом общем виде. Рисунок 8.7 показывает вычисление пределов функции $\tan(x)$ в точке $x=\pi/2$, а также слева и справа от нее. Для указания направления используются опции right (справа) и left (слева). Видно, что в самой точке предел не определен (значение undefined), а пределы справа и слева уходят в бесконечность.

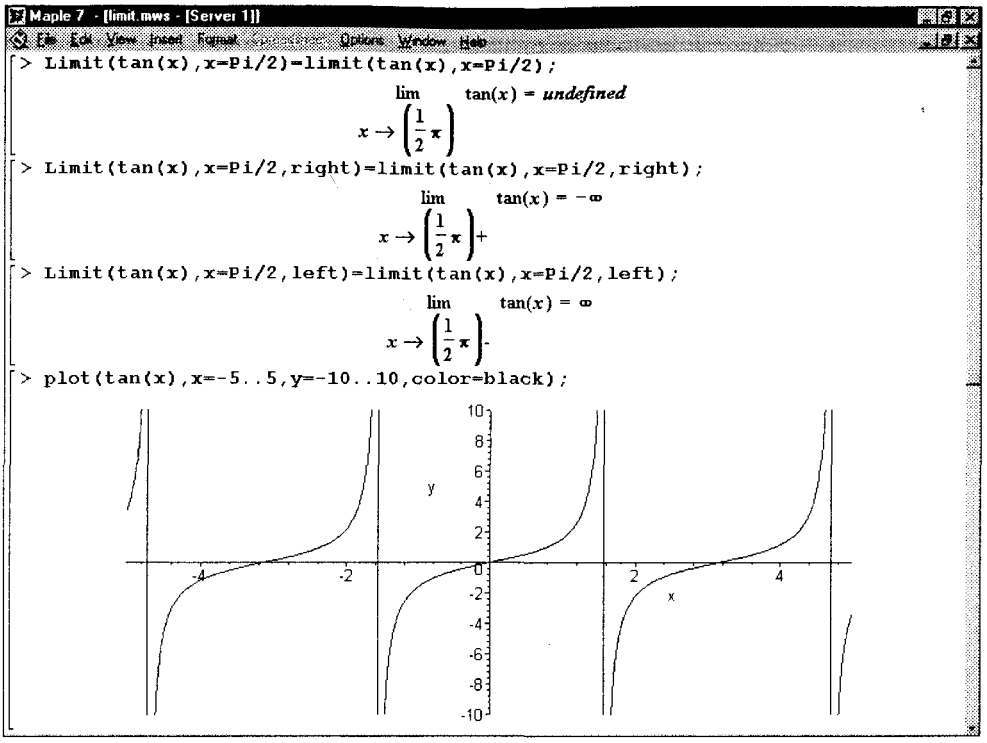


Рис. 8.7. Пример вычисления пределов функции tan(x) и построение ее графика

Показанный на рис. 8.7 график функции $\tan(x)$ наглядно подтверждает существование пределов справа и слева от точки $x = \pi/2$ и отсутствие их в самой этой точке, где функция испытывает разрыв от значения $+\infty$ до $-\infty$.

Разложение функций в ряды

Разложение в степенной ряд

Огромное разнообразие функций давно заставляло математиков задумываться над возможностями их приближенного, но единообразного представления. К таким представлениям относятся различные ряды, сходящиеся к значениям функций в окрестности заданной точки. Для разложения функции или выражения `expr` в обычный степенной ряд служат функции `series(expr, eqn)` и `series(expr, eqn, n)`. Здесь `expr` — разлагаемое выражение, `eqn` — условие (например, в виде `x=a`) или имя переменной (например, `x`) и `n` — необязательное и неотрицательное целое число, задающее число членов ряда (при его отсутствии оно по умолчанию берется равным 6, но может переустанавливаться системной переменной `Order`). Если в качестве `eqn` задано имя переменной, то это соответствует разложению по этой

переменной в области точки с ее нулевым значением. Задав eqn в виде x=x0, можно получить разложение по переменной x в окрестности точки $x = x_0$.

Разложение получается в форме степенного многочлена, коэффициенты которого задаются рациональными числами. Остаточная погрешность задается членом вида O(x)^n. При точном разложении этот член отсутствует. В общем случае для его удаления можно использовать функцию convert. Ниже представлены примеры разложения различных выражений в ряд:

```
> series(sinh(x),x=0);
```

$$x + \frac{1}{6}x^3 + \frac{1}{120}x^5 + O(x^6)$$

```
> series(sinh(x),x=1,3);
```

$$\left(\frac{1}{2}e - \frac{1}{2}\frac{1}{e}\right) + \left(\frac{1}{2}e + \frac{1}{e}\right)(x-1) + \left(\frac{1}{4}e - \frac{1}{4}\frac{1}{e}\right)(x-1)^2 + O((x-1)^3)$$

```
> series(sinh(x),x=1.0,3);
```

$$1.175201193 + 1.543080635\,(x-1.0) + .5876005967\,(x-1.0)^2 + O((x-1.0)^3)$$

```
> series(2*x^2-x+1,x=1,10);
```

$$2 + 3\,(x-1) + 2\,(x-1)^2$$

```
> f(x):=sin(x)/x;
```

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

```
> series(f(x),x=0,10);
```

$$1 - \frac{1}{6}x^2 + \frac{1}{120}x^4 - \frac{1}{5040}x^6 + \frac{1}{362880}x^8 + O(x^9)$$

```
> convert(%,polynom);
```

$$1 - \frac{1}{6}x^2 + \frac{1}{120}x^4 - \frac{1}{5040}x^6 + \frac{1}{362880}x^8$$

```
> s:=series(ln(x),x=2,4);
```

$$s := \ln(2) + \frac{1}{2}(x-2) - \frac{1}{8}(x-2)^2 + \frac{1}{24}(x-2)^3 + O((x-2)^4)$$

```
> evalf(convert(s,polynom));
```

$$-.3068528194 + .5000000000\,x - .1250000000\,(x-2.)^2 + .04166666667\,(x-2.)^3$$

Здесь видно, что член, обозначающий погрешность, отсутствует в тех разложениях, которые точны, например, в разложениях степенных многочленов. Для визуализации приближения рядами заданных аналитических зависимостей очень полезно построить на одном графике кривые аналитической зависимости и разложения в ряд. Мы это покажем чуть позже на примере ряда Тейлора.

Разложение в ряды Тейлора и Маклорена

Для разложения в ряд Тейлора используется функция taylor(expr, eq/nm, n). Здесь expr — разлагаемое в ряд выражение, eq/nm — равенство (в виде x=a) или имя переменной (например, x), n — необязательный параметр, указывающий на по-

рядок разложения и представленный целым положительным числом (при отсутствии указания порядка он по умолчанию принимается равным 6). При задании eq/nm в виде x=a разложение производится относительно точки $x = a$. При указании eq/nm в виде просто имени переменной разложение ищется в окрестности нулевой точки, то есть фактически вычисляется ряд Маклорена.

Ниже представлены примеры применения функции taylor:

```
> taylor(1-exp(x),x=1,4);
```

$$(1-e)-e\,(x-1)-\frac{1}{2}e\,(x-1)^2-\frac{1}{6}e\,(x-1)^3+O((x-1)^4)$$

```
> convert(%,polynom);
```

$$2\frac{x}{\sqrt{\pi}}-\frac{2}{3}\frac{x^3}{\sqrt{\pi}}+\frac{\frac{1}{5}x^5}{\sqrt{\pi}}$$

```
> taylor(sinh(x),x,10);
```

$$x+\frac{1}{6}x^3+\frac{1}{120}x^5+\frac{1}{5040}x^7+\frac{1}{362880}x^9+O(x^{10})$$

```
> taylor(int(sin(x)/x,x),x);
```

$$x-\frac{1}{18}x^3+\frac{1}{600}x^5+O(x^6)$$

```
> taylor(erf(x),x);
```

$$2\frac{1}{\sqrt{\pi}}x-\frac{2}{3}\frac{1}{\sqrt{\pi}}x^3+\frac{1}{5}\frac{1}{\sqrt{\pi}}x^5+O(x^6)$$

Не все выражения (функции) имеют разложение в ряд Тейлора. Ниже дан пример такого рода:

```
> taylor(1/x+x^2,x,5);
    Error, does not have a taylor expansion, try series()
> series(1/x+x^2,x,10);
```

$$x^{-1}+x^2$$

```
> taylor(1/x+x^2,x=1,5);
```

$$2+x-1+2\,(x-1)^2-(x-1)^3+(x-1)^4+O((x-1)^5)$$

Здесь Maple 7 отказалась от вычисления ряда Тейлора в окрестности точки $x = 0$ (по умолчанию) и предложил воспользоваться функцией series. Однако эта функция просто повторяет исходное разложение. В то же время в окрестности точки $x = 1$ ряд Тейлора вычисляется.

Для разложения в ряд Тейлора функций нескольких переменных используется библиотечная функция mtaylor:

```
mtaylor(f, v)
mtaylor(f, v, n)
mtaylor(f, v, n, w)
```

Здесь f — алгебраическое выражение, v — список имен или равенств, n — необязательное число, задающее порядок разложения, w — необязательный список целых чисел, задающих «вес» каждой из переменных списка v. Эта функция должна вызываться из библиотеки Maple 7 с помощью команды readlib:

```
> readlib(mtaylor);
> mtaylor(sin(x*y),[x,y],10,[2,1]);
 proc() ... end proc
```

$$x\,y-\frac{1}{6}x^3\,y^3$$

```
> mtaylor(exp(-x)*sin(y),[x,y],5);
```

$$y-x\,y-\frac{1}{6}y^3+\frac{1}{2}x^2\,y+\frac{1}{6}x\,y^3-\frac{1}{6}x^3\,y$$

Для получения только коэффициента при k-м члене ряда Тейлора можно использовать функцию `coeftayl(expr,var,k)`. Если `expr` — функция нескольких переменных, то `k` должен задаваться списком порядков коэффициентов.

Пример документа — разложение синуса в ряд

Полезно сочетать разложение выражений (функций) в ряд Тейлора с графической визуализацией такого разложения. Рассмотрим документ, в котором наглядно показаны возможности представления функции рядами Тейлора и Маклорена.

На рис. 8.8 показана первая часть документа. Она дает пример разложения в ряд Тейлора функции $\sin(x)$ с построением ее графика и графика по разложению в ряд. Поскольку выбрано разложение относительно точки $x = 0$, то полученный ряд является рядом Маклорена. Это хороший пример визуализации результатов математических вычислений — здесь наглядно видно, что при малых значениях x график ряда практически повторяет разлагаемую функцию, но затем начинает сильно от нее отходить.

Обратите внимание, несмотря на то что мы задали шестой порядок ряда, последний член имеет только пятый порядок. Это связано со спецификой данного разложения — в нем просто отсутствуют члены четного порядка.

Можно буквально в считанные секунды попробовать изменить число членов ряда или диапазон изменения переменной x, что и показано на рис. 8.9 (вторая часть документа). При этом легко убедиться в том, что при больших x поведение ряда не имеет ничего общего с поведением разлагаемой в ряд функции, в частности нет и намека на периодичность разложения, которая присуща тригонометрической функции $\sin(x)$.

В заключительной (третьей) части этого документа (рис. 8.10) представлено уже истинное разложение синуса в ряд Тейлора в окрестности смещенной от нуля точки $x = 1$. При смещении точки, относительно которой ведется разложение, выражение для ряда Тейлора существенно изменяется. В нем, во-первых, появляются члены четных степеней, а во-вторых, фигурирует аргумент вида $(x - 1)^n$.

Нетрудно заметить, что даже при представлении такой «простой» функции, как $\sin(x)$, приемлемая погрешность представления одного периода достигается при числе членов ряда Тейлора порядка 10 и более. Однако существенное повышение порядка ряда нецелесообразно из-за резкого возрастания вычислительных погрешностей. Кроме того, серьезным недостатком аппроксимации рядом Тейлора является непредсказуемое поведение полинома вдали от точки, относительно которой задается представление. Это хорошо видно на всех трех приведенных примерах.

Maple 7 - [ataylor.mws - [Server 1]]

Аппроксимация рядом Тейлора (Маклорена)

Вычислим представление функции sin(x) в форме ряда Тейлора (при x=0 - Маклорена):

```
> approx := taylor( sin(x), x=0, 6 );
```

$$approx := x - \frac{1}{6}x^3 + \frac{1}{120}x^5 + O(x^6)$$

Преобразуем это выражение в степенной многочлен - полином:

```
> poly := convert( approx, polynom ):
```

А теперь построим графики исходной функции и ее разложения в ряд:

```
> plot( { sin(x), poly }, x=-4..4, title=`Sin(x) and Taylor (Macloren)
  Series Approximation (order 6)`,color=black);
```

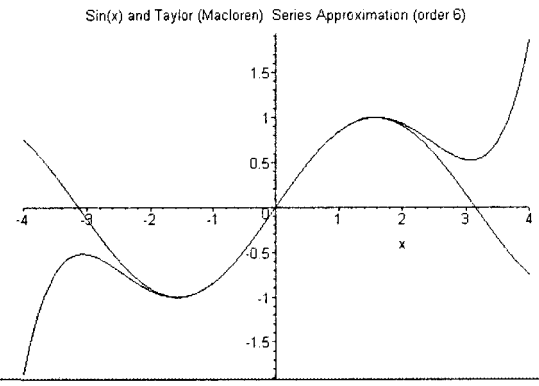


Рис. 8.8. Разложение функции sin(x) в ряд Маклорена 6-го порядка и построение ее графика

Maple 7 - [ataylor.mws - [Server 1]]

Зададим сразу полиномиальное представление для аппроксимации рядом Тейлора (Маклорена), но теперь уже 12-го порядка:

```
> poly2 := convert(taylor( sin(x), x=0, 12),polynom);
```

$$poly2 := x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 + \frac{1}{362880}x^9 - \frac{1}{39916800}x^{11}$$

Построим графики исходной функции и аппроксимации для этого случая:

```
> plot( { sin(x), poly2 }, x=-5..5, title=`Sin(x)and Taylor (Macloren) Series
  (order 12)`,color=black);
```

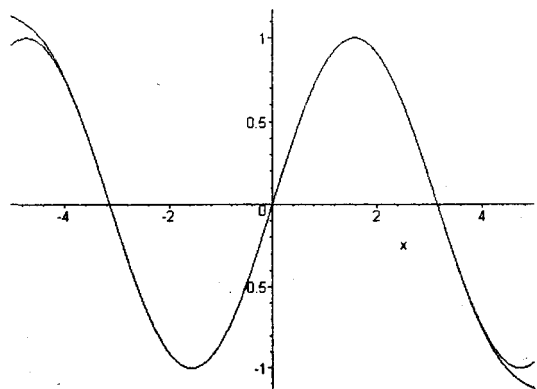


Рис. 8.9. Разложение функции sin(x) в ряд Маклорена 12-го порядка и построение ее графика

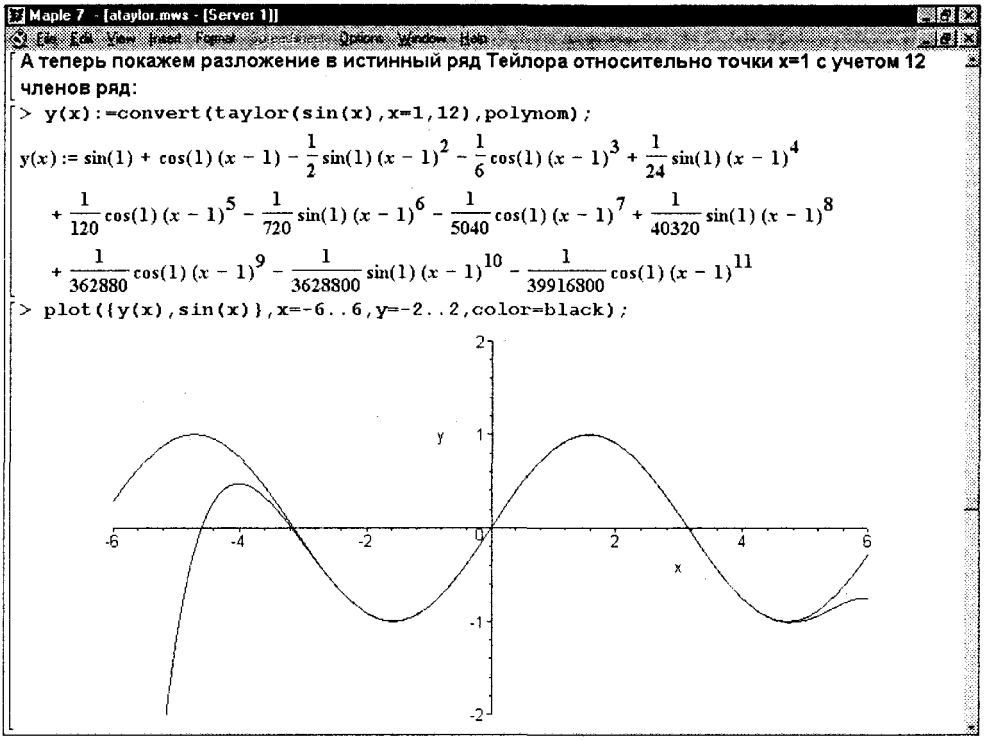


Рис. 8.10. Разложение функции sin(*x*) в ряд Тейлора 12-го порядка относительно точки *x* = 1 и построение ее графика

Помимо указанных выше разложений в ряд Maple 7 имеет множество функций для иных разложений. Например, в пакете numapprox имеется функция `laurent(expr,var,n)`, позволяющая получить разложение в ряд Лорана, функция `chebyshev(expr, eq/nm, eps)` дает разложение в форме полиномов Чебышева и т. д.

Решение уравнений и неравенств

Основная функция solve

Решение линейных и нелинейных уравнений и неравенств — еще одна важная область математического анализа. Maple 7 имеет мощные средства для такого решения. Так, для решения линейных и нелинейных уравнений в аналитическом виде используется достаточно универсальная и гибкая функция `solve(eqn, var)` или `solve({eqn1,eqn2,...},{var1,var2,...})`, где `eqn` — уравнение, содержащее функцию ряда переменных, `var` — переменная, по которой ищется решение. Если при записи `eqn` не используются знак равенства или знаки отношения, считается, что `solve` ищет корни уравнения `eqn=0`.

Характер решений можно изменить с помощью глобальных переменных:

- _SolutionsMayBeLost — при значении true дает решение, которое при обычном применении функции solve возвращает значения NULL;
- _MaxSols — задает максимальное число решений;
- _EnvAllSolutions — при значении true задает выдачу всех решений.

В решениях могут встречаться следующие обозначения:

- _NN — указывает на неотрицательные решения;
- _B — указывает на решения в бинарной форме;
- _Z — указывает на то, что решение содержит целые числа;
- %N — при текстовом формате вывода задает общие члены решения и обеспечивает более компактную форму его представления.

В форме solve[subtopic] возможны параметры subtopic функции solve следующих типов:

```
floats      functions     identity      ineq          linear
radical     scalar        series        system
```

При решении систем уравнений они и список переменных задаются как множества, то есть в фигурных скобках. При этом и результат решения получается в виде множества. Чтобы преобразовать его к обычному решению, нужно использовать функцию assign, которая обеспечивает присваивание переменным значений, взятых из множества.

Функция solve старается дать решение в аналитическом виде. Это не означает, что ее нельзя использовать для получения корней уравнений в численном виде. Просто для этого придется использовать функции evalf или convert. Если результат решения представлен через функцию RootOf, то зачастую можно получить все корни с помощью функции allvalues.

Решение одиночных нелинейных уравнений

Решение одиночных нелинейных уравнений вида $f(x) = 0$ легко обеспечивается функций solve(f(x),x). Это демонстрируют следующие примеры:

```
> solve(x^3-2*x+1,x);
```

$$1, -\frac{1}{2}+\frac{1}{2}\sqrt{5}, -\frac{1}{2}-\frac{1}{2}\sqrt{5}$$

```
> solve(x^(3/2)=3,x);
```

$$3^{(2/3)}$$

```
> evalf(%);
```

$$2.080083823$$

```
> solve(sqrt(ln(x))=2,x);
```

$$e^4$$

```
> evalf(%);
```

$$54.59815003$$

Часто бывает удобно представлять уравнение и его решение в виде отдельных объектов, отождествленных с определенной переменной:

```
> eq:=(2*x^2+x+3=0);
```

$$eq := 2x^2 + x + 3 = 0$$

```
> s:=[solve(eq,x)];
```

$$s := \left[-\frac{1}{4} + \frac{1}{4} I\sqrt{23}, -\frac{1}{4} - \frac{1}{4} I\sqrt{23} \right]$$

В частности, это позволяет легко проверить решение (даже если оно не одно, как в приведенном примере) подстановкой (subs):

```
> subs(x=s[1],eq);
```

$$2\left(-\frac{1}{4} + \frac{1}{4} I\sqrt{23} \right)^2 + \frac{11}{4} + \frac{1}{4} I\sqrt{23} = 0$$

```
> subs(x=s[2],eq);
```

$$2\left(-\frac{1}{4} - \frac{1}{4} I\sqrt{23} \right)^2 + \frac{11}{4} - \frac{1}{4} I\sqrt{23} = 0$$

```
> evalf(%);
```

$$0. + 0.\,I = 0.$$

Сводящиеся к одному уравнению равенства вида $f1(x) = f2(x)$ также решаются функцией solve(f1(x)=f2(x),x):

```
> solve(x^4=-x-1,x);
```

$$\text{RootOf}(\_Z^4 + \_Z + 1, index = 1), \text{RootOf}(\_Z^4 + \_Z + 1, index = 2),$$
$$\text{RootOf}(\_Z^4 + \_Z + 1, index = 3), \text{RootOf}(\_Z^4 + \_Z + 1, index = 4)$$

```
> evalf(%);
```

$$.7271360845 + .9340992895\,I, -.7271360845 + .4300142883\,I,$$
$$-.7271360845 - .4300142883\,I, .7271360845 - .9340992895\,I$$

```
> solve({exp(x)=sin(x)},x);
```

$$\{x = \text{RootOf}(\_Z - \ln(\sin(\_Z)))\}$$

```
> evalf(%);
```

$$\{x = .3627020561 - 1.133745919\,I\}$$

```
> solve(x^4=2*x,x);
```

$$0, 2^{(1/3)}, -\frac{1}{2}2^{(1/3)} + \frac{1}{2} I\sqrt{3}\,2^{(1/3)}, -\frac{1}{2}2^{(1/3)} - \frac{1}{2} I\sqrt{3}\,2^{(1/3)}$$

```
> evalf(%);
```

$$0., 1.259921050, -.6299605250 + 1.091123636\,I, -.6299605250 - 1.091123636\,I$$

Обратите внимание в этих примерах на эффективность применения функции evalf, позволяющей получить решения, выраженные через функцию RootOf, в явном виде.

Решение тригонометрических уравнений

Функция solve может использоваться для решения тригонометрических уравнений:

```
> solve(sin(x)=.2,x);
```

.2013579208

```
> solve(sin(x)-1/2,x);
```

$$\frac{1}{6}\pi$$

```
> solve(cos(x)=.5,x);
```

1.047197551

Однако из приведенных примеров видно, что при этом найдено только одно (главное) решение. Периодичность тригонометрических функций и связанная с этим множественность решений оказались проигнорированы. Однако можно попытаться найти все периодические решения, выполнив следующую команду:

```
> _EnvAllSolutions:=true;
```

_EnvAllSolutions := *true*

Указанная в ней системная переменная отвечает за поиск всех периодических решений, когда ее значение равно true, и дает поиск только главных решений при значении false, принятом по умолчанию. Так что теперь можно получить следующее:

```
> solve(sin(x)=1/2,x);
```

$$\frac{1}{6}\pi + \frac{2}{3}\pi \_B1\sim + 2\,\pi \_Z1\sim$$

На рис. 8.11 показан более сложный случай решения нелинейного уравнения вида $f_1(x) = f_2(x)$, где $f_1(x) = \sin(x)$ и $f_2(x) = \cos(x) - 1$.

Решение дано в графическом виде и в аналитическом для двух случаев — нахождения главных значений корней и нахождения всех корней.

В решениях встречаются переменные _B1~ и _Z1~, означающие ряд натуральных чисел. Благодаря этому через них можно представить периодически повторяющиеся решения.

Примеры решения уравнений с обратными тригонометрическими функциями показаны ниже:

```
> eqns := 2*arcsin(x) - arccos(5*x);
```

eqns := $2\arcsin(x) - \arccos(5x)$

```
> solve( eqns, {x} );
```

$$\left\{ x = \frac{1}{4} \frac{-5 + \sqrt{33}}{\sqrt{\left(-\frac{5}{4} + \frac{1}{4}\sqrt{33}\right)^2 - \frac{21}{8} + \frac{5}{8}\sqrt{33}}} \right\}$$

```
> eqns := arccos(x) - arctan(x/2);
```

$$eqns := \arccos(x) - \arctan\left(\frac{1}{2}x\right)$$

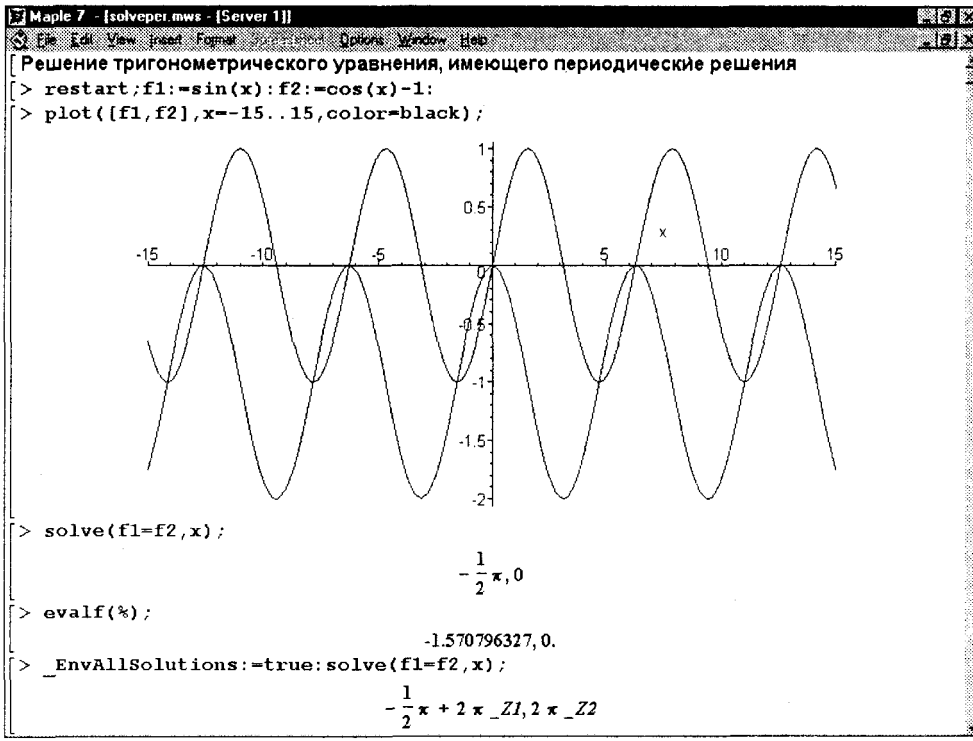


Рис. 8.11. Пример решения уравнения, имеющего периодические решения

```
> solve( eqns, {x} );
```

$$\{x = \frac{\sqrt{-2+2\sqrt{2}}}{\sqrt{(\sqrt{2}-1)^2-2+2\sqrt{2}}}\}$$

Решение систем линейных уравнений

Для решения систем линейных уравнений созданы мощные матричные методы, которые будут описаны отдельно. Однако функция `solve` также может с успехом решать системы линейных уравнений. Такое решение в силу простоты записи функции может быть предпочтительным. Для решения система уравнений и перечень неизвестных задаются в виде множеств (см. приведенные ниже примеры).

Рисунок 8.12 дает два примера решения систем из двух линейных уравнений. В первом примере функция `solve` возвращает решение в виде значений неизвестных x и y, а во втором отказывается это делать.

В чем дело? Оказывается, в том, что во втором случае система просто не имеет решения. Импликативная графика пакета расширения `plots` дает прекрасную возможность проиллюстрировать решение. Так, нетрудно заметить, что в первом случае геометрическая трактовка решения сводится к нахождению точки

пересечения двух прямых, отображающих два уравнения. При этом имеется единственное решение, дающее значения x и y.

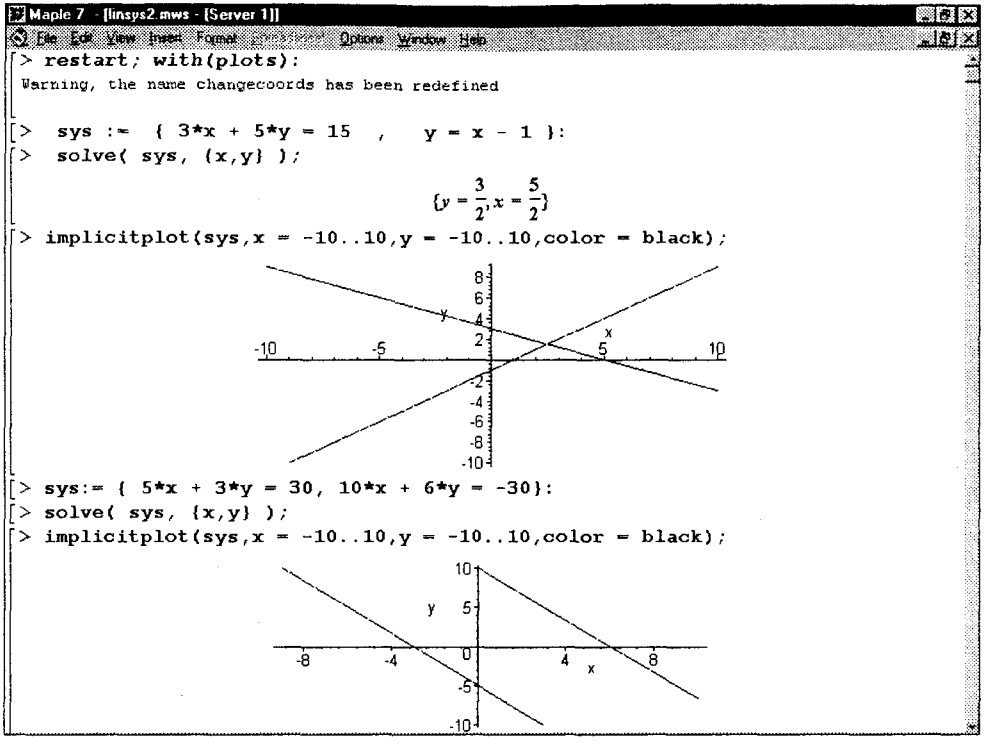


Рис. 8.12. Примеры решения системы из двух линейных уравнений с графической иллюстрацией

Во втором случае решения и впрямь нет, ибо уравнения задают параллельно расположенные прямые, которые никогда не пересекаются. Рекомендуем читателю самостоятельно проверить и третий случай — бесконечного множества решений. Он имеет место, если оба уравнения описывают одну и ту же зависимость и их графики сливаются в одну прямую.

Решение систем из трех линейных уравнений также имеет наглядную геометрическую интерпретацию — в виде точки, в которой пересекаются три плоскости, каждая из которых описывается функцией двух переменных. Для наглядности желательно представить и линии пересечения плоскостей. Это позволяет сделать функция импликативной трехмерной графики `implicitplot3d`, что и показано на рис. 8.13. Для объединения графиков площадей использована функция `display`.

Некоторые проблемы с решением систем из трех линейных уравнений иллюстрируют примеры, приведенные на рис. 8.14. В первом примере решения вообще нет. График показывает, в чем дело, — линии пересечения плоскостей идут параллельно и нигде не пересекаются. Во втором примере все три плоскости пересекаются по одной линии.

```
> restart; with(plots):
Warning, the name changecoords has been redefined

> sys := { z = 4, x+y= 10, x-y = 5 }:
> solve( sys, {x,y,z} );
```

$$\{z = 4, y = \frac{5}{2}, x = \frac{15}{2}\}$$

```
> display(implicitplot3d(sys, x = -10..10, y = -10..10, z=-10..10, shading =
  xyz,style=patchnogrid, orientation = [158,70]), spacecurve({[t,10-t,4],
  [t,t-5,4], [15/2,5/2,t]},t= -10..10, color = black, thickness = 2));
```

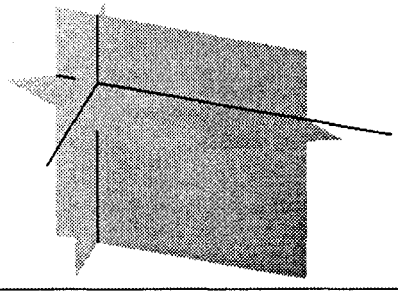
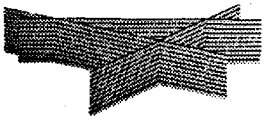


Рис. 8.13. Пример решения системы из трех линейных уравнений с графической иллюстрацией решения

```
> sys := { 2*x - 3*y = 10, 2*x+y= 7, y = 2*x + 4 }:
> solve( sys, {x,y,z} );
> implicitplot3d( sys, x = -10..10, y = -10..10, z=-10..10,
    style= patchcontour, orientation =[-25,30]);
```



```
> sys := { x + y + z = 1, x + 3*y - z = 2, 2*x + 4*y = 3 }:
> solve( sys, {x,y,z} );
```

$$\{x = \frac{1}{2} - 2z, z = z, y = \frac{1}{2} + z\}$$

```
> display(implicitplot3d( sys, x = -10..10, y = -10..10, z=-10..10,
  orientation = [-30,110], style = patchnogrid), spacecurve([1/2 - 2*t,
  1/2 + t,t],t = -10..10, color = black, thickness = 3));
```

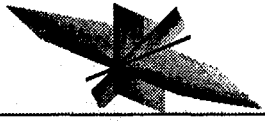


Рис. 8.14. Графическая иллюстрация особых случаев решения системы из трех линейных уравнений

Следующий пример показывает решение системы из четырех линейных уравнений:

```
> sys := { 4*x1 + 7*x2 - x3 + 3*x4 = 11,
                          -2*x1 + 2*x2 - 6*x3 + x4 = 4,
                          x1 - 3*x2 + 4*x3 - x4 = -3,
                          3*x1 - 5*x2 - 7*x3 + 5*x4 = 8 }:
> solve( sys, {x1, x2, x3, x4} );
```

$$\{x2 = \frac{8}{19}, x3 = \frac{-81}{19}, x1 = \frac{135}{19}, x4 = \frac{-156}{19}\}$$

Эта система имеет решение, но его простая графическая иллюстрация уже невозможна.

Случай решения неполной системы уравнений (уравнений — 3, а неизвестных — 4) иллюстрирует следующий пример:

```
> sys := { x1 + 2*x2 + 3*x3 + 4*x4 = 51,
                          x1 - 3*x2 + 4*x3 +  x4  = 32,
                          x1 + 2*x2 - 6*x3 +  x4  = -23   }:
> solve( sys, {x1, x2, x3, x4} );
```

$$\{x2 = \frac{2}{5}x1 - \frac{11}{15}, x4 = -\frac{3}{5}x1 + \frac{139}{15}, x3 = \frac{1}{5}x1 + \frac{77}{15}, x1 = x1\}$$

Как видно из приведенных примеров, функция `solve` неплохо справляется с решением систем линейных уравнений.

Решение систем нелинейных и трансцендентных уравнений

Функция `solve` может использоваться для решения систем нелинейных и трансцендентных уравнений. Для этого система уравнений и перечень неизвестных задаются в виде множеств. Ниже приведены примеры решения уравнений:

```
> restart;
> solve({x*y=a,x+y=b},{x,y});
```

$$\{y = \text{RootOf}(\_Z^2 - \_Z\, b + a), x = -\text{RootOf}(\_Z^2 - \_Z\, b + a) + b\}$$

```
> allvalues(%);
```

$$\{y = \frac{1}{2}b + \frac{1}{2}\sqrt{b^2 - 4\,a}, x = \frac{1}{2}b - \frac{1}{2}\sqrt{b^2 - 4\,a}\},$$

$$\{y = \frac{1}{2}b - \frac{1}{2}\sqrt{b^2 - 4\,a}, x = \frac{1}{2}b + \frac{1}{2}\sqrt{b^2 - 4\,a}\}$$

```
> s:=solve({x*y=2,x+y=3},{x,y});
```

$$s := \{y = 1, x = 2\}, \{y = 2, x = 1\}$$

```
> assign(s);x;y;
```

$$1$$

$$2$$

```
> unassign('x');y:='y';
```

$$y := y$$

```
> [x,y];
```

$$[x, y]$$

В этих примерах хорошо видна техника работы с функциями solve и assign. В конце примеров показано восстановление неопределенного статуса переменных x и y с помощью функции unassign и снятие определения переменных с помощью заключения их в прямые апострофы.

Функция RootOf

В решениях уравнений нередко появляется функция RootOf, означающая, что корни нельзя выразить в радикалах. Эта функция применяется и самостоятельно в виде RootOf(expr) или RootOf(expr, x), где expr — алгебраическое выражение или равенство, x — имя переменной, относительно которой ищется решение. Если x не указана, ищется универсальное решение по переменной _Z. Когда expr задано не в виде равенства, решается уравнение expr=0. Для получения решений вида RootOf в явном виде может использоваться функция allvalues.

Примеры применения функции RootOf:

```
> RootOf(x^2+1=0,x);
```

$$\mathrm{RootOf}(\_Z^2+1)$$

```
> allvalues(%);
```

$$I, -I$$

```
> RootOf(a*b^2+a/b,b);
```

$$\mathrm{RootOf}(\_Z^3+1)$$

```
> allvalues(%);
```

$$-1, \frac{1}{2}+\frac{1}{2}I\sqrt{3}, \frac{1}{2}-\frac{1}{2}I\sqrt{3}$$

```
> RootOf(x^3-1,x)mod 7;
```

$$\mathrm{RootOf}(\_Z^3+6)$$

```
> allvalues(%);
```

$$-6^{(1/3)}, \frac{1}{2}6^{(1/3)}-\frac{1}{2}I\sqrt{3}\,6^{(1/3)}, \frac{1}{2}6^{(1/3)}+\frac{1}{2}I\sqrt{3}\,6^{(1/3)}$$

```
> evalf(%);
```

-1.817120593, .9085602965 – 1.573672596 *I*, .9085602965 + 1.573672596 *I*

```
> RootOf(x^2-2*x+1,x)mod 5;
```

1

Итак, функция RootOf является эффективным способом представления решения в компактном виде. Как уже отмечалось, наряду с самостоятельным применением она часто встречается в составе результатов решения нелинейных уравнений.

Решение уравнений со специальными функциями

К важным достоинствам Maple 7 относится возможность решения уравнений, содержащих специальные функции как в записи исходных выражений, так и в результатах решения. Приведем несколько примеров такого рода:

```
> eqns := Psi(3*x-99) - Psi(3*x-100) + 3/x^2;
```

$$eqns := \Psi(3x-99) - \Psi(3x-100) + \frac{3}{x^2}$$

```
> solve( eqns, {x} );
```

$$\{x = -\frac{9}{2} + \frac{1}{2}\sqrt{1281}\}, \{x = -\frac{9}{2} - \frac{1}{2}\sqrt{1281}\}$$

```
> eqns := max(x,3*x-12)=min(10*x+8,22-x);
```

$$eqns := \max(-12+3x, x) = \min(10x+8, 22-x)$$

```
> solve( eqns, {x} );
```

$$\{x = \frac{-8}{9}\}, \{x = \frac{17}{2}\}$$

```
> eqns := LambertW(3*x)=ln(x);
```

$$eqns := \text{LambertW}(3x) = \ln(x)$$

```
> solve( eqns, {x} );
```

$$\{x = e^3\}$$

Решение неравенств

Неравенства в математике встречаются почти столь же часто, как и равенства. Они вводятся знаками отношений, например: > (больше), < (меньше) и т. д. Решение неравенств существенно расширяет возможности функции solve. При этом неравенства задаются так же, как и равенства. Приведенные на рис. 8.15 примеры поясняют технику решения неравенств.

Из приведенных примеров очевидна форма решений — представлены критические значения аргумента, вплоть до не включаемых значений области действия неравенства (они указываются словом Open). Всегда разумным является построение графика выражения, которое задает неравенство, — это позволяет наглядно убедиться в правильности решения.

Приведем еще несколько примеров решения неравенств в аналитической форме:

```
> solve(5*x>10,x);
```

$$\text{RealRange}(\text{Open}(2), \infty)$$

```
> solve(5*x>=10,x);
```

$$\text{RealRange}(2, \infty)$$

```
> solve(ln(x)>2,x);
```

$$\text{RealRange}(\text{Open}(e^2), \infty)$$

```
> solve(exp(x)>10,x);
```

$$\text{RealRange}(\text{Open}(\ln(10)), \infty)$$

```
> solve(a*x>b,{x});
```

$$\{-\text{signum}(a)\, x < -\frac{\text{signum}(a)\, b}{a}\}$$

```
> eqns := abs(z)^2/(z+1) < exp(2)/(exp(1)-1);
```

$$eqns := \frac{|z|^2}{z+1} < \frac{e^2}{e-1}$$

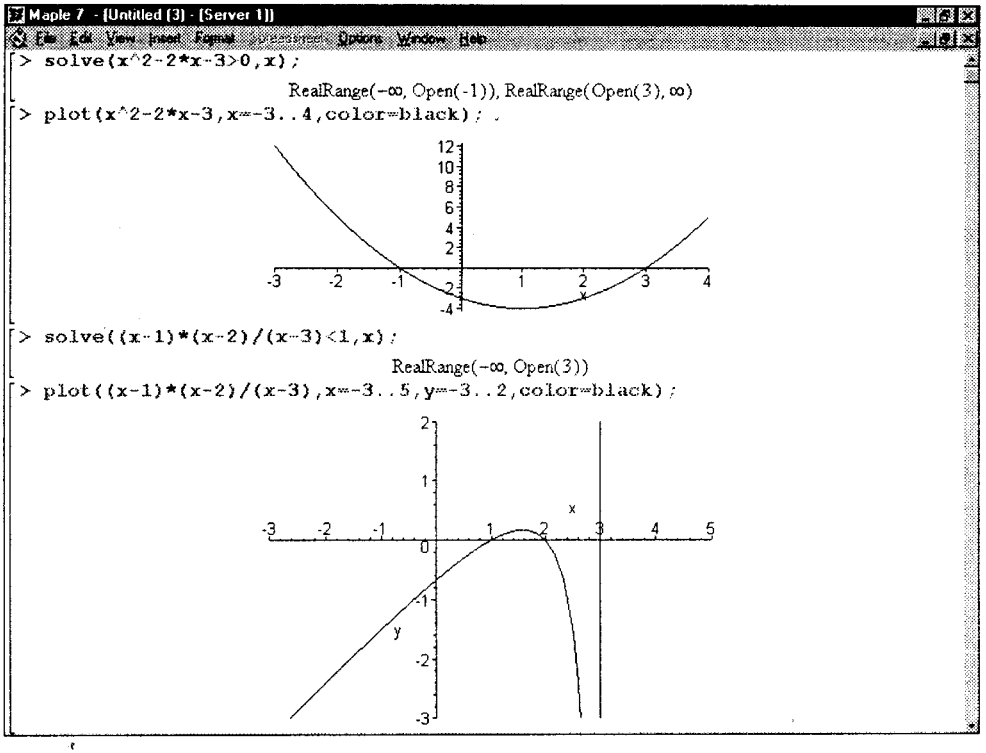


Рис. 8.15. Примеры, иллюстрирующие решение неравенств

```
> solve( eqns, {z} );
```

$$\left\{z<\frac{1}{2}\frac{e^2+\sqrt{(e^2)^2+4\,e^2\,e-4\,e^2}}{e-1},\ \frac{1}{2}\frac{e^2-\sqrt{(e^2)^2+4\,e^2\,e-4\,e^2}}{e-1}<z\right\},\ \{z<-1\}$$

```
> eqns := exp(x)*x^2 >= 1/2;
```

$$eqns := \frac{1}{2} \le e^x x^2$$

```
> solve( eqns, {x} );
```

$$\{2\ \text{Lambert W}\left(-1,-\frac{1}{4}\sqrt{2}\right)\le x, x\le 2\ \text{Lambert W}\left(-\frac{1}{4}\sqrt{2}\right)\},\{2\ \text{Lambert W}\left(-1,-\frac{1}{4}\sqrt{2}\right)\le x\}$$

```
> eqns := abs( (z+abs(z+2))^2-1 )^2 = 9;
```

$$eqns := \left|(z+|z+2|)^2-1\right|^2 = 9$$

```
> solve( eqns, {z} );
```

$$\{z=0\},\{z\le -2\}$$

```
> evalf(%);
```

$$\{-2.617866616\le x, x\le -1.487962064\},\{.5398352768\le x\}$$

```
> eqns := { x^2<1, y^2<=1, x+y<1/2 }:
```

$$eqns := \{x^2 < 1, y^2 \le 1, x + y < \frac{1}{2}\}$$

```
> solve( eqns, { x, y } );
```

$$\{y \le 1, -1 \le y, x + y < \frac{1}{2}, -1 < x, x < 1\}$$

```
> solve({x*y*z>0,x>-1,y+z>10},{x,y,z});
```

$$\{z = 0, -1 < x, 10 < y\}, \{y = 0, -1 < x, 10 < z\}$$

В последних примерах показано решение систем неравенств. При этом выдаются области определения нескольких переменных.

Решение функциональных уравнений

Решение функционального уравнения, содержащего в составе равенства некоторую функцию $f(x)$, заключается в нахождении этой функции. Для этого можно использовать функцию solve, что демонстрируют приведенные ниже примеры:

```
> A:=solve(f(x)^2-x+1,f);
```

$A := \text{proc}(x)\ \text{RootOf}(\_Z^2 - x + 1, label = \_L1)\ \text{end proc}$

```
> convert(A(x),radical);
```

$$\sqrt{x - 1}$$

```
> allvalues(%);
```

$$\sqrt{x - 1}$$

```
> B:=solve(f(x)*x=ln(x^2),f);
```

$B := \text{proc}(x)\ \ln(x^2)/x\ \text{end proc}$

```
> convert(B(x),radical);
```

$$\frac{\ln(x^2)}{x}$$

```
> C:=solve(f(x)*x^2=a*x^2+b*x+c,f);
```

$C := \text{proc}(x)\ (a \times x^2 + b \times x + c)/x^2\ \text{end proc}$

```
> convert(C(x),radical);
```

$$\frac{a x^2 + b x + c}{x^2}$$

Решение уравнений с линейными операторами

Maple 7 позволяет решать уравнения с линейными операторами, например с операторами суммирования рядов и дифференцирования. Ограничимся одним примером такого рода:

```
> S := sum( (a+b*exp(x[i])-y[i])^2, i=0..n );
```

$$S := (n+1)a^2 + \left(\sum_{i=0}^{n} \left(2b\,e^{x_i} a - 2a\,y_i + b^2 (e^{x_i})^2 - 2b\,e^{x_i} y_i + y_i^2 \right) \right)$$

```
> eqns := { diff(S, a), diff(S,b)};
```

$$eqns := \left\{ \sum_{i=0}^{n} \left(2\,e^{x_i} a + 2b (e^{x_i})^2 - 2\,e^{x_i} y_i \right), 2(n+1)a + \left(\sum_{i=0}^{n} \left(2b\,e^{x_i} - 2y_i \right) \right) \right\}$$

```
> solve( eqns, {a, b} );
```

$$\left\{ b = -\frac{n \left(\sum_{i=0}^{n} e^{x_i} y_i \right) + \left(\sum_{i=0}^{n} e^{x_i} y_i \right) - \left(\sum_{i=0}^{n} y_i \right) \left(\sum_{i=0}^{n} e^{x_i} \right)}{\left(\sum_{i=0}^{n} e^{x_i} \right)^2 - \left(\sum_{i=0}^{n} (e^{x_i})^2 \right) n - \left(\sum_{i=0}^{n} (e^{x_i})^2 \right)}, \right.$$

$$\left. a = \frac{-\left(\sum_{i=0}^{n} (e^{x_i})^2 \right) \left(\sum_{i=0}^{n} y_i \right) + \left(\sum_{i=0}^{n} e^{x_i} \right) \left(\sum_{i=0}^{n} e^{x_i} y_i \right)}{\left(\sum_{i=0}^{n} e^{x_i} \right)^2 - \left(\sum_{i=0}^{n} (e^{x_i})^2 \right) n - \left(\sum_{i=0}^{n} (e^{x_i})^2 \right)} \right\}$$

Решение в численном виде — функция fsolve

Для получения численного решения нелинейного уравнения или системы нелинейных уравнений в форме вещественных чисел удобно использовать функцию:

```
fsolve( eqns, vars, options )
```

Эта функция может быть использована со следующими параметрами:

- `complex` — находит один или все корни полинома в комплексной форме;
- `fulldigits` — задает вычисления для полного числа цифр, заданного функцией `Digits`;
- `maxsols=n` — задает нахождение только `n` корней;
- `interval` — задается в виде `a..b` или `x=a..b`, или `{x=a..b, y=c..d, ...}` и обеспечивает поиск корней в указанном интервале.

Функция `fsolve` дает решения сразу в форме вещественных или комплексных чисел, что и показывают следующие примеры:

```
> fsolve(sin(x)=Pi/4,x);
.9033391108
> fsolve(sin(x)=1/2,x=4..8);
```

```
6.806784083
> fsolve(2*x^2+x-1=10,x);
-2.608495283, 2.108495283
> fsolve(x^5-x,x);
-1., 0., 1.000000000
> fsolve(x^5-x,x,complex);
-1.000000000, -1.000000000I, 0., 1.000000000I, 1.000000000

> eqns := abs(x)*x+exp(x) > 0;
```

$$eqns := 0 < |x|\,x + e^{x}$$

```
> solve( eqns, {x} );
```

$$\left\{-2\,\mathrm{LambertW}\left(\frac{1}{2}\right) < x\right\}$$

```
> f := sin(x+y) - exp(x)*y = 0:
g := x^2 - y = 2:
fsolve({f,g},{x,y},{x=-1..1,y=-2..0});
{x = -.6687012050, y = -1.552838698}
```

Заметим, что локализация поиска корней в заданном интервале позволяет отыскивать такие решения, которые не удается получить с помощью функций `solve` и `fsolve` в обычном применении. В последнем из приведенных примеров дается решение системы нелинейных уравнений, представленных уравнениями `f` и `g`.

Чтобы еще раз показать различие между функциями `solve` и `fsolve`, рассмотрим пример решения с их помощью одного и того же уравнения erf(x) = 1/2:

```
> solve(erf(x)=1/2,x);
RootOf(2 erf(_Z) - 1)
> fsolve(erf(x)=1/2);
.4769362762
```

Функция `solve` в этом случае находит нетривиальное решение в комплексной форме через функцию `RootOf`, тогда как функция `fsolve` находит обычное приближенное решение.

Решение рекуррентных уравнений — rsolve

Функция `solve` имеет ряд родственных функций. Одну из таких функций — `fsolve` — мы рассмотрели выше. В справочной системе Maple 7 можно найти ряд и других функций, например `rsolve` для решения рекуррентных уравнений, `isolve` для решения целочисленных уравнений, `msolve` для решения по модулю m и т. д. Здесь мы рассмотрим решение уравнений важного класса — рекуррентных. Напомним, что это такие уравнения, у которых заданный шаг решения находится по одному или нескольким предшествующим шагам.

Для решения рекуррентных уравнений используется функция `rsolve`:

```
rsolve(eqns, fcns)
rsolve(eqns, fcns, 'genfunc'(z))
rsolve(eqns, fcns, 'makeproc')
```

Здесь eqns — одиночное уравнение или система уравнений, fcns — функция, имя функции или множество имен функций, z — имя, генерирующее функциональную переменную.

Ниже представлены примеры применения функции rsolve:

```
> restart;
> rsolve(f(n)=-2*f(n-1)-f(n-2),f(k));
```

$$(-f(0)-f(1))(k+1)(-1)^k+(f(1)+2f(0))(-1)^k$$

```
> rsolve({f(n)=-3*f(n-1)-2*f(n-2),f(1..2)=1},{f});
```

$$\{f(n)=-3(-1)^n+(-2)^n\}$$

```
> rsolve({y(n)=n*y(n-1),y(0)=1},y);
```

$$\Gamma(n+1)$$

```
> rsolve({y(n)*y(n-1)+y(n)-y(n-1)=0,y(0)=a},y);
```

$$\frac{a}{1+n\,a}$$

```
> rsolve({F(n)=F(n-1)+F(n-2),F(1..2)=1},F,'genfunc'(x));
```

$$-\frac{x}{-1+x+x^2}$$

```
> rsolve({y(n+1)+f(n)=2*2^n+n,f(n+1)-y(n)=n-2^n+3,y(k=1..5)=2^k-1,f(5)=6},{y,f});
```

$$\{f(n)=n+1,\ y(n)=2^n-1\}$$

А теперь приведем результат вычисления функцией rsolve n-го числа Фибоначчи. Оно задается следующим выражением:

```
> eq1 := {f(n+2) = f(n+1) + f(n) , f(0) = 1 , f(1) = 1};
```

$$eq1 := \{f(n+2)=f(n+1)+f(n),\ f(0)=1,\ f(1)=1\}$$

В нем задана рекуррентная формула для числа Фибоначчи — каждое новое число равно сумме двух предыдущих чисел, причем нулевое и первое числа равны 1. С помощью функции rsolve можно получить поистине ошеломляющий результат:

```
> a1:=rsolve(eq1,f);
```

$$a1 := \frac{2}{5}\frac{\sqrt{5}\left(2\dfrac{1}{-1+\sqrt{5}}\right)^n}{-1+\sqrt{5}}+\frac{\dfrac{2}{5}\sqrt{5}\left(-2\dfrac{1}{1+\sqrt{5}}\right)^n}{1+\sqrt{5}}$$

Числа Фибоначчи — целые числа. Поэтому представленный результат выглядит как весьма сомнительный. Но на самом деле он точный и с его помощью можно получить числа Фибоначчи. Ниже показан процесс получения чисел Фибоначчи для n = 5, 7, 10 и 20:

```
> [normal(subs(n=5,a1),expanded),normal(subs(n=7,a1),expanded),
                    normal(subs(n=10,a1),expanded),normal(subs(n=20,a1),expanded)];
```

[8, 21, 89, 10946]

Решение уравнений в целочисленном виде — isolve

Иногда бывает нужен результат в форме только целых чисел. Для этого используется функция `isolve(eqns, vars)`, дающая решение в виде целых чисел. Приведем примеры ее применения:

```
> isolve({2*x-5=3*y});
```

$$\{x = 4 + 3\ \_Z1, y = 1 + 2\ \_Z1\}$$

```
> isolve(y^4-z^2*y^2-3*x*z*y^2-x^3*z);
```

Во втором из приведенных примеров в выводе появилась вспомогательная переменная `%1`, которая упрощает запись результата при текстовом формате его вывода (Character Notation). Напоминаем, что в стандартной математической нотации вспомогательная переменная вида `%N` не формируется. В этом случае упомянутый пример будет выглядеть следующим образом:

```
> isolve(y^4-z^2*y^2-3*x*z*y^2-x^3*z);
```

$$\{z = \frac{\_Z3\ \_Z2^4}{\mathrm{igcd}(-\_Z1^2\,(\_Z2^2 - \_Z1^2), -\_Z1^3\ \_Z2, \_Z2^4)},$$

$$x = -\frac{\_Z3\ \_Z1^2\,(\_Z2^2 - \_Z1^2)}{\mathrm{igcd}(-\_Z1^2\,(\_Z2^2 - \_Z1^2), -\_Z1^3\ \_Z2, \_Z2^4)},$$

$$y = -\frac{\_Z3\ \_Z1^3\ \_Z2}{\mathrm{igcd}(-\_Z1^2\,(\_Z2^2 - \_Z1^2), -\_Z1^3\ \_Z2, \_Z2^4)}\}$$

Результат вычислений одинаков при любом формате вывода, но иногда вывод в текстовом формате с выделением вспомогательных переменных имеет преимущество, поскольку выглядит более компактным.

Функция msolve

Функция `msolve(eqns,vars,m)` или `msolve(eqns,m)` обеспечивает решение вида $Z \bmod m$ (то есть при подстановке решения левая часть при делении на m дает остаток, равный правой части уравнения). При отсутствии решения возвращается объект `NULL` (пустой список).

Ниже даны примеры использования функции `msolve`:

```
> msolve({3*x-4*y=1,7*x+y=2},12);
```

$$\{y = 5, x = 3\}$$

```
> msolve(2^i=3,19);
```

$$\{i = 13 + 18\ \_Z1\sim\}$$

```
> msolve(8^j=2,x,17);
```

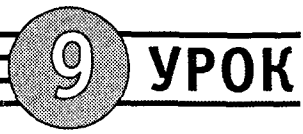
$$\{j = 3 + 8x\}$$

На этом мы завершаем рассмотрение функций для решения уравнений, неравенств и систем с ними.

Что нового мы узнали?

В этом уроке мы научились:

- Вычислять суммы членов последовательностей.
- Вычислять произведения членов последовательностей.
- Вычислять производные.
- Вычислять интегралы.
- Разлагать функции в ряды.
- Решать уравнения и неравенства.



Анализ функций и полиномов

- ❒ Анализ функций
- ❒ Поиск минимумов и максимумов аналитических функций
- ❒ Анализ функций на непрерывность
- ❒ Нахождение сингулярных точек функций
- ❒ Вычисление асимптотических и иных разложений
- ❒ Пример анализа сложной функции
- ❒ Операции с полиномами
- ❒ Интерполяция и аппроксимация функциональных зависимостей
- ❒ Прямое и обратное Z-преобразования

Анализ функций

Поиск экстремумов функций

Важным разделом математики является исследование аналитических функций. Оно обычно заключается в определении координат особых точек функции и ее значений в этих точках, а также в выяснении особенностей функции, таких как наличие точек разрыва, асимптот, точек перегибов, разрывов и т. д. К сожалению, пока нет средств, сразу выявляющих все особенности функций, поскольку даже средства, решающие частные задачи анализа функций, довольно сложны и специфичны. Достаточно отметить проблему поиска экстремумов функций (особенно функций нескольких переменных). Поэтому функции приходится анализировать индивидуально.

С помощью функции `fsolve` легко находятся значения независимой переменной x функций вида $f(x)$, при которых $f(x) = 0$ (корни этого уравнения). При этом данная функция позволяет (в отличие от функции `solve`) изолировать корни функции $f(x)$ указанием примерного интервала их существования. Ряд функций служит для вычисления экстремумов, максимумов и минимумов функций, а также для определения их непрерывности. Одна из таких функций, `extrema`, позволяет найти экстремумы выражения `expr` (как максимумы, так и минимумы) при ограничениях `constrs` и переменных `vars`, по которым ищется экстремум:

```
extrema(expr, constrs)
extrema(expr, constrs, vars)
extrema(expr, constrs, vars, 's')
```

Ограничения `contrs` и переменные `vars` могут задаваться одиночными объектами или списками ряда ограничений и переменных. Найденные координаты точки экстремума присваиваются переменной `'s'`. При отсутствии ограничений в виде равенств или неравенств вместо них записывается пустой список {}.

Эта функция в предшествующих версиях Maple находилась в стандартной библиотеке и вызывалась командой `readlib(extrema)`. Но в Maple 7 ее можно использовать без предварительного объявления. В этом убеждают приведенные ниже примеры:

```
> extrema(a*x^2+b*x+c,{},x,'s');s;
```

$$\left\{-\frac{1}{4}\frac{b^2-4\,c\,a}{a}\right\}$$

$$\left\{\left\{x=-\frac{1}{2}\frac{b}{a}\right\}\right\}$$

```
> extrema(x*exp(-x),{},x,'s');s;
```

$$\left\{e^{(-1)}\right\}$$

$$\{\{x=1\}\}$$

```
> extrema(sin(x)^2,{},x,'s');s;
```

$$\{0, 1\}$$

$$\{\{x = 0\}, \{x = \frac{1}{2}\pi\}\}$$

```
> extrema(x+y/z,x^2+y^2+z^2=1,{x,y,z},'s');s;
```

$$\{\max(1 - \text{RootOf}(\_Z^4 + 1)^2, -1 + \text{RootOf}(\_Z^4 + 1)^2),$$
$$\min(1 - \text{RootOf}(\_Z^4 + 1)^2, -1 + \text{RootOf}(\_Z^4 + 1)^2)\}$$

$$\{\{z = \text{RootOf}(\_Z^4 + 1), x = -1, y = \text{RootOf}(\_Z^4 + 1)^3\},$$
$$\{x = 1, z = \text{RootOf}(\_Z^4 + 1), y = -\text{RootOf}(\_Z^4 + 1)^3\}\}$$

Как видно из приведенных примеров, функция `extrema` возвращает как значения экстремумов, так и значения аргументов, при которых экстремумы наблюдаются.

Для проверки оптимизационных алгоритмов существует ряд тестовых функций. Одна из таких функций — функция двух переменных Розенброка. В представленном ниже примере она задана как `rf(x,y)`:

```
> rf:=(x,y)->100*(y-x^2)^2+(1-x)^2;
```

$$rf := (x, y) \rightarrow 100\,(y - x^2)^2 + (1 - x)^2$$

```
> extrema(rf(x,y),{x,y},'s');s;
s
```

Как нетрудно заметить, минимум этой функции при значениях $x = y = 1$, равный 0, функцией `extrema` не обнаружен. Однако это не недостаток данной функции, а просто неудачное ее применение. Функция Розенброка имеет минимум значения, и для его обнаружения надо использовать функцию `minimize`, описанную ниже.

ПРИМЕЧАНИЕ Функция extrema дает неплохие результаты при поиске экстремумов простых аналитических функций, не имеющих особенностей. Однако при анализе сложных функций, содержащих функции со сравнением аргумента (например, abs(x), signum(x) и др.), функция extrema часто отказывается работать и просто повторяет запись обращения к ней.

Поиск минимумов и максимумов аналитических функций

Часто нужно найти минимум или максимум заданной функции. Для поиска минимумов и максимумов выражений (функций) `expr` служат функции стандартной библиотеки:

```
minimize(expr, opt1, opt2, ..., optn)
maximize(expr, opt1, opt2, ..., optn)
```

Эти функции могут разыскивать максимумы и минимумы для функций как одной, так и нескольких переменных. С помощью опций `opt1, opt2, ..., optn` можно указывать

дополнительные данные для поиска. Например, параметр 'infinity' означает, что поиск минимума или максимума выполняется по всей числовой оси, а параметр location (или location=true) дает расширенный вывод результатов поиска — выдается не только значение минимума (или максимума), но и значения переменных в этой точке.

Примеры применения функции minimize приведены ниже:

```
> minimize(x^2-3*x+y^2+3*y+3);
```

$$\frac{-3}{2}$$

```
> minimize(x^2-3*x+y^2+3*y+3, location);
```

$$\frac{-3}{2}, \left\{\left[\{y=\frac{-3}{2}, x=\frac{3}{2}\}, \frac{-3}{2}\right]\right\}$$

```
> minimize(x^2-3*x+y^2+3*y+3, x=2..4, y=-4..-2, location);
```

$$-1, \{[\{x=2, y=-2\}, -1]\}$$

```
> minimize(x^2+y^2,x=-10..10,y=-10..10);
```

$$0$$

```
> minimize(x^2+y^2,x=-10..10,y=-10..10,location);
```

$$0, \{[\{y=0, x=0\}, 0]\}$$

```
> minimize(abs(x*exp(-x^2)-1/2), x=-4..4);
```

$$\frac{1}{2}-\frac{1}{2}\sqrt{2}\,e^{(-1/2)}$$

```
> minimize(abs(x*exp(-x^2)-1/2), x=-4..4,location=true);
```

$$\frac{1}{2}-\frac{1}{2}\sqrt{2}\,e^{(-1/2)}, \left\{\left[x=\frac{1}{2}\sqrt{2}, \frac{1}{2}-\frac{1}{2}\sqrt{2}\,e^{(-1/2)}\right]\right\}$$

Приведем подобные примеры и для функции поиска максимума — maximize:

```
> maximize(x*exp(-x));
```

$$e^{(-1)}$$

```
> maximize(x*exp(-x),location);
```

$$e^{(-1)}, \{[\{x=1\}, e^{(-1)}]\}$$

```
> maximize(sin(x)/x,x=-2..2,location);
```

$$\infty, \{[\{x=0\}, \infty]\}$$

```
> maximize(exp(-x)*sin(y),x=-10..10,y=-10..10,location);
```

$$e^{10}, \left\{\left[\{y=-\frac{3}{2}\pi, x=-10\}, e^{10}\right], \left[\{x=-10, y=\frac{5}{2}\pi\}, e^{10}\right], \left[\{y=\frac{1}{2}\pi, x=-10\}, e^{10}\right]\right\}$$

Обратите внимание на то, что в предпоследнем примере Maple 7 явно «оскандалилась» и вместо максимума функции $\sin(x)/x$, равного 1 при $x=0$, выдал результат в виде бесконечности. Другими словами, система обнаружила, что в данном случае ей незнакомо понятие предела $\sin(x)/x$ при $x\to 0$. Эта ситуация кажется более чем странной, если учесть, что в этом примере Maple 6 давал правильный результат.

Применим функцию minimize для поиска минимума функции Розенброка. Рисунок 9.1 показывает, что minimize прекрасно справляется с данной задачей. На рис. 9.1 представлено также построение функции Розенброка, хорошо иллюстрирующее ее особенности.

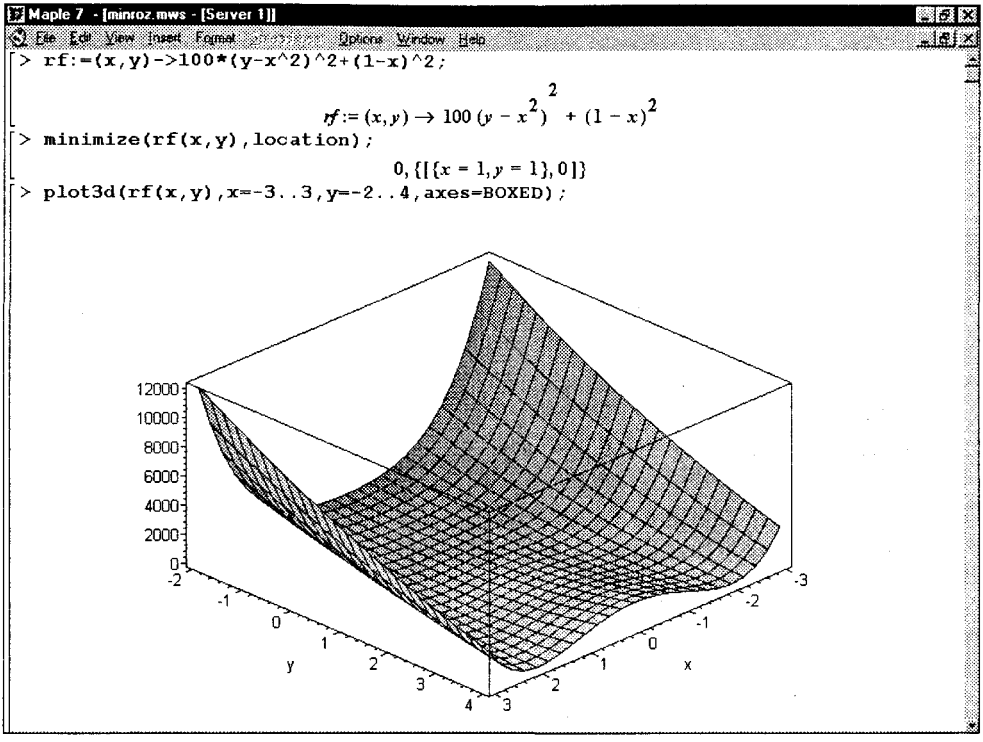


Рис. 9.1. Поиск минимума функции Розенброка и построение ее графика

Трудность поиска минимума функции Розенброка связана с ее характерными особенностями. Из рис. 9.1 видно, что эта функция представляет собой поверхность типа «глубокого оврага с почти плоским дном», в котором и расположена точка минимума. Такая особенность этой функции существенно затрудняет поиск минимума. То, что система Maple 7 справляется с данной тестовой функцией, вовсе не означает, что трудности в поиске минимума или максимума других функций остаются позади.

Анализ функций на непрерывность

Для исследования функций на непрерывность Maple 7 имеет функцию iscont, записываемую в ряде форм:

```
iscont(expr, x = a .. b)
iscont(expr, x = a .. b, 'closed')
iscont(expr, x = a .. b, 'open')
```

Она позволяет исследовать выражение expr, заданное в виде зависимости от переменной x, на непрерывность. Если выражение непрерывно, возвращается логическое значение true, иначе — false. Возможен также результат типа FAIL. Параметр 'closed' показывает, что конечные точки должны также проверяться, а указанный по умолчанию параметр 'open' — что они не должны проверяться.

Работу функции iscont иллюстрируют следующие примеры:

```
> iscont(1/x^2,x=-1..1);
```

false

```
> iscont(1/x^2,x=-1..1,'closed');
```

false

```
> iscont(1/x,x=0..1);
```

true

```
> iscont(1/x,x=0..1,'closed');
```

false

```
> iscont(1/(x+a),x=-1..1);
```

FAIL

Рекомендуется внимательно присмотреться к результатам этих примеров и опробовать свои собственные примеры.

Определение точек нарушения непрерывности

Функции, не имеющие непрерывности, доставляют много хлопот. Поэтому важным представляется анализ функций на непрерывность. В Maple 7 функция discont(f,x) позволяет определить точки, в которых нарушается непрерывность функции $f(x)$. Она вычисляет все точки в пределах изменения x от $-?$ до $+?$. Результаты вычислений могут содержать особые *экстрапеременные* с именами вида _Zn~ и _NNn~. В частности, они позволяют оценить периодические нарушения непрерывности функций.

Примеры применения функции discont приведены ниже:

```
> discont(1/(x-2),x);
```

$\{2\}$

```
> discont(1/((x-1)*(x-2)*(x-3)),x);
```

$\{1, 2, 3\}$

```
> discont(GAMMA(x/2),x);
```

$\{-2\ \_NNI\sim\}$

Весьма рекомендуется наряду с применением данной функции просмотреть график анализируемой функции.

ПРИМЕЧАНИЕ **В ряде примеров в выводе используются специальные переменные вида *_NameN~*, где *Name* — имя переменной и *N* — ее текущий номер. После выполнения команды restart отсчет *N* начинается с 1. Если вывод с такими переменными уже применялся, то их текущие номера могут казаться произвольными. Специальные переменные часто используются для упрощения выводимых выражений.**

Нахождение сингулярных точек функции

Многие операции, такие как интегрирование и дифференцирование, чувствительны к особенностям функций, в частности к их разрывам и особым точкам. Функция `singular(expr, vars)` позволяет найти особые (сингулярные) точки выражения `expr`, в которых она испытывает разрывы. Дополнительно в числе параметров может указываться необязательный список переменных.

Примеры применения этой функции приведены ниже:

```
> singular(ln(x)/(x^2-a));
```

$$\{a = a, x = 0\}, \{a = x^2, x = x\}$$

```
> singular(tan(x));
```

$$\{x = \_Z22\sim \pi + \frac{1}{2}\pi\}$$

```
> singular(1/sin(x));
```

$$\{x = \pi \_Z23\sim\}$$

```
> singular(Psi(x*y),{x,y});
```

$$\{y = y, x = -\frac{\_N1\sim - 1}{y}\}$$

```
> singular(x+y+1/x,{x,y});
```

$$\{y = y, x = 0\}, \{y = y, x = -\infty\}, \{y = \infty, x = x\}, \{y = -\infty, x = x\}, \{x = \infty, y = y\}$$

Вычисление асимптотических и иных разложений

Важным достоинством системы Maple является наличие в ней ряда функций, позволяющих выполнять детальный анализ функций. К такому анализу относится вычисление асимптотических разложений функций, которые представляются в виде рядов (не обязательно с целыми показателями степени). Для этого используется следующая функция:

```
asympt(f,x)
asympt(f,x,n).
```

Здесь `f` — функция переменной `x` или алгебраическое выражение; `x` — имя переменной, по которой производится разложение; `n` — положительное целое число (порядок разложения, по умолчанию равный 6). Ниже представлены примеры применения этой функции:

```
> asympt(x/(1-x^2),x);
```

$$-\frac{1}{x} - \frac{1}{x^3} - \frac{1}{x^5} + O\left(\frac{1}{x^7}\right)$$

```
> asympt(n!,n,3);
```

$$\frac{\frac{\sqrt{2}\sqrt{\pi}}{\sqrt{\frac{1}{n}}} + \frac{1}{12}\sqrt{2}\sqrt{\pi}\sqrt{\frac{1}{n}} + \frac{1}{288}\sqrt{2}\sqrt{\pi}\left(\frac{1}{n}\right)^{(3/2)} + O\left(\left(\frac{1}{n}\right)^{(5/2)}\right)}{\left(\frac{1}{n}\right)^{n} e^{n}}$$

```
> asympt(exp(x^2)*(1-exp(x)),x);
```

$$-e^{(x^2)}e^{x}+e^{(x^2)}$$

```
> asympt(sqrt(Pi/2)*BesselJ(0,x),x,3);
```

$$\sin\left(x+\frac{1}{4}\pi\right)\sqrt{\frac{1}{x}}-\frac{1}{8}\cos\left(x+\frac{1}{4}\pi\right)\left(\frac{1}{x}\right)^{(3/2)}-\frac{9}{128}\sin\left(x+\frac{1}{4}\pi\right)\left(\frac{1}{x}\right)^{(5/2)}+O\left(\left(\frac{1}{x}\right)^{(7/2)}\right)$$

Пример анализа сложной функции

Ниже мы рассмотрим типичный анализ достаточно «сложной» функции, имеющей в интересующем нас интервале изменения аргумента x от −4 до 4, нули, максимумы и минимумы. Определение функции $f(x)$, ее графики и график производной $dF(x)/dx$ даны на рис. 9.2. Этот рисунок является началом полного документа, описываемого далее.

Функция $F(x)$ на первый взгляд имеет не совсем обычное поведение вблизи начала координат (точки с $x = y = 0$). Для выяснения такого поведения разумно построить график функции при малых x и y. Он также представлен на рис. 9.2 (нижний график) и наглядно показывает, что экстремум вблизи точки (0, 0) является обычным минимумом, немного смещенным вниз и влево от начала координат.

Теперь перейдем к анализу функции $F(x)$. Для поиска нулей функции (точек пересечения оси x) удобно использовать функцию `fsolve`, поскольку она позволяет задавать область изменения x, внутри которой находится корень. Как видно из приведенных ниже примеров, анализ корней $F(x)$ не вызвал никаких трудностей, и все корни были уточнены сразу:

```
Поиск нулей функции
> fsolve(F(x),x,-2...-1);
-1.462069476
> fsolve(F(x),x,-.01..0.01);
0.
> fsolve(F(x),x,-.05..0);
-.02566109292
> fsolve(F(x),x,1..2);
1.710986355
> fsolve(F(x),x,2.5..3);
2.714104921
```

Нетрудно заметить, что функция имеет два очень близких (но различных) корня при x, близких к нулю.

Анализ функции на непрерывность, наличие ее нарушений и сингулярных точек реализуется следующим образом:

```
Анализ функции на непрерывность, наличие ее нарушений и наличие сингулярных точек
> iscont(F(x),x=-4..4);
true
> discont(F(x),x);
{ }
> singular(F(x));
{x = ∞}, {x = -∞}
```

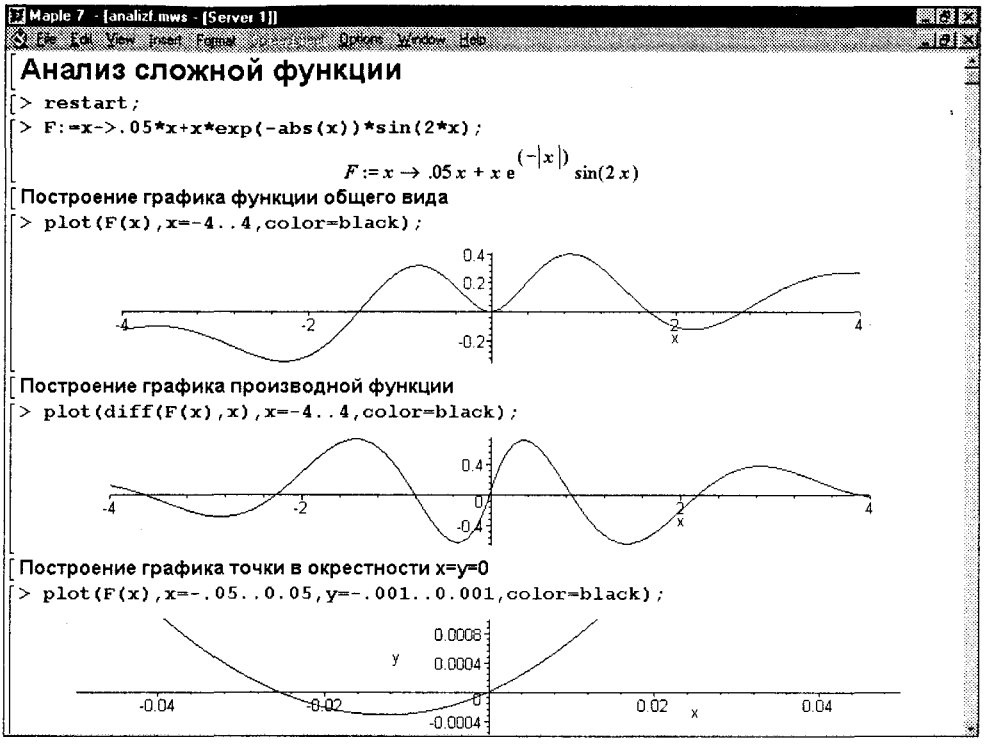


Рис. 9.2. Задание функции $F(x)$ и построение графиков функции и ее производной

Этот анализ не выявляет у заданной функции каких-либо особенностей. Однако это не является поводом для благодушия — попытка найти экстремумы $F(x)$ с помощью функции `extrema` и минимумы с помощью функции `minimize` завершаются полным крахом:

```
Неудачный поиск экстремумов и минимумов функции
> extrema(F(x),{},x,'s');s;
```

s

```
> minimize(F(x),x=-.1...1);
```

$$\text{minimize}\,(.05x + x\,e^{(-|x|)}\sin(2x), x = -.1 \,..\, 1)$$

```
> minimize(F(x),x=-2.5..-2);S
```

$$\text{minimize}\,(.05x + x\,e^{(-|x|)}\sin(2x), x = -2.5 \,..\, -2)$$

Приходится признать, что в данном случае система Maple 7 ведет себя далеко не самым лучшим образом. Чтобы довести анализ $F(x)$ до конца, придется вспомнить, что у функции без особенностей максимумы и минимумы наблюдаются в точках, где производная меняет знак и проходит через нулевое значение. Таким образом, мы можем найти минимумы и максимумы по критерию равенства производной нулю. В данном случае это приводит к успеху:

```
Поиск минимумов по критерию равенства нулю производной
> fsolve(diff(F(x),x)=0,x,-.5...5);
```

-.01274428224

```
> xm:=%;
```

$xm := -.0003165288799$

```
> [F(xm),F(xm+.001),F(xm-.001)];
```

[-.00001562612637, .00003510718293, -.00006236451216]

```
> fsolve(diff(F(x),x)=0,x,-2.5..-2);
```

-2.271212360

```
> fsolve(diff(F(x),x)=0,x,2..2.5);
```

2.175344371

```
Неудачный поиск максимума
> maximize(F(x),x=-1..-.5);
```

$\text{maximize}(.05\,x + x\,e^{(-|x|)}\sin(2\,x), x = -1 \,..\, -.5)$

```
Поиск максимумов по критерию равенства нулю производной
> fsolve(diff(F(x),x),x,-1..-.5);
```

-.8094838517

```
> fsolve(diff(F(x),x),x,.5..2);
```

.8602002115

```
> fsolve(diff(F(x),x),x,-4..-3);
```

-3.629879137

```
> fsolve(diff(F(x),x),x,3..4);
```

3.899664536

Итак, все основные особые точки данной функции (нули, минимумы и максимумы) найдены, хотя и не без трудностей и не всегда с применением специально предназначенных для такого поиска функций. В уроке 12 будет описана процедура, которая автоматизирует процесс анализа не очень сложных функций и обеспечивает его наглядную визуализацию.

Функции из отдельных кусков

Создание функций из отдельных кусков

Для создания функций, составленных из отдельных кусков, Maple 7 располагает интересной функцией:

```
piecewise(cond_1,f_1, cond_2,f_2, ..., cond_n,f_n, f_otherwise)
```

где `f_i` — выражение, `cond_i` — логическое выражение, `f_otherwise` — необязательное дополнительное выражение. В зависимости от того или иного условия эта функция позволяет формировать соответствующую аналитическую зависимость. К кусочным функциям (подчас в скрытой форме) приводят функции с элементами сравнения аргумента, например **abs**, **signum**, **max** и др. Поэтому в Maple 7 введен достаточно мощный аппарат обработки и преобразований таких функций по частям.

Простые примеры применения функции piecewise

Рисунок 9.3 показывает задание функции $f(x)$, содержащей три характерных участка. По определенной через функцию пользователя зависимости $f(x)$ можно, как обычно, построить ее график.

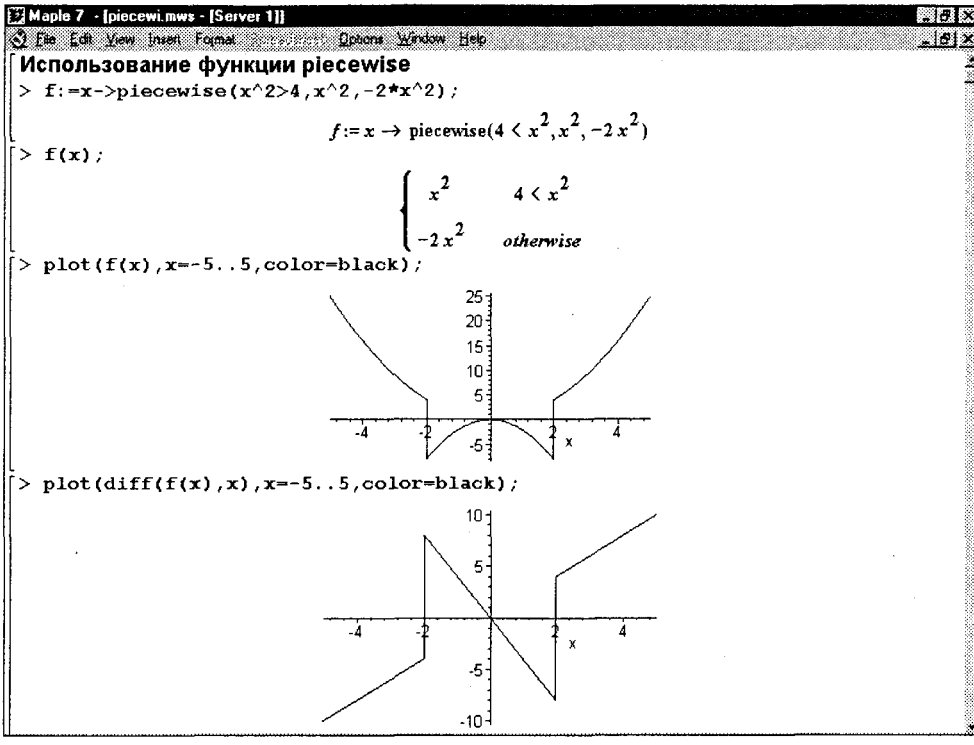


Рис. 9.3. Пример задания и применения функции, составленной из отдельных кусков

Важно отметить, что созданная с помощью функции piecewise зависимость может участвовать в различных преобразованиях. Например, на рис. 9.3 показано, что она легко дифференцируется и можно построить график производной этой функции. При этом каждая часть функции обрабатывается отдельно.

Работа с функциями piecewise

С функциями типа piecewise можно работать, как с обычными функциями. При этом необходимые операции и преобразования осуществляются для каждой из частей функции и возвращаются в наглядной форме.

Ниже приведен пример задания функции f в аналитической форме:

```
> restart;
> f := max(x^2 - 2, x-1);
```

$$f := \max(x^2 - 2, x - 1)$$

Для выявления характера функции воспользуемся функцией convert и создадим объект g в виде кусочной функции:

```
> g := convert(f, piecewise);
```

$$g := \begin{cases} x^2 - 2 & x \le \frac{1}{2} - \frac{1}{2}\sqrt{5} \\ x - 1 & x \le \frac{1}{2} + \frac{1}{2}\sqrt{5} \\ x^2 - 2 & \frac{1}{2} + \frac{1}{2}\sqrt{5} < x \end{cases}$$

Выполним дифференцирование и интегрирование функции:

```
> fprime := diff(f, x);
```

$$fprime := \begin{cases} 2x & x < \frac{1}{2} - \frac{1}{2}\sqrt{5} \\ undefined & x = \frac{1}{2} - \frac{1}{2}\sqrt{5} \\ 1 & x < \frac{1}{2} + \frac{1}{2}\sqrt{5} \\ undefined & x = \frac{1}{2} + \frac{1}{2}\sqrt{5} \\ 2x & \frac{1}{2} + \frac{1}{2}\sqrt{5} < x \end{cases}$$

```
> Int(g,x)=int(g,x);
```

$$\int \begin{cases} x^2 - 2 & x \le \frac{1}{2} - \frac{1}{2}\sqrt{5} \\ x - 1 & x \le \frac{1}{2} + \frac{1}{2}\sqrt{5} \\ x^2 - 2 & \frac{1}{2} + \frac{1}{2}\sqrt{5} < x \end{cases} dx = \begin{cases} \frac{1}{3}x^3 - 2x & x \le \frac{1}{2} - \frac{1}{2}\sqrt{5} \\ -x + \frac{1}{2}x^2 - \frac{7}{12} + \frac{5}{12}\sqrt{5} & x \le \frac{1}{2} + \frac{1}{2}\sqrt{5} \\ \frac{1}{3}x^3 - 2x + \frac{5}{6}\sqrt{5} & \frac{1}{2} + \frac{1}{2}\sqrt{5} < x \end{cases}$$

Как нетрудно заметить, результаты получены также в виде кусочных функций. Можно продолжить работу с функцией f и выполнить ее разложение в степенной ряд:

```
> series(f, x);
```

$$-1 + x + O(x^6)$$

Чтобы убрать член с остаточной погрешностью, можно выполнить эту операцию следующим образом:

```
> series(g, x);
```

$$-1 + x$$

Обратите внимание на то, что поскольку разложение в ряд ищется (по умолчанию) в окрестности точки x=0, то при этом используется тот кусок функции,

в котором расположена эта точка. Читатель может продолжить работу с кусочными функциями и далее.

Операции с полиномами

Определение полиномов

К числу наиболее известных и изученных аналитических функций относятся степенные многочлены — полиномы. Графики полиномов описывают огромное разнообразие кривых на плоскости. Кроме того, возможны рациональные полиномиальные выражения в виде отношения полиномов. Таким образом, круг объектов, которые могут быть представлены полиномами, достаточно обширен, и полиномиальные преобразования широко используются на практике, в частности, для приближенного представления других функций.

Под полиномом в системе Maple 7 понимается сумма выражений с целыми степенями. Многочлен для ряда переменных — *многомерный* полином. К одномерным полиномам относятся степенной многочлен:

$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots a_1 x + a_0$,

а также отдельная переменная x и константа. Большое достоинство полиномов состоит в том, что они дают единообразное представление многих зависимостей и для своего вычисления требуют только арифметических операций (их число значительно сокращается при использовании хорошо известной схемы Горнера). Производные от полиномов и интегралы с подынтегральными функциями-полиномами легко вычисляются и имеют простой вид. Есть и достаточно простые алгоритмы для вычисления всех (в том числе комплексных) корней полиномов на заданном промежутке.

Выделение коэффициентов полиномов

Для выделения коэффициентов полиномов в Maple 7 служат следующие функции:

- `coeff(p, x)` — возвращает коэффициент при `x` полинома `p`;
- `coeff(p,x,n)` — возвращает коэффициент для члена со степенью `n` полинома `p`;
- `coeff(p,x^n)` — возвращает коэффициенты при `x^n` полинома `p`;
- `coeffs(p, x, 't')` — возвращает коэффициенты полинома нескольких переменных, относящиеся к переменной `x` (или списку переменных) с опцией `'t'`, задающей имя переменной;
- `collect(p,x)` — возвращает полином, объединяя коэффициенты при степенях переменной `x`.

Ниже даны примеры применения этих функций:

```
> p:=a4*x^4+a3*x^3+a2*x^2+a1*x+a0;
```

$p := a4\,x^4 + a3\,x^3 + a2\,x^2 + a1\,x + a0$

```
> coeff(p,x);
```

```
> coeff(p,x^3);
```
$a1$
```
> coeff(p,x,4);
```
$a3$
```
> coeffs(p,x);
```
$a4$
```
> q:=x^2+2*y^2+3*x+4*y+5;
```
$a0, a4, a1, a3, a2$
```
> coeffs(q);
```
$q := x^2 + 2y^2 + 3x + 4y + 5$
```
> coeffs(q,y);
```
$5, 2, 3, 4, 1$
```
> coeffs(q,x,y);
```
$x^2 + 3x + 5, 2, 4$
```
> collect(q,x);
```
$5 + 2y^2 + 4y, 3, 1$
```
> collect(q,x,y);
```
$x^2 + 2(1, x^2, x)^2 + 3x + (4, 4x^2, 4x) + 5$
$y(1)x^2 + y(3)x + y(5 + 2y^2 + 4y)$

ПРИМЕЧАНИЕ Следует обратить внимание на то, что при выполнении операции collect в прежних версиях Maple довольно часто возникала фатальная ошибка. Как видно из приведенных примеров, в Maple 7 такой ошибки уже не возникает.

Оценка коэффициентов полинома по степеням

Полином может быть неполным, то есть не содержать членов со степенями ниже некоторой. Функция `lcoeff` возвращает старший, а функция `tcoeff` — младший коэффициент полинома нескольких переменных. Эти функции задаются в виде:

```
lcoeff(p)          tcoeff(p)
lcoeff(p, x)       tcoeff(p, x)
lcoeff(p, x, 't')  tcoeff(p, x, 't')
```

Функции `lcoeff` и `tcoeff` возвращают старший (младший) коэффициент полинома `p` относительно переменной `x` или ряда переменных при многомерном полиноме. Если `x` не определено, `lcoeff (tcoeff)` вычисляет старший (младший) коэффициент относительно всех переменных полинома `p`. Если третий аргумент `t` определен, то это имя назначено старшему (младшему) члену `p`. Если `x` — единственное неизвестное и `d` — степень `p` по `x`, то `lcoeff(p, x)` эквивалентно `coeff(p, x, d)`. Если `x` — список или множество неизвестных, `lcoeff (tcoeff)` вычисляет старший (младший) коэффициент `p`, причем `p` рассматривается как полином многих переменных. Имейте в виду, что `p` должен быть разложен по степеням неизвестного `x` до вызова функций `lcoeff` или `tcoeff`.

Приведем примеры применения функций `lcoeff`, `tcoeff` и `coeffs`:

```
> q:=1/x^2+2/x+3+4*x+5*x^2;
```

$$q := \frac{1}{x^2} + \frac{2}{x} + 3 + 4x + 5x^2$$

```
> lcoeff(q,x);
```

5

```
> lcoeff(q,x,'t');
```

5

```
> t;
```

x^2

```
> coeffs(q,x,'t');
```

3, 1, 4, 2, 5

```
> t;
```

$1, \frac{1}{x^2}, x, \frac{1}{x}, x^2$

Оценка степеней полинома

Функция degree возвращает высшую степень полинома, а ldegree — низшую степень. Эти функции задаются следующим образом:

```
degree(a,x)                    ldegree(a,x)
```

Функции degree и ldegree используются, чтобы определить высшую и низшую степени полинома от неизвестного (неизвестных) x, которое чаще всего является единственным, но может быть списком или множеством неизвестных. Полином может иметь отрицательные целые показатели степеней при x. Таким образом, degree и ldegree могут возвратить отрицательное или положительное целое число. Если выражение не является полиномом от x данным параметром, то возвращается FAIL.

Чтобы degree и ldegree возвратили точный результат, полином обязательно должен быть сгруппирован по степеням x. Например, для выражения $(x + 1)(x + 2) - x^2$ функция degree не обнаружит аннулирование старшего члена и неправильно возвратит результат 2. Во избежание этой проблемы перед вызовом degree следует применять к полиному функции collect или expand. Если x — множество неизвестных, degree/ldegree вычисляет полную степень. Если x — список неизвестных, degree/ldegree вычисляет векторную степень. Векторная степень определяется следующим образом:

```
degree(p,[]) = 0
degree(p,[x1,x2,…]) =degree(p,x1) degree(lcoeff(p,x1),[x2,…])
```

Полная степень тогда определяется следующим образом:

```
degree(p,{x1,…,xn}) = maximum degree(p,{x1,…xn})
```

или

```
degree(p,{x1,…,xn}) = degree(p,[x1,…,xn])
```

Обращаем внимание на то, что векторная степень зависит от порядка перечисления неизвестных, а полная степень не зависит.

Примеры применения функций degree и ldegree:

```
> restart;
> p:=a4*x^4+a3*x^3+a2*x^2;
```

$$p := a4\,x^4 + a3\,x^3 + a2\,x^2$$

```
> degree(p,x);
```

4

```
> ldegree(p,x);
```

2

```
> q:=1/x^2+2/x+3+4*x+5*x^2;
```

$$q := \frac{1}{x^2} + \frac{2}{x} + 3 + 4\,x + 5\,x^2$$

```
> degree(q,x);
```

2

```
> ldegree(q,x);
```

-2

```
> degree(x*sin(x),x);
```

FAIL

```
> zero := y*(x/(x+1)+1/(x+1)-1);
```

$$zero := y\left(\frac{x}{x+1} + \frac{1}{x+1} - 1\right)$$

```
> degree(zero,x);degree(zero,y);
```

FAIL

1

```
degree(collect(zero,x,normal),x);degree(collect(zero,y,
normal),y);
```

$-\infty$

$-\infty$

Разложение полинома на множители

Для контроля того, имеет ли полином несокращаемые множители, может использоваться функция irreduc(p) и ее вариант в инертной форме Ireduc(p,K), где K — RootOf-выражение. Ниже приведены примеры применения этих тестовых функций:

```
>irreduc(x^2-1);
```

false

```
>irreduc(x^2-2);
```

true

```
>Irreduc(2*x^2+6*x+6) mod 7;
```

false

```
>Irreduc(x^4+x+1) mod 2;
```

true

```
>alias(alpha=RootOf(x^4+x+1)):
>Irreduc(x^4+x+1,alpha) mod 2;
```

false

Разложение полинома по степеням

Для разложения полинома p по степеням служат инертные функции AFactor(p) и AFactors(p). Полином может быть представлен в виде зависимости от одной или нескольких переменных.

Функция Afactor(p) выполняет полную факторизацию (разложение) полинома p от нескольких переменных с коэффициентами в виде алгебраических чисел над полем комплексных чисел. При этом справедливо отношение evala(AFactor(p))=factor(p,complex). Таким образом, эта функция является, по существу, избыточной.

В случае одномерного полинома полное разложение на множители является разложением на линейные множители. Функция AFactors аналогична функции Afactor, но создает структуру данных формы [u,[[f[1],e[1]],...,[f[n],e[n]]]] так, что p=u*f[1]^e[1]*...*f[n]^e[n], где каждый f[i] — неприводимый полином.

Ниже даны примеры применения функции Afactor:

```
> evala(AFactor(2*x^2+4*x-6));
```

$$2\,(x+3)\,(x-1)$$

```
> evala(AFactor(x^2+2*y^2));
```

$$(x-\mathrm{RootOf}(\_Z^2+2)\,y)\,(x+\mathrm{RootOf}(\_Z^2+2)\,y)$$

```
> expand((x-1)*(x-2)*(x-3)*(x-4));
```

$$x^4-10\,x^3+35\,x^2-50\,x+24$$

```
> AFactor(%);
```

$$\mathrm{AFactor}(x^4-10\,x^3+35\,x^2-50\,x+24)$$

```
> evala(%);
```

$$(x-1)\,(x-2)\,(x-3)\,(x-4)$$

```
> expand((x-1+I*2)*(x+1-I*2)*(x-3));
```

$$x^3-3\,x^2+3\,x-9+4\,I\,x-12\,I$$

```
> evala(AFactor(%));
```

$$(x-3)\,(x^2+3+4\,I)$$

```
> evala(AFactors(x^2-2*y^2));
```

$$[1,[[x-\mathrm{RootOf}(\_Z^2-2)\,y,1],[x+\mathrm{RootOf}(\_Z^2-2)\,y,1]]]$$

Нетрудно заметить, что разложение полинома на множители позволяет оценить наличие у него корней. Однако для этого удобнее воспользоваться специальными функциями, рассмотренными ниже.

Вычисление корней полинома

Для вычисления действительных и комплексных корней полиномов служит уже известная нам функция solve(p,x), возвращающая список корней полинома p одной переменной. Кроме того, имеются следующие функции для вычисления корней полиномов:

```
roots(p)                    roots(p, K)
roots(p, x)                 roots(p, x, K)
```

Эти функции вычисляют точные корни в рациональной или алгебраической области чисел. Корни возвращаются в виде [[r1,m1], ..., [rn,mn]], где ri — это

корень полинома, а mi — порядковый номер полинома. С действиями этих функций можно разобраться с помощью приведенных ниже примеров:

```
> p:=x^4+9*x^3+31*x^2+59*x+60;
```

$$p := x^4 + 9x^3 + 31x^2 + 59x + 60$$

```
> solve(p,x);
```

$$-3, -4, -1 + 2I, -1 - 2I$$

```
> roots(p,x);
```

$$[[-4, 1], [-3, 1]]$$

```
> roots(x^2-4,x);
```

$$[[2, 1], [-2, 1]]$$

```
> expand((x-1)*(x-2)*(x-3)*(x-4));
```

$$x^4 - 10x^3 + 35x^2 - 50x + 24$$

```
> roots(%,x);
```

$$[[1, 1], [2, 1], [3, 1], [4, 1]]$$

Основные операции с полиномами

С полиномами могут выполняться различные операции. Прежде всего отметим некоторые функции, которые относятся к одному полиному:

- psqrt(p) — возвращает квадрат полинома;
- proot(p,n) — возвращает n-ю степень полинома;
- realroot(p) — возвращает интервал, в котором находятся действительные корни полинома;
- randpoly(vars, eqns) — возвращает случайный полином по переменным vars (список) с максимальной степенью eqns;
- discrim(p,var) — вычисление дискриминанта полинома по переменной var;
- Primitive(a) mod p — проверка полинома на примитивность (возвращает true, если полином примитивен).

Действие этих функций достаточно очевидно, поэтому ограничимся приведением примеров их использования:

```
> readlib(psqrt):
> readlib(proot):
> psqrt(x^2+2*x*y+y^2);
```

$$y + x$$

```
> proot(x^3+3*x^2+3*x+1, 3);
```

$$x + 1$$

```
> psqrt(x+y);
```

$$\_NOSQRT$$

```
> proot(x+y, 2);
```

$$\_NOROOT$$

```
> p:=x^3-3*x^2+5*x-10;
```

$$p := x^3 - 3x^2 + 5x - 10$$

```
> discrim(p,x);
```
$$-1355$$
```
> readlib(realroot):
> realroot(p);
```
$$[[0, 4]]$$
```
> randpoly([x],degree=10);
```
$$63\,x^{10} + 57\,x^8 - 59\,x^5 + 45\,x^4 - 8\,x^3 - 93$$
```
> randpoly([x],degree=10);
```
$$-5\,x^9 + 99\,x^8 - 61\,x^6 - 50\,x^5 - 12\,x^3 - 18\,x$$
```
> randpoly([x],degree=10);
```
$$41\,x^9 - 58\,x^8 - 90\,x^7 + 53\,x^6 - x^4 + 94\,x$$
```
> Primitive( x^4+x+1 ) mod 2;
```
$$true$$

Обратите внимание на то, что для использования некоторых из приведенных функций необходим вызов их из стандартной библиотеки. Для функции randpoly приведенные результаты случайны, так что, скорее всего, их повторение невозможно.

С полиномами можно выполнять обычные операции, используя для этого соответствующие операторы:

```
> readlib(psqrt):
> readlib(proot):
> Primitive( x^4+x+1 ) mod 2;
```
$$true$$
```
> p1:=a1*x^3+b1*x^2+c1*x+d1: p2:=a2*x^2+b2*x+c2:
> p1+p2;
```
$$a1\,x^3 + b1\,x^2 + c1\,x + d1 + a2\,x^2 + b2\,x + c2$$
```
> p1*p2;
```
$$(a1\,x^3 + b1\,x^2 + c1\,x + d1)\,(a2\,x^2 + b2\,x + c2)$$
```
> collect(%,x);
```
$$a1\,a2\,x^5 + (b1\,a2 + a1\,b2)\,x^4 + (c1\,a2 + b1\,b2 + a1\,c2)\,x^3 + (d1\,a2 + c1\,b2 + b1\,c2)\,x^2 + (d1\,b2 + c1\,c2)\,x + d1\,c2$$
```
> p1/p2;
```
$$\frac{a1\,x^3 + b1\,x^2 + c1\,x + d1}{a2\,x^2 + b2\,x + c2}$$
```
> expand(%,x);
```
$$\frac{a1\,x^3}{a2\,x^2 + b2\,x + c2} + \frac{b1\,x^2}{a2\,x^2 + b2\,x + c2} + \frac{c1\,x}{a2\,x^2 + b2\,x + c2} + \frac{d1}{a2\,x^2 + b2\,x + c2}$$

В целом надо отметить, что аппарат действий с полиномами в Maple 7 хорошо развит и позволяет выполнять с ними практически любые математические операции. В частности, можно вычислять производные от полиномов и интегралы, у которых полиномы являются подынтегральными функциями:

```
> diff(p1,x);
```
$$3\,a1\,x^2 + 2\,b1\,x + c1$$
```
> diff(p1,x$2);
```

$$6\,a1\,x+2\,b1$$

```
> Int(p1,x)=int(p1,x);
```

$$\int a1\,x^3+b1\,x^2+c1\,x+d1\,dx=\frac{1}{4}a1\,x^4+\frac{1}{3}b1\,x^3+\frac{1}{2}c1\,x^2+d1\,x$$

```
> Int(p1,x=0..1)=int(p1,x=0..1);
```

$$\int_0^1 a1\,x^3+b1\,x^2+c1\,x+d1\,dx=\frac{1}{4}a1+\frac{1}{3}b1+\frac{1}{2}c1+d1$$

Операции над степенными многочленами с отрицательными степенями

Хотя в подавляющем большинстве случаев используются степенные многочлены (полиномы) с положительными степенями, Maple 7 не накладывает особых ограничений и на многочлены с отрицательными степенями. Например, можно задать такой степенной многочлен:

```
> pp:=a*x^(-2)+b*x^(-1)+c*x+d+e*x^2+f*x^3;
```

$$pp:=\frac{a}{x^2}+\frac{b}{x}+c\,x+d+e\,x^2+f\,x^3$$

Нетрудно показать, что с ним можно выполнять различные операции:

```
> pp+pp;
```

$$2\frac{a}{x^2}+\frac{2\,b}{x}+2\,c\,x+2\,d+2\,e\,x^2+2\,f\,x^3$$

```
> pp-pp;
```

$$0$$

```
> pp^2;
```

$$\left(\frac{a}{x^2}+\frac{b}{x}+c\,x+d+e\,x^2+f\,x^3\right)^2$$

```
> simplify(%);
```

$$\frac{(a+b\,x+c\,x^3+d\,x^2+e\,x^4+f\,x^5)^2}{x^4}$$

```
> Diff(pp,x)=diff(pp,x);
```

$$\frac{\partial}{\partial x}\left(\frac{a}{x^2}+\frac{b}{x}+c\,x+d+e\,x^2+f\,x^3\right)=-2\frac{a}{x^3}-\frac{b}{x^2}+c+2\,e\,x+3\,f\,x^2$$

```
> Int(pp,x);
```

$$\int\frac{a}{x^2}+\frac{b}{x}+c\,x+d+e\,x^2+f\,x^3\,dx$$

```
> int(pp,x);
```

$$-\frac{a}{x}+b\ln(x)+\frac{1}{2}c\,x^2+d\,x+\frac{1}{3}e\,x^3+\frac{1}{4}f\,x^4$$

ПРИМЕЧАНИЕ Maple 7 не накладывает ограничений на применение степенных многочленов (полиномов) с отрицательными степенями. Однако свойства таких полиномов заметно отличаются от свойств полиномов с положительными степенями, поэтому при применении первых надо проявлять известную осторожность.

Интерполяция и аппроксимация функциональных зависимостей

Интерполяция, экстраполяция и аппроксимация

Вычисление многих функций, особенно специальных, требует больших затрат времени. Поэтому до сих пор широко применяются таблицы таких функций. Достаточно отметить знаменитые на весь мир таблицы в книге «Справочник по специальным функциям с формулами, графиками и таблицами» под редакцией М. Абрамовица и И. Стиган [59].

Если некоторая зависимость $y(x)$ представлена рядом табличных отсчетов $y_i(x_i)$, то *интерполяцией* принято называть вычисление значений $y(x)$ при заданном x, расположенном в интервале между отсчетами. За пределами общего интервала определения функции $[a, b]$, то есть при $x < a$ и $x > b$, вычисление $y(x)$ называют *экстраполяцией* (или иногда *предсказанием* значений функции). В данном случае речь идет об одномерной интерполяции, но возможны двумерная интерполяция функций двух переменных $z(x, y)$ и даже многомерная интерполяция для функций многих переменных.

Интерполяция и экстраполяция часто выполняются по некоторой скрытой, но подразумеваемой зависимости. Например, если узловые точки функции соединить отрезками прямых, то будем иметь многоинтервальную линейную интерполяцию данных. Если использовать отрезки параболы, то интерполяция будет параболической. Особое значение имеет многоинтервальная сплайн-интерполяция, области применения которой уже сейчас весьма обширны и непрерывно расширяются. Интерполяция рядом Фурье (набором синусоидальных функций) также достаточно хорошо известна; она эффективна при интерполяции периодических функций.

Аппроксимацией в системах компьютерной математики обычно называют получение приближенных значений какого-либо выражения. Однако под аппроксимацией функций подразумевается получение некоторой конкретной функции, вычисленные значения которой с некоторой точностью аналогичны аппроксимируемой зависимости. Обычно предпочитают найти одну зависимость, приближающую заданный ряд узловых точек. Часто для этого используют степенные многочлены — полиномы.

Здесь мы будем рассматривать такие виды аппроксимации, которые дают точные значения функции $y(x)$ в узловых точках в пределах погрешности вычислений по умолчанию. Если аппроксимирующая зависимость выбирается из условия наименьшей среднеквадратической погрешности в узловых точках (метод наименьших квадратов), то мы имеем *регрессию* или *приближение* функций по методу наименьших квадратов.

Аппроксимация аналитически заданных функций

Если функция задана аналитически, то наиболее простым способом нахождения ее аппроксимирующей зависимости является применение функции convert. Это поясняют следующие примеры:

```
> convert(taylor(exp(x),x,5),polynom);
```

$$1+x+\frac{1}{2}x^2+\frac{1}{6}x^3+\frac{1}{24}x^4$$

```
> f:=x->(x^3+x)/(x^2-1);
```

$$f:=x\to\frac{x^3+x}{x^2-1}$$

```
> convert(f(x),parfrac,x);
```

$$x+\frac{1}{x-1}+\frac{1}{x+1}$$

На рис. 9.4 представлен пример полиномиальной аппроксимации хорошо известной статистической функции erfc(x). Для полинома задана максимальная степень 12, но ввиду отсутствия в разложении четных степеней максимальная степень результата оказывается равна 11.

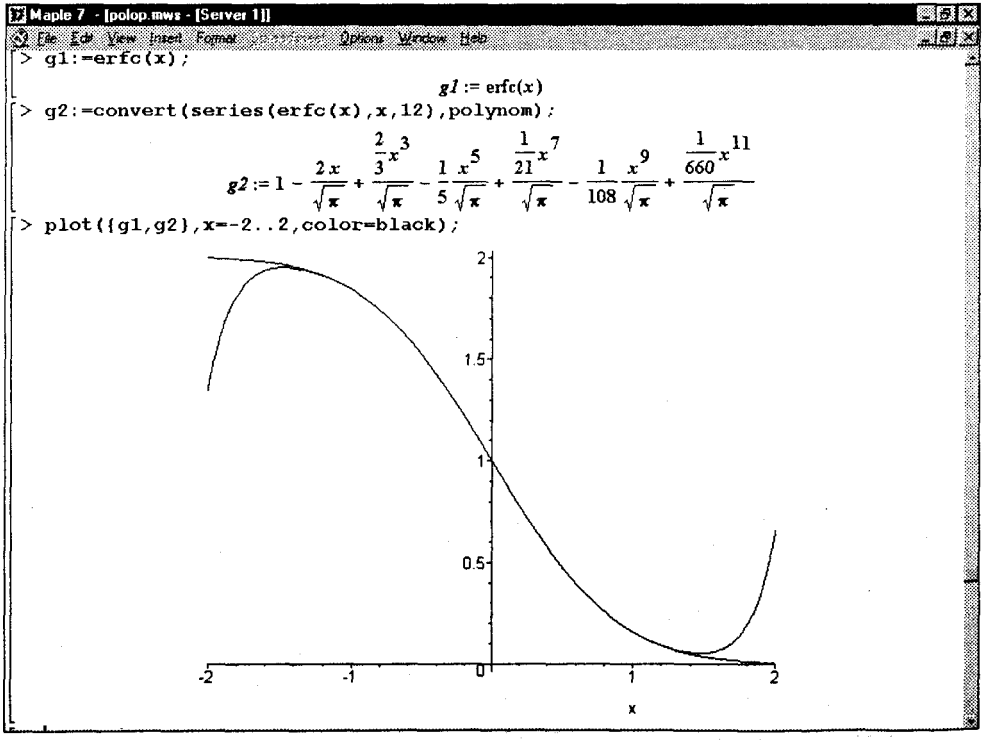


Рис. 9.4. Пример полиномиальной аппроксимации функции erfc(x)

Как видно из приведенного рисунка, в интервале изменения x от –1,5 до 1,5 аппроксимирующее выражение почти повторяет исходную зависимость. Однако затем график аппроксимирующей функции быстро отходит от графика исходной зависимости. При этом он ведет себя иначе даже качественно, никоим образом не показывая асимптотическое поведение, характерное для исходной зависимости.

Полиномиальная интерполяция табличных данных

Если данные некоторой зависимости $y(x)$ заданы векторами **X** и **Y** ее дискретных значений, то для получения интерполяционного степенного многочлена достаточно записать многочлен для всех N пар значений $y_i(x_i)$ при $i = 1...N$ (или $i = 0...N - 1$, если индексы отсчетов начинаются с нуля). Полученная при этом система линейных (относительно коэффициентов полинома) уравнений после решения дает коэффициенты аппроксимирующего полинома. Степень полинома на 1 меньше N, а вычисляемые при x значения $y(x)$ совпадают с табличными (узловыми) в пределах вычислительной погрешности.

На самом деле все это делать не нужно, поскольку Maple 7 имеет реализующую данный алгоритм встроенную функцию `interp(X,Y,v)` или в инертной форме `Interp(X,Y,v)`.

Переменная `v` указывает имя переменной интерполяционного полинома. Векторы `X` и `Y` должны содержать $n + 1 = N$ координат точек исходной зависимости, где n — степень интерполирующего полинома.

Рисунок 9.5 показывает технику применения полиномиальной аппроксимации на основе функции `interp` с построением графика исходных точек и аппроксимирующего полинома. Нетрудно заметить, что график полинома проходит точно через исходные точки — они показаны квадратиками.

В этом примере полезно присмотреться к визуализации результатов вычислений и совместному построению графика интерполирующего полинома и исходных точек. В частности, для построения последних использована обычная функция `plot`, позволяющая выводить на график точки с заданными координатами, причем не только в виде окружностей, но и в виде точек, маленьких крестиков, кружков, квадратов и других фигур. Для выбора типа точек и других параметров графика его надо выделить (установив указатель мыши в поле графика и щелкнув левой кнопкой) и нажать правую кнопку мыши — появится контекстно-зависимое меню с операциями форматирования графика. Это меню показано на рис. 9.5.

Приведем еще несколько примеров использования функции `Interp`:

```
> Interp([2,5,6], [9,8,3], x) mod 11;
```

$$8x^2+6x+9$$

```
> alias(alpha=RootOf(x^4+x+1));
```

$$\alpha$$

```
> a := Interp([0,1,alpha],[alpha,alpha^2,alpha^3], x) mod 2;
```

$$a := x^2+(\alpha^2+\alpha+1)x+\alpha$$

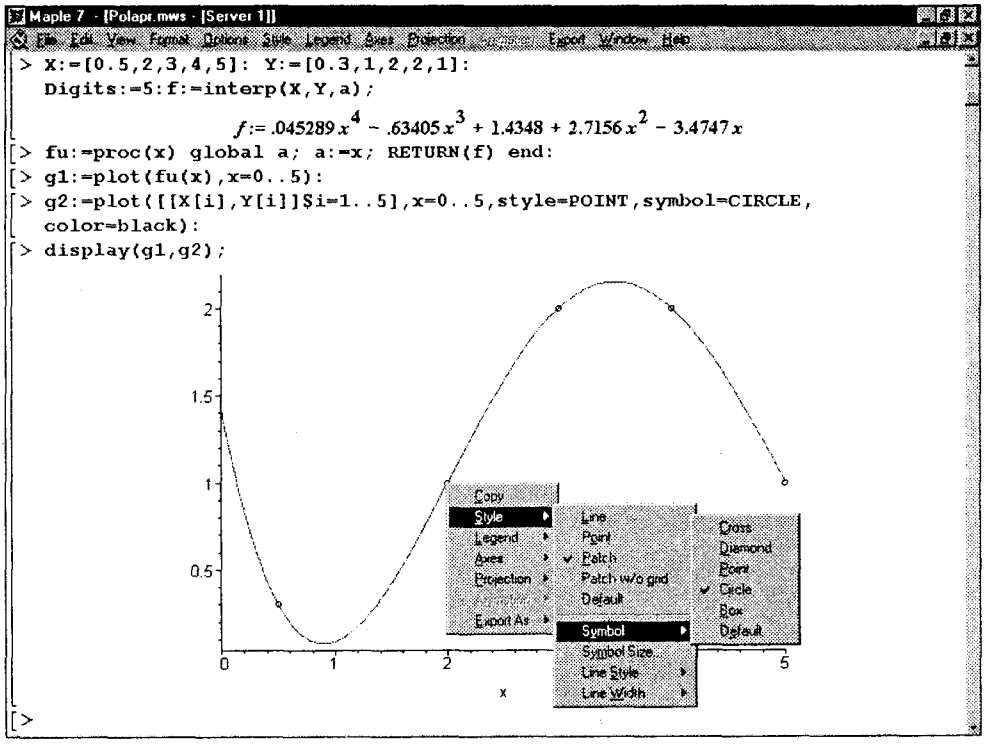


Рис. 9.5. Пример осуществления полиномиальной аппроксимации

Сплайн-интерполяция и аппроксимация

Точность полиномиальной аппроксимации катастрофически падает при увеличении степени аппроксимирующих полиномов. От этого недостатка можно избавиться, используя для аппроксимации *отрезки* полиномов невысокой степени, применяемые для представления части узловых точек. Самым известным методом такой аппроксимации является *сплайн-аппроксимация* на основе применения отрезков кубических полиномов. При этом аппарат сплайн-аппроксимации позволяет получить полиномы, которые дают в узловых точках непрерывность не только представляемой ими функции, но и ее первых и даже вторых производных.

Наглядно сплайн-функцию можно представить в виде гибкой стальной линейки, закрепленной в узловых точках и плавно изгибающейся. Благодаря указанным свойствам сплайнов они неплохо описывают функции, представленные как небольшим числом узловых точек (благодаря плавности сплайн-кривых), так и функции, представляемые очень большим числом узловых точек (поскольку порядок полиномов от этого числа уже не зависит). Недостатком сплайн-аппроксимации является отсутствие общего выражения для всей кривой. Фактически приходится использовать набор сплайн-функций для различных интервалов между узловыми точками.

Для получения сплайн-интерполяции используется Maple-функция `spline(X,Y,var,d)`. Здесь `X` и `Y` — одномерные векторы одинакового размера, несущие значения координат узловых точек исходной функции (причем в произвольном порядке), `var` — имя переменной, относительно которой вычисляется сплайн-функция, наконец, необязательный параметр `d` задает вид сплайна. Он может иметь следующие значения: `linear` — линейная функция, или полином первого порядка, `quadratic` — квадратичная функция, или полином второго порядка, `cubic` — полином третьего порядка, `quartic` — полином четвертого порядка. Если параметр `d` опущен, то сплайн-функция будет строиться на основе полиномов третьего порядка (кубические сплайны).

Технику *сплайновой аппроксимации наглядно поясняет* рис. 9.6. На нем представлено задание векторов узловых точек `X` и `Y` и четырех сплайновых функций, по которым построены их графики. Для одной из функций (с линейной интерполяцией между узлами) показан вид сплайновой функции.

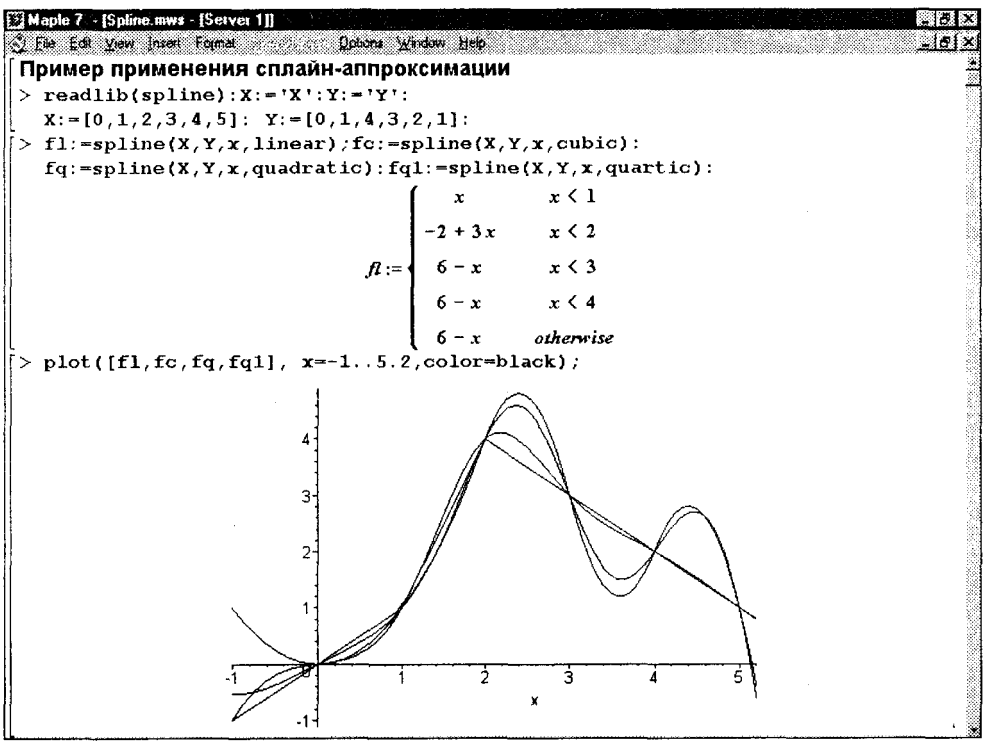


Рис. 9.6. Задание сплайновой аппроксимации и построение графиков полученных функций

Как видно из рис. 9.6, сплайновая функция представляет собой кусочную функцию, определяемую на каждом отдельном интервале. При этом на каждом участке такая функция описывается отдельным полиномом соответствующей степени. Функция `plot` «понимает» такие функции и позволяет без преобразования типов данных строить их графики. Для работы с кусочными функциями можно использовать функции `convert` и `piecewise`.

Прямое и обратное Z-преобразования

Прямое и обратное Z-преобразования функций широко используются при решении задач автоматического управления. Эти преобразования задаются следующими функциями:

- `ztrans(f, n, z)` — прямое преобразование функции $f(n)$ в $f(z)$;
- `invztrans(f, z, n)` — обратное преобразование $f(z)$ в $f(n)$.

Заметим, что прямое Z-преобразование базируется на соотношении `ztrans(f(n),n,z)= = sum(f(n)/z^n,n=0..infinity)`, записанном на Maple-языке. В первых версиях системы Maple Z-преобразования выполнялись средствами библиотеки и требовали вызова командой `readlib(ztrans)`. Но в Maple 7 они включены в ядро системы и предварительного вызова уже не требуют. В этом убеждают следующие примеры:

```
> a:=ztrans(n^2,n,z);
```

$$a := \frac{z(z+1)}{(z-1)^3}$$

```
> invztrans(a,z,n);
```

$$n^2$$

Родственные Z-преобразованиям интегральные преобразования Лапласа и Фурье реализуются с помощью пакета `inttrans` (интегральные преобразования).

Что нового мы узнали?

В этом уроке мы научились:

- Выполнять поиск экстремумов функций.
- Осуществлять поиск минимумов и максимумов аналитических функций.
- Анализировать функции на непрерывность.
- Находить сингулярные точки функций.
- Вычислять асимптотические и иные разложения.
- Анализировать сложные функции.
- Работать с кусочными функциями.
- Выполнять операции с полиномами.
- Интерполировать и аппроксимировать функциональные зависимости.
- Вычислять прямые и обратные Z-преобразования.

Символьные (аналитические) операции

- ❒ Основные операции с выражениями
- ❒ Подстановки
- ❒ Упрощение выражений
- ❒ Расширение выражений
- ❒ Факторизация выражений
- ❒ Комплектование по степеням
- ❒ Программирование символьных операций

Основные операции с выражениями

Работа с частями выражений

Выражения (expr) или уравнения (eqn) обычно используются как сами по себе, так и в виде равенств или неравенств. В последнем случае объекты с выражениями имеют левую и правую части. Для простейших манипуляций с выражениями полезны следующие функции:

- `cost(a)` — возвращает число сложений и умножений в выражении `a` (функция пакета codegen);
- `lhs(eqn)` — выделяет левую часть `eqn`;
- `rhs(eqn)` — выделяет правую часть `eqn`;
- `normal(expr)` — дает нормализацию (сокращение) `expr` в виде дроби;
- `numer(expr)` — выделяет числитель `expr`;
- `denom(expr)` — выделяет знаменатель `expr`.

Ввиду очевидности действия этих функций ограничимся наглядными примерами их применения:

```
> with(codegen,cost):
> cost(x^3+b^2-x);
```

$$2\ additions + 3\ multiplications$$

```
> lhs(sin(x)^2+cos(x)^2=1);
```

$$\sin(x)^2 + \cos(x)^2$$

```
> rhs(sin(x)^2+cos(x)^2=1);
```

$$1$$

```
> normal(2/4+3/6+6/12);
```

$$\frac{3}{2}$$

```
> f:=5*(a-b)^2/(a^2-2*a*b-b^2);
```

$$f := 5\,\frac{(a-b)^2}{a^2 - 2\,a\,b - b^2}$$

```
> numer(f);
```

$$5\,(a-b)^2$$

```
> denom(f);
```

$$a^2 - 2\,a\,b - b^2$$

ПРИМЕЧАНИЕ *Обратите внимание на то, что в предшествующих версиях Maple загрузка библиотечной функции cost выполнялась иначе — командой readlib(cost). Это обстоятельство может служить причиной неверной работы документов, созданных в старых версиях Maple, в среде описываемой версии Maple 7.*

Работа с уровнями вложенности выражений

В общем случае выражения могут быть многоуровневыми и содержать объекты, расположенные на разных уровнях вложенности. Приведем две функции для оценки уровней выражений и списков:

- `nops(expr)` — возвращает число объектов первого уровня (операндов) в выражении `expr`;
- `op(expr)` — возвращает список объектов первого уровня в выражении `expr`;
- `op(n,expr)` — возвращает n-й объект первого уровня в выражении `expr`.

Ниже представлены примеры применения этих функций:

```
> nops(a+b/c);
```

$$2$$

```
> op(a+b/c);
```

$$a, \frac{b}{c}$$

```
> op(1,a+b/c);
```

$$a$$

```
> op(2,a+b/c);
```

$$\frac{b}{c}$$

Рекомендуется просмотреть и более сложные примеры на применение этих функций в справке.

Преобразование выражений в тождественные формы

Многие математические выражения имеют различные тождественные формы. Порою преобразование выражения из одной формы в другую позволяет получить результат, более удобный для последующих вычислений. Кроме того, различные функции Maple 7 работают с разными формами выражений и разными типами данных. Поэтому большое значение имеет целенаправленное преобразование выражений и данных.

Основной функцией для такого преобразования является функция `convert`:

```
convert(expr, form, arg3, ...)
```

Здесь expr — любое выражение, form — наименование формы, arg3, ... — необязательные дополнительные аргументы.

convert — простая и вместе с тем очень мощная функция. Ее мощь заключается в возможности задания множества параметров. Их полный перечень (76 штук!) можно найти в справке по функции convert. Многие из этих параметров очевидны с первого взгляда, поскольку повторяют наименования типов чисел, данных или функций. Например, опции binary, decimal, hex и octal преобразуют заданные числа в их двоичное, десятичное, шестнадцатеричное и восьмеричное представление. Параметр vector задает преобразование списка в вектор (напоминаем, что список и вектор — разные типы данных), а параметр matrix — в матрицу. Приведем примеры применения функции convert:

```
> convert(123,binary);
```

$$1111011$$

```
> convert([a,b,c,d],`+`);
```

$$a+b+c+d$$

```
> f:=seq(x[i]^n,i=1..4);
```

$$f := x_1^{\,n}, x_2^{\,n}, x_3^{\,n}, x_4^{\,n}$$

```
> x:='x': convert(sinh(x),exp);
```

$$x := x$$

$$\frac{1}{2}e^{x} - \frac{1}{2}\frac{1}{e^{x}}$$

```
> convert(1.234567,fraction);
```

$$\frac{50737}{41097}$$

```
> convert(1/7,float);
```

$$.1428571429$$

```
> convert(sin(I*x),exp);
```

$$I\left(\frac{1}{2}e^{x} - \frac{1}{2}\frac{1}{e^{x}}\right)$$

```
> convert(sinh(x),exp);
```

$$\frac{1}{2}e^{x} - \frac{1}{2}\frac{1}{e^{x}}$$

```
> convert(arcsinh(x),ln);
```

$$\ln(x+\sqrt{x^2+1})$$

```
> convert(12345,list);
```

$$[12345]$$

```
> convert(binomial(m,n),factorial);
```

$$\frac{m!}{n!\,(m-n)!}$$

```
> convert([[1,2],[3,4],[5,6]],table);
```

$$\text{table}([(1,1)=1,(2,1)=3,(2,2)=4,(3,1)=5,(3,2)=6,(1,2)=2])$$

```
> convert(-Pi,signum);
```

$$-\pi$$

```
> s:=taylor(sin(x),x,8);
```

$$s := x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 + O(x^8)$$

```
> p:=convert(s,polynom);
```

$$p := x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7$$

```
> convert(p,float);
```

$$x - .1666666667\,x^3 + .008333333333\,x^5 - .0001984126984\,x^7$$

```
> f:=(x^4+x)/(x^2-1);
```

$$f := \frac{x^4 + x}{x^2 - 1}$$

```
> convert(f,parfrac,x);
```

$$x^2 + 1 + \frac{1}{x - 1}$$

```
> s:=series(f,x,5);
```

$$s := -x - x^3 - x^4 + O(x^5)$$

```
> convert(s,polynom);# Удаление члена ряда, описывающего погрешность
```

$$-x - x^3 - x^4$$

Из этих примеров (их список читатель может пополнить самостоятельно) следует, что функция преобразования `convert` является одной из самых мощных функций Maple. С ее помощью можно получить множество различных форм одного и того же выражения.

Преобразование выражений

Еще одним мощным средством преобразования выражений является функция `combine`. Она обеспечивает объединение показателей степенных функций и преобразование тригонометрических и некоторых иных функций. Эта функция может записываться в трех формах:

```
combine(f)
combine(f, n)
combine(f, n, opt1, opt2, …)
```

Здесь `f` — любое выражение, множество или список выражений; `n` — имя, список или множество имен; `opt1`, `opt2`, ... — имена параметров. Во втором аргументе можно использовать следующие функции:

`@@`	`abs`	`arctan`	`conjugate`	`exp`
`ln`	`piecewise`	`polylog`	`power`	`product`
`Ps`	`radical`	`range`	`signum`	`trig`

Примеры применения функции combine представлены ниже:

```
> combine(exp(2*x)^2,exp);
```

$$e^{(4x)}$$

```
> combine(2*sin(x)^2+2*cos(x)^2);
```

$$2$$

```
> combine(sin(x)*cos(x));
```

$$\frac{1}{2}\sin(2x)$$

```
> combine(Int(x,x=a..b)-Int(x^2,x=a..b));
```

$$\int_a^b -x^2 + x\,dx$$

Эти примеры далеко не исчерпывают возможностей функции combine в преобразовании выражений. Рекомендуется обзорно просмотреть примеры применения функции combine с разными параметрами, приведенные в справочной системе Maple 7.

Контроль за типами объектов

Выражения и их части в Maple 7 рассматриваются как объекты. В ходе манипуляций с ними важное значение имеет контроль за типом объектов. Одной из основных функций, обеспечивающих такой контроль, является функция whattype(object), возвращающая тип объекта, например string, integer, float, fraction, function и т. д. Могут также возвращаться данные об операторах. Примеры применения этой функции даны ниже:

```
> whattype(2+3);
integer
> whattype(Pi);
symbol
> whattype(123./5);
float
> whattype(1/3);
fraction
> whattype(sin(x));
function
> whattype([1,2,3,a,b,c]);
list
> whattype(a+b+c);
+
> whattype(a*b/c);
*
> whattype(a^b);
^
> whattype(1+2+3=4);
=
```

С помощью функции type(object,t) можно выяснить, относится ли указанный объект к соответствующему типу t, например:

```
> type(2+3,integer);
```
true
```
> type(sin(x),function);
```
true
```
> type(hello,string);
```
false
```
> type("hello",string);
```
true
```
> type(1/3,fraction);
```
true

При успешном соответствии типа объекта указанному (второй параметр) функция type возвращает логическое значение true, в противном случае — false.

Для более детального анализа объектов может использоваться функция hastype(expr, t), где expr — любое выражение и t — наименование типа подобъекта. Эта функция возвращает логическое значение true, если подобъект указанного типа содержится в выражении expr. Примеры применения этой функции даны ниже:

```
> hastype(2+3,integer);
```
true
```
> hastype(2+3/4,integer);
```
false
```
> hastype(2*sin(x),function);
```
true
```
> hastype(a+b-c/d,`+`);
```
true

Еще одна функция — has(f,x) — возвращает логическое значение true, если подобъект x содержится в объекте f, и false в ином случае:

```
> has(2*sin(x),2);
```
true
```
> has(2*sin(x),`/`);
```
false
```
> has(2*sin(x),3-1);
```
true

Следует отметить, что соответствие подобъекта выражения указанному подобъекту понимается в математическом смысле. Так, в последнем примере подобъект «3 – 1», если понимать его буквально, в выражении 2*sin(x) не содержится, но Maple-язык учитывает соответствие 3 – 1 = 2, и потому функция has в последнем примере возвращает true.

Подстановки

Функциональные преобразования подвыражений

Нередко бывает необходимо заменить некоторое подвыражение в заданном выражении на функцию от этого подвыражения. Для этого можно воспользоваться функцией applyop:

- applyop(f, i, e) — применяет функцию f к i-му подвыражению выражения e;
- applyop(f, i, e, …, xk, …) — применяет функцию f к i-му подвыражению выражения e с передачей необязательных дополнительных аргументов xk.

Ниже даны примеры применения этой функции:

```
> restart:applyop(sin,2,a+x);
```

$$a+\sin(x)$$

```
> applyop(f,1,g,2,a+b);
```

$$f(g,2,a+b)$$

```
> applyop(f,{2,3},a+x+b);
```

$$a+f(x)+f(b)$$

```
> applyop(f,{1,2},x/y+z);
```

$$f\left(\frac{x}{y}\right)+f(z)$$

```
> p:=y^2-2*y-3;
```

$$p:=y^2-2y-3$$

```
> applyop(f,2,p);
```

$$y^2+f(-2y)-3$$

```
> applyop(f,{2,3},p);
```

$$y^2+f(-2y)+f(-3)$$

```
> applyop(f,{[2,1],3},p);
```

$$y^2+f(-2)y+f(-3)$$

```
> applyop(abs,{[2,1],3},p);
```

$$y^2+2y+3$$

Функциональные преобразования элементов списков

Еще две функции, реализующие операции подстановки, указаны ниже:

```
map(fcn, expr, arg2, …, argn)
map2(fcn, arg1, expr, arg3, …, argn)
```

Здесь fcn — процедура или имя, expr — любое выражение, argi — необязательные дополнительные аргументы для fcn.

Первая из этих функций позволяет приложить fcn к операндам выражения expr. Приведенные далее примеры иллюстрируют использование функции map.

```
> f:=x->x^2;
```
$$f := x \to x^2$$
```
> map(f,[1,2,3]);
```
$$[1, 4, 9]$$
```
> map(f,[x,y,z]);
```
$$[x^2, y^2, z^2]$$
```
> map(x->x^n,[1,2,3]);
```
$$[1, 2^n, 3^n]$$
```
> L:=[1,2,3,4];
```
$$L := [1, 2, 3, 4]$$
```
> map(proc(x,y) x*y+1 end,[1,2,3,4],2);
```
$$[3, 5, 7, 9]$$
```
> map(int,L,x);
```
$$[x, 2x, 3x, 4x]$$
```
> map(F,[1,2,3],x,y,z);
```
$$[F(1, x, y, z), F(2, x, y, z), F(3, x, y, z)]$$

Из этих примеров нетрудно заметить, что если второй параметр функции `map` — список, то функция (первый параметр) прикладывается к каждому элементу списка, так что возвращается также список. Из последнего примера видно, что если за вторым параметром идет перечисление аргументов, то они включаются в список параметров функции.

Функция `map2` отличается иным расположением параметров. Ее действие наглядно поясняют следующие примеры:

```
> map2(w,g,{a,b,c});
```
$$\{w(g, a), w(g, b), w(g, c)\}$$
```
> map2(op,1,[a+b+i,c+d+k,e+f+j]);
```
$$[a, c, e]$$
```
> map2(op,3,[a+b+i,c+d+k,e+f+j]);
```
$$[i, k, j]$$
```
> map2(diff,[sin(x),cos(x),x^n],x);
```
$$\left[\cos(x), -\sin(x), \frac{x^n n}{x}\right]$$

Подстановки с помощью функций add, mul и seq

Заметим, что операции, подобные описанным выше, Maple 7 реализует и с рядом других функций. Ограничимся примерами на подстановки с помощью функций сложения `add`, умножения `mul` и создания последовательностей `seq`:

```
> add(i,i=[a,b,c]);
```
$$a + b + c$$
```
> add(i^2,i=[a,b,c]);
```
$$a^2 + b^2 + c^2$$

```
> add(i^2,i=[1,2,3]);
```
$$14$$
```
> mul(x-i,i=0..4);
```
$$x(x-1)(x-2)(x-3)(x-4)$$
```
> mul(x^i,i=0..4);
```
$$x^{10}$$
```
> seq(w(i),i={a,b,c});
```
$$w(a), w(b), w(c)$$
```
> seq(w(x,y,z),i={1,2,3});
```
$$w(x,y,z), w(x,y,z), w(x,y,z)$$
```
> seq(int(x^i,x),i={1,2,3,4});
```
$$\frac{1}{2}x^2, \frac{1}{3}x^3, \frac{1}{4}x^4, \frac{1}{5}x^5$$

Подстановки с помощью функций subs и subsop

Подстановки в общем случае служат для замены одной части выражения на другую. Частными видами подстановок являются такие виды операций, как замена одной переменной на другую или замена символьного значения переменной ее численным значением. Основные операции подстановки выполняют следующие функции:

- `subs(x=a,e)` — в выражении `e` заменяет подвыражение `x` на подвыражение `a`;
- `subs(s1,…,sn,e)` — в выражении `e` заменяет одни подвыражения на другие, выбирая их из списков `s1`, ..., `sn` вида `x=a`;
- `subsop(eq1, eq2, …, eqi, …,eqn, e)` — в выражении `e` заменяет указанные в `eqi` операнды другими, указанными в правой части равенств `eqi` вида `ni=ei`, где `ni` — номер операнда, `ei` — выражение для замены.

Все эти функции возвращают измененное после подстановки выражение. Ниже показаны примеры применения функций подстановок:

```
> subs(a=b,b^2-2*a*b-b^2);
```
$$-2\,b^2$$
```
> subs(a=2,b=1,b^2-2*a*b-b^2);
```
$$-4$$
```
> subs(c=a-b,(a^2-2*a*b+b^2)/c);
```
$$\frac{a^2-2\,a\,b+b^2}{a-b}$$
```
> normal(%);
```
$$a-b$$
```
> subs(a=x,b=y,c=z,[a,b,c]);
```
$$[x,y,z]$$
```
> subs({x=y,y=x},[x,y]);
```
$$[y,x]$$

```
> subs(a=sin(x),b=cos(x),a^2+b*b);
```
$$\sin(x)^2+\cos(x)^2$$
```
> simplify(%);
```
$$1$$
```
> subsop(1=x,a+b+c);
```
$$x+b+c$$
```
> subsop(2=x,a+b+c);
```
$$a+x+c$$
```
> subsop(3=x,a+b+c);
```
$$a+b+x$$
```
> subsop(3=x,a+b/c);
```
Error, improper op or subscript selector
```
> subsop(1=sin(x),(1+cos(x))/b);
```
$$\frac{\sin(x)}{b}$$
```
> subsop(2=sin(x),(1+cos(x))/b);
```
$$(1+\cos(x))\sin(x)$$
```
> subsop(1=sin(x),2=sin(x),(1+cos(x))/b);
```
$$\sin(x)^2$$

Следует обратить внимание на то, что результат подстановок, полученный с помощью функции subop, порой может не совпадать с ожидаемым. Поэтому полезно контролировать получаемые в результате подстановок выражения на их корректность.

Одним из важных применений подстановок является проверка правильности решений уравнений и систем уравнений. Ниже дан пример такой проверки:

```
> eqs:={x+y+z=6,y/x=z-1,z-x=2};
```
$$eqs := \{x+y+z=6, z-x=2, \frac{y}{x}=z-1\}$$
```
> res:=solve(eqs,{x,y,z});
```
$$res := \{z=-2, y=12, x=-4\}, \{y=2, z=3, x=1\}$$
```
> subs(res,eqs);
```
$$\{2=2, 6=6, -3=-3\}$$

Здесь задана система из трех нелинейных уравнений, которая затем решена функцией solve. В конце примера с помощью функции подстановки выполнена проверка правильности решения. Оно верно, поскольку у всех уравнений значение левой части совпадает со значением правой части.

Функции сортировки и селекции

Сортировка и селекция выражений широко используются в практике символьных преобразований. Для выполнения сортировки служит функция sort, применяемая в одной из следующих форм:

```
sort(L)          sort(L, F)          sort(A)          sort(A, V)
```

Здесь L — список сортируемых значений, F — необязательная булева процедура с двумя аргументами, A — алгебраическое выражение, V — необязательные дополнительные переменные.

```
> restart;
> sort([y,s,f,a,c,i]);t([2,5,1,7,3,8]);
```

$$[a, c, f, i, s, y]$$

$$t([2, 5, 1, 7, 3, 8])$$

```
> sort([y,s,f,a,c,i]);
```

$$[a, c, f, i, s, y]$$

```
> sort([y,s,f,a,c,i],lexorder);
```

$$[a, c, f, i, s, y]$$

```
> sort(1+x^4-x^2+x);
```

$$x^4 - x^2 + x + 1$$

```
> sort(y*x^2+x*y+y-x^2+x^4*y^5);
```

$$x^4 y^5 + x^2 y - x^2 + x y + y$$

```
> sort((y+z+x)/(y-x-z),{x,y});
```

$$\frac{x + y + z}{-x + y - z}$$

```
> names:=["Peter","Anna","Vladimir","Ivan"];
```

$$names := [\text{"Peter"}, \text{"Anna"}, \text{"Vladimir"}, \text{"Ivan"}]$$

```
> sort(names);
```

$$[\text{"Anna"}, \text{"Ivan"}, \text{"Peter"}, \text{"Vladimir"}]$$

```
> integers:=[$10..30];
```

$$integers := [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]$$

ПРИМЕЧАНИЕ Вы можете проверить, что функция sort в Maple 7 способна сортировать буквы и даже *слова русского языка.*

Если функция сортировки меняет порядок расположения членов в выражении (или порядок расположения выражений), то другая функция — select — служит для выделения требуемого выражения:

```
select(f, e)
select(f, e, b1, ..., bn)
```

Как бы обратной ей по действию служит функция remove, устраняющая заданные выражения:

```
remove(f, e)
remove(f, e, b1, ..., bn)
```

В этих функциях f — процедура, возвращающая логическое значение, e — список, множество, сумма, произведение или функция, b1, ..., bn — необязательные дополнительные аргументы.

Далее даны примеры применения этих функций.

```
> integers := [$10..30]:
> select(isprime,integers);
```
$[11, 13, 17, 19, 23, 29]$
```
> remove(isprime,integers);
```
$[10, 12, 14, 15, 16, 18, 20, 21, 22, 24, 25, 26, 27, 28, 30]$
```
> f:=2*ln(a*x)*exp(x)*ln(y);
```
$f := 2\ln(a\,x)\,e^{x}\ln(y)$
```
> select(has,f,x);
```
$\ln(a\,x)\,e^{x}$
```
> remove(has,f,x);
```
$2\ln(y)$
```
> f:=indets(f);
```
$f := \{a, x, y, e^{x}, \ln(y), \ln(a\,x)\}$
```
> select(type,f,name);
```
$\{a, x, y\}$
```
> remove(type,f,name);
```
$\{e^{x}, \ln(y), \ln(a\,x)\}$
```
> f:=2*ln(x)*(y+1);
```
$f := 2\ln(x)\,(y+1)$
```
> c:=remove(has,f,x);
```
$c := 2\,y + 2$
```
> f/c;
```
$$2\,\frac{\ln(x)\,(y+1)}{2\,y+2}$$
```
> select(has,f,x);
```
$\ln(x)$

Maple имеет также оператор селекции `A[expr]`. Его действие поясняют следующие примеры:

```
> restart;
> S:=[a+b*c,x^2,c,1,2,3];
```
$S := [a + b\,c, x^2, c, 1, 2, 3]$
```
> S[1];
```
$a + b\,c$
```
> S[1..2];
```
$[a + b\,c, x^2]$
```
> S[-2..-1];
```
$[2, 3]$
```
> S[3..3];
```
$[c]$
```
> S[3..2];
```
$[\,]$
```
> S[4..6];
```
$[1, 2, 3]$

```
> X:=S[];
```
$$X := a + b\,c, x^2, c, 1, 2, 3$$
```
> X[1];
```
$$a + b\,c$$
```
> X[1..2];
```
$$a + b\,c, x^2$$
```
> X[-2..-1];
```
$$2, 3$$
```
> S:={a,b,c};
```
$$S := \{a, b, c\}$$
```
> S[1];
```
$$a$$
```
> S[3];
```
$$c$$
```
> S[1..2];
```
$$\{a, b\}$$
```
> S[-2..-1];
```
$$\{b, c\}$$

Упрощение выражений

Функция `simplify` — одна из самых мощных в системах символьной математики. Она предназначена для упрощения математических выражений. «Все гениальное просто», — любим мы повторять, хотя это далеко не всегда так. Тем не менее стремление представить многие математические выражения в наиболее простом виде поощряется в большинстве вычислений и нередко составляет их цель.

В системе Maple 7 функция упрощения используется в следующем виде:

- `simplify(expr)` — возвращает упрощенное выражение `expr` или повторяет его, если упрощение в рамках правил Maple 7 невозможно;
- `simplify(expr, n1, n2, ...)` — возвращает упрощенное выражение `expr` с учетом *параметров с именами* `n1`, `n2`, ... *(в том числе заданных списком или множеством)*;
- `simplify(expr,assume=prop)` — возвращает упрощенное выражение `expr` с учетом всех условий.

Функция `simplify` — многоцелевая. Она обеспечивает упрощение математических выражений, выполняя следующие типовые действия (для простоты обозначим их как `->`):

- комбинируя цифровые подвыражения (`3*x*5->15*x`, `10*x/5->2*x`);
- приводя подобные множители в произведениях (`x^3*a*x->a*x^4`);
- приводя подобные члены в суммах (`5*x+2+3*x->8*x+2`);
- используя тождества, содержащие ноль (`a+0->a`, `x-0->x`);

- используя тождества, содержащие единицу (1*x->x);
- распределяя целочисленные показатели степени в произведениях ((3*x*y^3)^2 ->9*x^2*y^6);
- сокращая expr на наибольший общий полиномиальный или иной множитель;
- понижая степень полиномов там, где это возможно;
- используя преобразования, способные упростить выражения.

Несмотря на свою гибкость, функция simplify не всегда способна выполнить возможные упрощения. В этом случае ей надо подсказать, в какой области ищутся упрощения и где можно найти соответствующие упрощающие преобразования. С этой целью в функцию simplify можно включать дополнительные параметры.

В качестве параметров могут задаваться имена специальных математических функций и указания на область действия упрощений: BesselI, BesselJ, BesselK, BesselY, Ei, GAMMA, RootOf, LambertW, dilog, exp, ln, sqrt, polylog, pg, pochhammer, trig (для всех тригонометрических функций), hypergeom, radical, power и atsign (для операторов). Полезен также параметр symbolic, явно указывающий на проведение символьных преобразований.

Возможно также применение функции simplify в форме simplify[<name>], где <name> — одно из следующих указаний: atsign, GAMMA, hypergeom, power, radical, RootOf, sqrt, trig.

Ниже даны примеры применения функции simplify:

```
> simplify(4^(1/2)+3);
```

$$5$$

```
> simplify((x^y)^z+3^(3),power);
```

$$(x^y)^z + 27$$

```
> simplify(sin(x)^2+cos(x)^2,trig);
```

$$1$$

```
> e:=cos(x)^5+sin(x)^4+2*cos(x)^2-2*sin(x)^2-cos(2*x);;
```

$$e := \cos(x)^5 + \sin(x)^4 + 2\cos(x)^2 - 2\sin(x)^2 - \cos(2x)$$

```
> simplify(e);
```

$$\cos(x)^5 + \cos(x)^4$$

```
> simplify(GAMMA(n+4)/GAMMA(n),GAMMA);
```

$$n(n+1)(n+2)(n+3)$$

```
> r:=RootOf(x^2-2=0,x):
> simplify(r^2,RootOf);
```

$$2$$

```
> simplify(1/r,RootOf);
```

$$\frac{1}{2}\,\mathrm{RootOf}(\_Z^2 - 2)$$

```
> simplify(ln(x*y),power,symbolic);
```

$$\ln(x) + \ln(y)$$

```
> e:=(-5*b^2*a)^(1/2);
```

$$e := \sqrt{-5\,b^2\,a}$$

```
> simplify(e,radical);
```

$$\sqrt{5}\,\sqrt{-b^2 a}$$

```
> simplify(e,radical,symbolic);
```

$$b\sqrt{5}\,\sqrt{-a}$$

```
> simplify(GAMMA(n+1)/n!)[GAMMA];
```

$$1_{\Gamma}$$

Действие функции simplify существенно зависит от областей определения переменных. В следующем примере упрощение выражения не произошло, поскольку результат этой операции неоднозначен:

```
> restart;
> simplify(sqrt(x^4*y^2));
```

$$\sqrt{x^4 y^2}$$

Однако, определив переменные как реальные или положительные, можно легко добиться желаемого упрощения:

```
> simplify(sqrt(x^4*y^2),assume=positive);
```

$$x^2 y$$

```
> simplify(sqrt(x^4*y^2),assume=real);
```

$$x^2|y|$$

Читателю настоятельно рекомендуется просмотреть все разделы справочной системы, относящиеся к примерам применения функции simplify (в том числе с другими функциями символьных преобразований), поскольку их число очень велико и эти примеры наглядно демонстрируют необходимость правильного применения разнообразных параметров для придания упрощениям нужного характера. Если функция simplify не способна выполнить упрощение выражения expr, то она просто его повторяет. Это сигнал к применению опций.

Расширение выражений

Даже в жизни мы говорим: «не все так просто». Порою упрощенное выражение скрывает его особенности, знание которых является желательным. Функция expand «расширяет» выражение expr и записывается в виде:

```
expand(expr, expr1, expr2, …, exprn)
```

где expr — расширяемое выражение, expr1, expr2, ..., exprn — необязательные подвыражения — опции. Имеется также инертная форма данной функции — Expand(expr). Кроме того, возможно применение операторной конструкции frontend(expans,[expr]).

Функция expand раскладывает рациональные выражения на простые дроби, полиномы на полиномиальные разложения, она способна раскрыть многие математические функции, такие как sin, cos, tan, sinh, cosh, tanh, det, erf, exp, factorial, GAMMA, ln, max, min, Psi, binomial, sum, product, int, limit, bernoulli, euler, abs, signum, pochhammer, polylog, BesselJ, BesselY, BesselI, BesselK, AngerJ, Beta, Hankel, Kelvin,

Struve, WeberE и функция piecewise. С помощью дополнительных аргументов expr1, expr2, ..., exprn можно задать расширение отдельных фрагментов в expr.

Примеры применения функции expand приведены ниже:

```
> expand((x+2)*(x+3)*(x+4));
```

$$x^3 + 9\,x^2 + 26\,x + 24$$

```
> expand(sin(2*x));
```

$$2\sin(x)\cos(x)$$

```
> expand(sin(x+y));
```

$$\sin(x)\cos(y) + \cos(x)\sin(y)$$

```
> expand([(a+b)*(a-b),tan(2*x)]);
```

$$\left[a^2 - b^2, 2\,\frac{\tan(x)}{1-\tan(x)^2}\right]$$

```
> expand((a+d)*(b+d)*(c+d));
```

$$a\,b\,c + a\,b\,d + a\,d\,c + a\,d^2 + d\,b\,c + d^2\,b + d^2\,c + d^3$$

```
> expand((x+1)*(y+1));
```

$$x\,y + x + y + 1$$

```
> expand((y+1),(x+1));
```

$$y + 1$$

```
> expand((x+1)*(y+z));
```

$$x\,y + x\,z + y + z$$

```
> expand((x+1)*(y+z), x+1);
```

$$(x+1)\,y + (x+1)\,z$$

```
> frontend(expand,[(a+b)^3]);
```

$$a^3 + 3\,a^2\,b + 3\,a\,b^2 + b^3$$

Факторизация выражений

Разложение целых и рациональных чисел

Для разложения целых или рациональных чисел на множители в виде простых чисел служит функция:

```
ifactor(n) или ifactor(n,method)
```

где n — число, method — параметр, задающий метод разложения. Другая библиотечная функция, ifactors(n), возвращает результат разложения в форме вложенных списков:

```
> ifactor(123456789);
```

$$(3)^2\ (3803)\ (3607)$$

```
> ifactor(30!);
```

$$(2)^{26}\ (3)^{14}\ (5)^7\ (7)^4\ (11)^2\ (13)^2\ (17)\ (19)\ (23)\ (29)$$

```
> ifactor(12!/20!);
```

$$\frac{1}{(2)^8\ (3)^3\ (5)^2\ (7)\ (13)\ (17)\ (19)}$$

```
> ifactor(100/78);
```

$$\frac{(2)\ (5)^2}{(3)\ (13)}$$

```
> readlib(ifactors):
> ifactors(100/78);
```

$$[1, [[2, 1], [5, 2], [3, -1], [13, -1]]]$$

Разложение выражений (факторизация)

Для алгебраических выражений функция факторизации записывается в вычисляемой и невычисляемой (инертной) формах:

```
factor(a)                    Factor(a)
factor(a,K)                  Factor(a,K)
```

Здесь `a` — полином с несколькими переменными, `K` — необязательное алгебраическое расширение. Для получения результата от инертной формы функции факторизации надо использовать функции вычисления `evala` или `evalgf`.

Главная цель факторизации — это нахождение максимального числа независимых сомножителей выражения, линейных по заданным переменным с коэффициентами наиболее простой формы. Ниже представлены примеры применения функции `factor`:

```
> factor(a^2+2*a*b+b^2);
```

$$(a+b)^2$$

```
> factor(a^2-2*a*b-b^2);
```

$$a^2 - 2\,a\,b - b^2$$

```
> p:=expand((x-1)*(x-2)*(x-3)*(x-4));
```

$$p := x^4 - 10\,x^3 + 35\,x^2 - 50\,x + 24$$

```
> factor(p);
```

$$(x-1)\,(x-2)\,(x-3)\,(x-4)$$

```
> factor(x^5-2,2^(1/5));
```

$$(x - 2^{(1/5)})\,(x^4 + x^3\,2^{(1/5)} + x^2\,2^{(2/5)} + x\,2^{(3/5)} + 2^{(4/5)})$$

```
> alias(alpha=RootOf(x^2-2));
```

$$\alpha$$

```
> factor(x^2-2,alpha);
```

$$(x+\alpha)\,(x-\alpha)$$

```
> factor(x^3-y^3);
```

$$(x-y)\,(x^2+x\,y+y^2)$$

```
> factor(x^3-y^3,(-2)^(1/2));
```

$$(x-y)\,(x^2+x\,y+y^2)$$

```
> factor(x^3-y^3,(-3)^(1/2));
```

$$\frac{1}{4}(2x+y-y\sqrt{-3})(2x+y+y\sqrt{-3})(x-y)$$

```
> factor(x^3-3,complex);
```

$$(x+.7211247852+1.249024766\,I)(x+.7211247852-1.249024766\,I)$$
$$(x-1.442249570)$$

Комплектование по степеням

Еще одна функция общего назначения — `collect` — служит для комплектования выражения `expr` по степеням указанного фрагмента `x` (в том числе множества либо списка). Она задается в одной из следующих форм:

```
collect(a, x)
collect(a, x, form, func)
```

Во второй форме этой функции дополнительно задаются параметры `form` (форма) и `func` (функция или процедура). Параметр `form` может иметь два значения: `recursive` (рекурсивная форма) и `distributed` (дистрибутивная форма). Параметр `func` позволяет задать имя функции, по которой будет идти комплектование `expr`.

Примеры применения функции `collect` представлены ниже:

```
> collect(x+x^3-2*x,x);
```

$$-x+x^3$$

```
> collect(x+2*y^3+x+3+x^3*y,recursive,x);
```

$$x(2x+2y^3+3+x^3y)$$

```
> collect(x+2*y^3+x+3+x^3*y,distributive,y);
```

$$y(2x+2y^3+3+x^3y)$$

```
> f:=a*exp(x)-exp(x)*x-x;
```

$$f := a\,e^x - e^x x - x$$

```
> collect(f,exp(x));
```

$$(a-x)\,e^x - x$$

```
> g:=int(x*(exp(x)+exp(-x)),x);
```

$$g := e^x x - e^x - \frac{x}{e^x} - \frac{1}{e^x}$$

```
> collect(g,exp(x));
```

$$(x-1)\,e^x + \frac{-x-1}{e^x}$$

```
> p:=x*y+a*x*y+y*x^2-a*y*x^2+x+a*x;
```

$$p := xy + axy + yx^2 - ayx^2 + x + ax$$

```
> collect(p,[x,y],recursive);
```

$$(1-a)yx^2 + ((1+a)y+1+a)x$$

```
> collect(p,[x,y],distributed);
```

$$(1+a)x+(1+a)xy+(1-a)yx^2$$

```
> f:=a^3*x^2-x+a^3+a;
```

$$f := a^3x^2 - x + a^3 + a$$

```
> collect(f,x);
```

$$a^3x^2 - x + a^3 + a$$

```
> collect(f,x,factor);
```

$$a^3x^2 - x + a(a^2+1)$$

```
> p:=y/x+2*z/x+x^(1/3)-y*x^(1/3);
```

$$p := \frac{y}{x} + \frac{2z}{x} + x^{(1/3)} - yx^{(1/3)}$$

```
> collect(p,x);
```

$$(1-y)x^{(1/3)} + \frac{y+2z}{x}$$

Программирование символьных операций

Реализация итераций Ньютона в символьном виде

Найти достаточно простую и наглядную задачу, решение которой отсутствует в системе Maple 7, не очень просто. Поэтому для демонстрации решения задачи с применением аналитических методов воспользуемся примером, ставшим классическим, — реализуем итерационный метод Ньютона при решении нелинейного уравнения вида $f(x) = 0$.

Как известно, метод Ньютона сводится к итерационным вычислениям по следующей формуле:

$x_{i+1} = x_i + f(x_i)/f'(x_i)$.

Реализующая его процедура выглядит довольно просто:

```
> restart:
 NI:=proc(expr,x)
 local iter;
 iter:=x-expr/diff(expr,x);
 unapply(iter,x)
 end;
```

$NI := \mathbf{proc}(expr, x)\ \mathbf{local}\ iter;\ iter := x - expr/\mathrm{diff}(expr, x);\ \mathrm{unapply}(iter, x)\ \mathbf{end\ proc}$

Для получения итерационной формулы в аналитическом виде здесь используется функция `unapply`. Теперь, если задать решаемое уравнение, то можно получить искомое аналитическое выражение:

```
> expr:=sin(x)^2-0.5;
```

$$expr := \sin(x)^2 - .5$$

```
> F:=NI(expr,x);
```

$$F := x \to x - \frac{1}{2}\frac{\sin(x)^2 - .5}{\sin(x)\cos(x)}$$

Далее, задав начальное приближение для x в виде $x = x_0$, можно получить результаты вычислений для ряда итераций:

```
> x0:=0.2;
```

$$x0 := .2$$

```
> to 8 do x0:=F(x0);od;
```

$$x0 := 1.382611210$$

$$x0 := .117460944$$

$$x0 := 2.206529505$$

$$x0 := 2.360830634$$

$$x0 := 2.356194357$$

$$x0 := 2.356194490$$

$$x0 := 2.356194490$$

$$x0 := 2.356194490$$

Нетрудно заметить, что, испытав скачок в начале решения, значения x довольно быстро сходятся к конечному результату, дающему корень заданной функции. Последние три итерации дают одно и то же значение x. Заметим, что этот метод дает только одно решение, даже если корней несколько. Вычислить другие корни в таком случае можно, изменив начальное условие.

Можно попробовать с помощью полученной процедуры получить решение и для другой функции:

```
> expr:=ln(x^2)-0.5;
```

$$expr := \ln(x^2) - .5$$

```
> F:=NI(expr,x);
```

$$F := x \rightarrow x - \frac{1}{2}(\ln(x^2) - .5)\,x$$

```
> x0:=0.2;
```

$$x0 := .2$$

```
> to 8 do x0:=F(x0);od;
```

$$x0 := .5718875825$$

$$x0 := 1.034437603$$

$$x0 := 1.258023119$$

$$x0 := 1.283760340$$

$$x0 := 1.284025389$$

$$x0 := 1.284025417$$

$$x0 := 1.284025416$$

$$x0 := 1.284025417$$

Здесь итерационная формула имеет (и вполне естественно) уже другой вид, но сходимость к корню также обеспечивается за несколько итераций.

Возможна и иная форма задания итерационной процедуры с применением оператора дифференцирования D и заданием исходной функции также в виде процедуры:

```
> MI:=proc(f::procedure)
> (x->x)-eval(f)/D(eval(f));
> end;
```

$$MI := \text{proc}(f::procedure)\ (x \to x) - \text{eval}(f)/\text{D}(\text{eval}(f))\ \text{end proc}$$

```
> g:=x->x-cos(x);
```

$$g := x \to x - \cos(x)$$

```
> SI:=MI(g);
```

$$SI := (x \to x) - \frac{x \to x - \cos(x)}{x \to 1 + \sin(x)}$$

```
> x0:=0.1;
```

$$x0 := .1$$

```
> to 6 do x0:=SI(x0) od;
```

$$x0 := .9137633858$$

$$x0 := .7446642419$$

$$x0 := .7390919660$$

$$x0 := .7390851333$$

$$x0 := .7390851332$$

$$x0 := .7390851332$$

Вообще говоря, в программных процедурах можно использовать любые операторы и функции, присущие Maple-языку, в том числе и те, которые реализуют символьные вычисления. Это открывает широкий простор для разработки новых процедур и функций, обеспечивающих выполнение символьных операций.

Вычисление интеграла по известной формуле

Рассмотрим следующий пример:

```
> Int(e^x*x^n,x)=int(e^x*x^n,x);
```

$$\int e^x x^n\,dx = -(-1)^{(-n)} \ln(e)^{(-1-n)} (x^n (-1)^n \ln(e)^n n\,\Gamma(n) (-x \ln(e))^{(-n)}$$
$$- x^n (-1)^n \ln(e)^n e^{(x \ln(e))} - x^n (-1)^n \ln(e)^n n (-x \ln(e))^{(-n)} \Gamma(n, -x \ln(e)))$$

Прежние версии системы Maple не брали этот интеграл, поскольку он не имеет аналитического представления через обычные функции. Maple 7 блестяще вычисляет этот «крепкий орешек», но полученное выражение довольно сложно.

Из математики известно, что такой интеграл может быть представлен в следующем виде:

$$\int x^n e^x dx = n! e^x \sum_{i=0}^{n} (-1)^{n-i} \frac{x^i}{i!}$$

Используя эту формулу, мы можем создать простую процедуру для численного и аналитического вычисления данного интеграла:

```
> IntExpMonomial:=proc(n::anything,x::name)
 local i;
 n!*exp(x)*sum(((-1)^(n-i)*x^i)/i!,i=0..n);
 end;
```

IntExpMonomial := **proc** (*n*::*anything*, *x*::*name*)
local *i*;
n!×exp(*x*)×sum((-1)^(*n* − *i*)×*x*^*i*/*i*!, *i* = 0 .. *n*)
end proc

Проверим ее в работе:

```
> IntExpMonomial(3,x);
```

$$6\,e^x\left(-1+x-\frac{1}{2}x^2+\frac{1}{6}x^3\right)$$

```
> IntExpMonomial(5,x);
```

$$120\,e^x\left(-1+x-\frac{1}{2}x^2+\frac{1}{6}x^3-\frac{1}{24}x^4+\frac{1}{120}x^5\right)$$

```
> IntExpMonomial(n,x);
```

$$n!\,e^x\left((-1)^n e^{(-x)} + \frac{x^{(n+1)}\left((-x)^{(-1-n)} e^{(-x)}\,\Gamma(2+n)-(n+1)(-x)^{(-1-n)} e^{(-x)}\,\Gamma(n+1,-x)\right)}{(n+1)!}\right)$$

Результат в аналитическом виде довольно прост для данного интеграла с конкретным значением *n*. Более того, мы получили несколько иной результат и для *n* в общем случае. Но точен ли он? Для ответа на этот вопрос продифференцируем полученное выражение:

```
> diff(%,x);
```

$$n!\,e^x\left((-1)^n e^{(-x)} + \frac{x^{(n+1)}\left((-x)^{(-1-n)} e^{(-x)}\,\Gamma(2+n)-(n+1)(-x)^{(-1-n)} e^{(-x)}\,\Gamma(n+1,-x)\right)}{(n+1)!}\right) + n!\,e^x$$

$$\left(-(-1)^n e^{(-x)} + \frac{x^{(n+1)}(n+1)((-x)^{(-1-n)} e^{(-x)}\Gamma(2+n) - (n+1)(-x)^{(-1-n)} e^{(-x)}\Gamma(n+1,-x))}{x(n+1)!} + x^{(n+1)}\left(\frac{(-x)^{(-1-n)}(-1-n)e^{(-x)}\Gamma(2+n)}{x} - (-x)^{(-1-n)} e^{(-x)}\Gamma(2+n) - \frac{(n+1)(-x)^{(-1-n)}(-1-n)e^{(-x)}\Gamma(n+1,-x)}{x}\right.\right.$$

$$\left.\left. + (n+1)(-x)^{(-1-n)} e^{(-x)}\Gamma(n+1,-x) - (n+1)(-x)^{(-1-n)} e^{(-x)}(-x)^n e^x\right)/(n+1)!\right)$$

Результат дифференцирования выглядит куда сложнее, чем вычисленный интеграл. Однако с помощью функции `simplify` он упрощается к подынтегральной функции:

```
> simplify(%);
```

$$e^x x^n$$

Это говорит о том, что задача вычисления заданного интеграла в аналитической форме действительно решена. А что касается громоздкости результатов, так ведь системы, подобные Maple 7, для того и созданы, чтобы облегчить нам работу с громоздкими вычислениями — в том числе аналитическими.

Вложенные процедуры и интегрирование по частям

Теперь мы подошли к важному моменту, о котором читатель наверняка уже давно догадался — в составляемых пользователем процедурах можно использовать ранее составленные им (или кем-то еще) другие процедуры! Таким образом, Maple-язык позволяет реализовать процедуры, вложенные друг в друга.

Для иллюстрации применения вложенных процедур рассмотрим операцию интегрирования по частям. Пусть нам надо вычислить интеграл:

$$\int p(x)e^x dx.$$

где $p(x)$ — выражение, представляющее полином.

Приведенный ниже пример подготовлен в реализации Maple 7 [38]. Вначале подготовим процедуру `IntExpMonomialR`, реализующую вычисление уже рассмотренного ранее интеграла, но рекурсивным способом:

```
> IntExpMonomialR:=proc(n::nonnegint,x::name)
 local i;if n=0 then RETURN(exp(x)) fi;
 x^n*exp(x)-n*IntExpMonomialR(n-1,x);
 end;
```

IntExpMonomialR := **proc** (*n*::*nonnegint*, *x*::*name*)
local *i*;
    if *n* = 0 **then** RETURN(exp(*x*)) **end if**;
    x^*n*×exp(*x*) − *n*×IntExpMonomialR(*n* − 1, *x*)
end proc

Проверим ее в работе:

```
> IntExpMonomialR(4,x);
```

$$x^4 e^x - 4 x^3 e^x + 12 x^2 e^x - 24 x e^x + 24 e^x$$

```
> collect(%,exp(x));
```

$$(x^4 - 4 x^3 + 12 x^2 - 24 x + 24) e^x$$

Теперь составим процедуру для вычисления по частям нашего интеграла:

```
> IntExpPolynomial:=proc(p::polynom,x::name)
 local i,result;
 #### WARNING: degree(0,x) now returns -infinity
 result:=add(coeff(p,x,i)*IntExpMonomialR(i,x),i=0..degree(p,x));
 collect(result,exp(x));
 end;
```

IntExpPolynomial := **proc** (*p*::*polynom*, *x*::*name*)
local *i*, *result*;
    result := add(coeff(*p*, *x*, *i*)×IntExpMonomialR(*i*, *x*), *i* = 0 .. degree(*p*, *x*));
    collect(*result*, exp(*x*))
end proc

В этой процедуре имеется обращение к ранее составленной процедуре `IntExpMonomialR`. Обратите внимание на то, что в процедуре введено предупреждение об определенных проблемах, связанных с использованием функции `degree` (сообщение начинается с символов ####). Тем не менее процедура работает, в чем убеждают по крайней мере следующие примеры:

```
> p:=(x^2+1)*(1-3*x);
```

$$p := (x^2 + 1)(1 - 3x)$$

```
> expand(p);
```

$$x^2 - 3 x^3 + 1 - 3 x$$

```
> int(p*exp(x),x);
```

$$10 x^2 e^x - 23 x e^x + 24 e^x - 3 x^3 e^x$$

```
> IntExpPolynomial(p,x);
```

$$(10 x^2 - 23 x + 24 - 3 x^3) e^x$$

В заключение остается отметить, что данный пример в Maple V R4 дает неточный результат, хотя никаких сообщений об ошибках не выводится.

Что нового мы узнали?

В этом уроке мы научились:

- Осуществлять основные операции с выражениями.
- Выполнять приложения и подстановки.
- Упрощать и расширять выражения.
- Осуществлять факторизацию выражений.
- Выполнять комплектование выражений по степеням.
- Программировать некоторые символьные операции.

Типовые средства построения графиков

- Основные возможности построения двумерных графиков
- Основная функция построения двумерных графиков — plot
- Задание координатных систем двумерных графиков
- Управление цветом и стилем двумерных графиков
- Основные типы двумерных графиков
- Трехмерные графики
- Построение поверхностей
- Быстрое построение графиков
- Специальные приемы построения трехмерных графиков
- Структуры двумерной и трехмерной графики

Введение в построение двумерных графиков

Основные возможности двумерной графики

Maple 7 реализует все мыслимые (и даже «немыслимые») варианты математических графиков. Строятся как графики простых функций в декартовой и полярной системах координат, так и графики, показывающие реалистические образы сложных, пересекающихся в пространстве фигур с их функциональной окраской. Возможны наглядные графические иллюстрации решений самых разнообразных уравнений, включая системы дифференциальных уравнений.

В само ядро Maple 7 встроено ограниченное число функций построения графиков. Это прежде всего функция для построения двумерных графиков `plot` и функция для построения трехмерных графиков `plot3d`. Они позволяют строить графики *наиболее распространенных типов. Для построения специальных графиков* (например, векторных полей градиентов, решения дифференциальных уравнений, построения фазовых портретов и т. д.) в пакеты системы Maple 7 включено большое число различных графических функций. Для их вызова необходимы соответствующие указания.

Вообще говоря, средства для построения графиков в большинстве языков программирования принято считать графическими *процедурами*, или *операторами*. Однако мы сохраним за ними наименование *функций*, в силу двух принципиально важных свойств:

- графические средства Maple V *возвращают* некоторые графические объекты, которые размещаются в окне документа — в строке вывода или в отдельном графическом объекте;
- эти объекты можно использовать в качестве значений переменных, то есть переменным можно присваивать значения графических объектов и выполнять над ними соответствующие операции (например, с помощью функции `show` выводить на экран несколько графиков).

Графические функции заданы таким образом, что обеспечивают построение типовых графиков без какой-либо особой подготовки. Для этого нужно лишь указать функцию, график которой строится, и пределы изменения независимых переменных. Однако с помощью дополнительных необязательных параметров можно существенно изменить вид графиков — например, настроить стиль и цвет линий, вывести титульную надпись, изменить вид координатных осей и т. д.

Основная функция построения двумерных графиков plot

В математике широко используются зависимости вида $f(x)$ или $y(x)$. Их графики строятся на плоскости в виде ряда точек $y_i(x_i)$, обычно соединяемых отрезками прямых. Таким образом, используется кусочно-линейная интерполяция двумерных графиков. Если число точек графика достаточно велико (десятки или сотни), то приближенность построения не очень заметна.

Для построения двумерных графиков служит функция `plot`. Она задается в виде:

```
plot(f, h, v)
plot(f, h, v, o)
```

где `f` — визуализируемая функция (или функции), `h` — переменная с указанием области ее изменения, `v` — необязательная переменная с указанием области изменения, `o` — параметр или набор параметров, задающих стиль построения графика (толщину и цвет кривых, тип кривых, метки на них и т. д.).

Самыми простыми формами задания этой функции являются следующие:

- `plot(f,xmin..xmax)` — построение графика функции f, заданной только своим именем;
- `plot(f(x),x=xmin..xmax)` — построение графика функции $f(x)$.

Диапазон изменения независимой переменной `x` задается как `xmin..xmax`, где `xmin` и `xmax` — минимальное и максимальное значение `x`, `..` (две точки) — составной символ, указывающий на изменение независимой переменной. Разумеется, имя `x` здесь дано условно — независимая переменная может иметь любое допустимое имя.

Помимо построения самой кривой $y(x)$ или $f(x)$ необходимо задать ряд других свойств графиков, например вывод координатных осей, тип и цвет линий графика и др. Это достигается применением параметров графика — специальных указаний для Maple. Графики обычно (хотя и не всегда) строятся сразу в достаточно приемлемом виде. Это достигается тем, что многие параметры задаются по умолчанию и пользователь, по крайней мере начинающий, может о них ничего не знать. Однако язык общения и программирования Maple 7 позволяет задавать управляющие параметры и в явном виде.

Для двумерного графика возможны следующие параметры:

- `adaptive` — включение адаптивного алгоритма построения графиков (детали см. ниже);
- `axes` — вывод различных типов координат (`axes=NORMAL` — обычные оси, выводятся по умолчанию, `axes=BOXES` — график заключается в рамку с осями-шкалами, `axes=FRAME` — оси в виде перекрещенных линий, `axes=NONE` — оси не выводятся);
- `axesfont` — задание шрифтов для подписи делений на координатных осях (см. также параметр `font`);
- `color` — задает цвет кривых (см. далее);

- coords — задание типа координатной системы (см. далее);
- discont — задает построение непрерывного графика (значения true или false);
- filled — при filled=true задает окраску цветом, заданным параметром color, для области, ограниченной построенной линией и горизонтальной координатной осью x;
- font — задание шрифта в виде [семейство, стиль, размер];
- labels — задание надписей по координатным осям в виде [X, Y], где X и Y — надписи по осям x и y графика;
- labeldirections — задает направление надписей по осям [X, Y], где X и Y может иметь строковые значения HORISONTAL (горизонтально) и VERTICAL (вертикально);
- labelfont — задает тип шрифта подписей (см. font);
- legend — задает вывод легенды (обозначения кривых);
- linestyle — задание стиля линий (1 — сплошная, 2 — точками, 3 — пунктиром и 4 — штрихпунктиром);
- numpoints — задает минимальное количество точек на графике (по умолчанию numpoints=49);
- resolutions — задает горизонтальное разрешение устройства вывода (по умолчанию resolutions=200, параметр используется при отключенном адаптивном методе построения графиков);
- sample — задает список параметров для предварительного представления кривых;
- scaling — задает масштаб графика: CONSTRAINED (сжатый) или UNCONSTRAINED (несжатый — по умолчанию);
- size — задает размер шрифта в пунктах;
- style — задает стиль построения графика (POINT — точечный, LINE — линиями);
- symbol — задает вид символа для точек графика (возможны значения BOX — прямоугольник, CROSS — крест, CIRCLE — окружность, POINT — точка, DIAMOND — ромб);
- symbolsize — установка размеров символов для точек графика (в пунктах, по умолчанию 10);
- title — задает построение заголовка графика (title="string", где string — строка);
- titlefont — определяет шрифт для заголовка (см. font);
- thickness — определяет толщину линий графиков (0, 1, 2, 3, значение по умолчанию — 0);
- view=[A, B] — определяет максимальные и минимальные координаты, в пределах которых график будет отображаться на экране, A = [xmin..xmax], B=[ymin..ymax] (по умолчанию отображается вся кривая);

- xtickmarks — задает минимальное число отметок по оси x;
- ytickmarks — задает минимальное число отметок по оси y.

В основном задание параметров особых трудностей не вызывает, за исключением задания титульной надписи с выбором шрифтов по умолчанию — в этом случае не всегда поддерживается вывод символов кириллицы (русского языка). Подбором подходящего шрифта эту проблему удается решить. Модификация графиков с помощью управляющих параметров подробно рассматривается ниже.

Специальный параметр adaptive задает работу специального адаптивного алгоритма для построения графиков наилучшего вида. При этом Maple автоматически учитывает кривизну изменения графика и увеличивает число отрезков прямых в тех частях графиков, где их ход заметно отличается от интерполирующей прямой. При задании adaptive=false адаптивный алгоритм построения графиков отключается, а при adaptive=true включается (значение по умолчанию).

Задание координатных систем двумерных графиков

В версии Maple 7 параметр coords задает 15 типов координатных систем для двумерных графиков. По умолчанию используется прямоугольная (декартова) система координат (coords=cartesian). При использовании других координатных систем координаты точек для них (u, v) преобразуются в координаты (x, y) как (u, v) --> (x, y). Ниже приведены наименования систем координат (значений параметра coords) и соответствующие формулы преобразования.

```
bipolar:
                x = sinh(v)/(cosh(v)-cos(u))
                y = sin(u)/(cosh(v)-cos(u))
cardioid:
                x = 1/2*(u^2-v^2)/(u^2+v^2)^2
                y = u*v/(u^2+v^2)^2
cartesian:
                x = u
                y = v
cassinian:
                x = a*2^(1/2)/2*((exp(2*u)+2*exp(u)*cos(v)+1)^(1/2) +
                                                        exp(u)*cos(v)+1)^(1/2)
                y = a*2^(1/2)/2*((exp(2*u)+2*exp(u)*cos(v)+1)^(1/2) -
                                                        exp(u)*cos(v)-1)^(1/2)]
elliptic:
                x = cosh(u)*cos(v)
                y = sinh(u)*sin(v)
hyperbolic:
                x = ((u^2+v^2)^(1/2)+u)^(1/2)
                y = ((u^2+v^2)^(1/2)-u)^(1/2)
invcassinian:
                x = a*2^(1/2)/2*((exp(2*u)+2*exp(u)*cos(v)+1)^(1/2) +
                            exp(u)*cos(v)+1)^(1/2)/(exp(2*u)+2*exp(u)*cos(v)+1)^(1/2)
                y = a*2^(1/2)/2*((exp(2*u)+2*exp(u)*cos(v)+1)^(1/2) -
                            exp(u)*cos(v)-1)^(1/2)/(exp(2*u)+2*exp(u)*cos(v)+1)^(1/2)
invelliptic:
                x = a*cosh(u)*cos(v)/(cosh(u)^2-sin(v)^2)
                y = a*sinh(u)*sin(v)/(cosh(u)^2-sin(v)^2)
logarithmic:
```

```
                x = a/Pi*ln(u^2+v^2)
                y = 2*a/Pi*arctan(v/u)
logcosh:
                x = a/Pi*ln(cosh(u)^2-sin(v)^2)
                y = 2*a/Pi*arctan(tanh(u)*tan(v))
maxwell:
                x = a/Pi*(u+1+exp(u)*cos(v))
                y = a/Pi*(v+exp(u)*sin(v))
parabolic:
                x = (u^2-v^2)/2
                y = u*v
polar:
                x = u*cos(v)
                y = u*sin(v)
rose:
                x = ((u^2+v^2)^(1/2)+u)^(1/2)/(u^2+v^2)^(1/2)
                y = ((u^2+v^2)^(1/2)-u)^(1/2)/(u^2+v^2)^(1/2)
tangent:
                x = u/(u^2+v^2)
                y = v/(u^2+v^2)
```

Управление стилем и цветом линий двумерных графиков

Maple 7 позволяет воспроизводить на одном графике множество кривых. При этом возникает необходимость как-то идентифицировать их. Для этого можно использовать построение линий разными стилями, разными цветами и с разной толщиной. Набор средств выделения кривых позволяет уверенно различать их как на экране цветного дисплея и в распечатках, сделанных цветным струйным принтером, так и при печати монохромными принтерами.

Параметр `style` — позволяет задавать следующие стили для линий графиков:

- `POINT` или `point` — график выводится по точкам;
- `LINE` или `line` — график выводится линией.

Если задано построение графика точками, то параметр `symbol` позволяет представить точки в виде различных символов, например прямоугольников, крестов, окружностей или ромбов.

Другой параметр — `color` — позволяет использовать обширный набор цветов линий графиков:

aquamarine	black	blue	navy	coral
cyan	brown	gold	green	gray
grey	khaki	magenta	maroon	orange
pink	plum	red	sienna	tan
turquoise	violet	wheat	white	yellow

Различные цветовые оттенки получаются благодаря использованию RGB-комбинаций базовых цветов: `red` — красный, `gray` — зеленый, `blue` — синий. Приведем перевод ряда других составных цветов: `black` — черный, `white` — белый, `khaki` — цвет «хаки», `gold` — золотистый, `orange` — оранжевый, `violet` — фиолетовый, `yellow` — желтый и т. д. Перевод цветов некоторых оттенков на русский

язык не всегда однозначен и потому не приводится. Средства управления стилем графиков дают возможность легко выделять различные кривые на одном рисунке, даже если для выделения не используются цвета.

Основные типы двумерных графиков

Графики одной функции

При построении графика одной функции она записывается в явном виде на месте шаблона `f`. Примеры построения графика одной функции представлены на рис. 11.1. Обратите внимание на то, что график функции $\sin(x)/x$ строится без характерного провала в точке $x = 0$, который наблюдается при построении графиков этой функции многими программами. Он связан с используемым в них правилом — функция задается равной нулю, если ее числитель равен нулю. Данная функция в этой точке дает устранимую неопределенность `0/0->1`, что и учитывает графический процессор системы Maple 7.

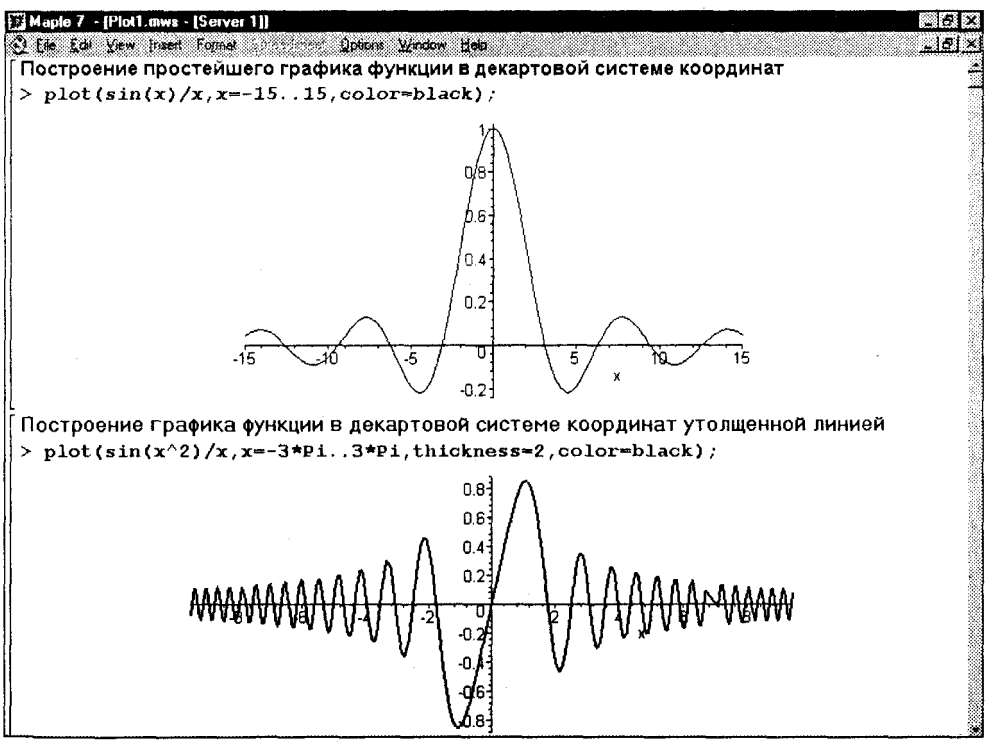


Рис. 11.1. Примеры построения графиков одной функции

При построении графиков одной функции могут быть введены описание диапазонов и различные параметры, например: для задания цвета кривой, толщины линии, которой строится график функции, и др. К примеру, запись в списке

параметров color=black задает вывод кривых черным цветом, а запись thikness=2 задает во втором примере рис. 11.1 построение графика линией, удвоенной по сравнению с обычной толщиной. Кстати говоря, запись color=red дает красный цвет, color=green — зеленый цвет, color=blue — синий цвет и т. д. При черно-белой печати цвета представляются оттенками серого цвета.

Управление диапазоном изменения переменной и значения функции

Для управления отображаемой на графике области служит задание диапазонов принимаемых значений для переменной и функции. В ряде случаев их можно не применять, тогда Maple автоматически задает приемлемые диапазоны. Однако их явное указание позволяет управлять областью графика вручную. Иногда соответствующее задание диапазонов случайно или целенаправленно ведет к отсечению части графика — например, на рис. 11.2 в первом примере отсечена верхняя часть графика.

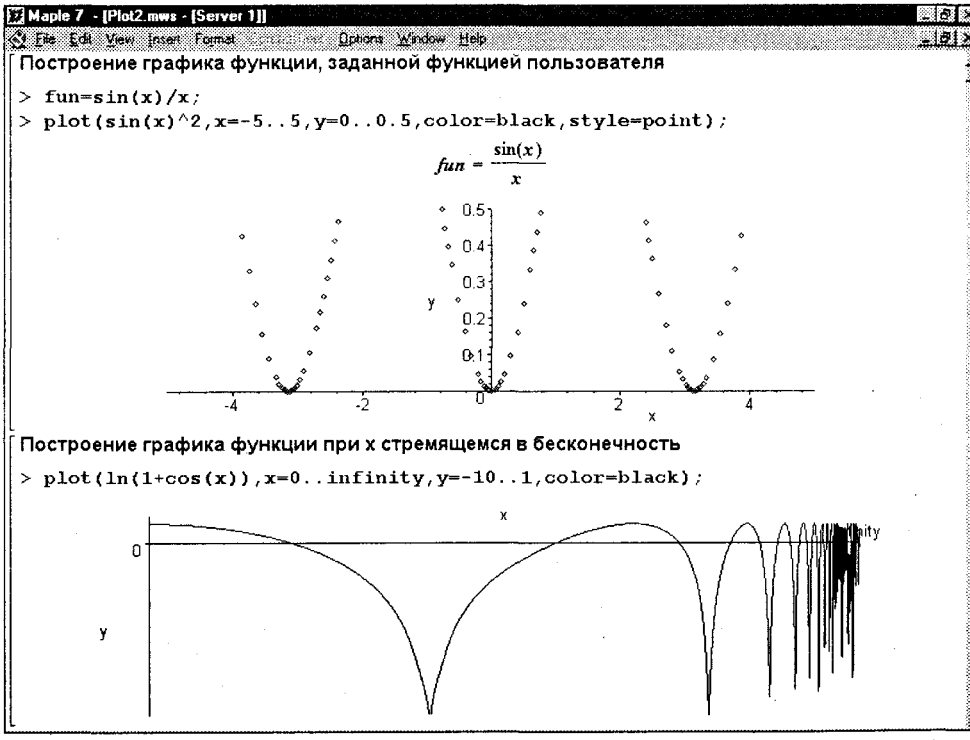


Рис. 11.2. Построение графиков функции с явным указанием масштаба

Правильный выбор диапазонов повышает представительность графиков функций. Рекомендуется вначале пробовать строить графики с автоматическим выбором диапазонов, а уже затем указывать их вручную.

Графики функций в неограниченном диапазоне

Изредка встречаются графики функций $f(x)$, которые надо построить при изменении значения x от нуля до бесконечности или даже от минус бесконечности до плюс бесконечности. Бесконечность в таких случаях задается как особая константа `infinity`. В этом случае переменной x, устремляющейся в бесконечность, откладывается значение $\arctan(x)$. Рисунок 11.2 (второй пример) иллюстрирует сказанное.

Графики функций с разрывами

Некоторые функции, например $\tan(x)$, имеют при определенных значениях x разрывы, причем случается, что значения функции в этом месте устремляются в бесконечность. Функция $\tan(x)$, к примеру, в точках разрывов устремляется к +? и –?. Построение графиков таких функций нередко дает плохо предсказуемые результаты. Графический процессор Maple 7 не всегда в состоянии определить оптимальный диапазон по оси ординат, а график функции выглядит весьма непредставительно, если не сказать безобразно (рис. 11.3, первый пример).

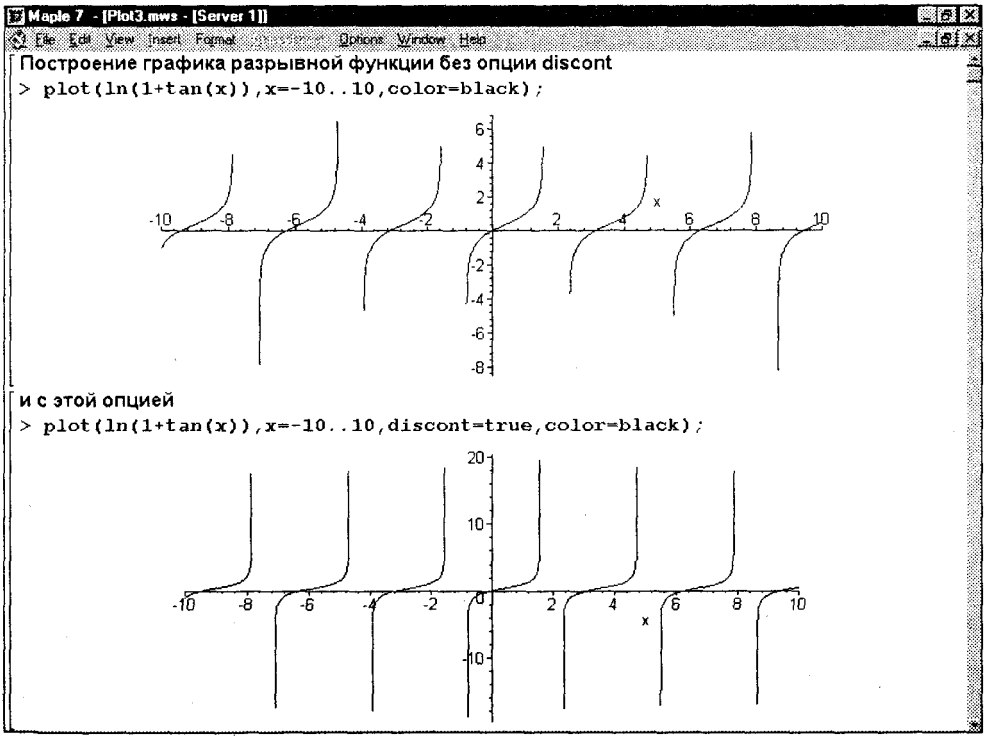


Рис. 11.3. Построение графиков функций с разрывами

Среди аргументов функции `plot` есть специальный параметр `discont`. Если задать его значение равным `true`, то качество графиков существенно улучшается,

см. второй пример на рис. 11.3. Улучшение достигается разбиением графика на несколько участков, на которых функция непрерывна, и более тщательным контролем за отображаемым диапазоном. При `discont=false` данный параметр отключен и строятся обычные графики.

ПРИМЕЧАНИЕ Следует отметить, что вид графика можно улучшить, просто задав диапазон по оси y (например, введя в параметры функции запись y=-10..10). При этом в точках разрыва могут появиться вертикальные линии. Иногда это бывает полезно.

Графики нескольких функций на одном рисунке

Важное значение имеет возможность построения на одном рисунке графиков нескольких функций. В простейшем случае (рис. 11.4, первый пример) для построения таких графиков достаточно перечислить нужные функции и установить для них общие интервалы изменения.

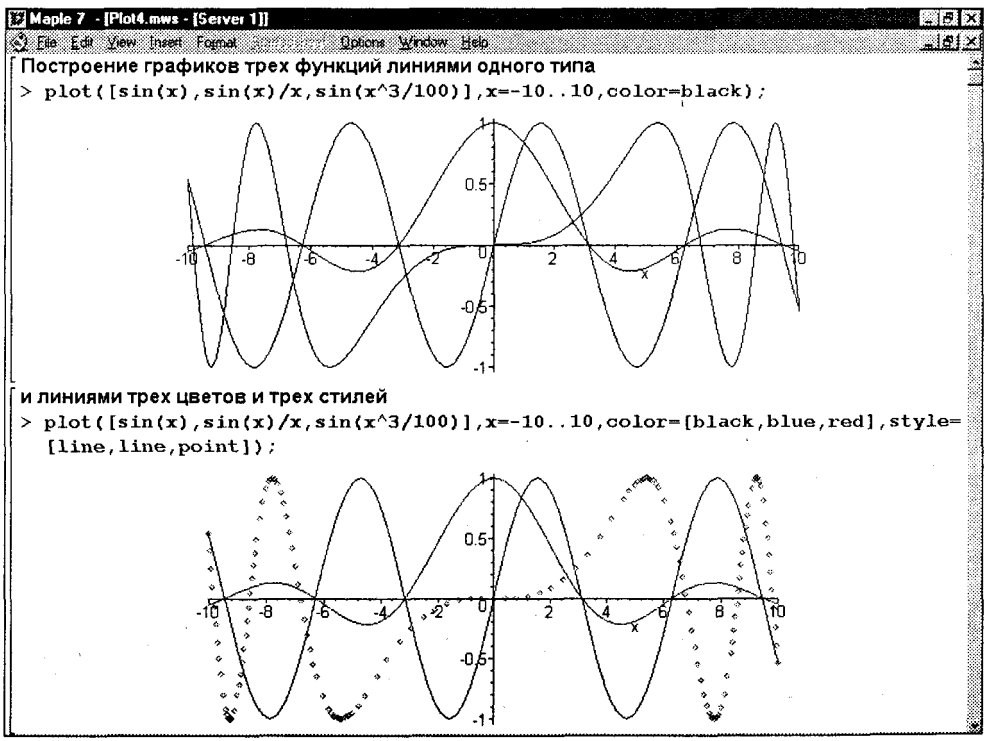


Рис. 11.4. Графики трех функций на одном рисунке

Обычно графики разных функций автоматически строятся разными цветами. Но это не всегда удовлетворяет пользователя — например, при распечатке графиков монохромным принтером некоторые кривые могут выглядеть слишком

блеклыми или даже не пропечататься вообще. Используя списки параметров `color` (цвет линий) и `style` (стиль линий), можно добиться выразительного выделения кривых — это показывает второй пример на рис. 11.4 для случая, когда линии графиков выделяются стилем. Однако если кривые задаются разным цветом, то при черно-белой печати они могут перестать различаться.

На рис. 11.5 показан еще один пример такого рода. Здесь построен график функции $\sin(x)/x$ и график ее полиномиальной аппроксимации. Она выполняется настолько просто, что соответствующие функции записаны прямо в списке параметров функции `plot`.

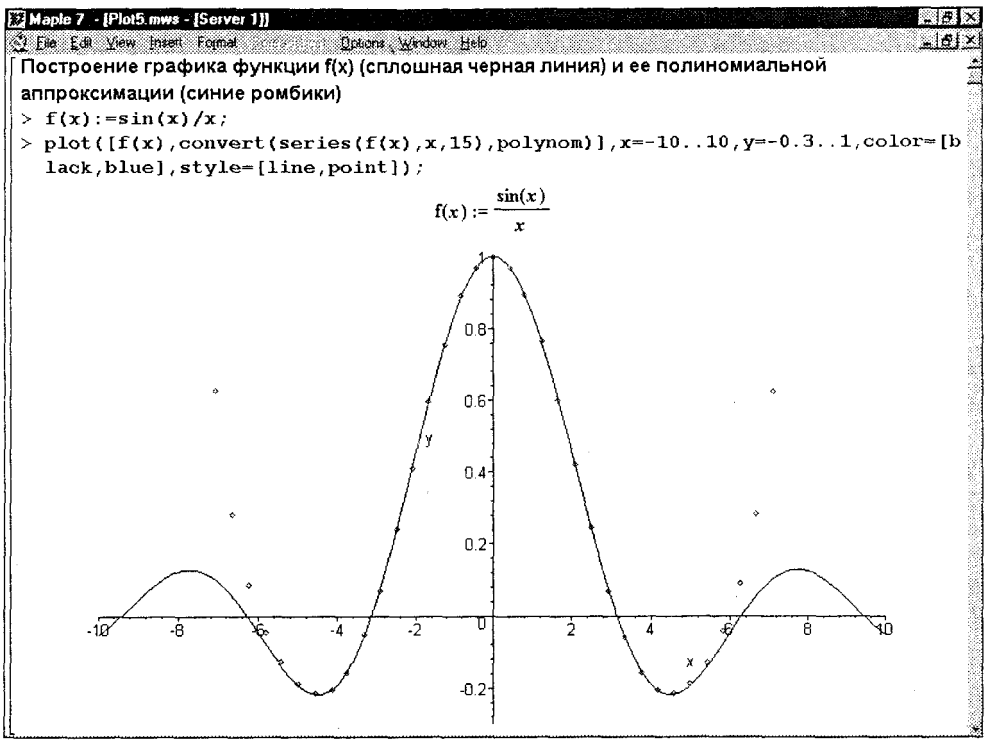


Рис. 11.5. График функции $\sin(x)/x$ и ее полиномиальной аппроксимации

В данном случае сама функция построена сплошной линией, а график полинома точками — ромбами. Хорошо видно, что при малых x аппроксимация дает высокую точность, но затем с ростом x ее погрешность резко возрастает.

Рисунок 11.6 показывает построение нескольких любопытных функций, полученных с помощью комбинаций элементарных функций. Такие комбинации позволяют получать периодические функции, моделирующие сигналы стандартного вида: в виде напряжения на выходе двухполупериодного выпрямителя, симметричных прямоугольных колебаний (меандр), пилообразных и треугольных импульсов, треугольных импульсов со скругленной вершиной.

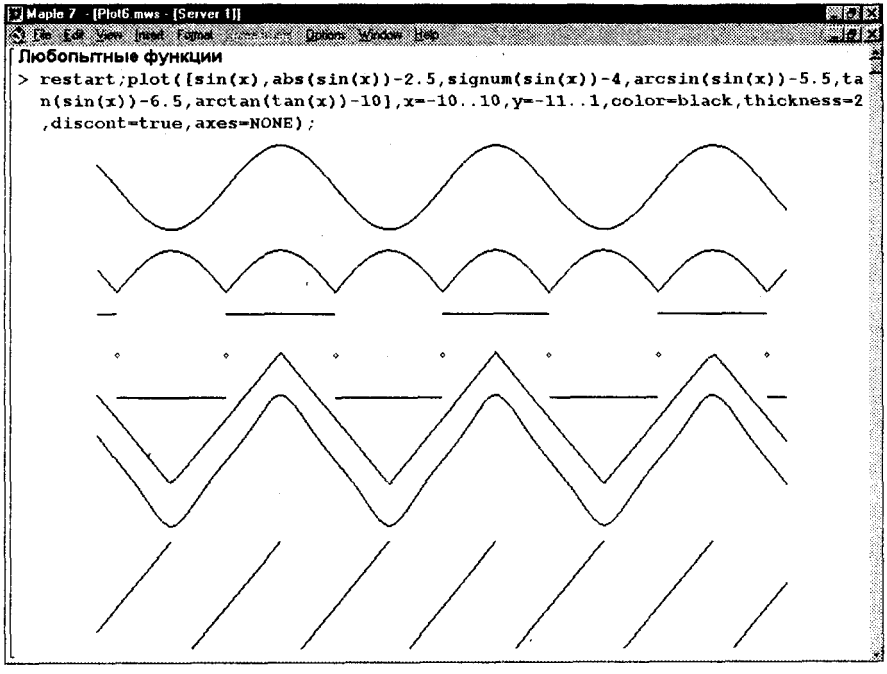


Рис. 11.6. Построение графиков нескольких любопытных функций

В этом рисунке запись `axes=NONE` убирает координатные оси. Обратите внимание, что смещение графиков отдельных функций вниз с целью устранения их наложения достигнуто просто прибавлением к значению каждой функции некоторой константы.

Графики функций, построенные точками

Показанный на рис. 11.5 график полинома, построенный ромбиками, не означает, что полином представлен отдельными точками. В данном случае просто выбран стиль линии в виде точек. Однако часто возникает необходимость построения графиков функций, которые представлены просто совокупностями точек. Такая совокупность может быть создана искусственно, как на рис. 11.7, либо просто задаваться списком координат x и значений функции.

В данном случае переменная `P` имеет вид списка, в котором попарно перечислены координаты точек функции $\sin(x)$. В этом нетрудно убедиться, заменив знак «:» после выражения, задающего `P`, на знак «;». Далее по списку `P` построен график точек в виде крестиков, которые отображают отдельные значения функции $\sin(x)$.

На рис. 11.8 показано построение графиков функций по точкам при явном задании функции списком координат ее отдельных точек. В первом примере эти точки соединяются отрезками прямых, так что получается кусочно-линейный график. Видно также, что указание типа точек после указания стиля линии игнорируется (а жаль, было бы неплохо, чтобы наряду с кусочно-линейной линией графика строились и выделенные окружностями точки).

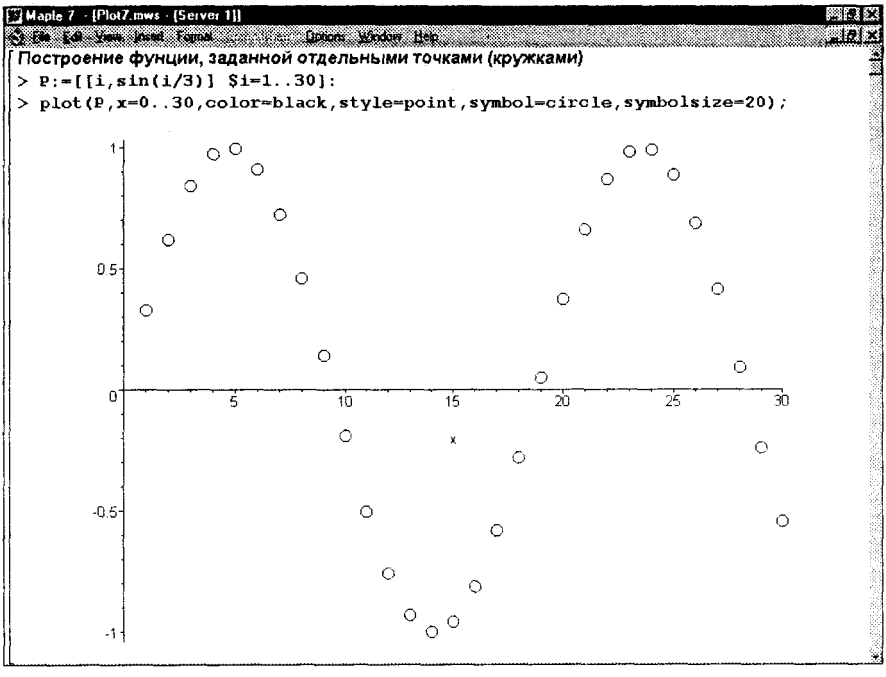


Рис. 11.7. Формирование списка отдельных точек функции и их построение на графике

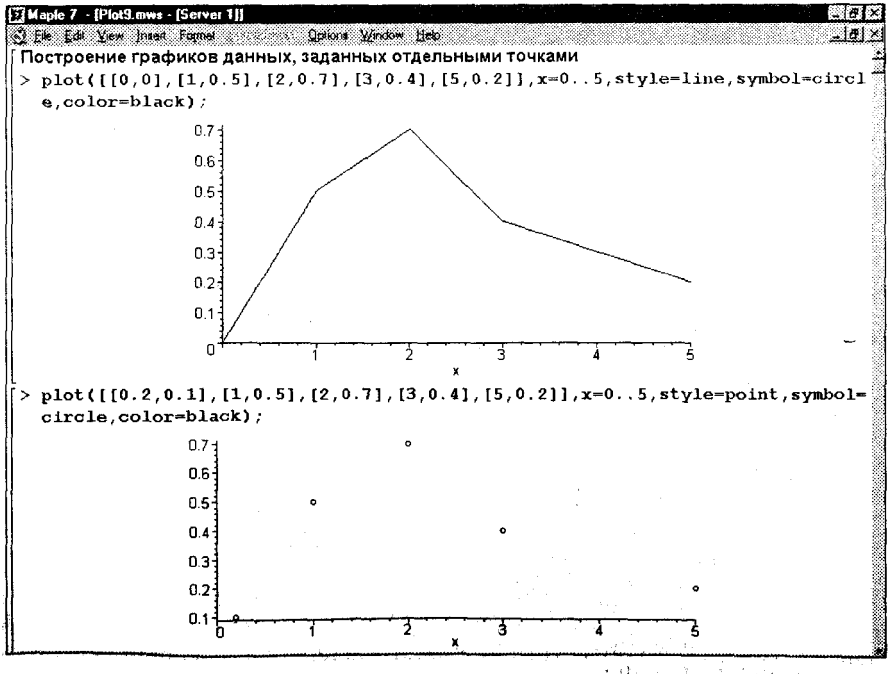


Рис. 11.8. Построение графика функции, явно заданной отдельными точками

Во втором примере рис. 11.8 показано построение только точек заданной функциональной зависимости. Они представлены маленькими кружками. Читателю предлагается самостоятельно совместить оба подхода к построению графиков по точкам и создать график в виде отрезков прямых, соединяющих заданные точки функции, представленные кружками или крестиками.

Графики функций, заданных своими именами

Способность Maple 7 к упрощению работы пользователя просто поразительна — жаль только, что многие возможности этого становятся ясными после основательного изучения программы, на что уходят, увы, не дни, а месяцы, а то и годы. Применительно к графикам одной из таких возможностей является построение графиков функций, заданных только их функциональными именами — даже без указания параметров в круглых скобках. Такую возможность наглядно демонстрирует рис. 11.9.

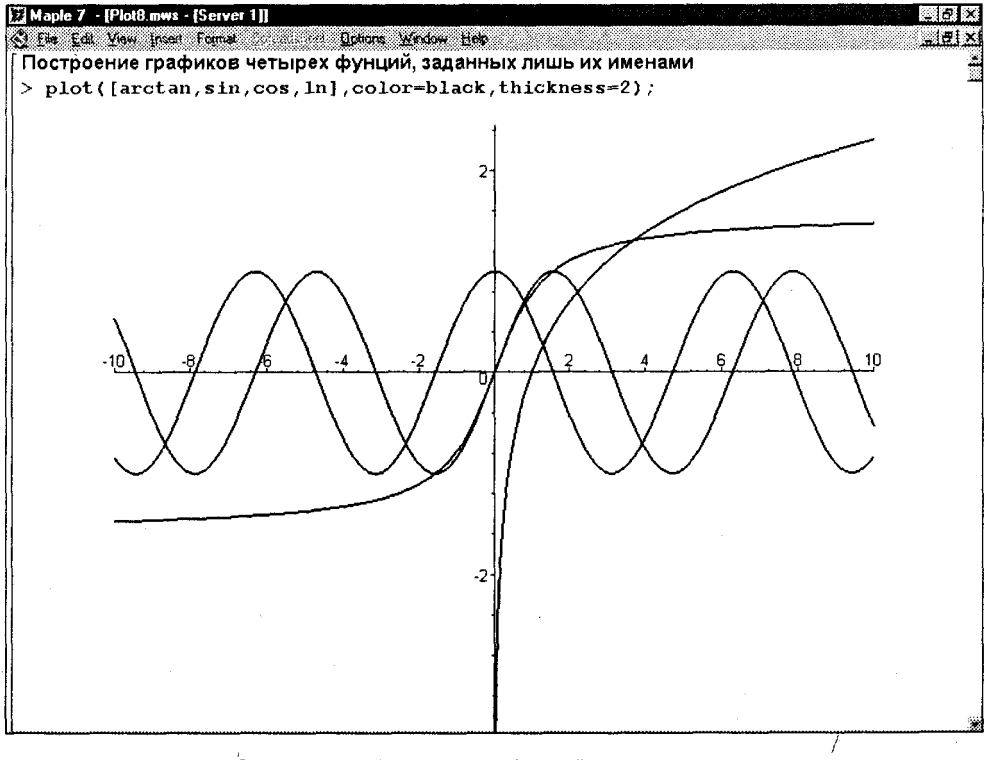


Рис. 11.9. Построение графиков четырех функций, заданных только их именами

Этот пример показывает, что возможно построение графиков функций даже без указания в команде `plot` диапазонов. При этом диапазон по горизонтальной оси устанавливается равным по умолчанию `-10..10`, а по вертикальной оси выбирается автоматически в соответствии с экстремальными значениями функций в указанном диапазоне изменения независимой переменной (условно *x*).

Графики функций с ординатами, заданными вектором

Часто возникает необходимость построения графика точек, ординаты которых являются элементами некоторого вектора. Обычно при этом предполагается равномерное расположение точек по горизонтальной оси. Пример построения такого графика дан на рис. 11.10.

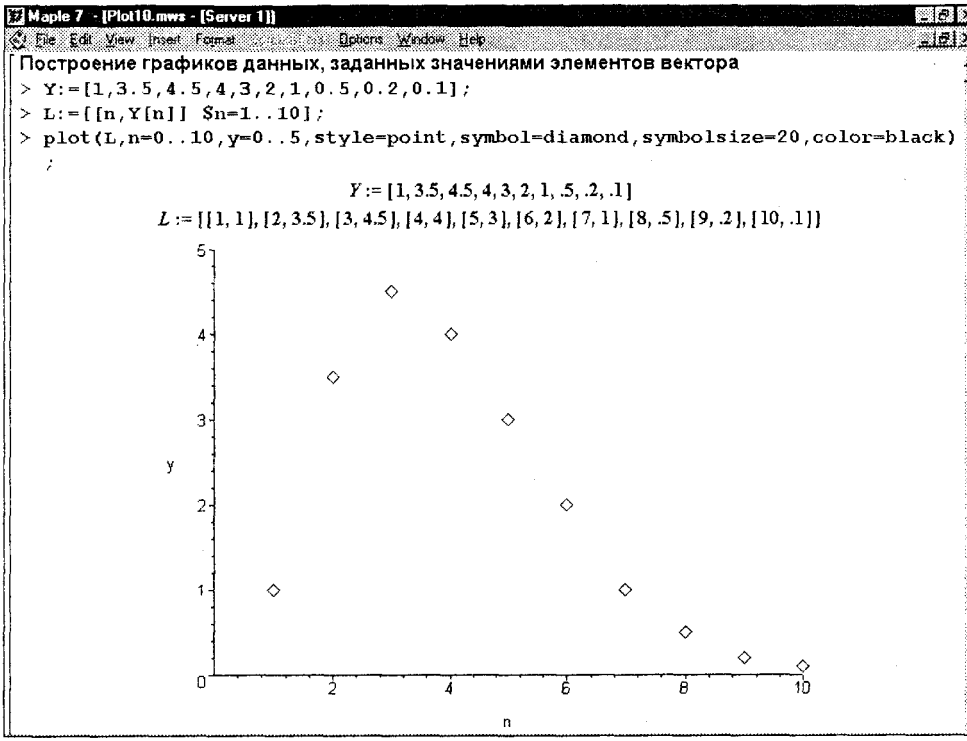


Рис. 11.10. Построение графика точек с ординатами, заданными элементами вектора

Из этого примера нетрудно заметить, что данная задача решается составлением списка парных значений координат исходных точек — к значениям ординат точек, взятых из вектора, добавляются значения абсцисс. Они задаются чисто условно, поскольку никакой информации об абсциссах точек в исходном векторе нет, так что фактически строится график зависимости ординат точек от их порядкового номера n.

Графики функций, заданных процедурами

Некоторые виды функций, например кусочные, удобно задавать процедурами. Построение графиков функций, заданных процедурами, не вызывает никаких трудностей и иллюстрируется рис. 11.11.

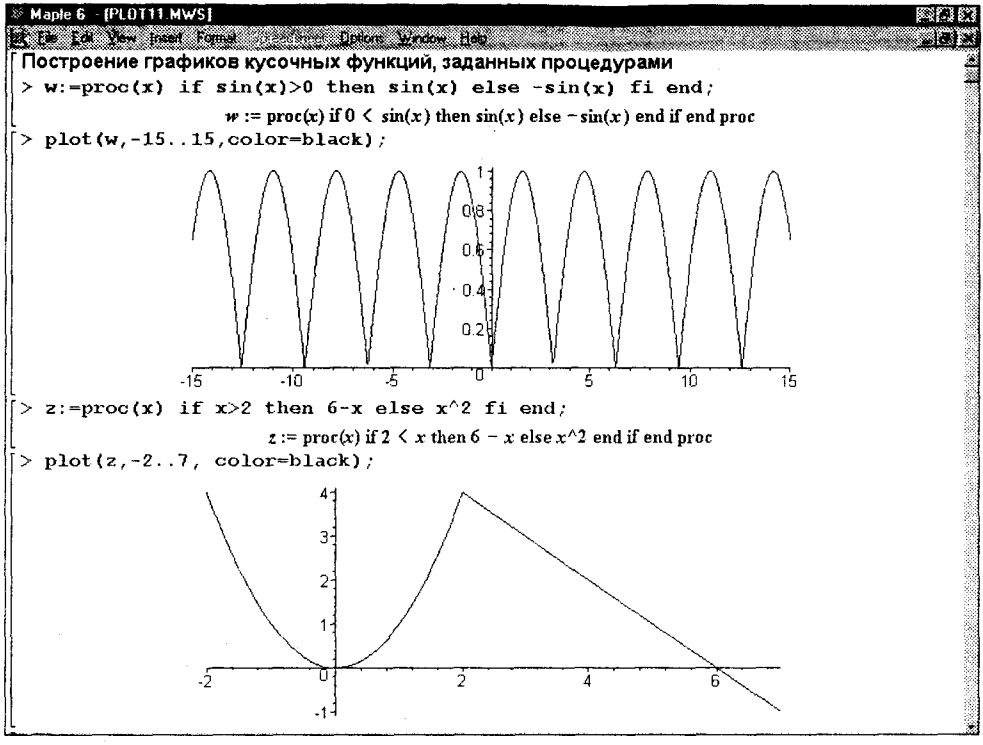


Рис. 11.11. Построение графика функций, заданных процедурами

Здесь, пожалуй, полезно обратить внимание на то, что в функции `plot` указывается имя процедуры без списка ее параметров.

Графики функций, заданных функциональными операторами

Еще одна «экзотическая» возможность функции `plot` — построение графиков функций, заданных функциональными операторами. Она иллюстрируется рис. 11.12.

Имена функций (без указания списка параметров в круглых скобках) тоже, по существу, являются функциональными операторами. Так что они также могут использоваться при построении графиков упрощенными способами.

Графики функций, заданных параметрически

В ряде случаев для задания функциональных зависимостей используются заданные параметрически уравнения, например $x = f_1(t)$ и $y = f_2(t)$ при изменении переменной t в некоторых пределах. Точки (x, y) наносятся на график в декартовой системе координат и соединяются отрезками прямых. Для этого используется функция `plot` в следующей форме:

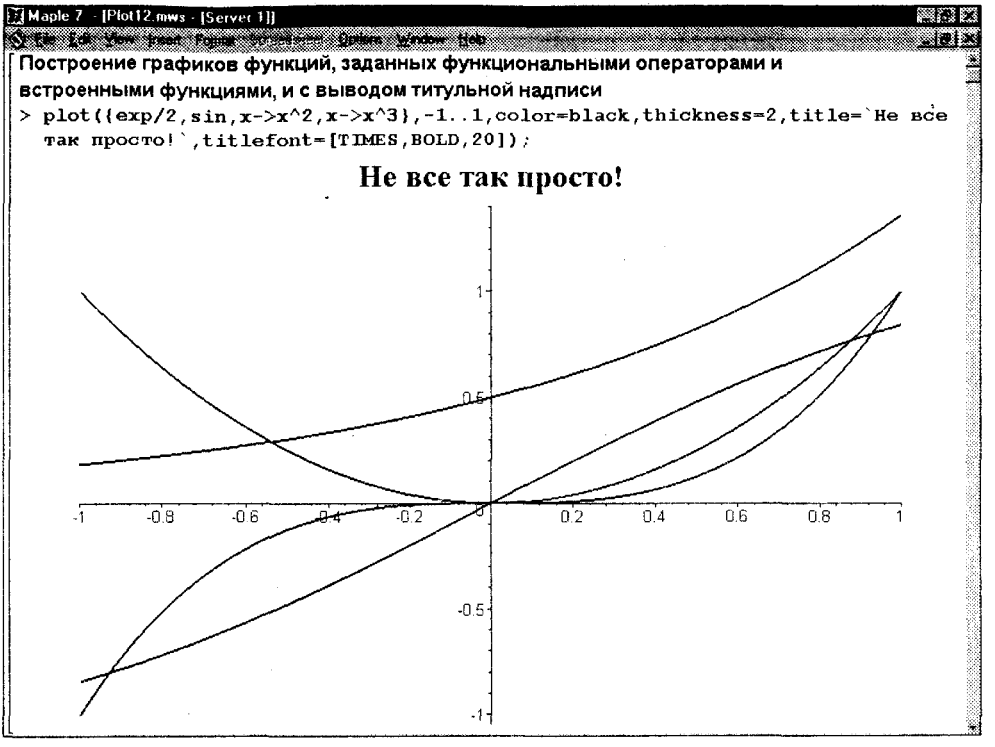


Рис. 11.12. Построение графиков функции, заданной функциональными операторами

```
plot([f1(t),f2(t),t=tmin..tmax],h,v,p)
```

Если функции $f_1(t)$ и $f_2(t)$ содержат периодические функции (например, тригонометрические), то для получения замкнутых фигур диапазон изменения переменной t обычно задается равным `0..2*Pi` или `-Pi..Pi`. К примеру, если задать в качестве функций $f_1(t)$ и $f_2(t)$ функции $\sin(t)$ и $\cos(t)$, то будет получен график окружности. Рисунок 11.13 показывает другие, чуть менее тривиальные примеры построения графиков такого рода.

Задание диапазонов для изменений `h` и `v`, а также параметров `p` не обязательно. Но, как и ранее, они позволяют получить вид графика, удовлетворяющий всем требованиям пользователя.

Графики функций в полярной системе координат

Графики в полярной системе координат представляют собой линии, которые описывают конец радиус-вектора $\mathbf{r}(t)$ при изменении угла t в определенных пределах — от t_{min} до t_{max}. Построение таких графиков также производится функцией `plot`, которая для этого записывается в следующем виде:

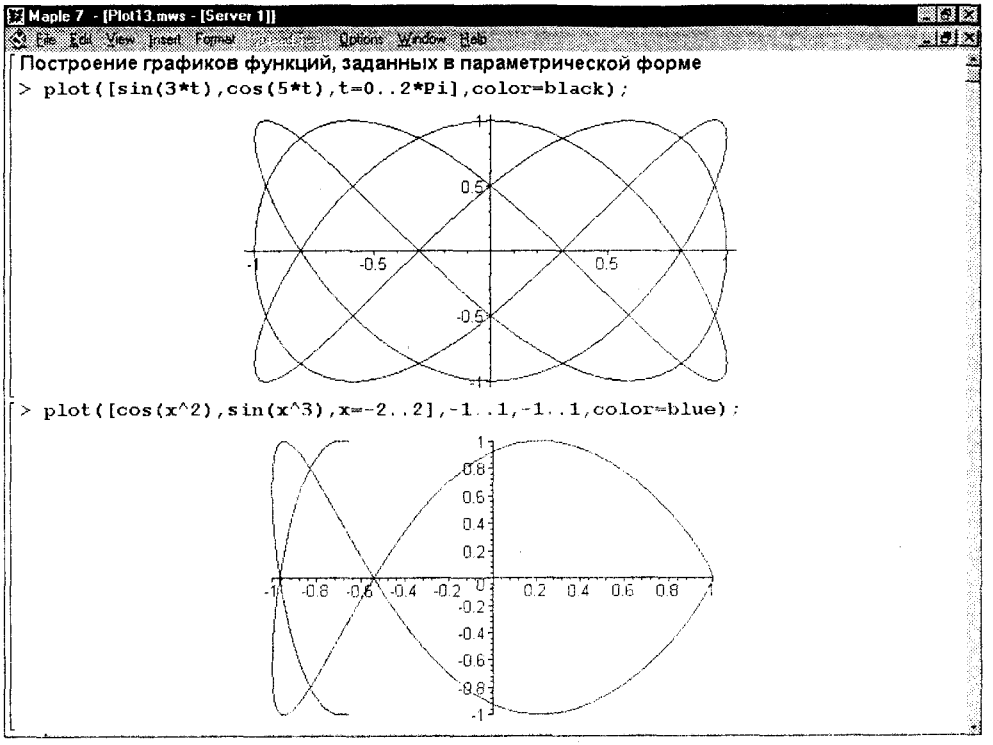


Рис. 11.13. Построение функций, заданных параметрически

```
plot([r(t),theta(t),t=tmin..tmax],h,v,p,coords=polar)
```

Здесь существенным моментом является задание полярной системы координат параметром `coords=polar`. Рисунок 11.14 дает примеры построения графиков функций в полярной системе координат.

Графики параметрических функций и функций в полярной системе координат отличаются огромным разнообразием. Снежинки и узоры мороза на стеклах, некоторые виды кристаллов и многие иные физические объекты подчиняются математическим закономерностям, положенным в основу построения таких графиков.

Построение трехмерных графиков

Особенности применения функции plot3d

Трехмерными называют графики, отображающие функции двух переменных $z(x, y)$. Каждая точка z_i таких графиков является высотой (аппликатой) точки, лежащей в плоскости XY и представленной координатами (x_i, y_i). Поскольку экран монитора компьютера в первом приближении является плоским, то на деле трехмерные графики представляют собой специальные проекции объемных объектов.

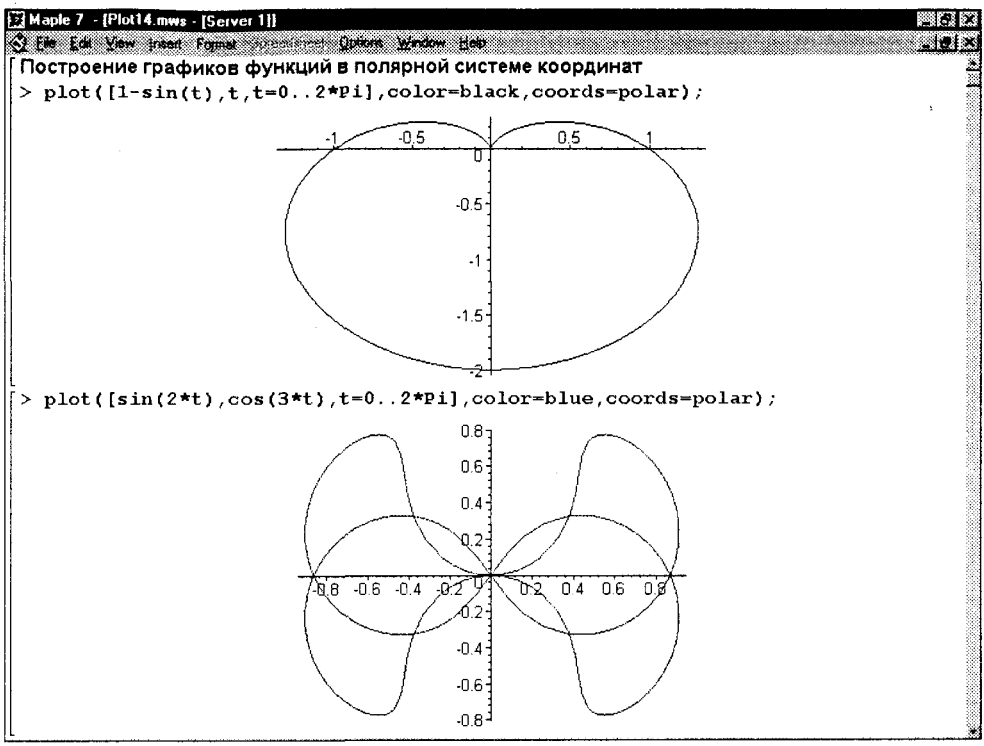


Рис. 11.14. Построение графиков функций в полярной системе координат

Для построения графиков трехмерных поверхностей Maple имеет встроенную в ядро функцию `plot3d`. Она может использоваться в следующих форматах:

```
plot3d(expr1, x=a..b, y=c..d,p)
plot3d(f, a..b, c..d,p)
plot3d([exprf,exprg,exprh], s=a..b, t=c..d,p)
plot3d([f,g,h], a..b, c..d,p)
```

В двух первых формах `plot3d` применяется для построения обычного графика одной поверхности, в других формах — для построения графика с параметрической формой задания поверхности. В приведенных формах записи `f`, `g` и `h` — функции; `expr1` — выражение, отражающее зависимость от `x` и `y`; `exprf`, `exprg` и `exprh` — выражения, задающие поверхность параметрически; `s`, `t`, `a` и `b` — числовые константы действительного типа; `c` и `d` — числовые константы или выражения действительного типа; `x`, `y`, `s` и `t` — имена независимых переменных; `p` — управляющие параметры.

Параметры функции plot3d

С помощью параметров `p` можно в широких пределах управлять видом трехмерных графиков, выводя или убирая линии каркасной сетки, вводя функциональную окраску поверхностей, меняя угол их обзора и параметры освещения, изме-

няя вид координатных осей и т. д. Следующие параметры функции plot3d задаются аналогично их заданию для функции plot:

axesfont	font	color	coords	font
labelfont	linestyle	numpoints	scaling	style
symbol	thickness	title	titlefont	

Однако функция plot3d имеет ряд дополнительных специфических параметров:

- ambientlight=[r,g,b] — задает интенсивность красного (r), зеленого (g) и синего (b) цветов подсветки в относительных единицах (от 0 до 1);
- axes=f — задает вид координатных осей (BOXED, NORMAL, FRAME и NONE, по умолчанию NONE);
- grid=[m,n] — задает число линий каркаса поверхности;
- gridstyle=x — задает стиль линий каркаса x ('rectangular' или 'triangular');
- labels=[x,y,z] — задает надписи по осям (x, y и z — строки, по умолчанию пустые);
- light=[phi,theta,r,g,b] — задает углы, под которыми расположен источник освещения поверхности и интенсивности составляющих цвета (r, g и b);
- lightmodel=x — задает схему освещения (соответственно 'none', 'light1', 'light2', 'light3' и 'light4');
- orientation=[theta,phi] — задает углы ориентации поверхности (по умолчанию 45°);
- projection=r — задает перспективу при обзоре поверхности (r может быть числом 0 или 1, задающим включение или выключение перспективы, а также одной из строк 'FISHEYE', 'NORMAL' или 'ORTHOGONAL' (это соответствует численным значениям r, равным 0, 0,5 или 1, причем по умолчанию задано projection=ORTHOGONAL));
- shading=s — задает направления, по которым меняется цвет функциональной окраски (значения s могут быть XYZ, XY, Z, ZGREYSCALE, ZHUE, NONE);
- tickmarks=[l,n,m] — задает характер маркировки по осям x, y и z (числа l, n и m имеют значения не менее 1);
- view=zmin..zmax или view=[xmin..xmax, ymin..ymax, zmin..zmax] — задает минимальные и максимальные координаты поверхности для ее видимых участков.

Выбор и пересчет координат трехмерных графиков

Для трехмерных графиков возможно задание 31 типа координатных систем с помощью параметра coords=Тип_координатной_системы. Поскольку на экране монитора поверхность отображается только в прямоугольной системе координат и характеризуется координатами x, y и z, то для представления поверхности, заданной в иной системе координат с координатами u, v и w, используются известные [46, 47] формулы для преобразования (u, v, w) --> (x, y, z). Ниже

перечислены типы трехмерных координатных систем и соответствующие формулы преобразования.

```
bipolarcylindrical:
                x = a*sinh(v)/(cosh(v)-cos(u))
                y = a*sin(u)/(cosh(v)-cos(u))
                z = w
bispherical:
                x = sin(u)*cos(w)/d
                y = sin(u)*sin(w)/d
                z = sinh(v)/d     где d = cosh(v) - cos(u)
cardioidal:
                x = u*v*cos(w)/(u^2+v^2)^2
                y = u*v*sin(w)/(u^2+v^2)^2
                z = (u^2-v^2)/2/(u^2+v^2)^2
cardioidcylindrical:
                x = (u^2-v^2)/2/(u^2+v^2)^2
                y = u*v/(u^2+v^2)^2
                z = w
casscylindrical:
                x = a*2^(1/2)/2*((exp(2*u)+2*exp(u)*cos(v)+1)^(1/2)+exp(u)*cos(v)+1)^(1/2)
                y = a*2^(1/2)/2*((exp(2*u)+2*exp(u)*cos(v)+1)^(1/2)-exp(u)*cos(v)-1)^(1/2)
                z = w
confocalellip:
                x = ((a^2-u)*(a^2-v)*(a^2-w)/(a^2-b^2)/(a^2-c^2))^(1/2)
                y = ((b^2-u)*(b^2-v)*(b^2-w)/(b^2-a^2)/(b^2-c^2))^(1/2)
                z = ((c^2-u)*(c^2-v)*(c^2-w)/(c^2-a^2)/(c^2-b^2))^(1/2)
confocalparab:
                x = ((a^2-u)*(a^2-v)*(a^2-w)/(b^2-a^2))^(1/2)
                y = ((b^2-u)*(b^2-v)*(b^2-w)/(b^2-a^2))^(1/2)
                z = (a^2+b^2-u-v-w)/2
conical:
                x = u*v*w/(a*b)
                y = u/b*((v^2 - b^2)*(b^2-w^2)/(a^2-b^2))^(1/2)
                z = u/a*((a^2 - v^2)*(a^2 - w^2)/(a^2-b^2))^(1/2)
cylindrical:
                x = u*cos(y)
                y = u*sin(y)
                z = w
ellcylindrical:
                x = a*cosh(u)*cos(v)
                y = a*sinh(u)*sin(v)
                z = w
ellipsoidal:
                x = u*v*w/a/b
                y = ((u^2-b^2)*(v^2-b^2)*(b^2-w^2)/(a^2-b^2))^(1/2)/b
                z = ((u^2-a^2)*(a^2-v^2)*(a^2-w^2)/(a^2-b^2))^(1/2)/a
hypercylindrical:
                x = ((u^2+v^2)^(1/2)+u)^(1/2)
                y = ((u^2+v^2)^(1/2)-u)^(1/2)
                z = w
invcasscylindrical:
                x = a*2^(1/2)/2*((exp(2*u)+2*exp(u)*cos(v)+1)^(1/2) +
```

```
                        exp(u)*cos(v)+1)^(1/2)/(exp(2*u)+2*exp(u)*cos(v)+1)^(1/2)
        y = a*2^(1/2)/2*((exp(2*u)+2*exp(u)*cos(v)+1)^(1/2) -
                        exp(u)*cos(v)-1)^(1/2)/(exp(2*u)+2*exp(u)*cos(v)+1)^(1/2)
        z = w
invellcylindrical:
        x = a*cosh(u)*cos(v)/(cosh(u)^2-sin(v)^2)
        y = a*sinh(u)*sin(v)/(cosh(u)^2-sin(v)^2)
        z = w
invoblspheroidal:
        x = a*cosh(u)*sin(v)*cos(w)/(cosh(u)^2-cos(v)^2)
        y = a*cosh(u)*sin(v)*sin(w)/(cosh(u)^2-cos(v)^2)
        z = a*sinh(u)*cos(v)/(cosh(u)^2-cos(v)^2)
invprospheroidal:
        x = a*sinh(u)*sin(v)*cos(w)/(cosh(u)^2-sin(v)^2)
        y = a*sinh(u)*sin(v)*sin(w)/(cosh(u)^2-sin(v)^2)
        z = a*cosh(u)*cos(v)/(cosh(u)^2-sin(v)^2)
logcylindrical:
        x = a/Pi*ln(u^2+v^2)
        y = 2*a/Pi*arctan(v/u)
        z = w
logcoshcylindrical:
        x = a/Pi*ln(cosh(u)^2-sin(v)^2)
        y = 2*a/Pi*arctan(tanh(u)*tan(v))
        z = w
maxwellcylindrical:
        x = a/Pi*(u+1+exp(u)*cos(v))
        y = a/Pi*(v+exp(u)*sin(v))
        z = w
oblatespheroidal:
        x = a*cosh(u)*sin(v)*cos(w)
        y = a*cosh(u)*sin(v)*sin(w)
        z = a*sinh(u)*cos(v)
paraboloidal:
        x = u*v*cos(w)
        y = u*v*sin(w)
        z = (u^2 - v^2)/2
paraboloidal2:
        x = 2*((u-a)*(a-v)*(a-w)/(a-b))^(1/2)
        y = 2*((u-b)*(b-v)*(b-w)/(a-b))^(1/2)
        z = u+v+w-a-b
paracylindrical:
        x = (u^2 - v^2)/2
        y = u*v
        z = w
prolatespheroidal:
        x = a*sinh(u)*sin(v)*cos(w)
        y=a*sinh(u)*sin(v)*sin(w)
        z=a*cosh(u)*cos(v)
rectangular:
        x = u
        y = v
        z = w
rosecylindrical:
```

```
                x = ((u^2+v^2)^(1/2)+u)^(1/2)/(u^2+v^2)^(1/2)
                y = ((u^2+v^2)^(1/2)-u)^(1/2)/(u^2+v^2)^(1/2)
                z = w
sixsphere:
                x = u/(u^2+v^2+w^2)
                y = v/(u^2+v^2+w^2)
                z = w/(u^2+v^2+w^2)
spherical:
                x = u*cos(v)*sin(w)
                y = u*sin(v)*sin(w)
                z = u*cos(w)
tangentcylindrical:
                x = u/(u^2+v^2)
                y = v/(u^2+v^2)
                z = w
tangentsphere:
                x = u*cos(w)/(u^2+v^2)
                y = u*sin(w)/(u^2+v^2)
                z = v/(u^2+v^2)
toroidal:
                x = a*sinh(v)*cos(w)/d
                y = a*sinh(v)*sin(w)/d
                z = a*sin(u)/d                  где d = cosh(v) - cos(u)
```

Эти формулы полезно знать, поскольку в литературе встречаются несколько отличные формулы пересчета. Вид графиков трехмерных поверхностей очень сильно различается в разных координатных системах. По умолчанию трехмерные графики строятся в прямоугольной системе координат — `rectangular`.

Построение поверхностей

Построение поверхностей с разными стилями

На рис. 11.15 показано два примера простейших построений графиков трехмерной поверхности. По умолчанию в Maple 7 строится поверхность с функциональной окраской и стилем `style=patch` (верхний рисунок). Функциональная окраска делает рисунки более информативными, но, увы, на рисунках в книге она превращается в окраску оттенками серого цвета.

Параметр `style=hidden` строит каркасную поверхность с функциональной окраской тонких линий каркаса и удалением невидимых линий. Чтобы график выглядел более четким, построение во втором примере задано линиями черного цвета с помощью параметра `color=black` (см. нижний рисунок на рис. 11.15).

Помимо значения `patch` для построения трехмерных поверхностей можно задавать ряд других стилей: `point` — точками, `contour` — контурными линиями, `line` — линиями, `hidden` — линиями каркаса с удалением невидимых линий, `wireframe` — линиями каркаса со всеми видимыми линиями, `patchnogrid` — с раскраской, но без линий каркаса, `patchcontour` — раскраска с линиями равного уровня.

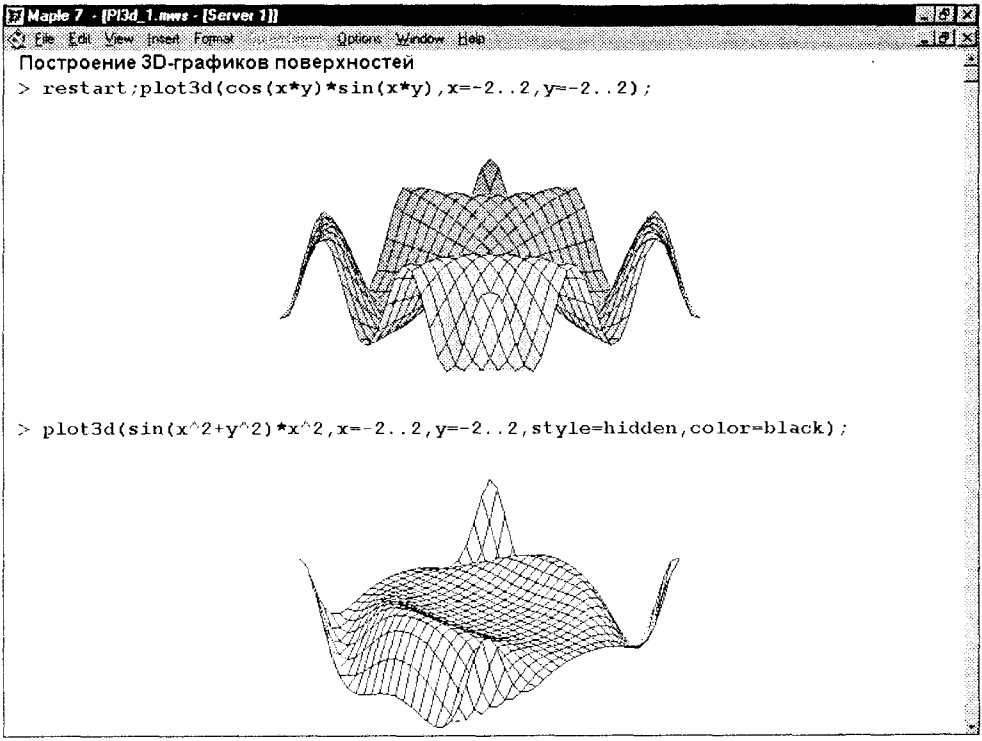


Рис. 11.15. Примеры простейшего построения трехмерных поверхностей

Цвет трехмерного графика может задаваться (как и для двумерного) параметром `color=c`, где `c` — цвет (оттенки цвета перечислялись ранее). Возможно еще два алгоритма задания цвета:

- `HUE` — алгоритм с заданием цвета в виде `color=f(x,y)`;
- `RGB` — алгоритм с заданием цвета в виде `color=[exprr,exprg,exprb]`, где выражения `exprr`, `exprg` и `exprb` задают относительную значимость (от 0 до 1) основных цветов (красного — `exprr`, зеленого — `exprg` и синего — `exprb`).

Удачный выбор углов обзора фигуры и применение функциональной окраски позволяют придать построениям трехмерных фигур весьма эффектный и реалистический вид.

Построение фигур в различных системах координат

Как отмечалось, вид графика трехмерной поверхности существенно зависит от выбора координатной системы. Рисунок 11.16 показывает пример построения нелинейного конуса в цилиндрической системе координат. Для задания такой системы координат используется параметр `coords=cylindrical`.

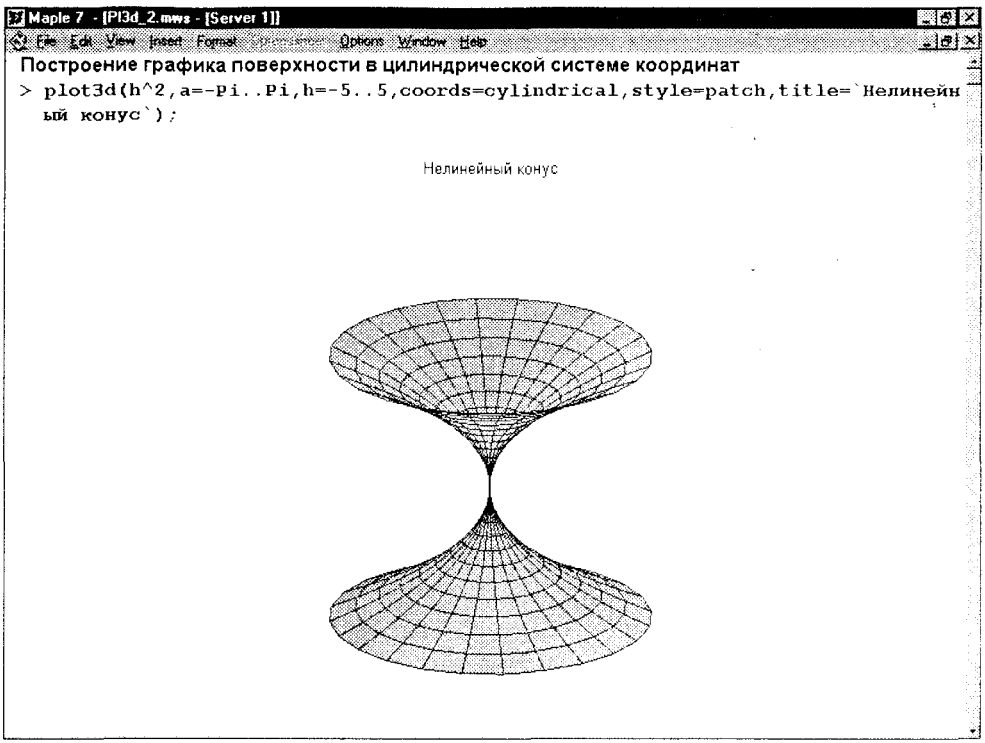


Рис. 11.16. Нелинейная цилиндрическая поверхность

При построении этой фигуры также использована цветная функциональная окраска. Кроме того, этот пример иллюстрирует вывод над рисунком титульной надписи (кстати, сделанной на русском языке).

Приведем еще один пример построения трехмерной поверхности — на этот раз в сферической системе координат (рис. 11.17). Здесь функция задана вообще элементарно просто — в виде числа 1. Но, поскольку выбрана сферическая система координат, в результате строится поверхность шара единичного радиуса.

О том, насколько необычным может быть график той или иной функции в различных системах координат, свидетельствует рис. 11.18. На нем показан график параметрически заданной функции от одной координаты t — $\sin(t^3)$, построенный в сферической системе координат.

Кстати, рис. 11.18 иллюстрирует возможность одновременного наблюдения нескольких окон. В одном окне задано построение графика, а в другом построен сам график. При построении графика в отдельном окне появляется панель форматирования графика. С помощью ее довольно наглядных кнопок можно легко скорректировать вспомогательные параметры графика (окраску, наличие линий каркаса, ориентацию и др.).

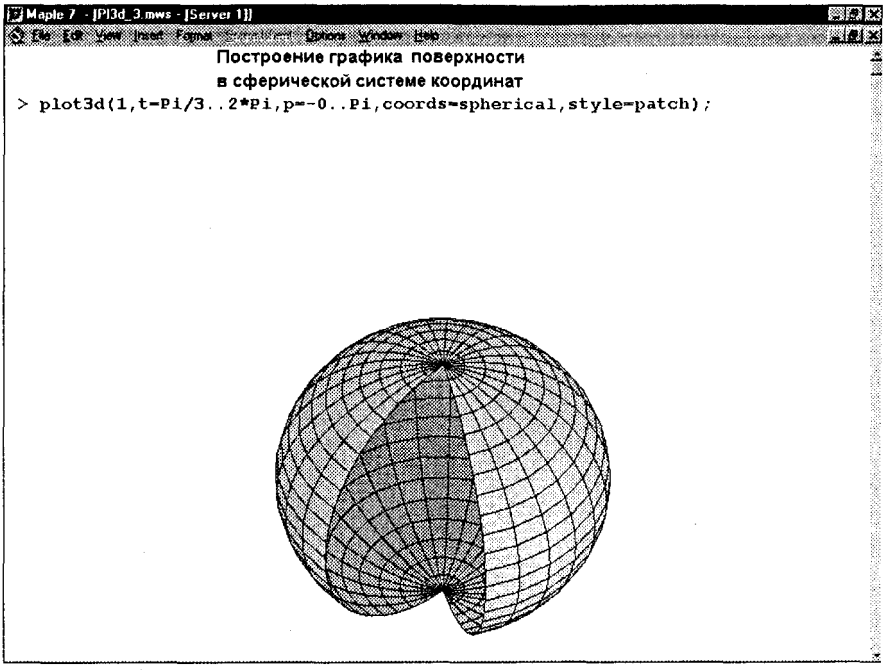


Рис. 11.17. *Построение шарообразной поверхности в сферической системе координат*

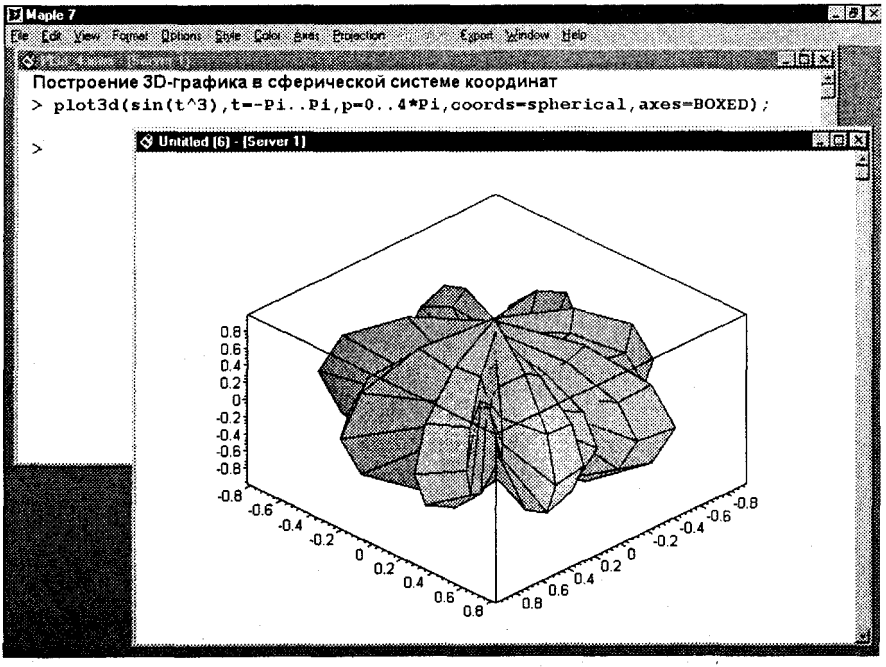


Рис. 11.18. **График еще одной поверхности в сферической системе координат**

3D-графики параметрически заданных поверхностей

На рис. 11.19 показано построение поверхности при полном ее параметрическом задании. В этом случае поверхность задается тремя формулами, содержащимися в списке.

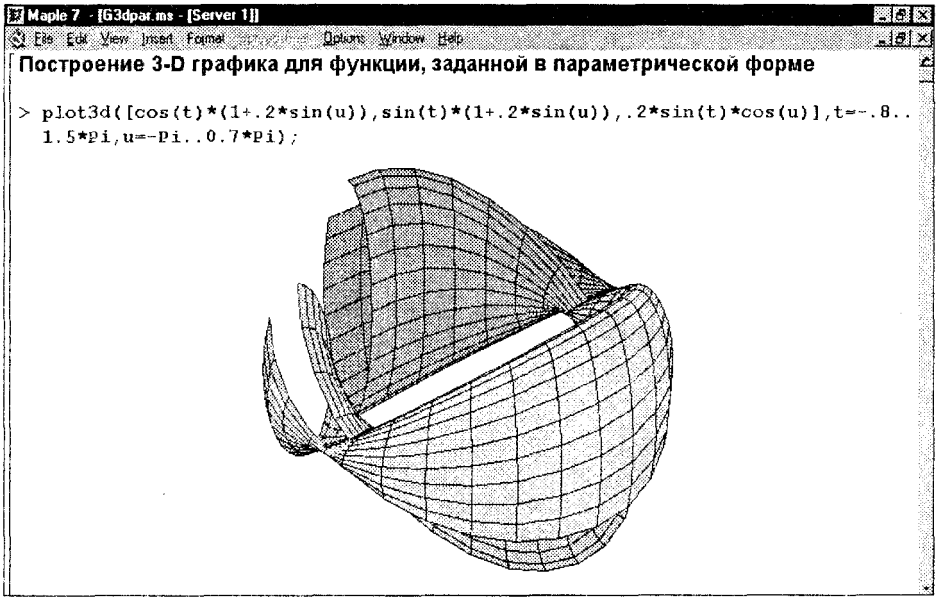


Рис. 11.19. График трехмерной поверхности при полном параметрическом ее задании

В данном случае функциональная окраска задана из меню, поэтому в состав функции соответствующий параметр не введен. Обратите внимание на технику удаления частей фигуры путем задания соответствующего диапазона изменения параметров t и u.

Следующий пример показывает построение простого тороида — цилиндра, свернутого в кольцо (рис. 11.20). Здесь также использован прием удаления части фигуры, что делает ее представление более наглядным и красочным. Кроме того, введены параметры, задающие функциональную окраску.

Тор на рис. 11.20 выглядит, как произведение искусства. Он дает полное и наглядное представление об этой фигуре.

Масштабирование трехмерных фигур и изменение углов их обзора

Полезно обратить внимание на параметр масштаба `scaling=constrained`, явно введенный в документ рис. 11.20. Его можно было бы и не вводить, поскольку

этот параметр задается по умолчанию. Он выравнивает масштабы представления фигуры по осям координат, обычно используется по умолчанию и позволяет снизить до минимума геометрические искажения фигур — тор, например, при этом виден как круглая труба, свернутая в кольцо. У таких графиков есть специфический недостаток — они занимают малую часть окна вывода.

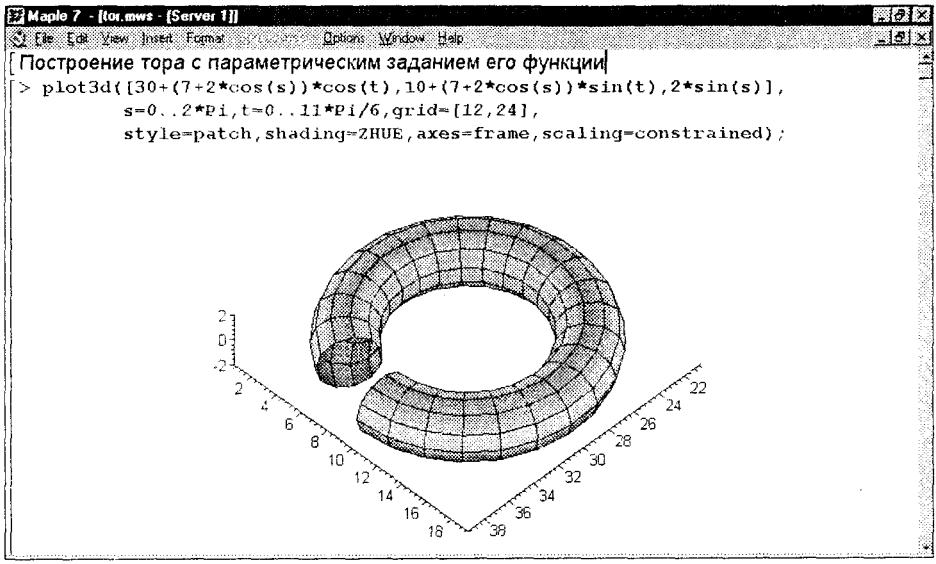


Рис. 11.20. Тор с функциональной окраской поверхности

Задание параметра `scaling=unconstrained` означает отказ от равного масштаба по осям. График при этом увеличивается в размерах, но становятся заметны его искажения по осям координат. В итоге тор превращается в толстую сплющенную трубу с эллиптическим сечением (рис. 11.21).

Весьма важным является учет углов, под которыми наблюдается трехмерная поверхность или объект. К примеру, построение рис. 11.21 неудачно в том плане, что оно не показывает наличия у тора дырки. В общем, как в поговорке: «кому бублик, а кому дырка от бублика» — ведь бублик и есть материально реализованный тор. Простейший и очень удобный способ изменить угол обзора заключается во вращении фигуры на рисунке мышью при нажатой левой кнопке. При этом можно повернуть фигуру так, что ее геометрические особенности будут видны (рис. 11.22).

В Maple есть способ явно задать углы обзора с помощью параметра `orientation=[theta, phi]`, где `theta` и `phi` — углы, через которые задаются параметрические уравнения трехмерной фигуры или поверхности. Рисунок 11.23 дает пример такого задания фигуры, которую можно назвать «квадратным» тором.

Обратите внимание, что значения заданных углов обзора повторяются в полях углов на контекстной панели инструментов. Разумеется, последние будут меняться, если начать вращать фигуру на рисунке мышью.

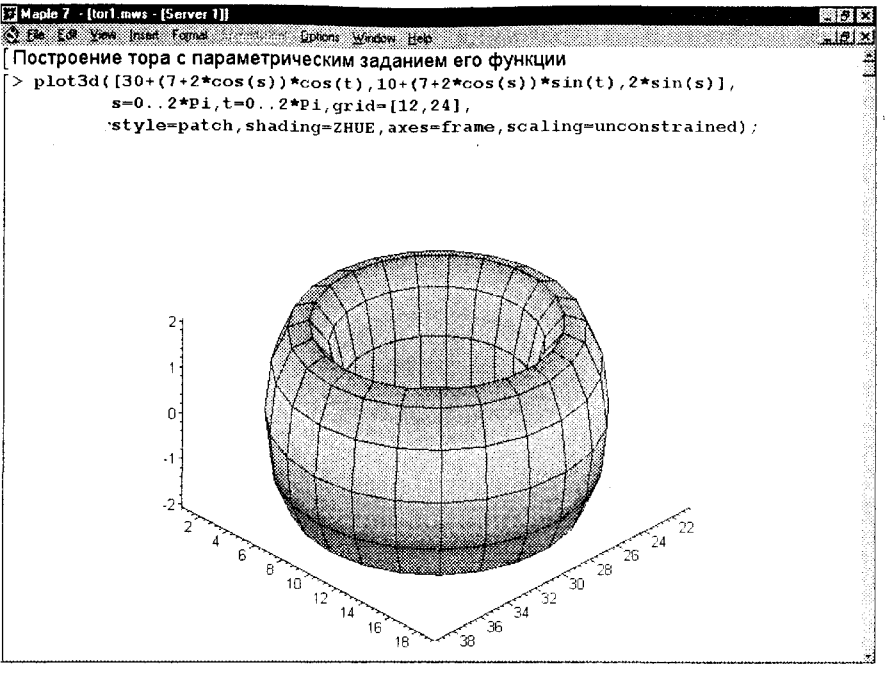


Рис. 11.21. Тор, построенный с применением значения параметра scaling=unconstrained

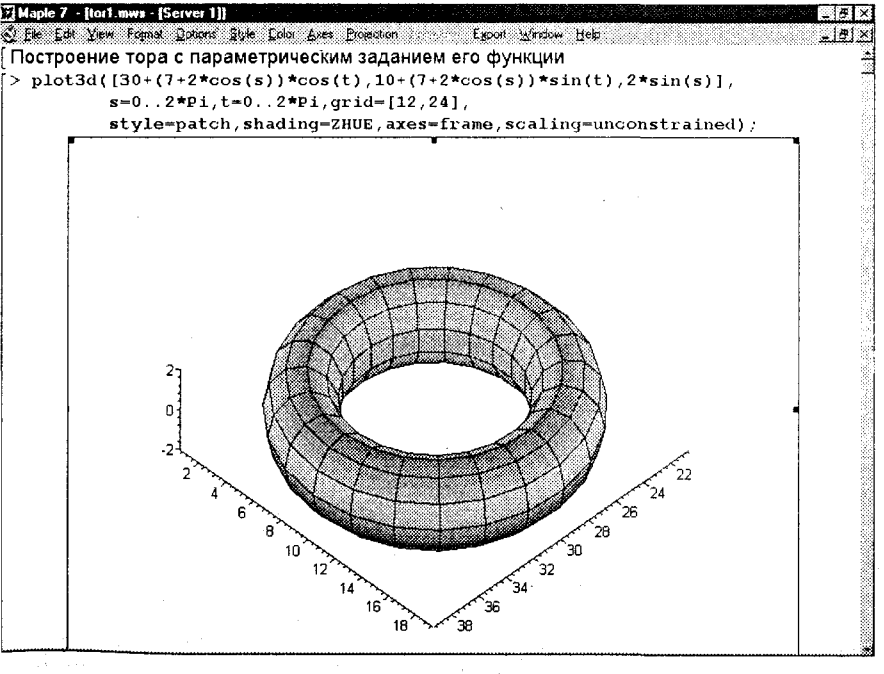


Рис. 11.22. Тор с рис. 11.21 после поворота мышью демонстрирует, что он и впрямь имеет дырку

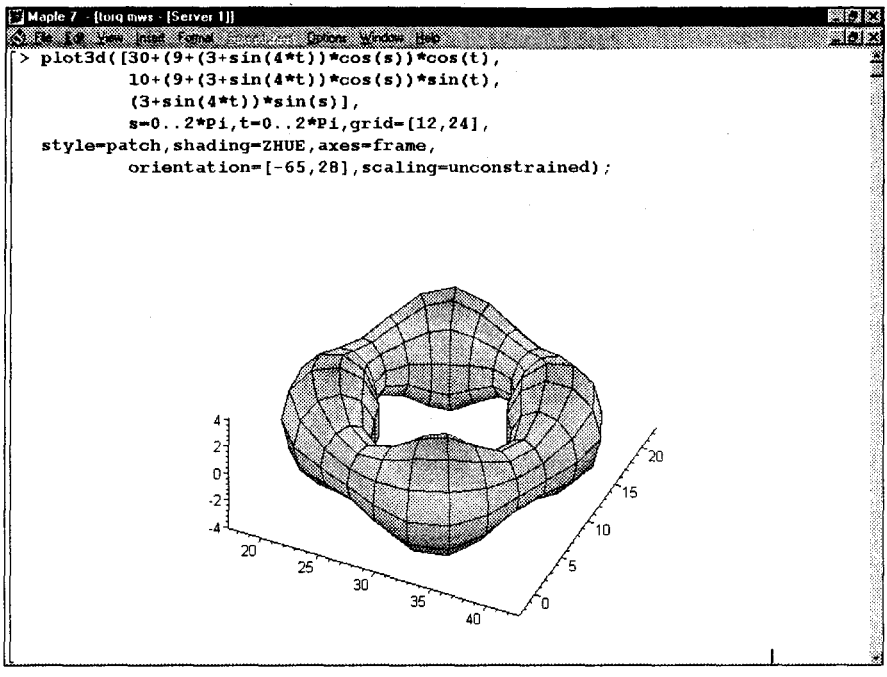


Рис. 11.23. «Квадратный» тор, представленный под заданными углами обзора

Занимательные фигуры — трехмерные графики

Параметрическое задание уравнений поверхности открывает почти неисчерпаемые возможности построения занимательных и сложных фигур самого различного вида. Приведем пару построений такого рода.

На рис. 11.24 показан тор, сечение которого имеет вид сплюснутой шестиконечной звезды. Вырез в фигуре дает прекрасный обзор ее внутренней поверхности, а цветная функциональная окраска и линии сетки, построенные с применением алгоритма удаления невидимых линий, дают весьма реалистичный вид фигуры. Замените параметр `scaling=unconstrained` на `scaling=constrained`, и вы получите тор с неискаженным сечением.

На рис. 11.25 показан еще один тор. На этот раз он круглого сечения, но сверху и снизу имеет вид пятиконечной звезды.

ПРИМЕЧАНИЕ

В приведенных на рис. 11.19–11.25 программах построения различных поверхностей и трехмерных фигур имеется ряд характерных констант и математических выражений, определяющих как вид фигур, так и их размеры и положение. Рекомендуется тщательно проанализировать эти примеры и попробовать их в работе с несколько измененными теми или иными данными. Полезно построить ряд подобных примеров самостоятельно. Все это будет способствовать привитию учащимся специального геометрического стиля мышления, при котором геометрические особенности фигур связываются с их расчетным описанием.

```
> plot3d([30+(11+(4+sin(6*s))*cos(s))*cos(t),
          20+(11+(4+sin(6*s))*cos(s))*sin(t),
          (4+sin(6*s))*sin(s)],
          s=0..2*Pi,t=0..3/2*Pi,grid=[36,24],orientation=[-60,64],
  style=patch,shading=ZHUE,axes=frame,scaling=unconstrained);
```

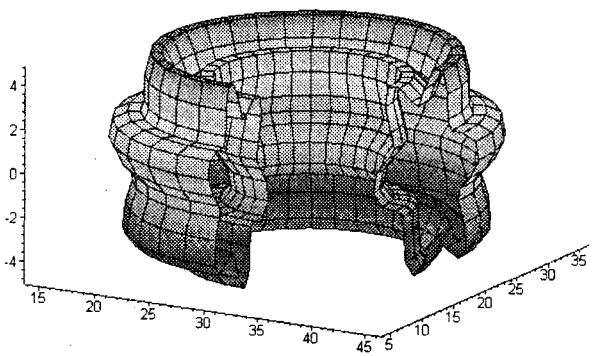


Рис. 11.24. Тор с сечением в виде шестиконечной звезды

```
> plot3d([30+(9+2*sin(5*t)+2*cos(s))*cos(t),
          10+(9+2*sin(5*t)+2*cos(s))*sin(t),
          2*sin(s)],s=0..2*Pi,t=0..11*Pi/6,grid=[12,36],
          style=patch,axes=frame,scaling=constrained);
```

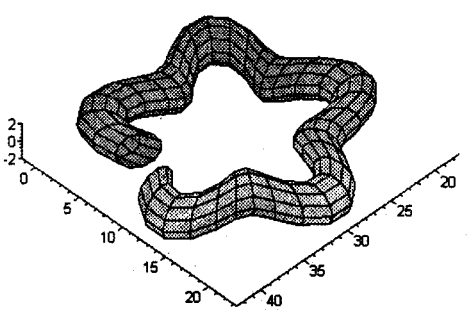


Рис. 11.25. Тор круглого сечения в виде пятиконечной звезды

Быстрое построение графиков

Двумерная быстрая графика — smartplot

В последние реализации системы Maple (5, 6 и 7) введены новые функции быстрого построения графиков. Функция `smartplot(f)` предназначена для создания двумерных графиков. Параметр `f` может задаваться в виде одиночного выражения или набора выражений, разделяемых запятыми. Задание управляющих парамтеров в этих графических функциях не предусмотрено; таким образом, их можно считать первичными, или черновыми. Для функции построения двумерного графика по умолчанию задан диапазон изменения аргумента `-10..10`.

Рисунок 11.26 иллюстрирует применение функции `smartplot` для построения трех (верхний пример) и двух (нижний пример) графиков функций на одном рисунке.

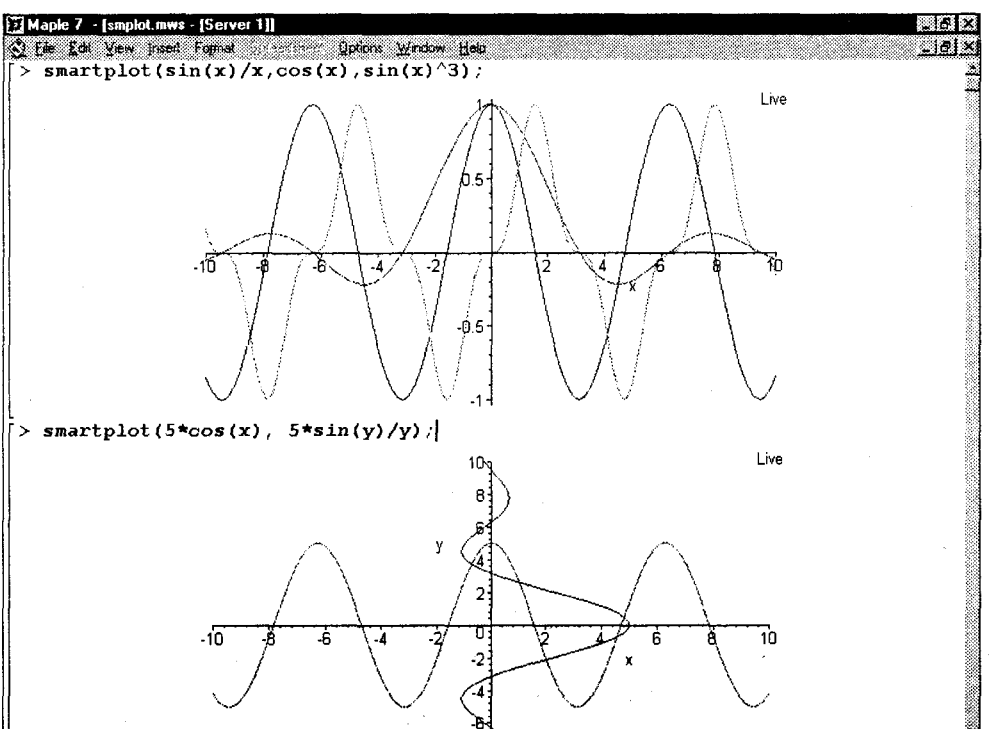


Рис. 11.26. Построение графиков с помощью функции smartplot

Обратите внимание на второй пример применения функции `smartplot`. Здесь график выражения $5\sin(y)/y$ построен относительно вертикальной оси. Поэтому он развернут на 90° относительно графика, построенного обычным образом.

ПРИМЕЧАНИЕ На графиках, построенных командой smartplot(x), присутствует надпись «Live», что видно на рис. 11.26.

Быстрое построение трехмерных графиков smartplot3d

Быстрое (не в смысле ускорения самого построения, а лишь в смысле более быстрого задания построения графиков) построение трехмерных графиков обеспечивает функция smartplot3d. Для этой функции задан диапазон изменения обоих аргументов -5..5. Рисунок 11.27 поясняет применение функции smartplot3d.

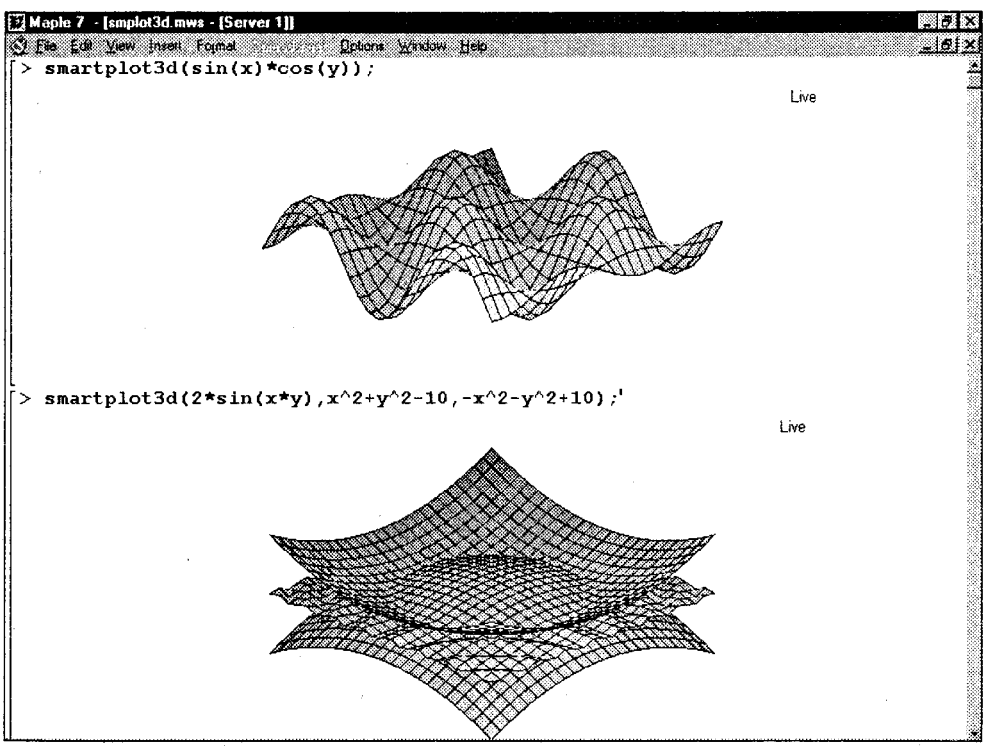


Рис. 11.27. Примеры применения функции smartplot3d

Как видно из второго примера, представленного на рис. 11.27, функция smartplot3d обеспечивает построение не только отдельных поверхностей, но и ряда пересекающихся поверхностей. При этом линии пересечения поверхностей строятся вполне корректно.

Специальные приемы построения трехмерных графиков

Трехмерный график как графический объект

Принадлежность функций plot и plot3d к функциям (в ряде книг их именуют операторами, командами или процедурами) наглядно выявляется при создании графических объектов. Графический объект — это, в сущности, обычная переменная, которой присваивается значение графической функции. После этого такая переменная, будучи вызванной, производит построение соответствующего графика. Пример этого дан на рис. 11.28.

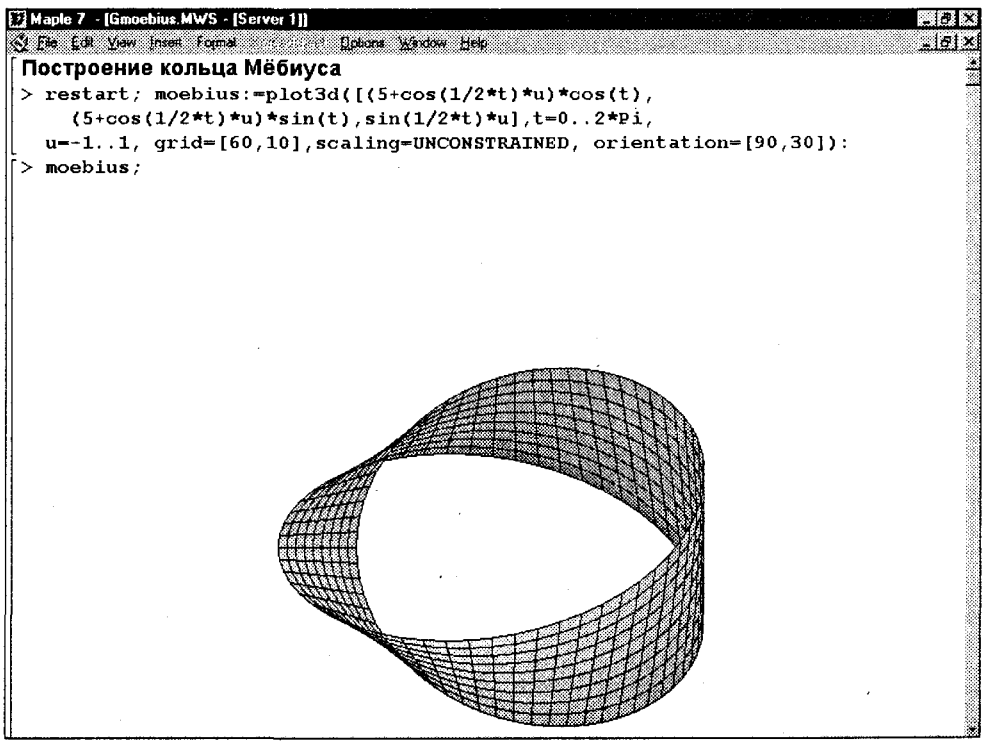


Рис. 11.28. Пример задания и вывода трехмерного графика — графического объекта

В данном случае строится лента Мебиуса, свойства которой (например, плавный переход с одной стороны ленты на другую) уже много веков будоражат воображение людей.

Поскольку можно говорить, что вызов переменной возвращает графический объект, то это дает повод считать plot и plot3d графическими функциями.

Задание трехмерных графиков в виде процедур

Язык программирования Maple 7 допускает применение в процедурах любых внутренних функций, в том числе графических. Пример такого применения дает рис. 11.29.

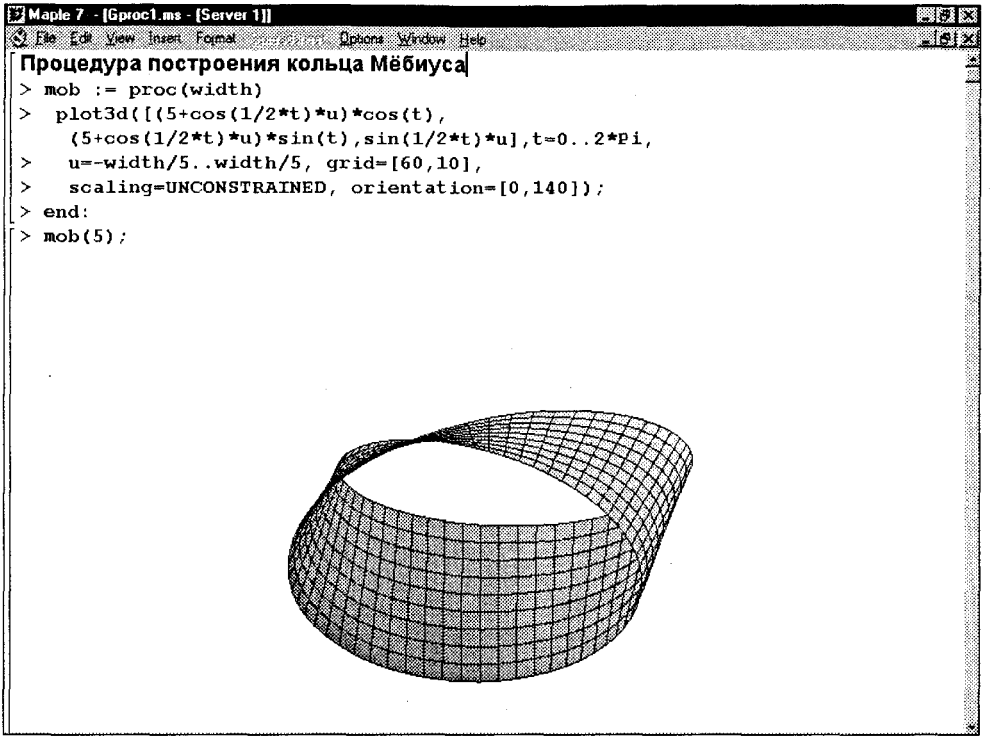


Рис. 11.29. Пример создания и применения процедуры трехмерной графики

Этот пример показывает еще один способ задания и построения кольца Мебиуса. Практически любые графические построения можно оформлять в виде процедур и использовать такие процедуры в своих документах.

Построение ряда трехмерных фигур на одном графике

Функция `plot3d` позволяет строить одновременно несколько фигур, пересекающихся в пространстве. Для этого достаточно вместо описания одной поверхности задать список описаний ряда поверхностей. При этом функция `plot3d` обладает уникальной возможностью — автоматически вычисляет точки пересечения фигур и показывает только видимые части поверхностей. Это создает изобра-

жения, выглядящие вполне естественно. Пример такого построения для двух функций показан на рис. 11.30.

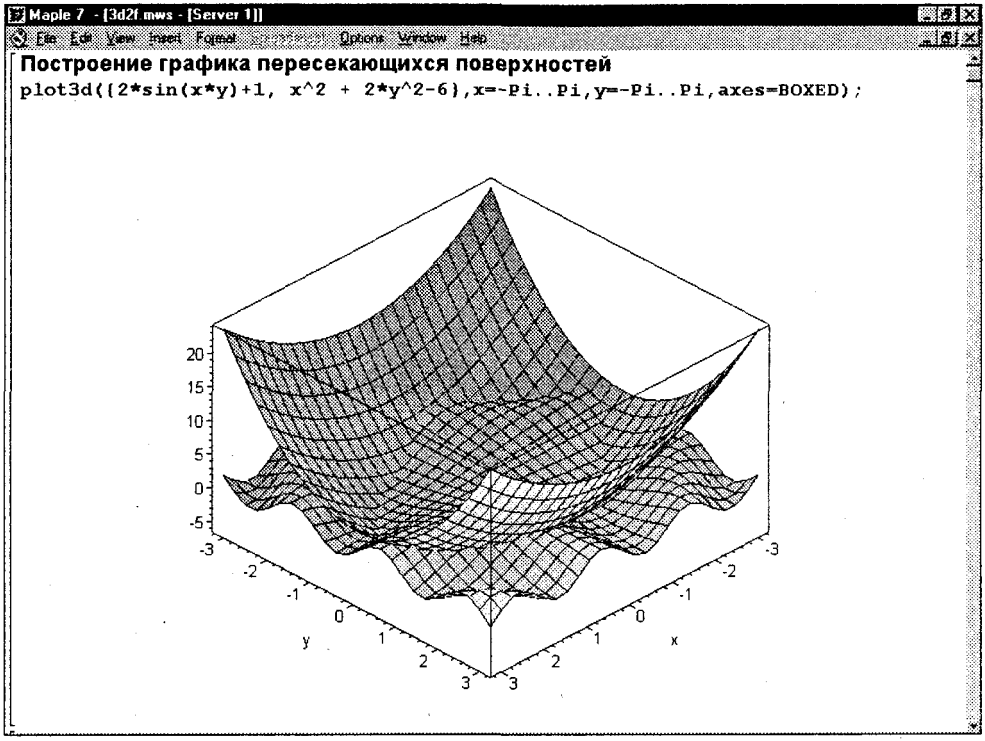


Рис. 11.30. Пример построения двух трехмерных фигур, пересекающихся в пространстве

Двумерные и трехмерные графические структуры

Понятие о графических структурах

Функции PLOT и PLOT3D (с именами, набранными большими буквами) позволяют создавать *графические структуры*, содержащие ряд графических объектов s1, s2, s3 и т. д. Каждый объект может представлять собой точку или фигуру, полигон, надпись и т. д., позиционированную с высокой точностью в заданной системе координат. Координатные оси также относятся к графическим объектам. Важно отметить, что функции PLOT и PLOT3D одновременно являются данными, описывающими графики. Их можно записывать в виде файлов и (после открытия файлов) представлять в виде графиков. Особые свойства этих функций подчеркиваются их записью прописными буквами.

Графические структуры двумерной графики

Графическая структура двумерной графики задается в виде:

```
PLOT(s1, s2, s3,...,o);
```

где s1, s2, s3 ... — графические объекты (или элементарные структуры — примитивы), o — общие для структуры параметры.

Основными объектами являются:

- POINTS([x1,y1],[x2,y2],...[xn,yn]) — построение точек, заданных их координатами;
- CURVES([[x11,y11],...[x1n,y1n]], [[x21,y21],...[x2n,y2n]],... [[xm1,ym1],...[xmn,ymn]]) — построение кривых по точкам;
- POLYGONS([[x11,y11],...[x1n,y1n]], [[x21,y21],...[x2n,y2n]],... [[xm1,ym1],...[xmn,ymn]]) — построение замкнутой области-полигона (многоугольника, так как последняя точка должна совпадать с первой);
- TEXT([x,y],`string`,horizontal,vertical) — вывод текстовой надписи `string`, позиционированной в точке с координатами [x,y], с горизонтальной или вертикальной ориентацией. Параметр horizontal может иметь значение ALIGNLEFT или ALIGNRIGHT, указывающие, в какую сторону (влево или вправо) идет надпись. Аналогично параметр vertical может иметь значение ALIGNABOVE или ALIGNBELOW, указывающее в каком направлении (вверх или вниз) идет надпись.

При задании графических объектов (структур) s1, s2, s3 и т. д. можно использовать описанные выше параметры и параметры, например, для задания стиля построения — STYLE (POINT, LINE, PATCH, PATCHNOGRID); толщины линий, — THICKNESS (кроме координатных осей); символа, которым строятся точки кривых — SYMBOL (BOX, CROSS, CIRCLE, POINT, DIAMOND и DEFAULT); стиля линий — LINESTYLE; цвета — COLOR (например, COLOR(HUE,0) для закраски непрерывной области), типа шрифта — FONT; вывода титульной надписи — TITLE(string); имени объекта — NAME(string); стиля координатных осей — AXESSTYLE (BOX, FRAME, NORMAL, NONE или DEFAULT) и т. д.

Следует отметить, что параметры в графических структурах задаются несколько иначе — с помощью круглых скобок. Например, для задания шрифта TIMES ROMAN с размером символов 16 пунктов надо записать FONT(TIMES,ROMAN,16), для задания стиля координатных осей в виде прямоугольника — AXESSTYLE(BOX) и т. д.

На рис. 11.31 показан пример графических построений при использовании основных структур двумерной графики.

Как видно из этого примера, графическая двумерная структура позволяет задавать практически любые двумерные графики и текстовые надписи в пределах одного рисунка.

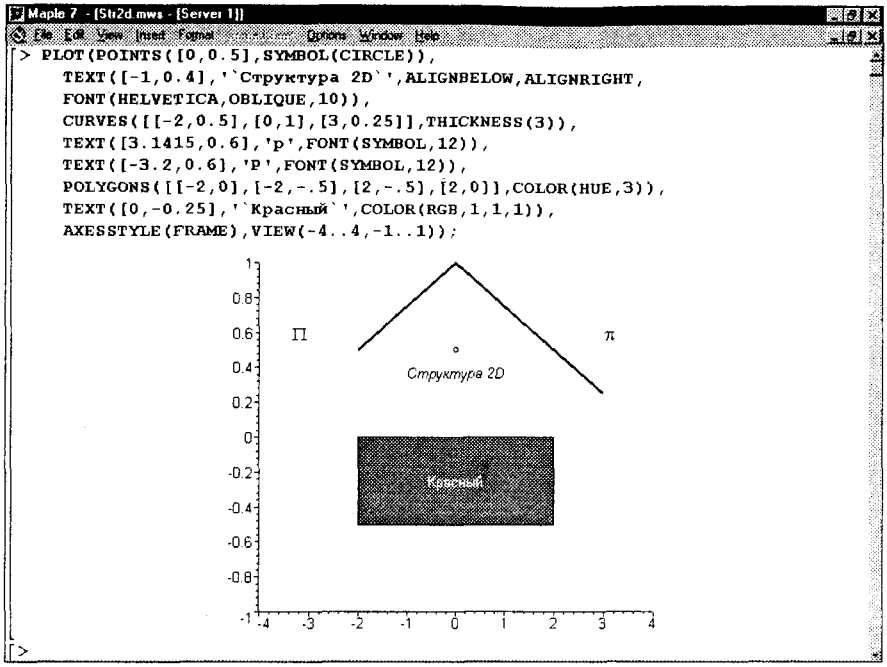


Рис. 11.31. Пример использования двумерных структур

Графические структуры трехмерной графики

Графические структуры трехмерной графики строятся функцией PLOT3D:

```
PLOT3D(s1,s2,s3,...,o)
```

В качестве элементарных графических структур можно использовать уже описанные выше объекты POINTS, CURVES, POLYGONS и TEXT — разумеется, с добавлением в списки параметров третьей координаты. Пример такого построения дан на рис. 11.32.

Кроме того, могут использоваться некоторые специальные трехмерные структуры. Одна из них — структура GRID:

- GRID(a..b,c..d,listlist) — задание поверхности над участком координатной плоскости, ограниченной отрезками $[a, b]$ и $[c, d]$, по данным, заданным переменной-списком listlist:=[[z11,...z1n],[z21,...z2n],...[zm1...zmn]] с размерностью $n?m$. Заметим, что эта переменная задает координату z для равноотстоящих точек поверхности.

На рис. 11.33 показан пример создания структуры трехмерной графики на базе GRID. Изображение представляет собой линии, соединяющие заданные точки.

Еще один тип трехмерной графической структуры — это MESH:

- MESH(listlist) — задание трехмерной поверхности по данным списочной переменной listlist, содержащей полные координаты всех точек поверхности (возможно задание последней при неравномерной сетке).

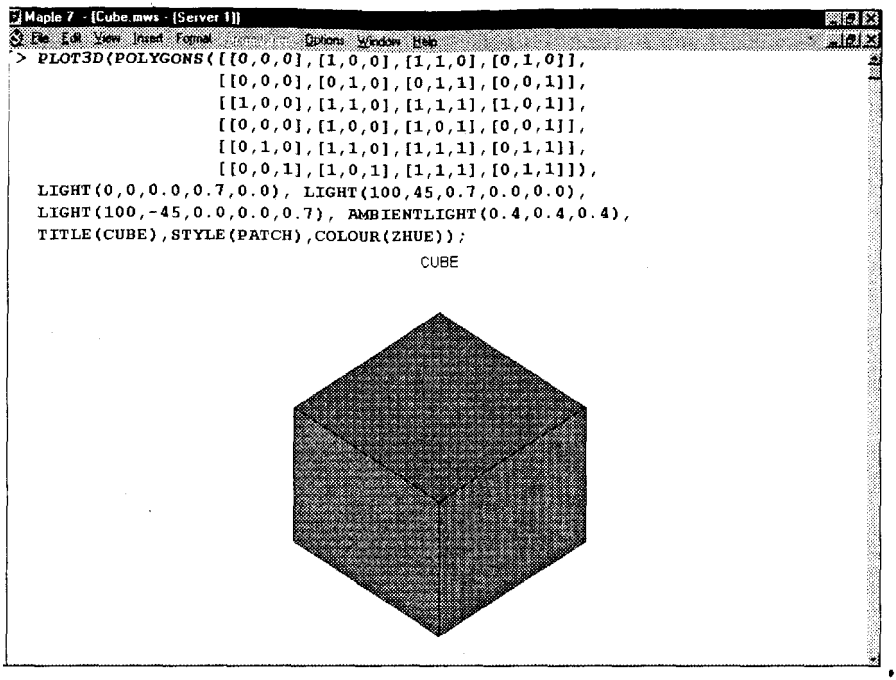


Рис. 11.32. Пример создания структуры трехмерной графики

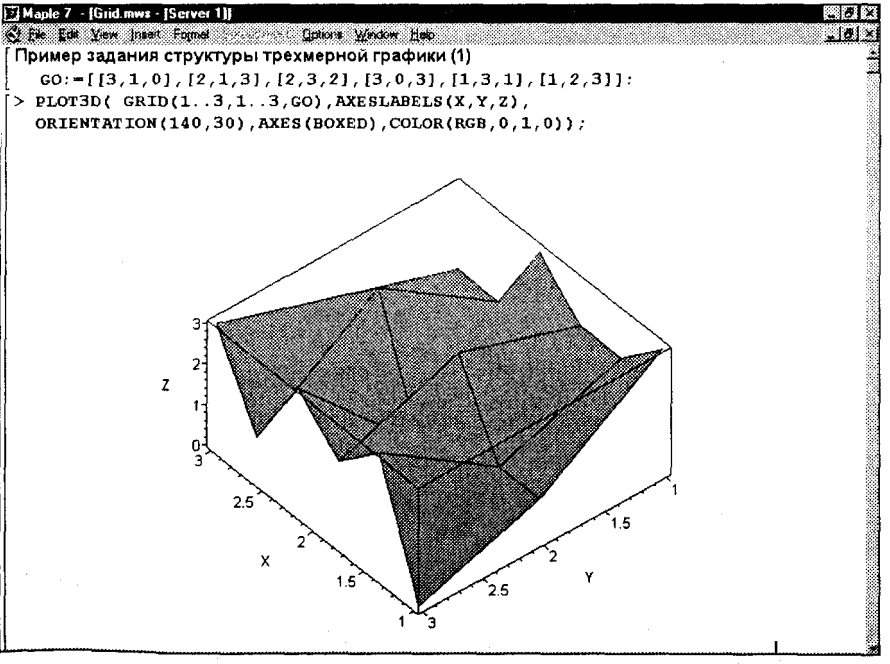


Рис. 11.33. Пример задания графической структуры типа GRID

Обычная форма задания этой структуры следующая:

```
MESH([[[x11,y11,z11],...[x1n,y1n,z1n]], [[x21,y21,z21],...[x2n,y2n,z2n]], ...
                                                    [[xm1,ym1,zm1]...[xmn,ymn,zmn]]])
```

Пример задания такой структуры представлен на рис. 11.34.

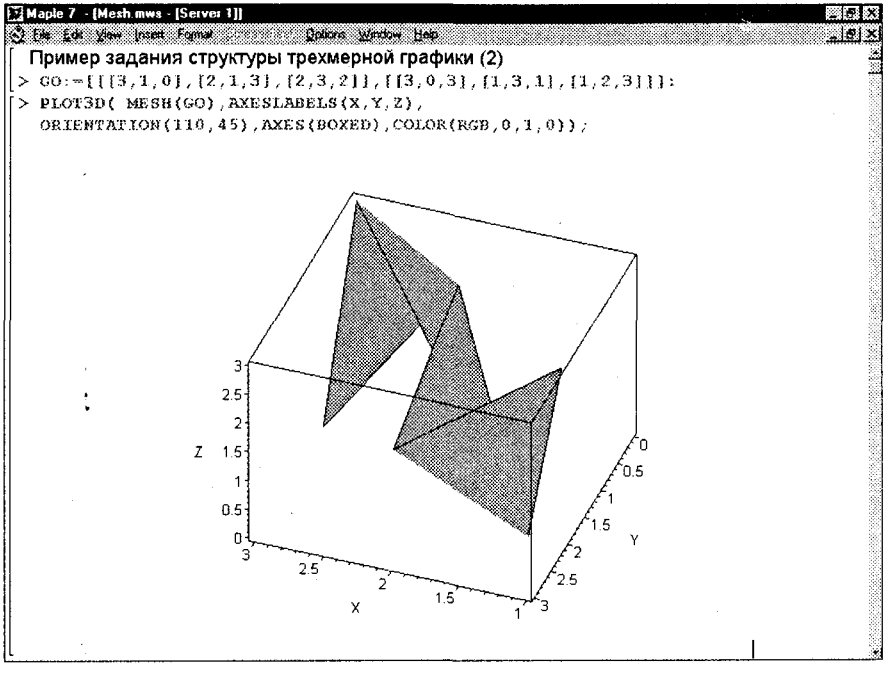


Рис. 11.34. Пример задания графической структуры типа MESH

Описанные структуры могут использоваться и в программных модулях. Много таких примеров описано в книгах, поставляемых с системой Maple 7.

Что нового мы узнали?

В этом уроке мы научились:

- Использовать основную функцию построения двумерных графиков — `plot`.
- Задавать координатные системы двумерных графиков.
- Управлять цветом и стилем двумерных графиков.
- Использовать основные типы двумерных графиков.
- Строить трехмерные графики.
- Строить различные поверхности.
- Выполнять быстрые (черновые) построения.
- Использовать специальные приемы для построения трехмерных графиков.
- Задавать графические структуры двумерной и трехмерной графики.

Расширенные средства графики

- ❒ Пакет plots
- ❒ Техника анимирования графиков
- ❒ Пакет plottools
- ❒ Расширенные средства графической визуализации
- ❒ Расширенная техника анимации
- ❒ Новая функция для построения стрелок arrow
- ❒ Построение сложных комбинированных графиков

Пакет plots

Общая характеристика пакета plots

Пакет plots содержит почти полсотни графических функций, существенно расширяющих возможности построения двумерных и трехмерных графиков в Maple 7:

```
> with(plots);
```

[*animate*, *animate3d*, *animatecurve*, *changecoords*, *complexplot*, *complexplot3d*, *conformal*, *contourplot*, *contourplot3d*, *coordplot*, *coordplot3d*, *cylinderplot*, *densityplot*, *display*, *display3d*, *fieldplot*, *fieldplot3d*, *gradplot*, *gradplot3d*, *implicitplot*, *implicitplot3d*, *inequal*, *listcontplot*, *listcontplot3d*, *listdensityplot*, *listplot*, *listplot3d*, *loglogplot*, *logplot*, *matrixplot*, *odeplot*, *pareto*, *pointplot*, *pointplot3d*, *polarplot*, *polygonplot*, *polygonplot3d*,

polyhedra_supported, *polyhedraplot*, *replot*, *rootlocus*, *semilogplot*, *setoptions*, *setoptions3d*, *spacecurve*, *sparsematrixplot*, *sphereplot*, *surfdata*, *textplot*, *textplot3d*, *tubeplot*]

Ввиду важности этого пакета отметим назначение всех его функций:

- `animate` — создает анимацию двумерных графиков функций;
- `animate3d` — создает анимацию трехмерных графиков функций;
- `animatecurve` — создает анимацию кривых;
- `changecoords` — смена системы координат;
- `complexplot` — построение двумерного графика на комплексной плоскости;
- `complexplot3d` — построение трехмерного графика в комплексном пространстве;
- `conformal` — конформный график комплексной функции;
- `contourplot` — построение контурного графика;
- `contourplot3d` — построение трехмерного контурного графика;
- `coordplot` — построение координатной системы двумерных графиков;
- `coordplot3d` — построение координатной системы трехмерных графиков;
- `cylinderplot` — построение графика поверхности в цилиндрических координатах;
- `densityplot` — построение двумерного графика плотности;

- `display` — построение графика для списка графических объектов;
- `display3d` — построение графика для списка трехмерных графических объектов;
- `fieldplot` — построение графика двумерного векторного поля;
- `fieldplot3d` — построение графика трехмерного векторного поля;
- `gradplot` — построение графика двумерного векторного поля градиента;
- `gradplot3d` — построение графика трехмерного векторного поля градиента;
- `implicitplot` — построение двумерного графика неявной функции;
- `implicitplot3d` — построение трехмерного графика неявной функции;
- `inequal` — построение графика решения системы неравенств;
- `listcontplot` — построение двумерного контурного графика для сетки значений;
- `listcontplot3d` — построение трехмерного контурного графика для сетки значений;
- `listdensityplot` — построение двумерного графика плотности для сетки значений;
- `listplot` — построение двумерного графика для списка значений;
- `listplot3d` — построение трехмерного графика для списка значений;
- `loglogplot` — построение логарифмического двумерного графика функции;
- `logplot` — построение полулогарифмического двумерного графика функции;
- `matrixplot` — построение трехмерного графика со значениями *Z*, определенными матрицей;
- `odeplot` — построение двумерного или трехмерного графика решения дифференциальных уравнений;
- `pareto` — построение диаграммы (гистограммы и графика);
- `pointplot` — построение точками двумерного графика;
- `pointplot3d` — построение точками трехмерного графика;
- `polarplot` — построение графика двумерной кривой в полярной системе координат;
- `polygonplot` — построение графика одного или нескольких многоугольников;
- `polygonplot3d` — построение одного или нескольких многоугольников;
- `polyhedraplot` — построение трехмерного многогранника;
- `replot` — перестроение графика заново;
- `rootlocus` — построение графика корней уравнения с комплексными неизвестными;
- `semilogplot` — построение графика функции с логарифмическим масштабом по оси абсцисс;
- `setoptions` — установка параметров по умолчанию для двумерных графиков;
- `setoptions3d` — установка параметров по умолчанию для трехмерных графиков;

- spacecurve — построение трехмерных кривых;
- sparsematrixplot — построение двумерного графика отличных от нуля значений матрицы;
- sphereplot — построение графика трехмерной поверхности в сферических координатах;
- surfdata — построение трехмерного графика поверхности по численным данным;
- textplot — вывод текста на заданное место двумерного графика;
- textplot3d — вывод текста на заданное место трехмерного графика;
- tubeplot — построение трехмерного графика типа «трубы».

Среди этих функций надо отметить прежде всего средства построения графиков ряда новых типов (например, в виде линий равного уровня, векторных полей и т. д.), а также средства объединения различных графиков в один. Особый интерес представляют две первые функции, обеспечивающие анимацию как двумерных (animate), так и трехмерных графиков (animate3d). Этот пакет вполне заслуживает описания в отдельной книге. Но, учитывая ограниченный объем данной книги, мы рассмотрим лишь несколько характерных примеров его применения. Заметим, что для использования приведенных функций нужен вызов пакета, например командой with(plots).

Построение графиков функций в двумерной полярной системе координат

В пакете plots есть функция для построения графиков в полярной системе координат. Она имеет вид polarplot(L,o), где L — объекты для задания функции, график которой строится, и o — необязательные параметры. На рис. 12.1, сверху, представлен пример построения графика с помощью функции polarplot.

В данном случае для большей выразительности опущено построение координатных осей, а график выведен линией удвоенной толщины. График очень напоминает лист клена, весьма почитаемого в Канаде и ставшего эмблемой Maple.

Построение двумерных графиков типа implicitplot

В математике часто встречается особый тип задания геометрических фигур, при котором переменные x и y связаны неявной зависимостью. Например, окружность задается выражением $x^2 + y^2 = R^2$, где R — радиус окружности. Для задания двумерного графика такого вида служит функция импликативной графики:

```
implicitplot(eqn,x=a..b,y=c..d,options)
```

Пример построения окружности с помощью этой функции показан на рис. 12.1, снизу. Чуть ниже мы рассмотрим подобную функцию и для трехмерного графика.

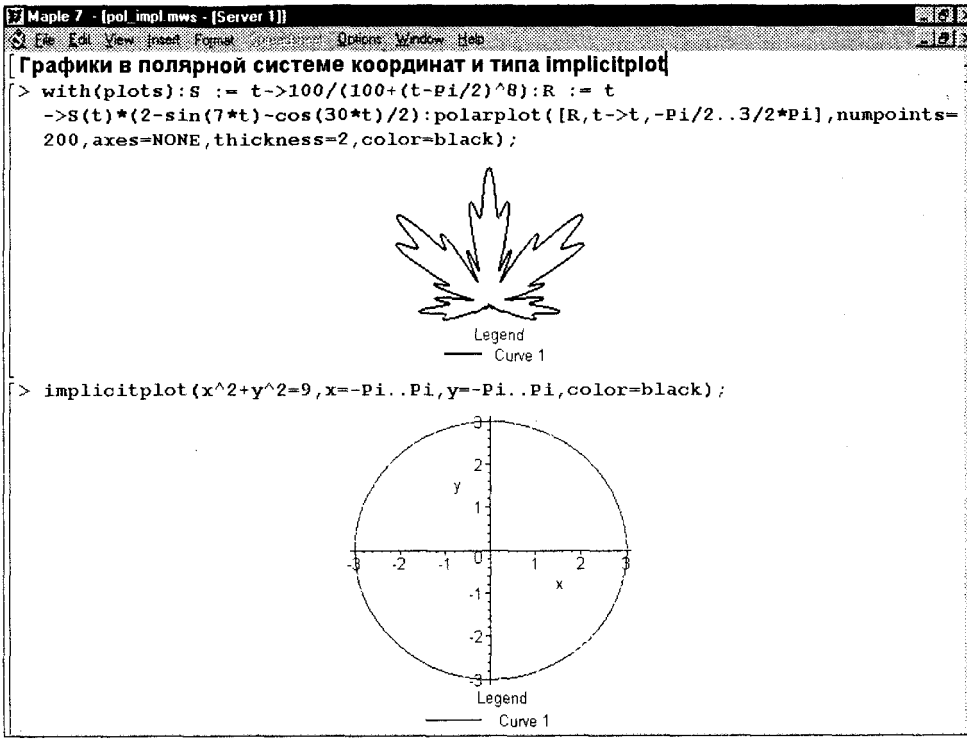


Рис. 12.1. Графики, построенные с помощью функций polarplot и implicitplot

Построение графиков линиями равного уровня

Графики, построенные с помощью линий равного уровня (их также называют контурными графиками), часто используются в картографии. Эти графики получаются, если мысленно провести через трехмерную поверхность ряд равноотстоящих плоскостей, параллельных плоскости, образованной осями X и Y графика. Линии равных высот образуются в результате пересечения этих плоскостей с трехмерной поверхностью.

Для построения таких графиков используется функция `contourplot`, которая может использоваться в нескольких форматах:

```
contourplot(expr1,x=a..b,y=c..d)
contourplot(f,a..b,c..d)
contourplot([exprf,exprg,exprh ],s=a..b,t=c..d)
contourplot([f,g,h ],a..b,c..d)
contourplot3d(expr1,x=a..b,y=c..d)
contourplot3d(f,a..b,c..d)
contourplot3d([exprf,exprg,exprh],s=a..b,t=c,.d)
contourplot3d([f,g,h ],a..b,c..d)
```

Здесь f, g и h — функции; expr1 — выражение, описывающее зависимость высоты поверхности от координат x и y; exprf, exprg и exprh — выражения, зависящие от s и t, описывающие поверхность в параметрической форме; a и b — константы вещественного типа; c и d — константы или выражения вещественного типа; x, y, s и t — имена независимых переменных.

На рис. 12.2 показано построение графика линиями равного уровня для одной функции. Параметр filled=true обеспечивает автоматическую функциональную окраску замкнутых фигур, образованных линиями равного уровня. Порою это придает графику большую выразительность, чем при построении только линий равного уровня.

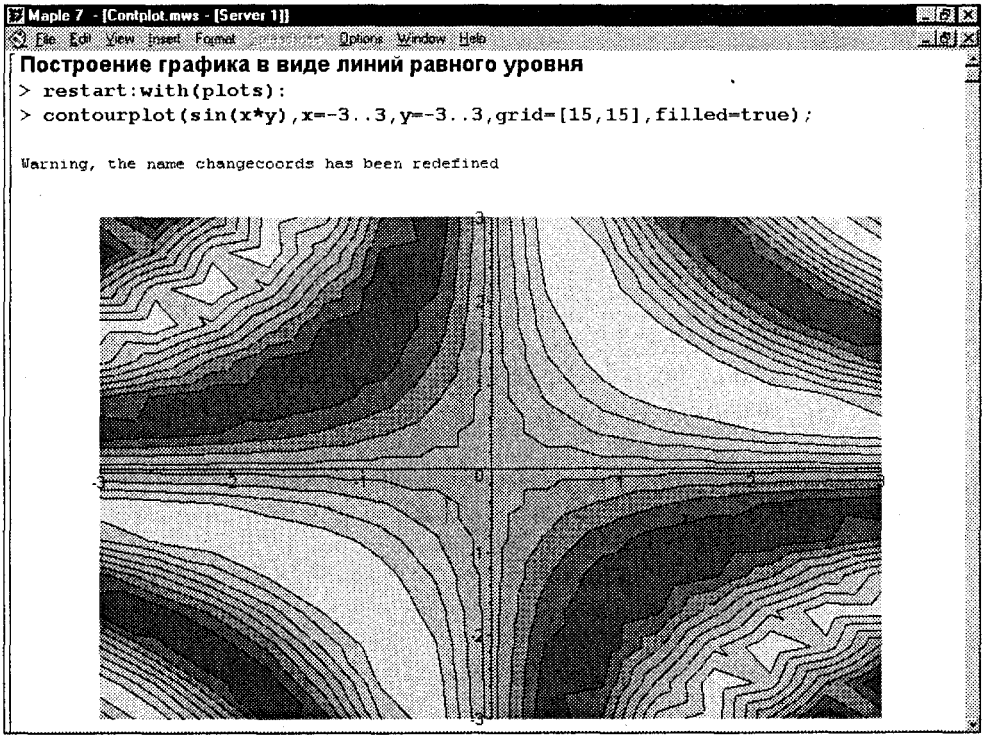


Рис. 12.2. Пример построения графика функции линиями равного уровня

Обратите внимание на то, что данная функция по умолчанию строит легенду — она видна под графиком в виде линий с надписями. К сожалению, в данном варианте окраски сами контурные линии получаются черными и их невозможно отличить. Однако если убрать параметр filled=true, то контурные линии (и линии легенды) будут иметь разный цвет и легко различаться.

Функция contourplot позволяет строить и графики ряда функций. Пример такого построения показан на рис. 12.3. Множество окружностей на этом рисунке создается четырьмя поверхностями, заданными функциями c1, c2, c3 и c4.

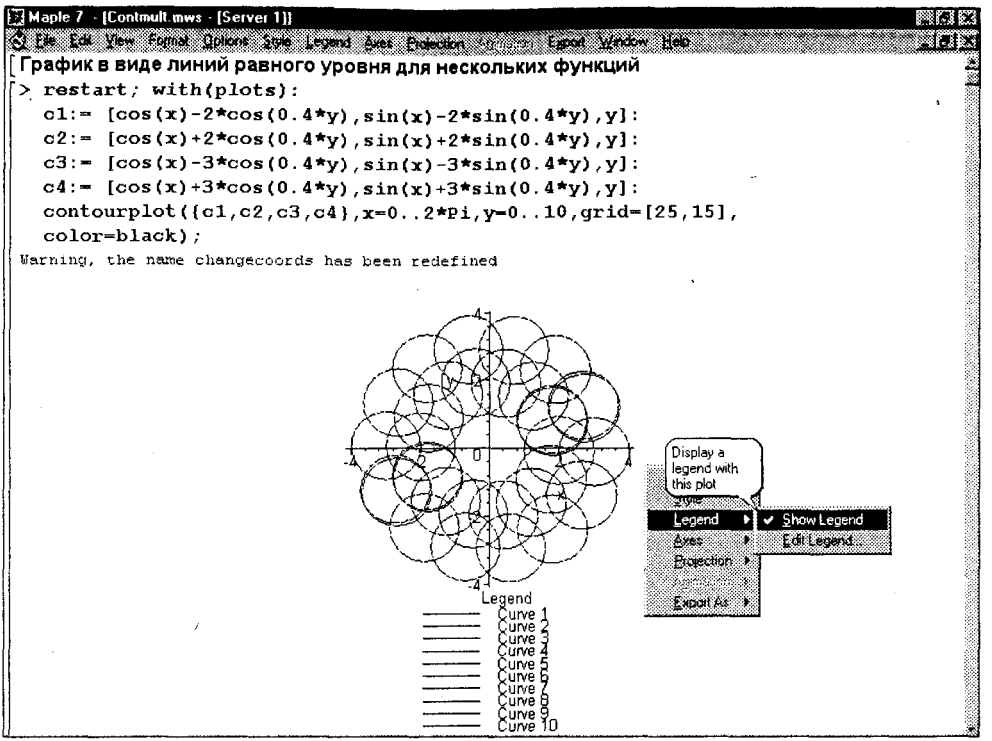


Рис. 12.3. Пример построения графиков многих функций линиями равного уровня

ВНИМАНИЕ

Обратите внимание, что на многих графиках Maple 7 по умолчанию вписывает легенду, то есть список линий с обозначениями. Иногда (как, например, на рис. 12.3) этот список оказывается просто некстати. Легенду можно убрать, расширив заодно место для графика, сняв флажок Show Legend в меню Legend, которое появляется при двойном щелчке на графике (это меню видно на рис. 12.3). То же самое можно сделать с помощью той же команды в контекстном меню. Заодно запомните, что легенду можно редактировать, выполнив команду Edit Legend.

Следует отметить, что хотя графики в виде линий равного уровня выглядят не так эстетично и естественно, как обычные графики трехмерных поверхностей (ибо требуют осмысления результатов), у них есть один существенный плюс — экстремумы функций на таких графиках выявляются порой более четко, чем на обычных графиках. Например, небольшая возвышенность или впадина за большой «горой» на обычном графике может оказаться невидимой, поскольку заслоняется «горой». На графике линий равного уровня этого эффекта нет. Однако выразительность таких графиков сильно зависит от числа контурных линий.

График плотности

Иногда поверхности отображаются на плоскости как графики плотности окраски — чем выше высота поверхности, тем плотнее (темнее) окраска. Такой вид графиков создается функцией densityplot. Она может записываться в двух форматах:

```
densityplot(exprl,x=a..b,y=c..d)
densityplot(f,a..b,c..d)
```

где назначение параметров соответствует указанному выше для функции contourplot.

На рис. 12.4 (верхняя часть) дан пример построения графика такого типа. Нетрудно заметить, что в плоскости *XY* график разбит на квадраты, плотность окраски которых различна. В нашем случае плотность окраски задается оттенками серого цвета.

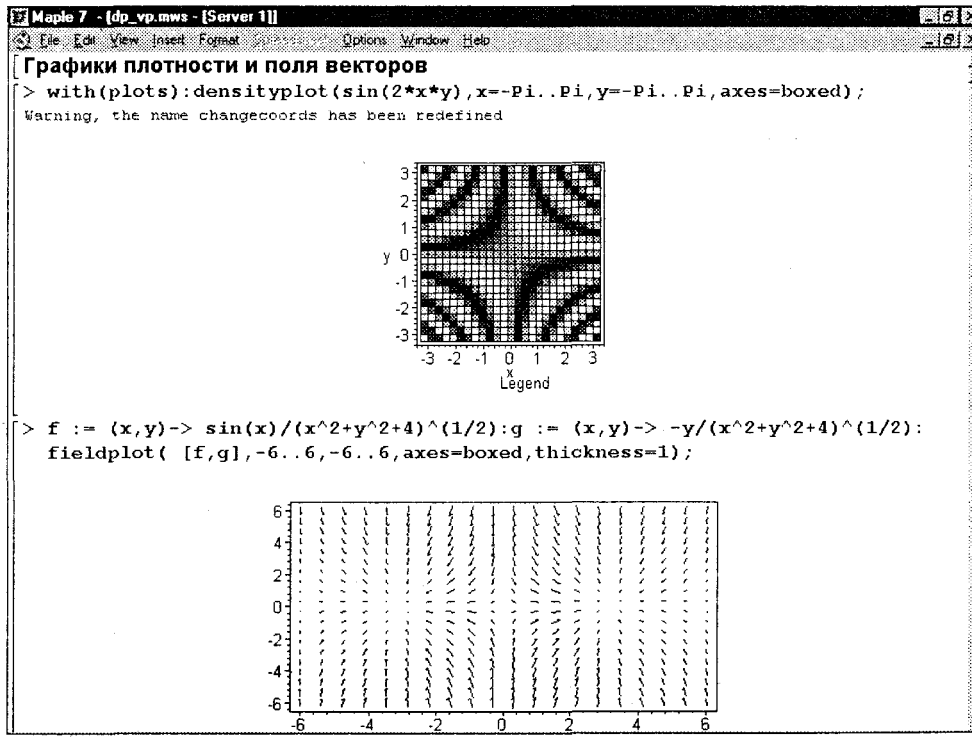


Рис. 12.4. Графики плотности и поля векторов

Обычно графики такого типа не очень выразительны, но имеют свои области применения. К примеру, оттенки окраски полупрозрачной жидкости могут указывать на рельеф поверхности дна емкости, в которой находится эта жидкость.

Двумерный график векторного поля

Еще один распространенный способ представления трехмерных поверхностей — графики полей векторов. Они часто применяются для отображения полей, например электрических зарядов. Особенность таких графиков в том, что для их построения используют стрелки, направление которых соответствует направлению изменения градиента поля, а длина — значению градиента. Так что термин «поле векторов» надо понимать в смысле, что поле графика заполнено векторами.

Для построения таких графиков в двумерной системе координат используется функция `fieldplot`:

```
fieldplot(f, r1, r2)
fieldplot(f, r1, r2, ...)
```

где `f` — вектор или множество векторов, задающих построение; `r1` и `r2` — пределы.

На рис. 12.4 в нижней части документа показан вид одного из таких графиков. Следует отметить, что для получения достаточного числа отчетливо видных стрелок надо поработать с форматированием графиков. Иначе графики этого типа могут оказаться не очень представительными. Так, слишком короткие стрелки превращаются в черточки и даже точки, не имеющие острия, что лишает графики наглядности.

Несколько позже мы рассмотрим построение на одном рисунке графиков плотности и векторного поля, а также создание более наглядных толстых стрелок.

Трехмерный график типа implicitplot3d

Трехмерные поверхности также могут задаваться уравнениями неявного вида. В этом случае для построения их графиков используется функция `implicitplot3d`:

```
implicitplot3d(expr1,x=a..b,y=c..d,z=p..q,<options>)
implicitplot3d(f,a..b,c..d,p..q,<options>)
```

На рис. 12.5 показаны два примера построения объемных фигур с помощью функции `implicitplot3d`.

Эти примеры хорошо иллюстрируют технику применения функции `implicitplot3d`. С ее помощью можно строить весьма своеобразные фигуры, что, впрочем, видно и из приведенных примеров. Для наглядности фигур на рис. 12.5 они несколько развернуты в пространстве с помощью мыши.

Графики в разных системах координат

В пакете `plots` имеется множество функций для построения графиков в различных системах координат. Объем книги не позволяет воспроизвести примеры всех видов таких графиков, ибо их многие сотни. Да это и не надо — во встроенных в справочную систему примерах можно найти все нужные сведения. Так что ограничимся лишь парой примеров применения функции `tubeplot(C, options)`,

позволяющей строить весьма наглядные фигуры в пространстве, напоминающие трубы или иные объекты, образованные фигурами вращения.

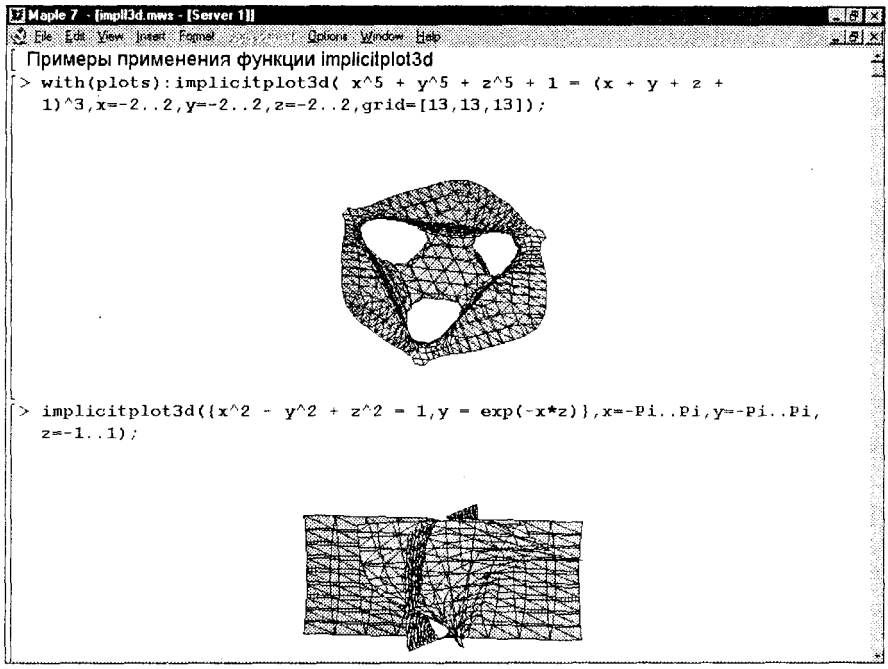


Рис. 12.5. Примеры применения функции implicitplot3d

На рис. 12.6 показана одна из таких фигур. Она поразительно напоминает раковину улитки. Функциональная окраска достигнута доработкой графика с помощью панели форматирования.

Эта функция может использоваться и для построения ряда трубчатых объектов в пространстве. При этом автоматически задается алгоритм удаления невидимых линий даже для достаточно сложных фигур. Это наглядно иллюстрирует пример на рис. 12.7, показывающий фигуру «цепи». Не правда ли, реалистичность этой фигуры поражает воображение?

Можно долго размышлять о том, как те или иные математические закономерности описывают предметы реального мира, положенные в основу тех или иных геометрических объектов, или, возможно, о гениальности людей, сумевших найти такие закономерности для многих из таких объектов. В наше время Maple 7 открывает огромные возможности для таких людей.

Графики типа трехмерного поля из векторов

Наглядность ряда графиков можно существенно увеличить, строя их в трехмерном представлении. Например, для такого построения графиков полей из векторов можно использовать графическую функцию `fieldplot3d`. В отличие от функции `fieldplot` она строит стрелки как бы в трехмерном пространстве (рис. 12.8).

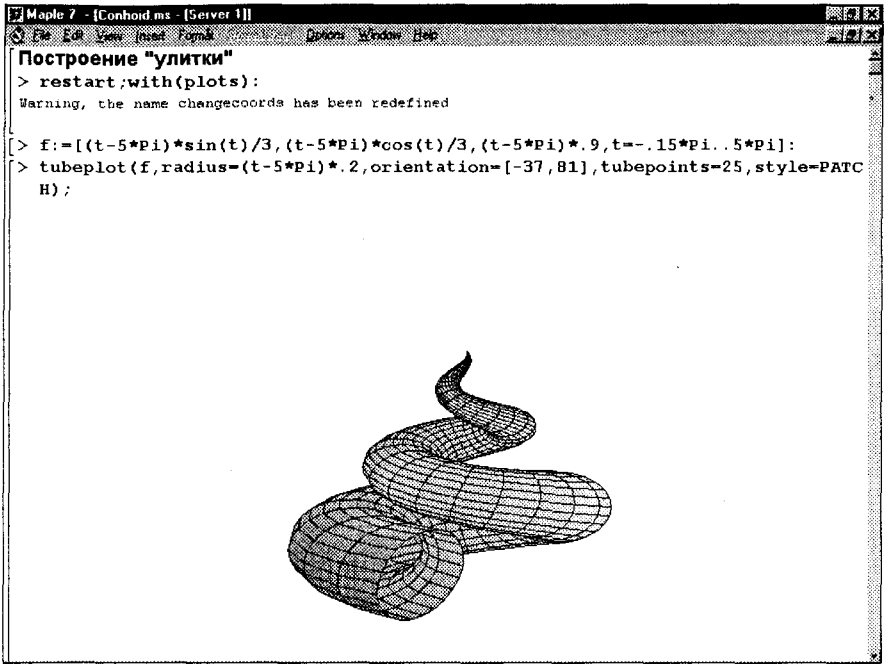


Рис. 12.6. Построение графика-«улитки»

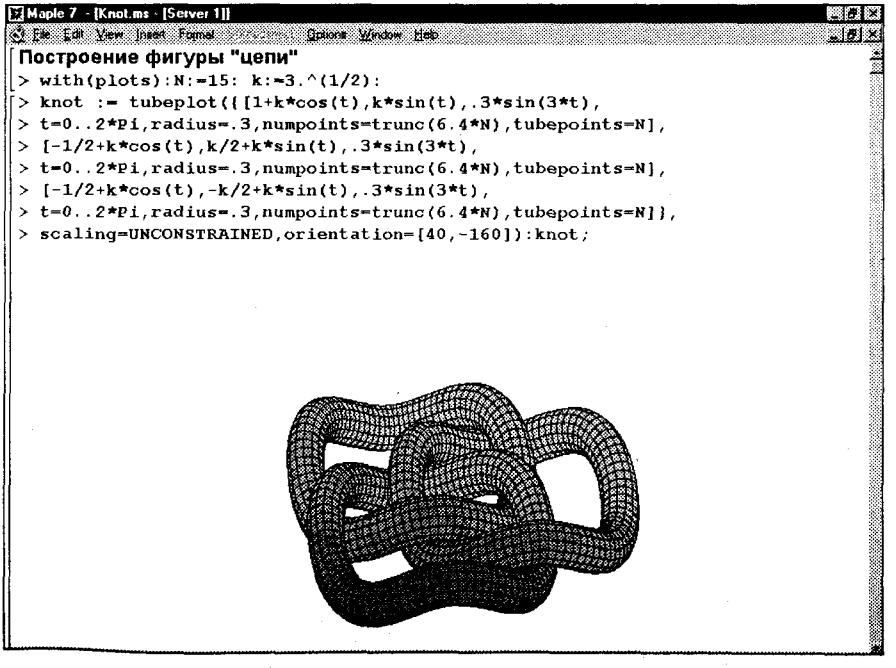


Рис. 12.7. Фигура «цепи», построенная с применением функции tubeplot

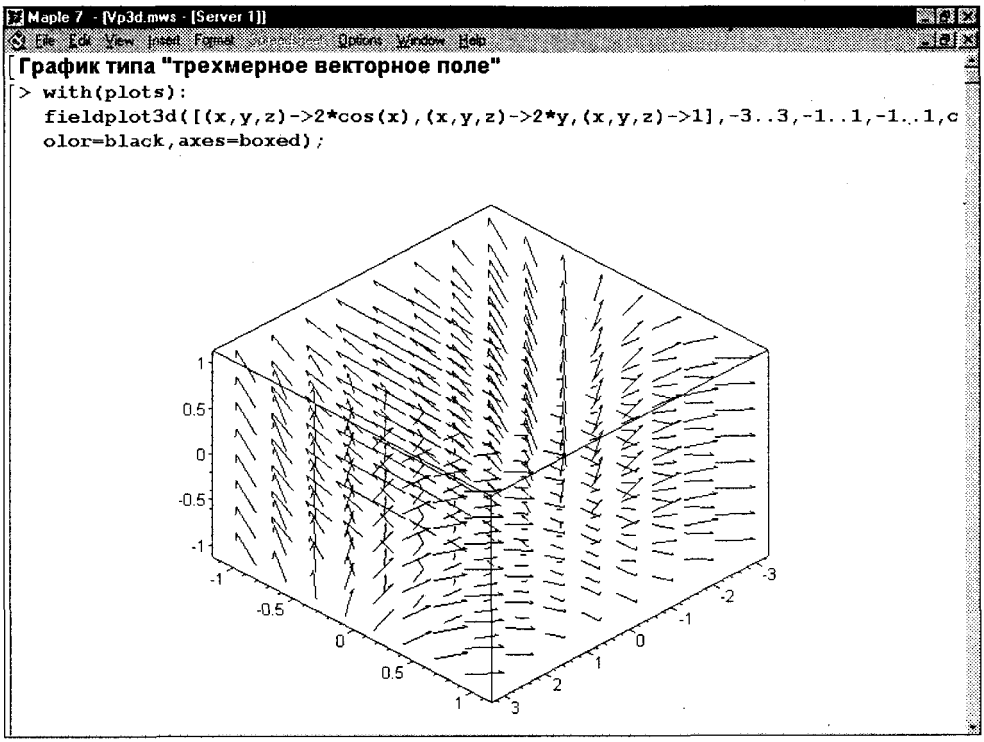


Рис. 12.8. Построение поля в трехмерном пространстве с помощью векторов

Все сказанное об особенностях таких двумерных графиков остается справедливым и для графиков трехмерных. В частности, для обеспечения достаточной наглядности нужно тщательно отлаживать форматы представления таких графиков.

Контурные трехмерные графики

В отличие от векторных графиков контурные графики поверхностей, наложенные на сами эти поверхности, нередко повышают восприимчивость таких поверхностей — подобно изображению линий каркаса. Для одновременного построения поверхности и контурных линий на них служит функция `contourplot3d`. Пример ее применения показан на рис. 12.9.

Для повышения наглядности этот график доработан с помощью контекстной панели инструментов графиков. В частности, включена функциональная окраска и подобраны углы обзора фигуры, при которых отчетливо видны ее впадина и пик.

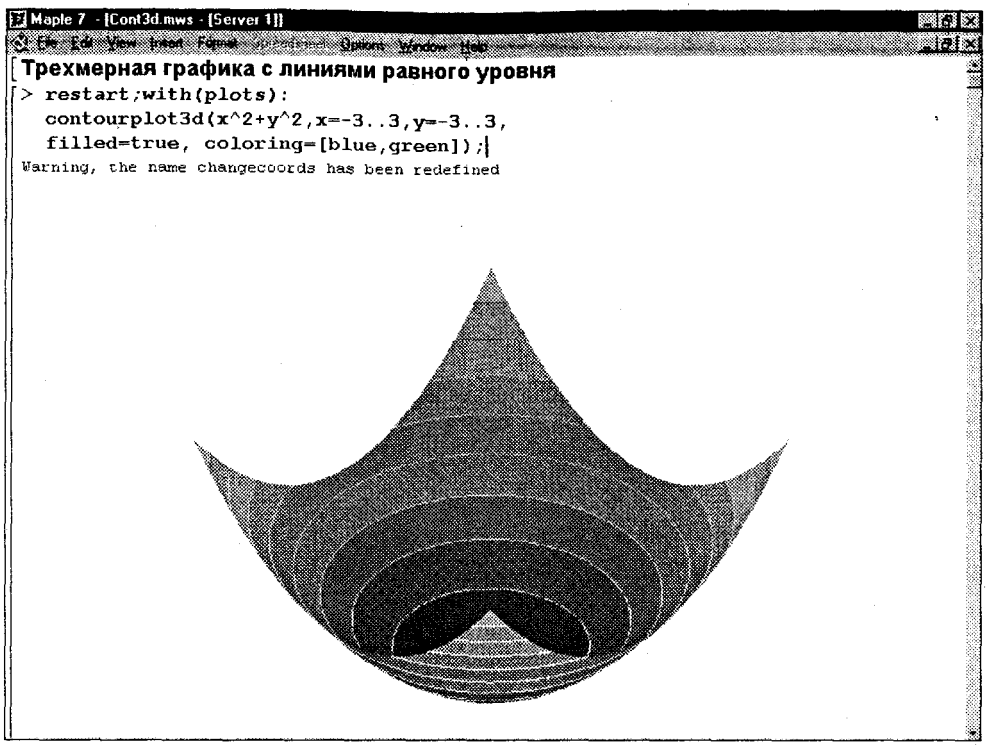


Рис. 12.9. График поверхности с контурными линиями

Техника визуализации сложных пространственных фигур

Приведенные выше достаточно простые примеры дают представление о высоком качестве визуализации геометрических фигур с помощью пакета `plots`. Здесь мы рассмотрим еще несколько примеров визуализации трехмерных фигур. Многие видели катушки индуктивности, у которых провод того или иного диаметра намотан на тороидальный магнитный сердечник. Некую математическую абстракцию такой катушки иллюстрирует рис. 12.10.

В документе рис. 12.10 для функции `tubeplot` использовано довольно большое число параметров. Не всегда их действие очевидно. Поэтому на рис. 12.11 показано построение трех взаимно пересекающихся торов с разными наборами параметров. Этот рисунок дает также наглядное представление о возможности построения нескольких графических объектов (представленных функциями `p1`, `p2` и `p3`) с помощью функции `tubeplot`.

Maple 7 - [Gtortub.ms - [Server 1]]

File Edit View Insert Format Options Window Help

Построение спирали на торе

```
> with(plots): N:=10:
> tortube := tubeplot([10*cos(t),10*sin(t),0,
> t=0..2*Pi,radius=2,numpoints=10*N,tubepoints=2*N]
> ,[cos(t)*(10+4*sin(9*t)),sin(t)*(10+4*sin(9*t)),4*cos(9*t),t=0..2*Pi,radius
  =1,numpoints=trunc(37.5*N),tubepoints=N]},scaling=UNCONSTRAINED,orientation
  =[60,40]):tortube;
```

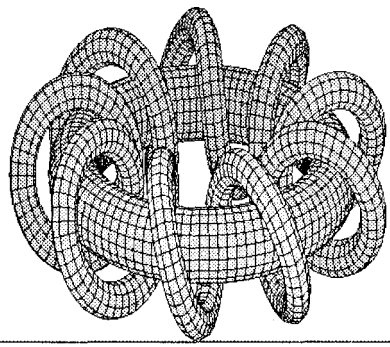


Рис. 12.10. Тор с обмоткой — толстой спиралью

Maple 7 - [3d3style.mws - [Server 1]]

File Edit View Insert Format Options Window Help

Задание трех трехмерных графиков в трех разных стилях

```
> with(plots):
> opts := t=0..2*Pi,numpoints=100,radius=.2:
> p1 := tubeplot([cos(t),sin(t),0],opts, style=PATCH):
> p2 := tubeplot([0,cos(t),sin(t)],opts, style=POINT,color=black,radius=.1):
> p3 := tubeplot([sin(t),0,cos(t)],opts,
  style=CONTOUR,color=black,radius=.1):
> display({p1,p2,p3});
```

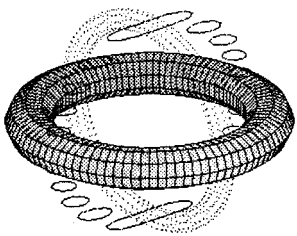


Рис. 12.11. Три пересекающихся тора с разными стилями построения

Наконец, на рис. 12.12 показано построение тора с тонкой обмоткой. Рекомендуется внимательно посмотреть на запись функции tubeplot в этом примере и в примере, показанном на рис. 12.11. Можно также поэкспериментировать с управляющими параметрами графика, от которых сильно зависят его представительность и наглядность.

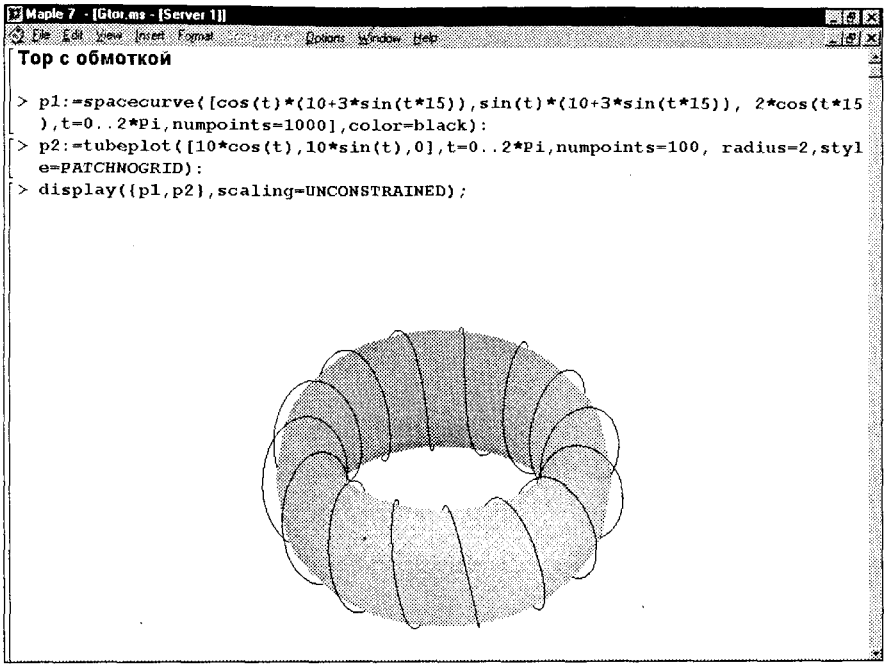


Рис. 12.12. Тор с тонкой обмоткой

В ряде случаев наглядно представленные фигуры можно строить путем объединения однотипных фигур. Пример графика подобного рода представлен на рис. 12.13. Здесь готовится список графических объектов s, смещенных по вертикали. С помощью функции display они воспроизводятся на одном графике, что повышает реалистичность изображения.

Последний пример имеет еще одну важную особенность — он иллюстрирует задание графической процедуры, в теле которой используются функции пакета plots. Параметр n этой процедуры задает число элементарных фигур, из которых строится полная фигура. Таким образом, высотой фигуры (или шириной «шины») можно управлять. Возможность задания практически любых графических процедур средствами Maple-языка существенно расширяет возможности Maple.

Наглядность таких графиков, как графики плотности и векторных полей может быть улучшена их совместным применением. Такой пример показан на рис. 12.14.

Maple 7 - [Gproc.ms - [Server 1]]

File Edit View Insert Format Options Window Help

Пример построения графика с использованием процедуры

```
> with(plots):
> torii := proc(n)
>  local s,k;
>  s := seq(tubeplot([cos(t),sin(t),k],
>   t=0..2*Pi,numpoints=100,radius=.25),k=0..n-1);
>  display({s});
> end:
> torii(5);
```

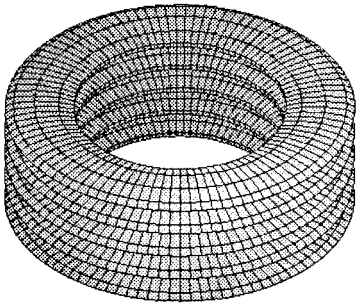


Рис. 12.13. Построение фигуры, напоминающей шину автомобиля

Maple 7 - [Gr_dens.mws - [Server 1]]

File Edit View Insert Format Options Window Help

Совместное применение функций gradplot и densityplot пакета plots

```
> with(plots): f:=-sin(5*x)+sin(5*y):
  p1:=gradplot(f,x=-1..1,y=-1..1,grid=[15,15],arrows=THICK):p2:=densityplot(f
  ,x=-1..1,y=-1..1,color=red):
  display(p1,p2,axes=BOX);
```

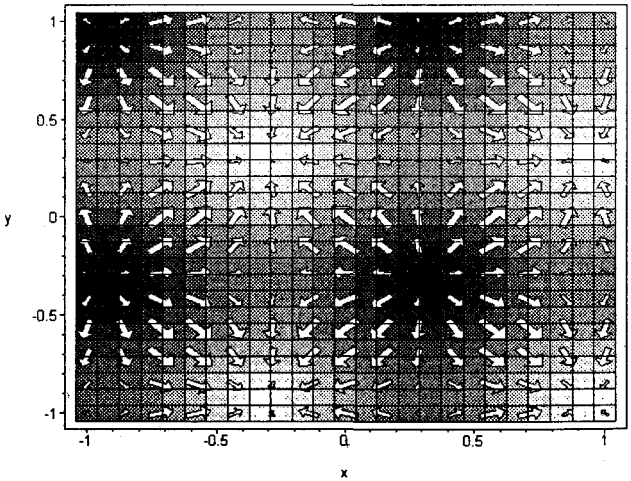


Рис. 12.14. Пример совместного применения графиков плотности и векторного поля

Этот пример иллюстрирует использование «жирных» стрелок для обозначения векторного поля. Наглядность графика повышается благодаря наложению стрелок на график плотности, который лучше, чем собственно стрелки, дает представление о плавности изменения высоты поверхности, заданной функцией *f*.

Техника анимирования графиков

Анимация двумерных графиков

Визуализация графических построений и результатов моделирования различных объектов и явлений существенно повышается при использовании средств «оживления» (анимации) изображений. Пакет plots имеет две простые функции для создания анимированных графиков.

Первая из этих функций служит для создания анимации графиков, представляющих функцию одной переменной $F(x)$:

```
animatecurve(F, r, ...)
```

Эта функция просто позволяет наблюдать медленное построение графика. Формат ее применения подобен используемому в функции `plot`.

При вызове данной функции вначале строится пустой шаблон графика. Если активизировать шаблон мышью, то в строке главного меню появляется меню Animation. Меню Animation содержит команды управления анимацией. Такое же подменю появляется и в контекстном (рис. 12.15).

Указанное подменю содержит следующие команды анимации:

- Play — запуск построения графика;
- Next — выполнение следующего шага анимации;
- Backward/Forward — переключение направления анимации (назад/вперед);
- Faster — ускорение анимации;
- Slower — замедление анимации;
- Continiuus/Single cycle — цикличность анимации.

При исполнении команды Play происходит построение кривой (или нескольких кривых). В зависимости от выбора команд Faster или Slower построение идет быстро или медленно. Команда Next выполняет один шаг анимации — построение очередного фрагмента кривой. Переключатель Backward/Forward позволяет задать направление построения кривой — от начала к концу или от конца к началу. Построение может быть непрерывным или циклическим в зависимости от состояния позиции Continiuus/Single cycle в подменю управления анимацией. При циклической анимации число циклов задается параметром `frames=n`.

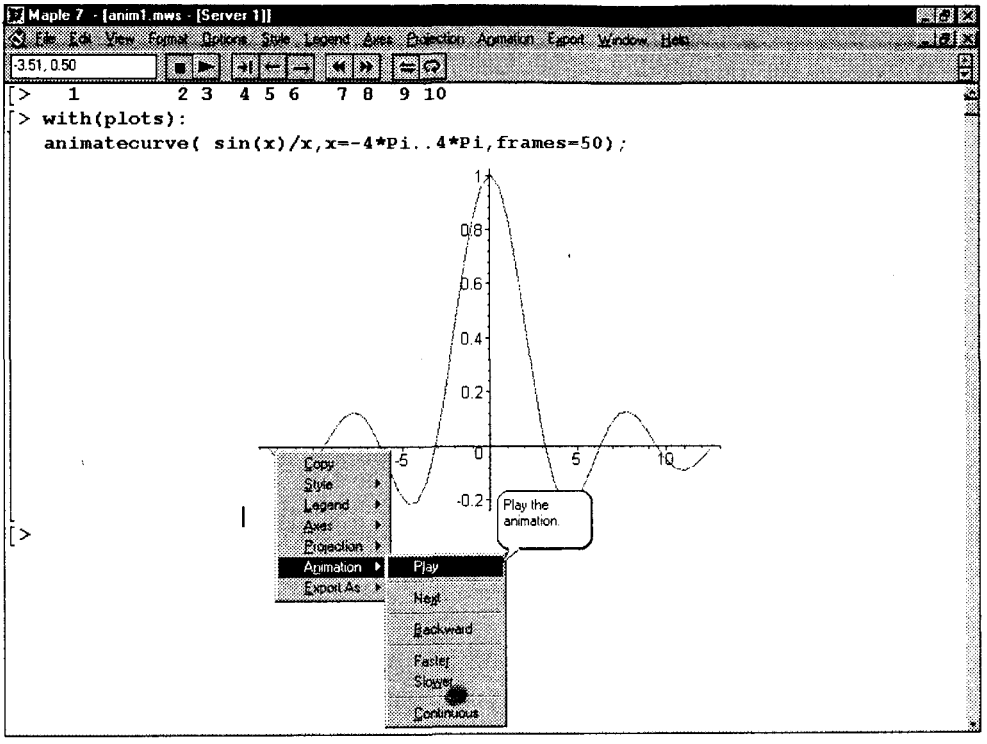


Рис. 12.15. Пример анимационного построения графика функцией animatecurve

Проигрыватель анимированной графики

При включенном выводе панели форматирования во время анимации она приобретает вид панели проигрывателя клипов (рис. 12.15). Эта панель имеет кнопки управления с обозначениями, принятыми у современных магнитофонов:

1. Поле координат перемещающейся точки графика.
2. Остановка анимации.
3. Пуск анимации.
4. Переход к следующему кадру (фрейму).
5. Установка направления анимации от конца в начало.
6. Установка направления анимации из начала в конец (по умолчанию).
7. Уменьшение времени шага анимации.
8. Увеличение времени шага анимации.
9. Установка одиночного цикла анимации.
10. Установка серии циклов анимации.

Итак, кнопки проигрывателя, по существу, повторяют команды подменю управления анимацией.

Нажав кнопку пуска (с треугольником, острием обращенным вправо), можно наблюдать изменение вида кривой для функции $\sin(x)/(x)$. Другие кнопки управляют характером анимации. Проигрыватель дает удобные средства для демонстрации анимации, например, во время занятий со школьниками или студентами.

Построение двумерных анимированных графиков

Более обширные возможности анимации двумерных графиков обеспечивает функция `animate`:

```
animate(F, x, t)
animate(F, x, t, o)
```

В ней параметр `x` задает пределы изменения переменной x, а параметр `t` — пределы изменения дополнительной переменной t. Суть анимации при использовании данной функции заключается в построении серии кадров (как в мультфильме), причем каждый кадр связан со значением изменяемой во времени переменной `t`. Если надо явно задать число кадров анимации `N`, то в качестве `o` следует использовать `frame=N`.

Рисунок 12.16 показывает применение функции `animate`.

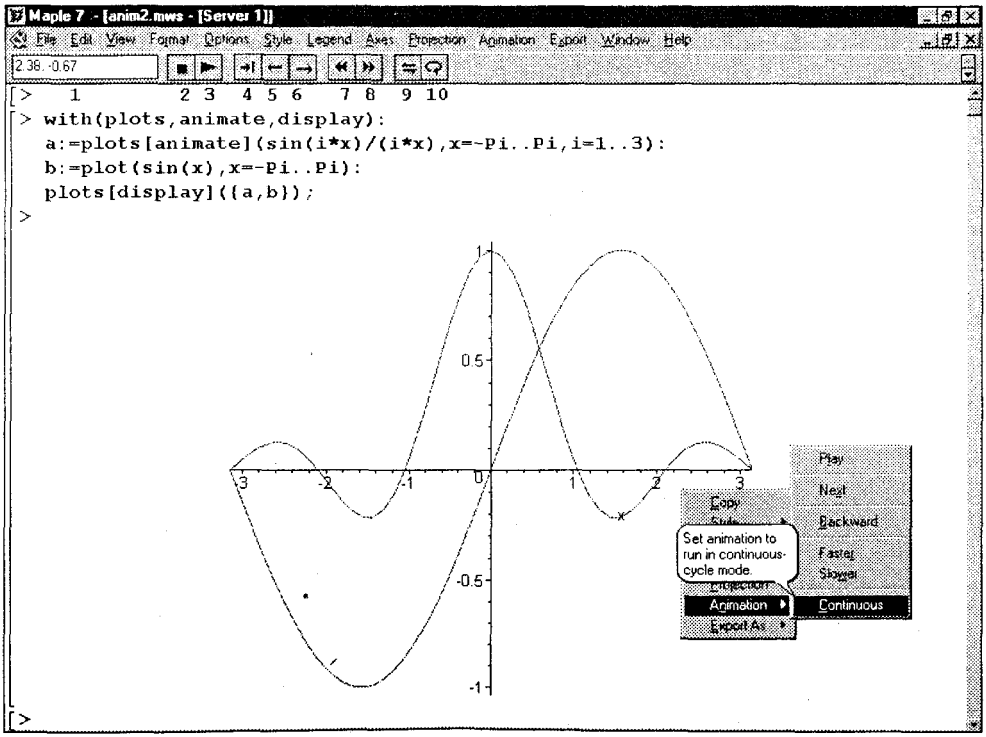


Рис. 12.16. Анимация функции $\sin(i*x)/(i*x)$ на фоне неподвижной синусоиды

В документе рис. 12.16 строятся две функции — не создающая анимации функция $\sin(x)$ и создающая анимацию функция $\sin(i*x)/(i*x)$, причем в качестве переменной t задана переменная i. Именно ее изменение и создает эффект анимации. Проигрыватель анимационных клипов и меню, описанные выше, могут использоваться для управления и этим видом анимации. Обратите внимание на вызов графических функций в этом примере командой `with` и на синтаксис записи этих функций.

К сожалению, картинки в книгах всегда неподвижны и воспроизвести эффект анимации невозможно. Можно лишь представить несколько текущих кадров анимации. Представленная на рис. 12.16 картина соответствует последнему кадру анимации.

Еще один пример анимации представлен на рис. 12.17. Этот документ показывает кадр анимированного процесса улучшения приближения синусоидальной функции рядом с различным числом членов (и порядком последнего члена ряда). Результирующая картина, изображенная на рис. 12.17, показывает как приближаемую синусоидальную функцию, так и графики всех рядов, которые последовательно выводятся в ходе анимации.

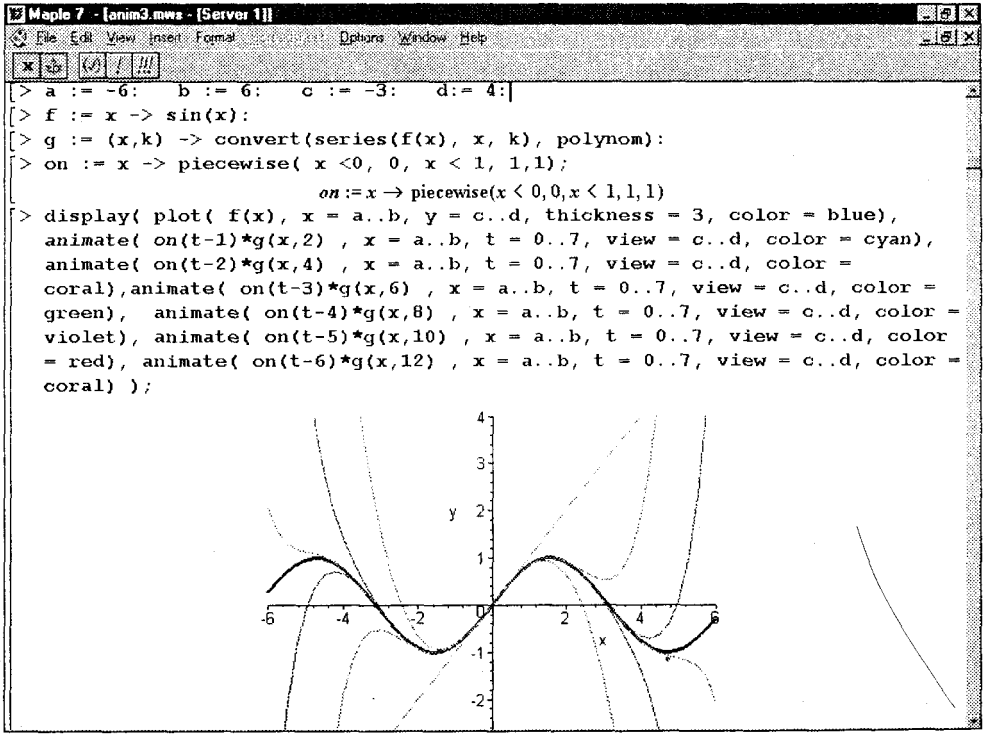


Рис. 12.17. Анимационная демонстрация приближения синусоиды рядом с меняющимся числом членов

Анимация графиков может найти самое широкое применение при создании учебных материалов. С ее помощью можно акцентировать внимание на отдель-

ных параметрах графиков и образующих их функций и наглядно иллюстрировать характер их изменений.

Построение трехмерных анимационных графиков

Аналогичным образом может осуществляться и анимирование трехмерных фигур. Для этого используется функция animate3d:

```
animate3d(F,x, y,t,o)
```

Здесь F — описание функции (или функций); x, y и t — диапазоны изменения переменных x, y и t. Для задания числа кадров N надо использовать необязательный параметр o в виде frame=N.

На рис. 12.18 показано построение анимированного графика. После задания функции, график которой строится, необходимо выделить график и запустить проигрыватель, как это описывалось для анимации двумерных графиков.

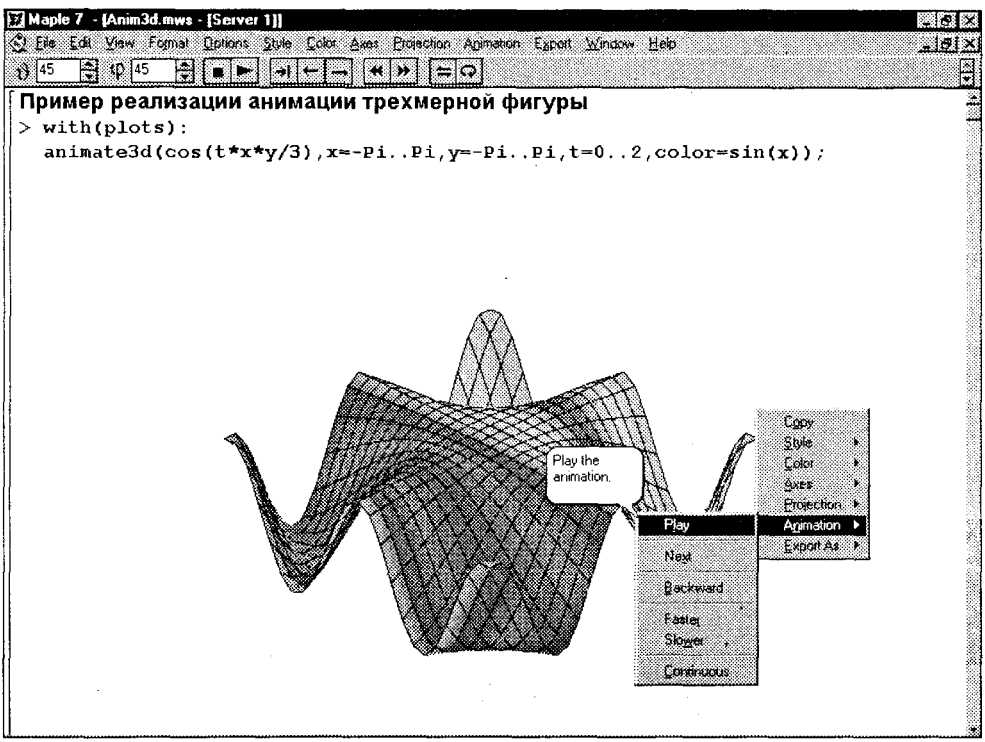


Рис. 12.18. Подготовка трехмерного анимационного графика

На рис. 12.18 показано также контекстное меню поля выделенного графика. Нетрудно заметить, что с помощью этого меню (и содержащихся в нем подменю) можно получить доступ к параметрам трехмерной графики и выполнить

необходимые операции форматирования, такие как включение цветовой окраски, выбор ориентации фигуры и т. д.

Назначение параметров, как и средств управления проигрывателем анимационных клипов, было описано выше.

Анимация с помощью параметра insequence

Еще один путь получения анимационных рисунков — создание ряда графических объектов p1, p2, p3 и т. д. и их последовательный вывод с помощью функций display или display3d:

```
display(p1,p2,p3,….,insequence=true)
display3d(p1,p2,p3….,insequence=true)
```

Здесь основным моментом является применение параметра insequence=true. Именно он обеспечивает вывод одного за другим серии графических объектов p1, p2, p3 и т. д. При этом объекты появляются по одному и каждый предшествующий объект стирается перед появлением нового объекта.

Графика пакета plottools

Примитивы пакета plottools

Инструментальный пакет графики plottools служит для создания графических примитивов, строящих элементарные геометрические объекты на плоскости и в пространстве: отрезки прямых и дуг, окружности, конусы, кубики и т. д. Его применение позволяет разнообразить графические построения и строить множество графиков специального назначения. В пакет входят следующие графические примитивы:

arc	arrow	circle	cone	cuboid
curve	cutin	cutout	cylinder	disk
dodecahedron	ellipse	ellipticArc	hemisphere	hexahedron
hyperbola	icosahedron	line	octahedron	pieslice
point	polygon	rectangle	semitorus	sphere
tetrahedron	torus			

ПРИМЕЧАНИЕ Вызов перечисленных примитивов осуществляется после загрузки пакета в память компьютера командой with(plottools). Только после этого примитивы пакета становятся доступными. Обычно примитивы используются для задания графических объектов, которые затем выводятся функцией display. Возможно применение этих примитивов совместно с различными графиками.

Большинство примитивов пакета plottools имеет довольно очевидный синтаксис. Например, для задания дуги используется примитив arc(c, r, a..b,…), где c — список с координатами центра окружности, к которой принадлежит дуга, r — радиус этой окружности, a..b — диапазон углов. На месте многоточия мо-

гут стоять обычные параметры, задающие цвет дуги, толщину ее линии и т. д. Конус строится примитивом cone(c,r,h...), где c — список с координатами центра, r — радиус основания, h — высота и т. д. Все формы записи графических примитивов и их синтаксис можно найти в справочной системе. В необходимых случаях стоит проверить синтаксис того или иного примитива с помощью справки по пакету plottools.

Примеры применения двумерных примитивов пакета plottools

На рис. 12.19 показано применение нескольких примитивов двумерной графики для построения дуги, окружности, закрашенного красным цветом эллипса и отрезка прямой. Кроме того, на графике показано построение синусоиды. Во избежание искажений пропорций фигур надо согласовывать диапазон изменения переменной *x*. Обычно параметр scalling=constrained выравнивает масштабы и диапазоны по осям координат, что гарантирует отсутствие искажений у окружностей и других геометрических фигур. Однако при этом размеры графика нередко оказываются малыми. Напоминаем, что этот параметр можно задать и с помощью подменю Projection.

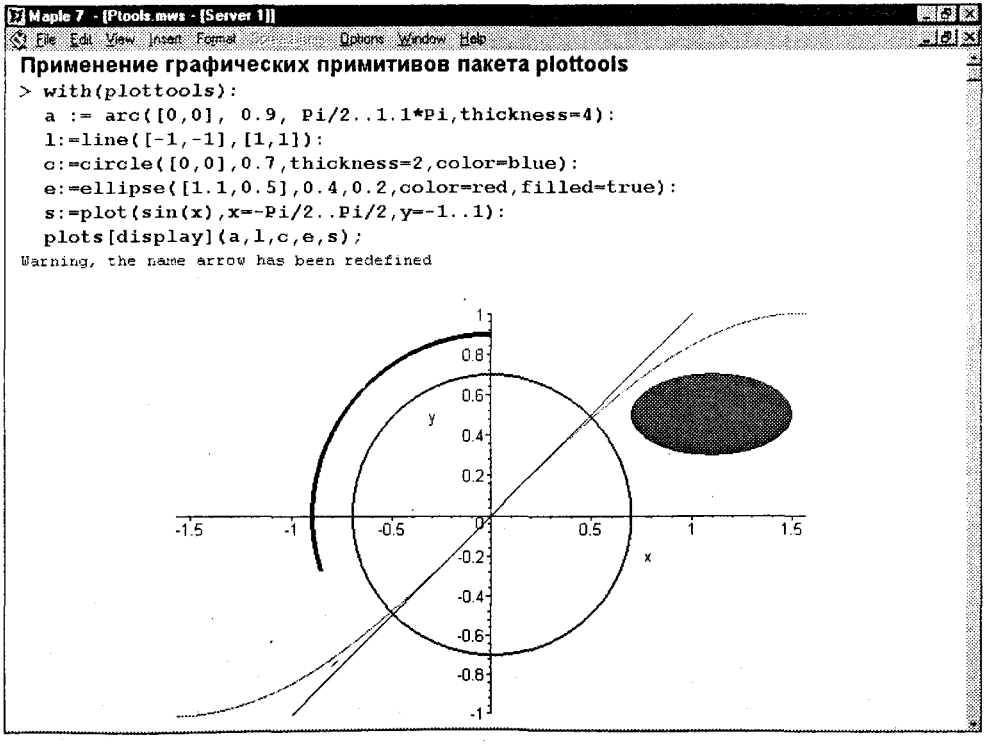


Рис. 12.19. Примеры применения примитивов двумерной графики пакета plottools

Рисунок 12.20 иллюстрирует построение средствами пакета plottools четырех разноцветных стрелок, направленных в разные стороны. Цвет стрелок задан списком цветов c, определенным после команды загрузки пакета. Для построения стрелок используется примитив arrow с соответствующими параметрами.

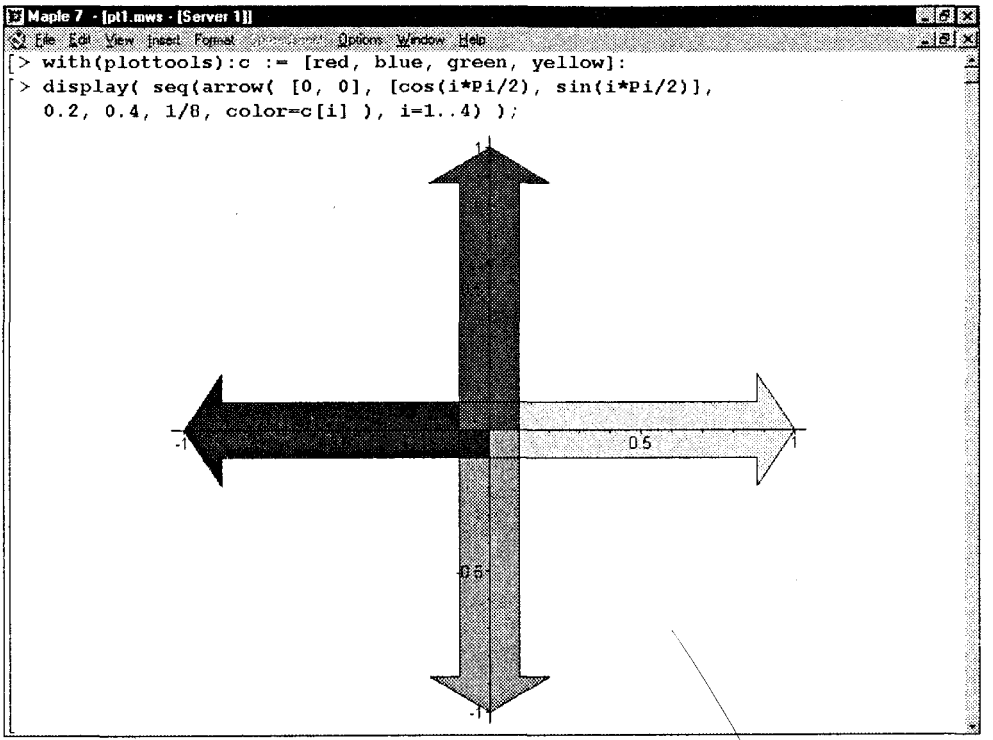


Рис. 12.20. Построение разноцветных стрелок, направленных в разные стороны

Примитивы могут использоваться в составе графических процедур, что позволяет конструировать практически любые типы сложных графических объектов. В качестве примера на рис. 12.21 представлена процедура SmithChart, которая строит хорошо известную электрикам диаграмму Смита (впрочем, несколько упрощенную). В этой процедуре используется примитив построения дуг arc. При этом задается верхняя часть диаграммы, а нижняя получается ее зеркальным отражением.

ПРИМЕЧАНИЕ

Обратите внимание на то, что, начиная с рис. 12.21, мы не указываем загрузку пакета plottools, поскольку она уже была проведена ранее. Однако надо помнить, что все примеры этого раздела предполагают, что такая загрузка обеспечена. Если вы использовали команду restart или только что загрузили систему Maple 7, то для исполнения примера рис. 12.21 и последующих примеров надо исполнить команду with(plottools).

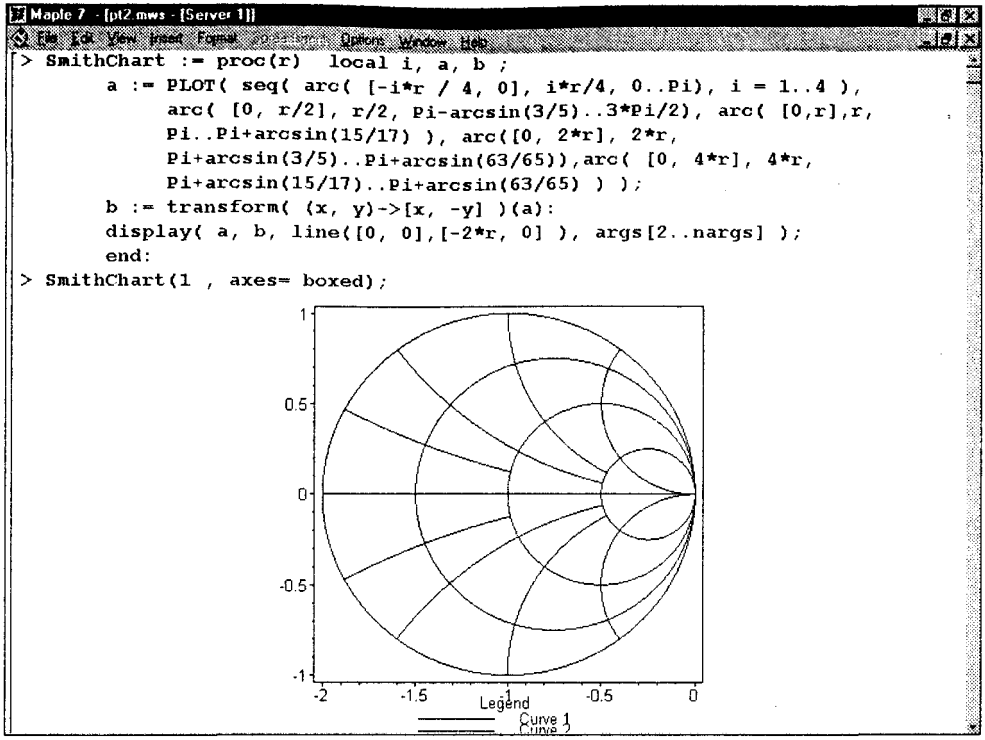


Рис. 12.21. Построение диаграммы Смита

Примеры применения трехмерных примитивов пакета plottools

Аналогичным описанному выше образом используются примитивы построения трехмерных фигур. Это открывает возможность создания разнообразных иллюстрационных рисунков и графиков, часто применяемых при изучении курса стереометрии. Могут строиться самые различные объемные фигуры и поверхности — конусы, цилиндры, кубы, полиэдры и т. д. Использование средств функциональной окраски делает изображения очень реалистичными.

Рисунок 12.22 показывает построение цилиндра и двух граненых шаров. Цилиндр строится примитивом `cylinder`, а граненые шары — примитивом `icosahedron`.

Другой пример (рис. 12.23) иллюстрирует построение на одном графике двух объемных фигур, одна из которых находится внутри другой фигуры. Этот пример демонстрирует достаточно корректное построение вложенных фигур.

На рис. 12.24 показано совместное построение двух пересекающихся кубов и сферы в пространстве. Нетрудно заметить, что графика пакета приблизительно (с точностью до сегмента) вычисляет области пересечения фигур. С помощью контекстно-зависимого меню правой кнопки мыши (оно показано на рис. 12.24) можно устанавливать условия обзора фигур, учитывать перспективу при построении и т. д. В частности, фигуры на рис. 12.24 показаны в перспективе.

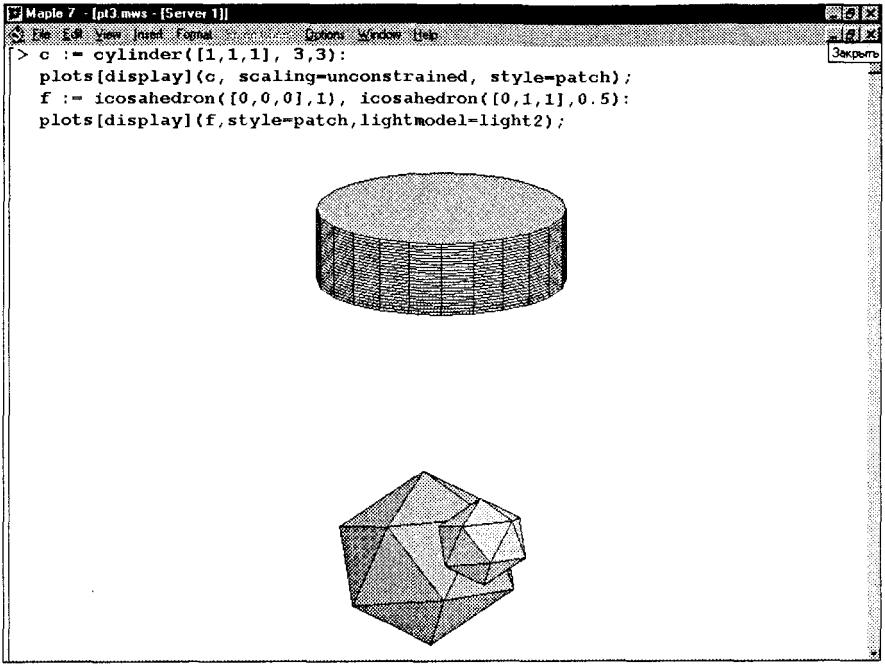


Рис. 12.22. Построение цилиндра и двух граненых шаров

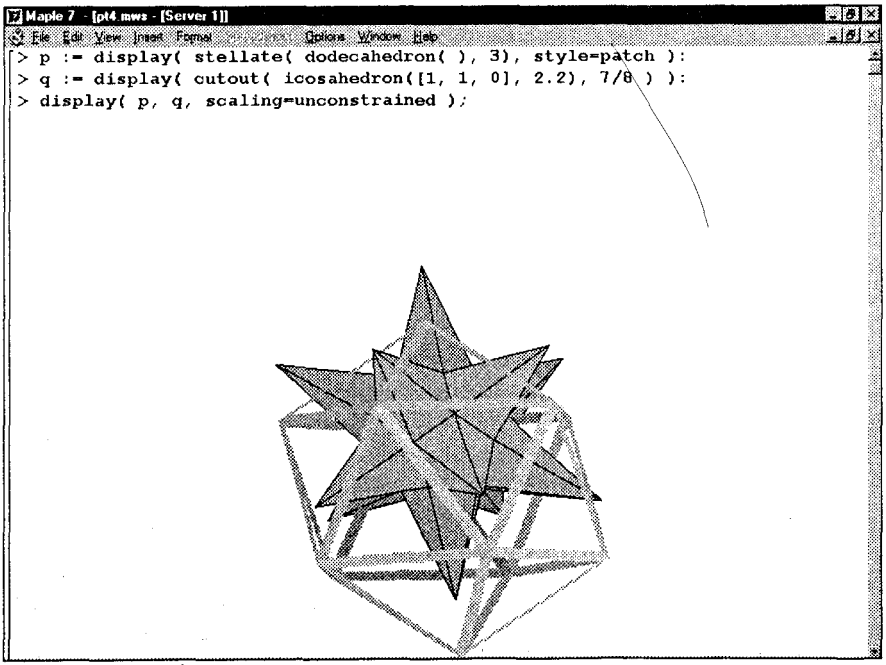


Рис. 12.23. Построение двух объемных фигур

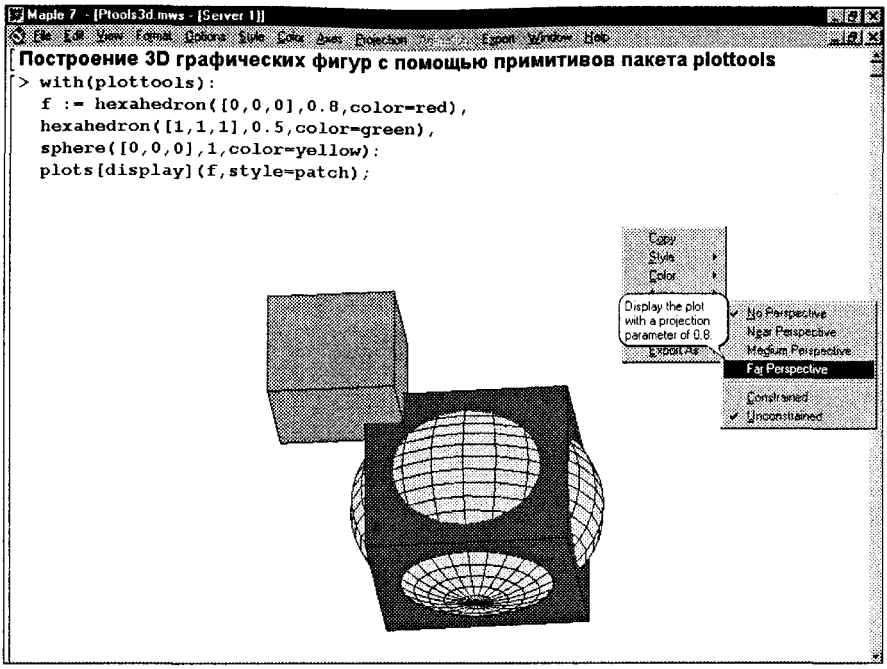


Рис. 12.24. Примеры применения примитивов трехмерной графики пакета plottools

Построение еще одной забавной трехмерной фигуры — «шкурки ежа» — демонстрирует рис. 12.25. В основе построения лежит техника создания полигонов.

Построение фигур, очень напоминающих улитки, показано на рис. 12.26. При построении этих фигур используется функция `tubeplot`. Обратите внимание на то, что строятся две входящие друг в друга «улитки».

Наконец, на рис. 12.27 показано построение фигуры — бутылки Клейна. Фигура задана рядом своих фрагментов, определенных в процедуре `cleinpoints`. Эта процедура является еще одним наглядным примером программирования графических построений с помощью Maple-языка.

С другими возможностями этого пакета читатель теперь справится самостоятельно или с помощью данных справочной системы. Много примеров построения сложных и красочных фигур с применением пакета plottools можно найти в Интернете на сайте фирмы Maple Software, в свободно распространяемой библиотеке пользователей системы Maple и в книгах по этой системе.

Построение графиков из множества фигур

В ряде случаев бывает необходимо строить графики, представляющие собой множество однотипных фигур. Для построения таких графиков полезно использовать функцию повторения `seq(f,i=a..b)`. На рис. 12.28 показано построение фигуры, образованной вращением прямоугольника вокруг одной из вершин..

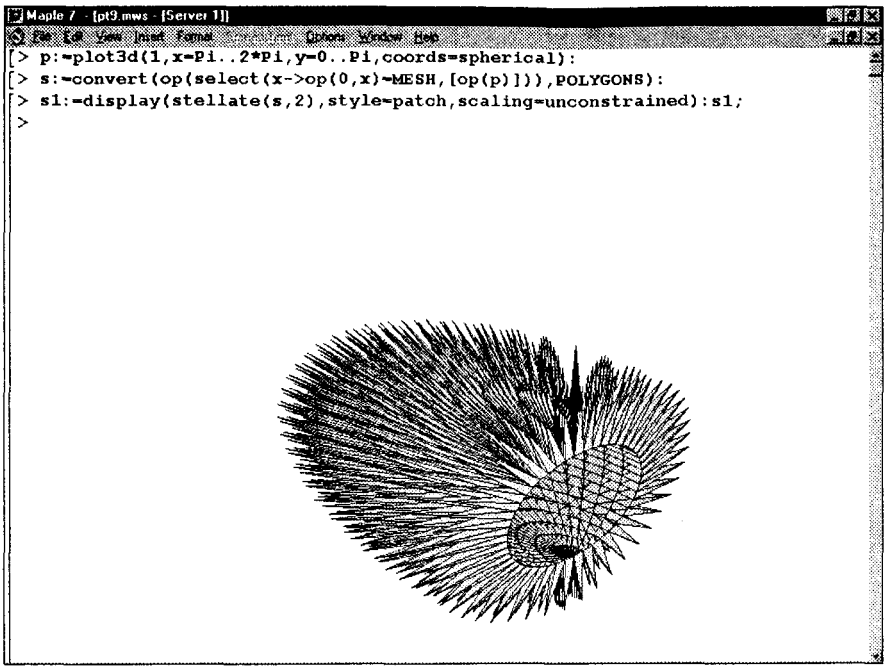


Рис. 12.25. Построение трехмерной фигуры — «шкурки ежа»

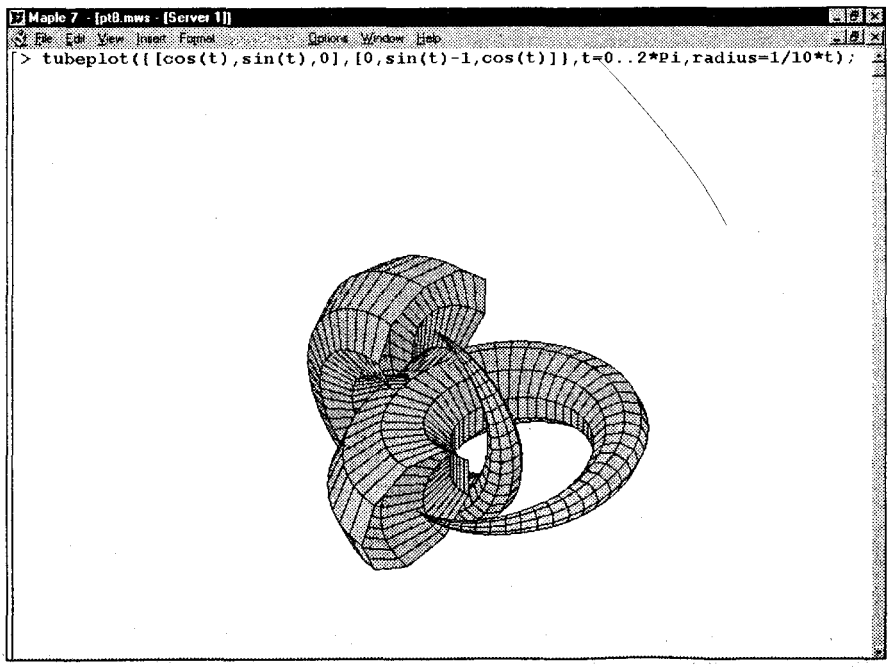


Рис. 12.26. Построение фигуры «улитка»

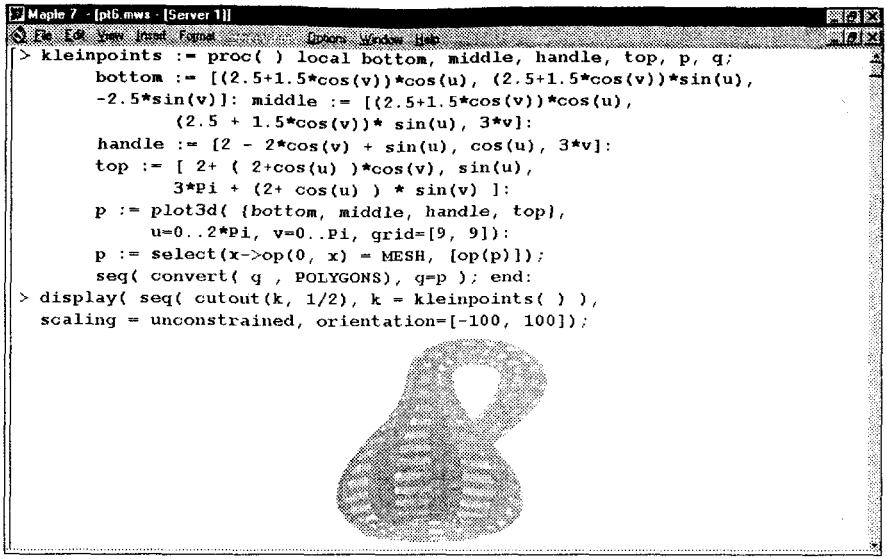


Рис. 12.27. Построение фигуры «бутылка Клейна»

В этом примере полезно обратить внимание еще и на функцию поворота фигуры — rotate. Именно сочетание этих двух функций (мультиплицирования и поворота базовой фигуры — прямоугольника) позволяет получить сложную фигуру, показанную на рис. 12.28.

Анимация двумерной графики в пакете plottools

Пакет plottools открывает возможности реализации анимационной графики. Мы ограничимся одним примером анимации двумерных графиков. Этот пример представлен на рис. 12.29. В этом примере показана анимационная иллюстрация решения дифференциального уравнения, описывающего незатухающий колебательный процесс. Строится качающийся объект — стрелка с острием вправо, решение дифференциального уравнения в виде синусоиды и большая стрелка с острием влево, которая соединяет текущую точку графика синусоиды с острием стрелки колеблющегося объекта.

Этот пример наглядно показывает возможности применения анимации для визуализации достаточно сложных физических и математических закономерностей. Перспективы применения системы Maple 7 в создании виртуальных физических и иных лабораторий трудно переоценить.

Анимация трехмерной графики в пакете plottools

Хорошим примером 3D-анимации является документ, показанный на рис. 12.30. Представленная на нем процедура springPlot имитирует поведение упругой системы, первоначально сжатой, а затем выстреливающей шар, установленный на ее верхней пластине. Упругая система состоит из неподвижного основания, на котором расположена упругая масса (например, из пористой резины), и верхней пластины.

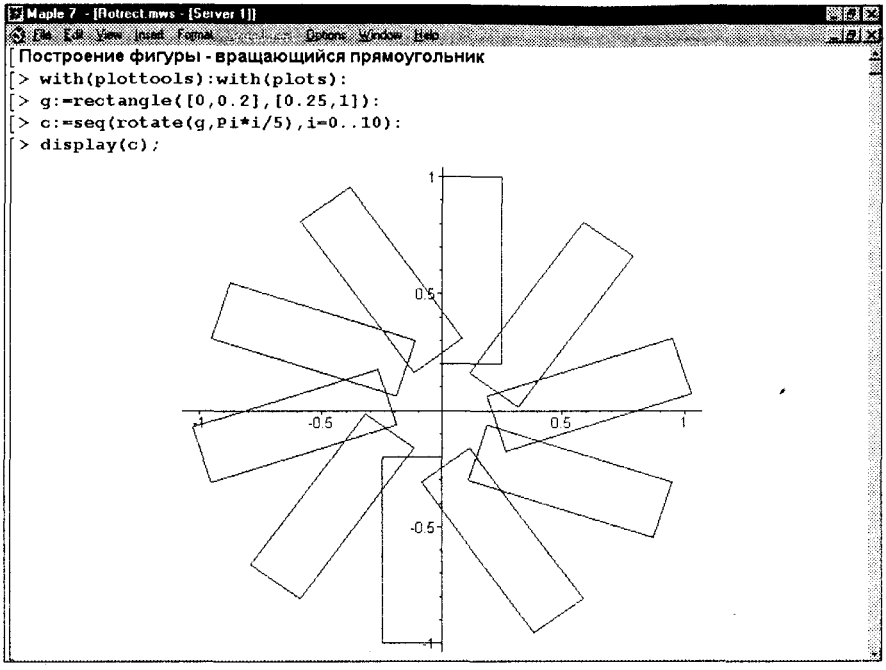


Рис. 12.28. Построение фигуры, образованной вращением прямоугольника

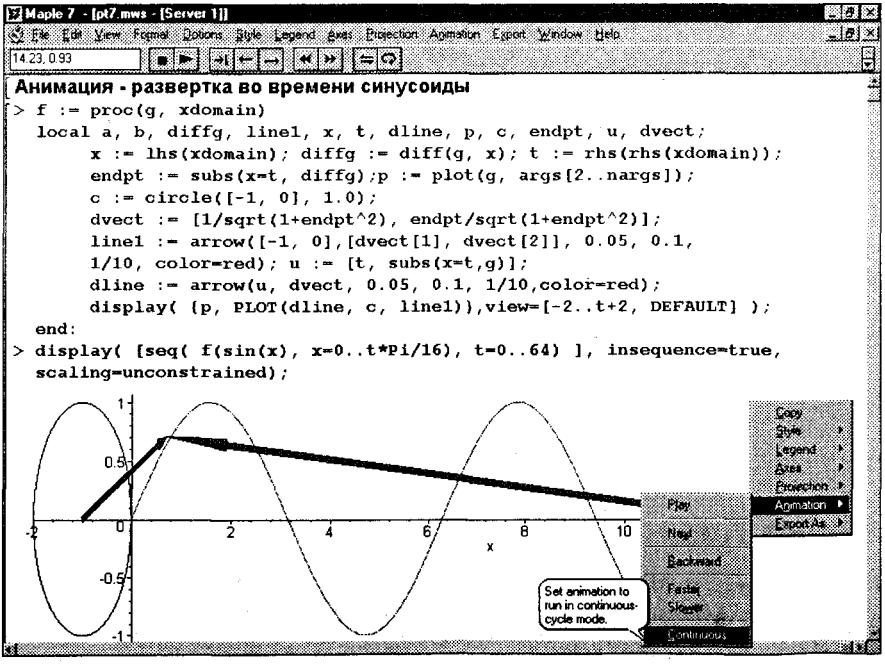


Рис. 12.29. Пример анимации двумерной графики

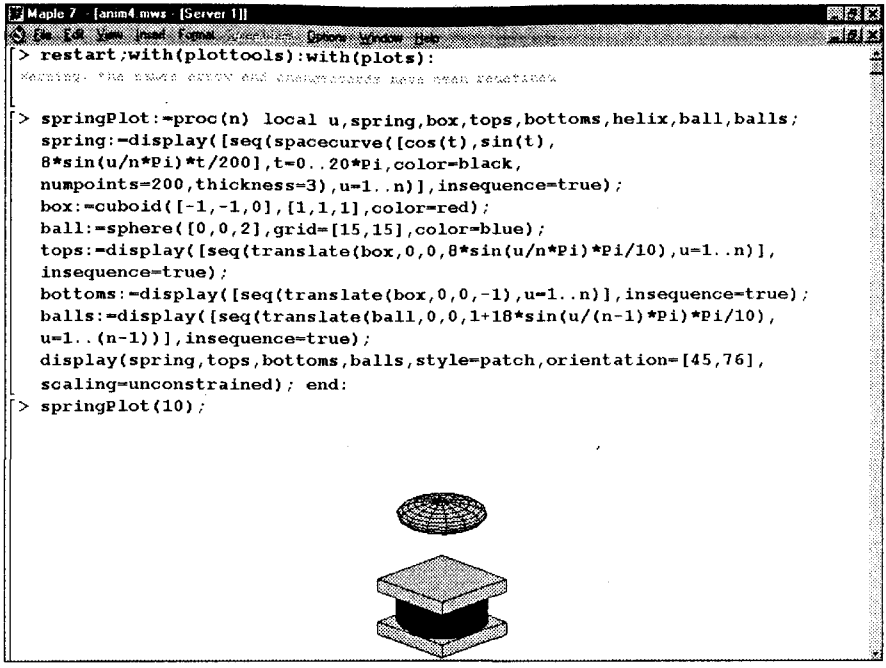


Рис. 12.30. Имитация отстрела шара сжатой упругой системой

Управление анимацией, реализованной средствами пакета plottools, подобно уже описанному ранее. Последний пример также прекрасно иллюстрирует возможности применения Maple 7 при математическом моделировании различных явлений, устройств и систем.

Расширенные средства графической визуализации

Построение ряда графиков, расположенных по горизонтали

Обычно если в строке ввода задается построение нескольких графиков, то в строке вывода все они располагаются по вертикали. Это не всегда удобно, например, при снятии копий экрана с рядом графиков, поскольку экран монитора вытянут по горизонтали, а не по вертикали. Однако при применении функций `plots` и `display` можно разместить ряд двумерных графиков в строке вывода по горизонтали. Это демонстрирует пример, показанный на рис. 12.31.

Этот пример достаточно прост и нагляден, так что читатель может пользоваться данной возможностью всегда, когда ему это нужно.

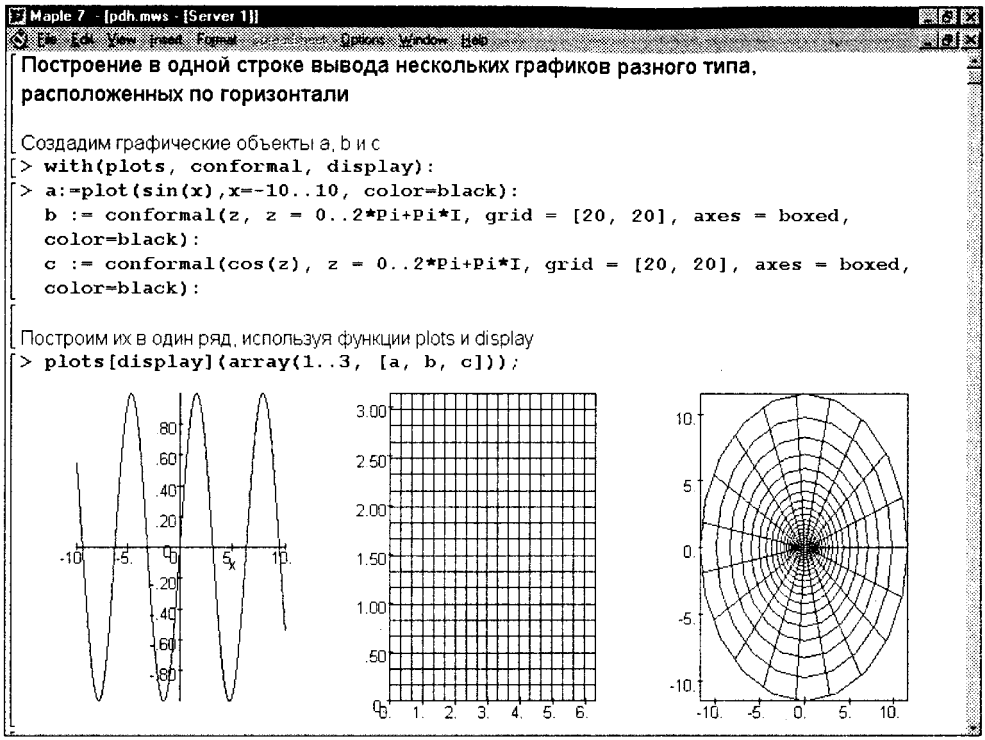


Рис. 12.31. Пример расположения трех графиков в строке вывода по горизонтали

Визуализация решения систем линейных уравнений

Мы уже не раз использовали графические возможности Maple для визуализации решений математических задач. Так, многие особенности даже функций одной переменной вида $f(x)$ могут быть выявлены с помощью графика этой функции. Затем можно точно вычислить корни функции (точки перехода через 0), экстремумы, крутизну наклона (производную) в заданных точках и т. д. Еще более информативна в этом отношении трехмерная графика — для большинства функций двух переменных вида $z(x, y)$ нужно очень богатое математическое воображение, чтобы представить их вид — особенно в одной из многих десятков координатных систем.

Однако некоторые виды графиков трудно представить себе даже при наличии такого воображения. В этом отношении Maple 7 предоставляет поистине уникальные возможности, обеспечивая простую и быструю визуализацию решений. Ниже мы рассмотрим несколько наиболее характерных примеров такой визуализации.

Системы линейных уравнений могут решаться как с помощью функции `solve`, так и с помощью матричных методов. Замечательной возможностью функции `solve` является возможность решения относительно ограниченного числа переменных. Например, систему линейных уравнений с переменными x, y, z, t и v можно решить относительно только первых трех переменных x, y и z. При этом

решения будут функциями относительно переменных t и v и можно будет построить наглядный график решения (рис. 12.32).

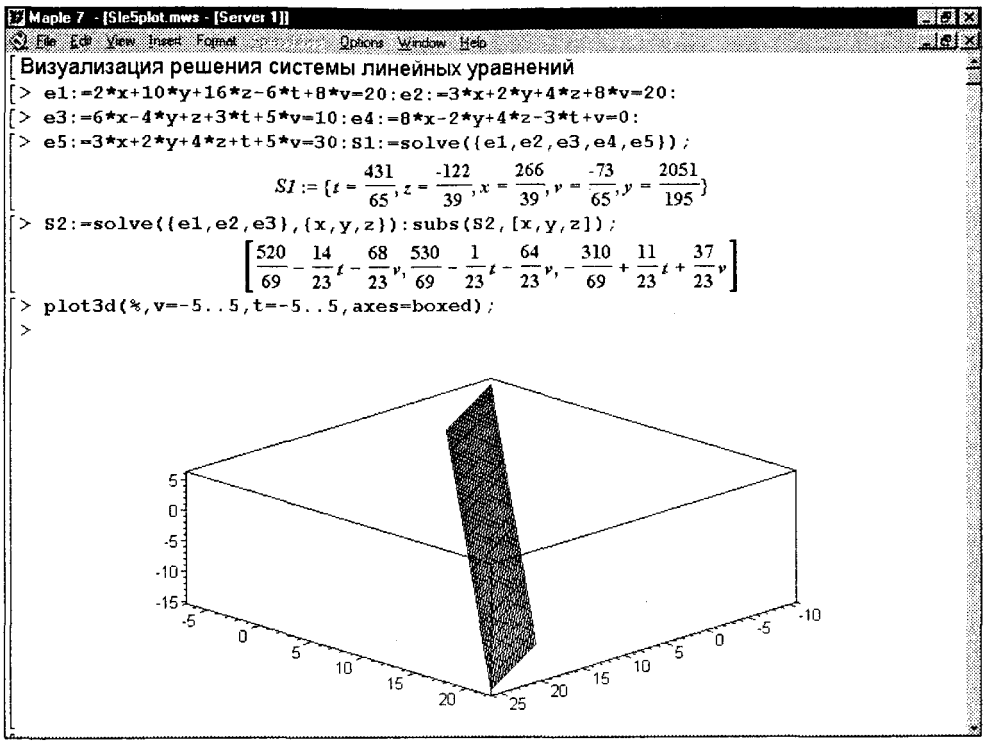


Рис. 12.32. График, представляющий решения системы линейных уравнений

На рис. 12.32 система задана пятью равенствами: e1, e2, e3, e4 и e5. Затем функцией solve получено вначале решение для всех переменных (для иллюстрации), а затем для трех переменных x, y и z. Для получения решения в виде списка, а не множества, как в первом случае для всех переменных, использована функция подстановки subs. После этого функция plot3d строит плоскость решения в пространстве.

Визуализация решения систем неравенств

Пожалуй, еще более полезным и наглядным средством является визуализация решения системы уравнений в виде неравенств. В пакете plots имеется специальная графическая функция inequal, которая строит все граничные линии неравенств и позволяет раскрасить разделенные ими области различными цветами:

```
inequal(ineqs, xspec, yspec, options)
```

Параметры этой функции следующие: ineqs — одно или несколько неравенств или равенств или список неравенств или равенств; xspec — xvar=min_x..max_x;

yspec — yvar=min_y..max_y; o — необязательные параметры, например указывающие цвета линий, представляющих неравенства или равенства, и областей, образованных этими линиями и границами графика.

Пример применения этой функции представлен на рис. 12.33.

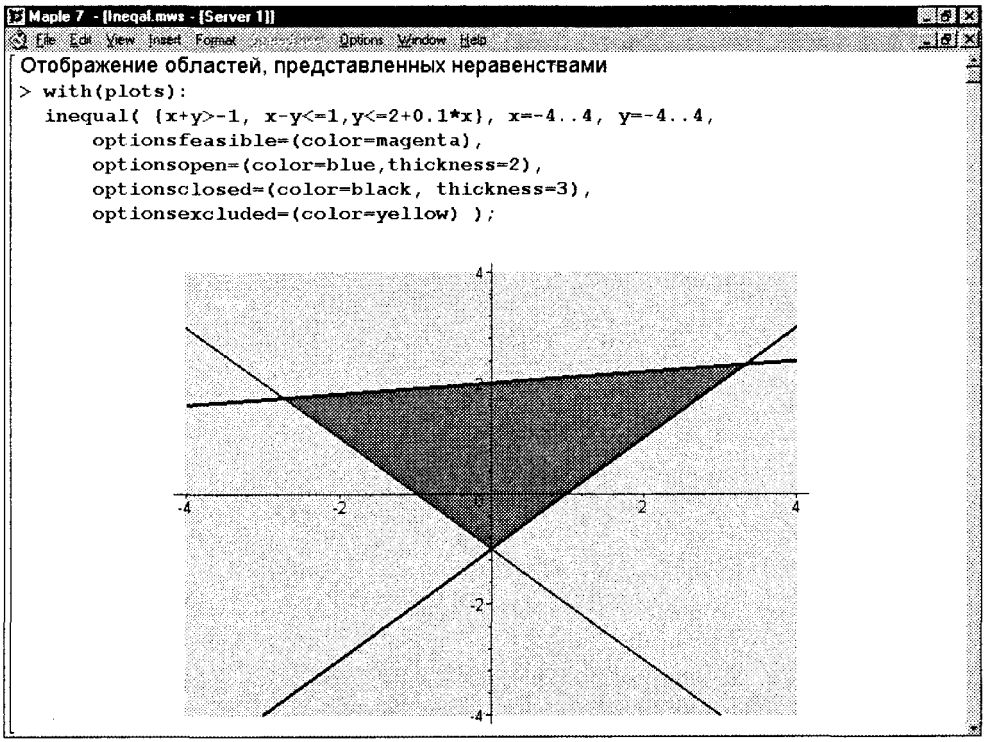


Рис. 12.33. Пример графической интерпретации решения системы неравенств

Обратите внимание на задание цветов: optionsfeasible задает цвет внутренней области, для которой удовлетворяются все неравенства (равенства), optionsopen и optionsclosed задают цвета открытых и закрытых границ областей графика, optionsexcluded используется для цвета внешних областей. График дает весьма наглядную интерпретацию действия ряда неравенств (или равенств).

Конформные отображения на комплексной плоскости

Объем данной книги не позволяет объяснить столь тонкое понятие, как конформные отображения на комплексной плоскости. Ограничимся лишь указанием на то, что в пакете plots имеется функция для таких отображений:

```
conformal(F,r1,r2,o);
```

где F — комплексная процедура или выражение; r1, r2 — области, задаваемые в виде a..b или name=a..b; o — управляющие параметры. Таким образом, для построения нужного графика достаточно задать нужное выражение и области изменения r1 и r2. Пример построения конформных изображений для трех выражений дан на рис. 12.34.

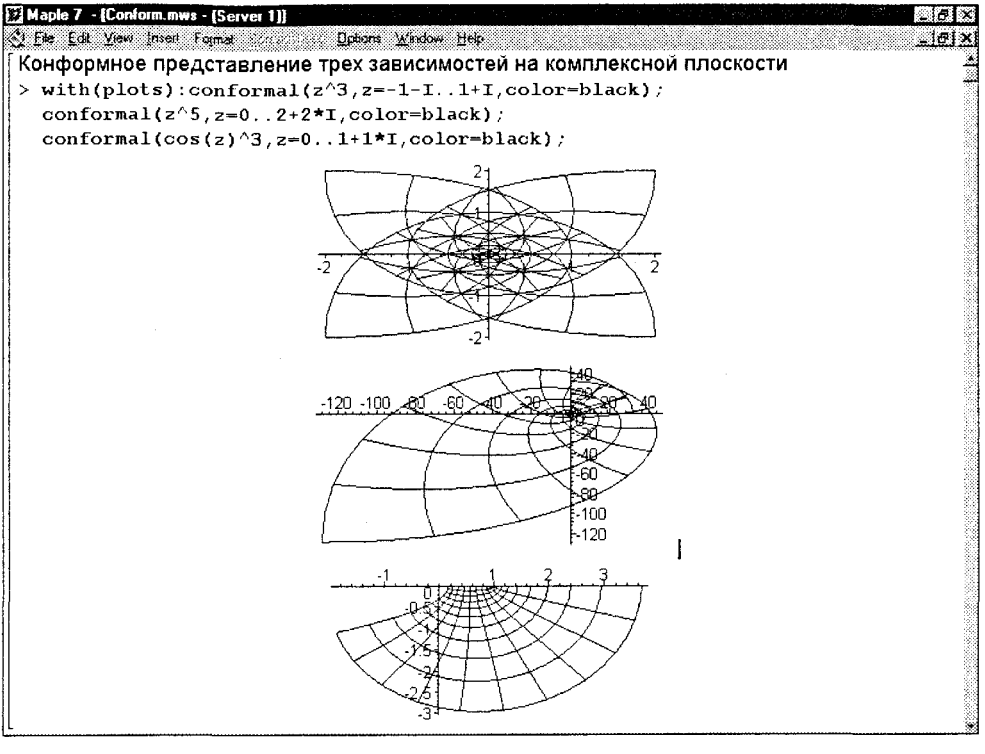


Рис. 12.34. Конформное отображение на комплексной плоскости графиков трех зависимостей

Средства конформного отображения в Maple 7, к сожалению, остаются рудиментарными и вряд ли достаточными для специалистов в этой области математики.

Графическое представление содержимого матрицы

Многие вычисления имеют результаты, представляемые в форме матриц. Иногда такие результаты можно наглядно представить графически, например в виде гистограммы. Она представляет собой множество столбцов квадратного сечения, расположенных на плоскости, образованной осями строк (row), и столбцов (column) матрицы. При этом высота столбцов определяется содержимым ячеек матрицы.

Такое построение обеспечивает графическая функция matrixplot из пакета plots. На рис. 12.35 показано совместное применение этой функции с двумя

функциями пакета linalg, формирующими две довольно экзотические матрицы **A** и **B**.

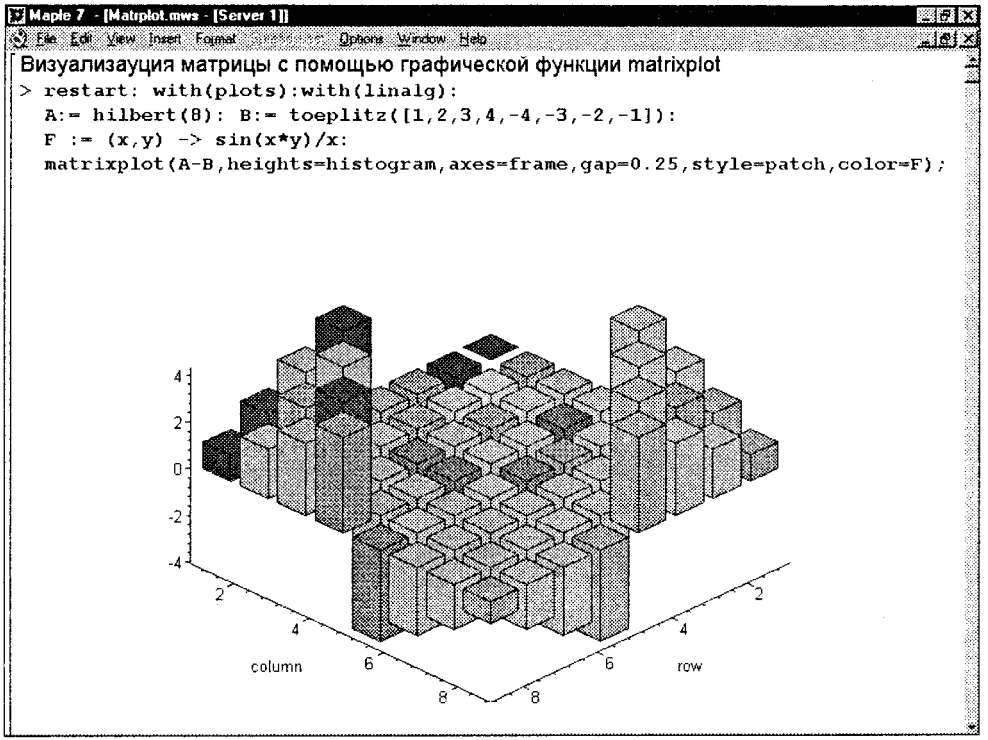


Рис. 12. 35. Графическое представление матрицы

На рис. 12.35 показана графическая визуализация матрицы, полученной как разность матриц **A** и **B**. Для усиления эффекта восприятия применяется функциональная закраска разными цветами. Для задания цвета введена процедура F.

Визуализация ньютоновских итераций в комплексной области

Теперь займемся довольно рискованным экспериментом — наблюдением ньютоновских итераций с их представлением на комплексной плоскости. На рис. 12.36 задана функция $f(z)$ комплексного аргумента. Проследить за поведением этой функции на комплексной плоскости в ходе ньютоновских итераций позволяет графическая функция complexplot3d из пакета plots.

Наблюдаемая картина весьма необычна и свидетельствует о далеко не простом ходе итерационного процесса.

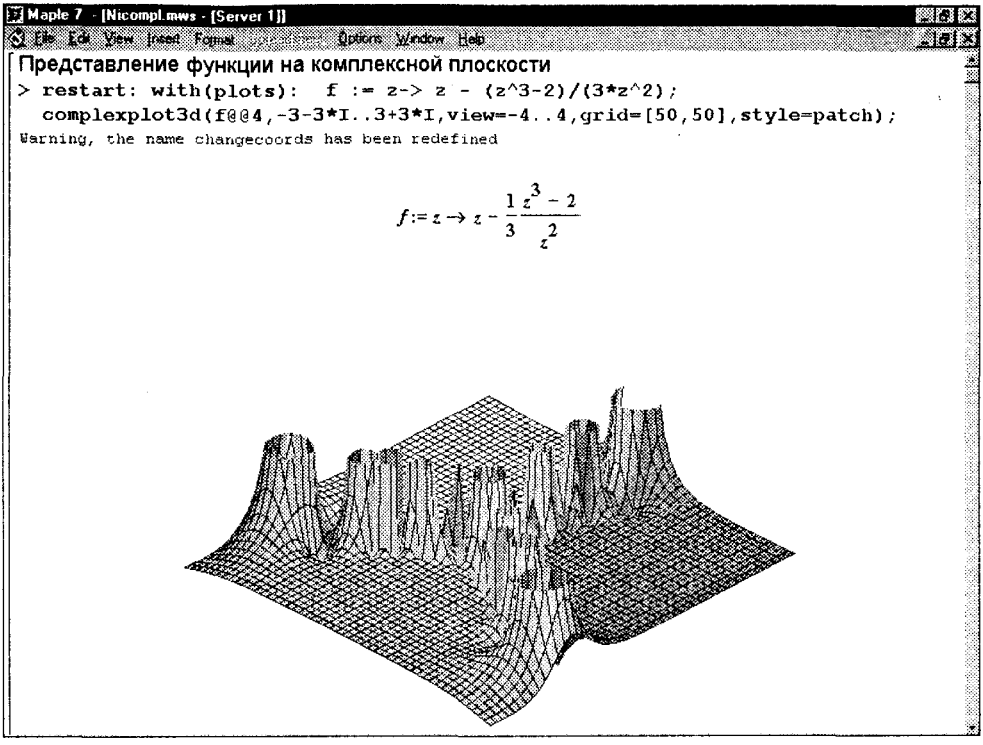


Рис. 12.36. Наблюдение за процессом ньютоновских итераций в трехмерном пространстве

ВНИМАНИЕ Риск работы с этим примером заключается в том, что в системе Maple 7 он иногда вызывает фатальные ошибки, ведущие к прекращению работы с системой. Обычно при запуске этого примера сразу после загрузки системы Maple такого не происходит, но, когда память загружена, сбой вполне возможен. Объективности ради надо заметить, что в системах Maple 6 и 7 подобное поведение системы не было замечено. Тем не менее рекомендуется записывать подобные примеры на диск перед их запуском.

Визуализация корней случайных полиномов

Наряду с традиционной для математических и статистических программ возможностью генерации случайных чисел Maple 7 предоставляет довольно экзотическую возможность генерации *случайных полиномов* с высокой максимальной степенью. Для этого используется функция:

```
randpoly(var,o)
```

Она возвращает случайный полином переменной `var`, причем максимальная степень полинома `nmax` может указываться параметром o вида `degree=nmax`.

Приведем примеры генерации случайного полинома с максимальной степенью 50:

```
> poly:=randpoly([x],degree=50);
```

$$poly := 56\,x^{44} + 49\,x^{36} + 63\,x^{34} + 57\,x^{33} - 59\,x^{8} + 45\,x^{6}$$

```
> poly:=randpoly([x],degree=50);
```

$$poly := 66\,x^{48} + 54\,x^{37} - 5\,x^{20} + 99\,x^{17} - 61\,x^{5} - 50\,x^{3}$$

```
>poly:=randpoly([x],degree=50);
```

$$poly := -91\,x^{49} - 47\,x^{32} - 61\,x^{27} + 41\,x^{12} - 58\,x^{9} - 90\,x^{8}$$

С помощью функции `allvalues` можно построить список `SA` корней случайного полинома. А с помощью команды вида:

```
> with(plots):complexplot(SA,x=-1.2..1.2,style=point);
```

построить комплексные корни полученного случайного полинома в виде точек на комплексной плоскости. Один из таких графиков (их можно построить множество) показан на рис. 12.37.

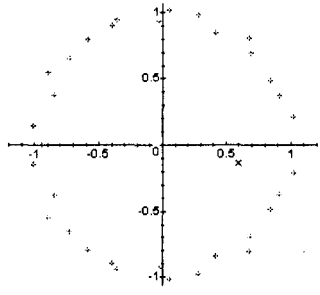


Рис. 12.37. Расположение корней случайного полинома на комплексной плоскости

Можно заметить любопытную закономерность — точки, представляющие корни случайного полинома, укладываются вблизи окружности единичного радиуса с центром в начале координат. Однако этот пример, приводимый в ряде книг по Maple, показывает, что порою вычисления могут давать довольно неожиданные результаты. Кстати говоря, аналитически можно вычислять корни полинома с максимальной степенью не более четырех.

Визуализация поверхностей со многими экстремумами

Maple 7 дает прекрасные возможности для визуализации поверхностей, имеющих множество пиков и впадин, другими словами, экстремумов. Рисунок 12.38 показывает задание «вулканической» поверхности с глубокой впадиной, окруженной пятью пиками. Здесь полезно обратить внимание на способ задания такой поверхности $f(a, b, c)$ как функции трех переменных a, b и c. Он обеспечивает индивидуальное задание координат каждого экстремума и его высоты (отрицательной для впадины).

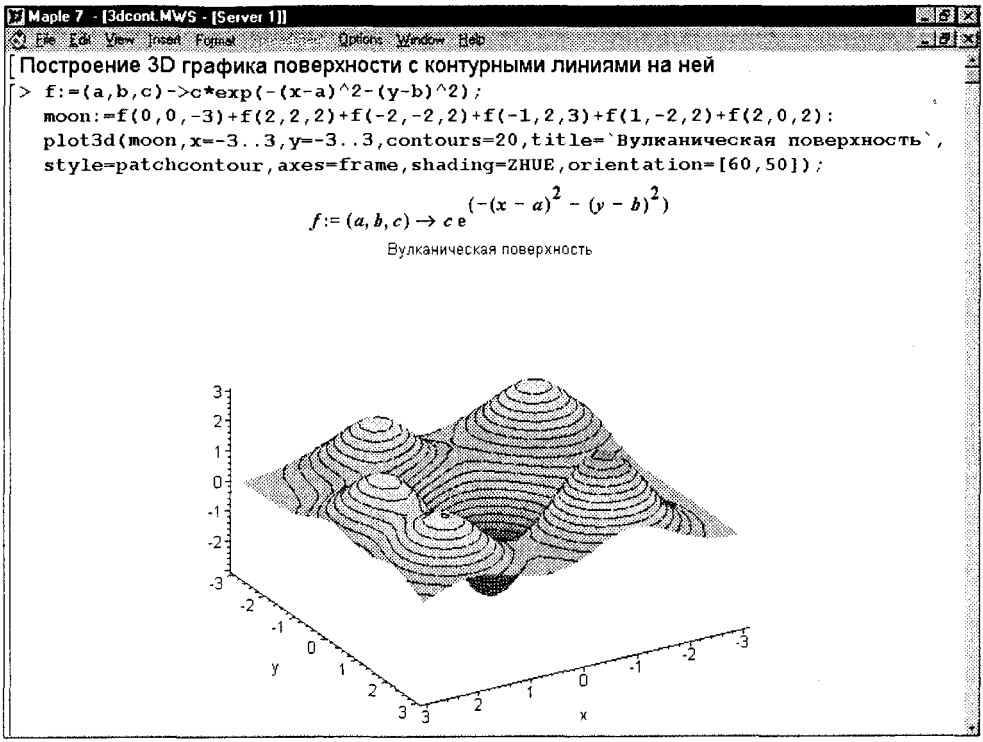


Рис. 12.38. Построение графика поверхности с множеством экстремумов

Наглядность этого графика усилена за счет применения функциональной окраски и контурных линий, нанесенных на саму поверхность. Все эти возможности обеспечивают параметры основной функции `plot3d`.

А на рис. 12.39 представлен еще один способ задания поверхности — с помощью функции двух угловых переменных $f(\theta, \varphi)$.

При построении этого рисунка также используются функциональная окраска и построение контурных линий.

Визуализация построения касательной и перпендикуляра

В ряде геометрических построений нужно строить касательную и перпендикуляр к кривой, отображающей произвольную функцию $f(x)$ в заданной точке $x = a$. Рисунок 12.40 поясняет, как это можно сделать. Линии касательной $T(x)$ и перпендикуляра $N(x)$ определены аналитически через производную в заданной точке.

Во избежание геометрических искажений положения касательной и перпендикуляра при построении графика функцией `plot` надо использовать параметр `scaling=constrained`.

Maple 7 - [3dl2v.mws - [Server 1]]

File Edit View Insert Format Options Window Help

3D график поверхности, заданной функцией двух переменных

```
> f:=(theta,phi)->  (8*sin(phi)-8*sin(theta)-6*sin(theta-phi))/
  (15+6*cos(theta)+6*cos(phi))^2;
```

$$f := (\theta, \phi) \rightarrow \frac{8\sin(\phi) - 8\sin(\theta) - 6\sin(\theta - \phi)}{(15 + 6\cos(\theta) + 6\cos(\phi))^2}$$

```
> plot3d(f(theta,phi),theta=-Pi..5*Pi/2,phi=-Pi/2..3*Pi,grid=[35,35],
  contours=20,title=`График поверхности, заданной функцией двух переменных
  f(theta,phi)`,style=patchcontour, axes=frame, shading=ZHUE,
  orientation=[115,55]);
```

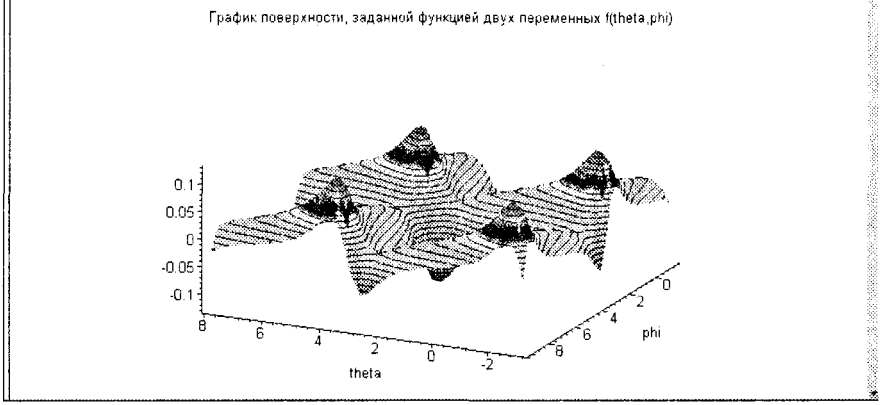


Рис. 12.39. Построение графика поверхности, заданной функцией двух угловых переменных

Maple 7 - [tn_curve.mws - [Server 1]]

File Edit View Insert Format Options Window Help

Построение касательной и перпендикуляра к функции f(x) в точке x=a

```
> f :=x -> 2+ sin(x^2)/(x-1):a :=2.1:
> T := x -> f(a) + (x-a)*D(f)(a);
```

$$T := x \rightarrow f(a) + (x - a)\,D(f)(a)$$

```
> N := x -> f(a) - (x-a)/D(f)(a);
```

$$N := x \rightarrow f(a) - \frac{x - a}{D(f)(a)}$$

```
> plot( {f(x), T(x), N(x)}, x=-5..5, y=-2..5, discont = true,
  scaling=constrained,color=black);
```

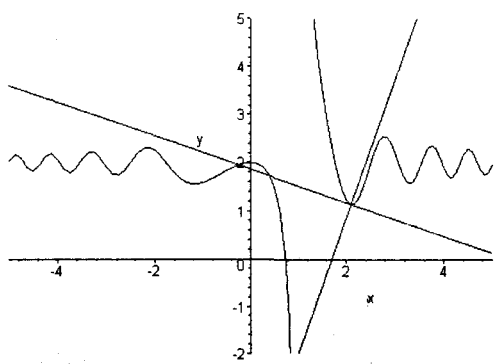


Рис. 12.40. Построение касательной и перпендикуляра к заданной точке графика функции *f(x)*

Визуализация вычисления определенных интегралов

Часто возникает необходимость в геометрическом представлении определенных интегралов в виде алгебраической суммы площадей, ограниченных кривой подынтегральной функции $f(x)$, осью абсцисс x и вертикалями $x = a$ и $x = b$ (пределами интегрирования). При этом желательно обеспечение закраски верхней и нижней (отрицательной и положительной) площадей разными цветами, например зеленым для верхней площади и красным для нижней. Как известно, численное значение определенного интеграла есть разность этих площадей.

К сожалению, в Maple 7 нет встроенной функции, явно дающей такое построение. Однако ее несложно создать. На рис. 12.41 представлена процедура a_plot, решающая эту задачу. Параметрами процедуры являются интегрируемая функция $f(x)$ (заданная как функция пользователя), пределы интегрирования a и b и пределы слева am и справа bm, задающие область построения графика $f(x)$.

Рисунок 12.41 дает прекрасное представление о сущности интегрирования для определенного интеграла. Приведенную на этом рисунке процедуру можно использовать для подготовки эффектных уроков по интегрированию разных функций.

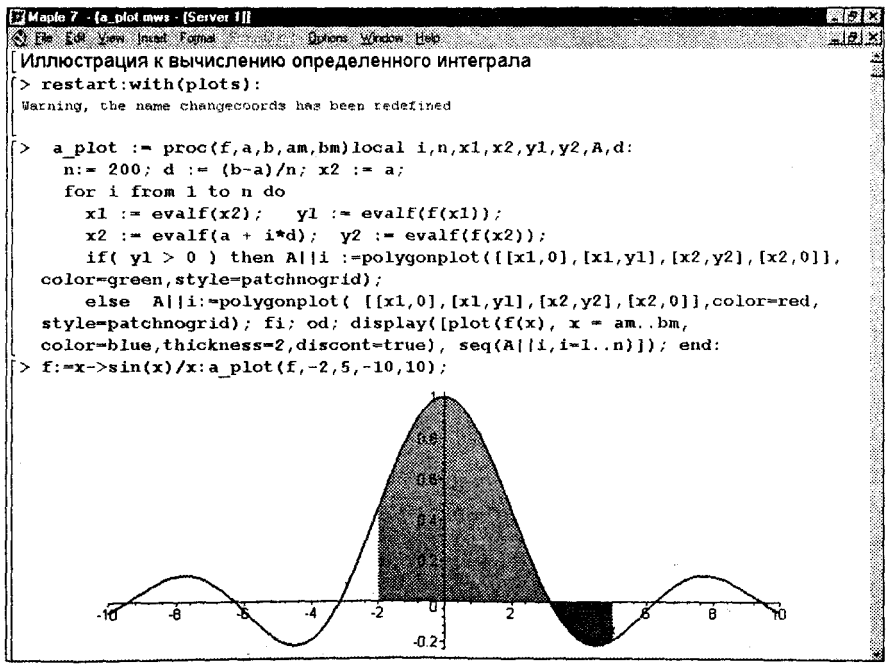


Рис. 12.41. Графическое представление определенного интеграла

Визуализация теоремы Пифагора

Еще один пример наглядного геометрического представления математических понятий — визуализация известной теоремы Пифагора (рис. 12.42).

В этом примере используется функция построения многоугольников. Наглядность построений усиливается выбором разной цветовой окраски треугольников и квадрата.

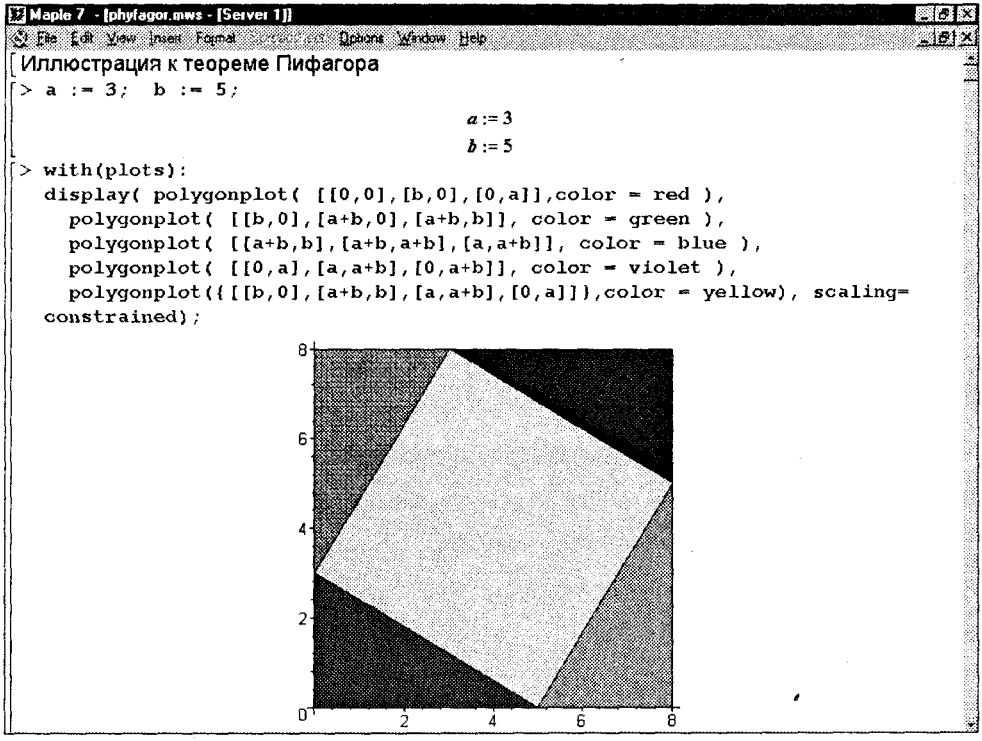


Рис. 12.42. Графическая иллюстрация к теореме Пифагора

Визуализация дифференциальных параметров кривых

Дифференциальные параметры функции $f(x)$, описывающей некоторую кривую, имеют большое значение для анализа ее особых точек и областей существования. Так, точки с нулевой первой производной задают области, где кривая нарастает (первая производная положительна) или убывает (первая производная отрицательна) с ростом аргумента x. Нули второй производной задают точки перегиба кривой.

Следующая графическая процедура служит для визуализации поведения кривой $f_1 = f(x)$ на отрезке изменения x от a до b:

```
> with(plots):
shape_plot := proc(f1,a,b)
local  i,n,x1,x2,xmid,d,y1,y2,A,B, m,M,slope, concav:
n:= 200;   d := (b-a)/n;   x2 := a;
M := maximize( f(x), x = a..b); m := minimize(f(x),x = a..b);
for i from 1 to n do
                        x1 := evalf(x2);      y1 := evalf( f(x1));
                        x2 := evalf(a + i*d); y2 := evalf( f(x2));
                                    xmid := x1 + d/2;
                        slope := evalf( subs( x = xmid, diff( f1,x )));
                        concav := evalf( subs( x = xmid, diff( f1,x $
                                                2)));
                        if( slope > 0 )
                        then A[i]:=plot(f1,x=x1..x2,color=blue,thickness=4,axes=box);
                        else A[i]:=plot(f1,x=x1..x2,color=red,thickness=2,axes=box);
                        fi;
                        if( concav > 0 ) then
                                    B[i]:=polygonplot([[x1,M],[x1,y1],[x2,y2],[x2,M]],
                                          color=green,style=patchnogrid);
                        else
                                    B[i]:=polygonplot( [[x1,m],[x1,y1],[x2,y2],[x2,m]],
                                          color=coral,style=patchnogrid);
                        fi;
od;
display({ seq( A[i],i=1..n ),seq( B[i],i=1..n ) } );
end:
```

В этой процедуре заданы следующие цвета (их можно изменить):

Таблица 12.1. Цвета при визуализации в процедуре shape_plot

Изменение f(x)	**Цвет**
Возрастание	Синий
Убывание	Красный
Площадь	**Цвет**
Над минимумом	Зеленый
Под максимумом	Коралловый

Например, для функции:

```
> f:=x->sin(x)+x/2;shape_plot(f(x),-6,6);
```

$$f := x \to \sin(x) + \frac{1}{2}x$$

построенный график будет иметь вид, представленный на рис. 12.43 (естественно, в книге цвета — лишь оттенки серого).

Рисунок 12.43 дает наглядное представление о поведении заданной функции. Рекомендуется опробовать данную процедуру на других функциях. Следует отметить, что, поскольку процедура использует функции **mimimize** и **maximize**, она может давать сбои при исследовании сложных функций, содержащих специальные математические функции или особенности. Иногда можно избежать такой

ситуации, исключив особенность. Например, для анализа функции $\sin(x)/x$ можно записать ее в виде:

```
> f:=x->if x=0 then 1 else sin(x)/x end if;
shape_plot(f(x),-10,10);
```

Исполнение приведенной выше строки ввода дает график, представленный на рис. 12.44.

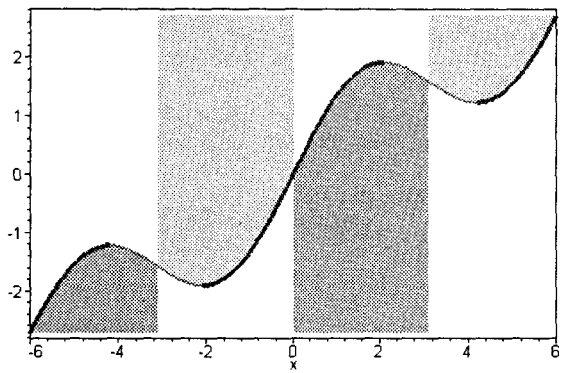


Рис. 12.43. Визуализация поведения функции $f(x)$

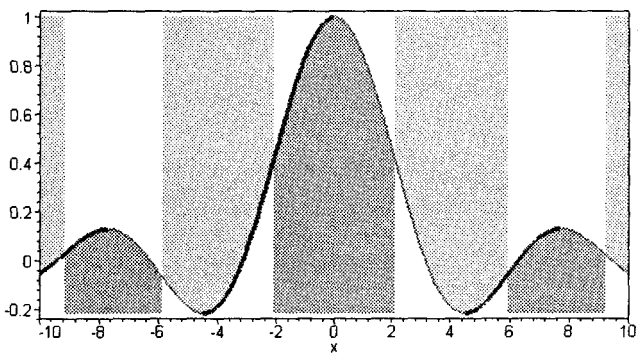


Рис. 12.44. Визуализация поведения функции $\sin(x)/x$

Данная процедура дает хорошие результаты при анализе функций, представленных полиномами. Вы можете сами убедиться в этом.

Иллюстрация итерационного решения уравнения f(x) = x

Классическим методом решения нелинейных уравнений является сведение их к виду $x = f(x)$ и применение метода простых итераций $x_k = s(x_{k-1})$ при заданном значении x_0. Приведем пример такого решения:

```
> f := x ->3*ln(x+1);
```
$f := x \to 3\ln(x+1)$
```
> x||0 := 0.5;
```
$x0 := .5$
```
> x0 := .5;
```
$x0 := .5$
```
> for k from 1 to 16 do x||k := evalf( f(x||(k-1) )); od;
```
$x1 := 1.216395324$

$x2 := 2.387646445$

$x3 := 3.660406248$

$x4 := 4.617307866$

$x5 := 5.177557566$

$x6 := 5.462768931$

$x7 := 5.598173559$

$x8 := 5.660378631$

$x9 := 5.688529002$

$x10 := 5.701181910$

$x11 := 5.706851745$

$x12 := 5.709388956$

$x13 := 5.710523646$

$x14 := 5.711030964$

$x15 := 5.711257755$

$x16 := 5.711359134$

Нетрудно заметить, что значения x_k в ходе итераций явно сходятся к некоторому значению. Проведем проверку решения, используя встроенную функцию `solve`:

```
> f(x) = x; solve(%, x);
```
$3\ln(x+1) = x$

$$0, -3\,\text{LambertW}\left(-1, -\frac{1}{3}e^{(-1/3)}\right) - 1$$

Результат выглядит необычно — помимо довольно очевидного корня $x = 0$ значение другого корня получено в виде специальной функции Ламберта. Впрочем, нетрудно найти и его численное значение:

```
> evalf(%);
```
0., 5.711441084

Однако как сделать процесс решения достаточно наглядным? Обычно для этого строят графики двух зависимостей — прямой x и кривой $f(x)$ — и наносят на них ступенчатую линию перемещения точки x_k. Специальной функции для графиков подобного рода Maple 7 не имеет. Однако можно составить специальную

процедуру для их построения. Ее листинг, заимствованный из примера, описанного в пакете обучения системе Maple 7 — PowerTools, представлен ниже:

```
> restart; with(plots):
                    Warning, the name changecoords has been redefined
> rec_plot := proc( f1, a , b, x0)
local x1, x2, y1, y2, i , p1, p2, p3, n, a1,a2;
x2 := x0;   y2:= 0;   n :=10;
for i from 1 to n do
x1 := x2;  y1 := y2;    y2:=f(x1);   x2 := y2;
p1[i] := plot([[x1, y1],[x1, y2]],x = a..b,color =black):
p2[i] := plot([[x1,y2],[x2, y2]],x=a..b,color=black):od:
display(   plot( f1(x), x = a..b, thickness = 2, color = blue),
plot(x, x = a..b, thickness = 2, color = black),
seq( p1[i], i = 1..n), seq( p2[i], i = 1..n) ); end:
```

Параметрами этой процедуры являются: f1 — функция $f(x)$; a и b — пределы изменения x при построении графика; x0 — значение x, с которого начинаются итерации. Исполнив команду:

```
> rec_plot( f(x), 0, 8, x0);
```

можно наблюдать график, иллюстрирующий итерационный процесс. Он представлен на рис. 12.45.

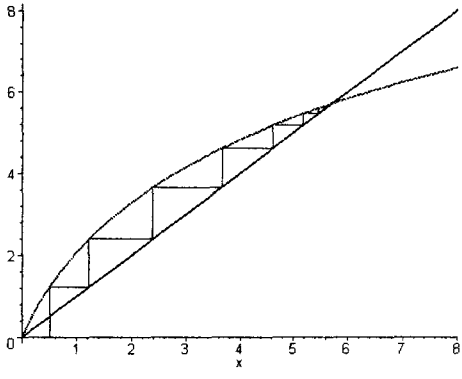


Рис. 12.45. Иллюстрация процесса итераций

Нетрудно заметить, что для данной функции процесс итераций хотя и не очень быстро, но уверенно сходится к точке пересечения прямой $y = x$ и кривой $y = f(x)$. Вы можете, меняя зависимость $f(x)$, провести исследования сходимости уравнений $x = f(x)$.

Построение сложных фигур в полярной системе координат

Некоторые виды математичеекой графики имеют определенную художественную ценность и фигурируют в символике различных стран и общественных

организаций. Остановимся на нескольких таких примерах применительно к графике в полярной системе координат. Представим фигуры, образованные множеством линий на плоскости.

Рисунок 12.46 демонстрирует одну из таких фигур. Это семейство из 10 кардиоид разного размера. Параметр `scalling=constrained` обеспечивает правильное отображение фигур — каждая кардиоида вписывается в огибающую ее невидимую окружность. Размер кардиоид задается значением параметра `a`.

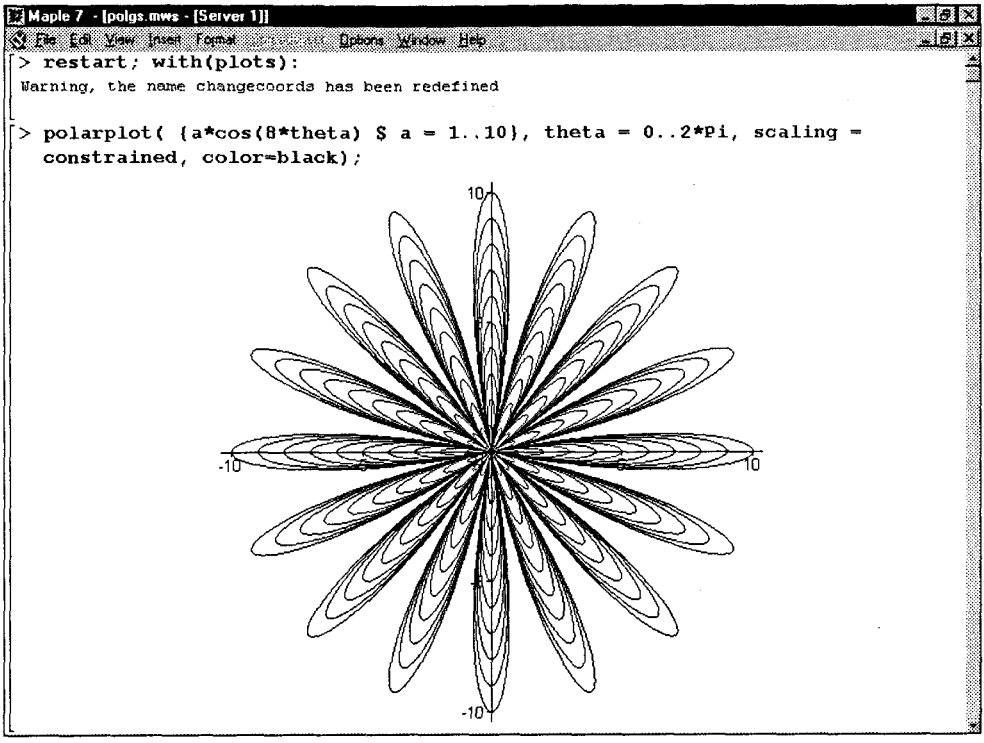


Рис.12.46. Семейство кардиоид на одном графике

Еще одно семейство кардиоид, на сей раз шестилепестковых, представлено на рис. 12.47. Здесь также изменяемым параметром каждой фигуры является ее размер, заданный параметром `a`.

Фигуре, представленной на рис. 12.48, трудно дать определенное название. Назовем ее волнообразной спиралью.

По образу и подобию приведенных фигур читатель может опробовать свои силы в создании новых красочных фигур в полярной системе координат. Некоторые из них поразительно напоминают снежинки, картинки в калейдоскопе и изображения морских звезд. Если убрать параметр `color=black`, введенный ради черно-белой печати картинок в книге, то можно усилить красочность фигур за счет их разноцветной окраски.

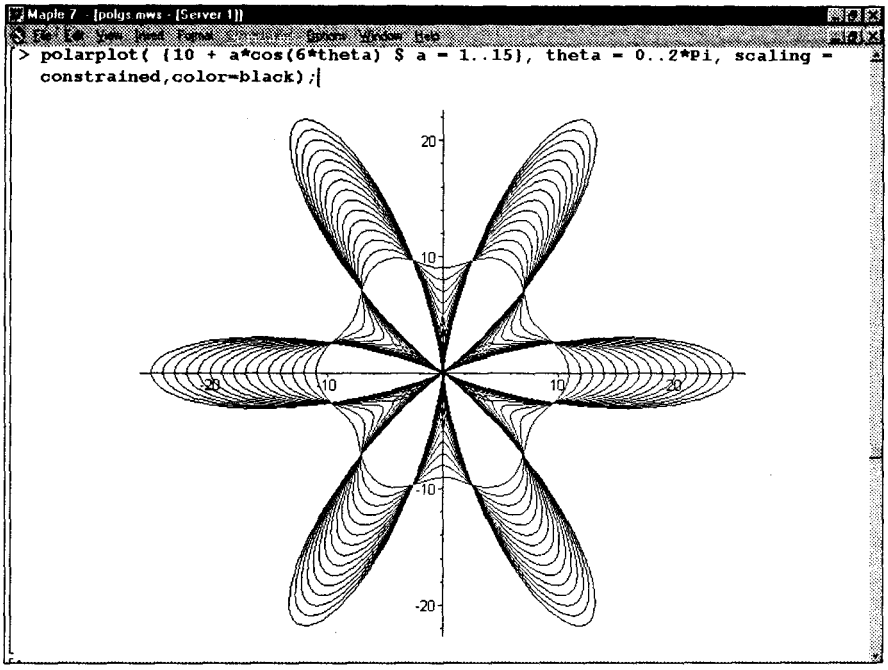


Рис. 12.47. Семейство шестилепестковых кардиоид

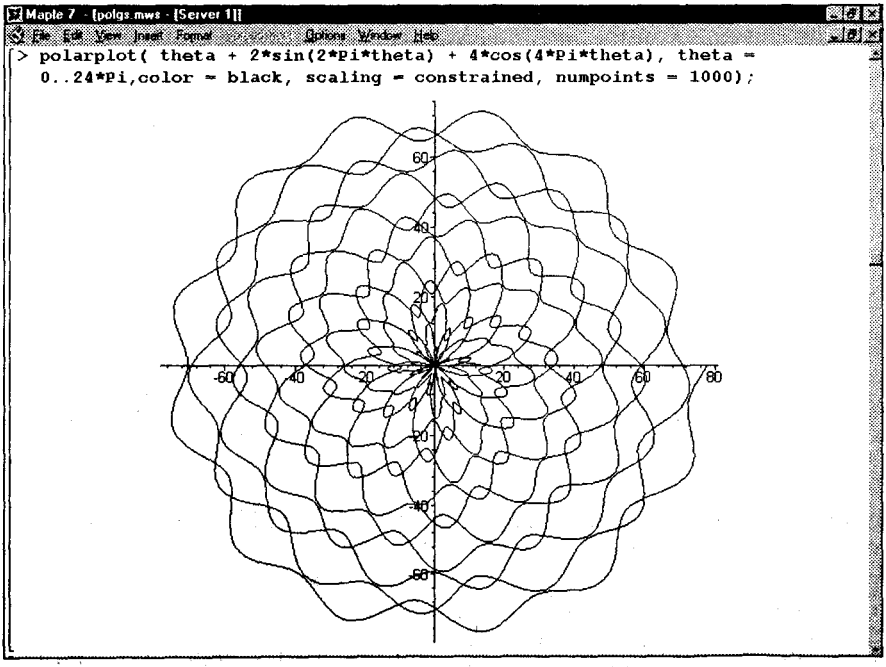


Рис. 12.48. Фигура — волнообразная спираль

Построение сложных фигур импликативной графики

Импликативные функции (см. урок 7) нередко имеют графики весьма любопытного вида. Ограничимся парой примеров построения таких графиков, представленных на рис. 12.49. Эти фигуры напоминают контурные графики функции двух переменных.

Приведенные примеры дают весьма наглядное представление о больших возможностях визуализации решений самых различных задач в системе Maple V. Можно значительно расширить их, эффектно используя описанные ранее приемы анимации изображений. В целом надо отметить, что графические возможности Maple 7 дают новый уровень качества графики современных математических систем, о котором с десяток лет тому назад можно было только мечтать.

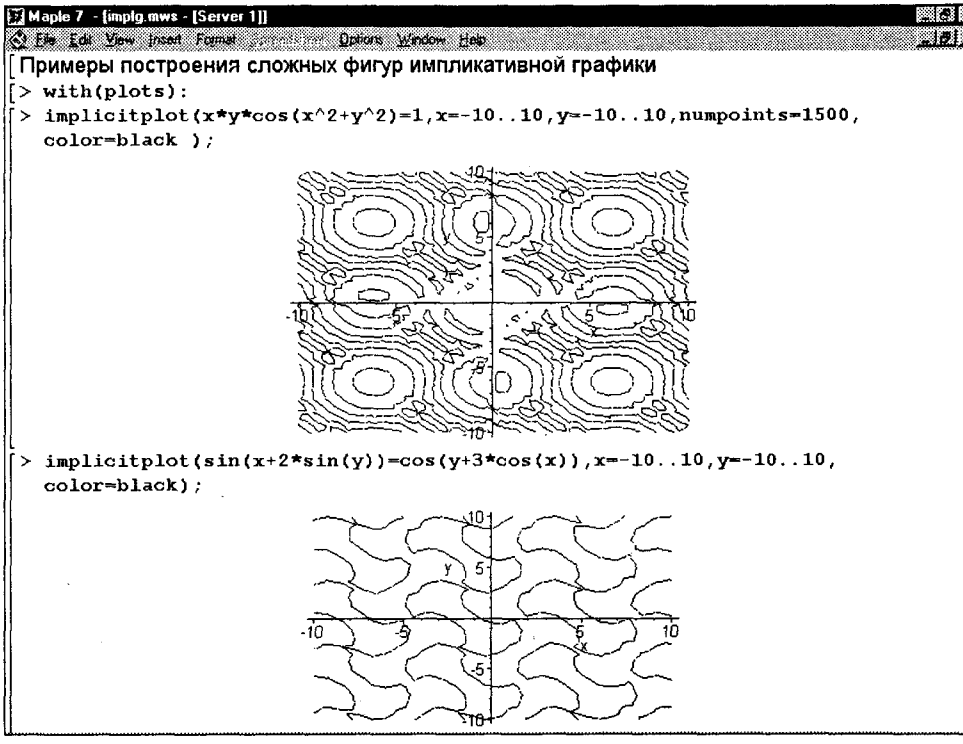


Рис. 12.49. Построение сложных фигур, заданных импликативными функциями

Расширенная техника анимации

Анимирование разложения импульса в ряд Фурье

Анимирование изображений является одним из самых мощных средств визуализации результатов моделирования тех или иных зависимостей или явлений.

Порою изменение во времени одного из параметров зависимости дает наглядное представление о его математической или физической сути.

Здесь мы расширим представление об анимации и рассмотрим не вполне обычный пример — наблюдение в динамике за гармоническим синтезом некоторой произвольной функции $f(x)$ на отрезке изменения x от 0 до 1. Значения функции $f(x)$ могут быть одного знака или разных знаков. В этом примере можно наблюдать в динамике синтез заданной функции рядом Фурье с ограниченным числом синусных членов (гармоник) — до 1, 2, 3...N. На рис. 12.50 представлен документ, реализующий такое разложение и затем синтез для пилообразного линейно нарастающего импульса, описываемого выражением $f(x) = -1 + 2 * x$. На графике строится исходная функция и результат ее синтеза в динамике анимации.

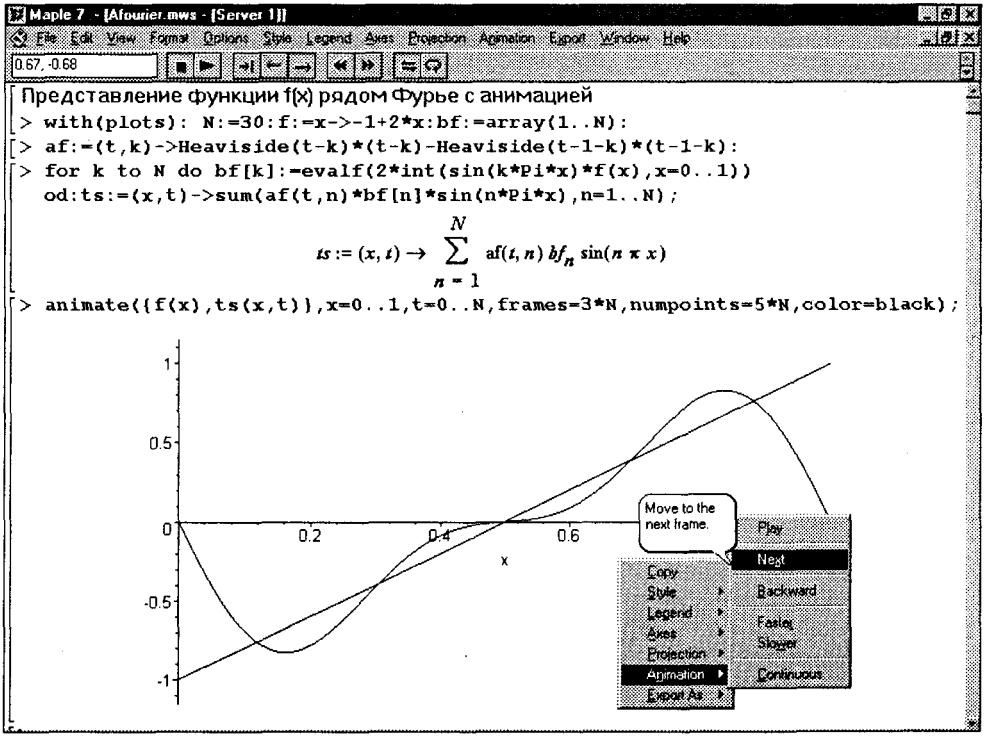


Рис. 12.50. Один из первых стоп-кадров анимации разложения импульса в ряд Фурье

Рисунок 12.51 показывает завершающий стоп-кадр анимации, когда число гармоник N равно 30. Нетрудно заметить, что такое число гармоник в целом неплохо описывает большую часть импульса, хотя в его начале и в конце все еще заметны сильные отклонения.

Для $f(x) = 1$ строится приближение для однополярного импульса с длительностью 1 и амплитудой 1, при $f(x) = x$ — приближение для пилообразного линейно

нарастающего импульса, при $f(x) = x\text{^}2$ — приближение для нарастающего по параболе импульса, при $f(x) = \text{signum}(x-1/2)$ — приближение для симметричного прямоугольного импульса-меандра и т. д. Фактически можно наблюдать анимационную картину изменения формы импульса по мере увеличения числа используемых для синтеза гармоник. Выбор используемого числа гармоник осуществляет амплитудный селектор — функция $af(t,k)$, основанная на применении функции Хевисайда.

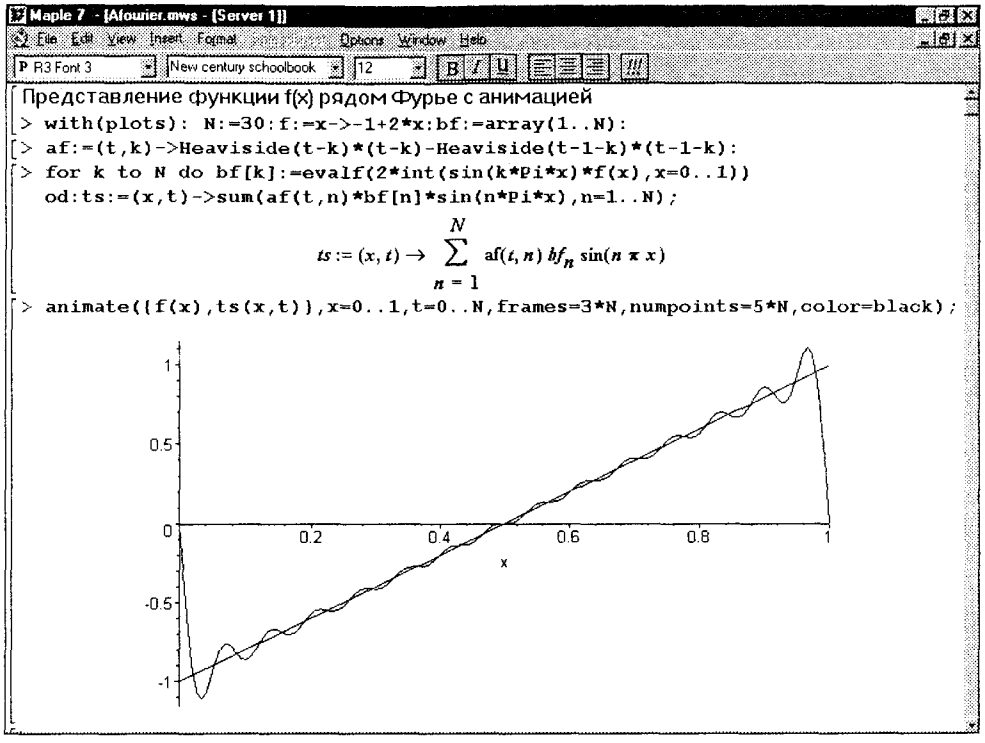


Рис. 12.51. Второй (завершающий) кадр анимации

Самым интересным в этом примере оказывается наблюдение за зарождением и эволюцией эффекта Гиббса — так называют волнообразные колебания на вершине импульса, связанные с ограничением числа гармоник при синтезе сигнала. С ростом числа гармоник эффект Гиббса не исчезает, просто обусловленные им выбросы вблизи разрывов импульса становятся более кратковременными. Амплитуда импульсов может достигать 18% от амплитуды перепадов сигнала, что сильно ухудшает приближение импульсных сигналов рядами Фурье и вынуждает математиков разрабатывать особые меры по уменьшению эффекта Гиббса.

Можно ли наблюдать одновременно все фазы анимации? Можно! Для этого достаточно оформить анимационную картину, созданную функцией `animate`, в виде

отдельного графического объекта, например g, после чего можно вывести все его фазы оператором display. Это и иллюстрирует рис. 12.52. На этот раз задано $f(x)$ = signum(x-1/2) и N = 25. Таким образом рассматриваются симметричные прямоугольные импульсы – меандр. У каждого рисунка координатные оси с делениями удалены параметром axes=none.

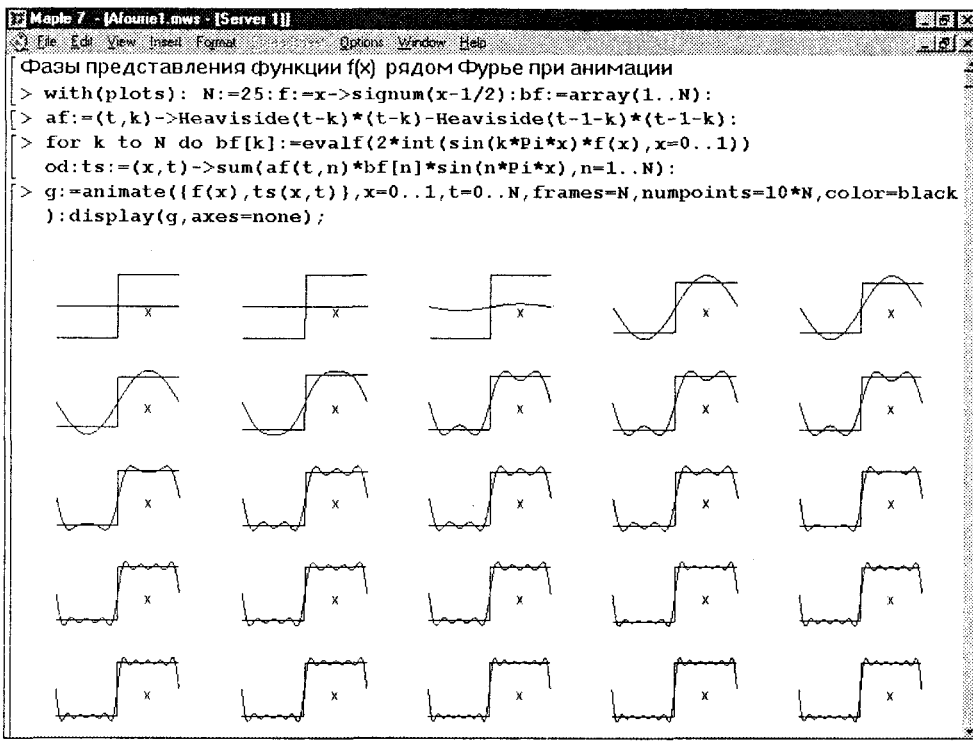


Рис. 12.52. Иллюстрация получения всех кадров анимации двумерного графика

Любопытно отметить, что при определенных числах гармоник связанная с колебательными процессами неравномерность вершины импульса резко уменьшается. Наблюдение этого явления и является наиболее интересным и поучительным при просмотре данного примера.

При внимательном просмотре рис. 12.52 заметно, что после некоторого периода установления фазы анимационной картинки практически повторяются. Это связано с известным обстоятельством — установившийся спектр меандра содержит только нечетные гармоники. Поэтому, к примеру, вид спектрального разложения при 22 гармониках будет тот же, что и при 21 гармонике, при 24 гармониках тот же, что при 23, и т. д. Однако эта закономерность проявляется только при установившемся (стационарном) спектре.

Наблюдение кадров анимации поверхности

Наблюдение за развитием поверхности производит на многих (особенно на студентов) большое впечатление. Оно позволяет понять детали создания сложных трехмерных графиков и наглядно представить их математическую сущность. Рассмотрим анимацию поверхности на примере рис. 12.18.

Как и для случая анимации двумерного графика, большой интерес представляет построение всех фаз анимации на одном рисунке. Делается это точно так же, как в двумерном случае. Это иллюстрирует рис. 12.53. На нем представлены 8 фаз анимации трехмерной поверхности $\cos(t*x*y/3)$, представленной функцией трех переменных t, x и y. При этом изменение первой переменной создает фазы анимации поверхности.

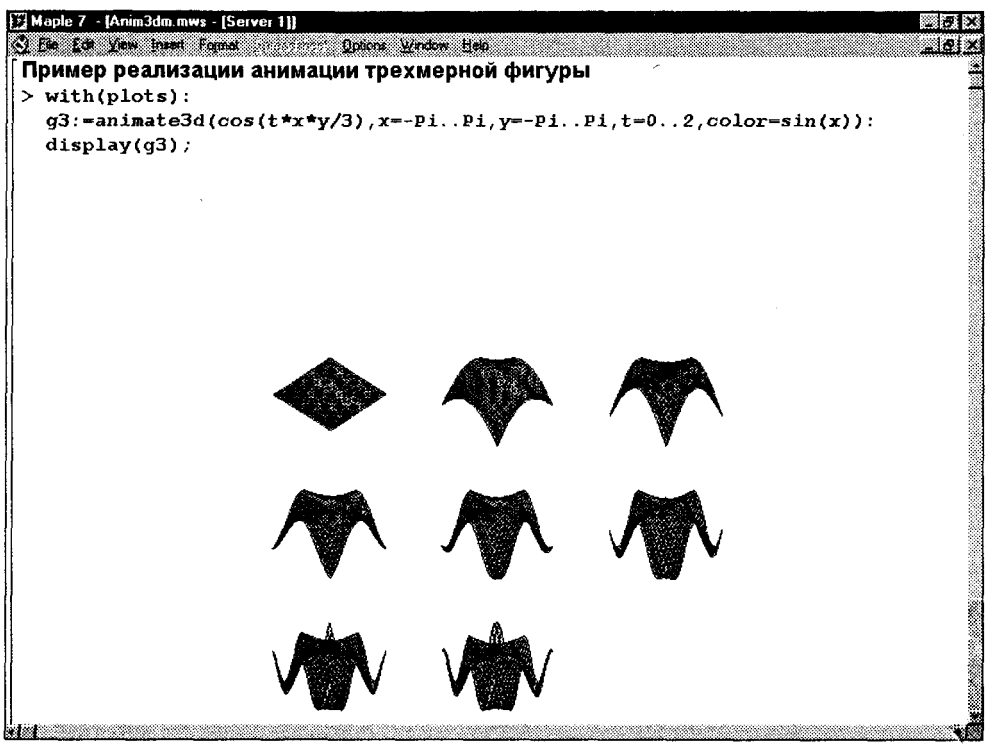


Рис. 12.53. Фазы анимации трехмерной поверхности

Применение анимации дает повышенную степень визуализации решений ряда задач, связанных с построением двумерных и трехмерных графиков. Следует отметить, что построение анимированных графиков требует дополнительных и достаточно существенных затрат оперативной памяти. Поэтому злоупотреблять числом стоп-кадров таких графиков не стоит.

Новая функция для построения стрелок arrow

В пакет plots системы Maple 7 введена новая функция построения стрелок `arrow`. Она задается в виде `arrow(u,[v,]opts)` или `arrow(U,opts)`

Построение стрелок задается одномерными массивами координат начала стрелок и их направления `u` и `v` или двумерным массивом `U`, которые могут быть представлены векторами, списками или множествами. Вид стрелок задается параметром `opts`, который может иметь значения `shape`, `length`, `width`, `head_width`, `head_length` или `plane` и задает вид стрелок (форму, длину, ширину и т. д.). Детали задания параметров можно найти в справке по данной функции. Рисунок 12.54 дает наглядное представление о ее возможностях.

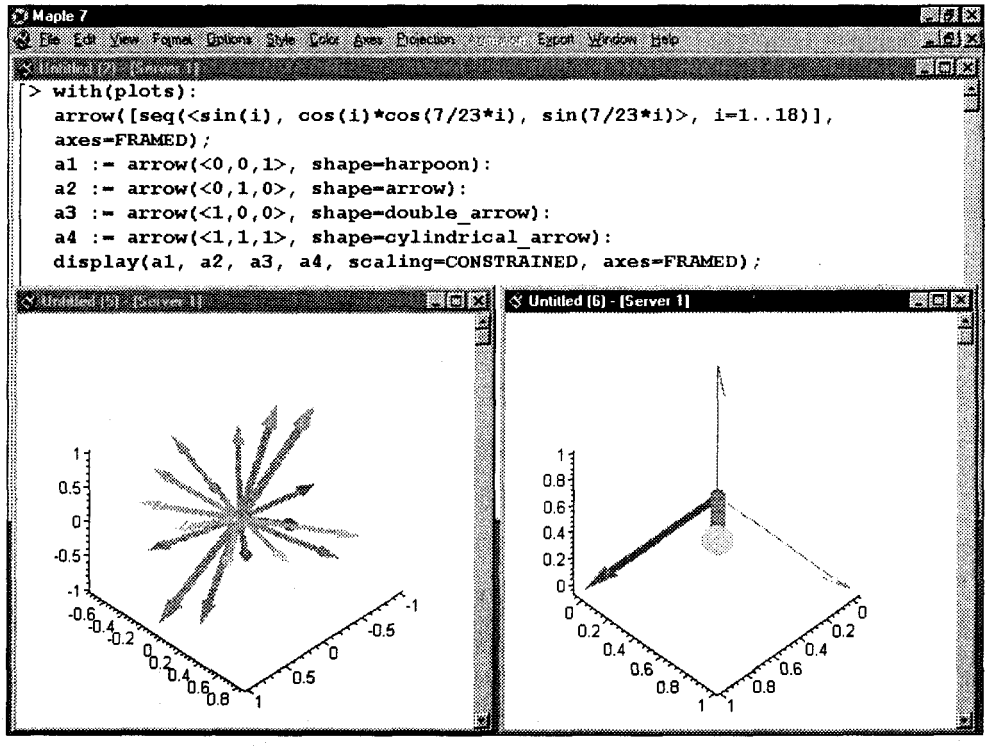


Рис. 12.54. Построение стрелок с помощью функции arrow

Построение сложных комбинированных графиков

В заключение этой главы рассмотрим построение с помощью системы Maple 7 достаточно сложных комбинированных графиков, содержащих различные графические и текстовые объекты. Пример построения такого графика представлен на рис. 12.55.

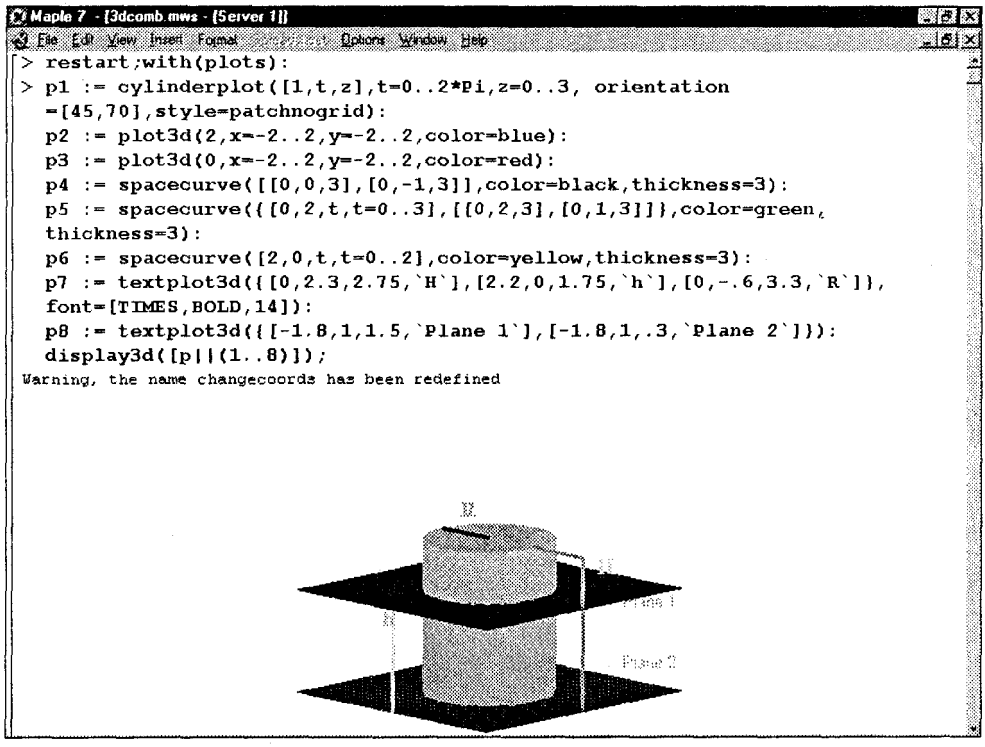


Рис. 12.55. Пример построения сложного объекта, состоящего из 8 графических и текстовых объектов

Представленный на рис. 12.55 объект задает построение восьми графических объектов от p1 до p8. Среди них цилиндр, две пересекающие его плоскости и иные (в том числе текстовые) объекты. Обратите внимание на способ вывода этих объектов функцией display3d. Этот пример показывает, что с помощью графических программных средств Maple 7 можно строить достаточно замысловатые графики, которые могут использоваться для визуализации тех или иных геометрических и иных объектов.

Что нового мы узнали?

В этом уроке мы научились:

- Пользоваться графикой пакета plots.
- Пользоваться техникой анимации графиков пакета plots.
- Применять графику пакета plottools.
- Использовать расширенные средства графической визуализации.
- Использовать расширенную технику создания графической анимации.
- Использовать новую функцию Maple 7 для построения стрелок.
- Строить сложные комбинированные графики.

Решение дифференциальных уравнений

- Основная функция dsolve
- Решение дифференциальных уравнений первого порядка
- Решение дифференциальных уравнений второго порядка
- Решение систем дифференциальных уравнений
- Численное решение дифференциальных уравнений
- Дифференциальные уравнения с кусочными функциями
- Структура неявного представления дифференциальных уравнений DESol
- Инструментальный пакет решения дифференциальных уравнений DEtools
- Графическое представление решений дифференциальных уравнений
- Углубленный анализ решений дифференциальных уравнений

Основные средства решения дифференциальных уравнений

Основная функция dsolve

Важное место в математических расчетах занимает решение дифференциальных уравнений. К нему, в частности, обычно относится анализ поведения различных систем во времени (анализ динамики), а также вычисление различных полей (тяготения, электрических зарядов и т. д.). Трудно переоценить роль дифференциальных уравнений в моделировании физических и технических объектов и систем.

Maple 7 позволяет решать одиночные дифференциальные уравнения и системы дифференциальных уравнений как аналитически, так и в численном виде. Разработчиками системы объявлено о существенном расширении средств решения дифференциальных уравнений и о повышении их надежности в смысле нахождения решений для большинства классов дифференциальных уравнений. Поэтому данный урок целиком посвящен решению уравнений данного класса.

Для решения системы простых дифференциальных уравнений (задача Коши) используется функция `dsolve` в разных формах записи:

```
dsolve(ODE)
dsolve(ODE, y(x), extra_args)
dsolve({ODE, ICs}, y(x), extra_args)
dsolve({sysODE, ICs}, {funcs}, extra_args)
```

Здесь `ODE` — одно обыкновенное дифференциальное уравнение или система из дифференциальных уравнений первого порядка с указанием начальных условий, `y(x)` — функция одной переменной, `Ics` — выражение, задающее начальные условия, `{sysODE}` —множество дифференциальных уравнений, `{funcs}` —множество неопределенных функций, `extra_argument` — опция, задающая тип решения.

Параметр `extra_argument` задает класс решаемых уравнений. Отметим основные значения этого параметра:

- `exact` — аналитическое решение (принято по умолчанию);
- `explicit` — решение в явном виде;
- `system` — решение системы дифференциальных уравнений;
- `ICs` — решение системы дифференциальных уравнений с заданными начальными условиями;

- `formal series` — решение в форме степенного многочлена;
- `integral transform` — решение на основе интегральных преобразований Лапласа, Фурье и др.;
- `series` — решение в виде ряда с порядком, указываемым значением переменной `Order`;
- `numeric` — решение в численном виде.

Для решения задачи Коши в параметры `dsolve` надо включать начальные условия, а при решении краевых задач — краевые условия. Если Maple способна найти решение при числе начальных или краевых условий меньшего порядка системы, то в решении будут появляться неопределенные константы вида `_C1`, `_C2` и т. д. Они же могут быть при аналитическом решении системы, когда начальные условия не заданы. Если решение найдено в неявном виде, то в нем появится параметр `_T`.

По умолчанию функция `dsolve` автоматически выбирает наиболее подходящий метод решения дифференциальных уравнений. Однако в параметрах функции `dsolve` в квадратных скобках можно указать предпочтительный метод решения дифференциальных уравнений. Допустимы следующие методы:

```
quadrature      linear          Bernoulli       separable
inverse_linear  homogeneous     Chini           lin_sym
exact           Abel            pot_sym
```

Информацию о каждом методе можно получить, используя команду `?dsolve,method` и указав в ней конкретный метод. Например, команда `?dsolve,linear` вызовет появление страницы справочной системы с подробным описанием линейного метода решения дифференциальных уравнений.

Производные при записи дифференциальных уравнений могут задаваться функцией `diff` или оператором `D`. Выражение `sysODE` должно иметь структуру множества и содержать помимо самой системы уравнений их начальные условия.

Решение ОДУ первого порядка

Начнем рассмотрение практических примеров с решения одиночных обыкновенных дифференциальных уравнений (ОДУ) первого порядка:

```
> dsolve(diff(y(x),x)-a*x=0,y(x));
```

$$y(x)=\frac{1}{2}a\,x^2+\_C1$$

```
> dsolve(diff(y(x),x)-y(x)=exp(-x),y(x));
```

$$y(x)=\left(-\frac{1}{2}e^{(-2x)}+\_C1\right)e^{x}$$

```
> dsolve(diff(y(x),x)-y(x)=sin(x)*x,y(x));
```

$$y(x)=-\frac{1}{2}\cos(x)\,x-\frac{1}{2}\cos(x)-\frac{1}{2}\sin(x)\,x+e^{x}\,\_C1$$

Следующие примеры иллюстрируют возможность решения одного и того же дифференциального уравнения `ode_1` разными методами:

```
> ode_L := sin(x)*diff(y(x),x)-cos(x)*y(x)=0;
```

$$ode\_L := \sin(x)\left(\frac{\partial}{\partial x}y(x)\right) - \cos(x)\,y(x) = 0$$

```
> dsolve(ode_L, [linear], useInt);
```

$$y(x) = \_C1\, e^{\left(\int \frac{\cos(x)}{\sin(x)}\,dx\right)}$$

```
> value(%);
```

$$y(x) = \_C1\,\sin(x)$$

```
> dsolve(ode_L, [separable], useInt);
```

$$\int \frac{\cos(x)}{\sin(x)}\,dx - \int^{y(x)} \frac{1}{\_a}\,d\_a + \_C1 = 0$$

```
> value(%);
```

$$\ln(\sin(x)) - \ln(y(x)) + \_C1 = 0$$

```
> mu := intfactor(ode_L);
```

$$\mu := \mathrm{intfactor}\left(\sin(x)\left(\frac{\partial}{\partial x}y(x)\right) - \cos(x)\,y(x) = 0\right)$$

```
> dsolve( mu*ode_L, [exact], useInt);
```

$$y(x) = \frac{\_C1}{\frac{1}{2}\tan\left(\frac{1}{2}x\right) + \frac{\frac{1}{2}}{\tan\left(\frac{1}{2}x\right)}}$$

Объем данной книги не позволяет остановиться на всех тонкостях аналитического решения дифференциальных уравнений. Множество примеров такого решения дано в справочной базе данных Maple 7. К ней нужно обратиться в случае, если решение того или иного дифференциального уравнения выходит за рамки учебного курса.

Решение дифференциальных уравнений второго порядка

Здесь видно, что для задания производной используется ранее рассмотренная функция `diff`. С помощью символа `$` можно задать производную более высокого порядка. Ниже представлено решение двух дифференциальных уравнений второго порядка:

```
> dsolve(diff(y(x),x$2)-diff(y(x),x)=sin(x),y(x));
```

$$y(x) = -\frac{1}{2}\sin(x) + \frac{1}{2}\cos(x) + e^{x}\,\_C1 + \_C2$$

```
> de:=m*diff(y(x),x$2)-k*diff(y(x),x);
```

$$de := m\left(\frac{\partial^2}{\partial x^2}y(x)\right) - k\left(\frac{\partial}{\partial x}y(x)\right)$$

```
> yx0:=y(0)=0,y(1)=1;
```

$$yx0 := y(0) = 0, y(1) = 1$$

```
> dsolve({de,yx0},y(x));
```

$$y(x) = -\frac{1}{-1+e^{\left(\frac{k}{m}\right)}} + \frac{e^{\left(\frac{kx}{m}\right)}}{-1+e^{\left(\frac{k}{m}\right)}}$$

Обратите внимание на решение второго из этих уравнений. Здесь использован прием визуализации исходного дифференциального уравнения, и оно задается значением переменной de. Кроме того, и это особенно важно, решение осуществляется при заданных начальных условиях. Именно поэтому в решении отсутствуют произвольные постоянные вида _CN.

Решение систем дифференциальных уравнений

На рис. 13.1 представлено решение системы из двух дифференциальных уравнений различными методами — в явном виде, в виде разложения в ряд и с использованием преобразования Лапласа. Здесь следует отметить, что решение в виде ряда является приближенным. Поэтому полученные в данном случае аналитические выражения отличаются от явного решения и решения с применением преобразования Лапласа.

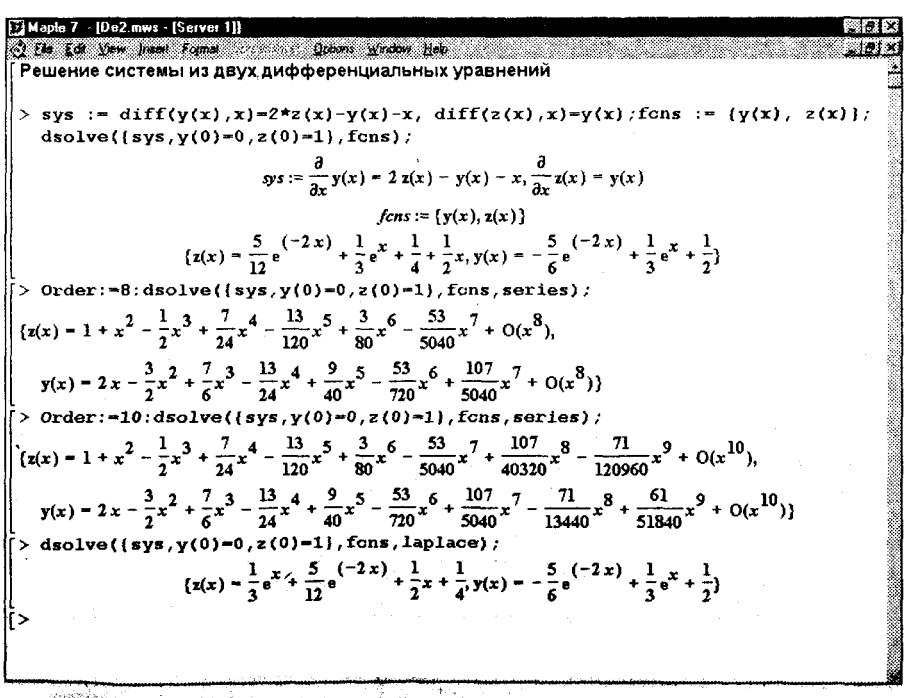


Рис. 13.1. Решение системы из двух дифференциальных уравнений различными методами

Следует отметить, что, несмотря на обширные возможности Maple 7 в области аналитического решения дифференциальных уравнений, оно возможно далеко не всегда. Поэтому, если не удается получить такое решение, полезно попытаться найти решение в численном виде.

Численное решение дифференциальных уравнений

Большинство нелинейных дифференциальных уравнений не имеет аналитического решения. Кроме того, часто аналитическое решение и не нужно, но требуется получить ответ в виде графических зависимостей.

В таких случаях для решения дифференциальных уравнений в численном виде используется функция `dsolve` с параметром `numeric` или `type=numeric`. При этом решение возвращается в виде специальной процедуры, по умолчанию реализующей широко известный метод решения дифференциальных уравнений Рунге—Кутта—Фелберга порядков 4 и 5 (в зависимости от условий адаптации решения к скорости его изменения). Эта процедура называется `rkf45` и символически выводится (без тела) при попытке решения заданной системы дифференциальных уравнений. Последнее достаточно наглядно иллюстрирует рис. 13.2.

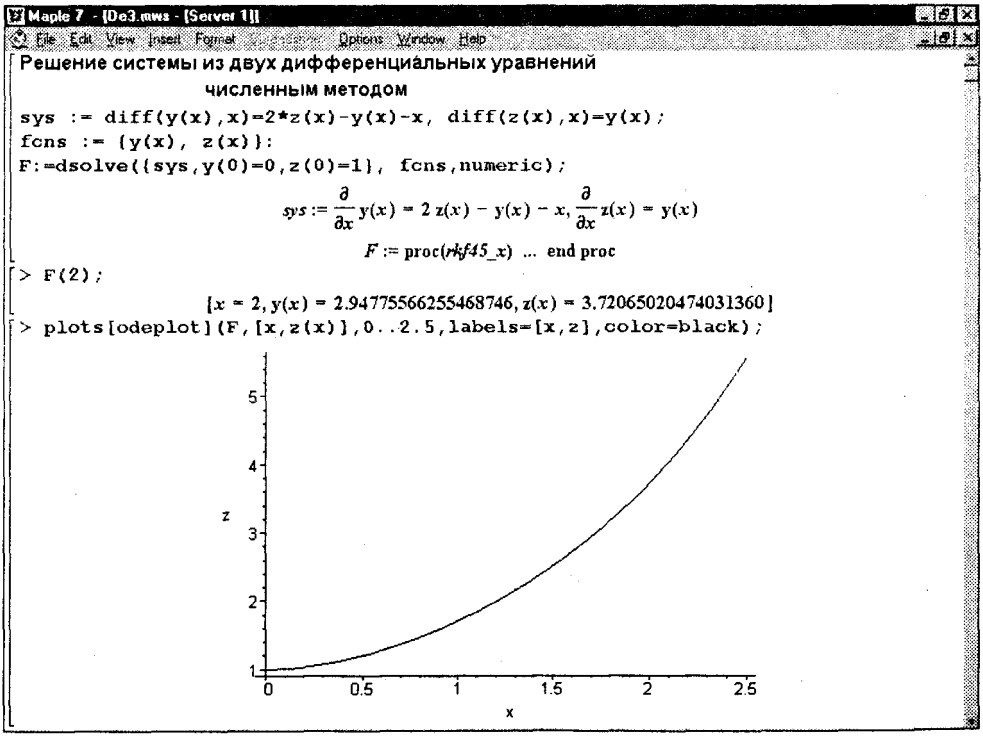


Рис. 13.2. Решение системы дифференциальных уравнений численным методом rkf45 с выводом графика решения

Указанная процедура возвращает особый тип данных, позволяющих найти решение в любой точке или построить график решения (или решений). Для графического отображения Maple 7 предлагает ряд возможностей, и одна из них представлена на рис. 13.2 — см. последнюю строку ввода. При этом используется функция `plot[odeplot]` из пакета `odeplot`, предназначенного для визуализации решений дифференциальных уравнений.

В список параметров функции `dsolve` можно явным образом включить указание на метод решения, например опция `mathod=dverk78` задает решение непрерывным методом Рунге—Кутта порядка 7 или 8. Вообще говоря, численное решение дифференциальных уравнений можно производить одним из следующих методов:

- `classical` — одна из восьми версий классического метода, используемого по умолчанию;
- `rkf45` — метод Рунге—Кутта 4 или 5 порядка, модифицированный Фелбергом;
- `dverk78` — непрерывный метод Рунге—Кутта порядка 7 или 8;
- `gear` — одна из двух версий одношагового экстраполяционного метода Гира;
- `mgear` — одна из трех версий многошагового экстраполяционного метода Гира;
- `lsode` — одна из восьми версий Ливенморского решателя жестких дифференциальных уравнений;
- `taylorseries` — метод разложения в ряд Тейлора.

Обилие используемых методов расширяет возможности решения дифференциальных уравнений в численном виде. Большинство пользователей Maple 7 вполне устроит автоматический выбор метода решения по умолчанию. Однако в сложных случаях возможна прямая установка одного из указанных выше методов. С деталями реализации методов можно ознакомиться по справочной системе.

С помощью параметра `'abserr'=aerr` можно задать величину абсолютной погрешности решения, а с помощью `'minerr'=mine` — минимальную величину погрешности. В большинстве случаев эти величины, заданные по умолчанию, оказываются приемлемыми для расчетов.

Maple 7 реализует адаптируемые к ходу решения методы, при которых шаг решения `h` автоматически меняется, подстраиваясь под условия решения. Так, если прогнозируемая погрешность решения становится больше заданной, шаг решения автоматически уменьшается. Более того, система Maple способна автоматически выбирать наиболее подходящий для решаемой задачи метод решения.

Еще один пример решения системы дифференциальных уравнений представлен на рис. 13.3. Здесь на одном графике представлены зависимости $y(x)$ и $z(x)$, представляющие полное решение заданной системы. При этом процедура имеет особый вид `listprocedure` и для преобразования списка выходных данных в векторы решения `Y` и `Z` используется функция `subs`.

Для решения достаточно сложных задач полезны специальная структура `DESol` для решения дифференциальных уравнений и инструментальный пакет `DEtools`, содержащий самые изысканные средства для графической визуализации результатов решения дифференциальных уравнений. Эти средства мы более подробно рассмотрим в дальнейшем.

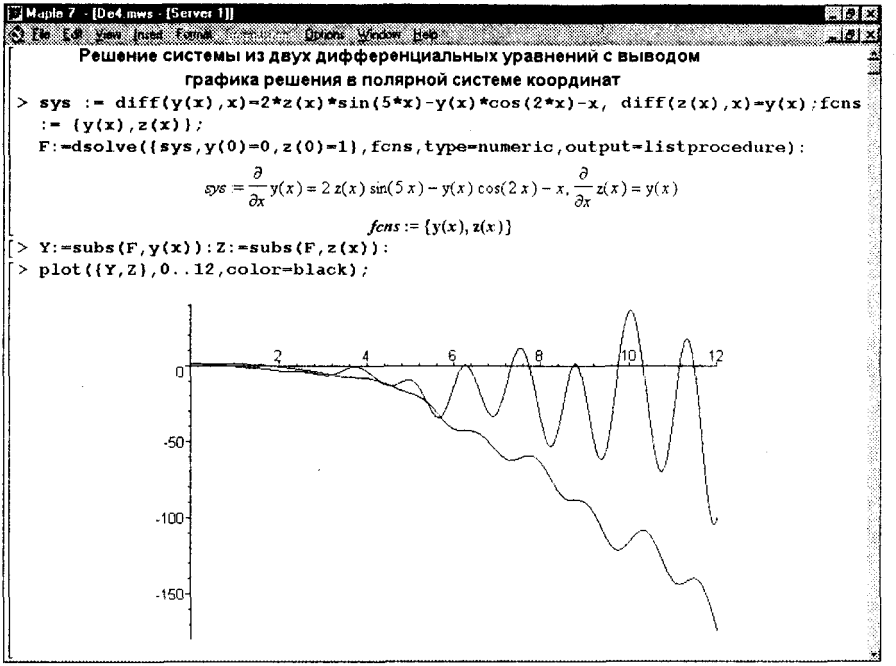


Рис. 13.3. Решение системы дифференциальных уравнений численным методом с выводом всех графиков искомых зависимостей

ВНИМАНИЕ

При решении некоторых задач физики и радиоэлектроники выбираемый по умолчанию шаг изменения аргумента x или t – h может привести к неустойчивости решения. Неустойчивости можно избежать рядом способов. Можно, например, нормировать уравнения, избегая необходимости использования малого шага. А можно задать заведомо малый шаг. Например, при method=classical для этого служит параметр stepsize=h. Примеры такого подхода будут даны в уроке 17 (см. Решение физических задач и моделирование цепи на туннельном диоде).

Дифференциальные уравнения с кусочными функциями

Функции кусочного типа широко используются при математическом моделировании различных физических объектов и систем. В основе такого моделирования обычно лежит решение дифференциальных уравнений, описывающих поведение объектов и систем. Покажем возможность применения кусочных функций для решения дифференциальных уравнений.

Ниже представлено задание дифференциального уравнения первого порядка, содержащего кусочную функцию:

```
> eq := diff(y(x), x)+ piecewise(x<x^2-3, exp(x/2))*y(x);
```

$$eq := \left(\frac{\partial}{\partial x} y(x)\right) + \left(\begin{cases} e^{(1/2)x} & x < x^2 - 3 \\ 0 & otherwise \end{cases} \right) y(x)$$

Используя функцию dsolve, выполним решение этого дифференциального уравнения:

```
> dsolve(eq, y(x));
```

$$y(x) = \begin{cases} \_C1\, e^{(-2\, e^{(1/2\, x)})} & x < \frac{1}{2} - \frac{1}{2}\sqrt{13} \\ \_C1\, e^{(-2\, e^{(1/4 - 1/4\sqrt{13})})} & x < \frac{1}{2} + \frac{1}{2}\sqrt{13} \\ \_C1\, e^{(-2\, e^{(1/4 - 1/4\sqrt{13})} - 2\, e^{(1/2\, x)} + 2\, e^{(1/4 + 1/4\sqrt{13})})} & \frac{1}{2} + \frac{1}{2}\sqrt{13} \le x \end{cases}$$

Нетрудно заметить, что результат получен также в форме кусочной функции, полностью определяющей решение на трех интервалах изменения x.

Приведем пример решения дифференциального уравнения второго порядка с кусочной функцией:

```
> eq := diff(y(x), x$2) + x*diff(y(x), x) + y(x) = piecewise(x > 0, 1);
```

$$eq := \left(\frac{\partial^2}{\partial x^2} y(x)\right) + x\left(\frac{\partial}{\partial x} y(x)\right) + y(x) = \begin{cases} 1 & 0 < x \\ 0 & otherwise \end{cases}$$

```
> dsolve(eq, y(x));
```

$$y(x) = \begin{cases} e^{(-1/2x^2)}\mathrm{erf}\left(\frac{1}{2}1\sqrt{2}x\right)\_C1 + e^{(-1/2x^2)}\_C2 & x < 0 \\ e^{(-1/2x^2)}\mathrm{erf}\left(\frac{1}{2}1\sqrt{2}x\right)\_C1 + e^{(-1/2x^2)}\_C2 + e^{(-1/2x^2)}e^{(1/2x^2)} - e^{(-1/2x^2)} & 0 \le x \end{cases}$$

В конце этого раздела приведем пример решения нелинейного дифференциального уравнения Риккати с кусочной функцией:

```
> eq := diff(y(x), x)=piecewise(x>0, x)*y(x)^2;
```

$$eq := \frac{\partial}{\partial x} y(x) = \left(\begin{cases} x & 0 < x \\ 0 & otherwise \end{cases}\right) y(x)^2$$

```
> dsolve({ y(0) = 1, eq }, y(x));
```

$$y(x) = \left(\begin{cases} \frac{1}{\_C1} & x < 0 \\ -2\frac{1}{x^2 - 2\,\_C1} & 0 \le x \end{cases}\right)\Bigg|_{\_C1 = 1}$$

В ряде случаев желательна проверка решения дифференциальных уравнений. Ниже показано, как она делается для последнего уравнения:

```
> simplify( eval(subs(%, eq)));
```

$$\begin{cases} 0 & x \le 0 \\ 4\frac{x}{(x^2-2)^2} & 0 < x \end{cases} = \begin{cases} 0 & x \le 0 \\ 4\frac{x}{(x^2-2)^2} & 0 < x \end{cases}$$

ПРИМЕЧАНИЕ Как видно из приведенных достаточно простых и наглядных примеров, результаты решения дифференциальных уравнений с кусочными функциями могут быть довольно громоздкими. Это, однако, не мешает эффективному применению функций данного класса.

Структура неявного представления дифференциальных уравнений — DESol

В ряде случаев иметь явное представление дифференциальных уравнений нецелесообразно. Для неявного их представления в Maple 7 введена специальная структура:

```
DESol(expr,vars)
```

где `exprs` — выражение для исходной системы дифференциальных уравнений, `vars` — заданный в виде опции список переменных (или одна переменная).

Структура `DESol` образует некоторый объект, дающий представление о дифференциальных уравнениях, чем-то напоминающее `RootOf`. С этим объектом можно обращаться, как с функцией, то есть его можно интегрировать, дифференцировать, получать разложение в ряд и вычислять численными методами. На рис. 13.4 показаны примеры применения структуры `DESol`.

Обратите внимание на последний пример — в нем структура `DESol` использована для получения решения дифференциального уравнения в виде степенного ряда.

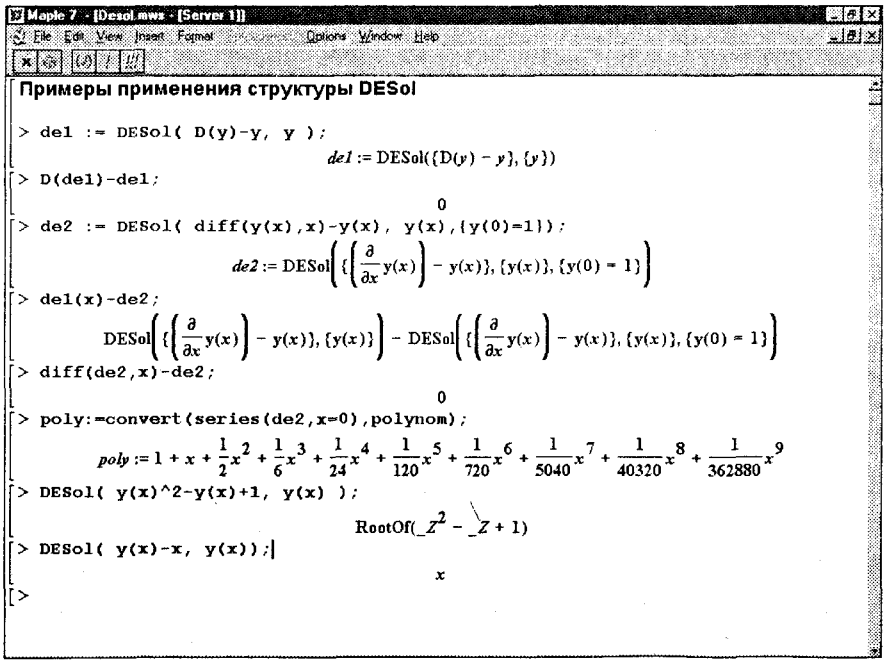


Рис. 13.4. Примеры применения структуры DESol

Инструментальный пакет решения дифференциальных уравнений DEtools

Средства пакета DEtools

Решение дифференциальных уравнений самых различных типов — одно из достоинств системы Maple 7. Пакет `DEtools` предоставляет ряд полезных функций для решения дифференциальных уравнений и систем с такими уравнениями:

```
> with(DEtools);
Warning, the name adjoint has been redefined
```

[*DEnormal, DEplot, DEplot3d, DEplot_polygon, DFactor, DFactorLCLM, DFactorsols, Dchangevar, GCRD, LCLM, PDEchangecoords, RiemannPsols, Xchange, Xcommutator, Xgauge, abelsol, adjoint, autonomous, bernoullisol, buildsol, buildsym, canoni, caseplot, casesplit, checkrank, chinisol, clairautsol, constcoeffsols, convertAlg, convertsys, dalembertsol, dcoeffs, de2diffop, dfieldplot, diffop2de, dsubs,*

eigenring, endomorphism_charpoly, equinv, eta_k, eulersols, exactsol, expsols, exterior_power, firint, firtest, formal_sol, gen_exp, generate_ic, genhomosol, gensys, hamilton_eqs, indicialeq, infgen, initialdata, integrate_sols, intfactor, invariants, kovacicsols, leftdivision, liesol, line_int, linearsol, matrixDE, matrix_riccati, moser_reduce, muchange, mult, mutest, newton_polygon, normalG2, odeadvisor, odepde, parametricsol, phaseportrait, poincare, polysols, ratsols, redode, reduceOrder, reduce_order, regular_parts, regularsp, remove_RootOf, riccati_system, riccatisol, rifsimp, rightdivision, rtaylor, separablesol, solve_group, super_reduce, symgen, symmetric_power, symmetric_product, symtest, transinv, translate, untranslate, varparam, zoom]

Этот пакет дает самые изысканные средства для аналитического и численного решения дифференциальных уравнений и систем с ними. По сравнению с версией Maple V R5 число функций данного пакета в Maple 7 возросло в несколько раз. Многие графические функции пакета `DEtools` были уже описаны. Ниже приводятся полные наименования тех функций, которые есть в реализациях R5, 6 и 7 системы Maple:

- `DEnormal` — возвращает нормализованную форму дифференциальных уравнений;
- `DEplot` — строит графики решения дифференциальных уравнений;
- `DEplot3d` — строит трехмерные графики для решения систем дифференциальных уравнений;
- `Dchangevar` — изменение переменных в дифференциальных уравнениях;

- PDEchangecoords — изменение координатных систем для дифференциальных уравнений в частных производных;
- PDEplot — построение графиков решения дифференциальных уравнений в частных производных;
- autonomous — тестирует дифференциальные уравнения на автономность;
- convertAlg — возвращает список коэффициентов для дифференциальных уравнений;
- convertsys — преобразует систему дифференциальных уравнений в систему одиночных уравнений;
- dfieldplot — строит график решения дифференциальных уравнений в виде векторного поля;
- indicialeq — преобразует дифференциальные уравнения в полиномиальные;
- phaseportrait — строит график решения дифференциальных уравнений в форме фазового портрета;
- reduceOrder — понижает порядок дифференциальных уравнений;
- regularsp — вычисляет регулярные особые точки для дифференциальных уравнений второго порядка;
- translate — преобразует дифференциальные уравнения в список операторов;
- untranslate — преобразует список операторов в дифференциальные уравнения;
- varparam — находит общее решение дифференциальных уравнений методом вариации параметров.

Применение этих функций гарантирует совместимость документов реализаций Maple R5, 6 и 7.

Основные функции пакета DEtools

Ввиду обилия функций пакета DEtools дать их полное описание в данной книге не представляется возможным. Поэтому выборочно рассмотрим наиболее важные функции этого пакета. Функция:

```
autonomous(des,vars,ivar)
```

тестирует дифференциальное уравнение (или систему) des. Ее параметрами помимо des являются независимая переменная ivar и зависимая переменная dvar. Следующие примеры поясняют применение этой функции:

```
> autonomous(sin(z(t)-z(t)^2)*(D@@4)(z)(t)-cos(z(t))-5,z,t);
```

true

```
> DE:=diff(x(s),s)-x(s)*cos(arctan(x(s)))=arctan(s):
> autonomous(DE,{x},s);
```

false

Функция Dchangevar используется для обеспечения замен (подстановок) в дифференциальных уравнениях:

```
Dchangevar(trans, deqns, c_ivar, n_ivar)
Dchangevar(tran1, tran2, ..., tranN, deqns, c_ivar, n_ivar)
```

В первом случае trans — список или множество уравнений, которые подставляются в дифференциальное уравнение, список или множество дифференциальных уравнений deqns. При этом c_ivar — имя текущей переменной, n_ivar — имя новой переменной (его задавать необязательно). Во второй форме для подстановки используются уравнения tran1, tran2, ...

Ниже представлены примеры применения функции Dchangevar:

```
# Преобразование 1-го типа
>Dchangevar(m(x)=l(x)*sin(x),n(x)=k(x),[D(m)(x)=m(x),(D@@2)(n)(x)=
n(x)^2],x);
```

$$[D(l)(x)\sin(x)+l(x)\cos(x)=l(x)\sin(x),(D^{(2)})(k)(x)=k(x)^2]$$

```
>Dchangevar(c=d,e=sin(f),{D(c),(D@@2)(e)},dummy);
```

$$[D(d),(D^{(2)})(\sin(f))]$$

```
# Преобразование 2-го типа
> Dchangevar(t=arctan(tau),diff(x(t),t)=sin(t),t,tau);
```

$$D(x)(\arctan(\tau))=\sin(\arctan(\tau))$$

```
> Dchangevar(x=sin(cos(t)),diff(y(x),x,x,x),x,t);
```

$$(D^{(3)})(y)(\sin(\cos(t)))$$

```
# Преобразование 3-го типа
> Dchangevar(x(t)=L*y(phi),diff(x(t),t$3) = tan(t),t,phi);
```

$$\frac{\partial^3}{\partial\phi^3}L\,y(\phi)=\tan(\phi)$$

```
# Дополнительные примеры
> Dchangevar({t=T*phi,x(t)=L*y(phi)},diff(x(t),
t$3)=tan(t),t,phi);
```

$$\frac{L\left(\frac{\partial^3}{\partial\phi^3}y(\phi)\right)}{T^3}=\tan(T\,\phi)$$

```
> de := diff(y(x),x$2) = y(x)*diff(y(x),x)/x;
```

$$de:=\frac{\partial^2}{\partial x^2}y(x)=\frac{y(x)\left(\frac{\partial}{\partial x}y(x)\right)}{x}$$

```
> Dchangevar({x=exp(t),y(x)=Y(t)},de,x,t);
```

$$\frac{-\frac{\partial}{\partial t}Y(t)}{e^t}+\frac{\frac{\partial^2}{\partial t^2}Y(t)}{e^t}}{e^t}=\frac{Y(t)\left(\frac{\partial}{\partial t}Y(t)\right)}{(e^t)^2}$$

Следует отметить, что подстановки являются мощным средством решения дифференциальных уравнений. Нередки случаи, когда дифференциальное уравнение не решается без их применения. Дополнительные примеры использования подстановок можно найти в справочной базе данных системы Maple 7.

Функция нормализации ОДУ DEnormal синтаксически записывается в виде:

```
DEnormal(des,ivar,dvar)
```

где des — система дифференциальных уравнений, ivar — независимая переменная и dvar — зависимая переменная. Применение этой функции поясняют следующие примеры:

```
> DE := x^3*y(x)+x^2*(x-1)*D(y)(x)+50*x^3*(D@@2)(y)(x)=x*sin(x);
```

$$DE := x^3\,y(x) + x^2\,(x-1)\,D(y)(x) + 50\,x^3\,(D^{(2)})(y)(x) = x\sin(x)$$

```
> DE2 := convertAlg(DE,y(x));
```

$$DE2 := [[x^3, x^3 - x^2, 50\,x^3], x\sin(x)]$$

```
> DEnormal(DE,x,y(x));
```

$$x\,y(x) + (x-1)\left(\frac{\partial}{\partial x}y(x)\right) + 50\,x\left(\frac{\partial^2}{\partial x^2}y(x)\right) = \frac{\sin(x)}{x}$$

```
> DEnormal(DE2,x);
```

$$\left[[x, x-1, 50\,x], \frac{\sin(x)}{x}\right]$$

Функция convertAlg(des,dvar) возвращает список коэффициентов формы системы дифференциальных уравнений des с зависимыми переменными dvar. Это поясняют следующие примеры:

```
> A := diff(y(x),x)*sin(x)-diff(y(x),x)-tan(x)*y(x)=5;
```

$$A := \left(\frac{\partial}{\partial x}y(x)\right)\sin(x) - \left(\frac{\partial}{\partial x}y(x)\right) - \tan(x)\,y(x) = 5$$

```
> convertAlg(A,y(x));
```

$$[[-\tan(x), \sin(x) - 1], 5]$$

```
> B := (D@@2)(y)(x)*cos(x) + (D@@2)(y)(x)*5*x^2;
```

$$B := (D^{(2)})(y)(x)\cos(x) + 5\,(D^{(2)})(y)(x)\,x^2$$

```
> convertAlg(B,y(x));
```

$$[[0, 0, \cos(x) + 5\,x^2], 0]$$

Для изменения переменных в системах дифференциальных уравнений используется функция convertsys:

```
convertsys(deqns, inits, vars, ivar, yvec, ypvec)
```

Здесь deqns — одно дифференциальное уравнение или список (множество), представляющие систему дифференциальных уравнений первого порядка, inits — множество или список начальных условий, vars — зависимые переменные, ivar — независимые переменные, yvec — вектор решений и ypvec — вектор производных.

Функция:

```
indicialeq(des,ivar,alpha,dvar)
```

обеспечивает полиномиальное представление для линейного однородного дифференциального уравнения второго порядка des. Параметр alpha намечает точку сингулярности.

```
> Y := (2*x^2+5*x^3)*diff(y(x),x,x)+(5*x-x^2)*diff(y(x),x)+(1+x)*y(x)=0:
> Y := convertAlg( Y, y(x) );
```

$$Y := [[1 + x, 5x - x^2, 2x^2 + 5x^3], 0]$$

```
> indicialeq( Y, x, -2/5, y(x) );
```

$$x^2 - \frac{37}{10}x = 0$$

```
> indicialeq( Y, x, 0, y(x) );
```

$$x^2 + \frac{3}{2}x + \frac{1}{2} = 0$$

```
> indicialeq( Y, x, 1, y(x) );
```

$$x^2 - x = 0$$

Функция:

```
reduceOrder(des,dvar,partsol, solutionForm)
```

обеспечивает понижение порядка дифференциального уравнения des (или системы уравнений, представленных списком или множеством) при зависимых переменных dvar, частном решении partsol (или списке частных решений) и флаге solutionForm, показывающем, что решение происходит явным методом (explicitly).

Для демонстрации действия этой функции воспользуемся примером из ее справочной страницы:

```
> de := diff(y(x),x$3) - 6*diff(y(x),x$2) + 11*diff(y(x),x) - 6*y(x);
```

$$de := \left(\frac{\partial^3}{\partial x^3} y(x)\right) - 6\left(\frac{\partial^2}{\partial x^2} y(x)\right) + 11\left(\frac{\partial}{\partial x} y(x)\right) - 6\, y(x)$$

```
> sol := exp(x);
```

$$sol := e^x$$

```
> reduceOrder( de, y(x), sol);
```

$$\left(\frac{\partial^2}{\partial x^2} y(x)\right) - 3\left(\frac{\partial}{\partial x} y(x)\right) + 2\, y(x)$$

```
> reduceOrder( de, y(x), sol, basis);
```

$$\left[e^x, e^{(2x)}, \frac{1}{2} e^{(3x)}\right]$$

Функция:

```
regularsp(des,ivar,dvar)
```

вычисляет регулярные особые (сингулярные) точки для дифференциального уравнения второго порядка или системы дифференциальных уравнений des. Следующий пример поясняет применение данной функции:

```
> coefs := [21*(x^2 - x + 1),0,100*x^2*(x-1)^2]:
> regularsp(coefs, x);
```

$$[0, 1]$$

Еще две функции пакета DEtools:

```
translate(des,ivar,pt,dvar)
untranslate(des,ivar,pt,dvar)
```

выполняют особую операцию трансляции дифференциального уравнения (или списка дифференциальных уравнений) из центрированного относительно 0 в цен-

трированное относительно 1 и наоборот. С деталями этого специфического процесса заинтересованный читатель может познакомиться в справочной базе данных.

И еще одна полезная функция пакета:

```
varparam(sols,v,ivar)
```

находит общее решение дифференциального уравнения (или системы уравнений) `sols` методом вариации параметров. Параметр `v` задает правую часть уравнения; если он равен `0`, ищется только частичное решение:

```
> varparam( [u1(x), u2(x)], g(x), x);
```

$$\_C_1\,\mathrm{u1}(x)+\_C_2\,\mathrm{u2}(x)+\int -\frac{\mathrm{u2}(x)\,\mathrm{g}(x)}{\mathrm{u1}(x)\left(\frac{\partial}{\partial x}\mathrm{u2}(x)\right)-\mathrm{u2}(x)\left(\frac{\partial}{\partial x}\mathrm{u1}(x)\right)}\,dx\,\mathrm{u1}(x)+$$

$$+\int \frac{\mathrm{u1}(x)\,\mathrm{g}(x)}{\mathrm{u1}(x)\left(\frac{\partial}{\partial x}\mathrm{u2}(x)\right)-\mathrm{u2}(x)\left(\frac{\partial}{\partial x}\mathrm{u1}(x)\right)}\,dx\,\mathrm{u2}(x)$$

Более подробную информацию об этих функциях читатель найдет в их справочных страницах, а также в информационном документе `detools.mws`, содержащем систематизированное описание пакета `DEtools` с многочисленными примерами его применения.

Графическое представление решений дифференциальных уравнений

Применение функции odeplot пакета plots

Для обычного графического представления результатов решения дифференциальных уравнений может использоваться функция `odeplot` из описанного выше пакета `plots`. Эта функция используется в следующем виде:

```
odeplot(s,vars,r,o)
```

где `s` — запись (в выходной форме) дифференциального уравнения или системы дифференциальных уравнений, решаемых численно функцией `dsolve`, `vars` — переменные, `r` — параметр, задающий пределы решения (например, `a..b`), и `o` — необязательные дополнительные опции.

На рис. 13.5 представлен пример решения одиночного дифференциального уравнения с выводом решения $y(x)$ с помощью функции `odeplot`.

В этом примере решается дифференциальное уравнение:

$$y'(x) = \cos(x^2 y(x))$$

при $y(0) = 2$ и x, меняющемся от -5 до 5. Левая часть уравнения записана с помощью функции вычисления производной `diff`. Результатом построения является график решения $y(x)$.

В другом примере (рис. 13.6) представлено решение системы из двух нелинейных дифференциальных уравнений. Здесь с помощью функции `odeplot` строятся графики двух функций — $y(x)$ и $z(x)$.

В этом примере решается система:

$$y'(x) = z(x),$$

$$z'(x) = 3\sin(y(x))$$

при начальных условиях $y(0) = 0$, $z(0) = 1$ и x, меняющемся от -4 до 4 при числе точек решения, равном 25.

Иногда решение системы из двух дифференциальных уравнений (или одного дифференциального уравнения второго порядка) представляется в виде фазового портрета — при этом по осям графика откладываются значения $y(x)$ и $z(x)$ при изменении x в определенных пределах. Рисунок 13.7 демонстрирует построение фазового портрета для системы, представленной выше.

Обычное решение, как правило, более наглядно, чем фазовый портрет решения. Однако для специалистов (например, в теории колебаний) фазовый портрет порою дает больше информации, чем обычное решение. Он более трудоемок для построения, поэтому возможность Maple 7 быстро строить фазовые портреты трудно переоценить.

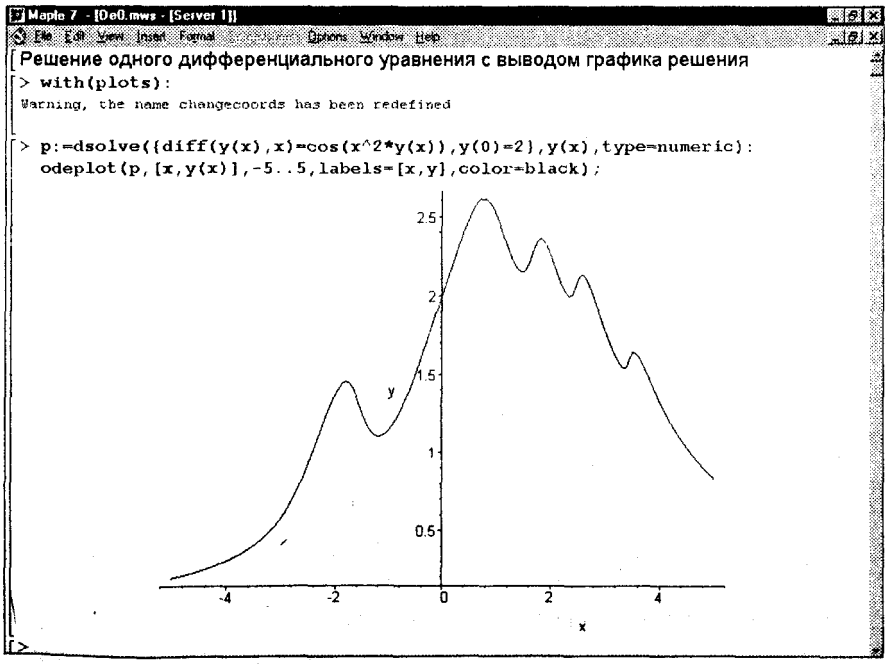


Рис. 13.5. Пример решения одиночного дифференциального уравнения

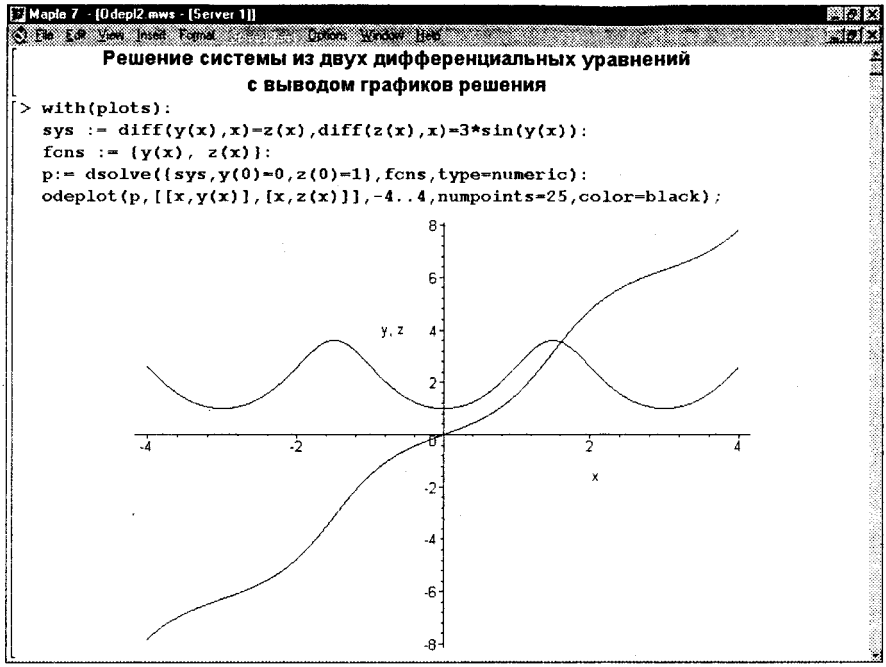


Рис. 13.6. Пример решения системы из двух дифференциальных уравнений

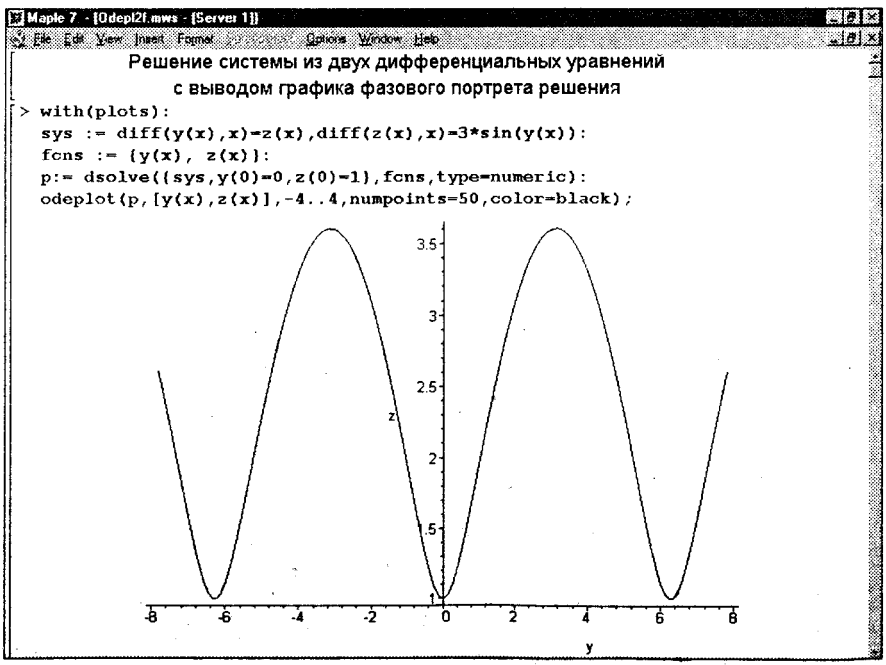


Рис. 13.7. Представление решения системы дифференциальных уравнений в виде фазового портрета

Функция DEplot из пакета DEtools

Специально для решения и визуализации решений дифференциальных уравнений и систем с дифференциальными уравнениями служит инструментальный пакет DEtools. В него входит ряд функций для построения наиболее сложных и изысканных графиков решения дифференциальных уравнений. Основной из этих функций является функция DEplot.

Функция DEplot может записываться в нескольких формах:

```
DEplot(deqns, vars, trange, eqns)
DEplot(deqns, vars, trange, inits, eqns)
DEplot(deqns, vars, trange, yrange, xrange, eqns)
DEplot(deqns, vars, trange, inits, xrange, yrange, eqns)
```

Здесь deqns — список или множество, содержащее систему дифференциальных уравнений первого порядка или одиночное уравнение любого порядка; vars — зависимая переменная или список либо множество зависимых переменных; trange — область изменения независимой переменной t; inits — начальные условия для решения; yrange — область изменения для первой зависимой переменной, xrange — область изменения для второй зависимой переменной; eqns — опция, записываемая в виде keyword=value. Замена имен переменных другими в данном случае недопустима.

Эта функция обеспечивает численное решение дифференциальных уравнений или их систем при одной независимой переменной t и строит графики решения. Для автономных систем эти графики строятся в виде векторного поля направлений, а для неавтономных систем — только в виде кривых решения. По умолчанию реализуется метод Рунге—Кутта 4-го порядка, что соответствует опции method=classical[rk4].

С функцией DEplot могут использоваться следующие параметры:

- arrows = type — тип стрелки векторного поля ('SMALL', 'MEDIUM', 'LARGE', 'LINE' или 'NONE');
- colour, color = arrowcolour — цвет стрелок (задается 7 способами);
- dirgrid = [integer,integer] — число линий сетки (по умолчанию [20, 20]);
- iterations = integer — количество итераций, представленное целым числом;
- linecolor, linecolor = line_info — цвет линии (задается 5 способами);
- method='rk4' — задает метод решения ('euler', 'backeuler', 'impeuler' или 'rk4');
- obsrange = TRUE,FALSE — задает (при TRUE) прерывание вычислений, если кривая решения выходит из области обзора;
- scene = [name,name] — задает имена зависимых переменных, для которых строится график;
- stepsize = h — шаг решения, по умолчанию равный abs((b-a))/20 и представленный вещественным значением.

На рис. 13.8 показано решение системы дифференциальных уравнений:

$$x'(t) = x(t)\,(1 - y(t)),$$
$$y'(t) = 0{,}3\,y(t)\,(x(t) - 1),$$

описывающих модель Лотки—Вольтерра при заданных в документе изменениях t, $x(t)$ и $y(t)$. Решение представлено в виде векторного поля, стрелки которого являются касательными к кривым решения (сами эти кривые не строятся). Обратите внимание на функциональную закраску стрелок векторного поля, делающую решение особенно наглядным (правда, лишь на экране цветного дисплея, а не на страницах книги).

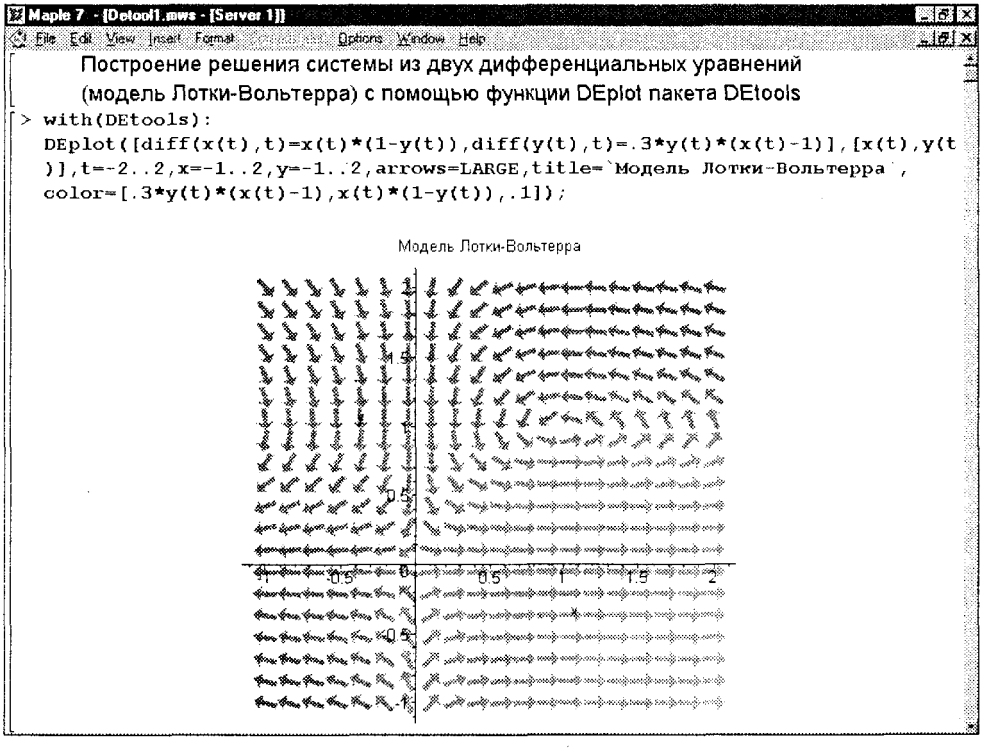


Рис. 13.8. Решение системы дифференциальных уравнений Лотки—Вольтерра с выводом в виде графика векторного поля

Еще интересней вариант графиков, представленный на рис. 13.9. Здесь помимо векторного поля несколько иного стиля построены фазовые портреты решения с использованием функциональной закраски их линий. Фазовые портреты построены для двух наборов начальных условий: $x(0) = y(0) = 1{,}2$ и $x(0) = 1$ и $y(0) = 0{,}7$.

Следует отметить, что функция `DEplot` может обращаться к другим функциям пакета `DEtools` для обеспечения специальных графических возможностей, таких как построение векторного поля или фазового портрета решения.

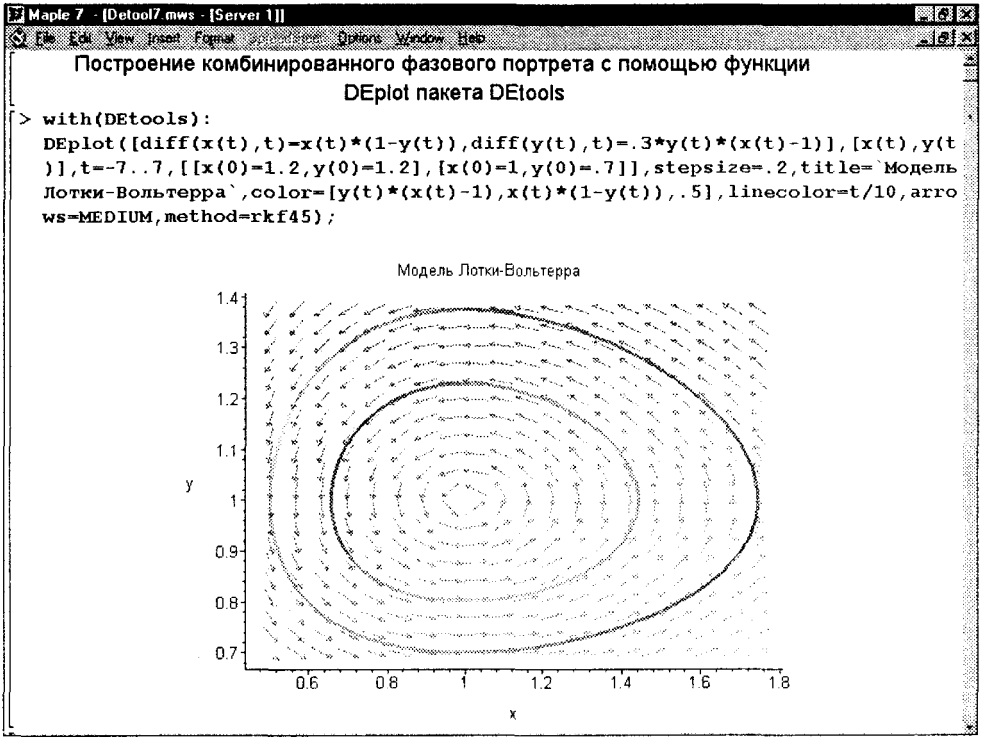


Рис. 13.9. Пример построения двух фазовых портретов на фоне векторного поля

Функция DEplot3d из пакета DEtools

В ряде случаев решение систем дифференциальных уравнений удобно представлять в виде пространственных кривых — например, линий равного уровня или просто в виде кривых в пространстве. Для этого служит функция DEplot3d:

```
DEplot3d(deqns, vars, trange, initset, o)
DEplot3d(deqns, vars, trange, yrange, xrange, initset, o)
```

Назначение параметров этой функции аналогично указанному для функции DEplot.

Рисунок 13.10 поясняет применение функции DEPlot3d для решения системы из двух дифференциальных уравнений с выводом фазового портрета колебаний в виде параметрически заданной зависимости $x(t)$, $y(t)$. В данном случае фазовый портрет строится на плоскости по типу построения графиков линий равной высоты.

Другой пример (рис. 13.11) показывает решение системы из двух дифференциальных уравнений с построением объемного фазового портрета. В этом случае используется трехмерная координатная система и графические построения соответствуют параметрическим зависимостям $x(t)$, $y(t)$ и $z(t)$. Вид фазового портрета напоминает разворачивающуюся в пространстве объемную спираль.

Функциональная окраска делает график пикантным, что, увы, теряется при черно-белом воспроизведении графика.

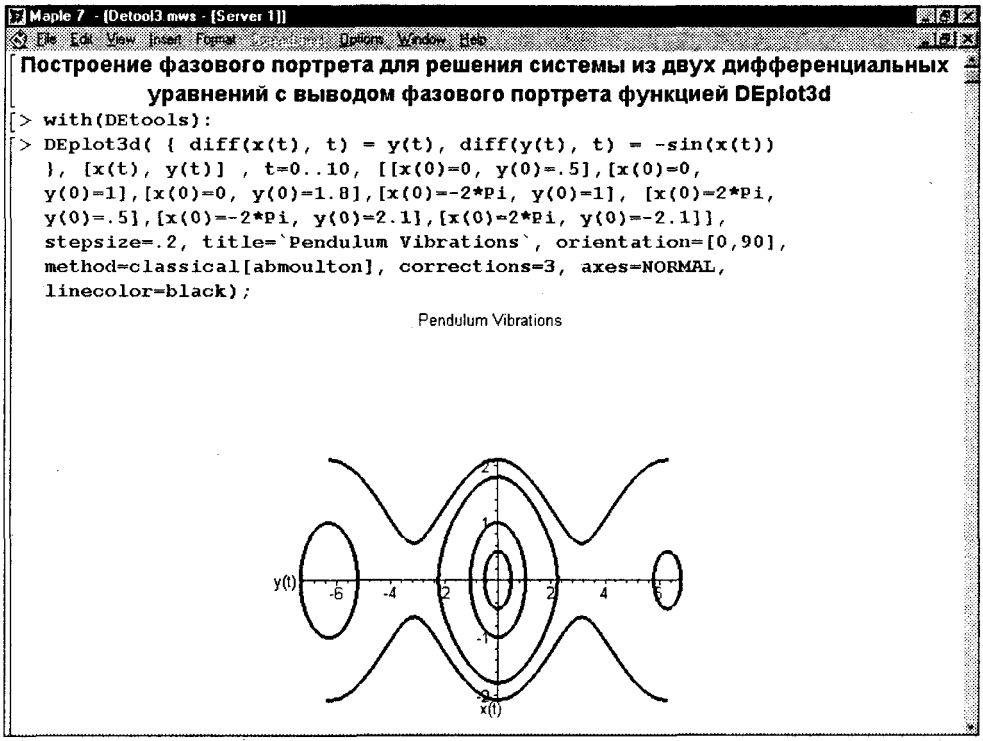


Рис. 13.10. Пример решения системы из двух дифференциальных уравнений с помощью функции DEplot3d

Возможности функции DEplot3d позволяют решать системы, состоящие более чем из двух дифференциальных уравнений. Однако в этом случае число решений, представляемых графически, выходит за пределы возможного для трехмерной графики. При этом от пользователя зависит, какие из зависимостей опускаются при построении, а какие строятся.

Функция PDEplot пакета DEtools

Еще одна функция пакета DEtools — DEtools[PDEplot] — служит для построения графиков решения систем с квазилинейными дифференциальными уравнениями первого порядка в частных производных.

Эта функция используется в следующем виде:

```
PDEplot(pdiffeq, var, i_curve, srange, o)
PDEplot(pdiffeq, var, i_curve, srange, xrange, yrange, urange, o)
```

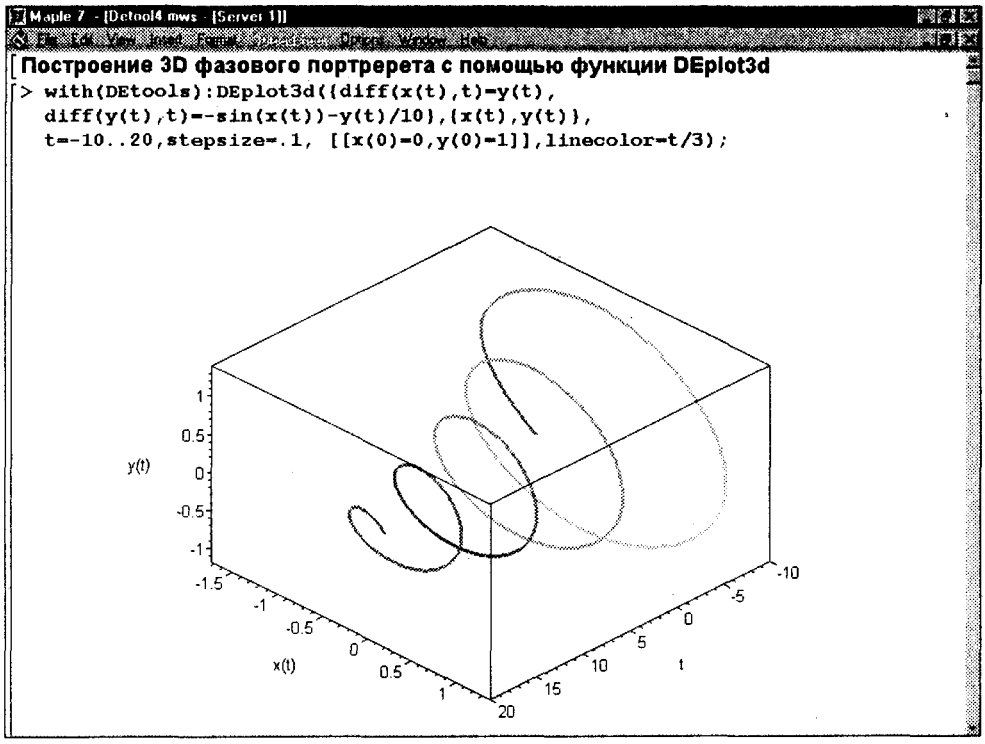


Рис. 13.11. Пример решения системы из двух дифференциальных уравнений с построением трехмерного фазового портрета

Здесь помимо упоминавшихся ранее параметров используются следующие: `pdiffeq` — квазилинейные дифференциальные уравнения первого порядка (PDE), `vars` — независимая переменная и `i_curve` — начальные условия для параметрических кривых трехмерной поверхности. Помимо опций, указанных для функции `DEplot`, здесь могут использоваться следующие опции:

- `basechar = TRUE, FALSE, ONLY` — устанавливает показ базовых характеристик кривых;
- `basecolor, basecolor = b_color` — устанавливает цвет базовых характеристик;
- `initcolor, initcolor = i_color` — инициализация цветов;
- `numchar = integer` — задает число отрезков кривых, которое не должно быть меньше 4 (по умолчанию 20);
- `numsteps = [integer1,integer2]` — задает число шагов интегрирования (по умолчанию [10, 10]).

Рисунок 13.12 демонстрирует применение функции `PDEplot`. Этот пример показывает, насколько необычным может быть решение даже простой системы дифференциальных уравнений в частных производных.

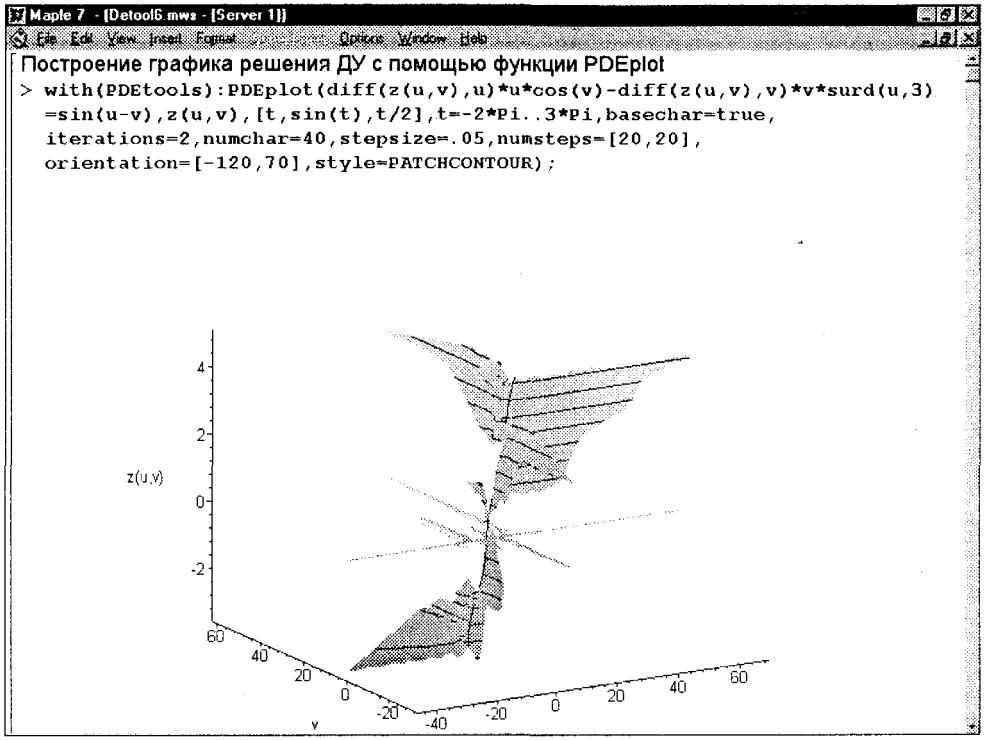


Рис. 13.12. Пример применения функции PDEplot

В данном случае решение представлено трехмерной фигурой весьма нерегулярного вида.

Другой пример использования функции `PDEplot` показан на рис. 13.13. Он иллюстрирует комбинированное построение графиков решения разного типа с применением функциональной закраски, реализуемой по заданной формуле с помощью опции `initcolor`.

Еще раз отметим, что, к сожалению, рисунки в данной книге не дают представления о цвете выводимого Maple графика. Поэтому наглядность решений, видимых на экране монитора, существенно выше.

Графическая функция dfieldplot

Графическая функция `dfieldplot` служит для построения поля направления с помощью векторов по результатам решения дифференциальных уравнений. Фактически эта функция как бы входит в функцию `DEplot` и при необходимости вызывается последней. Но она может использоваться и самостоятельно, что демонстрирует рис. 13.14, на котором показан пример решения следующей системы дифференциальных уравнений: $x'(t) = x(t)(1 - y(t))$, $y'(t) = 0{,}3\, y(t)(x(t) - 1)$.

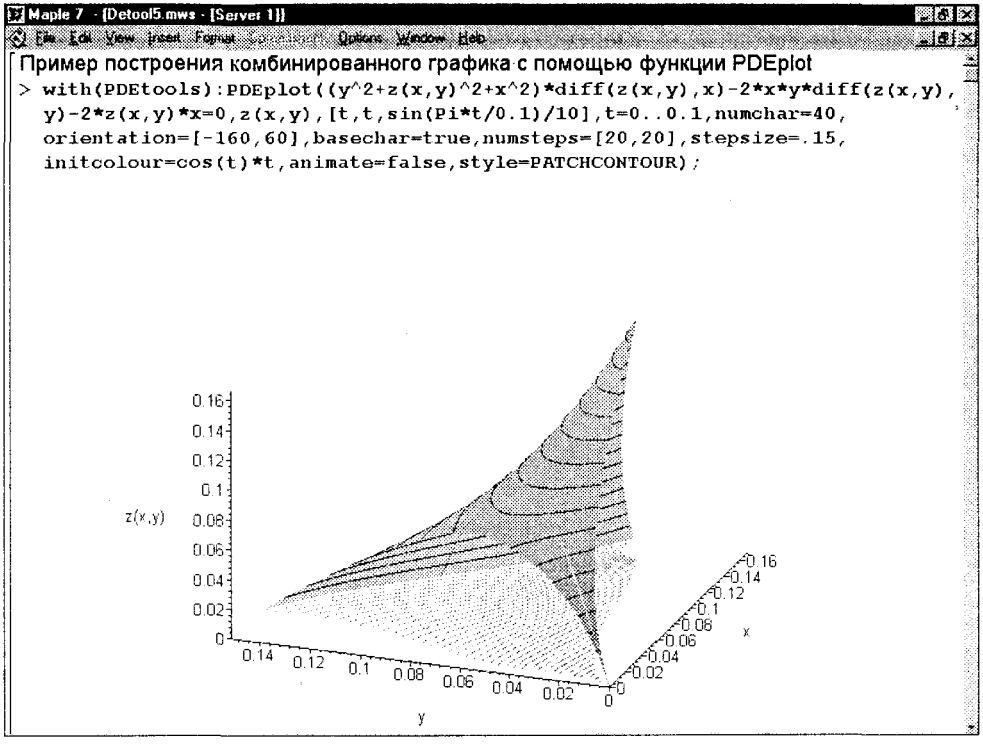


Рис. 13.13. Построение комбинированного графика с помощью функции PDEplot

Обратите внимание на использование опций в этом примере, в частности на вывод надписи на русском языке. В целом список параметров функции `phaseportrait` аналогичен таковому для функции `DEplot` (отсутствует лишь задание начальных условий).

Графическая функция phaseportrait

Графическая функция `phaseportrait` служит для построения фазовых портретов по результатам решения одного дифференциального уравнения или системы дифференциальных уравнений `deqns`. Она задается в следующем виде:

```
phaseportrait(deqns,vars,trange,inits,o)
```

При задании уравнений достаточно указать их правые части. На рис. 13.15 представлен пример применения функции `phaseportrait` для решения системы из трех дифференциальных уравнений первого порядка.

В этом примере система дифференциальных уравнений задана с помощью оператора дифференцирования `D`. Функциональная окраска линии фазового портрета достигается использованием параметра `linecolor`, в правой части которого задана формула для цвета.

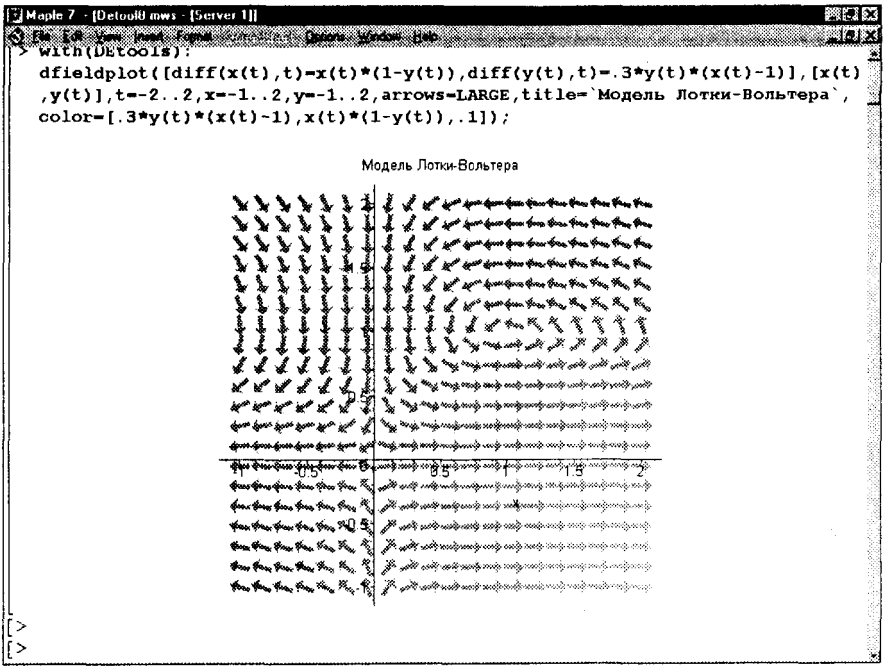


Рис. 13.14. Построение фазового портрета в виде графика векторного поля

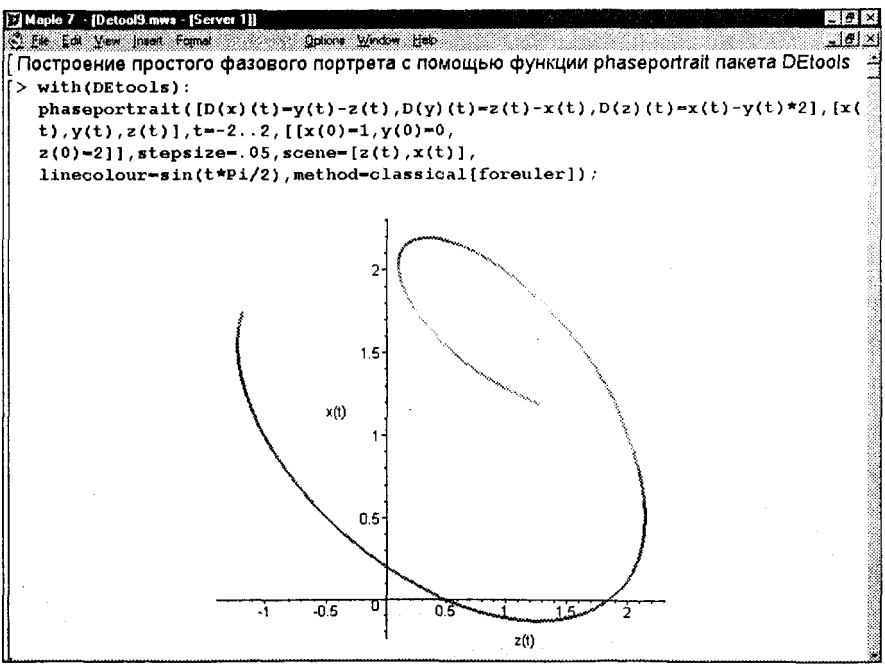


Рис. 13.15. Построение фазового портрета с помощью функции phaserportrait

Еще более интересный пример решения дифференциального уравнения представлен на рис. 13.16. Здесь построены фазовые портреты для асимптотических решений.

В целом надо отметить, что возможности визуализации решений дифференциальных уравнений с помощью системы Maple 7 весьма велики и приведенные выше примеры лишь частично иллюстрируют сказанное. В справочной системе можно найти ряд других весьма эффектных решений систем дифференциальных уравнений с визуализацией последних.

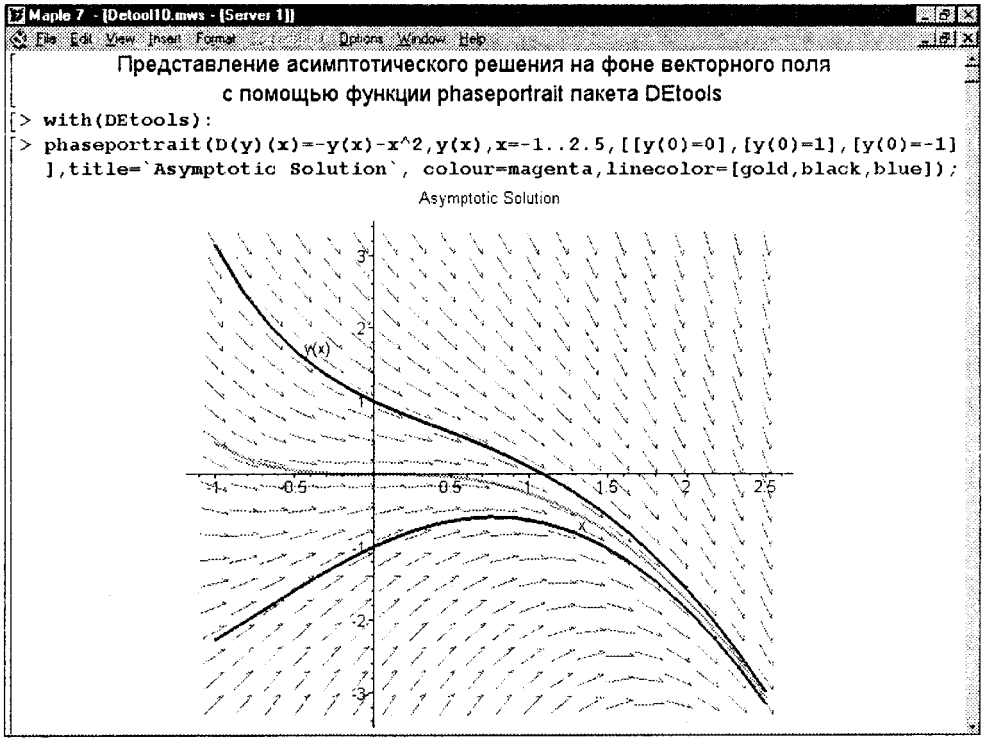


Рис. 13.16. Построение асимптотического решения на фоне графика векторного поля

Углубленный анализ дифференциальных уравнений

Задачи углубленного анализа ДУ

Maple 7 существенно доработана по части решения дифференциальных уравнений (ДУ) и систем с ДУ. Эта доработка прежде всего направлена на получение верных решений как можно большего числа ДУ разных классов и систем с ДУ.

В частности, расширен круг нелинейных дифференциальных уравнений, для которых Maple 7 способна дать аналитические решения.

Весь арсенал средств решения ДУ и методика их применения вполне заслуживают отражения в отдельной большой книге. Мы ограничимся описанием только трех средств системы Maple 7 — проверки ДУ на автономность, углубленным анализом решения с помощью контроля уровня выхода и получением приближенного полиномиального аналитического решения. Более подробное знакомство с новыми возможностями решения дифференциальных уравнений можно получить из соответствующей статьи справки `symbolics` в разделе `What is new...`.

Проверка ДУ на автономность

Одиночное дифференциальное уравнение или система дифференциальных уравнений называются автономными, если их правая часть явно не зависит от независимой переменной. Для автономных дифференциальных уравнений или систем при построении графиков решений функцией `DEplot` не обязательно задавать начальные условия, но нужно указывать диапазон изменения искомых переменных.

Для проверки уравнений (или систем) на автономность используется функция:

```
autonomous(des,vars,ivar)
```

где `des` — заданное дифференциальное уравнение или (в виде списка) система дифференциальных уравнений, `vars` — зависимые переменные и `ivar` — независимая переменная. Если система автономна, то эта функция возвращает `true`, в противном случае `false`.

Примеры:

```
> dif1:=diff(x(t),t)=x(t)*(1-y(t)); dif2:=diff(y(t),t)=.3*y(t)*(x(t)-1);
```

$$dif1 := \frac{\partial}{\partial t} x(t) = x(t)\,(1 - y(t))$$

$$dif2 := \frac{\partial}{\partial t} y(t) = .3\; y(t)\,(x(t) - 1)$$

```
> autonomous({dif1,dif2},[x(t),y(t)],t);
```

true

```
> autonomous(diff(x(t),t)=sin(t),x,t);
```

false

В первом случае система дифференциальных уравнений (модель Лотки–Вольтерра) автономна, а во втором случае дифференциальное уравнение не автономно.

Контроль уровня вывода решения ДУ

Для углубленного анализа аналитического решения ДУ (или системы ДУ) можно использовать специальную возможность управления уровнем вывода ре-

шения с помощью системной переменной `infilevel(dsolve):=level`. Значение `level=all` дает обычный вывод решения без комментариев, уровень 1 зарезервирован для информации, которую может сообщить пользователь, уровень 2 или 3 дает более детальный вывод (включая сообщения об использованном алгоритме и технике решения) и, наконец, уровни 4 и 5 дают наиболее детальную информацию (если таковая есть в дополнение к той информации, которую дает уровень 2 или 3).

Приведем пример аналитического решения ДУ третьего порядка с контролем уровня вывода решения:

```
> myDE := x^2*diff(y(x),x,x,x)-2*(n+1)*x*diff(y(x),x,x)+(a*x^2+6*n)*
                                                    *diff(y(x),x)-2*a*x*y(x)=0;
```

$$myDE := x^2\left(\frac{\partial^3}{\partial x^3}y(x)\right) - 2\,(n+1)\,x\left(\frac{\partial^2}{\partial x^2}y(x)\right) + (a\,x^2+6\,n)\left(\frac{\partial}{\partial x}y(x)\right) - 2\,a\,x\,y(x) = 0$$

```
> infolevel[dsolve] := all:

> dsolve(myDE);
```

$$y(x) = \_C1\,x^{(1/2+n)}\,\mathrm{BesselY}\left(-n-\frac{1}{2},\sqrt{a}\,x\right) + \_C2\,(4\,n-2+a\,x^2) + \_C3\,x^{(1/2+n)}\,\mathrm{BesselJ}\left(-n-\frac{1}{2},\sqrt{a}\,x\right)$$

```
> infolevel[dsolve] := 1:

> dsolve(myDE);
```

No Liouvillian solutions exists

$$y(x) = \_C1\,x^{(1/2+n)}\,\mathrm{BesselY}\left(-n-\frac{1}{2},\sqrt{a}\,x\right) + \_C2\,x^{(1/2+n)}\,\mathrm{BesselJ}\left(-n-\frac{1}{2},\sqrt{a}\,x\right) + \_C3\,(4\,n-2+a\,x^2)$$

```
> infolevel[dsolve] := 2:
> dsolve(myDE);
```

Methods for third order ODEs:
Trying to isolate the derivative d^3y/dx^3...
Successful isolation of d^3y/dx^3
--- Trying classification methods ---
trying a quadrature
checking if the LODE has constant coefficients
checking if the LODE is of Euler type
trying high order exact linear fully integrable
trying to convert to a linear ODE with constant coefficients
Equation is the LCLM of -2*x/(2*(2*n-1)/a+x^2)*y(x)+diff(y(x),x), a*y(x)-2*n/x*diff(y(x),x)+diff(diff(y(x),x),x)
checking if the LODE is of Euler type
checking if the LODE is of Euler type
exponential solutions successful

```
"Partial success" case. Entering dsolve with a lower order ODE
trying a quadrature
checking if the LODE has constant coefficients
checking if the LODE is of Euler type
trying a symmetry of the form [xi=0, eta=F(x)]
checking if the LODE is missing 'y'
trying a Liouvillian solution using Kovacic's algorithm
No Liouvillian solutions exists
trying a solution in terms of special functions:
-> Bessel
<- Bessel successful
<- solution in terms of special functions successful
```

$$y(x) = \_C1\, x^{(1/2+n)}\, \mathrm{BesselJ}\left(-n-\frac{1}{2}, \sqrt{a}\, x\right) + \_C2\, x^{(1/2+n)}\, \mathrm{BesselY}\left(-n-\frac{1}{2}, \sqrt{a}\, x\right) + \_C3\,(4n-2+a\,x^2)$$

В данном случае повышение уровня вывода до 4 или 5 бесполезно, поскольку вся информация о решении сообщается уже при уровне 2 (или 3).

Приближенное полиномиальное решение ДУ

Во многих случаях аналитические решения даже простых ДУ оказываются весьма сложными, например содержат специальные математические функции. При этом нередко полезна подмена такого решения другим, тоже аналитическим, но приближенным решением. Наиболее распространенным приближенным решением в этом случае может быть полиномиальное решение, то есть замена реального решения полиномом той или иной степени. При этом порядок полинома задается значением системной переменной `Order`, а для получения такого решения функция `dsolve` должна иметь параметр `series`.

На рис. 13.17 представлено решение ДУ третьего порядка различными методами: точное аналитическое и приближенное в виде полинома с максимальным заданным порядком 10 и 60. График дает сравнение этих решений для зависимости $y(t)$.

Дадим небольшой комментарий. Нетрудно заметить, что точное аналитическое решение весьма сложно и содержит специальные функции Бесселя и гамма-функции. При порядке полинома 8 (он несколько меньше заданного максимального) решение практически совпадает с точным до значений $t < 2$, а при максимальном заданном порядке 60 область совпадения расширяется до значений $t < 5{,}5$. Затем приближенное решение резко отходит от точного.

Этот пример, с одной стороны, иллюстрирует хорошо известный факт — быстрое нарастание погрешности полиномиального приближения за пределами области хорошего совпадения решений. С другой стороны, он показывает, что сте-

пень полинома более 60 (и даже выше) вовсе не так уж бесполезна, как это утверждается во многих статьях и книгах по полиномиальному приближению. Точность полиномиальных вычислений Maple 7 достаточно высока, чтобы обеспечить получение приближенных полиномиальных выражений со степенью порядка десятков и иногда даже сотен. Другое дело, что столь «длинный» полином не всегда удобен для аналитических расчетов, даже несмотря на его структурную простоту.

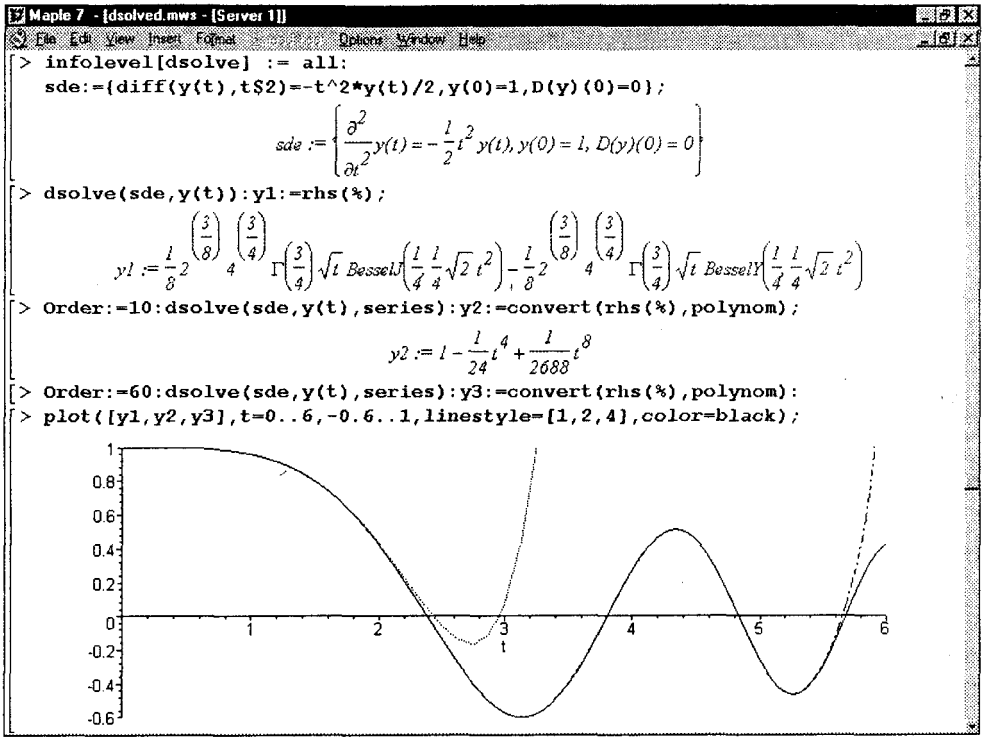


Рис. 13.17. Примеры решения ДУ третьего порядка

Что нового мы узнали?

В этом уроке мы научились:

- Использовать основную функцию решения дифференциальных уравнений dsolve.
- Решать дифференциальные уравнения первого порядка.
- Решать дифференциальные уравнения второго порядка.

- Решать системы дифференциальных уравнений.
- Выполнять численное решение дифференциальных уравнений.
- Решать дифференциальные уравнения с кусочными функциями.
- Использовать структуру неявного представления дифференциальных уравнений `DESol`.
- Применять инструментальный пакет решения дифференциальных уравнений DEtools.
- Осуществлять графическое представление решений дифференциальных уравнений.
- Осуществлять углубленный анализ аналитических решений дифференциальных уравнений.

Математические пакеты

- Назначение пакетов и обращение к ним
- Пакеты функций комбинаторики
- Пакет финансово-экономических функций finance
- Пакет ортогональных многочленов orthpoly
- Пакет для работы с суммами sumtools
- Пакет реализации степенных разложений powerseries
- Пакет числовой аппроксимации numapprox
- Пакет интегральных преобразований inttrans
- Пакет приближения кривых CurveFitting
- Пакет для работы с полиномами PolynomialTools

Назначение пакетов расширения и обращение к ним

Как уже отмечалось, некоторые функции системы Maple помимо их нахождения в ядре могут быть расположены в стандартной библиотеке и в пакетах, входящих в поставку системы. Это значит, что их не надо приобретать дополнительно, однако перед использованием таких функций надо загрузить их или отдельно, или вместе с целым пакетом, если большинство его функций представляет интерес для пользователя.

Обзор пакетов

В этом уроке дается выборочная информация о функциях, содержащихся в пакетах. Напоминаем, что список пакетов можно получить, используя команду:

```
>?packages
```

Этот список приведен ниже:

- DEtools — решение дифференциальных уравнений;
- Domains — создание областей определений в вычислениях;
- GF — поля Галуа;
- GaussInt — работа с целыми числами Гаусса;
- Groebner — вычисления в базисе Гробнера;
- LREtools — манипуляции с линейными рекуррентными отношениями;
- LinearAlgebra — линейная алгебра;
- Matlab — интеграция с MATLAB;
- Ore_algebra — основные вычисления в алгебре линейных операторов;
- PDEtools — решение дифференциальных уравнений в частных производных;
- Spread — работа с таблицами;
- algcurves — работа с алгебраическими кривыми;
- codegen — генерация кодов;
- combinat — функции комбинаторики;
- combstruct — структуры комбинаторики;
- context — контекстно-зависимые меню;
- diffalg — дифференциальная алгебра;

- difforms — дифференциальные формы;
- finance — финансовые расчеты;
- genfunc — рациональные функции;
- geom3d — трехмерная геометрия Евклида;
- geometry — евклидова геометрия;
- group — представление бесконечных групп;
- inttrans — интегральные преобразования;
- liesymm — симметрия Ли;
- linalg — линейная алгебра и структуры данных массивов;
- networks — графы;
- numapprox — численная аппроксимация;
- numtheory — теория чисел;
- orthopoly — ортогональные полиномы;
- padic — Пи-адические числа;
- plots — расширения графики;
- plottools — создание дополнительных графических объектов;
- polytools — действия с полиномами;
- powseries — формальные степенные ряды;
- process — мультипроцессы (для операционной системы Unix);
- simplex — линейная оптимизация (симплекс-метод);
- stats — статистика;
- student — функции в помощь студентам;
- sumtools — определенные и неопределенные суммы;
- tensor — тензоры и теория относительности.

Как следует из просмотра этого обширного списка, пакеты Maple 7 охватывают многие крупные разделы математики и существенно дополняют возможности системы, предоставляемые средствами ее ядра. Пакеты расширения пишутся на Maple-языке программирования, поэтому они могут легко модернизироваться и пополняться. Этим, в частности, объясняется тот факт, что набор пакетов расширения в Maple 7 существенно пополнен по сравнению с предшествующими реализациями системы.

Новые пакеты Maple 7

Система Maple 7 пополнилась рядом новых пакетов:

- CurveFutting — приближение кривых;
- ExternalCalling — внешние вычисления;

- LinearFunctionalSystem — линейные функциональные системы;
- MathML — поддержка средств языка MathML 2.0;
- OrthogonalSeries — серии с ортогональными полиномами;
- PolynomialTools — работа с полиномами.

Из этих пакетов надо особо выделить пакет приближения кривых. Он содержит наиболее важные функции для приближения кривых, которые до сих пор были разбросаны по ряду пакетов. В конце данного урока содержится описание пакета CurveFitting. Там же имеется описание другого нового и полезного пакета — PolynomialTools.

Получение информации о конкретном пакете

С помощью команды:

```
>?name_package;
```

можно получить информацию о любом пакете расширения и найти список входящих в него функций. Названия пакетов были приведены выше.

Для обращения к функциям того или иного пакета используется его полная загрузка командой:

```
>with(package):[;]
```

Знак : блокирует вывод списка функций пакета, а знак ; указывает вывести этот список.

Если вам необходима какая-то одна функция пакета или небольшая их часть, то не стоит загружать пакет целиком. Это может привести к избыточным затратам памяти компьютера и даже нарушить нормальную работу некоторых функций — следует помнить, что нередко пакеты переопределяют некоторые функции ядра. Для загрузки избранных функций используется команда `with` в форме

```
>with(package, f1, f2, …):
```

или

```
>with(package, [f1, f2, …]):
```

При этом загружаются функции `f1`, `f2`, ... из пакета с именем `packages`.

Может показаться, что было бы лучше иметь все функции в ядре. Однако создание ядра, реализующего все функции системы (в версии Maple 7 их около 3000), неразумно. Такое ядро занимало бы много места в памяти, имело большое время загрузки и затрудняло бы поиск конкретных нужных функций.

Поэтому ядро Maple 7 содержит определенный (и довольно обширный) минимум хорошо апробированных функций, а большинство других функций размещается в стандартной библиотеке и пакетах. Они готовятся на Maple-языке программирования и могут легко модернизироваться. К тому же пакеты могут модифицироваться (что не очень желательно) или дополняться (что приветствуется) пользователями. Некоторой платой за это является необходимость вызова того или иного пакета или функции перед их применением.

В этом разделе описана структура пакетов Maple 7, имеющих математическую направленность. Ограниченный объем книги и огромное число функций в пакетах не позволяют остановиться даже на описании синтаксических правил применения всех функций этих пакетов. Очевидно, что в этом нет и особого смысла — подавляющее большинство функций представляет малый интерес для конкретного пользователя. Те же, кто ими интересуются, могут легко восполнить пробелы в их описании с помощью справочной системы.

Однако в описании состава каждого пакета в данном уроке упомянуты имена всех без исключения его функций. Это позволяет оценить полноту того или иного пакета и без труда вызвать справочные страницы для любой функции. Описание функций пакетов в уроке дано выборочно, при этом предпочтение отдавалось тем функциям, которые используются в массовых математических и научно-технических расчетах и представляют интерес для достаточно широкого круга читателей.

Полезно отметить, что большинство функций имеет вполне понятные имена, отражающие их суть и назначение. К примеру, назначение функций `animate` или даже `textplot` в пакете `plots` или `Diff`, `Int` и `Limit` в пакете `student` понятны, пожалуй, всем. Но и в гораздо более специализированных пакетах имена входящих в них функций в значительной мере знакомы специалистам, поскольку почти всегда ассоциируются с общепринятыми названиями тех или иных специализированных функций или с их комбинациями.

Пакеты функций комбинаторики

Пакет combinat

Функции комбинаторики достаточно известны из обычного курса математики. При вызове пакета выводится (если вывод не заблокирован двоеточием) список его функций:

```
> with(combinat);
```

Warning, the protected name Chi has been redefined and unprotected

[Chi, *bell*, binomial, *cartprod*, *character*, *choose*, *composition*, *conjpart*, *decodepart*, *encodepart*, *fibonacci*, *firstpart*, *graycode*, *inttovec*, *lastpart*, *multinomial*, *nextpart*, *numbcomb*, *numbcomp*, *numbpart*, *numbperm*, *partition*, *permute*, *powerset*, *prevpart*, *randcomb*, *randpart*, *randperm*, *stirling1*, *stirling2*, *subsets*, *vectoint*]

Ввиду важности функций комбинаторики приведем их подные определения:

- `Chi(x)` — гиперболический косинусный интеграл;
- `bell(n)` — возвращает число `exp(exp(x)-1) = sum(bell(n)/n!*x^n, n=0..infinity)`, причем для вычислений используется рекуррентное соотношение `bell(n+1) = (bell(n)+1)^n`;

- `binomial(n, r)` — возвращает биноминальные коэффициенты, причем, если `n` и `r` — целые числа, удовлетворяющие условию `0 <= r<= n`, то функция возвращает `C(n,r)=n!/(r!(n-r)!)`, а в общем случае `C(n, r) = limit(GAMMA(N+1)/GAMMA(R+1)/GAMMA(N-R+1),R=r,N=n)`;
- `composition(n, k)` — возвращает списки композиций для целых неотрицательных `n` и `k`;
- `fibonacci(n)` — возвращает числа Фибоначчи, вычисляемые по рекуррентной формуле $F(n) = F(n-1) + F(n-2)$, где $F(0) = 0$ и $F(1) = 1$;
- `fibonacci(n, x)` — возвращает значение полинома Фибоначчи $F(n, x) = x\,F(n-1, x) + F(n-2, x)$, где $F(0, x) = 0$ и $F(1, x) = 1$, при этом $F(n) = F(n, 1)$;
- `firstpart(n)` — возвращает первую часть канонической последовательности ряда;
- `nextpart(l)` — возвращает следующую часть канонической последовательности ряда;
- `lastpart(n)` — возвращает последнюю часть канонической последовательности ряда;
- `prevpart(l)` — возвращает предыдущую часть канонической последовательности ряда;
- `conjpart(l)` — возвращает объединенный раздел в канонической последовательности ряда;
- `graycode(n)` — возвращает список кодов Грея для n-битовых чисел;
- `multinomial(n, k1, k2, ..., km)` — возвращает мультиномиальные коэффициенты;
- `numbcomb(n)` и `numbcomb(n, m)` — возвращает число комбинаций;
- `numbcomp(n, k)` — возвращает число композиций;
- `numbpart(n)` — возвращает список всех возможных сумм, дающих `n`;
- `permute(n)` и `permute(n, r)` — возвращает `numbperm(n, r) = nops(permute(n, r))`;
- `powerset(s)` — возвращает степень множества в множестве `s`;
- `randcomb(n, m)` — возвращает случайную комбинацию;
- `randpart(n)` — возвращает случайную часть;
- `randperm(n)` — возвращает случайную композицию;
- `stirling1(n, m)` — возвращает число Стирлинга первого рода;
- `stirling2(n, m)` — возвращает число Стирлинга второго рода;
- `subsets(L)` — задает итерационную процедуру над степенями множества или списка `L`;
- `vectoint(l)` — возвращает индекс вектора канонического упорядочения `l`;
- `inttovec(m, n)` — возвращает вектор канонического упорядочения для неотрицательных целых чисел `m` и `n`.

Ниже даны примеры применения некоторых из этих функций:

```
> choose(4,3);
[[1, 2, 3], [1, 2, 4], [1, 3, 4], [2, 3, 4]]
> choose([a,a,b,c],3);
[[a, a, b], [a, a, c], [a, b, c]]
> composition(3,2);
{[2, 1], [1, 2]}
> decodepart(4,2);
[1, 1, 2]
> fibonacci(10);
55
> seq(fibonacci(i),i=1..12);
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144
> partition(5);
[[1, 1, 1, 1, 1], [1, 1, 1, 2], [1, 2, 2], [1, 1, 3], [2, 3], [1, 4], [5]]
> firstpart(3);
[1, 1, 1]
> nextpart(%);
[1, 2]
> prevpart(%);
[1, 1, 1]
> lastpart(3);
[3]
> conjpart(%);
[1, 1, 1]
> multinomial(8,2,3,3);
560
> numbcomp(8,5);
35
> numpart(3);
numpart(3)
> numbperm(4);
24
> numbperm([a,b]);
2
> numbperm({a,b,c},2);
6
> permute(3,2);
[[1, 2], [1, 3], [2, 1], [2, 3], [3, 1], [3, 2]]
> permute([a,a,b],2);
[[a, a], [a, b], [b, a]]
> powerset([a,a,b]);
[[ ], [a], [b], [a, b], [a, a], [a, a, b]]
```

```
> randcomb([a,b,c,d],3);
```
$[a, c, d]$
```
> randcomb([a,b,c,d],3);
```
$[a, b, d]$
```
> randpart(10);
```
$[2, 8]$
```
> randpart(10);
```
$[10]$
```
> stirling1(10,5);
```
-269325
```
> stirling2(10,5);
```
42525
```
> S:=subsets({1,2}):
> while not S[finished] do S[nextvalue]() od;
```
$\{\}$

$\{1\}$

$\{2\}$

$\{1, 2\}$
```
> vectoint([1,0,0]);
```
1
```
> inttovec(6,3);
```
$[1, 0, 1]$

Читателю, желающему использовать данный пакет, рекомендуется внимательно ознакомиться с этими простыми примерами и просмотреть примеры из справочной базы данных для имеющихся в пакете функций.

Пакет combstruct

Еще девять функций, относящихся к структурам комбинаторики, содержит пакет combstruct:

```
> with(combstruct);
```

[*allstructs*, *count*, *draw*, *finished*, *gfeqns*, *gfseries*, *gfsolve*, *iterstructs*, *nextstruct*]

Эти функции служат для создания случайно однородных объектов, принадлежащих заданному комбинаторному классу. Ограничимся приведением примеров применения этих функций:

```
> allstructs(Subset({one,two}));
```
$\{\{\}, \{one, two\}, \{two\}, \{one\}\}$
```
> allstructs(Permutation([x,y,z]),size=2);
```
$[[x, y], [x, z], [y, x], [y, z], [z, x], [z, y]]$
```
> count(Subset({1,2,3}));
```
8
```
> draw(Combination(5),size=4);
```
$\{1, 3, 4, 5\}$

```
> count(Permutation([a,a,b]));
```
3
```
> it :=iterstructs(Permutation([a,a,b]),size=2);
```
it := table([*finished* = *false*, *nextvalue* = (proc() ... end proc)])
```
> draw(Partition(9));
```
[2, 2, 2, 3]
```
> allstructs(Composition(3), size=2);
```
[[2, 1], [1, 2]]

Для более полного знакомства с этими специфическими функциями обратитесь к справочной системе.

Пакет финансово-экономических функций finance

Пакет финансово-экономических расчетов открывается командой:

```
> with(finance);
```

[*amortization*, *annuity*, *blackscholes*, *cashflows*, *effectiverate*, *futurevalue*, *growingannuity*, *growingperpetuity*, *levelcoupon*, *perpetuity*, *presentvalue*, *yieldtomaturity*]

Этот пакет представлен рядом указанных выше функций в двух формах:

```
function(args)
finance[function](args)
```

Благодаря правилам задания аргументов можно реализовать практически все известные финансово-экономические расчеты, такие как амортизация, накопления и платежи по вкладам и т. д. В свете задач рыночной экономики эти функции полезны для приверженцев решения всего на свете без выхода из оболочки Maple. Все же надо отметить, что малозаметные тонкости в определении финансово-экономических функций затрудняют их применение. Есть множество специальных финансово-экономических пакетов, например «Бухгалтерия 1С», которые лучше подходят для наших экономических реалий, чем «заумный» и прозападный Maple 7.

Полный перечень функций можно найти в справке по этому пакету. Ограничимся упоминанием нескольких наиболее характерных функций, связанных с использованием вкладов:

- `annuity(cash,rate,nperiods)` — вычисление суммы, находящейся на вкладе с начальным значением `cash`, процентом начисления `rate` и числом периодов `nperiods`;
- `cashflows(flows,rate)` — вычисление общей суммы вклада по списку вложений `flows` и проценту обесценивания денег `rate`;

- futurevalue(amount, rate, nperiods) — вычисление будущего значения вклада при начальном вложении amount, проценте начисления rate и числе периодов nperiods;
- presentvalue(amount, rate, nperiods) — вычисление начального вклада для получения суммы в amount при проценте начислений rate и числе вкладов nperiods.

Примеры применения этих функций даны ниже:

```
> annuity(100, .10, 15);
```

760.6079506

```
> annuity(cash,rate,nperiods);
```

$$cash\left(\frac{1}{rate}-\frac{1}{rate\,(rate+1)^{nperiods}}\right)$$

```
> cashflows([1000,800,700,600],-0.05);
```

3492.146316

```
> s:=1000+800+700+600;
```

$s := 3100$

```
> futurevalue(1000,0.1,5);
```

1610.51000

```
> presentvalue(1610,0.1,5);
```

999.6833301

Поскольку формулы и обозначения в финансово-экономических расчетах в различной литературе порою заметно различаются (особенно сильны различия между нашей и западной литературой), это может создать серьезные ошибки при вычислениях. К примеру, в формулах Maple на самом деле используются не проценты начислений или обесценивания вкладов, а соответствующие им относительные единицы, например 10% соответствует 0,1. В нашей литературе проценты обычно задаются в явном виде, то есть rate = 10 при 10%. Надо следить и за знаком величины rate, поскольку она может трактоваться как процент начислений или процент обесценивания денег по вкладам, что соответствует различным ее знакам.

Расчеты такого рода для Maple 7 относятся к достаточно простым, так что даже начинающий пользователь может составить свои функции для таких расчетов по вполне понятным ему и апробированным формулам. Надо отметить, однако, что, используя символьное задание параметров функций, легко получить вывод именно тех формул, которые использует система Maple, и сравнить их со своими формулами. В случае совпадения применение функций Maple возможно и предпочтительно.

ПРИМЕЧАНИЕ В целом применение Maple 7 как системы с символьной и точной арифметикой весьма предпочтительно в финансово-экономических и статистических расчетах, поскольку обеспечивает принципиально повышенную точность и устойчивость таких расчетов.

Пакет ортогональных многочленов orthopoly

Ортогональные многочлены (полиномы) находят самое широкое применение в различных математических расчетах. В частности, они широко используются в алгоритмах интерполяции, экстраполяции и аппроксимации различных функциональных зависимостей. В пакете `orthopoly` задано 6 функций:

```
> with(orthopoly):
```

$[G, H, L, P, T, U]$

Однобуквенные имена этих функций отождествляются с первой буквой в наименовании ортогональных полиномов. Вопреки принятым в Maple 7 правилам, большие буквы в названиях этих полиномов не указывают на инертность данных функций — все они являются немедленно вычисляемыми. В данном разделе функции этого пакета будут полностью описаны.

Отметим определения указанных функций:

- `G(n,a,x)` — полином Гегенбауэра (из семейства ультрасферических полиномов);
- `H(n,x)` — полином Эрмита;
- `L(n,x)` — полином Лагерра;
- `L(n,a,x)` — обобщенный полином Лагерра;
- `P(n,x)` — полином Лежандра;
- `P(n,a,b,x)` — полином Якоби;
- `T(n,x)` — обобщенный полином Чебышева первого рода;
- `U(n,x)` — обобщенный полином Чебышева второго рода.

Свойства ортогональных многочленов хорошо известны. Все они характеризуются целочисленным порядком `n`, аргументом `x` и иногда дополнительными параметрами `a` и `b`. Существуют простые рекуррентные формулы, позволяющие найти полином n-го порядка по значению полинома $(n-1)$-го порядка. Эти формулы и используются для вычисления полиномов высшего порядка.

Ниже представлены примеры вычисления ортогональных полиномов:

```
> G(0,1,x):
```

1

```
> G(1,1,x):
```

$2x$

```
> G(1,1,5):
```

10

```
> H(3,x):
```

$8x^3 - 12x$

```
> L(3,x):
```

$$1 - 3x + \frac{3}{2}x^2 - \frac{1}{6}x^3$$

```
> L(2,a,x);
```

$$1+\frac{3}{2}a-2x+\frac{1}{2}a^2-ax+\frac{1}{2}x^2$$

```
> P(2,x);
```

$$\frac{3}{2}x^2-\frac{1}{2}$$

```
> P(2,1,1,x);
```

$$\frac{15}{4}x^2-\frac{3}{4}$$

```
> T(5,x);
```

$$16x^5-20x^3+5x$$

```
> U(5,x);
```

$$32x^5-32x^3+6x$$

Представляет интерес построение графиков ортогональных многочленов. На рис. 14.1 построены графики ряда многочленов Гегенбауэра и Эрмита.

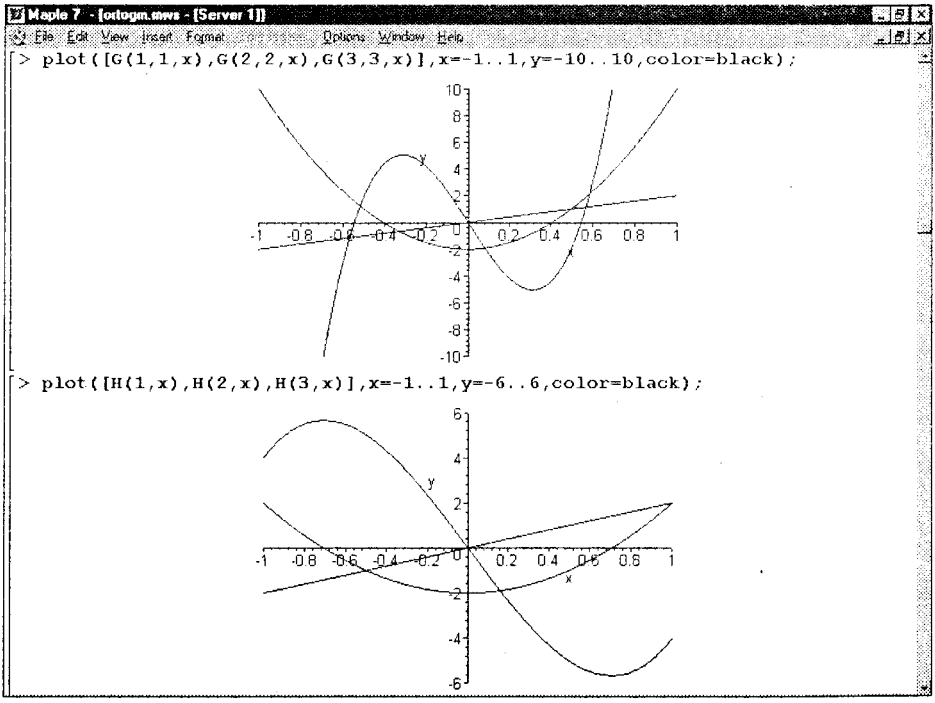


Рис. 14.1. Графики ортогональных многочленов Гегенбауэра и Эрмита

На рис. 14.2 построены графики ортогональных многочленов Лагерра и Лежандра.

Наконец, на рис. 14.3 даны графики ортогональных многочленов Чебышева $T(n, x)$ и $U(n, x)$.

Приведенные графики дают начальное представление о поведении ортогональных многочленов. К примеру, многочлены Чебышева имеют минимальное

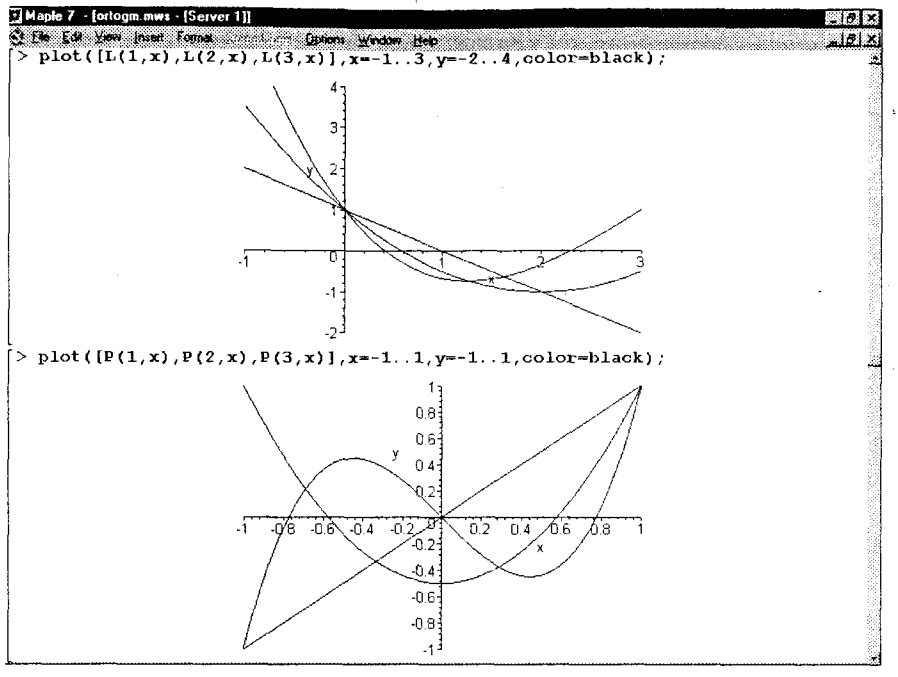


Рис. 14.2. Графики ортогональных многочленов Лагерра и Лежандра

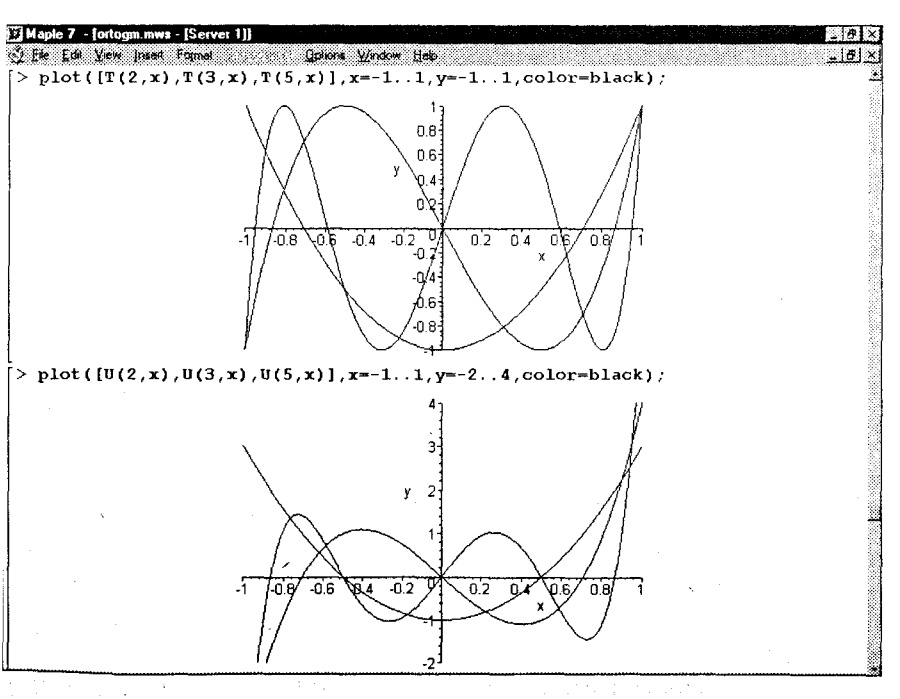


Рис. 14.3. Графики ортогональных многочленов Чебышева

отклонение от оси абсцисс в заданном интервале изменения x. Это их свойство объясняет полезное применение таких многочленов при решении задач аппроксимации функций. Можно порекомендовать читателю по их образу и подобию построить графики ортогональных многочленов при других значениях параметра n и диапазонах изменения аргумента x.

В отличие от ряда элементарных функций ортогональные многочлены определены только для действительного аргумента x. При комплексном аргументе просто повторяется исходное выражение с многочленом:

```
> evalf(U(2,2+3*I));
```

$$P(2, 2 + 3\,I)$$

```
> evalf(sqrt(2+3*I));
```

$$1.674149228 + .8959774761\,I$$

Ортогональные многочлены не определены также и для дробного показателя n. Впрочем, надо отметить, что такие многочлены на практике используются крайне редко.

Пакет для работы с суммами sumtools

Состав пакета sumtools

Этот инструментальный пакет предназначен для работы со специальными суммами. Он содержит указанные ниже функции:

```
> with(sumtools);
```

[*Hypersum, Sumtohyper, extended_gosper, gosper, hyperrecursion, hypersum, hyperterm, simpcomb, sumrecursion, sumtohyper*]

Назначение функций данного пакета перечислено ниже:

- `hypersum(U, L, z, n)` и `Hypersum(U, L, z, n)` — вычисление гиперсумм;
- `sumtohyper(f, k)` и `Sumtohyper(f, k)` — преобразование сумм в гиперсуммы;
- `extended_gosper(f, k)`, `extended_gosper(f, k=m..n)` и `extended_gosper(f, k, j)` — реализация расширенного алгоритма Госпера;
- `gosper(f, k)` и `gosper(f, k=m..n)` — реализация алгоритма Госпера;
- `hyperrecursion(U, L, z, s(n))` — реализация гиперрекурсионного алгоритма;
- `hyperterm(U, L, z, k)` и `Hyperterm(U, L, z, k)` — ввод гипергеометрического терма.

Работа с пакетом sumtools

Приведем примеры на применение этих функций:

```
> extended_gosper(k*(k/2)!,k);
```

$$2\left(\frac{1}{2}k\right)! + 2\left(\frac{1}{2}k + \frac{1}{2}\right)!$$

```
> extended_gosper(k*(k/2)!,k,2);
```

$$2\left(\frac{1}{2}k\right)!$$

```
> extended_gosper(k*(k/2)!,k=1..n);
```

$$2\left(\frac{1}{2}n+\frac{1}{2}\right)!+2\left(\frac{1}{2}n+1\right)!-2\left(\frac{1}{2}\right)!-2$$

```
> gosper(k*(k/2)!,k);
```

FAIL

```
> gosper(pochhammer(k,n),k);
```

$$\frac{(k-1)\ \mathrm{pochhammer}(k,n)}{n+1}$$

```
> hyperrecursion([-n,a],[b],1,f(n));
```

$$(-n+a-b+1)\,f(n-1)+(n+b-1)\,f(n)$$

```
> Hypersum([a,1+a/2,b,c,d,1+2*a-b-c-d+n,-n],
  [a/2,1+a-b,1+a-c,1+a-d,1+a-(1+2*a-b-c-d+n),1+a+n],1,n);
>
```

$$\mathrm{Hyperterm}([1,\ 1+a,\ a-d-c+1,\ a+1-d-b,\ a-c+1-b],$$
$$[1+a-d,\ 1+a-c,\ 1+a-b,\ a-b-c-d+1],\ 1,\ n)$$

```
> simpcomb(binomial(n,k));
```

$$\frac{\Gamma(n+1)}{\Gamma(n-k+1)\,\Gamma(k+1)}$$

```
> sumrecursion(binomial(n,k)^3,k,f(n));
```

$$-8\,(n-1)^2\,f(n-2)-(7\,n^2-7\,n+2)\,f(n-1)+f(n)\,n^2$$

```
> hyperterm([a,b],[c],z,k);
```

$$\frac{\mathrm{pochhammer}(a,k)\ \mathrm{pochhammer}(b,k)\ z^k}{\mathrm{pochhammer}(c,k)\ k!}$$

Из этих примеров применение функций данного пакета достаточно очевидно.

Пакет реализации степенных разложений powseries

Состав пакета powseries

Степенные разложения часто используются в математических расчетах для приближенного представления разнообразных функций и обеспечения единообразия такого представления. В пакете powseries сосредоточены расширенные средства по реализации таких разложений. Они представлены 22 функциями:

```
> with(powseries);
```

[*compose, evalpow, inverse, multconst, multiply, negative, powadd, powcos, powcreate, powdiff, powexp, powint, powlog, powpoly, powsin, powsolve, powsqrt, quotient, reversion, subtract, template, tpsform*]

Ниже представлено определение этих функций:

- `compose(a,b)` — объединяет ряды `a` и `b`;
- `evalpow(expr)` — вычисляет выражение `expr` и возвращает его в виде ряда;
- `inverse(p)` — инвертирует ряд `p`;
- `multconst(p,const)` — умножает ряд `p` на константу `const`;
- `multiply(a,b)` — умножает ряд `a` на ряд `b`;
- `negative(p)` — возвращает аддитивный, обратный по отношению к `p` ряд;
- `powadd(a,b,...)` — складывает ряды `a`, `b`, ...;
- `powcreate(expr)` — создает ряд для выражения `expr`;
- `powpoly(pol,var)` — создает ряд для полинома `pol` по переменной `var`;
- `powsolve(sys)` — создает ряд для решения дифференциальных уравнений `sys`;
- `quotient(a,b)` — возвращает частное для `a` и `b` в виде ряда;
- `reversion(a)` — дает обратное к композиции разложение ряда `a`;
- `subtract(a,b)` — дает разность рядов `a` и `b`.

В выражении `expr` могут использоваться операторы `+`, `-`, `*`, `/` и `^`. С ними могут комбинироваться встроенные функции и функции пользователя, например $f(g)$. Кроме того, могут использоваться следующие функции:

Powexp	powinv	powlog	powneg	powrev
Powdiff	powint	powquo	powsub	powcos
Powtan	powsec	powcsc	powcot	powsinh
Powcosh	powtanh	powsech	powcsch	powcoth
Powsqrt	powadd	multiply		

Примеры применения пакета powseries

Назначение большинства этих функций очевидно из их названий — они возвращают соответствующую функцию (указанную после слова `pow` в имени) в виде разложения в ряд или полинома. Например, `powexp` раскладывает выражения с экспоненциальными функциями в ряд.

Получаемые функциями ряды представляются в специальном формате. Поэтому для их применения в обычном виде необходимо использовать функцию `tpsform` в следующих видах:

- `tpsform(p, var, order)` — преобразует ряд `p` в обычную форму с заданием порядка `order`;
- `tpsform(p, var)` — преобразует ряд `p` в обычную форму с порядком, заданным переменной `Order`.

Здесь `p` — имя степенного ряда, `var` — переменная, относительно которой записан ряд, `order` — порядок ряда. Если параметр `order` не указан, используется зна-

чение глобальной переменной Order. Ниже даны примеры, иллюстрирующие технику работы со степенными разложениями:

```
> p1:=powexp(sin(x));
```

$$p1 := \text{proc}(powparm) \ \ldots \ \text{end proc}$$

```
> p2:=powexp(cos(x));
```

$$p2 := \text{proc}(powparm) \ \ldots \ \text{end proc}$$

```
> tpsform(p1,x);
```

$$1 + x + \frac{1}{2}x^2 - \frac{1}{8}x^4 - \frac{1}{15}x^5 + O(x^6)$$

```
> tpsform(p2,x);
```

$$e - \frac{1}{2}e\,x^2 + \frac{1}{6}e\,x^4 + O(x^6)$$

```
> a := powseries[powexp](x):
> b := powseries[tpsform](a, x, 5);
```

$$b := 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + O(x^5)$$

```
> c := powadd( powpoly(1+x^2+x,x), powlog(1+x) ):
> d := tpsform(c, x, 6);
```

$$d := 1 + 2x + \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4 + \frac{1}{5}x^5 + O(x^6)$$

Применение функций этого пакета достаточно просто и прозрачно, так что заинтересованный читатель может сам опробовать на примерах работу тех функций, которые не были использованы в приведенных примерах.

Пакет числовой аппроксимации numapprox

Состав пакета numapprox

Этот пакет содержит небольшое число безусловно очень важных функций:

```
> with(numapprox);
```

[*chebdeg*, *chebmult*, *chebpade*, *chebsort*, *chebyshev*, *confracform*, *hermite_pade*, *hornerform*, *infnorm*, *laurent*, *minimax*, *pade*, *remez*]

В их числе функции интерполяции и аппроксимации полиномами Чебышева, рядом Тейлора, отношением полиномов (Паде-аппроксимация) и др. Все они широко применяются не только в фундаментальной математике, но и при решении многих прикладных задач. Рассмотрим их, начиная с функций аппроксимации аналитических зависимостей.

Разложение функции в ряд Лорана

Для разложения функции f в ряд Лорана с порядком n в окрестности точки $x = a$ (или $x = 0$) служит функция laurent:

```
laurent(f, x=a, n)
laurent(f, x, n)
```

Представленный ниже пример иллюстрирует реализацию разложения в ряд Лорана:

```
> laurent(f(x),x=0,4);
```

$$f(0) + D(f)(0)\,x + \frac{1}{2}(D^{(2)})(f)(0)\,x^2 + \frac{1}{6}(D^{(3)})(f)(0)\,x^3 + O(x^4)$$

```
> laurent(exp(x),x,5);
```

$$1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + O(x^5)$$

Паде-аппроксимация аналитических функций

Для аппроксимации аналитических функций одной из лучших является Паде-аппроксимация, при которой заданная функция приближается отношением двух полиномов. Для осуществления такой аппроксимации используется функция pade:

```
pade(f, x=a, [m,n])
pade(f, x, [m,n])
```

Здесь f — аналитическое выражение или функция, x — переменная, относительно которой записывается аппроксимирующая функция, a — координата точки, относительно которой выполняется аппроксимация, m, n — максимальные степени полиномов числителя и знаменателя. Технику аппроксимации Паде поясняет рис. 14.4.

На рис. 14.4 представлена аппроксимация синусоидальной функции, а также построены графики этой функции и аппроксимирующей функции. Под ними дан также график абсолютной погрешности для этого вида аппроксимации. Нетрудно заметить, что уже в интервале $[-\pi, \pi]$ погрешность резко возрастает на концах интервала аппроксимации.

Важным достоинством Паде-аппроксимации является возможность довольно точного приближения разрывных функций. Это связано с тем, что нули знаменателя у аппроксимирующего выражения способны приближать разрывы функций, если на заданном интервале аппроксимации число разрывов конечно. На рис. 14.5 представлен пример Паде-аппроксимации функции $\tan(x)$ в интервале от −4,5 до 4,5, включающем два разрыва функции.

Как видно из рис. 14.5, расхождение между функцией тангенса и ее аппроксимирующей функцией едва заметно лишь на краях интервала аппроксимации. Оба разрыва прекрасно приближаются аппроксимирующей функцией. Такой характер аппроксимации подтверждается и графиком погрешности, которая лишь на концах интервала аппроксимации [−4,0, 4,0] достигает значений 0,01 (около 1%).

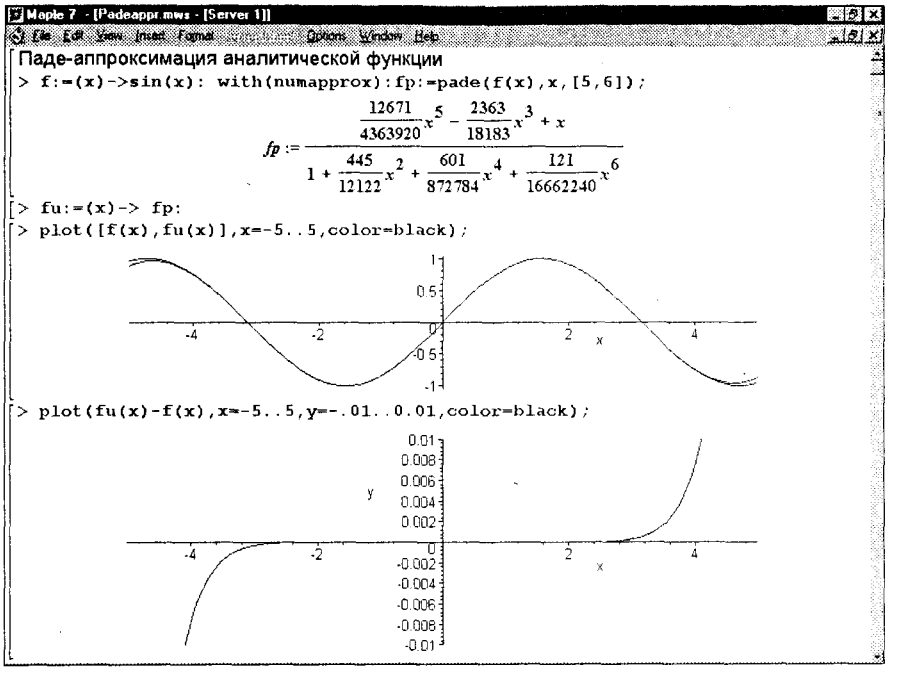


Рис. 14.4. Аппроксимация Паде для синусоидальной функции

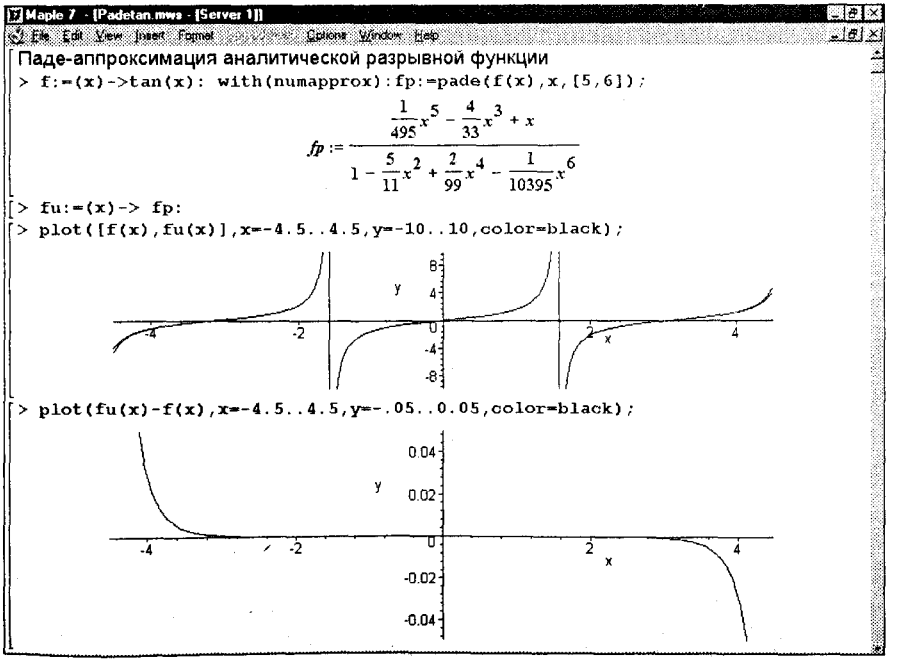


Рис. 14.5. Аппроксимация Паде для разрывной функции тангенса

Паде-аппроксимация с полиномами Чебышева

Для многих аналитических зависимостей хорошие результаты дает аппроксимация полиномами Чебышева. В общем случае применяется Паде-аппроксимация отношением таких полиномов. Она реализуется функциями chebpade:

```
chebpade(f, x=a..b, [m,n])
chebpade(f, x, [m,n])
chebpade(f, a..b, [m,n])
```

Здесь a..b задает отрезок аппроксимации, m и n — максимальные степени числителя и знаменателя полиномов Чебышева. Приведенный ниже пример показывает аппроксимацию Паде полиномами Чебышева для функции f=cos(x):

```
> Digits:=10:chebpade(cos(x),x=0..1,5);
```

$$.8235847380\,T(0, 2x-1) - .2322993716\,T(1, 2x-1) - .05371511462\,T(2, 2x-1) + .002458235267\,T(3, 2x-1) + .0002821190574\,T(4, 2x-1) - .7722229156\,10^{-5}\,T(5, 2x-1)$$

```
> chebpade(cos(x),x=0..1,[2,3]);
```

$$(.8162435876\,T(0, 2x-1) - .1852356296\,T(1, 2x-1) - .05170917481\,T(2, 2x-1))/(T(0, 2x-1) + .06067214549\,T(1, 2x-1) + .01097466398\,T(2, 2x-1) + .0005311640964\,T(3, 2x-1))$$

Наилучшая минимаксная аппроксимация

Минимаксная аппроксимация отличается от Паде-аппроксимации минимизацией максимальной абсолютной погрешности во всем интервале аппроксимации. Она использует алгоритм Ремеза (см. ниже) и реализуется следующей функцией:

```
minimax(f, x=a..b, [m,n], w, 'maxerror')
minimax(f, a..b, [m,n], w, 'maxerror')
```

Здесь помимо уже отмеченных параметров w — процедура или выражение, maxerror — переменная, которой приписывается значение minimax-нормы. Ниже дан пример аппроксимации функции cos(x) в интервале [–3, 3]:

```
> minimax(cos(x),x=-3..3,[2,3],1,'minmax');
```

$$\frac{.5458304182 + (.5119634586\,10^{-9} - .2308484266\,x)\,x}{.5217186403 + (.1496104420\,10^{-8} + .1062847466\,x)\,x}$$

```
> minmax;
```

$$.04621605601$$

Наилучшая минимаксная аппроксимация по алгоритму Ремеза

Для получения наилучшей полиномиальной аппроксимации используется алгоритм Ремеза, который реализует следующая функция:

```
remez(w, f, a, b, m, n, crit, 'maxerror')
```

Здесь w — процедура, представляющая функцию $w(x) > 0$ в интервале $[a, b]$, f — процедура, представляющая аппроксимируемую функцию $f(x)$, a и b — числа, задающие интервал аппроксимации $[a, b]$, m и n — степени числителя и знаменателя аппроксимирующей функции, crit — массив, индексированный от 1 до $m + n + 2$ и представляющий набор оценок в критических точках (то есть точек максимума/минимума кривых погрешности), maxerror — имя переменной, которой присваивается минимаксная норма w abs($f - r$).

Следующий пример иллюстрирует применение данной функции для аппроксимации функции erf(x):

```
> Digits:=12:w:=proc(x) 1.0 end;
```

$w := \mathbf{proc}(x)\ 1.0\ \mathbf{end\ proc}$

```
> f:=proc(x) evalf(erf(x)) end;
```

$f := \mathbf{proc}(x)\ \mathrm{evalf}(\mathrm{erf}(x))\ \mathbf{end\ proc}$

```
> crit:=array(1..7,[0,.1,.25,.5,.75,.9,1.]);
```

$crit := [0, .1, .25, .5, .75, .9, 1.]$

```
> remez(w,f,0,1,5,0,crit,'maxerror');
```

$$x \rightarrow .0000221268863 + (1.12678937620 + (.018447321509 + (-.453446232421 + (.141246775527 + .00966355213050\,x)\,x)\,x)\,x)\,x$$

```
> maxerror;
```

$.0000221268894463$

Другие функции пакета

Отметим назначение других функций пакета numapprox:

- chebdeg(p) — возвращает степень полинома Чебышева p;
- chebmult(p, q) — умножение полиномов Чебышева p и q;
- chebsort(e) — сортирует элементы ряда Чебышева;
- confracform(r) — преобразует рациональное выражение r в цепную дробь;
- confracform(r, x) — преобразует рациональное выражение r в цепную дробь с независимой переменной x;
- hornerform(r) — преобразует рациональное выражение r в форму Горнера;
- hornerform(r, x) — преобразует рациональное выражение r в форму Горнера с независимой переменной x;
- infnorm(f, x=a...b, 'xmax') — возвращает L-бесконечную норму функции на отрезке x $[a, b]$;
- infnorm(f, a...b, 'xmax') — возвращает L-бесконечную норму функции на отрезке $[a, b]$.

Действие этих функций очевидно, и читатель может самостоятельно опробовать их в работе.

Пакет интегральных преобразований inttrans

Общая характеристика пакета

Это один из пакетов, наиболее важных для общематематических и научно-технических приложений. Он содержит небольшой набор функций:

```
> with(inttrans);
```

[*addtable*, *fourier*, *fouriercos*, *fouriersin*, *hankel*, *hilbert*, *invfourier*, *invhilbert*, *invlaplace*, *invmellin*, *laplace*, *mellin*, *savetable*]

Однако эти функции охватывают такие практические важные области математики, как ряды Фурье, прямые и обратные преобразования Лапласа и Фурье и ряд других интегральных преобразований. Ниже они обсуждены более подробно.

В предшествующих реализациях системы Maple функции прямого и обратного Z-преобразований также входили в пакет `inttrans`, однако в Maple 6 и 7 они перенесены в ядро системы.

Прямое и обратное преобразования Лапласа

Прямое преобразование Лапласа заключается в переводе некоторой функции времени $f(t)$ в операторную форму $F(p)$. Это преобразование означает вычисление интеграла

$$F(p) = \int_0^{\infty} f(t)e^{-st}dt$$

Для осуществления прямого преобразования Лапласа Maple 7 имеет функцию

```
laplace(expr,t,p)
```

Здесь `expr` — преобразуемое выражение, `t` — переменная, относительно которой записано `expr`, и `p` — переменная, относительно которой записывается результат преобразования.

Обратное преобразование Лапласа означает переход от функции $F(p)$ к функции $f(t)$ с помощью формулы

$$F(t) = \int_{s-I\infty}^{s+I\infty} F(p)e^{-pt}dp$$

Для вычисления этого интеграла служит функция:

```
invlaplace(expr, p, t)
```

где `expr` — выражение относительно переменной `p`, `t` — переменная, относительно которой записывается результирующая зависимость. Оба преобразования широко применяются в практике научно-технических вычислений и отражают суть

операторного метода. При этом прямое преобразование создает *изображение*, а обратное — *оригинал* функции. Ниже приведены примеры применения прямого и обратного преобразований Лапласа:

```
> laplace(sin(t)+a*cos(t),t,p);
```

$$\frac{1}{p^2+1}+\frac{a\,p}{p^2+1}$$

```
> invlaplace(%,p,t);
```

$$\sin(t)+a\cos(t)$$

Нетрудно заметить, что в данном случае последовательное применение прямого, а затем обратного преобразования восстанавливает исходную функцию $\sin(t) + a\cos(t)$.

Прямое и обратное преобразования Фурье

Прямое преобразование Фурье преобразует функцию времени $f(t)$ в функцию частот и заключается в вычислении следующей интегральной функции:

$$F(w)=\int_{-\infty}^{\infty} f(t)e^{-jwt}dt$$

Оно реализуется следующей функцией пакета интегральных преобразований `inttrans`:

```
fourier(expr,t,w)
```

Здесь `expr` — выражение (уравнение или множество), `t` — переменная, от которой зависит `expr`, и `w` — переменная, относительно которой записывается результирующая функция.

Обратное преобразование Фурье задается вычислением интеграла:

$$f(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty} F(w)e^{jtw}dw.$$

Оно фактически переводит представление сигнала из частотной области во временную.

Примеры применения преобразования Фурье представлены ниже:

```
> fourier(sin(t),t,w);
```

$$-I\pi\,\mathrm{Dirac}(w-1)+I\pi\,\mathrm{Dirac}(w+1)$$

```
> invfourier(%,w,t);
```

$$\frac{1}{2}I(-\mathrm{e}^{(It)}+\mathrm{e}^{(-It)})$$

```
> simplify(%);
```

$$\sin(t)$$

```
> fourier(1-exp(-a*t),t,w);
```

$$2\pi\,\mathrm{Dirac}(w)-\mathrm{fourier}(\mathrm{e}^{(-at)},t,w)$$

```
> invfourier(%,w,t);
```

$$1-\mathrm{e}^{(-at)}$$

Обратите внимание на то, что даже в простом первом примере применение обратного преобразования Фурье вслед за прямым не привело к буквальному восстановлению исходной функции $sin(t)$. Потребовалась команда `simplify`, чтобы перевести результат в виде представления синуса через экспоненциальные функции к обычному виду $sin(t)$.

Вычисление косинусного и синусного интегралов Фурье

Разложение функции $f(t)$ в ряд Фурье требует вычисления интегралов следующего вида:

$$F(s)=\sqrt{\frac{2}{\pi}}\int_0^{\infty} f(t)\cos(st)dt$$

$$F(s)=\sqrt{\frac{2}{\pi}}\int_0^{\infty} f(t)\sin(st)dt$$

Они получили название косинусного и синусного интегралов Фурье и фактически задают вычисление коэффициентов ряда Фурье, в который может быть разложена функция $f(t)$.

Для вычисления этих интегралов в пакете используются следующие функции:

```
fouriercos(expr,t,s)
fouriersin(expr,t,s)
```

Поскольку формат задания этих функций вполне очевиден, ограничимся примерами их применения:

```
> fouriercos(5*t,t,s);
```

$$-5\frac{\sqrt{2}}{\sqrt{\pi}\,s^2}$$

```
> fouriersin(5*t,t,s);
```

$$-\frac{5}{2}\sqrt{2}\,\sqrt{\pi}\,\mathrm{Dirac}(1,s)$$

```
> fouriercos(exp(-t),t,s);
```

$$\frac{\sqrt{2}}{\sqrt{\pi}\,(s^2+1)}$$

Интегральное преобразование Ханкеля

Интегральное преобразование Ханкеля задается следующим выражением:

$$F[nu](s)=\int_0^{\infty} f(t)\cdot t\cdot \mathrm{BesselJ}(nu,s\cdot t)dt$$

и выполняется функцией:

```
hankel(expr, t, s, nu)
```

Здесь expr — выражение, равенство (или множество, или список с выражениями/равенствами), t — переменная в expr, преобразуемая в параметр преобразования s, nu — порядок преобразования.

Следующий пример демонстрирует применение функции Ханкеля:

```
> hankel(t^(1/3),t,s,2);
```

$$11\,\frac{2^{(5/6)}\,\Gamma\left(\frac{11}{12}\right)^2\sin\left(\frac{1}{12}\pi\right)}{s^{(4/3)}\,\pi}$$

Прямое и обратное преобразования Гильберта

Прямое преобразование Гильберта задается следующим выражением:

$$F(s)=\frac{1}{\pi}\int_{-\infty}^{\infty}\frac{f(t)}{t-s}\,dt$$

и превращает функцию $f(t)$ в $F(s)$.

Обратное преобразование Гильберта означает нахождение $f(t)$ по заданной $F(s)$. Эти преобразования выполняются функциями:

```
hilbert(expr, t, s)
invhilbert(expr, t, s)
```

где назначение параметров очевидно.

Приведенные ниже примеры иллюстрируют выполнение этих преобразований:

```
> assume(a>0);
> hilbert(exp(1), r, z);
```

$$0$$

```
> hilbert(f(u), u, t);
```

$$\mathrm{hilbert}(\mathrm{f}(u), u, t)$$

```
> hilbert(%, t, s);
```

$$-\mathrm{f}(s)$$

```
> hilbert(t*f(t), t, s);
```

$$\frac{s\,\mathrm{hilbert}(\mathrm{f}(t), t, s)\,\pi+\int_{-\infty}^{\infty}\mathrm{f}(\_U)\,d\_U}{\pi}$$

```
> hilbert(t/(t^2+1),t,s);
```

$$\frac{1}{s^2+1}$$

```
> invhilbert(%,s,t);
```

$$\frac{t}{t^2+1}$$

```
> hilbert(sin(x)/x,x,y);
```

$$\frac{\cos(y)-1}{y}$$

```
> hilbert(ln(I*a*x),x,y);
```

$$\text{hilbert}(\ln(I\,x),x,y)+\text{hilbert}(\ln(a\sim),x,y)$$

```
> hilbert(%,y,z);
```

$$-\ln(I\,z)$$

Как видно из этих примеров, обратное преобразование Гильберта, осуществленное над результатом прямого преобразования, не восстанавливает функцию $f(t)$ буквально.

Интегральное преобразование Меллина

Интегральное преобразование Меллина задается выражением:

$$M(s)=\int_0^{\infty} m(x)x^{s-1}dx$$

и реализуется функцией:

```
mellin(expr, x, s)
```

с очевидными параметрами expr, x и s.

Применение преобразования Меллина иллюстрируют следующие примеры:

```
> assume(a>0);
> mellin(x^a,x,s);
```

$$\frac{\Gamma(s+a\sim)\ \text{Heaviside}(s+a\sim)}{\Gamma(s+a\sim+1)}+\frac{\Gamma(-a\sim-s)\ \text{Heaviside}(-a\sim-s)}{\Gamma(1-s-a\sim)}$$

Функция addtable

Как видно из приведенных примеров, не всегда интегральные преобразования дают результат в явном виде. Получить его позволяет вспомогательная функция:

```
addtable(tname,patt,expr,t,s)
```

где tname — наименование преобразования, для которого образец patt должен быть добавлен к таблице поиска. Остальные параметры очевидны.

Следующие примеры поясняют применение этой функции:

```
> fouriersin(f(t),t,s);
```

$$\text{fouriersin}(f(t),t,s)$$

```
> addtable(fouriersin,f(t),F(s),t,s);
> fouriersin(f(x),x,z);
```

$$F(z)$$

Пакет приближения кривых CurveFitting

Общая характеристика пакета CurveFitting

Новый пакет приближения кривых CurveFitting весьма полезен тем, кто занимается столь распространенной задачей, как приближение кривых. Он содержит ряд функций:

```
> with(CurveFitting);
```

Доступ к функциям пакета возможен с помощью конструкций:

```
CurveFitting[function](arguments)
function(arguments)
```

Число функций пакета невелико и все они описаны ниже.

Функция вычисления B-сплайнов Bspline

Функция BSpline(k, v, opt) служит для вычисления B-сплайнов. Она имеет следующие параметры: k — порядок сплайна (целое число), v — имя и opt — параметр в виде knots=knotlist, где knotlist — список из k+1 элементов алгебраического типа. Используя функцию CurveFitting[BSplineCurve], можно строить кривые B-сплайнов. Примеры применения этой функции представлены ниже:

```
> BSpline(3, x);
```

$$\begin{cases} 0 & x<0 \\ \frac{1}{2}x^2 & x<1 \\ -\frac{1}{2}+x-(x-1)^2 & x<2 \\ \frac{5}{2}-x+\frac{1}{2}(x-2)^2 & x<3 \\ 0 & 3\le x \end{cases}$$

```
> BSpline(2, x, knots=[0,a,2]);
```

$$\begin{cases} 0 & x<0 \\ \frac{x}{a} & x<a \\ \frac{-x+a}{2-a}+1 & x<2 \\ 0 & 2\le x \end{cases}$$

Как нетрудно заметить из этих примеров, функция Bspline возвращает результат в виде кусочных функций типа piecewise.

Функция построения B-сплайновых кривых BsplineCurve

Функция BsplineCurve служит для построения кривых B-сплайнов. Она может использоваться в формах:

```
BSplineCurve(xydata, v, opts)
BSplineCurve (xdata, ydata, v, opts)
```

Здесь:

xydata — список, массив или матрица точек в форме [[x1,y1],[x2,y2],...,[xn,yn]];

xdata — список, массив или вектор значений независимой переменной [x1,x2,...,xn];

ydata — список, массив или вектор значений зависимой переменной в форме [y1,y2,...,yn];

v — имя независимой переменной;

opts — необязательный параметр в форме одного или более выражений вида order=k или knots=knotlist.

Примеры применения функции BSplineCurve с порядком, заданным по умолчанию, и с третьим порядком (кубический B-сплайн) представлены на рис. 14.6.

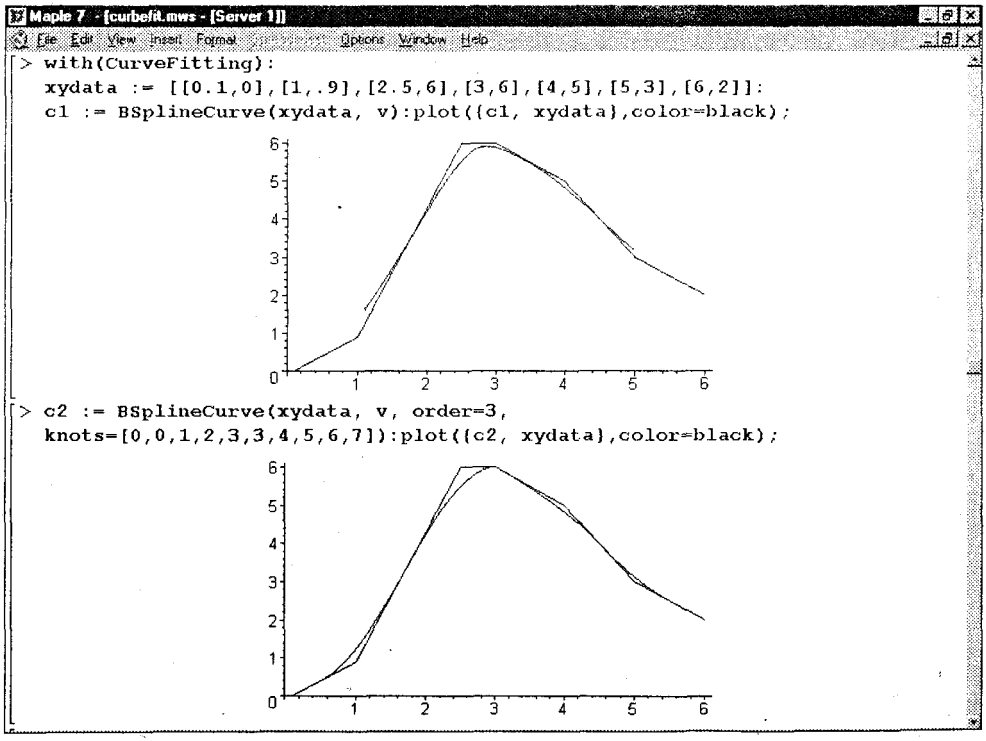


Рис. 14.6. Применение функции BSplineCurve

Следует отметить, что при малом числе точек аппроксимация B-сплайнами дает невысокую точность, что и видно из рис. 14.6.

Функция реализации метода наименьших квадратов LeastSquares

Функция LeastSquares служит для реализации аппроксимации по методу наименьших квадратов:

```
LeastSquares (xydata, v, opts)
LeastSquares(xdata, ydata, v, opts)
```

Все входящие в нее параметры были определены выше (см. параметры функции BSplineCurve). Параметр opts задается в форме выражений weight=wlist, curve=f или params=pset.

Следующие примеры иллюстрируют применение функции LeastSquares:

```
> with(CurveFitting):
LeastSquares([[0,.5],[1,2],[2,4],[3,8]], v);
```

$$-.0500000000000 + 2.44999999999999974\,v$$

```
> LeastSquares([0,1,2,3], [1,2,4,6], v, weight=[1,1,1,10]);
```

$$\frac{95}{146} + \frac{259}{146}\,v$$

```
> LeastSquares([0,1,3,5,6], [1,-1,-3,0,5], v, curve=a*v^2+b*v+c);
```

$$\frac{856}{637} + \frac{33}{49}\,v^2 - \frac{2231}{637}\,v$$

Функция полиномиальной аппроксимации PolynomialInterpolation

Функция PolynomialInterpolation реализует полиномиальную интерполяцию и может использоваться в виде:

```
PolynomialInterpolation (xydata, v)
PolynomialInterpolation(xdata, ydata, v)
```

Параметры функции были определены выше. Параметр v может быть как именем, так и численным значением. Примеры применения функции представлены ниже:

```
> with(CurveFitting):
PolynomialInterpolation([[0,0],[1,2],[2,4],[3,3]], z);
```

$$-\frac{1}{2}z^3 + \frac{3}{2}z^2 + z$$

```
> PolynomialInterpolation([0,2,5,8], [2,a,1,3], 3);
```

$$-\frac{1}{24} + \frac{5}{6}\,a$$

Функция рациональной аппроксимации RacionalInterpolation

Функция рациональной интерполяции задается в виде:

```
RationalInterpolation (xydata, z, opts)
RationalInterpolation(xdata, ydata, z, opts)
```

где необязательный параметр opts задается выражениями method=methodtype или degrees=[d1,d2]. Функция возвращает результат в виде отношения двух полиномов. Параметр methodtype может иметь значения `lookaround или subresultant, задающие учет или пропуск сингулярных точек.

Пример применения функции RationalInterpolation (загрузка пакета опущена, но предполагается):

```
> xpoints := [0,1,2,3,4,-1]: ypoints := [0,3,1,3,a,1/11]:
f := RationalInterpolation(xpoints, ypoints, x);
```

$$f := -3\,\frac{x}{4x^2 - 17x + 12}$$

```
> for i from 1 to 6 do normal(eval(f,x=xpoints[i])-ypoints[i]) end do;
```

$$0$$

$$0$$

$$0$$

$$0$$

$$-\frac{3}{2} - a$$

$$0$$

Функция вычисления обычных сплайнов Spline

Функция:

```
Spline(xydata, v, opts)          Spline(xdata, ydata, v, opts)
```

вычисляет обычные (не B-типа) сплайны. Примеры ее применения даны ниже:

```
> Spline([[0,1],[1,2],[2,5],[3,3]], x);
```

$$\begin{cases} 1 + \frac{2}{15}x + \frac{13}{15}x^3 & x < 1 \\ \frac{21}{5} - \frac{142}{15}x + \frac{48}{5}x^2 - \frac{7}{3}x^3 & x < 2 \\ -\frac{131}{5} + \frac{542}{15}x - \frac{66}{5}x^2 + \frac{22}{15}x^3 & otherwise \end{cases}$$

```
> Spline([0,1,2,3], [1,2,5,3], v, degree=1);
```

$$\begin{cases} 1 + v & v < 1 \\ -1 + 3v & v < 2 \\ 9 - 2v & otherwise \end{cases}$$

Функция аппроксимации непрерывными дробями ThieleInterpolation

Функция ThieleInterpolation осуществляет интерполяцию на основе непрерывных дробей (Thiele's-интерполяцию). Она задается в виде:

```
ThieleInterpolation (xydata, v)
ThieleInterpolation(xdata, ydata, v)
```

Примеры применения данной функции представлены ниже:

```
> ThieleInterpolation([[1,3],[2,5],[4,75],[5,4]], x);
```

$$3+\cfrac{x-1}{\cfrac{1}{2}+\cfrac{x-2}{-\cfrac{1944}{77}+\cfrac{402}{77}x}}$$

```
> ThieleInterpolation([1,2,a], [2,4,3], 3);
```

$$2+\cfrac{2}{\cfrac{1}{2}+\cfrac{1}{\cfrac{1-a}{-\cfrac{3}{2}+a}+2}}$$

Пакет для работы с полиномами PolynomialTools

Обзор возможностей пакета PolynomialTools

Пакет для работы с полиномами PolynomialTools предназначен для выполнения ряда специальных операций с полиномами или создания полиномов с заданными свойствами. Этот пакет имеет небольшое число функций:

```
> with(PolynomialTools);
```

[*IsSelfReciprocal*, *MinimalPolynomial*, *PDEToPolynomial*, *PolynomialToPDE*, *Shorten*, *Shorter*, *Sort*, *Split*, *Splits*, *Translate*]

В пакет входят функции расщепления, сортировки и преобразования полиномов (в том числе в дифференциальные уравнения и наоборот) и др.

Функции для работы с полиномами

Рассмотрим несколько функций пакета PolynomialTools общего характера.

Примеры применения этой функции представлены ниже:

```
> with(PolynomialTools):
IsSelfReciprocal(x^4+x^3+x+1, x, 'p');
```

ПРИМЕЧАНИЕ

Функция IsSelfReciprocal(a, x, 'p') проверяет полином a(x) на условие coeff(a,x,k) = =coeff(a,x,d-k) для всех k = 0..d, где d = degree(a, x) — порядок полинома. Если это условие выполняется, то возвращается логическое значение true, иначе — false. Если порядок d четный и если задан третий аргумент p, то p будет представлять полином P порядка d/2, такой, что x^(d/2)*P(x+1/x) = a. При нечетном d полином a будет взаимообратным, что подразумевает деление на x+1. В этом случае, если p указано, результат вычисляется в форме a/(x+1).

true

```
> p;
```

$-2+x+x^2$

```
> IsSelfReciprocal(x^5-3*x^4+x^3+x^2-3*x+1, x, 'p');
```

true

```
> p;
```

$3-4\,x+x^2$

```
> r := evalf( 1+sqrt(2) );
```

$r := 2.414213562$

Функция `MinimalPolynomial(r, n, acc)` возвращает полином минимальной степени не превышающей `n`, имеющий корень `r`. Необязательный аргумент `acc` задает погрешность приближения. Функция `MinimalPolynomial(r, n)` использует решетчатый алгоритм и находит полином степени `n` (или менее) с наименьшими целыми коэффициентами. Корень `r` может быть действительным или комплексным. Результат зависит от значения переменной окружения `Digits`. По умолчанию `acc` задано как `10^(Digits-2)`. Примеры применения данной функции:

```
> MinimalPolynomial( r, 2 );
```

$-1-2\,\_X+\_X^2$

```
> r := 1+sqrt(2);
```

$r := 1+\sqrt{2}$

```
> ( r, 2 );
```

$1+\sqrt{2},\ 2$

```
> MinimalPolynomial( 1.234, 3 );
```

$-109+61\,\_X-5\,\_X^2+22\,\_X^3$

```
> fsolve( %, _X );
```

1.234000001

Функция `Split(a, x, b)` служит для расщепления полинома `a` с независимой переменной `x`. Параметр `b` — необязательный. Функция `Split(a, x)` осуществляет комплексную факторизацию инвариантного полинома `a` по `x`. Если третий аргумент `b` задан, он представляет множество элементов `{t1,...,tm}`, таких что полином `a` расщепляется над `K=Q(t1,...,tm)`, где `Q` означает поле рациональных чисел. Примеры:

```
> Split(x^2+x+1,x);
```

$(x-\mathrm{RootOf}(\_Z^2+\_Z+1))\,(x+1+\mathrm{RootOf}(\_Z^2+\_Z+1))$

```
> Split(x^2+y*x+1+y^2, x, 'b');
```

$(x-\mathrm{RootOf}(\_Z^2+y\,\_Z+1+y^2))\,(x+y+\mathrm{RootOf}(\_Z^2+y\,\_Z+1+y^2))$

```
> b;
```
$$\{\text{RootOf}(\_Z^2 + y\_Z + 1 + y^2)\}$$

В пакете определена еще одна подобная функция Splits, с которой можно познакомиться по справке на нее.

Функция Translate(a, x, x0) преобразует полином a(x) с подстановкой x = x + x0, где x0 — константа. Примеры применения этой функции даны ниже:

```
> Translate(x^2,x,1);
```
$$1 + 2x + x^2$$
```
> expand(eval(x^2,x=x+1));
```
$$1 + 2x + x^2$$
```
> Translate(x^3,x,2);
```
$$8 + 12x + 6x^2 + x^3$$
```
> expand(eval(x^3,x=x+2));
```
$$8 + 12x + 6x^2 + x^3$$
```
> Translate((x+1)^3,x,-1);
```
$$x^3$$

Функции сортировки полиномов

Для сортировки полиномов предназначены следующие три функции:

```
Shorter(f, g, x)  Sort(v, x)  Shorten(f, x)
```

Здесь f и g полиномы, v — список полиномов и x — независимая переменная. Функции отличаются характером сортировки.

Функция Shorter определяет полином f как более короткий, чем g, по следующим признакам: меньшая длина, меньшее имя независимой переменной x, не дробный и меньшая степень других переменных. Функция Sort сортирует лист полиномов x по признакам, определяемым Shorter. Функция Shorten использует преобразования Мебиуса. Многочисленные детали ее применения можно найти в справке по данной функции.

Примеры применения функций сортировки:

```
> Shorten(x^2+x+1,x):
```
$$x^2 + 3$$
```
> Shorten(3*x^3+18*x+14,x);
```
$$x^3 - 6$$
```
> Shorten(x^4+32);
```
$$x^4 + 2$$
```
> Shorter(x^3,x+5,x);
```
$$false$$
```
> Sort([x^3,x^2,x+1,x+5]);
                    Error, (in sort_poly) sort_poly uses a 2nd argument, x,
                                        which is missing
> Sort([x^3,x^2,x+1,x+5],x);
```
$$[1 + x, x + 5, x^2, x^3]$$

Функции преобразования полиномов в PDE и обратно

Функция `PolynomialToPDE(polys, vars, depvars)` преобразует полиномы `polys` по независимым переменным `vars` в дифференциальные уравнения с частными производными (PDE). Другая функция `PDEToPolynomial(pdes, vars, depvars)` осуществляет обратное преобразование.

Следующие примеры иллюстрируют применение этих функций:

```
> S:= PolynomialToPDE([(x^2 - 2*x + 1)*u + x^3*v], [x], [u,v]);
```

$$S := \left[\left(\frac{\partial^2}{\partial x^2} u(x)\right) - 2\left(\frac{\partial}{\partial x} u(x)\right) + u(x) + \left(\frac{\partial^3}{\partial x^3} v(x)\right)\right]$$

```
> PDEToPolynomial(S, [x], [u,v]);
```

$$[(x^2 - 2x + 1)u + x^3 v]$$

Что нового мы узнали?

В этом уроке мы научились:

- Обращаться к пакетам расширения.
- Пользоваться функциями пакетов комбинаторики.
- Применять пакет финансово-экономических функций.
- Использовать ортогональные многочлены из пакета orthpoly.
- Работать с суммами пакета sumtools.
- Применять степенные разложения пакета powerseries.
- Работать с пакетом численной аппроксимации numapprox.
- Использовать интегральные преобразования пакета inttrans.
- Осуществлять приближение кривых с помощью пакета CurveFitting.
- Использовать пакет работы с полиномами PolynomialTools.

Пакеты линейной алгебры и функциональных систем

- Основные определения линейной алгебры
- Пакет решения задач линейной алгебры linalg
- Пакет линейной алгебры с алгоритмами NAG LinearAlgebra
- Интеграция системы Maple 7 с матричной системой MATLAB
- Пакет линейных функциональных систем LinearFunctionalSystems

Основные определения линейной алгебры

Прежде чем перейти к рассмотрению обширных возможностей пакетов Maple 7 по части решения задач линейной алгебры, рассмотрим краткие определения, относящиеся к ней.

Матрица (m×n) — прямоугольная двумерная таблица, содержащая *m* строк и *n* столбцов элементов, каждый из которых может быть представлен числом, константой, переменной, символьным или математическим выражением (расширительная трактовка матрицы).

Квадратная матрица — матрица, у которой число строк *m* равно числу столбцов *n*. Пример квадратной матрицы размера 3 × 3:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

Сингулярная (*вырожденная*) матрица — квадратная матрица, у которой детерминант (определитель) равен 0. Такая матрица обычно не упрощается при символьных вычислениях. Линейные уравнения с *почти сингулярными* матрицами могут давать большие погрешности при решении.

Единичная матрица — это квадратная матрица, у которой диагональные элементы равны 1, а остальные элементы равны 0. Ниже представлена единичная матрица размера 4 × 4:

$$\mathbf{E} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Сингулярные значения матрицы **A** — квадратные корни из собственных значений матрицы $\mathbf{A}^{\mathrm{T}}\cdot\mathbf{A}$, где $\mathbf{A}^{\mathrm{T}}$ – транспонированная матрица **A** (см. ее определение ниже);

Транспонированная матрица — матрица, у которой столбцы и строки меняются местами, то есть элементы транспонированной матрицы удовлетворяют условию $A^{T}(i,j)=A(j,i)$. Приведем простой пример.

Исходная матрица:

$$\mathbf{A} = \begin{bmatrix} a & b & c \\ d & e & f \\ i & k & l \end{bmatrix}$$

Транспонированная матрица:

$$\mathbf{A}^{\mathrm{T}} = \begin{bmatrix} a & d & i \\ b & e & k \\ c & f & l \end{bmatrix}$$

Обратная матрица — это матрица $\mathbf{M}^{-1}$, которая, будучи умноженной на исходную квадратную матрицу **M**, дает единичную матрицу **E**.

Ступенчатая форма матрицы соответствует условиям, когда первый ненулевой элемент в каждой строке есть 1 и первый ненулевой элемент каждой строки появляется справа от первого ненулевого элемента в предыдущей строке, то есть все элементы ниже первого ненулевого в строке — нули.

Диагональ матрицы — расположенные диагонально элементы $\mathbf{A}_{i,i}$ матрицы **A**. В приведенной ниже матрице элементы диагонали представлены заглавными буквами:

$$\mathbf{A} = \begin{bmatrix} A & b & c \\ d & E & f \\ i & k & L \end{bmatrix}.$$

Обычно указанную диагональ называют *главной* диагональю — для матрицы **A**, приведенной выше, это диагональ с элементами *A*, *E* и *L*. Иногда вводят понятия *поддиагоналей* (элементы *d* и *k*) и *наддиагоналей* (элементы *b* и *f*). Матрица, все элементы которой, расположенные кроме как на диагонали, поддиагонали и наддиагонали, равны нулю, называется *ленточной*.

Ранг матрицы — наибольший из порядков отличных от нуля миноров квадратной матрицы.

След матрицы — сумма диагональных элементов матрицы.

Определитель матрицы — это многочлен от элементов квадратной матрицы, каждый член которого является произведением *n* элементов, взятых по одному из каждой строки и каждого столбца со знаком произведения, заданным четностью перестановок:

$$\det \mathbf{A} = \sum a_{1j} (-1)^{j+1} \mathbf{M}_1^{<j>},$$

где $\mathbf{M}_1^{<j>}$ — определитель матрицы порядка $n - 1$, полученной из матрицы **A** вычеркиванием первой строки и *j*-го столбца. В таком виде определитель (он же детерминант) легко получить в символьных вычислениях. В численных расчетах мы будем подразумевать под определителем численное значение этого многочлена.

Матрица в *целой степени* — квадратная матрица в степени *n* (*n* — целое неотрицательное число), определяемая следующим образом: $\mathbf{M}^0 = \mathbf{E}$, $\mathbf{M}^1 = \mathbf{M}$, $\mathbf{M}^2 = \mathbf{M} \cdot \mathbf{M}$, ..., $\mathbf{M}^n = \mathbf{M}^{n-1} \cdot \mathbf{M}$.

Идемпотентная матрица — матрица, отвечающая условию $\mathbf{P}^2 = \mathbf{P}$.

Симметрическая матрица — матрица, отвечающая условию $\mathbf{A}^{\mathrm{T}} = \mathbf{A}$.

Кососимметрическая матрица — матрица, отвечающая условию $\mathbf{A}^T = -\mathbf{A}$.

Ортогональная матрица — матрица, отвечающая условию $\mathbf{A}^T = \mathbf{A}^{-1}$.

Нуль-матрица — матрица, все элементы которой равны 0.

Блок-матрица — матрица, составленная из меньших по размеру матриц, также можно представить как матрицу, каждый элемент которой — матрица. Частным случаем является блок-диагональная матрица — блок-матрица, элементы-матрицы которой вне диагонали — нуль-матрицы.

Комплексно-сопряженная матрица — матрица $\overline{\mathbf{A}}$, полученная из исходной матрицы **A** заменой ее элементов на комплексно-сопряженные.

Эрмитова матрица — матрица **A**, удовлетворяющая условию $\overline{\mathbf{A}} = \mathbf{A}^T$.

Собственный вектор квадратной матрицы **A** — любой вектор $\mathbf{x} \in V^n$, $\mathbf{x} \neq \mathbf{0}$, удовлетворяющий уравнению $\mathbf{Ax} = g\mathbf{x}$, где g — некоторое число, называемое *собственным значением* матрицы **A**.

Характеристический многочлен матрицы — определитель разности этой матрицы и единичной матрицы, умноженный на переменную многочлена, — $|\mathbf{A} - g\mathbf{E}|$.

Собственные значения матрицы — корни ее характеристического многочлена.

Норма — обобщенное понятие абсолютной величины числа.

Норма трехмерного вектора $\|\mathbf{x}\|$ — его длина.

Норма матрицы — значение $\sup(\|\mathbf{Ax}\|/\|\mathbf{x}\|)$.

L-норма матрицы **A** — число $\|\mathbf{A}\|_L = \max_j \sum_i |A_{i,j}|$.

Матричная форма записи системы линейных уравнений — выражение $\mathbf{A}\cdot\mathbf{X} = \mathbf{B}$, где **A** — матрица коэффициентов системы, **X** — вектор неизвестных и **B** — вектор свободных членов. Один из способов решения такой системы очевиден — $\mathbf{X} = \mathbf{A}^{-1}\mathbf{B}$, где $\mathbf{A}^{-1}$ — обратная матрица.

Пакет решения задач линейной алгебры linalg

Состав пакета linalg

Несомненно, что уникальной возможностью системы Maple 7, как и других систем компьютерной алгебры, является возможность решения задач линейной алгебры в символьном (формульном, аналитическом) виде. Однако такое решение представляет скорее теоретический, чем практический интерес, поскольку даже при небольших размерах матриц (уже при 4–5 строках и столбцах) символьные результаты оказываются очень громоздкими и труднообозримыми. Они полезны только при решении специфических аналитических задач, например

с разреженными матрицами, у которых большинство элементов имеют нулевые значения.

Поэтому разработчики Maple 7 были вынуждены реализовать в своей системе численные методы решения задач линейной алгебры, которые широко используются в основных сферах ее приложения — математическом моделировании систем и устройств, расчетах в электротехнике, механике, астрономии и т. д.

В ядро Maple 7, как отмечалось, введены очень скромные и минимально необходимые средства для решения задач линейной алгебры. Основной упор в их реализации сделан на подключаемые пакеты. Основным из них, унаследованным от предшествующих реализаций системы, является пакет решения задач линейной алгебры `linalg`. Это один из самых обширных и мощных пакетов в области решения задач линейной алгебры. Он содержит свыше ста функций:

```
> with(linalg):
```

Warning, the names fibonacci, inverse and multiply have been redefined
Warning, the protected names norm and trace have been redefined and unprotected

[*BlockDiagonal, GramSchmidt, JordanBlock, LUdecomp, QRdecomp, Wronskian, addcol, addrow, adj, adjoint, angle, augment, backsub, band, basis, bezout, blockmatrix, charmat, charpoly, cholesky, col, coldim, colspace, colspan, companion, concat, cond, copyinto, crossprod, curl, definite, delcols, delrows, det, diag, diverge, dotprod, eigenvals, eigenvalues, eigenvectors, eigenvects, entermatrix, equal, exponential,*

extend, ffgausselim, fibonacci, forwardsub, frobenius, gausselim, gaussjord, geneqns, genmatrix, grad, hadamard, hermite, hessian, hilbert, htranspose, ihermite, indexfunc, innerprod, intbasis, inverse, ismith, issimilar, iszero, jacobian, jordan, kernel, laplacian, leastsqrs, linsolve, matadd, matrix, minor, minpoly, mulcol, mulrow, multiply, norm, normalize, nullspace, orthog, permanent, pivot, potential, randmatrix,

randvector, rank, ratform, row, rowdim, rowspace, rowspan, rref, scalarmul, singularvals, smith, stackmatrix, submatrix, subvector, sumbasis, swapcol, swaprow, sylvester, toeplitz, trace, transpose, vandermonde, vecpotent, vectdim, vector, wronskian]

Ниже указано назначение тех функций пакета linalg, которые подробно не описаны:

- `addcol` — добавляет к одному из столбцов другой столбец, умноженный на некоторое число;
- `addrow` — добавляет к одной из строк другую строку, умноженную на некоторое число;
- `angle` — вычисляет угол между векторами;
- `augment` — объединяет две или больше матриц по горизонтали;
- `backsub` — реализует метод обратной подстановки при решении системы линейных уравнений (см. также `forwardsub`);
- `band` — создает ленточную матрицу;
- `basis` — находит базис векторного пространства;

- `bezout` — создает Bezout-матрицу двух полиномов;
- `BlockDiagonal` — создает блок-диагональную матрицу;
- `blockmatrix` — создает блок-матрицу;
- `cholesky` — декомпозиция Холесского для квадратной положительно определенной матрицы;
- `charmat` — создает характеристическую матрицу (charmat(M,v) — матрица, вычисляемая как v E-M);
- `charpoly` — возвращает характеристический полином матрицы;
- `colspace` — вычисляет базис пространства столбцов;
- `colspan` — находит базис линейной оболочки столбцов матрицы;
- `companion` — вычисляет сопровождающую матрицу, ассоциированную с полиномом;
- `cond` — вычисляет число обусловленности матрицы (cond(M) есть величина norm(M) norm(M^{-1}));
- `curl` — вычисляет ротор вектора;
- `definite` — тест на положительную (отрицательную) определенность матрицы;
- `diag` — создает блок-диагональную матрицу;
- `diverge` — вычисляет дивергенцию векторной функции;
- `eigenvals` — вычисляет собственные значения матрицы;
- `eigenvects` — вычисляет собственные векторы матрицы;
- `equal` — определяет, являются ли две матрицы равными;
- `exponential` — создает экспоненциальную матрицу;
- `ffgausselim` — свободное от дробей Гауссово исключение в матрице;
- `fibonacci` — матрица Фибоначчи;
- `forwardsub` — реализует метод прямой подстановки при решении системы линейных уравнений (например, для матрицы L и вектора b forwardsub(L, b) возвращает вектор решения x системы линейных уравнений **L·x=b**);
- `frobenius` — вычисляет форму Фробениуса (Frobenius) матрицы;
- `gausselim` — Гауссово исключение в матрице;
- `gaussjord` — синоним для `rref` (метод исключения Гаусса—Жордана);
- `geneqns` — генерирует элементы матрицы из уравнений;
- `genmatrix` — генерирует матрицу из коэффициентов уравнений;
- `grad` — градиент векторного выражения;
- `GramSchmidt` — вычисляет ортогональные векторы;
- `hadamard` — вычисляет ограничение на коэффициенты детерминанта;
- `hessian` — вычисляет гессиан-матрицу выражения;

- `hilbert` — создает матрицу Гильберта;
- `htranspose` — находит эрмитову транспонированную матрицу;
- `ihermite` — целочисленная эрмитова нормальная форма;
- `indexfunc` — определяет функцию индексации массива;
- `innerprod` — вычисляет векторное произведение;
- `intbasis` — определяет базис пересечения пространств;
- `ismith` — целочисленная нормальная форма Шмитта;
- `iszero` — проверяет, является ли матрица ноль-матрицей;
- `jacobian` — *вычисляет якобиан векторной функции*;
- `JordanBlock` — возвращает блок-матрицу Жордана;
- `kernel` — находит базис ядра преобразования, соответствующего данной матрице;
- `laplacian` — вычисляет лапласиан;
- `leastsqrs` — решение уравнений по методу наименьших квадратов;
- `linsolve` — решение линейных уравнений;
- `Ludecomp` — осуществляет LU-разложение;
- `minpoly` — вычисляет минимальный полином матрицы;
- `mulcol` — умножает столбец матрицы на заданное выражение;
- `mulrow` — умножает строку матрицы на заданное выражение;
- `multiply` — перемножение матриц или матрицы и вектора;
- `normalize` — нормализация вектора;
- `orthog` — тест на ортогональность матрицы;
- `permanent` — вычисляет перманент матрицы — определитель, вычисляемый без перестановок;
- `pivot` — вращение относительно элементов матрицы;
- `potential` — вычисляет потенциал векторного поля;
- `Qrdecomp` — осуществляет QR-разложение;
- `randmatrix` — генерирует случайные матрицы;
- `randvector` — генерирует случайные векторы;
- `ratform` — вычисляет рациональную каноническую форму;
- `references` — выводит список основополагающих работ по линейной алгебре;
- `rowspace` — вычисляет базис пространства строки;
- `rowspan` — вычисляет векторы охвата для места столбца;
- `rref` — реализует преобразование Гаусса–Жордана матрицы;
- `scalarmul` — умножение матрицы или вектора на заданное выражение;
- `singval` — вычисляет сингулярное значение квадратной матрицы;

- `singularvals` — возвращает список сингулярных значений квадратной матрицы;
- `smith` — вычисляет Шмиттову нормальную форму матрицы;
- `submatrix` — извлекает указанную подматрицу из матрицы;
- `subvector` — извлекает указанный вектор из матрицы;
- `sumbasis` — определяет базис объединения системы векторов;
- `swapcol` — меняет местами два столбца в матрице;
- `swaprow` — меняет местами две строки в матрице;
- `sylvester` — создает матрицу Сильвестра из двух полиномов;
- `toeplitz` — создает матрицу Теплица;
- `trace` — возвращает след матрицы;
- `vandermonde` — создает вандермондову матрицу;
- `vecpotent` — вычисляет векторный потенциал;
- `vectdim` — определяет размерность вектора;
- `wronskian` — вронскиан векторных функций.

Ниже мы рассмотрим более подробно наиболее часто используемые функции из этого пакета. С деталями синтаксиса (достаточно разнообразного) для каждой из указанных функций можно ознакомиться в справочной системе Maple. Для этого достаточно использовать команду `?name;`, где `name` — имя функции (из приведенного списка).

Интерактивный ввод матриц

Для интерактивного ввода матриц можно, определив размерность некоторого массива, использовать функцию `entermatrix`:

```
> A:=array(1..3,1..3);
```

$$A := \text{array}(1 .. 3, 1 .. 3, [\ ])$$

После исполнения этого фрагмента документа диалог с пользователем имеет следующий вид:

```
> entermatrix(A);
enter element 1,1 > 1;
enter element 1,2 > 2;
enter element 1,3 > 3;
enter element 2,1 > 4;
enter element 2,2 > 5;
enter element 2,3 > 6;
enter element 3,1 > 7;
enter element 3,2 > 8;
enter element 3,3 > 9;
```

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

```
> B:=(%);
```

$$B := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
> B[1,1];
 1
> B[2,2];
 5
> B[3,3];
 9
```

Основные функции для задания векторов и матриц

В библиотечном файле `linalg` имеются следующие функции для задания векторов и матриц:

- `vector(n,list)` — создание вектора с `n` элементами, заданными в списке `list`;
- `matrix(n,m,list)` — создание матрицы с числом строк `n` и столбцов `m` с элементами, заданными списком `list`.

Ниже показано применение этих функций:

```
> V:=vector(3,[12,34,56]);
```

$$V := [12, 34, 56]$$

```
> M:=matrix(2,3,[1,2,3,4]);
```

$$M := \begin{bmatrix} 1 & 2 & 3 \\ 4 & M_{2,2} & M_{2,3} \end{bmatrix}$$

```
> V[2];
 34
> M[1,3];
 3
> M[2,3];
```

$$M_{2,3}$$

Обратите внимание на последние примеры — они показывают вызов индексированных переменных вектора и матрицы.

Функции для работы с векторами и матрицами

Для работы с векторами и матрицами Maple 7 имеет множество функций, входящих в пакет `linalg`. Ограничимся приведением краткого описания наиболее распространенных функций этой категории.

Операции со структурой отдельного вектора **V** и матрицы **M**:

- `coldim(M)` — возвращает число столбцов матрицы `M`;
- `rowdim(M)` — возвращает число строк матрицы `M`;

- `vectdim(V)` — возвращает размерность вектора `V`;
- `col(M,i)` — возвращает *i*-й столбец матрицы `M`;
- `row(M,i)` — возвращает *i*-ю строку матрицы `M`;
- `minor(M,i,j)` — возвращает минор матрицы `M` для элемента с индексами `i` и `j`;
- `delcols(M,i..j)` — удаляет столбцы матрицы `M` от *i*-го до *j*-го;
- `delrows(V,i..j)` — удаляет строки матрицы `M` от *i*-й до *j*-й;
- `extend(M,m,n,x)` — расширяет матрицу `M` на `m` строк и `n` столбцов с применением заполнителя `x`.

Основные векторные и матричные операции:

- `dotprod(U,V)` — возвращает скалярное произведение векторов `U` и `V`;
- `crossprod(U,V)` — возвращает векторное произведение векторов `U` и `V`;
- `norm(V)` или `norm(M)` — возвращает норму вектора или матрицы;
- `copyinto(A,B,i,j)` — копирует матрицу `A` в `B` для элементов последовательно от `i` до `j`;
- `concat(M1,M2)` — возвращает объединенную матрицу с горизонтальным слиянием матриц `M1` и `M2`;
- `stack(M1,M2)` — возвращает объединенную матрицу с вертикальным слиянием `M1` и `M2`;
- `matadd(A,B)` и `evalm(A+B)` — возвращает сумму матриц `A` и `B`;
- `multiply(A,B)` и `evalm(A&*B)` — возвращает произведение матриц `A` и `B`;
- `adjoint(M)` или `adj(M)` — возвращает присоединенную матрицу, такую что M?adj(M) дает диагональную матрицу, определитель которой есть det(M);
- `charpoly(M,lambda)` — возвращает характеристический полином матрицы `M` относительно заданной переменной `lambda`;
- `det(M)` — возвращает детерминант (определитель) матрицы `M`;
- `Eigenvals(M,vector)` — инертная форма функции, возвращающей собственные значения матрицы `M` и (при указании необязательного параметра `vector`) соответствующие им собственные векторы;
- `jordan(M)` — возвращает матрицу `M` в форме Жордана;
- `hermite(M)` — возвращает матрицу `M` в эрмитовой форме;
- `trace(M)` — возвращает след матрицы `M`;
- `rank(M)` — возвращает ранг матрицы `M`;
- `transpose(M)` — возвращает транспонированную матрицу `M`;
- `inverse(M)` или `evalm(1/M)` — возвращает матрицу, обратную к `M`;
- `singularvals(A)` — возвращает сингулярные значения массива или матрицы `A`.

Приведем примеры применения некоторых из этих функций:

```
> M:=matrix(2,2,[a,b,c,d]);
```

$$M := \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

```
> transpose(M);
```

$$\begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

```
> inverse(M);
```

$$\begin{bmatrix} -\dfrac{d}{-a\,d+b\,c} & \dfrac{b}{-a\,d+b\,c} \\ \dfrac{c}{-a\,d+b\,c} & -\dfrac{a}{-a\,d+b\,c} \end{bmatrix}$$

```
> det(M);
```

$$a\,d-b\,c$$

```
> rank(M);
```

$$2$$

```
> trace(M);
```

$$a+d$$

```
> M:=matrix(2,2,[1,2,3,4]);
```

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

```
> ev:=evalf(Eigenvals(M,V));
```

$$ev := [-.372281323,\ 5.372281323]$$

```
> eval(V);
```

$$\begin{bmatrix} -.8245648401 & -.4222291504 \\ .5657674650 & -.9230523142 \end{bmatrix}$$

```
> charpoly(M,p);
```

$$p^2-5\,p-2$$

```
> jordan(M);
```

$$\begin{bmatrix} \dfrac{5}{2}+\dfrac{1}{2}\sqrt{33} & 0 \\ 0 & \dfrac{5}{2}-\dfrac{1}{2}\sqrt{33} \end{bmatrix}$$

```
> A:= array( [[1,0,1],[1,0,1],[0,1,0]] );
```

$$A := \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

```
> singularvals(A);
```

$$[0,\ 2,\ 1]$$

Читатель, понимающий суть матричных вычислений, легко справится с тестированием других функций, входящих в пакет `linalg`. В приведенных примерах полезно обратить внимание на то, что многие матричные функции способны выдавать результаты вычислений в аналитическом виде, что облегчает разбор выполняемых ими операций.

Решение систем линейных уравнений

Ниже представлен простой пример составления и решения трех систем линейных уравнений с применением функций, входящих в пакет `linalg`:

```
> with(linalg):
> C:=matrix(3,3,[[4,8,2],[6,2,3],[3,7,11]]);
```

$$C := \begin{bmatrix} 4 & 8 & 2 \\ 6 & 2 & 3 \\ 3 & 7 & 11 \end{bmatrix}$$

```
> B:=matrix(3,1,[5,6,1]);
```

$$B := \begin{bmatrix} 5 \\ 6 \\ 1 \end{bmatrix}$$

```
> A:=evalm(C);
```

$$A := \begin{bmatrix} 4 & 8 & 2 \\ 6 & 2 & 3 \\ 3 & 7 & 11 \end{bmatrix}$$

```
> A1:=copyinto(B,C,1,1):
```

$$A1 := \begin{bmatrix} 5 & 8 & 2 \\ 6 & 2 & 3 \\ 1 & 7 & 11 \end{bmatrix}$$

```
> C:=evalm(A):A2:=copyinto(B,C,1,2);
```

$$A2 := \begin{bmatrix} 4 & 5 & 2 \\ 6 & 6 & 3 \\ 3 & 1 & 11 \end{bmatrix}$$

```
> C:=evalm(A):A3:=copyinto(B,C,1,3);
```

$$A3 := \begin{bmatrix} 4 & 8 & 5 \\ 6 & 2 & 6 \\ 3 & 7 & 1 \end{bmatrix}$$

```
> x1:=det(A1)/det(A);
```

$$x1 := \frac{419}{380}$$

```
> x2:=det(A2)/det(A);
```

$$x2 := \frac{3}{20}$$

```
> x3:=det(A3)/det(A);
```

$$x3 := \frac{-29}{95}$$

А теперь рассмотрим пример решения матричного уравнения в символьном виде:

```
> A:=matrix(2,2,[a,b,c,d]);
```

$$A := \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

```
> B:=vector(2,[c,d]);
```

$$B := [c, d]$$

```
> X:=linsolve(A,B);
```

$$X := \left[-\frac{d\,(-b+c)}{b\,c-d\,a}, \frac{-d\,a+c^2}{b\,c-d\,a} \right]$$

Следующий пример показывает решение более сложной системы линейных уравнений с комплексными коэффициентами:

```
> A:=matrix(2,2,[[10+200*I,-200*I],[-200*I,170*I]]);
```

$$A := \begin{bmatrix} 10 + 200\,I & -200\,I \\ -200\,I & 170\,I \end{bmatrix}$$

```
> B:=vector(2,[5,0]);
```

$$B := [5, 0]$$

```
> X:=multiply(inverse(A),B);
```

$$X := \left[\frac{289}{7778} + \frac{510}{3889} I, \frac{170}{3889} + \frac{600}{3889} I \right]$$

```
> Digits:=5: convert(eval(X),float);
```

$$[.037156 + .13114\,I, .043713 + .15428\,I]$$

На этот раз решение получено использованием функций умножения матриц и вычисления обратной матрицы в виде $\mathbf{X} = \mathbf{A}^{-1}\,\mathbf{B}$, то есть в матричном виде. В конце примера показано преобразование результатов с целью их получения в обычной форме комплексных чисел с частями, представленными в форме чисел с плавающей точкой.

Пакет линейной алгебры с алгоритмами NAG LinearAlgebra

Назначение и загрузка пакета LinearAlgebra

В последние годы разработчики систем символьной математики осознали, что малая скорость выполнения векторных и матричных операций при решении задач линейной алгебры оборачивается потерей заметной части рынка систем компьютерной математики. Новые версии таких систем (Mathematica 4/4.1 и Maple 6/7) отличаются от прежних прежде всего резким повышением эффективности решения задач линейной алгебры в численном виде.

В новых реализациях систем Maple и MATLAB была сделана ставка на использование давно апробированных быстрых алгоритмов линейной алгебры, предложенных создателями Number Algorithm Group (NAG). Эти алгоритмы издавна применяются на больших ЭВМ и суперкомпьютерах, обеспечивая ускорение численных матричных операций от нескольких раз до нескольких десятков раз. Их применение обеспечивает эффективное использование систем символьной

математики в решении задач, сводящихся к задачам линейной алгебры. В числе таких задач многочисленные задачи теоретической электротехники, механики многих объектов, моделирования электронных устройств и т. д.

В Maple 7 использование алгоритмов NAG является одной из первых отличительных черт новой версии системы. Оно реализуется новым пакетом `LinearAlgebra`. Для его загрузки используются следующие команды:

```
> restart; with(LinearAlgebra):
```

[*Add*, *Adjoint*, *BackwardSubstitute*, *BandMatrix*, *Basis*, *BezoutMatrix*, *BidiagonalForm*, *BilinearForm*, *CharacteristicMatrix*, *CharacteristicPolynomial*, *Column*, *ColumnDimension*, *ColumnOperation*, *ColumnSpace*, *CompanionMatrix*, *ConditionNumber*, *ConstantMatrix*, *ConstantVector*, *CreatePermutation*, *CrossProduct*, *DeleteColumn*, *DeleteRow*, *Determinant*, *DiagonalMatrix*, *Dimension*, *Dimensions*, *DotProduct*, *Eigenvalues*, *Eigenvectors*, *Equal*, *ForwardSubstitute*, *FrobeniusForm*, *GenerateEquations*, *GenerateMatrix*, *GetResultDataType*, *GetResultShape*, *GivensRotationMatrix*, *GramSchmidt*, *HankelMatrix*, *HermiteForm*, *HermitianTranspose*, *HessenbergForm*, *HilbertMatrix*, *HouseholderMatrix*, *IdentityMatrix*, *IntersectionBasis*, *IsDefinite*, *IsOrthogonal*, *IsSimilar*, *IsUnitary*, *JordanBlockMatrix*, *JordanForm*, *LA_Main*, *LUDecomposition*, *LeastSquares*, *LinearSolve*, *Map*, *Map2*, *MatrixAdd*, *MatrixInverse*, *MatrixMatrixMultiply*, *MatrixNorm*, *MatrixScalarMultiply*, *MatrixVectorMultiply*, *MinimalPolynomial*, *Minor*, *Multiply*, *NoUserValue*, *Norm*, *Normalize*, *NullSpace*, *OuterProductMatrix*, *Permanent*, *Pivot*, *QRDecomposition*, *RandomMatrix*, *RandomVector*, *Rank*, *Row*, *RowDimension*, *RowOperation*, *RowSpace*, *ScalarMatrix*, *ScalarMultiply*, *ScalarVector*, *SchurForm*, *SingularValues*, *SmithForm*, *SubMatrix*, *SubVector*, *SumBasis*, *SylvesterMatrix*, *ToeplitzMatrix*, *Trace*, *Transpose*, *TridiagonalForm*, *UnitVector*, *VandermondeMatrix*, *VectorAdd*, *VectorAngle*, *VectorMatrixMultiply*, *VectorNorm*, *VectorScalarMultiply*, *ZeroMatrix*, *ZeroVector*, *Zip*]

```
> infolevel[LinearAlgebra]:=1;
```

$$infolevel_{LinearAlgebra} := 1$$

Нетрудно заметить, что многие функции этого пакета повторяет по назначению функции более старого пакета `linalg`, описанного выше. Поэтому мы не будем останавливаться на их повторном описании. Главное то, что эти функции задействуют возможности быстрых алгоритмов NAG и в отличие от функций пакета `linalg` ориентированы на численные расчеты в формате обработки вещественных чисел, характерном для компьютерной платформы. Знающий матричные методы читатель легко поймет назначение функций пакета `LinearAlgebra` по их составным названиям. Например, `DeleteColumn` означает удаление столбца матрицы, `ToeplitzMatrix` означает создание матрицы Теплица, `ZeroMatrix` — создание матрицы с нулевыми элементами и т. д. Все имена функций этого пакета начинаются с заглавной буквы.

Примеры матричных операций с применением пакета LinearAlgebra

Применение алгоритмов NAG особенно эффективно в том случае, когда используется встроенная в современные микропроцессоры арифметика чисел с плавающей запятой. С помощью специального флага такую арифметику можно отключать или включать:

```
> UseHardwareFloats := false;     # use software floats
```

$$UseHardwareFloats := false$$

```
> UseHardwareFloats := true;      # default behaviour
```

$$UseHardwareFloats := true$$

Матрицы в новом пакете линейной алгебры могут задаваться в угловых скобках, как показано ниже:

```
> M1:=<<1|2>,<4|5>>; M2:=<<1|2.>,<4|5>>;
```

$$M1 := \begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix}$$

$$M2 := \begin{bmatrix} 1 & 2. \\ 4 & 5 \end{bmatrix}$$

После этого можно выполнять с ними типовые матричные операции. Например, можно инвертировать (обращать) матрицы:

```
> M1^(-1); M2^(-1);
```

$$\begin{bmatrix} \frac{-5}{3} & \frac{2}{3} \\ \frac{4}{3} & \frac{-1}{3} \end{bmatrix}$$

```
MatrixInverse: "calling external function"
MatrixInverse: "NAG"  hw_f07adf
MatrixInverse: "NAG"  hw_f07ajf
```

$$\begin{bmatrix} -1.66666666666666652 & .666666666666666630 \\ 1.33333333333333326 & -.333333333333333315 \end{bmatrix}$$

Обратите внимание, что Maple 7 теперь выдает информационные сообщения о новых условиях реализации операции инвертирования матриц с вещественными элементами, и в частности об использовании алгоритмов NAG и арифметики, встроенной в сопроцессор.

Следующий пример иллюстрирует создание двух случайных матриц M1 и M2 и затем их умножение:

```
> M1:=RandomMatrix(2,3); M2:=RandomMatrix(3,3);
Multiply(M1,M2,'inplace'); M1;
```

$$M1 := \begin{bmatrix} -50 & 62 & -71 \\ 30 & -79 & 28 \end{bmatrix}$$

$$M2 := \begin{bmatrix} 20 & -34 & -21 \\ -7 & -62 & -56 \\ 16 & -90 & -8 \end{bmatrix}$$

$$\begin{bmatrix} -2570 & 4246 & -1854 \\ 1601 & 1358 & 3570 \end{bmatrix}$$

$$\begin{bmatrix} -2570 & 4246 & -1854 \\ 1601 & 1358 & 3570 \end{bmatrix}$$

Параметр `inplace` в функции умножения обеспечивает помещение результата умножения матриц на место исходной матрицы `M1` — излюбленный прием создателей быстрых матричных алгоритмов NAG. Поскольку матрицы `M1` и `M2` заданы как случайные, то при повторении этого примера результаты, естественно, будут иными, чем приведенные.

Следующий пример иллюстрирует проведение хорошо известной операции LU-разложения над матрицей `M`, созданной функцией `Matrix`:

```
> M:=Matrix([[14,-8,1],[-11,-4,18],[3,12,19]], datatype=float);
LUDecomposition(M,output=['NAG'],inplace);
ipiv:=%[1];
M;
```

$$M := \begin{bmatrix} 14. & -8. & 1. \\ -11. & -4. & 18. \\ 3. & 12. & 19. \end{bmatrix}$$

```
LUDecomposition: calling external function"
LUDecomposition: "NAG"   hw_f07adf
```

$$\begin{bmatrix} 1 \\ 3 \\ 3 \end{bmatrix}, \begin{bmatrix} 14. & -8. & 1. \\ .214285714285714274 & 13.7142857142857136 & 18.7857142857142848 \\ -.785714285714285698 & -.750000000000000000 & 32.8750000000000000 \end{bmatrix}$$

$$ipiv := \begin{bmatrix} 1 \\ 3 \\ 3 \end{bmatrix}$$

$$\begin{bmatrix} 14. & -8. & 1. \\ .214285714285714274 & 13.7142857142857136 & 18.7857142857142848 \\ -.785714285714285698 & -.750000000000000000 & 32.8750000000000000 \end{bmatrix}$$

Конечной целью большинства матричных операций является решение систем линейных уравнений. Для этого пакет `LinearAlgebra` предлагает великое множество методов и средств их реализации. Мы ограничимся простым примером одновременного решения сразу трех систем уравнений. Дабы не загромождать книгу массивными выражениями, ограничимся решением систем из двух линейных уравнений, матрица коэффициентов у которых одна, а векторы свободных членов разные. Ниже показан пример решения такой системы:

```
> M:=Matrix([[1..3],[4,5]],datatype=float);
V1:=<1..2>;
```

```
V2:=<7,-11>;
V3:=<-34,-67>;
```

$$M := \begin{bmatrix} 1. & 3. \\ 4. & 5. \end{bmatrix}$$

$$V1 := \begin{bmatrix} 1. \\ 2 \end{bmatrix}$$

$$V2 := \begin{bmatrix} 7 \\ -11 \end{bmatrix}$$

$$V3 := \begin{bmatrix} -34 \\ -67 \end{bmatrix}$$

```
> LinearSolve(M,<V1|V2|V3>);
```

LinearSolve: "calling external function"
LinearSolve: "NAG" hw_f07adf
LinearSolve: "NAG" hw_f07aef

$$\begin{bmatrix} .142857142857142905 & -9.71428571428571352 & -4.42857142857143060 \\ .285714285714285698 & 5.57142857142857118 & -9.85714285714285588 \end{bmatrix}$$

```
> M:=Matrix([[1.,3],[4,5]],datatype=float);
ipiv, M := LUDecomposition(M,output=['NAG'],inplace);
LinearSolve([ipiv, M], <V1|V2|V3>);
```

$$M := \begin{bmatrix} 1. & 3. \\ 4. & 5. \end{bmatrix}$$

LUDecomposition: "calling external function"
LUDecomposition: "NAG" hw_f07adf

$$ipiv, M := \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 4. & 5. \\ .250000000000000000 & 1.75000000000000000 \end{bmatrix}$$

LinearSolve: "calling external function"
LinearSolve: "NAG" hw_f07aef

$$\begin{bmatrix} .142857142857142905 & -9.71428571428571352 & -4.42857142857143060 \\ .285714285714285698 & 5.57142857142857118 & -9.85714285714285588 \end{bmatrix}$$

На этом, учитывая ограниченный объем книги, мы завершаем обзор пакета LinearAlgebra. Читатель, познающий или знающий методы линейной алгебры, может опробовать в работе любые функции этого пакета самостоятельно или познакомиться со множеством примеров, размещенных в справочной системе Maple 7. Возможности пакетов linalg и LinearAlgebra удовлетворят самых требовательных специалистов в этой области математики.

Интеграция Maple 7 с MATLAB

Краткие сведения о MATLAB

Несмотря на обширные средства линейной алгебры (да и многие другие), имеющиеся у системы Maple 7, есть системы компьютерной математики, решающие некоторые классы задач более эффективно, и прежде всего быстрее. В области

линейной алгебры к таким системам, безусловно, относится система MATLAB, созданная компанией MathWorks, Inc. Ее название происходит именно от слов MATrix LABoratory — матричная лаборатория.

MATLAB содержит в своем ядре многие сотни матричных функций и является одной из лучших матричных систем для персональных компьютеров. Она реализует самые современные алгоритмы матричных операций, включая, кстати, и алгоритмы NAG. Однако главное достоинство MATLAB — наличие множества дополнительных пакетов как по классическим разделам математики, так и по самым новейшим, таким как нечеткая логика, нейронные сети, идентификация систем, обработка сигналов и др. Знаменитым стал пакет моделирования систем и устройств Simulink, включаемый в пакет поставки системы MATLAB. Последней версией системы является MATLAB 6.0.

В то же время нельзя не отметить, что MATLAB — одна из самых громоздких математических систем. Инсталляция ее полной версии занимает около 1,5 Гбайт дискового пространства. Несмотря на это, интеграция различных математических систем с данной системой, похоже, становится своеобразной модой. Такая возможность предусмотрена и в системе Maple 7 с помощью пакета Matlab.

Загрузка пакета расширения Matlab

Для загрузки пакета Matlab используется команда:

```
> with(Matlab);
```

[*chol*, *closelink*, *defined*, *det*, *dimensions*, *eig*, *evalM*, *fft*, *getvar*, *inv*, *lu*, *ode45*, *openlink*, *qr*, *setvar*, *size*, *square*, *transpose*]

Использование этой команды ведет к автоматическому запуску системы MATLAB (гарантируется работа с версиями MATLAB до 5.3.1 включительно) и установлению необходимой объектной связи между системами Maple 7 и MATLAB.

ПРИМЕЧАНИЕ

Как нетрудно заметить, данный пакет дает доступ всего к 18 функциям системы MATLAB (из многих сотен, имеющихся только в ядре последней системы). Таким образом, есть все основания полагать, что возможности MATLAB в интеграции с системой Maple 7 используются пока очень слабо и носят рудиментарный характер. Стоит ли ради этих функций иметь на компьютере огромную систему MATLAB, пользователи должны решать сами. Если ответ положительный, то, скорее всего, пользователь решает тот класс задач, для которых лучше подходит MATLAB, и надо задуматься уже над тем, нужен ли в этом случае Maple.

Типовые матричные операции пакета расширения Matlab

Большинство функций пакета Matlab (не путайте с системой MATLAB, имя которой надо записывать прописными буквами) реализуют самые обычные матричные операции, что и иллюстрируют приведенные ниже примеры.

Зададим матрицу `M` в формате Maple:

```
> maplematrix_a:=array(1..3,1..3,[[6,4,2],[7,8,1],[3,7,3]]);
```

$$maplematrix\_a := \begin{bmatrix} 6 & 4 & 2 \\ 7 & 8 & 1 \\ 3 & 7 & 3 \end{bmatrix}$$

Ниже даны примеры транспонирования матрицы, ее инвертирования, вычисления детерминанта и собственных значений матрицы:

```
> Matlab[transpose](maplematrix_a);
```

$$\begin{bmatrix} 6. & 7. & 3. \\ 4. & 8. & 7. \\ 2. & 1. & 3. \end{bmatrix}$$

```
> Matlab[inv](maplematrix_a);
```

$$\begin{bmatrix} .212499999999999967 & .0250000000000000360 & -.150000000000000022 \\ -.224999999999999978 & .149999999999999966 & .100000000000000032 \\ .312500000000000000 & -.374999999999999944 & .249999999999999944 \end{bmatrix}$$

```
> Matlab[det](maplematrix_a);
```

80.

```
> Matlab[eig](maplematrix_a);
```

$$\begin{bmatrix} 13.8861968635740444 + 0.\,I \\ 1.55690156821298186 + 1.82679340924767875\,I \\ 1.55690156821298186 - 1.82679340924767875\,I \end{bmatrix}$$

Можно проверить, является ли матрица квадратной:

```
> Matlab[square](maplematrix_a);
```

true

а также вычислить размер матрицы:

```
> Matlab[dimensions](maplematrix_a);
```

[3, 3]

Можно также проверить, является ли данная матрица матрицей системы MATLAB:

```
> Matlab[defined]("maplematrix_a");
```

false

Здесь надо иметь в виду, что форматы матриц в системах Maple и MATLAB различны. Выполним LU-преобразование матрицы:

```
> Matlab[lu](maplematrix_a,output='L');
```

$$\begin{bmatrix} 7. & 8. & 1. \\ -.857142857142857094 & 3.57142857142857162 & 2.57142857142857162 \\ -.428571428571428548 & .799999999999999820 & 3.19999999999999972 \end{bmatrix}$$

Таким образом, видно, что пакет Maple в данном случае реализует типовые матричные операции, но средствами системы MATLAB. Загрузка последней происходит автоматически при загрузке пакета Matlab. Если система MATLAB не установлена на вашем компьютере, то доступ к функциям пакета Matlab будет отсутствовать, а Maple 7 при попытке использования данных функций будет выдавать сообщения об ошибках.

Выделение сигнала на фоне шумов

Среди небольшого числа доступных функций системы MATLAB в пакете Matlab нельзя не выделить особо функции быстрого прямого и обратного преобразований Фурье. В системе MATLAB эти функции реализуют наиболее эффективные алгоритмы быстрого преобразования Фурье (БПФ), обеспечивающие решение крупноразмерных задач (например, обработки сигналов, представленных векторами и матрицами больших размеров) в десятки раз быстрее, чем при обычных методах выполнения преобразований Фурье.

Покажем возможность применения БПФ на ставшем классическим примере — выделении спектра полезного сигнала на фоне сильных помех. Зададим некоторый двухчастотный сигнал, имеющий 1500 точек отсчета:

```
> num := 1500:
Time := [seq(.03*t, t=1..num)]:
data := [seq((3.6*cos(Time[t]) + cos(6*Time[t])), t=1..num)]:
plots[pointplot](zip((x,y)->[x,y],Time,data), style=line);
```

График сигнала представлен на рис. 15.1.

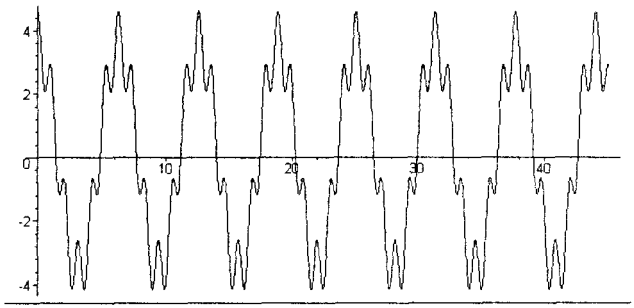


Рис. 15.1. График исходного сигнала

Теперь с помощью генератора случайных чисел наложим на этот сигнал сильный «шум» (слово «шум» взято в кавычки, поскольку речь идет о математическом моделировании шума, а не о реальном шуме физической природы):

```
> tol := 10000:
r := rand(0..tol):
noisy_data := [seq(r()/(tol)*data[t], t=1..num)]:
plots[pointplot](zip((x,y)->[x,y],Time,noisy_data), style=line);
```

Нетрудно заметить, что теперь форма сигнала настолько замаскирована шумом (рис. 15.2), что можно лишь с трудом догадываться, что сигнал имеет периодическую составляющую малой амплитуды. Эта высокочастотная составляющая сигнала скрыта шумом.

Подвергнем полученный сигнал (в виде временной зависимости) прямому преобразованию Фурье, реализованному функцией `fft`:

```
> ft := fft(noisy_data):
> VectorOptions(ft, datatype);
```

$complex_8$

Эта операция переводит задачу из временного представления сигнала в частотное, что позволяет использовать частотные методы анализа сигнала. Выделим, к примеру, действительную и мнимую части элементов вектора ft и проверим его размер:

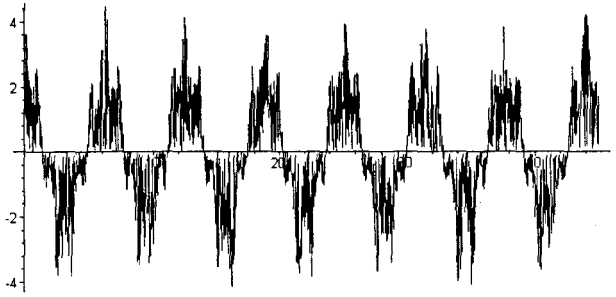


Рис. 15.2. Временная зависимость сигнала с шумом

```
> real_part := map(Re, ft):
imag_part := map(Im, ft):
> dimensions(ft);
```

[1500]

Теперь выполним обычные операции вычисления спектра и зададим построение графика частотного спектра мощности сигнала:

```
> setvar("FT", ft);setvar("n", num);
> evalM("result = FT.*conj(FT)/n");
> pwr := getvar("result"):
> VectorOptions(pwr, datatype);
```

$float_8$

```
>pwr_list := convert(pwr, list):
>pwr_points := [seq([(t-1)/Time[num], pwr_list[t]], t=1..num/2)]:
>plots[pointplot](pwr_points, style=line);
```

Спектрограмма сигнала представлена на рис. 15.3.

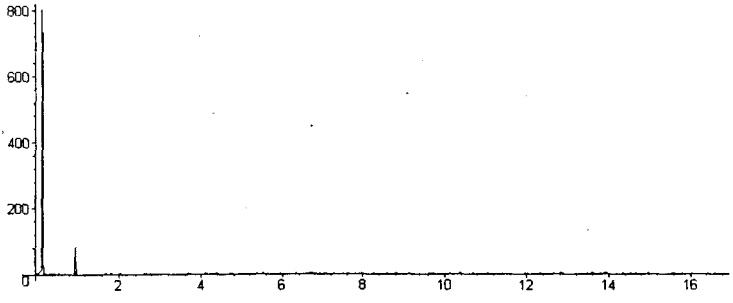


Рис. 15.3. Спектрограмма сигнала

Из нее отчетливо видно, что сигнал представлен двумя частотными составляющими с разной амплитудой. Таким образом, задача четкого выделения полезных компонент спектра из зашумленного сигнала с применением средств системы MATLAB успешно решена.

Пакет анализа линейных функциональных систем LinearFunctionalSystems

Назначение пакета LinearFunctionalSystems

Пакет LinearFunctionalSystems содержит набор функций для решения задач, связанных с анализом линейных функциональных систем. Обычно такие системы описываются линейными дифференциальными уравнениями, имеющими то или иное решение. Пакет LinearFunctionalSystems позволяет провести тестирование подготовленной системы, оценить ряд ее параметров и получить решение одним из ряда методов.

Вызов всех функций пакета осуществляется командой:

```
> with(LinearFunctionalSystems);
```

[*AreSameSolution*, *CanonicalSystem*, *ExtendSeries*, *HomogeneousSystem*, *IsSolution*, *MatrixTriangularization*, *PolynomialSolution*, *Properties*, *RationalSolution*, *SeriesSolution*, *UniversalDenominator*]

Тестовые функции пакета LinearFunctionalSystems

Прежде чем рассматривать основные функции пакета, рассмотрим две тестовые функции. Они представлены следующими формами записи:

```
IsSolution(sol, sys, vars)      IsSolution(sol, A, b, x, case)
IsSolution(sol, A, x, case)     AreSameSolution(sol, sol1)
```

В них: `sol` — тестируемое решение, `sys` — система функциональных уравнений, `x` — независимая переменная решения, `A` и `b` — матрица и вектор с рациональными элементами, `case` — имя метода решения (`'differential'`, `'difference'` или `'qdifference'`).

Функции решения линейных функциональных систем

Группа основных функций пакета LinearFunctionalSystems имеет идентичный синтаксис и записывается в виде:

```
name(sys,vars,[method])  или    name(A[,b],x,case,[method])
```

Здесь `name` — одно из следующих имен:

- `PolynomialSolution` — решение в форме полинома;
- `RationalSolution` — решение в форме рационального выражения;
- `SeriesSolution` — решение в виде ряда;
- `UniversalDenominator` — решение с универсальным знаменателем (и числителем, равным 1).

Система функциональных уравнений задается либо в виде полной системы sys со списком переменных vars, либо в матричном виде с заданием матрицы коэффициентов системы A и вектора свободных членов b (может отсутствовать) с указанием независимой переменной x и параметра case, имеющего значения 'differential', 'difference' или 'qdifference'. Параметр method, задающий метод EG-исключения, может иметь значения 'quasimodular' или 'ordinary'.

Вспомогательные функции

Несколько вспомогательных функций пакета LinearFunctionalSystems представлено ниже:

- MatrixTriangularization(mat, m, n, x, lt) — триангуляция матрицы mat размера m×n с указанием типа lt ('lead' или 'trail');
- CanonicalSystem(shift, sys, vars) или CanonicalSystem(shift, A[, b], x, case) — возвращает систему в каноническом виде (параметр shift задается как 'difference' или 'q-difference', назначение других параметров соответствует указанным выше для других функций);
- ExtendSeries(sol, deg) — расширяет ряд решения sol до расширенного ряда степени deg;
- HomogeneousSystem(homo, sys, vars) или HomogeneousSystem(homo, A[, b], x, case) — преобразует исходную систему в гомогенную с именем homo.
- Properties(sys, vars) или Properties(A[, b], x, case) — возвращает основные свойства системы.

Примеры применения пакета LinearFunctionalSystems

Ниже представлен ряд примеров применения пакета LinearFunctionalSystems, иллюстрирующих его возможности:

```
> with(LinearFunctionalSystems):
sys := [diff(y1(x), x) - y2(x),
diff(y2(x), x) - y3(x) - y4(x),
diff(y3(x), x) - y5(x),
diff(y4(x), x) - 2*y1(x) - 2*x*y2(x) - y5(x),
diff(y5(x), x) - x^2*y1(x) - 2*x*y3(x) - y6(x),
diff(y6(x), x) - x^2*y2(x) + 2*y3(x)]:
vars := [y1(x), y2(x), y3(x), y4(x), y5(x), y6(x)]:
sol:=PolynomialSolution(sys, vars);
```

$$sol := \left[-\_c_2 - \frac{1}{2} x \_c_3, -\frac{1}{2} \_c_3, \_c_1 + x \_c_2 + \frac{1}{2} x^2 \_c_3, -\_c_1 - x \_c_2 - \frac{1}{2} x^2 \_c_3, \_c_2 + x \_c_3, \right.$$
$$\left. \_c_3 - 2 x \_c_1 - x^2 \_c_2 - \frac{1}{2} x^3 \_c_3 \right]$$

```
> IsSolution(sol, sys, vars);
```

true

```
> sol1 := [-_c[1]+x*_c[3], _c[3], -_c[2]+x*_c[1]-x^2*_c[3], _c[2]-x*_c[1]+x^2*_c[3], _c[1]-
2*x*_c[3], -2*_c[3]+2*x*_c[2]-x^2*_c[1]+x^3*_c[3]]:
```

$$sol1 := [-\_c_1 + x\,\_c_3, \_c_3, -\_c_2 + x\,\_c_1 - x^2\,\_c_3, \_c_2 - x\,\_c_1 + x^2\,\_c_3, \_c_1 - 2\,x\,\_c_3,$$
$$-2\,\_c_3 + 2\,x\,\_c_2 - x^2\,\_c_1 + x^3\,\_c_3]$$

```
> IsSolution(sol1, sys, vars);
```

true

```
> AreSameSolution(sol,sol1);
```

true

```
> sol[1] := 1;
```

$$sol_1 := 1$$

```
> IsSolution(sol, sys, vars);
```

$$[0, -\infty, -\infty, 1, 3, -\infty]$$

```
> sol2 := eval(sol1, _c[1]=0);
```

$$sol2 := [x\,\_c_3, \_c_3, -\_c_2 - x^2\,\_c_3, \_c_2 + x^2\,\_c_3, -2\,x\,\_c_3, -2\,\_c_3 + 2\,x\,\_c_2 + x^3\,\_c_3]$$

```
> IsSolution(sol2, sys, vars);
```

true

```
> AreSameSolution(sol1,sol2);
```

false

```
> sys := [-x^2*y2(x) + 2*x^2*y1(x) + 4*y1(x)*x - y1(x + 1)*x^2 - 4*y1(x + 1)*x + 2*y1(x) -
4*y1(x + 1), y2(x + 1) - y1(x)]:
vars:= [y1(x), y2(x)]:
UniversalDenominator(sys, vars);
```

$$\frac{1}{(x+1)^2\,x^2}$$

```
> Properties(sys,vars);
```

res

Множество дополнительных примеров на анализ и решение линейных функциональных систем можно найти в справке по функциям данного пакета.

Что нового мы узнали?

В этом уроке мы научились:

- Применять основные операции и определения линейной алгебры.
- Использовать пакет решения задач линейной алгебры linalg.
- Использовать пакет линейной алгебры LinearAlgebra с алгоритмами NAG.
- Осуществлять интеграцию системы Maple 7 с матричной системой MATLAB.
- Использовать некоторые функции системы MATLAB при работе в Maple 7.
- Применять функции нового пакета LinearFunctionalSystems.

Обзор пакетов специального назначения

- Пакет решения задач оптимизации simplex
- Пакет двумерной геометрии geometry
- Пакет трехмерной геометрии geom3d
- Пакет для работы с алгебраическими кривыми algcurves
- Пакет функций теории графов networks
- Пакет статистических расчетов stats
- Пакет для студентов student
- Пакет работы с тензорами tensor
- Пакет Domains
- Обзор пакетов узкого назначения
- Новые пакеты системы Maple 7

Пакет решения задач линейной оптимизации simplex

Обзор средств пакета

Задачи линейной оптимизации важны как в фундаментальных, так и в прикладных приложениях математики. В пакете simplex имеется небольшой, но достаточно представительный набор функций и определений для решения таких задач:

```
> with(simplex);
```

Warning, the protected names maximize and minimize have been redefined and unprotected

[*basis*, *convexhull*, *cterm*, *define_zero*, *display*, *dual*, *feasible*, *maximize*, *minimize*, *pivot*, *pivoteqn*, *pivotvar*, *ratio*, *setup*, *standardize*]

Приведем краткое назначение этих функций:

- `basis` — возврат списка основных переменных для множества линейных уравнений;
- `convexhull` — вычисление выпуклой оболочки для набора точек;
- `cterm` — задание констант для системы уравнений или неравенств;
- `define_zero` — определение наименьшего значения, принимаемого за ноль (по умолчанию увязано со значением системной переменной `Digits`);
- `display` — вывод системы уравнений или неравенств в матричной форме;
- `dual` — выдача сопряженных выражений;
- `equality` — параметр для функции `convert`, указывающий на эквивалентность;
- `feasible` — выяснение возможности решения заданной задачи:
- `maximize` — вычисление максимума функции;
- `minimize` — вычисление минимума функции;
- `pivot` — создание новой системы уравнений с заданным главным элементом;
- `pivoteqn` — выдача подсистемы уравнений для заданного главного элемента;
- `pivotvar` — выдача переменных с положительными коэффициентами в целевой функции;
- `ratio` — выдача отношений для определения наиболее жесткого ограничения;
- `setup` — задание системы линейных уравнений;
- `standardize` — приведение заданной системы уравнений или неравенств к стандартной форме неравенств типа «меньше или равно».

Функции maximize и minimize

Главными из этих функций являются maximize и minimize, оптимизирующие задачу симплекс-методом. Они записываются в следующих формах:

```
maximize(f, C)
minimize(f, C)
maximize(f , C, vartype)
minimize(f , C, vartype)
maximize(f , C, vartype, 'NewC', 'transform')
minimize(f , C, vartype, 'NewC', 'transform')
```

Здесь f — линейное выражение, C — множество или список условий, vartype — необязательно задаваемый тип переменных NONNEGATIVE или UNRESTRICTED, NewC и transform — имена переменных, которым присваиваются соответственно оптимальное описание и переменные преобразования. Ниже даны примеры применения этих функций:

```
> minimize( x-y, {4*x+2*y <= 10, 3*x+4*y <= 16}, NONNEGATIVE,'NC','vt' );
```

$$\{x = 0, y = 4\}$$

```
> NC;vt;
```

$$\{y = -\frac{1}{4}\_SL2 + 4 - \frac{3}{4}x, \_SL1 = 2 - \frac{5}{2}x + \frac{1}{2}\_SL2\}$$

$$\{\ \}$$

```
> maximize( x+y, {4*x+2*y <= 10, 3*x+4*y <= 16}, NONNEGATIVE );
```

$$\{x = \frac{4}{5}, y = \frac{17}{5}\}$$

Прочие функции пакета simplex

Функция basis(C) возвращает базис для системы линейных уравнений C. Например:

```
> basis( [ x = 2*z+w , z = 2*y - w ] );
```

$$[x, z]$$

Функция convexhull(ps) возвращает выпуклую оболочку множества точек ps:

```
> convexhull({[0,0],[1,1],[2,-1],[1,1/3],[1,1/2]});
```

$$[[0, 0], [2, -1], [1, 1]]$$

Для определения констант для системы линейных уравнений или неравенств служит функция cterm(C):

```
> cterm([2*x+y<=6,7*y-z-3=4]);
```

$$[6, 7]$$

Функция define_zero(C) возвращает ближайшее ненулевое значение, зависящее от установки переменной Digits:

```
> define_zero();
```

$$.1000000000\ 10^{-7}$$

```
> Digits:=40;
```
$$Digits := 40$$
```
> define_zero();
```
$$.1000000000000000000000000000000000000000 \ 10^{-37}$$
```
> define_zero(1*10^(-10));
```
$$.1000000000000000000000000000000000000000 \ 10^{-9}$$

Функция display(C) имеет еще и форму display(C,[x, y, z]). Она задает вывод линейных уравнений и неравенств в матричной форме:

```
> display({2*x+5*y-z<= 0, 2*w-4*y-z<=2});
```
$$\begin{bmatrix} 0 & -4 & 2 & -1 \\ 2 & 5 & 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \\ z \end{bmatrix} \le \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

Функция dual(f, C, y) имеет следующие параметры: f — линейное выражение , C — множество неравенств и y — имя. Эта функция возвращает сопряженное с f выражение:

```
> dual( x-y,{2*x+3*y<=5,3*x+6*y<=15},z);
```
$$15\ z1 + 5\ z2, \{1 \le 3\ z1 + 2\ z2, -1 \le 6\ z1 + 3\ z2\}$$

Функция feasible может быть задана в трех формах:

```
feasible(C)
feasible(C,vartype)
feasible(C,vartype,'NewC','Transform')
```

Здесь параметр vartype может иметь значения NONNEGATIVE или UNRESTRICTED. Эта функция определяет систему как осуществимую или нет:

```
> feasible({2*x+3*y<=5, 3*x+6*y<=15}, NONNEGATIVE);
```
true
```
> feasible({2*x+3*y<=5, 3*x+6*y<=-15}, NONNEGATIVE);
```
false

Если функция возвращает логическое значение true, то заданная система осуществима, а если false — неосуществима, то есть ни при каких значениях переменных не способна удовлетворить записанным неравенствам и равенствам.

Функция pivot(C, x, eqn) конструирует новую систему с заданным главным элементом:

```
> pivot({_SL1=5-4*x-3*y,_SL2=4-3*x-4*y},x,[_SL1=5-4*x-3*y]);
```
$$\{x = -\frac{1}{4}\_SL1 + \frac{5}{4} - \frac{3}{4}y, \_SL2 = \frac{1}{4} + \frac{3}{4}\_SL1 - \frac{7}{4}y\}$$

Функция pivoteqn(C, var) возвращает подсистему для заданного диагонального элемента C:

```
> pivoteqn( {_SL1 = 5-3*x-2*y, _SL2 = 4-2*x-2*y}, x );
```
$$[\_SL1 = 5 - 3\,x - 2\,y]$$

Функция pivotvar(f, List) или pivotvar(f) возвращает список переменных, имеющих положительные коэффициенты в выражении для целевой функции:

```
> pivotvar( x1-2*x2+3*x3-x4);
```

x1

```
> pivotvar( x1 + 2*x3 - 3*x4 , [x4,x3,x1] );
```

x3

Функция ratio(C, x) возвращает список отношений, задающих наиболее жесткие ограничения:

```
> ratio( [_SL1=10-3*x-2*y,_SL2=8-2*x-4*y],x );
```

$$\left[\frac{10}{3}, 4\right]$$

Функция setup может иметь три формы:

```
setup(C)
setup(C, NONNEGATIVE)
setup(C, NONNEGATIVE, 't')
```

Она обеспечивает конструирование множества уравнений с переменными в левой части:

```
> setup( {2*x+3*y<=5,3*x+5*y=15});
```

$$\{\_SL1 = -5 + \frac{1}{3}y, x = -\frac{5}{3}y + 5\}$$

Последняя функция — standartize(C) — конвертирует список уравнений (неравенств) в неравенства типа «меньше или равно»:

```
> standardize({2*x+3*y<=5,3*x+5*y=15});
```

$$\{2x + 3y \le 5, 3x + 5y \le 15, -3x - 5y \le -15\}$$

Пакет планиметрии geometry

Набор функций пакета geometry

Пакет геометрических расчетов geometry в системе Maple 7 получил как бы второе рождение — число его функций по сравнению с версией этого пакета в системе Maple V R5 возросло в несколько раз. Теперь загрузка пакета возвращает весьма внушительный список из более чем 100 функций:

```
> with(geometry);
```

[*Appolonius*, *AreCollinear*, *AreConcurrent*, *AreConcyclic*, *AreConjugate*, *AreHarmonic*, *AreOrthogonal*, *AreParallel*, *ArePerpendicular*, *AreSimilar*, *AreTangent*, *CircleOfSimilitude*, *CrossProduct*, *CrossRatio*, *DefinedAs*, *Equation*, *EulerCircle*, *EulerLine*, *ExteriorAngle*, *ExternalBisector*, *FindAngle*, *GergonnePoint*, *GlideReflection*, *HorizontalCoord*, *HorizontalName*, *InteriorAngle*, *IsEquilateral*,

IsOnCircle, *IsOnLine*, *IsRightTriangle*, *MajorAxis*, *MakeSquare*, *MinorAxis*, *NagelPoint*, *OnSegment*, *ParallelLine*, *PedalTriangle*, *PerpenBisector*, *PerpendicularLine*, *Polar*, *Pole*, *RadicalAxis*, *RadicalCenter*, *RegularPolygon*, *RegularStarPolygon*, *SensedMagnitude*, *SimsonLine*, *SpiralRotation*, *StretchReflection*, *StretchRotation*, *TangentLine*, *VerticalCoord*, *VerticalName*,

altitude, *apothem*, *area*, *asymptotes*, *bisector*, *center*, *centroid*, *circle*, *circumcircle*, *conic*, *convexhull*, *coordinates*, *detail*, *diagonal*, *diameter*, *dilatation*, *directrix*, *distance*, *draw*, *dsegment*, *ellipse*, *excircle*, *expansion*, *foci*, *focus*, *form*, *homology*, *homothety*, *hyperbola*, *incircle*, *inradius*, *intersection*, *inversion*, *line*, *medial*, *median*, *method*, *midpoint*, *orthocenter*, *parabola*, *perimeter*, *point*, *powerpc*, *projection*,

radius, *randpoint*, *reciprocation*, *reflection*, *rotation*, *segment*, *sides*, *similitude*, *slope*, *square*, *stretch*, *tangentpc*, *translation*, *triangle*, *vertex*, *vertices*]

Этот пакет содержит средства расчета основных параметров ряда геометрических объектов. Для каждого объекта возможно задание различных исходных величин, так что пакет охватывает практически все виды классических геометрических расчетов на плоскости. Несомненно, этот пакет заинтересует всех, кто работает в области геометрии и смежных областях.

Обратите внимание на то, что многие функции этого пакета вовсе не рисуют на экране соответствующие фигуры, а лишь выполняют типовые геометрические расчеты. Разумеется, в дальнейшем, используя результаты этих расчетов, можно построить соответствующую фигуру с помощью графических функций.

Пример применения расчетных функций пакета geometry

К сожалению, описание всех функций этого пакета потребует привести справочные данные практически по всей геометрии на плоскости, объем которых намного превышает объем данной книги. Учитывая идентичность идеологии при работе с функциями этого пакета, большинство из которых имеет вполне прозрачные имена (правда, англоязычные), работу с пакетом поясним на примере одной из функций — `circle`. Она позволяет математически задать окружность и определить все ее геометрические параметры. Функция может иметь несколько форм записи. Например, в форме:

```
circle(c, [A, B, C], n, 'centername'=m)
```

она определяет построение окружности, проходящей через три точки `A`, `B` и `C`. Необязательный параметр `n` — список с именами координатных осей. Параметр `'centername'=m` задает имя центра.

В форме `circle(c, [A, B], n, 'centername'=m)` задается окружность, проходящая через две точки `A` и `B`, а в форме `circle(c, [A, rad], n, 'centername'=m)` задается окружность, проходящая через одну точку `A` с заданным (и произвольным) радиусом `rad`

и центром c. Наконец, функция circle в форме circle (c, eqn, n, 'centername'=m) позволяет задать окружность по заданным уравнению eqn и центру c.

Проиллюстрируем применение функции circle на следующих примерах. Зададим характеристические переменные:

```
> _EnvHorizontalName := m: _EnvVerticalName := n:
```

Определим окружность c1, проходящую через три заданные точки A, B и C с указанными после их имен координатами, и найдем координаты центра этой окружности:

```
> circle(c1,[point(A,0,0), point(B,2,0),point(C,1,2)],'centername'=O1):
> center(c1), coordinates(center(c1));
```

$$O1, \left[1, \frac{3}{4}\right]$$

Далее найдем радиус окружности:

```
> radius(c1);
```

$$\frac{1}{16}\sqrt{25}\sqrt{16}$$

и уравнение окружности, заданное в аналитическом виде:

```
> Equation(c1);
```

$$m^2 + n^2 - 2\,m - \frac{3}{2}\,n = 0$$

Наконец, с помощью функции detail получим детальное описание окружности:

```
> detail(c1);
```

name of the object: c1

form of the object: circle2d

name of the center: O1

coordinates of the center: [1, 3/4]

*radius of the circle: 1/16*25^(1/2)*16^(1/2)*

*equation of the circle: m^2+n^2-2*m-3/2*n = 0*

Заинтересованный в таких расчетах читатель может самостоятельно ознакомиться с другими функциями аналогичным образом, тем более, что в справочной системе этого пакета имеется множество примеров работы с его функциями.

Визуализация геометрических объектов с помощью пакета geometry

Одно из важных достоинств пакета geometry — возможность наглядной визуализации различных геометрических понятий, например графической иллюстрации доказательства теорем или геометрических преобразований на плоскости. Проиллюстрируем это на нескольких характерных примерах, заодно показывающих технику работы с рядом функций этого пакета.

Рисунок 16.1 показывает построение из множества окружностей фигуры — кардиоиды. Вопреки обычному построению этой фигуры, используется алгоритм случайного (но удовлетворяющего требованиям построения данной фигуры) выбора положений центров и радиусов окружностей.

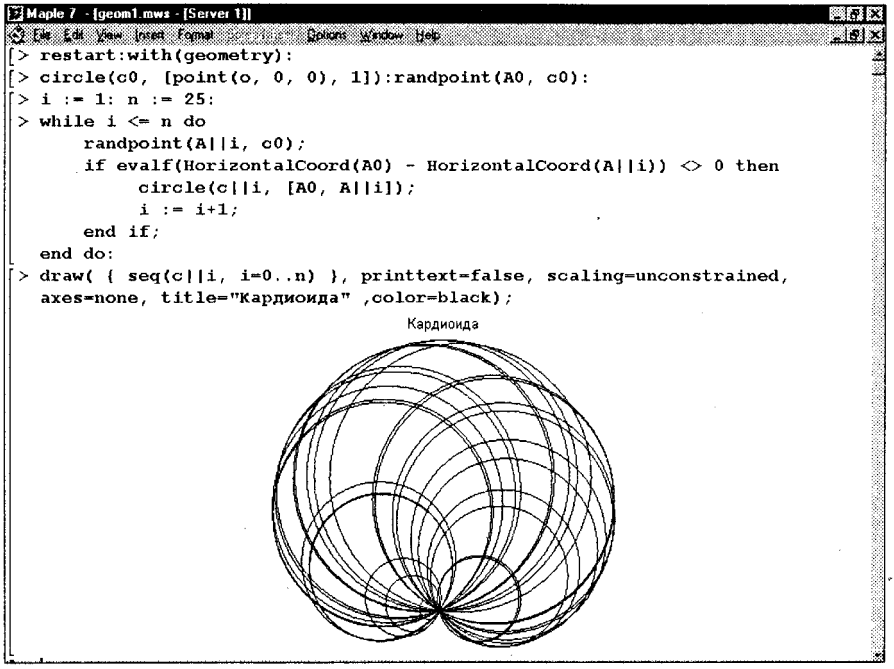


Рис. 16.1. Построение кардиоиды из окружностей

Рисунок 16.2 дает графическую иллюстрацию к одной из теорем Фейербаха. Здесь эффектно используются средства выделения геометрических фигур цветом, что, увы, нельзя оценить по книжной черно-белой иллюстрации.

На следующем рисунке (рис. 16.3) показано построение фигуры, образованной вращением множества квадратов относительно одной из вершин. Это хороший пример применения функций `point`, `square`, `rotation` и `draw` из пакета `geometry`.

Рисунок 16.4 показывает гомологические преобразования квадрата. Заинтересовавшийся читатель может легко разобраться с деталями простого алгоритма этой программы.

ПРИМЕЧАНИЕ Обратите особое внимание на последний параметр в функции **draw**. Он задает построение титульной надписи с заданными шрифтом и размером символов. Сравните титульные надписи на рис. 16.4 и 16.3, где титульная надпись сделана шрифтом, выбранным по умолчанию. Приятно, что в обоих случаях нет преград для использования символов кириллицы и создания надписей на русском языке.

Наконец, на рис. 16.5 показан пример построения трех окружностей, имеющих две общие точки. Обратите внимание на вывод надписей «o», «o1» и «o2», указывающих положение центров окружностей на рисунке.

```
> restart:with(geometry):
> triangle( T, [point(A, 0, 0), point(B, 3, -1), point(C, 6, 2)] ):
incircle( inc, T, 'centername'=II ):
excircle( obj, T, [ex1(I1), ex2(I2), ex3(I3)] ):
EulerCircle( ec, T, 'centername'=X ):
> draw( [ inc(color=yellow), ec(color=plum), ex1(color=green),
  T(color=blue, filled=false) ], filled=true,
  scaling=constrained, title="Иллюстрация к теореме Фейербаха" );
```

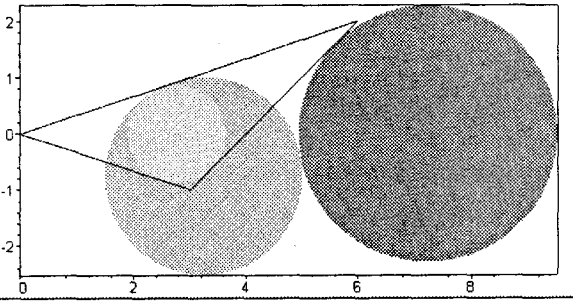


Рис. 16.2. Графическая иллюстрация к теореме Фейербаха

```
> MakeSquare( s||1,[point(A,1,0),point(B,2,0),'diagonal']):n := 30:
> for i to n do rotation( t||i, s1, i*2*Pi/n, counterclockwise, A ) end do:
> square(sq, [point(a, 0, -1), point(b, 2, -1), point(c, 2, 1),
  point(d, 0, 1)]):
> white := [1, 1, 1]: color1 := (100-2.4*x)/100*white:
> draw( [ seq( t||i( color=COLOR( RGB, op( subs(x=i,
  color1) ) ) ), i=1..30), sq(color=black, filled=true)],
   printtext=false, scaling=constrained, axes=none,
  thickness=3, title="Вращение квадрата" );
```

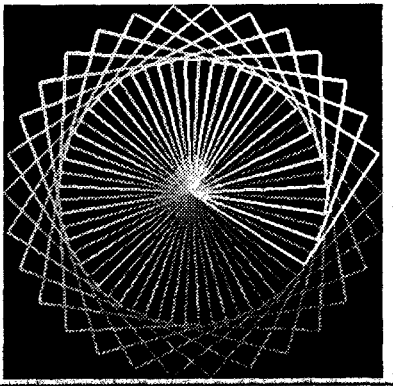


Рис. 16.3. Фигура, полученная вращением квадрата

```
> point(o,0,0): square( sq1, [point(a1, -2, -2),
  point(a2, 2, -2), point(a3, 2, 2),point(a4, -2, 2)] ):
  for i from 2 to 10 do
    homology(sq||i, sq||(i-1), o, Pi/4,
    counterclockwise,1/sqrt(2));
  end do:
  square(sq, [point(aa1, -11/5, -11/5),
  point(aa2,11/5,-11/5),point(aa3,11/5,11/5),point(aa4, -11/5,11/5)]):
> draw( [seq(sq||i,i=1..10),sq(color=COLOR(RGB,1/2,1/2,1/2), filled=true)],
  color='white', thickness=1, scaling=constrained,axes=none,
  printtext=false,title="Гомологические преобразования квадрата",
  titlefont=[TIMES, ROMAN,16] );
```

Гомологические преобразования квадрата

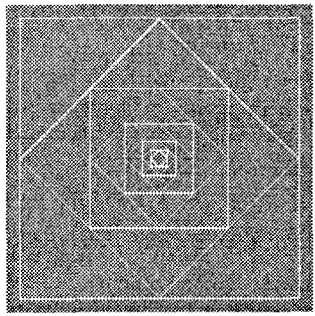


Рис. 16.4. Гомологические преобразования квадрата

```
> with(geometry):_EnvHorizontalName := 'x': _EnvVerticalName := 'y':
  circle(c1, x^2 + y^2 = 1, 'centername' = o1):
  circle(c2, [point(A,3,3), 4], 'centername' = o2):
  CircleOfSimilitude(c,c1,c2,'centername'=o);
```

c

```
> detail(c);
```

name of the object: c
form of the object: circle2d
name of the center: o
coordinates of the center: [-1/5, -1/5]
*radius of the circle: 1/50*128^(1/2)*25^(1/2)*
*equation of the circle: x^2-6/5*y^2+2/5*x+2/5*y = 0*

```
> draw([c,c1,c2],printtext=true,view=[-1.5..3.5,-1.5..3.5],color=black);
```

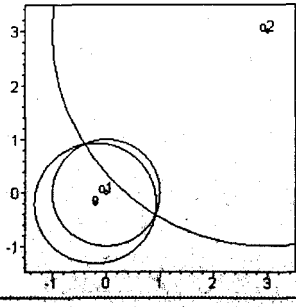


Рис. 16.5. Три окружности, имеющие две общие точки

Множество примеров применения всех функций пакета geometry дано в справочной системе Maple 7. Рекомендуется просмотреть те из них, которые нужны вам.

Пакет стереометрии geom3d

Набор функций пакета geom3d

Помимо существенного расширения пакета geometry в систему Maple 7 введен новый геометрический пакет geom3d. Он предназначен для решения задач в области трехмерной геометрии. При загрузке пакета появляется доступ к большому (свыше 140) числу новых функций:

```
> with(geom3d);
```

[*Archimedean*, *AreCollinear*, *AreConcurrent*, *AreConjugate*, *AreCoplanar*, *AreDistinct*, *AreParallel*, *ArePerpendicular*, *AreSameObjects*, *AreSamePlane*, *AreSkewLines*, *DefinedAs*, *DirectionRatios*, *Equation*, *FindAngle*, *FixedPoint*, *GlideReflect*, *GlideReflection*, *GreatDodecahedron*, *GreatIcosahedron*, *GreatRhombicuboctahedron*, *GreatRhombiicosidodecahedron*,

GreatStellatedDodecahedron, *HarmonicConjugate*, *HexakisIcosahedron*, *HexakisOctahedron*, *InRadius*, *IsArchimedean*, *IsEquilateral*, *IsFacetted*, *IsOnObject*, *IsQuasi*, *IsRegular*, *IsRightTriangle*, *IsStellated*, *IsTangent*, *MidRadius*, *NormalVector*, *OnSegment*, *ParallelVector*, *PentagonalHexacontahedron*, *PentagonalIcositetrahedron*, *PentakisDodecahedron*, *QuasiRegularPolyhedron*,

RadicalCenter, *RadicalLine*, *RadicalPlane*, *RegularPolyhedron*, *RhombicDodecahedron*, *RhombicTriacontahedron*, *RotatoryReflect*, *RotatoryReflection*, *ScrewDisplace*, *ScrewDisplacement*, *SmallRhombicuboctahedron*, *SmallRhombiicosidodecahedron*, *SmallStellatedDodecahedron*, *SnubCube*, *SnubDodecahedron*, *StereographicProjection*, *StretchRotate*, *TangentPlane*,

TetrakisHexahedron, *TrapezoidalHexecontahedron*, *TrapezoidalIcositetrahedron*, *TriakisIcosahedron*, *TriakisOctahedron*, *TriakisTetrahedron*, *TruncatedCuboctahedron*, *TruncatedDodecahedron*, *TruncatedHexahedron*, *TruncatedIcosahedron*, *TruncatedIcosidodecahedron*, *TruncatedOctahedron*, *TruncatedTetrahedron*, *altitude*, *area*, *center*, *centroid*, *circle*, *coordinates*, *cube*,

cuboctahedron, *detail*, *dilate*, *distance*, *dodecahedron*, *draw*, *dsegment*, *duality*, *faces*, *facet*, *form*, *gtetrahedron*, *hexahedron*, *homology*, *homothety*, *icosahedron*, *icosidodecahedron*, *identity*, *incident*, *intersection*, *inverse*, *inversion*, *line*, *midpoint*, *octahedron*, *parallel*, *parallelpiped*, *plane*, *point*, *polar*, *pole*, *powerps*, *projection*, *radius*, *randpoint*, *reflect*, *reflection*, *rotate*, *rotation*, *schlafli*, *segment*, *sides*, *sphere*,

stellate, *tetrahedron*, *tname*, *transform*, *translate*, *translation*, *transprod*, *triangle*, *unit*, *valuesubs*, *vertices*, *volume*, *xcoord*, *xname*, *ycoord*, *yname*, *zcoord*, *zname*]

Функции этого пакета обеспечивают задание и определение характеристик и параметров многих геометрических объектов: точек в пространстве, сегментов, отрезков линий и дуг, линий, плоскостей, треугольников, сфер, регулярных и квазирегулярных полиэдров, полиэдров общего типа и др. Для описания функций этого пакета пришлось бы воспроизвести обширное справочное руководство по стереометрии. В то же время назначение функций ясно из их названия, а характер применения тот же, что для функций описанного выше пакета `geometry`.

Пример применения пакета geom3d

Учитывая сказанное, ограничимся единственным примером применения этого пакета (рис. 16.6).

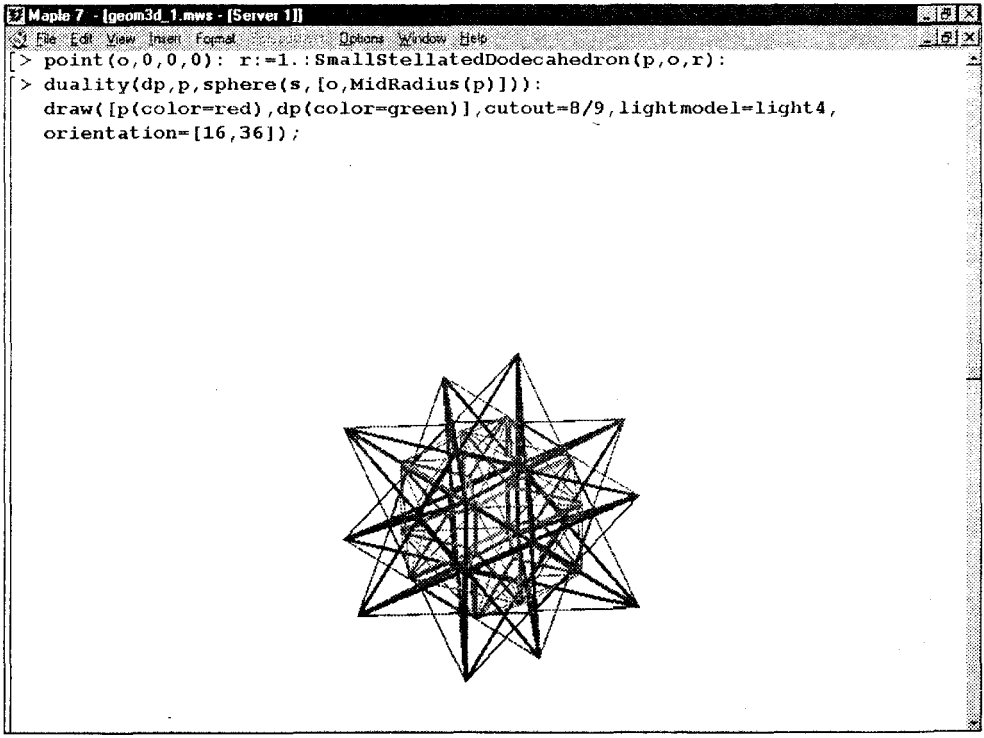


Рис. 16.6. Иллюстрация применения пакета geom3d

Напоминаем, что цель пакета не в построении рисунков геометрических фигур, а в аналитическом представлении объектов в пространстве. Поэтому в обширной базе данных справочной системы по этому пакету вы встретите очень много рисунков.

Пакет для работы с алгебраическими кривыми algcurves

Пакет для работы с алгебраическими кривыми прекрасно дополняет возможности геометрических пакетов. При обращении к нему он дает доступ к полутора десяткам функций:

```
> restart:with(algcurves);
```

[*Weierstrassform*, *differentials*, *genus*, *homogeneous*, *homology*, *integral_basis*, *is_hyperelliptic*, *j_invariant*, *monodromy*, *parametrization*, *periodmatrix*, *plot_knot*, *puiseux*, *singularities*]

Ввиду важности функций пакета и их сравнительно небольшого числа приведем полную форму записи функций и их назначение:

- `Weierstrassform(f,x,y,x0,y0,opt)` — вычисление нормальной формы для эллиптических или гиперболических алгебраических кривых;
- `differentials(f, x, y, opt)` — вычисление голоморфных дифференциалов алгебраических кривых;
- `genus(f,x,y,opt)` — проверка подлинности алгебраической кривой;
- `homogeneous(f,x,y,z)` — создание полинома двух переменных, гомогенного в трех переменных;
- `homology(f, x, y)` — нахождение канонического гомологического базиса по алгоритму Треткоффа;
- `integral_basis(f, x, y, S)` — нахождение интегрального базиса алгебраического поля функции;
- `is_hyperelliptic(f, x, y)` — тестирование кривой на ее принадлежность к гиперболической;
- `j_invariant(f,x,y)` — вычисление *j*-инварианта алгебраической кривой;
- `monodromy(f, x, y, opt)` — вычисляет монодромию алгебраической кривой;
- `parametrization(f,x,y,t)` — нахождение параметризации для кривой с родом (даваемым функцией `genuc`), равным 0;
- `periodmatrix(f, x, y, opt)` — вычисление периодической матрицы кривой;
- `plot_knot(f,x,y,opt)` — построение узла — несамопересекающейся замкнутой кривой в трехмерном евклидовом пространстве;
- `puiseux(f,x=p,y,n,T)` — определение Пуизе-расширения алгебраической функции (может иметь и более простые формы записи);
- `singularities(f,x,y)` — анализ кривой на сингулярность.

Примеры применения пакета algcurves

Приведем примеры применения функций пакета algcurves:

```
> Weierstrassform( (y^2-1)^2+x*(x^2+1)^2 ,x,y,x0,y0);
```

$$\left[y0^2 - 1 - x0 - x0^5, \frac{y^2-1}{x^2+1}, -y, -x0^2, -y0\right]$$

```
> f:=y^3+x^3*y^3+x^4;
```

$$f := y^3 + y^3 x^3 + x^4$$

```
> differentials(f,x,y);
```

$$\left[\frac{x^2\,dx}{y^2(1+x^3)}, \frac{x\,dx}{y(1+x^3)}, \frac{x^2\,dx}{y(1+x^3)}\right]$$

```
> differentials(f,x,y,skip_dx);
```

$$[x^2, y\,x, y\,x^2]$$

```
> nops(%);
```

$$3$$

```
> genus(f,x,y);
```

$$3$$

```
> homogeneous(f,x,y,z);
```

$$x^4 z^2 + y^3 z^3 + y^3 x^3$$

```
> homology(y^3-x^2-1,x,y);
```

table([*cycle*=table([1 = [1,[-1.00000000000 *I*, [1, 3, 2]], 3, [1.00000000000 *I*, [1, 3, 2]]],
2 = [1,[-1.00000000000 *I*, [1, 3, 2]], 2, [1.00000000000 *I*, [1, 3, 2]]],
3 = [1,[-1.00000000000 *I*, [1, 3, 2]], 3, [∞, [1, 3, 2]]],
4 = [1,[-1.00000000000 *I*, [1, 3, 2]], 2, [∞, [1, 3, 2]]],
]), *basepoint* = -.899000000000 *genus* = 1, *canonycalcycle* = table([
*a*1 = [[1, [-1.0000000000 *I* , 1], 3, [1.0000000000 *I*, -1]]],
*b*1 = [[1, [-1.0000000000 *I* , 1], 2, [1.0000000000 *I*, -1]]],

]), *linearcombination* = $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$,
sheets =
[-.609142506531 – 1.05506577036 *I*, -609142506531 + 1.05506577036*I*, 1.2182501306]
])

```
> g := y^3-x*y^2+2*2^(1/2)*y^2+x^2-2*2^(1/2)*x+2+y^6;:
```

$$g := y^3 - x\,y^2 + 2\sqrt{2}\,y^2 + x^2 - 2\sqrt{2}\,x + 2 + y^6$$

```
> integral_basis(g,x,y);
```

$$\left[1, y, y^2, y^3, y^4, \frac{y^2 + y^5 + \sqrt{2}\,y}{-\sqrt{2} + x}\right]$$

```
> is_hyperelliptic(f,x,y);
```

false

```
> f1:=y^2+x^5+1:is_hyperelliptic(f1,x,y);
```

true

```
> j_invariant(g,x,y);
```

$$\frac{71936606821}{38521803} + \frac{3803393323}{38521803}\sqrt{2}$$

```
> monodromy(f1,x,y);
```

```
[-1.47022820183, [-2.42270058708, 2.42270058708], [[-.309016994375- .951056516295I, [[1, 2]]],
    [.809016994375- .587785252292I, [[1, 2]]], [-1., [[1, 2]]],
    [.809016994375+ .587785252292I, [[1, 2]]], [-.309016994375+ .951056516295I, [[1, 2]]],
    [∞, [[1, 2]]]]]
> parametrization(x^4+y^4+a*x^2*y^2+b*y^3,x,y,t);
```

$$\left[-\frac{t^3 b^3}{b^8 + t^2 a b^4 + t^4}, -\frac{t^4 b}{b^8 + t^2 a b^4 + t^4}\right]$$

```
> Z := periodmatrix(f1,x,y,Riemann);
```

$$Z := \begin{bmatrix} .4999999918 + 1.213922064\,I & -.9999999899 - .5257311260\,I \\ -1.000000004 - .5257311066\,I & -.5000000106 + .6881909548\,I \end{bmatrix}$$

Построение алгебраических кривых класса knot

Функция plot_knot позволяет строить одну или несколько алгебраических кривых — узлов. Пример построения целого семейства узлов показан на рис. 16.7.

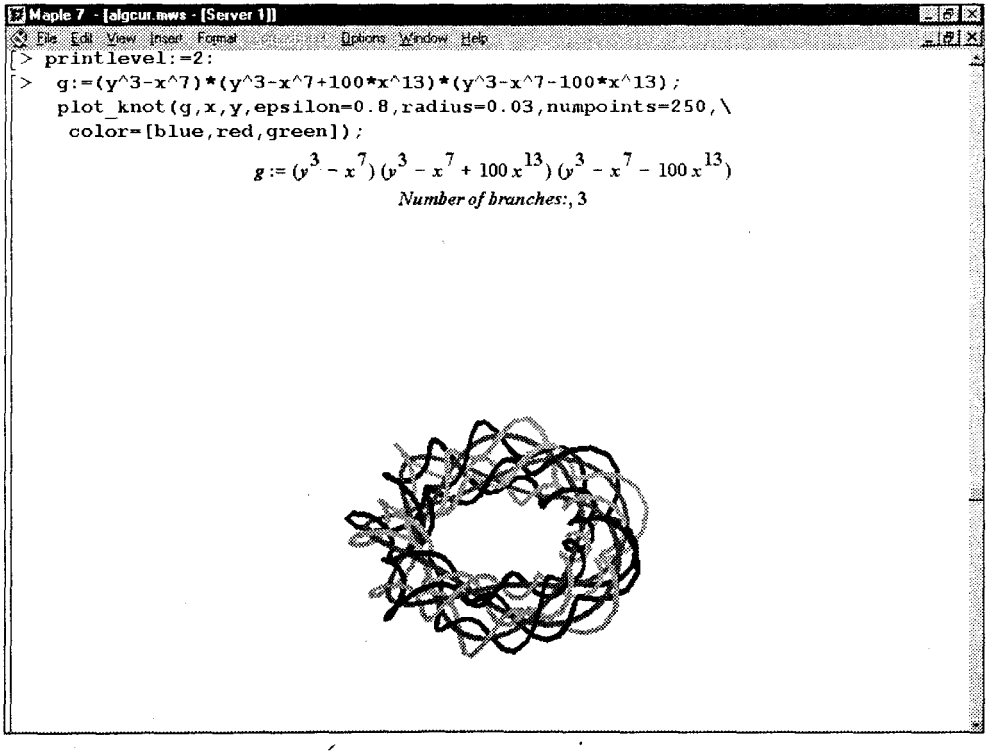


Рис. 16.7. Семейство узлов

Для лучшего обзора таких кривых рекомендуется воспользоваться возможностью вращения трехмерных фигур мышью для уточнения угла, под которым рассматривается фигура — в нашем случае семейство алгебраических кривых. По-

строение на рис. 16.7 выполняется довольно медленно — даже на компьютере с процессором Pentium III 600 МГц оно занимает около минуты.

Новая функция Maple 7 plot_real_curve

В пакет расширения algcurves добавлена новая функция импликативной графики plot_real_curve. Она строит алгебраическую кривую для действительной части полиномиального выражения и записывается в виде:

```
plot_real_curve (p, x, y, opt)
```

Функция имеет следующие параметры:

- p — полиномиальное выражение переменных x и y задающее алгебраическую кривую;
- opt — параметр, который может быть записан в форме приведенных ниже выражений:
  - showArrows = true или false — задает показ стрелок касательных или перпендикулярных к точкам вдоль кривой (по умолчанию false);
  - arrowIntervalStep = posint — задает число точек, пропускаемых до показа очередной пары стрелок (по умолчанию 10);
  - arrowScaleFactor = positive — задает масштаб для длины стрелок (по умолчанию 1);
  - colorOfTangentVector = c — задает цвет касательных стрелок, по умолчанию заданный как зеленный, COLOR(RGB,0,1,0);
  - colorOfNormalVector = c — задает цвет перпендикулярных стрелок, по умолчанию заданный как красный, COLOR(RGB,1,0,0);
  - colorOfCurve = c — задает цвет кривой, по умолчанию заданный как синий, COLOR(RGB, 0, 0, 1);
  - eventTolerance = positive — задает погрешность при представлении сингулярных точек (по умолчанию 0,01);
  - NewtonTolerance = positive — задает погрешность при выполнении ньютоновских итераций в ходе построений.

Функция plot_real_curve вычисляет и строит алгебраическую кривую по точкам и может (при использовании параметра opt) строить стрелки-векторы по касательным и перпендикулярным направлениям к каждой точке или к части точек. Возможно задание разных цветов для кривой и стрелок. Применение функции plot_real_curve показывает рис. 16.8.

При задании построения стрелок касательные стрелки строятся по внешней части кривой и указывают направление ее построения. Перпендикулярные стрелки строятся по внутренней части кривой. Нетрудно заметить, что острие стрелки указывается только для длинных стрелок. Короткие стрелки острия не имеют и отображаются как отрезки прямых линий. Построение стрелок оказывается не

вполне точным, если разнятся масштабы графика по горизонтали и вертикали, что можно подметить при внимательном рассмотрении графика на рис. 16.8, сверху.

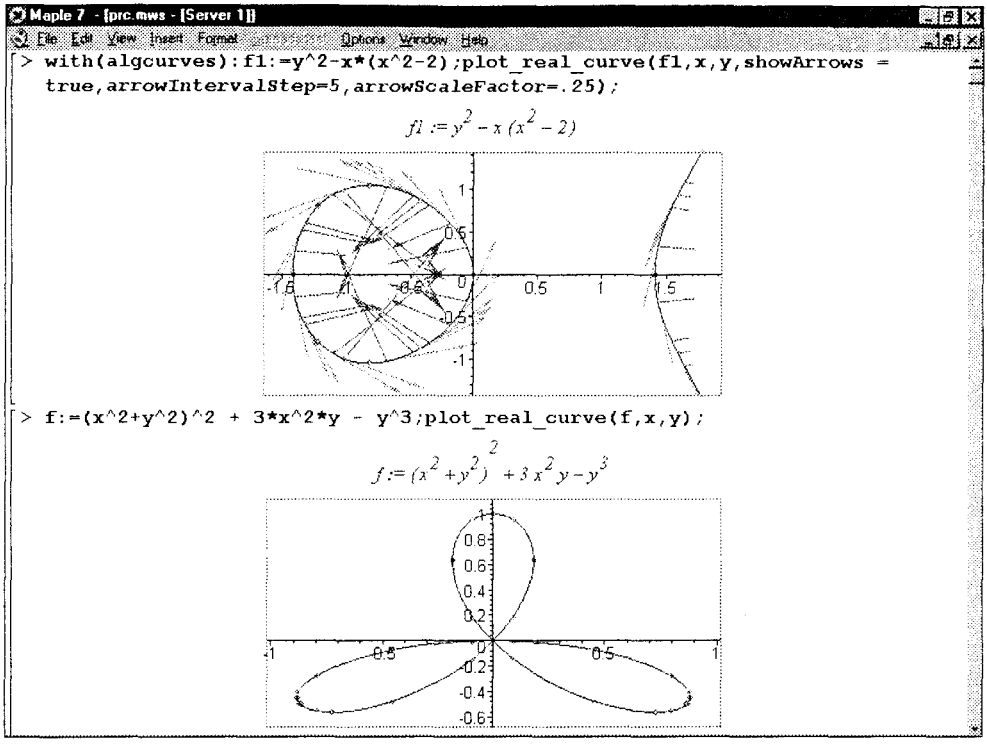


Рис. 16.8. Примеры применения функции plot_real_curve

Пакет функций теории графов networks

Набор функций пакета networks

Графы широко используются при решении многих прикладных и фундаментальных задач. Пользователей, занятых решением таких задач, наверняка порадует пакет networks, содержащий весьма представительный набор функций:

```
> with(networks);
```

Warning, the names diameter, draw and vertices have been redefined

[*acycpoly*, *addedge*, *addvertex*, *adjacency*, *allpairs*, *ancestor*, *arrivals*, *bicomponents*, *charpoly*, *chrompoly*, *complement*, *complete*, *components*, *connect*, *connectivity*, *contract*, *countcuts*, *counttrees*, *cube*, *cycle*, *cyclebase*, *daughter*, *degreeseq*, *delete*, *departures*, *diameter*, *dinic*, *djspantree*, *dodecahedron*, *draw*, *duplicate*, *edges*, *ends*, *eweight*, *flow*, *flowpoly*, *fundcyc*, *getlabel*, *girth*, *graph*, *graphical*, *gsimp*, *gunion*,

head, *icosahedron*, *incidence*, *incident*, *indegree*, *induce*, *isplanar*, *maxdegree*, *mincut*, *mindegree*, *neighbors*, *new*, *octahedron*, *outdegree*, *path*, *petersen*, *random*, *rank*, *rankpoly*, *shortpathtree*, *show*, *shrink*, *span*, *spanpoly*, *spantree*, *tail*, *tetrahedron*, *tuttepoly*, *vdegree*, *vertices*, *void*, *vweight*]

Объективности ради надо отметить, что в Maple 7 из этого пакета удалено несколько второстепенных функций, которые были в версии Maple V R5.

Теория графов используется достаточно широко даже при решении прикладных задач — например, для вычисления оптимальных маршрутов движения железнодорожных составов, наиболее целесообразной раскройки тканей и листов из различных материалов и т. д.

Примеры применения пакета networks

Рассмотрим некоторые избранные функции этого пакета, которые наиболее часто используются при работе с графами. Детали синтаксиса функций можно найти в справочной базе данных Maple 7.

Функции создания графов:

- `new` — создает пустой граф (без ребер и узлов);
- `void` — создает пустой граф (без ребер);
- `duplicate` — создает копию графа;
- `complete` — создает полный граф;
- `random` — возвращает случайный граф;
- `petersen` — создает граф Петерсена.

Функции модификации графов:

- `addedges` — добавляет в граф ребро;
- `addvertex` — добавляет в граф вершины;
- `connect` — соединяет одни заданные вершины с другими;
- `delete` — удаляет из графа ребро или вершину.

Функции контроля структуры графов:

- `draw` — рисует граф;
- `edges` — возвращает список ребер графа;
- `vertices` — возвращает список узлов графа;
- `show` — возвращает таблицу с полной информацией о графе;
- `ends` — возвращает имена вершин графа;
- `head` — возвращает имя вершины, которая является головой ребер;

- `tail` — возвращает имя вершины, которая является хвостом ребер;
- `incidence` — возвращает матрицу инцидентности;
- `adjacency` — возвращает матрицу смежности;
- `eweight` — возвращает веса ребер;
- `vweight` — возвращает веса вершин;
- `isplanar` — упрощает граф, удаляя циклы и повторяющиеся ребра, и проверяет его на планарность (возвращает `true`, если граф оказался планарным, и `false` — в противном случае).

Функции с типовыми возможностями графов:

- `flow` — находит максимальный поток в сети от одной заданной вершины к другой;
- `shortpathtree` — находит кратчайший путь в графе с помощью алгоритма Дейкстры.

Каждая из этих команд имеет одну или несколько синтаксических форм записи. Их можно уточнить с помощью справочной системы. С ее помощью можно ознакомиться и с назначением других функций этого обширного пакета. Проиллюстрируем его применение на нескольких типичных примерах.

На рис. 16.9 показан пример создания графа, имеющего четыре вершины, и графа Петерсона с выводом их графиков графической функцией `draw`.

На рис. 16.10 показан другой пример работы с графами — построение графа функцией `complete` и затем его преобразование путем удаления части вершин. Исходный и преобразованный графы строятся функцией `draw`.

В третьем примере (рис. 16.11) граф формируется по частям — вначале задается пустой граф функцией `new`, а затем с помощью функций `addvertex` и `addedge` в него включаются вершины и ребра. Далее функция `connect` соединяет вершину *a* с вершиной *c*, делая граф замкнутым. Функция `draw` строит сформированный таким образом граф, а функции `head` и `tail` используются для выявления «голов» и «хвостов» графа.

В четвертом примере, представленном на рис. 16.12, показано создание графа `G2` (его изображение было приведено на рис. 16.10) с вычислением для этого графа максимального потока от вершины 1. Обратите внимание, что в параметрах функции `flow`, использованной для этого, заданы две переменные: `eset` — принимает значение множества с *ребрами*, по которым проходит максимальный поток, и `comp` — принимает значение множества, в котором содержатся *вершины*, по которым проходит максимальный поток. Значения этих переменных выведены в области вывода. В заключительной части этого примера показано применение функции `shortpathtree`, ищущей наиболее короткий путь от вершины `1` до других вершин.

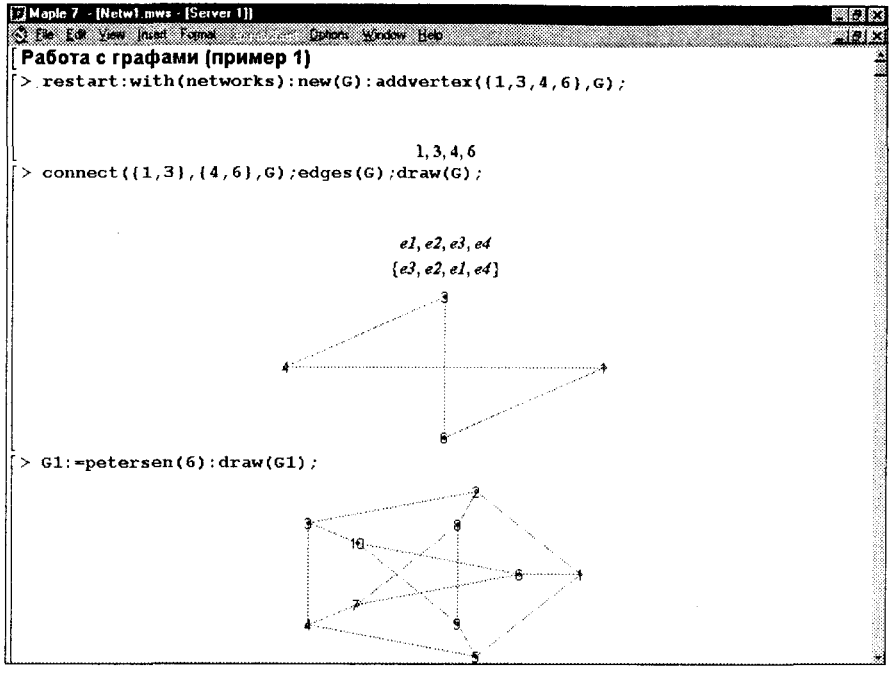


Рис. 16.9. Построение графов

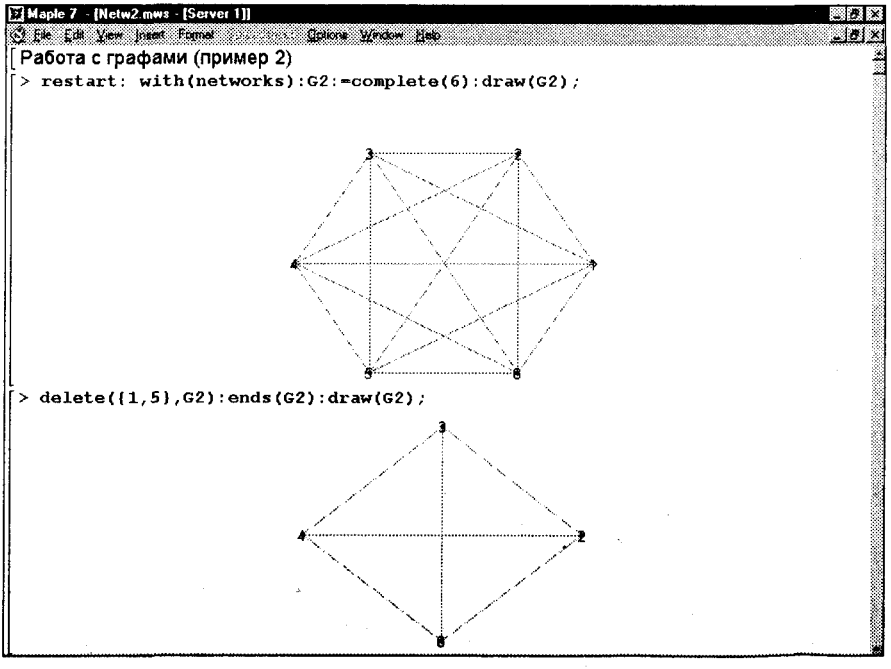


Рис. 16.10. Преобразование графа удалением части вершин

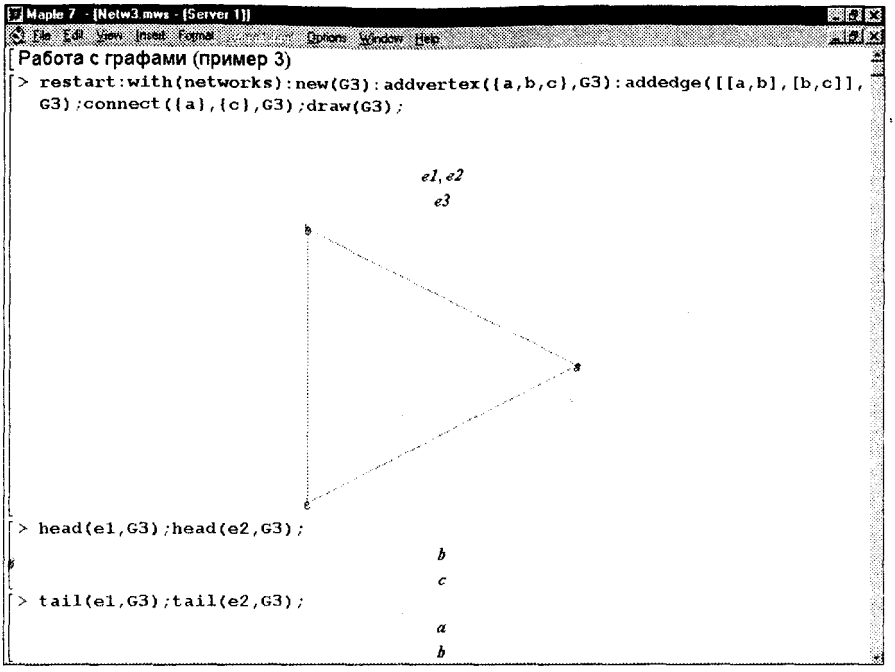


Рис. 16.11. Формирование графа и определение его «голов» и «хвостов»

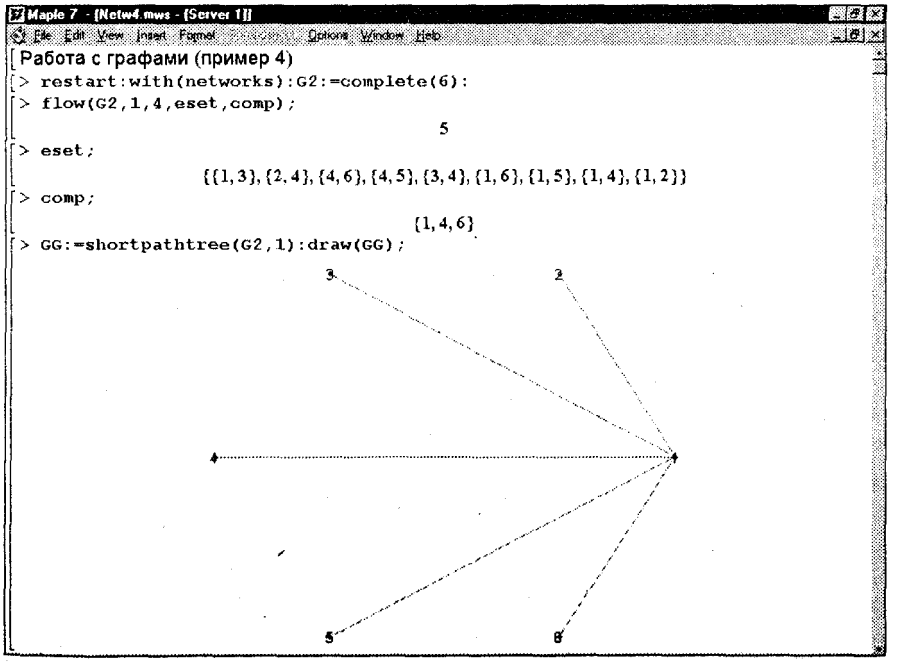


Рис. 16.12. Пример вычисления максимального потока и наиболее коротких путей для заданного графа

Получение информации о графе

Еще один пример, приведенный ниже, иллюстрирует работу функции show, выдающей таблицу с полной информацией о графе, созданном функцией complete:

```
> restart:with(networks):G2:=complete(4):
> show(G2);
```

```
table([_Neighbors = table([1 = {2, 3, 4}, 2 = {1, 3, 4}, 3 = {1, 2, 4}, 4 = {1, 2, 3}]),
    _EdgeIndex = table(symmetric, [(1, 3) = {e2}, (2, 4) = {e5}, (1, 2) = {e1},
    (1, 4) = {e3}, (3, 4) = {e6},
    (2, 3) = {e4}
    ]), _Edges = {e1, e2, e3, e4, e5, e6}, _Bicomponents = _Bicomponents,

    _Tail = table([]), _Countcuts = _Countcuts, _Head = table([]),
    _Eweight = table([e6 = 1, e2 = 1, e5 = 1, e1 = 1, e4 = 1, e3 = 1]),
    _Vweight = table(sparse, []), _Econnectivity = _Econnectivity, _Ends =
    table([e6 = {3, 4}, e2 = {1, 3}, e5 = {2, 4}, e1 = {1, 2}, e4 = {2, 3}, e3 = {1, 4}]),
    _Counttrees = _Counttrees, _Emaxname = 6, _Vertices = {1, 2, 3, 4},

    _Status = {COMPLETE, SIMPLE}
    ])
```

Разумеется, приведенные примеры далеко не исчерпывают всех задач, которые можно решать с применением графов. Но они наглядно демонстрируют, что для большинства пользователей пакет networks превращает графы из окутанного ореолом таинственности модного средства в простой рабочий инструмент.

Пакет статистических расчетов stats

Характеристика пакета stats

Мир математических систем сейчас насыщен статистическими системами, например такими, как Statistica или StatGraphics. Они прекрасно приспособлены для решения задач статистической обработки обширных массивов данных. Тем не менее проведение статистических расчетов в Maple 7 возможно и в ряде случаев весьма целесообразно — например, когда они являются частью исследовательского проекта.

Пакет stats для таких расчетов представлен всего двумя многоцелевыми статистическими функциями:

```
stats[subpackage, function](args)
subpackage[function](args)
```

Однако благодаря специальной форме задания параметров (в частности, в виде *подпакетов* — subpackages) возможно вычисление самых разнообразных статистических функций. Имеются следующие подпакеты:

- anova — вариационный анализ;
- describe — функции распределения вероятности;
- fit — регрессионный анализ;
- random — генерация случайных чисел с различными законами распределения;
- statevalf — вычисление статистических функций и получение оценок для массивов данных;
- statplots — построение графиков статистических функций;
- transform — функции преобразования данных.

Генерация случайных чисел с заданным распределением

Основой этого подпакета является функция random:

```
random[distribution] (quantity,uniform,method)
```

или

```
stats[random, distribution] (quantity,uniform,method)
```

где

- distribution — описание закона распределения случайных чисел;
- quantity — положительное число, указывающее на количество получаемых случайных чисел (по умолчанию 1, возможен параметр 'generator');
- uniform — процедура генерации чисел с равномерным распределением или ключевое слово 'default' (по умолчанию);
- method — указание на один из трех методов ('auto', 'inverse' или 'builtin').

Возможно задание дискретных и непрерывных распределений, например binomiald — дискретное биномиальное распределение, discreteuniform — дискретное равномерное распределение, empirical — дискретное эмпирическое распределение, poisson — дискретное распределение Пуассона, beta — бета-распределение, cauchi — распределение Коши, exponential — экспоненциальное и др. (есть функции практически для всех известных распределений).

Следующие примеры демонстрируют технику получения случайных чисел с заданным законом распределения:

```
> with(stats):Digits:=5:
> stats[random, normald](5);
.79697, -.40654, -.085304, .71297, 1.1119
> stats[random, normald](5,'default','inverse');
-.63631, .72218, -.063423, -1.5123, .84664
> seed:=1:
> uniform_generator:=proc()
>                                global seed;
>                                seed:=irem(seed*11,101);
>                                RETURN(seed/101)
> end;
> random[gamma[3]](5,uniform_generator);
1.1719, 1.5144, 3.3684, 1.9976, 4.8183
```

Графика статистического пакета stats

Статистический пакет stats имеет свою небольшую библиотечку для построения графиков. Она вызывается в следующем виде:

```
stats[statplots, function](args)
```

или

```
statplots[function](args)
```

Вид графика задается описанием function: boxplot, histogram, notchedbox, quantile, quantile2, scatter1d, scatter2d и symmetry. Данные функции обеспечивают построение типовых графиков, иллюстрирующих статистические расчеты. В качестве примера на рис. 16.13 показано задание множества случайных точек и его отображение на плоскости в ограниченном прямоугольником пространстве.

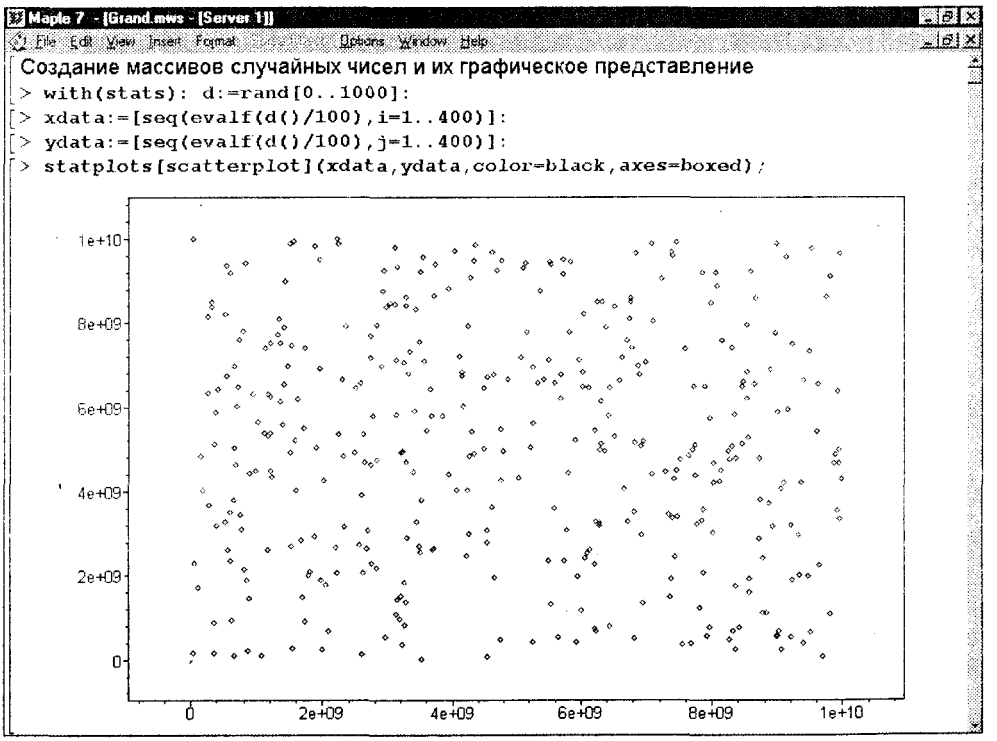


Рис. 16.13. Создание случайных точек и построение их на плоскости

По равномерности распределения точек можно судить о качестве программного генератора случайных чисел, встроенного в Maple 7.

Довольно часто для визулизации вычислений используется построение гистограмм. Для их создания пакет stats имеет функцию histogram:

```
stats[statplots, histogram](data)
statplots[histogram](data)
stats[statplots, histogram[scale](data)
statplots[histogram[scale](data)
```

Здесь data — список данных, scale — число или описатель. Детали применения этой простой функции поясняет рис. 16.14. На нем дан два примера — построение столбцов заданной ширины и высоты и построение гистограммы 100 случайных чисел с нормальным распределением.

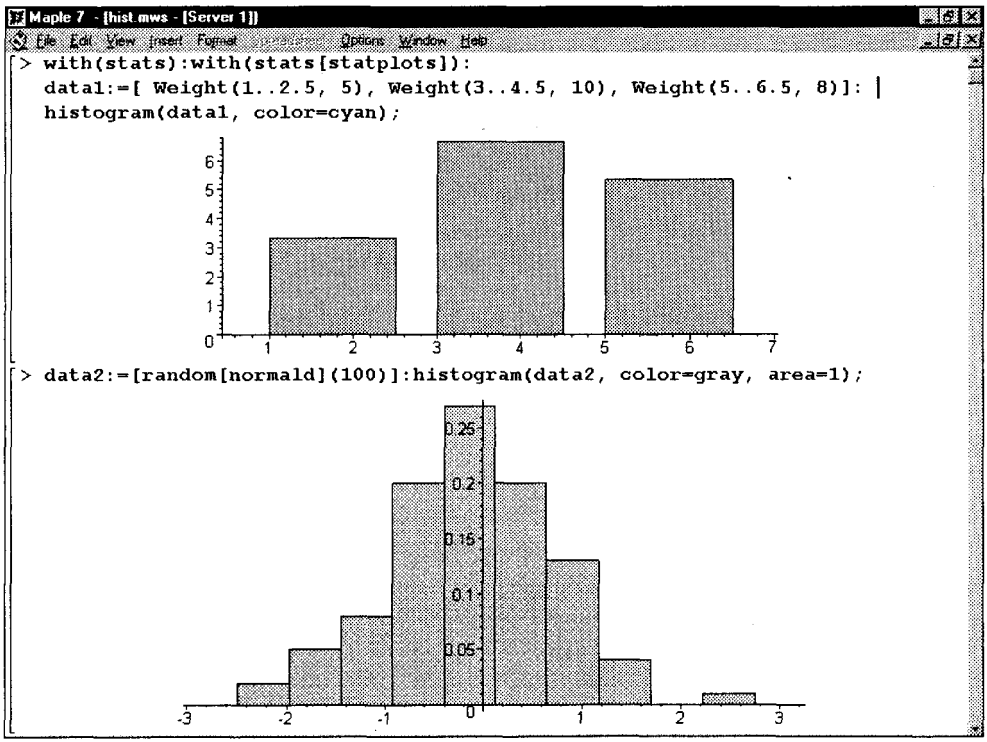


Рис. 16.14. Построение гистограмм

Обратите внимание на то, что для второго примера гистограмма будет несколько меняться от пуска к пуску, так как данные для ее построения генерируются случайным образом.

Регрессионный анализ

Под регрессионным анализом (или просто регрессией) обычно подразумевают нахождение некоторой формальной аналитической зависимости, которая приближенно (по критерию минимума среднеквадратической ошибки) аппроксимирует исходную зависимость. Последняя чаще всего бывает представлена некоторым набором точек (например, полученных в результате эксперимента).

Для проведения регрессионного анализа служит функция fit, которая вызывается следующим образом:

```
stats[fit,leastsquare[vars,eqn,parms]](data)
```

или

```
fit[leastsquare[vars,eqn,parms]](data)
```

где data — список данных, vars — список переменных для представления данных, eqn — уравнение, задающее аппроксимирующую зависимость (по умолчанию линейную), parms — множество параметров, которые будут заменены вычисленными значениями.

На приведенных ниже примерах показано проведение регрессии с помощью функции fit для зависимостей вида $y(x)$:

```
> with(stats):Digits:=5;
```

Digits := 5

```
> fit[leastsquare[[x,y]]]([[1,2,3,4],[3,3.5,3.9,4.6]]);
```

y = 2.4500 + .52000 x

```
> fit[leastsquare[[x,y], y=a*x^2+b*x+c]]([[1,2,3,4],[1.8,4.5,10,16.5]]);
```

2

y = .95000 x + .21000 x + .55000

В первом примере функция регрессии не задана, поэтому реализуется простейшая линейная регрессия, и функция fit возвращает полученное уравнение регрессии для исходных данных, представленных списками координат узловых точек. Это уравнение аппроксимирует данные с наименьшей среднеквадратичной погрешностью. Во втором примере задано приближение исходных данных степенным многочленом второго порядка. Вообще говоря, функция fit обеспечивает приближение любой функцией полиномом.

Рисунок 16.15 показывает регрессию для одних и тех же данных полиномами первой, второй и третьей степени с построением их графиков и точек исходных данных. Нетрудно заметить, что лишь для полинома третьей степени точки исходных данных точно укладываются на кривую полинома, поскольку в этом случае (4 точки) регрессия превращается в полиномиальную аппроксимацию. В других случаях точного попадания точек на линии регрессии нет, но обеспечивается минимум среднеквадратической погрешности для всех точек — следствие реализации метода наименьших квадратов.

Функция fit может обеспечивать регрессию и для функций нескольких переменных. При этом надо просто увеличить размерность массивов исходных данных. В качестве примера ниже приведен пример регрессии для функции двух переменных:

```
>f:=fit[leastsquare[[x,y,z],z=a+b*x+c*y,{a,b,c}]]\
([[1,2,3,5,5],[2,4,6,8,8],[3,5,7,10,Weight(15,2)]]);
```

f := z = 1 + 13/3 x – 7/6 y

```
> fa:=unapply(rhs(f),x,y);
```

fa := (x, y) -> 1 + 13/3 x – 7/6 y

```
z> fa(1.,2.);
```

2.999999999

```
> fa(2,3);
```

37/6

```
> with(stats):
> r1:=rhs(fit[leastsquare[[x,y]]]([[1,2,3,4],[3,3.5,3.9,4.8]]));
```

$$r1 := 2.350000000 + .5800000000\,x$$

```
> r2:=rhs(fit[leastsquare[[x,y],y=a*x^2+b*x+c]]([[1,2,3,4],
  [3,3.5,3.9,4.8]]));
```

$$r2 := .1000000000\,x^2 + .08000000000\,x + 2.850000000$$

```
> r3:=rhs(fit[leastsquare[[x,y],y=a*x^3+b*x^2+c*x+d]]([[1,2,3,4],
  [3,3.5,3.9,4.8]]));
```

$$r3 := .1000000000\,x^3 - .6500000000\,x^2 + 1.750000000\,x + 1.800000000$$

```
> plot([r1,r2,r3,[[1,3],[2,3.5],[3,3.9],[4,4.8]]],x=0..5,2..5,
  style=[line,line,line,point],color=black);
```

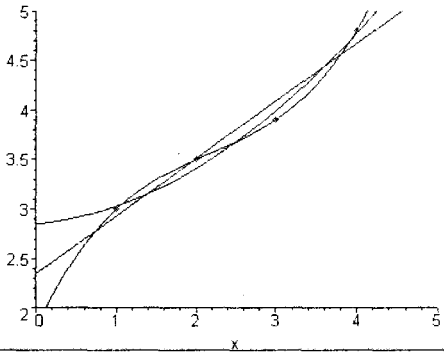


Рис. 16.15. Примеры регрессии полиномоми первой, второй и третьей степени

В данном случае уравнение регрессии задано в виде $z = a + b\,x + c\,y$. Обратите внимание на важный момент в конце этого примера — применение полученной функции регрессии для вычислений или построения ее графика. Прямое применение функции `f` в данном случае невозможно, так как она представлена в невычисляемом формате. Для получения вычисляемого выражения она преобразуется в функцию двух переменных `fa(x,y)` путем отделения правой части выражения для функции `f`. После этого возможно вычисление значений функции `fa(x,y)` для любых заданных значений `x` и `y`.

К сожалению, функция `fit` неприменима для нелинейной регресии. При попытке ее проведения возвращается структура процедуры, но не результат регресии — *см. пример ниже*:

```
> fit[leastsquare[[x,y],y=a*exp(x)+b*x+c],{a,b,c}]([[1,2,3,4],\
[3,3.5,3.9,4.6]]);
```

$$fit_{\substack{leastsquare \\ [x,y],\,y = a\,e^{x} + b\,x + c}, \{a,\,b,\,c\}}([[1, 2, 3, 4], [3, 3.5, 3.9, 4.6]])$$

Для проведения нелинейной регрессии произвольного вида нужно обратиться к средствам нового пакета CurveFitting, включенного в состав Maple 7. Этот пакет был описан в главе 14.

Пакет для студентов student

Функции пакета student

Пакет student — это, несомненно, один из пакетов, наиболее привлекательных для студентов и аспирантов. В нем собраны наиболее распространенные и нужные функции, которые студенты университетов и иных вузов обычно используют на практических занятиях, при подготовке курсовых и дипломных проектов. Набор этих функций, разумеется, не ограничивается «скромными» потребностями студентов — просто это наиболее распространенные функции, в основном относящиеся к математическому анализу. Наряду со студентами эти функции широко используют профессионалы-математики и ученые, применяющие математические методы в своей работе.

В этом пакете имеется почти полсотни функций:

- `D` — дифференциальный оператор;
- `Diff` — инертная форма функции вычисления производной;
- `Doubleint` — инертная форма функции вычисления двойного интеграла;
- `Int` — инертная форма функции интегрирования `int`;
- `Limit` — инертная форма функции вычисления предела `limit`;
- `Lineint` — инертная форма функции вычисления линейного интеграла `lineint`;
- `Point` — тестирование объекта на соответствие типу точки (point);
- `Product` — инертная форма функции вычисления произведения членов последовательности;
- `Sum` — инертная форма функции вычисления суммы членов последовательности;
- `Tripleint` — инертная форма функции вычисления тройного интеграла;
- `changevar` — замена переменной;
- `combine` — объединение подобных членов;
- `completesquare` — вычисление полного квадрата (многочлена);
- `distance` — вычисление расстояния между точками;
- `equate` — создание системы уравнений из списков, таблицы, массивов;
- `extrema` — вычисление экстремума выражения;
- `integrand` — вывод подынтегрального выражения из-под знака инертного интеграла;
- `intercept` — нахождение точки пересечения двух кривых;
- `intparts` — интегрирование по частям;
- `isolate` — выделение подвыражения;
- `leftbox` — графическая иллюстрация интегрирования методом левых прямоугольников;
- `leftsum` — числовое приближение к интегралу левыми прямоугольниками;

- makeproc — преобразование выражения в процедуру Maple;
- maximize — вычисление максимума функции;
- middlebox — графическая иллюстрация интегрирования методом центральных прямоугольников;
- middlesum — числовое приближение к интегралу центральными прямоугольниками;
- midpoint — вычисление средней точки сегмента линии;
- minimize — вычисление минимума функции;
- powsubs — подстановка для множителей выражения;
- rightbox — графическая иллюстрация интегрирования методом правых прямоугольников;
- rightsum — числовое приближение к интегралу правыми прямоугольниками;
- showtangent — график функции и касательной линии;
- simpson — числовое приближение к интегралу по методу Симпсона;
- slope — вычисление и построение касательной к заданной точке функции;
- trapezoid — числовое приближение к интегралу методом трапеций;
- value — вычисление инертные функции.

Функции интегрирования пакета student

В пакетах Maple 7 можно найти множество специальных функций для вычисления интегралов различного типа. Например, в пакете student имеются следующие функции:

- Int(expr,x) — инертная форма вычисления неопределенного интеграла;
- Doubleint(expr,x,y,Domain) — вычисление двойного интеграла по переменным x и y по области Domain;
- Tripleint(expr,x,y,z) — вычисление тройного интеграла;
- intparts(f,u) — интегрирование по частям.

Ниже дан пример применения функции Tripleint пакета student:

```
> Tripleint(f(x,y,z),x,y,z);
```

$$\int\int\int f(x,y,z)\,dx\,dy\,dz$$

```
> Tripleint(x*y*z^2,x=0..2,y=0..3,z=0..5);
```

$$\int_0^5\int_0^3\int_0^2 x\,y\,z^2\,dx\,dy\,dz$$

```
> evalf(%);
 375.0000000
> int(int(int(x*y*z^2,x=0..2),y=0..3),z=0..5);
 375
```

Объективности ради надо отметить, что вычисление тройного интеграла с помощью функции Tripleint занимает много времени (около 20 с на компьютере с процессором Pentium II 350 МГц). Однако тот же результат (см. последний пример) получается за доли секунды при использовании тройного интегрирования с помощью функции int.

Иллюстративная графика пакета student

Пакет student имеет три графические функции для иллюстрации интегрирования методом прямоугольников:

- leftbox(f(x), x=a..b, o) или leftbox(f(x), x=a..b, n, 'shading'=<color>, o);
- rightbox(f(x), x=a..b, o) или rightbox(f(x), x=a..b, n, o);
- middlebox(f(x), x=a..b, o) или middlebox(f(x), x=a..b, n, o);

Здесь f(x) — функция переменной x, x — переменная интегрирования, a — левая граница области интегрирования, b — правая граница области интегрирования, n — число показанных прямоугольников, color — цвет прямоугольников, o — параметры (см. ?plot,options).

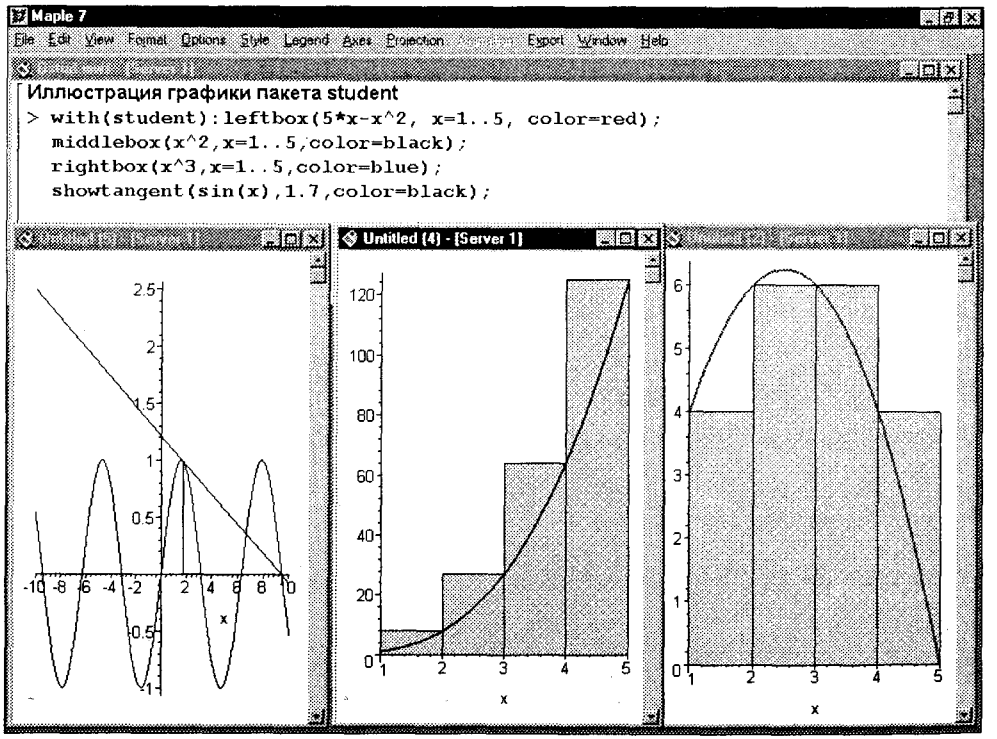


Рис. 16.16. Примеры иллюстративной графики пакета student

В этих функциях прямоугольники строятся соответственно слева, справа и посередине относительно узловых точек функции $f(x)$, график которой также строится. Кроме того, имеется функция для построения касательной к заданной точке $x = a$ для линии, представляющей $f(x)$:

```
showtangent (f(x), x = a)
```

Рисунок 16.16 показывает все эти возможности пакета student. Три вида графиков здесь построены в отдельных окнах.

Графические средства пакета student ограничены. Но они предоставляют как раз те возможности, которые отсутствуют в основных средствах построения графиков.

Пакет для работы с тензорами tensor

Этот пакет впервые появился в реализации Maple V R5. Он дает средства для работы с тензорами и вычислениями, используемыми в общей теории относительности. В нем использован специальный тип данных `tensor_type` в виде таблиц с двумя полями: компонентов и характеристик индексов. Поле компонентов — массив с размерностью, эквивалентной рангу объекта. Поле характеристик индексов задается списком чисел 1 и -1. При этом 1 на i-й позиции означает, что соответствующий индекс контрвариантный, а -1 — что он ковариантный.

Процедура `tensor_type` возвращает логическое значение `true`, если ее первый аргумент удовлетворяет свойствам тензора, и `false`, если он этому свойству не удовлетворяет.

Каждому тензору соответствуют еще две таблицы. Таблица коэффициентов вращения задает коэффициенты вращения Ньюмена—Пенроуза, которые вычисляются функцией `tensor[npspin]` и индексируются именами греческих букв `alpha`, `beta`, `gamma`, `epsilon` и т. д. Другая таблица (компонент кривизны) содержит компоненты кривизны Ньюмена—Пенроуза. Они представлены тремя полями: полем `Phi` в виде массива размерности `(0..2,0..2)` с компонентами Риччи, поле `Psi` с массивом размерности `(0..4)` с компонентами Вейля и полем `R` со скаляром Риччи.

Объявление:

```
> with(tensor);
```

[*Christoffel1*, *Christoffel2*, *Einstein*, *Jacobian*, *Killing_eqns*, *Levi_Civita*, *Lie_diff*, *Ricci*, *Ricciscalar*, *Riemann*, *RiemannF*, *Weyl*, *act*, *antisymmetrize*, *change_basis*, *commutator*, *compare*, *conj*, *connexF*, *contract*, *convertNP*, *cov_diff*, *create*, *d1metric*, *d2metric*, *directional_diff*, *displayGR*, *display_allGR*, *dual*, *entermetric*, *exterior_diff*, *exterior_prod*, *frame*, *geodesic_eqns*, *get_char*, *get_compts*, *get_rank*, *init*, *invars*, *invert*, *lin_com*, *lower*, *npcurve*, *npspin*, *partial_diff*, *permute_indices*, *petrov*, *prod*, *raise*, *symmetrize*, *tensorsGR*, *transform*]

дает доступ к следующим функциям пакета:

- `Christoffel1` — вычисление символов Кристоффеля первого рода;
- `Christoffel2` — вычисление символов Кристоффеля второго рода;
- `Einstein` — возвращение тензора Эйнштейна;
- `display_alJGR` — описывает ненулевые компоненты всех тензоров и параметров, вычисленных командой `tensorsGR` (общая теория относительности);
- `displayGR` — описывает ненулевые компоненты конкретного тензора (общая теория относительности);
- `Jacobian` — Якобиан преобразования координат;
- `Killing_eqns` — вычисление компонентов для уравнений Киллинга (имеет отношение к симметриям пространства);
- `LeviCivita` — вычисление ковариантных и контрвариантных псевдотензоров Леви—Чивита;
- `Lie_diff` — вычисляет производную Ли тензора по отношению к контравариантному векторному полю;
- `Ricci` — тензор Риччи;
- `Ricciscalar` — скаляр Риччи;
- `Riemann` — тензор Римана;
- `RiemannF` — тензор кривизны Римана в жесткой системе отсчета;
- `tensorsGR` — вычисляет тензор кривизны в данной системе координат (общая теория относительности);
- `Weyl` — тензор Вейля;
- `act` — применяет операции к элементам тензора, таблицам вращений или кривизны;
- `antisymmetrize` — антисимметризация тензора по любым индексам;
- `change_basis` — преобразование системы координат;
- `commutator` — коммутатор двух контравариантных векторных полей;
- `compare` — сравнивает два тензора, таблицы вращений или кривизны;
- `conj` — комплексное сопряжение;
- `connexF` — вычисляет связующие коэффициенты для жесткой системы координат;
- `contract` — свертка тензора по парам индексов;
- `convertNP` — преобразует связующие коэффициенты или тензор Римана к формализму Ньюмена—Пенроуза;
- `cov_diff` — ковариантное дифференцирование;
- `create` — создает тензорный объект;
- `d1metric` — первая частная производная метрики;
- `d2metric` — вторая частная производная метрики;
- `directional_diff` — производная по направлению;
- `dual` — осуществляет дуальную операцию над индексами тензора;

- `entermetric` — обеспечивает ввод пользователем координатных переменных и ковариантных компонент метрического тензора;
- `exterior_diff` — внешнее дифференцирование полностью антисимметричного ковариантного тензора;
- `exterior_prod` — внешнее произведение двух ковариантных антисимметричных тензоров;
- `frame` — задает систему координат, которая приводит метрические компоненты к диагональной сигнатурной матрице (с положительными или отрицательными единицами);
- `geodesic_eqns` — уравнение Эйлера—Лагранжа для геодезических кривых;
- `get_char` — возвращает признак (ковариантный/контравариантный) объекта;
- `getcompts` — возвращает компоненты объекта;
- `get_rank` — возвращает ранг объекта;
- `invars` — инварианты тензора кривизны Римана (общая теория относительности);
- `invert` — обращение тензора второго ранга;
- `lincom` — линейная комбинация тензорных объектов;
- `lower` — опускает индексы;
- `npcurve` — компонента кривизны Ньюмена—Пенроуза в формализме Дебевера (общая теория относительности);
- `npspin` — компонент вращения Ньюмена—Пенроуза в формализме Дебевера (общая теория относительности);
- `partial_diff` — частная производная тензора;
- `permute_indices` — перестановка индексов;
- `petrov` — классификация Петрова тензора Вейля;
- `prod` — внутреннее и внешнее тензорные произведения;
- `raise` — поднятие индекса;
- `symmetrize` — симметризация тензора по любым индексам;
- `transform` — преобразование системы координат.

Примеры применения этого пакета можно найти в справочной базе данных системы. Пакет представляет интерес для физиков-теоретиков, работающих в области общей теории относительности и ее приложений. Для них (но не для большинства пользователей) приведенные данные полезны и понятны.

Пакет Domains

Этот небольшой пакет служит для создания *доменов* — таблиц операций для вычислений. При его загрузке появляется сообщение о переопределениях объектов и список из всего лишь шести функций:

```
> restart;with(Domains);
---------- Domains version 1.0 ----------
Initially defined domains are Z and Q the integers and rationals
Abbreviations, e.g. DUP for DenseUnivariatePolynomial, also made
Warning, the protected names Array, Matrix and Vector have been redefined and
unprotected
[Array, Matrix, MatrixInverse, Vector, init, show]
```

Пакет допускает применение следующих конструкций:

```
Domains[domain]                    Domains[evaldomains]
Domains[example]                   Domains[coding]
```

Приведенный ниже пример поясняет создание и использование доменов Q (для рациональных данных) и Z (для целочисленных данных):

```
> Q[`+`](1/2,2/5,3/8);
```

$$\frac{51}{40}$$

```
> Z[Gcd](660,130);
```

$$10$$

Следующая операция показывает, что домен Z является таблицей:

```
> type(Z,table);
```

true

А функция show позволяет вывести полный перечень всех операций, доступных для домена Z:

```
> show(Z,operations);
`     Signatures for constructor Z`
`     note: operations prefixed by  --  are not available`
`      * : (Integers,Z) -> Z`
`      * : (Z,Z*) -> Z`
`      + : (Z,Z*) -> Z`
`      - : (Z,Z) -> Z`
`      - : Z -> Z`
`      0 : Z`
`      1 : Z`
`      < : (Z,Z) -> Boolean`
`      <= : (Z,Z) -> Boolean`
`      <> : (Z,Z) -> Boolean`
`      = : (Z,Z) -> Boolean`
`      > : (Z,Z) -> Boolean`
`      >= : (Z,Z) -> Boolean`
`      Abs : Z -> Z`
`      Characteristic : Integers`
`      Coerce : Integers -> Z`
`      Div : (Z,Z) -> Union(Z,FAIL)`
`      EuclideanNorm : Z -> Integers`
`      Factor : Z -> [Z,[[Z,Integers]*]]`
`      Gcd : Z* -> Z`
`      Gcdex : (Z,Z,Name) -> Z`
`      Gcdex : (Z,Z,Name,Name) -> Z`
`      Input : Expression -> Union(Z,FAIL)`
`      Inv : Z -> Union(Z,FAIL)`
`      Lcm : Z* -> Z`
`      Max : (Z,Z*) -> Z`
```

```
`      Min : (Z,Z*) -> Z`
`      Modp : (Z,Z) -> Z`
`      Mods : (Z,Z) -> Z`
`      ModularHomomorphism : () -> (Z -> Z,Z)`
`      Normal : Z -> Z`
`      Output : Z -> Expression`
`      Powmod : (Z,Integers,Z) -> Z`
`      Prime : Z -> Boolean`
`      Quo : (Z,Z,Name) -> Z`
`      Quo : (Z,Z) -> Z`
`      Random : () -> Z`
`      RelativelyPrime : (Z,Z) -> Boolean`
`      Rem : (Z,Z,Name) -> Z`
`      Rem : (Z,Z) -> Z`
`      Sign : Z -> UNION(1,-1,0)`
`      SmallerEuclideanNorm : (Z,Z) -> Boolean`
`      Sqrfree : Z -> [Z,[[Z,Integers]*]]`
`      Type : Expression -> Boolean`
`      Unit : Z -> Z`
`      UnitNormal : Z -> [Z,Z,Z]`
`      Zero : Z -> Boolean`
`      ^ : (Z,Integers) -> Z`
```

Домены позволяют передавать в качестве параметра процедур набор функций в виде единого целого, что и объясняет название этих объектов. Предполагается, что это может привести к заметному сокращению кодов программ вычислений в будущих реализациях системы Maple. Пока же возможности доменов скорее выглядят как очередная экзотика, чем как реальное средство для оптимизации вычислений. Потребуется время, чтобы показать, что это не так.

Обзор пакетов узкого назначения

Мы уже не раз обращали внимание читателя на выборочный характер описания системы Maple 7 в данной книге. Хотя она и является одной из самых полных книг по данной системе, книга не претендует на роль детального справочника по Maple 7. Более того, такого справочника в виде книги нет и, вероятно, учитывая быстрые темпы модернизации программы, так и не будет. Для подобного описания Maple пришлось бы подготовить многотомное издание, охватывающее практически все области математики.

Учитывая это, мы вынуждены отказаться от попытки описать ряд пакетов специального назначения. Такими пакетами интересуются серьезные специалисты в области математики и им (не без труда, разумеется) под силу разобраться с назначением функций таких пакетов и примерами их применения, приведенными в справочной системе Maple 7. В связи с вышесказанным, мы ограничимся перечислением оставшихся неизученными пакетов.

Пакет функций теории чисел numtheory

В этом обширном пакете собрано 46 функций, относящихся к теории чисел:

```
> with(numtheory);
```

```
Warning, the protected name order has been redefined and unprotected
```

[*GIgcd*, *bigomega*, *cfrac*, *cfracpol*, *cyclotomic*, *divisors*, *factorEQ*, *factorset*, *fermat*, *imagunit*, *index*, *integral_basis*, *invcfrac*, *invphi*, *issqrfree*, *jacobi*, *kronecker*, λ, *legendre*, *mcombine*, *mersenne*, *minkowski*, *mipolys*, *mlog*, *mobius*, *mroot*, *msqrt*, *nearestp*, *nthconver*, *nthdenom*, *nthnumer*, *nthpow*, *order*, *pdexpand*, ϕ, π, *pprimroot*, *primroot*, *quadres*, *rootsunity*, *safeprime*, σ, *sq2factor*, *sum2sqr*, τ, *thue*]

В новой реализации Maple 7 число функций было уменьшено. Большинство функций этого пакета достаточно просты и заинтересовавшийся читатель вполне в состоянии провести их тестирование самостоятельно.

Пакет для работы с p-адическими числами padic

Этот весьма специфический пакет содержит следующие функции для работы с p-адическими числами:

```
> with(padic);
```

[*arccoshp*, *arccosp*, *arccothp*, *arccotp*, *arccschp*, *arccscp*, *arcsechp*, *arcsecp*, *arcsinhp*, *arcsinp*, *arctanhp*, *arctanp*, *coshp*, *cosp*, *cothp*, *cotp*, *cschp*, *cscp*, *evalp*, *expansion*, *expp*, *lcoeffp*, *logp*, *orderp*, *ordp*, *ratvaluep*, *rootp*, *sechp*, *secp*, *sinhp*, *sinp*, *sqrtp*, *tanhp*, *tanp*, *valuep*]

В Maple 7 число функций этого пакета увеличено почти в четыре раза. Однако ввиду специфичности данных функций их изучение мы оставляем за читателем для самостоятельной работы.

Пакет для работы с гауссовыми целыми числами GaussInt

Гауссово целое число — это число вида $a + I*b$, где a и b — любые целые рациональные числа. Таким образом, они образуют решетку всех точек с целыми координатами на плоскости комплексных чисел. Пакет GaussInt содержит достаточно представительный набор функций для работы с этими числами:

```
> with(GaussInt);
```

```
Warning, the name GIgcd has been redefined
```

[*GIbasis*, *GIchrem*, *GIdivisor*, *GIfacpoly*, *GIfacset*, *GIfactor*, *GIfactors*, *GIgcd*, *GIgcdex*, *GIhermite*, *GIissqr*, *GIlcm*, *GImcmbine*, *GInearest*, *GInodiv*, *GInorm*, *GInormal*, *GIorder*, *GIphi*, *GIprime*, *GIquadres*, *GIquo*, *GIrem*, *GIroots*, *GIsieve*, *GIsmith*, *GIsqrfree*, *GIsqrt*, *GIunitnormal*]

Нетрудно заметить, что в этот набор входят уже известные числовые функции, к именам которых добавлены буквы `GI`. Например, функция `GIfactor(c)` раскладывает гауссово число (в том числе комплексное) на простые множители, `GIgcd(c1,c2)` находит наибольший общий делитель гауссовых чисел `c1` и `c2` и т. д. Функции этого пакета достаточно просты, так что ограничимся приведенными примерами.

Гауссовы целые числа в большинстве научно-технических расчетов встречаются крайне редко. Так что этот пакет рассчитан на специалистов-математиков, работающих в области теории чисел.

Пакет алгебры линейных операторов Ore_algebra

Пакет Ore_algebra содержит набор функций алгебры линейных операторов, состав которого виден после обращения к пакету:

```
> with(Ore_algebra);
```

[*Ore_to_DESol*, *Ore_to_RESol*, *Ore_to_diff*, *Ore_to_shift*, *annihilators*, *applyopr*, *diff_algebra*, *poly_algebra*, *qshift_algebra*, *rand_skew_poly*, *shift_algebra*, *skew_algebra*, *skew_elim*, *skew_gcdex*, *skew_pdiv*, *skew_power*, *skew_prem*, *skew_product*]

Этот пакет поддерживает решение задач в области алгебры линейных операторов.

Инструментальный пакет для линейных рекуррентных уравнений LREtools

Этот пакет полезен математикам, часто использующим рекуррентные отношения и формулы. Он дополняет функцию `rsolve` основной библиотеки и содержит следующие функции:

```
> with(LREtools);
```

[*REcontent*, *REcreate*, *REplot*, *REprimpart*, *REreduceorder*, *REtoDE*, *REtodelta*, *REtoproc*, *autodispersion*, *constcoeffsol*, δ, *dispersion*, *divconq*, *firstlin*, *hypergeomsols*, *polysols*, *ratpolysols*, *riccati*, *shift*]

С назначением функций этого пакета можно познакомиться по справочной системе Maple 7.

Пакет функций дифференциальных форм difforms

В пакете дифференциальных форм содержится следующий ряд функций:

```
> with(difforms);
```

[*&^*, *d*, *defform*, *formpart*, *parity*, *scalarpart*, *simpform*, *wdegree*]

Демонстрационные материалы по применению этого пакета входят в поставку Maple 7.

Пакет для работы с рациональными производящими функциями genfunc

В пакете genfunc, предназначенном для работы с производящими функциями, содержатся следующие средства:

```
> with(genfunc);
```

[*rgf_charseq*, *rgf_encode*, *rgf_expand*, *rgf_findrecur*, *rgf_hybrid*, *rgf_norm*, *rgf_pfrac*, *rgf_relate*, *rgf_sequence*, *rgf_simp*, *rgf_term*, *termscale*]

Эти функции представляют специальный интерес для пользователей, работающих в области теории чисел и рациональных функций.

Пакет операций для работы с конечными группами group

Этот пакет содержит довольно представительный набор функций для работы с конечными группами:

```
> with(group);
```

[*DerivedS, LCS, NormalClosure, RandElement, SnConjugates, Sylow, areconjugate, center, centralizer, core, cosets, cosrep, derived, elements, groupmember, grouporder, inter, invperm, isabelian, isnormal, issubgroup, mulperms, normalizer, orbit, parity, permrep, pres, transgroup*]

Функции этого пакета представляют интерес для математиков, работающих в области конечных групп. Но вряд ли они будут полезны большинству пользователей. Тем не менее, наличие таких функций говорит о полноте функциональных возможностей системы Maple 7.

Пакет для работы с симметрией Ли liesymm

В этом пакете, являющемся реализацией алгоритма Харрисона–Эстабрука, имеется ряд функций:

```
> with(liesymm);
```

Warning, the protected name close has been redefined and unprotected

[*&^, &mod, H, Lie, Lrank, TD, annul, autosimp, close, d, depvars, determine, dvalue, extgen, extvars, getcoeff, getform, hasclosure, hook, indepvars, makeforms, mixpar, prolong, reduce, setup, translate, vfix, wcollect, wdegree, wedgeset, wsubs*]

Эти функции достаточно специфичны и могут пригодиться лишь узким специалистам.

Пакет команд для решения уравнений SolveTools

Пакет команд с весьма многообещающим названием SolveTools на самом деле содержит вовсе не средства для решения уравнений, а несколько весьма специфических функций:

```
> with(SolveTools);
```

[*Basis, Complexity, GreaterComplexity, RationalCoefficients, SortByComplexity*]

Функции этого пакета позволяют найти базис выражений, дескрипторы и рациональные коэффициенты. Примеры применения этого пакета очень просты, и с ними несложно ознакомиться. Однако при этом возникает вопрос «Зачем это надо?», который (увы!) остается без ответа.

Пакет для работы с таблицами Spread

Загрузка этого пакета дает средства для работы с таблицами:

```
> with(Spread);
```

[*CopySelection*, *CreateSpreadsheet*, *EvaluateCurrentSelection*, *EvaluateSpreadsheet*, *GetCellFormula*, *GetCellValue*, *GetFormulaeMatrix*, *GetMaxCols*, *GetMaxRows*, *GetSelection*, *GetValuesMatrix*, *InsertMatrixIntoSelection* , *IsStale*, *SetCellFormula*, *SetMatrix*, *SetSelection*]

Функции пакета не имеют самостоятельного значения и призваны поддерживать работу с электронными таблицами, которая уже была подробно описана. Они дают такие средства, как создание в документе шаблона таблиц, проведение операций по заполнению и редактированию ячеек таблиц, копированию содержимого таблиц в буфер памяти и т. д. Назначение функций достаточно очевидно из их составных имен.

Пакет генерации кодов codegen

Пакет codegen представляет собой набор команд, предназначенных для организации взаимодействия системы Maple 7 с другими программными средствами:

```
> with(codegen);
```

[*C*, *GRAD*, *GRADIENT*, *HESSIAN*, *JACOBIAN*, *cost*, *declare*, *dontreturn* , *eqn*, *fortran* , *horner*, *intrep2maple* , *joinprocs* , *makeglobal*, *makeparam* , *makeproc* , *makevoid* , *maple2intrep* , *optimize* , *packargs* , *packlocals*, *packparams* , *prep2trans* , *renamevar* , *split* , *swapargs*]

Этот пакет очень полезен программистам, занимающимся разработкой сложных программных комплексов. Пакет позволяет создавать процедуры на языке Maple 7 и транслировать их в программные модули, записанные на других языках программирования, таких как Фортран или Си.

Пакет создания контекстных меню context

Пакет context служит для создания контекстных меню. Он содержит небольшое число функций:

```
> with(context);
```

[*buildcontext*, *clearlabels*, *defaultcontext*, *display*, *installcontext*, *restoredefault*, *testactions*, *troubleshoot*]

Этот пакет используется довольно редко и в основном пользователями, решающими в среде Maple не вычислительные, а системные задачи. Описание таких задач выходит за рамки данной книги.

Пакет организации многопроцессорной работы process

Этот узкоспециализированный пакет содержит ряд функций по организации работы на нескольких процессорах:

```
> with(process);
```

[*block, exec, fork, kill, pclose, pipe, popen, wait*]

Данные функции представляют интерес для пользователей операционной системы UNIX, так что в проблематику данной книги не входят.

Новые пакеты системы Maple 7

Пакет поддержки вычислений с размерными величинами Units

При выполнении большинства вычислений рекомендуется использовать безразмерные величины. Однако в некоторых областях науки и техники, например в физике, широко используются размерные величины, у которых помимо их значения указываются единицы измерения. Довольно развитую поддержку таких расчетов обеспечивает новый пакет расширения системы Maple 7 — Units. Он содержит следующие функции:

```
> with(Units);
```

[*AddBaseUnit, AddDimension, AddSystem, AddUnit, Converter, GetDimension, GetDimensions, GetSystem, GetSystems, GetUnit, GetUnits, HasDimension, HasSystem, HasUnit, Natural, RemoveDimension, RemoveSystem, Standard, Unit, UseContexts, UseSystem, UsingContexts, UsingSystem*]

Большинство функций этого пакета достаточно просты и даже очевидны. В связи с этим ограничимся несколькими характерными примерами их применения:

```
> convert(4.532, units, N/m^2, (lb*ft/s^2)/ft^2);
```

3.045363395

```
> convert(W, dimensions), convert(W, dimensions, base);
```

$$power, \frac{length^2\ mass}{time^3}$$

```
> with(Units[Standard]):
> distance := 3.5*Unit(ft) + 2.4*Unit(m);
```

$distance := 3.466800000\ [m]$

```
> force := distance*Unit(lb)/Unit(s)^2;
```

$force := 1.572514028\ [N]$

```
> convert(force, units, lbf);
```

$.3535152166\ [lbf]$

```
> V := i*R;
```

$V := i\,R$

```
> eval(V, [i = 2.3*Unit(mA), R = 50.0*Unit(uOmega)]);
```

$.1150000000\ 10^{-6}\ [V]$

```
> convert(%, units, nV );
```

$115.0000000\ [nV]$

Пакет для работы с рядами ортогональных многочленов OrthogonalSeries

Новый пакет для работы с рядами ортогональных многочленов OrthogonalSeries имеет довольно представительный набор функций:

```
> with(OrthogonalSeries);
```

[*Add*, *ApplyOperator*, *ChangeBasis*, *Coefficients*, *ConvertToSum*, *Copy*, *Create*, *Degree*, *Derivate*, *DerivativeRepresentation*, *Evaluate*, *GetInfo*, *Multiply*, *PolynomialMultiply*, *ScalarMultiply*, *SimplifyCoefficients*, *Truncate*]

Поскольку этот пакет представляет интерес в основном для опытных математиков, мы не будем рассматривать его функции (в целом достаточно простые) подробно и ограничимся несколькими примерами. В следующем примере с помощью функции Create создается бесконечный ряд с ортогональным многочленом Эрмита в составе базового выражения ряда:

```
> OrthogonalSeries[Create](u(n),HermiteH(n,x));
```

$$\sum_{n=0}^{\infty} \mathrm{u}(n)\,\mathrm{HermiteH}(n,x)$$

В другом примере показано представление полиномиального выражения в новом базисе с ортогональными многочленами Чебышева с помощью функции ChangeBasis:

```
> OrthogonalSeries[ChangeBasis](1+3*y*x^2+y^3*x,ChebyshevT(n,x),\
ChebyshevU(m,y)) :
```

$$1+\frac{3}{4}\,\mathrm{ChebyshevT}(2,x)\,\mathrm{ChebyshevU}(1,y)+\frac{3}{4}\,\mathrm{ChebyshevU}(1,y)$$
$$+\frac{1}{2}\,\mathrm{ChebyshevT}(1,x)\,\mathrm{ChebyshevU}(1,y)$$

```
> OrthogonalSeries[Evaluate](%) :
```

$$3x^2y+yx+1$$

Обратите внимание на то, что новое выражение после исполнения команды Evaluate приняло вид исходного выражения.

Следующий пример демонстрирует создание ряда на основе ортогональных многочленов Чебышева и его копирование с помощью функции Copy:

```
> S:=Create((-1)^n/n!,ChebyshevT(n,x));
```

$$S := \sum_{n=0}^{\infty} \frac{(-1)^n\,\mathrm{ChebyshevT}(n,x)}{n!}$$

```
> T:=Copy(S);
```

$$T := \sum_{n=0}^{\infty} \frac{(-1)^n\,\mathrm{ChebyshevT}(n,x)}{n!}$$

Вычисление производной от ряда с ортогональными многочленами представлено ниже:

```
> S := Create(u(n),ChebyshevT(n,x));
```

$$S := \sum_{n=0}^{\infty} \mathrm{u}(n)\,\mathrm{ChebyshevT}\,(n, x)$$

```
> Derivate(S,x) ;
```

$$\sum_{n=0}^{\infty} (n+1)\,\mathrm{u}(n+1)\,\mathrm{ChebyshevU}\,(n, x)$$

Еще один пример демонстрирует операцию скалярного умножения ряда с помощью функции ScalarMultiply:

```
> S := Create(n+1,Kravchouk(n,p,q,x));
```

$$S := \sum_{n=0}^{\infty} (n+1)\,\mathrm{Kravchouk}\,(n, p, q, x)$$

```
> ScalarMultiply(alpha,S) ;
```

$$\sum_{n=0}^{\infty} \alpha\,(n+1)\,\mathrm{Kravchouk}\,(n, p, q, x)$$

Приведенные примеры показывают, что применение этого пакета достаточно просто. С деталями (порой довольно многочисленными) применения функций этого пакета можно познакомиться по справке на данный пакет.

Пакет поддержки стандарта MathML

Для представления математической информации на страницах Интернета в последние годы был создан специальный язык MathML. Пока для большинства пользователей MathML — просто «экзотика», но так как наряду с XML его поддерживает World Wide Web Consortium, его вынуждены поддерживать все солидные фирмы — причем не только создающие системы компьютерной математики. Среди них такие крупные корпорации, как Intel, IBM и Microsoft. В Maple 7 предусмотрена новая возможность поддержки стандарта MathML 2.0. Для такой поддержки используются MathML Viewer (см. урок 2) и пакетMathML.

Пакет MathML дает минимальный набор функций для использования языка MathML:

```
> with(MathML);
```

[*Export*, *ExportContent*, *ExportPresentation*, *Import*, *ImportContent*]

В нем всего 5 функций, что позволяет разобрать их достаточно детально. Первые три функции служат для экспорта выражений:

- `Export(expr)` — преобразует Maple-выражение `expr` в параллельное MathML-выражение;
- `ExportContent (expr)` — преобразует Maple-выражение `expr` в MathML-выражение в формате содержания;

- `ExportPresentation (expr)` — преобразует Maple-выражение `expr` в MathML-выражение в формате представления.

Еще две функции служат для импорта строки в формате MathML и его преобразования в Maple-выражение:

`Import(mmlstring)` и `ImportContent(mmlstring)`

Следующий пример наглядно иллюстрирует применение функций пакета расширения MathML для преобразования математического выражения `a*x+b` вначале в запись на MathML, а затем преобразование этой записи `str` в Maple-выражение:

```
> str:=MathML[Export]( a*x + b );
```

str := "<math xmlns='http://www.w3.org/1998/Math/MathML'><semantics><mrow xre\
f='id5'><mrow xref='id3'><mi xref='id1'>a</mi><mo>⁢</mo><mi\
xref='id2'>x</mi></mrow><mo>+</mo><mi xref='id4'>b</mi></mrow><annotati\
on-xml encoding='MathML-Content'><apply id='id5'><plus/><apply id='id3'><ti\
mes/><ci id='id1'>a</ci><ci id='id2'>x</ci></apply><ci id='id4'>b</ci></apply>\

</annotation-xml><annotation encoding='Maple'>a*x+b</annotation></semantics\
></math>"

```
> Import(str);
```

$a\,x + b$

Этот пример показателен тем, что дает представление о виде записей на языке MathML. Нетрудно заметить, что это достаточно громоздкий язык. Функции, представленные выше, автоматизируют процесс составления записей на языке MathML обычных математических выражений, что делает доступной для обычных пользователей подготовку публикации в Интернете, содержащей математические выражения.

Пакет XMLTools

В Maple 7 резко расширены средства интеграции с Интернетом. Это сделано не только на уровне специфических для математических приложений средств, таких как рассмотренный выше пакет MathML, но и на уровне обычных языков сети, таких как HTML и XML.

Пакет XMLTools служит для поддержки средств языка XML. Этот пакет предоставляет весьма внушительное число функций для этого:

```
> with(XMLTools);
```

[*AddAttribute*, *AddChild*, *AttrCont*, *AttributeCount*, *AttributeNames*, *AttributeValue*, *AttributeValueWithDefault*, *Attributes*, *CData*, *CDataData*, *CleanXML*, *Comment*, *CommentText*, *ContentModel*, *ContentModelCount*, *Element*, *ElementName*, *ElementStatistics*, *Equal*, *FirstChild*, *FromString*, *GetAttribute*, *GetChild*, *HasAttribute*, *HasChild*, *IsCData*, *IsComment*, *IsElement*, *IsProcessingInstruction*,

IsTree, *JoinEntities*, *LastChild*, *MakeElement*, *Print*, *PrintToFile*, *PrintToString*, *ProcessAttributes*, *ProcessingInstruction*, *ProcessingInstructionData*, *ProcessingInstructionName*, *ReadFile*, *RemoveAttribute*, *RemoveAttributes*, *RemoveChild*, *RemoveContent*, *SecondChild*, *SelectAttributes*, *SelectContent*, *SelectRemoveAttributes*, *SelectRemoveContent*, *SeparateEntities*, *Serialize*, *StripAttributes*, *StripComments*, *SubsAttribute*, *SubsAttributeName*, *ThirdChild*, *ToString*, *WriteFile*]

Рассмотрение этих средств (как и средств поддержки HTML) далеко выходит за пределы тематики данной книги, хотя многие из них достаточно просты. Поэтому ограничимся единственным примером применения функции `Print` для получения программы на языке XML соответствующей программе задания выражения, рассмотренного в предшествующем разделе:

```
> XMLTools[Print]( % ):
<math xmlns='http://www.w3.org/1998/Math/MathML'>
  <semantics>
    <mrow xref='id5'>
      <mrow xref='id3'>
        <mi xref='id1'>a</mi>
        <mo>&InvisibleTimes;</mo>
        <mi xref='id2'>x</mi>
      </mrow>
      <mo>+</mo>
      <mi xref='id4'>b</mi>
    </mrow>
    <annotation-xml encoding='MathML-Content'>
      <apply id='id5'>
        <plus/>
        <apply id='id3'>
          <times/>
          <ci id='id1'>a</ci>
          <ci id='id2'>x</ci>
        </apply>
        <ci id='id4'>b</ci>
      </apply>
    </annotation-xml>
    <annotation encoding='Maple'>a*x+b</annotation>
  </semantics>
</math>
```

Пакет создания внешних программ ExternalCalling

Пакет ExternalCalling служит для создания внешних программ, записанных на языке Maple (или C++). Состав пакета представлен небольшим числом функций:

- `DefineExternal(fn,extlib)` — используя функцию `define_external`, Maple 7 задает внешнюю функцию `fn` в таблице функций внешних библиотек `extlib`;
- `ExternalLibraryName (basename, precision)` — задает имя `basename` функции и точность вычислений **precision** для функции внешней библиотеки.

Детальное знакомство с этим пакетом мы опускаем. Заинтересованный читатель найдет нужные сведения в справке по этому пакету.

Пакет линейных операторов LinearOperators

Пакет линейных операторов LinearOperators — новый пакет, содержащий средства для работы с линейными операторами. Состав пакета виден из его вызова:

```
> with(LinearOperators);
```

[*Apply*, *DEToOrePoly*, *FactoredAnnihilator*, *FactoredGCRD*, *FactoredMinimalAnnihilator*, *FactoredOrePolyToDE*, *FactoredOrePolyToOrePoly*, *FactoredOrePolyToRE*, *IntegrateSols*, *MinimalAnnihilator*, *OrePolyToDE*, *OrePolyToRE*, *REToOrePoly*, *dAlembertianSolver*]

Набор функций пакета достаточно представителен. Но, поскольку область применения пакета весьма специфична, рекомендуется знакомиться с его возможностями по справкам на его функции и обзорной статье по нему, имеющейся в обзоре новых пакетов расширения Maple 7 (также размещенной в справочной базе данных пакета).

Пакет для работы со случайными объектами RandomTools

Пакет для работы со случайными объектами RandomTools служит для расширения базовых возможностей системы Maple 7 (для большинства пользователей и так вполне достаточных) в части генерации различных случайных объектов, таких как числа различных форматов, векторов, матриц, строковых символов, таблиц и т. д. Они образно названы `Flavor` (в буквальном переводе «букет (вина)»), что подчеркивает возможную сложность структуры создаваемых объектов.

Пакет представлен небольшим числом основных функций:

```
> with(RandomTools);
```

[*AddFlavor*, *Generate*, *GetFlavor*, *GetFlavors*, *HasFlavor*, *RemoveFlavor*]

Однако функции `AddFlavor` и `Generate` могут использоваться с внушительным набором типов случайных объектов:

Choose	complex	exprseq	float	identical
Integer	list	listlist	negative	negint
Nonnegative	nonnegint	nonposint	nonpositive	nonzero
Nonzeroint	polynom	posint	positive	rational
Set	structured	truefalse		

Действие большинства из них вполне очевидно из названий. Основной функцией является функция генерации случайных объектов `Generate(expr)`. Если тип объекта не задан (например, функцией `AddFlavor`), то использование функции `Generate` будет порождать сообщение об ошибке. Примеры применения функций представлены ниже:

```
> with(RandomTools):
Generate(a);
Error, (in RandomTools:-Generate) the flavor `a` does not exist
> AddFlavor(a = rand(1..20)):
Generate(a);
```

```
 2
> Generate(a);
 11
> Generate(alphachar);
```

Error, (in RandomTools:-Generate) the flavor `alphachar` does not exist

```
> AddFlavor(alphachar = proc() [a,b,c,d,e,f][rand(1..6)()] end proc):
GetFlavors();
```

a, *alphachar*, *choose*, *complex*, *exprseq*, *float*, *integer*, *list*, *listlist*, *negative*, *negint*, *nonnegative*, *nonnegint*, *nonposint*, *nonpositive*, *nonzero*, *nonzeroint*, *polynom*, *posint*, *positive*, *rational*, *set*, *string*, *truefalse*

```
> Generate(integer);
-72580330913
> Generate(list(float, 3));
```

$[.00007110693270, .02633073697, .7256143563\ 10^{-7}]$

```
> Generate([a, integer, float]);
```

Error, (in RandomTools:-Generate) the flavor `a` does not exist

```
> Matrix(3, 3, Generate(rational(denominator=10), makeproc));
```

$$\begin{bmatrix} \frac{3}{10} & \frac{-1}{2} & \frac{4}{5} \\ \frac{1}{2} & \frac{-1}{10} & 0 \\ \frac{3}{10} & \frac{-2}{5} & \frac{-1}{2} \end{bmatrix}$$

```
> Matrix(3, 3, Generate(rational(denominator=10), makeproc));
```

$$\begin{bmatrix} \frac{-1}{5} & \frac{-1}{10} & \frac{-9}{10} \\ \frac{1}{2} & \frac{4}{5} & \frac{2}{5} \\ \frac{-3}{10} & \frac{4}{5} & \frac{-9}{10} \end{bmatrix}$$

```
> Vector(4, Generate(complex(integer(range=1..100)), makeproc));
```

$$\begin{bmatrix} 67 + 10\,I \\ 74 + 82\,I \\ 75 + 67\,I \\ 74 + 43\,I \end{bmatrix}$$

Функция `GetFlavor(flvr)` представляют случайный объект (или объекты) в форме процедуры:

```
> AddFlavor(a = rand(1..20)):
Generate(a);
 15
> GetFlavor(a);
```

```
proc()
local t;
global _seed;
    _seed := irem(a×_seed, p);
    t := _seed;

    to concats do _seed := irem(a×_seed, p); t := s×t + _seed end do ;
    irem(t, divisor) + offset
end proc
> GetFlavor(integer);
module() local Defaults; export Main; end module
```

Другая функция `GetFlavors()` возвращает все типы случайных объектов:

```
> GetFlavors();
```

a, alphachar, choose, complex, exprseq, float, integer, list, listlist, negative, negint, nonnegative, nonnegint, nonposint, nonpositive, nonzero, nonzeroint, polynom, posint, positive, rational, set, string, truefalse

Функция `HasFlavor(flvr)` служит для проверки наличия объекта данного типа в списке типов объектов, а функция `RemoveFlavor(flvr)` — для удаления типа объекта. Следующие примеры иллюстрируют применение этих функций:

```
> HasFlavor(a);
true
> RemoveFlavor(a);
> HasFlavor(a);
false
> GetFlavors();
```

alphachar, choose, complex, exprseq, float, integer, list, listlist, negative, negint, nonnegative, nonnegint, nonposint, nonpositive, nonzero, nonzeroint, polynom, posint, positive, rational, set, string, truefalse

Обратите внимание на то, что после уничтожения объекта типа **a** он исчез из списка, выводимого функцией `GetFlavors`. Этот пакет, несмотря на довольно специфические возможности, наверняка будет полезен тем читателям, которые всерьез заняты реализацией «продвинутых» методов Монте-Карло, основанных на моделировании случайных объектов и ситуаций. Он служит серьезным дополнением к пакету статистических расчетов stats, описанному в этом уроке выше.

Пакет для работы со списками ListTools

Новый пакет ListTools содержит ряд полезных команд для работы со списками. Их набор представлен ниже:

```
> with(ListTools);
Warning, the assigned name Group now has a global binding
```

[*BinaryPlace, BinarySearch, Categorize, DotProduct, FindRepetitions, Flatten, FlattenOnce, Group, Interleave, Join, JoinSequence, MakeUnique, Pad, PartialSums, Reverse, Rotate, Sorted, Split, Transpose*]

Пакет содержит набор известных функций для работы со списками, например скалярного умножения списков, их обращения, транспонирования, поворота, объединения и т. д. Ограничимся примером на вычисление скалярного произведения векторов:

```
> L := [0., .84, .91, .14, -.76, -.96, -.28, .66, .99, .41, -.54];
```

$L := [0., .84, .91, .14, -.76, -.96, -.28, .66, .99, .41, -.54]$

```
> M := [1., .54, -.42, -.99, -.65, .28, .96, .75, -.15, -.91, -.84];
```

$M := [1., .54, -.42, -.99, -.65, .28, .96, .75, -.15, -.91, -.84]$

```
> DotProduct(L, L);
```

5.0063

```
> DotProduct(L, M);
```

.3162

Другие новые пакеты, CurveFitting и LinearFunctionalSystem, мы рассмотрели ранее достаточно подробно (см. уроки 14 и 15). В целом надо отметить, что состав пакетов Maple 7 существенно расширен по сравнению с предшествующими версиями системы. В то же время все пакеты, вошедшие в состав Maple 6, сохранены в новой версии программы — Maple 7, что гарантирует полную совместимость с ней. Практически это было подтверждено проверкой всех (а их многие сотни) примеров применения системы Maple 6 применительно к новой версии Maple 7.

Что нового мы узнали?

В этом уроке мы научились:

- Применять пакет решения задач оптимизации simplex.
- Использовать пакет двумерной геометрии geometry.
- Избранно применять пакет трехмерной геометрии geom3d.
- Использовать пакет для работы с алгебраическими кривыми algcurves.
- Строить и модернизировать графы с помощью пакета networks.
- Использовать возможности пакета статистических расчетов stats.
- Применять пакет для студентов student.
- Использовать средства поддержки MathML.
- Использовать средства ряда новых пакетов Maple 7.

Примеры решения научно-технических задач

- ❒ Небольшое введение
- ❒ Выбор аппроксимации для сложной функции
- ❒ Моделирование физических явлений
- ❒ Моделирование и расчет электронных схем

Небольшое введение

Выше при изложении данного учебного курса приводились многие сотни примеров применения системы Maple 7. При этом намеренно подбирались достаточно простые примеры, занимающие немного места и не требующие чрезмерных ухищрений для решения.

Многие читатели полагают, что системы компьютерной математики хорошо работают на таких простых примерах, но от них мало толку при решении реальных задач математики, физики или радиоэлектроники. Это, конечно, заблуждение. Дело просто в том, что при решении таких задач руководящая роль пользователя сильно возрастает. Вы должны понимать, что не Maple 7 решает вашу задачу, а вы! И система Maple 7 лишь помогает в этом трудном деле. Так что при неудачах в решении своих специфических задач следует прежде всего пенять на себя и на свое незнание возможностей системы Maple 7, а вовсе не на свою помощницу.

В том, что Maple можно успешно использовать при решении вполне конкретных научных и практических задач, призваны убедить примеры, приведенные ниже. Разумеется, и их нельзя отнести к таким сложнейшим задачам, как проектирование ядерного реактора или расчет траектории полета космического корабля, — не стоит забывать, что такие расчеты делают на суперкомпьютерах, а не на домашнем компьютере, который стоит перед вами. И объем материалов по сопровождению и результатам таких расчетов многократно превосходит объем всей этой книги. Тем не менее в этом уроке вы встретите решение вполне реальных и полезных задач в области математики, физики и радиоэлектроники. Почему не в механике, гидродинамике или в оптике? Да потому, как верно сказал наш народный пророк Козьма Прутков: «нельзя объять необъятное». Приведенные примеры отчасти обусловлены личными пристрастиями автора; но они полезны каждому пользователю, желающему всерьез оценить возможности Maple 7.

Описанные в этом уроке задачи являются реальными документами, созданными и отлаженными в среде Maple 7 и лишь затем перенесенными в рукопись книги. Так что они заодно служат примерами того, как надо оформлять такие документы. В то же время от некоторых «излишеств» оформления (например, закрывающихся и открывающихся секций) мы отказались, дабы не усложнять описание документов явно второстепенными деталями. Начнем этот урок с решения весьма актуальной для многих областей применения математики задачи — **аппроксимации сложной функции.**

Выбор аппроксимации для сложной функции

Задание исходной функции и построение ее графика

Трудно представить себе область более широкую и почитаемую, чем аппроксимация различных функциональных зависимостей. С получения простой аппроксимации сложной зависимости нередко начинаются (а часто и заканчиваются) научные исследования во многих областях как прикладной, так и фундаментальной науки. Покажем возможности в этом системы Maple 7 на одном из примеров, давно помещенных в библиотеку пользователей системы Maple V R2, и переработанных для Maple 7.

Воспользуемся возможностями пакета numapprox, для чего прежде всего подключим его:

```
> restart:with(numapprox);
```

[*chebdeg*, *chebmult*, *chebpade*, *chebsort*, *chebyshev*, *confracform*, *hermite_pade*, *hornerform*, *infnorm*, *laurent*, *minimax*, *pade*, *remez*]

Будем искать приемлемую аппроксимацию для следующей, отнюдь не простой, тестовой функции:

```
> f := x -> int(1/GAMMA(t), t=0..x ) / x^2;
```

$$f := x \to \frac{\int_0^x \frac{1}{\Gamma(t)}\,dt}{x^2}$$

```
> plot(f,0..4,color=black);
```

График этой функции представлен на рис. 17.1. С первого взгляда — это простой график, но тут как раз тот случай, когда простота обманчива. Вы сразу заметите, что график строится необычно медленно, поскольку в каждой из множества его точек системе Maple 7 приходится вычислять значение интеграла с подынтегральной функцией, содержащей довольно каверзную гамма-функцию. И делает это Maple 7 по сложному и медленному алгоритму адаптивного численного интегрирования.

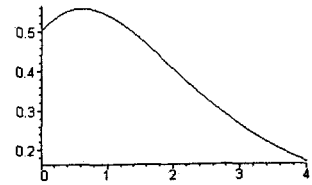


Рис. 17.1. График аппроксимируемой функции

Итак, вычисление $f(x)$ по ее интегральному представлению совершенно не эффективно. Наша цель состоит в разработке процедуры вычислений, которая дала бы 6 точных цифр результата в интервале [0..4] и требовала, по возможности, наименьшего числа арифметических операций для каждого вычисления. Втайне не вредно помечтать о том, чтобы после аппроксимации время вычислений уменьшилось бы хотя в несколько раз. Что получится на деле, вы увидите чуть позже. А пока войдем в дебри аппроксимации.

Аппроксимации рядом Тейлора

Начнем с аппроксимации функции хорошо известным рядом Тейлора степени 8 относительно середины интервала (точки с x=2):

```
> s := map(evalf, taylor(f(x), x=2, 9));
```

$$s := .4065945998 - .1565945998(x-2) + .00209790791(x-2)^2 + .01762626393(x-2)^3 - .006207547150(x-2)^4 + .0005733566(x-2)^5 + .00024331163(x-2)^6 - .00010010532(x-2)^7 + .00001414212(x-2)^8 + O((x-2)^9)$$

```
> TaylorApprox := convert(s, polynom):
```

Такой ряд позволяет использовать для вычислений только арифметические действия, что само по себе здорово! Для удобства преобразуем аппроксимацию в функцию, чтобы она соответствовала форме, указанной для первоначальной функции $f(x)$. Тогда мы сможем построить график кривой ошибок для аппроксимации полиномом Тейлора:

```
> TaylorApprox := unapply(TaylorApprox, x);
```

$$TaylorApprox := x \rightarrow .7197837994 - .1565945998x + .00209790791(x-2)^2 + .01762626393(x-2)^3 - .006207547150(x-2)^4 + .0005733566(x-2)^5 + .00024331163(x-2)^6 - .00010010532(x-2)^7 + .00001414212(x-2)^8$$

Кривая ошибок для аппроксимации полиномом Тейлора строится командой:

```
> plot(f - TaylorApprox, 0..4,color=black);
```

и имеет вид, представленный на рис. 17.2. Эта кривая нас, прямо скажем, не слишком радует, поскольку погрешность в сотни раз превышает заданную.

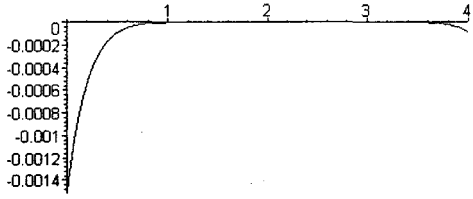


Рис. 17.2. Кривая погрешности при аппроксимации рядом Тейлора

Типичное свойство аппроксимации рядом Тейлора состоит в том, что ошибка мала вблизи точки разложения и велика вдали от нее. В данном случае самая боль-

шая ошибка имеет место в левой оконечной точке. Чтобы вычислить значение ошибки в точке $x = 0$, что ведет к делению на нуль (см. определение для $f(x)$), мы должны использовать значение предела:

```
> maxTaylorError := abs( limit(f(x), x=0) - TaylorApprox(0) );
```

maxTaylorError := .0015029620

Итак, в самом начале наших попыток мы потерпели полное фиаско. Но отчаиваться не стоит, ибо, как говорят, «даже у хорошей хозяйки первый блин — комом».

Паде-аппроксимация

Теперь опробуем рациональную аппроксимацию Паде (Pade) функции $f(x)$ степени (4,4). Приближения по этому разложению будут аппроксимировать функцию более точно, и потому ошибки округления в вычислениях станут более заметными. Поэтому зададим еще два дополнительных знака для точности вычислений.

```
> Digits := 12:
> s := map(evalf, taylor(f(x), x=2, 9)):
> PadeApprox := pade(s, x=2, [4,4]);
```

$$PadeApprox := (.341034780922 + .0327799093746x - .0061278352789\,(x-2)^2 + .00452991086617(x-2)^3 - .000431506275874(x-2)^4)/(.068484878062 + .465757560969x + .159149617732(x-2)^2 + .0266813702995(x-2)^3 + .00346967814937(x-2)^4)$$

```
> PadeApprox := unapply(PadeApprox, x):
```

Кривая ошибки для интервала [0, 4] строится командой:

```
> plot(f - PadeApprox, 0..4,color=black);
```

и имеет вид, показанный на рис. 17.3.

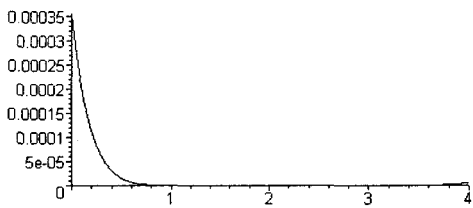


Рис. 17.3. Кривая погрешности при Паде-аппроксимации степени (4,4)

Как и при аппроксимации рядом Тейлора, ошибка здесь мала вблизи точки разложения и велика вдали от нее. Мы снова видим из графика, что для указанной функции, самая большая ошибка — в левой оконечной точке. Однако максимальная ошибка в Паде-аппроксимации уже на порядок меньше, чем при аппроксимации полиномом Тейлора:

```
> maxPadeError := abs(limit(f(x), x=0) - PadeApprox(0));
```

maxPadeError := .000353777214

Это успех, показывающий, что мы на верном пути. Но пока погрешность остается слишком большой по сравнению с заданной.

Аппроксимация полиномами Чебышева

Знатоки техники аппроксимации знают, что лучшие приближения на заданном интервале могут быть получены при использовании разложения в ряд Чебышева. Это связано с тем, что ортогональные полиномы Чебышева позволяют получить аппроксимацию, погрешность которой в заданном диапазоне изменения аргумента распределена более равномерно, чем в предшествующих случаях. Выбросы погрешности на краях интервала аппроксимации в этом случае исключены.

Разложим функцию $f(x)$ на [0, 4] в ряд Чебышева с точностью $1*10^{-8}$. Это означает, что все члены с коэффициентами меньше чем эта величина, будут опущены. Такая точность обеспечивается полиномом 13 степени:

```
> evalf( limit(f(x), x=0) );
.500000000000
> fproc := proc(x) if x=0 then 0.5 else evalf(f(x)) fi end:
> ChebApprox := chebyshev(fproc, x=0..4, 1E-8);
```

$$ChebApprox := .379206274272T\left(0,\frac{1}{2}x-1\right)-.202632813998T\left(1,\frac{1}{2}x-1\right)$$

$$-.0369064836430T\left(2,\frac{1}{2}x-1\right)+.0370131431541T\left(3,\frac{1}{2}x-1\right)$$

$$-.00888944143050T\left(4,\frac{1}{2}x-1\right)-.000149789336636T\left(5,\frac{1}{2}x-1\right)$$

$$+.000642974620794T\left(6,\frac{1}{2}x-1\right)-.000170677949427T\left(7,\frac{1}{2}x-1\right)$$

$$+.0000126917283881T\left(8,\frac{1}{2}x-1\right)+.43987492873810^{-5}T\left(9,\frac{1}{2}x-1\right)$$

$$-.15628413987610^{-5}T\left(10,\frac{1}{2}x-1\right)+.20498054066410^{-6}T\left(11,\frac{1}{2}x-1\right)$$

$$+.45625427777810^{-8}T\left(12,\frac{1}{2}x-1\right)-.69432395526110^{-8}T\left(13,\frac{1}{2}x-1\right)$$

Можно проверить для этого примера, что кривая ошибки при аппроксимации рядом Чебышева колеблется. Поскольку ряд Чебышева был оборван на члене 8-й степени (как и полином ряда Тейлора), то максимальная ошибка оказалась равной приблизительно $0,6*10^{-5}$. Эта величина уже на два порядка меньше, чем ошибка при Паде-аппроксимации, вычисленная выше. Но все же немного не дотягивает до наших требований.

Для последующих вычислений полезно заметить, что мы можем использовать процедуру для нахождения численных значений $f(x)$, которая будет намного эффективнее, чем прямое определение, которое требует численного интегрирования для каждого значения x. А именно определим процедуру численной оценки, основанную на разложении в ряд Чебышева степени 13, так как максимальная ошибка

при такой аппроксимации меньше чем 10^{-8}, и обеспечивает для нашей цели достаточную точность. Мы определим полином Чебышева T(x) из пакета orthopoly и затем для эффективной оценки преобразуем его в форму Горнера:

```
> F := hornerform( eval(subs(T=orthopoly[T], ChebApprox)) );
> F := unapply(F, x):
```

$$F := x \to .500001 + (.192405 + (-.163970 + (-.0083867 + (.0277086 + (-.00593183 + (-.00132727 + (.000910060 + (-.000180351 + (.576869\,10^{-5} + (.448884\,10^{-5} + (-.990278\,10^{-6} + (.925434\,10^{-7} - .347162\,10^{-8}\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x$$

Схема Горнера минимизирует число арифметических операций, заменяя операции возведения в степень операциями последовательного умножения.

Аппроксимация Чебышева–Паде

Теперь рассмотрим еще более точную рациональную аппроксимацию Чебышева-Паде. Это такая рациональная функция r [m, n] (x) с числителем степени m и знаменателем степени n такой же, как и для разложения в ряд Чебышева. Функция r [m, n] (x) согласуется с разложением в ряд Чебышева f(x) членом степени m+n. Мы вычислим аппроксимацию Чебышева-Паде степени (4,4), подобную обычной Паде-аппроксимации, успешно выполненной ранее:

```
> ChebPadeApprox := chebpade(F, 0..4, [4,4]);
```

$$ChebPadeApprox := x \to \left(.285648384503\mathrm{T}\left(0,\frac{1}{2}x-1\right)+.0896033645410\mathrm{T}\left(1,\frac{1}{2}x-1\right)\right.$$
$$-.00626546513721\mathrm{T}\left(2,\frac{1}{2}x-1\right)+.00537846708321\mathrm{T}\left(3,\frac{1}{2}x-1\right)$$
$$\left.-.000414939086957\mathrm{T}\left(4,\frac{1}{2}x-1\right)\right)\Big/\left(\mathrm{T}\left(0,\frac{1}{2}x-1\right)+.879308989228\mathrm{T}\left(1,\frac{1}{2}x-1\right)\right.$$
$$+.289575811178\mathrm{T}\left(2,\frac{1}{2}x-1\right)+.0487963355059\mathrm{T}\left(3,\frac{1}{2}x-1\right)$$
$$\left.+.00650272206660\mathrm{T}\left(4,\frac{1}{2}x-1\right)\right)$$

Построим кривую ошибок:

```
> with(orthopoly, T):
> plot(F - ChebPadeApprox, 0..4,color=black);
```

Она представлена на рис. 17.4.

Максимальная ошибка и на этот раз имеет место в левой оконечной точке. Величина максимальной ошибки несколько меньше, чем ошибка при аппроксимации рядом Чебышева. Главное преимущество представления в виде рациональной функции — высокая эффективность вычислений, которая может быть достигнута преобразованием в непрерывную (цепную) дробь (см. ниже). Однако полученная максимальная ошибка чуть-чуть больше заданной:

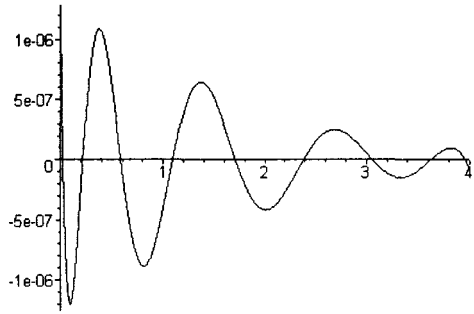


Рис. 17.4. Кривая ошибки при Паде–Чебышева рациональной аппроксимации

```
> maxChebPadeError := abs( F(0) - ChebPadeApprox(0) );
```

$$maxChebPadeError := .1236746\ 10^{-5}$$

Мы достигли впечатляющего успеха и остается сделать еще один шаг в направлении повышения точности аппроксимации.

Минимаксная аппроксимация

Классический результат теории аппроксимации заключается в том, что минимакс как наилучшая аппроксимация рациональной функции степени (m, n) достигается, когда кривая ошибки имеет m+n+2 равных по величине колебаний. Кривая ошибки аппроксимации Чебышева–Паде имеет нужное число колебаний, но эта кривая должна быть выровнена (по амплитуде выбросов кривой ошибки) с тем, чтобы обеспечить наилучшее минимаксное приближение. Эта задача решается с помощью функции `minimax`:

```
> MinimaxApprox := minimax(F, 0..4, [4,4], 1, 'maxerror');
```

$$MinimaxApprox := x \rightarrow (.174932901740 + (.0833009461857 + (-.0201932631012 + (.00368157773344 - .000157697316206x)x)x)x)/(.349866213807 + (.031945313924 + (.0622933587074 + (-.0011478761209 + .00336343612168x)x)x)x)$$

Максимальная ошибка в аппроксимации `MinimaxApprox` дается значением переменной `maxerror`. Заметим, что мы наконец достигли нашей цели получения аппроксимации с ошибкой меньшей, чем $1*10^{-6}$:

```
> maxMinimaxError := maxerror;
```

$$maxMinimaxError := .585025375366\ 10^{-6}$$

Построим график погрешности для данного типа аппроксимации:

```
> plot(F - MinimaxApprox,0..4,color=black);
```

График ошибки, представленный на рис. 17.5, показывает равные по амплитуде колебания.

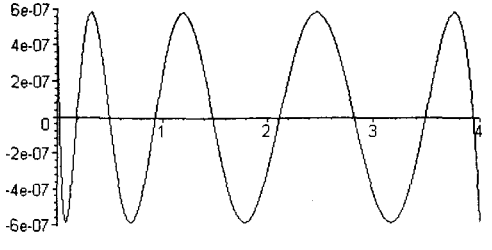


Рис. 17.5. График ошибки при минимаксной аппроксимации

Таким образом, мы добились блестящего успеха в снижении погрешности до требуемого и довольно жесткого уровня. Если бы мы задались целью получить только четыре или пять точных знаков аппроксимации, что в целом ряде случаев вполне приемлемо, то могли бы получить нужный результат гораздо раньше. Нам остается оптимизировать полученную аппроксимацию по минимуму арифметических операций и проверить реальный выигрыш по времени вычислений.

Эффективная оценка рациональных функций

Полиномы числителя и знаменателя в минимаксной аппроксимации уже выражены в форме Горнера (то есть в форме вложенного умножения). Оценка полиномом степени n в форме Горнера при n-умножениях и n-суммированиях — это наиболее эффективная схема оценки для полинома в общей форме. Однако для рациональной функции степени (m, n) мы можем делать кое-что даже лучше, чем просто представить выражения числителя и знаменателя в форме Горнера. Мы можем нормализовать рациональную функцию так, что полином знаменателя будет со старшим коэффициентом, равным 1. Мы можем также заметить, что вычисление рациональной функции степени (m, n) в форме Горнера требует выполнения все m+n сложений , m+n–1 умножений и 1 деления. Другими словами, общий индекс действия есть:

- m+n операций умножения/деления;
- m+n операций сложения/вычитания.

Вычисление рациональной функции можно значительно сократить и далее, преобразуя ее в непрерывную (цепную) дробь. Действительно, рациональная функция степени (m, n) может быть вычислена, при использовании только

- max(m,n) операций умножения/деления;
- m+n операций сложения/вычитания.

Например, если m = n, тогда эта новая схема требует выполнения только половины числа действий умножения/деления по сравнению с предшествующим методом. Для рациональной функции `MinimaxApprox` вычисление в форме, выраженной выше, сводится к 9 действиям умножения/деления и 8 действиям сложения/вычитания. Число операций умножения/деления можно сократить до 8, нормализуя знаменатель к форме monic. Мы можем теперь вычислить непре-

рывную (цепную) дробь для той же самой рациональной функции. Вычисление по этой схеме, как это можно видеть из вывода Maple, сводятся только к 4 действиям деления и 8 действиям сложения/вычитания:

```
> MinimaxApprox := confracform(MinimaxApprox):
> lprint(MinimaxApprox(x));
```

```
-.468857770747e-1+1.07858705749/(x+4.41994843227+16.1901737091/
(x+4.29121842830+70.1948525272/(x-10.2912843004+4.77536150167/(x+1.23883665458))))
```

Сравнение времен вычислений

Теперь определим время, необходимое для вычисления функции $f(x)$ в 1000 точек, используя первоначальное интегральное определение, и сравним его с временем, требующимся для схемы `MinimaxApprox` в виде непрерывной дроби. Так как наше приближение будет давать только 6 точных цифр, мы также потребуем 6 точных цифр и от интегрального представления функции:

```
> Digits := 6: st := time():
> seq( evalf(f(i/250.0)), i = 1..1000 ):
> oldtime := time() - st;
```

oldtime := 81.805

В процессе вычислений с использованием представления рациональной функции в виде непрерывной дроби иногда требуется внести несколько дополнительных цифр точности для страховки. В данном случае достаточно внести две дополнительные цифры. Итак, новое время вычислений:

```
> Digits := 8: st := time():
> seq( MinimaxApprox(i/250.0), i = 1..1000 ):
> newtime := time() - st;
```

newtime := .694

Ускорение вычисления при аппроксимации есть:

```
> SpeedUp := oldtime/newtime;
```

SpeedUp := 117.87464

Мы видим, что процедура вычислений, основанная на `MinimaxApprox`, выполняется почти в 120 раз быстрее процедуры с использованием исходного интегрального определения. Это просто феноменальный успех, полностью оправдывающий время, потерянное на предварительные эксперименты по аппроксимации и ее оптимизации! Разумеется, при условии, что вы будете применять эту аппроксимацию многократно.

Преобразование в код Фортрана или C

Один из поводов разработки эффективной аппроксимации для вычисления математической функции заключается в создании библиотек подпрограмм для популярных языков программирования высокого уровня, таких как Фортран или C. В Maple имеются функции преобразования на любой из этих языков. Например, мы можем преобразовывать формулу для минимаксной аппроксимации в код Фортрана.

```
> fortran(MinimaxApprox(x));
```

$$\text{fortran}\left(-.0468860934488 + \frac{1.07859095973}{x+4.41993871351+\dfrac{16.1901858102}{x+4.29117952035+\dfrac{70.1941645008}{x-10.2912413941+\dfrac{4.77540023037}{x+1.23883860342}}}}\right)$$

Итак, нами показано, что правильный выбор аппроксимации для сложной функции обеспечивает уменьшение времени ее вычисления более чем на два порядка (!) при весьма приличной точности в 6 верных знаков и при использовании для вычислений минимального числа арифметических операций. Применение при этом средств системы Maple 7 позволяет генерировать разложения в различные ряды, быстро вычислять рациональные аппроксимации функций и выполнять преобразования в различные специальные формы, сочетая это с мощными средствами интерактивной работы и графической визуализации, в частности с построением графиков функции и кривых ошибок при разных видах аппроксимации. Все это обеспечивает идеальную среду для решения таких задач.

Моделирование физических явлений

Расчет траектории камня с учетом сопротивления воздуха

Вы хотите метнуть камень в огород вашего вредного соседа? Разумеется, во время его отсутствия. Давайте промоделируем эту ситуацию, предположив два актуальных случая: дело происходит на Луне и на Земле. В первом случае сопротивления воздуха (как и его самого) нет, а в другом — сопротивление воздуха есть и его надо учитывать. Иначе камень упадет в ваш огород, а не в огород соседа!

Итак, пусть подвернувшиеся под руку камни с массой 500 и 100 г брошены под углом 45° к горизонту со скоростью Vo = 20 м/с. Найдем их баллистические траектории, если сила сопротивления воздуха Fтр=A*V, где A=0,1 Н*с/м. Сравним их с траекториями, получающимися без учета сопротивления воздуха.

Начнем с подключения пакета plots, нужного для визуализации данной задачи:

```
> restart;
> with(plots):
Warning, the name changecoords has been redefined
```

Составим параметрические уравнения для проекций скорости на оси координат:

```
> Vox:=Vo*cos(alpha);Voy:=Vo*sin(alpha):
```

$$Vox := Vo\cos(\alpha)$$

$$Voy := Vo\sin(\alpha)$$

Мы рассматриваем два случая: камень массой 500 г и камень массой 100 г. Поскольку для каждого случая мы предусматриваем расчет в двух вариантах (с учетом сопротивления воздуха и без такого учета), то мы должны составить 4 системы дифференциальных уравнений (ДУ). Каждая система состоит из двух ДУ второго порядка и вид этих систем известен из курса физики. Ниже представлено задание этих систем ДУ (для первой системы дан вывод ее вида):

```
> sys1:=massa[1]*diff(x(t),t$2)=-A[1]*diff(x(t),t),
massa[1]*diff(y(t),t$2)=-A[1]*(diff(y(t),t))-massa[1]*g;
```

$$sys1 := massa_1\left(\frac{\partial^2}{\partial t^2}\mathrm{x}(t)\right) = -A_1\left(\frac{\partial}{\partial t}\mathrm{x}(t)\right), massa_1\left(\frac{\partial^2}{\partial t^2}\mathrm{y}(t)\right) = -A_1\left(\frac{\partial}{\partial t}\mathrm{y}(t)\right) - massa_1\, g$$

```
> sys2:=massa[1]*diff(x(t),t$2)=-A[2]*diff(x(t),t),
massa[1]*diff(y(t),t$2)=-A[2]*(diff(y(t),t))-massa[1]*g:
> sys3:=massa[2]*diff(x(t),t$2)=-A[1]*diff(x(t),t),
massa[2]*diff(y(t),t$2)=-A[1]*(diff(y(t),t))-massa[2]*g:
> sys4:=massa[2]*diff(x(t),t$2)=-A[2]*diff(x(t),t),
massa[2]*diff(y(t),t$2)=-A[2]*(diff(y(t),t))-massa[2]*g:
```

Зададим исходные числовые безразмерные данные для расчета:

```
> Vo:=20;massa:=[0.5,0.1];A:=[0.1,0];alpha:=Pi/4;g:=9.8;
```

$$Vo := 20$$

$$massa := [.5, .1]$$

$$A := [.1, 0]$$

$$\alpha := \frac{1}{4}\pi$$

$$g := 9.8$$

Выполним решение заданных систем ДУ:

```
> p1:=dsolve({sys1,x(0)=0,D(x)(0)=Vox,y(0)=0,D(y)(0)=Voy}, {y(t),x(t)},
type=numeric ,output=listprocedure);
```

$$p1 := \left[t = (\mathrm{proc}(t) \ \ldots\ \mathrm{end\ proc}),\ \mathrm{x}(t) = (\mathrm{proc}(t) \ \ldots\ \mathrm{end\ proc}),\ \frac{\partial}{\partial t}\mathrm{x}(t) = (\mathrm{proc}(t) \ \ldots\ \mathrm{end\ proc}),\right.$$
$$\left.\mathrm{y}(t) = (\mathrm{proc}(t) \ \ldots\ \mathrm{end\ proc}),\ \frac{\partial}{\partial t}\mathrm{y}(t) = (\mathrm{proc}(t) \ \ldots\ \mathrm{end\ proc})\right]$$

```
> p2:=dsolve({sys2,x(0)=0,D(x)(0)=Vox,y(0)=0,D(y)(0)=Voy}, {y(t),x(t)},
type=numeric ,output=listprocedure):
> p3:=dsolve({sys3,x(0)=0,D(x)(0)=Vox,y(0)=0,D(y)(0)=Voy},
{y(t),x(t)},
type=numeric ,output=listprocedure):
> p4:=dsolve({sys4,x(0)=0,D(x)(0)=Vox,y(0)=0,D(y)(0)=Voy}, {y(t),x(t)},
type=numeric ,output=listprocedure):
```

Создадим графические объекты — результаты решения систем ДУ:

```
> a1:=odeplot(p1,[x(t),y(t)],0..3,color=green, view=[0..50,0..15], thickness=2):
> a2:=odeplot(p2,[x(t),y(t)],0..3,color=red, view=[0..50,0..15], thickness=2):
> a3:=odeplot(p3,[x(t),y(t)],0..3,color=blue, view=[0..50,0..15], thickness=2:
> a4:=odeplot(p4,[x(t),y(t)],0..3,color=black, view=[0..50,0..15], thickness=2):
```

Построим графики траекторий для первого случая:

```
> t:=textplot([[25,8,`A=0.1`],[35,9,`A=0`]],color=blue,
font=[TIMES,ROMAN,12]):
> t1:=textplot([[17,3,`A=0.1`],[35,9,`A=0`]],color=blue,
font=[TIMES,ROMAN,12]):
> display({a1,a2,t},title=`Траект. полета тела массой 500г`,
labels=[x,y],labelfont=[TIMES,ROMAN,14]);
```

Графики траекторий полета камня с массой 500 г представлены на рис. 17.6.

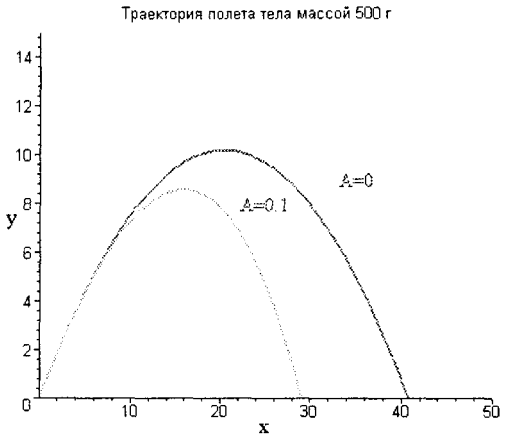


Рис. 17.6. Баллистические траектории камня с массой 500 г

Теперь построим графики траекторий для второго случая:

```
> display({a3,a4,t1},title=`Траект. полета тела массой 100 г`,
labels=[x,y],labelfont=[TIMES,ROMAN,14]);
```

Они представлены на рис. 17.7.

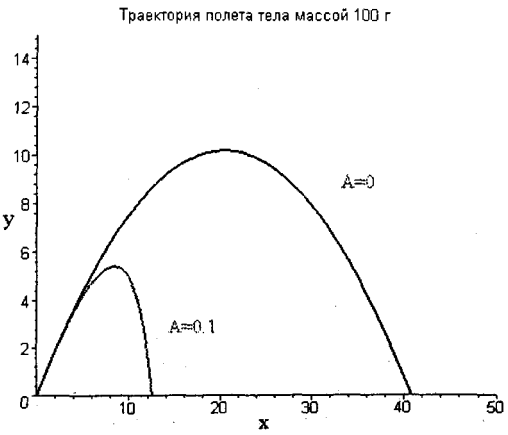


Рис. 17.7. Баллистические траектории камня при массе 100 г

Из проведенных расчетов и графиков видно, что при учете силы сопротивления воздуха дальность и высота полета сильно уменьшаются по сравнению с полетом в вакууме, и эта разница зависит от массы тела, поэтому при небольшой массе тела сопротивлением воздуха пренебрегать нельзя.

Движение частицы в магнитном поле

От реального мира перейдем к микромиру. Пусть микрочастица массой $9*10^{-31}$ кг и зарядом $+1{,}6*10^{-19}$ Кл влетает в магнитное поле с индукцией B = 0,1 Тл под углом a=80°. Рассчитаем траекторию движения частицы при начальной скорости Vo = $1*10^{7}$ м/с:

```
> restart;
```

Сила Лоренца, действующая на движущуюся частицу F = q*(E+[v, B]). Проекции векторного произведения [v, B] на оси x, y, z:

```
[v,B]x=vy*Bz-vz*By      [v,B]y=vz*Bx-vx*Bz    [v,B]z=vx*By-vy*Bz
```

В соответствии с этим известные из курса физики дифференциальные уравнения, описывающие траекторию полета частицы по осям x, y, z имеют вид:

```
> sys:=diff(x(t),t$2)=q*(Ex+(diff(y(t),t)*Bz-diff(z(t),t)*By))/massa,diff(y(t),t$2)=
                =q*(Ey+(diff(z(t),t)*Bx-diff(x(t),t)*Bz))/massa,diff(z(t),t$2)=
                                              =q*(Ez+(diff(x(t),t)* By-diff(y(t),t)*Bx))/massa;
```

$$sys := \frac{\partial^2}{\partial t^2}\mathrm{x}(t) = \frac{q\left(Ex + \left(\frac{\partial}{\partial t}\mathrm{y}(t)\right)Bz - \left(\frac{\partial}{\partial t}\mathrm{z}(t)\right)By\right)}{massa},$$

$$\frac{\partial^2}{\partial t^2}\mathrm{y}(t) = \frac{q\left(Ey + \left(\frac{\partial}{\partial t}\mathrm{z}(t)\right)Bx - \left(\frac{\partial}{\partial t}\mathrm{x}(t)\right)Bz\right)}{massa},$$

$$\frac{\partial^2}{\partial t^2}\mathrm{z}(t) = \frac{q\left(Ez + \left(\frac{\partial}{\partial t}\mathrm{x}(t)\right)By - \left(\frac{\partial}{\partial t}\mathrm{y}(t)\right)Bx\right)}{massa}$$

Зададим исходные числовые данные (опустив размерности):

```
> q:=-1.6e-19:massa:=9.1e-31:V:=1e7:alpha:=80*Pi/180:
> Vx:=V*cos(alpha):Vy:=V*sin(alpha):Ex:=0:Ey:=0:Ez:=0: Bx:=0.1:By:=0: Bz:=0:
```

Построим траекторию движения частиц в пространстве:

```
> with(DEtools):DEplot3d({sys},{x(t),y(t),z(t)},t=0..2e-9,[[x(0)=0,D(x)(0)=Vx,y(0)=
                    =0,D(y)(0)=Vy,z(0)=0,D(z)(0)=0]],stepsize=1e-11,orientation=[24,117]);
```

Полученная траектория представлена на рис. 17.8. Она имеет вид спирали в пространстве. При этом скорость движения частицы вдоль оси x неизменна, а вдоль осей y и z имеет характерную колебательную компоненту. Случай явно куда менее тривиальный, чем полет камня, описанный выше.

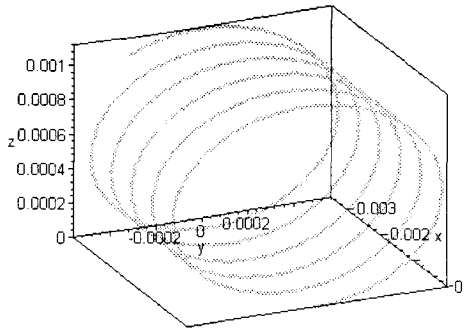


Рис. 13.8. Траектория движения частицы в магнитном поле

Мы можем найти аналитическое представление для траектории частицы в виде параметрически заданной (с параметром времени t) системы из трех уравнений:

```
>xyz:=dsolve({sys,x(0)=0,D(x)(0)=Vx,y(0)=0,D(y)(0)=
                                    =Vy,z(0)=0,D(z)(0)=0},{x(t),y(t),z(t)}, method=laplace):
```

$$xyz := \{z(t) = \frac{1000000}{879120879}\sin\left(\frac{4}{9}\pi\right)\sin(8791208790\,t)^2,\ x(t) = 10000000\cos\left(\frac{4}{9}\pi\right)t,$$
$$y(t) = \frac{500000}{879120879}\sin\left(\frac{4}{9}\pi\right)\sin(17582417580\,t)\}$$

Моделирование движения заряженной частицы в пространстве с магнитным полем показывает, что для принятых для моделирования параметров решаемой задачи, движение частицы происходит по спиралеобразной траектории. Получен как график траектории движения частицы, так и аналитические уравнения, описывающие это движение.

Разделение изотопов

Рассмотрим еще одну классическую задачу ядерной физики — разделение изотопов (атомов с одинаковым зарядом ядра, но разной массой). Для этого используют различные способы. В частности, это может быть масс-спектроскопический метод. Из точки A вылетают однозарядные ионы (q = e = $1.6*10^{-19}$ Кл) разной массы (от 20 до 23 а.е.м.) и под разными углами в пределах от 80 до 100° к оси x в плоскости xy (рис. 17.9). Вдоль оси z приложено магнитное поле B=10^{-2} Тл. Рассчитаем траектории полета частиц. Будем надеяться, что это подскажет способ разделения изотопов.

Приступим к решению данной задачи. Сила Лоренца, действующая на движущуюся частицу, F = q*(E+[v, B]). Проекции векторного произведения [v, B] на оси x, y, z заданы выражениями:

```
[v,B]x=vy*Bz-vz*By      [v,B]y=vz*Bx-vx*Bz    [v,B]z=vx*By-vy*Bz
```

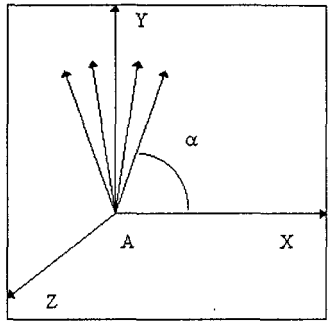


Рис. 17.9. Иллюстрация к методу разделения изотопов

В соответствии с этим дифференциальные уравнения, описывающие траекторию полета частицы по осям x, y, z имеют вид:

```
> restart;
> sys:=diff(x(t),t$2)=q*(Ex+(diff(y(t),t)*Bz-diff(z(t),t)*By))/massa,diff(y(t),t$2)=
                =q*(Ey+(diff(z(t),t)*Bx-diff(x(t),t)*Bz))/massa,diff(z(t),t$2)=
                                              =q*(Ez+(diff(x(t),t) *By-diff(y(t),t)*Bx))/massa;
```

$$sys := \frac{\partial^2}{\partial t^2}\mathrm{x}(t) = \frac{q\left(Ex + \left(\frac{\partial}{\partial t}\mathrm{y}(t)\right)Bz - \left(\frac{\partial}{\partial t}\mathrm{z}(t)\right)By\right)}{massa},$$

$$\frac{\partial^2}{\partial t^2}\mathrm{y}(t) = \frac{q\left(Ey + \left(\frac{\partial}{\partial t}\mathrm{z}(t)\right)Bx - \left(\frac{\partial}{\partial t}\mathrm{x}(t)\right)Bz\right)}{massa},$$

$$\frac{\partial^2}{\partial t^2}\mathrm{z}(t) = \frac{q\left(Ez + \left(\frac{\partial}{\partial t}\mathrm{x}(t)\right)By - \left(\frac{\partial}{\partial t}\mathrm{y}(t)\right)Bx\right)}{massa}.$$

Зададим исходные числовые данные для расчета:

```
> q:=1.6e-19:V:=1e4:
> Vx:=V*cos(alpha):Vy:=V*sin(alpha):Ex:=0:Ey:=0:Ez:=0:Bx:=0:
  By:=0:Bz:=1e-2:
```

Выполним решение составленной выше системы дифференциальных уравнений:

```
>xyz:=dsolve({sys,x(0)=0,D(x)(0)=Vx,y(0)=0,D(y)(0)=Vy,z(0)=0,D(z)(0)=0},
                                                    {x(t),y(t),z(t)},method=laplace):
> XX:=(massa,alpha)->.6250000000e25*massa*(sin(alpha)-1.*sin(alpha)*
                    *cos(.1600000000e-20*t/massa)+cos(alpha)*sin(.1600000000e-20*t/massa));
> YY:=(massa,alpha)->.6250000000e25*massa*(-1.*cos(alpha)+cos(alpha)*
                    *cos(.1600000000e-20*t/massa)+sin(alpha)*sin(.1600000000e-20*t/massa));
```

$$XX := (massa, \alpha) \to .625000000010^{25}\, massa$$
$$\left(\sin(\alpha) - 1.\sin(\alpha)\cos\left(.160000000010^{-20}\frac{t}{massa}\right) + \cos(\alpha)\sin\left(.160000000010^{-20}\frac{t}{massa}\right)\right)$$

$$YY := (massa, \alpha) \to .625000000010^{25}\, massa$$
$$\left(-1.\cos(\alpha) + \cos(\alpha)\cos\left(.160000000010^{-20}\frac{t}{massa}\right) + \sin(\alpha)\sin\left(.160000000010^{-20}\frac{t}{massa}\right)\right)$$

Построим графики решения:

```
> aem:=1.67e-27: ur:=3.14/180:
> plot([[XX(20*aem,80*ur),YY(20*aem,80*ur),
t=0..10e-5],[XX(20*aem,90*ur),YY(20*aem,90*ur),
t=0..10e-5],[XX(28*aem,80*ur),YY(28*aem,80*ur),
t=0..10e-5],[XX(28*aem,90*ur),YY(28*aem,90*ur),
t=0..10e-5],[XX(24*aem,80*ur),YY(24*aem,80*ur),
t=0..10e-5],[XX(24*aem,90*ur),YY(24*aem,90*ur),
t=0..10e-5]],view=[0..0.65,0..0.65],
color=[red,red,blue,blue,black,black],labels=[x,y]);
```

Эти графики показаны на рис. 17.10.

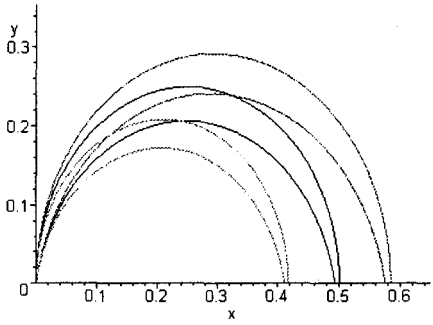


Рис. 17.10. Траектории движения частиц

Полученные графики (рис. 17.10) наглядно показывают на одну из возможностей разделения изотопов. Как говорится, осталось подставить «стаканчик» в нужное место для ловли нужных изотопов. Разумеется, это только изложение идеи одного из методов разделения изотопов. Увы, на практике приходится использовать сложнейшие и дорогие физические установки для решения этой актуальной задачи.

Моделирование рассеивания альфа-частиц

Одним из фундаментальных доказательств существования ядра у атомов стал опыт с бомбардировкой тонкой фольги из металла альфа-частицами с высокой энергией. Если бы «массивных» ядер не существовало, то альфа-частицы должны были бы спокойно пролетать сквозь тонкую фольгу, практически не отклоняясь. Однако, как физики и ожидали, некоторая часть частиц испытывала сильное отклонение и даже поворачивала назад. Очевидно, что имели место отскоки (упругие столкновения) с малыми, но массивными ядрами металла фольги.

В нашем распоряжении, увы (а может быть и к счастью), нет ускорителя альфа-частиц. Так что мы, не опасаясь облучения и очередной Чернобыльской катастрофы, сможем смоделировать это интереснейшее физическое явление с помощью математической системы Maple 7. Причем спокойно сидя перед своим домашним компьютером и глубокомысленно наблюдая за траекториями полета альфа-частиц.

Итак, пусть в нашем теоретическом опыте альфа-частицы с энергией 4 МэВ рассеиваются тонкой золотой фольгой. Рассчитать траекторию частицы, приближающейся к ядру атома Au. Прицельное расстояние p равно $2*10^{-15}$ м. Приступим к решению задачи и зададим вначале систему дифференциальных уравнений для траектории альфа-частицы:

```
> restart;
> sys:=diff(x(t),t$2)=q1*q2*x(t)/(4*Pi*E0*massa* (x(t)^2+
        +y(t)^2)^(3/2)),diff(y(t),t$2)=q1*q2*y(t)/(4*Pi*E0*massa*(x(t)^2+y(t)^2)^ (3/2));
```

$$sys := \frac{\partial^2}{\partial t^2}\mathrm{x}(t) = \frac{1}{4}\frac{q1\ q2\ \mathrm{x}(t)}{\pi\ E0\ massa\ (\mathrm{x}(t)^2 + \mathrm{y}(t)^2)^{(3/2)}},$$

$$\frac{\partial^2}{\partial t^2}\mathrm{y}(t) = \frac{1}{4}\frac{q1\ q2\ \mathrm{y}(t)}{\pi\ E0\ massa\ (\mathrm{x}(t)^2 + \mathrm{y}(t)^2)^{(3/2)}}$$

Введем исходные числовые данные для вычислений:

```
> q1:=2*1.6e-19:q2:=79*1.6e-19:massa:=4*1.67e-27:E0:=8.85e-12: a:=4e-13:
                                                    p:=5e-15:T:=4e6*1.6e-19:V0x:=sqrt(2*T/massa):
```

Создадим графическую структуру решения нашей системы дифференциальных уравнений для нескольких расчетных отклонений линии движения альфа-частицы от центра ядра атома, находящегося на ее пути:

```
> with(DEtools):ss:=DEplot({sys},{y(t),x(t)},t=0..7e-20,
[[x(0)=-a,D(x)(0)=V0x,y(0)=p,D(y)(0)=0],
[x(0)=-a,D(x)(0)=V0x,y(0)=p*4,D(y)(0)=0],
[x(0)=-a,D(x)(0)=V0x,y(0)=p*8,D(y)(0)=0],
[x(0)=-a,D(x)(0)=V0x,y(0)=p*12,D(y)(0)=0],
[x(0)=-a,D(x)(0)=V0x,y(0)=p*16,D(y)(0)=0],
[x(0)=-a,D(x)(0)=V0x,y(0)=p*20,D(y)(0)=0],
[x(0)=-a,D(x)(0)=V0x,y(0)=p*24,D(y)(0)=0],
[x(0)=-a,D(x)(0)=V0x,y(0)=p*28,D(y)(0)=0]],
x(t)=-a..a,scene=[x(t),y(t)],stepsize=1e-21,linecolor=black):
> with(plottools):yy:=circle([0,0],2E-14,color=red,thickness=2):
```

Warning, the name translate has been redefined

Построим центр ядра (кружок со знаком +) и траектории альфа-частиц:

```
> ss2:=PLOT(TEXT([0,-0.3e-14],`+`), FONT(HELVETICA, OBLIQUE,14)):
```

Осталось построить график траекторий движения альфа-частиц вблизи центра атома:

```
> with(plots):
```

Warning, the name changecoords has been redefined

```
> display([ss,yy,ss2],title=`Рассеивание а-частиц`,axes=framed);
```

График траекторий движения альфа-частиц вблизи ядра представлен на рис. 17.11. Этот график настолько нагляден, что не требует пояснения.

Моделирование движения альфа-частиц вблизи малого и «массивного» ядра атома дает наглядное представление о математической и физической сути данного опыта. Надо лишь помнить, что нельзя нацеливать альфа-частицы прямо

в центр ядра. Более сложные, чем приведенные, расчеты показывают, что при этом альфа-частица настолько близко подходит к ядру, что надо учитывать новые факторы, возникающие при близком взаимодействии. Они могут привести к тому, что частица будет поглощена ядром. Но это уже тема нового разговора, выходящего за рамки данной книги.

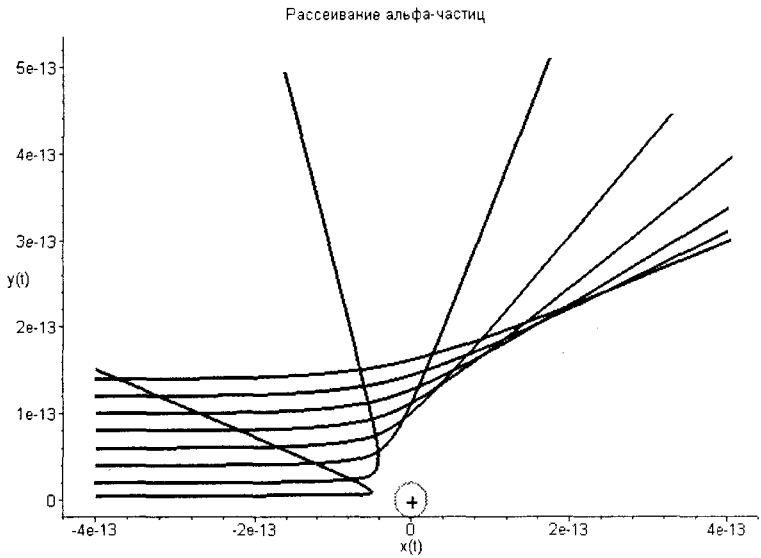


Рис. 17.11. Траектории движения альфа-частиц вблизи ядра атома

Моделирование и расчет электронных схем

Нужно ли применять Maple для моделирования и расчета электронных схем?

Нужно ли применять системы компьютерной математики для анализа, расчета и моделирования электронных схем? Ответ на этот вопрос не так прост, как кажется с первого взгляда. С одной стороны, к услугам пользователя компьютера сейчас имеется ряд программ схемотехнического моделирования, например Micro-CAP, Electronics Workbench, PSpice, Design Labs и др., автоматически составляющих и решающих большие системы уравнений состояния электронных схем и моделирующих работу бесчисленного множества электронных схем без кропотливого «ручного» составления уравнений.

Но, с другой стороны, анализ схем в таких программах настолько автоматизирован, что начисто теряется его физическая и математическая сущность. Это не так уж страшно, когда моделируются типовые схемы на давно известных или, скорее, просто хорошо знакомых электронных приборах. Но это явно плохо, когда объек-

том исследования и моделирования являются новые нетрадиционные схемы на новых или малоизвестных приборах или когда знание физических и математических основ работы таких схем принципиально необходимо. Например, при изучении их в вузах и университетах. В этом случае применение систем компьютерной математики не только возможно, но и принципиально необходимо.

Малосигнальный анализ усилителя на полевом транзисторе

Рассмотрим классический усилительный каскад на полевом транзисторе, схема которого приведена на рис. 17.12, а. Его эквивалентная малосигнальная схема представлена на рис. 17.12, б.

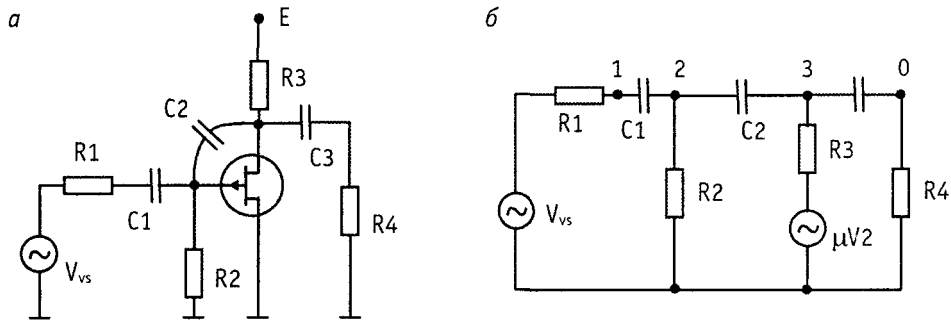


Рис. 17.12. Принципиальная (*а*) и эквивалентная (*б*) схемы усилителя на полевом транзисторе

Наша цель заключается в расчете характеристик усилителя операторным методом. Подключим нужный нам пакет plots:

```
> restart:with(plots):
```

Warning, the name changecoords has been redefined

Из законов Киргофа вытекает, что сумма токов, втекающих в каждый узел и вытекающих из него равна 0. Следовательно, для узлов эквивалентной схемы рис. 17.12 можно записать следующую систему уравнений в операторной форме:

$$> \ eq1 := 0 = \frac{V1 - Vvs}{R1} + \frac{V1 - V2}{\frac{1}{s\,C1}},$$

$$> \ eq2 := 0 = \frac{V2 - V1}{\frac{1}{s\,C1}} + \frac{V2}{R2} + \frac{V2 - V3}{\frac{1}{s\,C2}},$$

$$> \ eq3 := 0 = \frac{V3 - V2}{\frac{1}{s\,C2}} + \frac{V3 + \mu\,V2}{R3} + \frac{V3 - Vo}{\frac{1}{s\,C3}},$$

$$> \; eq4 := 0 = \frac{Vo - V3}{\frac{1}{s\,C3}} + \frac{Vo}{R4}.$$

Переменные напряжения на узлах схемы находятся из аналитического решения данной системы. При этом заблокируем вывод их аналитических значений, поскольку он очень громоздок. Тем не менее вы можете посмотреть на полученные формулы, поставив знак точки с запятой вместо знака двоеточия в приведенных ниже выражениях:

```
> solve({eq1,eq2,eq3,eq4},{V1,V2,V3,Vo}):
```

Обеспечим присвоение переменным Vo, V1, V2 и V3 найденных из решения системы уравнений значений:

```
> assign(%);
```

Теперь найдем операторную передаточную функцию в аналитическом виде:

$$> \; H := \frac{Vo}{Vvs}$$

$$\begin{aligned} H := ((&s\,C2\,R3 - \mu)\,C1\,R2\,s^2\,C3\,R4) \,/\, (1 + s\,C3\,R3 + s^2\,C2\,R2\,C1\,R1 + s\,C2\,R3 + s\,C3\,R4 \\ &+ C3\,R4\,s^2\,C2\,R2\,\mu + s^2\,C3\,R3\,C2\,R2 + s\,C2\,R2\,\mu + C3\,R4\,s^2\,C2\,R2 + s\,C2\,R2 + s\,C1\,R2 \\ &+ s^3\,C2\,R3\,C3\,R4\,C1\,R2 + s^2\,C3\,R4\,C1\,R2 + s^2\,C2\,R3\,C1\,R2 + s^2\,C3\,R3\,C1\,R2 \\ &+ C3\,R4\,s^3\,C2\,R3\,C1\,R1 + s^2\,C2\,R2\,\mu\,C1\,R1 + C3\,R4\,s^3\,C2\,R2\,C1\,R1 \\ &+ C3\,R4\,s^3\,C2\,R2\,\mu\,C1\,R1 + C3\,R4\,s^2\,C1\,R1 + C3\,s^3\,R3\,C2\,R2\,C1\,R1 + C3\,s^2\,R3\,C1\,R1 \\ &+ s^2\,C2\,R3\,C3\,R4 + C2\,R3\,s^2\,C1\,R1 + s\,C1\,R1) \end{aligned}$$

В соответствии с выбранным операторным методом анализа введем обозначения:

$$> \; \omega := 2\,\pi\,f$$

$$> \; s := I\,\omega$$

$$s := 2\,I\,\pi\,f$$

Это позволяет найти H как функцию от частоты f также в аналитическом виде:

```
> H;
```

$$\begin{aligned} -4\,((&2\,I\,\pi\,f\,C2\,R3 - \mu)\,C1\,R2\,\pi^2\,f^2\,C3\,R4) \,/\, (1 + 2\,I\,\pi\,f\,C2\,R3 + 2\,I\,\pi\,f\,C3\,R3 \\ &- 4\,\pi^2\,f^2\,C2\,R2\,C1\,R1 + 2\,I\,\pi\,f\,C3\,R4 - 4\,C3\,R4\,\pi^2\,f^2\,C2\,R2\,\mu - 4\,\pi^2\,f^2\,C3\,R3\,C2\,R2 \\ &+ 2\,I\,\pi\,f\,C2\,R2\,\mu - 4\,C3\,R4\,\pi^2\,f^2\,C2\,R2 + 2\,I\,\pi\,f\,C1\,R2 - 8\,I\,\pi^3\,f^3\,C2\,R3\,C3\,R4\,C1\,R2 \\ &- 4\,\pi^2\,f^2\,C3\,R4\,C1\,R2 - 4\,\pi^2\,f^2\,C3\,R3\,C1\,R2 - 8\,I\,C3\,R4\,\pi^3\,f^3\,C2\,R3\,C1\,R1 \\ &- 4\,\pi^2\,f^2\,C2\,R2\,\mu\,C1\,R1 - 8\,I\,C3\,R4\,\pi^3\,f^3\,C2\,R2\,C1\,R1 - 8\,I\,C3\,R4\,\pi^3\,f^3\,C2\,R2\,\mu\,C1\,R1 \\ &- 4\,C3\,R4\,\pi^2\,f^2\,C1\,R1 - 8\,I\,C3\,\pi^3\,f^3\,R3\,C2\,R2\,C1\,R1 - 4\,C3\,\pi^2\,f^2\,R3\,C1\,R1 \\ &- 4\,\pi^2\,f^2\,C2\,R3\,C3\,R4 - 4\,C2\,R3\,\pi^2\,f^2\,C1\,R1 + 2\,I\,\pi\,f\,C1\,R1 + 2\,I\,\pi\,f\,C2\,R2 \\ &- 4\,\pi^2\,f^2\,C2\,R3\,C1\,R2) \end{aligned}$$

Это тоже довольно громоздкое выражение, и его применение при «ручном» анализе потребовало бы от нас немало изобретательности. Между тем Maple 7 позволяет «в два счета» определить из него амплитудно-частотную (AVM) и фазо-частотную (PhaseAV) характеристики усилителя как функции частоты:

```
> AVM:=evalc(abs(H)):
> PhaseAV:=evalc(argument(H)):
```

Преобразуем AVD в логарифмическую характеристику, выражающую усиление в децибелах (dB):

```
> AVdB:=20*log10(AVM):
```

Такая характеристика более привычна для специалистов в радиоэлектронике. Соответственно фазо-частотную характеристику выразим в градусах:

```
> R2D:=evalf(360/(2*Pi));
```

$$R2D := 57.29577950$$

```
> AVdeg:=R2D*PhaseAV:
```

Теперь можно перейти к обычным численным расчетам. Зададим конкретные значения компонент эквивалентной схемы усилителя:

```
> R1:=100: R2:=100000: R3:=1000: R4:=10000:
C1:=.1*10^(-6): C2:=5*10^(-12): C3:=1*10^(-6):
mu:=50:
```

Построим амплитудно-частотную характеристику усилителя:

```
> gaindata:=NULL: phasedata:=NULL:
for a from 0 to 8 do:
  for i from 2*10^a to 10^(a+1) by 10^a do
   gaindata:=gaindata, [i, evalf(subs(f=i,AVdB))];
   phasedata:=phasedata, [i, evalf(subs(f=i,AVdeg))];
  od:
od:
> loglogplot([gaindata], thickness=2, color=black, style=line, axes=boxed,
                         title=`Коэффициент усиления K(f)`, labels=[`Частота (Hz)`,`K(dB)`]);
```

Она показана на рис. 17.13.

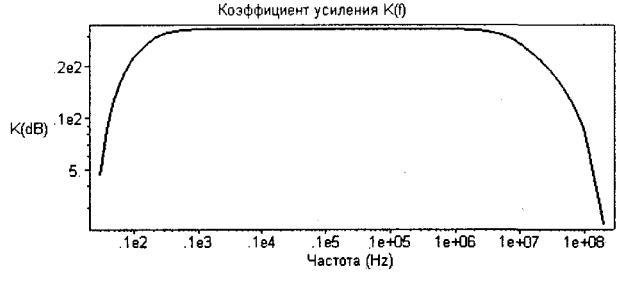


Рис. 17.13. Амплитудно-частотная характеристика усилителя

Далее зададим построение фазо-частотной характеристики усилителя:

```
> loglogplot([phasedata], thickness=2, color=blue, style=line, axes=boxed, title=`Фазовый
сдвиг (в градусах)`, labels=[`Частота (Hz)`,`Фаза`]);
```

Она представлена на рис. 17.14.

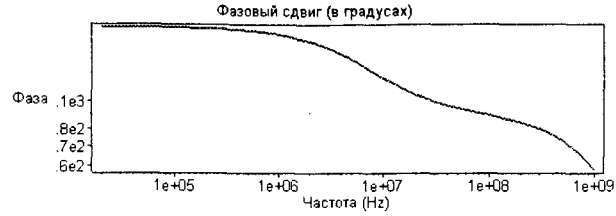


Рис. 17.14. Фазо-частотная характеристика усилителя

Найдем номинальный коэффициент усиления на частоте f=1000 (Гц):

```
> AVmid:=evalf(subs(f=1000, AVdB));
```

AVmid := 33.12074854

Имея аналитическое выражение для амплитудно-частотной характеристики, можно составить уравнения для вычисления граничных частот (по спаду усиления на -dAV в dB):

```
> dAV:=3:    #Ослабление (в dB на граничных частотах)
> eq5:=AVmid-dAV=20*log10(AVM):
```

Теперь можно найти эти частоты — нижнюю и верхнюю:

```
> flow:=fsolve(eq5,f, f=10..2000);
```

flow := 23.61659476

```
> fhigh:=fsolve(eq5,f, f=2000..100*10^6);
```

fhigh := .5737800225 10^7

Мы можем построить и более наглядную амплитудно-частотную характеристику с точками, соответствующими граничным частотам:

```
> with(plottools) :h:=log10(AVmid-dAV):
aplot:= loglogplot([gaindata], thickness=2, color=black, style=line, axes=boxed,
title=`Частоты flow и fhigh среза`, labels=[`Частота (Hz)`,`K(dB)`]):
bplot:=line([0.1,h], [7.1,h], color=black, linestyle=3):
cplot:=line([log10(flow),0.58], [log10(flow),1.6], color=blue, linestyle=3):
dplot:=line([log10(fhigh),0.58], [log10(fhigh),1.6], color=red, linestyle=3):
display([aplot,bplot,cplot,dplot]);
```

Эта характеристика показана на рис. 17.15.

На ней проставлены синяя и красная пунктирные вертикали, соответствующие найденным граничным частотам flow и fhigh, а также пунктирная горизонталь, соответствующая коэффициенту усиления на этих частотах. Это позволяет наглядно оценить частотный диапазон работы усилителя.

Таким образом, задача расчета усилителя в малосигнальном режиме полностью решена. Мы получили значение номинального коэффициента усиления, рассчитали нижнюю и верхнюю граничные частоты, получили аналитические выражения для амплитудно-частотной и фазо-частотной характеристик усилителя и построили их наглядные графики.

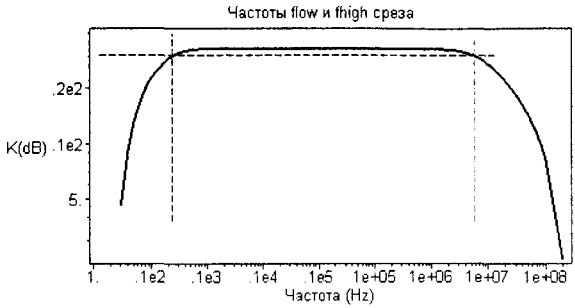


Рис. 17.15. Амплитудно-частотная характеристика с выделенными точками граничных частот

Расчет аналогового фильтра на операционном усилителе

Теперь рассмотрим проектирование аналогового полосового фильтра на операционном усилителе, схема которого приведена на рис. 17.16.

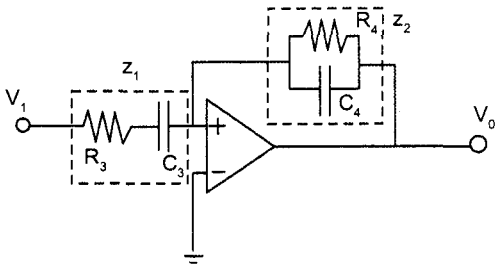


Рис. 17.16. Схема полосового фильтра на интегральном операционном усилителе

Подготовимся к расчету фильтра:

```
> restart:
```

Зададим основные уравнения, описывающие работу фильтра на малом сигнале:

```
> Vo := (-Z2/Z1)* Vi;
```

$$Vo := -\frac{Z2\ Vi}{Z1}$$

```
> Z1 := R3 + 1/(I*omega*C3);
```

$$Z1 := R3 - \frac{I}{\omega\ C3}$$

```
> Z2 := R4*1/(I*omega*C4) / (R4 + 1/(I*omega*C4));
```

$$Z2 := \frac{-I\ R4}{\omega\ C4\left(R4 - \frac{I}{\omega\ C4}\right)}$$

Введем круговую частоту:

```
> omega := 2*Pi*f;
```

$$\omega := 2\pi f$$

Найдем коэффициент передачи фильтра и его фазо-частотную характеристику как функции от частоты:

```
> gain := abs(evalc(Vo/Vi)):
> phase := evalc(op(2,convert(Vo/Vi,polar))):
```

Для просмотра громоздких аналитических выражений для этих параметров замените знаки двоеточия у выражений для gain и phase на знак точки с запятой. Далее введем конкретные исходные данные для расчета:

```
> R3 := 1000:
> R4 := 3000:
> C3 := 0.08*10^(-6):
> C4 := 0.01*10^(-6):
```

Построим АЧХ фильтра как зависимость коэффициента передачи в децибелах (dB) от частоты f в Гц:

```
> plot([log10(f), 20*log10(gain), f=10..50000], color=black,
      title=`Коэффициент передачи dB как функция от частоты f в Гц`);
```

Эта характеристика представлена на рис. 17.17. Здесь полезно обратить внимание на то, что спад усиления на низких и высоких частотах происходит довольно медленно из-за малого порядка фильтра.

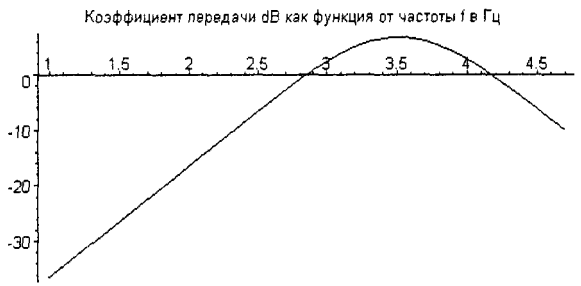


Рис. 17.17. АЧХ фильтра на операционном усилителе

Далее построим фазо-частотную характеристику фильтра как зависимость фазы в радианах от частоты f в Гц:

```
> plot ([log10(f), phase, f=10..50000], color=black,
                          title=`Фазо-частотная характеристика фильтра`);
```

Фазо-частотная характеристика (ФЧХ) фильтра показана на рис. 17.18.

На ФЧХ фильтра можно заметить характерный разрыв, связанный с превышением фазовым углом граничного значения π. Такой способ представления фазового сдвига общепринят, поскольку его изменения стремятся вписать в диапазон от $-\pi$ до π.

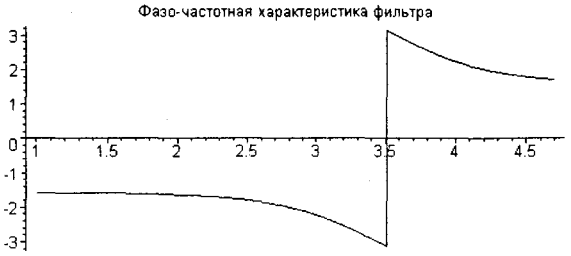


Рис. 17.18. ФЧХ фильтра на операционном усилителе

Проектирование цифрового фильтра

Основной недостаток аналоговых активных фильтров, подобных описанному выше, заключается в их малом порядке. Его повышение за счет применения многих звеньев низкого порядка ведет к значительному повышению габаритов фильтров и их стоимости. От этого недостатка свободны современные цифровые фильтры, число ячеек которых N даже при однокристальном исполнении может достигать десятков и сотен. Это обеспечивает повышенную частотную селекцию.

Спроектируем фильтр N+1-го порядка класса FIR (Finite Impulse Response или с конечной импульсной характеристикой). Каждая из N ячеек временной задержки фильтра удовлетворяет следующей зависимости выходного сигнала **y** от входного **x** вида:

$$y_n = \sum_{k=0}^{N} h_k x_{n-k}$$

Подключим пакет расширения plots, нужный для графической визуализации проектирования:

```
> restart:with(plots):
```
Warning, the name changecoords has been redefined

Зададим исходные данные для проектирования полосового цифрового фильтра, выделяющего пятую гармонику из входного сигнала в виде зашумленного меандра с частотой 500 Гц:

```
> N := 64:      # Число секций фильтра (на 1 меньше порядка фильтра)
> fs := 10000:  # Частота квантования
> fl := 2300:   # Нижняя граничная частота
> fh := 2700:   # Верхняя граничная частота
> m := 10:      # 2^m > N - число точек для анализа
```

Вычислим:

```
> T := 2^m-1:
```
$T := 1023$

```
> Fl := evalf(fl/fs);
```

$$F1 := .2300000000$$

```
> F2 := evalf(fh/fs);
```

$$F2 := .2700000000$$

```
> Dirac(0) := 1: # Функция Дирака
> fp1:=2*Pi*F1: fp2:=2*Pi*F2:
```

Зададим характеристику полосового фильтра:

```
> g := (sin(t*fp2)-sin(t*fp1))/(t*Pi);
```

$$g := \frac{\sin(.5400000000\,t\,\pi) - \sin(.4600000000\,t\,\pi)}{t\,\pi}$$

Вычислим FIR-коэффициенты для прямоугольного окна фильтра:

```
> C := (n)  ->  limit(g,t=n):h := aray(0..N): N2:=N/2:
> for n from 0 to N2 do h[N2-n]:= evalf(C(n)); h[N2+n] := h[N2-n]; od:
```

Определим массивы входного x(n) и выходного y(n) сигналов:

```
> x := array(-N..T):y := array(0..T):
```

Установим значение x(n) равным 0 для времени меньше 0 и 1 для времени >=0:

```
> for n from -N to -1 do x[n] := 0; od:
> for n from 0 to T do x[n] := Dirac(n); od:
```

Вычислим временную зависимость для выходного сигнала:

```
> for n from 0 to T do y[n] := sum(h[k]*x[n-k],k=0..N); od:
```

Построим график импульсной характеристики фильтра, отражающей его реакцию на сигнал единичной площади с бесконечно малым временем действия:

```
> p := [seq([j/fs,y[j]],j=0..T)]:
> plot(p, time=0..3*N/fs, labels=[time,output], axes=boxed, xtickmarks=4, title=`Импульсная
характеристика фильтра`,color=black);
```

Он показан на рис. 17.19. Нетрудно заметить, что эта характеристика свидетельствует об узкополосности фильтра, поскольку его частоты fl и fh различаются несильно. В этом случае полосовой фильтр по своим свойствам приближается к резонансному, хотя само по себе явление резонанса не используется.

Рис. 17.19. Импульсная характеристика цифрового фильтра

Вычислим АЧХ фильтра, используя прямое преобразование Фурье. Оно после подготовки обрабатываемых массивов реализуется функцией FFT:

```
> ro := array(1..T+1):io := array(1..T+1):
> for n from 0 to T do ro[n+1] := y[n]; io[n+1] := 0; od:
> FFT(m,ro,io):
```

Построим график АЧХ фильтра:

```
> p := [seq([j*fs/(T+1),abs(ro[j+1]+io[j+1]*I)],j=0..T/2)]:
> plot(p, frequency=0..fs/2, labels=[frequency,gain], title=`АЧХ фильтра`,color=black);
```

Он представлен на рис. 17.20. Нетрудно заметить, что и впрямь АЧХ фильтра напоминает АЧХ резонансной цепи — она имеет вид узкого пика. Вы можете легко проверить, что раздвижением частот fl и fh можно получить АЧХ с довольно плоской вершиной и резкими спадами (говорят, что такая характеристика приближается к прямоугольной).

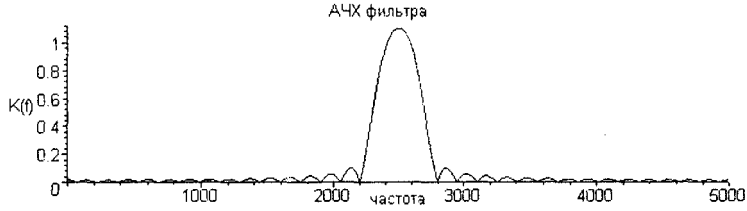


Рис. 17.20. АЧХ цифрового полосового фильтра

Теперь приступим к тестированию фильтра. Зададим входной сигнал в виде зашумленного меандра с частотой 500 Гц и размахом напряжения 2 В:

```
> l := round(fs/2/500):
> for n from 0 by 2*l to T do
>     for n2 from 0 to l-1 do
>         if n+n2 <= T then
>             x[n+n2] := evalf(-1+rand()/10^12-0.5);
>         fi;
>         if n+n2+l <= T then
>             x[n+n2+l] := evalf(1+rand()/10^12-0.5);
>         fi;
>     od:
> od:
```

Временная зависимость синтезированного входного сигнала представлена на рис. 17.21.

Рис. 17.21. Синтезированный входной сигнал

Вычислим реакцию фильтра на входной сигнал:

```
> for n from 0 to T do
>      y[n] := sum(h[k]*x[n-k],k=0..N);
> od:
```

Построим график выходного сигнала:

```
> p := [seq([j/fs,x[j]],j=0..T)]:q := [seq([j/fs,y[j]],j=0..T)]:
> plot(p,time=0..T/fs/4,labels=[time,volts],title=`Входной сигнал`,color=black);
> plot(q,time=0..T/fs/4,labels=[time,volts],title=`Выходной сигнал`,color=black);
```

Временная зависимость выходного сигнала показана на рис. 17.22. Нетрудно заметить, что в конце концов выходной сигнал вырождается в пятую гармонику входного сигнала, но этому предшествует довольно заметный переходной процесс. Он связан с узкополосностью данного фильтра.

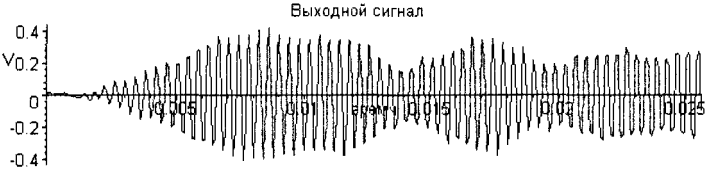


Рис. 17.22. Временная зависимость выходного сигнала цифрового фильтра

Вычислим спектры входного и выходного сигналов, подготовив массивы выборок сигналов и применив прямое преобразование Фурье с помощью функции FFT:

```
> ri := array(1..T+1):ii := array(1..T+1):
> for n from 0 to T do
>      ri[n+1] := x[n]*2/T; ii[n+1] := 0;
>      ro[n+1] := y[n]*2/T; io[n+1] := 0;
> od:
> FFT(m,ri,ii):FFT(m,ro,io):
```

Построим график спектра входного сигнала, ограничив масштаб по амплитуде значением 0,5 В:

```
> p := [seq([j*fs/(T+1),abs(ri[j+1]+ii[j+1]*I)],j=0..T/2)]:
> q := [seq([j*fs/(T+1),abs(ro[j+1]+io[j+1]*I)],j=0..T/2)]:
> plot(p, frequency=0..fs/2,y=0..0.5,labels=[частота,V], title=`Частотный спектр входного
сигнала`,color=black);
```

Этот график представлен на рис. 17.23. Из него хорошо видно, что спектральный состав входного сигнала представлен только нечетными гармониками, амплитуда которых убывает по мере роста номера гармоники. Пятая гармоника на частоте 2500 Гц находится посередине полосы пропускания фильтра, ограниченной граничными частотами фильтра 2300 и 2700 Гц. Заметны также беспорядочные спектральные линии шума сигнала в пределах полосы прозрачности фильтра.

Теперь построим график спектра выходного сигнала:

```
> plot(q, frequency=0..fs/2,y=0..0.5,labels=[частота,V], title=`Частотный спектр выходного
сигнала`,color=black);
```

Он представлен на рис. 17.24. Хорошо видно эффективное выделение пятой гармоники сигнала и прилегающей к ней узкой полосы шумового спектра.

Рис. 14.23. Спектрограмма входного сигнала

Рис. 17.24. Спектрограмма выходного сигнала цифрового фильтра

Приведенные данные свидетельствуют, что спроектированный фильтр полностью отвечает заданным требованиям и обеспечивает уверенное выделение пятой гармоники зашумленного меандра. По образу и подобию данного документа можно выполнить проектирование и других видов цифровых фильтров.

Моделирование цепи на туннельном диоде

А теперь займемся моделированием явно нелинейной цепи. Выполним его для цепи, которая состоит из последовательно включенных источника напряжения Es, резистора Rs, индуктивности L и туннельного диода, имеющего N-образную вольтамперную характеристику (ВАХ). Туннельный диод обладает емкостью C, что имитируется конденсатором C, подключенным параллельно туннельному диоду. Пусть ВАХ реального туннельного диода задана выражением:

```
> restart;
> A:=.3: a:=10: B:=1*10^(-8): b:=20:
> Id:=Ud->A*Ud*exp(-a*Ud)+B*(exp(b*Ud-1));
```

$$Id := Ud \rightarrow A\ Ud\ e^{(-a\ Ud)} + B\ e^{(b\ Ud - 1)}$$

Построим график ВАХ:

```
> plot(Id(Ud), Ud=-.02..0.76,color=black);
```

Этот график представлен на рис. 17.25. Нетрудно заметить, что ВАХ туннельного диода не только резко нелинейна, но и содержит протяженный участок отрицательной дифференциальной проводимости, на котором ток падает с ростом напряжения. Это является признаком того, что такая цепь способна на переменном токе отдавать энергию во внешнюю цепь и приводить к возникновению колебаний в ней различного типа.

Работа цепи описывается системой из двух дифференциальных уравнений:

```
di/dt=(Es-i(t)*Rs-u(t))/L
du/dt=(i(t)-Id(u(t))/C
```

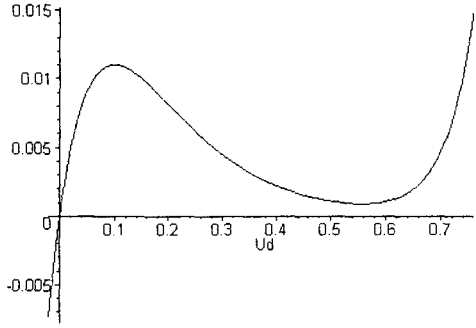


Рис. 17.25. ВАХ туннельного диода

Пусть задано Es = 0,35 В, Rs = 15 Ом, C = 10*10-12, L = 30*10-9 и максимальное время моделирования tm=10*10-9. Итак, задаем исходные данные:

```
> Es:=.35:Rs:=15:C:=10*10^(-12):L:=30*10^(-6):tm:=10*10^(-9):
```

Составим систему дифференциальных уравнений цепи и выполним ее решение с помощью функции dsolve:

```
> se:=diff(i(t),t)=(Es-i(t)*Rs-u(t))/L,
    diff(u(t),t)=(i(t)-Id(u(t)))/C:
```

$$se := \frac{\partial}{\partial t} i(t) = .116666666710^{8} - 500000000 i(t) - \frac{100000000}{3} u(t),$$

$$\frac{\partial}{\partial t} u(t) = 100000000000 i(t) - .300000000010^{11} u(t) e^{(-10 u(t))} - 1000 e^{(20 u(t) - 1)}$$

```
> F:=dsolve({se,i(0)=0,u(0)=0},{i(t),u(t)},type=numeric, method=classical, stepsize=10^(-
11), output=listprocedure);
```

$$F := [t = (\mathrm{proc}(t) \ \ldots \ \mathrm{end\ proc}), u(t) = (\mathrm{proc}(t) \ \ldots \ \mathrm{end\ proc}),$$
$$i(t) = (\mathrm{proc}(t) \ \ldots \ \mathrm{end\ proc})]$$

Поскольку заведомо известно, что схема имеет малые значения L и C, мы задали с помощью парметров достаточно малый шаг решения для функции dsolve — stepsize=10^(-11) (с). При больших шагах возможна численная неустойчивость решения, искажающая форму колебаний, получаемую при моделировании. Используя функции odeplot и display пакета plots, построим графики решения в виде временных зависимостей u(t) и 10*i(t) и линии, соответствующей напряжению Es источника питания:

```
> gu:=odeplot(F,[t,u(t)],0..tm,color=black,
     labels=[`t`,`u(t),10*i(t)`]):
> gi:=odeplot(F,[t,10*i(t)],0..tm,color=black):
> ge:=odeplot(F,[t,Es],0..tm,color=red):
```

```
> display(gu,gi,ge);
```

Эти зависимости представлены на рис. 17.26. Из них хорошо видно, что цепь создает автоколебания релаксационного типа. Их форма сильно отличается от синусоидальной.

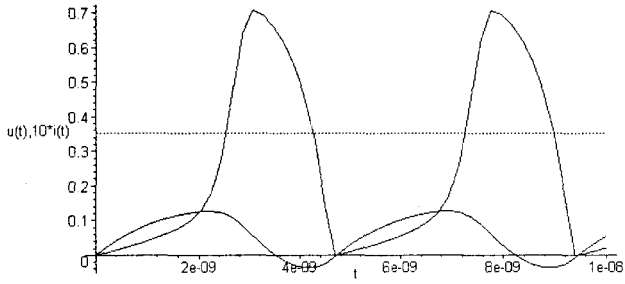


Рис. 17.26. Временные зависимости напряжения на туннельном диоде и тока

Решение можно представить также в виде фазового портрета, построенного на фоне построенных ВАХ и линии нагрузки резистора Rs:

```
> gv:=plot({Id(Ud),(Es-Ud)/Rs},Ud=-.05..0.75,color=black,
                                                labels=[Ud,Id]):
> gpp:=odeplot(F,[u(t),i(t)],0..tm,color=blue):
> display(gv,gpp);
```

Фазовый портрет колебаний показан на рис. 17.27.

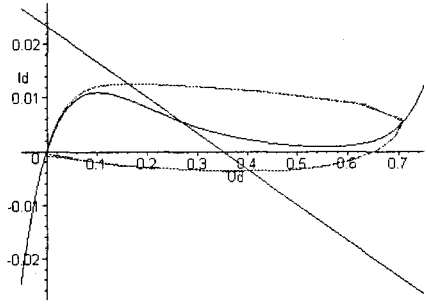


Рис. 17.27. Фазовый портрет колебаний на фоне ВАХ туннельного диода и линии нагрузки резистора Rs

О том, что колебания релаксационные можно судить по тому, что уже первый цикл колебаний вырождается в замкнутую кривую — предельный цикл, форма которого заметно отличается от эллиптической.

Итак, мы видим, что данная цепь выполняет функцию генератора незатухающих релаксационных колебаний. Хотя поставленная задача моделирования цепи на туннельном диоде успешно решена, в ходе ее решения мы столкнулись с проблемой обеспечения малого шага по времени при решении системы дифферен-

циальных уравнений, описывающих работу цепи. При неудачном выборе шага можно наблюдать явную неустойчивость решения.

Применение интеграла Дюамеля для расчета переходных процессов

Вернемся к линейным цепям и рассмотрим еще один полезный метод расчета электрических цепей — с помощью интеграла Дюамеля. При нем можно рассчитать временную зависимость выходного напряжения u2(t) цепи по известному входному сигналу u1(t) и переходной характеристике цепи a(t). Возьмем в качестве первого классического примера дифференцирующую RC-цепь и вычислим ее реакцию на экспоненциально нарастающий перепад напряжения. Соответствующие расчеты приведены на рис. 17.28.

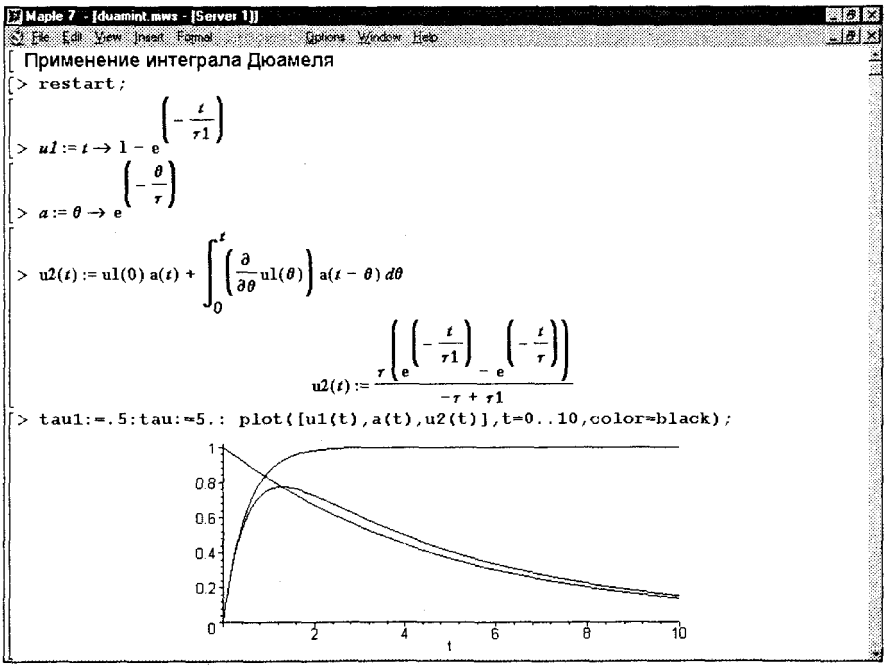


Рис. 17.28. Расчет реакции дифференцирующей цепи на экспоненциальный перепад напряжения

Рисунок 17.28 представляет начало документа, в котором выполнен указанный выше расчет. Представлены заданные зависимости u1(t) и a(t), аналитическое выражение для интеграла Дюамеля (одна из 4 форм) и аналитическое выражение для искомой зависимости u2(t). Пока последнее выражение довольно простое. В конце этого фрагмента документа построены графики зависимостей u1(t), a(t) и u2(t).

Окончание документа, представленное на рис. 17.29, демонстрирует расчет на основе интеграла Дюамеля реакции дифференцирующей RC-цепи на экспоненциально затухающий синусоидальный сигнал u1(t).

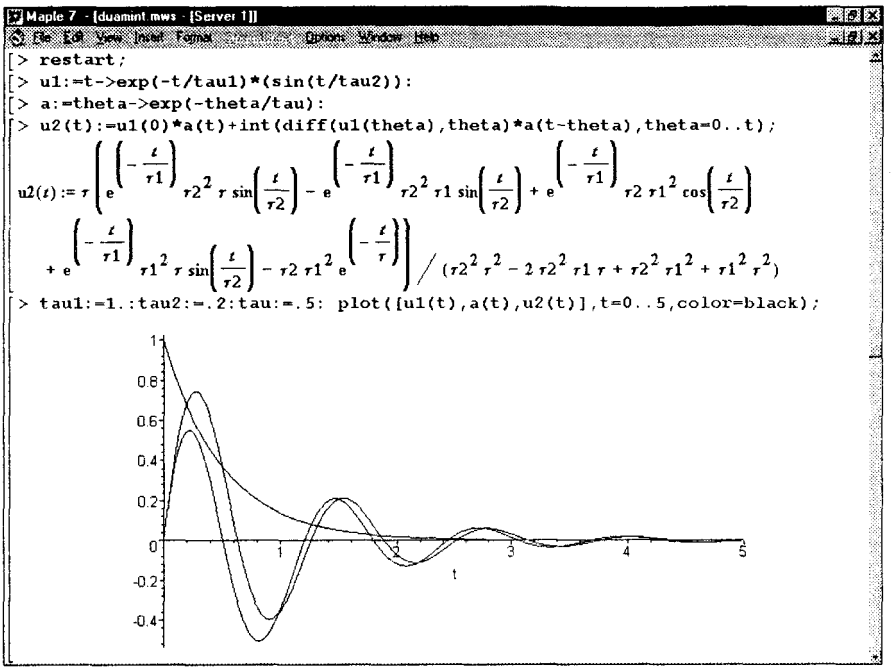


Рис. 17.29. Расчет реакции дифференцирующей цепи на синусоидальный сигнал с экспоненциально уменьшающейся амплитудой

Обратите внимание на то, что выражение для u2(t), получаемое с помощью интеграла Дюамеля, стало намного сложнее. Тем не менее получено как аналитическое выражение для реакции цепи u2(t), так и графики u1(t), a(t) и u2(t). Они показаны внизу графика.

Что нового мы узнали?

В этом уроке мы научились:

- Оценивать возможности Maple 7 в решении конкретных прикладных задач.
- Выбирать аппроксимацию для сложной функции по заданной точности.
- Моделировать различные физические явления (полет камня, движение частицы в магнитном поле и др.).
- Моделировать и проектировать различные электронные схемы (усилителя, аналогового и цифрового фильтра, нелинейной цепи на туннельном диоде).
- Применять интеграл Дюамеля для расчета переходных процессов в линейных цепях.

Заключение

Программа Maple корпорации Waterloo Maple Inc. — патриарх в мире систем компьютерной математики. Эта система, снискавшая себе мировую известность и огромную популярность, является одной из лучших среди систем символьной математики, позволяющих решать математические задачи в аналитическом виде.

Эта книга познакомила читателей с новейшей версией Maple — Maple 7. Она вобрала в себя не только обширные и мощные возможности предшествующих реализаций системы, но и предоставила в распоряжение пользователя ряд новых возможностей. Прежде всего это целый букет пакетов: CurveFitting, PolynomialTools, OrthogonalSeries и др.

Maple как система компьютерной математики развивается по ряду характерных направлений. Одно из них — повышение мощности и достоверности аналитических (символьных) вычислений. Это направление представлено в Maple наиболее сильно. Maple 7 уже сегодня способна выполнять сложнейшие аналитические вычисления, которые нередко не под силу даже опытным математикам. Конечно, Maple не способна на «гениальные догадки», но зато рутинные и массовые расчеты система выполняет с блеском. В новой версии ее возможности существенно расширены, особенно в области решений дифференциальных уравнений.

Другое важное направление — повышение эффективности численных расчетов. И тут успехи налицо — начиная с версии Maple 6 в систему включены эффективные алгоритмы группы NAG, лидирующей в области численных расчетов. Повышена эффективность и алгоритмов самой системы Maple 7. В результате этого заметно возросла перспектива использования Maple в численном моделировании и выполнении сложных численных расчетов — в том числе с произвольной точностью.

Интеграция Maple с другими программными средствами — еще одно важное направление развития этой системы. Ядро символьных вычислений Maple уже включено в состав целого ряда систем компьютерной математики — от систем «для всех» класса Mathcad до одной из лучших систем для численных расчетов и моделирования — MATLAB. Имеется целый ряд автоматизированных рабочих мест для математиков на основе ядра системы Maple: Math Office, Scientific Word, Scientific WorkPlace и др.

Предусмотрена и интеграция Maple 7 с Excel 2000 и MATLAB. Однако альянс Maple 7 с Excel трудно назвать удачным. Во-первых, потому, что куда более распространенная версия Excel 97 связь с Maple 7 не поддерживает. Во-вторых, введенные в Maple 7 средства работы с таблицами (в том числе новые) в большинстве случаев оказываются более удобными, чем обычные средства работы с таблицами у Excel. Достаточно отметить, что таблицы в Maple могут работать с формульными данными и построение рисунков в Maple не требует создания таблицы данных для них, как это нужно в Excel.

Существенно расширена поддержка системы Maple через Интернет. Появление на сайте корпорации Waterloo Maple Inc. массы информационных материалов, и прежде всего обучающих программ и примеров применения Maple, разгрузило саму программу и предоставило ее пользователям обширные возможности в пополнении своих знаний и навыков работы с Maple 7.

С другой стороны, резко повышены возможности Maple 7 для создания web-страниц — основы Интернета. Здесь прежде всего надо отметить включение в пакеты средств поддержки языков HTML, XML и (что особенно важно) MathML.

Все эти возможности в сочетании с прекрасно выполненным и удобным пользовательским интерфейсом и мощной справочной системой делают Maple 7 первоклассной программной средой для решения самых разнообразных математических задач: от самых простых до самых сложных. Особо следует отметить возможность создания превосходных электронных документов, статей, книг и учебников в среде Maple 7 с «живыми» и модифицируемыми примерами.

Maple — быстро развивающаяся система, и работа с ней не только полезна, но и приятна для всех категорий пользователей и учащихся. Автор надеется, что эта книга привлечет внимание наших читателей, и прежде всего специалистов, преподавателей вузов, аспирантов, студентов и даже школьников, к такому уникальному программному продукту, как система компьютерной математики Maple 7, и поможет им в решении учебных и реальных научно-технических задач.

Список литературы

1. *Дьяконов В. П.* Справочник по расчетам на микрокалькуляторах. 3-е издание, дополненное и переработанное. М.: Наука, Физматлит, 1989.
2. *Дьяконов В. П.* Справочник по алгоритмам и программам на языке Бейсик для персональных ЭВМ. М.: Наука, Физматлит, 1987.
3. *Дьяконов В. П.* Форт-системы программирования персональных ЭВМ. М.: Наука, Физматлит, 1992.
4. *Дьяконов В. П* . Справочник по применению системы Eureka. М.: Наука, Физматлит, 1993.
5. *Дьяконов В. П.* Mercury — отличная система для всех // Монитор-Аспект. 1995. № 5.
6. *Дьяконов В. П.* Как выбрать математическую систему? // Монитор-Аспект. 1993. № 2.
7. *Дьяконов В. П.* Компьютерная математика. Теория и практика. М.: Нолидж, 2001.
8. *Дьяконов В. П.* Расширяемые системы для численных расчетов MatLAB. Монитор-Аспект, № 2, 1993.
9. *Дьяконов В. П.* Справочник по применению системы PC MatLAB. М.: Наука, Физматлит, 1993.
10. *Дьяконов В. П.* Система MathCAD: Справочник. М.: Радио и связь, 1993.
11. MathCAD 6.0 PLUS. Финансовые, инженерные и научные расчеты в среде Windows 95. Пер. с англ. М.: Филинъ, 1996.
12. *Дьяконов В. П.* Справочник по MathCAD PLUS 6.0 PRO. М.: СК-ПРЕСС, 1997.
13. *Дьяконов В. П.* Справочник по MathCAD 7.0 PRO. М.: СК-ПРЕСС, 1998.
14. *Очков В. Ф.* MathCAD 7 Pro для студентов и инженеров. М.: Компьютер Press, 1998.
15. *Дьяконов В. П., Абраменкова И. В.* Mathcad 7 в математике, в физике и в Интернет. М.: Нолилж, 1998.
16. *Дьяконов В. П., Абраменкова И. В.* Mathcad 8 в математике, в физике и в Интернет. М.: Нолилж, 1999.
17. *Очков В. Ф.* MathCAD 8 Pro для студентов и инженеров. М.: Компьютер Press, 1999.
18. *Плис А. И., Сливина Н. А.* Mathcad: математический справочник. М.: Финансы и статистика, 1999.

19. *Дьяконов В. П.* Mathcad 8/2000: Специальный справочник. СПБ: Питер, 2000.
20. *Дьяконов В. П.* Mathcad 2000: Учебный курс. СПБ: Питер, 2000.
21. *Дьяконов В. П.* Mathcad 2001: Учебный курс. СПБ: Питер, 2001.
22. *Дьяконов В. П.* Справочник по применению системы Derive. М: Наука, Физматлит, 1996.
23. *Дьяконов В. П.* Справочник по системе символьной математики Derive. М.: СК-ПРЕСС, 1998.
24. *Лобанова О. В.* Практикум по решению задач в математической системе Derive. М.: Финансы и статистика, 1999.
25. *Дьяконов В. П.* Справочник по математической системе Mathematica 2 и 3. М.: СК-ПРЕСС, 1998.
26. *Воробьев Е. М.* Введение в систему «Mathematica». М.: Финансы и статистика, 1998.
27. *Дьяконов В. П.* Mathematica 4 с пакетами расширений. М.: Нолидж, 2000.
28. *Дьяконов В. П.* Mathematica 4: Учебный курс. СПБ: Питер, 2001.
29. *Потемкин В. Г.* MATLAB: Справочное пособие. М.: Диалог-МИФИ, 1997.
30. *Дьяконов В. П., Абраменкова И. В.* MATLAB 5.0/5.3 — система символьной математики. М.: Нолидж, 1999.
31. *Дьяконов В. П.* MATLAB 5.3: Учебный курс. СПБ: Питер, 2001.
32. *Дьяконов В., Новиков Ю., Рычков В.* Самоучитель. Компьютер для студента. СПб: Питер, 2000.
33. *Дьяконов В. П.* Maple V — мощь и интеллект компьютерной алгебры! // Монитор-Аспект. 1993. № 2.
34. *Прохоров Г. В., Леденев М. А., Колбеев В. В.* Пакет символьных вычислений Maple V. М.: Петит, 1997.
35. *Манзон Б. М.* Maple V Power Edition. М.: Филинъ, 1998.
36. *Дьяконов В. П.* Математическая система Maple V R3/R4/R5. М.: Солон, 1998.
37. *Дьяконов В. П.* Maple 6: Учебный курс. СПБ: Питер, 2001.
38. *Матросов А.* Maple 6. Решение задач высшей математики и механики. СПБ: БХВ-Санкт-Петербург, 2001.
39. *Heal K. M., Hansen L. M., Rickard K. M.*. Maple 6. Learning Guide. Waterloo Maple Inc., 2000.
40. *Monagan M. B., Geddes K. O., Heal K. M., Labahn G., Vorkoetter S. M.*. Maple 6. Programming Guide. Waterloo Maple Inc., 2000.
41. *Дэвенпорт Дж., Сирэ И., Турнье Э.*. Компьютерная алгебра. Системы и алгоритмы алгебраических вычислений. М.: Мир, 1991.
42. *Саймон Барри.* Символьная математика: новые времена — новые формы // PC Magazine/RE. 1992. № 5.
43. *Гантмахер Ф.* Теория матриц. М.: Наука, Физматлит, 1988.

44. Справочник по специальным функциям с формулами, графиками и математическими таблицами / Под ред. М. Абрамовица и И. Стиган. М.: Наука, Физматлит, 1979.

45. *Корн Г., Корн Т.* Справочник по математике для научных работников и инженеров. М.: Наука, 1973.

46. *Воднев В. Т., Наумович А. Ф., Наумович Н. Ф.* Основные математические формулы. Минск: Высшая школа, 1988.

47. *Moon P., Spencer D. E.* Field Theory Handbook, 2nd Ed. Berlin: Springer-Verlag, 1971.

48. *Spiegel, Murray R.* Mathematical Handbook of Formulas and Tables. New York: McGraw Hill Book Company, 1968.

49. *Дьяконов В. П.* Windows 98. 98 вопросов по Windows 98 с ответами. М.: Солон, 1999.

50. *Дьяконов В. П.* Мой Word 95/97. М.: АСТ, 1998.

51. *Дьяконов В. П.* Internet. Настольная книга пользователя. 3-е издание. М.: Нолидж, 2001.

Алфавитный указатель

Символы

А

Б

В

Г

Д

З

И

К

Л

М

Н

О

П

Р

С

У

Ф

Ц

Ч

Э

Я

A

B

C

D

E

F

H

I

L

M

N

O

P

R

S

T

U

V

W

Z