ПЕРМСКИЙ
ГОСУДАРСТВЕННЫЙ
НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ

СОВРЕМЕННЫЕ ПАКЕТЫ ПРИКЛАДНЫХ ПРОГРАММ

ОСНОВЫ РАБОТЫ В MAPLE

ИЗДАТЕЛЬСКИЙ ЦЕНТР
"Perm University Press"

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«ПЕРМСКИЙ ГОСУДАРСТВЕННЫЙ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»

СОВРЕМЕННЫЕ ПАКЕТЫ ПРИКЛАДНЫХ ПРОГРАММ

ОСНОВЫ РАБОТЫ В MAPLE

Допущено методическим советом
Пермского государственного национального
исследовательского университета в качестве
учебного пособия для студентов, обучающихся
по направлению подготовки бакалавров
«Прикладные математика и физика»

Пермь 2022

УДК 519.67(075.8)
ББК 22.195я73
С568

Современные пакеты прикладных программ. Основы работы
С568 в Maple [Электронный ресурс] : учебное пособие / сост. В. Г. Гилев ; Пермский государственный национальный исследовательский университет. – Электронные данные. – Пермь, 2022. – 2,61 Мб ; 95 с. – Режим доступа: http://www.psu.ru/files/docs/science/books/uchebnie-posobiya/Gilev-Sovremennye-Pakety-Prikladnyh-Programm-Osnovy-Raboty-V-Maple.pdf. – Заглавие с экрана.

ISBN 978-5-7944-3850-5

Учебное пособие содержит материал, необходимый при анализе и обработке экспериментальных данных. Основное внимание в уделено операциям с символьными выражениями, решению линейных и трансцендентных уравнений и их систем, обыкновенных дифференциальных уравнений и их систем, основам построения двумерной и трехмерной графики, аппроксимации экспериментальных данных, процедурам импорта числовых значений в Maple из сторонних программ. Каждый раздел пособия включает задания для самостоятельной работы.

Пособие предназначено для студентов, обучающихся по направлениям подготовки 03.03.01, 03.04.01 «Прикладные математика и физика».

УДК 519.67(075.8)
ББК 22.195я73

Издается по решению ученого совета физического факультета Пермского государственного национального исследовательского университета

Рецензенты: кафедра физики и технологии Пермского государственного гуманитарно-педагогического университета (зав. кафедрой д-р физ.-мат. наук, профессор ***В. Г. Козлов***);

зав. лабораторией гидродинамической устойчивости института механики сплошных сред УрО РАН, д-р физ.-мат. наук, доцент ***А. И. Мизев***

ISBN 978-5-7944-3850-5

ВВЕДЕНИЕ

Maple – это интерактивная среда для научных и технических расчетов, включает в себя аналитические вычисления, визуализацию и программирование в удобной визуальной среде, где задачи и решения выражаются в форме, близкой к математической.

Основное назначение: математические вычисления и аналитические расчеты, анализ данных, моделирование, научная графика, разработка приложений, включая создание графического интерфейса пользователя и многое другое.

Детальное освоение любой большой программной системы – достаточно длительный процесс, основу которого составляет самостоятельная работа пользователя. Любая, даже самая «толстая» книга, не в состоянии охватить всех особенностей этой интерактивной оболочки. Настоящее пособие рассчитано на пользователей, знакомых со средой Maple, включающей: интерфейс программы, набор инструментов, средства для управления переменными в рабочем пространстве, основные функции и элементы программирования.

Цель пособия – дать основы тех функций и элементов Maple, которые могут быть полезны при анализе и обработке экспериментальных данных спецпрактикумов «Физика фазовый переходов», «Оптика анизотропных сред», «Экспериментальные методы в физике» по направлениям подготовки 03.03.01, 03.04.01 «Прикладные математика и физика».

В соответствии с этим главное внимание в пособии уделено операциям с символьными выражениями, решению линейных и трансцендентных уравнений и их систем, решению обыкновенных дифференциальных уравнений и их систем, основам построения двумерной и трехмерной графики, аппроксимации экспериментальных данных, процедурам импорта числовых значений в Maple из сторонний программ. Каждый раздел пособия содержит заданиями для самостоятельной работы.

1. ОСНОВНЫЕ ОПЕРАЦИИ С СИМВОЛЬНЫМИ ВЫРАЖЕНИЯМИ

При работе с математическими выражениями практически всегда приходиться выполнять множество таких операций, как раскрытие скобок, разложение выражений на множители, приведение подобных членов, различные подстановки, и многие другие.

Для всех пакетов аналитических вычислений такие операции являются базовыми. Maple обладает широкими возможностями для проведения выкладок аналитических преобразований математических формул и выражений, что позволяет сосредоточиться над основными преобразованиями, избегая рутинной работы. Символьных команд в Maple довольно много и их количество продолжает увеличиваться. Здесь мы рассмотрим только основные функции символьных преобразований:

- expand(expr, opt) – раскрывает и расширяет (разлагает) выражение expr в соответствии с дополнительными опциями opt;
- collect(expr, x) – представляет выражение expr по степеням x;
- combine(expr,opt_1..opt_i) – преобразует выражение expr с помощью дополнительных опций opt_1..opt_i;
- factor(expr) – разлагает выражение expr на множители;
- simplify(expr) – упрощает выражение expr;
- numer(dr) – выделение числителя дроби dr;
- denom(dr) – выделение знаменателя дроби dr;
- normal(dr) – нормализация (сокращение) дроби dr;
- Rhs(ur) – выделение правой части уравнения ur;
- lhs(ur) – выделение левой части уравнения ur;

Более подробную информацию об этих и многих других командах и функциях Maple (их более трех тысяч) можно найти в справочной системе.

Справку о любой команде из рабочего листа программы можно получить, выделив эту команду или наведя на нее курсор и нажав клавишу F1. Рассмотрим основные команды более подробно.

expand() .....

Назначение	Раскладывает рациональные выражения на простые дроби, полиномы на полиномиальные разложения, раскрывает скобки, выполняет символьное умножение (деление), раскрывает математические функции.
Синтаксис	expand(expr**,** expr1**,** expr2**, ...,** exprn).
Параметры	• expr – некоторое алгебраическое выражение; • expr1,expr2,...,exprn – дополнительные аргументы, которые используются при преобразовании выражения.

Основное назначение команды expand() – представить произведение некоторого выражения виде эквивалентной суммы. Другими словами, данная команда раскрывает скобки в алгебраическом выражении. Она выполняется для любого полинома. Для частного двух полиномов (рациональная алгебраическая дробь) эта команда раскрывает скобки в числителе и делит каждый член полученного выражения на знаменатель, с которым она не производит никаких преобразований.

Кроме того, данная команда умеет работать с большинством математических функций и знает, как раскрывать скобки в выражениях, содержащих тригонометрические функции: sin(x), cos(x), tg(x), sh(x), th(x), ln(x), exp(x), abs(x), специальные математические функции и многие другие.

Примеры применения команды expand() приведены на рис. 1.1.

expand((x+1)*(x+2));

$$x^2+3x+2$$

expand((x+1)/(x+2));

$$\frac{x}{x+2}+\frac{1}{x+2}$$

expand(sin(x+y));

$$\sin(x)\cos(y)+\cos(x)\sin(y)$$

expand((x+1)*(y+z));

$$xy+xz+y+z$$

expand((x+1)*(y + z), x+1);

$$(x+1)y+(x+1)z$$

expand((x + 1)*(x+2));

$$x^2+3x+2$$

expand((x + 1)/(x + 2));

$$\frac{x}{x+2}+\frac{1}{x+2}$$

expand(sin(x + y));

$$\sin(x)\cos(y)+\cos(x)\sin(y)$$

expand((x + 1)*(y + z));

$$xy+xz+y+z$$

expand((x + 1)*(y + z), x + 1);

$$(x+1)y+(x+1)z$$

expand(ln(x/(1 - x)^2));

$$\ln\left(\frac{x}{(1-x)^2}\right)$$

В последнем примере команда не произвела вычислений, а только лишь повторила исходное выражение. Это произошло потому, что в данном случае такая операция неоднозначная. По умолчанию Maple оперирует с комплексными числами, но в этом выражении это невозможно. Наложим ограничение на переменную *x*, представив ее действительным числом (real):

assume(x, real);
expand(ln(x/(1 - x)^2));

$$\ln\left(\frac{1}{(1-x\sim)^2}\right)+\ln(x\sim)$$

Теперь всё получилось.

Рис. 1.1. Раскрытие выражений с помощью команды expand()

collect() ……..

Назначение	Приведение подобных членов в выражении относительно переменной или подвыражения, в том числе по степеням указанного фрагмента. Может применяться для множества или списка.
Синтаксис	collect(expr, x); collect(expr, x, form, func).
Параметры	• expr – преобразуемое выражение; • x – параметр комплектования, может быть списком или множеством. • form – дополнительный параметр преобразования. Может иметь два значения: ✓ recursive – рекурсивная форма; ✓ distributed – дистрибутивная форма; • func – позволяет задать имя функции или процедуры, по которой будет идти комплектование выражения expr.

Примеры применения команды collect():

Необходимо осуществить приведение подобных членов в следующих выражениях:

1). $x(x+1)+y(x+1)$,

2). $a\ln(x)-\ln(x)\cdot x-x$,

3). $x^2\exp(x)-2x\exp(x)+\frac{x^2}{\exp(x)}-\frac{2x}{\exp(x)}$,

4). $x^3\ y+x^2\ y^3+3$.

Решение примера показано на рис. 1.2.

```
collect(x·(x + 1) + y·(x + 1), x);
```

$$x^2 + (1+y)\,x + y$$

```
collect(a· ln(x) - ln(x) ·x -x, ln(x));
```

$$(a-x)\,\ln(x\ \ -x$$

```
collect(x^2 exp(x) - 2 x exp(x) + x^2/exp(x) - 2 x/exp(x), x);
```

$$\left(e^x+\frac{1}{e^x}\right)x^2+\left(-2\,e^x-\frac{2}{e^x}\right)x$$

```
collect(x^2 exp(x) - 2 x exp(x) + x^2/exp(x) - 2 x/exp(x), exp(x));
```

$$\left(x^2-2x\right)e^x+\frac{x^2-2x}{e^x}$$

```
collect(x^2 exp(x) - 2 x exp(x) + x^2/exp(x) - 2 x/exp(x), x, exp(x));
```

$$\left(e^x\right)\left(e^x+\frac{1}{e^x}\right)x^2+\left(e^x\right)\left(-2\,e^x-\frac{2}{e^x}\right)x$$

Рис. 1.2. Приведение подобных членов с помощью команды collect()

combine() ……..……..

Назначение	Приведение подобных членов в выражении, представленном суммой, произведением или степенями неизвестных.

Синтаксис	combine(**f**);
	combine(**f**, **n**);
	combine(**f**, **n**, **opt1**, **opt2**, ...).

Параметры	• **f –** выражение или список выражений; • **n –** имя или список имен подобных членов; • **opt1 –** дополнительные опции**.**

Команда combine() приводит несколько членов выражения, представленного суммой, произведением или степенями неизвестных, к одному члену, используя разнообразные правила, которые, по существу, противоположны правилам, применяемым командой expand().

Примеры применения команды collect():

Рассмотрим известное тригонометрическое соотношение:

$$\sin(a + b) = \sin(a)\cos(b) + \cos(a)\sin(b)$$

и применим к этому соотношению последовательно команды expand() и combine(). Результат показан на рис. 1.3.

expand (sin(a + b));

$$\sin(a)\cos(b) + \cos(a)\sin(b)$$

combine (sin(a) cos(b)+cos(a) sin(b));

$$\sin(a + b)$$

Рис. 1.3. Применение команд **expand() и combine()**

Как видно, команда expand() использует преобразование выражения слева-направо, тогда как команда combine() действует наоборот: справа-налево.

Пример сравнения команд expand() и combine() показан на рис. 1.4. Для сравнения преобразуем выражение $\sin(a + b)^2$.

```
g := sin(a + b)^2
```

$$g := \sin(a + b)^2$$

```
s:=expand(g);
```

$$s := \sin(a)^2 \cos(b)^2 + 2 \sin(a) \cos(b) \cos(a) \sin(b) + \cos(a)^2 \sin(b)^2$$

```
f:=combine(s);
```

$$f := \frac{1}{2} - \frac{1}{2} \cos(2\,a + 2\,b)$$

Рис. 1.4. Преобразование выражений командами **expand() и combine()**

Как видно из примера, команда combine() преобразовала выражение *s* не к исходному выражению *g*, которое мы раскрыли функцией expand(). Это происходит потому, что Maple осуществляет приведение членов выражения по своим внутренним алгоритмам, которые завершаются, как только получилось (или не получилось) представление в соответствии с идеологией разработчиков команды combine(). В нашем примере – представление через тригонометрическую функцию с аргументом, являющимся линейной комбинацией аргументов тригонометрических функций преобразуемого выражения. Если мы хотим получить исходный вид выражения *g*, то можно воспользоваться командой подстановки subs(), параметры которой определяют, что на что следует заменить в том или ином выражении. Пример такого преобразования показан на рис. 1.5.

```
subs (cos(2*a+2*b)=-2*sin (a+b)^2+1, f);
```

$$\sin(a + b)^2$$

Рис. 1.5. Преобразование выражения командой subs()

Команда combine () «знакома» с практически всеми правилами преобразования элементарных математических функций. Если вторым ее параметром задать одно из следующих имен:

abs	exp	piecewise	Psi	Signum	arctan	icombine
polylog	radical	trig	conjugate	ln	powerrange	

,

которые соответствуют используемым в Maple функциям, то при преобразовании выражения будут использоваться только правила преобразования соответствующих функций. Для функций, правила преобразования которых зависят от

значения их аргументов (arctan) или которые имеют ограничения на значения аргументов (ln, radical), можно задать третий параметр symbolic, который будет предписывать функции combine() не обращать внимания на интервалы изменения аргументов подобных функций, а осуществлять формальные символические преобразования в соответствии с формулами преобразования этих функций.

Для того чтобы оценить действие различных дополнительных опций команды combin(), рассмотрим пример, представленный на рис. 1.6.

F:=4*sin(x)^3+exp(y)*exp(x); #Исходное выражение:

$$F := 4\sin(x)^3 + e^y e^x$$

combine(F);

$$-\sin(3x) + 3\sin(x) + e^{(y+x)}$$

Здесь процедура применена без дополнительных параметров.

combine(F, trig);

$$-\sin(3x) + 3\sin(x) + e^y e^x$$

Здесь дано указание применять преобразования только к тригонометрическим членам.

combine(F, exp);

$$4\sin(x)^3 + e^{(y+x)}$$

Здесь дано указание применять преобразования только к экспоненте.

combine(F, exp, trig);

$$-\sin(3x) + 3\sin(x) + e^{(y+x)}$$

Здесь дано указание применять преобразования и тригонометрическим членам и экспоненциальным.

Рис. 1.6. Применение дополнительных опций команды combine()

factor() ……..……..

Назначение	Разложение выражения на множители.
Синтаксис	factor(f); factor(f, K).
Параметры	• **f –** выражение или список выражений; • K – ключевое слово real или complex или радикал.

Команда factor() разлагает на множители полином от нескольких переменных. Под *полиномом* в Maple понимается выражение, содержащее неизвестные величины. Каждый член в этом выражении представлен в виде произведения целых неотрицательных степеней неизвестных величин с числовым или алгебраическим коэффициентом, т.е. коэффициент может быть целым, дробным, с плавающей точкой, комплексным числом и даже представлять собой алгебраическое выражение с другими переменными.

Эта команда достаточно проста и прозрачна. Для примера рассмотрим два выражения, представленные на рис. 1.7.

```
factor(6*x^2+18*x - 24);
```

$$6\,(x+4)\,(x-1)$$

```
factor((x^3-y^3)/(x^4-y^4));
```

$$\frac{y^2+yx+x^2}{(x+y)\,(y^2+x^2)}$$

Рис. 1.7. Применение команды factor()

При применении этой команды к алгебраической рациональной дроби (отношение двух полиномов) сначала осуществляется приведение дроби к нормальной форме (сокращение общих множителей числителя и знаменателя), а после этого и числитель и знаменатель раскладываются на множители (с учетом поля коэффициентов).

Следует помнить правило: команда раскладывает полином на множители над числовым полем, которому принадлежат коэффициенты полинома. Если все коэффициенты целые, то и в получаемых сомножителях будут только целые коэффициенты и не обязательно будут получены линейные сомножители. Второй необязательный параметр этой команды указывает, над каким числовым полем следует осуществлять разложение полинома. Он может иметь значение real, complex, а также один радикал или список /множество радикалов. Следующий

пример демонстрирует результаты разложения одного и того же полинома над разными числовыми полями (рис. 1.8).

factor (x^3+2); #над полем целых чисел (целые коэффициенты).

$$(x + 1.259921050)(x^2 - 1.259921050\ x + 1.587401052)$$

factor (x^3+2, real); #над полем вещественных чисел

$$(x + 1.259921050)(x^2 - 1.259921050\ x + 1.587401052)$$

factor (x^3+2, complex); #над полем комплексных чисел

$$(x + 1.259921050)(x - 0.6299605249 + 1.091123636I)$$
$$(x - 0.6299605249 - 1.091123636I)$$

factor (x^3+2,2^(1/3)); #над полем целых чисел и радикала 2^(1/3)

$$(x^2 - x \cdot 2^{(1/3)} + 2^{(2/3)})(x + 2^{(1/3)})$$

Рис. 1.8. Разложение выражения над разными полями

normal() ……..

Назначение	Сокращение (нормализация) алгебраической дроби
Синтаксис	normal(f) normal(f, expanded))
Параметр	• f – выражение или список выражений • expanded – дополнительный аргумент

Команда normal() приводит выражение, содержащее алгебраические дроби, к общему знаменателю и упрощает полученную алгебраическую дробь, сократив и числитель и знаменатель на наибольший общий делитель. Команда имеет две формы вызова: normal(f); normal(f, expanded), где f – алгебраическая дробь, а параметр expanded указывает на то, что после сокращения дроби в числителе и знаменателе раскрываются скобки. Если параметр f задан в виде списка, множества, последовательности, ряда, уравнения, отношения или функции, то команда normal() последовательно применяется к компонентам f. Например, для уравнения это означает, что процедура сокращения применяется как к правой, так и к левой части уравнения. В случае выражения с несколькими функциями, аргументы которых представлены алгебраическими дробями, процедура сокращения применяется к аргументу каждой функции. Если выражение нормализации не подлежит, возвращается значение 0.

Примеры применения команды normal() показаны на рис. 1.9.

f:=1/x+1/(x+1)^2+1/(x+1);

$$f := \frac{1}{x} + \frac{1}{(x+1)^2} + \frac{1}{x+1}$$

normal(f);

$$\frac{2x^2+4x+1}{x(x+1)^2}$$

s:=sin (x/(x+1) – x)^2+cos (-x/(x+1)+x);

$$s := \sin\left(\frac{x}{x+1} - x\right)^2 + \cos\left(-\frac{x}{x+1} + x\right)$$

normal(s);

$$\sin\left(\frac{x^2}{x+1}\right)^2 + \cos\left(\frac{x^2}{x+1}\right)$$

normal(1/x+y=x/y+(3*y)/x);

$$\frac{1+yx}{x} = \frac{x^2+3y^2}{yx}$$

Следующий пример демонстрирует работу параметра **expanded**
normal(1/x+x/(x+1));

$$\frac{x+1+x^2}{x(x+1)}$$

normal(1/x+x/(x+1), expanded);

$$\frac{x+1+x^2}{x^2+x}$$

Рис. 1.9. Примеры применения команды normal()

simplify () ……..……..

Назначение	Упрощение выражений

Синтаксис	simplify(expr);
	simplify(expr, n1, n2, ...);
	simplify(expr, assume=prop);
	simplify(expr, symbolic).

Параметры	• expr – выражение или список выражений; • n1, n2… – имя функции или выражения, по которому проводятся упрощения; • prop – дополнительное условие.

Команда simplify() предназначена для упрощения выражений, составленных из чисел, переменных и элементарных функций. Заметим, что Maple может упростить данное выражение, а может и не упростить, так как программа использует свои внутренние алгоритмы упрощения, заложенные разработчиками. При этом результат упрощения может не совсем соответствовать взглядам пользователя на то, в каком виде он хотел бы получить этот результат.

Команда имеет несколько форм вызова. Ее самый простой синтаксис имеет следующий вид: simplify (выражение). В скобках указывается выражение, подлежащее упрощению.

Реально команда simplify() реализована в виде набора процедур упрощения, хранящихся в основной библиотеке Maple. Перечислим основные из них:

- simplify/exp – для упрощения выражений с экспоненциальными функциями;
- simplify/ln – для упрощения выражений с логарифмами;
- simplify/sqrt – для упрощения выражений, содержащих квадратные корни;
- simplify/trig – для упрощения выражении с тригонометрическими функциями;
- simplify/radical – для упрощения выражений с радикалами;
- simplify/power – для упрощения выражений со степенями, экспонентами и логарифмами и т.д.

По умолчанию Maple пытается использовать максимальный набор функций упрощения, подходящий к конкретному выражению. Однако можно задать конкретные процедуры упрощения, тогда только они будут использоваться для

упрощения заданного выражения. Это обеспечивается следующим синтаксисом команды: simplify(выражение, nl, n2,…), где nl, n2 и т.д. являются именами процедур упрощения: RootOf, ln, polar, power, radical, sqrt, trig и т.п.

Упрощения выражений можно проводить с различными видами чисел, например положительными или принадлежащими некоторому отрезку действительных чисел. Это достигается с помощью параметра assume=свойство. Форма вызова команды при этом имеет вид simplify (выражение, assume=свойство), где параметр «свойство» может принимать одно из следующих значений: complex – комплексная область, real – действительная область, positive – положительные действительные числа, integer – целые числа, RealRange (a, b) – интервал (a, b) действительных чисел.

Примеры использования команды упрощения выражений simplify() представлены на рис. 1.10.

d:= 1/sqrt(8)*(((1+sqrt(8))/10)^5+((1-sqrt(8))/10)^5);

$$d := \frac{1}{4}\sqrt{2}\left(\left(\frac{1}{10}+\frac{1}{5}\sqrt{2}\right)^5+\left(\frac{1}{10}-\frac{1}{5}\sqrt{2}\right)^5\right)$$

simplify(d);

$$\frac{401}{200000}\sqrt{2}$$

c:= ln (exp(x))+x*ln (exp(x));

$$c := \ln\left(e^x\right) + x\,\ln\left(e^x\right)$$

simplify(c);

$$\ln\left(e^x\right) + x\,\ln\left(e^x\right)$$

simplify(c, assume=real);

$$x + x^2$$

Рис. 1.10. Примеры использования простого вызова команды simplify()

Как видно из последнего примера, использование команды без параметров не упростило выражения ln (exp(x))+x·ln (exp(x)), тогда как второй оператор с предположением о действительной области изменения переменной x упростил заданное выражение.

При вызове команды упрощения можно последним, или единственным, не считая самого упрощаемого выражения, параметром задать параметр с именем ***symbolic***. В этом случае если выражение содержит многозначные функции, например квадратный корень, то относительно таких функций будет осуществ-

лено формальное символическое упрощение. Это означает, что не будет приниматься во внимание различное поведение многозначных функций при нахождении их аргумента в разных областях комплексной плоскости или действительной оси. Поясним это следующим примером

f:=(sqrt(x^2));

$$f := \sqrt{x^2}$$

simplify(f, symbolic);

$$x$$

Видно, что в случае неоднозначной функции $f := \sqrt{x^2}$ задание параметра symbolic снимает неопределенность, зависящую от знака переменной, используя при этом формальное правило: квадратный корень из квадрата какой-либо величины равен этой величине.

Команда simplify() позволяет задать правила упрощения пользователя. Использование собственных правил для упрощения выражений позволяет получить именно тот его вид, который необходим пользователю для дальнейшей работы. Таким параметром можно дополнительно определить, в какой последовательности должны отображаться неизвестные в упрощенном выражении. Такой параметр задается в двух формах: в виде множества или списка, которые должны быть заключены в фигурные или квадратные скобки соответственно. Если параметр задан в виде множества, то алгоритм упрощения сортирует в выражении неизвестные по убыванию их степени в слагаемых выражения, учитывая степени всех неизвестных, а потом начинает упрощения в соответствии с заданными правилами. В случае списка сначала выражение сортируется по степеням первой неизвестной в списке, затем упрощается в соответствии с заданными правилами и сортируется по степеням второй неизвестной списка.

Пример упрощения выражений в соответствии с правилами пользователя показан на рис. 1.11.

equ:={sin(x)^2+cos(x)^2=1}; # зададим правило пользователя:

$$equ := \{ \sin(x)^2 + \cos(x)^2 = 1 \}$$

e:=sin(x)^3–11*sin(x)^2*cos(x)+3*cos(x)^3-sin(x)*cos(x)+2:

$$e := \sin(x)^3 - 11\sin(x)^2\cos(x) + 3\cos(x)^3 - \sin(x)\cos(x) + 2$$

simplify (e, equ, [sin(x), cos(x)]);

$$14\cos(x)^3 - \sin(x)\cos(x) + 2 - \sin(x)\cos(x)^2 + \sin(x) - 11\cos(x)$$

simplify (e, equ, [cos(x), sin(x)]); # изменим порядок сортировки

$$\sin(x)^3 - 14\sin(x)^2\cos(x) - \sin(x)\cos(x) + 2 + 3\cos(x)$$

Рис. 1.11. Примеры использования правил пользователя

Может показаться, что команда simplify() одна из самых полезных команд Maple. Однако, это не всегда так. Как мы видели, эта команда упрощает выражение в соответствии со своими внутренними представлениями о том, что считать более простым видом выражения. Например, она всегда считает, что сумма проще произведения, хотя в некоторых случаях может оказаться и наоборот.

Задачи для самостоятельного решения

1. Показать, что имеет место следующее тождество:

$$(a^2+b^2+c^2)(x^2+y^2+z^2)-(ax+by+cz)^2=(bx-ay)^2+(cy-bz)^2+(az-cx)^2.$$

2. Доказать следующие тождества:

$$(a^2+b^2)^2+(2ab)^2=(a^2+b^2)^2,$$
$$(6a^2-4ab+4b^2)^3=(3a^2+5ab-5b^2)^3+(4a^2-4ab\_6b^2)^3+(5a^2-5ab-3b^2)^3.$$

3. Доказать тождество:

$$X^2 + XY + Y^2 = Z^3,$$

если

$$X = q^3 + 3pq^2 - p^3,$$

$$Y = -3pq(p + q),$$

$$Z = p^2 + pq + q^2.$$

4. Показать, что

$$(p^2 - q^2)^4 + (2pq + q^2)^4 + (2pq + p^2)^4 = 2(p^2 + pq + q^2)^4.$$

5. Доказать тождество

$$(a + b + c)^4 + (b + c - a)^4 + (c + a - b)^4 + (a + b - c)^4 =$$
$$= 4(a^4 + b^4 + c^4) + 24(b^2c^2 + c^2a^2 + a^2b^2).$$

6. Доказать, что если $a+b+c=2s$, то

$$a(s - a)^2 + b(s - b)^2 + c(s - c)^2 + 2(s - a)(s - b)(s - c) = abc.$$

7. Показать, что если $a + b + c = 0$, то

$$\left(\frac{a-b}{c} + \frac{b-c}{a} + \frac{c-a}{b}\right)\left(\frac{c}{a-b} + \frac{a}{b-c} + \frac{b}{c-a}\right) = 9.$$

8. Показать, что

$$\cos(\alpha+\beta)\sin(\alpha-\beta) + \cos(\beta+\gamma)\sin(\beta-\gamma) + \cos(\gamma+\delta)\sin(\gamma-\delta) +$$
$$+ \cos(\delta+\alpha)\sin(\delta-\alpha) = 0.$$

9. Пусть $a+b+c=2s$ и $a^2+b^2+c^2=2\sigma^2$. Показать, что

$$(\sigma^2 - a^2)(\sigma^2 - b^2) + (\sigma^2 - b^2)(\sigma^2 - c^2) + (\sigma^2 - c^2)(\sigma^2 - a^2) =$$
$$= 4s(s-a)(s-b)(s-c).$$

2. ФУНКЦИИ, УРАВНЕНИЯ, СИСТЕМЫ УРАВНЕНИЙ В MAPLE

2.1. Способы задания функций. Замена переменных

В Maple имеется несколько способов представления функции.

Способ 1. Определение функции с помощью оператора присваивания (**:=**). Например:

```
f:=sin(x)+cos(x);
```

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

Если задать конкретное значение переменной *x*, то будет вычислено значение функции *f* для этого *x*. Например, если продолжить предыдущий пример и вычислить значение **f** при $x = \pi/4$, то следует записать

```
x:= Pi/4;
```

$$x := \frac{\pi}{4}$$

```
f;
```

$$\sqrt{2}$$

После выполнения этих команд переменная *x* будет иметь заданное значение $\pi/4$ до тех пор пока вы не очистите значение этой переменной процедурой: x:='x' или командой restart**.**

Все вычисления в Maple по умолчанию производятся символьно, т. е. результат будет содержать в явном виде иррациональные константы, такие как e, π и др. Чтобы получить приближенное значение в виде числа с плавающей запятой, следует использовать команду: evalf(expr, t),

где expr – выражение; t – точность, выраженная в количестве цифр после запятой. Например, в продолжение предыдущего примера вычислим полученное значение функции приближенно:

```
evalf(%);
```

0.7357588824

Здесь использован символ (**%**) для вызова предыдущей команды.

Способ 2. Определение функции с помощью функционального оператора, который ставит в соответствие набору переменных **(x1, x2,…)** одно или несколько выражений **(f1, f2,…)**.

Например, определение функции двух переменных с помощью функционального оператора выглядит следующим образом:

f:=(x,y)→sin(x+y);

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

f(Pi/2, 0);

1

Обращение к этой функции осуществляется наиболее привычным в математике способом, когда в скобках вместо аргументов функции указываются конкретные значения переменных x и y.

Способ 3. С помощью команды **unapply(expr, x1, x2,…)**, где **expr** – выражение, **x1, x2,…** – набор переменных, от которых оно зависит. Например:

f:=unapply(x^2+y^2, x, y);

$$f := (x, y) -> x^2 + y^2$$

f(-7,5);

74

Рассмотрим следующий пример на выполнение преобразований: определить функцию $f = \sqrt{1 - x^2 - y^2}$. Перейти к полярным координатам $x = \rho\cos\varphi$, $y = \rho\sin\varphi$ и упростить полученное выражение.

Результат преобразований показан ниже.

f:=sqrt(1-x^2-y^2);

$$f = \sqrt{1 - x^2 - y^2}$$

f:=subs({x=rho*cos(phi), y=rho*sin(phi)}, f);

$$f = \sqrt{1 - \rho^2\cos(\phi)^2 - \rho^2\sin(\phi)^2}$$

f:=simplify(%);

$$f = \sqrt{1 - \rho^2}$$

2.2. Решение обыкновенных уравнений

Решение уравнений в аналитическом виде в системе Maple осуществляется с помощью встроенной функции solve(), которая имеет вид

solve(eq, x),

где **eq** – уравнение, записанное в произвольном виде; ***x*** – переменная, относительно которой это уравнение надо разрешить (искомая неизвестная).

Рассмотрим технологию определения корней уравнения в аналитическом виде на примерах.

Пример 2.1. Необходимо определить корни следующих уравнений:

a·x + b = c;

x^2 – a = 0;

2·x^2 + 3·x – 1 = 0;

sin(a·x) + cos(a·x) = 0.

Решение приведено на рис. 2.1.

solve(a*x+b=c, x);

$$-\frac{b-c}{a}$$

ur:=solve(x^2-a=0, x);

$$ur := -\sqrt{a}, \sqrt{a}$$

Когда уравнение имеет несколько решений, которые могут понадобиться для дальнейших расчетов, команде solve() следует присвоить какое-либо имя. В нашем случае ur. Обращение к какому-либо k–му решению данного уравнения производится указанием его имени с номером решения **k** в квадратных скобках: ur[k]. Например:

ur[1];

$$-\sqrt{a}$$

ur[2];

$$\sqrt{a}$$

solve(2*x^2 + 3*x -1, x);

$$-\frac{3}{4}+\frac{\sqrt{17}}{4}, \; -\frac{3}{4}-\frac{\sqrt{17}}{4}$$

solve(sin(a*x)+cos(a*x) = 0, x);

$$-\frac{\pi}{4}$$

Как видно из последнего примера, найдено только одно (главное) решение. Однако можно найти все периодические решения, предварительно выполнив следующую команду: _EnvAllSolutions:= true;

EnvAllSolutions:= true;

$$\_EnvAllSolutions := true$$

solve(sin(a*x)+cos(a*x) = 0, x);

$$\frac{\pi(4\_Z1 \sim -1)}{4\,a}$$

Рис. 2.1. Технология определения корней уравнения в аналитическом виде

Команда solve() способна решать и, так называемые функциональные уравнения, где неизвестная представляет собой некую функцию от аргумента. Конечно, подобные задачи редко встречаются в экспериментальной практике, но могут быть полезны при установлении эмпирических зависимостей.

Пример 2.2. Необходимо найти функцию f(x), удовлетворяющую уравнению: $f^2(x) - 2 \cdot f(x) = x$.

Решение показано на рис.2.2.

F:=solve(f(x)^2-3*f(x)+2*x, f);

F:= **proc**(*x*) RootOf(_*Z*^2 - 3*_*Z* + 2**x*) **end.**

В результате получается решение в неявном виде: RootOf. Однако Maple способен работать с такими решениями. Неявное решение функционального уравнения можно попытаться преобразовать в какую-либо элементарную функцию с помощью команды convert(), как показано ниже.

f:=convert(F(x), radical); $f := \frac{3}{2} + \frac{1}{2}\sqrt{9 - 8x}$

Рис. 2.2. Пример решения функционального уравнения

2.3. Численное решение уравнений

Встроенная функция solve() всегда пытается найти решение в аналитическом виде. Это не означает, что ее нельзя использовать для получения корней в численном виде. Просто для этого придется использовать функции evalf() или convert().

Для численного решения уравнений, в тех случаях когда уравнения не имеют аналитических решений, может быть полезна специальная команда:

fsolve(Ur, Var, Option),

где первым ее параметром указывается решаемое уравнение Ur, после чего следует ввести переменную Var, относительно которой решается уравнение. Кроме того, функция допускает использование некоторых опций Option. Основные опции функции fsolve() указаны ниже:

- complex – находит один или все корни полинома в комплексной форме;
- filldigts – 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 и обеспечивает поиск корней только в заданном интервале.

Функция fsolve() может решать все уравнения, которые решает функция solve(). Ее отличие – лишь в форме представления ответов, которые она представляет сразу в форме вещественных или комплексных чисел.

Пример 2.3. Необходимо определить корни следующих уравнений:

$\cos(x) = x$;

$2 \cdot x^2 + 3 \cdot x - 1 = 0$;

$2 \cdot x^3 + x - 1 = 10$;

$x \cdot (x - 2)\,(x^2 + 4) = 0$;

$x^2 = \cos(x)$.

Решение приведено на рис. 2.3

```
restart:
x:=fsolve(cos(x)=x,  x);
                                  x:=.7390851332
fsolve(2*x^2+3*x-1 = 0, x);
                          -1.780776406, 0.2807764064
fsolve(2*x^3+x-1 = 10, x);
                                  1.670849626
fsolve(2*x^3+x-1 = 10, x, complex);
                    -0.835424813224904 - 1.61052906079201 I,
                    -0.835424813224904 + 1.61052906079201 I,
                                  1.67084962644981
fsolve(x*(x-2)*(x^2+4) = 0, x = 0 .. 2);
                                  0, 2
fsolve(x^2 = cos(x);
                                  0.82413231
```

Рис. 2.3. Технология определения корней уравнения в численном виде

2.4. Решение систем уравнений

Системы уравнений решаются с помощью команды **solve({eq1, eq2,…},{x1, x2,…})**, только теперь в параметрах команды следует указывать в первых фигурных скобках через запятую уравнения, а во вторых фигурных скобках перечисляются через запятую переменные, относительно которых требуется решить систему. При решении трансцендентных уравнений для получения решения в явном виде перед командой solve() следует ввести дополнительную команду _EnvExplicit:= true.

Пример 2.4. Найти все решения системы уравнений $\begin{cases} x^2 - y^2 = 1 \\ x^2 + xy = 2 \end{cases}$.

Результат расчетов показан на рис. 2.4.

eq:={x^2-y^2 = 1, x^2 + x·y = 2};

s:=solve(eq,{x, y});

$$s := \{x = \frac{2}{3}\sqrt{3}, y = \frac{1}{3}\sqrt{3}\}, \{x = -\frac{2}{3}\sqrt{3}, y = -\frac{1}{3}\sqrt{3}\}$$

Теперь найдем сумму первого набора решений:

X1:=subs(s[1],x):

Y1:=subs(s[1], y):

X1 +Y1: evalf(%);

1.732059808

Рис. 2.4. Пример решения системы уравнений

Если вам будет необходимо для дальнейших вычислений использовать полученные решения уравнений, то команде solve() следует присвоить какое-нибудь имя. Для этого служит команда присвоения имени: assign(name). После чего над решениями можно будет производить математические операции. Пример такого присвоения показан на рис. 2.5.

s:=solve({a·x - y = 1.5·x+a·y = 1},{x, y});

$$\mathbf{s:=\ \{} x = \frac{a+1}{5+a^2}, y = \frac{a-5}{5+a^2} \mathbf{\}}$$

assign(s); simplify(x-y);

$$\frac{6}{5+a^2}$$

Рис. 2.5. Пример присвоения имени решениям системы уравнений

Пример 2.6. Найти решения системы трансцендентных уравнений и упростить их:

$$\begin{cases} 7 \cdot 3^x - 3 \cdot 2^{z+y-x+2} = 15 \\ 2 \cdot 3^{(x+1)} + 3 \cdot 2^{(z+y-x)} = 66 \\ \ln(x+y+z) - 3 \cdot \ln(x) - \ln(y \cdot z) = -\ln(4). \end{cases}$$

Решение системы трансцендентных уравнений и упрощения вида решений показано на рис. 2.6.

```
eq:={7*3^x-3*2^(z+y-x+2) = 15, 2*3^(x+1)+
      + 3*2^(z+y-x) = 66, ln(x+y+z)-3*ln(x) - ln(y*z)=-ln(4) }:
_EnvExplicit:=true:
s:=solve(eq,{x, y, z});
                                  s := ()
# Увы, функция solve() не смогла найти решения. Конечно, можно поработать # с
исходными уравнениями, но можно поступить проще, использовав
# функцию fsolve()
s:=fsolve(eq,{x, y, z});
simplify(s[1]);simplify(s[2]);
                    {x = 2, y = 3, z = 1}, {x = 2, y = 1, z = 3}
# Теперь решения найдены!
```

Рис. 2.6. Пример решения системы трансцендентных уравнений

Задания для самостоятельного решения

1. Записать функцию $f(x,y)=\left(\dfrac{\operatorname{arctg}(x+y)}{\operatorname{arctg}(x-y)}\right)^2$ в виде функционального оператора и вычислить ее значения при x=1, y=0 и при $x=(1+\sqrt{3})/2$, $y=(1-\sqrt{3})/2$.
2. Записать функцию $f(x,y)=\dfrac{x^3y^2-x^2y^3}{(xy)^5}$ с помощью оператора присваивания и вычислить ее значение при x=a, y=1/a, используя команду подстановки subs().
3. Найти все точные решения системы уравнений $\begin{cases}x^2-5xy+6y^2=0,\\ x^2+y^2=10.\end{cases}$ в аналитическом виде.
4. Найти все решения тригонометрического уравнения $\sin^4 x-\cos^4 x=1/2$.
5. Найти численное решение уравнения $e^x=2(1-x)^2$.

6. Решить уравнение $\dfrac{\sqrt{a+x}+\sqrt{a-x}}{\sqrt{a+x}-\sqrt{a-x}}=\sqrt{b}.\quad (a>0)$.

7. Решить систему уравнений и проверить решение подстановкой

$$\begin{cases} x^2+y^2=cxyz \\ x^2+z^2=bxyz \\ y^2+z^2=cxyz. \end{cases}$$

8. Численно решить систему уравнений и проверить решение подстановкой

$$2.7\ x_1+3.3\ x_2+1.3\ x_3=2.1$$

$$3.5\ x_1-1.7\ x_2+2.8\ x_3=1.7$$

$$4.1\ x_1+5.8\ x_2-1.7\ x_3=0.8.$$

9. Численно решить систему уравнений и проверить решение подстановкой

$$5.4\ x_1-6.2\ x_2-0.5\ x_3=0.52$$
$$3.4\ x_1+2.3\ x_2+0.8\ x_3=-0.8$$
$$2.4\ x_1-1.1\ x_2+3.8\ x_3=1.8\ .$$

3. РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Для нахождения аналитических решений простых дифференциальных уравнений и их систем, в том числе с начальными условиями (задача Коши) в *Maple,* применяется команда ***dsolve***

- **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 началь-ными условиями;
- formal series – решение в форме степенного многочлена;
- numeric – решение в численном виде.

Для решения задачи Коши в параметры функции dsolve() необходимо включить 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, _Cl2* 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 • inversejinear • exact • linear • homogeneous	• Abel • Bernoulli • Chini • pot_sym • separable

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.

3.1. Аналитическое решение дифференциальных уравнений

Для нахождения аналитических решений обыкновенных дифференциальных уравнений в *Maple* обычно применяется команда:

dsolve(ODE, y(x), extra_args).

Пояснения параметров даны выше. Общее решение дифференциального уравнения зависит от произвольных постоянных, число которых равно порядку дифференциального уравнения. В Maple такие постоянные, как правило, обозначаются как *_C1*, *_C2*, и т.д.

Общее решение неоднородного линейного дифференциального уравнения всегда выводится так, чтобы была четко видна, структура этого решения. Как известно, общее решение неоднородного линейного дифференциального уравнения равно сумме общего решения соответствующего однородного дифференциального уравнения и частного решения этого же неоднородного дифференциального уравнения. Поэтому в строке вывода решение неоднородного линейного дифференциального уравнения всегда состоит из слагаемых, которые содержат произвольные постоянные (это общее решения соответствующего однородного дифференциального уравнения), и слагаемых без произвольных постоянных (это частное решения этого же неоднородного дифференциального уравнения).

Команда dsolve выдает решение дифференциального уравнения в невычисляемом формате. Для того чтобы с решением можно было бы работать далее (например, построить график решения), следует отделить правую часть полученного решения командой rhs(%).

Рассмотрим ряд примеров.

Пример 3.1. Найти общее решение дифференциальных уравнений (рис. 3.1).

$y' + y \cdot \cos(x) = \sin(x) \cdot \cos(x)$,

$y'' - 2 \cdot y' + y = \sin(x) + e^{-x}$.

restart:
de:=diff(y(x), x)+y(x)*cos(x)=sin(x)*cos(x);

$$de := \left(\frac{\partial}{\partial x} y(x)\right) + y(x)\cos(x) = \sin(x)\cos(x)$$

dsolve(de,y(x));

$$y(x) = \sin(x) - 1 + e^{(-\sin(x))} \_C1$$

deq:=diff(y(x),x$2)-2*diff(y(x), x)+y(x) =sin(x)+exp(-x);

$$deq := \left(\frac{\partial^2}{\partial x^2} y(x)\right) - 2\left(\frac{\partial}{\partial x} y(x)\right) + y(x) = \sin(x) + e^{(-x)}$$

dsolve(deq,y(x));

$$y(x) = \_C1 e^{x} + \_C2 e^{x} x + \frac{1}{2}\cos(x) + \frac{1}{4} e^{(-x)}$$

Рис. 3.1. Решение дифференциальных уравнений

Так как второе уравнение было второго порядка, то полученное решение содержит две произвольные константы *_C1* и *_C2*. Первые два слагаемых представляют собой общее решение соответствующего однородного дифференциального уравнения, а вторые два – частное решение неоднородного дифференциального уравнения.

Пример 3.2. Решить одно и то же дифференциальное уравнение $\sin(x)\left(\frac{d}{dx} y(x)\right) - \cos(x)\, y(x) = 0$ разными методами.

Результаты расчетов показаны на рис. 3.2.

restart:

ode_L := sin(x)*diff(y(x), x)-cos(x)*y(x)=0;

$$ode\_L := \sin(x)\left(\frac{d}{dx}y(x)\right)-\cos(x)\,y(x)=0$$

dsolve(ode_L, [linear], uselnt);

$$y(x)=\_Cl\,e^{\left(\int\frac{\cos(x)}{\sin(x)}dx\right)}$$

value(%);

$$y(x)=\_Cl\ \sin(x)$$

ode_L := sin(x)*diff(y(x),x)-cos(x)*y(x)=0;

$$ode\_L := \sin(x)\left(\frac{d}{dx}y(x)\right)-\cos(x)\,y(x)=0$$

dsolve(ode_L, [separable], useInt);

$$\int\frac{\cos(x)}{\sin(x)}dx-\int\frac{1}{\_a}d\_a\ +\_Cl=0$$

value(%);

$$\ln(\sin(x))-\ln(y(x))+\_Cl=0$$

mu := intfactor(ode_L);

$$\mu := \mathrm{int}\,factor\left(\sin(x)\left(\frac{d}{dx}y(x)\right)-\cos(x)\,y(x)=0\right)$$

dsolve(mu*ode_L, [exact], uselnt);

$$y(x)=\frac{\_Cl}{\frac{1}{2}\tan\left(\frac{1}{2}x\right)+\frac{\frac{1}{2}}{\tan\left(\frac{1}{2}x\right)}}$$

Рис. 3.2. Решение дифференциального уравнения разными методами

3.2. Решение задачи Коши или краевой задачи

Команда ***dsolve()*** может найти решение задачи Коши или краевой задачи, если помимо дифференциального уравнения задать начальные или краевые условия для неизвестной функции. Для обозначения производных в начальных или краевых условиях удобно использовать дифференциальный оператор D, например условие y''(0)=2 можно записать в виде $(D@@2)(y)(0)=2$ или условие $y'(1)=0$: $D(y)(1)=0$. Напомним, что производная n-го порядка записывается в виде $(D@@n)(y)$.

Рассмотрим следующие примеры.

Пример 3.3. Найти решение задачи Коши: $y^{(4)}+y''=2\cos x$, $y(0)=-2$, $y'(0)=1$, $y''(0)=0$, $y'''(0)=0$ (рис.3.3).

```
restart;
de:=diff(y(x),x$4)+diff(y(x),x$2)=2*cos(x);
```

$$de := \left(\frac{\partial^4}{\partial x^4}\,\mathrm{y}(x)\right)+\left(\frac{\partial^2}{\partial x^2}\,\mathrm{y}(x)\right)=2\cos(x)$$

```
cond:=y(0)=-2, D(y)(0) = 1, (D@@2)(y)(0)=0, (D@@3)(y)(0) = 0;
```

$$cond:=\mathrm{y}(0)=-2,\ \mathrm{D}(\mathrm{y})(0)=1,\ (\mathrm{D}^{(2)})(\mathrm{y})(0)=0,\ (\mathrm{D}^{(3)})(\mathrm{y})(0)=0$$

```
dsolve({de,cond},y(x));
```

$$\mathrm{y}(x)=-2\cos(x)-x\sin(x)+x$$

Рис. 3.3. Решение задачи Коши

Пример 3.4. Найти решение краевой задачи: $y''+y=2x-\pi$, $y(0)=0$, $y(\pi/2)=0$ (рис. 3.4).

```
restart;
de:=diff(y(x),x$2)+y(x)=2*x - Pi;
```

$$de:=\left(\frac{\partial^2}{\partial x^2}\,\mathrm{y}(x)\right)+\mathrm{y}(x)=2x-\pi$$

```
cond:=y(0)=0, y(Pi/2)=0;
```

$$cond := y(0)=0,\ y\left(\frac{\pi}{2}\right)=0$$

```
dsolve({de, cond},y(x));
```

$$\mathrm{y}(x)=2x-\pi+\pi\cos(x)$$

Рис. 3.4. Решение краевой задачи

3.3. Системы дифференциальных уравнений

Для решения системы дифференциальных уравнений (или задачи Коши) в команде dsolve() необходимо указать: **dsolve({sys},{x(t),y(t),…}),** где **sys** – система дифференциальных уравнений, **x(t), y(t),…** – набор неизвестных функций.

Пример 3.5. Найти решение системы дифференциальных уравнений (рис.3.5):

$$\begin{cases} x' = -4x - 2y + \dfrac{2}{e^t - 1}, \\ y' = 6x + 3y - \dfrac{3}{e^t - 1}. \end{cases}$$

Результаты расчетов показаны на рис. 3.5.

```
sys:={diff(x(t), t) =- 4*x(t)-2*y(t)+2/(exp(t)-1),
diff(y(t), t) = 6*x(t)+3*y(t)-3/(exp(t)-1)}:
dsolve(sys,{x(t),y(t)});
```

$$\{\mathrm{x}(t) = -3\_C1 + 4C1\_e^{(-t)} - 2C2\_ + 2C2\_e^{(-t)} + 2e^{(-t)}\ln(e^t - 1),$$

$$\mathrm{y}(t) = 6\_C1 - 6\_C1e^{-t} - 3\_C2e^{(-t)} + 4\_C2 - 3e^{(-t)}\ln(e^{(t)} - 1)\}$$

Рис. 3.5. Решение системы дифференциальных уравнений

Итак, найдены две функции $x(t)$ и $y(t)$, которые зависят от двух произвольных постоянных *_C1* и *_C2*.

3.4. Приближенное решение дифференциальных уравнений с помощью степенных рядов

Для многих типов дифференциальных уравнений точное аналитическое решение не может быть найдено. В этом случае дифференциальное уравнение можно решить с помощью приближенных методов и, в частности, с помощью разложения в степенной ряд неизвестной функции.

Чтобы найти приближенное решение дифференциального уравнения в виде степенного ряда, в команде dsolve следует после переменных указать параметр type = series (или просто series). Для того чтобы указать порядок разложения n, т.е. порядок степени, до которой производить разложение, следует перед командой dsolve вставить определение порядка с помощью команды Order:=n.

Если ищется общее решение дифференциального уравнения в виде разложения в степенной ряд, то коэффициенты при степенях *x* найденного разложения будут содержать неизвестные значения функции в нуле y(0) и ее производных D(y)(0), (D@@2)(y)(0) и т.д. Полученное в строке вывода выражение будет иметь вид, похожий на разложение искомого решения в ряд Маклорена, но с другими коэффициентами при степенях *x*. Для выделения частного решения следует задать начальные условия y(0) = y1, D(y)(0) = y2, (D@@2)(y)(0) = y3 и т.д., причем количество этих начальных условий должно совпадать с порядком соответствующего дифференциального уравнения.

Разложение в степенной ряд имеет тип ***series***, поэтому для дальнейшей работы с этим рядом его следует преобразовать в полином с помощью команды convert(%, polynom), а затем выделить правую часть полученного выражения командой rhs(%).

Пример 3.6. Найти решение задачи Коши: $y' = y + xe^y$, $y(0) = 0$ в виде степенного ряда с точностью до 5-го порядка.

Результаты расчетов показаны на рис. 3.6.

```
restart;

Order:=5:

dsolve({diff(y(x),x)=y(x)+x*exp(y(x)), y(0)=0}, y(x), type=series);
```

$$\mathrm{y}(x) = \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{6}x^4 + \mathrm{O}(x^5)$$

Рис. 3.6. Решение задачи Коши

В полученном решении слагаемое $\mathrm{O}(x^5)$ означает, что точность разложения была до 5-го порядка.

Пример 3.7. Найти общее решение дифференциального уравнения $y''(x) - y^3(x) = e^{-x}\cos x$ в виде разложения в степенной ряд до 4-го порядка. Найти разложение при начальных условиях: $y(0) = 1$, $y'(0) = 0$.

Результаты расчетов показаны на рис. 3.7.

```
restart:
Order:=4:
de:=diff(y(x), x$2)-y(x)^3=exp(-x)*cos(x):
f:=dsolve(de,y(x), series);
```

$$f := \mathrm{y}(x) = \mathrm{y}(0) + \mathrm{D}(y)(0)x + \left(\frac{1}{2}\mathrm{y}(0)^3 + \frac{1}{2}\right)x^2 +$$

$$\left(\frac{1}{2}\mathrm{y}(0)^2\,\mathrm{D}(y)(0) - \frac{1}{6}\right)x^3 + \mathrm{O}(x^4)$$

```
#Для нахождения частого решения осталось задать начальные условия:
y(0):=1: D(y)(0):=0: f;
```

$$\mathrm{y}(x) = 1 + x^2 - \frac{1}{6}x^3 + \mathrm{O}(x^4)$$

Рис. 3.7. Общее решение дифференциального уравнения

Пример 3.8. Найти приближенное решение в виде степенного ряда до 6-го порядка и точное решение задачи Коши: $y''' - y' = 3(2 - x^2)\sin x$, $y(0) = 1$, $y'(0) = 1$.

Результаты расчетов показаны на рис. 3.8.

```
restart;
Order:=6:
de:=diff(y(x), x$3)-diff(y(x), x)= 3*(2-x^2)*sin(x);
```

$$de := \left(\frac{\partial^3}{\partial x^3}\,y(x)\right) - \left(\frac{\partial}{\partial x}\,y(x)\right) = 3(2 - x^2)\sin(x)$$

```
cond:=y(0) = 1, D(y)(0) = 1, (D@@2)(y)(0) = 1;
```

$$cond := y(0)=1,\ D(y)(0)=1,\ D^{(2)}(y)(0)=1$$

```
dsolve({de, cond},y(x));
```

$$y(x) = \frac{21}{2}\cos(x) - \frac{3}{2}x^2\cos(x) + 6x\sin(x) - 12 + \frac{7}{4}e^{x} + \frac{3}{4}e^{(-x)}$$

```
y1:=rhs(%):
dsolve({de, cond},y(x), series);
```

$$y(x) = 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{7}{24}x^4 + \frac{1}{120}x^5 + O(x^6)$$

Рис. 3.8. Приближенное решение задачи Коши

Тип решения дифференциального уравнения в виде ряда есть series, поэтому для дальнейшего использования такого решения (вычислений или построения графика) его обязательно следует конвертировать в полином с помощью команды convert().

3.5. 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. Такая процедура возвращает особый тип данных, позволяющих найти решение в любой точке или построить график решения (или решений).

В список параметров функции dsolve() можно явным образом включить указание в метод решения, например опция method = dverk78 задает решение непрерывным методом Рунге– Кутта порядка 7 или 8. 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;|
- **0dverk78** – 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;
- **Isode** – одна из восьми версий Ливенморского решателя жестких дифференциальных уравнений;
- **taylorserles** – метод разложения в ряд Тейлора.

Пример 3.9. Найти численное решение задачи Коши: $y''-x\sin(y)=\sin 2x$, $y(0)=0$, $y'(0)=1$.

Результаты расчетов показаны на рис. 3.9.

```
restart; Ordev=6:
eq:=diff(y(x),x$2)-x*sin(y(x))=sin(2*x):
cond:=y(0)=0, D(y)(0)=1:
de:=dsolve({eq,cond},y(x),numeric);
                    de:=proc(rkf45_x)...end
```

Рис. 3.9. Численное решение задачи Коши

В строке вывода появляется сообщение о том, что при решении использован метод *rkf45*. Во избежание вывода строк, не несущих полезной информации, рекомендуется отделять промежуточные команды двоеточием. Если необходимо получить значение решения при каком-то фиксированном значении переменной *x* (при этом дополнительно будет выведено значение производной решения в этой точке), например, при *x* = 0.5, то следует набрать

```
de(0.5);
```

$$\left[x=.5,\ \mathrm{y}(x)=.54492611538626 30,\ \frac{\partial}{\partial \mathrm{x}}\,\mathrm{y}(x)=1.272503082225380\right]$$

Задания для самостоятельного решения

1. Найти общее решение дифференциального уравнения:

$$y''-2y'-3y = xe^{4x}\sin x.$$

2. Найти фундаментальную систему решений дифференциального уравнения:

$$y'''+y''=1-6x^2e^{-x}.$$

3. Найти решение задачи Коши: $y'''-y'=\operatorname{tg}x,\ y(0)=3,\ y'(0)=-1,\ y''(0)=1.$

4. Найти решение системы дифференциальных уравнений:

$$\begin{cases} x''+5x'+2y'+y=0 \\ 3x''+5x+y'+3y=0 \end{cases}$$

при начальных условиях $x(0)=1$, $x'(0)=0$; $y(0)=1$.

5. Найти решение нелинейного уравнения $y''+y=y^2$ при начальных условиях $y(0)=2a$, $y'(0)=a$ в виде разложения в степенной ряд до 6-го порядка.

6. Построить график численного решения задачи Коши $y'=\sin(xy)$, $y(0)=1$.

7. Численно решить задачу Коши: $y''=xy'-y^2,\ y(0)=1,\ y'(0)=2$. Найти приближенное решение этого уравнения в виде разложения в степенной ряд.

8. Решить уравнение и построить график решения. y'3 + (3x – 6) y' = 3y .

9. Решить уравнение и построить график решения y'' – 5y' = 3x2 + sin(5x).

10. Решить систему дифференциальных уравнений

$$\begin{cases} \dfrac{dx}{dt}=x-1 \\ \dfrac{dy}{dt}=x+2\cdot y-3. \end{cases}$$

11. Решить уравнения:

$$y'=2\ln x+x-2,\ y(0)=1;$$
$$y''=3y-e^{2x}+x-1,\ y(0)=1,\ y'(0)=0;$$
$$y'''=-5x-1,\ y(1)=y'(1)=y''(1)=0.$$

4. ТИПОВЫЕ СРЕДСТВА ПОСТРОЕНИЯ ДВУМЕРНЫХ ГРАФИКОВ В MAPLE

Система аналитических вычислений Maple обладает не только широкими возможностями в реализации алгоритмов аналитических вычислений, но и развитой графикой. Здесь доступны построения двумерных кривых, сложных трехмерных поверхностей, а также анимация двумерных и трехмерных изображений.

В само ядро (основную библиотеку) Maple встроено две графические команды: *plot()* и *plot3d()*. Первая предназначена для построения графиков функций одной переменной (двумерная графика), вторая позволяет строить графические изображения поверхностей и трехмерных кривых (пространственная графика).

Универсальные графические команды собраны в пакетах *plots и plottools*. Чтобы воспользоваться их графическими средствами, необходимо обязательное подключение соответствующих пакетов командой *with (plots)* или *with (plottools)*.

4.1. Описание процедуры двумерной графики plot()

Процедура построения двумерных графиков задается в виде:

Синтаксис процедуры	Аргументы процедуры
plot(f, x) ***plot(f, x, [v]*)*** ***plot (f, x, [v], [o])***	***f*** – визуализируемая функция (или функции); ***x*** – аргумент функции с указанием области ее изменения: $x = x_{min} \,..\, x_{max}$; ***v*** – необязательная переменная, определяющая область изменения функции: $y = y_{min} \,..\, y_{max}$; ***o*** – необязательный параметр или набор параметров, задающих стиль построения графика (толщину и цвет кривых, тип кривых, метки на них и т. д.).

*) Здесь и далее все необязательные параметры заключены в квадратные скобки.

В простейшем случае для построения графика достаточно указать функцию *f(x)*, график которой необходимо построить и, через запятую, аргумент этой функции, например, *p1ot(f(x) , x)*.

Если диапазон изменения аргумента не задан, т.е. второй параметр представляет собой просто имя независимой переменной, то по умолчанию принимается интервал от –10 до +10. Второй параметр (с диапазоном или нет) обязательно должен присутствовать при задании графика функции процедурой *plot()*.

Команда *plot()* может отображать графики функций не только на конечном интервале изменения независимой переменной, но и на бесконечном:

- ***p1ot(f(x), x = xmin . . infinity)***.

Исследуемая функция может быть включена в процедуру *p1ot()* как в явном виде, заданном только своим именем, например *plot(sin(x),x),* так и заранее определена (предварительно рассчитана) в виде функциональной зависимости и задана ее идентификатором: *a:=x-> f(x); plot(a(x),x)* или просто с помощью оператора присваивания: *a:= f(x); plot(a, x).*

Пример 4.1. Проверить, действительно ли любой из перечисленных выше способов формирования процедуры plot() эквивалентен, т.е. приводит к одинаковому результату (рис. 4.1).

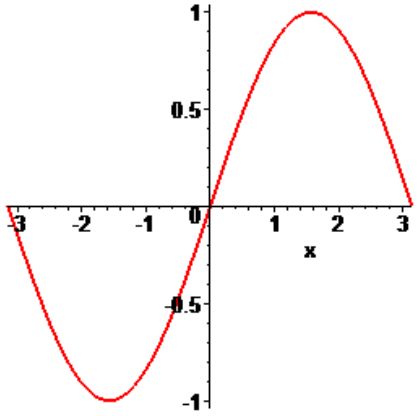
Наберите в командной строке:	• Результат исполнения команды
plot(sin(x), x= -Pi..Pi);	
a:=sin(x): **plot(a, x= - Pi..Pi);**	
a:=x->sin(x): **plot(a(x), x=-Pi..Pi);**	

Рис. 4.1. Решение примера 4.1

Обратите внимание, что в случае задания исследуемой функции с помощью оператора присваивания указание аргумента функции не обязательно.

4.2. Параметры процедуры plot()

Как правило, графики строятся сразу в достаточно приемлемом виде. Это достигается тем, что многие типовые параметры графиков заданы в Maple по умолчанию. Однако язык общения Maple позволяет указать параметры графика – т.н. опции и явным образом. Опции определяют вид отображаемого графика: толщину, цвет и тип линий графика, тип координатных осей, размещение надписей, легенды, размер, гарнитуру шрифта и многое другое, необходимое для создания полноценной графической иллюстрации.

Ниже приведены наиболее употребительные стилевые опции процедуры *plot()*. Дано их краткое описание и примеры использования.

Параметр	Описание параметра
axes	Определяет тип координатных осей графика. Допускаются значения: FRAME *(рамка)* – точка пересечения осей в левом нижнем углу рисунка, BOXED (обрамление) – график отображается в рамке с осями по левой вертикальной и нижней граням,

Параметр	Описание параметра
	NORMAL – обычные, пересекающиеся в начале координат оси, NONE – координатные оси не отображаются.
> plot(f(x),x, **axes** = frame);	

axesfont	Задает шрифт для отображения надписей координатной шкалы. Значение опции задается в виде списка [тип шрифта, стиль, размер]. Возможные значения:
	• Тип шрифта: TIMES, COURIER, HELVETICA, SYMBOL;
	• Стиль шрифта:
	  • для гарнитуры TIMES возможны значения ROMAN, BOLD, ITALIC, BOLDITALIC;
	  • для гарнитур COURIER и HELVETICA стиль можно опустить или задать BOLD, OBLIQUE, BOLD OBLIQUE.
	  • Для шрифта SYMBOL стиль не указывается.
	• размер шрифта – задает размер шрифта в пунктах (point).

> plot(f(x),x, **axesfonts** = [TIMES, ITALIC,14]);	
color	Задает цвет линии для отображения графика функции. Допускается использование цветов: aquamarine (зеленовато-голубой), black (черный), blue (синий), navy (темно-синий), coral (светло-красный), cyan(голубой), brown (коричневый), gold (ярко-желтый), green (зеленый), gray (серый), khaki (хаки), magenta (пурпурный), maroon (красно-коричневый), orange (оранжевый), pink (розовый), plum (темно-фиолетовый), red (красный), sienna (охра), an (желто-коричневый), turquoise (бирюзовый), violet (фиолетовый), wheat (бледно-желтый), white (белый), и yellow (желтый). Кроме того, пользователь может определить собственный цвет.
> plot(f(x), x, **color** = brown);	
coords	Задает систему координат, в которой отображается график. По умолчанию используется декартова система координат. Кроме декартовой, поддерживаются такие координатные системы как: bipolar (биполярная), cardioid (кардиоидная), cassinian (кассинианова), elliptic (эллиптическая), hyperbolic (гиперболическая), invcassinian (инверсная кассинианова), invelliptic (инверсная эллиптическая), logarithmic(логарифмическая), logcosh (логарифмо-гиперболическая), maxwell (максвеллова), parabolic (параболическая), polar (полярная), rose (круговая).
> plot(f(x), x, ***coords*** = *polar*);	
filled	Если задать значение этого параметра равным *true*, то область между графиком функции и осью абсцисс будет закрашена цветом, соответствующим цвету линии графика, заданным в опции *color*. Значение по умолчанию – *false.*
> plot(f(x), x, *color=blue*, ***filled*** = *true*);	
font	Определяет шрифт для отображения текстовых надписей на графике. Значением параметра является список с типом, стилем и размером шрифта аналогично параметру axesfont.

Параметр	Описание параметра
> plot(f(x), x, **font=[** TIMES, ITALIC,14]**);**	
labels	Параметр предназначен для того, чтобы задавать надписи для осей координат. В качестве значения параметра указывается список из двух строк: первая является надписью для оси абсцисс, вторая – для оси ординат. Если параметр не указан, то отображаются названия переменных.
> plot(f(x),x, ***labels*** **=[** "Ось абсцисс","Ось ординат"]**);**	
labeldirections	Параметр задает ориентацию надписей у координатных осей. Значением параметра является список с двумя элементами, каждый из которых может принимать значение либо HORIZONTAL (по горизонтали), либо VERTICAL (по вертикали). По умолчанию обе надписи ориентируются по горизонтали.
> plot(f(x),x, *labels* =["Ось абсцисс","Ось ординат"], **labeldirections** =[HORIZONTAL, VERTICAL]);	
labelfont	Задает параметры шрифта надписей у координатных осей. Допустимые значения те же, что и у параметров font и axesfont.
> plot(f(x),x, *labels* =["Ось абсцисс","Ось ординат"], labeldirections= [HORIZONTAL, VERTICAL), **labelfont**=[TIMES, ITALIC,14]);	
legend	Задает вывод легенды – обозначения кривых для нескольких графиков. Значением параметра есть список, элементы которого – строки, являющиеся легендами этих графиков.
> plot(sin(x),x, labels =["Ось абсцисс","Ось ординат"],**legend**=["sin(x)"]);	
linestyle	Определяет тип линии графика. Значением может быть целое число в диапазоне от 1 до 4 или, что одно и то же, одно из перечисленных названий: solid (сплошная), DOT (пунктирная), dash (штрихованная), DASHDOT (штрихпунктирная).
> plot(sin(x), x, labels = ["Ось абсцисс","Ось ординат"], **linestyle** =3);	
numpoits	Задает минимальное количество вычисляемых точек, по которым строится график. Значение по умолчанию равно 50.
> plot(sin(x), x, labels =["Ось абсцисс","Ось ординат"], **numpoits** =200);	
scaling	Задает масштаб отображения графика. Возможны два значения: CONSTRAINED (единый) и UNCONSTRAINED (различный). В первом случае масштабы по оси абсцисс и ординат одинаковы. Во втором случае (UNCONSTRAINED) рисунок масштабируется таким образом, что длины координатных осей равны линейным размерам области вывода рисунка. Значением по умолчанию является UNCONSTRAINED.
> plot(sin(x),x, labels =["X","Y"], **scaling** = CONSTRAINED);	
style	Задает стиль построения графика. Допускаются значения: LINE (линия), POINT (точка), PATCH (заливка), PATCHNOGRID (заливка без отображения границ). По умолчанию используется значение LINE, что соответствует отображению графика функции посредством линии, соединяющей базовые точки. При значении POINT отображаются только базовые точки. Значения PATCH и PATCHNOGRID имеет смысл использовать, только если график содержит замкнутые многоугольники. В этом случае

Параметр	Описание параметра
	многоугольники заполняются цветом, указанным в качестве значения параметра color. Разница между PATCH и PATCHNOGRID состоит в том, что в последнем случае граница многоугольника не отображается.
> plot(sin(x),x, labels =["X","Y"], **style** = POINT);	
symbol	Тип символов для отображения базовых точек. Возможные значения таковы: BOX (квадрат), CROSS (крест), CIRCLE (круг), POINT (точка) или diamond (ромб). Параметр имеет смысл использовать, если значение параметра style равно POINT.
> plot(sin(x),x, labels =["X","Y"], style = POINT, **symbol=** CROSS);	
symbolsize	Размер символов отображения базовых точек. Этот параметр не влияет на размер точек (если symbol=POINT). По умолчанию размер символов равен 10 пунктам.
> plot(sin(x),x, style = POINT, symbol= CROSS, **symbolsize=20**);	
thickness	Параметр задает толщину линии отображения графика. Возможные значения: 0 – 15). Значением по умолчанию является 0.
> plot(sin(x), x, **thickness =** 10);	
tickmarks	Задает минимальное число отметок по осям координат. Значением параметра является список: первый элемент – для оси абсцисс, второй – для оси ординат. Если нужно задать такой элемент только для оси абсцисе (ординат), используют параметр xtickmarks (ytickmarks).
> plot(sin(x), x, thickness=5, **tickmarks**=[4, 8]);	
title	Заголовок рисунка. Значением параметра является строка символов. По умолчанию заголовок не отображается. Если нужно вывести многострочный заголовок, для перехода на новую строку используют сочетание символов "\n".
> plot(sin(x), x, **title**="График гармонической функции \n sin(x)");	
titlefont	Шрифт заголовка рисунка. Определяется точно так же, как и параметр font.
> plot(sin(x), x, title="График \n sin(x)", **titlefont=**[TIMES, ITALIC,14]);	
xtickmarks	Задает минимальное число отметок по оси абсцисс. Значением может быть число или список. Если значение – список, то элементы этого списка определяют надписи у меток. В качестве элементов списка можно использовать равенства. Тогда левая часть равенства определяет координаты выделенных точек, а правая (в обратных кавычках) – отображаемый при этом текст.
> plot(sin(x), x, **xtickmarks =** [2=`ЖК-фаза`, 4=`плавление`,6=`жидкость`]);	
ytickmarks	То же, что и параметр *xtickmarks*, но только для оси ординат.

В основном задание параметров процедуры *plot()* не вызывает особых трудностей – «к хорошему быстро привыкают». Самое трудное запомнить правильное написание опций, но все они происходят от английских слов с латинскими корнями, их можно сначала угадать, затем постепенно запомнить. И, наконец, незаменимую услугу здесь, как и в любой иной аналогичной ситуации, может оказать справочная система Maple.

Для получения оперативной справки по контексту достаточно установить курсор на соответствующий объект, например имя какой-либо функции, и открыть меню *Help*. Допустим, вы хотите уточнить синтаксис опции *title*. Если установить курсор на это слово и открыть меню *Help*, в нем можно обнаружить строчку " *Help on title"*. Выполните эту команду – тут же появится окно со справкой о процедуре *plot()* и всех ее опциях. Забыли написание опции – установите курсор на слове *plot()* и получите справку об этой процедуре и всех ее опциях. Скопируйте нужный термин и вставьте в рабочий лист.

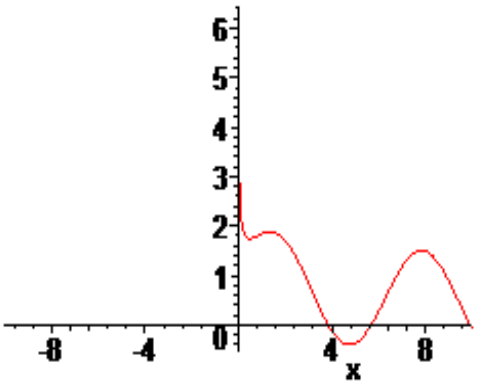
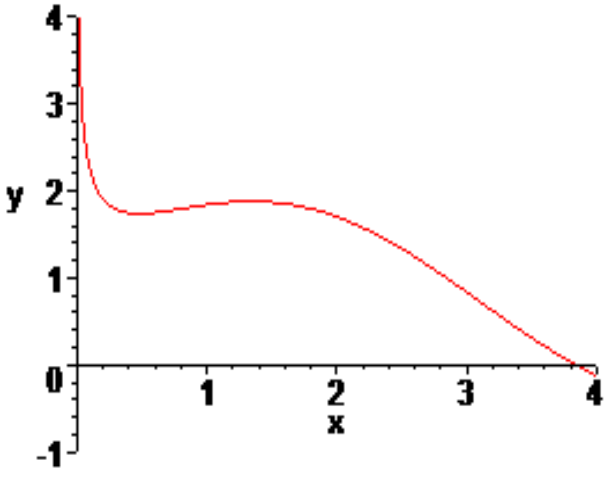
4.3. Основные приемы построения двумерных графиков

4.3.1. Графики функции одной переменной

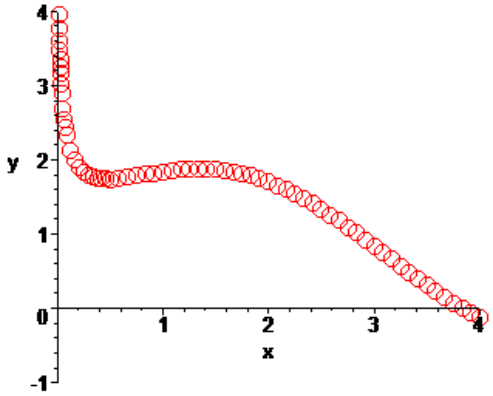
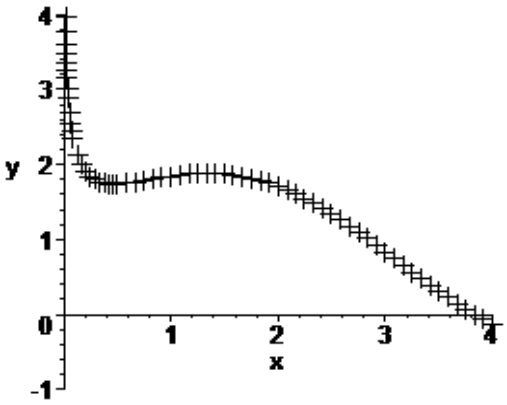
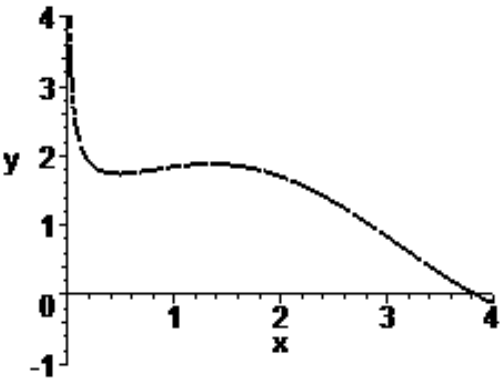
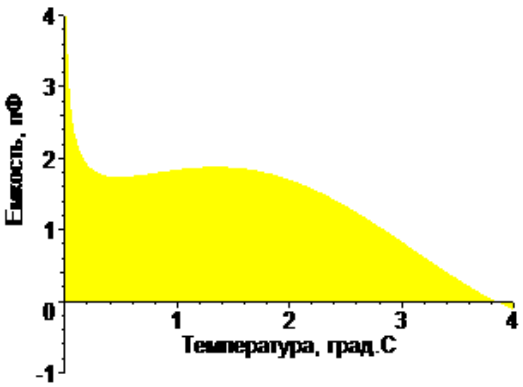
Рассмотрим несколько примеров применения наиболее употребительных опций процедуры *plot()*, которые не только помогут с ними освоиться, но и выработать тактику действий при создании качественной графической иллюстрации.

Все последующие упражнения 4.2 – 4.5 выполнены на примере построения графика функции: $y = \sin(x) + 1/x^{1/3}$.

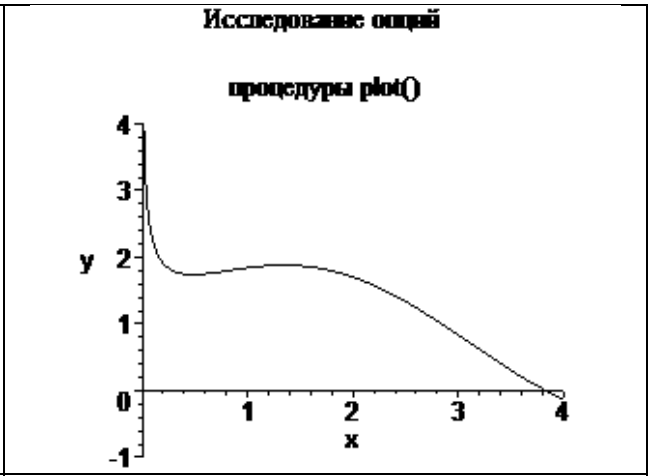
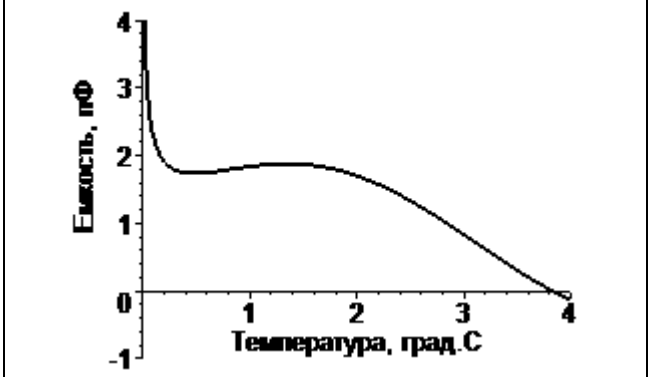
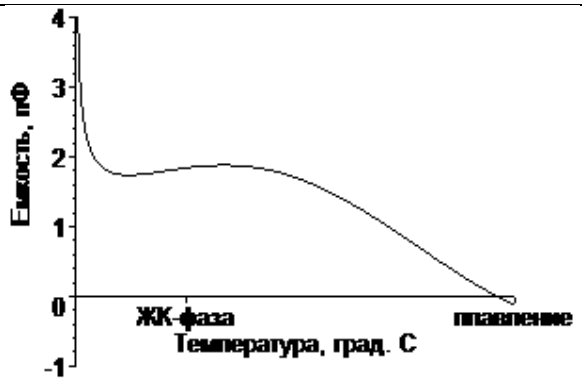
Пример 4.2. Управление диапазоном изменения переменной и значения функции

Наберите в командной строке:	• Результат исполнения команды
> f:=sin(x)+1/(x^(1/3)); > **plot(f,x);** # Здесь параметры графика заданы по умолчанию. Начинать подготовку графической иллюстрации рекомендуется именно с этого режима.	
#Проведем коррекцию области существования переменных, выделив только необходимую для иллюстрации часть: > **plot(f,x=0..4.y=-1..4);** # Правильный выбор диапазонов повышает представительность графиков функций.	

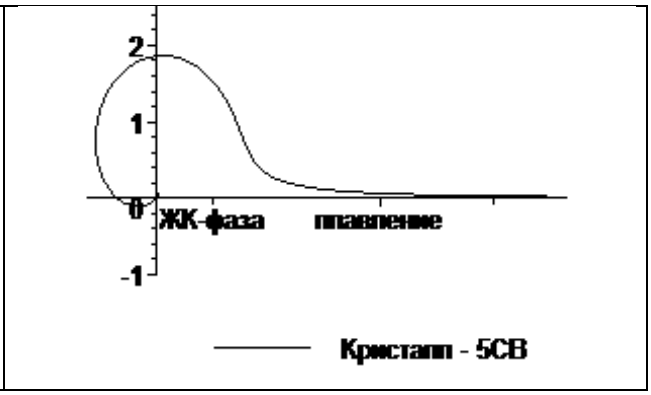
Пример 4.3. Управление стилем и цветом линий графика.

# Изобразим график точками размером 20 пт. Цвет оставим по умолчанию: > **plot(f,x=0..4,y=-1..4,** **style= point, symbol=circle,** **symbolsize=20);** **#** По умолчанию *Maple* использует красный цвет линий или символов.	
# Изменим цвет графического изображения на черный и изобразим этот же график крестиками. > **plot(f,x=0..4,y=-1..4,** **style= point), symbol=cross,** **symbolsize=20, color=black);**	
# Изменим толщину и стиль линии: > **plot(f,x=0..4,y=-1..4,** **thickness = 3, linestyle=4);**	
# Выполним заливку графика, например, желтым цветом: > **plot(f,x=0..4,y=-1..4,** **color=yellow, filled=true);** **#** Опции для подписи координатный осей добавьте самостоятельно.	

Пример 4.4. Создание заголовка и подписей координатных осей.

# Создадим заголовок графика. Толщину и стиль линии оставим по умолчанию: > **plot(f,x=0..4,y=-1..4, titlefont=[COURIER,BOLD,12], title="Исследование опций \n процедуры plot()");** # Для перехода на новую строку используют сочетание символов «\n».	
# Добавим названия координатных осей и расположим их - одно по горизонтали, другое по вертикали: **>plot(f,x=0..4,y=-1..4, titlefont= [COURIER,BOLD,12], title="Исследование опций \n процедуры plot()", labels = ["Температура, град. С"," Емкость, пФ"], label-directions = [HORIZONTAL, VERTICAL]);**	
# Зададим поясняющие отметки на оси абцисс: **>plot(f,x=0..4,y=-1..4, , labels = ["Температура, град. С"," Емкость, пФ"], labeldirections = [HORIZONTAL, VERTICAL], xtickmarks =[1=`ЖК-фаза`, 4=`плавление`);**	

Пример 4.5. Изменение системы координат.

plot(f,x=0..4,y=-1..4, labels = ["Температура, град. С"," Емкость, пФ"], labeldirections = [HORIZONTAL, VERTICAL], xtickmarks = [1=`ЖК-фаза`, 4 =`плавление`, 6 =`жидкость`], color=black, legend="Кристалл – 5CB", coords = polar);	

Выполните эти упражнения «руками» и вы убедитесь, что это совсем несложно.

Большинство команд управления стилем построения графиков доступны из основного или контекстного меню. Такая панель появляется, если двумерный график выделен или на нем находится маркер ввода. Вид контекстной панели зависит от выделенного в рабочем поле объекта. В приложении приведено описание и назначение кнопок контекстной панели для редактирования параметров двумерных графиков.

4.3.2. Графики функции с бесконечными интервалами

Иногда 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*.

Пример 4.6. Построить график функции $y = \ln(1+\cos(x))$ без указания интервала изменения *x* и с явным указанием интервала изменения *x* от 0 до ∞.

Программный код и результаты расчетов показаны на рис. 4.1.

```
plot( ln(1+cos(x)), x, labelfont = [TIMES,ITALIC,12], color=blue);
plot(ln(1+cos(x)), x=0..infinity, labelfont = [TIMES,ITALIC,12]);
```

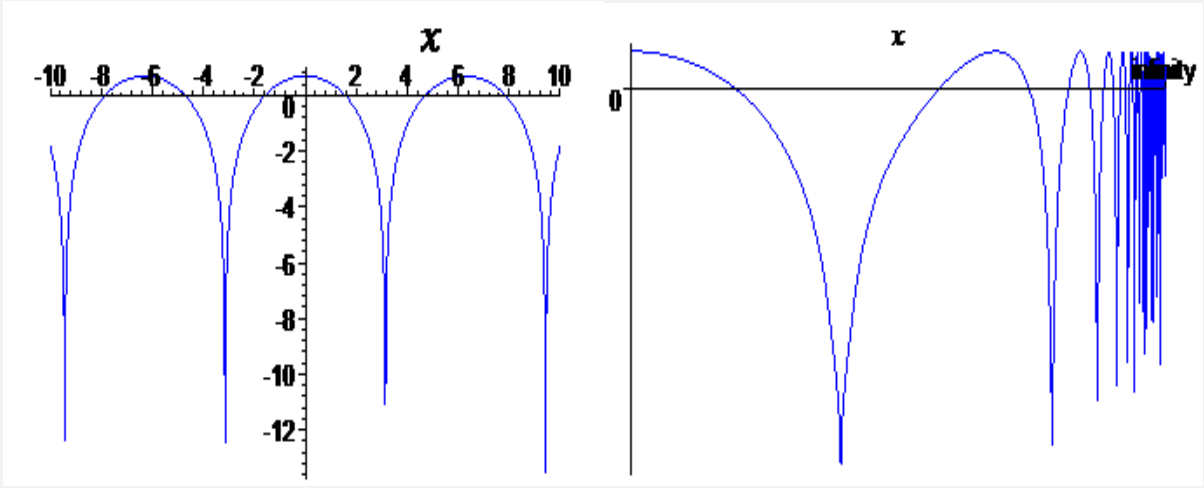


Рис. 4.1. Построение графика функции $y = \ln(1+\cos(x))$

4.3.3. Графики функций с разрывами

Некоторые функции, например tg*(x)*, имеют при определенных значениях *x* разрывы, причем случается, что значения функции в этом месте устремляются в бесконечность. Функция tg*(x)*, к примеру, в точках разрывов устремляется к $+\infty$

и $-\infty$. Построение графиков таких функций по умолчанию нередко дает плохо предсказуемые результаты.

Пример 4.7. Построить график функции $y = \mathrm{tg}(x)$ без указания интервала изменения *x* и на интервале изменения *y* от –7 до 7.

Программный код и результаты расчетов показаны на рис. 4.2.

```
plot(tan(x), x, color=blue);
plot(tan(x), x= -7..7, y = -10..15, color=blue);
```

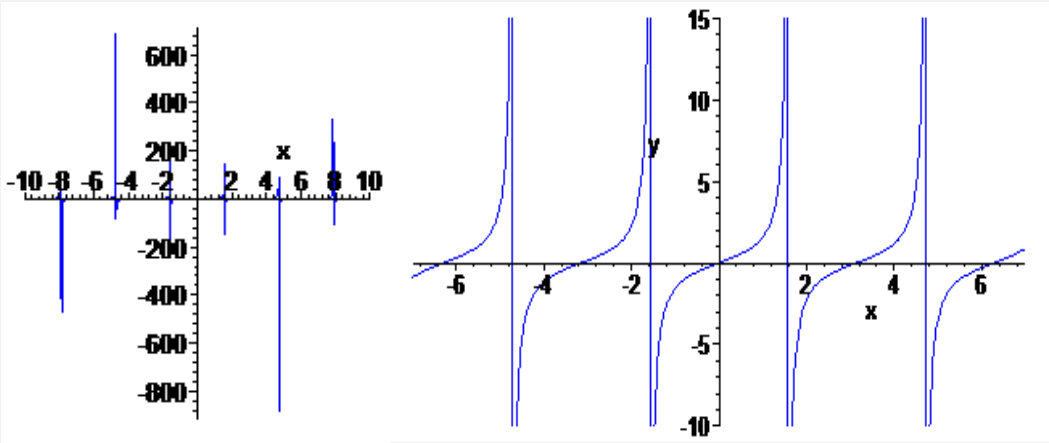


Рис. 4.2. График разрывной функции y = tg(x)

Графический процессор Maple не всегда в состоянии определить оптимальный диапазон по оси ординат, а график функции выглядит весьма непредставительно. Графики функций с разрывами следует всегда строить с обязательным указанием интервала изменения ее значений. При этом в точках разрыва могут появиться вертикальные линии – асимптоты. Иногда это бывает полезно.

Maple умеет работать с разрывными функциями, имеющими разрывы любого рода. Для задания кусочно-непрерывных функций следует использовать команду piecewise.

Пример 4.8. Построить график функции:

$$f(x) = \begin{Bmatrix} -1 & x \le 1 \\ x^2 & -x < -1 and\ x < 2 \\ 3 & x > 2 \end{Bmatrix}.$$

Программный код и результаты расчетов показаны на рис. 4.3.

```
f:=x → piecewise(x ≤ 1,-1, 1< x and x < 2, x^2,  3);
plot( f(x), x=0..3, color=black, thickness = 2);   # a)
plot( f(x), x=0..3, color=black, thickness = 2, discount = true);  # b)
```

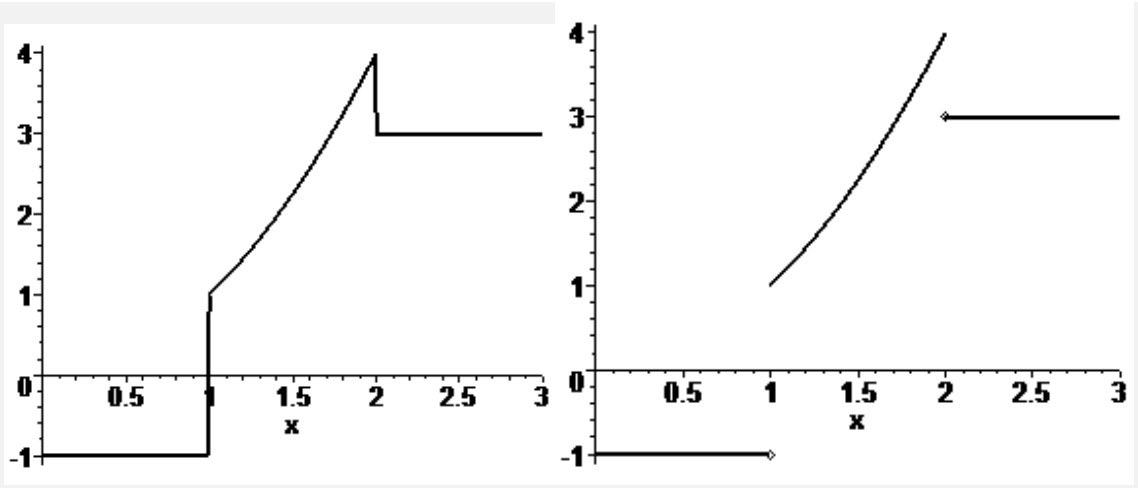


Рис. 4.3. График кусочно-непрерывной функции

Левый график на рис.4.3 построен не очень корректно, так как Maple вычертил вертикальные линии в точках разрыва, соединяя значения функции, справа и слева от точки разрыва. Чтобы избежать такого некорректного отображения функции, следует использовать специальный параметр *discont* функции *plot()*. Если задать его значение равным *true*, то качество графика существенно улучшается.

Улучшение достигается разбиением графика на несколько участков, на которых функция непрерывна, и более тщательным контролем за отображаемым диапазоном. При *discont= false* (значение по умолчанию) данный параметр отключен и строятся обычные непрерывные графики. Отметим, что для удаления из графика функции с разрывами второго рода вертикальных прямых – асимптот в точках разрыва следует также воспользоваться опцией *discont*.

4.3.4. Графики быстро осциллирующих функций

Maple вычерчивает графики функций, вычисляя их значения на некотором множестве равноотстоящих точек независимой переменной. Затем он анализирует изменение функции на полученных интервалах и принимает решение вычислить значения функции в дополнительных точках тех интервалов, где функция является быстро осциллирующей. К сожалению, не во всех случаях такой адаптивный алгоритм срабатывает. В этих случаях необходимо использовать явное задание количества точек на графике.

Пример 4.9. Построить график функции:

$$S = \sum_{k=0}^{40} (-1)^k \left| -x + \frac{1}{10} k \right|$$

с параметрами по умолчанию и с явным заданием количества точек на графике.

Программный код и результаты расчетов показаны на рис. 4.4.

```
s :=Sum((-1)^k* abs(x - k/10), k=0..40):
plot(value(s), x=-1..5, color = black);
plot(value(s), x=-1..5, color = black, numpoints = 600);
```

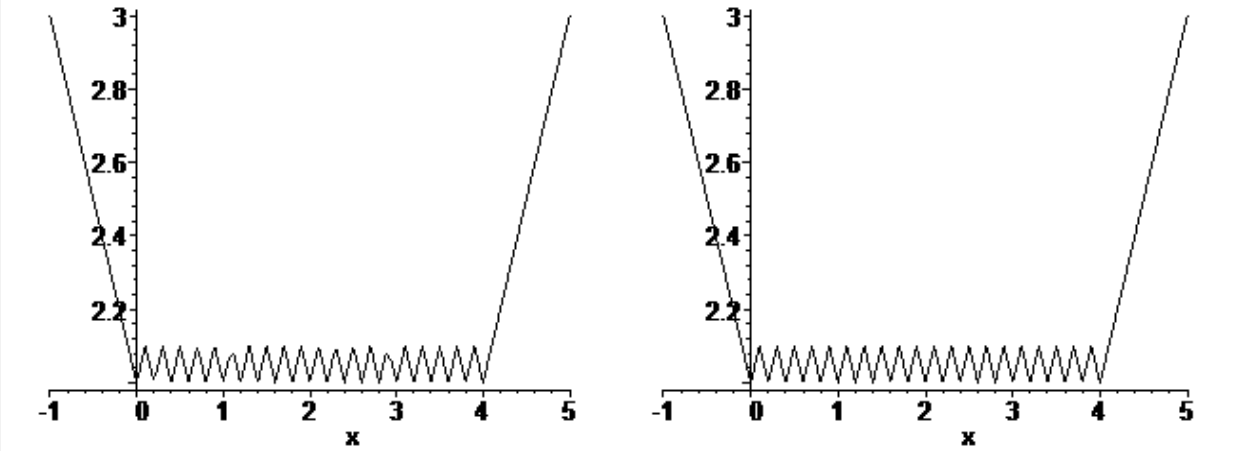


Рис. 4.4. График быстро осциллирующей функции

Как видно, при построении по умолчанию зубчики на графике рис.4.4 не одинаковой величины, что является свидетельством недостаточности вычисленных адаптивным алгоритмом точек на графике. Задание опции *numpoints* позволило получить его приемлемое отображение.

4.3.5. Графики функций, построенных точками

Показанный ранее в примере 4.3 график функции, построенный различными значками, не означает, что этот график представлен отдельными точками. В данном случае был выбран стиль линии в виде тех или иных графических примитивов для непрерывной в целом функциональной зависимости. Однако часто возникает необходимость построения графиков функций, которые представлены просто совокупностями точек, например, при отображении экспериментальных данных. Такая совокупность может быть создана искусственно либо просто задаваться списком значений координат *x* и *y*.

Пример 4.10. Построить график функции y = sin(x) с аргументом, заданным списком из 30 равноотстоящих значений.

Программный код и результаты расчетов показаны на рис. 4.5.

```
P:=[[i,sin(i/3)]$i=1..30]:
plot(P, x= 0..30,color = black, style = point, symbol = circle, symbolsize =1 0);
```

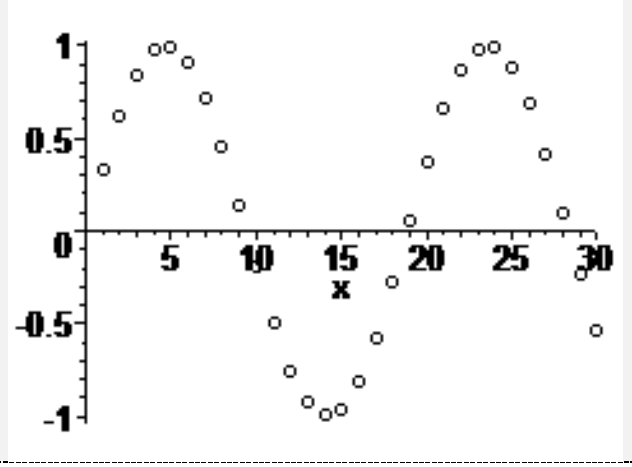


Рис. 4.5. Построение графика функции y = sin(x), заданного списком

В данном случае переменная *P* имеет вид искусственного списка, в котором попарно перечислены координаты точек функции sin*(x)*. Далее по списку *P* построен график точек в виде окружностей, которые отображают отдельные значения функции непрерывной функции sin*(x)*.

Пример 4.11. Построить график функции при явном задании координат точек: [0, 0], [1, 0.5],[2, 0.7],.[3, 0.4],[5, 0.2].

Программный код и результаты расчетов показаны на рис. 4.6.

```
plot([[0,0], [1,0.5],[2,0.7],[3,0.4],[5,0.2]],x = 0..5, style=point, symbol = circle,
symbolsize = 20);
plot([[0,0], [1,0.5],[2,0.7],[3,0.4],[5,0.2]],x = 0..5, style=line, symbol = circle, sym-
bolize = 20);
```

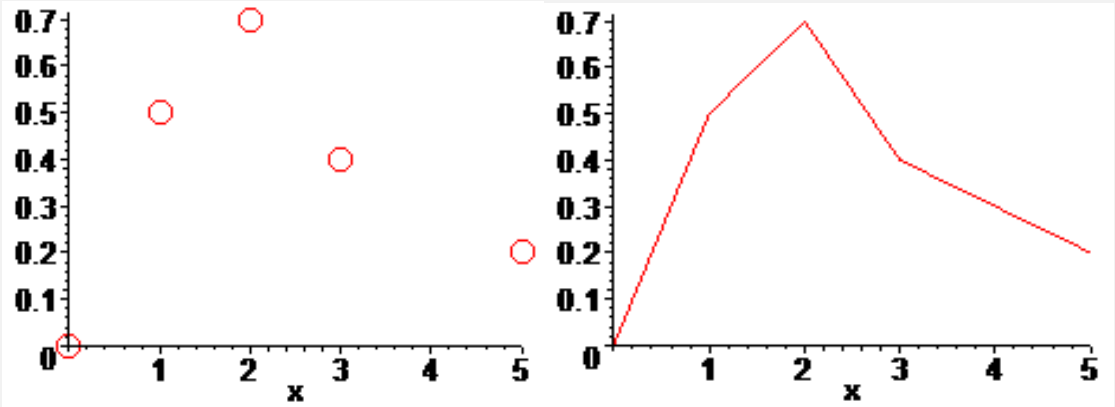


Рис. 4.6. Построение графика функции заданной списком координат

На левом графике (рис. 4.6) показано построение только точек заданной зависимости. Они представлены маленькими кружками. На правом – точки соединяются отрезками прямых, так что получается кусочно-линейный график.

Видно, 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).

4.3.6. 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 4.12. Построить график функции с ординатами, заданными вектором: [1, 3.5, 4.5, 4, 3, 2, 1, 0.5, 0.2, 0.1].

Пример построения такого графика дан на рис. 4.7.

```
Y:=[1,3.5,4.5,4,3,2,1,0.5,0.2,0.1]:
L:=[[n,Y[n]] $n=1..10]:
plot(L, n=0..10, style = point, symbol = diamond, symbolsize = 20);
```

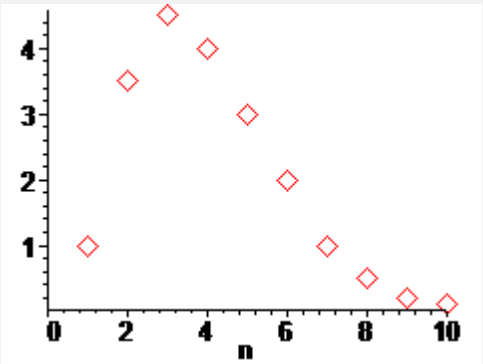


Рис. 4.7. Построение графика с ординатами, заданными вектором

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*.

4.3.7. Графики функций, заданные процедурами

Некоторые виды функций, например, кусочные, удобно задавать процедурами. Построение графиков функций, заданных процедурами, рассмотрено в следующем примере.

Пример 4.13. Построить график функции y = |sin(x)|.

Пример построения такого графика показан на рис. 4.8.

```
Y:=proc(x):
if sin(x)>0 then sin(x) else - sin(x) fi end;
plot(Y, -6..6, color=blue);
```

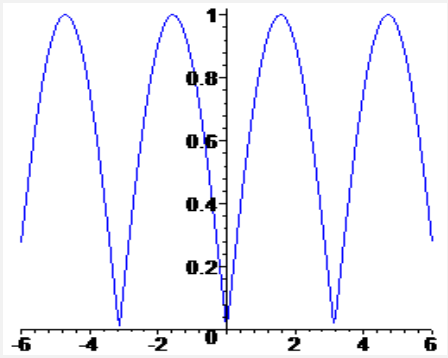


Рис. 4.8. Построение графика функции, заданной процедурой

Здесь полезно обратить внимание на то, что в функции *plot()* указывается только имя процедуры без списка ее параметров. Разумеется, все опции процедуры *plot()* здесь уместны.

4.3.8. Графики функций, заданных параметрически

В ряде случаев для задания функциональных зависимостей используются параметрические уравнения, например $x = f(t)$ и $y = f(t)$ при изменении переменной t в некоторых пределах. Точки (x, y) наносятся на график в декартовой системе координат и соединяются отрезками прямых. Для задания графического объекта здесь следует использовать список.

Пример 4.14. Построить график параметрической кривой y = sin(2t), x = cos(3t) в диапазоне $0 \le t \le 2\pi$ в рамке.

Программный код и пример построения такого графика показан на рис. 4.9.

```
plot([sin(2*t),cos(3*t),t= 0..2*Pi], axes = BOXED, thickness=2);
```

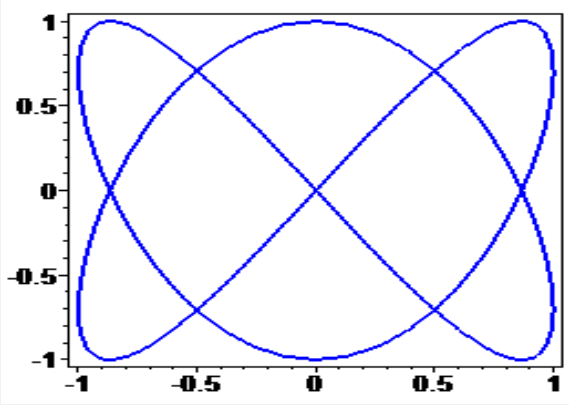


Рис. 4.9. Построение графика функции заданной параметрически

Задание диапазонов изменения аргумента, а также параметров процедуры *plot()* не обязательно, но, как и ранее, они позволяют получить вид графика, удовлетворяющий всем требованиям пользователя.

4.3.9. Графики нескольких функций на одном рисунке

Часто возникает необходимость построения на одном рисунке графиков нескольких функций. В простейшем случае для построения таких графиков достаточно создать список или перечислить в процедуре *plot()* нужные функции и установить для них общие интервалы изменения аргумента.

Пример 4.15. Построить на одном рисунке график функции $y = \ln(3x-1)$ и касательную к нему $y = 1.5x - \ln 2$.

Программный код и пример построения такого графика показан на рис. 4.10.

```
plot( [ ln (3*x-1), 3*x/2- ln (2) ], x = 0..6);
```

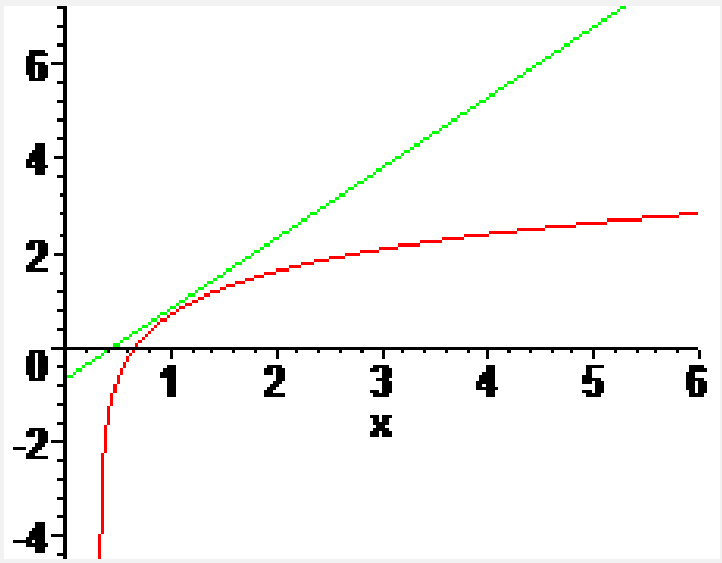


Рис. 4.10. Графики двух функций на одном рисунке

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. Используя списки параметров *color* и *style* , можно добиться контрастного выделения кривых.

Задача о совмещении кусочно-линейной линии и символьного представления графика, заданного в виде пар точек решается аналогичным образом.

Пример 4.16. Построить на одном рисунке график функции при явном задании координат точек: [0, 0], [1, 0.5], [2, 0.7], [3, 0.4], [5, 0.2] и ее кусочно-линейную аппроксимацию (рис. 4.11).

```
Y:=[[0,0], [1,0.5],[2,0.7],[3,0.4],[5,0.2]];
plot([Y,Y],x=0..5,style=[line,point],color=[black,blue], symbol = circle,  symbolsize = 20);
```

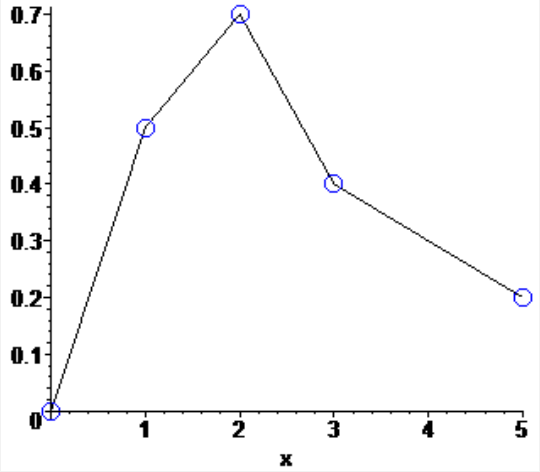


Рис. 4.11. График функции, заданной числовой последовательностью и ее линейно-кусочная аппроксимация

Здесь для наглядности список координат точек задан с помощью дополнительной переменной Y.

Как правило, если в строке ввода задается построение нескольких графиков, то в строке вывода все они располагаются по вертикали. В Maple можно разместить ряд двумерных графиков в строке вывода по горизонтали. Для этого, правда, необходимо подключить дополнительный графический пакет *plots*().

Пример 4.17. Построить два графика: $y = \ln(3x-1)$ и $z = \sin(x)$ в одной строке вывода (рис.4.12).

```
with (plots, display):
a:= plot(sin(x), x= - 6..10, color=blue ):  b := plot (ln(3*x-1), x = 0..6,):
plots[display](array(1..2, [a,b]));
```

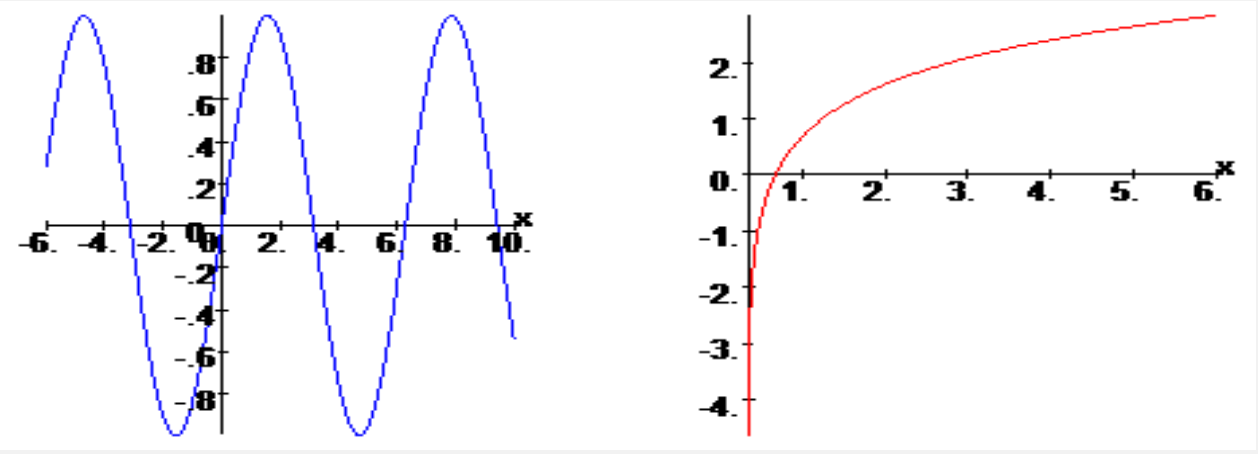


Рис. 4.12. Пример расположения графиков в строке вывода по горизонтали

Приложение 4.1

Контекстная панель инструментов для двумерных графиков

Двумерные графики строятся с заданием ряда параметров, определяющих стиль графика. Все параметры имеют значения по умолчанию – они определяют вид графика, при формировании которого параметры не указаны явным образом.

Однако ряд параметров можно изменять уже после построения графика с помощью контекстной панели. Такая панель появляется, если двумерный график выделен или на нем находится маркер ввода.

На рис. 4.13 показано назначение кнопок контекстной панели инструментов для редактирования двумерных графиков. Отметим, что в разных версиях Maple вид контекстной панели может отличатся, но функции кнопок остаются неизменными.

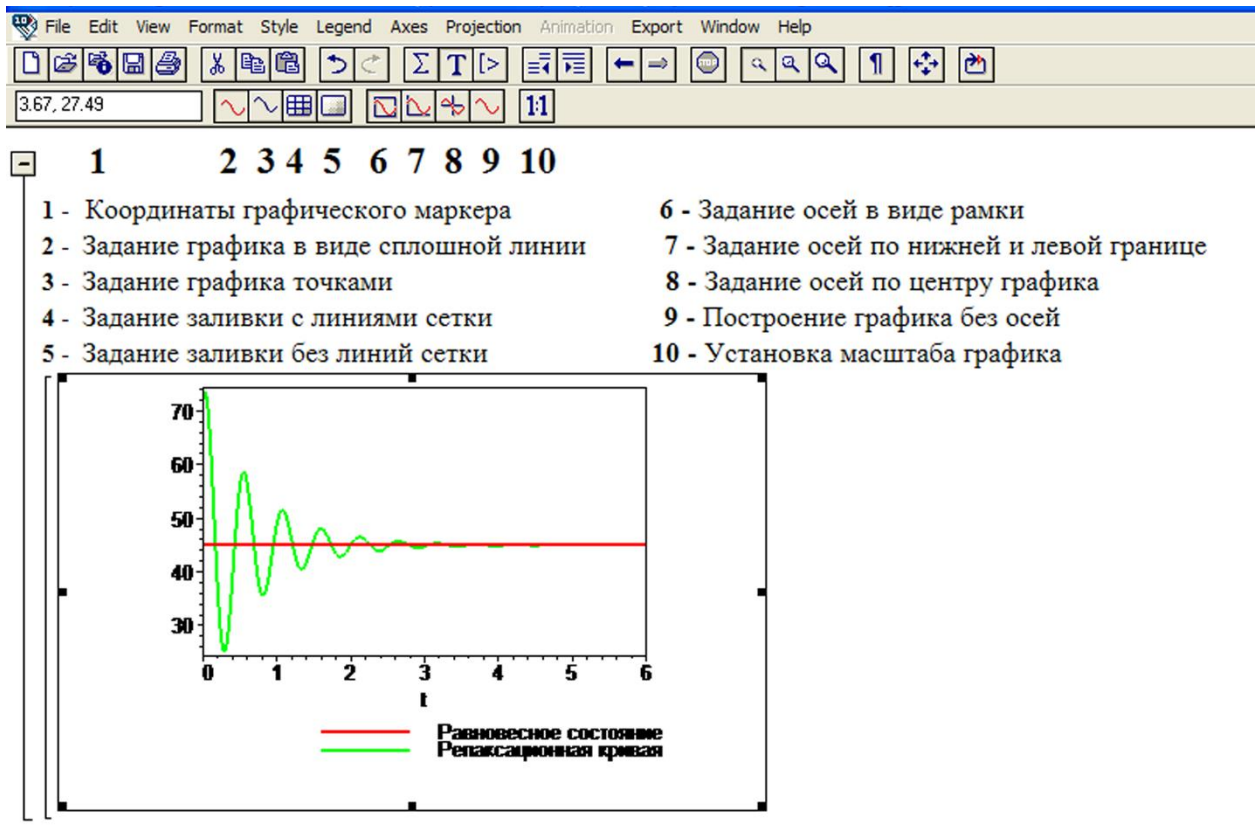


Рис. 4.13. Назначение кнопок панели форматирования двумерных графиков

Задания для самостоятельной работы

1. Построить графики функций:

$y = x \cdot \sin(1\text{-}x) + \dfrac{\cos x}{\sqrt{1+x^2}}, -3 \le x \le 3.$

$y = \dfrac{e^x}{(x-1)^2+1} + \sin(\sqrt{x^2+1}).\ -3 \le x \le 4.$

$y = \sin(\sqrt{|x|} + \dfrac{\ln(x^2+1)}{x^4+1};\ -4 \le x \le 4.$

$y = 2\sin(2x^2+1) - \dfrac{x}{x^2+x+1};\ -4 \le x \le 4.$

2. Построить графики функций, заданных параметрически:

$x = 2(t-\sin t),\ y = 2(1-\cos t);\ \ -8 \le t \le 8.$

$x = \dfrac{1+\sin t}{1+t^2},\ y = \dfrac{1-\cos t}{1+t^2};\ \ -10 \le t \le 10.$

$x = \dfrac{t^2-1}{t^2+1},\ y = \dfrac{t(t^2-1)}{t^2+1};\ \ -10 \le t \le 10.$

$x = \dfrac{2 \cdot t^2}{1+t^2},\ y = \dfrac{2 \cdot t^3}{1+t^3};\ \ -10 \le t \le 10.$

3. Построить два графика вместе:

$y = e^x \cos x$ и $y = \dfrac{x}{(1+x^2)(1+x+x^2)};\ \ -4 \le x \le 4.$

$y = e^{-\sin x} + e^{-\cos x}$ и $y = \dfrac{1+x^3}{1+x^2};\ \ -2 \le x \le 2.$

$y = \dfrac{x}{\sqrt{x^2+x+1}}$ и $y = \dfrac{\sin x}{1+x^2};\ \ -4 \le x \le 4.$

$y = \dfrac{\cos(x^2)}{x^4+x^2+1}$ и $y = \dfrac{\cos x}{x^2+1};\ \ -4 \le x \le 4.$

5. ТРЕХМЕРНАЯ ГРАФИКА В MAPLE

Трехмерными называют графики, отображающие функции двух переменных *z(x, y)*. Каждая точка 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.

5.1. Процедура трехмерной графики plot3d()

Для построения графиков трехмерных поверхностей Maple имеет встроенную в ядро функцию *plot3d*. Она может использоваться в следующих форматах.

Синтаксис процедуры	Аргументы процедуры	
• ***plot3d(exprl. 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)***	***f, g, h***	– визуализируемые функции;
	x, y, s, t	– имена независимых переменных;
	exprl	– выражение, отражающее зависимость от *x* и *y*;
	exprf, exprg* и *exprh	– выражения, задающие поверхность в параметрическом виде;
	s, t, a* и *b	– числовые константы действительного типа;
	c* и *d	– числовые константы или выражения действительного типа;
	p	– управляющие параметры.

В двух первых формах процедура *plot3d()* применяется для построения графика одной поверхности, в других формах – для построения графика с параметрической формой задания поверхности.

5.2. Параметры процедуры plot3d()

С помощью параметров процедуры *plot3d*() можно в широких пределах управлять видом трехмерных графиков. Многие параметры процедуры пространственной графики полностью соответствуют своим двумерным аналогам, правда, в некоторых добавлена дополнительная функциональность, но есть и специальные опции, отражающие специфику пространственной графики. Ниже перечислены такие опции с их кратким описанием и возможными значениями.

Параметр	Описание параметра
ambienlight	Задает цвет внешнего источника интенсивностями его интенсивность красной (r), зеленой (g) и синей (b) составляющих. Значением параметра является список [r,g,b]. Величины r, g и b должны быть числами в диапазоне от 0 до 1.
axes	Определяет тип координатных осей. Возможные значения – BOXED, NORMAL, FRAME и NONE. Значения параметра имеют тот же смысл, что и соответствующие значения для двухмерных графиков, но только теперь речь идет не о двух, а о трех координатных осях. Значение по умолчанию NONE.
color	Задает цвет отображаемой поверхности в случае ее закрашивания или цвет линий сетки в случае отображения поверхности в виде каркаса или линий уровня. В качестве значений этой опции может выступать одно из зарезервированных значений цвета в Maple. Можно также определить собственный цвет, соответствующий смешению в заданных частях красного, зеленого и синего цветов.
contours	Определяет количество линий уровня (контуров) при отображении их на поверхности. Значение по умолчанию равно 10.
coords	Параметр задает тип координатной системы. По умолчанию используется декартова система координат.
filled	Параметр может принимать значения true и false. Если значение параметра равно true, область, ограниченная поверхностью и плоскостью xy отображается как непрозрачное тело, закрашенное в соответствии с используемой цветовой схемой. Значением по умолчанию является false.
grid	Эта опция определяет прямоугольную равномерную сетку значений независимых переменных отображаемой функции, на которой вычисляются ее значения. Задается в виде списка [m,n], в котором каждый элемент является целым числом, определяющим количество точек по соответствующей координате. По умолчанию используется сетка [25,25].
gridstyle	Задает тип отображаемой сетки. Параметр может принимать значение *rectangular* (сетка поверхности состоит из прямоугольных ячеек) или *triangular* (сетка поверхности состоит из треугольных ячеек).
labels	Задание названий координатных осей в виде списка [x,y,z]. Параметры x, y и z задаются в виде строк. По умолчанию оси не подписываются.

Параметр	Описание параметра
lebeldirections	Параметр определяет ориентацию надписей у координатных осей. Значением параметра является список из трех элементов [x,y,z]. Каждый элемент, соответствующий координатной оси, может принимать значение HORIZONTAL (по горизонтали, используется по умолчанию) или VERTICAL (по вертикали).
light	Эта опция определяет расположение и цвет напарвленного источника света при использовании пользовательской схемы подсветки. Ее значения задаются в виде списка [*phi, theta, r, g, b*]. *Phi* и *theta* - определяют углы направления, из которого исходит направленный свет (задаются в сферической системе координат), *a r, g, b* – числовые интенсивности от 0 до 1 составляющих цвета источника.
lightmodel	Параметр задает одну из стандартных моделей подсветки поверхности. Допускаются такие значения: *none* (не используется ни одна из моделей), *lightl, light2, light3 и light4.*
orientation	Параметр задает углы в сферической системе координат направления, из которого наблюдатель смотрит на отображаемую поверхность. Значением является список из двух элементов [*phi, theta*] – азимутального и полярного углов, определяющих такое направление. Углы задаются в градусах и по умолчанию равны [45,45]
projection	Значением параметра является число в диапазоне от 0 до 1, определяющее ракурс, в котором просматривается поверхность. Значением могут быть также FISHEYE (выгнутый), NORMAL (нормальный) и ORTHOGONAL (ортогональный), что соответствует значениям 0; 0.5; и 1 .
scalling	Задает масштаб, в котором отображается поверхность. Если значение данной опции равно CONSTRAINED, то это соответствует заданию абсолютных значений по осям координат, т.е. одна единица измерения по оси независимой переменной равна одной единице измерения по оси значений функции. Значение по умолчанию равно UN CONSTRAINED. Это соответствует тому, что оси растягиваются таким образом, чтобы их размеры соответствовали размерам графического окна вывода.
shading	Параметр определяет, какая схема закрашивания применяется при отображении поверхности. Возможные значения: • XYZ – цвет точки поверхности зависит от значений трех ее координат; • XY – цвет точки поверхности зависит от значений двух ее независимых координат; • Z – цвет точки поверхности зависит от значения функции: минимальное представляется синим цветом, максимальное – красным, остальные оттенками при переходе от синего к красному; • ZGRAYSCALE – цвет точки поверхности зависит от значения функции: минимальное представляется черным цветом, максимальное – бледно-серым, остальные оттенками при переходе от черного к бледно-серому;

Параметр	Описание параметра
	• ZHUE – цвет точки поверхности зависит от значения функции от минимального сиреневого, через синий, зеленый и желтый к максимальному красному; • NONE – поверхность не закрашена.
style	Определяет, как будет отображаться поверхность. Допустимые значения: POINT – точками; HIDDEN – каркасная модель с удалением невидимых линий; PATCH – закрашенная поверхность с линиями сетки; LINE - каркасная модель без удаления невидимых линий; CONTOUR - линиями уровня; PATCHNOGRID – закрашенная поверхность без линий сетки; PATCHCONTOUR – закрашенная поверхность с линиями уровня.
symbol	Определяет тип символа, которым помечаются точки поверхности функции при опции *style = point.*
view	Параметр определяет область отображения графика. Значением может быть либо диапазон, либо список из трех диапазонов: [x_{min} .. x_{max}, y_{min} .. y_{max}, z_{min} .. z_{max}]. В первом случае определяются минимальное и максимальное значения вдоль вертикальной оси, во втором – аналогичные значения для каждой из трех осей. По умолчанию отображается вся поверхность без обрезания.

5.3. Основные приемы построения трехмерных графиков

5.3.1. График поверхности, заданной явной функцией

Как видно из описания процедуры *plot3d()* и ее параметров, работа с ней аналогична двумерному аналогу *plot()*. Рассмотрим следующий пример (рис.5.1).

Пример 5.1. Построить график функции $z = y\,tg\,x$ с заголовком и значениями опций по умолчанию.

```
plot3d(tan(x)*y, x=-3..3, y=-3..3, title="График функции z=y tg(x)");
```

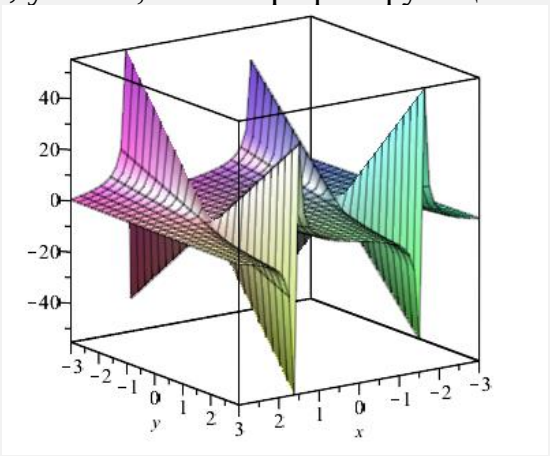


Рис. 5.1. Пример построения трехмерной поверхности

По умолчанию в Maple строится поверхность стилем *sty1e = patch* и закрашивается в соответствии с цветовой схемой XYZ, которая выбирает цвет точки поверхности в зависимости от значения трех ее координат.

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. Такая функциональная окраска делает рисунки более информативными.

Пример 5.2. Построить график функции $z = y \cdot \mathrm{tg}\, x$ стилем *hidden* и ориентацией направления наблюдения (рис.5.2).

```
plot3d(tan(x)*y, x=-3..3, y=-3..3, style=hidden, orientation=[60,65], color=black,
grid=[10,10]);
```

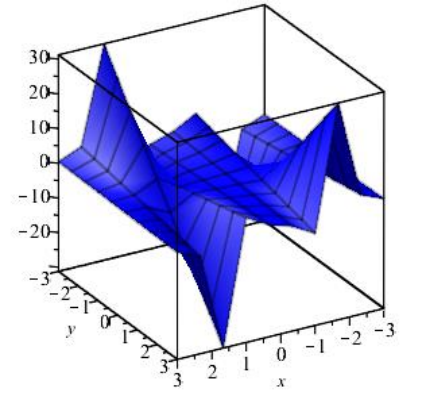


Рис. 5.2. График поверхности, выполненный стилем hidden

Пример 5.3. Построить поверхность

$$z = \frac{1}{x^2 + y^2} + \frac{0{,}2}{(x+1{,}2)^2 + (y-1{,}5)^2} + \frac{0{,}3}{(x-0{,}9)^2 + (y+1{,}1)^2}$$

в пределах от x = –2 -- 2 и y = -2 – 2.5 вместе с линиями уровня, без осей.

plot3d(1/(x²+y²) + 0.2/((x+1.2)² + (y-1.5)²)+0.3/((x-0.9)² + (y+1.1)²), x=-2..2, y=-2..2.5, axes=NONE, light=[100, 30, 1, 1, 1], style = PATCHCONTOUR, grid =[60, 60], view = [-2..2, -2..2.5, 0..6]);

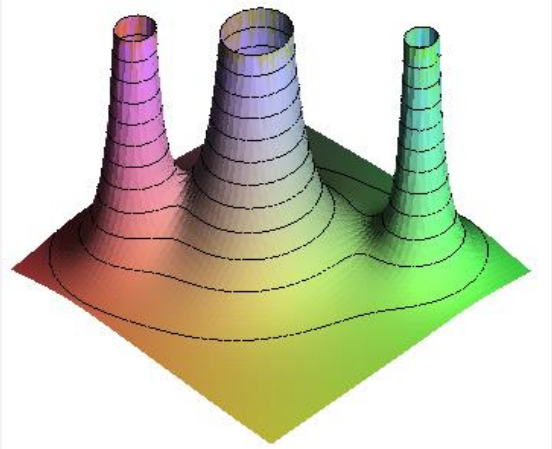


Рис. 5.3. Пример построения трехмерной поверхности стилем patch contour

Как и в случае с двумерной графикой, после построения командой *plot3d()* пространственного образа функции двух переменных можно изменить его внешний вид, переустановив значения некоторых опций с помощью команд основного или контекстного меню. Для этого надо выделить график на рабочем листе. Напомним, что при этом меняется основное меню интерфейса пользователя, а также заменяется контекстная панель инструментов. В приложении приведено описание и назначение кнопок контекстной панели для редактирования параметров трехмерных графиков. Заметим, что вид этой панели зависит от версии Maple, но основные функции интуитивно понятны (рис.5.3).

5.3.2. График поверхности, заданной параметрически

Командой *plot3d* можно отображать параметрически заданные поверхности. В этом случае поверхность задается тремя формулами в виде функций или выражений двух переменных, оформленными как список (рис.5.4).

Пример 5.4. Построить поверхность прямого геликоида, заданного параметрическими уравнениями: x = u cos (v), y = u sin(v), z = 2 v.

```
x:= u*cos(v); y:= u*sin(v); z:=1*v;
plot3d([x, y, z],u=0..3,v=-2*Pi..2*Pi, style=hidden, color=black, grid=[20,20]);
```

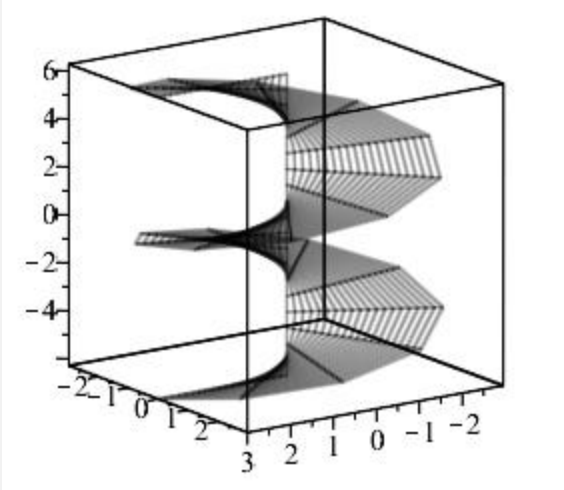


Рис. 5.4. Построение поверхности геликоида, заданного параметрически

Вид графика трехмерной поверхности существенно зависит от выбора координатной системы. Следующее упражнение демонстрирует пример построения геликоида в цилиндрической системе координат.

Пример 5.5. Построить поверхность прямого геликоида, заданного параметрическими уравнениями: x = u cos (v), y = u sin(v), z = 2 v в цилиндрической системе координат (рис.5.5).

```
x:= u*cos(v); y:= u*sin(v); z:=1*v;
plot3d([x, y, z],u=0..3,v=0..2*Pi, style=hidden, color=black, grid=[20,20], coords =
cylindrical);
```

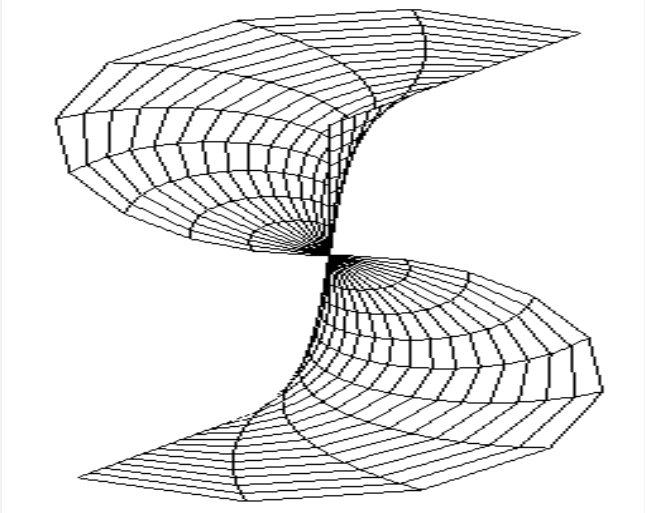


Рис. 5.5. Поверхность геликоида в цилиндрической системе координат

5.3.3. График ряда поверхностей на одном рисунке

Функция *plot3d()* позволяет строить одновременно несколько фигур, пересекающихся в пространстве. Для этого необходимо 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. Пример такого построения представлен ниже (рис.5.6).

Пример 5.6. Построить на одном рисунке графики двух поверхностей: плоскости z = x+y и псевдосферы, заданной параметрически – x = sin (u) cos(v), y = sin (u) sin(v), z = ln(tg(u/2) + cos(u)) с параметрами по умолчанию.

```
plot3d([[sin(u)*cos(v), sin(u)*sin(v), ln(tan(u/2)+cos(u))], u + v],
u = -1*Pi..1*Pi, v = -2*Pi..1*Pi);
```

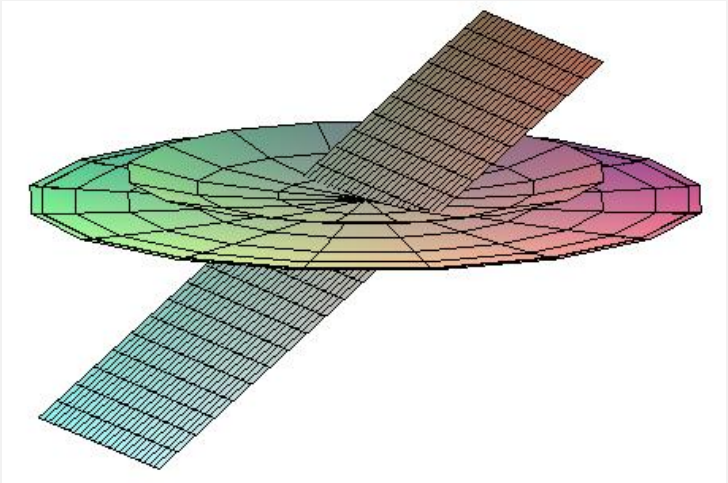


Рис. 5.6. Графики двух поверхностей построенных на одном рисунке

5.3.4. Масштабирование трехмерных фигур

Полезно обратить внимание на параметр масштаба *scaling*. Значение по умолчанию этого параметра равно *constrained*. Он выравнивает масштабы представления фигуры по осям координат и позволяет снизить до минимума геометрические искажения фигур.

Пример 5.7. Построить график сферы, заданной параметрически: x = 2 sin (t) cos(s), y = 2 cos (t) cos(s), z = 2 sin(s) с параметрами по умолчанию *scaling = constrained* и заданием параметра *scaling = unconstrained*.

```
plot3d([2*sin(t)*cos(s), 2*cos(t)*cos(s), 2*sin(s)], s=-Pi..Pi, t=-Pi..Pi, scaling = constrained);
plot3d([2*sin(t)*cos(s), 2*cos(t)*cos(s), 2*sin(s)], s=-Pi..Pi, t=-Pi..Pi, scaling = unconstrained);
```

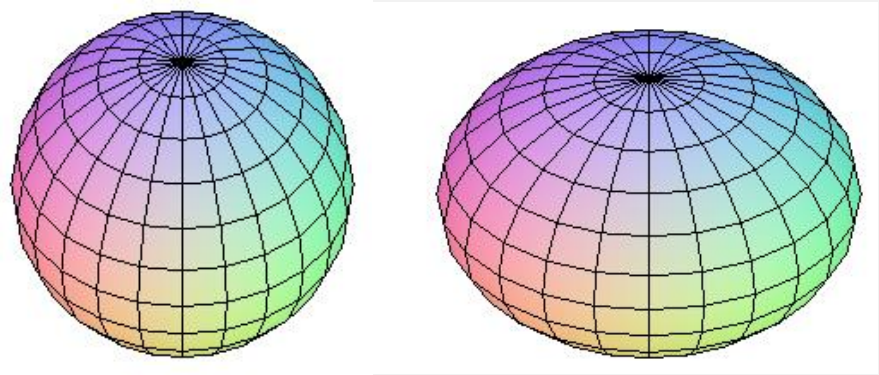


Рис. 5.8. Графики поверхности сферы, построенные с параметрами scaling = constrained и scaling = unconstrained

Разница, как говорится, «на лицо». Задание параметра *scaling = unconstrained* означает отказ от равного масштаба по осям. График при этом увеличивается в размерах занимает большую часть окна вывода, но становятся заметными его искажения по осям координат. В итоге шар превращается в эллипс (рис.5.8).

В математике часто встречается особый тип задания геометрических фигур, при котором переменные связаны неявной зависимостью. В качестве примера рассмотрим эллипсоид вращения:

$$\frac{x^2 + y^2}{a^2} + \frac{z^2}{c^2} = 1.$$

Для того чтобы построить такую поверхность процедурой *plot3d()*, можно преобразовать неявное уравнение к явному, а лучше к параметрическому виду. Это нетрудно сделать и в рамках оболочки Maple и просто "руками". Правда, здесь есть подводные камни.

Пример 5.9. Построить поверхность эллипсоида вращения, заданного неявным уравнением: $\frac{x^2 + y^2}{1^2} + \frac{z^2}{4^2} = 1.$

Преобразуем уравнение к явному виду: $z = 4\sqrt{1 - x^2 - y^2}$ и наберем в командной строке

plot3d(4*sqrt(1-x^2-y^2), x=- 2.. 2, y=-2..2);

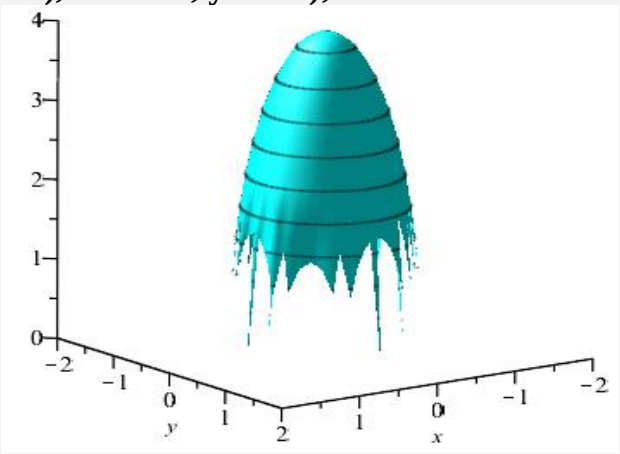


Рис. 5.10. График поверхности эллипсоида вращения

Однако, это не совсем то, что мы хотели увидеть. Причина, впрочем, понятна: квадратное уравнение имеет два корня. Подставив второе решение, возможно подбором пределов изменения независимых переменных, можно будет добиться нужного результата, но придется потрудиться (рис.5.10).

Конечно, разработчики *Maple* не могли заставить пользователя выполнять такие монотонные рутинные операции и предусмотрели расширенные средства графики, которые мы рассмотрим далее.

Задания для самостоятельного решения

1. Построить поверхность, заданную параметрически:

$x = \sin t,\ \ y = \cos t,\ \ z = u;\ \ 0 \le t \le 2\pi, 0 \le u \le 4.$

$x = \sin t,\ y = \sin(2\,t)\sin u,\ z = \sin(2t)\cos u;\ \ -\pi/2 \le t \le \pi, 0 \le u \le 2\pi.$

$x = \cos t + \cos u,\ \ y = \sin t - \sin u,\ \ z = \sin(t)\cos u;\ -\pi \le t \le \pi, 0 \le u \le 2\pi.$

$x = \sin(2\,t\,u),\ \ y = t + u,\ \ z = \cos\frac{t+u}{2};\ \ -2 \le t, u \le 2\pi.$

2. Построить поверхность, заданную уравнением в неявном виде:

$z = \sin x + \cos(x \cdot y);\ \ -\pi \le x, y \le \pi.$

$z = \sin(x + \cos y) + \cos(y + \sin x);\ \ -2\pi \le x, y \le 2\pi\,.$

$z = x\cos y + y\sin x;\ -2\pi \le x, y \le 2\pi.$

6. РАСШИРЕННЫЕ СРЕДСТВА ГРАФИКИ В MAPLE

Универсальные графические команды собраны в пакетах *plots() и plottools().* Чтобы воспользоваться их графическими средствами, необходимо обязательное подключение соответствующих пакетов командой *with (plots)* или *with (plottools).*

6.1. Основные функции пакета plots

Пакет *plots* содержит более шестидесяти графических функций, существенно расширяющих возможности построения двумерных и трехмерных графиков. Ниже приведены наиболее употребительные стилевые функции процедуры *plots.* Дано их краткое описание и примеры использования.

Параметр	Описание параметра
changecoords	Изменение системы координат.
contourplot	Построение контурного графика.
contourplot3d	Построение контурного трехмерного графика.
coordplot	Построение координатной системы двумерных графиков.
coordplot3d	Построение координатной системы трехмерных графиков.
densityplot	Построение двумерного графика плотности.
display	Построение графика для списка графических объектов.
display3d	Построение графика для списка трехмерных 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.
listcontplot	Построение двумерного контурного графика.
listcontplot3d	Построение трехмерного контурного графика.
listdensityplot	Построение двумерного графика плотности.
listplot	Построение двумерного графика для списка значений.
listplot3d	Построение трехмерного графика для списка значений.
loglogplot	Построение логарифмического двумерного графика.
logplot	Построение полулогарифмического двумерного графика.
odeplot	Построение двумерного или трехмерного графика решения дифференциальных уравнений.

Параметр	Описание параметра
pointplot	Построение точками двумерного графика.
pointplot3d	Построение точками трехмерного графика.
polarplot	Построение графика в полярной системе координат.
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.
sphereplot	Построение трехмерной поверхности в сферических координатах.
surfdata	Построение трехмерного графика по численным данным.
textplot	Вывод текста на заданное место двумерного графика
textplot3d	Вывод текста на заданное место трехмерного графика

Среди этих функций необходимо особо отметить средства: для построения графиков, заданных в неявном виде; объединения различных графиков в один; функции для построения графиков новых типов, таких как графиков в виде линий равного уровня, векторных полей. Особый интерес представляют функции, обеспечивающие вывод текста на заданное место пространства, как для двумерных, так и для трехмерных графиков.

6.2. Основные приемы построения графиков в пакете plots

6.2.1. Графики функций, заданные уравнениями неявного типа

Для задания двумерной графики такого вида служит импликативная функция, в которой задается уравнение графика и диапазоны изменения всех ее переменных:

- ***implicitplot(expr1, x = a..b, y = c..d, [options]),***
- ***implicitplot(f, a..b, c..d, [options]).***

Здесь

- *expr1* – выражение, описывающее исследуемую зависимость от координат *x, y;*
- *a, b, c, d* – константы вещественного типа, определяющие область изменения переменных;
- *f* – функция или процедура;
- *[options]* – параметры графика.

Пример 6.1. Построить график окружности, заданной неявным уравнением: $x^2 + y^2 = 16$.

```
with(plots):
implicitplot(x^2+y^2=16, x=-2*Pi..2*Pi, y=-2*Pi..2*Pi, color=blue);
```

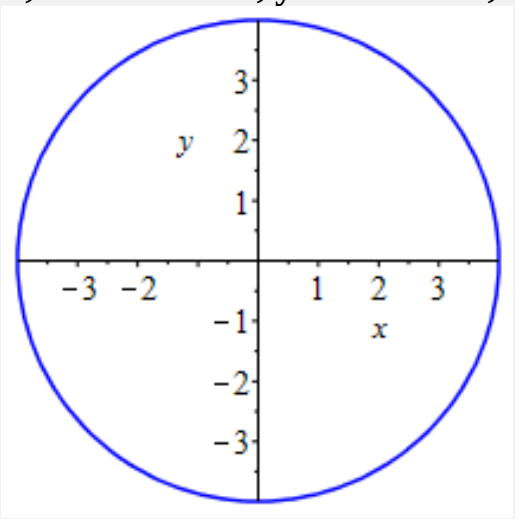


Рис. 6.1. График окружности, заданной неявным уравнением

Трехмерные поверхности также могут задаваться уравнениями неявного вида. Для построения таких графиков используется функции:

- ***implicitplot3d(expr1, x = a..b, y = c..d, z = p..q, <options>)***
- ***implicitplot3d(f, a..b, c..d, p..q, <options>).***

Все значения параметров такие же, как в предыдущем случае.

Пример 6.2. Построить поверхность эллипсоида вращения, заданного неявным уравнением: $\frac{x^2 + y^2}{1^2} + \frac{z^2}{4^2} = 1$.

```
with(plots):
implicitplot3d(x^2+y^2+z^2/16=1,x=-1..1,y=-2..2,z=-5..5);
```

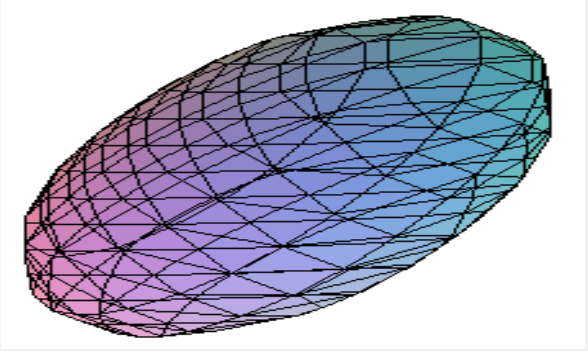


Рис. 6.2. График поверхности эллипсоида, заданный неявным уравнением

Эти примеры хорошо иллюстрируют технику применения функций i*mplicitplot()* и *implicitplot3d()*. Для наглядности фигуры на рис. 6.2 ее несколько развернули в пространстве с помощью мыши. Сравните этот результат с примером 5.8.

6.2.2. Совмещение графического вывода

Функция *plot3d()* позволяет строить одновременно несколько фигур на одном рисунке. Для этого необходимо вместо описания одной поверхности задать список описаний ряда поверхностей.

Пакет *plots* содержит две уникальные функции *display() и display3d()*, предлагающие более широкие возможности для вывода графических структур. С ее помощью можно строить весьма своеобразные фигуры. Синтаксис функции приведен ниже:

- ***display(p1, p2, p3,…., insequence);***
- ***display3d (p1, p2, p3,…., insequence).***

Здесь

- p1, p2, p3,… – графические структуры, в простейшем случае представляющие собой соответствующие функции графического вывода;
- insequence – управляет последовательностью вывода графиков. При значении *insequence = true* объекты появляются по одному и каждый предшествующий объект стирается перед появлением нового объекта. Если установить значение *insequence = false (значение по умолчанию)*, то все объекты появляются одновременно.

В примере 4.16 мы уже использовали функцию *display()* для вывода ряда графиков в строке по горизонтали. Сравните результаты вывода.

Пример 6.3. В качестве забавного примера наберите в командной строке:

```
with(plots, plottools):
p1:=sphere([3/2, 1/4, 1/2],1/4, color=red):
p2:=sphere([3/2, -1/4, 1/2], 1/4, color=red):
p3:=translate(rotate(cone([0, 0, 0], 1/2, 2, color=khaki), 0, Pi/2,-0.6), 3, 0, 1/4):
p4:=stellate(rotate(hemisphere(), Pi, 0, 0), 2):
display(p1, p2, p3, p4, scaling = constrained, style = patch);
```

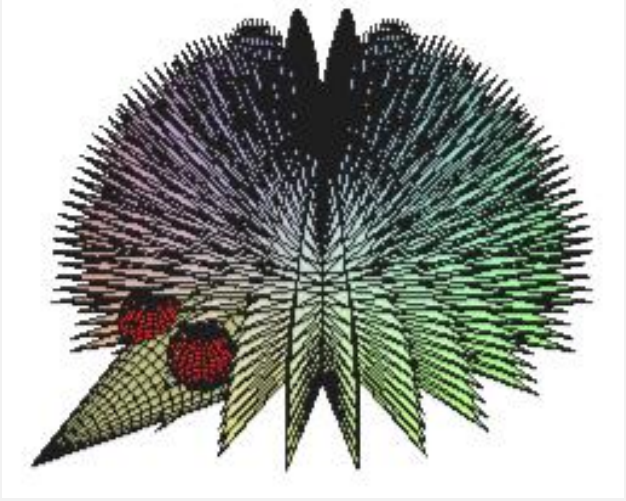


Рис. 6.3. Пример совмещения графического вывода командой display()

При построении этой фигуры использованы примитивы графического пакета *plottools,* содержащие ряд команд для создания трехмерных геометрических объектов. Наиболее употребительные команды пакета *plottools* приведены в приложении 4.1. Назначение других специальных функций, использованных при построении фигуры на рис. 6.3, определите самостоятельно, используя меню *Help*.

6.2.3. Отображение текста в пространстве графика

Для создания надписей в пакете *plots* служат две удобные команды:

- **textplot(*L, align = t, options*),**
- **textplot3d(*L, align = t, options*).**

Здесь

L – список или последовательность списков, задающих координаты и тест надписи: [[x,y, `текст1`],[x,y, `текст2`],…] или [[x,y,z, `текст1`],[x,y,z, `текст2`],…] для отображения текста в пространстве;

align – выравнивает текст в соответствии со следующими возможными значениями *t*: BELOW, RIGHT, ABOVE, LEFT;

option – опции графика, аналогичные процедуре *plot*.

Пример 6.4. Построить график функции y = sin(x) и подписать максимальное и минимальное значения функции.

```
with(plots):
p := plot(sin(x),x=-Pi..Pi,color=black,thickness = 2): delta := 0.05:
t1 := textplot([Pi/2,1+delta,`Максимум`],align=ABOVE):
t2 := textplot([-Pi/2,-1,`Минимум`],align=BELOW):
display({p, t1, t2});
```

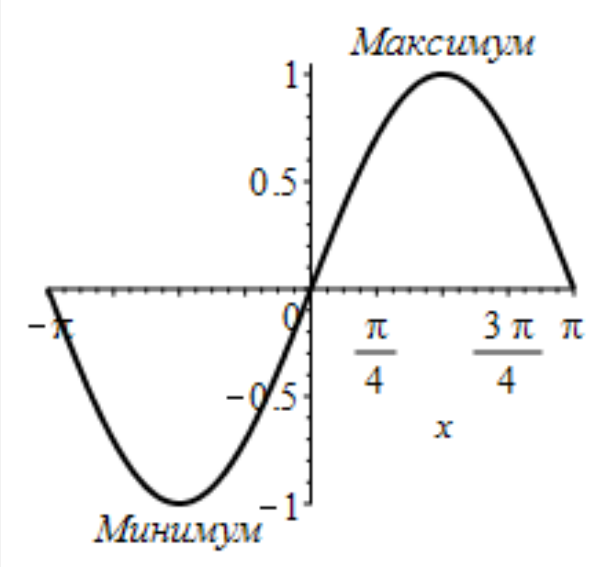


Рис. 6.4. Отображение текста на графике

Обратите внимание, что строковые литералы задаются в обратных кавычках.

6.3. Специальные приемы построения трехмерных графиков

Наглядность и информативность ряда графиков функций и результатов вычисления можно существенно увеличить, используя специальные функции пакета *plots*.

6.3.1. Построение графиков линиями равного уровня

Такие 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 графиков линиями равного уровня применяется функция *contourplot*, которая может использоваться в нескольких форматах:

- **contourplot(expr1, x=a..b, y=c..d);**
- **contourplot(f, a..b, c..d);**
- **contourplot([exprX, exprY, exprZ,…] s=a..b, t=c..d);**
- **contourplot([f,g,h,…] a..b, c..d).**

Здесь

f, g, h	–	функции;
expr1	–	выражение, описывающее зависимость высоты поверхности от координат *x* и *y*;
exprX..Z	–	выражения, описывающие поверхность в параметрической форме;
x, y, s и t	–	имена независимых переменных.

Пример 6.5. Построить график функции z = sin(a + b)cos(b - a) и отобразить линии равной высоты.

```
with(plots):
plot3d(sin(a+b)*cos(b - a), a=-Pi..Pi,b=-Pi..Pi, thickness = 2);
contourplot(sin(u + v)*cos(v - u), u=-Pi..Pi, v=-Pi..Pi, thickness = 2);
```

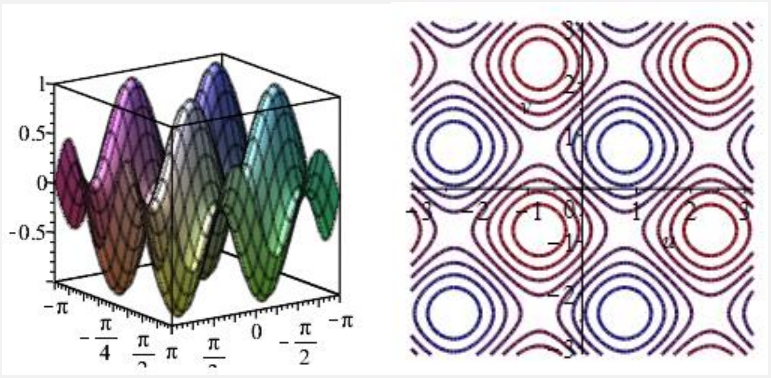


Рис. 6.4. Исходная поверхность и отображение двумерных линий равного уровня

Трехмерная команда *contourplot3d()*, так же как и ее двумерный аналог, отображает линии уровня поверхности и имеет такой же синтаксис. Отличие заключается в том, что двумерная команда рисует линии на плоскости, тогда как трехмерная отображает их в пространстве в плоскостях z = const, где постоянная z равняется значению, принимаемому функцией на соответствующей линии уровня.

Пример 6.6. Построить график функции z = sin(a + b)cos(b - a) и отобразить линии равной высоты командой *contourplot3d*.

```
with(plots):
plot3d(sin(a+b)*cos(b - a), a=-Pi..Pi,b=-Pi..Pi, thickness=2);
contourplot3d(sin(u + v)*cos(v - u), u=-Pi..Pi,v=-Pi..Pi, thickness=2);
```

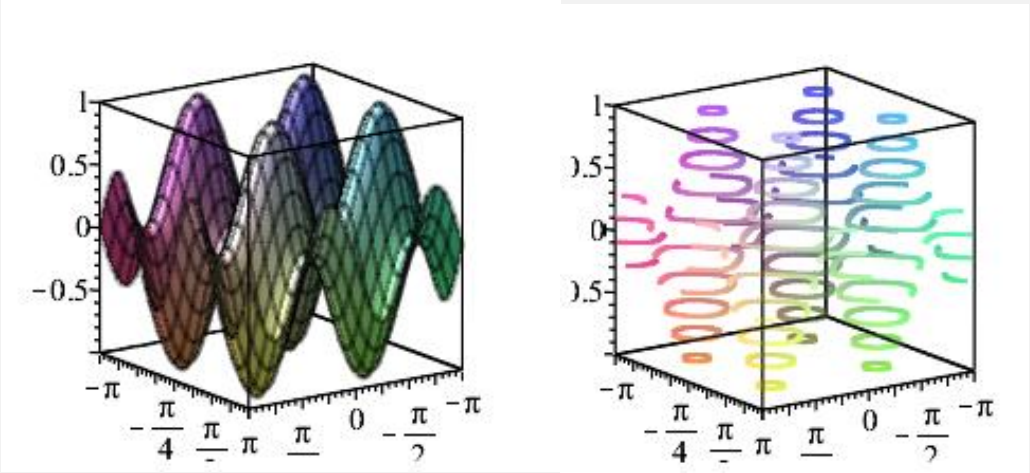


Рис. 6.5. Исходная поверхность и отображение трехмерных линий равного уровня

Следует отметить, что, хотя графики в виде линий равного уровня выглядят не так естественно, как обычные графики трехмерных поверхностей, у них есть одно существенное достоинство – экстремумы на таких графиках выявляются 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. Обнаружив скрытый экстремум, можно затем подобрать нужную проекцию и для трехмерного графика.

6.3.2. Построение графиков плотности

Иногда поверхности удобно отобразить на плоскости как графики плотности окраски. Чем выше высота поверхности, тем плотнее(темнее) окраска. Такой вид графиков создается функцией *densityplot()*. Она может быть использована в двух форматах:

- densityplot(exper1,x=a..b,y=c..d);
- densityplot(f, a..b, c..d).

Здесь назначения параметров аналогичны функции *contourplot()*.

Пример 6.6. Построить график плотности функции z = sin(a + b)cos(b - a).

```
with(plots):
plot3d(sin(a+b)*cos(b-a), a=-Pi..Pi, b=-Pi..2*Pi, thickness=2);
densityplot(sin(a+b)*cos(b-a), a=-Pi..Pi, b=-Pi..2*Pi);
```

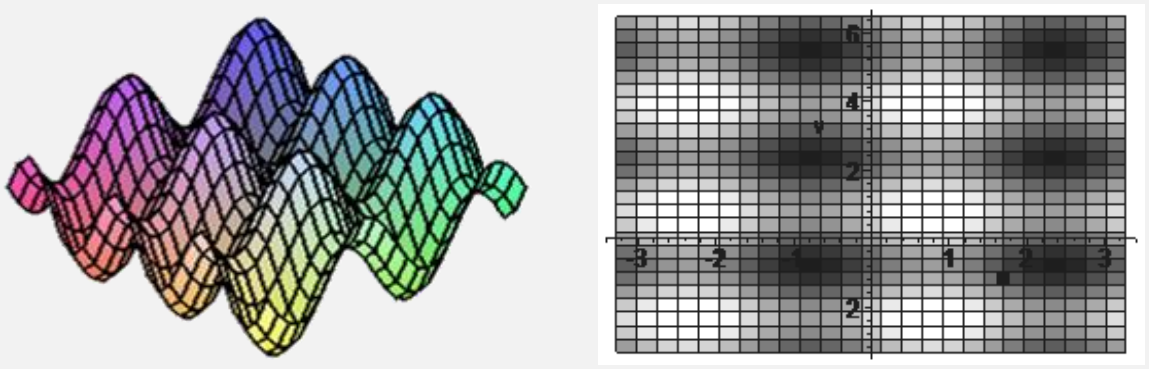


Рис. 6.6. Исходная поверхность и ее отображение в виде графика плотности

Обычно графики такого типа не очень выразительны, но часто применяются в физических исследованиях.

6.3.3. Графики векторного поля

Еще один распространенный способ графического представления результатов – графики векторного поля. Они часто применяются для отображения полей, например электрических зарядов. Особенность таких графиков в том, что для их построения используются стрелки, направления которых соответствуют направлению изменения градиента поля, а длина – значению градиента. Для построения таких графиков используются: функции для отображения двумерных векторные полей:

- **fieldplot(v, r1, r2,opt),**
- **gradplot(f, r1, r2, opt)**

и их трехмерные аналоги для отображения векторных полей в пространстве:

- **fieldplot3d(v, r1, r2, r3,opt),**
- **gradplot3d(f, r1, r2, r3,opt).**

Здесь

v – вектор или множество векторов;
f – функция или список функций;
r1 – пределы изменения переменных;
opt – необязательные дополнительные опции.

Команды *gradplot()* и *gradplot3d()* отображают поле градиента выражения, заданного в виде функции, зависящей от двух(трех) переменных в соответствии с диапазонами их изменения.

Команды *fieldplot()* и *fieldplot3d()* выполняют построение векторного поля по заданным двум(трем) компонентам вектора, зависящим от некоторых переменных в соответствии с диапазонами их изменения.

В командах можно использовать опцию *arrows*, значение которой определяет вид отображаемой стрелки.

Наглядность графиков векторных может быть улучшена совместным применением с графиком плотности.

Пример 6.7. Построить график плотности функции z = sin(a + b)cos(b - a).

```
with(plots):
f:=- sin(3*x)+sin(3*y):
p1:=gradplot(f,x=-2..1,y=-2..1,grid=[10,10], arrows=THICK):
p2:=densityplot(f, x=-2..1,y=-2..1, color=white):
display(p1,p2, axes=BOX);
```

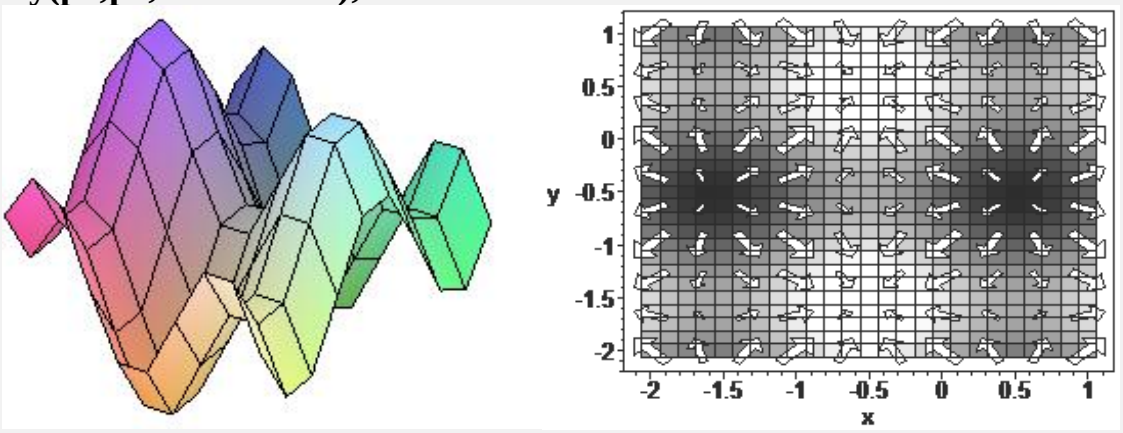


Рис. 6.7. Исходная поверхность и ее отображение в виде графиков плотности и векторного поля

6.3.4. Графическое представление решений дифференциальных уравнений

И последняя команда, на которой мы остановимся, это команда графического представления решения дифференциальных уравнений. Для этих целей в пакет *plots* включена команда *odeplot*.

Эта команда используется в следующем виде:

- **odeplot(s, vars, r, o).**

Здесь

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;
o – необязательные дополнительные опции.

Пример 6.8. Построить график решения дифференциального уравнения: y'(x) = y(x) в диапазоне от 0 до 2 при условии, что y(0) = 1.

Наберите в командной строке:

```
with(plots):
p:= dsolve({D(y)(x) = y(x), y(0)=1}, type=numeric, range=0..2):
odeplot(p, color=blue, thickness=2);
```

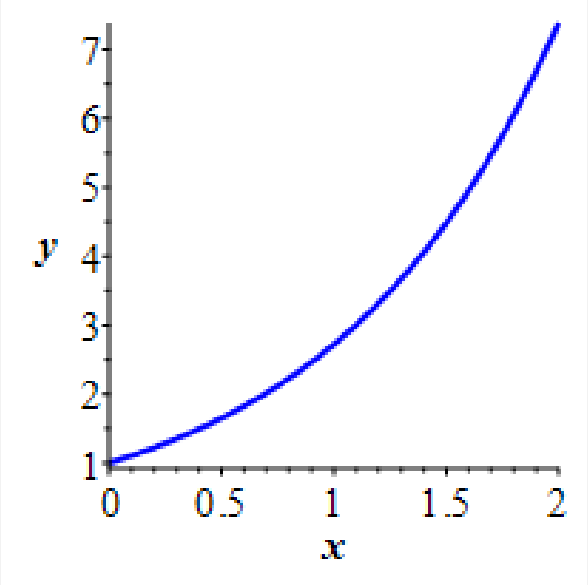


Рис. 6.8. Пример графического решения дифференциального уравнения

Приложения

6.1. Основные функции пакета *plots*

Параметр	Описание параметра
changecoords	Изменение системы координат.
complexplot	Построение двумерного графика на комплексной плоскости.
complexplot3d	Построение трехмерного графика на комплексной плоскости.
contourplot	Построение контурного графика.
contourplot3d	Построение контурного трехмерного графика.
coordplot	Построение координатной системы двумерных графиков.
coordplot3d	Построение координатной системы трехмерных графиков.
densityplot	Построение двумерного графика плотности.
display	Построение графика для списка графических объектов.
display3d	Построение графика для списка трехмерных 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.
listcontplot	Построение двумерного контурного графика .
listcontplot3d	Построение трехмерного контурного графика.
listdensityplot	Построение двумерного графика плотности.
listplot	Построение двумерного графика для списка значений.
listplot3d	Построение трехмерного графика для списка значений.
loglogplot	Построение логарифмического двумерного графика.
logplot	Построение полулогарифмического двумерного графика.
matrixplot	Построение трехмерного графика со значениями, определенными матрицей.
odeplot	Построение двумерного или трехмерного графика решения дифференциальных уравнений.
pointplot	Построение точками двумерного графика.
pointplot3d	Построение точками трехмерного графика.
polarplot	Построение графика в полярной системе координат.

Параметр	Описание параметра
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.
sphereplot	Построение трехмерной поверхности в сферических координатах.
surfdata	Построение трехмерного графика по численным данным.
textplot	Вывод текста на заданное место двумерного графика
textplot3d	Вывод текста на заданное место трехмерного графика

6.2. Основные функции пакета *plottools*

Инструментальный пакет графики *plottools* служит для создания графических примитивов типа: отрезков прямых, дуг, окружностей, конусов, кубиков и т.п. на плоскости и в пространстве.

Большинство примитивов имеют довольно очевидные название и синтаксис. Ниже приведены только наиболее употребительные из них.

Параметр	Описание параметра
arc([x,y], r, diap, opt)	Дуга окружности с центром в точке с координатами x и y и радиусом r. Параметр diap определяет углы в радианах начальной и конечной точек дуги.
arrow([x_n,y_n], [x_k,y_k], wd, wh, hh, opt]	Стрелка с началом в точке с координатами [x_n,y_n] и концом в точке с координатами [x_k,y_k]. Параметр wd задает толщину стрелки, wh ширину головки стрелки и hh высоту головки стрелки в частях ее длины.
cone([x,y,z], r, h, opt)	Конус с вершиной в точке, заданной координатами [x,y,z], направленный в положительном направлении оси z высотой h. В сечении z = h окружность имеет радиус r.
cuboid([x_1,y_1,z_1],[x_2,y_2,z_2], opt)	Прямоугольный параллелепипед с главной диагональю, определяемой двумя заданными точками.
curve([[x_1,y_1], …[x_n,y_n]], opt)	Кривая, заданная координатами своих точек. Отображается линейными сегментами, соединяющими соседние точки.

Параметр	Описание параметра
cylinder([x,y,z], r, h, opt)	Круговой цилиндр высотой h с образующей окружностью радиуса r с центром в точке [x,y,z]. Значение опции capped = true отображает цилиндр с закрытыми основаниями.
disk([x,y], r, opt)	Круг радиуса r с центром в точке с координатами [x,y]. По умолчанию круг на закрашивается.
ellipse([x,y], a, b, opt)	Эллипс с центром в точке с координатами [x,y] и полуосями a и b.
ellipticArc([x,y], a,b, diap, opt)	Дуга эллипса с центром в точке с координатами [x,y] и полуосями a и b. Параметр diap определяет углы в радианах начальной и конечной точек дуги и задается в виде диапазона.
hemisphere([x,y,z], r, opt)	Полусфера радиуса r с центром в точке с координатами [x,y,z]. Значение опции capped = true отображает полусферу с закрытым сечением.
hyperbola([x,y], a,b, diap, opt)	Гипербола с центром симметрии в точке [x,y] и эксцентриситетом $e^2 = a^2 + b^2$. Гипербола симметрична относительно вертикальной оси. Параметр diap определяет диапазон изменения значений гиперболы.
line([[x_1,y_1],[x_2,y_2]], opt)	Отрезок, соединяющий две заданные точки.
pieslice([x,y], r, diap, opt)	Центральный сектор круга с центром в точке с координатами [x,y] и радиусом r. Параметр diap определяет диапазон изменения угловой координаты сектора.
point([x,y], opt)	Точка с заданными координатами. Опции opt определяют цвет, размер, символ точки.
rectangle([[x_1,y_1],[x_2,y_2]], opt)	Прямоугольник, заданный координатами своих верхней левой и нижней правой точек. Опцией color можно определить цвет заливки прямоугольника.
sphere([x,y,z], r, opt)	Сфера радиуса r с центром в точке с координатами [x,y,z].
tetrahedron([x,y,z], s, opt)	Четырехгранник с центром в точке с координатами [x,y,z]. Масштабный параметр s по умолчанию равен 1.
torus([x,y,z], r, R opt)	Тор с радиусом меридиана r, центром в точке с координатами [x,y,z] и радиусом образующей окружности R.

7. ИНТЕРПОЛЯЦИЯ И АППРОКСИМАЦИЯ ЭКСПЕРИМЕНТАЛЬНЫХ ЗАВИСИМОСТЕЙ

Трудно представить себе область научно-технических расчетов более широкую и почитаемую, чем интерполяция и аппроксимация экспериментальных данных различными функциональными зависимостями.

7.1. Интерполяция экспериментальных данных

Предположим, имеется набор значений y_i реализации некоторой функции для значений аргумента x_i. Задача интерполяции заключается в том, чтобы построить функцию $\varphi(x)$, которая в узловых точках x_i принимает значения y_i, т.е. $\varphi(x) = y_i$ или, другими словами, идентична исходной последовательности в узлах. В этом смысле такую задачу часто называют *аппроксимацией точной в узлах.*

Maple имеет встроенную в ядро интерполяционную функцию interp(X, Y, V) Здесь X иY – векторы, содержащие значения узловых точек x_i и функции y_i. Векторы X иY должны содержать n + 1 =N координат точек исходной зависимости, где n – степень интерполяционного полинома. Переменная V указывает имя переменной интерполяционного полинома.

Приведем пример использования функции interp(X, Y, V). Пояснения даны непосредственно в листинге (рис. 7.1).

restart:

with(CurveFitting): with(plots): # *Подключаем необходимые пакеты*

X:=[0.6, 2, 3, 4, 5]; **Y:**=[0.4, 1, 2, 2, 1]; # *Задаем базовые векторы данных*

$$X := [0.6, 2, 3, 4, 5]$$
$$Y := [0.4, 1, 2, 2, 1]$$

f:=interp(X,Y, x); # *Вычислим интерполяционный полином*

$$f := 0.04933791686x^4 - 0.6907308363x^3 + 3.002992101x^2 - 4.098039208x + 1.920550035$$

Подготовим данные для графической иллюстрации
Pare := (X,Y) → [X,Y]:
XY:=zip(Pare,X,Y,2);

$$XY := [[0.6, 0.4], [2, 1], [3, 2], [4, 2], [5, 1]]$$

Создадим графический объект по экспериментальным данным
F1:=plot([XY], x=0..5,0..2.5, color = blue, style = POINT, symbol = BOX, symbolsize=18):
Создадим графический объект по аппроксимирующей функции f

F0:=plot(f(x), x=0..5, 0..2.5, color = red, style = LINE):

Строим график

display(F0, F1);

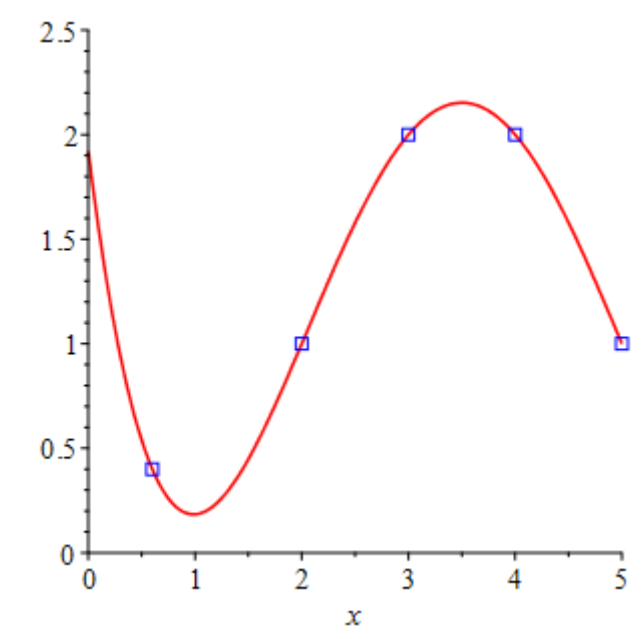


Рис. 7.1. Пример использования функции interp()

В ряде случаев – точная интерполяция в узлах для построения сглаженной огибающей экспериментальных значения и вычисления промежуточных данных. Однако точность такой интерполяции падает с увеличением степени аппроксимирующих полиномов. Этого недостатка можно избежать, используя для аппроксимации отрезки полиномов невысокой степени, применяемые для представления части узловых точек. Самым известным методом такой аппроксимации является сплайн-аппроксимация. При этом аппарат сплайн-аппроксимации позволяет получить полиномы, которые дают в узловых точках непрерывность не только представляемой ими функции, но и ее первых производных.

В Maple есть пакет, который предназначен специально для выполнения различного ряда интерполяций и аппроксимаций. Речь идет о пакете CurveFitting. Ниже описываются основные функции и процедуры, которые становятся доступными после подключения этого пакета.

7.2. Функция построения B-сплайновых кривых BsplineCurve

Для построения кривых B-сплайнов используется функция BsplineCurve. Она может использоваться в следующих формах:

BsplineCurve(xydata, v, opts),

BsplineCurve(xdata, ydata v, opts).

Здесь

- xydata – список, массив или матрица точек;

- xdata – список, массив или вектор значений независимой переменной;
- ydata – список, массив или вектор значений зависимой переменной;
- v – имя независимой переменной;
- opts – необязательный параметр в форме одного или более выражений вида order = k или knots = knotlist.

Пример применения функции BsplineCurve с порядком, заданным по умолчанию, и с пятым порядком показаны на рис.7.2. Пояснения даны непосредственно на рисунке.

```
restart:
with(CurveFitting): with(plots): # Подключаем необходимые пакеты
xydata:=[[0.1,0],[1,0.9],[2.5,6],[3,6],[4,5],[5,3],[6,2]]: # Задаем значения данных
# Создадим графический объект по исходным экспериментальным точкам
F0:=plot(xydata, x=0..6, 0..6, color=blue, style = POINT, symbol = asterisk, symbolsize=30):
# Вычислим интерполяционный полином без указания параметра
C1:=BSplineCurve(xydata,x):
# Создадим графический объект интерполяционного полинома
F1:=plot(C1,x=0..6,0..6,color=red,style=LINE, thickness=8):
# Вычислим интерполяционный полином явным указанием параметра
C2:=BSplineCurve(xydata,x, order=5):
Создадим графический объект нового интерполяционного полинома
F2:=plot(C1,x=0..6,0..6,color=GREEN,style=point,symbolsize=8):
# Строим графики
display(F0,F1,F2);
```

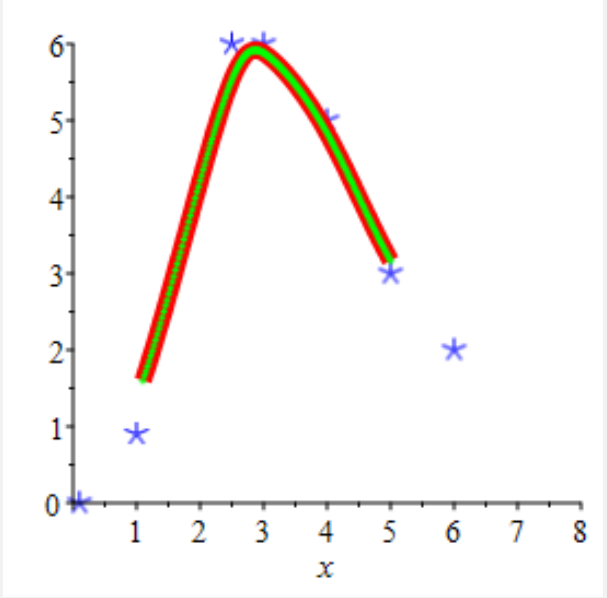


Рис. 7.2. Пример использования функции BsplineCurve ()

Следует отметить, что при малом числе точек аппроксимация B-сплайнами дает невысокую точность, что и видно на рис. 7.2.

7.3. Аппроксимация экспериментальных данных

Задача аппроксимации данных, в отличие от интерполяции (*аппроксимации точной в узлах)* несколько иная, построить приближенную аналитическую зависимость табличных значений.

В Maple этой цели служит процедура LeastSquare(), которая позволяет аппроксимировать табулированную функцию, используя метод наименьших квадратов. Такой метод применяется, как правило, в тех случаях, когда число подгоночных параметров (параметров, которые могут выбираться пользователем) существенно меньше числа базовых табличных точек. Идея метода состоит в том, чтобы найти такие значения параметров аппроксимирующей функции $\varphi(x)$, чтобы ее отклонение от табличных значений в узловых точках $y_i(x_i)$ было минимальным. С получения простой аппроксимации сложной зависимости нередко начинаются (а часто и заканчиваются) научные исследования во многих областях как прикладной, так и фундаментальной науки.

Использование процедуры LeastSquare() позволяет сразу получить желаемый результат. Для этого следует ввести соответствующую команду, указав параметрами либо два списка с узлами и значениями функции, либо один с элементами-списками, в которых первый элемент является узловой точкой, а второй – значением функции в этой точке. Ниже представлено описание опций, которые помимо обязательных элементов могут использоваться вместе с программой LeastSquare.

Опция	Описание опции
curve	Определяет общий вид аппроксимирующей кривой. Зависимость должна быть линейной по параметрам.
params	Может использоваться в тех случаях, когда явно задана опция curve. Значения этой опции задаются в виде списка переменных, которые следует считать параметрами оптимизации. Если значения не заданы, параметрами оптимизации считаются все переменные, отличные от той, которая указана в параметрах процедуры как независимая.
weight	Определяет весовые множители. Ее значением может быть список из неотрицательных чисел, а число элементов списка должно строго соответствовать числу узловых точек.

Для того чтобы продемонстрировать работу процедуры LeastSquare, поступим следующим образом. Сначала для тренировки создадим базовую функцию, которую затем будем восстанавливать по табличным данным. В качестве примера рассмотрим функцию $f = 10 + \exp(-x) \cdot \sin(2\,\pi\,x)$. Построим график этой функции и попробуем аппроксимировать ее процедурой LeastSquare.

Программный код и результаты расчетов показаны на рис. 7.3.

```
restart:
with(CurveFitting):with(plots): # Подключаем необходимые пакеты
f:= (x) → 10+exp(-x)·sin(2·π·x);  # Задаем базовую функцию
```

$$f := x \mapsto 10 + e^{-x} \sin(2\pi x)$$

```
# Создадим графический объект этой функции
F0:=plot(f(x),x=0..4, 9.5..11, color=blue, style = LINE):
# Выполним аппроксимацию в виде по умолчанию
F1:=LeastSquares(B, x);
```

$$F1 := 10.1172925726147 - 0.0430148483311683\,x$$

```
# Создадим графический объект аппроксимационной функции
F1:=plot(F1,x=0..4, 9.5..11, color=red, style = LINE, linestyle = DASH):
# Построим исходную функцию и ее аппроксимацию на одном графике
```

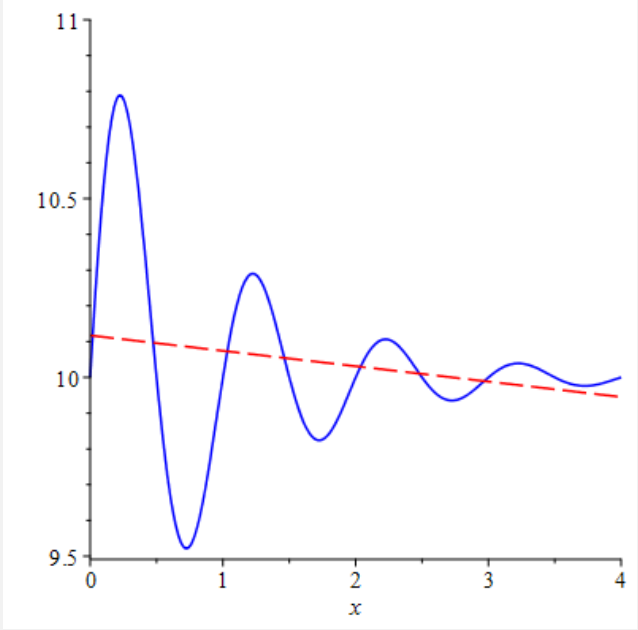


Рис. 7.3. Аппроксимация функции процедурой LeastSquare без указания параметров

Между тем результат не очень впечатляет. Дело в том, что процедура LeastSquare по умолчанию, т.е. без указания параметров выполняет аппроксимацию только линейной функцией. В ряде случаев это бывает полезно, так как большинство нелинейных зависимостей удается свести к линейным с помощью простых линеаризующих преобразований.

Попробуем пойти другим путем. В большинстве случаев аппроксимационные модели той или иной задачи, как правило, известны либо из теоретических предсказаний, либо из принципа "здравого смысла". Кстати, моделей, описывающих физические процессы, не более десятка: экспонента, гипербола, степенная функция, гармонические функции, логарифмическая функция и т.п. Применим знание аппроксимируемой зависимости и в наших расчетах. Чтобы задача была не слишком простой, внесем некий элемент случайности. Для этого воспользуемся генератором случайных чисел. Будем табулировать значения функции так, чтобы после вычисления значений функции в узле к каждому значения добавлялось случайное число в диапазоне от 0 до 1. В этом случае табличные значения функции будут иметь погрешность порядка 1%.

Программный код и результаты вычислений показаны на рис. 7.4.

```
restart:
with(CurveFitting): with(plots): # Подключаем необходимые пакеты
f:= (x) → 10+exp(-x)·sin(2·π·x);  # Задаем базовую функцию
```

$$f := x \mapsto 10 + \mathrm{e}^{-x}\sin(2\pi x)$$

```
# Создадим графический объект этой функции
F0:=plot(f(x), x=0..5, 9.5..11,colo r= blue, style = LINE);
k:=rand(1..10)/100: evalf(k(), 2): #Генератор случайных чисел
# Испортим исходные данные и создадим новый графический объект
F1:=plot((f(x)+evalf(k(), 2)), x = 0..5, 9.5..11, color = blue, style = POINT,
symbol = BOX, symbolize = 14):
# Построим исходную и «испорченную» функции
display(F0, F1);
```

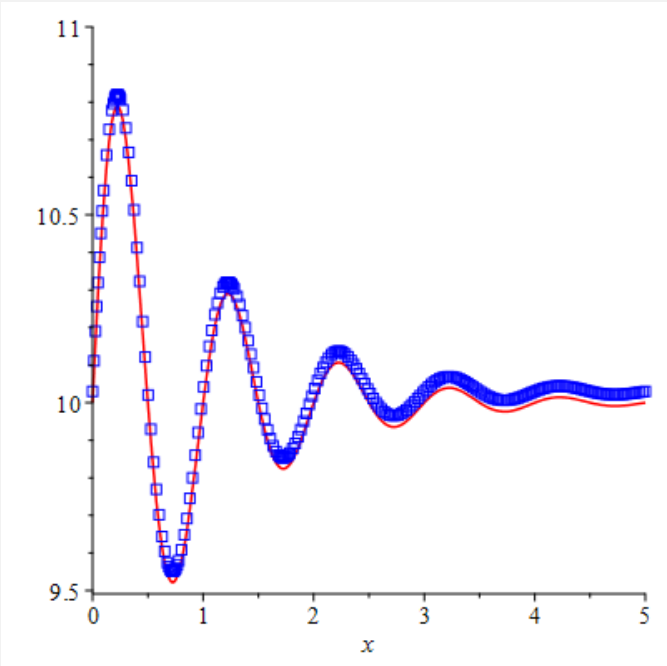


Рис. 7.4. Аппроксимация исследуемой функции процедурой LeastSquare с полным указанием ее параметров: сплошная линия – исходная функция, «квадраты» – ее аппроксимация

Результат, как видно, налицо! Однако, чтобы в общем случае методом наименьших квадратов эффективно строить аппроксимирующие функции, желательно знать вид аппроксимируемой функциональной зависимости.

Задания для самостоятельного решения

1. В результате эксперимента была определена табличная зависимость.

S	0	1	1.5	2	2.5	3	3.5	4	4.5	5
P	12	10.1	11.58	17.4	30.68	53.6	87.78	136.9	202.5	287

Подобрать функциональную зависимость. Построить график зависимости и найденной функции.

2. В результате эксперимента была определена табличная зависимость.

S	0.5	1.5	2	2.5	3	3.5	4	4.5	5	
P	3.99	5.65	6.41	6.71	7.215	7.611	7.83	8.19	8.3	

Подобрать функциональную зависимость. Построить график зависимости и найденной функции.

3. В результате эксперимента была определена табличная зависимость.

S	0	0.5	1	1.5	2	2.5	3	3.5	4	
P	2.31	2.899	3.534	4.412	5.578	6.92	8.699	10.69	13.39	

Подобрать функциональную зависимость. Построить график зависимости и найденной функции.

4. В результате эксперимента была определена табличная зависимость.

S	0.2	0.7	1.2	1.7	2.2	2.7	3.2			
V	2.3198	2.8569	3.5999	4.4357	5.5781	6.9459	8.6621			

Определить параметры функции $V(S) = A\, S^b\, e^{CS}$. Построить график зависимости и найденной функции.

5. Подобрать аналитическую зависимость вида $Y = X/(aX - b)$, используя данные таблицы

X	3	3.1	3.2	3.3	3.4	3.5	3.6	3.7	3.8	3.9
Y	0.61	0.6	0.592	0.58	0.585	0.583	0.582	0.57	0.572	0.571

Построить график зависимости и найденной функции.

8. ЭКСПОРТ И ИМПОРТ ДАННЫХ

Решение практических задач предполагает взаимодействие пакета Maple с другими пакетами и приложениями и в первую очередь офисными и графическими программами.

8.1. Запись данных в файлы

Maple умеет все или почти все. Тем не менее часто возникает необходимость передачи рассчитанных данных в другие узкоспециализированные программы или непосредственно в экспериментальные установки. Обмен информацией между Maple и внешней средой чаще всего осуществляется через файлы текстового формата, поскольку именно с такими файлами могут работают практически все программы. Для *записи* данных в файл служит оператор writedata, который используется в следующих формах:

- writedata[APPEND](fileID, data);
- writedata[APPEND](fileID, data, format).

Здесь fileID – имя или дескриптор файла данных, data – список, вектор или матрица данных, format – спецификация формата данных (integer, float или string). Необязательный указатель APPEND используется, если данные должны дописываться в уже созданный файл.

Пример использования операции записи данных в файл показан на рис. 8.1.

```
# Для примера создадим модельный массив.
A := array([[1.5, 2.2, 3.4], [2.7, 3.4, 5.6], [1.8, 3.1, 6.7]]):
fd := fopen(`C:\\1\\A. txt`, WRITE):  # Откроем канал для записи данны.х
writedata(fd, A); # Запишем данные в указанную директорию.
fclose(fd); # Закроем  канал для записи данных.
# Вот что получилось:
```

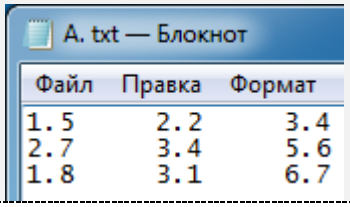


Рис. 8.1. Запись данных во внешнюю программу

8.2. Считывание данных из файлов

Обширные возможности Maple делают привлекательным применение этой программы и для автоматической обработки данных, поступающих от каких-либо экспериментальных установок или внешних узкоспециализированных программ.

Считывание данных из файла filename обеспечивает функция readdata:

- readdata(fileID, n);
- readdata(fileID, format, n).

Здесь n – целое положительное число, задающее число считываемых столбцов.

Ниже представлены пример такой операции. Для этого используем сохранённый в предыдущем примере текстовый файл A.txt (рис. 8.2).

```
fd := fopen(`C:\\1\\A.txt`, READ):
L:=readdata(fd, 3 ):  #3столбца
M:=convert(L, array);
fclose(fd):
```

$$M := \begin{bmatrix} 1.5 & 2.2 & 3.4 \\ 2.7 & 3.4 & 5.6 \\ 1.8 & 3.1 & 6.7 \end{bmatrix}$$

Рис. 8.2. Считывание данных в Maple

В Maple начиная с 10-й версии есть и более мощное средство ввода данных – ассистент импорта данных ImportData. Он позволяет вводить данные (в том числе матрицы рисунков) из файлов множества форматов (рис. 8.3).

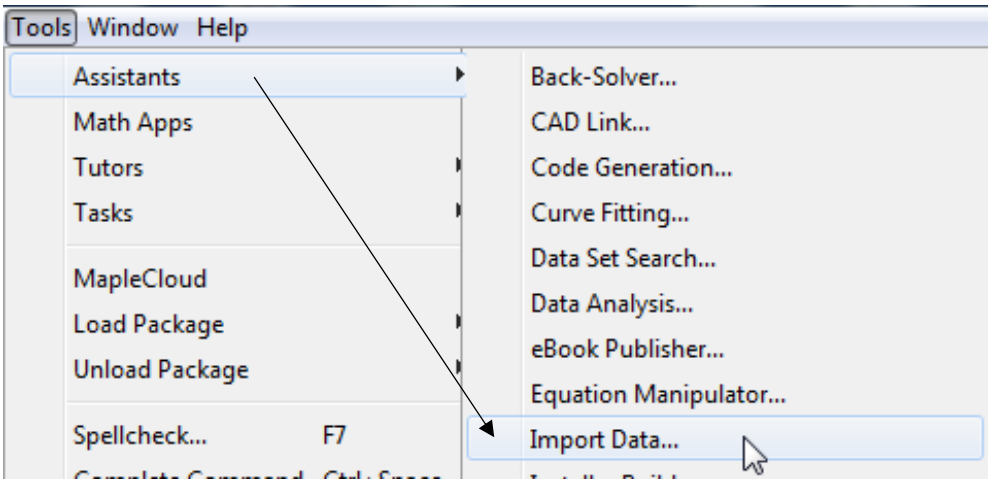


Рис. 8.3. Выбор ассистента импорта данных в Maple

При обращении к нему из подменю Assistants позиции меню Tools открываются два окна. Верхнее окно (рис. 8.4) служит для выбора файла с данными.

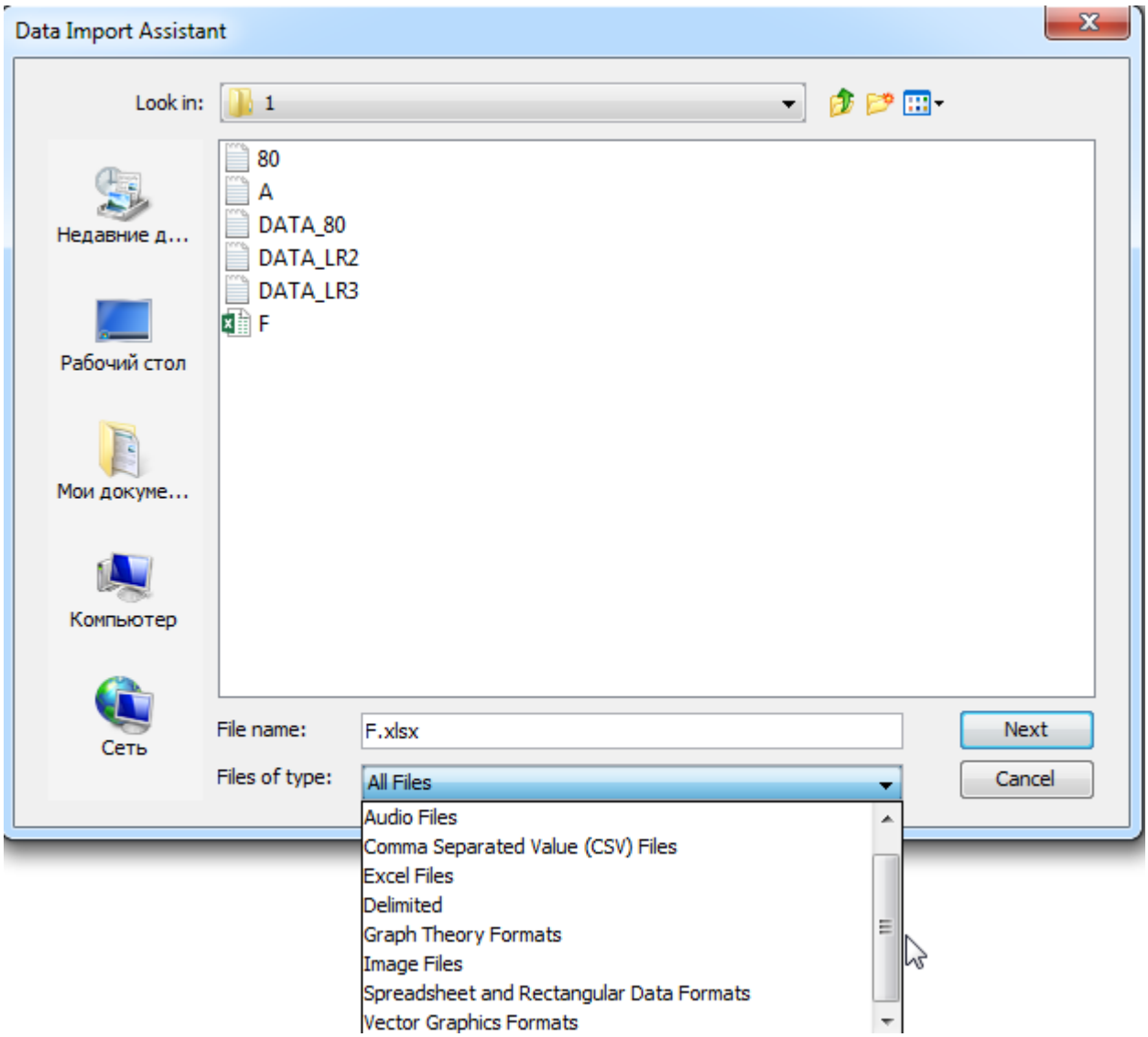
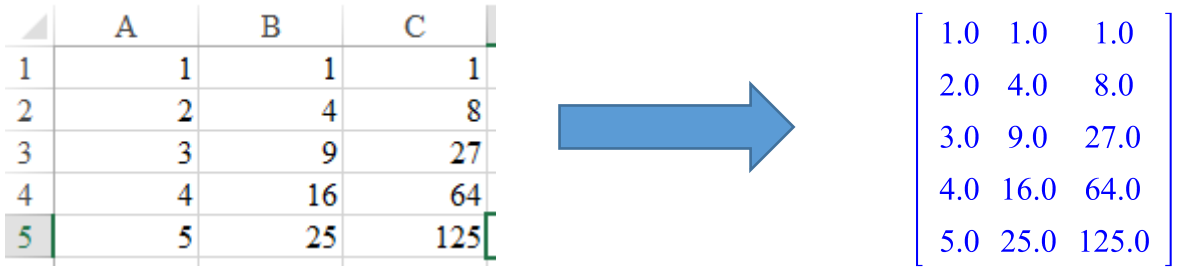


Рис. 8.4. Окно выбора файлов с данными

Обратите внимание на обширный список возможных типов файлов внизу этого окна. Он включает в себя аудио- и видеофайлы, файлы рисунков различного формата и другие типы файлов, представляемые которыми файлы могут быть представлены в векторной или матричной форме. Число типов матричных и векторных данных постоянно расширено и систему можно использовать для обработки таких данных, как звуки и изображения.

Выберем для тренировки модельный файл Excel: F.xlsx и нажмем Next. Появится новое окно (см. рис. 8.5), в котором нужно указать идентификатор рабочего листа Excel и диапазон нужных данных. Последующие два окна можно пропустить. В результате на рабочем поле Maple появится матрица, с которой можно проводить необходимые вычисления. При этом способе импорта указывать явный путь к файлу не нужно.

Исходная таблица и внедренная матрица показаны на рис. 7.5.



	A	B	C
1	1	1	1
2	2	4	8
3	3	9	27
4	4	16	64
5	5	25	125

$$\begin{bmatrix} 1.0 & 1.0 & 1.0 \\ 2.0 & 4.0 & 8.0 \\ 3.0 & 9.0 & 27.0 \\ 4.0 & 16.0 & 64.0 \\ 5.0 & 25.0 & 125.0 \end{bmatrix}$$

Рис. 8.5. Результат импорта таблицы Excel в Maple

Аналогичный результат можно получить, используя функцию ImportData(). При активации функции ImportData() без указания параметров открывается окно выбора файла, аналогичное рис. 8.4. В дальнейшем процедура полностью аналогична описанной выше. Достоинством такого импорта данных является то, что данным можно сразу присвоить необходимый идентификатор: M:= =ImportData().

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Ефремов Ю. С. Методы математической физики в пакете символьной математики Maple : уч. пособие для академического бакалавриата/М.: Из-во Юрайт, 2019. – 302 с.

2. Егоров, А. И. Обыкновенные дифференциальные уравнения и система Maple / М: СОЛОН-ПРЕСС, 2016. – 392 с.

3. Дьяконов В. П. Maple 11.10.12/13/14 в математических расчетах: самоучитель/М.: ДМК-пресс, 2011. – 699–701 с.

4. Васильев А.Н. Maple 8. Самоучитель./М.: Издательский дом «Вильямс», 2003. –352 с.

СОДЕРЖАНИЕ

Учебное издание

Составитель

Гилев Валерий Григорьевич

СОВРЕМЕННЫЕ ПАКЕТЫ ПРИКЛАДНЫХ ПРОГРАММ. ОСНОВЫ РАБОТЫ В MAPLE

Учебное пособие

Редактор *Л. В. Хлебникова*
Корректор *М. Н. Демидова*
Компьютерная верстка: *В. Г. Гилев*

Объем данных 2,61 Мб
Подписано к использованию 27.07.2022

Размещено в открытом доступе
на сайте www.psu.ru
в разделе НАУКА / Электронные публикации
и в электронной мультимедийной библиотеке ELiS

Издательский центр
Пермского государственного
национального исследовательского университета
614990, г. Пермь, ул. Букирева, 15