ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение высшего профессионального образования
«ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

А.В. Богданов

ПАКЕТ MATHEMATICA ДЛЯ ИНЖЕНЕРНЫХ ВЫЧИСЛЕНИЙ

Часть I

Учебное пособие

Издательство
Томского политехнического университета
2008

УДК 681.3
ББК
Б73
Богданов А.В.
Б73 Пакет Mathematica для инженерных вычислений. Часть I: учебное пособие. – Томск: Изд-во Томского политехнического университета, 2008. – 97 с.

ISBN

В настоящей части пособия раскрываются основные сведения по работе в пакете Mathematica фирмы Wolfram Research признанного лидера в разработке пакетов аналитической математики. Учебное пособие предназначено для магистров, обучающихся по программам «Физика ускорителей» и «Медицинская физика». Кроме того, материал, изложенный в пособии, может быть полезен при изучении курсов «Компьютерный практикум» и «Летний учебный практикум» студентами специальностей 140302 «Физика атомного ядра и частиц» и 140307 «Радиационная безопасность человека и окружающей среды», и при выполнении лабораторных работ по курсу «Теоретическая физика» студентами специальностей: 140305 «Ядерные реакторы и энергетические установки», 140306 «Электроника и автоматика физических установок» и др.

УДК 681.3

Рецензенты
Доктор физико-математических наук, профессор МГТУ СТАНКИН
А.М. Кольчужкин

Доктор физико-математических наук, профессор ТПУ
Ю.Н. Адищев

ISBN

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Многие математические вычисления в экономике, физике и других современных науках требуют достаточно большого количества расчетов. Очень часто приходится обрабатывать полученные результаты и визуализировать их. Выполнение таких операций вручную является достаточно трудоемкой работой. Поэтому для упрощения аналитических и численных расчетов были придуманы системы компьютерной математики. Они интегрируют в себе современный интерфейс пользователя, возможности решения, как аналитических, так и численных задач, а также мощные средства визуализации полученных результатов с использованием различных типов графиков.

Ярким представителем систем компьютерной математики (СКМ) является пакет *Mathematica,* разработанный фирмой Wolfram Research Inc, во главе с ее президентом и главным разработчиком программ Стивеном Вольфрамом (*Stephan Wolfram*). Этот пакет считается одной из лучших СКМ наряду с *Maple* и *MatLab*.

В данной части пособия рассматриваются основные принципы работы в пакете *Mathematica* на примере достаточно простых задач. Рассмотрены вопросы оформления расчетов с использованием стилей пакета и возможностей форматирования. Кроме того, достаточно подробно рассмотрены возможности визуализации результатов вычислений с помощью различных типов графиков.

1. ИНТЕРФЕЙС ПРОГРАММЫ

Так как процесс установки и запуска пакета *Mathematica* не сильно отличается от процесса установки и запуска других программ и может быть изучен по другой литературе, например по [1] или [2], то перейдем сразу к рассмотрению интерфейса программы, представленного на рис. 1.

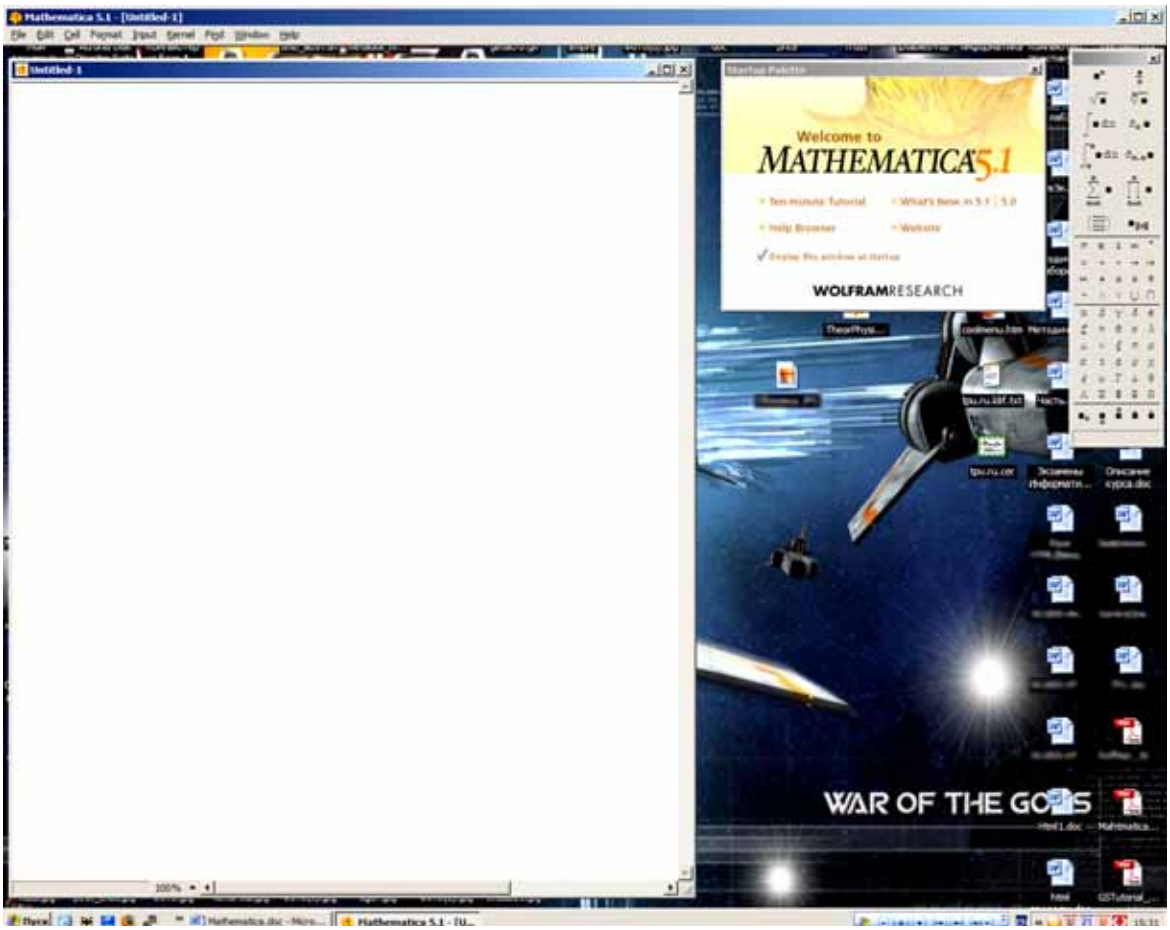


Рис. 1. Интерфейс пакета Mathematica

Из этого рисунка видно, что внешний вид пакета *Mathematica* немного отличается от других программ. Прежде всего тем, что интерфейс программы состоит из нескольких окон, которые не имеют одного родительского окна. Вверху представленного рисунка находиться окно с основным меню программы. Оно является общим для всех дополнительных окон ввода[1]. В данном примере открыто одно окно ввода, и оно располагается на рисунке слева. В этом окне производится ввод информации для последующих вычислений.

[1] В версии пакета для операционных систем, отличных от Windows, например Linux-меню располагается на каждом окне ввода.

Справа от окна ввода располагаются две дополнительные панели. Первая панель от основного окна позволяет получить быстрый доступ к некоторой информации, например о новых возможностях пакета *Mathematica 5.1*, вызов справки пакета, 10-минутное руководство по работе с пакетом и т. д. Вторая панель – это так называемая панель быстрого ввода, которая позволяет упростить ввод часто встречающихся операций и символов.

В пакете *Mathematica* существуют и другие панели для быстрого набора и поиска необходимых функций. О том, как их можно будет открыть, вы узнаете из разд. 3.

Далее рассмотрим применение пакета *Mathematica* для решения различных задач.

2. ИСПОЛЬЗОВАНИЕ ПАКЕТА *MATHEMATICA* В КАЧЕСТВЕ ПРОСТОГО КАЛЬКУЛЯТОРА

Пакет *Mathematica* позволяет решать достаточно большое количество задач, но мы рассмотрим вначале самые простые операции, которые он может решать, а именно используем пакет *Mathematica* в качестве калькулятора. Например, если мы хотим вычислить выражение $4^2 + 5$, необходимо набрать это выражение в пакете *Mathematica* и запустить текущий блок на выполнение посредством нажатия на клавиатуре клавиш **<Shift>+<Enter>** или **<GreyEnter>** (клавиша *Enter* на дополнительной цифровой клавиатуре). В результате выполнения данной операции получим:

После выполнения вычисления пакет *Mathematica* готов для ввода следующих значений. Признаком этого является горизонтальная черта, появляющаяся после результатов вычислений. При начале ввода информации на месте черты появится новый блок, в котором и будет отображаться информация. При вычислении исходное выражение присваивается объекту **In[n]**, а результат вычисления – объекту **Out[n]**, где n – номер проводимого вычисления. Мы можем использовать синонимы **In[n]** и **Out[n]** при дальнейших вычислениях. Например, прибавим 2 к полученному в предыдущем примере результату.

```
In[4]:= Out[1] + 2

Out[4]= 23
```

Как видно из предыдущих примеров, исходные данные и результаты вычислений присваиваются соответствующим объектам с использованием различных операций присваивания. Вычисляемое выражение присваивается объекту **In[n]** с помощью операции **:=**, а результат вычисления объекту **Out[n]** с помощью операции =. Знаки **:=** и = имеют существенно различный смысл и называются **отложенным** и **немедленным присвоением**. Отличием между немедленным и отложенным присваиванием является то, что выражение, присвоенное первым способом, вычисляется сразу, а во втором случае вычисление сразу не происходит, а присваивается аналитическое выражение и вычисляется оно только в момент использования.

Далее в табл. 1 приведем основные арифметические операции.

Таблица 1

Основные арифметические операции

Операция	**Описание**
B1+B2	сложение двух чисел (выражений)
B1–B2	вычитание выражения 2 из выражения 1
B1*B2 **или** B1 B2	умножение двух выражений. Данная операция может записываться двумя способами. Первый способ – использование знака * для умножения. Второй способ – использование пробела вместо знака *
B1/B2	деление B1 на B2
B^n	возведение выражения в степень n

3. ИСПОЛЬЗОВАНИЕ ПАКЕТА В КАЧЕСТВЕ ИНЖЕНЕРНОГО КАЛЬКУЛЯТОРА

Прежде чем мы начнем дальнейшее рассмотрение возможностей пакета *Mathematica,* обозначим некоторые правила набора выражений в нем.

Первое, на что нужно обратить внимание, – это то, что при наборе выражений различаются заглавные и прописные буквы. Также следует помнить, что имена встроенных функций всегда начинаются с заглавной буквы, например **Sin.** Если имя функции составное, т. е. состоит из нескольких корней, то каждая составная часть в имени функции начинается с заглавной буквы, например **ArcSin**.

Второй важной особенностью пакета *Mathematica* является то, что в нем использование различных типов скобок жестко регламентировано.

Круглые скобки «( )» используются для группировки выражений и изменения порядка вычисления выражений, т.е. используются для тех же целей, что и при обычных математических вычислениях. Использование других типов скобок, как это принято при обычных математических вычислениях, не допускается.

Квадратные скобки «[ ]» являются основным признаком функций и используются при записи функций, внутри которых записываются аргументы, разделенные запятыми, например **Sin[x]**.

Фигурные скобки «{ }» используются при работе с массивами, списками и матрицами. Использование фигурных скобок более подробно будет изучено позже.

Любые инженерные калькуляторы, как электронные, так и программные, имеют набор констант, которые можно использовать при вычислениях. Пакет *Mathematica* тоже имеет достаточно большой набор таких предопределенных констант. В табл. 2 приведены лишь некоторые из них.

Таблица 2

Список основных констант пакета Mathematica

Обозначение	**Описание**
Pi или π	константа, хранящая значение числа π = 3.14159…
E или ©	константа, хранящая значение числа e = 2.71828…
I или ™	мнимая единица
Infinity или ∞	обозначает бесконечность, используется при некоторых вычислениях, например при вычислении пределов, сумм и интегралов
Degree или °	константа для преобразования радиан в градусы, которая численно равно количеству радиан в одном градусе

Стоит заметить, что в пакете *Mathematica* возможен ввод некоторых обозначений, функций и констант двумя способами. Первый способ заключается в наборе полного имени (в табл. 2 первое обозначение), а второй – в использовании панелей быстрого ввода. Одна из панелей появляется при первом запуске программы. Эта и дополнительные панели могут быть вызваны посредством пункта меню ***File=>Palettes***.

Помимо констант, в каждом калькуляторе имеется набор алгебраических и тригонометрических функций, используемых при вычислениях. Рассматриваемый пакет тоже имеет такой набор функций. Основные функции приведены табл. 3.

Таблица 3

Основные функции

Функция	**Описание**
Abs[x]	вычисляет модуль числа x
Sqrt[x]	вычисление квадратного корня от числа x
Exp[x]	вычисление экспоненты в степени x
Log[x]	вычисление натурального логарифма от числа x
Log[b,x]	вычисление логарифма x по основанию b
Sin[x]	вычисление синуса от числа x
Cos[x]	вычисление косинуса от числа x
Sec[x]	вычисление секанса от числа x
Csc[x]	вычисление косеканса от числа x
Tan[x]	вычисление тангенса от числа x
Cot[x]	вычисление котангенса от числа x
Sinh[x]	вычисление гиперболического синуса от числа x
Cosh[x]	вычисление гиперболического косинуса от числа x
Sech[x]	вычисление гиперболического секанса от числа x
Csch[x]	вычисление гиперболического косеканса от числа x
Tanh[x]	вычисление гиперболического тангенса от числа x
Coth[x]	вычисление гиперболического котангенса от числа x
ArcSin[x]	вычисление обратной функции синуса для числа x
ArcCos[x]	то же для косинуса
ArcSec[x]	то же для секанса
ArcCsc[x]	то же для косеканса
ArcTan[x]	то же для тангенса
ArcCot[x]	то же для котангенса
ArcSinh[x]	то же для гиперболического синуса
ArcCosh[x]	то же для гиперболического косинуса
ArcSech[x]	то же для гиперболического секанса
ArcCsch[x]	то же для гиперболического косеканса
ArcTanh[x]	то же для гиперболического тангенса
ArcCoth[x]	то же для гиперболического котангенса

Проиллюстрируем изученную информацию на примере вычисления выражения $3\sin(\pi/2)(1-2\cos\pi)$.

```
In[5]:= 3 Sin[Pi / 2] (1 - 2 * Cos[Pi])

Out[5]= 9
```

Большинство калькуляторов имеет некоторое количество ячеек памяти для запоминания вычисленных значений для последующего их использования. Как мы уже знаем, пакет *Mathematica* запоминает не толь-

ко результат вычисления выражений, но и сами выражения в специальных ячейках. К запомненным ячейкам можно обратиться в любое время, зная их номер. Проблема заключается в том, что номер определяется порядком выполненной операции, а он во время выполнения может измениться. Для решения данной проблемы существует два способа.

Первым способом можно воспользоваться, если все операции для решения задачи находятся в одной ячейке. В данном случае можно воспользоваться операцией **%**, которая означает, что вместо данного знака необходимо подставить результат предыдущей операции. Перепишем предыдущий пример с использованием данной операции.

```
In[10]:= Pi / 2
         Pi
         3 Sin[%%] (1 - 2 * Cos[%])

Out[10]=
          π
          2

Out[11]=
         π

Out[12]=
         9
```

Из приведенного примера видно, что результат вычисления каждого выражения запоминается в собственной ячейке **Out**, а также видно, что количество знаков % означает насколько необходимо вернуться назад, чтобы взять оттуда результат вычисления. Данным способом не всегда удобно пользоваться и на практике им пользуются достаточно редко.

Вторым способом является использование вспомогательных переменных. **Переменной является любая последовательность чисел, начинающаяся с буквы, которой присваивается результат вычисления или значение.** Удобством переменных является то, что ими можно воспользоваться в любой момент, после того как она была инициализирована, в любом месте вычислений. Следует помнить, что значения переменных, после того как они были просчитаны, хранятся только в течение текущего сеанса вычислений. После того как вы закроете программу, их значения при следующем запуске пакета автоматически не восстанавливаются. Для того чтобы их восстановить, необходимо запустить соответствующие ячейки на просчет.

Для иллюстрации использования переменных перепишем предыдущий пример, запомнив первый и второй углы, в переменные a1 и a2, соответственно.

```
In[13]:= a1 = Pi / 2
         a2 = Pi
         3 Sin[a1] (1 - 2 * Cos[a2])

Out[13]=
          π
          2

Out[14]=
         π

Out[15]=
         9
```

Такой подход более изящен и понятен.

Иногда результат промежуточных вычислений бывает не нужен и неплохо было бы иметь в своем распоряжении механизм, позволяющий подавлять вывод ненужных результатов. В пакете существует такой механизм. Для подавления вывода промежуточного результата достаточно поставить знак **точка с запятой** «**;**» в конце строки. Дополнительным преимуществом этого знака является то, что мы можем записывать несколько строк промежуточных вычислений, разделенных точками с запятыми, в одну строку. Перепишем предыдущий пример, подавив вывод результатов вычисления a1 и a2. В результате получим:

```
In[16]:= a1 = Pi / 2; a2 = Pi;
         3 Sin[a1] (1 - 2 * Cos[a2])

Out[17]=
         9
```

Как видно из данного примера, при подавлении вывода ненужных результатов мы можем получить более компактный и понятный вид и избавиться от вывода ненужной информации.

Кроме того, следует помнить, что все тригонометрические функции по умолчанию вычисляются в радианах. Для того чтобы вычислить значение какой-либо функции в градусах, необходимо воспользоваться

константой **Degree**[2] или ее символьным обозначением, которое может быть найдено на панели быстрого ввода. Для того чтобы показать это на примере, вычислим выражение $3\sin(\pi/2)(1-2\cos\pi)$, предварительно заменив углы в радианах на градусы. В результате получим:

```
In[22]:= a1 = 90 * Degree
         a2 = 180 °
         3 Sin[a1] (1 - 2 Cos[a2])

Out[22]=
         90 °

Out[23]=
         180 °

Out[24]=
         9
```

Легко догадаться, что мы можем перенести перевод из радиан в градусы в само выражение.

```
In[25]:= a1 = 90 ; a2 = 180;
         3 Sin[a1 * Degree] (1 - 2 Cos[a2 * °])

Out[26]=
         9
```

3.1. Вычисление выражений с заданной точностью

Так как пакет *Mathematica*, в основном, оптимизирован на проведение аналитических вычислений, то не все выражения могут по умолчанию вычисляться численно. Например, если мы попытаемся вычислить $\sin(3)$, то в результате получим следующее:

```
In[29]:= Sin[3]

Out[29]=
         Sin[3]
```

То есть получили не полностью вычисленное выражение. Для борьбы с данной проблемой существует два способа.

---

[2] Как уже было сказано ранее, Degree – это обычное константное выражение, численно равное количеству радиан в одном градусе. Поэтому для преобразования выражения, записанного в градусах, в радианы необходимо это выражение умножить на Degree.

Первым способом является представление числа, в данном случае тройки, в качестве числа с плавающей точкой (вещественного числа). Для этого достаточно после тройки поставить точку.

```
In[30]:= Sin[3.]

Out[30]=
         0.14112
```

Данный способ удобен, когда нам необходимо вычислить достаточно простое выражение.

Вторым способом является использование специальной функции, которая говорит пакету, что данное выражение необходимо вычислить численно. Эта функция называется **N** и имеет следующий формат записи **N[выражение, точность]**. Если точность не указывается, то вычисления проводятся с точностью по умолчанию в пять знаков после точки. Покажем пример использования данной функции на простом выражении $\sin(3)$, вычисленном с точностью по умолчанию:

```
In[32]:= N[Sin[3]]

Out[32]=
         0.14112
```

Данная запись может быть упрощена посредством постфиксного выполнения простых выражений, суть которой в следующем: если над каким-то выражением, перед выводом на экран, нужно выполнить функцию с одним аргументом (или функцию с несколькими аргументами, часть из которых можно не указывать), то можно воспользоваться записью вида: **выражение//функция**. Такая запись бывает очень удобна при вычислениях. Например, вычисление предыдущего выражения мы можем записать в виде:

```
In[31]:= Sin[3] // N

Out[31]=
         0.14112
```

Эта форма вычисления используется достаточно часто, так как позволяет не загромождать основное выражение.

Еще одним важным фактором, на который необходимо обратить внимание, является то, что *Mathematica*, например, в отличие от языков программирования, не имеет пределов по точности вычислений, точнее она ограничена только ресурсами компьютера, на котором проводятся

вычисления. Например, мы можем вычислить число π с точностью до 300-го знака.

```
In[9]:= N[Pi, 300]

Out[9]= 3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127
```

На этом мы закончим изложение возможности использования пакета *Mathematica* в качестве калькулятора и далее начнем рассматривать те возможности, которые, в основном, не имеются в калькуляторах, но присутствуют в данном пакете.

Задачи и упражнения

1. Придумать несколько простых выражений (2–4) и опробовать пакет в качестве простого калькулятора.
2. Придумать несколько выражений (2–3), содержащих математические и тригонометрические функции, и просчитать их. При работе с тригонометрическими функциями получить несколько значений в радианах и градусах.
3. Для одного из выражений предыдущей задачи получить ответ с разной точностью. Оцените время, затрачиваемое пакетом на вычисления, с помощью функции **Timing.** Описание данной функции посмотреть в справке.

4. НАХОЖДЕНИЕ КОРНЕЙ И РЕШЕНИЕ СИСТЕМ УРАВНЕНИЙ

4.1. Нахождение корней уравнений аналитически

Для нахождения корней уравнений в пакете *Mathematica* существует две функции:

Roots[уравнение, переменные];

Solve[уравнение, переменные].

Первая из них может решать только полиномиальные выражения, а вторая может решать более широкий спектр уравнений, в том числе некоторые трансцендентные уравнения и системы уравнений. Рассмотрим решение уравнений с помощью этих функций на примере уравнения $x^2+2bx+c=0$.

```
In[33]:= Roots[x^2 + 2 b x + c == 0, x]

Out[33]= x == -b - Sqrt[b^2 - c] || x == -b + Sqrt[b^2 - c]
```

```
In[34]:= Solve[x^2 + 2 b x + c == 0, x]

Out[34]= {{x -> -b - Sqrt[b^2 - c]}, {x -> -b + Sqrt[b^2 - c]}}
```

Как видно из данного примера, обе функции могут решить искомую задачу, а какую использовать для ее решения вам выбирать самим. Помимо этого, стоит обратить внимание, что при записи уравнения используется двойной знак «=», т. е. используется не присвоение, а проверка на равенство.

4.2. Решение систем линейных уравнений

Функция **Solve,** помимо решения простых линейных уравнений, может вычислять и более сложные задачи, например, решать системы линейных уравнений. Для этого решим следующую систему уравнений:

$$\begin{cases} x = 1 + 2ay \\ y = 9 + 2x \end{cases}$$

с использованием данной функции.

```
In[35]:= Solve[{x == 1 + 2 a y, y == 9 + 2 x}, {x, y}]

Out[35]= {{x -> -(1 + 18 a)/(-1 + 4 a), y -> -11/(-1 + 4 a)}}
```

Как видно из этого примера, в данном случае оба уравнения записываются вместе через запятую, но берутся в фигурные скобки.

4.3. Нахождение корней трансцендентных уравнений

Помимо этого, функцию **Solve[]** можно использовать при решении уравнений или систем уравнений, содержащих тригонометрические и показательные функции. Например, решим следующее уравнение: $-\arccos(3x) + 2\arcsin x = 0$.

```
In[4]:= Solve[-ArcCos[3 x] + 2 ArcSin[x] == 0, x]

Out[4]= {{x → 1/4 (-3 + Sqrt[17])}}
```

К сожалению, лишь малый класс трансцендентных уравнений и систем может быть решен аналитически. В большинстве случаев корни можно найти только приближенно с помощью функции **FindRoot[],** которая в общем виде записывается следующим образом:

FindRoot[уравнение, {x,x_0,x_1,…}],

где **x** обозначает, для какой переменной будут найдены корни, а $\mathbf{x_i}$ – указывают на начальные приближения к корню.

Для того чтобы понять, как пользоваться этой функцией, найдем корни следующего уравнения:

$$exp(x) = x^3.$$

В результате решения данного уравнения получим следующий результат:

```
In[1]:= FindRoot[Exp[x] == x^3, {x, 0, 4}]

Out[1]= {x → 1.85718}
```

К сожалению, недостатком данной функции является то, что за один раз можно найти только один корень уравнения. Для того чтобы найти другой корень, необходимо задать другие приближения, например:

```
In[2]:= FindRoot[Exp[x] == x^3, {x, 5}]

Out[2]= {x → 4.5364}
```

С помощью данной функции можно находить численное решение системы трансцендентных уравнений. Рассмотрим решение систем трансцендентных уравнений на примере системы функций:

$$\begin{cases} x^2 + y^2 = 1 \\ y = x^2 \exp(x) \end{cases}.$$

В результате решения данной системы будет получен следующий результат:

```
In[3]:= FindRoot[{x^2 + y^2 == 1, y == x^2 Exp[x]},
          {x, 0.5}, {y, 1}]

Out[3]= {x → 0.637907, y → 0.770113}
```

Аналогичным образом можно решать другие более сложные уравнения и системы уравнений с несколькими уравнениями.

4.4. Численное решение уравнений

Из-за аналитической направленности пакета *Mathematica* многие вычисления получаются в аналитическом или полуаналитическом виде, когда результат выражается через дроби. Очень часто бывает необходимо заставить пакет выдать результат в виде числа. В данном случае мы можем воспользоваться функцией **N**, как было рассказано ранее, или воспользоваться функцией **NSolve**, которая специально предназначена для численного решения как уравнений, так и систем уравнений. Например, получим результат нахождения корней уравнения $-\arccos(3x) + 2\arcsin x = 0$ с помощью функции **NSolve[].**

```
In[7]:= NSolve[-ArcCos[3 x] + 2 ArcSin[x] == 0, x]

Out[7]= {{x → 0.280776}}
```

Для более подробной информации можно посмотреть справку по данным функциям в пакете *Mathematica* или в литературе.

Задачи и упражнения

1. Придумать линейное уравнение и решить его с помощью функций **Roots** и **Solve**.
2. Придумать уравнение, содержащее тригонометрические функции, и получить его решение.
3. Придумать систему линейных уравнений с тремя неизвестными и получить его решение. Оценить время выполнения.

5. РАСШИРЕННАЯ РАБОТА С ЯЧЕЙКАМИ

Как мы уже знаем, исходные данные для вычисления (выражения, которые необходимо вычислить) и результаты расчета располагаются в разных ячейках, а обе ячейки являются единым целым, относящимся к одной группе вычислений. Таких групп в одном документе может быть множество. Также известно, что в каждой группе вычислений ячейка с исходными данными может быть одна, а результатов – несколько. Количество ячеек с результатами зависит от количества выражений, которые вычисляются в ячейке с исходными данными. Пояснение этих слов показано на рис. 2.

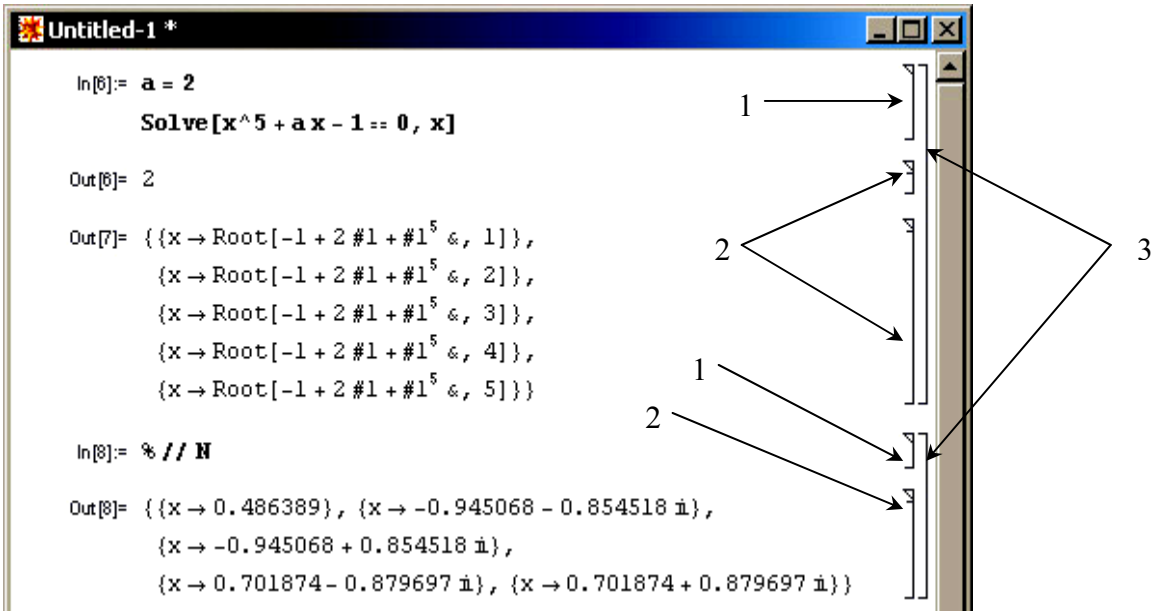


Рис. 2. Деление ячеек на группы: 1 – ячейки с исходными данными; 2 – ячейки с результатами; 3 – группа вычислений

При такой организации пользователь сам решает, как нужно разбивать вычисления на блоки. В принципе можно задать все вычисления и в одной ячейке с исходными результатами, но в данном случае затрудняется интерпретация вычисленных результатов, так как все результаты вычислений будут помещены в отдельные блоки и пользователю будет труднее идентифицировать, к какому выражению относится каждый результат. Задача, разбитая на несколько групп вычислений, чаще всего понимается лучше. Правда, в данном случае возникает проблема с тем, что пользователю придется каждый раз пересчитывать все группы вычислений при повторном вычислении, например при изменении некоторых параметров. Эту проблему можно решить, если воспользоваться

пунктом меню ***Kernel=> Evaluation=> Evaluate Notebook***, который запускает на выполнение последовательно все ячейки с исходными данными.

Как мы уже знаем, посредством знака «;» в конце выражения мы можем подавить вывод результата того или иного выражения. Иногда этого оказывается недостаточно. Пакет *Mathematica* позволяет скрывать выходные ячейки для целой группы вычислений. Для этого достаточно кликнуть два раза левой кнопкой мыши на скобку 3 (см. рис. 1). Например, если мы захотим спрятать промежуточные вычисления в предыдущем примере, т. е. спрятать результат вычисления первой группы вычислений, то после манипуляций мышью мы получим следующий вид документа:

```
In[6]:= a = 2
        Solve[x^5 + a x - 1 == 0, x]

In[8]:= % // N

Out[8]= {{x → 0.486389}, {x → -0.945068 - 0.854518 i},
         {x → -0.945068 + 0.854518 i},
         {x → 0.701874 - 0.879697 i},
         {x → 0.701874 + 0.879697 i}}
```

Как видно из данного примера, результаты вычислений первого блока вычислений спрятались и скобка, ограничивающая данный блок, приняла другой вид. Такой вид скобки говорит пользователю, что в данном блоке результаты вычислений спрятаны. Для того чтобы заставить систему заново показать результаты, достаточно кликнуть два раза левой кнопкой мыши на скобку еще раз.

Следует заметить, что ячейки с исходными данными являются редактируемыми, т. е. вы можете в любое время изменить исходные данные и пересчитать их. Ячейки с результатами вычислений по умолчанию изменению не подлежат. При попытке изменения ячейки с результатом ее содержимое будет скопировано во вновь созданную ячейку, которая уже будет интерпретироваться как исходные данные для нового блока.

При вводе выражений могут произойти ошибки или возникнет необходимость изменить или скопировать часть выражения в какой-либо входной ячейке. В этом случае следует прибегнуть к редактированию. Для входных ячеек это можно сделать стандартными приемами, принятыми в текстовых редакторах, работающих под *Windows*, т. е. с помощью *Clipboard* и команд **Cut, Copy, Paste**, расположенных в меню **Edit**.

Удалить, скопировать, вставить можно не только любую часть содержимого входной ячейки, но и всю ячейку в целом. Для этого ее следует выделить, подведя указатель мыши справа к скобке ячейки и кликнув левой кнопкой мыши. Скобка будет выделена черным цветом. После этого можно воспользоваться описанными ранее командами для редактирования всего содержимого ячейки. С помощью буфера обмена ячейку или ее часть можно вставить в любой другой документ, например Microsoft Word. Но при этом следует помнить, что скопированные таким образом данные могут некорректно отображаться на тех компьютерах, на которых не установлен рассматриваемый пакет.

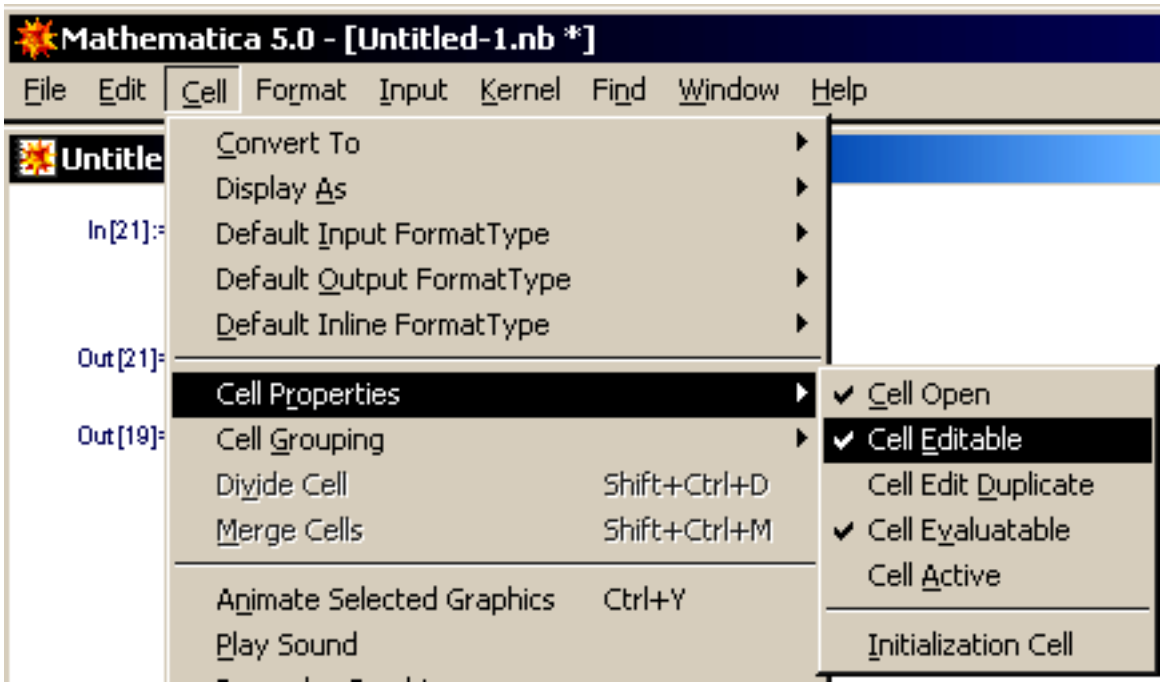
Иногда бывает необходимо добавить некоторые блоки между уже созданными блоками. Для того чтобы это сделать, необходимо поместить курсор мыши между блоками. При этом курсор сменит свой вид с вертикального на горизонтальный. Затем кликнуть левой кнопкой мыши, и тогда в этом месте появится горизонтальная черта приглашения ввода, как показано на следующем примере:

```
In[1]:= a = 2
        Solve[x^5 + a x - 1 == 0, x]

Out[1]= 2

Out[2]= {{x → Root[-1 + 2 #1 + #1^5 &, 1]},
         {x → Root[-1 + 2 #1 + #1^5 &, 2]},
         {x → Root[-1 + 2 #1 + #1^5 &, 3]},
         {x → Root[-1 + 2 #1 + #1^5 &, 4]},
         {x → Root[-1 + 2 #1 + #1^5 &, 5]}}
```

```
In[3]:= % // N

Out[3]= {{x → 0.486389},
         {x → -0.945068 - 0.854518 i},
         {x → -0.945068 + 0.854518 i},
         {x → 0.701874 - 0.879697 i},
         {x → 0.701874 + 0.879697 i}}
```

Если после этих манипуляций начать набирать текст, то в этом месте появится новая ячейка для ввода информации.

В пакете *Mathematica* есть также возможность задавать различные стили оформления документа. Можно менять как оформление одной ячейки, так и стиль всего документа. Для изменения оформления предназначены пункты из меню ***Cell*** и ***Format***. Например, если мы захотим запретить изменение информации в какой-нибудь отдельно взятой ячейке, то мы можем изменить свойства данной ячейки в меню ***Cell=>Cell Properties***. В данном случае мы должны убрать галочку в пункте ***Cell Editable***, как показано на следующем примере:



Этот булевый пункт меню указывает на то, редактируемый данный блок или нет. После снятия данного флага скобка блока станет другой, указывающей на то, что изменение данного блока невозможно.

```
In[4]:= a = 2

        Solve[x^5 + a x - 1 == 0, x]
```

Кратко рассмотрим другие пункты данного меню. Флаг ***Cell Open*** указывает на то, показывать или спрятать данную ячейку. ***Cell Edit Duplicate*** указывает на то, что при попытке изменения данной ячейки необходимо создать ее копию и там продолжить изменения. Данный флаг установлен, по умолчанию, для ячеек с результатами вычислений. Флаг ***Cell Evaluatable*** устанавливается для ячеек, для которых возможен запуск на вычисление (по умолчанию установлен для ячеек с исходными данными). ***Cell Active*** указывает на то, содержит или не содержит ячейка

активные элементы, например кнопки. Последний из пунктов данного меню (***Initialization Cell)*** бывает иногда очень полезен. При установке данного флага ячейка считается инициализирующей для всего документа и выполняется сразу при открытии документа.

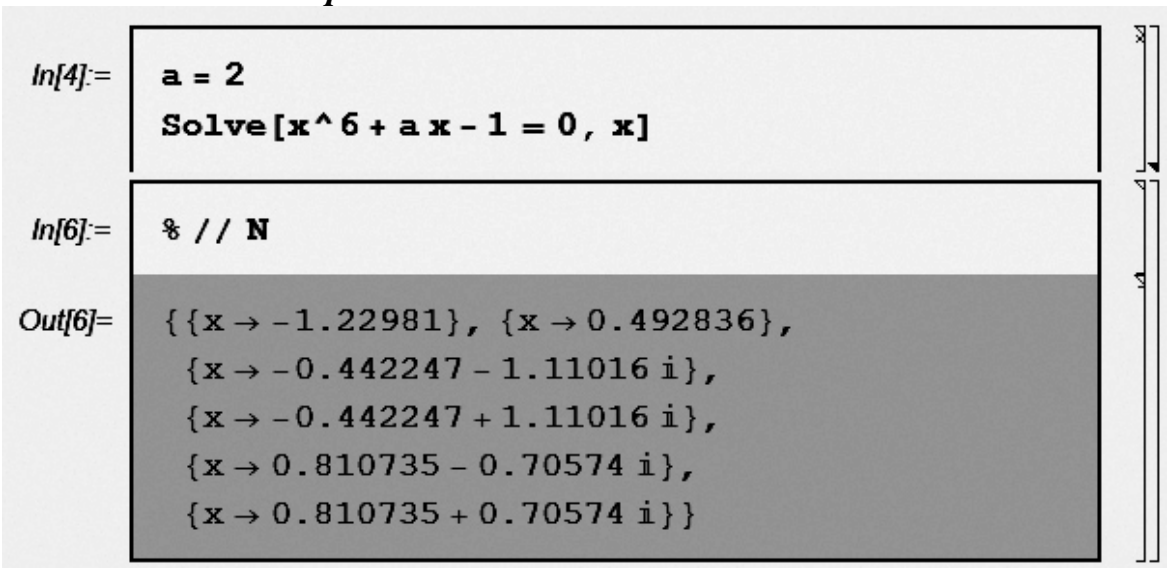
В меню ***Format*** находятся пункты для изменения стиля блока или всего документа. Например, можно менять размер шрифта, цвет текста, тип текста, тип фона и мн. др. Рассмотрим лишь несколько пунктов меню, так как объем данного пособия не позволяет описать все возможности форматирования.

Первое подменю (***Style***), отвечающее за стиль ячейки, будет рассмотрено более подробно чуть позже.

Пункты меню ***Screen Style Environment*** и ***Printing Style Environment*** отвечают за вид отображения текущего документа на экране и при печати. Подпункты этого меню позволяют установить некоторые заданные стили при отображении документа, которые влияют на общее форматирование и размер шрифта. Например, существуют рабочий, презентационный и компактный стили отображения содержимого.

При включении флага ***Show Expression*** из меню ***Format*** текущая ячейка выводится во внутреннем формате, т. е. она записывается в реализации самого документа.

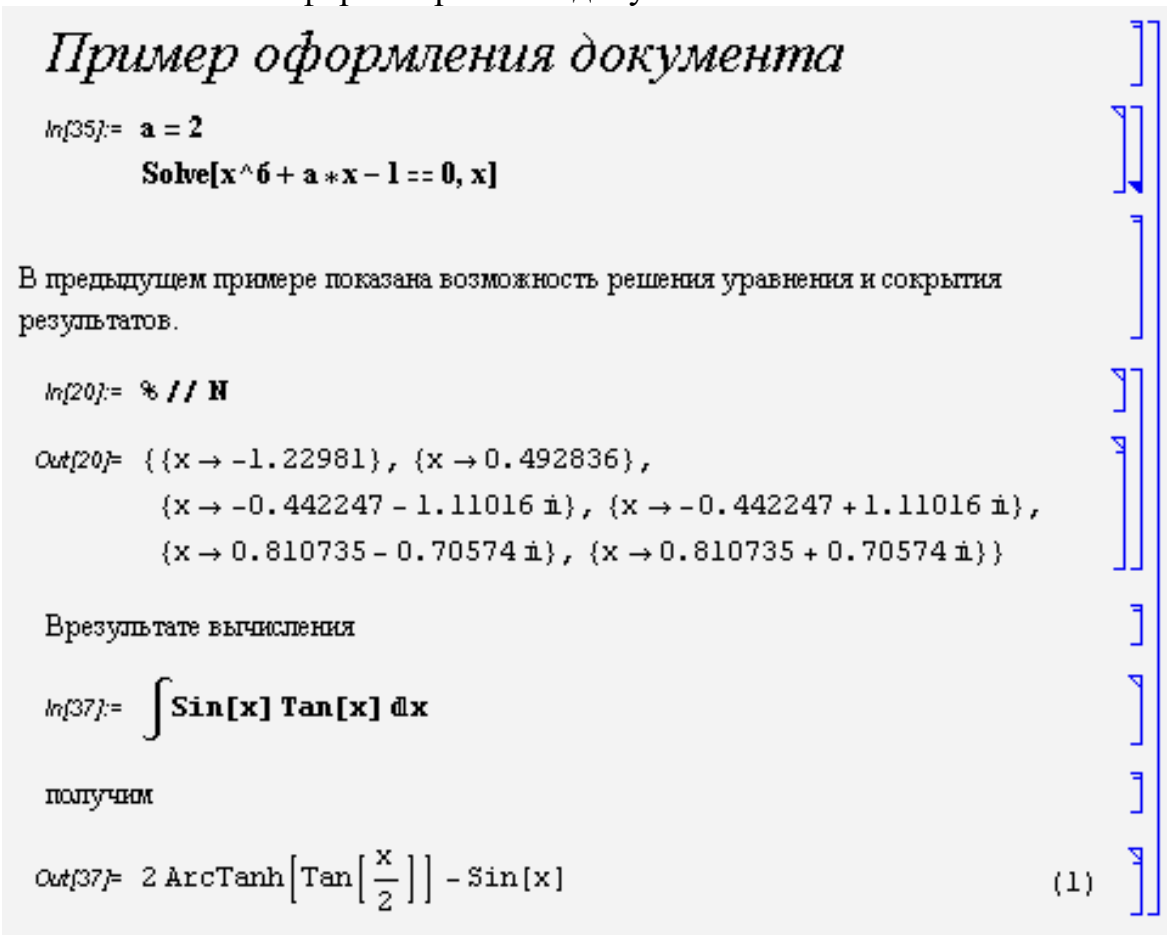
Пункт меню ***Style Sheet*** позволяет выбрать общий стиль документа из одного из предустановленных. На следующем примере показан установленный стиль ***Report***.



Пакет *Mathematica* позволяет задавать собственные стили оформления или переопределять существующие для текущего документа в пункте меню ***Edit Style Sheet***.

Остальные пункты меню позволяют устанавливать тип шрифта, размер букв, цвет текста и фона, выравнивание и другие параметры и могут быть легко изучены пользователем.

Теперь рассмотрим более подробно меню ***Format=>Style***. С использованием данного пункта мы можем устанавливать для ячеек стили, отличные от двух известных нам стилей (стиля ввода (***Input***) и вывода (***Output)***), используемых для ввода выражений и вывода результатов, соответственно. Остальные стили позволяют создавать в пакете *Mathematica* полноценные текстовые документы. В данном пункте меню определены, например следующие стили: заголовок документа (***Title***), подзаголовок документа (***Subtitle***), заголовок (***Section***), подзаголовок (***Subsection***), обычный текст (***Text***), комментарий (***Commentary***), нумерованная формула (***Numbered Equation***). На следующем примере показана возможность форматирования документа.



Для более подробной информации по данной теме можно обратиться к справке пакета *Mathematica* или к литературе.

6. ОСНОВНЫЕ ТИПЫ ДАННЫХ ПАКЕТА MATHEMATICA

Пакет *Mathematica* поддерживает все простые основные типы аналогично многим языкам программирования, а именно несколько типов чисел, булевы переменные и строки.

Числа в пакете *Mathematica* бывают четырех типов: целые, рациональные, вещественные и комплексные. Все типы чисел могут содержать любое количество цифр. Чтобы число рассматривалось как вещественное, оно должно содержать точку в его записи, даже в случае, если имеет нулевую мантиссу (дробную часть). По умолчанию пакет сам определяет тип выражения, но это поведение можно переопределить. При вычислении *Mathematica* сохраняет, если это возможно, тип чисел. Например, при вычислении синуса от 2, представленного выражением ***Sin[2]***, после вычисления будет также записан в виде ***Sin[2]***. Для того чтобы получить приближенное значение этого выражения, следует его вычислить для вещественного числа 2, представленного как (2.) или как 2.0.

Булевы выражения получаются при сравнении выражений. Результатом таких выражений может быть ***True*** или ***False***.

Строки – это заключенные в кавычки последовательности букв, цифр и специальных символов: **“Это строка”**. Если внутри строки используются кавычки, то их следует представить в виде последовательности **\"**. Строки также могут содержать последовательности: **\n** – перехода на новую строчку, ***\t*** – табуляции и $\backslash\mathbf{n_1n_2n_3}$, где $\mathbf{n_1n_2n_3}$ – восьмеричный код ASCII.

Еще одним типом, не встречающимся в большинстве языков программирования, можно считать выражения. Выражения – это строки описывающие порядок вычисления, незаключенные в кавычки.

Все перечисленные типы могут храниться в переменных, а затем использоваться и, следовательно, мы можем получить результат вычисления в одном из этих типов.

7. СОЗДАНИЕ СПИСКОВ

Список является фундаментальным способом структурирования данных и очень часто применяется при вычислениях в пакете *Mathematica*. Он может задаваться с помощью функции ***List***. Например:

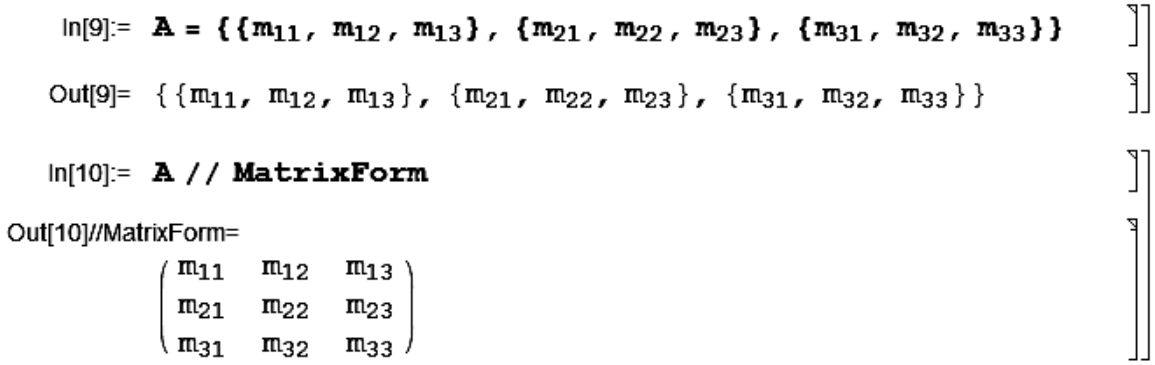
```
In[7]:= List[1, 2, 3, a + b, Sin[x]]

Out[7]= {1, 2, 3, 2 + b, Sin[x]}
```

Как видно из данного примера, элементами списка могут быть любые выражения, использование которых возможно в пакете *Mathematica*. Так же виден еще один способ задания списков, при котором мы можем просто заключить элементы, разделенные запятыми, в фигурные скобки. Вывод по умолчанию можно представить в более привычном стиле, например, с помощью функции ***TableForm***. Для примера перепишем предыдущий пример с применением данной функции.

```
In[7]:= List[1, 2, 3, a + b, Sin[x]]

Out[7]= {1, 2, 3, 2 + b, Sin[x]}

In[8]:= % // TableForm

Out[8]//TableForm=
1
2
3
2 + b
Sin[x]
```

Как видно из данного примера ***TableForm*** выводит результат вычисления в виде таблицы. При использовании данной функции следует помнить, что ее стоит использовать только для вывода результатов и ни в коем случае не стоит запоминать вывод этой функции в переменную для последующего использования в вычислениях, так как большинство операций не работают с результатами в табличном или матричном виде.

Элементами списка могут быть и другие списки. Наиболее часто встречающаяся конструкция из списка списков – матрицы.



Как видно из данного примера, функция ***MatrixForm*** представляет вывод списка в матричной форме.

Помимо ручного задания списков, существуют функции, создающие списки по определенным правилам. Приведем основные из них в табл. 4.

Таблица 4

Основные функции для задания списков

Функция	**Описание**
Range[imin,imax,di]	Генерирует список, начиная с imin до imax, с шагом di. Если шаг не указан, то по умолчанию он равен единице. Если еще не указано imin, т. е. указана одна переменная, то генерируется список из imax элементов, начинающихся с единицы.
Table[выр.,{i,imin,imax,di}]	Генерирует список по выражению, зависящему от i = imin…imax, с шагом равном di. Если шаг не указан, то по умолчанию он равен 1, а если еще не указан imin, то i = 1…imax
Table[выр.,{i,imin,imax,di}, {j,jmin,jmax,dj},…]	То же самое, но для многомерных списков
Array[f,n]	Генерирует список длиною n из элементов f[i]. Если второй аргумент записать как список из нескольких элементов, то будет создан список соответствующей элементам размерности. Если же добавить третий аргумент, то он будет задавать, с какого числа начинать нумерацию элементов.
DiagonalMatrix[list]	Генерирует двумерную диагональную матрицу, где элементы из списка list располагаются на главной диагонали.

Поясним работу данных функций на следующих примерах.

```
In[11]:= Range[5, 12]

Out[11]= {5, 6, 7, 8, 9, 10, 11, 12}

In[12]:= Range[5, 12, 3]

Out[12]= {5, 8, 11}

In[13]:= Table[i^2, {i, 10}]

Out[13]= {1, 4, 9, 16, 25, 36, 49, 64, 81, 100}

In[14]:= Table[j^3, {j, -3, 3}]

Out[14]= {-27, -8, -1, 0, 1, 8, 27}

In[15]:= Table[x^(i + j), {i, 1, 3}, {j, 1, i}] //
          TableForm

Out[15]//TableForm=
          x^2
          x^3       x^4
          x^4       x^5       x^6
```

Еще одним способом задания списков может быть импорт данных из файла с помощью функции ***Import["имя файла","формат чтения"]***. При использовании данной функции следует помнить, что при считывании данных из файла, в первом аргументе функции ***Import***, нужно указывать полный путь до файла. Кроме того, следует учитывать, что в пакете *Mathematica* при записи пути используется форма записи, принятая в Unix-подобных операционных системах. Это означает, что при записи пути, вместо обратной косой черты "\", принятой в Windows, необходимо использовать прямую косую черту "/". Например, если файл находится на диске **U** в папке **test** и называется **input.dat**, то для записи пути необходимо использовать следующую запись **"U:/test/input.dat"**. Особо нужно обратить внимание на то, что в пути нельзя использовать русские буквы! Второй параметр данной функции указывает на формат чтения. Форматов чтения существует несколько, но для численных данных главными являются ***List*** и ***Table***. Первый из них считывает данные в виде списка (одномерного массива), а второй – в виде таблицы (матрицы). Например, при выполнении следующей записи:

```
In[3]:= data = Import["E:/work/input.txt", "List"]

Out[3]= {1.44755, -4.60692, 24.9079, 3.23948, -25.6717,
          -8.83294, -23.2626, 23.0333, 13.8353, 23.7639}
```

переменная **data** будет содержать содержимое файла **input.txt**, расположенного в директории **work** на диске **E:**. Данные будут считаны (импортированы) из файла в виде списка.

Более подробно о работе данной функции можно посмотреть в справке пакета *Mathematica* или в литературе.

8. РАБОТА С ВЕКТОРАМИ И МАТРИЦАМИ

Вектора в пакете *Mathematica* трактуются как линейные, т. е. одноуровневые, списки: ***v*={ul,u2,u3};** матрицы как двухуровневые: ***mn* = {{mll,ml2,ml3},{m21,m22,m23}}**, хотя компоненты векторов и матриц могут быть произвольными выражениями. Другими словами можно сказать, что матрица является списком списков. Ввиду важной роли этих объектов в теоретических и прикладных вопросах математики, в пакете определены несколько функций, предназначенных для работы с векторами и матрицами.

Функция ***Det*** вычисляет детерминант квадратной матрицы.

```
In[16]:= m1 = {{2, 3, 2, 3}, {1, 8, 5, 3}, {3, 5, 2, 1},
            {4, 8, 1, 3}};
         Det[m1]

Out[17]= -140
```

Выражение ***Minors[matrix,k]*** имеет результатом список миноров k-го порядка матрицы *matrix.*

```
In[18]:= Minors[m1, 2]

Out[18]= {{13, 8, 3, -1, -15, -9},
          {1, -2, -7, -4, -12, -4}, {4, -6, -6, -13, -15, 3},
          {-19, -13, -8, -9, -7, -1},
          {-24, -19, -9, -32, 0, 12}, {4, -5, 5, -11, 7, 5}}
```

Функция ***Inverse*** вычисляет обратную матрицу для невырожденных квадратных матриц.

```
In[20]:= Inverse[m1] // MatrixForm

Out[20]//MatrixForm=
```

$$\begin{pmatrix} \frac{1}{5} & -\frac{8}{35} & \frac{3}{7} & -\frac{4}{35} \\ -\frac{1}{4} & \frac{3}{28} & -\frac{3}{28} & \frac{5}{28} \\ \frac{3}{20} & \frac{11}{140} & \frac{9}{28} & -\frac{47}{140} \\ \frac{7}{20} & -\frac{1}{140} & -\frac{11}{28} & \frac{17}{140} \end{pmatrix}$$

Для транспонирования матрицы используется функция ***Transpose***. На следующем примере показан результат транспонирования матрицы **m1**.

```
In[21]:= m1 = {{2, 3, 2, 3}, {1, 8, 5, 3}, {3, 5, 2, 1},
             {4, 8, 1, 3}};
         m1 // MatrixForm
```

Out[22]//MatrixForm=

$$\begin{pmatrix} 2 & 3 & 2 & 3 \\ 1 & 8 & 5 & 3 \\ 3 & 5 & 2 & 1 \\ 4 & 8 & 1 & 3 \end{pmatrix}$$

```
In[23]:= Transpose[m1] // MatrixForm
```

Out[23]//MatrixForm=

$$\begin{pmatrix} 2 & 1 & 3 & 4 \\ 3 & 8 & 5 & 8 \\ 2 & 5 & 2 & 1 \\ 3 & 3 & 1 & 3 \end{pmatrix}$$

Скалярное произведение векторов, произведение вектора и матрицы, а также произведение матриц вычисляются с помощью функции ***Dot[s1,s2]***. Возможна также упрощенная запись **s1.s2** для вычисления произведения.

Приведем несколько примеров употребления функции ***Dot***.

```
In[1]:= v1 = {1, 0, 0}; v2 = {a, b, c};
        v1.v2

Out[2]= a

In[3]:= m1 = {{2, 3, 2, 3}, {1, 8, 5, 3}, {3, 5, 2, 1},
            {4, 8, 1, 3}};
        {2, 3, 5, 1}.m1

Out[4]= {26, 63, 30, 23}

In[5]:= {2, 3, 5, 1}.Inverse[m1]

Out[5]= {3/4, 1/4, 7/4, -5/4}
```

Для векторов также определена операция векторного произведения ***Cross***.

```
In[17]:= Cross[{0, 0, 1}, {1, 0, 0}]

Out[17]=
         {0, 1, 0}
```

С помощью функции ***MatrixPower*** можно возводить матрицы в целую положительную степень. Например, возведем заданную выше матрицу **m1** в квадрат. В результате получим:

```
In[3]:= MatrixPower[m1, 2]

Out[3]= {{25, 64, 26, 26}, {37, 116, 55, 41},
         {21, 67, 36, 29}, {31, 105, 53, 46}}
```

Собственные числа и собственные векторы матриц можно найти, используя функции ***Eigenvalues*** и ***Eigenvectors***. С помощью функции ***Eigensystem*** можно получить и собственные числа, и функции матриц одновременно. Функция ***CharacteristicPolynomial*** выводит характеристический полином матрицы:

```
In[4]:= Eigenvalues[m1] // N

Out[4]= {14.476, -2.89874,
         1.71137 + 0.6384 i, 1.71137 - 0.6384 i}
```

```
In[5]:= Eigenvectors[m1] // N

Out[5]= {{0.597757, 1.05479, 0.646621, 1.},
         {-0.56396, -0.542193, 0.694636, 1.},
         {0.908636 - 1.76941 i, -0.59336 + 1.10318 i,
          -0.176296 - 1.10937 i, 1.},
         {0.908636 + 1.76941 i, -0.59336 - 1.10318 i,
          -0.176296 + 1.10937 i, 1.}}
```

```
In[6]:= Eigensystem[m1] // N

Out[6]= {{14.476, -2.89874,
          1.71137 + 0.6384 i, 1.71137 - 0.6384 i},
         {{0.597757, 1.05479, 0.646621, 1.},
          {-0.56396, -0.542193, 0.694636, 1.},
          {0.908636 - 1.76941 i, -0.59336 + 1.10318 i,
           -0.176296 - 1.10937 i, 1.},
          {0.908636 + 1.76941 i, -0.59336 - 1.10318 i,
           -0.176296 + 1.10937 i, 1.}}}
```

```
In[9]:= m2 = {{1, x1, x1^2}, {1, x2, x2^2},
           {1, x3, x3^2}};
        CharacteristicPolynomial[m2, x]
```

Out[10]=

$$x^2 - x^3 + x\,x1 + x\,x1^2 - x\,x2 + x^2\,x2 - x1^2\,x2 + x1\,x2^2 + x1^2\,x3 - x2^2\,x3 + x\,x2^2\,x3 - x\,x3^2 + x^2\,x3^2 - x1\,x3^2 + x2\,x3^2 - x\,x2\,x3^2$$

Сумму и разность векторов и матриц можно получить с помощью обычных операций сложения (+) и вычитания (–), соответственно. Например, вычислим сумму двух квадратных матриц (**a** и **b**) размерности 2. В результате получим:

```
In[14]:= a = {{1, 2}, {3, 4}}; b = {{2, 3}, {4, 5}};
         a + b // MatrixForm
         a - b // MatrixForm
```

Out[15]//MatrixForm=

$$\begin{pmatrix} 3 & 5 \\ 7 & 9 \end{pmatrix}$$

Out[16]//MatrixForm=

$$\begin{pmatrix} -1 & -1 \\ -1 & -1 \end{pmatrix}$$

Начиная с версии 5 пакета *Mathematica*, для списков определена функция ***Norm***, вычисляющая норму списка.

```
In[19]:= Norm[{a, b, c}]
```

Out[19]=

$$\sqrt{\text{Abs[a]}^2 + \text{Abs[b]}^2 + \text{Abs[c]}^2}$$

Как видно из данного примера, для векторов нормой является модуль вектора.

Задачи и упражнения

1. Для векторов $\vec{a} = \{1,3,5\}; \vec{b} = \{2,3,8\}; \vec{c} = \{-1,3,-3\}$ найти:
   a) $(\vec{a}\cdot\vec{b}),(\vec{a}\cdot\vec{c}),(\vec{c}\cdot\vec{b})$;
   b) $\vec{a}\vec{b}\vec{c},\vec{c}\vec{b}\vec{a},\vec{a}\vec{c}\vec{b}$;
   c) $[\vec{a}\times\vec{c}],[\vec{b}\times\vec{c}],[\vec{a}\times\vec{b}]$;
   d) найти высоту пирамиды, построенной на векторах $\vec{a},\vec{b}$ и $\vec{c}$.

2. Для матрицы

$$\begin{pmatrix} 2 & 3 & 2 & 5 \\ 1 & 2 & -3 & 3 \\ 3 & 4 & 1 & -3 \\ 16 & 25 & 81 & 2 \end{pmatrix}$$

вычислить:

a) определитель матрицы;
b) транспонированную матрицу;
c) обратную матрицу (при вычислении проверьте результат, используя свойства обратной матрицы);
d) возвести матрицу в 3-ю степень;
e) произвести сложение, вычитание и умножение матрицы с диагональной матрицей, полученной из списка $\{1,-4,3,2\}$.

3. Создать матрицу по выражению $\sin(x+y)$, где $0 \le x \le 6$, с шагом 0.1, а $0 \le y \le 3$, с шагом 1.5. На основании полученной таблицы построить график, воспользовавшись функцией ***ListPlot3D***.

9. РАСШИРЕННАЯ РАБОТА СО СПИСКАМИ

Рассмотренные в предыдущих главах возможности работы пакета *Mathematica* со списками основаны, в основном, на математической обработке таких объектов, как вектора и матрицы. Но в данном пакете существует достаточно большой набор функций, предназначенных для выполнения различных операций над списками. Их можно разделить на насколько групп:

1) выделение элементов списков;
2) выявление структуры списков;
3) операции над списками;
4) изменение структуры списков.

Рассмотрим каждую из групп функций более подробно.

9.1. Выделение элементов списков

Функции из данной группы позволяют выделять один или несколько элементов из списка.

Запись вида ***Part[сисок,i]***, или ее более короткая форма – ***список[[i]]***, позволяет выделить i-й элемент списка, например при выполнении следующего блока

```
In[1]:= li = {4, 5, 3, 6, 2, 7, 2, 8, 9};
        li[[5]]

Out[2]= 2
```

получим вывод 5-го элемента списка, т. е. в данном случае мы можем сказать, что был выделен элемент из списка (массива) ***li*** с индексом 5. Похожая запись может быть встречена в большинстве языков программирования. Но благодаря более свободной записи в пакете *Mathematica* возможно производить с помощью данной операции более сложные выборки из списка. Например, если для того же массива ***li*** выражение внутри двойных квадратных скобок записать в виде списка, как показано на следующем примере

```
In[3]:= li[[{1, 3, 5, 7}]]

Out[3]= {4, 3, 2, 2}
```

то в результате получим новый список, составленный из элементов массива ***li***, с индексами, заданными в качестве параметра данной операции.

Эта же операция может применяться и для выделения необходимых элементов многомерных массивов (матриц), например

```
In[7]:= matr = {{4, 5, 3, 6}, {2, 7, 2, 8}};
        matr // MatrixForm
        matr[[2, 2]]

Out[8]//MatrixForm=
        ( 4  5  3  6 )
        ( 2  7  2  8 )

Out[9]= 7
```

В результате выполнения данных операций будет выведен элемент списка с индексами ***2,2***. Данную операцию можно получить и для выделения определенной строки из данного двумерного списка. Например, для выделения первой строки можно воспользоваться записью

```
In[11]:= matr[[1]]

Out[11]=
         {4, 5, 3, 6}
```

Аналогично приведенному ранее примеру с выбором нескольких элементов из массива, можно выбирать несколько элементов или строк из матрицы и сгруппировать на их основе новые списки или матрицы.

Для получения первого элемента списка можно воспользоваться функцией ***First[список]***. Например, если использовать данную функцию на списке ***li***, то в результате получим

```
In[14]:= First[li]

Out[14]=
         4
```

В противоположность ***First[]***, функция ***Last[список]*** возвращает последний элемент списка:

```
In[5]:= Last[li]

Out[5]= 9
```

Функция ***Extract[список, список позиций]*** работает аналогично ***Part*** и позволяет выделять элементы списка

```
In[6]:= matr = {{4, 5, 3, 6}, {2, 7, 2, 8}};
        Extract[matr, {2, 3}]

Out[7]= 2
```

Отличие заключается в том, что в данной функции второй аргумент должен быть оформлен в виде списка (или матрицы-столбца при выделении нескольких элементов).

Иногда бывает необходимо выделить несколько первых или последних элементов из списка. Для решения этой задачи можно воспользоваться функцией ***Take[список,n]***, которая возвращает в качестве результата первые ***n*** элементов, если ***n*** **положительно**, и ***n*** последних элементов, если оно **отрицательно**, например:

```
In[1]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};

In[2]:= Take[l, 2]

Out[2]= {4, 5}

In[3]:= Take[l, -2]

Out[3]= {8, 9}
```

Данную функцию можно использовать и для того, чтобы выделять часть элементов матрицы и составить на ее основе другую, например:

```
In[1]:= matr1 = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
        matr1 // MatrixForm
```

Out[2]//MatrixForm=

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

```
In[3]:= Take[matr1, -2, 2] // MatrixForm
```

Out[3]//MatrixForm=

$$\begin{pmatrix} 4 & 5 \\ 7 & 8 \end{pmatrix}$$

Функция ***Drop[список,n]*** возвращает в качестве результата список, в котором отброшены ***n*** элементов с начала или конца, в зависимости от значка числа **n**.

```
In[4]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
        Drop[l, 2]
        Drop[l, -2]

Out[5]= {3, 6, 2, 7, 2, 8, 9}

Out[6]= {4, 5, 3, 6, 2, 7, 2}
```

Если в данной функции второй параметр записать в списковом формате, то будет удален только один элемент, стоящий на ***i***-й позиции.

```
In[1]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
        Drop[l, {2}]

Out[2]= {4, 3, 6, 2, 7, 2, 8, 9}
```

Для удаления нескольких элементов, с номера ***a*** до номера ***b*** из списка, можно воспользоваться записью ***Drop[список,{a,b}]***, как показано на следующем примере:

```
In[3]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
        Drop[l, {2, 4}]

Out[4]= {4, 2, 7, 2, 8, 9}
```

В результате будут удалены элементы со второго по четвертый.

Этой же функцией можно воспользоваться для удаления некоторых столбцов (строк) из матриц. Например, в матрице

```
In[7]:= m = Table[10 i + j, {i, 3}, {j, 4}];
        m // MatrixForm

Out[8]//MatrixForm=
```

$$\begin{pmatrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \end{pmatrix}$$

отбросим первую строку, для этого выполним следующую команду:

```
In[10]:= Drop[m, 1] // MatrixForm

Out[10]//MatrixForm=
```

$$\begin{pmatrix} 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \end{pmatrix}$$

Затем отбросим четвертый столбец матрицы.

```
In[12]:= Drop[m, {}, {4}] // MatrixForm

Out[12]//MatrixForm=
```

$$\begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix}$$

И наконец, первый и четвертый столбцы:

```
In[13]:= Drop[m, {1}, {4}] // MatrixForm

Out[13]//MatrixForm=
```

$$\begin{pmatrix} 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix}$$

С помощью данной функции можно удалять также несколько строк и/или столбцов в определенном диапазоне, аналогично тому, как это делалось в случае со списками.

Функция ***Select[список, выражение]*** возвращает те элементы списка, для которых выражение второго аргумента принимает значение ***True.*** Например, если мы хотим выделить все элементы (больше 5) из списка ***l***, мы можем записать

```
In[3]:= Select[l, # > 5 &]

Out[3]= {6, 7, 8, 9}
```

Более подробную информацию по этим и другим функциям данного раздела вы можете получить в книгах или справке по пакету в разделе ***Lists and Matrices => Element Extraction***.

9.2. Выявление структуры списков

Для выявления структуры списков существует достаточно большой набор функций. Рассмотрим лишь основные из них.

Функция ***Length[список]*** возвращает количество элементов в списке, как показано на следующем примере:

```
In[4]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
        Length[l]

Out[5]= 9
```

Если использовать данную функцию на матрице, то в результате получим количество строк, так как матрица представляется в виде списка списков. Например:

```
In[6]:= matr = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
        Length[matr]

Out[7]= 3
```

Для того чтобы узнать размерность матрицы, необходимо воспользоваться функцией ***Dimensions[список]***, например

```
In[9]:= Dimensions[matr]

Out[9]= {3, 3}
```

Функция ***VectorQ[выражение]*** возвращает ***True***, если выражение является одномерным вектором, т. е. не содержит вложенных списков и ***False***, если не является:

```
In[1]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
        VectorQ[l]

Out[2]= True

In[3]:= VectorQ[{2, 3, 5, {6, 7}, 5}]

Out[3]= False
```

Помимо этого, в данной функции можно воспользоваться вторым аргументом, в котором можно задать дополнительное условие. Например, мы можем сказать, что все элементы списка должны быть больше 0.

```
In[4]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
        VectorQ[l, # > 0 &]

Out[5]= True
```

Функция ***MatrixQ[выражение]*** возвращает ***True***, если выражение является матрицей и возвращает ***False*** в противном случае.

```
In[1]:= matr = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
        MatrixQ[matr]

Out[2]= True
```

Достаточно часто приходится подсчитывать, сколько элементов списка удовлетворяют определенному условию (шаблону). Для этого можно воспользоваться функцией ***Count[выражение, шаблон]***. Например, если мы захотим проверить, сколько элементов в списке ***l*** равняется 2, то можем записать:

```
In[3]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
        Count[l, 2]

Out[4]= 2
```

Если необходимо проверить, сколько элементов из ***l*** являются целыми, то для этого можно воспользоваться следующим шаблоном:

```
In[1]:= l = {4, 5, 3, 6, 2, 7, 2.1, 2, 8};
        Count[l, _Integer]

Out[2]= 8
```

Мы можем использовать также другие типы в этих шаблонах, например, для вещественных чисел с плавающей точкой мы можем воспользоваться записью ***_Real***, а ***_Complex*** – для комплексных и т. д.

Иногда также бывает необходимо узнать, на каких позициях расположены элементы, удовлетворяющие шаблону. Для этого можно воспользоваться функцией ***Position[выражение, шаблон]***. Например, для предыдущих двух примеров мы получим:

```
In[1]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
        Position[l, 2]
        Position[l, _Integer]

Out[2]= {{5}, {7}}

Out[3]= {{1}, {2}, {3}, {4},
         {5}, {6}, {7}, {8}, {9}}
```

Более подробную информацию по этим и другим функциям данного раздела вы можете посмотреть в книгах или справке по пакету в разделе ***Lists and Matrices =>Lists Testing***, или в соответствующей литературе.

9.3. Операции над списками

Основными операциями при работе со списками является добавление и удаление элементов списков[3]. В пакете *Mathematica* существует несколько функций, позволяющих производить такие операции. Рассмотрим их на примерах.

Функция ***Append[список, элемент]*** возвращает список с добавленным в конце элементом.

```
In[1]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
        Append[l, 13]

Out[2]= {4, 5, 3, 6, 2, 7, 2, 8, 9, 13}
```

В свою очередь, функция ***Prepend[список, элемент]*** возвращает список с добавленным элементом в начале списка.

```
In[3]:= Prepend[l, 13]

Out[3]= {13, 4, 5, 3, 6, 2, 7, 2, 8, 9}
```

Функция ***Insert[список, элемент, n]*** позволяет получить список с элементом, добавленным на ***n***-ю позицию.

```
In[4]:= Insert[l, 13, 3]

Out[4]= {4, 5, 13, 3, 6, 2, 7, 2, 8, 9}
```

[3] Стоит заметить, что, как правило, в пакете *Mathematica* не происходит изменение основного списка, а происходит создание нового списка с необходимыми изменениями. Поэтому, для того чтобы использовать полученные результаты, их необходимо либо присваивать какой-либо переменной (возможно присваивание той же переменной), либо ссылаться на результаты одним из рассмотренных ранее способов. Это относится ко многим модифицирующим функциям.

С помощью функции ***ReplacePart[список, элемент, n]*** можно получить список, в котором элемент, стоящий на позиции с номером ***n***, заменяется на другой:

```
In[5]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
        ReplacePart[l, 9, 2]

Out[6]= {4, 9, 3, 6, 2, 7, 2, 8, 9}
```

Были рассмотрены возможности пакета по добавлению элементов в список. Теперь рассмотрим возможности пакета по удалению элементов из списка.

Функция ***Delete[список, n]*** позволяет получить список, в котором удален элемент, расположенный на позиции n. Причем, если число во втором аргументе отрицательное, то позиция будет отсчитываться с конца.

```
In[7]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
        Delete[l, 2]
        Delete[l, -2]

Out[8]= {4, 3, 6, 2, 7, 2, 8, 9}

Out[9]= {4, 5, 3, 6, 2, 7, 2, 9}
```

Кроме того, с помощью данной функции можно получить список, в котором удалено несколько элементов. Для этого необходимо второй аргумент записать в виде списка. Например, если мы захотим получить список из ***l***, в котором удалены элементы, стоящие на второй позиции с начала и на первой позиции с конца, то в этом случае мы можем записать:

```
In[10]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
         Delete[l, {{2}, {-1}}]

Out[11]=
         {4, 3, 6, 2, 7, 2, 8}
```

Функция ***DeleteCases[список, шаблон]*** позволяет производить удаление по шаблону. Например, если мы хотим получить список из списка ***l***, в котором удалены все элементы, равные 2, то мы можем записать:

```
In[12]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
         DeleteCases[l, 2]

Out[13]=
         {4, 5, 3, 6, 7, 8, 9}
```

Шаблоны могут быть и более сложные, например запись вида ***_Integer*** удалит все целые числа в выражении, а запись ***x^_*** удалит выражения, где ***x*** возводится в любую степень.

Далее мы рассмотрим функции, которые позволяют изменить порядок расположения элементов, а также функции, которые позволяют получать списки на основе нескольких списков по тому или иному правилу.

Для того чтобы просто объединить несколько списков в один, можно воспользоваться функцией ***Join[cn1,cn2,…]***.

```
In[14]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
         Join[l, {4, 6, 2, 6}]

Out[15]=
         {4, 5, 3, 6, 2, 7, 2, 8, 9, 4, 6, 2, 6}
```

Как видно из данного примера, в результате выполнения данной функции получается список, в котором к первому списку, в конец, добавляются элементы второго. Естественно предположить, что во втором аргументе может использоваться не явно заданный список, а переменная, содержащая его или выражение, выделяющее часть списка.

Иногда бывает необходимо получить список на основе нескольких списков, из которых все повторения элементов удаляются, а сам список сортируется по возрастанию. Если выражаться математическим языком, то мы должны найти объединение двух множеств. Это можно сделать с помощью функции ***Union[cn1,cn2,…]***.

```
In[16]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
         Union[l, {4, 6, 2, 6}]

Out[17]=
         {2, 3, 4, 5, 6, 7, 8, 9}
```

Легко догадаться, что если мы запишем один список в качестве аргумента данной функции, то мы получим отсортированный список, из которого убраны повторения элементов.

```
In[18]:= Union[l]

Out[18]=
         {2, 3, 4, 5, 6, 7, 8, 9}
```

В данном случае мы получим тот же результат, что и в предыдущем примере.

Функция ***Intersection[cn1,cn2,…]*** позволяет найти пересечение нескольких множеств. В результате работы данной функции мы получим

отсортированный список, состоящий из элементов, встречающихся во всех перечисленных списках.

```
In[19]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
         Intersection[l, {4, 6, 2, 6}]

Out[20]=
         {2, 4, 6}
```

В результате работы функции ***Complement[cn1,cn2…]*** получается отсортированный список, состоящий из элементов, которые есть только в первом списке и не встречаются в остальных.

```
In[1]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
        Complement[l, {4, 6, 2, 6}]

Out[2]= {3, 5, 7, 8, 9}
```

С помощью функции ***Sort[список, условие сортировки]*** можно отсортировать элементы списка.

```
In[21]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
         Sort[l]

Out[22]=
         {2, 2, 3, 4, 5, 6, 7, 8, 9}

In[23]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
         Sort[l, Greater]

Out[24]=
         {9, 8, 7, 6, 5, 4, 3, 2, 2}
```

Как видно из данного примера, по умолчанию данная функция сортирует элементы по возрастанию. Для того чтобы отсортировать элементы по другому закону, нужно использовать второй аргумент. Например, если мы хотим отсортировать по убыванию, то необходимо использовать аргумент ***Greater***.

Иногда бывает необходимо получить не отсортированный список, а номер, на котором был бы расположен элемент в отсортированном списке. Для этого можно воспользоваться функцией ***Ordering[список, n, условие сортировки]***. Данная функция возвращает список из ***n*** элементов, состоящий из номеров позиций, на которых расположен элемент. Если ***n*** отрицательно, то позиции элементов отсчитываются с конца.

```
l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
Ordering[l]
```

```
In[26]:= Ordering[l, Length[l], Greater]

Out[26]=
{9, 8, 6, 4, 2, 1, 3, 7, 5}
```

Как видно из данного примера, для списка, отсортированного по возрастанию, порядок расположения для всех элементов находится просто. Если то же самое нужно получить для списка, отсортированного по другому закону, необходимо проделать более сложные манипуляции.

Для того чтобы получить список, в котором элементы расположены в обратном порядке, можно воспользоваться функцией ***Reverse[список]***.

```
In[27]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
         Reverse[l]

Out[28]=
{9, 8, 2, 7, 2, 6, 3, 5, 4}
```

Иногда бывает необходимо циклически переставить элементы списка на n позиций вправо или влево. Для этого можно воспользоваться функциями ***RotateLeft[список, n]*** и ***RotateRight[список, n]***, в результате работы которых мы можем получить список, в котором элементы циклически сдвинуты на ***n*** позиций влево или вправо, соответственно. Если необходимо сдвинуть элементы списка на один, в ту или иную сторону, то можно опустить второй аргумент в необходимой функции.

```
In[32]:= l = {4, 5, 3, 6, 2, 7, 2, 8, 9};
         RotateLeft[l, 2]

Out[33]=
{3, 6, 2, 7, 2, 8, 9, 4, 5}
```

```
In[34]:= RotateLeft[l, 2]

Out[34]=
{3, 6, 2, 7, 2, 8, 9, 4, 5}
```

Более подробную информацию по этим и другим функциям данного раздела вы можете посмотреть в книгах или справке по пакету в разделе ***Lists and Matrices => List Operations***.

Задачи и упражнения

1. Считать данные из заранее приготовленного текстового файла, содержащего колонку целых чисел. Для полученного массива найти:
   a) является ли полученный массив вектором (списком);
   b) количество элементов в массиве;
   c) сколько элементов, равных 5, в массиве и на каких местах они находятся;
   d) найти максимальный и минимальный элементы в массиве (см. справку по функциям ***Max*** и ***Min***);
   e) сколько и каких элементов (больше 5) находится в массиве.
2. Над полученным ранее массивом проделать следующие операции:
   a) удалить элементы, равные 5;
   b) отсортировать массив по возрастанию и по убыванию.
3. Считать данные из другого подготовленного текстового файла, содержащего таблицу чисел. Используя этот массив и полученный ранее, выполнить следующие действия над ними:
   a) соединить два списка в один и подсчитать количество элементов в полученном списке;
   b) найти объединение двух списков, а также расположить элементы полученного списка в обратном порядке;
   c) найти пересечение списков;
   d) найти дополнение для первого списка. Другими словами, найти те элементы, которые находятся в первом списке и не встречаются во втором.

10. СИМВОЛЬНЫЕ ВЫЧИСЛЕНИЯ

Пакет *Mathematica* позволяет ускорить решение многих аналитических задач в сферах применения современной прикладной и теоретической математики. Ускорение происходит за счет того, что человеку не нужно проводить весь цикл вычислений вручную, так как часть из них можно сделать в данном пакете. Но следует помнить, что не все задачи, на которые пакет выдает ответ, являются правильными. Так как алгоритмы символьной математики, заложенные в данный пакет, достаточно сложны, то могут быть получены неверные результаты. Из этого следует, что всегда необходимо проверять правильность решения известными методами.

Далее мы рассмотрим часть возможностей пакета *Mathematica* по проведению символьных вычислений.

10.1. Еще раз о списках

Так как списки являются одним из фундаментальных способов представления данных и часто используются в пакете Mathematica, их можно применять и при символьных или численных математических вычислениях. Например, нам необходимо вычислить выражение $y = x^n + 1$, где n=1,2,3,4, при $x = 1.5, 2.3, 4.0$. Одним из вариантов решения данной проблемы является написание программы для вычисления данной задачи. Во втором варианте решения данной задачи можно использовать рассматриваемый пакет и его возможности по работе со списками:

```
In[35]:= n = {1, 2, 3, 4};
         y = x^n + 1
         x = {1.5, 2.3, 4};
         y // TableForm

Out[36]=
         {1 + x, 1 + x^2, 1 + x^3, 1 + x^4}

Out[38]//TableForm=
         2.5     3.3      5
         3.25    6.29     17
         4.375   13.167   65
         6.0625  28.9841  257
```

Из приведенного примера видно, что вначале были заданы значения для параметра **n**. Потом было вычислено необходимое выражение в аналитическом виде. В данном случае вместо **n** был подставлен список значений и, в результате, мы получили тоже список, но состоящий из математических выражений. Потом мы задали список значений для **x** и провели новый просчет выражения, хранящегося в переменной **y**, но уже с новыми численными значениями величины **x** и вывели полученный результат в табличном виде.

Более того, большинство из рассмотренных ранее функций для работы со списками применимо и для символьных представлений.

Например:

```
In[43]:= First[x^3 - 2 x^2 + 3 x - 5]
         Last[x^3 - 2 x^2 + 3 x - 5]
         Part[x^3 - 2 x^2 + 3 x - 5, 3]

Out[43]=
         -5

Out[44]=
         x^3

Out[45]=
         -2 x^2
```

Как видно из данного примера, с использованием списковых функций на математических выражениях тоже возможно, но первое, что бросается в глаза, это то, что результат работы функции ***Part*** не совсем тот которого, можно ожидать с первого взгляда. Это получилось потому, что пакет перед вычислением функций перевел выражение $x^3 - 2x^2 + 3x - 5$ в свое внутреннее представление, узнать которое можно с помощью функции ***FullForm***:

```
In[46]:= FullForm[x^3 - 2 x^2 + 3 x - 5]

Out[46]//FullForm=
         Plus[-5, Times[3, x],
          Times[-2, Power[x, 2]], Power[x, 3]]
```

Отсюда видно, что *Mathematica* переставила слагаемые в обратном порядке, и это объясняет полученные результаты.

Списки можно использовать при решении аналитических и численных задач во многих случаях и комбинациях, а круг их применения очень обширен, поэтому нельзя показать на примерах весь спектр задач. Нужно просто помнить о возможностях работы со списками и стараться их применять для облегчения решения задач.

11. ОПЕРАЦИИ МАТЕМАТИЧЕСКОГО АНАЛИЗА

В данном разделе мы рассмотрим некоторые возможности пакета *Mathematica* по операциям математического анализа, детали которых можно найти в любом справочнике по высшей математике. Данные операции, чаще всего, используются при проведении математических и научно-технических расчетов и поэтому описаны достаточно полно.

11.1. Вычисление сумм

В числе операций математического анализа, прежде всего, следует отметить суммы.

$$\sum_{i=i\min}^{i\max} f_i.$$

В математическом представлении индекс **i** принимает целочисленные значения от **imin** до **imax**, с шагом 1.

Для вычисления сумм в пакете *Mathematica* существует функция ***Sum[f(i),{i,imin,imax}]***, которая позволяет, если это возможно, получать значение суммы в аналитическом виде. Например, вычислим следующую сумму:

$$\sum_{k=1}^{n} \frac{(-1)^k k}{4k^2 - 1}.$$

Это можно проделать двумя способами: использовать панель быстрого набора или использовать функцию ***Sum***. Оба способа, чаще всего, равнозначны, поэтому рассмотрим второй способ. В результате вычислений получим:

```
In[49]:= Sum[(-1)^k k/(4 k^2 - 1), {k, 1, n}]

Out[49]=
                -1 + (-1)^n - 2 n
           ---------------------------
                  4 (1 + 2 n)
```

Следует заметить, что запись вида «**число!**» – означает вычисление соответствующего факториала.

С помощью данной функции можно вычислять не только конечные суммы, но и бесконечные. Например:

```
In[50]:= Sum[x^k/k!, {k, 1, Infinity}]

Out[50]=
                  x
           -1 + e
```

С помощью данной функции можно также вычислять суммы, для которых шаг по параметру ***i*** не равен 1. Для этого используется запись вида ***Sum[f(i),{i,imin,imax,di}]***. Например:

```
In[51]:= Sum[i^2, {i, 1, 10, 0.25}]

Out[51]=
         1382.88
```

Помимо всего прочего, в данном пакете возможно вычисление многократных сумм:

```
In[54]:= Sum[Sin[x^m + y^n], {m, 1, 3}, {n, 1, m}]

Out[54]=
         Sin[x + y] + Sin[x^2 + y] + Sin[x^3 + y] +
          Sin[x^2 + y^2] + Sin[x^3 + y^2] + Sin[x^3 + y^3]
```

Бывает так, что *Mathematica* не может свести результат к аналитическому решению и выдает результат в численном виде, но не всегда ее результаты бывает легко интерпретировать, например:

```
In[55]:= Sum[1/n^2 + n, {n, 1, 50}]

Out[55]=
         2 452 161 173 950 367 144 669 121 831 504 545 /
           815 454 461 709 /
          1 920 815 367 859 463 099 600 511 526 151 929 /
           560 192 000
```

В этом случае можно воспользоваться уже известной функцией ***N***

```
In[56]:= Sum[1/n^2 + n, {n, 1, 50}] // N

Out[56]=
         1276.63
```

или специальной функцией для численных расчетов сумм – ***Nsum***

```
In[57]:= NSum[1/n^2 + n, {n, 1, 50}]

Out[57]=
         1276.63
```

Если изначально значение суммы нужно получить не в аналитическом, а в численном виде, то использование функции ***NSum*** для таких

вычислений более предпочтительно, так как данная функция оптимизирована для такого рода расчетов, и результат, чаще всего, получается быстрее, чем в первом случае. Для функции ***NSum*** справедливы те же способы записи, что и для ***Sum.***

11.2. Вычисление произведений

Для вычисления выражений вида:

$$\prod_{i=i\min}^{i\max} f_i$$

существуют функции ***Product*** и ***NProduct***, которые позволяют вычислять произведения в аналитическом и численном виде, соответственно. Правила записи аргументов и возможности вычислений те же, что и для функций ***Sum*** и ***NSum***, поэтому в данном разделе приведем только примеры использования данных функций.

Вычисление произведений с конечными пределами:

```
In[58]:= Product[Csc[(k Pi) / n]^2, {k, 1, n - 1}]
```

Out[58]=

$$\frac{4^{-1+n}}{n^2}$$

Вычисление произведений с бесконечными пределами:

```
In[59]:= Product[1 - x/k^2, {k, 1, ∞}]
```

Out[59]=

$$\frac{\text{Sin}\left[\pi\sqrt{x}\right]}{\pi\sqrt{x}}$$

11.3. Вычисление пределов

Вычисление выражений вида

$$\lim_{x\to x_0} f(x),$$

т. е. вычисление предела функции при ***x***, стремящемся к x_0, осуществляется с помощью функции ***Limit[f(x),x→x0]***.

Например:

```
In[60]:= Limit[(4 x^2 - 1)/(4 x^2 + 8 x + 3), x → -1/2]
```

Out[60]=

$$-1$$

С помощью данной функции можно вычислить предел и при стремлении к бесконечности:

```
In[61]:= Limit[n/(n - 1), n → ∞]

Out[61]=
1
```

Как известно из курса высшей математики, некоторые функции требуют вычисления предела слева и справа от значения x_0. Для этого можно воспользоваться третьим аргументом данной функции. Например, вычислим предел от функции

$$\lim_{x \to \pi/2} \tan x$$

слева

```
In[62]:= Limit[Tan[x], x -> π/2, Direction → 1]

Out[62]=
∞
```

и справа

```
In[63]:= Limit[Tan[x], x -> π/2, Direction → -1]

Out[63]=
-∞
```

Как видно из данных примеров, предел стремления задается флагом ***Direction***. Если он равен 1, то ищется предел стремления слева, а если –1, то справа.

11.4. Вычисление производных

К числу наиболее часто используемых математических операций принадлежит вычисление производных. Для дифференцирования существуют две функции: ***D[f,{x,n},{y,m},…]*** и ***Dt[f]***. Первая функция позволяет вычислять частную производную разного порядка от функций нескольких переменных. Естественно, что с помощью данной функции можно вычислять и производные от одной переменной. Для этого достаточно не указывать ненужные аргументы. Например, вычислим первую производную от функции $\sin x$:

```
In[64]:= D[Sin[x], x]

Out[64]=
Cos[x]
```

Как видно из данного примера, если нам необходимо вычислить первую производную от функции, то мы можем не указывать порядок дифференцирования. То же самое правило действует и для функций нескольких переменных, когда нам нужно вычислить частные производные 1 порядка:

```
In[67]:= D[Sin[x - y] + Cos[2 y], x, y]

Out[67]=
         Sin[x - y]
```

Приведем примеры вычисления производных для порядка, отличного от первого:

```
In[68]:= D[Tan[x], {x, 2}]

Out[68]=
         2 Sec[x]^2 Tan[x]
```

```
In[69]:= D[Sin[x - y] + Cos[2 y], {x, 2}, y]

Out[69]=
         Cos[x - y]
```

Любое наименование, не содержащее чисел (при условии, что ему ранее не было задано некоторое выражение, зависящее от параметров дифференцирования), функция ***D*** рассматривает как константу:

```
In[70]:= D[a Tan[x], {x, 2}]

Out[70]=
         2 a Sec[x]^2 Tan[x]
```

Для рассматриваемой функции существует опция ***NonConstants***, которая позволяет задавать список объектов, находящихся в неявной зависимости от переменных дифференцирования. По умолчанию этот список пустой.

Функция ***Dt[f,{x,n},{y,m},…]*** позволяет вычислять полный дифференциал. Для данной функции справедливы те же способы записи, что и для ***D***. Отличие заключается в том, что по умолчанию данная функция считает все выражения, не содержащие символов, по которым производится дифференцирование, не явно зависящих от параметров дифференцирования. И поэтому для нее существует опция ***Constants***, по умолчанию пустая, которая указывает какие переменные считать независимыми от параметров дифференцирования. Проиллюстрируем это несколькими примерами:

```
In[71]:= Dt[Sin[x*y], x]

Out[71]=
         Cos[x y] (y + x Dt[y, x])

In[72]:= Dt[Sin[x*y], x, Constants → y]

Out[72]=
         y Cos[x y]
```

11.5. Вычисление интегралов

Операцией, обратной дифференцированию, является нахождение первообразной (вычисление неопределенного интеграла). Первообразная – это функция F(x), удовлетворяющая условию

$$(F(x))' = f(x).$$

Вычисление определенного интеграла с пределами – верхним (**b**) и нижним (**a**) – производится по формуле:

$$\int_a^b f(x)dx = F(b) - F(a).$$

Следует заметить, что определенный интеграл может быть вычислен как аналитически, так и численно.

В пакете *Mathematica* существует функция, которая позволяет получать, если это возможно, аналитически значения как определенных, так и неопределенных интегралов. Называется она ***Integrate***. Для нахождения значений неопределенных интегралов можно воспользоваться следующей формой записи данной функции ***Integrate[f,x,y,…]***. Например, вычислим первообразную от функции $\cos x$:

```
In[73]:= Integrate[Cos[x], x]

Out[73]=
         Sin[x]
```

Вычислим также кратный неопределенный интеграл:

```
In[74]:= Integrate[Sin[x - y], x, y]

Out[74]=
         Sin[x - y]
```

Как видно из данных примеров, для первого случая мы получили точное соответствие полученного выражения с дифференцируемой функцией, а во втором – нет. Для того что бы этого добиться и для второй функции, необходимо воспользоваться функциями преобразования выражений, но об этом будет рассказано позже.

Для вычисления определенных интегралов существует вторая форма записи данной функции:

Integrate[f,{x,xmin,xmax},{y,ymin,ymax},…].

Приведем примеры вычисления нескольких определенных интегралов:

```
In[79]:= Integrate[1/Sqrt[1 + x^π], {x, 0, Infinity}]

Out[79]= Gamma[1/2 - 1/π] Gamma[1/π] / π^(3/2)

In[81]:= Integrate[x/Sqrt[1 + x^2 - x^4], {x, 0, 1}]

Out[81]= ArcCot[2]
```

Следует заметить, что лишь небольшая группа интегралов, встречающихся в реальных математических задачах, берется аналитически. Такие интегралы вычисляются численно. Для численных вычислений определенных интегралов существует специальная функция и, как легко догадаться, называется она ***NIntegrate***. Более подробную информацию по данной функции можно получить из литературы или справки пакета *Mathematica*.

11.6. Преобразование выражений

Иногда бывает необходимо, перед тем как применить один из способов вычисления, рассмотренных ранее, преобразовать выражения, например попытаться упростить вычисление той или иной функции путем раскрытия выражений и приведения к другим формам записи. Рассмотрим некоторые функции из пакета *Mathematica*, позволяющие тем или иным образом преобразовывать выражения.

Рассмотрим вначале функции, позволяющие упрощать выражения, так как упрощение математических выражений является одной из самых важных задач символьной математики. Часто бывает, что невероятно сложное математическое выражение сводится к простому выражению после ряда вполне заурядных (хотя, порою, и довольно сложных) преобразований. *Mathematica* всегда старается упростить, там, где это возможно, то или иное выражение, если для этого не требуется каких-либо особых средств. Например, сложные выражения, содержащие элементарные или специальные функции, преобразуются в более простые в том лишь случае, если они состоят из более простых функций:

```
In[82]:= (Csc[x] Tan[y]) / (Cot[x] Sec[y])

Out[82]=
         Sec[x] Sin[y]
```

Следует заметить, что так бывает далеко не всегда. Например, в результате некоторых вычислений могут получиться выражения, которые *Mathematica* без дополнительных указаний не сможет упростить

```
In[83]:= D[x Exp[x^2] / Sqrt[1 - x^2], x]
```

Out[83]=

$$\frac{e^{x^2}\, x^2}{\left(1 - x^2\right)^{3/2}} + \frac{e^{x^2}}{\sqrt{1 - x^2}} + \frac{2\, e^{x^2}\, x^2}{\sqrt{1 - x^2}}$$

В таких случаях для упрощения выражений можно воспользоваться функцией ***Simplify[выр]***. Она исполняет последовательность алгебраических преобразований над выражением и возвращает простейшую из найденных форм. Например, в предыдущем случае мы можем получить

```
In[84]:= D[x Exp[x^2] / Sqrt[1 - x^2], x] //
          Simplify
```

Out[84]=

$$\frac{e^{x^2}\, \left(1 + 2\, x^2 - 2\, x^4\right)}{\left(1 - x^2\right)^{3/2}}$$

В результате данной операции мы получили более компактное выражение по сравнению с предыдущим случаем.

Функция ***Simplify*** работает с самыми различными математическими выражениями: многочленами, рациональными выражениями (состоящими из полиномов и их отношений), элементарными и специальными функциями, алгебраическими и тригонометрическими выражениями и т. д. Данная функция очень часто выполняется после некоторых операций по умолчанию, например после ***Integrate***.

Иногда с помощью рассмотренной только что функции не получается упростить выражения. Для более сильного упрощения с возможностью приведения к специальным математическим функциям можно воспользоваться функцией ***FullSimplify[выр].*** Ее следует использовать там, где ***Simplify*** не смогла получить достаточно простых результатов.

```
In[87]:= D[Integrate[Sqrt[(2-x)/(3+x)]/Sqrt[2-x], x] x]

         Simplify[%]
         FullSimplify[%%]
```

Out[87]=

$$\frac{2\sqrt{2-x}\;x}{\sqrt{\frac{2-x}{3+x}}}$$

Out[88]=

$$\frac{2\sqrt{2-x}\;x}{\sqrt{\frac{2-x}{3+x}}}$$

Out[89]=

$$\frac{2x}{\sqrt{\frac{1}{3+x}}}$$

Как видно из данного простого примера, функция ***FullSimplify*** с упрощением данного выражения справилась более успешно.

Время от времени бывает необходимо проделать обратную операцию с выражениями, а именно раскрыть выражения. В пакете *Mathematica* для этих случаев существует целый набор функций класса ***Expand***.

Первой функцией данного класса является функция, давшая название всему классу и записывается она следующим образом: ***Expand[выр]***. Данная функция раскрывает произведения и положительные степени сумм. Например, попытаемся раскрыть следующее выражение:

```
In[90]:= Expand[(1 + x)^3 + (2 + x)^4 + (1 + y)^2]
```

Out[90]=

$$18 + 35x + 27x^2 + 9x^3 + x^4 + 2y + y^2$$

У функции ***Expand*** может быть второй аргумент. При вычислении выражения ***Expand[выр,шаблон]*** раскрываются только те элементы выражения, которые удовлетворяют шаблону.

```
In[91]:= Expand[(1 + x)^3 + (2 + x)^4 + (1 + y)^2,
          _Integer + x]
```

Out[91]=

$$17 + 35x + 27x^2 + 9x^3 + x^4 + (1 + y)^2$$

Вместо данного аргумента также можно использовать аргумент ***Trig→True***. Данный флаг указывает, что тригонометрические функции должны трактоваться как рациональные функции экспонент. Также, вместо указания данного флага, можно просто воспользоваться другой функцией из данного класса, а именно ***TrigExpand***. Рассмотрим описанное в данном разделе на следующем примере:

```
In[95]:= Expand[Sin[2 ArcTan[t]]]
         Expand[Sin[2 ArcTan[t]], Trig → True]
         TrigExpand[Sin[2 ArcTan[t]]]

Out[95]=
         Sin[2 ArcTan[t]]

Out[96]=
           2 t
         1 + t^2

Out[97]=
           2 t
         1 + t^2
```

Для экономии места приведем остальные функции класса ***Expand*** с кратким описанием в табл. 5.

Таблица 5

Функции класса Expand

1.1.1. Функция	**Описание**
ComplexExpand[выр]	раскрывает выражение, полагая, что все переменные являются вещественными числами
ComplexExpand[выр,{x1,x2,…]	раскрывает выражение, полагая, что переменные *xi* являются комплексными
ExpandAll[выр]	раскрывает все произведения и целочисленные степени в любой части выражения
ExpandAll[выр,шаблон]	то же самое, но при раскрытии исключает элементы не соответствующие шаблону
ExpandDenominator[выр]	раскрывает произведения и степени, которые присутствуют в выражении в роли знаменателей
ExpandNumerator[выр]	то же, но для числителей
PowerExpand[выр]	раскрывает вложенные степени, степени произведений и логарифмы от произведений
FunctionExpand[выр]	рроизводит расширение выражений, содержащих специальные функции

К функциям, расширяющим выражения, относится также функция ***Collect[выр,{x1,x2,…}]***. Данная функция выполняет приведение общих членов выражения по степеням переменных x^i. Поясним работу данной функции на следующем примере:

```
In[98]:= Collect[x + 4 y + 5 x y, x]
         Collect[x + 4 y + 5 x y, y]
         Collect[x^4 + (a + y) (x + y^3) x^2 + (y^3 + 3) x^2,
          {x, y}]
```

Out[98]=

$$4\,y + x\,(1 + 5\,y)$$

Out[99]=

$$x + (4 + 5\,x)\,y$$

Out[100]=

$$x^4 + x^3\,(a + y) + x^2\,\left(3 + (1 + a)\,y^3 + y^4\right)$$

Противоположной функциям класса ***Expand*** по действию являются функции класса ***Factor***. Например, функция ***Factor[выр]*** раскладывает, над полем рациональных чисел, полиномы на множители:

```
In[101]:=
         Factor[x^5 + x^4 + x + 1]
         Factor[x^17 - 1]
```

Out[101]=

$$(1 + x)\,\left(1 + x^4\right)$$

Out[102]=

$$(-1 + x)\left(1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{10} + x^{11} + x^{12} + x^{13} + x^{14} + x^{15} + x^{16}\right)$$

Для более подробной информации по этим и другим функциям можно обратиться к литературе или справке пакета *Mathematica*.

11.7. Подстановки

Одним из наиболее распространенных видов алгебраических преобразований являются подстановки, в результате выполнения которых какая-либо часть алгебраического выражения заменяется новым выражением. В пакете *Mathematica* подстановку можно осуществить двумя способами.

Один способ реализуется с помощью функции ***Set[перемен,выр]*** или ***перемен=выр*** и выполняется следующим образом. Сначала вычисляется выражение, а затем вычисленное выражение присваивается как значение переменной. Далее везде при вычислениях вместо переменной будет подставляться ее значение. Например, заменим в следующем выражении все x на a и проведем расширение выражения:

```
In[103]:=
a1 = (1 + x)^3 + (2 + x)^4 + (1 + y)^2
x = a
Expand[a1]

Out[103]=
(1 + x)^3 + (2 + x)^4 + (1 + y)^2

Out[104]=
a

Out[105]=
18 + 35 a + 27 a^2 + 9 a^3 + a^4 + 2 y + y^2
```

Неудобством этого способа является то, что теперь везде, где бы не встретился символ ***x***, он будет заменяться на его значение, в данном случае на ***a***. Это бывает не совсем удобно, если подстановку нужно проделать несколько раз, а потом нам будет необходимо использование не присвоенного символа.

Отменить глобальную подстановку для переменной можно с помощью функции ***Clear[пер1,пер2,…]***. Данная функция сбрасывает значения переменных переданных, в качестве аргументов. Для сброса значения одной переменной можно воспользоваться функцией ***Unset[переменная]*** (или ее короткой записью ***переменная=.***), например

```
In[106]:=
x = .
a1

Out[107]=
(1 + x)^3 + (2 + x)^4 + (1 + y)^2
```

Как видно из данного примера, после данной операции переменная ***x*** стала простым символом, не хранящим свое значение.

Функция **Set** задает «глобальную» подстановку, оказывающую влияние на все последующие вычисления с ее первым аргументом.

Если же подстановку одного выражения вместо другого нужно сделать в одном конкретном выражении или же в нескольких, то по мере возникновения в этом необходимости применяется другой механизм, использующий функцию ***Rule***.

Выражение ***Rule[lhs,rhs]***, или ***lhs→rhs***, задает правило, в соответствии с которым может быть сделана подстановка вычисленного выражения ***rhs*** вместо ***lhs***. Выражение ***rhs*** вычисляется в момент задания ***Rule***. Если ***rhs*** целесообразно вычислять в момент применения правила, то употребляется функция ***RuleDelayed***, или ***lhs :> rhs***.

После того как подстановка определена в виде правила **rl = x→a**, она может быть применена к конкретному выражению с помощью функций ReplaceAll (постфиксная форма/.):

```
In[108]:=
rl = x → a;
ReplaceAll[Expand[(y + x + a) ^ 2], rl]
Expand[(y + x + a) ^ 2] /. rl

Out[109]=
4 a^2 + 4 a y + y^2

Out[110]=
4 a^2 + 4 a y + y^2
```

Подстановку целесообразно отдельно задавать и присваивать в качестве значения какому-либо символу только тогда, когда есть основания полагать, что она будет применяться несколько раз в ходе вычислений. Если же подстановка применяется один раз, то ее можно задать в виде второго аргумента функции ***ReplaceAll*** непосредственно в момент совершения подстановки:

```
In[111]:=
(y + x + a) ^ 2 /. x → a

Out[111]=
(2 a + y)^2
```

Функция ***ReplaceAll*** позволяет осуществить несколько подстановок одновременно. Тогда эти подстановки должны быть оформлены в виде списка и заданы вторым аргументом этой функции:

```
In[112]:=
(y + x + a) ^ 2 /. {x → a, y → a}

Out[112]=
9 a^2
```

Иногда возникает необходимость делать подстановки повторно. Предположим, что в выражении ***Expand[(x + y + a)^2]*** нужно избавиться от ***x*** и ***y***, заменив ***x*** на ***a***, ***y*** на ***ax***, т. е. в конечном счете ***y*** на a^2. Тогда можно сделать либо подстановку ***{y→ax,x→a}***, либо применить два раза подстановку ***{x→a,y→ax}***. Последнее делают с помощью функции ***ReplaceRepeated*** (в постфиксной форме ***//.***).

```
In[113]:=
Expand[(y + x + a) ^ 2 //. {x → a, y → a x}]

Out[113]=
4 a^2 + 4 a^3 + a^4
```

Очень часто подстановки применяются при проверке полученных решений с помощью функции ***Solve***, т. е. для проверки правильности решения уравнений, например,

```
In[114]:=
Solve[a = x^3 - 5 x^2 + 2 x + 8 == 0, x]

Out[114]=
{{x → -1}, {x → 2}, {x → 4}}

In[115]:=
a /. %

Out[115]=
{True, True, True}
```

Как видно из данного примера, результат решения уравнения записан в виде подстановок и после проведения подстановки в начальное уравнение мы получили, что все три корня обращают его в истину.

Задачи и упражнения

1. Преобразовать выражения:
   a) $\dfrac{\ln(x-5)-\ln(2x^2+1)}{x^2+1}$;
   b) $(8x+3)^3\cdot(5x^2+3x+4)^4$;
   c) $\dfrac{\sin(x^2+\pi)\cos(x^2-\pi)}{\cot x^2}$;
   d) $((x-y)^3+(y-x)^4)^2$;
   e) $(2x-1)^2+(2x-2)^3(5x-1)^4$. При преобразовании этого выражения оставить нераскрытым множитель $(5x-1)$.
2. Вычислить аналитически и численно. Оценить время, затрачиваемое на вычисления:
   a) $\lim\limits_{x\to-1}\dfrac{x^3}{2(1-x)^2}$;
   b) $\sum\limits_{i=1}^{\infty}\dfrac{1}{i^2}$;
   c) $\prod\limits_{i=1}^{\infty}\dfrac{1}{i^2}$;
   d) $\sum\limits_{i=1}^{1000}\dfrac{1}{(1+i)^2}$
3. Найти корни уравнения $x^8-2x^5+x^3-3=3$ и выполнить проверку, используя подстановку.
4. Решить системы уравнений и проверить результат вычислений:
   a) $\begin{cases} 2x_1-x_2+x_3=4 \\ x_1+x_2-x_3=2 \\ 2x_1-x_2+3x_3=6 \end{cases}$;
   b) $\begin{cases} 2x_1+x_2-5x_3+x_4=8 \\ -3x_2-6x_4=9 \\ x_1+2x_2-x_3+2x_4=-5 \\ x_1+4x_2-7x_3+6x_4=0 \end{cases}$.
5. Вычислить интеграл $\displaystyle\int\frac{dx}{\arcsin(x)\sqrt{1-x^2}}$ и проверить результат дифференцированием. Постарайтесь добиться полного совпадения подынтегрального выражения результату дифференцирования.

12. РАБОТА С ГРАФИКОЙ

При решении многих задач очень часто оказывается, что полученные аналитические или численные данные достаточно трудно осмыслить. Во многих случаях решением проблемы может оказаться построение графиков по полученным результатам. Пакет *Mathematica* имеет в своем распоряжении не очень большой набор функций для построения графиков, но, несмотря на это, он позволяет строить практически все типы графиков. Благодаря большому количеству опций и директив для графических функций пакет позволяет строить достаточно продвинутые и сложные графики с красивым оформлением. Далее мы постараемся рассмотреть основные возможности по построению графиков.

12.1. Построение графиков функций, заданных аналитически

Следует заметить, что все функции для построения двухмерных графиков оканчиваются на ***Plot***, а трехмерных – на ***Plot3D***. Приведем в табл. 6 основные функции для построения графиков функций.

Таблица 6

Функции для построения графиков заданных аналитически

Функция	***Описание***
Plot[f(x),{x,xmin,xmax}]	Построение графика функции ***f (x)*** в декартовых координатах для ***x*** в пределах от ***xmin*** до ***xmax***
ParametricPlot[{f_x,f_y},{t,tmin,tmax}]	Построение графика функции, заданной в параметрической форме
Plot3D[f(x,y),{x,xmin,xmax}, {y,ymin,ymax}]	Построение трехмерного графика функции ***f(x,y)*** в декартовых координатах при изменении аргументов функции от минимального до максимального значения
ParametricPlot3D[{f_x,f_y,f_z},{t,tmin,tmax}, {u,umin,umax}]	Построение трехмерного графика функции, заданной в параметрической форме
ContourPlot[f(x,y),{x,xmin,xmax}}, {y,ymin,ymax}]	Построение контурного графика от функции двух переменных
DensityPlot[f(x,y),{x,xmin,xmax}}, {y,ymin,ymax}]	Построение графика плотности от функции двух переменных

Далее на простых примерах рассмотрим возможности пакета *Mathematica* по построениям графиков функций.

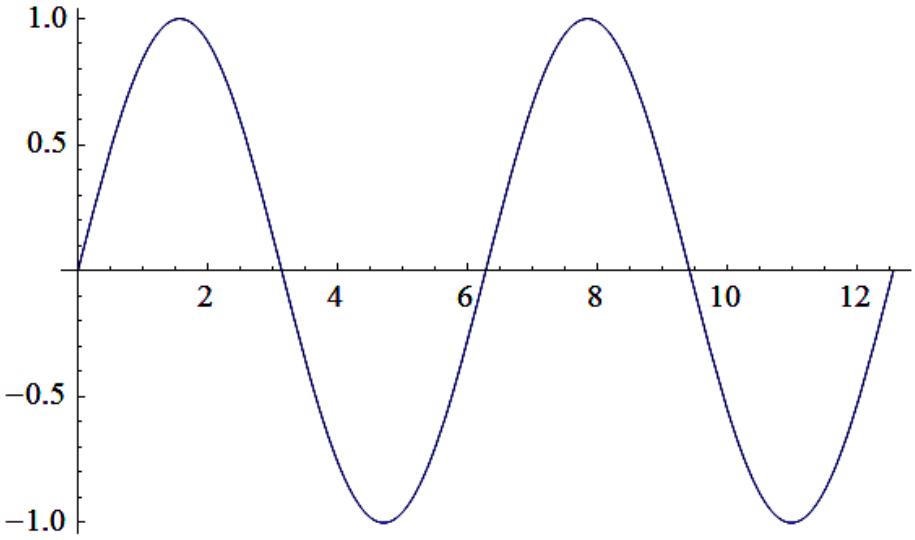
Одним из самых распространенных типов графиков являются простые графики функций одной переменной. Для построения таких гра-

фиков, как показано в табл. 6, используется функция ***Plot***. Для примера нарисуем график функции ***Sin(x)*** в пределах от 0 до 4π.

In[116]:=

```
Plot[Sin[x], {x, 0, 4 Pi}]
```

Out[116]=

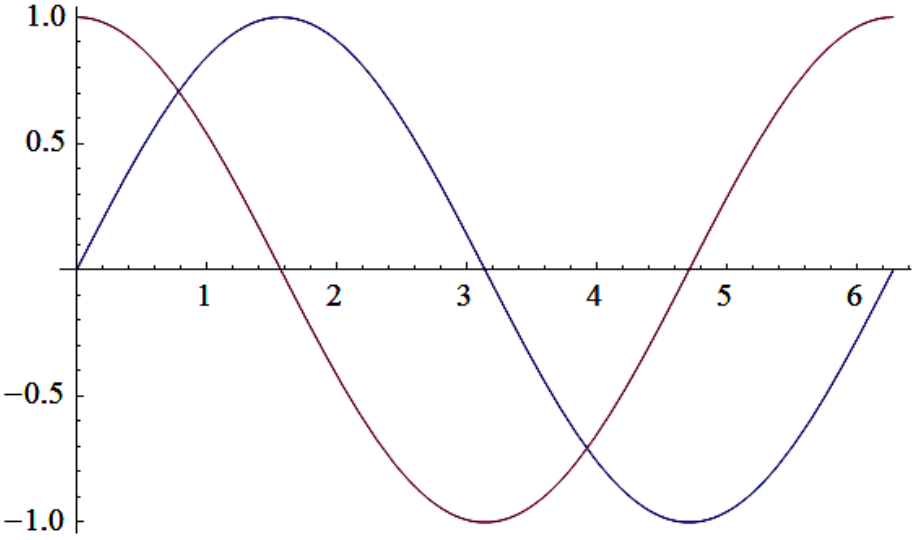


С помощью данной функции также можно построить графики нескольких функций в одних координатах. Для этого необходимо функции в первом аргументе записать в виде списка, например нарисуем графики функций $\sin x$ и $\cos x$ на одном графике.

In[117]:=

```
Plot[{Sin[x], Cos[x]}, {x, 0, 2 Pi}]
```

Out[117]=



Как видно, в данном случае мы получили графики двух функций на одном графике.

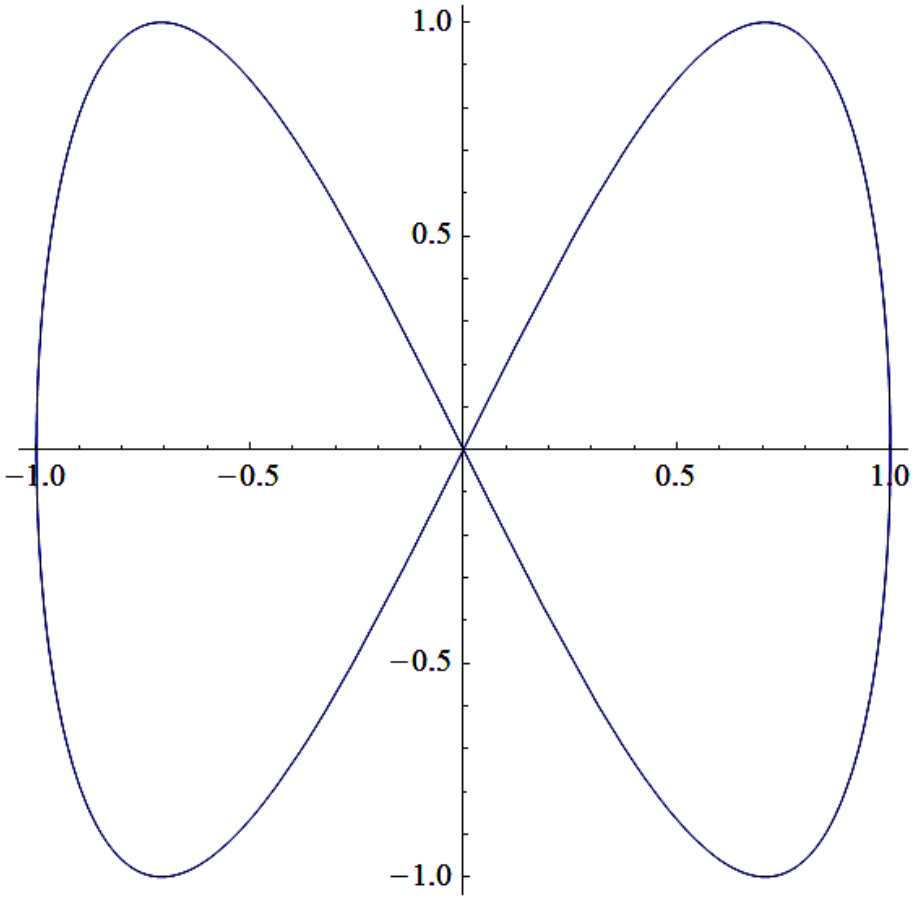
Для построения графиков в полярных координатах можно использовать два способа: собственно использование функций, заданных в полярных координатах, и построение графиков функций, заданных в параметрической форме. В пакете *Mathematica* встроенным является второй способ. Данный способ основан на использовании обычных декартовых координатах. Координаты каждой точки при этом задаются в параметрическом виде: $x = f_x(t)$, $y = f_y(t)$, где независимая переменная t меняется от минимального значения ***tmin*** до максимального ***tmax***. Особенно удобно применение таких функций для построения замкнутых линий, таких как окружности, эллипсы, циклоиды и т. д. Покажем отображение графика функции

$$\begin{cases} x = \sin(t) \\ y = \sin(2t) \end{cases}, \text{ где } 0 \le t \le 2\pi$$

на следующем примере:

```
In[118]:=
ParametricPlot[{Sin[t], Sin[2 t]},
 {t, 0, 2 Pi}]
```

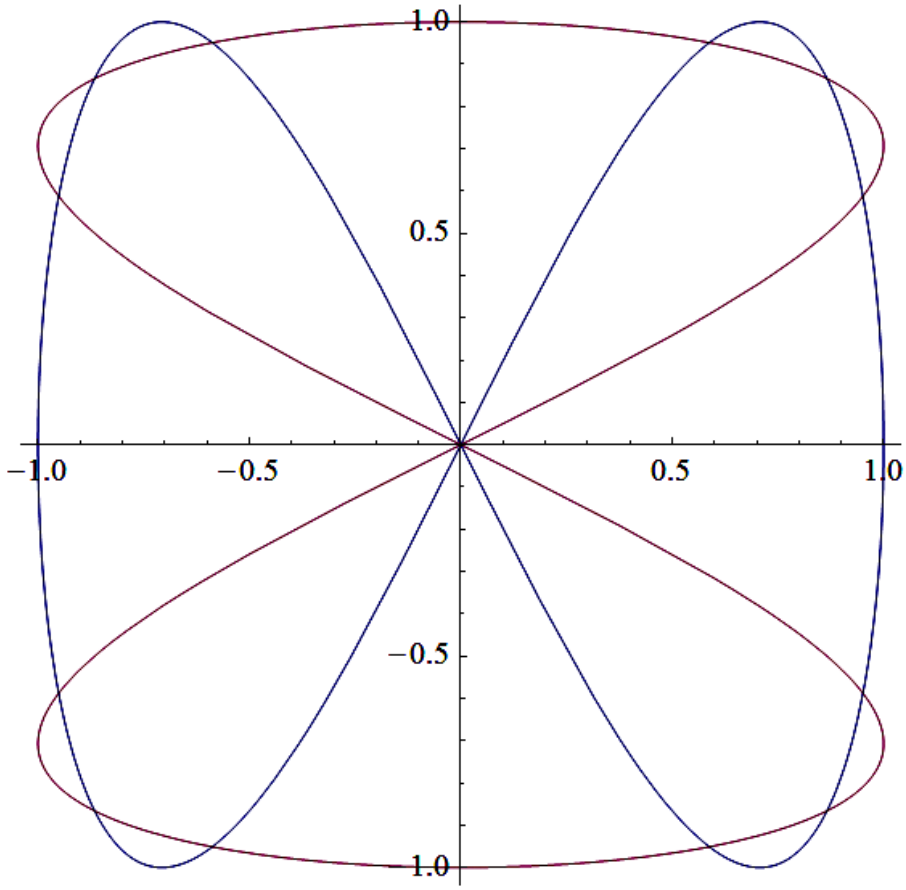
Out[118]=



Для построения нескольких графиков в данном случае необходимо проделать практически то же самое, что и для предыдущей команды. Отличие заключается в том, что в данной команде первый аргумент уже записан как список и для представления придется воспользоваться вложенным списком. Для примера нарисуем только что построенную функцию вместе с нею же, но повернутой на 90 градусов. В результате получим:

```
In[120]:=
ParametricPlot[
 {{Sin[t], Sin[2 t]}, {Sin[2 t], Sin[t]}},
 {t, 0, 2 Pi}]
```

Out[120]=

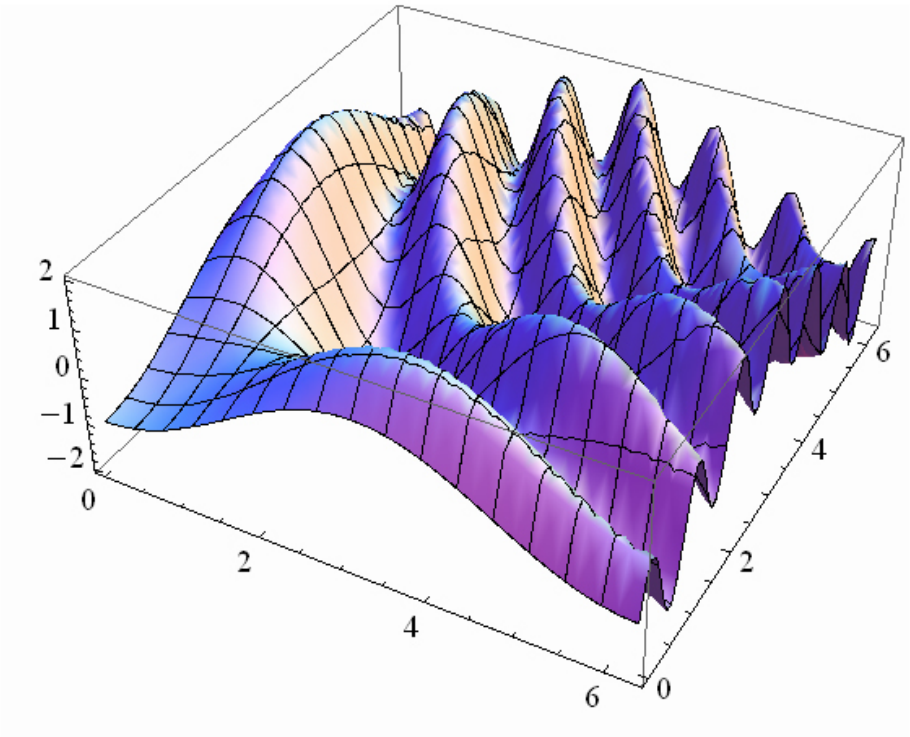


Следует заметить, что построение нескольких графиков на одном возможно практически для всех типов графиков, за исключением контурных и графиков плотности. Правила задания для построения во всех случаях одни и те же. Если аргумент простой, то функции графиков, которые необходимо построить, записываются в виде списка, а если функции для задания одного графика уже записаны в виде списка, то используются вложенные списки.

Теперь рассмотрим команды для построения графиков функций двух переменных. Одним из самых распространенных типов графиков функций является простой график от выражения, зависящего от двух переменных. Для построения графиков такого типа необходимо воспользоваться встроенной функцией пакета *Mathematica* ***Plot3D***. На следующем примере показан график функции $f(x,y) = \sin(xy) - \cos(x-y)$, где ***x*** и ***y*** изменяются от ***0*** до ***2π***.

```
In[123]:=
Plot3D[Sin[x y] - Cos[x - y], {x, 0, 2 Pi},
 {y, 0, 2 Pi}]
```

Out[123]=



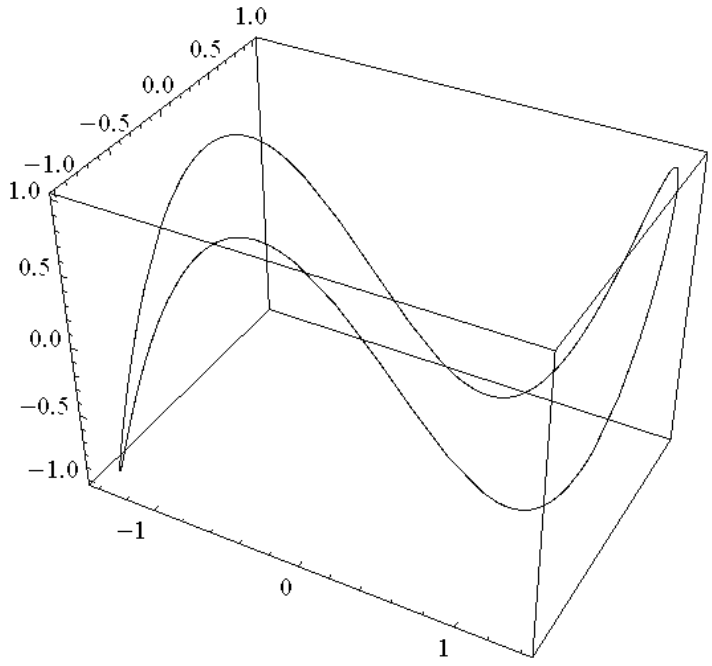
Следующая функция позволяет строить функции, заданные в параметрической форме, и называется ***ParametricPlot3D***. Причем, если координаты всех трех координат, заданных в параметрической форме, зависят только от одного параметра, например ***t***, то графиком будет кривая нарисованная, в трехмерном пространстве. Для примера построим график функции:

$$\begin{cases} x = \sin t + \cos t \\ y = \cos t \qquad\qquad ,0 \le t \le 2\pi. \\ z = \sin 3t \end{cases}$$

In[124]:=

```
ParametricPlot3D[
 {Sin[t] + Cos[t], Cos[t], Sin[3 t]},
 {t, 0, 2 Pi}]
```

Out[124]=

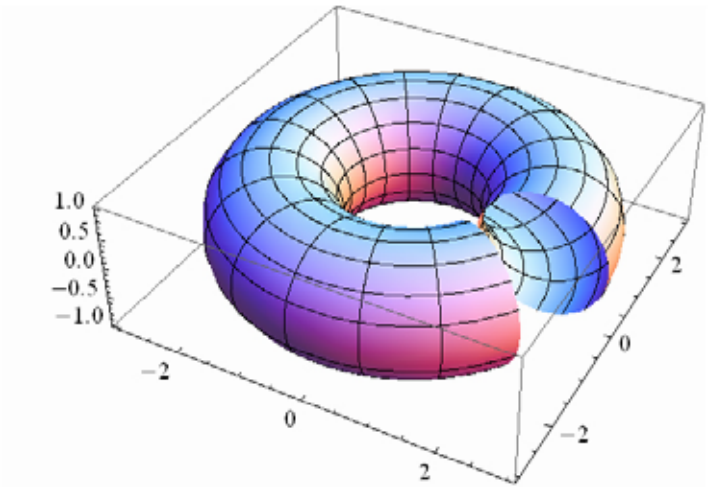


В том случае, если координаты будут зависеть от двух параметров, например ***t*** и ***u***, на графике будет отображаться объемная поверхность. Для примера построим график тороида с вырезом.

In[125]:=

```
ParametricPlot3D[
 {Cos[t] (2 + Cos[u]), Sin[t] (2 + Cos[u]),
  Sin[u]}, {t, 0, 2 Pi - 0.6}, {u, 0, 2 Pi}]
```

Out[125]=

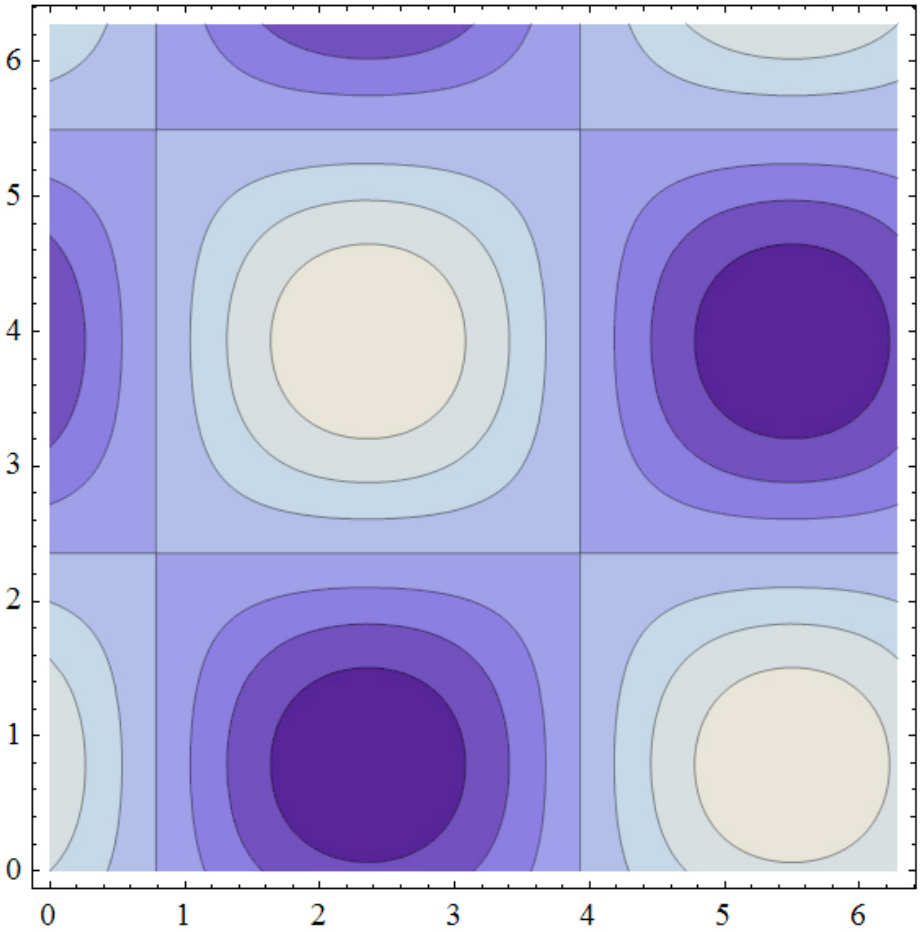


Следующие два типа графиков являются специфическими, но довольно популярными в некоторых задачах.

Первый тип графиков – это *контурные* графики, построение которых возможно с помощью функции ***ContourPlot***. ***Контурные графики***, или графики ***линий равных высот***, используются для отображения поверхностей на плоскости. Они удобны для выявления всех экстремумов функций в пределах области графика. Такие графики являются линиями пересечения поверхности с секущими горизонтальными плоскостями, расположенными параллельно друг другу. Для примера построим контурный график функции $\cos(x+y)-\sin(x-y)$, где ***x*** и ***y*** изменяются в пределах от ***0*** до ***2π***.

```
In[1]:= ContourPlot[Cos[x + y] - Sin[x - y],
          {x, 0, 2 Pi}, {y, 0, 2 Pi}]
```

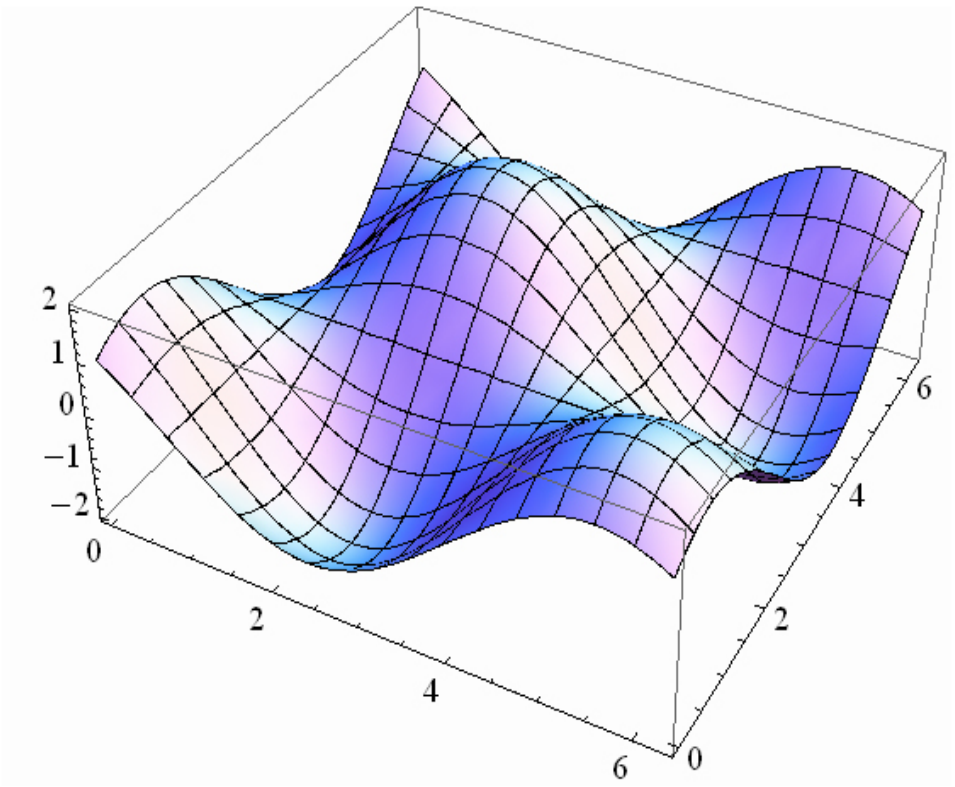
Out[1]=



Как видно из данного графика, в рассматриваемом случае хорошо наблюдаются положения максимумов и минимумов. Для сравнения построим график той же самой функции в обычных трехмерных координатах.

```
In[2]:= Plot3D[Cos[x + y] - Sin[x - y], {x, 0, 2 Pi},
         {y, 0, 2 Pi}]
```

Out[2]=

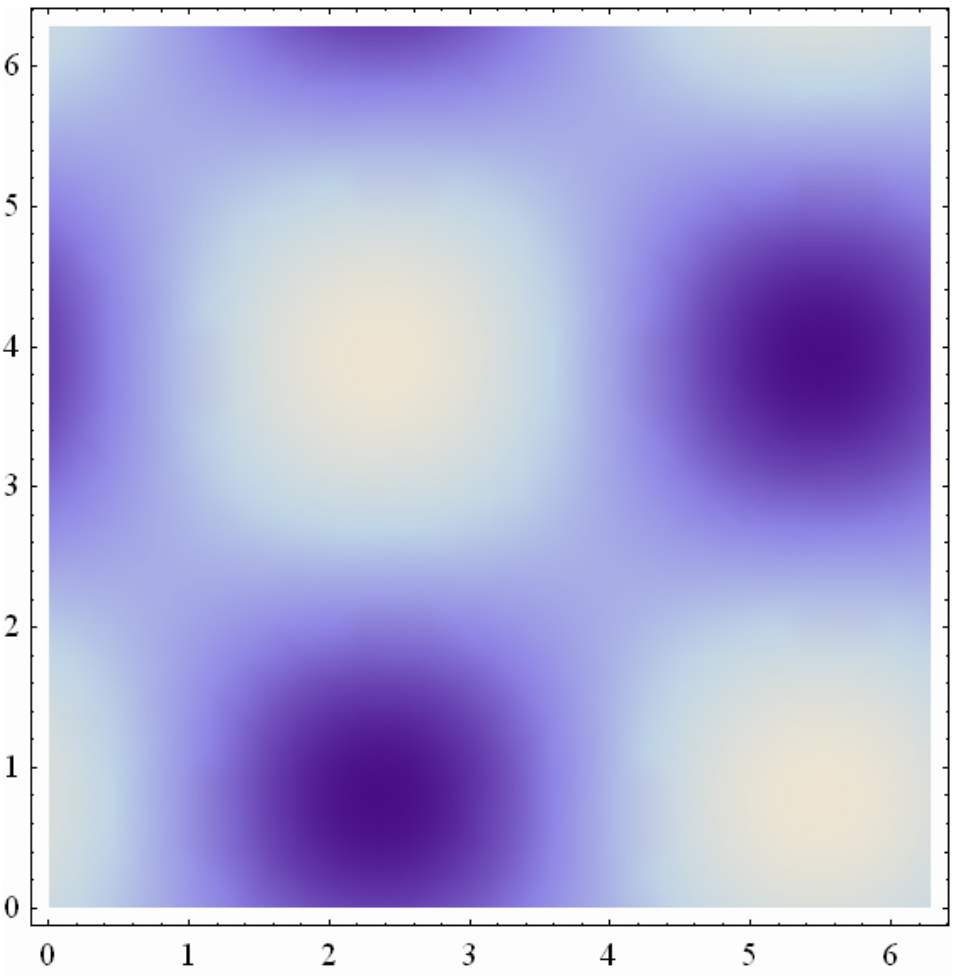


Как видно в данном случае положение максимумов и минимумов определить сложнее, но зато хорошо видно общее поведение функции.

Второй тип графиков – это *графики плотности*, построение которых возможно с помощью функции ***DensityPlot***. Данный тип графиков используется, в основном, в том случае, если некоторая функция двух переменных описывает плотность среды. Внешне график данного типа похож на контурный график, с тем отличием, что на данном графике участки не соединяются изолиниями. Для примера построим график плотности для рассмотренной ранее функции.

```
In[3]:= DensityPlot[Cos[x + y] - Sin[x - y],
         {x, 0, 2 Pi}, {y, 0, 2 Pi}]
```

Out[3]=



12.2. Опции построения графиков

Все рассмотренные ранее функции для построения графиков имеют достаточно большой набор функций, позволяющих изменять вид отображения графиков. Список опций можно узнать с помощью функции ***Options,*** в качестве параметра которой передается имя функции.

Вначале рассмотрим основные опции, используемые при построении двухмерных графиков. Например, выведем список опций, которые можно задать для функции ***Plot***[4].

[4] Стоит заметить, что в новой версии пакета количество опций, выводимых оператором ***Options***, значительно увеличилось.

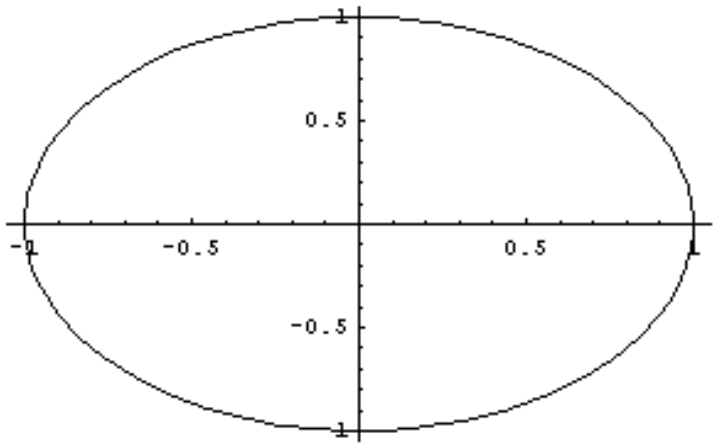
```
In[25]:= Options[Plot]

Out[25]= {AspectRatio → 1/GoldenRatio, Axes → Automatic,
          AxesLabel → None, AxesOrigin → Automatic,
          AxesStyle → Automatic, Background → Automatic,
          ColorOutput → Automatic, Compiled → True,
          DefaultColor → Automatic, DefaultFont :→ $DefaultFont,
          DisplayFunction :→ $DisplayFunction, Epilog → {},
          FormatType :→ $FormatType, Frame → False,
          FrameLabel → None, FrameStyle → Automatic,
          FrameTicks → Automatic, GridLines → None,
          ImageSize → Automatic, MaxBend → 10.,
          PlotDivision → 30., PlotLabel → None, PlotPoints → 25,
          PlotRange → Automatic, PlotRegion → Automatic,
          PlotStyle → Automatic, Prolog → {}, RotateLabel → True,
          TextStyle :→ $TextStyle, Ticks → Automatic}
```

Ранее, при построении, не пользовались опциями или, точнее будет сказать, мы использовали опции, установленные по умолчанию. В данном разделе мы рассмотрим некоторые опции при построении графиков.

Опции задаются в виде правил подстановок, например ***AspectRatio→Automatic***. По определению, данная опция задает отношение высоты к ширине двухмерного графика. Пользователь может установить его равным числу ***k***, набрав в качестве третьего аргумента выражение ***AspectRation→k***. По умолчанию величина данного параметра равна выражению ***1/GoldrnRatio***, где ***GoldenRatio*** есть константа «золотого сечения», приближенно равная 1.61803. Если установить данную опцию в ***Automatic,*** то величина данной опции будет определяться внутренними алгоритмами пакета *Mathematica*. Для иллюстрации возможностей работы данной опции попытаемся нарисовать окружность, заданную в параметрической форме, с помощью функции ***ParametricPlot***, без указания дополнительных опций, т. е. со значениями опций, заданных по умолчанию.

Из следующего графика видно, что получившееся изображение больше напоминает эллипс, чем круг.

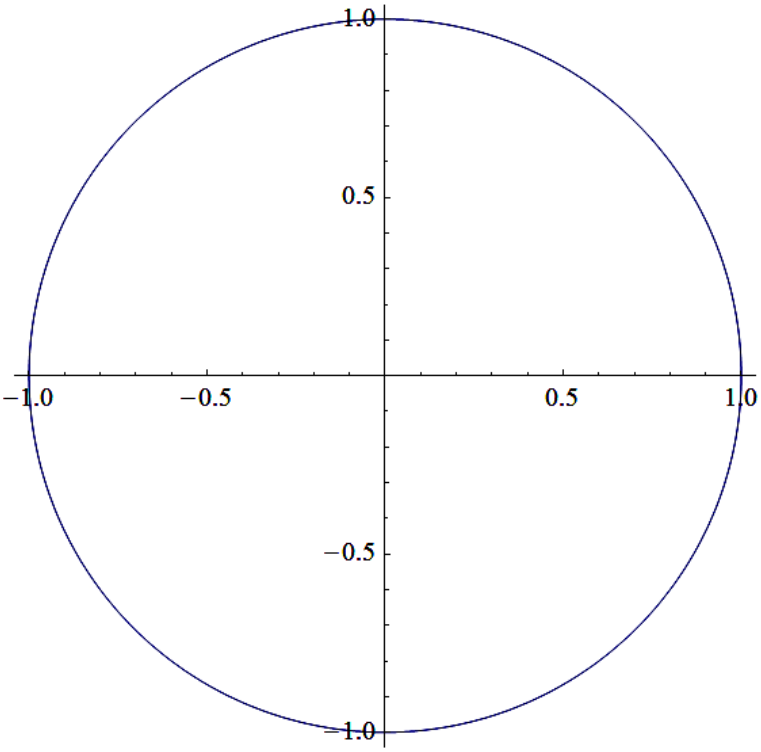
```
In[26]:= ParametricPlot[{Cos[t], Sin[t]}, {t, 0, 2 Pi}]
```




```
Out[26]= - Graphics -
```

Для выхода из данного положения построим график данной функции с параметром ***AspectRatio***, установленным в 1[5].

```
In[6]:= ParametricPlot[{Cos[t], Sin[t]},
        {t, 0, 2 Pi}, AspectRatio → 1]
```




```
Out[6]=
```

[5] Это свойственно только для версии пакета *Mathematica* ниже 6, так как в последней версии представления по умолчанию изменены, и опция ***AspectRatio*** изначально установлена в ***Automatic***, а не в ***1/GoldenRatio***.

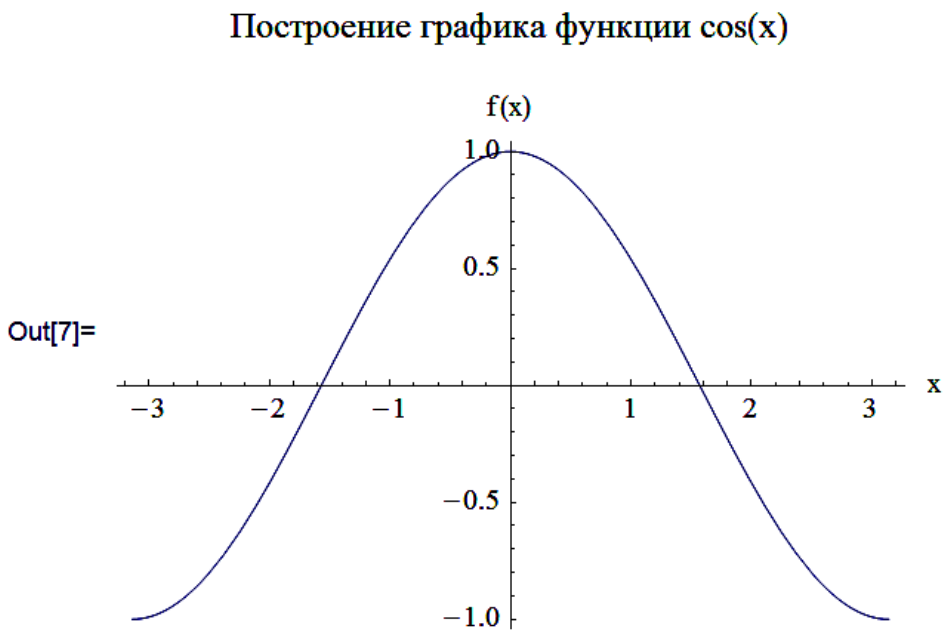
В данном случае, мы получили нормальный вид окружности. Стоит заметить, что во многих случаях достаточно установить значение данной опции в ***Automatic*** для нормального отображения графиков.

Опция ***Axes*** определяет, какие из координатных осей будут нарисованы при построении графика. Для данной опции возможны три установки: ***False***, при которой ни одна из осей не будет показана, ***True*** – обе оси будут отображены и ***{Boolean, Boolean}***, где ***Boolean*** принимает значения ***True*** или ***False***. Как видно из представленного ранее примера, по умолчанию данная опция установлена в ***Automatic***, в данном случае *Mathematica* сама решает, когда следует рисовать оси, а когда нет. Следует заметить, что для того чтобы убрать все оси можно вместо опции ***False*** воспользоваться ее аналогом – ***None***.

По умолчанию пакет *Mathematica* строит графики, не указывая надписей ни по осям координат, ни в верхней части графика (надпись, расположенная по центру, сверху, называется титульной). Для того чтобы создать подписи осей, необходимо воспользоваться опцией ***AxesLabel→{“подпись для оси x”,”подпись для оси y”}***. Для задания титульной надписи можно воспользоваться опцией ***PlotLabel,*** в качестве параметра которой задается надпись, взятая в кавычки. В качестве примера построим график косинуса, сделаем подписи к осям и титульную надпись.

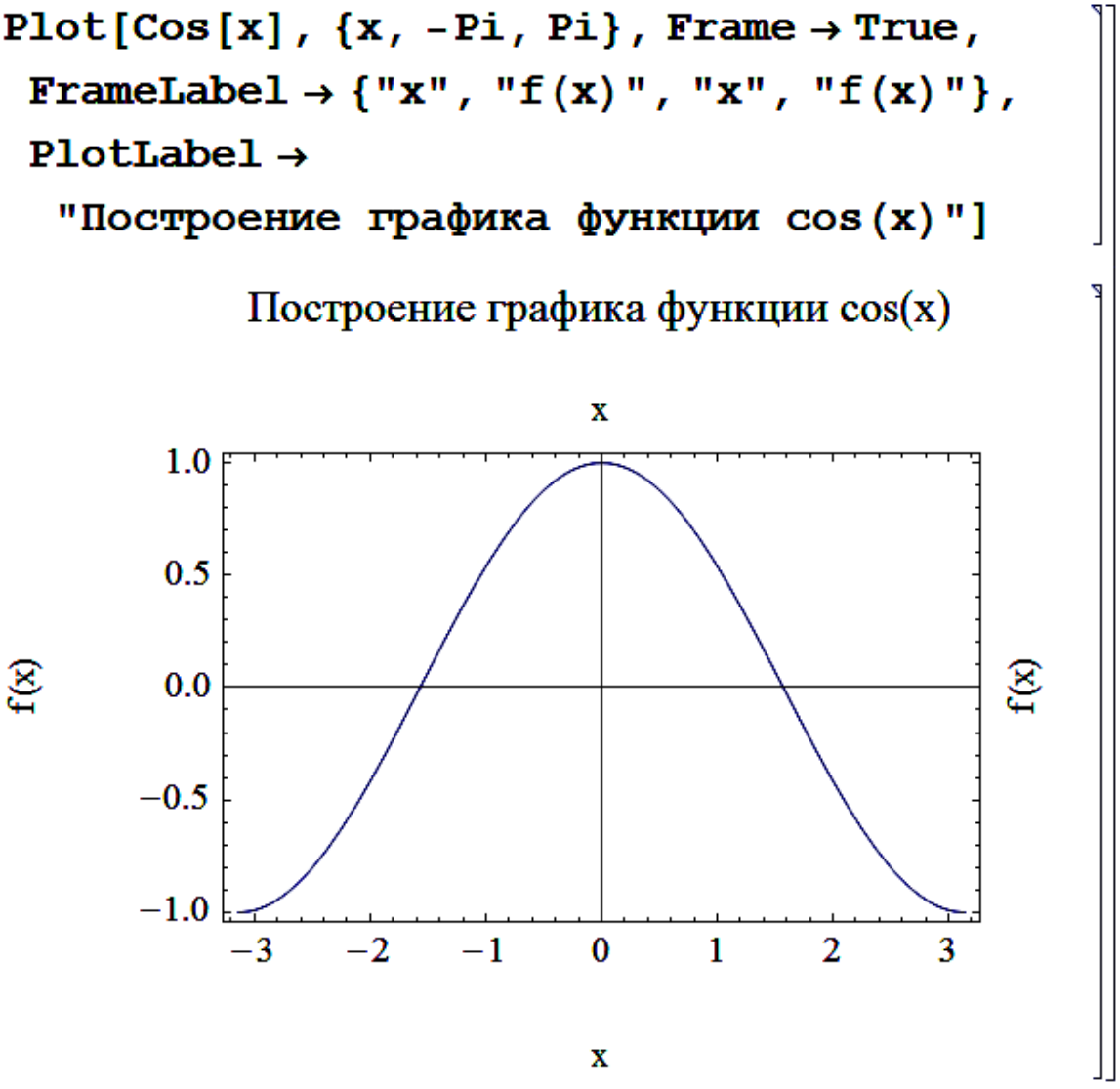
```
In[7]:= Plot[Cos[x], {x, -Pi, Pi},
         AxesLabel → {"x", "f(x)"},
         PlotLabel →
          "Построение графика функции cos(x)"]
```

Out[7]=



Как можно заметить из представленных примеров, в пакете *Mathematica* по умолчанию одномерные графики рисуются не совсем привычным образом, принятом при представлении графиков. В отличие от математического представления осей, когда линии координат проходят через 0, при построении графиков, чаще всего, значения осей перемещаются на прямоугольную область нарисованную вокруг графика. Для представления осей графика в такой форме можно воспользоваться опцией ***Frame,*** которая может принимать значения ***True,*** в этом случае прямоугольная рамка рисуется вокруг графика, и туда переносятся значения осей или ***False***, когда прямоугольная рамка не рисуется. Следует заметить, что при установленной опции ***Frame*** в ***True*** можно создать четыре подписи к осям, с помощью опции ***FrameLabel*** и четырех параметров данной опции.

```
Plot[Cos[x], {x, -Pi, Pi}, Frame → True,
 FrameLabel → {"x", "f(x)", "x", "f(x)"},
 PlotLabel →
  "Построение графика функции cos(x)"]
```

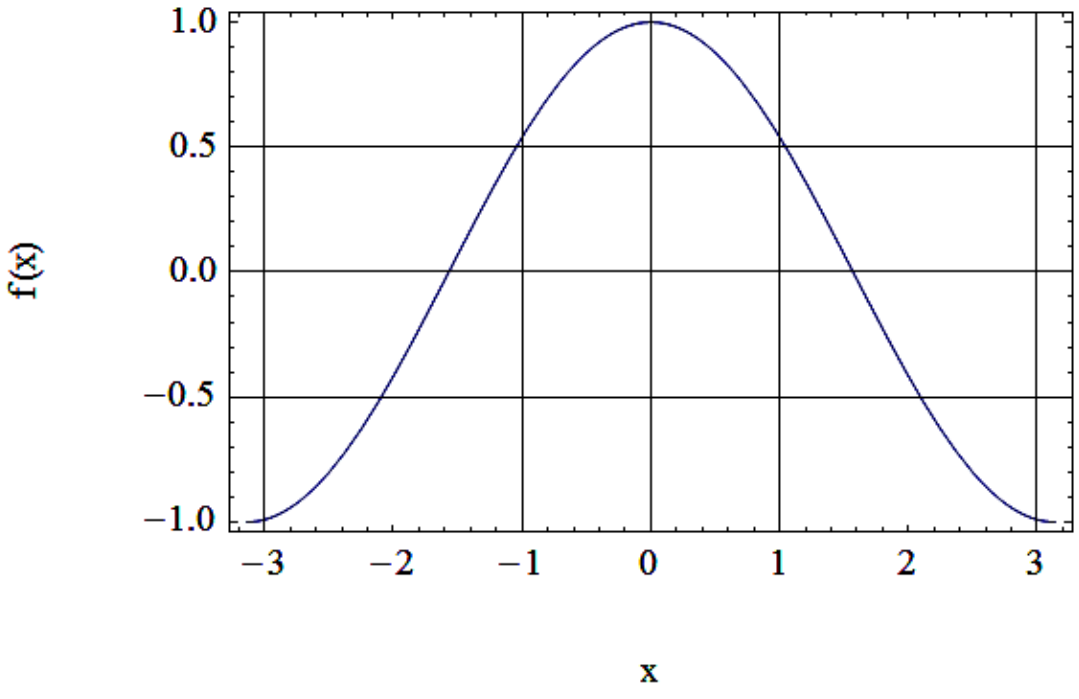


Как видно из данного примера, в этом случае график получился в более привычном виде и надписи, указанные в качестве параметров

данной функции располагаются по часовой стрелке, причем первая располагается снизу. Переделаем предыдущий график с использованием только что рассмотренных функций. Следует заметить, что в опции ***FrameLabel*** последние два параметра можно не указывать при построении графика. Из примера видно также, что подписи осей по умолчанию располагаются вертикально и читаются снизу вверх. Для того чтобы расположить подписи горизонтально, можно установить опцию ***RotateLabel*** в ***False***.

Чтобы график принял совсем законченный вид, отобразим на нем сетку. Для этого необходимо воспользоваться опцией ***GridLines***, значения которой могут быть следующими: ***None*** – не рисовать сетку, ***Automatic*** – рисовать сетку, расстояние между ячейками по x и по y выбирается автоматически, ***{xgrid, ygrid}***, где ***xgrid*** и ***ygrid*** задаются списками с координатами прохождения линий для оси x и y, соответственно. В качестве примера построим предыдущий график с отображенными линиями сетки. Причем по оси x расположим сетку вручную, а по оси y предоставим построение на усмотрение системы.

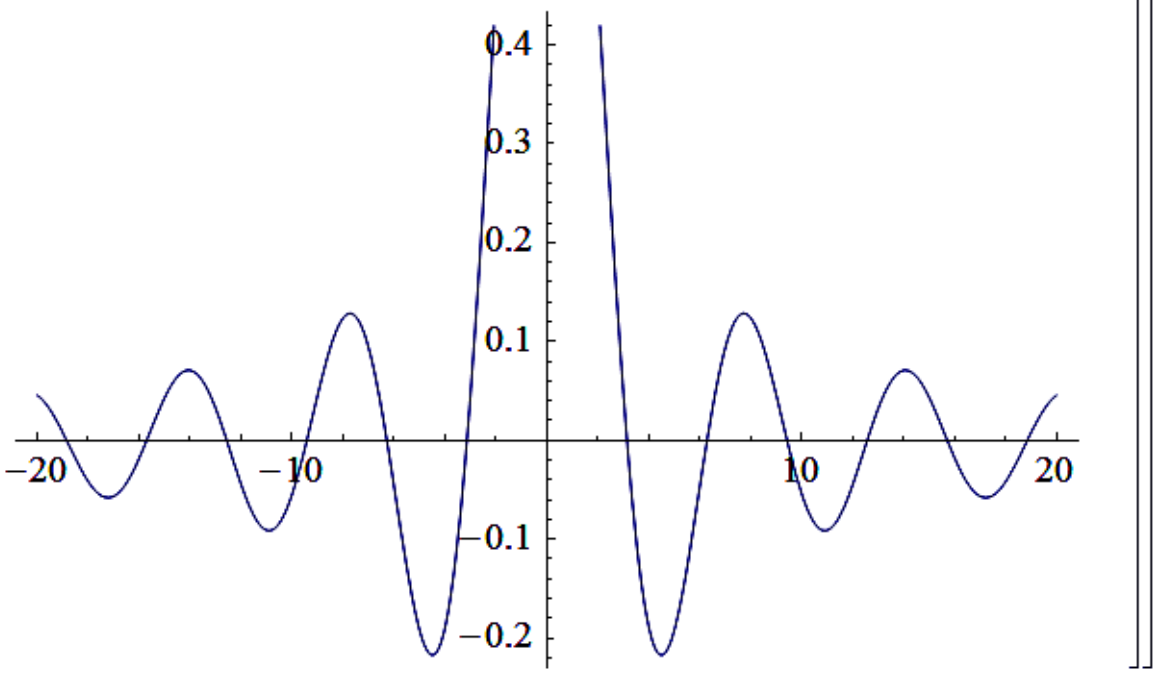
```
Plot[Cos[x], {x, -Pi, Pi}, Frame → True,
 FrameLabel → {"x", "f(x)"},
 GridLines → {{-3, -1, 1, 3}, Automatic}]
```



Следует заметить, что в качестве аргумента данной опции можно задать функцию, которая выдает значения в интервале изменения по обеим осям.

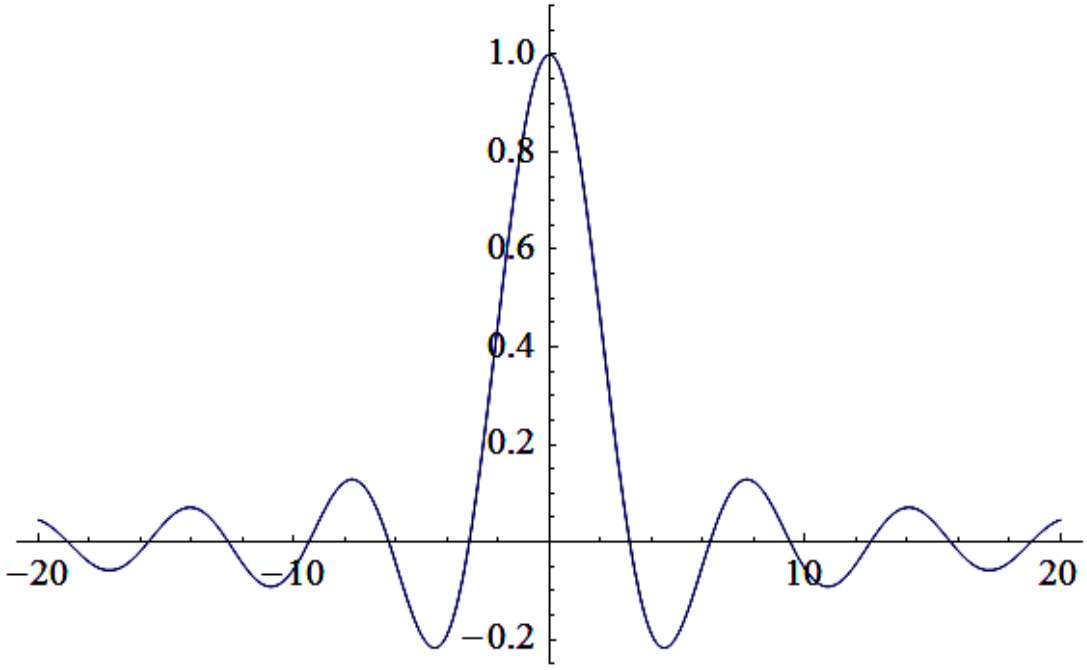
Иногда бывает, что при построении графика функции она отображается по оси ***y*** неполностью (чаще всего, это происходит с функциями, содержащими разрывы), например нарисуем график функции $\sin x/x$, где *x* изменяется в пределах от –20 до 20.

```
Plot[Sin[x] / x, {x, -20, 20}]
```



Как видно из данного примера, график в данном случае нарисован неполностью. Виной тому является то, что пакет не смог определить по умолчанию максимальные и минимальные значения для оси ***y***. Для того чтобы установить размеры самому, необходимо воспользоваться опцией ***PlotRange→{{xmin,xmax},{ymin,ymax}}***. Причем, если вам нужно установить максимальное и минимальное значение шкалы только для оси ***y***, то в этом случае можно воспользоваться упрощенной записью ***PlotRange→{ymin,ymax}***. Теперь попробуем построить этот график более корректно, с использованием данной функции.

```
Plot[Sin[x] / x, {x, -20, 20},
 PlotRange -> {-0.25, 1.1}]
```



Как видно, в данном случае график функции нарисован нормально, т. е. он поместился полностью.

До сих пор все двухмерные графики отображались в черно-белом варианте, за исключением случая, когда рисовалась сетка[6]. В пакете *Mathematica* есть набор функций, позволяющих задавать стили объектов, а именно цвета, толщины и тип линий, размеры точек. В табл. 7 приведены основные функции, с помощью которых можно составлять стили практически любых объектов.

[6] *Mathematica* 6 с графиками изначально поступает умнее и раскрашивает их автоматически в разные цвета.

Таблица 7

Основные функции для задания цветов объектов, толщин и типов линий и размера точек

Функция	***Описание***
Задание цвета	
GreyLevel[g]	данная функция позволяет задать цвет в оттенках серого цвета. Величина g принимает значения от 0 до 1 и характеризует степень насыщенности черным цветом
RGBColor[r,g,b]	функция позволяет задать цвет путем смеси трех основных цветов: красного (r), зеленого (g) и голубого (b). Все аргументы могут принимать значение от 0 до 1 и задают степень насыщенности соответствующим цветом
Hue[h]	данная функция задает цвет из палитры, в зависимости от параметра h, который может принимать значения от 0 до 1. Следует заметить, что палитра на разных компьютерах зависит от глубины цвета, установленного на компьютере, и может отличаться. В данной функции можно задать еще два аргумента, которые будут отвечать за насыщенность и яркость выбранного цвета
CMYKColor[c,m,y,k]	в отличие от ***RGBColor***, данная функция позволяет получить цвет путем смеси четырех цветов: синего (c), пурпурного (m), желтого (y) и черного (k)
Задание толщины и типа линий	
Thickness[t]	задает толщину линии в ***t*** (***t<<1***). Следует заметить, что в данном случае толщина линии будет зависеть от размера графического объекта, на котором отображается линия. Для того чтобы задать толщину линии в абсолютных единицах, необходимо воспользоваться функцией ***AbsoluteThickness[t]***. В данном случае параметр ***t*** должен быть больше 1
Dashing[{l1,l2,l3,…}]	с помощью данной функции можно рисовать линии разного типа по шаблону. Шаблон составляется по аргументам функции (***li***), где первый элемент отвечает за закрашенную область, второй – за незакрашенную и т. д. После задания шаблона он будет повторяться циклически на протяжении всей линии. Например, чтобы задать штрих пунктирную линию можно написать ***Dashing[{0.02,0.02,0.005, 0.02}]***. Следует заметить, что в данном случае длина отрезков задается в относительных единицах. Для того чтобы получить ее в абсолютных единицах нужно воспользоваться соответствующим аналогом
Задание размера точек	
PointSize[r]	данная функция задает размер точек при построении графиков по точкам в относительных единицах. Для задания точек в абсолютных единицах необходимо воспользоваться функцией ***AbolutePointSize[r]***

На графиках с помощью данных функций можно задавать значения опций, приведенных в табл. 8.

Таблица 8

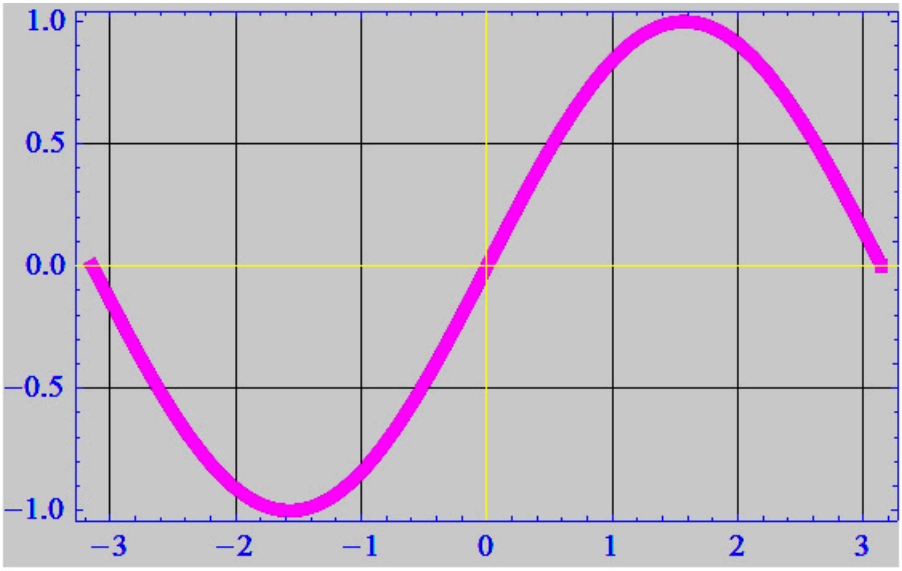
Основные опции изменения стилей отображения графиков

Опция	***Описание***
AxesStyle	позволяет задать цвет и тип линии для осей
FrameStyle	аналогична ***AxesStyle***, но позволяет задавать стиль для прямоугольной области вокруг графика
PlotStyle	позволяет задать цвет и тип линий при рисовании графика функции
Background	с помощью данной опции можно задать цвет заднего фона, на котором отображается график
DefaultColor	позволяет установить цвет по умолчанию.

Для примера построим график функции $\sin x$, в котором зададим цвет линии, ее толщину, а также цвет фона, прямоугольной области вокруг графика и осей, и с помощью опции ***TextStyle*** зададим вид отображаемого текста. После набора команды:

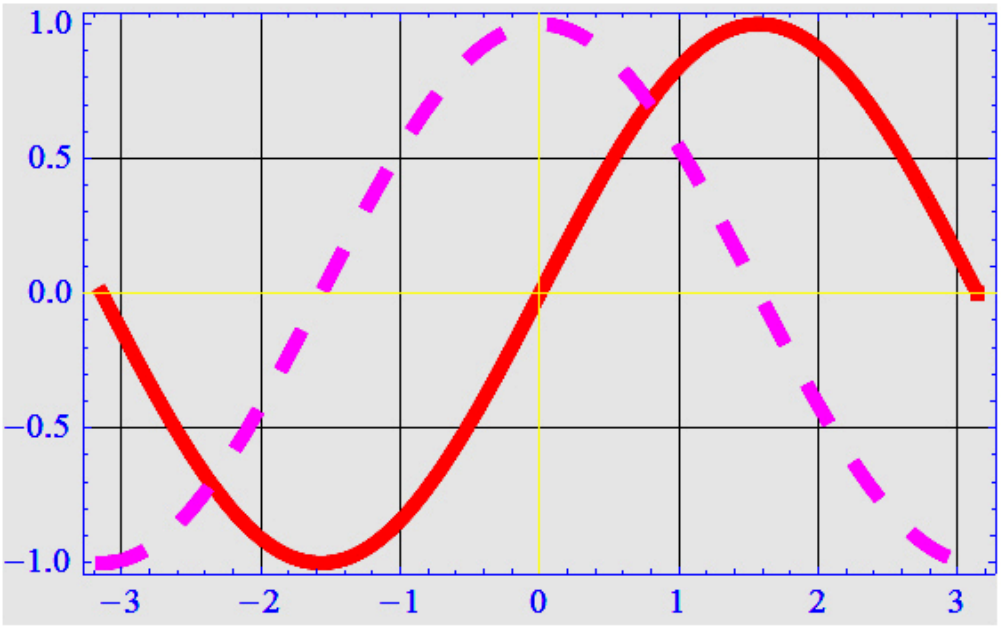
```
Plot[Sin[x], {x, -Pi, Pi}, Frame → True,
 GridLines → Automatic,
 PlotStyle → {Thickness[0.015],
   RGBColor[1, 0, 1]},
 Background → GrayLevel[0.95],
 FrameStyle -> RGBColor[0, 0, 1],
 AxesStyle → CMYKColor[0, 0, 1, 0]]
```

Мы получим следующий график функции:



Как видно из данного примера, при одновременном задании стиля и цвета, для отображения графика функции, их необходимо указывать в списке. Если графиков функций несколько и для каждого из них предполагается задание стиля, то в данном случае необходимо воспользоваться списком. В качестве примера добавим к предыдущему графику график функции $\cos x$, который нарисуем прерывистой линией пурпурного цвета с заданной толщиной.

```
Plot[{Sin[x], Cos[x]}, {x, -Pi, Pi},
 Frame → True, GridLines → Automatic,
 PlotStyle →
  {{Thickness[0.015], RGBColor[1, 0, 0]},
   {Thickness[0.015],
    Dashing[{0.05, 0.05}],
    CMYKColor[0, 1, 0, 0]}},
 Background → GrayLevel[0.95],
 FrameStyle -> RGBColor[0, 0, 1],
 AxesStyle → CMYKColor[0, 0, 1, 0]]
```



Иногда бывает необходимо не выводить сам график при выполнении функции, а только присвоить полученный результат, в виде графического объекта, какой-нибудь переменной для дальнейшего использования, на-

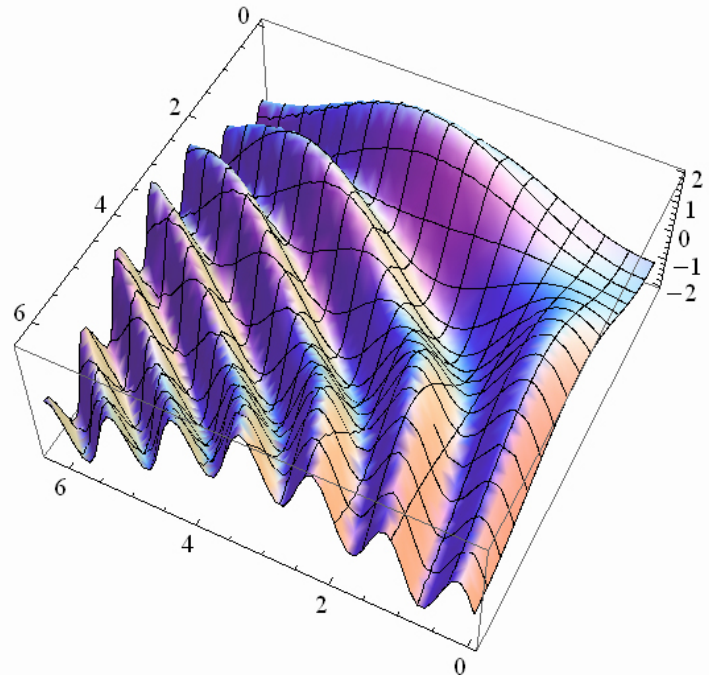
пример наложения нескольких графиков с различными параметрами на один, с помощью функции ***Show***. Для этого можно воспользоваться опцией ***DisplayFunction→Identity***, которая позволяет присвоить объект какой-нибудь переменной, но при этом не отображает сам график.

Теперь рассмотрим основные опции форматирования для функций, предназначенных для построения трехмерных графиков.

Вначале стоит отметить, что большинство из опций, рассмотренных ранее, подходят и для функций, строящих трехмерные графики, например ***AxesLabel***. Только при использовании данной опции для этих функций необходимо добавить третий параметр. Для данной опции он будет задавать подпись для третьей оси координат. Для получения более подробной информации можно воспользоваться функцией ***Options***, чтобы посмотреть, какие опции устанавливаются для данного типа графиков, а потом посмотреть в справке пакета или в литературе их описания.

Одной из часто использующихся опций при построении трехмерных графиков является опция ***ViewPoint→{x,y,z}***, которая позволяет задавать точку, из которой рассматривается рисуемая поверхность. Например, построим график функции $\sin(xy) - \cos(x - y)$ и изменим точку обзора[7].

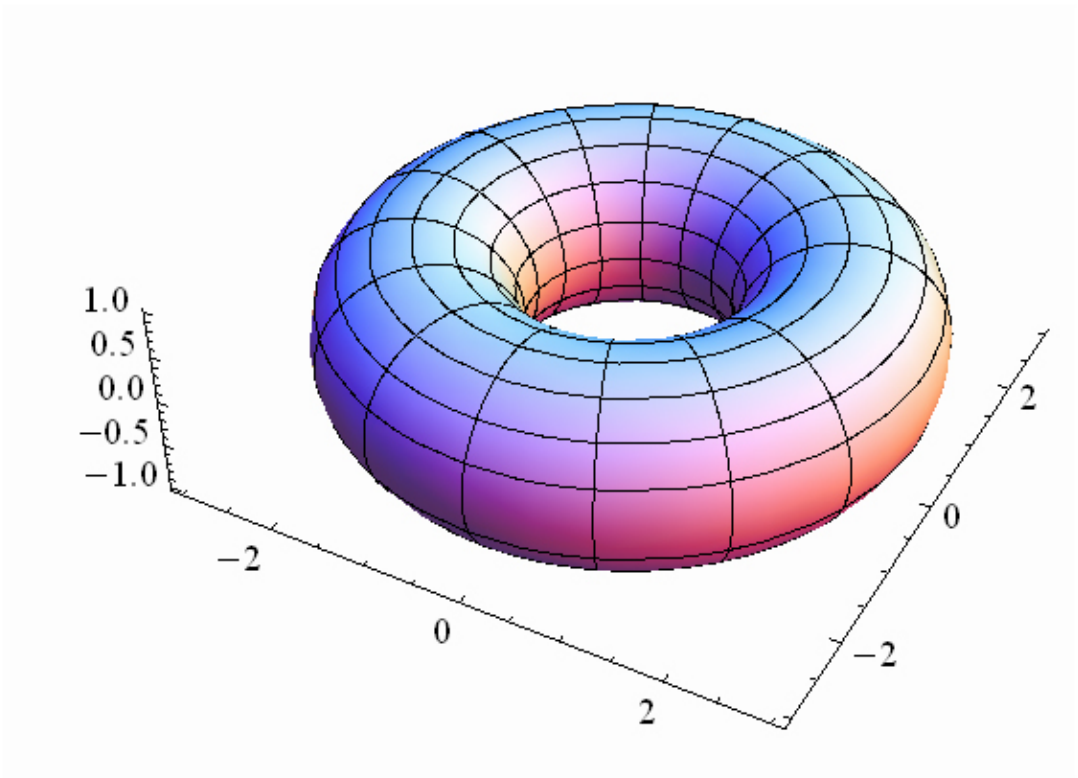
```
Plot3D[Sin[x y] - Cos[x - y], {x, 0, 2 Pi},
 {y, 0, 2 Pi}, ViewPoint → {-1, 2, 3}]
```



[7] Начиная с версии 6 пакета *Mathematica*, трехмерные графики можно просто вращать мышью и подбирать угол зрения без использования специальных функций.

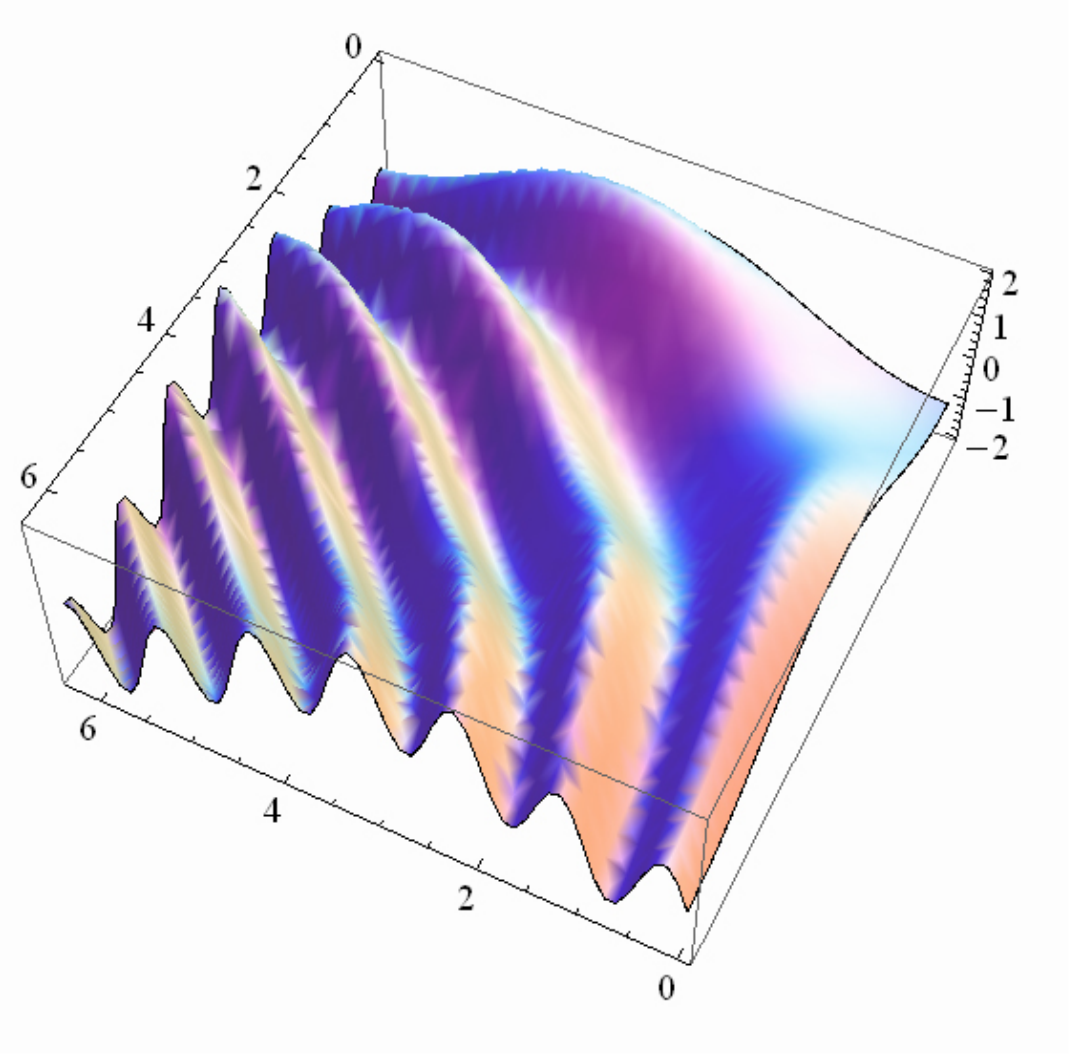
С помощью опции ***Boxed*** можно построить график функции, на которой не отображаются параллелепипед, обрамляющий график функции, например,

```
ParametricPlot3D[
 {Cos[t] (2 + Cos[u]), Sin[t] (2 + Cos[u]),
  Sin[u]}, {t, 0, 2 Pi}, {u, 0, 2 Pi},
 Boxed -> False]
```



С помощью установленной в ***False*** опции ***Mesh*** можно избавиться от линий каркаса фигуры. В таком виде некоторые поверхности могут выглядеть более естественно.

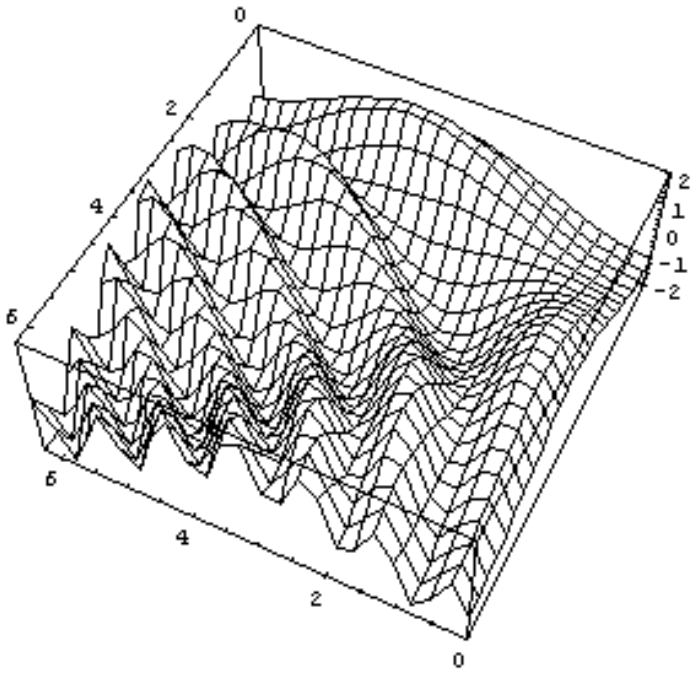
```
Plot3D[Sin[x y] - Cos[x - y], {x, 0, 2 Pi},
 {y, 0, 2 Pi}, ViewPoint → {-1, 2, 3},
 Mesh → False]
```



С помощью опции ***Shading*** можно убрать закрашивание поверхности и представить график в виде сетчатой поверхности[8].

[8] Стоить заметить, что в 6-й версии пакета данная функция заменена функцией ***ColorFunction***, позволяющей производить различные способы раскраски поверхности в трехмерных графиках.

```
In[9]:= Plot3D[Sin[x y] - Cos[x - y], {x, 0, 2 Pi}, {y, 0, 2 Pi},
          ViewPoint → {-1, 2, 3}, Shading → False]
```




```
Out[9]= - SurfaceGraphics -
```

Для более подробной информации по этим и другим опциям можно обратиться к литературе или справке пакета *Mathematica*.

12.3. Построение графиков функций, заданных в виде таблиц

Далеко не все задачи можно решить в аналитическом виде. В этом случае на помощь приходят численные методы. Результатом таких вычислений являются табличные данные, и было бы не плохо иметь инструмент для построения графиков по полученным результатам. Пакет *Mathematica* имеет несколько встроенных функций, аналогичных рассмотренным ранее функциям. В табл. 9 приведены основные функции для построения графиков по точкам.

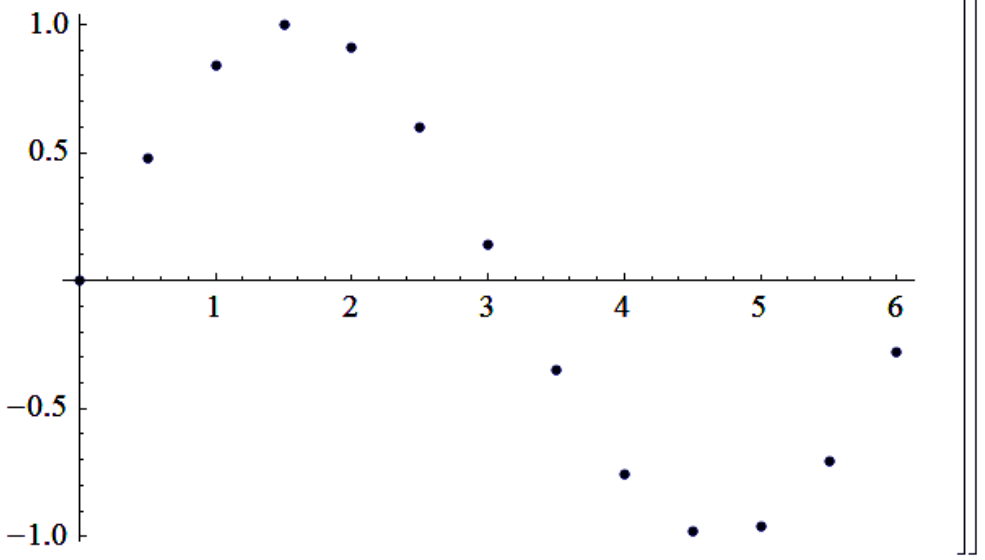
Таблица 9

Основные функции для визуализации табличных данных

Функция	***Описание***
ListPlot[список]	построение графика по табличным данным. Если в качестве параметра передается обычный список, то его значения будут взяты в качестве координат по ***y*** при построении графика, а координаты ***x*** будут заданы из списка 1, 2, 3,… Если же в качестве параметра будет передана матрица, состоящая из двух строк, то данные из первого столбца будут взяты в качестве ***x***-координат, а второго столбца, соответственно, в качестве ***y***-координат для точек
ListPlot3D[матрица]	по данным, представленным в матрице с тремя столбцами, строится поверхность, где столбцы задают значения ***x,y*** и ***z***-координат
ListContourPlot[матрица]	по данным, представленным в матрице с тремя столбцами, строится контурный график
ListDensityPlot[матрица]	по данным, представленным в матрице с тремя столбцами, строится график плотности

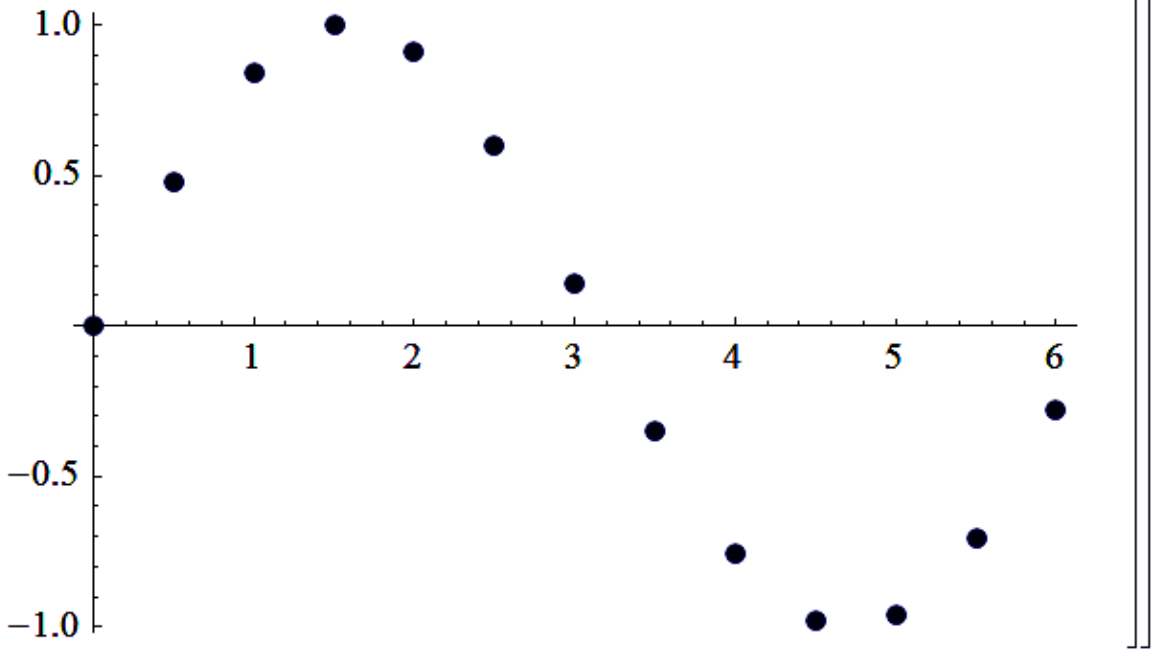
Для примера построим таблицу из функции $f(x) = \sin x^2$, и по ней построим двухмерный график:

```
s = Table[{x, Sin[x]}, {x, 0, 2 Pi, 0.5}];
ListPlot[s]
```



Как и размер линий при построении графиков, размер точек тоже может быть изменен. Для увеличения размеров точек можно воспользоваться уже знакомой опцией ***PlotStyle,*** задав значение размера точек с помощью функции ***PointSize***. Изменим в полученном графике размер точек:

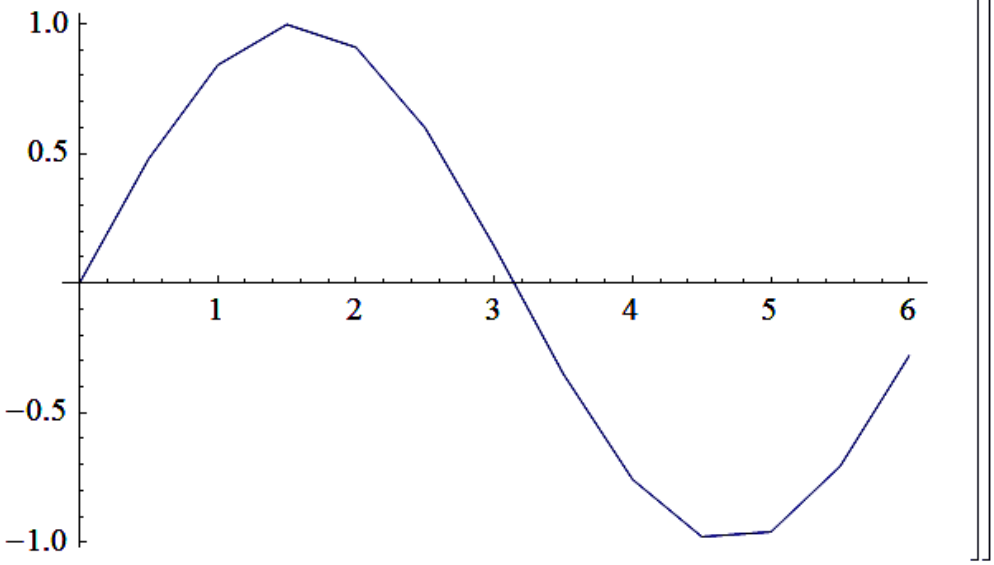
```
s = Table[{x, Sin[x]}, {x, 0, 2 Pi, 0.5}];
ListPlot[s, PlotStyle → PointSize[0.02]]
```



В данном случае точки видны лучше, но достаточно сложно понять вид функции. Для того чтобы получить более подробное представление о виде функции, можно установить опцию ***PlotJoined*** в ***True***.[9] При установлении данной опции точки соединяются между собой прямыми линиями. На следующем примере показана работа данной функции.

[9] В 6-й версии пакета имя данной опции изменено на просто ***Joined***. Так же введен специальный тип графика ***ListLinePlot***, который по умолчанию строит график, в котором точки соединяются линиями.

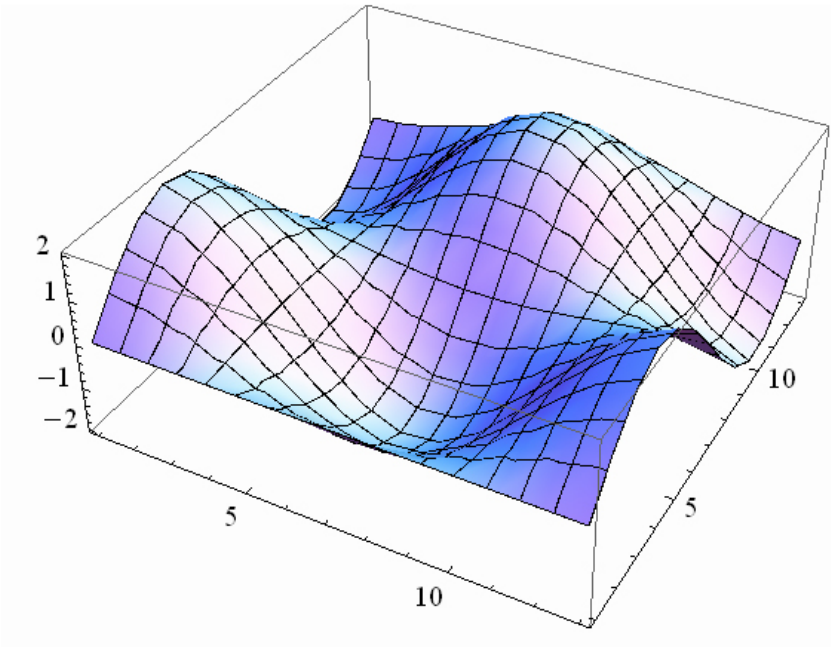
```
ListPlot[s, PlotJoined -> True]
```



Следует заметить, что большинство опций форматирования графиков, рассмотренных ранее, применимы и для данного типа графиков.

Для примера покажем построенный по точкам график функции $f(x) = \sin(x - y) + \sin(x + y)$.

```
s = Table[Sin[x - y] + Sin[x + y],
    {x, 0, 2 Pi, 0.5}, {y, 0, 2 Pi, 0.5}];
ListPlot3D[s]
```



В заключение данного раздела хочется сказать, что описанные здесь возможности по построению графиков – лишь малая часть всех возможностей пакета *Mathematica*. Помимо встроенных типов графиков, в дополнительном пакете расширения ***Graphics***[10] содержится намного большее количество функций для построения других типов графиков. Более подробно с ними можно ознакомиться в справке пакета или посмотреть в литературе.

Задачи и упражнения

1. Построить графики функций:
   a) $\ln(x) \cdot \dfrac{\sin(3x)}{\exp(-x)}$;
   b) $\begin{cases} x = 5\sin t \\ y = 7\cos t \end{cases}$;
   c) $\rho = 3(1 - \cos\varphi)$ (при построении данного графика воспользоваться функцией ***PolarPlot***).
2. Найти, если существуют, первые три корня трансцендентных уравнений графическим способом:
   a) $x - 3\ln x - 1 = 0$;
   b) $2\sin^2 x - \sin x = 1$;
   c) $x^2 - \exp(x) = \lg x$.
3. Построить график функции на основе подготовленного текстового файла, содержащего две колонки цифр.
4. Исследовать и построить график функции $\dfrac{3x^2 + 7}{5 - 2x}$.

13. РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ И СИСТЕМ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Из курсов физики и высшей математики известно, что большинство процессов, происходящих в реальной жизни, можно описать с помощью дифференциальных уравнений. Поэтому решение многих задач сводится к решению соответствующих дифференциальных уравнений, например уравнений диффузии и теплопроводности. Проблема заключается в том, что многие уравнения достаточно сложны для ручного аналитиче-

[10] В 6-й версии пакета ***Graphics*** разбили на несколько частей: ***BarCharts***, ***Histograms***, ***PieCharts*** и т. д.

ского решения, а решение многих, в аналитическом, виде затруднены. Для последних уравнений разработаны специальные численные методы, позволяющие получить численное решение уравнений для определенных условий, аналитическое решение которых затруднительно или невозможно. Пакет *Mathematica* позволяет расширить класс решаемых уравнений и сократить время на поиск решения. Кроме того, пакет позволяет, без глубокого знания численных методов, получить численное решение достаточно сложных уравнений за приемлемое время, которое естественно зависит от мощности компьютера, на котором проводятся вычисления.

Для решения дифференциальных уравнений в пакете *Mathematica* существует функция вида:

DSolve[уравнение, y,x][11]***,***

которая позволяет решать дифференциальное уравнение для переменой ***y***, которая, в свою очередь, зависит от независимой переменной ***x***.

Далее на примерах рассмотрим использование данной функции в различных случаях.

13.1. Получение решений в общем виде

Иногда бывает необходимо получить общее решение уравнения с использованием констант дифференцирования. В качестве примера решим уравнение $x'(t) = k \cdot x(t)$.

```
In[45]:= DSolve[D[x[t], t] == k * x[t], x[t], t]

Out[45]=
{{x[t] → e^(k t) C[1]}}
```

Как видно из данного примера, решение получается в виде подстановки (замены переменных) с соответствующей константой дифференцирования[12]. Аналогично функции ***Solve***, при записи уравнения в функции ***Dsolve***, необходимо следить за тем, чтобы в описании уравнения стояла **операция сравнения** (==). Ни в коем случае не стоит писать там операцию присваивания (=) – это может привести к трудноуловимым ошибкам.

Следует заметить, что производная в приведенном уравнении в функции ***DSolve*** может быть записана двумя способами: с использованием функции ***D***, как было показано выше, и с помощью сокращенной записи производной:

---

[11] Имя данной функции получается комбинацией имен двух функций: ***D*** – вычисление производных и ***Solve*** – решения линейных уравнений.

[12] Количество констант, естественно, зависит от степени дифференциального уравнения. Так для уравнения со второй производной таких констант должно быть две, для уравнения с третьей производной – три и т. д.

```
In[46]:= DSolve[x'[t] == k * x[t], x[t], t]

Out[46]=
{{x[t] → e^(k t) C[1]}}
```

Но стоит помнить, что некоторые уравнения могут быть решены только при использовании первого варианта записи.

13.2. Решение уравнений с начальными условиями

Решение уравнений в общем виде, чаще всего, представляет лишь академический интерес, а для решения реальных задач необходимо решение с начальными условиями. Для примера рассмотрим решение уравнения $x''(t) = 0$, где координата в начальный момент времени равно ***x0***, а начальная скорость – ***v0***[13]. В результате решения получим

```
In[47]:= DSolve[{x''[t] == 0, x[0] == x0, x'[0] == v0},
           x[t], t]

Out[47]=
{{x[t] → t v0 + x0}}
```

Как и следовало ожидать, при движении с нулевым ускорением, которое описывается этим уравнением[14], получили решение для движения с постоянной скоростью, зависящей только от начальной скорости и начальных координат. Как видно из данного примера, начальные условия задаются как дополнительные уравнения в первом аргументе функции ***DSolve***. Так как аргументов в данной функции должно быть 3, то уравнение и начальные условия записываются в виде списка, т. е. взятыми в фигурные скобки.

Аналогичным образом могут быть получены решения более сложных дифференциальных уравнений.

13.3. Решение систем дифференциальных уравнений

Как можно понять из предыдущего раздела, решение систем дифференциальных уравнений заключается в простом добавлении необходимых уравнений в список функции ***DSolve***. Кроме того, второй аргумент тоже станет списком, в котором необходимо указать, относительно каких переменных будет решаться уравнение. В качестве примера решим систему вида:

$$\begin{cases} x'(t) = y(t) \\ y'(t) = x(t) \end{cases}.$$

[13] Напомним, что в дифференциальном исчислении скорость – это первая производная по времени.
[14] В данном случае предполагается, что масса равна 1 и нет силы трения.

В результате получим:

```
In[48]:= DSolve[{x'[t] == y[t], y'[t] == x[t]},
           {x[t], y[t]}, t]
```

Out[48]=

$$\left\{\left\{x[t] \to \frac{1}{2} e^{-t}\left(1+e^{2t}\right) C[1] + \frac{1}{2} e^{-t}\left(-1+e^{2t}\right) C[2],\ y[t] \to \frac{1}{2} e^{-t}\left(-1+e^{2t}\right) C[1] + \frac{1}{2} e^{-t}\left(1+e^{2t}\right) C[2]\right\}\right\}$$

В данном примере получено решение в общем виде. Для того чтобы получить частное, решение зададим начальные условия $x(0) = x0,\ y(0) = y0$.[15] Решение этой системы в данном случае будет выглядеть следующим образом:

```
In[49]:= DSolve[{x'[t] == y[t], y'[t] == x[t], x[0] == x0,
           y[0] == y0}, {x[t], y[t]}, t]
```

Out[49]=

$$\left\{\left\{x[t] \to \frac{1}{2} e^{-t}\left(x0 + e^{2t}\, x0 - y0 + e^{2t}\, y0\right),\ y[t] \to \frac{1}{2} e^{-t}\left(-x0 + e^{2t}\, x0 + y0 + e^{2t}\, y0\right)\right\}\right\}$$

В большинстве случаев решение дифференциальных уравнений получается слишком громоздким и поэтому результат работы функции ***DSolve*** полезно упрощать с помощью функций ***Simplify*** или ***FullSimplify***.

```
In[50]:= DSolve[{x'[t] == y[t], y'[t] == x[t], x[0] == x0,
           y[0] == y0}, {x[t], y[t]}, t] // FullSimplify
```

Out[50]=

```
{{x[t] → x0 Cosh[t] + y0 Sinh[t],
  y[t] → y0 Cosh[t] + x0 Sinh[t]}}
```

Как видно из данного примера, решение в данном случае получается более простым, но при использовании функций упрощения следует помнить, что время, затрачиваемое на решение, может значительно увеличиться (особенно это качается функции ***FullSimplify***).

---

[15] Начальные условия задаются аналогичным образом, как и в случае решения уравнений с начальными условиями.

13.4. Численное решение дифференциальных уравнений

Наверное, ни для кого связанного с математикой не будет секретом то, что лишь малая часть дифференциальных уравнений может быть решена аналитически. Пакет Mathematica позволяет увеличить это количество, но и то не значительно. Этот пакет может решать аналитически большинство уравнений линейных дифференциальных уравнений с постоянными коэффициентами и некоторое количество уравнений с переменными коэффициентами. Кроме того, пакет может решить определенный класс нелинейных уравнений. Остальные уравнения и системы уравнений могут быть решены с использованием численных методов[16].

Для решения дифференциальных уравнений используется функция:

NDSolve[уравнение, y,{x,x_{min},x_{max}}]

Видно, что аргументы функции ***NDSolve*** практически аналогичны функции ***DSolve***, за исключением последнего аргумента, который задается в виде списка и указывает в каком диапазоне, в данном случае – иксов, необходимо искать решение уравнение. Рассмотрим на примере численное решение уравнения первого порядка $y''(x) = 2y(x)$ с граничными условиями[17] $y(0) = 0,\ y'(0) = -0.05$. Найдем решение в интервале иксов от 0 до 5:

```
In[51]:= sol1 =
          NDSolve[{y''[x] == 2 y[x], y[0] == 0,
            y'[0] == -0.05}, y, {x, 0, 5}]

Out[51]=
         {{y → InterpolatingFunction[{{0., 5.}}, <>]}}
```

В результате получим решение в виде специальной интерполирующей функции, которую можно использовать практически как обычную функцию, например для проверки начального условия:

```
In[52]:= y[0] /. sol1[[1, 1]]

Out[52]=
         3.70577 × 10^-22
```

которое в данном случае можно считать верным, так как полученный результат очень близок к нулю.

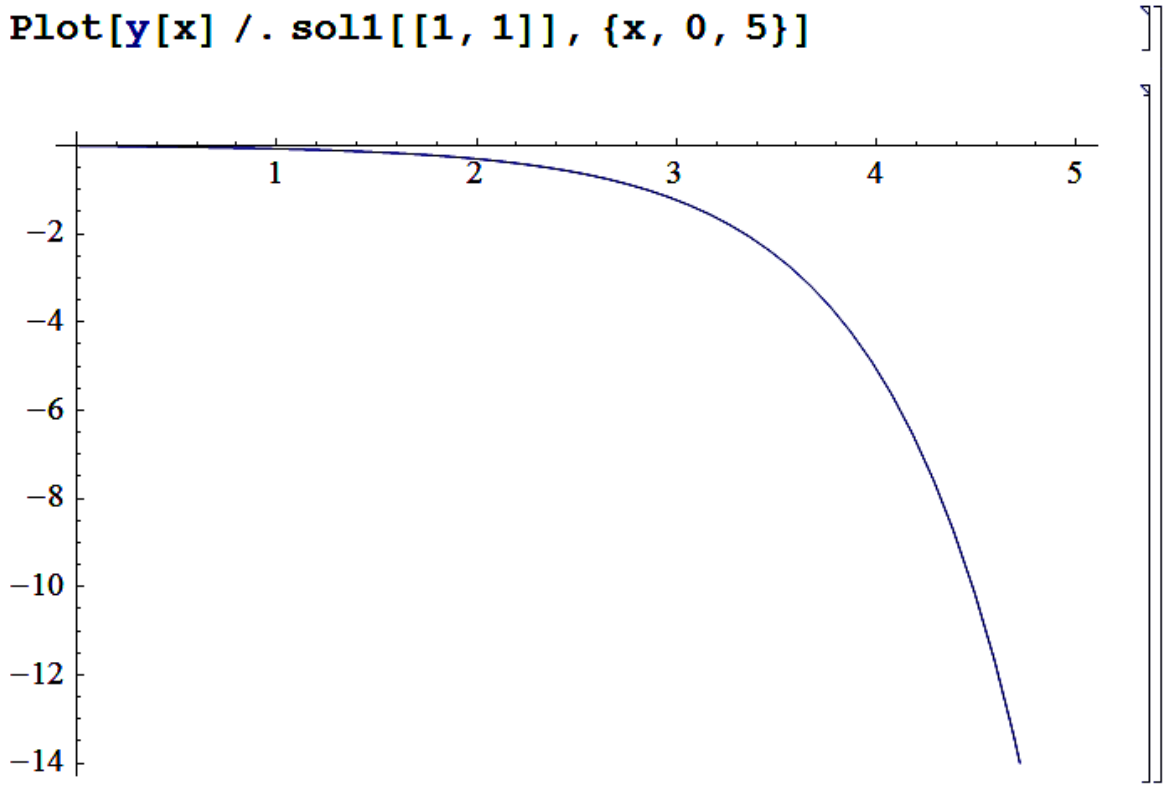
---

[16] Стоит заметить, что часть уравнений, решение которых в аналитическом виде может быть получено, тоже лучше решать численно, так как время, затраченное на аналитическое решение, может быть существенно больше, чем при численном решении.

[17] Следует помнить, что при численном решении уравнения граничные условия могут быть заданы только в виде численных значений и не могут быть заданы в виде символов.

Используя полученный результат, можно также построить график поведения функции в рассматриваемом диапазоне значений, как показано на следующем примере:

```
Plot[y[x] /. sol1[[1, 1]], {x, 0, 5}]
```



При использовании результата решений следует помнить, что результат определен только в интервале чисел, который был задан при решении уравнения. Для других значений результат хоть может и выводится, но является неверным.

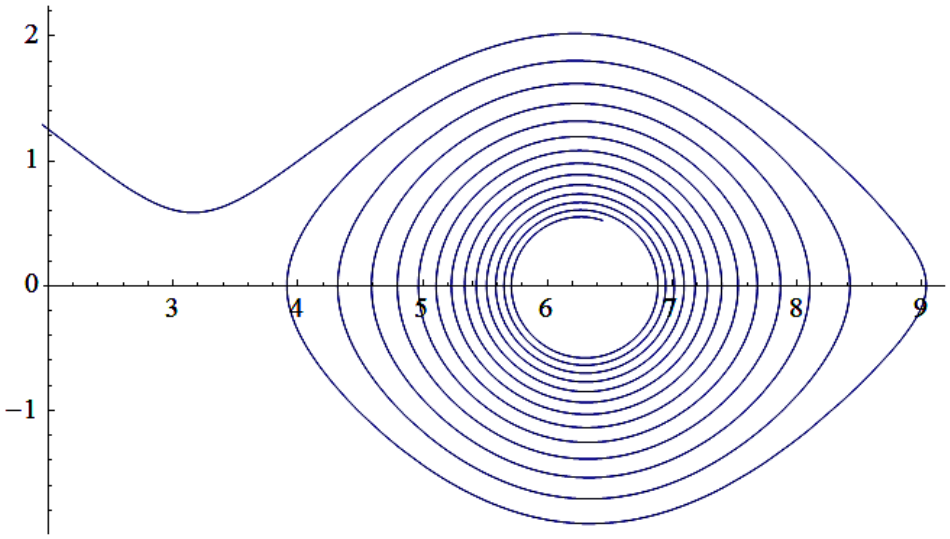
Аналогичным образом можно решать системы уравнений. Например, решим систему уравнений:

$$\begin{cases} x'(t) = y(t) \\ y'(t) = -0.03y(t) - \sin x(t)' \end{cases}$$

для граничных условий: $x(0) = 0,\ y(0) = 2.15,\ \mathrm{x} \in (0{,}100)$ и построим график функции:

```
In[56]:= sol2 =
          NDSolve[{x'[t] == y[t],
            y'[t] == -0.03` y[t] - Sin[x[t]], x[0] == 0,
            y[0] == 2.15`}, {x, y}, {t, 0, 100}];
         ParametricPlot[Evaluate[{x[t], y[t]} /. sol2],
          {t, 0, 100}]
```

Out[57]=



В заключение данного раздела хотелось бы сказать, что функция ***NDSolve*** имеет достаточно большое количество аргументов, позволяющих управлять процессом вычисления, например задавать шаг разбиения сетки, выбор метода решения и т. д. Дополнительные сведения можно узнать из справочного руководства пакета *Mathematica*.

Задачи и упражнения

1. Решить дифференциальное уравнение $x''(t) - ax'(t) = 0$:
   a) в общем виде и оценить время, затрачиваемое пакетом на вычисления;
   b) с заданием произвольных начальных условий. Оценить время вычисления и исследовать зависимость $x(t)$ от начальных координат, построив несколько графиков на одном.
2. Решить систему уравнений $\begin{cases} x''(t) = y(t) \\ y'(t) = x(t) \end{cases}$, задав начальные условия в общем виде. Построить график полученных решений.

3. Провести численные решения уравнений из задач 1 и 2, и сравнить время, затраченное на выполнение при численном решении с временем аналитических вычислений.
4. Решить уравнение Ньютона для гармонического осциллятора $x''(t) = -\omega^2 x(t)$, где $\omega = \sqrt{k/m}$. Найти зависимость потенциальной, кинетической и полной энергий от времени, и построить их на одном графике.

СПИСОК ЛИТЕРАТУРЫ

1. В. Дьяконов. Mathematica 4: учебный курс. – Спб.: Питер, 2001.
2. В. Дьяконов. Mathematica 4 с пакетами расширений. – СПб.: Нолидж, 2000.
3. В.П. Дьяконов. Mathematica 4.1.4.2.5.0 в математических и научно-технических расчетах. – М.: СОЛОН-Пресс, 2004.
4. Я.К. Шмидский. Mathematica 5: Самоучитель. – М.: Диалог, 2004.

Учебное издание

ПАКЕТ MATHEMATICA ДЛЯ ИНЖЕНЕРНЫХ ВЫЧИСЛЕНИЙ

Часть I

Учебное пособие

Научный редактор А.П. Потылицын

Редактор *М.В. Пересторонина*
Верстка *В.П. Аршинова*
Дизайн обложки *О.Ю. Аршинова*
О.А. Дмитриев

Подписано к печати 30.07.2008. Формат 60x84/16. Бумага «Снегурочка».
Печать RISO. Усл. печ. л. 5,64. Уч.-изд. л. 5,1.
Заказ XXX. Тираж 100 экз.

Томский политехнический университет
Система менеджмента качества
Томского политехнического университета сертифицирована
NATIONAL QUALITY ASSURANCE по стандарту ISO 9001:2000

ИЗДАТЕЛЬСТВО ТПУ. 634050, г. Томск, пр. Ленина, 30.