А.Н.Прокопеня

РЕШЕНИЕ ФИЗИЧЕСКИХ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ СИСТЕМЫ *MATHEMATICA*

Допущено в качестве пособия
для студентов технических специальностей
Брестского государственного технического
университета

Брест 2005

УДК 530.1:530.10:681.3(075.8)
ББК 22.3я73
П 80

Рецензенты:
доктор физико-математических наук, профессор Р.Краглер,
кандидат физико-математических наук А.С.Гаркун

Прокопеня А.Н.

П 80 Решение физических задач с использованием системы *Mathematica*: Пособие. Брест: Издательство БГТУ, 2005 – 260 с.

ISBN 985-493-024-6

В пособии рассматривается применение системы *Mathematica* к решению физических задач. Дается общая характеристика системы, особенности создаваемых в ней документов, описание основных встроенных функций с примерами их использования. Приводится описание восьми лабораторных работ, в которых с помощью системы *Mathematica* моделируются различные физические структуры. В описании каждой работы приводятся необходимые теоретические сведения, примеры решения задач и набор заданий, предназначенных для самостоятельного выполнения.

Пособие предназначено для студентов технических специальностей Брестского государственного технического университета.

УДК 530.1:530.10:681.3(075.8)
ББК 22.3я73

ISBN 985-493-024-6

Предисловие

Характерная черта современного этапа развития общества – массовое внедрение компьютерных технологий во все сферы человеческой деятельности. Этому во многом способствует появление таких универсальных программных продуктов как система *Mathematica*, разработанная американской компанией Wolfram Research, Inc. Ее первая версия была выпущена в 1988 году и сразу привлекла к себе внимание пользователей. С помощью специально разработанного символьного языка в ней была реализована идея представления различных объектов с единых позиций на основе небольшого набора простейших понятий. Это позволило системе весьма эффективно выполнять как численные, так и символьные вычисления, оперировать графической и текстовой информацией, т.е. стать мощным инструментом для выполнения технических расчетов. Поэтому первыми пользователями системы *Mathematica* стали физики, математики и инженеры. Однако дальнейшее развитие системы обеспечило ей широкое применение и в таких областях, как кибернетика, биология, химия, экономика, финансовое и банковское дело. В настоящее время *Mathematica* занимает первое место по количеству приложений в различных областях науки, техники и образования. Система постоянно обновляется и расширяется, впитывая в себя все новейшие достижения современных информационных технологий. Следует отметить не только исключительно мощные вычислительные возможности системы, но и простоту и удобство работы с ней. *Mathematica* имеет широкий набор средств, позволяющих представлять результаты работы в виде презентации, отчета, доклада, статьи, удовлетворяющих всем современным требованиям. Поэтому пользователями системы являются люди самых разных профессий и уровня математической подготовки, общее число зарегистрированных пользователей превышает два миллиона. Важное место занимает *Mathematica* и в образовании. В настоящее время она используется в учебных и научных целях более чем в пятидесяти крупнейших университетах мира, на ее основе разработаны сотни различных курсов для студентов и школьников. О системе *Mathematica* и ее приложениях опубликовано более 300 книг на 18 различных языках (см., например, [1-9]). К сожалению, только малая часть из них написана на русском языке. Полную информацию о системе, а также о компании Wolfram Research Inc., можно найти в интернете на сайте http://www.wolfram.com.

Данное пособие написано на основе спецкурса "Решение прикладных физических задач с помощью системы *Mathematica*, читаемого студентам технических специальностей в Брестском государственном техническом университете с 1995 года. Основная его цель – познакомить студентов с системой *Mathematica* и научить использовать ее как для выполнения технических расчетов, так и для оформления полученных результатов. В основу курса положено решение физических задач, важных с точки зрения приложений в технике. Естественно, опыт работы с ситемой *Mathematica*, приобретаемый при решении конкретных проблем, позволит использовать ее в качестве рабочего инструмента не только при выполнении любых расчетов, подготовке курсовых и дипломных проектов, но и в последующей практической деятельности инженера.

Пособие состоит из двух частей. Первая часть представляет собой введение в систему *Mathematica*, в котором дается общая характеристика

системы, особенности создаваемых в ней документов, описание основных встроенных функций и примеры их использования. Вторая часть содержит описание лабораторных работ, в которых с помощью этой системы решаются различные физические и математические задачи. В описании каждой работы приводятся необходимые теоретические сведения, примеры решения задач и набор заданий, предназначенных для самостоятельного выполнения. Следует отметить, что многие предлагаемые задачи обычно не рассматриваются в стандартном курсе общей физики из-за математических сложностей, возникающих при их решении. Использование же системы *Mathematica* дает возможность переложить эти трудности на компьютер и сосредоточиться на проблеме постановки задачи и анализе получаемых результатов. Поэтому решение каждой задачи превращается в небольшую исследовательскую работу, которую студент выполняет самостоятельно, используя знания, полученные в курсах физики и математики. Изложение базируется на версии *Mathematica* 5.1, причем все пособие подготовлено как обычный документ, получаемый при работе с этой системой. Поскольку объем пособия ограничен, в его первой части в основном приводятся только те сведения, которые необходимы для выполнения лабораторных работ. Полное описание системы *Mathematica* содержится в оригинальной книге С.Вольфрама [10]. Необходимые теоретические сведения по физике можно найти в стандартных учебниках [11-15].

Автор выражает искреннюю признательность Р.Краглеру и А.С.Гаркуну за ряд замечаний, способствовавших улучшению пособия, а также преподавателям кафедры физики БГТУ за полезное обсуждение проблем, рассматриваемых в пособии.

Часть I

Введение в систему *Mathematica* *

1. Структура и особенности системы

Mathematica состоит из двух основных частей: вычислительное ядро (**The Kernel**) и оболочка (**The Front End**). Ядро представляет собой программное обеспечение, непосредственно выполняющее вычисления и работающее одинаково на всех типах компьютеров. Оболочка обеспечивает взаимодействие между ядром и пользователем. При этом вычислительное ядро и оболочка могут находиться как на одном, так и на разных компьютерах, в последнем случае связь между ними осуществляется по сети. Важной особенностью системы *Mathematica* является также возможность взаимодействия вычислительного ядра не только с пользователем, но и с другими компьютерными системами посредством специального интерфейса **MathLink**.

Наиболее общей и удобной для пользователей является оболочка, используемая при работе в операционной системе Windows. Она позволяет работать с ядром в интерактивном режиме, при котором команды, предназначенные для выполнения, и получаемые результаты изображаются на экране монитора. Для набора команд и редактирования введенной информации можно использовать весь спектр возможностей клавиатуры и мыши, включая копирование (**Copy**), вырезание (**Cut**), вставки (**Paste**), удаление (**Delete**) и т.д.. Кроме того, выбор одного из значений опции **Palettes** в разделе **File** главного меню позволяет открыть одно или несколько дополнительных окон с клавишами, нажатие которых с помощью левой кнопки мыши приводит к появлению соответствующих символов или команд в том месте основного документа, где находится курсор. Это позволяет представлять команды и формулы в виде, близком к традиционным математическим обозначениям.

Характерной особенностью документов, создаваемых в системе *Mathematica*, является возможность соединять в них различные элементы, такие как текст, команды, выполняемые ядром, получаемые при этом результаты в виде чисел, выражений, таблиц, графиков, звука, а также рисунки. Весь документ легко структурируется, т.е. разбивается на главы, параграфы, может содержать заголовок, подзаголовок и т.д. Выбор одного из значений опции **Style Sheet** в разделе **Format** главного меню позволяет изменять стиль всего документа, придавая ему вид статьи, отчета, доклада, учебника и т.п. Опытные пользователи могут определять свои собственные стили для различных документов. Созданный документ можно сохранить на диске, распечатать на бумаге, переслать по электронной почте. В англоязычной литерату-

---

* **Полужирным шрифтом** в тексте набраны названия разделов меню, команд, функций и опций системы *Mathematica*, секции, содержащие команды и формулы, предназначенные для обработки ядром, а также получаемые результаты.

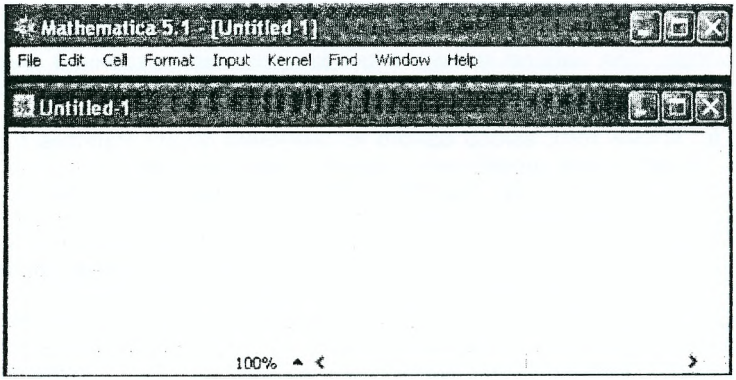
Курсивом набраны название системы *Mathematica*, математические формулы и определения, а также метки секций *In[n]:=* и *Out[n]=*, содержащие соответственно данные, предназначенные для обработки ядром, и получаемые результаты.

ре такой документ называется *Mathematica notebook*, в русскоязычной литературе встречаются термины "тетрадь", "блокнот", "записная книжка". Важно помнить, что он является "живым" документом, который можно редактировать, изменять или улучшать отдельные его части, алгоритмы вычислений и способы представления получаемых результатов для получения желаемого эффекта.

2. Работа с системой *Mathematica*

2.1. Загрузка системы. Первые вычисления

Для запуска системы достаточно подвести указатель мыши к иконке *Mathematica* и дважды щелкнуть ее левой кнопкой. После загрузки на экране появляются два окна, одно из которых является основным и имеет заголовок **Mathematica 5.1 - [Untitled-1]**, где число **5.1** обозначает номер версии системы, а слово **Untitled-1** – имя текущего документа. В этом окне размещается главное меню системы со стандартным набором разделов (**File**, **Edit**, **Cell**, **Format**, **Input**, **Kernel**, **Find**, **Window**, **Help**), характерным для работы в операционной системе Windows.

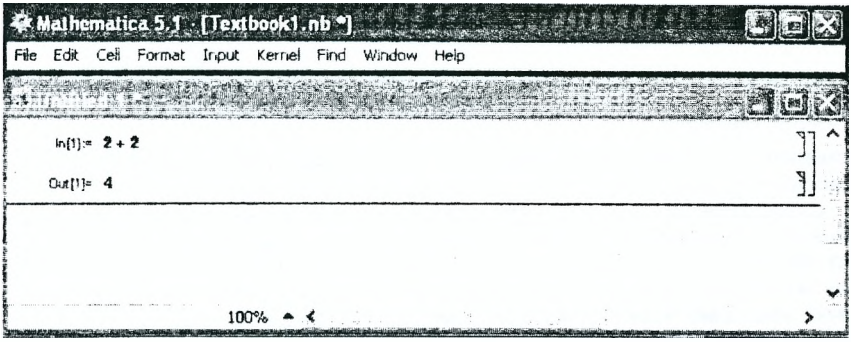


Второе окно имеет заголовок **Untitled-1** и представляет собой бланк нового документа, в который пользователь вводит свои данные. В верхней части рабочего поля этого окна имеется тонкая горизонтальная линия, показывающая, что вводимые символы будут изображаться под ней. Введем, например, простейшее выражение:

```
2 + 2
```

С появлением первого символа горизонтальная линия исчезает и появляется курсор в виде мигающего вертикального отрезка, который показывает место, в котором будет изображаться следующий вводимый символ. Кроме того, вводимые данные автоматически выделяются квадратной скобкой в правой части рабочего поля окна и составляют одну секцию или ячейку (**cell**) документа. Секция содержит по крайней мере одну строку текста и является наименьшей частью данных, которые обрабатываются ядром системы *Mathematica*. Для обработки секции (т.е. выполнения содержащихся в ней команд и

вычислений) достаточно выделить ее, подведя указатель мыши к]-скобке и щелкнув левой кнопкой мыши (при этом соответствующая скобка выделяется темным фоном), или просто поместить курсор в любое место этой секции и нажать клавиши: **Shift+Enter** (удерживая клавишу **Shift**, нажимаем клавишу **Enter**). В качестве альтернативы комбинации **Shift+Enter** можно использовать клавишу **Enter** на цифровой части клавиатуры. После вычислений получаем:



При обработке введенных данных *Mathematica* снабжает их меткой *In[1]:=*, а результат вычислений помещает в отдельную секцию с меткой *Out[1]=* . Две секции, содержащие команду, вводимую пользователем, и получаемый результат, объединяются в группу с помощью еще одной]-скобки. Под этой группой секций появляется горизонтальная линия, показывающая, что вводимые символы следующего выражения будут изображаться под ней в новой секции.

Следует отметить, что при запуске системы ее вычислительное ядро не загружается автоматически, это происходит при выполнении первого вычисления. Поскольку *Mathematica* – очень большая система, загрузка ядра и его инициализация требует некоторого времени. Поэтому кажется, что первое вычисление даже очень простого выражения выполняется медленно. Однако повторное вычисление этого выражения показывает, что оно выполняется практически мгновенно. При этом числа в квадратных скобках меток *In* и *Out* увеличиваются на единицу.

Итак, в простейшем случае работа с системой *Mathematica* напоминает работу с обычным калькулятором и представляет собой диалог, состоящий из пар:

In[n] := Команда, вводимая пользователем
Out[n] = Результат, выводимый системой

Подчеркнем, что метки *In[n]* не вводятся пользователем, а появляются только после обработки соответствующей секции, причем число *n* увеличивается на единицу при выполнении каждой последующей команды. Обычно команда пользователя сводится либо к выражению, которое нужно вычислить, либо к применению какой-либо встроенной функции, число которых в системе *Mathematica* превышает тысячу, к полученному ранее выражению. Номер *n*, которым помечается очередной диалог, можно использовать для ссылки на любой предыдущий результат. Так, выражение *%n* соответствует

результату, полученному в секции с меткой *Out[n]=* . Один символ % соответствует последнему полученному результату, два символа %%, набираемые без пробела – предпоследнему результату и т.д. Рассмотрим несколько примеров.

In[1]:= `3^12`

Out[1]= `531441`

In[2]:= `%^(1/6)`

Out[2]= `9`

In[3]:= `%1+%`

Out[3]= `531450`

Отметим, что символ шляпка " ^ " обозначает операцию возведения в степень.

Существует и другой способ ссылки на полученные результаты. Любому объекту в системе *Mathematica* (числу, выражению, функции, уравнению и его решению, графику и т.д.) можно присвоить имя, которое также позволяет ссылаться на него в последующих вычислениях. Операция присваивания имеет вид: **имя = значение**. Например, команда **a = 3** означает, что символу *a* присвоено численное значение *3*. Далее, в любом выражении, где встречается символ *a*, при обработке секции он будет заменяться на *3*. Примеры:

In[4]:= `a=3`

Out[4]= `3`

In[5]:= `x=(a^2+3b)/c`

Out[5]= $\frac{9+3b}{c}$

In[6]:= `a/x`

Out[6]= $\frac{3c}{9+3b}$

Имя может состоять из любого количества букв, цифр и знака $, но не должно начинаться с цифры и содержать пробелы, поскольку пробел в системе *Mathematica* соответствует знаку умножения. Так, выражение *a2* может обозначать имя какого-либо объекта, а *2a* соответствует произведению *2*a*, даже если пробел между символами *2* и *a* отсутствует. Следует помнить, что имя сохраняет присвоенное ему значение до конца текущего сеанса работы с системой или до тех пор, пока оно не удалено. Для удаления значения, присвоенного символу *a*, например, можно использовать команду **Clear[a]** или ввести выражение **a =.** , причем символы знак равенства "=" и точка "." набираются один за другим без пробела. Чтобы удалить значения, присвоенные нескольким переменным, можно набрать соответствующие имена в качестве аргументов функции **Clear**, разделяя их запятыми, например,

In[7]:= `Clear[a, x]`

Вообще, полезно начинать любой новый документ с команды (обратите внимание на символ " ` ", набираемый после слова **Global**):

```
Clear["Global`*"]
```

В этом случае при его обработке сначала производится удаление значений, присвоенных всем символам в других документах, которые обрабатывались ранее в течение данного сеанса работы с системой. Это позволяет избежать ошибок и использования неправильных значений переменных при вычислениях.

Если используется имя, похожее на уже введенное ранее, то при обработке соответствующей секции *Mathematica* выдает предупреждение о возможной ошибке. Пример:

```
In[8]:= cosx = 0.5

Out[8]= 0.5

In[9]:= cosy = 1

General::spell1 :
 Possible spelling error: new symbol name "cosy" is
   similar to existing symbol "cosx". More...

Out[9]= 1
```

Как видим, появилось сообщение о том, что при записи нового имени **cosy** возможно была сделана ошибка, поскольку символ **cosy** похож на уже существующий символ **cosx**. Это сообщение является только предупреждением и не означает наличие ошибки. Исключить появление таких сообщений можно с помощью команды:

```
Off[General::spell, General::spell1]
```

В этом случае вся ответственность за правильность набора всех имен ложится на пользователя.

Mathematica различает заглавные и строчные буквы. Существует правило, согласно которому имена всех встроенных функций и постоянных начинаются с заглавной буквы и защищены от изменений. Попробуем, например, присвоить какое-либо значение символу **E**.

```
In[10]:= E = 5

Set::wrsym : Symbol e is Protected. More...

Out[10]= 5

In[11]:= 3 E

Out[11]= 3 e
```

Поскольку заглавная буква **E** в системе *Mathematica* используется для обозначения основания натурального логарифма, то выдается сообщение, что соответствующий символ защищен, т.е. ему нельзя присвоить какое-либо другое значение. Чтобы приблизить обозначения математических констант к традиционным, в третьей и последующих версиях системы *Mathematica* для основания натурального логарифма используется также дополнительное обозначение в виде буквы e с двойным начертанием. Поскольку *Mathematica* считает символы с двойным начертанием отличными от обычных, строчную букву e можно использовать в качестве имени, что легко видеть из следующих примеров.

In[12]:= `e = 5`

Out[12]= `5`

In[13]:= `3 e`

Out[13]= `15`

Если в конце выражения стоит точка с запятой, то при его обработке *Mathematica* производит все указанные вычисления, но результат не выводит на экран, т.е. соответствующая секция с меткой *Out[n]* отсутствует.

In[14]:= `c = a + b;`

Чтобы убедиться в том, что команда присваивания выполнена, вычислим

In[15]:= `c^2`

Out[15]= $(a + b)^2$

Таким образом, если в ходе вычислений нас не интересует какой-либо промежуточный результат, в конце соответствующего выражения необходимо поставить точку с запятой ";". При этом в одной стороке можно ввести сразу несколько выражений, разделяя их точками с запятой. Если после последнего выражения точка с запятой отсутствует, то будет выведен только окончательный результат. Это справедливо и в том случае, если отдельные выражения вводятся в разных строчках одной и той же секции.

In[16]:= `x = 2; y = a - b; y + c^x`

Out[16]= $a - b + (a + b)^2$

In[17]:= `x = c - a;`

`x y`

Out[18]= `(a - b) b`

Если вводимое выражение не помещается в одной строке, автоматически происходит переход на следующую строку. При этом отдельные слова в выражении не разрываются и смысл выражения сохраняется. Место, в котором происходит разрыв выражения при переходе на новую строку, зависит от размера рабочего поля окна по горизонтали. Если же для перехода на следующую строку нажимается клавиша **Enter**, то в общем случае новая строка будет восприниматься как начало самостоятельного выражения. Чтобы этого не случилось, нужно делать переход на новую строку так, чтобы предыдущая строка не представляла собой законченное выражение, которое может быть вычислено самостоятельно. Например, первая строка может заканчиваться знаком "+" или "," , что предполагает наличие продолжения, или не все скобки в ней оказываются закрытыми. Другая возможность состоит в том, чтобы в месте разрыва выражения перед переносом ввести знак " \ ". Примеры:

In[19]:=
```
Integrate[Sqrt[z - 1] / (1 - Sqrt[z + 1]),
                        {z, 1, 2}]
```

Out[19]= $-2 - \sqrt{3} + \frac{5\pi}{6}$

```
In[20]:= (a - b) ^ 3 (a + b) ^ 2 \
                  * (a + 2 b)
Out[20]= (a - b)^3 (a + b)^2 (a + 2 b)
```

2.2. Виды скобок и разделителей

В системе *Mathematica* используется пять видов скобок, причем каждый вид имеет свое назначение. Например, круглые скобки используются только для группирования выражений и изменения стандартного порядка выполнения операций. В квадратные скобки заключаются аргументы всех функций, как встроенных, так и вводимых пользователем. Фигурные скобки применяются для обозначения списков (они подробно рассматриваются в разделе 7). Двойные квадратные скобки служат для выделения части выражения или списка. Круглая скобка и звездочка, набираемые без пробела, используются для обозначения комментария, который может располагаться в одной секции с командой и не влияет на ее выполнение. Все виды скобок и разделителей, а также примеры их использования, показаны в следующей таблице:

Название	Обозначение	Назначение	Пример
Круглые скобки	( )	группирование	**(a + b) c**
Квадратные скобки	[ ]	аргумент функции	**Cos[x]**
Фигурные скобки	{ }	список	**{x, y, z}**
Двойные квадратные скобки	[[ ]]	часть выражения	**s[[2]]**
Круглые скобки со звездочками	(* *)	комментарий	**(* пример *)**
Запятая	,	разделение элементов	**{x, y, z}**
Точка с запятой	;	разделение выражений	**a + b; x + y**
Кавычки	" "	текстовая строка	**" Строка "**

Для разделения элементов списка и аргументов функций используется запятая. Напомним, что точка с запятой позволяет не только разделить две команды, набираемые в одной строке, но и отменяет вывод результата вычислений. Выражение, выделенное с помощью кавычек, является одним из базовых элементов системы *Mathematica* и называется *строкой*. Следует отметить, что использование различных видов скобок позволяет избежать неточностей при записи выражений.

2.3. Прерывание вычислений. Сохранение результатов работы. Выход

Иногда при работе с системой *Mathematica* возникает необходимость прервать вычисления. Обычно для этого достаточно с помощью мыши выбрать команду **Interrupt Evaluation** (прервать вычисления) или **Abort Evaluation** (прекратить вычисления) в разделе **Kernel** главного меню (соответствующие горячие клавиши "**Alt + ,**" и "**Alt + .** "). Если в первом случае открывается дополнительное окно с набором клавиш, которые позволяют управлять

дальнейшим процессом вычисления, то во втором случае вычисления сразу же прекращаются.

Однако при выполнении громоздких расчетов даже выбор команды **Abort Evaluation** иногда не приводит к остановке вычислений. В таких случаях приходится удалять вычислительное ядро из оперативной памяти компьютера, выбирая с помощью мыши последовательно следующие опции в разделе **Kernel** главного меню: **Kernel** ↦ **Quit Kernel** ↦ **Local**. При этом все присвоения и результаты, полученные в ходе текущего сеанса работы с системой, удаляются из оперативной памяти компьютера, хотя документ остается открытым для дальнейшей работы с ним. При следующем запуске вычислений ядро будет опять загружаться, что потребует некоторого времени. Если для выполнения расчетов в обрабатываемой секции требуются некоторые результаты, полученные в предыдущих секциях, то предварительно нужно повторно запустить эти секции. Заметим, что для обработки всех секций документа в порядке их следования достаточно выбрать следующие опции в разделе **Kernel** главного меню: **Kernel** ↦ **Evaluation** ↦ **Evaluate Notebook**..

Для сохранения полученного документа достаточно с помощью мыши выбрать опцию **Save** или **Save as** в разделе **File** главного меню. Кроме того, можно нажать клавиши "**Ctrl+S**" или "**Shift+Ctrl+S**" соответственно. При этом открывается дополнительное окно, в котором нужно указать директорию или папку, куда следует поместить соответстующий файл (в строке **Save in**), и имя файла (в строке **File name**). По умолчанию сохраняемый файл будет иметь тип ***Mathematica* Notebook** и расширение **.nb**. После сохранения имя файла будет автоматически внесено в заголовок окна вместо слова **Untitled-1**. Следует отметить, что работая с документом, полезно периодически сохранять полученные результаты, так как в случае каких-либо сбоев в работе компьютера несохраненные результаты будут потеряны.

Для окончания работы с системой *Mathematica* достаточно с помощью мыши выбрать опцию **Exit** в разделе **File** главного меню. Если перед выходом документ не был сохранен, автоматически появляется вопрос, следует ли сохранить введенные данные. В случае положительного ответа открывается дополнительное окно, в котором можно определить имя сохраняемого файла. При следующем сеансе работы с системой *Mathematica* сохраненный ранее файл может быть открыт путем выбора опции **Open** в разделе **File** главного меню для дальнейшей работы с ним.

2.4. Получение справочной информации

Наиболее полную информацию о системе и работе с ней можно получить, выбрав опцию **Help Browser** в разделе **Help** главного меню. В этом случае открывается дополнительное окно с набором закладок, выбирая которые, можно получить информацию об инсталяции системы и особенностях работы с ней (закладка **Getting Started**), о разделах главного меню и опциях, управляющих работой оболочки (закладка **Front End**), о всех встроенных функциях (закладка **Built-in Functions**), об имеющихся дополнительных программах (**Packages**), расширяющих вычислительные возможности системы *Mathematica* (закладка **Add-ons & Links**), а также прочитать любой раздел из полного описания системы (закладка **The *Mathematica* book**) и посмотреть множество демонстрационных программ (закладки **Tour** и **Demos**). Кроме того, имеется алфавитный указатель (закладка **Master Index**), позволяющий получить информацию о любом встроенном объекте. Заметим, что про-

сматривая примеры использования встроенных функций, можно копировать соответствующие секции или их части, вставлять в свой документ и редактировать, приспосабливая их к своим вычислениям.

Получить информацию о любой функции или постоянной, которая является встроенной или введена пользователем, можно также путем набора ее имени после знака вопроса и обработки соответствующей секции. Набирая, например, **?Sin**, получаем:

In[21]:= **? Sin**

```
Sin[z] gives the sine of z. More...
```

Набирая два знака вопроса вместо одного, получаем некоторую дополнительную информацию о функции.

In[22]:= **?? Sin**

```
Sin[z] gives the sine of z. More...

Attributes[Sin] = {Listable, NumericFunction, Protected}
```

В обоих случаях после основной информации о функции появляется выделенное цветом или подчеркнутое слово More... , которое осуществляет связь с программой **Help Browser**. Если подвести указатель мыши к этому слову и щелкнуть левой кнопкой мыши, то запускается программа помощи **Help Browser** и открывается дополнительное окно, в котором содержится вся информация о функции, примеры ее использования и ссылки на соответствующие разделы описания системы (The *Mathematica* book [10]).

Для получения информации о функциях с похожими именами используется символ звездочка " * ", который заменяет любую последовательность символов. Так, команда **?*Sin** вызывает список всех объектов, имена которых заканчиваются словом **Sin**.

In[23]:= **? *Sin**

```
System`

ArcSin  Sin
```

Команда **?*Tan*** вызывает список всех объектов, в именах которых содержится слово **Tan**.

In[24]:= **? *Tan***

```
System`

ArcTan  ArcTanh  Tan  Tanh
```

Имена всех объектов в системе *Mathematica* помещаются в различные папки, которые называются контекстами (**Context**). Текущая папка, в которой находятся имена объектов, вводимых пользователем, называется **Global**. Имена всех встроенных функций, которые доступны без загрузки каких-либо дополнительных программ (**Packages**), находятся в папке **System**. Поскольку одно и то же имя может встречаться в различных контекстах и относиться к разным функциям, при вызове списка функций с помощью знака вопроса выводится также название контекста, в котором находятся соответствующие

функции (контекст **System** в предыдущих примерах). Кроме того, каждое имя в списке подчеркнуто, что означает наличие связи с программой помощи **Help Browser** и возможность быстрого вызова подробной информации о функции.

Следует отметить, что существует и другая возможность получить подробную информацию о функции, имя которой вам известно. Для этого достаточно набрать имя функции, затем выделить его с помощью левой кнопки мыши или путем перемещения курсора с помощью клавиш-стрелок при нажатой клавише **Shift** и нажать клавишу **F1**. При этом запускается программа помощи **Help Browser** и открывается дополнительное окно, в котором содержится вся информация о функции.

2.5. Дополнительные возможности системы *Mathematica*

Наиболее важная особенность системы *Mathematica* состоит в том, что она является открытой системой, которую можно совершенствовать и расширять. Это значит, что в дополнение к уже имеющимся встроенным функциям, которые доступны при каждом запуске вычислительного ядра, можно определить новые функции и поместить их в отдельные модули, которые называются **Packages**. Каждый модуль представляет собой документ, создаваемый в системе *Mathematica* и сохраняемый в файл специального формата, который содержит все необходимые определения и правила обращения к функции. Можно выделить несколько типов таких модулей.

Во-первых, стандартные модули (**Standard Packages**), входящие в обычный комплект системы *Mathematica*. В этих модулях определяются функции, которые используются не очень часто. Именно поэтому они не загружаются при запуске ядра системы, но могут быть загружены по команде пользователя. Чтобы получить список этих модулей и описание их возможностей, достаточно запустить программу помощи **Help Browser** и выбрать в разделе **Add-ons & Links** закладку **Standard Packages**. При этом в дополнительном окне появится список модулей: **Algebra**, **Calculus**, **DiscreteMath** и т.д.

Во-вторых, дополнительные модули (**Extra Packages**), которые разрабатываются сотрудниками компании Wolfram Research для решения специальных задач. Впоследствии некоторые из них поставляются с обычной копией системы *Mathematica*. Информацию о таких модулях можно получить, запустив программу помощи и выбрав последовательно разделы **Add-ons & Links** ↦ **Wolfram Research Products**.

В-третьих, коммерческие модули, которые разрабатываются и поставляются независимыми производителями. Для их использования требуются отдельные лицензии и система *Mathematica* в качестве базовой программы. Информацию о таких программных продуктах можно найти в интернете на сайте компании Wolfram Research.

Четвертый вид дополнительных модулей – это модули, разрабатываемые самими пользователями для решения своих задач.

Любая дополнительная функция, определенная в каком-либо модуле, становится доступной только после его загрузки в оперативную память компьютера. Чтобы узнать, какие модули уже загружены, достаточны выполнить следующую команду:

```
In[25]:= $Packages
Out[25]= {Global`, System`}
```

Поскольку мы пока не использовали дополнительные модули, в полученном списке указаны только два контекста: **Global** и **System**. Чтобы загрузить модуль, который во время текущего сеанса работы с системой *Mathematica* еще не использовался (т.е. его нет в списке, выдаваемом командой **$Packages**), применяется функция **Needs**. Ее аргумент должен указывать имя модуля и название контекста, в котором он хранится. Формат обращения к этой функции понятен из следующего примера, в котором производится загрузка модуля **FilledPlot**, находящегося в контексте **Graphics**.

```
In[26]:= Needs["Graphics`FilledPlot`"];
```

Подчеркнем, что наличие символов " ` " (не путать со штрихом), между которыми набирается имя модуля, и кавычек, в которые заключается аргумент функции **Needs**, является необходимым условием ее нормальной работы. Как только модуль загружается, в списке доступных функций появляются новые имена. Действительно, выполняя команду **$Packages** еще раз, получаем следующий список:

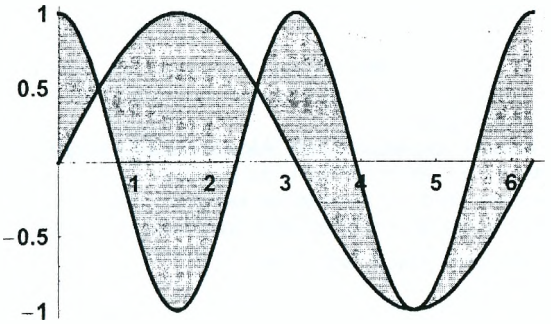
```
In[27]:= $Packages
Out[27]= {Utilities`FilterOptions`,
          Graphics`FilledPlot`, Global`, System`}
```

Получить информацию о любой из новых доступных функциях можно стандартным образом, например,

```
In[28]:= ? FilledPlot

  FilledPlot[function,{var,varmin,varmax}] generates a plot
    with the area between the curve and the var axis
    filled. FilledPlot[{f1,f2,...},range] generates a
    plot with the areas between curve f1 and f2, f2 and
    f3, etc. filled. The shade of the fill and other
    specifications can be given by the Fills option. More...
```

Функция **FilledPlot** генерирует график одной или нескольких функций, на котором области между кривыми заполняются заданным фоном. Пример:

```
In[32]:= FilledPlot[{Cos[2 x], Sin[x]}, {x, 0, 2 π},
           PlotStyle → Thickness[0.008],
           Fills → GrayLevel[.7], Curves → Front];
```



Если обращение к какой-либо дополнительной функции, например, **InequalitySolve**, происходит до того, как соответствующий модуль **Algebra`InequalitySolve** был загружен, то функция не будет работать. Действительно, при попытке решить систему двух неравенств с помощью функции **InequalitySolve**, получаем:

```
In[35]:= Clear[x, y];
         InequalitySolve[x^2 + z^2 < 1 && x < z, {x, z}]
Out[36]= InequalitySolve[x^2 + z^2 < 1 && x < z, {x, z}]
```

При этом в контексте **Global** появляется новое имя **InequalitySolve**, которое не имеет никакого отношения к вызываемой функции. Попробуем получить информацию об этом объекте.

```
In[37]:= ? InequalitySolve

Global`InequalitySolve
```

Как видим, никакой информации о функции нет, имеется только имя **InequalitySolve** в контексте **Global**. Однако наличие одинаковых имен в различных контекстах **Global** и **Algebra`InequalitySolve** приводит к тому, что функция **Needs** не загружает соответствующий модуль.

```
In[38]:= Needs["Algebra`InequalitySolve`"]

InequalitySolve::shdw :
 Symbol InequalitySolve appears in multiple
   contexts {Algebra`InequalitySolve`, Global`};
   definitions in context Algebra`InequalitySolve` may
   shadow or be shadowed by other definitions. More
```

Чтобы решить эту проблему, нужно сначала удалить имя **InequalitySolve** из контекста **Global**, используя команду **Remove**.

```
In[39]:= Remove[InequalitySolve]
```

После этого модуль **Algebra`InequalitySolve** легко загружается, и определяемая в нем функция **InequalitySolve** может быть использована для решения системы неравенств. Пример:

```
In[40]:= Needs["Algebra`InequalitySolve`"]

In[41]:= InequalitySolve[x^2 + y < 1 && x < y, {x, y}]
```

$$\text{Out[41]}= \frac{1}{2}\left(-1-\sqrt{5}\right) < x < \frac{1}{2}\left(-1+\sqrt{5}\right) \;\&\&\; x < y < 1 - x^2$$

Отметим также, что функция **Needs** не загружает модуль и в том случае, если он уже был загружен ранее, и его имя присутствует в списке, выдаваемом по команде **$Packages**.

В отличие от **Needs**, функция **Get["Algebra`InequalitySolve`"]** загружает дополнительный модуль, не проверяя, загружался ли он ранее. При обращении к этой функции вместо ее имени можно набрать два знака "<" без пробела между ними и указать полное имя соответствующего модуля.

```
<< Algebra`InequalitySolve`
```

Следует отметить, что при использовании функции **Get** возможно появление совпадающих имен в различных контекстах, что может привести к ошибкам в работе системы.

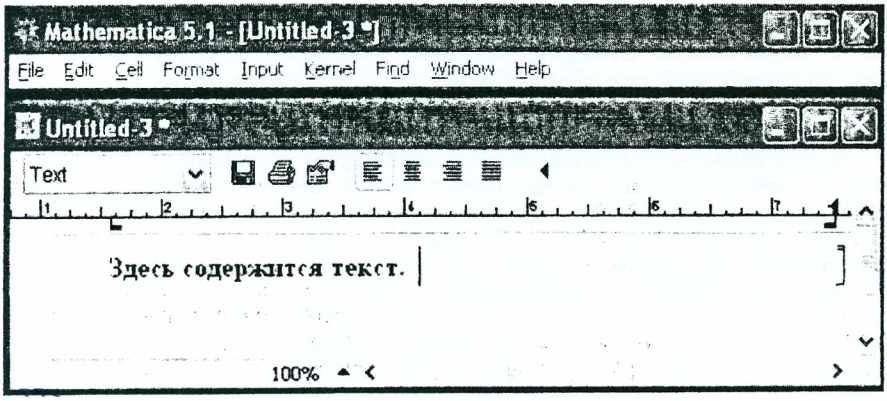
3. Создание документа в системе *Mathematica*

3.1. Выбор стиля документа

При запуске системы *Mathematica* открывается окно с бланком нового документа. Однако создать новый документ можно на любом этапе работы, поскольку эта система позволяет работать с несколькими документами одновременно. Для этого достаточно выбрать с помощью мыши опцию **New** в разделе **File** главного меню или нажать клавиши **Ctrl+N**. Переход от одного документа к другому при работе с несколькими документами осуществляется путем выбора соответствующего имени в разделе **Window** главного меню. При этом заголовок активного документа выделяется более ярким цветом.

Стандартный пакет *Mathematica* содержит более десяти шаблонов, предназначенных для подготовки документов различного вида, например, научных докладов, презентаций, статей, книг, отчетов. Опытные пользователи могут редактировать данные шаблоны, а также создавать свои собственные. Названия шаблонов являются значениями опции **Style Sheet** в разделе **Format** главного меню, а сами они определяют стиль всего документа. По умолчанию каждый новый документ имеет стиль **Default**. Для изменения стиля документа достаточно выбрать другое значение опции **Style Sheet**.

Чтобы с документом было удобнее работать, рекомендуется в разделе **Format** поставить "птичку" у опции **Show ToolBar**. При этом в верхней части рабочего окна появляется дополнительный ряд кнопок и маленькое окно, в котором указывается тип той секции, где находится курсор (см. рисунок).



Если подвести указатель мыши к треугольнику ▼, находящемуся в правой части этого маленького окна, и щелкнуть левой кнопкой мыши, то появится выпадающее меню, содержащее список всех типов секций, которые можно создавать в данном документе (этот список совпадает с соответствующим списком значений опции **Style** в разделе **Format** главного меню). Значки на кнопках поясняют операции, выполняемые при их нажатии. Кроме того,

соответствующая подсказка появляется, если подвести указатель мыши к кнопке. Если же поставить "птичку" у опции **Show Ruler** в разделе **Format**, то в верхней части документа появится масштабная линейка, на которой отмечены границы рабочего поля в текущей секции и правая граница страницы при печати документа.

3.2. Секционная структура документа

По умолчанию каждая новая секция имеет тип **Input** и предназначается для ввода команды или выражения, обрабатываемого ядром. Результаты вычислений помещаются в секции, имеющие тип **Output**. Поэтому в простейшем случае получаемый документ представляет собой последовательность секций этих двух видов. Однако в системе *Mathematica* существуют и другие типы секций, причем в зависимости от стиля всего документа, определяемого опцией **Style Sheet**, их количество может изменяться. Тип секции определяет ее роль в документе. Например, секция типа **Text** используется для размещения текста, поясняющего вычисления, и не обрабатывается ядром. Секция, содержащая заголовок документа, имеет тип **Title**. Для определения подзаголовков имеются секции типа **Subtitle** и **Subsubtitle**, а для выделения названий глав и параграфов – секции типа **Section**, **Subsection** и **Subsubsection**. Каждая секция выделяется скобкой в правой части рабочего поля окна, причем в зависимости от типа секции вид скобки несколько изменяется. Две или более секций могут объединяться в группу, которая выделяется справа еще одной скобкой. Отдельные секции и группы секций в свою очередь могут быть объединены в параграфы и главы, снабжены соответствующими заголовками и выделены дополнительными скобками. Чем правее расположена скобка на экране, тем больше секций она охватывает.

По умолчанию процесс группирования секций в документе происходит автоматически, что определяется выбором опции **Automatic Grouping** в подразделе **Cell Grouping** раздела **Cell** главного меню. Соответственно выбор опции **Manual Grouping** позволяет пользователю осуществлять процесс группирования секций самостоятельно, используя опции **Group Cells** (объединить выделенные секции в группу) и **Ungroup Cells** (расформировать группу) из подраздела **Cell Grouping**.

Если в документе установлено автоматическое группирование секций, то этот процесс происходит следующим образом. Секция каждого типа имеет свой приоритет. Самым высоким приоритетом обладает секция типа **Title**, в которой обычно определяется название документа. Поэтому эта секция всегда является первой в группе. За ней следуют секции, определяющие подзаголовки (**Subtitle** и **Subsubtitle**). Далее в группе могут содержаться одна или несколько подгрупп, в каждой из которых первая секция имеет тип **Section**. Эти подгруппы соответствуют главам книги. В каждой такой подгруппе могут быть свои подгруппы или параграфы с заголовками типа **Subsection** и т.д. Общее правило автоматического группирования секций таково: секция, имеющая более низкий приоритет, включается в расположенную выше нее группу, заголовок которой имеет более высокий приоритет.

Секционная система позволяет создавать в системе *Mathematica* документы, которые по структуре напоминают книгу, т.е. имеют заголовок, подзаголовки, главы, параграфы и т.д. Основное их отличие от книги состоит в том, что это интерактивные документы, которые можно не только читать, но

и редактировать, изменяя отдельные их секции и добавляя новые, а также производить в них все необходимые вычисления. Имеется и еще одна особенность таких документов. Если подвести указатель мыши к скобке, ограничивающей группу секций, и дважды щелкнуть левой кнопкой мыши, то все секции в группе, за исключением первой, будут скрыты, а в нижней части ограничивающей ее скобки появится маленький черный треугольник. Чтобы открыть и показать на экране монитора остальные секции группы, достаточно еще раз дважды щелкнуть левой кнопкой мыши, подведя ее указатель к соответствующей скобке с черным треугольником внизу. Эта возможность скрывать и открывать содержимое группы секций особенно удобна при презентациях и чтении лекций.

3.3. Изменение стиля, создание, деление и объединение секций

Для изменения типа секции необходимо выделить ее и затем с помощью мыши выбрать нужный тип в подразделе **Style** раздела **Format** главного меню. Если же открыто дополнительное окно, в котором указан тип выделенной секции (т.е. установлена "птичка" у опции **Show ToolBar**), то выбрать нужный тип можно и в меню, которое появляется при нажатии кнопки ▼, расположенной в правой части этого окна.

Перемещая мышь по коврику, легко видеть, что форма указателя мыши изменяется в зависимости от его положения. Если указатель мыши находится в пределах какой-либо секции, то он имеет вид вертикального отрезка. Щелкнув в этот момент левой кнопкой мыши, мы помещаем курсор в соответствующее место этой секции и можем редактировать введенное ранее выражение или продолжить ввод данных. Если же указатель мыши находится между секциями, то он принимает вид горизонтального отрезка. Если щелкнуть левой кнопкой мыши в этот момент, то между секциями появится тонкая горизонтальная линия, которая показывает, что следующее выражение будет вводиться под ней в новой секции. Следует отметить, что точно так же будет изменяться форма курсора при его перемещении с помощью стрелок на клавиатуре.

Любую секцию можно разделить на две части, поместив курсор в соответствующую точку и выбрав команду **Divide Cell** в разделе **Cell** главного меню (или нажав клавиши **Shift+Ctrl+D**). При этом в первой секции останутся все символы, расположенные выше и левее курсора. Если же выделить две или несколько рядом расположенных секций и выбрать опцию **Merge Cells** в том же разделе (или нажать клавиши **Shift+Ctrl+M**), то секции будут объединены в одну, которая имеет тот же тип, что и самая верхняя выделенная секция.

Любую секцию или ее часть можно вырезать, копировать и затем вставить в другое место документа, используя стандартные возможности редактирования, предоставляемые соответствующими командами в разделе **Edit** главного меню. Используя опции из раздела **Format**, можно изменять размер (**Size**), тип (**Font**) и цвет (**Text Color**) шрифта как во всей секции, так и в любой ее части. Чтобы установить, например, размер букв в каком-либо выражении или во всей секции в 18 пикселей, необходимо выделить это выражение или секцию и затем последовательно выбрать соответствующую опцию в разделе **Format** главного меню: **Format** ↦ **Size** ↦ **18 Point**. Таким же образом можно изменить тип шрифта или цвет какого-либо выражения, выделить его с помощью цветового фона (опция **Background Color**) и т.д.

3.4. Подготовка документа к печати

Чтобы распечатать готовый документ, достаточно в разделе **File** главного меню выбрать опцию **Print** или нажать клавиши **Ctrl+P**. Как обычно при работе в операционной системе **Windows** появится дополнительное окно, в котором можно определить тип принтера и его свойства, количество печатаемых копий, а также указать, следует ли печатать весь документ или только отдельные его страницы. Кроме того, можно распечатать только одну или несколько секций документа. Для этого нужно выделить их и выбрать опцию **Print Selection** в разделе **File** или нажать клавиши **Shift+Ctrl+P**. Предварительно рекомендуется указать размеры бумаги и полей, оставляемых при печати, сделав соответствующие установки в дополнительном окне, которое появляется при выборе опции **File** ↦ **Printing Settings** ↦ **Page Setup**. Если вместо **Page Setup** выбрать опцию **Headers and Footers**, то можно указать, какие дополнительные надписи в верхней и нижней частях листа следует сделать. Например, можно написать название соответствующей главы или всего документа, номер страницы, отделить эту надпись от основного текста линией и т.д.

Документы, создаваемые в системе *Mathematica*, являются интерактивными документами, с которыми большинство пользователей работает, сидя за компьютером. Поэтому они должны хорошо смотреться на экране монитора. Однако качество документа, распечатанного на бумаге, также должно быть высоким. Чтобы документ хорошо выглядел в обоих случаях, в системе *Mathematica* имеется возможность устанавливать его стили отдельно для каждого вида работы. По умолчанию в разделе **Format** главного меню для опции **Screen Style Environment** установлено значение **Working**, т.е. пользователь видит экранную версию документа. Чтобы увидеть, как будет выглядеть документ при печати, необходимо выбрать для этой опции значение **Printout**. После этого, установив "птичку" у опции **Show Page Breaks** в разделе **Format**, можно посмотреть разбивку документа на страницы. В зависимости от стиля всего документа, который определяется значением опции **Style Sheet** в разделе **Format**, кроме **Working** и **Printout** возможны и другие значения опции **Screen Style Environment**, например, **Presentation**, **Condensed** или **Slide Show**. В режиме **Presentation** текст печатается более крупными буквами, что позволяет представить документ в виде, удобном для демонстрации в лекционном зале при проектировании его на большой экран с помощью мультимедийного проектора. Если требуется поместить на экране как можно больше текста, используется режим **Condensed**. Режим **Slide Show** позволяет представить документ в виде последовательности слайдов, что особенно удобно при презентациях.р

Стиль документа при печати определяется опцией **Printing Style Environment** в разделе **Format** главного меню. По умолчанию для нее установлено значение **Printing**. Выбирая другое значение для этой опции, например, **Working**, при распечатке документа получим его в том же виде, как и на экране монитора.

4. Численные расчеты в системе *Mathematica*

4.1. Основные арифметические операции

Mathematica обеспечивает выполнение всех стандартных арифметических операций – изменение знака, сложение, вычитание, умножение, деление, возведение в степень. Кроме того, вычисляется факториал и произведение матриц. Все эти операции перечислены в таблице вместе с их обозначениями и примерами.

Операция	Обозначение	Пример
Изменение знака	-	`-a`
Сложение	+	`a + b`
Вычитание	-	`a - b`
Умножение	`*` или **пробел**	`a * b` или `a b`
Деление	`/` или `Ctrl + /`	`a / b` или $\frac{a}{b}$
Возведение в степень	`^` или `Ctrl + ^`	`a^b` или a^b
Извлечение квадратного корня	`^` или `Ctrl + 2`	`a^(1/2)` или $\sqrt{a}$
Факториал	!	`a!`
Матричное умножение	.	`a . b`

Следует отметить, что в системе *Mathematica* часто существует несколько эквивалентных обозначений одной и той же операции. Например, для обозначения умножения можно использовать звездочку " * " или пробел, хотя *Mathematica* предпочитает использовать пробел, как видно из следующего примера.

In[42]:= `a * b`

Out[42]= `a b`

Аналогичная ситуация имеет место с операцией деления. Вводя выражение *a / b*, получаем:

In[43]:= `a / b`

Out[43]= $\frac{a}{b}$

Как видим, для отношения $\frac{a}{b}$ *Mathematica* использует стандартное обозначение. Говорят, что выражение ***a / b*** является одномерным, поскольку все символы вводятся в одной строке, тогда как выражение вида $\frac{a}{b}$ является двумерным. Чтобы ввести двумерное выражение с помощью клавиатуры, нужно последовательно нажать клавиши ***a*** **Ctrl+/** ***b*** вместо простого набора ***a/b***, причем клавиша " **/** " нажимается при нажатой клавише **Ctrl**. Аналогично, возведение в степень производится путем нажатия клавиш "**^**" или **Ctrl+^**. Для ввода более сложных двумерных выражений можно использовать дополнительные панели с клавишами, которые открываются путем выбора одного из значений опции **Palettes** в разделе **File** главного меню, например,

File ↦ Palettes ↦ BasicInput.

При отсутствии в выражении скобок арифметические операции выполняются в стандартном порядке: сначала производится изменение знака числа, затем выполняется возведение в степень, потом умножение и деление, и, наконец, сложение и вычитание. Наличие у каждой операции своего приоритета позволяет в некоторых случаях опускать скобки, не нарушая порядка вычислений. Например, при возведение числа в отрицательную степень показатель степени не обязательно заключать в скобки.

```
In[44]:= 2.15 10^-1 + 0.785
Out[44]= 1.
```

4.2. Точные и приближенные вычисления

В системе *Mathematica* используются четыре вида численных данных, приведенные в следующей таблице:

Тип	Описание	Пример
Integer	Целое число	5
Real	Действительное число вида *nn.mm* (содержит десятичную точку)	4.6
Rational	Рациональное число вида $\frac{m}{n}$, где m и n – целые числа	$\frac{2}{3}$
Complex	Комплексное число вида $x + i\,y$, где x и y – целые, действительные или рациональные числа	2 + 5.1 I

При этом целые и рациональные числа (а также иррациональные) считаются заданными точно. Работая с ними, *Mathematica* выдает точный результат, даже если получается очень длинное выражение. Примеры:

```
In[45]:= 1/3 + 24 / 37
Out[45]= 109/111

In[53]:= 28^198
Out[53]= 3445801103611154449738188803252480961371719219593365086260817543380509968554906947071991018134081673781447258859538278927960635480916000563994070700308764524770279907989278341538962290009971045075964030296179743077331962422080719245804899506945304178072564098596458364935783778464720486 4
```

Действительные числа, содержащие десятичную точку, считаются заданными приближенно и имеют тип **Real**. Так как из приближенного числа нельзя получить точное, при обработке выражений, содержащих действительные числа, *Mathematica* выдает приближенный результат. Пример:

```
In[54]:= 1 / 3 + 0.479

Out[54]= 0.812333
```

Для обозначения мнимой единицы, входящей в состав комплексного числа, в системе *Mathematica* зарезервирована заглавная буква **I**. Кроме того, в третьей и последующих версиях мнимая единица обозначается символом i, что позволяет приблизить написание комплексных чисел к традиционному виду. Если действительная и мнимая части комплексного числа заданы точно, то и само число считается заданным точно. Если же мнимая часть комплексного числа есть точный ноль, то число не является комплексным, и его тип совпадает с типом действительной части. Убедиться в этом можно с помощью функции **Head**, которая выдает заголовок любого выражения (заголовком числа является его тип). Заметим, что понятие выражения и его структура подробно рассматриваются в разделе 8.1. Примеры:

```
In[55]:= Head[12]

Out[55]= Integer

In[56]:= Head[0.325]

Out[56]= Real

In[57]:= Head[0.35 I]

Out[57]= Complex

In[58]:= Head[0.325 + I 0]

Out[58]= Real
```

Если в последнем примере после нуля ввести десятичную точку, то мнимая часть комплексного числа уже не будет точным нулем, и тип числа изменится. Действительно,

```
In[59]:= Head[0.325 + i 0.]

Out[59]= Complex
```

Тот факт, что число задано точно, означает, что его можно вычислить с любой точностью. Конечно, невозможно получить бесконечную последовательность чисел, стоящих после десятичной точки в выражении $\sqrt{3} \approx 1.7320508\ldots$, например, так как возможности любого компьютера ограничены. Однако вычислительные алгоритмы, используемые в системе *Mathematica*, позволяют, в принципе, это сделать, и в этом смысле иррациональное число $\sqrt{3}$ считается заданным точно. Заданными точно считаются и известные математические постоянные, для которых в системе *Mathematica* зарезервированы специальные обозначения, приведенные в следующей таблице.

Постоянная	Обозначение	Название	Численное значение
π	**Pi** или π	Отношение длины окружности к ее диаметру	≈ 3.141592
e	**E** или e	Основание натуральных логарифмов	≈ 2.71828
i или j	**I** или i	Мнимая единица	$\sqrt{-1}$
$\pi/180$	**Degree** или °	Множитель для перевода градусов в радианы	≈ 0.017453
$(1+\sqrt{5})/2$	**GoldenRatio**	Золотое отношение	≈ 1.61803
γ	**EulerGamma**	Постоянная Эйлера	≈ 0.577216

Для преобразования точных чисел в приближенные используется функция **N**. Если эта функция вызывается с одним аргументом, то она выдает результат с точностью до шести значащих цифр, например:

```
In[60]:= N[2^9 / 7!]

Out[60]= 0.101587
```

В случае выражения вида **N[expr, m]**, где *m* – целое число, функция **N** пытается получить результат с точностью до *m* значащих цифр. Пример:

```
In[61]:= N[Sqrt[3], 43]

Out[61]= 1.7320508075688772935274463415058723669428052
```

По умолчанию все вычисления с числами, заданными приближенно, производятся с машинной точностью, значение которой задается постоянной **MachinePrecision**. Ее численное значение равно $53\log_{10}2$, т.е. приближенно 16 знаков. Естественно, в ходе вычислений по разным причинам возникают ошибки, и в полученном результате не все 16 значащих цифр могут быть верными. Определить точность результата можно с помощью функции **Precision**. Так, применяя эту функцию к последнему результату, получаем:

```
In[62]:= Precision[%]

Out[62]= 43.
```

Точность вычисления некоторой величины *x*, определяемая функцией **Precision**, равняется $\log_{10}(dx/x)$, где dx – абсолютная погрешность вычисления *x*. Именно поэтому мы получили для точности вычислений действительное, а не целое число. Для чисел, заданных точно, функция **Precision** выдает следующий результат:

```
In[63]:= Precision[Sqrt[3]]

Out[63]= ∞
```

Знак бесконечности подчеркивает то обстоятельство, что из точного результата можно получить приближенный с любой точностью.

Иногда получаемый результат содержит очень маленькую величину. Например, извлечем корень четвертой степени из (–2), а затем возведем полученный результат в четвертую степень.

In[64]:= `a1 = N[(-2)^(1/4), 20]`

Out[64]= `0.84089641525371454303 + 0.84089641525371454303 i`

In[65]:= `a1^4`

Out[65]= $-2.00000000000000000 + 0.\times 10^{-20}\,i$

Как видим, появилась очень малая мнимая часть. Поскольку вычисления производятся с конечной точностью, такие малые величины могут появляться в любом выражении. Чтобы удалить их, не нарушая самого выражения, используется функция **Chop**. По умолчанию она заменяет точным нулем все числа, которые меньше 10^{-11}.

In[66]:= `Chop[a1^4]`

Out[66]= `-2.00000000000000000`

В качестве второго аргумента у этой функции можно указать другое пороговое число, например, 10^{-5}.

In[67]:= `Chop[1.10^-6, 10^-5]`

Out[67]= `0`

In[68]:= `Chop[1.10^-4, 10^-5]`

Out[68]= `0.0001`

В системе *Mathematica* имеется еще несколько функций, позволяющих преобразовывать приближенно заданные числа в точные (см. таблицу ниже). Отметим среди них функцию **Rationalize**, которая заменяет действительное число x близким к нему рациональным числом $\frac{m}{n}$, причем по умолчанию числа x и $\frac{m}{n}$ считаются близкими, если выполняется неравенство

$$\left| x - \frac{m}{n} \right| < \frac{c}{n^2},$$

где $c = 10^{-4}$. Кроме того, можно непосредственно указать допустимую погрешность в качестве второго аргумента этой функции. Пример:

In[69]:= `Rationalize[Sqrt[3.], 10^-10]`

Out[69]= $\frac{191861}{110771}$

В приведенном примере мы находим рациональное число, которым можно заменить $\sqrt{3}$ с точностью до 10^{-10}.

Тип	Описание	Пример
Rationalize [x]	Преобразует приближенное число в рациональное	**Rationalize[0.3] → 3 / 10**
Ceiling[x]	Наименьшее целое, не меньшее *x*	**Ceiling[2.3] → 3**
Floor[x]	Наибольшее целое, не большее *x*	**Floor[3.6] → 3**
Round[x]	Ближайшее к *x* целое (округление)	**Round[3.7] → 4** **Round[5.3] → 5**
Chop[x]	Замена близких к нулю чисел точным нулем	**Chop[1. 10 ^ – 11] → 0**

4.3. Стандартные математические функции

Система *Mathematica* оперирует со всеми стандартными математическими функциями. В качестве их имен используются либо стандартная аббревиатура, например, **Cos**, либо соответствующее английское слово, например, **Factorial**, причем все имена пишутся с заглавной буквы. Если название функции состоит из нескольких слов, то слова пишутся одно за другим без пробелов, причем каждое слово пишется с заглавной буквы. Аргументы функций обязательно заключаются в квадратные скобки. Примеры некоторых функций приведены в таблице.

Обозначение	Название
Sqrt[x]	Квадратный корень ($\sqrt{x}$)
Exp[x]	Экспонента (e^x)
Log[x]	Натуральный логарифм ($ln\, x$)
Log[b, x]	Логарифм по основанию b ($\log_b x$)
Sin[x], Cos[x], Tan[x], Cot[x], Sec[x], Csc[x]	Тригонометрические функции
ArcSin[x], ArcCos[x], ArcTan[x], ArcCot[x], ArcSec[x], ArcCsc[x]	Обратные тригонометрические функции
Sinh[x], Cosh[x], Tanh[x], Coth[x], Sech[x], Csch[x]	Гиперболические функции
ArcSinh[x], ArcCosh[x], ArcTanh[x], ArcSech[x]	Обратные гиперболические функции
Abs[z]	Модуль комплексного числа | z |
Arg[z]	Аргумент φ комплексного числа
Re[z], Im[z]	Действительная и мнимая части комплексного числа
Mod[n, m]	Остаток деления *n* на *m*
Min[x, y, ...]	Минимальное число из множества *x*, *y*, ...
Max[x, y, ...]	Максимальное число из множества *x*, *y*, ...

По умолчанию предполагается, что аргументы тригонометрических функций выражены в радианах. Для перевода градусов в радианы есть встроенная постоянная **Degree**, которую следует включать в аргумент тригонометрической функции в качестве множителя.

```
In[70]:= Cos[31. Degree]

Out[70]= 0.857167
```

Следует отметить, что *Mathematica* может работать не только со стандартными, но и со специальными математическими функциями. Информацию о них можно найти в подразделе **Mathematical Functions** раздела **Built-in Functions** программы помощи **Help Browser**. Кроме того, компанией Wolfram Research создан специальный сайт http://functions.wolfram.com, на котором можно найти полную информацию о всех известных функциях, их свойствах, соотношениях между ними. Имеющиеся там формулы написаны в кодах системы *Mathematica* и могут быть легко скопированы, а затем вставлены в любой ее документ.

4.4. Численное решение уравнений

Одна из важных особенностей системы *Mathematica* состоит в том, что она позволяет находить точные решения некоторых уравнений в символьной форме. В тех случаях, когда получить точное решение не удается, желательно найти хотя бы приближенное численное решение уравнения или системы уравнений. Для получения таких решений используются функции **NSolve** и **FindRoot**.

Функция **NSolve** позволяет найти численные значения всех корней полиномиального уравнения или системы уравнений. Чтобы выяснить правила обращения к этой функции, воспользуемся справкой.

```
In[71]:= ? NSolve

   NSolve[lhs==rhs, var] gives a list of numerical
      approximations to the roots of a polynomial equation.
      NSolve[{eqn1, eqn2, ... }, {var1, var2, ... }]
      solves a system of polynomial equations. More...
```

Как видим, у функции **NSolve** должно быть два аргумента, разделенных запятой. Первый аргумент представляет собой уравнение или список уравнений, заключенных в фигурные скобки и разделенных запятыми, а второй – имя переменной или список имен, относительно которых необходимо разрешить уравнение или систему соответственно. Следует запомнить, что признаком уравнения в системе *Mathematica* является двойной знак равенства "==", причем символы "=" набираются без пробела. В качестве примера найдем корни уравнения: $x^4 + x^2 + 1 = 0$.

```
In[72]:= NSolve[x^4 + x^2 + 1 == 0, x]

Out[72]= {{x → -0.5 - 0.866025 i}, {x → -0.5 + 0.866025 i},
          {x → 0.5 - 0.866025 i}, {x → 0.5 + 0.866025 i}}
```

Функция **NSolve** выдает список из четырех комплексных корней, которые представлены в виде *правил подстановки*, т.е. конструкций вида: **имя → значение**, причем каждое решение заключено в фигурные скобки. Это

значит, что каждое решение является списком, который состоит из одного элемента при решении одного уравнения и из *n* элементов при решении системы *n* уравнений. Это легко видеть из следующего примера.

```
In[73]:= sol1 = NSolve[ {x^2 + y == 0, y^3 - x == 2}, {x, y}]

Out[73]= {{x → 1.02362 - 0.639381 i, y → -0.638991 + 1.30897 i},
          {x → 1.02362 + 0.639381 i, y → -0.638991 - 1.30897 i},
          {x → -0.102554 - 1.13685 i, y → 1.28191 - 0.233176 i},
          {x → -0.102554 + 1.13685 i, y → 1.28191 + 0.233176 i},
          {x → -0.921067 + 0.453263 i, y → -0.642916 + 0.834971 i},
          {x → -0.921067 - 0.453263 i, y → -0.642916 - 0.834971 i}}
```

Список решений, которому мы присвоили имя *sol1*, состоит из шести элементов, каждый из которых также есть список, состоящий из двух правил подстановки для переменых *x* и *y*. Соответственно, если решается одно уравнение, которое имеет один корень, то список решений будет состоять из одного элемента, который в свою очередь является списком, содержащим одно правило подстановки. Поэтому решение будет заключаться в двойные фигурные скобки, как видно из следующего примера:

```
In[74]:= NSolve[3 x + 8 == 0, x]

Out[74]= {{x → -2.66667}}
```

Следует отметить, что найденные решения могут быть подставлены в любое выражение. Подставим, например, второй корень из решения *sol1* в выражение $y^3 - x$. Напомним, что для выделения элементов списка используются двойные квадратные скобки.

```
In[75]:= y^3 - x /. sol1[[2]]

Out[75]= 2. + 9.99201 × 10^-16 i
```

Хотя, согласно второму уравнению системы, при подстановке корней это выражение должно равняться 2, мы получаем дополнительное чисто мнимое слагаемое, которое очень мало. Это обычная ситуация, так как при выполнении численных расчетов мы находим результат с определенной точностью. Действительно, применяя к последнему результату функцию **Precision**, получаем:

```
In[76]:= Precision[%]

Out[76]= MachinePrecision
```

Это означает, что по умолчанию, выполняя численные расчеты, *Mathematica* оперирует шестнадцатью значащими цифрами. Однако, как видно из последнего примера, получаемый результат может быть менее точным, т.е. уравнение удовлетворяется с точностью до 10^{-15}. Чтобы увеличить точность вычисления корней уравнения или системы, достаточно в качестве третьего аргумента функции **NSolve** указать количество значащих цифр, которые должны использоваться при расчетах. Так, в следующем примере вычисление корней уравнения производится с точностью до 20 значащих цифр.

```
In[77]:= sol2 = NSolve[x^3 - 3 x^2 + 7 == 0, x, 20]

Out[77]= {{x → -1.2790187861665935795},
          {x → 2.1395093930832967897 - 0.9462795415600985046 i},
          {x → 2.1395093930832967897 + 0.9462795415600985046 i}}
```

Подставляя, например, третий корень в левую часть уравнения, убеждаемся, что оно удовлетворяется с точностью до 10^{-18}.

```
In[78]:= x^3 - 3 x^2 + 7 /. sol2[[3]]

Out[78]= 0. × 10^-18 + 0. × 10^-18 i
```

Следует отметить, что введение третьего аргумента у функции **NSolve**, который определяет точность вычислений, эквивалентно применению функции **N** к результату, даваемому функцией **Solve**, которая используется для нахождения корней уравнений в символьном виде. Соответствующая команда может быть записана в виде:

```
In[79]:= N[ Solve[x^3 - 3 x^2 + 7 == 0, x], 20]

Out[79]= {{x → -1.2790187861665935795},
          {x → 2.1395093930832967897 - 0.9462795415600985046 i},
          {x → 2.1395093930832967897 + 0.9462795415600985046 i}}
```

Для решения произвольных уравнений и их систем используется функция **FindRoot**, которая находит только одно численное решение при каждом обращении к ней. Правила обращения к ней легко видеть из следующего сообщения.

```
In[80]:= ? FindRoot

   FindRoot[lhs==rhs, {x, x0}] searches for
      a numerical solution to the equation lhs==rhs,
      starting with x=x0. FindRoot[{eqn1, eqn2, ... },
      {{x, x0}, {y, y0}, ... }] searches for a numerical
      solution to the simultaneous equations eqni. More...
```

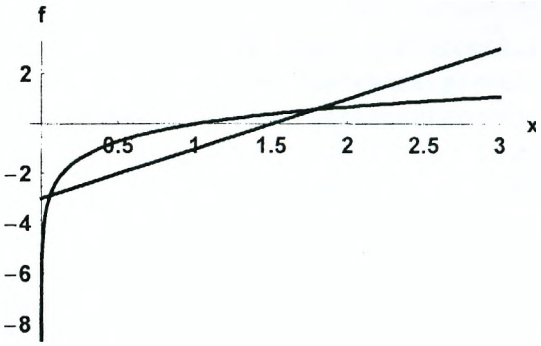
Следует подчеркнуть, что при обращении к функции **FindRoot** необходимо указать приближенное значение корня *x0*, с которого начинается поиск его точного значения. Найти это приближенное значение проще всего графически. В качестве примера найдем корни уравнения

$$\ln x = 2x - 3$$

на отрезке $x \in [0, 3]$.

Чтобы найти приближенные значения корней, построим графики левой и правой частей уравнения.

```
In[81]:= Plot[{Log[x], 2 x - 3}, {x, 0, 3},
           PlotStyle → Thickness[0.008], AxesLabel → {"x", "f"}];
```



Легко видеть, на заданном отрезке есть два корня (значения x, соответствующие точкам пересечения кривых $y = \ln x$ с прямой $y = 2x - 3$). Из графика находим приближенные значения корней.

```
In[82]:= x1 = 0.1; x2 = 1.8;
```

Точные значения корней находим с помощью функции **FindRoot**.

```
In[83]:= FindRoot[Log[x] == 2 x - 3, {x, x1}]

Out[83]= {x → 0.0556483}

In[84]:= FindRoot[Log[x] == 2 x - 3, {x, x2}]

Out[84]= {x → 1.79154}
```

Как видим, при отыскании обоих корней используется одна и та же процедура с той лишь разницей, что используются различные стартовые значения для x. Таким образом, выбор корня, находимого функцией **FindRoot**, осуществляется путем подбора для него соответствующего начального значения. Если начальное значение выбрано неудачно, то корень может быть не найден, как видно из следующего примера.

```
In[85]:= FindRoot[Log[x] == 2 x - 3, {x, 0.5}]

    FindRoot::jsing :
      Encountered a singular Jacobian at the point {x} =
        {0.5}. Try perturbing the initial point(s). More

Out[85]= {x → 0.5}
```

В таких случаях необходимо последовать выведенной подсказке и попробовать изменить начальное значение для x.

Следует помнить, что функция **FindRoot** может работать только в том случае, если решаемое уравнение не содержит неизвестных параметров, т.е. при подстановке в уравнение вместо x числа в левой и правой частях уравнения также получаются числа. Следующий пример показывает, что получается в противном случае.

```
In[86]:= FindRoot[Cos[x] == k x, {x, 0.1}]
```

```
FindRoot::nlnum :
 The function value {0.995004 - 0.1 k} is not a list of
   numbers with dimensions {1} at {x} = {0.1}. More...
```

Out[86]= `FindRoot[Cos[x] == k x, {x, 0.1}]`

Если у функции **FindRoot** указано одно начальное значение для переменной, то при нахождении корней используется метод касательных. Этот метод является достаточно эффективным и быстро приводит к нужному корню, если в уравнение входят функции, производные которых могут быть вычислены в символьном виде. Однако иногда это условие оказывается невыполненным, и метод касательных не может быть применен. Тогда для нахождения корня можно применить метод хорд, который не требует вычисления производных. Указанием для системы *Mathematica* применить метод хорд для поиска решений является задание двух стартовых значений, например,

In[87]:= `FindRoot[Log[x] == 2 x - 3, {x, 0.02, 0.2}]`

Out[87]= `{x → 0.0556483}`

Если второй аргумент функции **FindRoot** указан в виде **{x, xstart, xmin, xmax}**, то это означает, что мы ищем решение на отрезке **{xmin, xmax}**, используя в качестве начального приближения для *x* значение **xstart**. Если в ходе вычислений какое-либо промежуточное значение *x* оказывается за пределами отрезка **{xmin, xmax}**, то вычисления прекращаются и выдается последний результат и соответствующее сообщение. Пример:

In[88]:= `FindRoot[Sin[x^3] == x - 2, {x, 0.8, 0, 2}]`

```
FindRoot::reged :
 The point {2.} is at the edge of the search
   region {0., 2.} in coordinate 1 and the computed
   search direction points outside the region. More...
```

Out[88]= `{x → 2.}`

Таким образом, и в этом случае результат работы функции **FindRoot** во многом зависит от выбора начального приближения для корня **xstart**.

Если коэффициенты уравнения и начальное значение для корня являются действительными числами, то функция **FindRoot** также ищет только действительные корни. Чтобы найти комплексный корень, достаточно указать в качестве начального значения комплексное число, например,

In[89]:= `FindRoot[Sin[x^3] == x - 2, {x, -0.3 + 2 I}]`

Out[89]= `{x → -0.625976 + 1.53159 i}`

Для решения системы уравнений необходимо в качестве второго и последующих аргументов функции **FindRoot** указать в произвольном порядке списки, каждый из которых содержит искомую переменную и используемое для нее начальное приближение. Пример:

In[90]:= `FindRoot[{Sin[x] == y^2 - 1, Cos[y] == x}, {x, 0.3}, {y, 1}]`

Out[90]= `{x → 0.386857, y → 1.17358}`

И в этом случае нужно следить за тем, чтобы в уравнениях не оказалось других параметров, кроме переменных, относительно которых решается система.

4.5. Численное решение дифференциальных уравнений

Для численного решения дифференциальных уравнений и их систем используется функция **NDSolve**, которая вызывается по крайней мере с тремя аргументами: **NDSolve[eqns, y, {x, xmin, xmax}]**, где *eqns* – список, в котором содержатся дифференциальное уравнение и необходимое число граничных или начальных условий, *y* – имя функции, относительно которой решается уравнение, *x* – имя независимой переменной, причем решение ищется на отрезке от *xmin* до *xmax*. В качестве примера найдем решение дифференциального уравнения

$$y'' + 2y' + 12xy = 0$$

с начальными условиями $y(0) = 1$, $y'(0) = 0$ на отрезке $x \in [0, 3]$ и назовем его *sol3*.

```
In[91]:= sol3 = NDSolve[{y''[x] + 2 y'[x] + 12 x y[x] == 0,
                         y[0] == 1, y'[0] == 0}, y[x], {x, 0, 3}]

Out[91]= {{y[x] → InterpolatingFunction[{{0., 3.}}, <>][x]}}
```

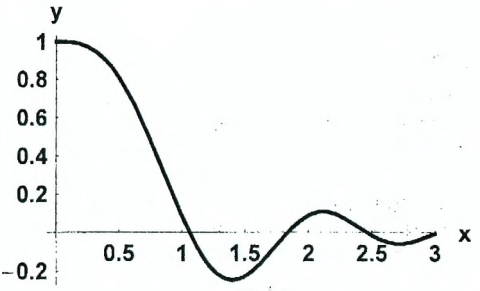
Обратите внимание на наличие двойного знака равенства как в самом дифференциальном уравнении, так и в уравнениях, выражающих начальные условия. Это общее правило записи всех уравнений. Кроме того, функция и все ее производные должны быть записаны с аргументами, заключенными в квадратные скобки: **y[x]**, **y'[x]** и т.д. Для производных можно использовать стандартные обозначения, набирая после *y* необходимое число штрихов (для обозначения второй производной нельзя использовать кавычки). Старшие производные удобно записывать с помощью оператора дифференцирования **D**. Например, *n*-ая производная функции *y(x)* записывается в виде: **D[y[x], {x, n}]**. Следует подчеркнуть также, что ни в самом дифференциальном уравнении, ни в начальных условиях не должно содержаться никаких неизвестных параметров, в противном случае численное решение уравнения не может быть найдено и выдается сообщение об ошибке. Пример:

```
In[92]:= NDSolve[{y''[x] + 2 k y'[x] + 12 x y[x] == 0,
                  y[0] == 1, y'[0] == 0}, y[x], {x, 0, 3}]
NDSolve::ndnum : Encountered non-numerical
   value for a derivative at x == 0.`. More

Out[92]= NDSolve[{12 x y[x] + 2 k y'[x] + y''[x] == 0,
                  y[0] == 1, y'[0] == 0}, y[x], {x, 0, 3}]
```

Решение дифференциального уравнения выдается в виде правила подстановки для искомой функции *y(x)*, в правой части которого находится объект, называемый интерполяционной функцией (**InterpolatingFunction**). У этой функции есть два аргумента, первый из которых указывает ее область определения, а второй представляет собой таблицу значений функции $y(x_i)$ в различных точках отрезка *[xmin, xmax]*. Эта таблица достаточно большая и потому обычно не выводится, а заменяется скобками < >. С интерпо-

ляционной функцией можно обращаться как с обычной функцией, например, дифференцировать, интегрировать, построить ее график. Так, график функции, найденной в *sol3*, имеет вид:

```
In[94]:= Plot[y[x] /. sol3, {x, 0, 3},
          AxesLabel → {"x", "y"}, PlotStyle → Thickness[0.01]];
```

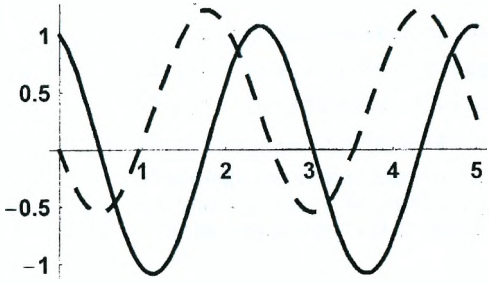


При решении системы дифференциальных уравнений в качестве первого аргумента функции **NDSolve** должен быть список, состоящий из дифференциальных уравнений и уравнений, выражающих граничные условия. Вторым аргументом является список функций, относительно которых необходимо разрешить систему уравнений, а третьим – список, состоящий из имени независимой переменной, ее начального и конечного значений. В качестве примера рассмотрим следующую систему двух дифференциальных уравнений:

```
In[95]:= sol4 = NDSolve[{x'[t] - 3 y[t] == -1, y'[t] + 2 x[t] == 0,
                         x[0] == 1, y[0] == 0}, {x, y}, {t, 0, 5}]
Out[95]= {{x → InterpolatingFunction[{{0., 5.}}, <>],
           y → InterpolatingFunction[{{0., 5.}}, <>]}}
```

Обе искомые функции *x(t)* и *y(t)* находятся в виде интерполяционных функций. Соответствующие графики имеют вид:

```
In[97]:= Plot[{x[t] /. sol4[[1]], y[t] /. sol4}, {t, 0, 5},
          PlotStyle → {{Thickness[0.01]},
                       {Dashing[{0.05}], Thickness[0.01]}}];
```



На рисунке сплошная кривая соответствует функции *x(t)*, а штриховая – функции *y(t)*.

Решая систему дифференциальных уравнений, функция **NDSolve** выдает список решений, причем каждое решение само является списком правил подстановки для всех искомых функций. Именно поэтому оба найденные выше решения *sol3* и *sol4* заключены в двойные фигурные скобки. При подстановке вида ***x*[*t*] /. sol4**, например,

```
In[98]:= x[t] /. sol4
Out[98]= {InterpolatingFunction[{{0., 5.}}, <>][t]}
```

одни фигурные скобки переносятся на результат, т.е. получается список, единственным элементом которого является интерполяционная функция. Как видно из предыдущего примера, эти дополнительные скобки не влияют на процесс построения графика с помощью функции **Plot**. Однако они оказываются существенными, если мы хотим, например, найти наименьшее значение параметра *t*, при котором функция *x(t)* имеет локальный минимум. Для этого можно воспользоваться функцией **FindMinimum**.

```
In[99]:= FindMinimum[ x[t] /. sol4, {t, 1.2}]
FindMinimum::nnum : The function value
   {-1.06161} is not a number at {t} = {1.2}. More...
Out[99]= FindMinimum[x[t] /. sol4, {t, 1.2}]
```

Поскольку в результате подстановки **x[t] /. sol4** первым аргументом функции **FindMinimum** оказывается список, содержащий интерполяционную функцию, а не сама эта функция, ее минимум не вычисляется, а выдается сообщение об ошибке. Чтобы избавиться от дополнительных фигурных скобок, с помощью двойных квадратных скобок выделяем первый и единственный элемент списка *sol4* и используем его для подстановки вместо *x(t)*.

```
In[100]:= FindMinimum[x[t] /. sol4[[1]], {t, 1.2}]
Out[100]= {-1.08012, {t → 1.1243}}
```

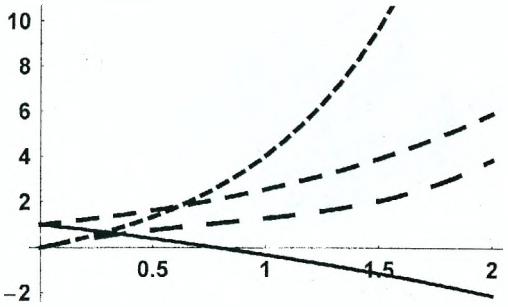
Теперь функция **FindMinimum** легко находит минимальное значение функции *x(t)* в окрестности точки *t* = 1.2, а также соответствующее значение *t*.

В следующем примере система двух уравнений относительно функций *x(t)* и *y(t)* имеет два решения.

```
In[101]:= sol5 = NDSolve[{x'[t]^2 - y[t] == 1, x[t]^2 - y'[t] == -1,
             x[0] == 1, y[0] == 0}, {x, y}, {t, 0, 2}]
Out[101]= {{x → InterpolatingFunction[{{0., 2.}}, <>],
            y → InterpolatingFunction[{{0., 2.}}, <>]},
           {x → InterpolatingFunction[{{0., 2.}}, <>],
            y → InterpolatingFunction[{{0., 2.}}, <>]}}
```

В таком случае при построении графиков функций использование двойных квадратных скобок для выделения решений является обязательным. Соответствующая команда может быть записана в виде:

```
In[103]:= Plot[{x[t] /. sol5[[1]], y[t] /. sol5[[1]],
    x[t] /. sol5[[2]], y[t] /. sol5[[2]]}, {t, 0, 2},
  PlotStyle → {{Thickness[0.008]},
      {Dashing[{0.06}], Thickness[0.01]},
      {Dashing[{0.04}], Thickness[0.01]},
      {Dashing[{0.02}], Thickness[0.01]}}];
```



Чтобы убедиться в необходимости двойных квадратных скобок, попробуйте построить график любой из функций, найденных в *sol5*, без них.

Функция **NDSolve** позволяет находить численные решения не только обыкновенных дифференциальных уравнений, но и некоторых дифференциальных уравнений в частных производных. При этом в список уравнений, который является первым аргументом этой функции, должно быть включено достаточное количество начальных и граничных условий. В качестве примера рассмотрим одномерное уравнение теплопроводности, определяющее зависимость температуры в различных поперечных сечениях однородного теплоизолированного стержня от времени *t*, если температуры его оснований изменяются по заданному закону. Температура является функцией двух переменных *T(x,t)* (предполагается, что ось стержня совпадает с осью *Ox*) и удовлетворяет следующему дифференциальному уравнению:

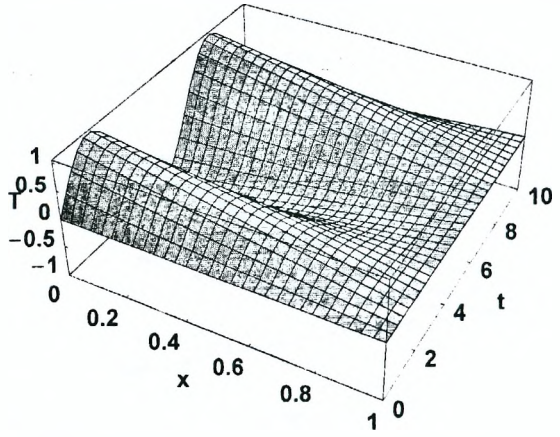
$$\frac{\partial T}{\partial t} = k \frac{\partial^2 T}{\partial x^2}.$$

Предположим, что коэффициент теплопроводности *k=1*, начальная температура во всех точках стержня одинакова и равна нулю, температура одного конца стержня изменяется по закону синуса, а второго – поддерживается равной нулю. Считая длину стержня равной 1, находим решение уравнения теплопроводности при заданных начальных и граничных условиях.

```
In[104]:= sol6 = NDSolve[
    {D[T[x, t], t] == D[T[x, t], {x, 2}], T[x, 0] == 0,
      T[0, t] == Sin[t], T[1, t] == 0},
    T, {x, 0, 1}, {t, 0, 10}]
Out[104]= {{T → InterpolatingFunction[{{0., 1.}, {0., 10.}}, <>]}}
```

Найденная зависимость $T(x,t)$ определяет поверхность в трехмерном пространстве (x t T), которую можно построить с помощью функции **Plot3D**.

In[105]:=
```
Plot3D[T[x, t] /. sol6[[1]], {x, 0, 1}, {t, 0, 10},
  PlotPoints → 30, AxesLabel → {"x", "t", "T"}];
```



Заметим, что при подстановке найденного решения в качестве первого аргумента функции **Plot3D** следует обязательно выделить первый элемент с списке решений, используя, например, двойные квадратные скобки.

4.6. Обработка численных данных

На практике часто возникает ситуация, когда некоторая функция $y = y(x)$ задана в виде таблицы, в которой содержатся пары чисел вида (x_j, y_j). Такая таблица может появиться как результат измерений, выполненных в ходе какого-либо эксперимента. Обычно из физических соображений можно сделать некоторые предположения относительно вида этой функции. Однако всегда остаются свободные параметры, которые следует подобрать так, чтобы получаемая в результате функция $y = y(x)$ соответствовала имеющимся табличным данным. Для решения подобных задач в системе *Mathematica* имеется целый набор встроенных функций. Рассмотрим работу некоторых из них на конкретных примерах.

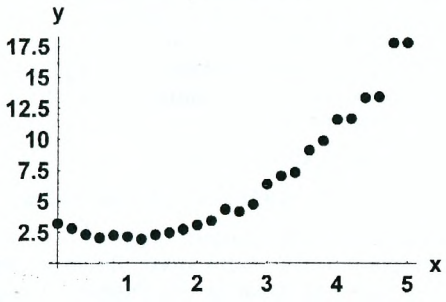
Предположим, что в изучаемом явлении две физические величины связаны соотношением: $y = a + b\,x + c\,x^2$. Задавая значения постоянных $a = 3$, $b = -2$, $c = 1$, с помощью встроенной функции **Table** генерируем таблицу пар чисел (x, y), где переменная x изменяется от 0 до 5 с шагом 0.2.

In[122]:=
```
dat1 =
  Table[{x, (3 - 2 x + x^2) (1 + 0.1 Random[Real, {-1, 1}])},
                                        {x, 0, 5., 0.2}]
```

```
Out[122]= {{0, 3.21673}, {0.2, 2.80947},
          {0.4, 2.31171}, {0.6, 2.04584}, {0.8, 2.24358},
          {1., 2.15015}, {1.2, 1.94362}, {1.4, 2.29277},
          {1.6, 2.45886}, {1.8, 2.73615}, {2., 3.11035},
          {2.2, 3.44792}, {2.4, 4.3558}, {2.6, 4.19676},
          {2.8, 4.77535}, {3., 6.42731}, {3.2, 7.10533},
          {3.4, 7.37769}, {3.6, 9.17338}, {3.8, 9.94059},
          {4., 11.6462}, {4.2, 11.7192}, {4.4, 13.3825},
          {4.6, 13.4717}, {4.8, 17.8621}, {5., 17.8786}}
```

Поскольку любое измерение производится с некоторой погрешностью, мы добавили к каждому вычисляемому значению y_j некоторое возмущение, используя функцию **Random**, которая генерирует действительные случайные числа из интервала [–1, 1]. Чтобы лучше представить себе полученную зависимость, с помощью функции **ListPlot** отметим точки (x_j, y_j) на координатной плоскости *xOy*.

```
In[123]:= p1 = ListPlot[dat1, PlotStyle → PointSize[0.03],
                        AxesLabel → {"x", "y"}];
```



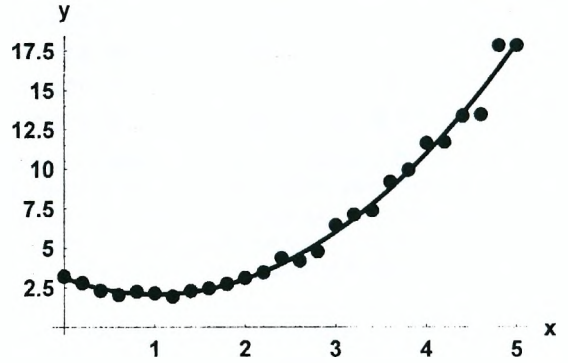
Заметим, что мы задали коэффициенты *a*, *b*, *c* только для того, чтобы смоделировать результаты измерений. На практике обычно встречаются с обратной задачей, когда эти коэффициенты нужно определить, имея результаты измерений. Это можно сделать с помощью функции **Fit**, например,

```
In[124]:= y1 = Fit[dat1, {1, x, x^2}, x]

Out[124]= 3.11458 - 2.04028 x + 1.00325 x^2
```

Ее второй аргумент, заключенный в фигурные скобки, содержит набор функций, линейная комбинация которых $a*1+bx+cx^2$ и должна соответствовать имеющимся численным данным. При этом имя списка численных данных *dat1* помещается в качестве первого аргумента функции **Fit**, а ее третий аргумент указывает имя независимой переменной. Наилучшие значения коэффициентов в выражении $y=a+bx+cx^2$ определяются по методу наименьших квадратов. Как видим, в полученном выражении для *y1* значения коэффициентов *a*, *b*, *c* близки к тем, которые мы использовали при генерации данных. Небольшие отличия возникли вследствие возмущения численных данных функцией **Random**. На следующем рисунке построен

график найденной функции вместе с точками, изображающими численные данные.

```
In[125]:= p2 = Plot[y1, {x, 0, 5}, PlotStyle → Thickness[0.01],
             DisplayFunction → Identity];

In[126]:= Show[p1, p2, DisplayFunction → $DisplayFunction];
```



Выражение для *y=y(x)*, получаемое с помощью функции **Fit**, зависит от набора функций, линейная комбинация которых используется для аппроксимации *y(x)*. Чем лучше этот набор соответствует реальной зависимости *y=y(x)*, тем точнее находятся значения неизвестных коэффициентов. Попробуем, например, искать функцию *y(x)* в виде $y = a + b\,x + c\,x^2 + d\,x^3 + e\,x^4$. Соответствующая команда имеет вид:

```
In[129]:= Fit[dat1, {1, x, x^2, x^3, x^4}, x]

Out[129]= 3.21188 - 2.54812 x + 1.49981 x^2 - 0.160385 x^3 + 0.0163286 x^4
```

Хотя в полученном выражении коэффициенты при x^3 и x^4 достаточно малы, значения остальных коэффициентов стали хуже. Легко убедиться в том, что добавление слагаемых с более высокими степенями *x* также не улучшает результат.

Итак, используя команду **Fit**, мы задаем набор функций и ищем такую их линейную комбинацию, которая аппроксимирует табличные данные наилучшим образом. Часто искомая функция *y(x)* является более сложной и не может быть представлена в виде линейной комбинации других функций. Рассмотрим, например, функцию $y = 2\,x^2\,e^{-x}$. Создадим таблицу численных данных этой функции, добавляя к каждому ее значению случайное возмущение из интервала [0, 0.1].

```
In[130]:= dat2 = Table[{x, 2 x^2 Exp[-x] + Random[Real, {0, 0.1}]},
              {x, 0, 4, 0.3}]

Out[130]= {{0, 0.00368491}, {0.3, 0.210032},
          {0.6, 0.488498}, {0.9, 0.717734}, {1.2, 0.944747},
          {1.5, 1.10004}, {1.8, 1.09355}, {2.1, 1.17095},
          {2.4, 1.05399}, {2.7, 1.02466}, {3., 0.918602},
          {3.3, 0.884024}, {3.6, 0.711484}, {3.9, 0.674949}}
```

Пусть из некоторых соображений мы можем предположить, что численные данные определяют функцию вида: $y = a x^2 e^{bx}$. Тогда задача поиска функции сводится к определению наилучших значений постоянных *a* и *b*. Это можно сделать с помощью функции **FindFit**, которая вызывается с четырьмя аргументами. На первом месте располагаются численные данные *dat2*, на втором – искомая функция, на третьем – список неизвестных коэффициентов, содержащихся в функции, а на четвертом – независимая переменная. В результате получаем:

In[131]:=
```
coeff1 = FindFit[dat2, a x^2 Exp[b x], {a, b}, x]
```

Out[131]=
```
{a → 2.22529, b → -1.02133}
```

Теперь функцию, аппроксимирующую численные данные, можно представить в виде:

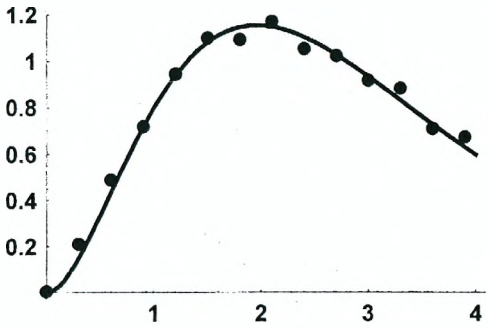
In[132]:=
```
y1 = a x^2 Exp[b x] /. coeff1
```

Out[132]=
```
2.22529 E^(-1.02133 x) x^2
```

Соответствующий график вместе с экспериментальными точками показан на следующем рисунке.

In[136]:=
```
p3 = ListPlot[dat2, PlotStyle → PointSize[0.03],
                    DisplayFunction → Identity];
p4 = Plot[y1, {x, 0, 4}, PlotStyle → Thickness[0.01],
                    DisplayFunction → Identity];
```

In[138]:=
```
Show[p3, p4, DisplayFunction → $DisplayFunction];
```



Как и функция **Fit**, функция **FindFit** находит наилучшие значения неизвестных коэффициентов, используя метод наименьших квадратов. Получаемый при этом результат зависит от того, насколько удачно выбран вид искомой функции. Найдем, например, коэффициенты *a*, *b*, *c* и *d* у функции вида $y = (a + b x + c x^2) e^{dx}$, используя численные данные *dat2*.

In[143]:=
```
FindFit[dat2, (a + b x + c x^2) Exp[d x], {a, b, c, d}, x]
```

Out[143]=
```
{a → -0.0667837, b → 1.24746, c → -0.24687, d → -0.134002}
```

Найденные коэффициенты существенно отличаются от полученных ранее. Построив график функции $y = (a + b x + c x^2) e^{dx}$ с этими коэффициентами, легко убедиться в том, что выбранная функция плохо соответствует численным данным.

Отметим еще раз, что команды **Fit** и **FindFit** позволяют найти выражения для функций из выбранного класса, аппроксимирующих численные данные наилучшим, с точки зрения метода наименьших квадратов, образом.

5. Символьные вычисления

5.1. Преобразование алгебраических выражений с помощью встроенных функций

Для преобразавания алгебраических выражений и формул в системе *Mathematica* имеется целый ряд встроенных функций. Их перечень и описание можно найти в подразделах **Basic Algebra** и **Formula Manipulation** раздела **Algebraic Computation**, запустив программу помощи **Help Browser** и выбрав закладку **Built-in Functions**. Некоторые из этих функций приведены в таблице (*expr* обозначает алгебраическое выражение).

Обозначение	Назначение
Expand[expr]	Раскрыть скобки в *expr*, представив его в виде суммы
ExpandAll[expr]	Применить функцию **Expand** как к числителю, так и к знаменателю выражения *expr*
Factor[expr]	Представить *expr* в виде произведения множителей
Together[expr]	Привести сумму дробей в *expr* к общему знаменателю
Apart[expr]	Разложить *expr* на сумму простых дробей
Cancel [expr]	Сократить общие множители в числителе и знаменателе дроби *expr*
Simplify[expr]	Найти для *expr* простейшую форму, применяя к нему стандартные правила преобразования
FullSimplify[expr]	Упростить *expr*, применяя к нему более широкий спектр преобразований
Collect[expr, x]	Представить *expr* в виде полинома по степеням x
FactorTerms[expr, x]	Вынести за скобки множители, не зависящие от x
Coefficient[expr, x]	Коэффициент при множителе x в *expr*
Coefficient[expr, x, n]	Коэффициент при множителе $x\hat{}n$ в *expr*
Exponent[expr, form]	Максимальная степень множителя *form* в *expr*
Part[expr, n], expr[[n]]	n – й член в expr
Length[expr]	Общее число членов в *expr*
Numerator[expr]	Числитель рационального выражения *expr*
Denominator[expr]	Знаменатель рационального выражения *expr*
ExpandNumerator[expr]	Раскрыть скобки в числителе рационального выражения *expr*
ExpandDenominator[expr]	Раскрыть скобки в знаменателе *expr*
ComplexExpand[expr]	Раскрыть скобки, предполагая, что все переменные в *expr* являются действительными
PowerExpand[expr]	Раскрыть скобки, преобразуя $(x\,y)^n$ в $x^n\,y^n$

Рассмотрим несколько примеров преобразования алгебраических выражений, используя функции, приведенные в таблице. Сначала присвоим имя t_1 выражению $(x^2+y+1)^2(y^2-1)+y^3+1$, которое будем рассматривать как объект преобразования.

```
In[1]:= t1 = (x^2 + y + 1)^2 (y^2 - 1) + y^3 + 1
Out[1]= 1 + y^3 + (1 + x^2 + y)^2 (-1 + y^2)
```

Обратите внимание, что *Mathematica* не делает никаких преобразований введенного выражения самостоятельно, она только переписывает его, размещая числа и символы в стандартном порядке. Чтобы раскрыть скобки и представить выражение t_1 в виде суммы, применим к нему функцию **Expand**. Полученный результат обозначим через t_2.

```
In[2]:= t2 = Expand[t1]
Out[2]= -2 x^2 - x^4 - 2 y - 2 x^2 y + 2 x^2 y^2 + x^4 y^2 + 3 y^3 + 2 x^2 y^3 + y^4
```

Обратная операция, т.е. представление выражения t_2 в виде произведения множителей, выполняется функцией **Factor**.

```
In[3]:= t3 = Factor[t2]
Out[3]= (1 + y) (-2 x^2 - x^4 - 2 y + x^4 y + 2 y^2 + 2 x^2 y^2 + y^3)
```

С помощью функции **FactorTerms** вынесем в t_2 за скобки множитель, не зависящий от *x*.

```
In[6]:= FactorTerms[t2, x]
Out[6]= (1 + y) (-2 x^2 - x^4 - 2 y + x^4 y + 2 y^2 + 2 x^2 y^2 + y^3)
```

Функция **Collect** позволяет представить t_2 в виде полинома по степеням *x*.

```
In[7]:= Collect[t2, x]
Out[7]= -2 y + 3 y^3 + y^4 + x^4 (-1 + y^2) + x^2 (-2 - 2 y + 2 y^2 + 2 y^3)
```

Выделим в t_2 коэффициент при x^2.

```
In[8]:= Coefficient[t2, x^2]
Out[8]= -2 - 2 y + 2 y^2 + 2 y^3
```

Выделим в сумме t_2 пятое слагаемое.

```
In[9]:= Part[t2, 5]
Out[9]= 2 x^2 y^2
```

Число членов в выражении t_2 найдем с помощью функции **Length**.

```
In[10]:= Length[t2]
Out[10]= 9
```

Действительно, t_2 представляет собой сумму девяти слагаемых. Если же выражение является произведением, то функция **Length** определяет число множителей, например,

```
In[11]:= Length[t3]
Out[11]= 2
```

Если применить функцию **Expand** к выражению, которое представляет собой произведение нескольких множителей, возводимое в степень, то скобки не раскрываются. Действительно,

```
In[12]:= Expand[(x y z^2)^n]

Out[12]= (x y z^2)^n
```

Это связано с тем, что преобразование вида $(xy)^n \to x^n y^n$ справедливо не всегда, а только в случаях, когда n есть целое число или x и y являются положительными действительными числами. Для выполнения такого преобразования используется функция **PowerExpand**. При этом его правомерность определяется самим пользователем.

```
In[13]:= PowerExpand[(x y z^2)^n]

Out[13]= x^n y^n z^(2 n)
```

Легко подобрать такие значения переменных x, y, z и n, для которых преобразованное с помощью функции **PowerExpand** выражение не совпадает с исходным. Пример:

```
In[14]:= (x y z^2)^n /. {x → -4, y → 9, z → -7, n → 1 / 2}

Out[14]= 42 i

In[15]:= PowerExpand[(x y z^2)^n] /. {x → -4, y → 9, z → -7, n → 1 / 2}

Out[15]= -42 i
```

Если переменные в выражении могут принимать только действительные значения, то для раскрытия скобок можно использовать функцию **ComplexExpand**, например,

```
In[16]:= ComplexExpand[E^(I x y)]

Out[16]= Cos[x y] + i Sin[x y]
```

Если одна или несколько переменных в выражении могут быть комплексными числами, то их следует указать в качестве второго аргумента функции **ComplexExpand**. Предполагая, что y в предыдущем выражении является комплексным числом, и применяя к нему функцию **ComplexExpand**, получаем:

```
In[17]:= ComplexExpand[E^(I x y), y]

Out[17]= e^(-x Im[y]) Cos[x Re[y]] + i e^(-x Im[y]) Sin[x Re[y]]
```

Следующий пример показывает, что при преобразовании рациональных выражений функция **Expand** раскрывает скобки только в числителе, оставляя знаменатель выражения неизменным.

```
In[18]:= t4 = (x + 2)^3 (x - 3) / ((x^2 - 4) (x^2 - 6 x + 9));
         Expand[t4]
```

$$Out[19]= -\frac{24}{(-4+x^2)\ (9-6x+x^2)} - \frac{28x}{(-4+x^2)\ (9-6x+x^2)} - \frac{6x^2}{(-4+x^2)\ (9-6x+x^2)} + \frac{3x^3}{(-4+x^2)\ (9-6x+x^2)} + \frac{x^4}{(-4+x^2)\ (9-6x+x^2)}$$

Функция **ExpandDenominator** раскрывает скобки только в знаменателе рационального выражения.

```
In[20]:= ExpandDenominator[t4]
```

$$Out[20]= \frac{(-3+x)\ (2+x)^3}{-36+24x+5x^2-6x^3+x^4}$$

Раскрыть скобки и в числителе, и в знаменателе выражения можно с помощью функции **ExpandAll**.

```
In[21]:= ExpandAll[t4]
```

$$Out[21]= -\frac{24}{-36+24x+5x^2-6x^3+x^4} - \frac{28x}{-36+24x+5x^2-6x^3+x^4} - \frac{6x^2}{-36+24x+5x^2-6x^3+x^4} + \frac{3x^3}{-36+24x+5x^2-6x^3+x^4} + \frac{x^4}{-36+24x+5x^2-6x^3+x^4}$$

Для приведения суммы дробей к общему знаменателю используется функция **Together**.

```
In[22]:= Together[%]
```

$$Out[22]= \frac{(2+x)^2}{(-3+x)\ (-2+x)}$$

Как видим, сумма дробей не только приведена к общему знаменателю, но и упрощена, так как в полученном выражении сокращены одинаковые множители $(x-3)$ и $(x+2)$ в числителе и знаменателе. Выполнить такое сокращение в исходном выражении t_4 можно и с помощью функции **Cancel**. Действительно, применяя к нему эту функцию, получаем:

```
In[23]:= Cancel[t4]
```

$$Out[23]= \frac{(2+x)^2}{(-3+x)\ (-2+x)}$$

Чтобы разложить выражение на сумму дробей с наиболее простыми знаменателями, достаточно применить к нему функцию **Apart**. Пример:

```
In[24]:= Apart[t4]
```

$$Out[24]= 1 + \frac{25}{-3+x} - \frac{16}{-2+x}$$

Кроме функций, выполняющих конкретные операции, в системе *Mathematica* имеется функция **Simplify**, которая стремится представить выражение

в наиболее простом виде, применяя к нему стандартный набор правил преобразования до тех пор, пока оно не перестанет изменяться. Поскольку понятие "простейшая форма выражения" является довольно условным, получаемый при этом результат не всегда совпадает с тем, который мы ожидаем. Рассмотрим, например, выражение t_5 и применим к нему функцию **Simplify**.

```
In[25]:= t5 = (x - 1)^2 (x + 1)^2 (x^3 - 1);  Simplify[t5]

Out[25]= (-1 + x^2)^2 (-1 + x^3)
```

Теперь раскроем в выражении t_5 скобки и применим функцию **Simplify** к результату.

```
In[26]:= Simplify[Expand[t5]]

Out[26]= (-1 + x)^3 (1 + x)^2 (1 + x + x^2)
```

Как видим, полученное выражение не совпадает с предыдущим, и оба отличаются от t_5. Таким образом, результат работы функции **Simplify** зависит от формы исходного выражения. Вообще говоря, задача упростить выражение, является довольно сложной для компьютера, поскольку трудно определить, какая из форм выражения является простейшей. Кроме того, результат преобразований может зависеть от некоторых дополнительных условий. Попробуем, например, упростить выражение $\sqrt{(x-1)^2}$.

```
In[27]:= Simplify[Sqrt[(x - 1)^2]]

Out[27]= Sqrt[(-1 + x)^2]
```

Поскольку результат зависит от того, какие значения принимает *x*, *Mathematica* только переписывает выражение, размещая числа и переменные в стандартном порядке. Чтобы помочь ей сделать правильный выбор при упрощении выражения, у функции **Simplify** можно добавить второй аргумент, который содержит дополнительные условия, например,

```
In[28]:= Simplify[Sqrt[(x - 1)^2], x < 1]

Out[28]= 1 - x
```

В этом случае получается правильный результат. Аналогично работает функция **FullSimplify**, которая также используется для упрощения выражений, но применяет для этого более широкий набор правил преобразования.

Для преобразования выражений, содержащих тригонометрические функции, в системе *Mathematica* имеются дополнительные команды, приведенные ниже в таблице. Действие этих функций можно проследить на следующих примерах.

```
In[29]:= TrigExpand[Cos[5 x] Cos[2 x]]

Out[29]= Cos[x]^3/2 + Cos[x]^7/2 -
         3/2 Cos[x] Sin[x]^2 - 21/2 Cos[x]^5 Sin[x]^2 +
         35/2 Cos[x]^3 Sin[x]^4 - 7/2 Cos[x] Sin[x]^6
```

```
In[30]:= TrigReduce[Cos[5 x] Cos[2 x]^2]
```

$$Out[30]= \frac{1}{4}\,(\mathrm{Cos}[x] + 2\,\mathrm{Cos}[5x] + \mathrm{Cos}[9x])$$

```
In[31]:= TrigFactor[Cos[3 x] + Sin[2 x]]
```

$$Out[31]= \left(\mathrm{Cos}\left[\frac{x}{2}\right] - \mathrm{Sin}\left[\frac{x}{2}\right]\right)\left(\mathrm{Cos}\left[\frac{x}{2}\right] + \mathrm{Sin}\left[\frac{x}{2}\right]\right)\,(-1 + 2\,\mathrm{Cos}[2x] + 2\,\mathrm{Sin}[x])$$

Обозначение	Назначение
TrigExpand[expr]	Заменить тригонометрические функции от кратных углов в *expr* на степени и произведения тригонометрических функций и раскрыть скобки
TrigFactor[expr]	Разложить *expr* на множители, используя соотношения между тригонометрическими функциями
TrigReduce[expr]	Заменить произведения и степени тригонометрических функций в *expr* через тригонометрические функции от кратных углов
TrigToExp[expr]	Записать тригонометрические функции в *expr* через экспоненту
ExpToTrig[expr]	Записать экспоненту от комплексного агрумента в *expr* через тригонометрические функции

Функции **TrigToExp** и **ExpToTrig** выполняют преобразование тригонометрических функций в экспоненциальные и наоборот, используя формулу Эйлера: $e^{ix} = \cos x + i \sin x$.

```
In[32]:= TrigToExp[Cos[3 x] Cot[x]]
```

$$Out[32]= -\frac{i\,(e^{-ix} + e^{ix})\,(e^{-3ix} + e^{3ix})}{2\,(e^{-ix} - e^{ix})}$$

```
In[33]:= ExpToTrig[E^(I x) (Cos[x] + E^(-2 I x))]
```

$$Out[33]= (\mathrm{Cos}[x] + i\,\mathrm{Sin}[x])\,(\mathrm{Cos}[x] + \mathrm{Cos}[2x] - i\,\mathrm{Sin}[2x])$$

5.2. Преобразование выражений с помощью правил замены

Кроме встроенных функций, в системе *Mathematica* имеется такое мощное средство преобразования выражений как *правила замены* или *подстановки*, которые определяются самим пользователем и представляет собой конструкции вида:

старое выражение → новое выражение.

Стрелка получается при наборе двух символов "–" и ">" без пробела. Для выполнения подстановки *rule* в выражении *expr* используется следующая команда:

expr /. rule	или	**ReplaceAll[expr, rule]**

Между преобразуемым выражением *expr* и правилом подстановки *rule* стоит оператор подстановки "/.", который состоит из двух символов "/" и ".", набираемых без пробела. Сделаем, например, в выражении $\frac{x-y}{x+y}$, которое обозначим через t_6, замену $x \to (z^2+1)$.

```
In[34]:= t6 = (x - y)/(x + y) ; t6 /. x → (z^2 + 1)
```

$$Out[34]= \frac{1 - y + z^2}{1 + y + z^2}$$

Легко убедиться в том, что само выражение t_6 при этом не изменилось. Действительно,

```
In[35]:= t6
```

$$Out[35]= \frac{x - y}{x + y}$$

В выражении можно одновременно выполнить несколько подстановок. При этом соответствующие правила замены должны быть представлены в виде списка, т.е. заключены в фигурные скобки и разделены запятыми. Порядок правил замены в списке значения не имеет.

expr /. {$x \to a$, $y \to b$}	заменить x на a и y на b в выражении *expr*

Легко убедиться в том, что выражение, в котором две замены производятся одновременно, может принимать другой вид по сравнению со случаем, когда замены выполняются последовательно одна за другой.

```
In[36]:= t6 /. {x → y, y → x}
```

$$Out[36]= \frac{-x + y}{x + y}$$

```
In[37]:= t6 /. x → y /. y → x
Out[37]= 0
```

Очевидно, во втором случае после первой замены числитель выражения t_6 обращается в ноль. Поэтому все выражение становится равным нулю, и вторая замена уже не имеет смысла.

В выражении можно заменять не только отдельные переменые, но и целые части. Выполним, например, в выражении t_6 замены $x - y \to z$, $x + y \to 3x^2$. Поскольку порядок замен в данном случае не существенен, можно представить их в виде списка.

```
In[38]:= t6 /. {x - y → z , x + y → 3 x^2}
```

$$Out[38]= \frac{z}{3x^2}$$

Следует отметить, что правила замены могут быть любыми, даже не имеющими смысла. Выполним, например, следующую подстановку:

In[39]:= $\frac{\text{Cos[x]}}{\text{Sin[x]} + 1}$ `/. Cos[x] → 2`

Out[39]= $\frac{2}{1 + \text{Sin[x]}}$

Хотя значения косинуса не могут превышать единицу, при выполнении соответствующей подстановки *Mathematica* не отвечает на вопрос, имеет ли смысл данная операция.

5.3. Дифференцирование и интегрирование функций

Для вычисления частной производной выражения *expr* по переменной *var* используется функция **D[expr, var]**, например:

In[40]:= `D[Sin[k x], x]`

Out[40]= `k Cos[k x]`

Для этой функции можно использовать также следующее обозначение: ∂_x *expr*, причем переменная *x*, по которой производится дифференцирование выражения *expr*, является индексом у буквы ∂.

In[41]:= ∂_{x} `Sin[k x]`

Out[41]= `k Cos[k x]`

Для вычисления производной *k*-ого порядка в качестве второго аргумента функции **D** необходимо указать список вида **{var, k}**. Например, третья производная от x^n равна:

In[42]:= `D[x^n, {x, 3}]`

Out[42]= $(-2 + n)\ (-1 + n)\ n\ x^{-3+n}$

При вычислении смешанных производных в качестве второго, третьего и т.д. аргументов функции **D** в произвольном порядке должны стоять списки, каждый из которых содержит два элемента: первый элемент определяет имя переменной, по которой производится дифференцирование, а второй – порядок соответствующей производной. В качестве примера вычислим производную шестого порядка вида $\frac{\partial^6}{\partial x^3 \partial y\, \partial z^2}$ выражения $(x+y)^4 y^2 \ln^2 z$.

In[43]:= `D[(x + y)^4 y^2 Log[z]^2, {x, 3}, y, {z, 2}]`

Out[43]= $24\ y^2 \left(\frac{2}{z^2} - \frac{2\ \text{Log[z]}}{z^2}\right) + 48\ y\ (x + y) \left(\frac{2}{z^2} - \frac{2\ \text{Log[z]}}{z^2}\right)$

Если выражение *expr* содержит функции от переменной *var*, то при его дифференцировании необходимо указать у этих функций соответствующий аргумент. В противном случае они будут считаться не зависящими от *var*, и их производные будут равны нулю. Примеры:

In[44]:= `D[Cos[x] + f, x]`

Out[44]= `-Sin[x]`

In[45]:= `D[Cos[x] + f[x], x]`

Out[45]= `-Sin[x] + f'[x]`

Mathematica легко вычисляет производные специальных функций и пытается выразить их через другие функции, используя известные соотношения между ними. В качестве примера вычислим вторую производную функции Бесселя $J_n(x)$.

In[46]:=
```
D[BesselJ[n, x], {x, 2}]
```

Out[46]=
$$\frac{1}{2}\left(\frac{1}{2}\,(\texttt{BesselJ[-2 + n, x] - BesselJ[n, x]}) + \frac{1}{2}\,(\texttt{-BesselJ[n, x] + BesselJ[2 + n, x]})\right)$$

Вычисление полного дифференциала некоторого выражения *expr* производится с помощью функции **Dt**, которая в этом случае вызывается с одним аргументом, например,

In[47]:=
```
Dt[x^3 + Cos[x y]]
```

Out[47]=
```
3 x^2 Dt[x] - (y Dt[x] + x Dt[y]) Sin[x y]
```

При этом переменные *x* и *y*, которые есть в выражении *expr*, предполагаются независимыми, и результат вычисления содержит их дифференциалы **Dt[x]** и **Dt[y]**. Если у переменной *y* в предыдущем примере добавить в квадратных скобках аргумент *x*, то останется только одна независимая переменная и результат вычисления будет содержать только один дифференциал.

In[48]:=
```
Dt[x^3 + Cos[x y[x]]]
```

Out[48]=
```
3 x^2 Dt[x] - Sin[x y[x]] (Dt[x] y[x] + x Dt[x] y'[x])
```

Если у функции **Dt** добавить одну из переменных в качестве второго аргумента, то она будет вычислять полную производную выражения *expr* по этой переменной: **Dt[expr, var]**. При этом предполагается, что все остальные переменные в *expr* являются функциями от *var*, даже если соответствующий аргумент отсутствует. Пример:

In[49]:=
```
Dt[x^3 z + Cos[x y], x]
```

Out[49]=
```
3 x^2 z + x^3 Dt[z, x] - (y + x Dt[y, x]) Sin[x y]
```

Аналогично вычисляются кратные полные производные, например,

In[50]:=
```
Dt[x^3 y^2, {x, 2}, y]
```

Out[50]=
```
12 x y + 6 y^2 Dt[x, y] + 12 x^2 Dt[y, x] +
  24 x y Dt[x, y] Dt[y, x] + 2 x^3 Dt[y, {x, 2}] +
  3 x^2 Dt[x, y] (2 Dt[y, x]^2 + 2 y Dt[y, {x, 2}])
```

Если какая-либо из переменных в *expr* не зависит от *var*, т.е. ее производная по *var* равна нулю, то такую переменную необходимо определить как постоянную, добавив у функции **Dt** соответствующий третий аргумент. Так, в следующем примере предполагается, что *a* и *k* не зависят от *x*.

In[51]:=
```
Dt[a (x + y) + Sin[k x], x, Constants -> {a, k}]
```

Out[51]=
```
k Cos[k x] + a (1 + Dt[y, x, Constants → {a, k}])
```

Интегрирование выражения *expr* выполняется функцией **Integrate**. При обращении к ней необходимо указать два аргумента, первый из которых есть само выражение, которое нужно проинтегрировать, а второй – переменная, по которой производится интегрирование. Пример:

In[52]:= `Integrate[Log[k x]/x^3, x]`

Out[52]= $-\frac{1}{4x^2}-\frac{\text{Log}[k\,x]}{2x^2}$

Заметим, что результат, выдаваемый функцией **Integrate**, не содержит произвольной постоянной, возникающей при интегрировании. При записи соответствующей команды вместо имени функции **Integrate** можно использовать стандартное обозначение $\int f\,dx$ (обратите внимание на букву d с двойным начертанием при записи знака дифференциала *dx*). Пример:

In[53]:= `∫ Sin[x]^3 dx`

Out[53]= $-\frac{3\,\text{Cos}[x]}{4}+\frac{1}{12}\text{Cos}[3x]$

Для вычисления кратных интегралов у функции **Integrate** следует добавить соответствующие переменные в качестве третьего и последующих аргументов. Вычислим, например, двойной интеграл от выражения $x^2\cos(x\,y)$.

In[54]:= `Integrate[x^2 Cos[x y], x, y]`

Out[54]= $\frac{-x\,y\,\text{Cos}[x\,y]+\text{Sin}[x\,y]}{y^2}$

Следует отметить, что *Mathematica* знает все интегралы, которые можно найти в справочной литературе. Кроме того, при вычислении интегралов она использует все известные соотношения между элементарными и специальными функциями. Поэтому результат интегрирования выражения, содержащего только элементарные функции, может выражаться через специальные функции. Пример:

In[55]:= `∫ Sin[x^3] dx`

Out[55]= $-\frac{1}{2}\,i\left(-\frac{x\,\text{Gamma}[\frac{1}{3},-i\,x^3]}{3\,(-i\,x^3)^{1/3}}+\frac{x\,\text{Gamma}[\frac{1}{3},i\,x^3]}{3\,(i\,x^3)^{1/3}}\right)$

Если же *Mathematica* не может вычислить интеграл, то она просто переписывает исходное выражение, размещая в нем числа и символы в стандартном порядке. Пример:

In[56]:= `Integrate[1/(Tan[x]+x), x]`

Out[56]= $\int\frac{1}{x+\text{Tan}[x]}\,dx$

Если второй аргумент функции **Integrate** является списком вида **{x, xmin, xmax}**, то будет вычисляться определенный интеграл по *x* на отрезке [*xmin*, *xmax*]. Следующий пример показывает, что пределы интегрирования могут быть любыми.

In[57]:= `Integrate[Sin[x]/x, {x, -∞, 0}]`

Out[57]= $\frac{\pi}{2}$

В некоторых случаях, когда интегрируемое выражение содержит произвольные параметры, вид получаемого результата может зависеть от значений этих параметров. Поэтому получаемое выражение будет очень громоздким, включающим в себя много условий на параметры. Если же условия на параметры заранее известны, то список этих условий можно добавить в качестве значения опции **Assumptions**, помещаемой в качестве третьего аргумента функции **Integrate**. В этом случае получаемый результат будет существенно упрощен. В следующем примере использована такая возможность.

```
In[58]:= Integrate[x/Sqrt[a x^2 + x], {x, 0, c}, Assumptions -> {a > 0, c > 0}]
```

$$Out[58]= \frac{\sqrt{a\,c\,(1+a\,c)} - \text{ArcSinh}\left[\sqrt{a\,c}\right]}{a^{3/2}}$$

Легко убедиться в том, что без условий на параметры *a* и *c* получаемый результат будет очень громоздким.

При вычислении определенных интегралов функция **Integrate** сначала находит неопределенный интеграл, а затем вычисляет разность полученных значений на верхнем и нижнем пределах. И если подынтегральное выражение не содержит параметров, а все числа в нем и пределы интегрирования заданы точно, то будет получен точный результат. Пример:

```
In[59]:= Integrate[1/(1 + x^4)^(1/3), {x, 0, 1}]
```

$$Out[59]= \text{Hypergeometric2F1}\left[\frac{1}{4}, \frac{1}{3}, \frac{5}{4}, -1\right]$$

Если же добавить у любого числа десятичную точку, указав тем самым, что оно задано приближенно, то и результат вычислений будет получен с машинной точностью.

```
In[60]:= Integrate[1./(1 + x^4)^(1/3), {x, 0, 1}]

Out[60]= 0.949768
```

Такая замена эквивалентна применению к точному результату функции **N**. Кроме того, в системе *Mathematica* есть функция **NIntegrate**, которая *не эквивалентна* последовательному применению функций **Integrate** и **N**, как это было, например, в случае функции **NSolve**. Функция **NIntegrate** не вычисляет неопределенный интеграл, а сразу анализирует подынтегральное выражение и пределы интегрирования и на основе этого анализа применяет для вычисления заданного интеграла некоторый численный метод. Следует отметить, что функция **NIntegrate** позволяет вычислить определенный интеграл даже в том случае, когда неопределенный интеграл не вычисляется. Примеры:

```
In[61]:= Integrate[1/(Cos[x] + x), {x, 0, 1.}]
```

```
In[61]:= Integrate[1/(x + Cos[x]), {x, 0, 1}]
```

$$\text{Out[61]}= \int_0^1 \frac{1}{x + \text{Cos}[x]}\, dx$$

```
In[62]:= NIntegrate[1/(Cos[x] + x), {x, 0, 1}]
Out[62]= 0.756656
```

При вычислении кратных определенных интегралов с помощью функции **Integrate** существенным оказывается порядок, в котором указаны переменные и пределы интегрирования. Следует помнить, что порядок вычисления интегралов является обратным к порядку следования аргументов у функции **Integrate**. Так, в следующем примере сначала вычисляется интеграл по *y*, а затем выполняется интегрирование по *x*.

```
In[63]:= Integrate[Sin[x y], {x, 0, π}, {y, 0, x}]
```

$$\text{Out[63]}= \frac{1}{2}\left(\text{EulerGamma} - \text{CosIntegral}[\pi^2]\right) + \text{Log}[\pi]$$

5.4. Решение уравнений и систем уравнений

Mathematica располагает целым набором функций для решения уравнений в символьной форме. Полную информацию о них можно получить, запустив программу помощи **Help Browser** и выбрав последовательно разделы **Built-in Functions** ↦ **Algebraic Computation** ↦ **Equation Solving**. Мы рассмотрим только некоторые из этих функций.

Основной функцией для решения уравнений и их систем является **Solve**, при вызове которой нужно указать по крайней мере два аргумента. Первым аргументом является уравнение или список уравнений, а вторым – соответственно переменная или список переменных, относительно которых нужно разрешить уравнение или систему. В качестве первого примера найдем решение квадратного уравнения:

```
In[64]:= Solve[a x^2 - b x + c == 0, x]
```

$$\text{Out[64]}= \left\{\left\{x \to \frac{b - \sqrt{b^2 - 4ac}}{2a}\right\}, \left\{x \to \frac{b + \sqrt{b^2 - 4ac}}{2a}\right\}\right\}$$

Как обычно, в любом уравнении обязательным является двойной знак равенства. Решения выдаются в виде списка, каждый элемент которого в свою очередь является списком правил замены и заключается в фигурные скобки. В случае системы двух уравнений, например, каждое решение является списком из двух правил замены для обеих переменных.

```
In[65]:= Solve[{3 x + b y == 1, x y == a}, {x, y}]
```

$$\text{Out[65]}= \left\{\left\{x \to \frac{1}{6}\left(1 - \sqrt{1 - 12ab}\right), y \to \frac{1 + \sqrt{1 - 12ab}}{2b}\right\},\right.$$
$$\left.\left\{x \to \frac{1}{3}\left(\frac{1}{2} + \frac{1}{2}\sqrt{1 - 12ab}\right), y \to \frac{1 - \sqrt{1 - 12ab}}{2b}\right\}\right\}$$

При наличии кратных корней каждый корень выводится столько раз, какова его кратность. Пример:

```
In[66]:= Solve[
          x^4 + 3 (1 - a) x^3 + 3 a (a - 3) x^2 - a^2 x (a - 9) - 3 a^3 == 0, x]
Out[66]= {{x → -3}, {x → a}, {x → a}, {x → a}}
```

Если уравнение является достаточно сложным, то его решение не может быть записано в виде формулы. Однако уравнение может иметь численные решения. В этом случае они представляются в виде **Root**-объектов, имеющих два аргумента. Первым аргументом является функция, нули которой есть корни рассматриваемого уравнения, записанная в виде неименованной или "чистой" функции, а второй аргумент соответствует номеру корня. Пример:

```
In[67]:= Solve[x^5 + 5 x - 3 == 0, x]
Out[67]= {{x → Root[-3 + 5 #1 + #1^5 &, 1]},
          {x → Root[-3 + 5 #1 + #1^5 &, 2]},
          {x → Root[-3 + 5 #1 + #1^5 &, 3]},
          {x → Root[-3 + 5 #1 + #1^5 &, 4]},
          {x → Root[-3 + 5 #1 + #1^5 &, 5]}}
```

Численные значения корней всегда могут быть найдены путем применения функции **N** к полученному результату.

```
In[68]:= N[%, 10]
Out[68]= {{x → 0.5861607183}, {x → -1.183348026 - 1.073307836 i},
          {x → -1.183348026 + 1.073307836 i},
          {x → 0.890267667 - 1.101222754 i},
          {x → 0.890267667 + 1.101222754 i}}
```

Если уравнение или система уравнений не имеет решений, то функция **Solve** выдает одни фигурные скобки, например,

```
In[69]:= Solve[{x^2 + 2 x y + y^2 == 1,  x + y == 2}, {x, y}]
Out[69]= {}
```

Если же уравнение удовлетворяется при любом значении переменной, то выдаются двойные фигурные скобки.

```
In[70]:= Solve[x 0 == 0, x]
Out[70]= {{}}
```

Часто при решении системы уравнений нас интересуют значения не всех переменных, а только некоторых из них. В таком случае в качестве второго аргумента функции **Solve** необходимо указать список только тех переменных, которые следует найти. Остальные переменные должны быть указаны в качестве третьего аргумента, и они не будут выдаваться. Тем не менее, общее число переменных должно равняться числу уравнений системы. Примеры:

```
In[71]:= Solve[{x + 3 y - z == 1, x^2 - y^2 == 2, x + z == 7}, {y, z}, x]
```

Out[71]= $\left\{\left\{z \to \frac{3}{5}\left(17 - \sqrt{74}\right),\ y \to \frac{2}{5}\left(12 - \sqrt{74}\right)\right\},\right.$

$\left.\left\{z \to \frac{1}{2}\left(\frac{102}{5} + \frac{6\sqrt{74}}{5}\right),\ y \to \frac{2}{5}\left(12 + \sqrt{74}\right)\right\}\right\}$

In[72]:=
```
Solve[{x + 3 y - z == 1, x^2 - y^2 == 2, x + z == 7}, z, {x, y}]
```

Out[72]= $\left\{\left\{z \to \frac{3}{5}\left(17 - \sqrt{74}\right)\right\},\ \left\{z \to \frac{3}{5}\left(17 + \sqrt{74}\right)\right\}\right\}$

Хотя функция **Solve** является наиболее эффективной при решении полиномиальных уравнений, она может находить решения и других видов уравнений. В качестве примера рассмотрим уравнение, содержащее тригонометрические функции.

In[73]:=
```
Solve[Cos[2 x] - a Sin[x] == 1, x]
```

```
Solve::ifun :
  Inverse functions are being used by Solve, so some
    solutions may not be found; use Reduce
    for complete solution information. More
```

Out[73]= $\left\{\{x \to 0\},\ \{x \to -\pi\},\ \{x \to \pi\},\right.$

$\left\{x \to -\text{ArcCos}\left[-\frac{1}{2}\sqrt{4 - a^2}\right]\right\},\ \left\{x \to \text{ArcCos}\left[-\frac{1}{2}\sqrt{4 - a^2}\right]\right\},$

$\left.\left\{x \to -\text{ArcCos}\left[\frac{\sqrt{4 - a^2}}{2}\right]\right\},\ \left\{x \to \text{ArcCos}\left[\frac{\sqrt{4 - a^2}}{2}\right]\right\}\right\}$

Получили семь корней и сообщение о том, что при их вычислении использовались обратные функции. Поэтому некоторые корни могли быть потеряны. И действительно, потерянными оказались, например, корни $x = \pi k$ ($k = \pm 2, \pm 3, \ldots$). В этой связи следует заметить, что функция **Solve** имеет ряд опций, которые управляют ее работой. Одна из них – **InverseFunctions** – по умолчанию имеет значение **Automatic**. Это означает, что при решении уравнения обратные функции могут использоваться, но при этом следует выдавать сообщение о возможной потере корней. Если добавить в качестве третьего аргумента функции **Solve** правило **InverseFunctions→True**, то обратные функции будут всегда использоваться, но сообщений об этом не будет. В случае правила **InverseFunctions→False** могут быть найдены только такие решения, для поиска которых не нужно использовать обратные функции.

In[74]:=
```
Solve[Cos[2 x] - a Sin[x] == 1,
    x, InverseFunctions → False]
```

```
Solve::ifun2 : Cannot obtain solution with
    InverseFunctions->False option setting. More
```

Out[74]=
```
Solve[Cos[2 x] - a Sin[x] == 1,
    x, InverseFunctions → False]
```

Если уравнение не может быть решено, *Mathematica* перепечатывает введенные данные и выдает соответствующее сообщение.

Решения, которые существуют только при выполнении некоторых условий, налагаемых на параметры уравнения, функция **Solve** отбрасывает. Этого недостатка лишена функция **Reduce**, заменяющая исходное уравнение системой более простых уравнений, которая содержит все возможные решения, включая условия на параметры, обеспечивающие существование этих решений. Применим эту функцию к уравнению, рассмотренному в предыдущем примере.

In[75]:=
```
Reduce[Cos[2 x] - a Sin[x] == 1]
```

Out[75]=
```
(Sin[x] == 0 && Cos[2 x] == 1) ||
    (Sin[x] ≠ 0 && a == (-1 + Cos[2 x]) Csc[x])
```

Полученный результат состоит из ряда уравнений и неравенств, соединенных символами || и &&, которые обозначают логические операции "или" и "и" соответственно. Как видим, возможны два варианта: в первом $\sin x = 0$, и тогда $\cos(2x) = 1$, а во втором – $\sin x \neq 0$ и уравнение сохраняет свою форму. Поскольку мы не указали, относительно какой переменной следует разрешить уравнение, функция **Reduce** просто выделила два возможных варианта. Очевидно, они соответствуют тому случаю, когда уравнение решается относительно *a*. При попытке разрешить это уравнение относительно *x* количество и возможные значения корней будут определяться параметром *a*, и ответ, выдаваемый функцией **Reduce**, будет очень громоздким. Чтобы уменьшить его, дополним уравнение условием: $a > 2$.

In[76]:=
```
Reduce[{Cos[2 x] - a Sin[x] == 1, a > 2}, x]
```

Out[76]=
$$\left(a > 2 \;\&\&\; C[1] \in \text{Integers} \;\&\&\; \left(x == 2\pi C[1] \;||\; x == -2\,\text{ArcTan}\left[\frac{2}{a} + i\sqrt{\frac{-4+a^2}{a^2}}\right] + 2\pi C[1] \;||\; x == -2\,\text{ArcTan}\left[\frac{2}{a} - i\sqrt{\frac{-4+a^2}{a^2}}\right] + 2\pi C[1]\right)\right) \;||$$
$$(C[1] \in \text{Integers} \;\&\&\; a > 2 \;\&\&\; x == \pi + 2\pi C[1])$$

Чтобы выделить в полученном результате возможные варианты, применим к нему функцию **LogicalExpand**, которая представляет его в виде отдельных решений, соединенных символами || .

In[77]:=
```
LogicalExpand[%]
```

Out[77]= (C[1] ∈ Integers && 2 π C[1] == x && a > 2) ||
(C[1] ∈ Integers && π + 2 π C[1] == x && a > 2) ||
$$\left(C[1] \in \text{Integers} \;\&\&\; -2\,\text{ArcTan}\left[\frac{2}{a} - i\sqrt{\frac{-4+a^2}{a^2}}\right] + 2\pi C[1] == x \;\&\&\; a > 2\right) \;||\; \left(C[1] \in \text{Integers} \;\&\&\; -2\,\text{ArcTan}\left[\frac{2}{a} + i\sqrt{\frac{-4+a^2}{a^2}}\right] + 2\pi C[1] == x \;\&\&\; a > 2\right)$$

Теперь легко видеть, что при $a > 2$ существуют четыре бесконечные последовательности решений (C[1] – произвольное целое число), две из которых определяют комплексные корни. В каждом решении уравнения и неравенства соединены символами &&, т.е. они должны удовлетворяться одновременно. Если мы хотим отобрать только такие корни, при подстановке которых в уравнение функции *cos(2x)* и *sin(x)* принимают действительные значения, необходимо добавить слово **Reals** в качестве третьего аргумента функции **Reduce**. В результате выделяются только две последовательности действительных корней.

In[78]:= Reduce[{Cos[2 x] - a Sin[x] == 1, a > 2}, x, Reals]

Out[78]= (C[1] ∈ Integers && a > 2 && x == 2 π C[1]) ||
(C[1] ∈ Integers && a > 2 && x == π + 2 π C[1])

При анализе системы уравнений часто возникает необходимость упростить ее, выражая одну или несколько переменных из некоторых уравнений и подставляя в остальные. Естественно, количество уравнений и переменных в системе при этом уменьшается. Для реализации этой процедуры служит функция **Eliminate**, которая вызывается с двумя аргументами. С помощью этой функции из системы трех уравнений, например, можно получить систему двух уравнений относительно любых двух переменных или одно уравнение относительно одной из переменных. Так, удаляя переменную *z* в следующем примере, получаем:

In[79]:= Eliminate[{x + y^2 + 3 z == 1, x^2 - y + z^2 == 0,
2 x + y^2 - z == 5}, z]

Out[79]= $7x == 16 - 4y^2 \;\&\&\; -49y - 122y^2 + 17y^4 == -265$

Удаляя переменные *y* и *z*, получаем одно уравнение относительно *x*.

In[80]:= Eliminate[{x + y^2 + 3 z == 1, x^2 - y + z^2 == 0,
2 x + y^2 - z == 5}, {y, z}]

Out[80]= $192x + 608x^2 - 272x^3 + 289x^4 == 768$

Функция **Eliminate** хорошо работает с линейными и полиномиальными уравнениями. Однако уже в случае уравнений с простейшими трансцендентными функциями, например, возникают проблемы, связанные с

использованием обратных функций. При этом появляются сообщения о возможной потере корней. Пример:

```
In[81]:= Eliminate[{x^2 - y^2 == 0, Cos[x + y] - Sin[x] == 1}, y]

    Eliminate::ifun :
     Inverse functions are being used by Eliminate,
       so some solutions may not be found; use
       Reduce for complete solution information. More...
```

$$\text{Out[81]}= -2\,x\,\text{ArcCos}[1+\text{Sin}[x]] + \text{ArcCos}[1+\text{Sin}[x]]^2 == 0 \;||\; 2\,x\,\text{ArcCos}[1+\text{Sin}[x]] + \text{ArcCos}[1+\text{Sin}[x]]^2 == 0$$

Для решения таких уравнений наиболее эффективной оказывается функция **Reduce**. Так, для рассмотренной выше системы она выдает полный набор решений:

```
In[82]:= LogicalExpand[
           Reduce[{x^2 - y^2 == 0, Cos[x + y] - Sin[x] == 1}, {x, y}] ]
```

$$\begin{aligned}\text{Out[82]}= {} & (C[1] \in \text{Integers} \;\&\&\; y == -x \;\&\&\; 2\pi C[1] == x) \;||\\ & (C[1] \in \text{Integers} \;\&\&\; y == -x \;\&\&\; \pi + 2\pi C[1] == x) \;||\\ & (C[1] \in \text{Integers} \;\&\&\; y == x \;\&\&\; 2\pi C[1] == x) \;||\\ & \left(C[1] \in \text{Integers} \;\&\&\; y == x \;\&\&\; -\frac{5\pi}{6} + 2\pi C[1] == x\right) \;||\\ & \left(C[1] \in \text{Integers} \;\&\&\; y == x \;\&\&\; -\frac{\pi}{6} + 2\pi C[1] == x\right) \;||\\ & (C[1] \in \text{Integers} \;\&\&\; y == x \;\&\&\; \pi + 2\pi C[1] == x)\end{aligned}$$

Чтобы разделить решения, мы применили к результату работы функции **Reduce** команду **LogicalExpand**.

Для решения систем линейных уравнений вместо **Solve** часто оказывается более удобным использовать функцию **LinearSolve**. Эта функция предполагает, что система задана в матричном виде: $m.x = b$, где $x = \{x_1, x_2, \ldots, x_n\}$ – список переменных, m – матрица коэффициентов при x_j ($j = 1, 2, \ldots, n$), а b – список правых частей системы. Пусть, например, матрица m и вектор b имеют вид:

```
In[83]:= m = {{10, 2, 13, 4}, {0, -2, 7, 1},
            {-3, 1, -6, 2}, {0, 8, 1, 0}}; b = {11, 3, 9, 6};
```

Соответствующую систему можно записать в стандартном виде:

```
In[84]:= m1 = Thread[ m.{x1, x2, x3, x4} == b ]
```

$$\text{Out[84]}= \{10\,x_1 + 2\,x_2 + 13\,x_3 + 4\,x_4 == 11,\; -2\,x_2 + 7\,x_3 + x_4 == 3,\; -3\,x_1 + x_2 - 6\,x_3 + 2\,x_4 == 9,\; 8\,x_2 + x_3 == 6\}$$

Решение, находимое функцией **LinearSolve**, представляет собой список из четырех элементов.

```
In[85]:= LinearSolve[m, b]
```

$$\text{Out[85]}= \left\{-\frac{87}{136}, \frac{499}{680}, \frac{11}{85}, \frac{1211}{340}\right\}$$

Удаляя у матрицы *m* четвертую строку, а у вектора *b* – четвертый элемент, мы получим систему из трех уравнений относительно четырех переменных x_1, x_2, x_3, x_4.

```
In[86]:= Delete[m1, 4]
```

$$Out[86]= \{10\,x_1 + 2\,x_2 + 13\,x_3 + 4\,x_4 == 11,\ -2\,x_2 + 7\,x_3 + x_4 == 3,\ -3\,x_1 + x_2 - 6\,x_3 + 2\,x_4 == 9\}$$

Хотя эта система имеет бесконечное множество решений, функция **LinearSolve** выдает только одно их них.

```
In[87]:= LinearSolve[Delete[m, 4], Delete[b, 4]]
```

$$Out[87]= \left\{-7,\ \frac{66}{5},\ \frac{21}{5},\ 0\right\}$$

В отличие от нее, функция **Solve** выдает общее решение.

```
In[88]:= Solve[Delete[m1, 4]]

Solve::svars : Equations may not
   give solutions for all "solve" variables. More
```

$$Out[88]= \left\{\left\{x_2 \to -\frac{13}{25} - \frac{49\,x_1}{25},\ x_3 \to -\frac{7}{25} - \frac{16\,x_1}{25},\ x_4 \to \frac{98}{25} + \frac{14\,x_1}{25}\right\}\right\}$$

Поскольку мы не указали, относительно каких переменных следует решать систему, выдается сообщение, что уравнения не позволяют найти все переменные. При этом переменная x_1 становится произвольным параметром.

Если при обращении к функции **LinearSolve** вектор *b* не указан, то она создает специальный объект **LinearSolveFunction**, который можно применять к различным векторам *b*, получая соответствующие решения. Так, в случае вектора *b* = {1, 2, 3, 5}, например, получаем:

```
In[89]:= LinearSolve[m][{1, 2, 3, 5}]
```

$$Out[89]= \left\{-\frac{55}{68},\ \frac{199}{340},\ \frac{27}{85},\ \frac{161}{170}\right\}$$

Следует отметить также функцию **RSolve**, которая позволяет находить решения линейных и нелинейных разностных уравнений. К уравнениям такого типа относятся, например, рекуррентные соотношения, которым удовлетворяют некоторые функции. *Mathematica* распознает такие соотношения и выдает соответствующие решения. И хотя трудно ожидать, что решение любого уравнения можно представить в виде формулы, в некоторых случаях это возможно. Примеры:

```
In[90]:= RSolve[
           {y[k + 1] - 2 y[k] + y[k - 1] == k^2, y[0] == 0, y[1] == 1}, y, k]
```

$$Out[90]= \left\{\left\{y \to \text{Function}\left[\{k\},\ \frac{1}{12}\,(12\,k - k^2 + k^4)\right]\right\}\right\}$$

```
In[91]:= RSolve[{y[x] == x^2 y[x - 1], y[0] == 1}, y, x]
```

$$Out[91]= \{\{y \to \text{Function}[\{x\}, \text{Gamma}[1 + x]^2]\}\}$$

5.5. Вычисление сумм и произведений

Для вычисления сумм в системе *Mathematica* используется функция **Sum**, первый аргумент которой является суммируемым выражением, а второй представляет собой *итератор*, определяющий порядок суммирования. Возможные виды итераторов приведены в таблице.

Обозначение	Интервал изменения переменной
{n}	операция выполняется *n* раз
{i, n}	*i* изменяется в пределах отрезка [*1*, *n*] с шагом *1*
{i, imin, imax}	*i* изменяется в пределах отрезка [*imin*, *imax*] с шагом *1*
{i, imin, imax, step}	*i* изменяется в пределах отрезка [*imin*, *imax*] с шагом *step*

Простейшим итератором является число, заключенное в фигурные скобки. Такой итератор используется в том случае, когда суммируемое выражение не содержит параметра. Вычислим, например, сумму пяти случайных чисел, генерируемых функцией **Random**. Напомним, что при отсутствии аргумента эта функция генерирует действительные случайные числа из интервала [0,1].

```
In[92]:= Sum[Random[], {5}]

Out[92]= 2.32904
```

В следующем примере вычисляется сумма кубов первых двенадцати натуральных чисел.

```
In[93]:= Sum[k^3, {k, 12}]

Out[93]= 6084
```

При выполнении суммирования *Mathematica* производит возможные упрощения в каждом из слагаемых и выдает результат в наиболее простом виде. Примеры:

```
In[94]:= Sum[x^k / k!, {k, 0, 5}]
```

$$Out[94]= 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120}$$

```
In[95]:= Sum[k^2, {k, 0, n, 1 / 3}]
```

$$Out[95]= \frac{1}{54}\,\mathtt{Floor[3\,n]\,(1 + Floor[3\,n])\,(1 + 2\,Floor[3\,n])}$$

Хотя в последнем примере значение верхнего предела суммирования *n* не задано, соответствующая сумма допускает представление в виде произведения. Поскольку переменная *k* изменяется с шагом $\frac{1}{3}$, то ее максимальное значение определяется из условия $\frac{k}{3} \le n$. Поэтому полученный результат содержит встроенную функцию **Floor**. Кроме того, результат суммирования может выражаться через элементарные и специальные функции. Примеры:

```
In[96]:= Sum[k / (2 k + 1) ^ 3, {k, -1, Infinity}]
```

Out[96]= $\frac{1}{16}\,(16 + \pi^2 - 7\,\mathrm{Zeta}[3])$

In[97]:=
$$\sum_{k=1}^{\infty} \frac{1 + (1 - x)^k}{k}$$

Out[97]= `EulerGamma - Log[x]`

Как видим, вместо имени **Sum** можно использовать стандартный знак суммирования. Однако в этом случае необходимо указывать оба предела суммирования, а шаг изменения переменной всегда равняется единице.

Если сумма не может быть вычислена или представлена в более простом виде, то *Mathematica* переписывает введенные данные. Пример:

In[98]:= `Sum[n / (n! + 1), {n, Infinity}]`

Out[98]=
$$\sum_{n=1}^{\infty} \frac{n}{1 + n!}$$

В таком случае можно попытаться получить приближенное значение суммы, применяя к результату функцию **N**. Кроме того, можно сразу использовать функцию **NSum**.

In[99]:= `NSum[n / (n! + 1), {n, Infinity}]`

Out[99]= `1.8065`

Если в качестве второго и последующих аргументов функции **Sum** указаны несколько итераторов, то будет выполняться многократное суммирование, причем его порядок будет обратным порядку следования итераторов. Так, в следующем примере сумма по *k* является внешней, а переменная *j* изменяется в пределах $1 \le j \le k$ при каждом значении *k*.

In[100]:= `Sum[x^k y^j, {k, 3}, {j, k}]`

Out[100]= $x\,y + x^2\,y + x^3\,y + x^2\,y^2 + x^3\,y^2 + x^3\,y^3$

Для вычисления произведений используются функции **Product** и **NProduct**, обращение к которым аналогично обращению к функциям **Sum** и **NSum**. Поэтому приведем лишь несколько примеров их использования.

In[101]:= `Product[i, {i, n}]`

Out[101]= `n!`

In[102]:= `Product[k / (k^2 + 1), {k, n}]`

Out[102]=
$$\frac{\pi\,\mathrm{Csch}[\pi]\,\mathrm{Gamma}[1 + n]}{\mathrm{Gamma}[(1 - i) + n]\,\mathrm{Gamma}[(1 + i) + n]}$$

In[103]:= `NProduct[k / (k^2 + 1), {k, 7}]`

Out[103]= `0.0000616363`

5.6. Разложение функций в ряд и вычисление пределов

Встроенная функция **Series** позволяет получить разложение функции или выражения в степенной ряд в окрестности заданной точки. Если вид функции не определен, то это будет обычный ряд Тейлора, например,

In[104]:=
```
Series[f[x], {x, x0, 2}]
```

Out[104]= $f[x0] + f'[x0]\ (x - x0) + \frac{1}{2} f''[x0]\ (x - x0)^2 + O[x - x0]^3$

Первым аргументом функции **Series** является выражение *f(x)*, которое следует разложить в ряд. Второй аргумент этой функции представляет собой список вида {*x*, *x0*, 2}, который указывает, что разложение следует произвести по переменной *x* в окрестности точки *x0* с точностью до второго порядка, т.е. максимальная степень разности $(x - x0)$ в разложении должна быть равна двум. Символ $O[x - x0]^3$ в полученном разложении показывает порядок следующего члена ряда.

Ряд, получаемый с помощью функции **Series**, является более общим по сравнению с рядом Тейлора, так как может содержать отрицательные и дробные степени переменной. Примеры:

In[105]:=
```
Series[Tan[x] / x^3, {x, 0, 5}]
```

Out[105]= $\frac{1}{x^2} + \frac{1}{3} + \frac{2x^2}{15} + \frac{17x^4}{315} + O[x]^6$

In[106]:=
```
Series[Sinh[Sqrt[x]], {x, 0, 4}]
```

Out[106]= $\sqrt{x} + \frac{x^{3/2}}{6} + \frac{x^{5/2}}{120} + \frac{x^{7/2}}{5040} + O[x]^{9/2}$

Если в точке *x0* аргумент функции имеет сингулярность, то она не может быть разложена в ряд в окрестности этой точки. В таком случае выдается соответствующее сообщение, например,

In[107]:=
```
Series[Sin[1 / x], {x, 0, 5}]
```

```
Series::esss :
```

```
Essential singularity encountered in Sin[1/x + O[x]^6]. More
```

Out[107]= $\mathrm{Sin}\left[\frac{1}{x}\right]$

Тем не менее, можно разложить эту функцию по степеням $(\frac{1}{x})$ в окрестности бесконечно удаленной точки.

In[108]:=
```
Series[Sin[1 / x], {x, Infinity, 5}]
```

Out[108]= $\frac{1}{x} - \frac{1}{6}\left(\frac{1}{x}\right)^3 + \frac{1}{120}\left(\frac{1}{x}\right)^5 + O\left[\frac{1}{x}\right]^6$

Mathematica позволяет разложить в ряд выражение, содержащее логарифмическую функцию. Поскольку эта функция имеет в точке $x = 0$ сингулярность, то она не разлагается и входит в окончательное выражение без изменений. Пример:

In[109]:= `Series[(x^2 + 1 + Log[x])^3, {x, 0, 5}]`

Out[109]= $(1 + \text{Log}[x])^3 + 3\,(1 + \text{Log}[x])^2\,x^2 + 3\,(1 + \text{Log}[x])\,x^4 + O[x]^6$

Однако разложение логарифмической функции в ряд в окрестности любой другой точки может быть легко получено, например,

In[110]:= `Series[Log[x], {x, a, 3}]`

Out[110]= $\text{Log}[a] + \frac{x-a}{a} - \frac{(x-a)^2}{2a^2} + \frac{(x-a)^3}{3a^3} + O[x-a]^4$

При подстановке вместо *a* числа log *a* будет заменяться соответствующим численным значением.

С помощью функции **Series** можно разложить выражение в ряд сразу по нескольким переменным. Для этого достаточно добавить соответствующие списки вида {*y*, y0, *k*} в качестве третьего и последующих ее аргументов. В таком случае сначала производится разложение выражения по переменной, определенной во втором аргументе, затем коэффициенты полученного выражения разлагаются по переменной, определенной в третьем аргументе, и т.д. В качестве примера разложим sin(*x y*) по *x* и *y*.

In[111]:= `Series[Sin[x y], {x, 0, 5}, {y, 0, 6}]`

Out[111]= $(y + O[y]^7)\,x + \left(-\frac{y^3}{6} + O[y]^7\right)x^3 + \left(\frac{y^5}{120} + O[y]^7\right)x^5 + O[x]^6$

Следующий пример показывает, что такой же результат получается при последовательном применении функции **Series**, выполняющей каждый раз разложение по одной переменной.

In[112]:= `Series[Series[Sin[x y], {x, 0, 5}], {y, 0, 6}]`

Out[112]= $(y + O[y]^7)\,x + \left(-\frac{y^3}{6} + O[y]^7\right)x^3 + \left(\frac{y^5}{120} + O[y]^7\right)x^5 + O[x]^6$

Следует отметить, что символ $O[x]^n$, который содержится в выражениях, выдаваемых функцией **Series**, не только указывает порядок следующего члена разложения, но и является признаком выражения, заданного приближенно. При работе с разложениями функций этот символ играет ту же роль, что и десятичная точка при работе с числами. Если некоторое выражение содержит символ $O[x]^n$, то при любых операциях с ним получаемый результат будет представлять собой степенной ряд по *x*. Пример:

In[113]:= `ex1 = (x + a)^5 (1 + a x + x^2 + O[x]^4)`

Out[113]= $a^5 + (5a^4 + a^6)\,x + (10a^3 + 6a^5)\,x^2 + (10a^2 + 15a^4)\,x^3 + O[x]^4$

С такими рядами можно поступать как с обычными функциями, т.е. складывать, умножать, возводить в степень, подставлять в качестве аргумента в другие функции, дифференцировать, интегрировать и т.д. При этом все вычисления будут выполняться с максимально возможной точностью, а результат будет представляться в виде степенного ряда, содержащего характерный символ $O[x]^n$. Примеры:

In[114]:= `ex1^2`

Out[114]= $a^{10} + 2\,a^5\,(5\,a^4 + a^6)\,x + \left(2\,a^5\,(10\,a^3 + 6\,a^5) + (5\,a^4 + a^6)^2\right)x^2 +$
$(2\,a^5\,(10\,a^2 + 15\,a^4) + 2\,(10\,a^3 + 6\,a^5)\,(5\,a^4 + a^6))\,x^3 + O[x]^4$

In[115]:= `Cos[ex1]`

Out[115]= $\mathrm{Cos}[a^5] + (-5\,a^4 - a^6)\,\mathrm{Sin}[a^5]\,x +$
$\left(-\frac{1}{2}\,(5\,a^4 + a^6)^2\,\mathrm{Cos}[a^5] + (-10\,a^3 - 6\,a^5)\,\mathrm{Sin}[a^5]\right)x^2 +$
$\left(-(10\,a^3 + 6\,a^5)\,(5\,a^4 + a^6)\,\mathrm{Cos}[a^5] + (-10\,a^2 - 15\,a^4)\right.$
$\left.\mathrm{Sin}[a^5] + \frac{1}{6}\,(5\,a^4 + a^6)^3\,\mathrm{Sin}[a^5]\right)x^3 + O[x]^4$

In[116]:= `D[ex1, x]`

Out[116]= $(5\,a^4 + a^6) + 2\,(10\,a^3 + 6\,a^5)\,x + 3\,(10\,a^2 + 15\,a^4)\,x^2 + O[x]^3$

In[117]:= `Integrate[%, x]`

Out[117]= $a^4\,(5 + a^2)\,x + 2\,a^3\,(5 + 3\,a^2)\,x^2 + 5\,a^2\,(2 + 3\,a^2)\,x^3 + O[x]^4$

Поскольку символ $O[x]^n$ показывает, что выражение, в котором он содержится, задано приближенно, то при сложении или умножении точного и приближенного выражений будет получаться приближенный результат. Пример:

In[118]:= `Sin[x] ex1`

Out[118]= $a^5\,x + (5\,a^4 + a^6)\,x^2 + \left(10\,a^3 + \frac{35\,a^5}{6}\right)x^3 +$
$\left(10\,a^2 + 15\,a^4 + \frac{1}{6}\,(-5\,a^4 - a^6)\right)x^4 + O[x]^5$

Как видим, при умножении выражения *ex1* на функцию sin*x* результат автоматически разлагается в ряд по *x*. Поскольку первый член в разложении синуса равен *x*, а максимальная степень *x* в выражении *ex1* равняется трем, то результат умножения определяется с точностью до x^4, а следующий член разложения имеет порядок x^5.

Чтобы удалить символ $O[x]^n$, не изменяя выражения, достаточно применить к нему функцию **Normal**. Эта функция аналогична функции **Chop**, используемой при работе с числами. Пример:

In[119]:= `Normal[%]`

Out[119]= $a^5\,x + (5\,a^4 + a^6)\,x^2 + \left(10\,a^3 + \frac{35\,a^5}{6}\right)x^3 +$
$\left(10\,a^2 + 15\,a^4 + \frac{1}{6}\,(-5\,a^4 - a^6)\right)x^4$

Получаемый результат является выражением, заданным точно. Умножая его на $(x-a)^2$, например, убеждаемся, что никаких разложений в ряд не происходит.

In[120]:= `% (x - a)^2`

Out[120]= $(-a+x)^2\left(a^5x+(5a^4+a^6)x^2+\left(10a^3+\frac{35a^5}{6}\right)x^3+\left(10a^2+15a^4+\frac{1}{6}(-5a^4-a^6)\right)x^4\right)$

Для преобразования точно заданных выражений нужно использовать встроенные функции, рассмотренные ранее. Пример:

In[121]:=
```
Collect[%, x, Simplify]
```

Out[121]= $a^7x+a^6(3+a^2)x^2+\left(a^5+\frac{23a^7}{6}\right)x^3-\frac{1}{6}a^4(30-21a^2+a^4)x^4+\frac{1}{6}a^3(-60-135a^2+2a^4)x^5+\frac{1}{6}(60a^2+85a^4-a^6)x^6$

Чтобы найти значение функции в некоторой точке, обычно достаточно подставить заданное значение переменной в соответствующее выражение. Как показывает следующий пример, такая подстановка не всегда приводит к желаемому результату.

In[122]:=
```
Sin[π x]/(x - 1) /. x → 1
```

```
Power::infy : Infinite expression 1/0 encountered. More...

∞::indet :
  Indeterminate expression 0 ComplexInfinity encountered. More...
```

Out[122]=
```
Indeterminate
```

Поскольку *Mathematica* представляет отношение двух величин $\frac{a}{b}$ в виде произведения a и $\frac{1}{b}$, то при $a = b = 0$ получается произведение нуля и бесконечно большого числа, т.е. возникает неопределенность вида $0\times\infty$. Об этом выводится соответствующее сообщение, и результат вычислений оказывается неопределенным. Используя функцию **Series**, можно разложить приведенное выше выражение в ряд по x в окрестности точки $x = 1$. В результате получим:

In[123]:=
```
Series[Sin[π x]/(x - 1), {x, 1, 3}]
```

Out[123]= $-\pi+\frac{1}{6}\pi^3(x-1)^2+O[x-1]^4$

Из этого разложения видно, что при $x = 1$ функция принимает значение $(-\pi)$. Действительно, выполняя подстановку, находим:

In[124]:=
```
% /. x → 1
```

Out[124]= $-\pi$

Другим способом устранить возникшую неопределенность является использование функции **Limit**, которая вычисляет предел функции при

стремлении ее аргумента к заданному значению. Соответствующая команда может быть записана в виде:

```
In[125]:= Limit[ Sin[π x] / (x - 1) , x → 1 ]
```

Out[125]= $-\pi$

Предел функции не обязательно является конечным. Так, в следующем примере пределом является бесконечно большое положительное число.

```
In[126]:= Limit[ Sin[2 π x] / (x - 1)^2 , x → 1 ]
```

Out[126]= ∞

Если знаменатель рассматриваемого выражения в окрестности точки $x = 1$ положителен, то числитель принимает отрицательные и положительные значения при $x < 1$ и $x > 1$ соответственно. Поэтому значение предела зависит от направления, вдоль которого происходит приближение к точке $x = 1$. Из полученного результата ясно, что $x \to 1$ со стороны больших значений ($x > 1$). Это обеспечивается значением опции **Direction → – 1**, которое используется по умолчанию. Чтобы приближаться к точке $x = 1$ со стороны меньших значений ($x < 1$), достаточно добавить соответствующую опцию в качестве третьего аргумента функции **Limit**.

```
In[127]:= Limit[ Sin[2 π x] / (x - 1)^2 , x → 1, Direction → 1 ]
```

Out[127]= $-\infty$

В следующих двух примерах результат также зависит от направления, вдоль которого происходит приближение к предельной точке. Однако предельные значения функции оказываются конечными.

```
In[128]:= Limit[ Abs[x] / x, x → 0, Direction → 1]
```

Out[128]= -1

```
In[129]:= Limit[ Abs[x] / x, x → 0, Direction → -1]
```

Out[129]= 1

Если поведение функции в предельной точке неизвестно, то предел не вычисляется.

```
In[130]:= Limit[ x Log[x] + f[x] , x → 0]
```

Out[130]= Limit[f[x] + x Log[x], x → 0]

Этот результат определяется значением опции **Analytic → False**, используемым по умолчанию. Если же функция является аналитической в окрестности предельной точки, то предел может быть вычислен, даже если вид функции не задан. Для этого достаточно установить для опции **Analytic** значение **True**.

```
In[131]:= Limit[ x Log[x] + f[x] , x → 0, Analytic → True]
```

Out[131]= f[0]

В некоторых случаях предельное значение функции не может быть найдено. Например, функция $\cos(\frac{1}{x})$ является быстро осциллирующей при $x \to 0$. В такой ситуации применение функции **Limit** позволяет найти интервал, в пределах которого может находиться предельное значение.

In[132]:= `Limit[Cos[1 / x], x → 0]`

Out[132]= `Interval[{-1, 1}]`

Числа в фигурных скобках обозначают минимальное и максимальное значения функции. С полученным **Interval**-объектом можно выполнять арифметические и другие действия. Прибавляя к нему 1, например, получим другой интервал.

In[133]:= `% + 1`

Out[133]= `Interval[{0, 2}]`

Единица, деленная на полученный интервал, дает область изменения функции $\frac{1}{x}$, если $x \in [0, 2]$.

In[134]:= $\frac{1}{\%}$

Out[134]= $\texttt{Interval}\left[\left\{\frac{1}{2}, \infty\right\}\right]$

Функция **IntervalMemberQ** позволяет определить, находится ли число 5, например, в пределах полученного интервала.

In[135]:= `IntervalMemberQ[ %, 5]`

Out[135]= `True`

Более подробную информацию об **Interval**-объекте можно найти в разделе **Built-in functions** программы помощи **Help Browser**.

5.7. Решение дифференциальных уравнений

Для решения дифференциальных уравнений и их систем в символьной форме в системе *Mathematica* используется функция **DSolve**. При обращении к ней в качестве первого аргумента нужно указать дифференциальное уравнение или список уравнений. Вторым аргументом является искомая функция или список искомых функций. Имя независимой переменной указывается в качестве третьего обязательного аргумента. Пример:

In[136]:= `sol1 = DSolve[ y''[x] + k y[x] == 0, y[x], x]`

Out[136]= $\left\{\left\{\texttt{y[x]} \to \texttt{C[1]}\ \texttt{Cos}\left[\sqrt{\texttt{k}}\ \texttt{x}\right] + \texttt{C[2]}\ \texttt{Sin}\left[\sqrt{\texttt{k}}\ \texttt{x}\right]\right\}\right\}$

Как и функции **Solve**, **NSolve**, **NDSolve**, функция **DSolve** выдает решение в виде списка правил замены. Если на искомую функцию не наложены граничные условия, ищется общее решение дифференциального уравнения. Произвольные постоянные, входящие в общее решение, по умолчанию обозначаются заглавной буквой *C*, за которой в квадратных скобках следует порядковый номер. Такое обозначение определяется значением опции **GeneratedParameters→C**, используемым по умолчанию. При необходимости для произвольных постоянных можно ввести другое обозначение, добавив

соответствующую опцию в качестве четвертого аргумента функции **DSolve**. Пример:

```
In[137]:= DSolve[ y'[x] ^ 2 + k y[x] == 0,
            y[x], x, GeneratedParameters → BB]
```

$$\text{Out[137]= } \left\{\left\{y[x] \to \frac{1}{4}\left(-k\,x^2 - 2\,i\,\sqrt{k}\,x\,BB[1] + BB[1]^2\right)\right\},\right.$$
$$\left.\left\{y[x] \to \frac{1}{4}\left(-k\,x^2 + 2\,i\,\sqrt{k}\,x\,BB[1] + BB[1]^2\right)\right\}\right\}$$

Так как последнее уравнение имеет два решения, список решений состоит из двух элементов. Каждое решение также является списком, состоящим из одного правила замены для искомой функции $y(x)$. Именно поэтому единственное решение, найденное в **sol1**, заключено в двойные фигурные скобки. В случае системы дифференциальных уравнений каждое решение будет содержать несколько правил замены.

Напомним, что у функции **NDSolve**, которая находит численные решения дифференциальных уравнений, присутствие аргумента у искомой функции не является обязательным. Если же у функции **DSolve** в качестве второго аргумента указана искомая функция без аргумента, то решение находится в виде неименованной или "чистой" функции (**pure function**), которая обозначается как **Function**. Пример:

```
In[138]:= sol2 = DSolve[ y''[x] + k y[x] == 0, y, x]
```

$$\text{Out[138]= } \left\{\left\{y \to \text{Function}\left[\{x\}, C[1]\,\text{Cos}\left[\sqrt{k}\,x\right] + C[2]\,\text{Sin}\left[\sqrt{k}\,x\right]\right]\right\}\right\}$$

Хотя решение, полученное в форме неименованной функции, выглядит несколько более громоздко, часто оно оказывается удобнее для дальнейшей работы. Подставим, например, первое решение, найденное в **sol1**, в следующее выражение:

```
In[139]:= y''[x] + k y[x] /. sol1[[1]]
```

$$\text{Out[139]= } k\left(C[1]\,\text{Cos}\left[\sqrt{k}\,x\right] + C[2]\,\text{Sin}\left[\sqrt{k}\,x\right]\right) + y''[x]$$

Как видим, производится подстановка только самой функции $y(x)$. Для вычисления производной требуется дополнительно определить соответствующее правило замены в виде:

```
In[140]:= y''[x] + k y[x] /. sol1[[1]] /.
            y''[x] → D[y[x] /. sol1[[1]], {x, 2}]
```

$$\text{Out[140]= } -k\,C[1]\,\text{Cos}\left[\sqrt{k}\,x\right] - k\,C[2]\,\text{Sin}\left[\sqrt{k}\,x\right] +$$
$$k\left(C[1]\,\text{Cos}\left[\sqrt{k}\,x\right] + C[2]\,\text{Sin}\left[\sqrt{k}\,x\right]\right)$$

Следующий пример показывает, что решение, найденное в **sol2**, может быть легко подставлено в любое выражение.

```
In[141]:= y''[x] + k y[x] /. sol2
```

$$\text{Out[141]= } \left\{-k\,C[1]\,\text{Cos}\left[\sqrt{k}\,x\right] - k\,C[2]\,\text{Sin}\left[\sqrt{k}\,x\right] +\right.$$
$$\left.k\left(C[1]\,\text{Cos}\left[\sqrt{k}\,x\right] + C[2]\,\text{Sin}\left[\sqrt{k}\,x\right]\right)\right\}$$

Чтобы найти частное решение дифференциального уравнения, необходимо в качестве первого аргумента функции **DSolve** привести список, содержащий само уравнение и необходимое количество уравнений, выражающих граничные условия. Пример:

```
In[142]:= DSolve[{y'[x] + k y[x] == 0, y[0] == 2}, y[x], x]
```

$$Out[142]= \{\{y[x] \to 2\, e^{-k\,x}\}\}$$

В следующем примере ищется решение системы двух уравнений, удовлетворяющее заданным граничным условиям.

```
In[143]:= DSolve[{y'[x] == z[x], z'[x] z''[x] == 1,
    y[0] == 1, z[0] == -1, z'[0] == 0}, {y[x], z[x]}, x]
```

$$Out[143]= \left\{\left\{y[x] \to \frac{1}{15}\left(15 - 15\,x - 4\sqrt{2}\,x^{5/2}\right),\right.\right.$$
$$\left.z[x] \to \frac{1}{3}\left(-3 - 2\sqrt{2}\,x^{3/2}\right)\right\},$$
$$\left\{y[x] \to \frac{1}{15}\left(15 - 15\,x + 4\sqrt{2}\,x^{5/2}\right),\right.$$
$$\left.\left.z[x] \to \frac{1}{3}\left(-3 + 2\sqrt{2}\,x^{3/2}\right)\right\}\right\}$$

Как видим, существует два решения, каждое из которых содержит два правила замены для функций $y(x)$ и $z(x)$.

Отметим, что первым аргументом функции **DSolve** может быть не только система дифференциальных уравнений, но и смешанная система, включающая дифференциальные и алгебраические уравнения. Однако такая комбинация уравнений может быть решена только при использовании пятой и последующих версий системы *Mathematica*. Пример:

```
In[144]:= DSolve[{y'[x] + z''[x] == 1, y[x] + z[x] == x, z'[0] == 1},
    {y[x], z[x]}, x]
```

$$Out[144]= \left\{\left\{y[x] \to \frac{1}{8}\,(1 - 8\,e^{x} + 8\,x - C[2]),\right.\right.$$
$$\left.\left.z[x] \to \frac{1}{8}\,(-1 + 8\,e^{x} + C[2])\right\}\right\}$$

Поскольку задано только одно граничное условие, найденное решение системы содержит произвольную постоянную $C[2]$.

Хорошо известно, что поиск решений дифференциальных уравнений в общем случае представляет собой очень трудную задачу. Более того, только в некоторых случаях решение может быть найдено в классе элементарных функций. Иногда решение может быть выражено через специальные функции. Примеры:

```
In[145]:= DSolve[y''[x] + Exp[x] y[x] == 0, y[x], x]
```

$$Out[145]= \left\{\left\{y[x] \to \right.\right.$$
$$\left.\left.\text{BesselJ}\left[0,\, 2\sqrt{e^{x}}\right] C[1] + 2\,\text{BesselY}\left[0,\, 2\sqrt{e^{x}}\right] C[2]\right\}\right\}$$

```
In[146]:= DSolve[y''[x] - x^2 y[x] == 0, y[x], x]

Out[146]= {{y[x] → (x^2)^(1/4) BesselI[-1/4, x^2/2] C[2] Gamma[3/4] / Sqrt[2] +
                   E^(-x^2/2) C[1] HermiteH[-1/2, x]}}
```

Следует отметить, что *Mathematica* может работать со всеми известными специальными функциями и находит решение практически любого линейного дифференциального уравнения, которое можно встретить в современной справочной литературе. Кроме того, она может находить решения многих нелинейных дифференциальных уравнений. Иногда решение может быть найдено в неявной форме. Пример:

```
In[147]:= DSolve[y'[x] + x y[x]^3 + y[x]^2 == 0, y[x], x]

Solve::tdep :
  The equations appear to involve the variables to be
    solved for in an essentially non-algebraic way. More

Out[147]= Solve[1/2 (2 ArcTanh[(-1-2 x y[x])/Sqrt[5]] / Sqrt[5] +
                     Log[(-1 - x y[x] (-1 - x y[x])) / (x^2 y[x]^2)]) == C[1] - Log[x], y[x]]
```

В данном случае исходное дифференциальное уравнение свелось к алгебраическому уравнению. И хотя разрешить это уравнение относительно искомой функции $y(x)$ с помощью функции **Solve** не удается, задача интегрирования дифференциального уравнения оказалась решенной.

Функция **DSolve** может находить решения не только обыкновенных дифференциальных уравнений, в которых искомые функции зависят от одной независимой переменной, но и дифференциальных уравнений в частных производных. Общие решения таких уравнений определяются с точностью до произвольных функций, которые в системе *Mathematica* обозначаются *C*[*n*]. В качестве примера найдем решение одномерного волнового уравнения.

```
In[148]:= DSolve[D[u[t, x], {t, 2}] == v^2 D[u[t, x], {x, 2}],
            u[t, x], {t, x}]

Out[148]= {{u[t, x] → C[1][-t Sqrt[v^2] + x] + C[2][t Sqrt[v^2] + x]}}
```

Получили известное решение в виде линейной комбинации двух произвольных функций *C*[1] и *C*[2] от аргументов $(x \mp v\,t)$ (обратите внимание, что аргументы функций заключены в квадратные скобки). Вид этих функций определяется граничными условиями. Поскольку само существование решения дифференциального уравнения в частных производных существенно зависит от граничных условий, *Mathematica* находит решения таких уравнений в явном виде очень редко.

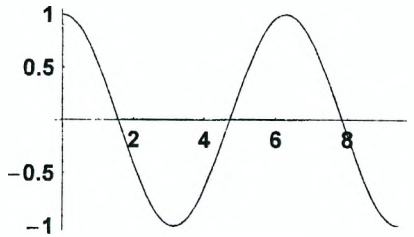
6. Графика в системе *Mathematica*

6.1. Построение графиков функций одной переменной

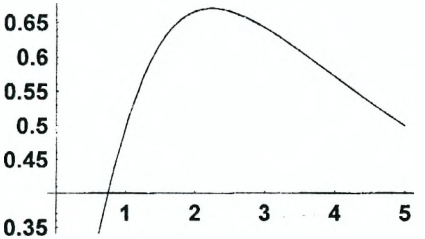
6.1.1. Графики функций, заданных явно

Наиболее часто встречаются случаи, когда необходимо построить график функции, заданной явно, т.е. в виде $y = f(x)$. В таких случаях используется функция **Plot**, при обращении к которой необходимо указать по крайней мере два аргумента: **Plot[f[x], {x, xmin, xmax}]**. В результате получается график функции $f(x)$ на отрезке $x \in [xmin, xmax]$. Примеры:

In[6]:= `Plot[Cos[x], {x, 0, 3 π}];`



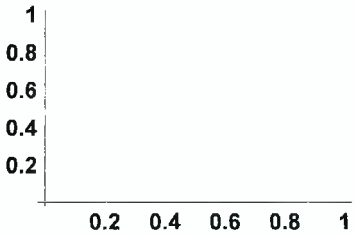
In[7]:= $\texttt{Plot}\left[\frac{3x}{x^2+5}, \{x, 0, 5\}\right];$



Следует отметить, что в качестве первого аргумента используется либо имя функции, известной или определенной заранее, либо конкретное выражение, в котором имеется только одна символьная переменная. Вторым аргументом является список, состоящий из трех элементов: имя независимой переменной, ее начальное и конечное значения. Если, кроме независимой переменной, в выражении содержится еще какой-либо неизвестный символ, то значение функции в любой точке x из отрезка $[xmin, xmax]$ не может быть получено в численном виде и график построен не будет. При этом выводятся сообщения, что соответствующее выражение не является числом при подстановке в него некоторых конкретных значений независимой переменной из заданного интервала и процесс построения графика прекращается. В результате изображаются только координатные оси. Пример:

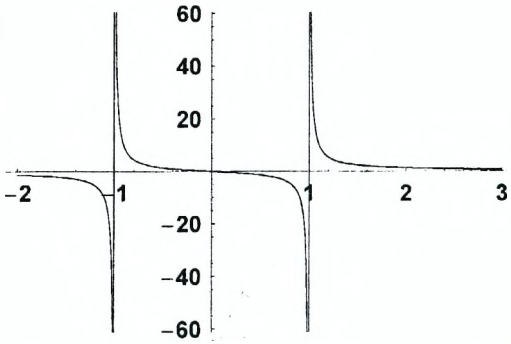
In[8]:= `Plot[a x^2 + 1, {x, -2, 3}];`

```
Plot::plnr :
 a x^2 + 1 is not a machine-size real number at x = -2.. More...

Plot::plnr : a x^2 + 1 is not a
   machine-size real number at x = -1.79717. More...

Plot::plnr : a x^2 + 1 is not a
   machine-size real number at x = -1.57596. More...

General::stop : Further output of Plot::plnr will
   be suppressed during this calculation. More...
```



К достоинствам системы *Mathematica* следует отнести тот факт, что она строит графики даже таких функций, которые могут принимать на отрезке [*xmin*, *xmax*] бесконечно большие значения, например,

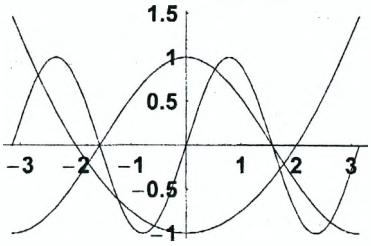
In[9]:= `Plot[2 x/(x^2 - 1), {x, -2, 3}];`



Хотя при $x = \pm 1$ рассматриваемая функция имеет сингулярности, ее график легко строится. При этом вертикальные отрезки, пересекающие ось x в точках $x = \pm 1$, изображают асимптоты функции и не являются частями ее графика. Причина их появления объясняется особенностями работы функции **Plot** и станет понятна в дальнейшем.

Функция **Plot** позволяет построить на одной координатной плоскости графики сразу нескольких функций. В этом случае ее первым аргументом должен быть соответствующий список функций, например,

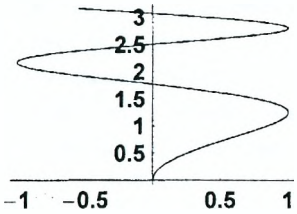
In[10]:= `Plot[{Cos[x], Sin[2 x], x^2/4 - 1}, {x, -π, π}];`



6.1.2. Графики функций, заданных параметрически

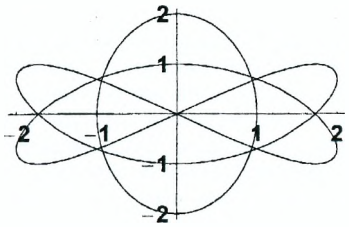
Функция $y = f(x)$ является заданной параметрически, если указаны две зависимости: $x = x(t)$, $y = y(t)$. Для построения графиков таких функций используется команда **ParametricPlot**, при обращении к которой нужно указать по крайней мере два аргумента: **ParametricPlot[{x[t], y[t]}, {t, tmin, tmax}]**. Первым аргументом является список двух выражений, определяющих функцию, а вторым – список, определяющий интервал изменения параметра *t*. Пример:

In[11]:= `ParametricPlot[{Sin[t], √t}, {t, 0, 10}];`



Если на одной координатной плоскости необходимо изобразить сразу несколько функций, заданных параметрически, то в качестве первого аргумента функции **ParametricPlot** нужно указать список, элементами которого являются списки вида: {*x*[*t*], *y*[*t*]}. Пример:

In[12]:= `ParametricPlot[{ {Cos[t], 2 Sin[t]},`
`{2 Sin[2 t], Sin[3 t]}}, {t, 0, 2 π}];`



6.1.3. Управление процессом построения графика

При построении графика функции $y = f(x)$ *Mathematica* поступает стандартным образом – вычисляет значения функции $y_i = f(x_i)$ в последовательности точек x_i из интервала **[xmin, xmax]**, отмечает точки (x_i, y_i) на координатной плоскости и соединяет их отрезками. При этом она должна принять ряд решений, а именно: определить, в каких точках отрезка **[xmin, xmax]** следует вычислить значения функции, какую область изменения функции изобразить на графике, какой при этом выбрать масштаб, как провести координатные оси и т.д.. Управление этими процессами осуществляется с помощью набора опций, каждая из которых имеет свое имя, а соответствующее значение опции задается с помощью правила замены вида: **Опция → значение**. По умолчанию для каждой опции задано некоторое значение, так что при стандартном обращении к функции **Plot** вида **Plot[f[x], {x, xmin, xmax}]** обычно получается хороший график. Чтобы вывести список опций, которые имеются у данной функции, вместе с их значениями, заданными по умолчанию, используем команду **Options**.

```
In[13]:= Options[Plot]

Out[13]= {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}
```

Легко убедиться в том, что у функции **Plot** имеется тридцать опций.

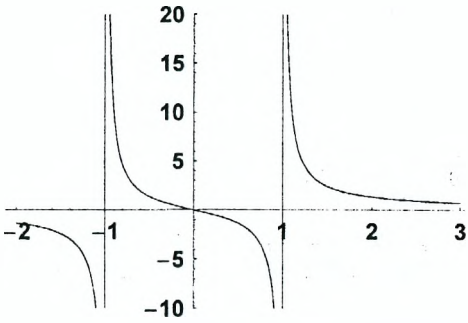
```
In[14]:= Length[%]

Out[14]= 30
```

Точно такой же набор опций имеет функция **ParametricPlot**. При этом управление процессом построения графика сводится к замене значений некоторых опций таким образом, чтобы получить желаемый результат. Для этого достаточно в качестве третьего и последующих аргументов функции **Plot** или **ParametricPlot** ввести соответствующие правила замены в любом порядке, указав в правой их части необходимые значения. Далее мы рассмотрим назначение и возможные значения наиболее часто используемых опций.

Опция **PlotRange** определяет диапазон изменения функции, который сле-

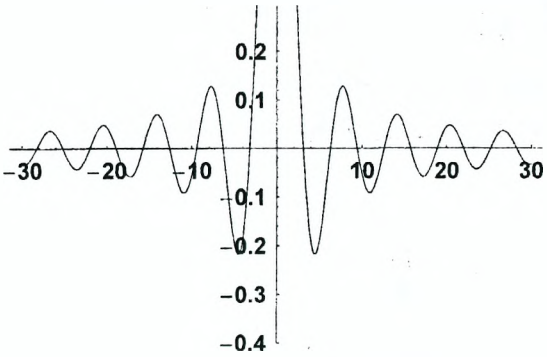
дует изобразить на графике. Чтобы при построении графика функции $y = \frac{2x}{x^2-1}$, например, выделить только область $y \in [-10, 20]$, достаточно ввести следующую команду:

```
In[15]:= Plot[2 x / (x^2 - 1), {x, -2, 3}, PlotRange → {-10, 20}];
```



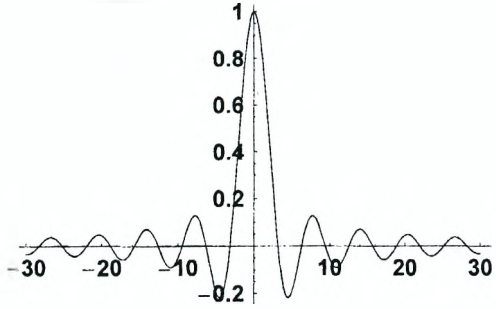
Ранее мы уже строили график этой функции и значение **Automatic**, используемое для опции **PlotRange** по умолчанию, обеспечивало изображение функции в области $y \in [-60, 60]$. По умолчанию *Mathematica* стремиться показать на графике ту область изменения функции, в которой находится большая часть ее вычисленных значений. При этом удаленные от основной области точки не изображаются, что дает возможность показать в ней характер изменения функции более детально. Значение **Automatic** можно установить в комбинации с конкретным числом, ограничивающим минимальное или максимальное значения функции. Так, в следующем примере мы установим только нижнюю границу значений функции.

```
In[16]:= Plot[Sin[x] / x, {x, -30, 30},
          PlotRange → {-0.4, Automatic}];
```



Если же на графике необходимо изобразить всю область изменения функции, то для опции **PlotRange** следует установить значение **All**.

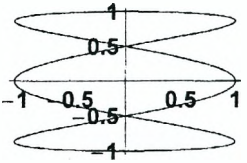
In[17]:= `Plot[Sin[x]/x, {x, -30, 30}, PlotRange → All];`



Однако при наличии у функции сингулярностей в области **[xmin, xmax]** выбор значения **All** для опции **PlotRange** не приведет к построению хорошего графика.

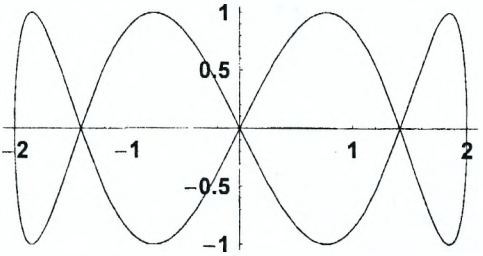
Размеры области на экране, в которой изображается график, определяются значением опции **ImageSize** и задаются в точках принтера, например, **ImageSize → {200, 200}**. При этом предполагается, что на один дюйм приходится 72 точки. Используемое по умолчанию значение этой опции **ImageSize → Automatic** обычно соответствует 288 точкам или 4 дюймам. Следует отметить, что при работе в операционной системе **Windows** можно легко увеличить или уменьшить размер построенного графика с помощью мыши, причем установленный размер графика запоминается и воспроизводится при его повторном построении. По умолчанию вся область, отводимая для графика опцией **ImageSize**, используется по назначению, что определяется значением опции **PlotRegion→ Automatic**. Всю эту область можно рассматривать как прямоугольник на плоскости *xOy*, причем координаты его левого нижнего угла равны (0, 0), а правого верхнего угла – (1, 1). В пределах этой области координаты *x* и *y* принимают значения от 0 до 1. Чтобы выделить часть прямоугольника, в которой следует изобразить график, нужно указать в качестве значения опции **PlotRegion** конкретные интервалы изменения *x* и *y*, например, **PlotRegion → {{0.2, 0.8}, {0.1, 0.7}}**. Это дает возможность оставить дополнительные поля слева, справа, снизу и сверху от графика. Пример:

In[18]:=
```
ParametricPlot[{Cos[3 t], Sin[t]}, {t, 0, 2 π},
    PlotRegion → {{0.1, 0.8}, {0.1, 0.7}}];
```



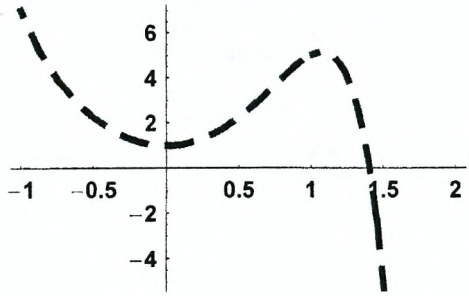
Отношение высоты к ширине прямоугольника, в который помещается график, определяется опцией **AspectRatio**, для которой по умолчанию установлено значение $\frac{2}{1+\sqrt{5}}$, равное обратному золотому соотношению. Эту величину можно изменить на любую другую. Например, при значении опции **AspectRatio → 1** график будет нарисован в квадрате. Чтобы масштаб по осям *x* и *y* был одинаковым, для опции **AspectRatio** нужно установить значение **Automatic**.

```
In[19]:= ParametricPlot[{2 Cos[t], Sin[4 t]},
          {t, -π, π}, AspectRatio → Automatic];
```



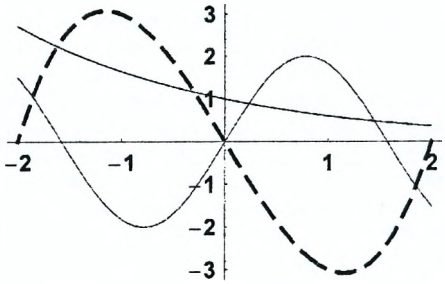
Опция **PlotStyle** позволяет определять параметры кривой, изображающей поведение функции. Толщина линии определяется директивой **Thickness**, которая вызывается с одним аргументом. Его величина равняется отношению толщины линии к ширине всего графика и по умолчанию примается равной 0.004. Легко убедиться в том, что использование директивы **Thickness[1]** приведет к закрашиванию всей области графика в черный цвет, который используется для изображения линий по умолчанию. Для определения цвета линии в системе *Mathematica* имеется несколько директив. Например, директива **GrayLevel** задает различные оттенки серого цвета, начиная от черного **GrayLevel[0]** и заканчивая белым **GrayLevel[1]**. Ее аргумент может изменяться в пределах от 0 до 1. Директива **RGBColor[r, g, b]** определяет цвет как комбинацию красного, зеленого и синего цветов. Каждый из трех ее аргументов может изменяться в пределах от 0 до 1 и определяет интенсивность соответствующей составляющей цвета. При этом значение аргумента 0 означает, что данная составляющая в цвете отсутствует, а значение 1 обеспечивает ее максимальное включение. Например, **RGBColor[1,0,0]** соответствует красному цвету, а **RGBColor[1,1,1]** – белому. Аналогично работает директива **CMYKColor[c,m,y,b]**, которая вызывается с четырьмя аргументами, и обеспечивает смешение четырех цветов, причем иинтенсивность каждого цвета зависит от значения соответствующего аргумента. Директива **Hue[h]** определяет все цвета радуги в зависимости от значения аргумента. При его изменении от 0 до 1 цвет изменяется от красного до фиолетового и обратно, причем **Hue[0]** и **Hue[1]** соответствуют красному цвету. Директива **Dashing** позволяет делать разрывы в линии, превращая ее в штриховую или штрих-пунктирную линию. Ее аргументом является список чисел, соответствующих длине рисуемых и нерисуемых участков. Если этот список содержит только одно число, то длины рисуемых и нерисуемых участков будут одинаковы. При этом само число равняется отношению длины участка в ширине всего графика.

```
In[20]:= p1 = Plot[-x^7 + 5 x^2 + 1, {x, -1, 2},
            PlotStyle → {Thickness[0.015],
              RGBColor[1, 0, 0.8], Dashing[{0.05}]}];
```



Если на графике изображаются несколько кривых, то для каждой из них можно выбрать свой стиль, указав в качестве значения опции **PlotStyle** список, элементами которого являются списки директив, последовательно определяющие стиль каждой кривой.

```
In[21]:= p2 = Plot[{2 Sin[2 x], Exp[-x / 2], x^3 - 4 x},
            {x, -2, 2}, PlotStyle → {{RGBColor[1, 0, 0]},
                {}, {Thickness[0.009], Hue[0.7],
                    Dashing[{0.04, 0.03}]}}];
```

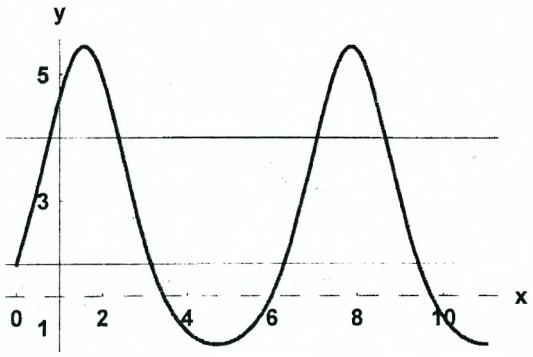


Как видно из приведенного примера, директивы, относящиеся к одной кривой, должны заключаться в фигурные скобки даже в том случае, если указывается только одна директива. При отсутствии в фигурных скобках каких-либо директив для соответствующей кривой используются директивы, заданные по умолчанию. Если же значением опции **PlotStyle** является набор директив, заключенный в двойные фигурные скобки, например, **PlotStyle → {{ Thickness[0.01], Hue[0.3]}}**, то все кривые будут изображаться одинаково указанным стилем.

По умолчанию для опции **Axes** установлено значение **Automatic**, которое означает, что на графике должны быть изображены координатные оси. При этом имеется возможность определить стиль изображения осей, точку их пересечения, снабдить их надписями и делениями, а также нанести на

график координатную сетку. Стиль изображения координатных осей определяется значением опции **AxesStyle**, которое должно быть списком, состоящим из двух элементов. Каждый элемент в свою очередь есть список, содержащий набор директив, используемый при изображении соответствующей оси. Опция **AxesOrigin → {x,y}** указывает, что на рисунке координатные оси должны пересекаться в точке с координатами *(x,y)*. Опция **AxesLabel** позволяет снабдить координатные оси надписями, которые должны быть заключены в кавычки и заданы в виде списка из двух элементов. По умолчанию на осях отмечаются масштабные деления, что определяется значением опции **Ticks → Automatic**. Выбирая для этой опции значение **False** или **None**, можно удалить масштабные метки на одной или обеих осях. Кроме того, для каждой оси можно задать конкретный список делений, которые нужно отметить на ней. Чтобы изобразить на графике координатную сетку, для опции **GridLines** нужно установить значение **Automatic** или указать для каждой оси список координат точек, через которые должны проходить линии сетки. Пример:

In[22]:=
```
p3 = Plot[2 Exp[Sin[x]], {x, 0, 11},
    AxesStyle → {{Dashing[{0.04}]}, {}},
    AxesOrigin → {1, 1.5}, AxesLabel → {"x", "y"},
    PlotStyle → {Thickness[0.008], Hue[0.9]},
    Ticks → {{0, 2, 4, 6, 8, 10}, {1, 3, 5}},
    PlotRange → All, GridLines → {None, {2, 4}}];
```



Заметим, что выбор для опции **Axes** значения **False** или **None** отменяет действие опций **AxesStyle**, **AxesOrigin**, **AxesLabel** и **Ticks**.

Выбирая для опции **Frame** значение **True**, можно нарисовать вокруг графика рамку. При этом автоматически происходит перенос масштабных делений с координатных осей на стороны рамки. Чтобы этого не происходило, для опции **FrameTicks** необходимо установить значение **None**. Стиль рамки определяется опцией **FrameStyle**, которая позволяет задать толщину линии рамки, ее цвет и штриховку. С помощью опции **FrameLabel** вокруг рамки с каждой стороны можно сделать надписи, задавая их в виде списка и заключая каждую надпись в кавычки. Первым элементом списка является надпись снизу, затем надписи слева, сверху и справа. Используемое по

умолчанию значение опции **RotateLabel→True** обеспечивает размещение надписей вдоль соответствующих сторон рамки. Естественно, выбор для опции **Frame** значения **False** отменяет действие опций **FrameTicks**, **FrameStyle** и **FrameLabel**. Отметим также опцию **PlotLabel**, которая позволяет снабдить график заголовком. Пример:

```
In[23]:= p4 = Plot[x^2 Exp[x^2 / 2], {x, -2, 2},
            PlotStyle →
            {Thickness[0.012], RGBColor[1, 0, 1]},
          AxesOrigin → {0, 5}, Frame → True,
          FrameStyle → {Hue[0.6], Thickness[0.009]},
           FrameLabel → {FontForm["Координата  X",
                              {"Courier", 12}], "Y",
            "Н а д п и с ь     с в е р х у", "с п р а в а"},
           PlotLabel → FontForm["Г р а ф и к  ф у н к ц и и
              y = x^2 Exp[x^2/2] \n", {"Times-Bold", 12}],
            FrameTicks → None, DefaultFont → {"Times", 11}];
```

Г р а ф и к ф у н к ц и и $y = x^2 \, Exp[x^2/2]$

Как видно из приведенного примера, для надписи под нижней стороной рамки "Координата X" и заголовка графика с помощью директивы **FontForm** тип и размер шрифта установлены отдельно. Для всех остальных надписей используется шрифт такого типа и размера, который определяется опцией **DefaultFont**. Использование символа перехода к новой строке **\n** позволяет поднять заголовок графика на одну строку вверх и тем самым отделить его от надписи, сделанной над верхней стороной рамки. Заметим также, что цвет, которым изображаются оси, деления на них и надписи, определяется опцией **DefaultColor**.

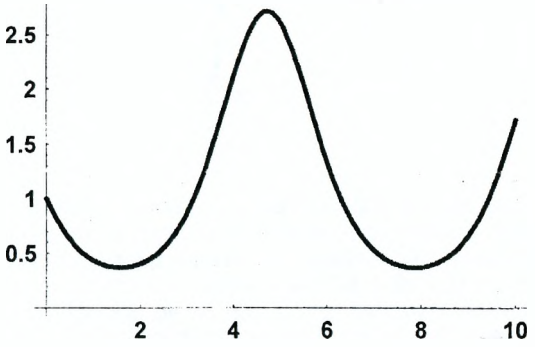
Опции **Prolog** и **Epilog** позволяют нанести на график дополнительные графические объекты, следуя стандартному порядку: сначала изображаются объекты, определяемые опцией **Prolog**, затем строится график, а потом наносятся объекты, определяемые опцией **Epilog**. В тех местах, где происходит наложение различных объектов, видимыми оказываются те из них, которые строятся последними.

Чтобы сформировать графический объект, но не изображать его, для опции **DisplayFunction** нужно установить значение **Identity**, например,

```
In[24]:= p5 = Plot[Exp[-Sin[x]], {x, 0, 10},
             PlotStyle → {Hue[0], Thickness[0.01]},
             DisplayFunction → Identity];
```

Чтобы показать полученный график в нужном месте, используется функция **Show**. При обращении к ней нужно изменить значение опции **DisplayFunction** на **$DisplayFunction**.

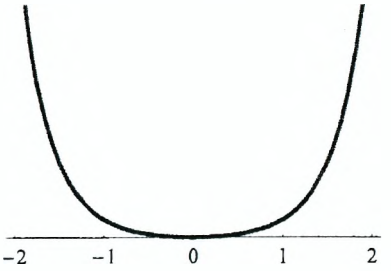
```
In[25]:= p6 = Show[p5, DisplayFunction → $DisplayFunction];
```



6.2. Наложение нескольких графиков, создание таблицы графиков

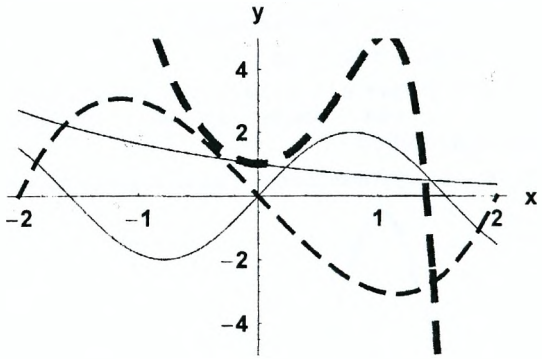
Построенный ранее график можно показать в любом месте документа, используя функцию **Show**. При этом можно изменить некоторые опции, установленные ранее при обращении к функциям **Plot** и **ParametricPlot**. Убирая на графике *p4*, например, ось *y*, заголовок графика и рамку, получаем:

```
In[26]:= p7 = Show[p4, Axes → {True, None},
             PlotLabel → None, Frame → False];
```



С помощью функции **Show** можно изобразить на одной координатной плоскости сразу несколько построенных ранее графиков. Для этого достаточно указать их имена в качестве первого и последующих аргументов функции **Show**.

```
In[27]:= Show[p1, p2, PlotRange → {-5, 5},
          AxesLabel → {"x", "y"}];
```



Несколько графиков можно представить в виде таблицы, используя функцию **GraphicsArray**. Ее аргументом является список, элементы которого представляют собой списки имен графиков, размещаемых в одной строке. Представить графики **p1**, **p2**, **p6** и **p7**, например, в виде таблицы, в первой строке которой находятся графики **p1** и **p2**, а во второй – **p6** и **p7**, и поместить таблицу в рамку, можно с помощью следующей команды:

```
In[28]:= Show[GraphicsArray[{{p1, p2}, {p6, p7}}],
          Frame → True, PlotLabel →
           FontForm["Т а б л и ц а   г р а ф и к о в",
            {"Times-Bold", 14}]];
```

С помощью опции **PlotLabel** мы также добавили к таблице заголовок.

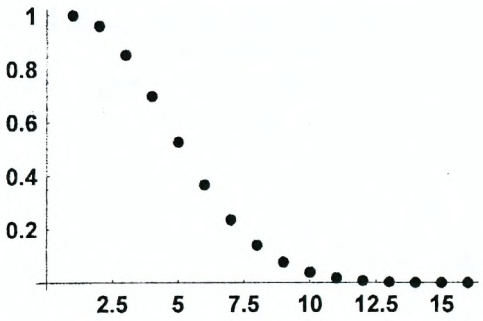
6.3. Графическое представление численных данных

Для графического представления численных данных на плоскости служит функция **ListPlot**. Ее первым аргументом является список, который может быть двух видов. В первом случае элементами списка являются действительные числа. В качестве примера рассмотрим список значений функции e^{-x^2}.

```
In[29]:= dat1 = Table[Exp[-i^2], {i, 0, 3, 0.2}] // N
Out[29]= {1., 0.960789, 0.852144, 0.697676, 0.527292,
          0.367879, 0.236928, 0.140858, 0.0773047,
          0.0391639, 0.0183156, 0.00790705, 0.00315111,
          0.00115923, 0.000393669, 0.00012341}
```

При обращении к функции **ListPlot** на координатной плоскости будут отмечены точки, координатами *x* которых являются натуральные числа 1, 2, 3 и т.д., а координатами *y* – числа из списка **dat1** в порядке их следования. Стиль изображения точек определяется опцией **PlotStyle**, которая позволяет задать цвет и размер точки (директива **PointSize**). Отметим, что аргумент директивы **PointSize** имеет смысл отношения диаметра точки к ширине всего графика и должен быть достаточно мал. В результате получаем следующий рисунок:

```
In[30]:= p8 = ListPlot[dat1,
            PlotStyle → {RGBColor[1, 0, 0], PointSize[0.025]}];
```



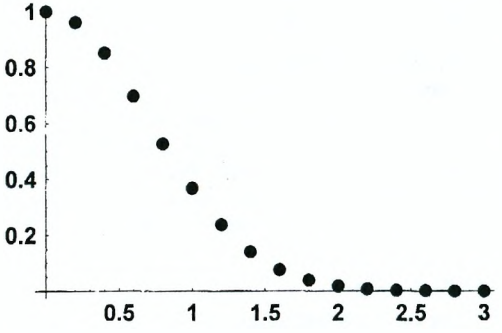
Очевидно, полученный рисунок неточно отображает поведение функции, поскольку вдоль оси *Ox* отложены не значения координаты *x*, а номер точки в списке. Чтобы избежать этой неточности, в качестве элементов списка необходимо задать пары чисел – координаты *x* и *y* каждой точки. Соответствующий список значений функции e^{-x^2} имеет вид:

```
In[31]:= dat2 = Table[{x, Exp[-x^2]}, {x, 0, 3, 0.2}] // N
```

```
Out[31]= {{0., 1.}, {0.2, 0.960789}, {0.4, 0.852144},
         {0.6, 0.697676}, {0.8, 0.527292}, {1., 0.367879},
         {1.2, 0.236928}, {1.4, 0.140858}, {1.6, 0.0773047},
         {1.8, 0.0391639}, {2., 0.0183156}, {2.2, 0.00790705},
         {2.4, 0.00315111}, {2.6, 0.00115923},
         {2.8, 0.000393669}, {3., 0.00012341}}
```

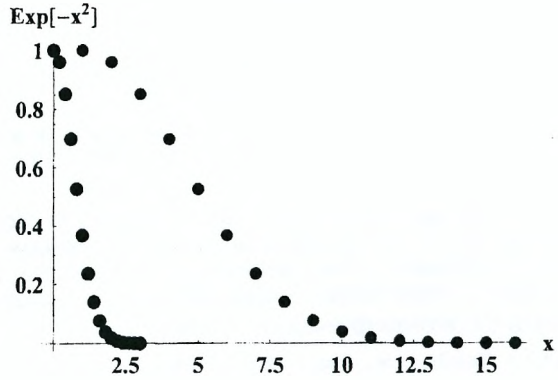
В результате график принимает вид:

```
In[32]:= p9 = ListPlot[dat2,
           PlotStyle → {RGBColor[0, 0, 1], PointSize[0.028]}];
```



Для сравнения изобразим два графика на одной плоскости с помощью функции **Show**, добавляя обозначения осей.

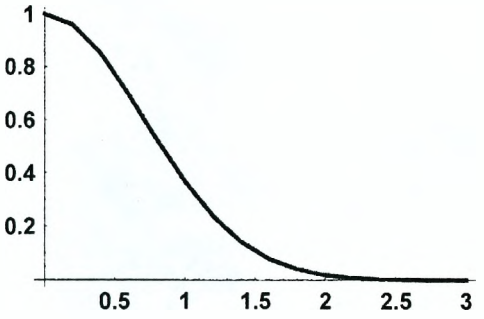
```
In[33]:= Show[p8, p9, AxesLabel → {"x", "Exp[-x^2]"},
           DefaultFont → {"Times-Bold", 12}];
```



Функция **ListPlot** также имеет ряд опций, изменение значений которых позволяет управлять процессом построения графика. Большая их часть имеют те же названия и назначение, что и опции функции **Plot**. Отметим только опцию **PlotJoined**, значение **True** которой указывает на необходимость

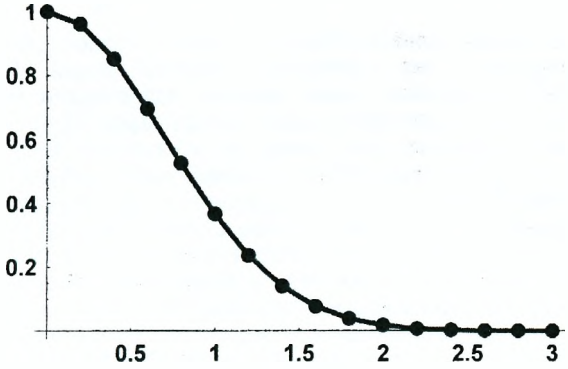
соединить точки на плоскости прямолинейными отрезками. В этом случае вместо директивы **PointSize** необходимо использовать директиву **Thickness**, определяющую ширину линии. Соединяя точки на графике *p9*, например, получаем:

```
In[34]:= p10 = ListPlot[dat2, PlotJoined → True,
           PlotStyle → {RGBColor[1, 0, 0], Thickness[0.01]}];
```



С помощью функции **Show** можно совместить графики **p9** и **p10**.

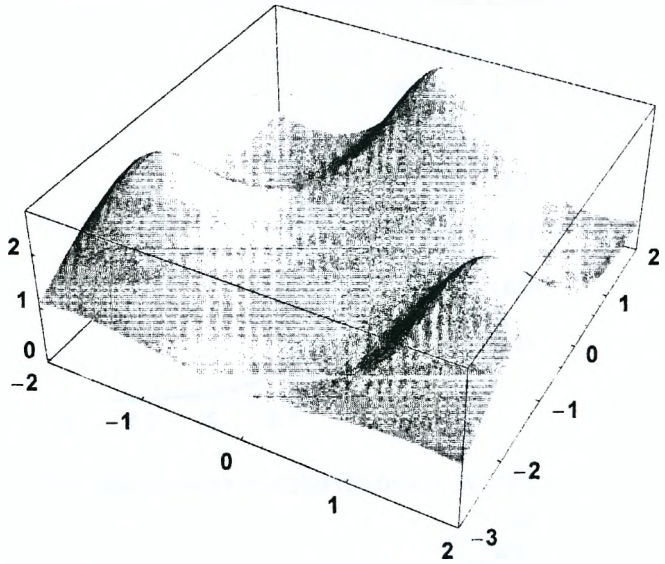
```
In[35]:= Show[p9, p10];
```



6.4. Постороение графиков функций двух переменных

График функции двух переменных $z = f(x, y)$ представляет собой некоторую поверхность в трехмерном пространстве. Для ее построения в системе *Mathematica* используется функция **Plot3D**, при обращении к которой необходимо указать три обязательных аргумента. Первым аргументом является имя функции, встроенной или определенной заранее, или выражение, содержащее не более двух переменных. Второй и третий аргументы представляют собой списки, элементами которых являются имена переменных, их минимальные и максимальные значения. Пример:

```
In[36]:= Plot3D[Exp[Cos[2 x] Sin[y]], {x, -2, 2},
          {y, -3, 2}, PlotPoints → 50, Mesh → False];
```



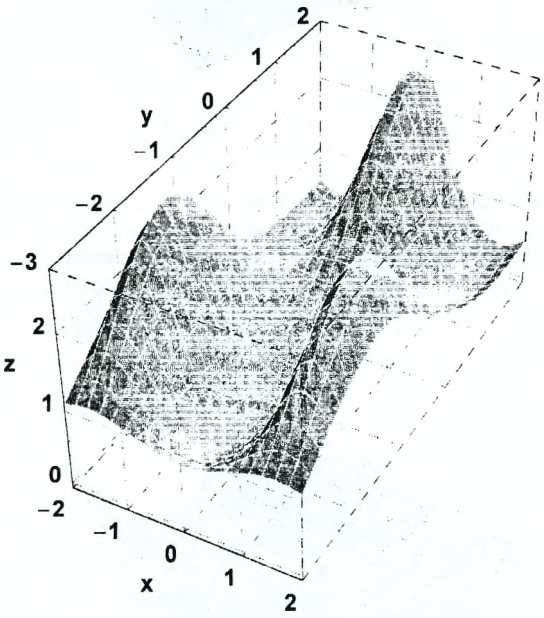
Как и в случае функции **Plot**, управление процессом построения графика осуществляется путем изменения значений опций. Так, в предыдущем примере для получения более плавной поверхности мы увеличили число точек, в которых вычисляются значения функции, до 50 (опция **PlotPoints**), а также убрали границы всех площадок, из которых составляется изображаемая поверхность (опция **Mesh**). Полный набор опций, которые имеются у функции **Plot3D**, вместе с из значениями, используемыми по умолчанию, можно получить с помощью команды **Options[Plot3D]**. Далее мы рассмотрим только некоторые опции из списка, включающего 40 наименований.

По умолчанию для опции **Mesh** установлено значение **True**, что приводит к появлению на изображаемой поверхности линий, показывающих границы площадок, на которые разбита поверхность. При этом стиль изображаемых линий можно задать с помощью опции **MeshStyle**. Чтобы легче было представить себе пространственную структуру поверхности, она помещается внутрь прямоугольного параллелепипеда (**Boxed→True**). Опция **BoxStyle** позволяет определить стиль изображения ребер параллелепипеда. Опция **BoxRatios** аналогична опции **AspectRatio** у функции **Plot** и определяет соотношение длин ребер параллелепипеда. Выбор для нее значения **{1, 2, 1}**, например, приводит к тому, что длина ребра, параллельного оси *y*, будет в два раза больше по сравнению с двумя другими ребрами. Сама же опция **AspectRatio** по-прежнему определяет соотношение сторон прямоугольного окна, внутри которого изображается график. Опция **FaceGrids** позволяет нанести на грани параллелепипеда координатную сетку. Значения этой опции являются списками, состоящими из трех элементов, один из которых есть 1 или –1, а остальные – нули. При этом значение **{-1,0,0}**, например, соответствует дальней грани, проходящей перпендикулярно к оси *x*.

Чтобы лучше представить себе форму поверхности, она раскрашивается

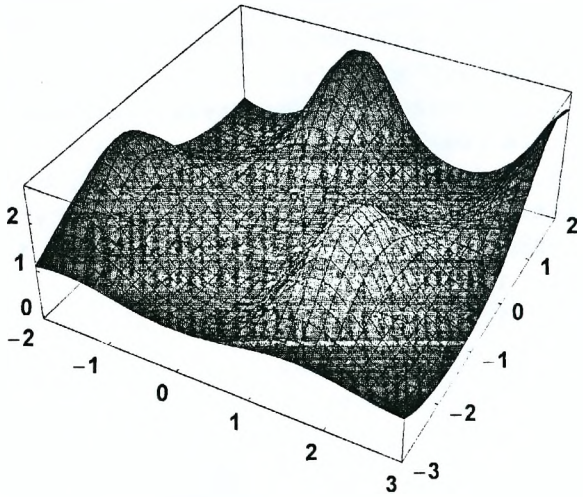
и на нее наносятся тени (опция **Shading→True**), причем режим раскрашивания определяется опцией **Lighting**. В случае значения **Lighting→True** на поверхность наносится равномерный фон, цвет которого определяется опцией **AmbientLight**, и включаются дополнительные источники света. Расположение источников и их цвет определяются опцией **LightSources**. Следующий пример демонстрирует использование описанных выше опций.

```
In[37]:= Plot3D[Exp[Cos[2 x] Sin[y]], {x, -2, 2}, {y, -3, 2},
          PlotPoints → {20, 30}, MeshStyle → GrayLevel[0.8],
          BoxStyle → Dashing[{0.02}], FaceGrids →
           {{-1, 0, 0}, {0, 0, -1}, {0, -1, 0}, {0, 1, 0}},
          BoxRatios → {1, 2, 1}, AxesLabel → {"x", "y", "z"},
          LightSources → {{{1., 0., 1.}, RGBColor[0.5, 0, 0]},
            {{1., 1., 1.}, RGBColor[0, 0.8, 0]},
            {{0., 1., 1.}, RGBColor[0.5, 0, 1]}}];
```



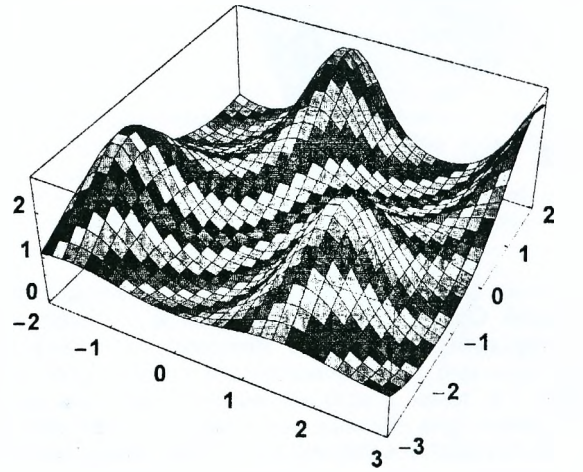
В случае значения опции **Lighting → False** цвет раскраски поверхности определяется опцией **ColorFunction**, для которой по умолчанию установлено значение **GrayLevel**. При этом в зависимости от значения функции $z = f(x, y)$ соответствующие точки поверхности приобретают различные оттенки серого цвета, т.е. точки, в которых значения функции одинаковы, имеют один цвет. В следующем примере мы изменяем цветовую функцию так, что поверхность окрашивается в различные оттенки синего цвета.

```
In[166]:= Plot3D[Exp[Cos[2 x] Sin[y]], {x, -2, 3},
          {y, -3, 2}, PlotPoints → 30, Lighting → False,
          ColorFunction → (RGBColor[0, 0, #] &)];
```



Характер раскрашивания поверхности можно изменить, поместив в фигурных скобках в качестве первого аргумента функции **Plot3D** вместе с функцией *f(x,y)* соответствующую цветовую функцию, например,

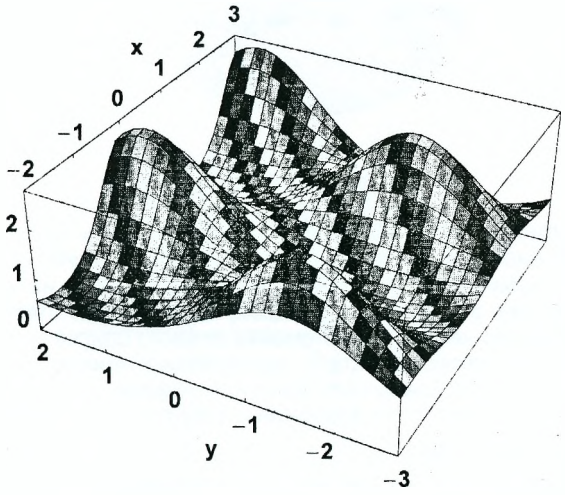
```
In[167]:= Plot3D[{Exp[Cos[2 x] Sin[y]], Hue[x - y]},
          {x, -2, 3}, {y, -3, 2}, PlotPoints → 30];
```



Как видим, на графике одинаковый цвет имеют площадки, расположенные вдоль прямых $x - y = \text{const}$. Отметим еще опцию **ViewPoint**, которая позво-

ляет определить координаты точки, из которой мы смотрим на поверхность. С помощью этой опции можно также посмотреть на уже построенный ранее график из другой точки. Пример:

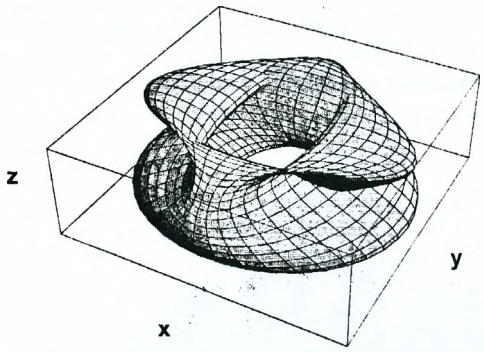
In[40]:=
```
Show[%, AxesLabel → {"x", "y", ""},
      ViewPoint -> {-2.659, -1.389, 1.859}];
```



Для удобства выбора точки обзора в разделе **Input** главного меню имеется команда **3D ViewPoint Selector**. По этой команде открывается дополнительное окно, в котором изображается ограничивающий поверхность параллелепипед и указываются координаты *x*, *y*, *z* точки обзора. Изменить их можно либо указав нужные значения в специальных окошках, либо перемещая соответствующие ползунки, либо непосредственно вращая параллелепипед с помощью мыши и наблюдая за изменением его положения. Как только точка обзора выбрана, необходимо нажать клавишу **Paste**. При этом опция **ViewPoint** вместе с выбранными координатами точки обзора будет вставлена в то место документа, в котором находился курсор до вывова команды **3D ViewPoint Selector**.

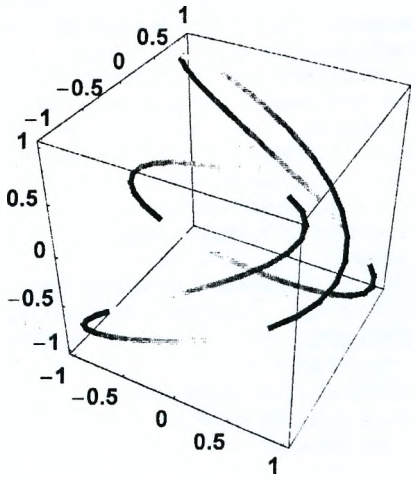
Для построения поверхности, заданной параметрически, т.е. в виде трех функций $x = x(u, v)$, $y = y(u, v)$, $z = z(u, v)$ от двух переменных u и v, используется функция **ParametricPlot3D**. Ее первым аргументом должен быть список функций **{x,y,z}**, определяющих поверхность, вторым и третьим – списки, определяющие области изменения переменных u и v, а далее могут следовать опции, аналогичные опциям функции **Plot3D**. Пример построения поверхности, заданной параметрически, приводится ниже.

In[41]:=
```
ParametricPlot3D[
      {(2 + Sin[2 u]) Cos[v], (2 - Sin[u]) Sin[v], 3 + Cos[u]},
      {v, 0, 2 π}, {u, 0, 2 π}, Ticks → None,
      PlotPoints → 40, AxesLabel → {"x", "y", "z"}];
```



Функция **ParametricPlot3D** позволяет также построить пространственную кривую, заданную параметрически, т.е. в виде $x = x(t)$, $y = y(t)$, $z = z(t)$. В этом случае обязательными являются два аргумента, первый из которых есть список функций **{x,y,z}**, а второй – список, определяющий область изменения параметра *t*. В качестве четвертого элемента первого списка можно добавить список директив, определяющих стиль изображения кривой. Так, в следующем примере заданы толщины изображения кривой и функция, определяющая ее цвет в зависимости от значения параметра.

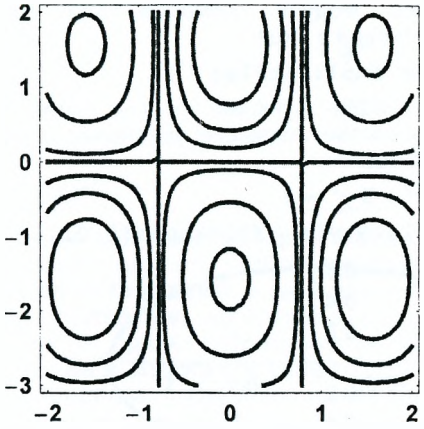
```
In[42]:= ParametricPlot3D[
          {Cos[3 t], Sin[t], Sin[2 t], {Thickness[0.015],
           GrayLevel[Mod[t, 1]]}}, {t, 0, 2 π}];
```



Для отображения формы поверхности на плоскости часто используют метод контурных линий. В этом случае поверхность $z = f(x, y)$ пересекается множеством плоскостей $z = \text{const}$, и получаемые в сечении линии уровня изображаются на плоскости. В системе *Mathematica* эта процедура реализуется с помощью функции **ContourPlot**, первые три обязательных аргумен-

та которой такие же, как и у функции **Plot3D**. Как обычно, управление процессом построения линий уровня осуществляется рядом опций, из которых отметим лишь несколько. Так, с помощью опции **ContourStyle** можно определить стиль изображения линий, опция **Contours** позволяет определить конкретные значения уровней *z*, на которых необходимо провести сечения, а опция **ContourShading** показывает, следует или нет раскрашивать области между контурными линиями. Пример:

```
In[43]:= ContourPlot[Exp[Cos[2 x] Sin[y]], {x, -2, 2}, {y, -3, 2},
          ContourStyle → {Thickness[0.012], RGBColor[0.8, 0, 1]},
          Contours → {0.4, 0.6, 0.9, 1, 1.2, 1.5, 2},
          PlotPoints → 50, ContourShading → False];
```



Если прямоугольную область на плоскости *xOy* разбить на маленькие прямоугольники и раскрасить их так, что прямоугольники, для которых значения функции $z = f(x, y)$ одинаковы, будут иметь один цвет, то можно отбразить форму поверхности $z = f(x, y)$ на плоскости с помощью цвета. Соответствующая функция называется **DensityPlot** и также имеет три обязательных аргумента, смысл которых понятен из следующего примера.

```
DensityPlot[Exp[Cos[2 x] Sin[y]], {x, -2, 2},
  {y, -3, 2}, Mesh → False, PlotPoints → 80];
```

После обязательных аргументов могут следовать опции, которые несколько изменяют вид получаемого графика. Так, чтобы наблюдался плавный переход цвета, с помощью опции **PlotPoints** мы увеличили количество отрезков, на которые разбиваются интервалы изменения координат *x* и *y*, а также убрали на плоскости *xOy* координатную сетку (опция **Mesh→False**). По умолчанию для раскрашивания прямоугольников используется цветовая функция **GrayLevel**, причем прямоугольники, для которых значения *z* минимальны и максимальны, на рисунке будут соответственно черными и белыми. Заметим, что с помощью опции **ColorFunction** цветовую функцию можно изменить.

6.5. Дополнительные графические возможности

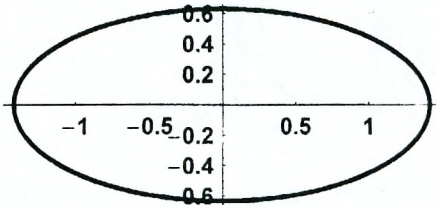
Как уже ранее отмечалось, при запуске системы *Mathematica* доступными являются не все ее встроенные функции, а только те из них, которые используются наиболее часто. Кроме того, имеется ряд дополнительных модулей (**packages**), в которых определяются функции, существенно расширяющие графические возможности системы. Чтобы получить список модулей и их описание, достаточно запустить программу помощи **Help Browser** и выбрать последовательно следующие закладки: **Add-ons & Links** ↦ **Standard Packages** ↦ **Graphics**. Далее мы рассмотрим только некоторые из имеющихся функций.

В качестве первого примера рассмотрим модуль **ImplicitPlot**. Напомним, что любая дополнительная функция становится доступной только после загрузки соответствующего модуля. Команда для загрузки модуля **ImplicitPlot** может быть записана в виде:

In[44]:=
```
Needs["Graphics`ImplicitPlot`"];
```

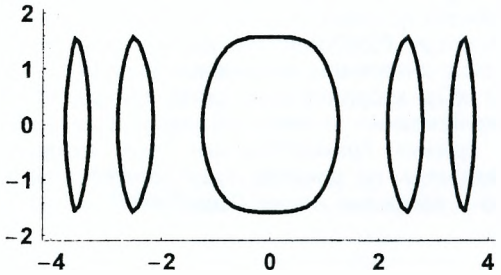
Теперь можно построить график функции $y = y(x)$, заданной неявно. Для этого достаточно записать соответствующее уравнение в качестве первого аргумента функции **ImplicitPlot**.

In[45]:=
```
ImplicitPlot[x^2 + 5 y^2 == 2,
    {x, -2, 2}, PlotStyle → Thickness[0.01]];
```



Поскольку уравнение $x^2 + 5y^2 = 2$ разрешимо относительно y, такой же график можно получить и с помощью стандартной функции **Plot**. Однако, функция **ImplicitPlot** позволяет изобразить кривую без каких-либо дополнительных вычислений и в случае более сложных уравнений, например,

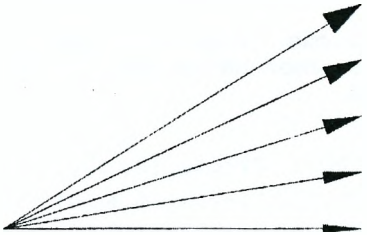
In[46]:=
```
ImplicitPlot[Cos[x^2] + Cos[y] == 1, {x, -4, 4},
    {y, -2, 2}, PlotStyle → Thickness[0.009]];
```



Ясно, что для получения такого же графика с помощью функции **Plot** потребуется гораздо больше усилий.

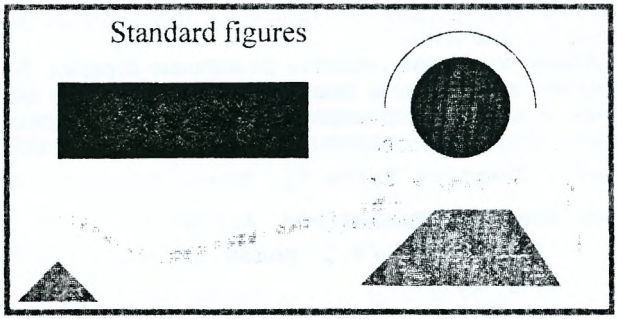
Модуль **Arrow** позволяет рисовать различные стрелки. Загрузка модуля **Colors** позволяет использовать вместо цветовой функции обычные английские названия. В качестве примера нарисуем набор красных стрелок различной формы, загрузив предварительно соответствующие модули.

```
In[52]:= Needs["Graphics`Arrow`"]; Needs["Graphics`Colors`"];

In[53]:= Show[Graphics[Table[{Red, Arrow[{0, 0}, {1, 0.2 i},
           HeadLength → 0.1, HeadWidth → 0.1 (i + 2)]},
         {i, 0, 4}]]];
```



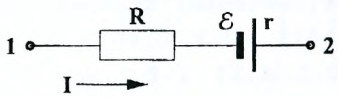
Кроме построения графиков функций, *Mathematica* позволяет генерировать рисунки, используя для этого набор простейших объектов, называемых примитивами (**Graphics Primitives**). К ним относятся прямая линия, ломаная, дуга окружности, треугольник, многоугольник, диск и т.д. Для получения списка примитивов и правил обращения к ним достаточно запустить программу помощи **Help Browser** и выбрать последовательно следующие закладки: **Built-in Functions** ↦ **Graphics and Sound** ↦ **Graphics Primitives** ↦ **Graphics**. В качестве примера изобразим в прямоугольной рамке несколько стандартных фигур. Соответствующая команда имеет вид:

```
In[54]:= Show[Graphics[ {{Cyan, Polygon[
            {{-0.08, -0.08}, {0, 0}, {0.08, -0.08}}]},
          {Cyan, Polygon[{{0.6, -0.05},
             {0.75, 0.1}, {0.9, 0.1}, {1, -0.05}}]},
          {Red, Disk[{0.8, 0.3}, 0.1]},
          {Red, Circle[{0.8, 0.3}, 0.15, {0, π}]},
          {Blue, Rectangle[{0, 0.2}, {0.5, 0.35}]},
          {Thickness[0.02], Yellow,
           Line[{{0, 0.1}, {0.2, 0}, {1, 0.2}, {1.05, 0}}]},
          {Magenta, Text[FontForm["Standard figures",
             {"Times", 16}], {0.3, 0.45}]} }],
        PlotRange → {{-0.1, 1.1}, {-0.1, 0.5}},
        AspectRatio → Automatic,
        Frame → True, FrameTicks → None,
        FrameStyle → {Green, Thickness[0.01]}];
```



Следующая команда позволяет получить изображение участка электрической цепи, на котором стрелкой указано направление тока и нанесены соответствующие обозначения.

```
Show[Graphics[ {
    {Thickness[0.007], Circle[{0, 0}, 0.007]},
    {Thickness[0.007], Circle[{0.6, 0}, 0.007]},
    Line[{{0, 0}, {0.15, 0}}],
    Line[{{0.31, 0}, {0.45, 0}}],
    Line[{{0.48, 0}, {0.6, 0}}],
    Line[{{0.15, -0.03}, {0.15, 0.03}, {0.31, 0.03},
                        {0.31, -0.03}, {0.15, -0.03}}],
    {Thickness[0.02],
     Line[{{0.45, -0.02}, {0.45, 0.02}}]},
    {Thickness[0.01], Line[
      {{0.48, -0.05}, {0.48, 0.05}}]},
    Arrow[{0.1, -0.08}, {0.25, -0.08}],
    Text["1", {-.04, 0}], Text["2", {.64, 0}],
    Text["R", {.23, 0.06}], Text["ℰ", {.42, 0.05}],
    Text["r", {.51, 0.05}], Text["I", {.08, -0.08}] }],
  AspectRatio → Automatic,
  DefaultFont → {"Times-Bold", 12},
  PlotRange → {{-0.1, 0.7}, {-0.15, 0.15}} ];
```



7. Списки и работа с ними

7.1. Простейшие операции со списками

Список является одной из достаточно общих и важных конструкций в системе *Mathematica*, которая позволяет объединить несколько различных объектов и рассматривать их как одно целое. Элементы списка заключаются в фигурные скобки, которые являются его характерным признаком, и разделяются запятыми. Определим, например, список *s1*.

```
In[55]:= s1 = {x, a + x, 5, Cos[y + x], Sqrt[17]}

Out[55]= {x, a + x, 5, Cos[x + y], Sqrt[17]}
```

Список можно умножить на число или символьную переменную. При этом каждый элемент списка умножается на это число или символ. Аналогично, добавляя к списку какую-либо величину, мы добавляем ее к каждому элементу списка. Например, умножая список *s1* на 7 и добавляя к нему *x*, получаем следующий список:

```
In[56]:= 7 s1 + x

Out[56]= {8 x, x + 7 (a + x), 35 + x, x + 7 Cos[x + y], 7 Sqrt[17] + x}
```

Списки, имеющие одинаковое количество элементов, можно складывать и умножать друг на друга. При этом получается список, элементами которого являются суммы и произведения соответствующих элементов исходных списков. Определим, например, список *s2*, затем умножим его на 3 и сложим со списком *s1*. Получаем:

```
In[57]:= s2 = {2, b, x, y - 2 x, 2 + x^2};
          s1 + 3 s2

Out[58]= {6 + x, a + 3 b + x, 5 + 3 x,
           3 (-2 x + y) + Cos[x + y], Sqrt[17] + 3 (2 + x^2)}
```

При умножении списков *s1* и *s2* получаем следующий список:

```
In[59]:= s1 * s2

Out[59]= {2 x, b (a + x), 5 x, (-2 x + y) Cos[x + y], Sqrt[17] (2 + x^2)}
```

Легко видеть, что элементы этого списка представляют собой произведения соответствующих элементов списков-сомножителей. Аналогично можно возводить список в степень, дифференцировать и интегрировать. Примеры:

```
In[60]:= s1^3

Out[60]= {x^3, (a + x)^3, 125, Cos[x + y]^3, 17 Sqrt[17]}

In[61]:= D[s2, x]

Out[61]= {0, 0, 1, -2, 2 x}

In[62]:= Integrate[s1, y]

Out[62]= {x y, (a + x) y, 5 y, Cos[y] Sin[x] + Cos[x] Sin[y], Sqrt[17] y}
```

7.2. Генерирование списков

В системе *Mathematica* имеется несколько функций, предназначенных для генерирования списков. Наиболее часто используется функция **Table**, которая позволяет создавать списки, элементы которых можно задать в виде некоторой формулы, содержащей параметр. Эта функция вызывается по крайней мере с двумя аргументами. Первым аргументом является выражение, определяющее обший вид элементов списка, а вторым – итератор, который аналогичен итератору у функций **Sum** и **Product** (см. таблицу на стр. 58). Напомним, что в простейшем случае итератор представляет собой натуральное число, заключенное в фигурные скобки. Это число определяет количество элементов в списке. Например, для создания списка из пяти элементов, которые являются действительными случайными числами из интервала (0, 1), достаточно выполнить следующую команду:

```
In[63]:= Table[Random[], {5}]

Out[63]= {0.459649, 0.974904, 0.961079, 0.970511, 0.691982}
```

Если значения элементов списка зависят от параметра *par*, который изменяется от 1 до n_{max} с шагом 1, то итератор должен иметь вид: {par, n_{max}}. Пример:

```
In[64]:= Table[x/(x^2 + 1), {x, 3}]
```

$$Out[64]= \left\{\frac{1}{2}, \frac{2}{5}, \frac{3}{10}\right\}$$

В случае итератора вида {*par*, n_{min}, n_{max}} параметр *par* изменяется от n_{min} до n_{max} с шагом 1.

```
In[65]:= Table[Sin[k π], {k, 0.3, 3.1}]

Out[65]= {0.809017, -0.809017, 0.809017}
```

Наиболее общим является итератор вида {par, n_{min}, n_{max}, step}, в котором задаются не только начальное n_{min} и конечное n_{max} значения параметра *par*, на и шаг его изменения *step*.

```
In[66]:= Table[Exp[x], {x, -1., 0.7, 0.4}]

Out[66]= {0.367879, 0.548812, 0.818731, 1.2214, 1.82212}
```

Если элементы списка сами являются списками, то получается многомерный список. Такой список генерируется в том случае, когда функция **Table** содержит несколько итераторов, которые являются ее вторым и последующими аргументами. При этом выражение, определяющее общий вид элементов списка, может зависеть от двух и более параметров. Определим, например, двумерный список *s3*:

```
In[67]:= s3 = Table[x/y, {x, 5}, {y, 7, 9}]
```

$$Out[67]= \left\{\left\{\frac{1}{7}, \frac{1}{8}, \frac{1}{9}\right\}, \left\{\frac{2}{7}, \frac{1}{4}, \frac{2}{9}\right\}, \left\{\frac{3}{7}, \frac{3}{8}, \frac{1}{3}\right\}, \left\{\frac{4}{7}, \frac{1}{2}, \frac{4}{9}\right\}, \left\{\frac{5}{7}, \frac{5}{8}, \frac{5}{9}\right\}\right\}$$

Как видим, при каждом значении параметра *x*, определяемого первым итератором, который является вторым аргументом функции **Table**, генерируется список, число элементов которого определяется вторым итератором. Таким образом, первый итератор определяет размер внешнего списка, а второй – размер каждого из вложенных списков. Количество элементов в многомерных списках можно определить с помощью функции **Dimensions**. Так, для полученного выше списка *s3* имеем:

In[68]:= `Dimensions[s3]`

Out[68]= `{5, 3}`

Полученный результат означает, что *s3* является двумерным списком из пяти элементов, причем каждый его элемент представляет собой список из трех элементов. Напомним, что для определения размерности внешнего списка можно использовать функцию **Length**.

In[69]:= `Length[s3]`

Out[69]= `5`

Для генерирования списка, элементами которого являются числа из заданного интервала, используется функция **Range**. Если ее аргументом является положительное действительное число *k*, то она генерирует список натуральных чисел, максимальное из которых равняется целой части *k*. Пример:

In[70]:= `Range[5.7]`

Out[70]= `{1, 2, 3, 4, 5}`

Задавая два аргумента k_1 и k_2, получаем список, первый элемент которого равен k_1, а каждый последующий увеличивается на единицу до тех пор, пока не получится число, превышающее k_2. Таким образом, k_2 определяет верхнюю границу списка. Пример:

In[71]:= `Range[-2.1, 3.4]`

Out[71]= `{-2.1, -1.1, -0.1, 0.9, 1.9, 2.9}`

Добавляя у функции **Range** третий аргумент, мы указываем, с каким шагом должны изменяться числа в списке.

In[72]:= `Range[2.1, -3.4, -0.6]`

Out[72]= `{2.1, 1.5, 0.9, 0.3, -0.3, -0.9, -1.5, -2.1, -2.7, -3.3}`

Последний пример показывает, что шаг может быть и отрицательным. Отметим также, что элементы списка определяются с такой же точностью, что и аргументы функции **Range**. Если все аргументы являются точными числами, то и получаемые элементы списка будут заданы точно. Пример:

In[73]:= `Range[Sqrt[2], 4, 1/2]`

Out[73]= $\left\{\sqrt{2},\ \frac{1}{2}+\sqrt{2},\ 1+\sqrt{2},\ \frac{3}{2}+\sqrt{2},\ 2+\sqrt{2},\ \frac{5}{2}+\sqrt{2}\right\}$

7.3. Выделение и поиск элементов списка

Положение каждого элемента в списке фиксировано и определяется его номером, который можно использовать для выделения элемента. Отметим, что для нумерации элементов используются натуральные числа, причем первый элемент всегда имеет номер 1. Чтобы выделить первый элемент определенного ранее списка *s1*, например, достаточно после его имени указать соответствующий номер в двойных квадратных скобках.

In[74]:= `s1[[1]]`

Out[74]= `x`

В качестве альтернативы для выделения первого элемента списка можно использовать команду **First**, например,

In[75]:= `First[s1]`

Out[75]= `x`

Аналогично, последний элемент списка выделяется с помощью команды **Last**.

In[76]:= `Last[s1]`

Out[76]= $\sqrt{17}$

Добавление знака "–" у номера элемента означает, что при выделении элемента отсчет производится в обратном порядке, начиная с последнего элемента списка. Так, предпоследний элемент списка *s1* равен:

In[77]:= `s1[[-2]]`

Out[77]= `Cos[x + y]`

Если указанный номер превышает число элементов списка, то выдается сообщение, что элемент с таким номером не существует.

In[78]:= `s2[[7]]`

```
Part::partw :
  Part 7 of {2, b, x, -2 x + y, 2 + x^2} does not exist. More
```

Out[78]= $\{2, b, x, -2x + y, 2 + x^2\}[[7]]$

Элементы каждого из вложенных списков нумеруются отдельно. Поэтому положение любого элемента определяется набором чисел, каждое из которых является номером элемента в соответствующем списке. Так, например, второй элемент списка *s3* сам является список вида:

In[79]:= `s3[[2]]`

Out[79]= $\left\{\frac{2}{7}, \frac{1}{4}, \frac{2}{9}\right\}$

Для выделения третьего, например, элемента в этом списке достаточно в двойных квадратных скобках указать после двойки число 3, разделив два числа запятыми.

In[80]:= `s3[[2, 3]]`

Out[80]= $\frac{2}{9}$

Для определения положения элемента в списке используется функция **Position**. При обращении к ней в качестве первого аргумента нужно указать имя списка, а на втором месте поместить сам элемент. Пример:

```
In[81]:= Position[s3, 2/7]

Out[81]= {{2, 1}}
```

В результате получаем список чисел {2,1}, который показывает, что число 2/7 является первым элементом второго из вложенных списков. Следует отметить, что список {2,1} заключен в дополнительные фигурные скобки. Это связано с тем, что в исходном списке может быть несколько одинаковых элементов, положение каждого из которых определяется своим набором чисел, заключенных в фигурные скобки. В таком случае функция **Position** выдает двумерный список, элементы которого определяют все положения заданного элемента. Пример:

```
In[82]:= s4 = {{x, y, z}, {y, z^2, 5 z}, {z, x^2, y}};
         Position[s4, y]

Out[83]= {{1, 2}, {2, 1}, {3, 3}}
```

Функция **Count** позволяет определить, сколько заданных элементов содержится в списке. Попробуем, например, подсчитать, сколько элементов *y* имеется в списке *s4*.

```
In[84]:= Count[s4, y]

Out[84]= 0
```

Хотя в списке *s4* содержится три элемента *y*, функция **Count** выдает ноль. Причиной этого является то обстоятельство, что *s4* является двумерным списком, элементы которого сами являются списками. При этом символы *y* являются элементами вложенных списков, т.е. находятся как бы на втором уровне списка *s4*. Чтобы произвести подсчет количества элементов *y* во вложенных списках, нужно добавить число 2 в качестве третьего аргумента функции **Count**. Это число означает, что подсчет производится на втором уровне исходного списка. Соответственно, положение каждого элемента *y* определяется списком из двух чисел, что и показала ранее функция **Position**.

```
In[85]:= Count[s4, y, 2]

Out[85]= 3
```

В результате получили число 3, что и ожидалось.

Проверить, имеется ли заданный элемент в списке можно и с помощью функции **MemberQ**. Если при обращении к ней указаны два аргумента, то проверка производится только на первом уровне. Пример:

```
In[86]:= MemberQ[s4, x]

Out[86]= False
```

Поскольку на первом уровне списка *s4* символ *x* отсутствует, получаем отрицательный ответ. Указав в качестве третьего аргумента номер уровня, на котором следует производить проверку, получаем положительный ответ.

In[87]:= **MemberQ[s4, x, 2]**

Out[87]= **True**

Итак, при наличии двух аргументов функции **MemberQ** и **Count** выполняют соответственно поиск и подсчет элементов только на первом уровне заданного списка. В отличие от них, функция **FreeQ** производит поиск элемента на всех уровнях и выдает положительный или отрицательный ответ в зависимости от того, отсутствует или присутствует соответственно данный элемент в списке. Так, поскольку элемент *y* имеется в списке *s4*, функция **FreeQ** выдает отрицательный ответ.

In[88]:= **FreeQ[s4, y]**

Out[88]= **False**

Добавляя в качестве третьего аргумента число 1, например, мы ограничиваем поиск только первым уровнем и получаем положительный ответ, поскольку на этом уровне элемент *y* действительно отсутствует.

In[89]:= **FreeQ[s4, y, 1]**

Out[89]= **True**

7.4. Преобразование списков

Под *преобразованиями списков* мы будем понимать любые операции с одним или несколькими списками, приводящие к появлению новых списков. Примерами таких операций являются удаление одного или нескольких элементов из списка, добавление к списку новых элементов, замена некоторых элементов списка другими элементами, изменение порядка следования элементов в списке, объединение списков и т.п.. Далее мы рассмотрим основные функции, используемые для преобразования списков.

Удалить некоторые элементы из списка можно с помощью функций **Delete** и **Drop**. Чтобы продемонстрировать их работу, генерируем список из 12 натуральных чисел *s5*.

In[90]:= **s5 = Range[1, 12]**

Out[90]= **{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}**

Применим к этому списку обе функции, указав у них одинаковые аргументы следующего вида:

In[91]:= **Delete[s5, 5]**

Out[91]= **{1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12}**

In[92]:= **Drop[s5, 5]**

Out[92]= **{6, 7, 8, 9, 10, 11, 12}**

Как видим, второй аргумент в виде натурального числа у функции **Delete** обозначает номер элемента, который должен быть удален. Такой же аргумент у функции **Drop** определяет количество удаляемых элементов, причем отсчет производится от первого элемента списка. Если же это число заключено в фигурные скобки, то обе функции удаляют только один элемент с заданным номером. Напомним, что отрицательное число обозначает, что отсчет производится с конца списка.

```
In[93]:= Delete[s5, {2}]

Out[93]= {1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

In[94]:= Drop[s5, {-2}]

Out[94]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12}
```

Второй аргумент вида $\{n_1, n_2\}$ у функции **Drop** приводит к удалению с n_1-го элемента по n_2-ой.

```
In[95]:= Drop[s5, {3, 7}]

Out[95]= {1, 2, 8, 9, 10, 11, 12}
```

Если же этот аргумент представляет собой список трех чисел вида $\{n_1, n_2, step\}$, то номера удаляемых элементов будут начинаться с n_1 и увеличиваться с шагом *step* до тех пор, пока не будет достигнут верхний предел n_2. Так, например, применение к списку *s5* функции **Drop** со вторым аргументом вида {2,10,3} приведет к удалению второго, пятого и восьмого элементов. В результате получится следующий список:

```
In[96]:= Drop[s5, {2, 10, 3}]

Out[96]= {1, 3, 4, 6, 7, 9, 10, 11, 12}
```

В отличие от **Drop**, у функции **Delete** нельзя указать некоторый интервал номеров элементов, которые должны быть удалены. Ее второй аргумент должен содержать список номеров удаляемых элементов. Для удаления в списке *s5* первого, третьего и десятого элементов, например, используется следующая команда:

```
In[97]:= Delete[s5, {{1}, {3}, {10}}]

Out[97]= {2, 4, 5, 6, 7, 8, 9, 11, 12}
```

Обратите внимание на дополнительные фигурные скобки, в которые заключены номера удаляемых элементов. Их появление объясняется возможностью удалять элементы во внутренних списках, когда положение элементов определяется не одним, а несколькими числами. Чтобы продемонстрировать такую возможность, создадим двумерный список *s6*.

```
In[98]:= s6 = Table[Range[3 k + 1, 3 k + 3], {k, 0, 4}]

Out[98]= {{1, 2, 3}, {4, 5, 6},
          {7, 8, 9}, {10, 11, 12}, {13, 14, 15}}
```

Применяя функцию **Delete**, удаляем второй элемент в первом вложенном списке, третий элемент в четвертом списке, а также последний элемент исходного списка *s6*.

```
In[99]:= Delete[s6, {{1, 2}, {4, 3}, {-1}}]

Out[99]= {{1, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11}}
```

Заметим, что для удаления первого и последнего элементов списка можно также использовать функции **Rest** и **Most** соответственно. Примеры:

```
In[100]:= Rest[s6]

Out[100]= {{4, 5, 6}, {7, 8, 9}, {10, 11, 12}, {13, 14, 15}}
```

```
In[101]:= s7 = Most[s6]

Out[101]= {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12}}
```

Если функции **Delete**, **Drop**, **Rest**, **Most** создают новый список, удаляя часть элементов в исходном списке, то функция **Take** поступает наоборот, отбирая указанные элементы исходного списка и создавая из них новый список. Ее второй аргумент, определяющий отбираемые элементы, может принимать такие же значения, как и соответствующий аргумент функции **Drop**. Так, например, для выделения в списке *s6* второго, третьего и четвертого элементов можно использовать следующую команду:

```
In[102]:= Take[s6, {2, 4}]

Out[102]= {{4, 5, 6}, {7, 8, 9}, {10, 11, 12}}
```

Аналогичную операцию выполняет функция **Extract**. Однако, в отличие от функции **Take**, ее второй аргумент должен содержать конкретный список тех элементов, из которых будет состоять новый список. Пример:

```
In[103]:= Extract[s7, {{1}, {2, 3}, {4}}]

Out[103]= {{1, 2, 3}, 6, {10, 11, 12}}
```

Чтобы вставить дополнительный элемент в начало списка, используем функцию **Prepend**. Формат обращения к ней понятен из следующего примера:

```
In[104]:= Prepend[s7, {2, 11, 7}]

Out[104]= {{2, 11, 7}, {1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12}}
```

Для добавления элемента в конец списка используется функция **Append**. Пример:

```
In[105]:= Append[s7, {8, 1, 13}]

Out[105]= {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12}, {8, 1, 13}}
```

Вставить дополнительный элемент в любое место списка можно с помощью функции **Insert**. Отметим, что ее первым аргументом является исходный список, вторым – вставляемый элемент, а третьим – список, в котором указаны одно или несколько положений, в которых должен находиться заданный элемент. В следующем примере число 77 вставляется в трех местах списка *s7*.

```
In[106]:= Insert[s7, 77, {{1, 3}, {3, 2}, {4}}]

Out[106]= {{1, 2, 77, 3}, {4, 5, 6}, {7, 77, 8, 9}, 77, {10, 11, 12}}
```

Для замены некоторых элементов списка на заданный элемент служит функция **ReplacePart**. В следующем примере с ее помощью производится замена двух элементов, находящихся на разных уровнях списка *s8*, символом *Z*.

```
In[107]:= s8 = {{1, 2, 3}, {a, b, c}, {7, x, 9}};
          ReplacePart[s8, Z, {{2, 2}, {3}}]

Out[108]= {{1, 2, 3}, {a, Z, c}, Z}
```

Второй аргумент функции **ReplacePart** может быть и списком элементов, которыми будут заменяться элементы исходного списка. В этом случае функция **ReplacePart** вызывается с четырьмя аргументами, причем третий

аргумент является списком, определяющим положения заменяемых элементов в исходном списке, а четвертый указывает соответствующие положения элементов в списке замен. Пример:

```
In[109]:= ReplacePart[s8, {a, b, {c, d}},
            {{1, 1}, {3, 2}}, {{2}, {3, 1}}]

Out[109]= {{b, 2, 3}, {a, b, c}, {7, c, 9}}
```

Заметим, что создать новый список из элеметнов определенного выше списка *s8*, например, можно и непосредственно, указав после его имени в двойных квадратных скобках список соответствующих номеров его элементов. При этом можно изменить не только порядок следования, но и число элементов исходного списка. Следующий пример показывает, как из списка *s8* можно получить новый список из пяти элементов.

```
In[110]:= s8[[ {3, 1, 2, 1, 3} ]]

Out[110]= {{7, x, 9}, {1, 2, 3}, {a, b, c}, {1, 2, 3}, {7, x, 9}}
```

Поскольку элементы списка *s8* сами являются списками, получили двумерный список, первым элементом которого является третий элемент списка *s8*, вторым – первый элемент *s8* и т.д.

В системе *Mathematica* имеется несколько функций, которые изменяют порядок следования элементов в списке. Так, функция **Sort**, вызываемая с одним аргументом, размещает элементы списка в "так называемом" стандартном порядке. Чтобы продемонстрировать ее работу, определим список *s9*, состоящий из различных элементов.

```
In[111]:= s9 = {a, 3.4, D, {3, 4}, -2, A, 1 + Sqrt[3], d, Cos[x]};
```

Применяя к нему функцию **Sort**, получаем:

```
In[112]:= Sort[s9]

Out[112]= {-2, 3.4, 1 + Sqrt[3], a, A, d, D, Cos[x], {3, 4}}
```

Первыми в полученном списке идут числа в порядке возрастания. Затем в алфавитном порядке следуют буквы, причем заглавные буквы располагаются после строчных, затем более сложные объекты. Проверить, соответствует ли расположение элементов в списке стандартному порядку, можно с помощью функции **OrderedQ**. Применяя ее к полученному выше списку, получаем положительный ответ.

```
In[113]:= OrderedQ[%]

Out[113]= True
```

Соответственно исходный список *s9* не является упорядоченным.

```
In[114]:= OrderedQ[s9]

Out[114]= False
```

В качестве второго аргумента функции **Sort** можно указать функцию или правило, которые будут управлять процессом сортировки. Так, например, функция **GreaterEqual** указывает, что для любых соседних элементов списка x и y должно выполняться неравенство: $x \geq y$. В результате числа располагаются в порядке убывания.

```
In[115]:= Sort[{2, 3, 6, 3, 1, 12}, GreaterEqual]
```

Out[115]= {12, 6, 3, 3, 2, 1}

Функция **Reverse** позволяет записать элементы списка в обратном порядке.

In[116]:= Reverse[s9]

Out[116]= $\{\text{Cos}[x], d, 1+\sqrt{3}, A, -2, \{3, 4\}, D, 3.4, a\}$

Функции **RotateLeft** и **RotateRight** производят циклическую перестановку элементов списка влево и вправо соответственно. Следующие два примера демонстрируют их работу.

In[117]:= RotateLeft[s9]

Out[117]= $\{3.4, D, \{3, 4\}, -2, A, 1+\sqrt{3}, d, \text{Cos}[x], a\}$

In[118]:= RotateRight[s9, 4]

Out[118]= $\{A, 1+\sqrt{3}, d, \text{Cos}[x], a, 3.4, D, \{3, 4\}, -2\}$

С помощью функций **Join** и **Union** два или несколько списков можно объединить в один. При этом функция **Join** просто удаляет граничные фигурные скобки у соседних списков, создавая один список и не нарушая порядка следования элементов.

In[119]:= Join[{a, b, c, d}, {e, a, b, {x, y}}, {x, e, h}]

Out[119]= {a, b, c, d, e, a, b, {x, y}, x, e, h}

Как видим, в полученном списке сохранены все одинаковые элементы, имеющиеся в объединяемых списках. В отличие от **Join**, функция **Union** оставляет только один из совпадающих элементов, а к результирующему списку применяет функцию **Sort**. В результате получается упорядоченный список.

In[120]:= Union[{a, b, c, d}, {e, a, b, {x, y}}, {x, e, h}]

Out[120]= {a, b, c, d, e, h, x, {x, y}}

К аналогичному результату приводит применение функции **Union** к одному списку. Пример:

In[121]:= Union[{a, 3, b, 4, 5, c, a, c, d, 1, 3, 4}]

Out[121]= {1, 3, 4, 5, a, b, c, d}

Функция **Intersection[list1, list2, list3, ...]** создает упорядоченный список элементов, имеющихся в каждом из списков *list1*, *list2*, *list3*, ... Пример:

In[122]:= Intersection[{a, f, g, h, 5}, {3, x, f, h, 5, y}]

Out[122]= {5, f, h}

Как видно из следующего примера, функция **Complement[list, list1, list2, ...]** также создает упорядоченный список. Однако она отбирает среди элементов списка *list* только такие, которые отсутствуют в остальных списках.

In[123]:= Complement[{11, 2, b, 24, y, a, s},
{b, c, d}, {1, 23, 24}]

Out[123]= {2, 11, a, s, y}

В системе *Mathematica* имеется несколько функций, позволяющих изменять структуру списка. Отметим из них только две: **Partition** и **Flatten**. Чтобы продемонстрировать их работу, определяем список *s10*.

```
In[124]:= s10 = Range[14]

Out[124]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}
```

Функция **Partition** позволяет, например, получить из *s10* двумерный список, разбивая его на части.

```
In[125]:= Partition[s10, 4]

Out[125]= {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}}
```

Число 4, которое является вторым аргументом у **Partition**, указывает на то, что элементы списка *s10* должны быть последовательно разбиты на отдельные списки, каждый из которых содержит четыре элемента. Поскольку остающихся двух чисел 13 и 14 недостаточно для создания списка из четырех элементов, они отбрасываются. Добавляя число 3 в качестве третьего аргумента, получаем:

```
In[126]:= Partition[s10, 4, 3]

Out[126]= {{1, 2, 3, 4}, {4, 5, 6, 7},
           {7, 8, 9, 10}, {10, 11, 12, 13}}
```

Легко видеть, что разбиение списка *s10* производится следующим образом. Сначала выделяются первые четыре элемента *s10*. Затем происходит смещение на три элемента вправо и опять выделяются четыре последовательных элемента *s10*, и т.д.. В результате лишним оказывается только один элемент *s10* – число 14. Заметим, что функцию **Partition** можно применять не только к одномерным спискам, но и к более сложным объектам.

Функция **Flatten** выполняет обратную по отношению к **Partition** операцию. Применяя ее к предыдущему списку, получаем:

```
In[127]:= Flatten[%]

Out[127]= {1, 2, 3, 4, 4, 5, 6, 7, 7, 8, 9, 10, 10, 11, 12, 13}
```

Как видим, функция **Flatten** удаляет все внутренние фигурные скобки, превращая двумерный список в одномерный. В случае списков более сложной структуры получается аналогичный результат. Пример:

```
In[128]:= s11 = {{1, 2, 3}, 4, {x, {a, {b}}, {c, 4, 7}}};
          Flatten[s11]

Out[129]= {1, 2, 3, 4, x, a, b, c, 4, 7}
```

Добавляя натуральное число *n* в качестве второго аргумента функции **Flatten[list, n]**, можно ограничить число уровней, на которых производится удаление внутренних фигурных скобок. Так, в случае $n = 1$ удаление скобок производится только на первом уровне.

```
In[130]:= Flatten[s11, 1]

Out[130]= {1, 2, 3, 4, x, {a, {b}}, {c, 4, 7}}
```

При $n = 2$ скобки удаляются на первом и втором уровнях.

```
In[131]:= Flatten[s11, 2]
```

Out[131]= `{1, 2, 3, 4, x, a, {b}, c, 4, 7}`

Отметим также, что с помощью функции **FlattenAt** операцию удаления фигурных скобок можно произвести не только на целом уровне списка, но и у любого из внутренних списков. Для этого достаточно указать положение соответствующих элементов в качестве второго аргумента этой функции. Пример:

In[132]:= `FlattenAt[s11, {{1}, {3, 2}}]`

Out[132]= `{1, 2, 3, 4, {x, a, {b}, {c, 4, 7}}}`

7.5. Операции с векторами и матрицами

Векторы и матрицы в системе *Mathematica* представляются в виде списков и вложенных списков соответственно. Поэтому к ним применимы все рассмотренные выше операции. Кроме того, *Mathematica* позволяет выполнять с векторами и матрицами все операции линейной алгебры. В этом параграфе мы рассмотрим некоторые из них.

Зададим, например, два вектора: *r1* и *r2*.

In[133]:=
```
r1 = {x1, y1, z1};
r2 = {x2, y2, z2};
```

Такие векторы определяют, например, положения двух частиц в трехмерном пространстве, а их компоненты соответствуют прямоугольным декартовым координатам частиц.

Скалярное произведение двух векторов определяется как сумма произведений их соответствующих координат. Для вычисления скалярного произведения используется функция **Dot**.

In[135]:= `Dot[r1, r2]`

Out[135]= $x_1 x_2 + y_1 y_2 + z_1 z_2$

Ее эквивалентное обозначение имеет вид:

In[136]:= `r1 . r2`

Out[136]= $x_1 x_2 + y_1 y_2 + z_1 z_2$

Напомним, что обычное умножение списков (точка в выражении *r1.r2* отсутствует или заменена звездочкой) приводит к появлению списка, элементы которого являются произведениями соответствующих элементов перемножаемых списков. Действительно,

In[137]:= `r1 * r2`

Out[137]= $\{x_1 x_2, y_1 y_2, z_1 z_2\}$

Длина вектора определяется как корень квадратный из суммы квадратов модулей его координат. Для ее вычисления служит функция **Norm**. Пример:

In[138]:= `Norm[r1]`

Out[138]= $\sqrt{\text{Abs}[x_1]^2 + \text{Abs}[y_1]^2 + \text{Abs}[z_1]^2}$

Очевидно, определяемая таким образом длина будет неотрицательным действительным числом и в том случае, когда координаты вектора являются комплексными числами. При этом действие функции **Norm** эквивалентно вычислению квадратного корня из скалярного произведения вектора и

комплексно сопряженного ему вектора (для комплексного сопряжения используется функция **Conjugate**).

In[139]:=
```
Sqrt[r1.Conjugate[r1]]
```

Out[139]=
$$\sqrt{\text{Abs}[x_1]^2 + \text{Abs}[y_1]^2 + \text{Abs}[z_1]^2}$$

Если координаты вектора являются действительными числами, то длина вектора равняется квадратному корню из суммы квадратов его координат. Действительно, добавляя у функции **Simplify** соответствующее условие в качестве второго аргумента, получаем:

In[140]:=
```
Simplify[Norm[r1], r1 ∈ Reals]
```

Out[140]=
$$\sqrt{x_1^2 + y_1^2 + z_1^2}$$

Для вычисления векторного произведения используется функция **Cross**. В результате получается вектор, ортогональный каждому из векторов-сомножителей, причем его направление зависит от порядка следования сомножителей. Пример:

In[141]:=
```
Cross[r1, r2]
```

Out[141]=
$$\{-y_2\, z_1 + y_1\, z_2,\ x_2\, z_1 - x_1\, z_2,\ -x_2\, y_1 + x_1\, y_2\}$$

In[142]:=
```
r1 . Cross[r1, r2] // Simplify
```

Out[142]= 0

Следует отметить, что функции **Dot**, **Norm**, **Cross** могут оперировать не только с векторами, заданными в трехмерном пространстве, но и с векторами произвольной размерности.

Определим теперь список *mat1*, элементы которого сами являются списками, содержащими одинаковое число элементов, например,

In[143]:=
```
mat1 =
    {{18, 2, -7, -5}, {50, -6, -21, -11}, {6, 9, -3, -5}};
```

Такой двумерный список представляет собой матрицу, которая содержит три строки и четыре столбца. Чтобы записать ее в привычном виде, применим к списку *mat1* функцию **MatrixForm**. Отметим, что эта функция не создает матрицу, она лишь определяет формат представления списка.

In[144]:=
```
MatrixForm[mat1]
```

Out[144]//MatrixForm=
$$\begin{pmatrix} 18 & 2 & -7 & -5 \\ 50 & -6 & -21 & -11 \\ 6 & 9 & -3 & -5 \end{pmatrix}$$

Теперь легко видеть, что вложенные списки в *mat1* определяют строки матрицы. Соответственно транспонированная матрица будет представлять собой список, элементами которого являются столбцы матрицы *mat1*. Чтобы убедиться в этом, применим к списку *mat1* функцию **Transpose**. В результате получаем:

In[145]:=
```
Transpose[mat1]
```

Out[145]=
```
{{18, 50, 6}, {2, -6, 9}, {-7, -21, -3}, {-5, -11, -5}}
```

Для перемножения матриц используется функция **Dot**. Напомним, что элемент $(A.B)_{ij}$ результирующей матрицы вычисляется как скалярное произведение i-ой строки матрицы A и j-ого столбца матрицы B. Так, умножая вектор *r1* на матрицу *mat1*, получаем следующий вектор:

In[146]:= **r1.mat1**

Out[146]= $\{18\,x_1 + 50\,y_1 + 6\,z_1,\ 2\,x_1 - 6\,y_1 + 9\,z_1,$
$-7\,x_1 - 21\,y_1 - 3\,z_1,\ -5\,x_1 - 11\,y_1 - 5\,z_1\}$

При умножении матриц следует помнить, что число элементов в строке первой матрицы должно равняться числу элементов в столбце второй. Если это условие не выполняется, произведение матриц не может быть найдено. Именно поэтому нельзя умножить матрицу *mat1* на вектор *r1*. Действительно,

In[147]:= **mat1 . r1**

```
Dot::dotsh :
  Tensors {{18, 2, -7, -5}, {50, -6, -21, -11}, {6, 9, -3, -5}}
    and {x1, y1, z1} have incompatible shapes. More...
```

Out[147]= {{18, 2, -7, -5}, {50, -6, -21, -11}, {6, 9, -3, -5}}.
{x_1, y_1, z_1}

Аналогичное сообщение об ошибке будет выдано при вычислении произведения *mat1.mat1*. Однако можно умножить *mat1* на транспонированную матрицу *mat1*. При этом получаем следующую матрицу третьего порядка:

In[148]:= **mat1 . Transpose[mat1]**

Out[148]= {{402, 1090, 172}, {1090, 3098, 364}, {172, 364, 151}}

Если у транспонированной матрицы *mat1* удалить вторую строку, например, то получится квадратная матрица третьего порядка. Присвоим этой матрице имя *mat2*.

In[149]:= **mat2 = Delete[Transpose[mat1], 2]**

Out[149]= {{18, 50, 6}, {-7, -21, -3}, {-5, -11, -5}}

Определитель квадратной матрицы можно вычислить с помощью функции **Det**. Так, определитель матрицы *mat2* равен:

In[150]:= **Det[mat2]**

Out[150]= 128

Поскольку определитель не равен нулю, у матрицы *mat2* существует обратная матрица. Чтобы найти ее, используем функцию **Inverse**.

In[151]:= **mat3 = Inverse[mat2]**

Out[151]= $\left\{\left\{\frac{9}{16}, \frac{23}{16}, -\frac{3}{16}\right\},\right.$
$\left\{-\frac{5}{32}, -\frac{15}{32}, \frac{3}{32}\right\}, \left\{-\frac{7}{32}, -\frac{13}{32}, -\frac{7}{32}\right\}\}$

Легко убедиться в том, что при перемножении матриц *mat2* и *mat3* получается единичная матрица.

In[152]:= **mat2 . mat3**

Out[152]= `{{1, 0, 0}, {0, 1, 0}, {0, 0, 1}}`

Для такой матрицы в системе *Mathematica* имеется специальное обозначение: **IdentityMatrix[n]**, где число *n* в квадратных скобках указывает порядок матрицы. Если же вместо единиц на некоторых местах главной диагонали квадратной матрицы располагаются другие элементы, то получается диагональная матрица, для которой можно использовать следующее обозначение: **DiagonalMatrix**. При этом в качестве аргумента этой функции следует указать соответствующий список элементов, например,

In[153]:= `DiagonalMatrix[{1, x, 3, a}]`

Out[153]= `{{1, 0, 0, 0}, {0, x, 0, 0}, {0, 0, 3, 0}, {0, 0, 0, a}}`

При анализе многих физических систем часто возникает необходимость найти такие значения параметра λ, при которых линейная система уравнений

$$\boldsymbol{A}\,\boldsymbol{x} = \lambda\,\boldsymbol{x},$$

где *A* – некоторая квадратная матрица, имеет отличные от нуля решения *x*. Такие значения λ и векторы *x* называются *собственными значениями* и *собственными векторами* матрицы *A* соответственно. Для матрицы *mat2*, например, задачу по определению собственных значений можно записать в виде следующей системы:

In[154]:= `ColumnForm[ Thread[ mat2.r1 == λ r1 ] ]`

Out[154]=
$$\begin{aligned} 18\,x_1 + 50\,y_1 + 6\,z_1 &= \lambda\,x_1 \\ -7\,x_1 - 21\,y_1 - 3\,z_1 &= \lambda\,y_1 \\ -5\,x_1 - 11\,y_1 - 5\,z_1 &= \lambda\,z_1 \end{aligned}$$

Поскольку записанная система уравнений является однородной, для существования ненулевых решений необходимо, чтобы ее определитель был равен нулю, т.е. $det\,(mat2 - \lambda\,I) = 0$, где *I* – единичная матрица. В результате получаем следующее уравнение для определения собственных значений λ :

In[155]:= `eq1 = Det[mat2 - λ IdentityMatrix[3] ] == 0`

Out[155]= $128 + 16\,\lambda - 8\,\lambda^2 - \lambda^3 = 0$

Это уравнение называется *характеристическим уравнением*, а многочлен, стоящий в левой его части – *характеристическим полиномом*. Следует отметить, что в системе *Mathematica* имеется специальныя функция **CharacteristicPolynomial** для вычисления характеристических полиномов. С ее помощью легко находим характеристический полином для матрицы *mat2*:

In[156]:= `pol1 = CharacteristicPolynomial[mat2, λ]`

Out[156]= $128 + 16\,\lambda - 8\,\lambda^2 - \lambda^3$

Легко видеть, что *pol1* совпадает с выражением в левой части уравнения *eq1*. Решая характеристическое уравнение, находим собственные значения матрицы *mat2*.

In[157]:= `Solve[pol1 == 0, λ]`

Out[157]= `{{λ → -8}, {λ → -4}, {λ → 4}}`

Заметим, что для вычисления собственных значений матрицы в системе *Mathematica* имеется спициальная функция **Eigenvalues**. Так, применяя ее к матрице *mat2*, получаем:

In[158]:= **Eigenvalues[mat2]**

Out[158]= **{-8, -4, 4}**

Как видим, функция **Eigenvalues** выдает список собственных значений матрицы. Каждому собственному значению λ соответствует вектор x, который является решением системы $A\,x = \lambda\,x$. Полный набор собственных векторов можно получить с помощью функции **Eigenvectors**. В случае матрицы *mat2* соответствующие собственные векторы имеют вид:

In[159]:= **vectors = Eigenvectors[mat2]**

Out[159]= **{{-6, 3, 1}, {2, -1, 1}, {-4, 1, 1}}**

Отметим также функцию **Eigensystem**, которая полностью решает задачу по определению собственных значений и собственных векторов матрицы. Применяя ее к матрице *mat2*, получаем следующий список:

In[160]:= **Eigensystem[mat2]**

Out[160]= **{{-8, -4, 4}, {{-6, 3, 1}, {2, -1, 1}, {-4, 1, 1}}}**

Первым элементом списка является список собственных значений матрицы, а вторым – список соответствующих собственных векторов.

8. Элементы программирования

8.1. Выражение и его структура

Любой язык программирования, в том числе и *Mathematica*, содержит набор объектов и правил работы с ними, позволяющих записать программу в понятном компьютеру виде. В системе *Mathematica* имеется много различных объектов: переменные, функции, математические формулы, списки, графики и т.д.. Все они рассматриваются системой как *выражения*. При обработке выражения *Mathematica* сначала преобразует его к некоторой стандартной форме, называемой *полной формой* выражения, затем выполняет все необходимые операции и, наконец, выдает результат в привычном для нас виде. Именно с полной формой выражений оперирует вычислительное ядро системы *Mathematica*. Поэтому рассмотрим более подробно, что она собой представляет.

Структура полной формы выражения напоминает математическую функцию, т.е. имеет вид: *f*[*x*, *y*, ...]. При этом имя функции *f* называется *заголовком* (**Head**) выражения и определяет тип объекта, имя функции или команду, которая должна быть выполнена с аргументами *x*, *y*, ... Если один или несколько аргументов также имеет вид *g*[*z*, *h*], то появляется двухуровневое выражение, например, *f*[*x*, *g*[*y*, *z*]] и т.д.. Естественно, такая структура выражения предполагает, что имеются некоторые простейшие объекты, из которых строятся более сложные выражения.

Простейшими или *атомарными выражениями* являются числа, символы и строки. Напомним, что в системе *Mathematica* используются четыре типа чисел: целые (**Integer**), рациональные (**Rational**), действительные (**Real**) и

комплексные (**Complex**), например, 5, $\frac{2}{3}$, 3.14 и 2.1+7 I, причем величина любого числа не ограничена.

Символ представляет собой любую последовательность заглавных и строчных латинских букв, цифр и знака $, набираемых без пробела, которая не начинается с цифры. Кроме того, символ может содержать некоторые специальные буквы, например, греческие. Отметим еще раз, что в системе *Mathematica* заглавные и строчные буквы различаются. Символы используются в качестве имен различных объектов, как встроенных в систему, так и вводимых пользователем.

Строка есть последовательность букв, цифр и специальных символов, заключенная в кавычки, т.е. выражение вида: "This is a string". Строками являются, например, обозначения координатных осей, заголовок, а также надписи, наносимые на график.

Чтобы выяснить, является ли объект атомарным выражением, достаточно применить к нему функцию **AtomQ**, которая принимает одной из двух значений: **True** (Да) или **False** (Нет). Примеры:

```
In[1]:= AtomQ[x]

Out[1]= True

In[2]:= AtomQ["x+y"]

Out[2]= True

In[3]:= AtomQ[x + y]

Out[3]= False
```

Последний пример показывает, что сумма двух символов является не атомарным, а более сложным выражением. Для записи полной формы выражений служит функция **FullForm**. Применяя ее к сумме двух символов, получаем:

```
In[4]:= FullForm[x + y]

Out[4]//FullForm=
        Plus[x, y]
```

Итак, операция сложения двух атомарных выражений имеет заголовок **Plus**, а само выражение содержит только один уровень. Следующий пример показывает структуру более сложного выражения, которому мы присвоим имя *expr1*:

```
In[5]:= expr1 = -3 x^2 Cos[y] + z;  FullForm[expr1]

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

Легко видеть, что в выражении *expr1* имеется несколько уровней. Чтобы выяснить их количество, применим к *expr1* функцию **Depth**, которая определяет *глубину* выражения. В результате получаем:

```
In[6]:= Depth[expr1]

Out[6]= 4
```

Отметим, что функция **Depth** выдает число, которое на единицу больше, чем имеющееся количество уровней. Следовательно, в выражении *expr1* имеется

только три уровня. Чтобы увидеть это более четко, применим к *expr1* функцию **TreeForm**, которая так же как и функция **FullForm** показывает внутреннюю структуру выражения.

```
In[7]:= TreeForm[expr1]

Out[7]//TreeForm=
        Plus[z,  |                                  ]
                 Times[-3,  |            ,  |       ]
                            Power[x, 2]    Cos[y]
```

Однако, в отличие от **FullForm**, функция **TreeForm** изображает полную форму выражения в виде "дерева", причем его элементы, находящиеся на одном уровне, располагаются на одной горизонтали. При этом элементы, не являющиеся атомарными выражениями, изображаются в виде вертикальных отрезков, которые представляют собой "ветви", ведущие на следующий уровень. Теперь легко видеть, что в выражении *expr1* имеется три уровня.

Список также является выражением, и его структуру можно выяснить с помощью функций **FullForm** и **TreeForm**. Так, применяя к списку *s1* функцию **TreeForm**, получаем:

```
In[9]:= s1 = {{c, b}, a, {1, 2, 3}}; TreeForm[s1 ]

Out[9]//TreeForm=
        List[ |            , a,  |              ]
              List[c, b]        List[1, 2, 3]
```

Как видим, список *s1* имеет заголовок **List** и содержит два уровня. Мы уже знаем, что с помощью двойных квадратных скобок можно выделить любой элемент списка, указав последовательно его номер на каждом из уровней. Например, s1[[1, 2]] и s1[[2]] соответствуют символам *b* и *a*. Оказывается, точно так же можно выделять элементы любого выражения. При этом набор чисел, которые фиксируют положение элемента в выражении, определяется по полной форме выражения, выдаваемой функциями **FullForm** или **TreeForm**. Так, используя записанную выше полную форму выражения *expr1*, можно легко проследить, что положение выражения Cos[*y*], например, должно определяться списком чисел вида: {2,3}. Действительно, указав после имени выражения *exp1* соответствующий набор чисел в двойных квадратных скобках, получаем:

```
In[10]:= expr1[[2, 3]]

Out[10]= Cos[y]
```

Следует отметить, что заголовку в выражении соответствует нулевой номер. Поэтому для выделения имени функции *Cos* в выражении *expr1*, например, достаточно в предыдущем примере после цифры 3 добавить ноль.

```
In[11]:= expr1[[2, 3, 0]]

Out[11]= Cos
```

Заголовок всего выражения *expr1* можно выделить следующим образом:

```
In[12]:= expr1[[0]]

Out[12]= Plus
```

Кроме того, для выделения заголовка выражения можно использовать функцию **Head**, например,

In[13]:= **Head[expr1]**

Out[13]= **Plus**

Функция **Level** позволяет выделить все элементы, из которых состоит выражение, и вызывается с двумя или тремя аргументами. В случае команды вида **Level[expr, n]** выдается список элементов выражения *expr*, находящихся на уровнях от первого до *n*-ого. Примеры:

In[14]:= **Level[expr1, 1]**

Out[14]= **{z, -3 x^2 Cos[y]}**

In[15]:= **Level[expr1, 2]**

Out[15]= **{z, -3, x^2, Cos[y], -3 x^2 Cos[y]}**

Если второй аргумент функции **Level[expr, {n}]** заключен в фигурные скобки, то выдается список элементов выражения *expr*, находящихся только на *n*-ом уровне.

In[16]:= **Level[expr1, {2}]**

Out[16]= **{-3, x^2, Cos[y]}**

В случае команды вида **Level[expr, {m, n}]** выдается список элементов, находящихся на уровнях с *m*-ого по *n*-ый включительно.

In[17]:= **Level[expr1, {2, 3}]**

Out[17]= **{-3, x, 2, x^2, y, Cos[y]}**

Если второй аргумент функции **Level** является отрицательным числом, т.е. имеет вид $-n$, то выдается список элементов исходного выражения, глубина которых больше или равна *n*. Легко убедиться в том, что в выражении *expr1* можно выделить только три элемента, имеющих один или более уровней.

In[18]:= **Level[expr1, -2]**

Out[18]= **{x^2, Cos[y], -3 x^2 Cos[y]}**

Приведенные примеры показывают, что функция **Level**, вызываемая с двумя аргументами, выдает список элементов, т.е. выражение, имеющее заголовок **List**. Этот заголовок можно легко заменить другим, добавив его в качестве третьего аргумента функции **Level**. Следующий пример показывает, как можно получить сумму элементов выражения *expr1*, находящихся на втором уровне.

In[19]:= **Level[expr1, {2}, Plus]**

Out[19]= **-3 + x^2 + Cos[y]**

Как и в случае списков, набор чисел, определяющий положение элементов в выражении, можно получить с помощью функции **Position**. Определим, например, положение элемента x^2 в выражении *expr1*.

In[20]:= **Position[expr1, x^2]**

Out[20]= **{{2, 2}}**

Получаем, что в выражении *expr1* элемент x^2 встречается только один раз и является вторым элементом на втором уровне. Подчеркнем еще раз, что список чисел, характеризующий положение элементов в выражении, однозначно определяется только по полной форме выражения, даваемой функциями **FullForm** или **TreeForm**.

8.2. Преобразование выражений

Под *преобразованием выражения* будем понимать любые изменения его структуры. Это может быть удаление элементов выражения, добавление новых элементов, замена одних элементов другими и т.п.. В любом случае получается новое выражение, полная форма которого отлична от формы исходного выражения. В системе *Mathematica* имеется множество функций для преобразования выражений. Мы уже имели дело с некоторыми из них, рассматривая преобразование списков. Следует отметить,что все они могут применяться для преобразования любых выражений. Чтобы выяснить особенности их использования, рассмотрим выражение *expr2*.

In[21]:= **expr2 = a x - b + 7 + 3 y^2 - (x - 1)/(x^2 - y^2);**

Попробуем удалить первое слагаемое в *expr2*, используя функцию **Delete**.

In[22]:= **Delete[expr2, 1]**

Out[22]= $-b + a\,x + 3\,y^2 - \frac{-1 + x}{x^2 - y^2}$

Как видим, в полученном выражении отсутствует число 7, которое было третьим слагаемым в *expr2*. Здесь следует напомнить, что положение элементов в выражении определяется не формой записи, а полной формой выражения, даваемой функцией **FullForm**. В зависимости от свойств заголовка, при обработке выражения *Mathematica* может автоматически произвести перестановку элементов выражения так, что оно будет выглядеть по-другому, хотя его смысл не изменяется. Свойства заголовка определяются его *атрибутами*, т.е. такими характеристиками, список которых выдается по команде **Attributes**. Поскольку выражение *expr2*, очевидно, имеет заголовок **Plus**, его свойства определяются следующими атрибутами:

In[23]:= **Attributes[Plus]**

Out[23]= **{Flat, Listable, NumericFunction, OneIdentity, Orderless, Protected}**

Атрибут **Orderless** означает, что порядок следования элементов в любом выражении с заголовком **Plus** не является существенным, и при обработке соответствующего выражения его элементы будут автоматически записаны в стандартном порядке. Функция **Times**, например, также имеет атрибут **Orderless**. Посмотрим, как *Mathematica* записывает выражение *expr2*.

In[24]:= **expr2**

Out[24]= $7 - b + a\,x + 3\,y^2 - \frac{-1 + x}{x^2 - y^2}$

Напомним, что стандартный порядок в расположении элементов предполагает, что сначала следуют числа, затем символы в алфавитном порядке, а далее более сложные выражения. Таким образом, первым элементом выра-

жения *expr2* оказывается число 7, которое и было удалено функцией **Delete**. Элемент *a x* является третьим, и команда для его удаления имеет вид:

In[25]:= `Delete[expr2, 3]`

Out[25]= $7 - b + 3\,y^2 - \frac{-1+x}{x^2-y^2}$

Чтобы удалить рациональное выражение, которое в *expr2* является последним слагаемым, можно использовать функцию **Most**.

In[26]:= `Most[expr2]`

Out[26]= $7 - b + a\,x + 3\,y^2$

Функции **Prepend**, **Append** и **Insert** позволяют добавлять элементы в начало, в конец или в указанное место выражения соответственно. Однако, если заголовок выражения имеет атрибут **Orderless**, то добавленный элемент окажется в том месте, которое определяется стандартным порядком расположения элементов. Добавляя, например, выражение $x\,y^3$ в конец выражения *expr2*, получаем:

In[27]:= `Append[expr2, x y^3]`

Out[27]= $7 - b + a\,x + 3\,y^2 + x\,y^3 - \frac{-1+x}{x^2-y^2}$

Как видим, добавленное выражение оказалось не последним, а пятым элементом. Это связано с тем, что при записи элементов в стандартном порядке первыми будут следовать более простые выражения, т.е. такие, у которых количество элементов меньше. Именно поэтому рациональное выражение осталось на последнем месте. С помощью функции **FullForm** легко убедиться в том, что элементы $x\,y^3$ и $3\,y^2$ имеют одинаковую структуру.

In[28]:= `FullForm[x y^3]`

Out[28]//FullForm=

```
Times[x, Power[y, 3]]
```

In[29]:= `FullForm[3 y^2]`

Out[29]//FullForm=

```
Times[3, Power[y, 2]]
```

Поэтому первым из них будет следовать элемент, имеющий в качестве множителя число, а не символ, т.е. выражение $3\,y^2$. В следующем примере мы добавляем символ *z* в качестве множителя к рациональному выражению в *expr2*.

In[30]:= `Insert[expr2, z, {5, 1}]`

Out[30]= $7 - b + a\,x + 3\,y^2 - \frac{(-1+x)\,z}{x^2-y^2}$

Хотя список чисел {5,1} указывает на то, что символ *z* должен располагаться на первом месте в пятом элементе выражения *expr2*, он оказывается последним элементом в числителе рационального выражения. Причиной этого является тот факт, что *Mathematica* представляет рациональное выражение в виде произведения, т.е. выражения с заголовком **Times**:

```
In[31]:= FullForm[expr2[[5]]]
Out[31]//FullForm=
        Times[-1, Plus[-1, x],
          Power[Plus[Power[x, 2], Times[-1, Power[y, 2]]], -1]]
```

А этот заголовок имеет атрибут **Orderless** и, следовательно, элементы выражения с таким заголовком располагаются в стандартном порядке.

Функция **ReplacePart** позволяет произвести замену элементов в любом выражении. В качестве примера заменим рациональное выражение в *expr2* выражением Cos[*x*].

```
In[32]:= ReplacePart[expr2, Cos[x], 5]
```

$$Out[32]= 7 - b + a\,x + 3\,y^2 + \text{Cos}[x]$$

В следующем примере производится замена символа *x*, который имеется в третьем и пятом элементах выражения *expr2*.

```
In[33]:= ReplacePart[expr2, Cos[x], {{3, 2}, {5, 2, 2}}]
```

$$Out[33]= 7 - b + 3\,y^2 - \frac{-1 + \text{Cos}[x]}{x^2 - y^2} + a\,\text{Cos}[x]$$

В отличие от **ReplacePart**, функция **ReplaceAll** производит замену элементов на всех уровнях исходного выражения. Ее вторым аргументом должно быть соответствующее правило замены. Примеры:

```
In[34]:= ReplaceAll[expr2, x → Cos[x]]
```

$$Out[34]= 7 - b + 3\,y^2 + a\,\text{Cos}[x] - \frac{-1 + \text{Cos}[x]}{-y^2 + \text{Cos}[x]^2}$$

```
In[35]:= ReplaceAll[expr2, -1 → k]
```

$$Out[35]= 7 + b\,k + a\,x + 3\,y^2 + k\,(k + x)\,(x^2 + k\,y^2)^k$$

Заметим, что краткой формой записи функции **ReplaceAll** является уже известная нам комбинация двух символов " */.* ".

8.3. Шаблоны и их использование для преобразования выражений

Полная форма однозначно определяет структуру любого выражения. Это позволяет ввести конструкцию, которая описывает целый класс однотипных выражений и называется *шаблоном* (**Pattern**). Основным элементом любого шаблона является символ подчеркивания "_", полная форма которого имеет вид:

```
In[36]:= FullForm[_]
Out[36]//FullForm=
        Blank[]
```

Этот символ является шаблоном самого общего вида и используется для обозначения любого выражения, как атомарного, так и более сложного. Поэтому правило подстановки вида _ → *expr*, например, с одним символом

подчеркивания в левой части позволяет заменить исходное выражение другим выражением *expr*. Пример:

In[37]:=
```
a x^2 + b /. _ → Tan[y] + 5 z
```

Out[37]=
```
5 z + Tan[y]
```

Произвольный символ и символ подчеркивания, набираемые без пробела, (например, x_) также представляют собой шаблон общего вида, называемый *именованным шаблоном*. Имя шаблона не влияет на класс выделяемых им выражений. Однако его можно использовать в правой части правила замены, что дает возможность преобразовать все выражение. Пример:

In[38]:=
```
a x^2 + b /. y_ → Sin[y] + y^2
```

Out[38]=
```
(b + a x^2)^2 + Sin[b + a x^2]
```

Шаблоны позволяют изменять не только все выражение целиком, но и любую его часть. Например, выражения вида x^n, где n – произвольное выражение, принадлежат к классу, определяемому шаблоном x^n_. Чтобы убедиться в этом, определим выражение *expr3* и применим к нему правило замены с шаблоном x^n_ в левой части.

In[39]:=
```
expr3 = a x^3 + b x^(2/3) - c/x + d x + x^2 - Cos[k x];

expr3 /. x^n_ → 1 / x^n
```

Out[40]=
$$\frac{a}{x^3} + \frac{1}{x^2} + \frac{b}{x^{2/3}} - c\,x + d\,x - \text{Cos}[k\,x]$$

Как видим, *Mathematica* находит в выражении *expr3* элементы вида x^n и заменяет их на x^{-n}. При этом замены x в первой степени на $1/x$ не происходит, хотя для любого x справедливо соотношение: $x = x^1$ и, казалось бы, атомарное выражение x также принадлежит к классу выражений, определяемых шаблоном x^n_. Причина этого кроется в том, что соответствие выражения шаблону проверяется по его полной форме. И если x является атомарным выражением, то полная форма выражения x^n имеет вид:

In[41]:=
```
FullForm[x^n]
```

Out[41]//FullForm=
```
Power[x, n]
```

Очевидно, что выражения x и x^n имеют различную полную форму. Именно поэтому символ x в первой степени в выражении *expr3* не был заменен на $1/x$. Чтобы добиться такой замены, достаточно в левой части соответствующего правила замены непосредственно после символа подчеркивания добавить точку. Это будет означать, что показатель степени может принимать значение $n = 1$. В результате получаем:

In[42]:=
```
expr3 /. x^n_. → 1 / x^n
```

Out[42]=
$$\frac{a}{x^3} + \frac{1}{x^2} + \frac{d}{x} + \frac{b}{x^{2/3}} - c\,x - \text{Cos}\left[\frac{k}{x}\right]$$

Шаблон вида x_ * y_ соответствует произведению двух произвольных выражений. Применим правило замены с таким шаблоном к произвольному выражению, имеющему заголовок **Times**. В результате получаем:

In[43]:= **a b c^2 /. x_ y_ → z**

Out[43]= **z**

Хотя в рассматриваемом выражении имеется три множителя, оно соответствует структуре шаблона. Действительно, выражение можно записать, например, в виде: $a\,b\,c^2 = a\,(b\,c^2)$, т.е. оно представимо в виде произведения двух выражений. Теперь применим такое же правило замены к выражению *expr3*, которое имеет заголовок **Plus**.

In[44]:= **expr3 /. x_ y_ → z**

Out[44]= **x^2 + 5 z**

Как видим, пять слагаемых в *expr3*, каждое их которых можно представить в виде произведения, заменены символом z. Следует отметить, что выражение (–Cos[*k x*]) соответствует структуре шаблона, поскольку его полная форма имеет вид:

In[45]:= **FullForm[-Cos[k x]]**

Out[45]//FullForm=

```
Times[-1, Cos[Times[k, x]]]
```

Таким образом, операция присваивания знака "минус" произвольному выражению в системе *Mathematica* рассматривается как произведение числа (–1) и выражения. Рациональные выражения, например, c/x, также представляются в виде произведения.

In[46]:= **FullForm[c / x]**

Out[46]//FullForm=

```
Times[c, Power[x, -1]]
```

Однако выражение x^2, которое также можно представить в виде произведения $x^2 = x\,x$, имеет другую полную форму.

In[47]:= **FullForm[x^2]**

Out[47]//FullForm=

```
Power[x, 2]
```

Поэтому оно не подвергается замене. Отметим также, что проверка соответствия струтуры выражения шаблону начинается на уровне заголовка всего выражения. Если на этом уровне выражение не соответствует шаблону, поиск продолжается на первом уровне, затем на втором и т.д.. Именно поэтому производится замена всего выражения (–Cos[*k x*]) в *expr3* символом *z*, а не аргумента косинуса, который также является произведением двух символов.

Рассмотрим еще несколько примеров использования шаблонов для преобразования выражений.

In[48]:= **expr4 = 3 Cos[x] + $\sqrt{y}$ - 2 x^2 y + 5 a Sin[3 y];**

```
expr4 /. f_[x] → h
```

Out[49]= **3 h + $\sqrt{y}$ - 2 x^2 y + 5 a Sin[3 y]**

Шаблон вида f_[*x*] соответствует произвольной функции одного фиксированного аргумента *x*. Поскольку в выражении *expr4* имеется только одна

функция от *x*, а именно, Cos[*x*], то только она заменяется символом *h*. Заметим, что замена $\sqrt{x}$ на *h* не производится, так как полная форма функции *Sqrt[x]* не соответствует шаблону f_[*x*]. Действительно,

In[50]:= **FullForm[Sqrt[x]]**

Out[50]//FullForm=

```
Power[x, Rational[1, 2]]
```

Добавляя символ подчеркивания у аргумента функции *x*, получаем шаблон f_[x_], который обозначает произвольную функцию одного произвольного аргумента. Очевидно, при таком шаблоне в левой части правила замены в последнем примере как Cos[*x*], так и Sin[3*y*] в выражении *expr4* будут заменены символом *h*.

In[51]:= **expr4 /. f_[x_] → h**

Out[51]= $3\,h + 5\,a\,h + \sqrt{y} - 2\,x^2\,y$

Добавляя шаблон a_ в качестве множителя к шаблону, рассмотренному в последнем примере, получаем новый шаблон, который позволяет выделить произведения произвольного выражения на любую функцию одного аргумента. Заменяя такие выражения символом *h*, приводим *expr4* к виду:

In[52]:= **expr4 /. a_ f_[x_] → h**

Out[52]= $2\,h + \sqrt{y} - 2\,x^2\,y$

Подобным образом можно построить множество различных шаблонов. Но иногда требуется ограничить класс выделяемых выражений. Пусть, например, в последнем примере мы хотим заменить символом *h* не все произведения выражений на функции одной переменной, а только такие, у которых множителями функций являются целые числа. Решить такую задачу можно путем замены шаблона a_ шаблоном с заголовком *Integer*, который имеет вид: **_Integer**. Заметим, что заголовок набирается непосредственно после символа подчеркивания без пробела. Легко видеть, что полученный шаблон решает задачу.

In[54]:= **expr4 /. _Integer * f_[x_] → h**

Out[54]= $h + a\,h + \sqrt{y} - 2\,x^2\,y$

Итак, шаблон **_Integer** выделяет класс атомарных выражений, которые являются целыми числами. Аналогично можно выделить класс действительных (**_Real**), рациональных (**_Rational**) и комплексных (**_Complex**) чисел, а также произвольный символ (**_Symbol**) или список (**_List**). Кроме того, можно добавить у шаблона имя, например, **x_Symbol**, что позволяет использовать это имя в правой части правила замены. Пример:

In[55]:= **a x^2 + b x^m + c /. x^n_Symbol → x^(n - 1)**

Out[55]= $c + a\,x^2 + b\,x^{-1+m}$

Заметим, что проверка эквивалентности математических выражений в общем случае представляет собой очень сложную задачу. В системе *Mathematica* она решается путем сравнения не самих выражений, а их структур или полных форм. Очевидно, при таком подходе некоторые математически эквивалентные выражения не будут признаны таковыми, так как их структуры

различны. Примером такого рода являются следующие эквивалентные выражения $(x-1)^2$ и x^2-2x+1. Их полные формы имеют вид:

```
In[56]:= FullForm[(x - 1)^2]
Out[56]//FullForm=
        Power[Plus[-1, x], 2]

In[57]:= FullForm[x^2 - 2 x + 1]
Out[57]//FullForm=
        Plus[1, Times[-2, x], Power[x, 2]]
```

Поэтому шаблон **_^_Integer**, который позволяет выделить первое выражение, не годится для выделения второго. Чтобы проверить, соответствует ли выражение шаблону, можно использовать функцию **MatchQ**.

```
In[58]:= MatchQ[(1 - x)^2, _^_Integer]
Out[58]= True

In[59]:= MatchQ[1 - 2 x + x^2, _^_Integer]
Out[59]= False
```

Очевидно, можно добиться эквивалентности рассмотренных двух выражений, преобразовав одно из них с помощью функций **Expand** или **Factor**. После этого выражение будет соответствовать шаблону, например,

```
In[60]:= MatchQ[Factor[1 - 2 x + x^2], _^_Integer]
Out[60]= True
```

Из большого количества функций, позволяющих работать с шаблонами, отметим еще одну, которая позволяет получить список элементов, находящихся на заданном уровне исходного выражения и соответствующих шаблону. Это функция **Cases**, которая вызывается с двумя, тремя или четырьмя аргументами. В первом случае она используется для выделения элементов выражения, соответствующих шаблону и находящихся на первом уровне. Пример:

```
In[61]:= Cases[{a, a + b, a b, a^2 - b^2, Sin[(a - b)/(2 a + b) + k]}, x_ + y_]
Out[61]= {a + b, a^2 - b^2}
```

Здесь производится выделение элементов списка, которые представляют собой сумму двух произвольных выражений. Заметим, что шаблон _ + _ соответствует сумме двух одинаковых выражений. Легко видеть, что в рассмотренном списке такие элементы отсутствуют.

```
In[62]:= Cases[{a, a + b, a b, a^2 - b^2, Sin[(a - b)/(2 a + b) + k]}, _ + _]
Out[62]= {}
```

Число *n*, подставляемое в качестве третьего аргумента функции **Cases**, указывает, что поиск элементов следует производить на уровнях с первого до *n*-ого. Пример:

In[63]:= `Cases[{a, a+b, ab, a^2-b^2, Sin[(a-b)/(2a+b)+k]}, x_+y_, 2]`

Out[63]= $\left\{a+b,\ a^2-b^2,\ \frac{a-b}{2a+b}+k\right\}$

Подстановка вместо числа *n* бесконечности (**Infinity**) означает, что поиск следует производить на всех уровнях выражения.

In[64]:=
```
Cases[{a, a+b, ab, a^2-b^2, Sin[(a-b)/(2a+b)+k]},
  x_+y_, Infinity]
```

Out[64]= $\left\{a+b,\ a^2-b^2,\ a-b,\ 2a+b,\ \frac{a-b}{2a+b}+k\right\}$

Если же мы хотим ограничиться первыми *m* элементами, то число *m* следует добавить в качестве четвертого аргумента. Пример:

In[65]:=
```
Cases[{a, a+b, ab, a^2-b^2, Sin[(a-b)/(2a+b)+k]},
  x_+y_, Infinity, 4]
```

Out[65]= `{a+b, a^2-b^2, a-b, 2a+b}`

Заметим, что для выделения элементов только *k*-ом уровне следует использовать третий аргумент вида: {*k*}.

8.4. Определение функций

Любая функция определяет правило, по которому каждому значению переменной, например *x*, ставится в соответствие некоторое выражение *expr*. Одна из возможностей установить такое соответствие в системе *Mathematica* – это задать правило замены вида $x\_ \to expr$, где *expr* определяет форму зависимости от *x*. При этом в каждом случае, когда необходимо применить это правило, мы должны использовать оператор подстановки "**/.**" или функцию **ReplaceAll**. Вместо этого часто оказывается удобным определить дополнительную функцию, которая будет автоматически выполнять такую же подстановку при обращении к ней. Использование функций избавляет от необходимости многократной записи однотипных выражений и упрощает процесс программирования.

Пусть, например, требуется определить функцию $f(x) = x/(x+3)$. Соответствующая команда в системе *Mathematica* имеет вид:

In[66]:= $\mathbf{f}[\mathbf{x\_}] = \frac{\mathbf{x}}{\mathbf{x}+3}$;

На первом месте в левой части этого определения стоит имя функции *f*, которое представляет собой произвольный символ (т.е. произвольное атомарное выражение) и выбирается пользователем исходя из соображений удобства. Поскольку имена всех встроенных в систему *Mathematica* функций начинаются с заглавной буквы, во избежание ошибок рекомендуется писать имена определяемых функций с малой буквы. Следует обратить также

особое внимание на символ подчеркивания, который набирается непосредственно за аргументом функции в левой части ее определения. Напомним, что выражение x_ является шаблоном общего вида. Именно поэтому при обращении к функции *f* в качестве ее аргумента можно будет подставить любое выражение. В правой части определения функции приводится функциональная зависимость от *x*, причем символ подчеркивания за переменной не ставится. Как мы уже видели в предыдущем параграфе, аналогичным образом поступают при определении правила замены с шаблоном в его левой части.

После определения функции ее заголовок можно использовать точно так же, как и заголовок любой встроенной функции. Отметим, что при обращении к функции символ подчеркивания после аргумента не ставится. Примеры:

```
In[67]:= Simplify[ f[x^2 - 12] - 3 x f[x] ]
```

$$Out[67]= \frac{-12 + 10\,x^2 - 3\,x^3}{-9 + x^2}$$

```
In[68]:= Solve[f[x - 1] == 2 x - 5, x]
```

$$Out[68]= \left\{\left\{x \to \frac{1}{2}\left(1 - \sqrt{19}\right)\right\}, \left\{x \to \frac{1}{2}\left(1 + \sqrt{19}\right)\right\}\right\}$$

Если определяемая функция зависит от нескольких переменных, то в левой части соответствующего определения аргументы разделяются запятыми, причем каждый аргумент обязательно набирается с символом подчеркивания. Пример:

```
In[69]:= fun[x_, y_, z_] = Sin[x + y - z];

In[70]:= fun[π / 2, π / 3, π / 6]
```

$$Out[70]= \frac{\sqrt{3}}{2}$$

Следует отметить, что знак равенства "=" в определении функции обозначает операцию присваивания имени *f* соответствующего выражения. Поэтому при обработке правой части этого определения *Mathematica* пытается произвести все указанные там вычисления. Если некоторые из них не могут быть выполнены, выдается сообщение об ошибке. В качестве примера попробуем определить следующую функцию:

```
In[71]:= int[a_, y_] = NIntegrate[Cos[a x^2], {x, 0, y}]

NIntegrate::nlim :
 x = y is not a valid limit of integration. More...

NIntegrate::nlim :
 x = y is not a valid limit of integration. More...
```

$$Out[71]= \text{NIntegrate}\left[\text{Cos}\left[a\,x^2\right], \{x, 0, y\}\right]$$

При этом получаем сообщение, что верхний предел интегрирования не задан численно, и операция численного интегрирования не может быть выполнена. Кроме того, подынтегральныя функция зависит от параметра, и ее численное значение также не может быть найдено. В результате функция *int[a, y]* оказывается неопределенной. Тем не менее эту функцию можно определить, поскольку при любых численных значениях аргументов *a* и *y*

значение функции *int* может быть найдено. Для этого достаточно вместо оператора присваивания "=" использовать оператор отложенного присваивания ":=", который представляет собой комбинацию двух символов ":" и "=", набираемых без пробела.

```
In[72]:= int[a_, y_] := NIntegrate[Cos[a x^2], {x, 0, y}]
```

При наличии оператора отложенного присваивания вычисления в правой части определения функции не производятся. Именно поэтому результат вычислений не выводится, хотя точка с запятой в конце выражения отсутствует. Тем не менее, имя этой функции можно использовать в любом выражении. Например, можно найти численное значение функции при некоторых значениях параметров.

```
In[73]:= int[2, 3]

Out[73]= 0.379161
```

Подчеркнем еще раз, что вычисление значения функции, в определении которой использован оператор отложенного присваивания, производится только при обращении к ней. Тем не менее, возможна ситуация, когда при обращении к функции мы указали, например, численное значение только для одного из аргументов: int[*a*, 3].

```
In[74]:= int[a, 3]

    NIntegrate::inum :
      Integrand Cos[a x^2] is not numerical at {x} = {1.5}. More...

Out[74]= NIntegrate[Cos[a x^2], {x, 0, 3}]
```

В результате опять получаем сообщение об ошибке. Чтобы предотвратить попытку вычисления функции в том случае, когда ее аргументы не заданы численно, можно при ее определении использовать шаблон, выделяющий только числа. В таком шаблоне непосредственно после символа подчеркивания следует набрать знак вопроса и имя функции **NumberQ**. Эта функция анализирует заголовок аргумента и выдает "Да" или "Нет" в зависимости от того, является он числом или нет.

```
In[75]:= NumberQ[2]

Out[75]= True

In[76]:= NumberQ[a]

Out[76]= False
```

Таким образом, определение функции, аргументы которой должны быть заданы только численно, имеет вид:

```
In[77]:= int1[a_?NumberQ, y_?NumberQ] :=
           NIntegrate[Cos[a x^2], {x, 0, y}]
```

Теперь легко убедиться в том, что вычисление значения функции производится только тогда, когда оба ее аргумента являются числами.

```
In[78]:= int1[a, 3]

Out[78]= int1[a, 3]
```

In[79]:= **int1[2, 3]**

Out[79]= **0.379161**

Имена определяемых пользователем функций, как и имена других объектов, сохраняются в памяти компьютера до конца текущего сеанса работы с системой *Mathematica* или до тех пор, пока они не удалены. Напомним, что для удаления присвоенных имен используется функция **Clear**.

In[80]:= **Clear[f, fun, int, int1]**

В рассмотренных выше примерах каждая из определяемых функций имела свое имя, т.е. являлась *именованной* функцией. Введение таких функций целесообразно, если они будут использоваться во многих местах программы. Кроме того, в системе *Mathematica* имеется возможность определить *анонимную (неименованную)* или *чистую* функцию (*pure function*), для которой не требуется никакого имени. Она представляет собой выражение вида:

Function[*x*, body] или **Function[{x_1, x_2, ...}, body]**,

где второй аргумент *body* определяет требуемую функциональную зависимость от одной или нескольких независимых переменных, список которых указывается в качестве первого аргумента. При этом символы *x*, x_1, x_2, ... являются локальными переменными, которые используются только в пределах данной процедуры. Это значит, что символ *x*, например, может использоваться в остальной части программы в качестве имени любого другого объекта, и он не будет влиять на работу функции. Чтобы продемонстрировать это, присвоим символу *x* какое-либо численное значение, например,

In[81]:= **x = 5;**

Далее определяем чистую функцию, в которой используем символ *x* в качестве независимой переменной, и применяем ее, например, к выражению $(a + b)$.

In[82]:= **Function[x, x^2 + 4][a + b]**

Out[82]= $4 + (a + b)^2$

Обратите внимание, что для применения чистой функции к какому-либо выражению достаточно поместить его непосредственно за определением функции в квадратных скобках. Отметим также, что никаких ошибок в работе функции не возникло, хотя символу *x* было присвоено значение 5. Более того, применяя функцию к символу *x*, получаем правильное численное значение функции.

In[83]:= **Function[x, x^2 + 4][x]**

Out[83]= **29**

При определении чистой функции можно вообще избежать употребления имен для обозначения независимых переменных, а заголовок **Function** заменить символом **&**. В этом случае при записи функциональной зависимости в качестве локальных переменных используются выражения вида **#1**, **#2**, **#3** и т.д., где число, следующее за символом **#** (выражение с заголовком **Slot**), обозначает номер аргумента, а символ **&** пишется в конце выражения. Так, рассмотренную выше функцию можно записать в виде: **(#1^2+4&)**. Заметим, что номер у первого аргумента ставить не обязательно, т.е. вместо **#1** можно просто записать **#**, а все выражение для чистой

функции, включая символ **&**, рекомендуется заключать в круглые скобки. При обращении к такой функции достаточно после соответствующего выражения указать в квадратных скобках нужное количество аргументов. Примеры:

```
In[84]:= ( #^2 + 4 &) [x + y]
```

$$Out[84]= 4 + (5 + y)^2$$

```
In[85]:= (√(#1^2 + #2^2) &) [a + b, a - b]
```

$$Out[85]= \sqrt{(a - b)^2 + (a + b)^2}$$

Отметим, что используемая функция может быть довольно сложной, и для ее определения требуется несколько команд. В таком случае всю правую часть в определении функции можно заключить в круглые скобки, а отдельные команды разделить точками с запятой. В качестве примера рассмотрим функцию, которая вычисляет среднее арифметическое *n* случайных чисел, принадлежащих интервалу [0,1].

```
In[86]:= averageValue[n_] := (data = Table[Random[], {n}];
                              average = 1/n Sum[data[[k]], {k, n}]; average)
```

Выполняя вычисления при различных значениях *n*, получаем:

```
In[87]:= Table[averageValue[10^k], {k, 6}]
Out[87]= {0.454996, 0.494021, 0.502923,
          0.504009, 0.500597, 0.499902}
```

Интересно отметить, что с увеличением количества случайных чисел их среднее оказывается все ближе к $\frac{1}{2}$. Недостатком приведенного выше определения функции *averageValue* является тот факт, что в нем использованы вспомогательные переменные *data* и *average*, которые являются глобальными и по окончании вычислений сохраняют последние присвоенные им значения. Символ *average*, например, имеет значение

```
In[88]:= average
Out[88]= 0.499902
```

Поэтому, во избежание ошибок, по окончании работы с функцией *averageValue* рекомендуется удалить все использованные переменные.

```
In[89]:= Clear[x, data, average, averageValue]
```

В системе *Mathematica* имеется специальная функция **Module**, которая позволяет локализовать вспомогательные переменные, вводимые при определении функций, и тем самым избежать появления дополнительных глобальных переменных. При обращении к ней необходимо указать два аргумента: **Module[{ x, y, ...}, expr]**. Первый из них представляет собой список локальных переменных, а второй содержит определение функциональной зависимости *expr*. Если в *expr* содержится несколько команд, их следует разделять точкой с запятой. В качестве примера определим с помощью функции **Module** рассмотренную выше функцию для вычисления среднего арифметического *n* случайных чисел. Соответствующая команда имеет вид:

```
In[90]:= averageValue[n_] := Module[{data, average},
                    data = Table[Random[], {n}];
                    average = 1/n Sum[data[[k]], {k, n}]; average]
```

Найдем значение этой функции при $n = 10000$.

```
In[91]:= averageValue[10000]

Out[91]= 0.50213
```

Очевидно, именно такое значение было присвоено локальной переменной *average* в ходе вычислений. Однако на вопрос, что представляет собой объект с именем *average*, *Mathematica* выдает следующий ответ:

```
In[92]:= ? average

    Global`average
```

Таким образом, как глобальная переменная, *average* не имеет никакого значения.

Еще одной функцией, которая позволяет использовать локальные переменные, является функция **Block[{ x, y, ...}, expr]**, при обращении к которой нужно указать такие же аргументы, что и у функции **Module**. Их отличие состоит только в механизмах локализации вспомогательных переменных.

8.5. Применение функций к выражениям

Стандартной формой обращения к любой функции, как встроенной, так и определенной пользователем, является выражение вида *f*[*x*, *y*, ...], где *f* – имя функции, а аргументы *x*, *y*, ... представляют собой некоторые выражения, в простейшем случае атомарные. Именно такая форма используется наиболее часто при записи математических формул и определении функциональных операций и команд. Кроме того, в системе *Mathematica* имеются еще три формы записи выражений: *префиксная*, *постфиксная* и *инфиксная*. В некоторых случаях использование этих форм позволяет записать выражение в более наглядном и понятном виде.

В случае префиксной формы сначала записывается имя функции, затем символ "@", а потом выражение, к которому функция должна быть применена. В качестве примера вычислим число π с точностью до 20 значащих цифр.

```
In[93]:= N[#, 20] &  @  Pi

Out[93]= 3.1415926535897932385
```

Чтобы вычисления были выполнены с требуемой точностью, мы не просто указали имя используемой функции **N**, а определили ее в виде чистой функции. Следует отметить, что префиксная форма записи имеет очень высокий приоритет. Следующий пример показывает, что при отсутствии круглых скобок заданная функция применяется только к объекту, следующему непосредственно за символом "@".

```
In[94]:= Cos  @  Pi + Pi

Out[94]= -1 + π
```

Как видим, сначала вычисляется $\cos\pi$, а только потом сумма двух чисел. Поэтому при использовании префиксной формы все выражение, к которому нужно применить функцию, следует заключать в круглые скобки.

```
In[95]:= Cos @ (Pi + Pi)

Out[95]= 1
```

В случае постфиксной формы имя функции, которую нужно применить к выражению, записывается в самом конце после двойной наклонной черты. Это означает, что постфиксная форма имеет очень низкий приоритет, и указанная функция применяется к выражению после выполнения других операций. Пример:

```
In[96]:= (a^2 + a b + 1)^3 + 2 // Collect[#, a] &

Out[96]= 3 + a^6 + 3 a b + 3 a^5 b + a^2 (3 + 3 b^2) + a^4 (3 + 3 b^2) + a^3 (6 b + b^3)
```

Инфиксную форму удобно применять в случае функций, зависящих от двух переменных. При этом имя функции располагается между переменными и отделяется от них символами "~". Так, инфиксная форма команды, которая объединяет два списка в один, имеет вид:

```
In[97]:= {1, 2, 3} ~ Join ~ {4, 5, 6}

Out[97]= {1, 2, 3, 4, 5, 6}
```

Заметим, что префиксная, постфиксная и инфиксная формы могут использоваться не только при обращении к какой-либо функции, но и при записи любых выражений. В таких случаях вместо имени функции необходимо указать заголовок выражения, например,

```
In[98]:= a ~ Less ~ b

Out[98]= a < b
```

Напомним, что свойства каждой встроенной функции в системе *Mathematica* определяются набором ее атрибутов, который выдается по команде **Attributes[function]**. В случае косинуса, например, список атрибутов имеет вид:

```
In[99]:= Attributes[Cos]

Out[99]= {Listable, NumericFunction, Protected}
```

Атрибут **Listable** означает, что функция обладает свойством дистрибутивности, т.е. при применении такой функции к списку автоматически происходит ее применение к каждому элементу списка. Пример:

```
In[100]:= Cos[ π/2 Range[10] ]

Out[100]= {0, -1, 0, 1, 0, -1, 0, 1, 0, -1}
```

Добиться аналогичного эффекта в случае функции, не обладающей свойством дистрибутивности, можно с помощью функции **Map**. Пример:

```
In[101]:= Map[f, {x, y, z}]

Out[101]= {f[x], f[y], f[z]}
```

Как видим, функция **Map** обеспечивает применение заголовка *f* к каждому элементу списка. Если она вызывается с двумя аргументами, то заголовок *f*

применяется только к элементам первого уровня, что хорошо видно из следующего примера.

In[102]:= `Map[f, {{a, b, c}, d, {x, y, {z}}}]`

Out[102]= `{f[{a, b, c}], f[d], f[{x, y, {z}}]}`

Добавляя третий аргумент, можно определить уровни, на которых будет применяться заголовок *f*. Примеры:

In[103]:= `Map[f, {{a, b, c}, d, {x, y, {z}}}, 2]`

Out[103]= `{f[{f[a], f[b], f[c]}], f[d], f[{f[x], f[y], f[{z}]}]}`

In[104]:= `Map[f, {{a, b, c}, d, {x, y, {z}}}, {2}]`

Out[104]= `{{f[a], f[b], f[c]}, d, {f[x], f[y], f[{z}]}}`

Как и в случае функции **Cases**, например, число *n*, указанное в качестве третьего аргумента функции **Map**, означает, что заголовок *f* следует применить на всех уровнях с первого по *n*-ый включительно. Если же число *n* заключено в фигурные скобки, то заголовок применяется только на *n*-ом уровне.

Для применения заголовка *f* к конкретным элементам выражения используется функция **MapAt**. При этом ее третий аргумент должен содержать список чисел, определяющих положения требуемых элементов. Примеры:

In[105]:= `MapAt[f, {{a, b, c}, d, {x, y, {z}}}, {{1, 2}, {3, 3, 1}}]`

Out[105]= `{{a, f[b], c}, d, {x, y, {f[z]}}}`

In[106]:= `MapAt[f, x + y + a b, {3}]`

Out[106]= `a b + x + f[y]`

Напомним, что положение элемента в выражении определяется не формой записи, а его полной формой. Именно поэтому в последнем примере третим элементом является символ *y*, а не произведение *a b*. Действительно,

In[107]:= `FullForm[x + y + a b]`

Out[107]//FullForm=

```
Plus[Times[a, b], x, y]
```

Чтобы применить заголовок *f* на всех уровнях выражения, используется функция **MapAll**.

In[108]:= `MapAll[f, {{a, b, c}, d, {x, y, {z}}}]`

Out[108]=

```
f[{f[{f[a], f[b], f[c]}],
    f[d], f[{f[x], f[y], f[{f[z]}]}]}]
```

Функция **Apply[f, expr]** позволяет заменить заголовок выражения *expr* на *f*. Очевидно, в случае такой замены смысл выражения полностью изменяется. Так, в следующем примере с помощью функции **Apply** мы преобразуем сумму трех элементов в произведение, заменяя заголовок **Plus** на **Times**.

In[109]:= `Apply[Times, x + y + z^2]`

Out[109]= $x\,y\,z^2$

Как и в случае функции **Map**, третий аргумент функции **Apply** позволяет задать уровень, на котором следует производить замену заголовков. Примеры:

In[110]:= `Apply[Times, Cos[x] + y + z^2, 1]`

Out[110]= `x + y + 2 z`

In[111]:= `Apply[Times, Sin[x + y] + Sin[2 x], {2}]`

Out[111]= `Sin[2 x] + Sin[x y]`

In[112]:= `Apply[Times, Sin[x + y] + Sin[2 x], {0, 2}]`

Out[112]= $2\,x^2\,y$

Заметим, что в последнем примере третий аргумент вида {0,2} указывает на то, что требуется заменить на **Times** как заголовок всего выражения, так и все заголовки на первом и втором уровнях.

Если аргументами функции *f* являются списки, т.е. выражение имеет вид $f[\{a_1, b_1\}, \{a_2, b_2\}]$, например, то применение к нему функции **Thread** приводит к появлению нового списка $\{f[a_1, a_2], f[b_1, b_2]\}$. Таким образом, действие функции **Thread** сводится к переносу заголовка *f* на соответствующие элементы списков, которые являются аргументами *f*. Заметим, что такая операция возможна лишь в том случае, когда размерности этих списков одинаковы. Пример:

In[113]:= `Thread[ f[{a, b, c}, {x, y, z}]]`

Out[113]= `{f[a, x], f[b, y], f[c, z]}`

Если некоторые аргументы являются символами или числами, то функция **Thread** рассматривает их как списки одинаковых элементов. В результате получается следующий список:

In[114]:= `Thread[ f[{a, b}, c, {x, y}, 2]]`

Out[114]= `{f[a, c, x, 2], f[b, c, y, 2]}`

Использование функции **Thread** позволяет, например, легко получить из матричного уравнения соответствующую систему уравнений. Примеры:

In[115]:= `Thread[{g[x, y], h[x, y]} == {a, b}]`

Out[115]= `{g[x, y] == a, h[x, y] == b}`

In[116]:= `{g[x, y], h[x, y]} == 0 // Thread`

Out[116]= `{g[x, y] == 0, h[x, y] == 0}`

Функция **Thread** позволяет переносить заголовок *f* не только на элементы списков, но и на элементы других выражений. Если аргументы функции *f* в выражении **Thread[*f*[arg1, arg2], *h*]** имеют одинаковые заголовки *h*, то добавив этот заголовок в качестве второго аргумента функции **Thread**, получим новое выражение с таким же заголовком, причем элементы этого выражения будут представлять собой результат действия функции *f* на соответствующие элементы выражений *arg1* и *arg2*. Пример:

In[117]:= `Thread[f[a + b, c + d], Plus]`

Out[117]= `f[a, c] + f[b, d]`

Если же заголовки у *arg1* и *arg2* разные, то в качестве заголовка *h* можно выбрать любой их них. Естественно, получаемый результат будет зависеть от выбора *h*. Примеры:

In[118]:=
```
Thread[f[a + b, x^y], Power]
```

Out[118]=
$$f[a+b,\,x]^{f[a+b,y]}$$

In[119]:=
```
Thread[f[a + b, x^y], Plus]
```

Out[119]=
$$f[a,\,x^y] + f[b,\,x^y]$$

В качестве еще одного примера приведем команду, позволяющую умножить обе стороны уравнения на один и тот же множитель.

In[120]:=
```
Thread[ a (f[x] == g[x]), Equal]
```

Out[120]=
$$a\,f[x] == a\,g[x]$$

8.6. Циклические и условные операторы

Основной функцией, которая позволяет циклически повторять вычисление одного или нескольких выражений, является в системе *Mathematica* функция **Do[expr, iterator]**. Ее первый аргумент *expr* содержит выражения, предназначенные для вычисления. Если их несколько, то выражения отделяются друг от друга точкой с запятой. Второй аргумент *iterator* является итератором, который определяется точно так же, как и для функций **Table**, **Sum** и **Product**. В простейшем случае он указывает, сколько раз следует вычислить *expr*. Пример:

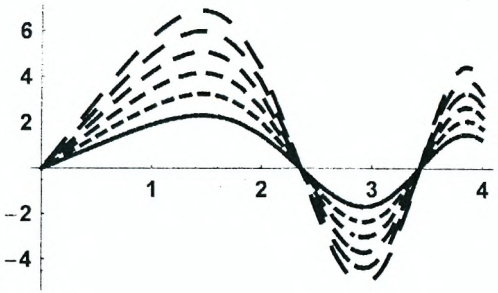
In[121]:=
```
t = a ; Do[t = 1 - 1 / t, {3}] ; t
```

Out[121]=
$$1 - \frac{1}{1 - \frac{1}{1-\frac{1}{a}}}$$

Легко убедиться в том, что результат вычислений *expr*, получаемый с помощью функции **Do**, на экран не выводится. Чтобы увидеть его, мы поместили символ *t*, которому и присвоено полученное выражение, после функции **Do**.

В следующем примере с помощью функции **Do** мы генерируем список численных решений дифференциального уравнения при различных начальных условиях, а затем изображаем полученные решения на одной координатной плоскости. При этом мы используем более сложный итератор.

In[127]:=
```
Clear["Global`*"]; graphs = {};
    Do[
      sol = NDSolve[{y''[x] + x^2 y[x] == 0, y[0] == 0, y'[0] == k},
        y[x], {x, 0, 4}];
      graphs = Append[graphs, y[x] /. sol[[1]] ],
      {k, 2, 6, 0.8}];
    Plot[graphs // Evaluate, {x, 0, 4},
      PlotStyle → Table[{Thickness[0.01],
         Dashing[{0.01 k}]}, {k, Length[graphs]}]];
```



Отметим, что перед построением графиков функций, содержащихся в списке *graphs*, который генерируется функцией **Do**, требуется сформировать этот список. Для этого к списку *graphs* применяется функция **Evaluate**.

В некоторых случаях вместо функции **Do** целесообразно использовать функцию **Nest[f, expr, n]**, которая позволяет применить функцию *f* к выражению *expr n* раз. Чтобы продемонстрировать ее работу, вычислим выражение из первого примера. Соответствующая команда имеет вид:

In[130]:= `Nest[1 - 1/# &, a, 3]`

Out[130]= $1 - \dfrac{1}{1 - \frac{1}{1-\frac{1}{a}}}$

К достоинствам функции **Nest** следует отнести тот факт, что с ее помощью мы выполнили вычисления, не вводя ни одной вспомогательной переменной. Если же мы хотим получить не только окончательный результат вычислений, но и список всех промежуточных результатов, можно воспользоваться функцией **NestList**. Формат обращения к этой функции точно такой же, как и к функции **Nest**. Пример:

In[131]:= `NestList[1 - 1/# &, a, 3]`

Out[131]= $\left\{a,\ 1 - \dfrac{1}{a},\ 1 - \dfrac{1}{1 - \frac{1}{a}},\ 1 - \dfrac{1}{1 - \frac{1}{1-\frac{1}{a}}}\right\}$

В некоторых случаях при многократном применении функции к выражению на некотором шаге результат перестает изменяться. Вычислим, например, первые десять чисел, которые получаются при многократном применении функции **Sqrt** к числу 3.

In[132]:= `NestList[Sqrt, 3., 10]`

```
Out[132]= {3., 1.73205, 1.31607, 1.1472, 1.07108, 1.03493,
          1.01731, 1.00862, 1.0043, 1.00215, 1.00107}
```

Как и ожидалось, получили убывающую последовательность чисел, которая стремится к 1. Однако, заранее неизвестно, сколько раз следует применить функцию **Sqrt**, чтобы результат перестал изменяться. В таком случае можно воспользоваться функцией **FixedPoint[f, expr]**, которая применяет *f* к

выражению *expr* до тех пор, пока не получится неизменяемый результат. В результате находим:

In[133]:= **FixedPoint[Sqrt, 3.]**

Out[133]= **1.**

Ясно, что для получения точной единицы нужно применить функцию **Sqrt** к числу 3 бесконечное число раз. Поскольку вычисления производятся с машинной точностью, то неизменяемый результат получается после конечного числа шагов. При этом его точность равна:

In[134]:= **Precision[%]**

Out[134]= **MachinePrecision**

С помощью функции **NestWhile[f, expr, test]** можно последовательно применять *f* к выражению *expr* до тех пор, пока будет выполняться условие *test*. В следующем примере мы применяем функцию **Sqrt** к числу 3 до тех пор, пока результат вычислений отличается от 1 больше, чем на 0.01. При этом условие *test* определено в виде чистой функции.

In[135]:= **NestWhile[Sqrt, 3., (Abs[# - 1] > 0.01) &]**

Out[135]= **1.00862**

Добавляя число 2 в качестве четвертого аргумента функции **NestWhile**, мы указываем, что условие *test* следует применять к последним двум результатам. В данном случае вычисления прекращаются, когда два последовательных результата отличаются меньше, чем на 0.005.

In[136]:= **NestWhile[Sqrt, 3., (Abs[#1 - #2] > 0.005) & , 2]**

Out[136]= **1.0043**

В системе *Mathematica* имеются еще две функции, которые позволяют многократно производить вычисления выражений при выполнении заданных условий. Это **While[test, expr]** и **For[start, test, incr, expr]**. Функция **While** вызывается с двумя аргументами и повторяет вычисление *expr* до тех пор, пока условие *test* принимает значение **True**. В качестве примера вычислим, сколько раз следует бросить две кости, чтобы сумма выпадающих на них чисел оказалась больше 10.

```
In[144]:= n = 1; While[ Random[Integer, {1, 6}] +
                        Random[Integer, {1, 6}] < 11,
        n = n + 1]; n
```

Out[144]= **6**

Заметим, что функция **While** сначала проверяет условие *test* и только в случае его выполнения вычисляет *expr*. Именно поэтому при многократной обработке предыдущей секции можно получить результат $n = 1$, который означает, что условие *test* оказывается невыполненным уже после первого броска.

У функции **For** первый аргумент *start* содержит некоторые начальные данные, второй аргумент *test* определяет условие, при выполнении которого происходит увеличение текущей переменной (аргумент *incr*) и вычисляется выражение *expr*. Чтобы продемонстрировать ее работу, вычислим корень

уравнения $x^3 - 3x + 1 = 0$ с точностью до 0.01. Используя метод касательных, запишем рекурентную формулу для вычисления корней:

$$x_{n+1} = x_n - f(x_n) / f'(x_n),$$

где функция $f(x)$ соответствует левой части рассматриваемого уравнения. Тогда процедуру для вычисления корней можно записать в виде:

```
In[145]:= f[x_] := x^3 - 3x + 1;
          For[i = 1; x0 = 1.6, Abs[f[x0]] > 0.01, i++,
           x0 = x0 - f[x0] / f'[x0]; Print[{i, x0}]]; {i, x0}
```

From In[145]:=

```
{1, 1.53675}
```

From In[145]:=

```
{2, 1.53211}
```

```
Out[146]= {3, 1.53211}
```

Здесь *i* и *x0* обозначают номер итерации и текущее значение искомого корня соответственно. Поскольку имеется начальные значения для двух переменных *i* и *x0*, то соответствующие им операторы присваивания разделены точкой с запятой. Требуемая точность вычисления корня определяется условием $|f(x0)| > 0.01$, которое является вторым аргументом функции **For**. Операция увеличения параметра *i* на единицу обозначена как *i*++, что является короткой формой записи выражения $i = i + 1$. Как видим, требуемая точность вычислений достигается уже после двух итераций. Обратите внимание, что при нарушении условия *test* параметр *i* тем не менее увеличивается на 1, а вычисление *expr* не производится. Именно поэтому последнее значение корня, выводимое по команде **Print**, совпадает с результатом, получаемым по команде {*i*, *x0*}, которая записана после функции **For**, а соответствующие значения параметра *i* отличаются на единицу.

В системе *Mathematica* имеется несколько условных операторов, которые позволяют ветвить вычисления в зависимости от выполнения одного или нескольких условий. Оператор **If[test, expr1, expr2]** аналогичен условному оператору **If** в других языках программирования. Он вычисляет выражение *expr1* или *expr2*, если условие *test* принимает значения **True** или **False** соответственно. Если же условие *test* не может быть проверено, то ни одно из этих выражений не вычисляется. В такой ситуации можно добавить выражение *expr3* в качестве четвертого аргумента функции **If**. Именно это выражение будет вычисляться, если условие *test* не выдает ни **True**, ни **False**. Пример:

```
In[147]:= {If[2 < 3, expr1, expr2, expr3],
           If[5 < 3, expr1, expr2, expr3],
           If[x < 3, expr1, expr2, expr3]}
```

```
Out[147]= {expr1, expr2, expr3}
```

Заметим, что оператор **If** может вызываться только с двумя аргументами: **If[test, expr]**. В этом случае выражение *expr* вычисляется только в том случае, если условие *test* принимает значение **True**.

Функция **Which[test1, expr1, test2, expr2, ...]** позволяет сделать выбор

между большим количеством альтернативных вариантов вычисления. При обращении к ней сначала проверяется условие *test1*. Если оно выдает **True**, вычисляется выражение *expr1*, и на этом работа функции **Which** оканчивается. Если же условие *test1* выдает **False**, проверяется следующее условие *test2*, и в случае положительного ответа вычисляется *expr2*. В случае отрицательного ответа проверяется условие *test3*, и т.д. Пример:

```
In[148]:= f1[x_] := Which[x < 0, -x, x ≥ 0 && x < 2, x, x ≥ 2, 2];
          {f1[-1], f1[1], f1[5]}

Out[149]= {1, 1, 2}
```

Заметим, что второе условие состоит из двух частей, соединенных двумя символами **&&**, набираемыми без пробела. Эти символы обозначают логическое условие "**и**", что соответствует одновременному выполнению обоих неравенств. Логическое "**или**" обозначается двойной вертикальной линией " || " и также может использоваться при записи условий.

Функция **Switch[expr, form1, value1, form2, value2, ...]** также выполняет роль переключателя. Она последовательно сравнивает форму выражения *expr* с шаблонами *form1*, *form2* и т.д. и при первом же совпадении формы с шаблоном $form_i$ выдает соответствующее значение $value_i$. На этом ее работа заканчивается. Пример:

```
In[150]:= f1[x_] := Switch[ x, _Integer, 1,
                  _Rational, 2, _Real, 3, _Complex, 4, _, x]

In[151]:= {f1[1], f1[2 / 3], f1[0.8], f1[2 i], f1[a]}

Out[151]= {1, 2, 3, 4, a}
```

Чтобы учесть возможность несовпадения формы *expr* ни с одним из шаблонов, в определении функции *f1(x)* мы добавили в качестве последних двух аргументов шаблон общего вида (символ подчеркивания) и аргумент функции *x* соответственно.

Как мы уже видели выше, количество вычислений выражения *expr* с помощью оператора **Do** определяется его итератором. Используя условные операторы, можно прервать выполнение этой циклической процедуры или пропустить в ней некоторые циклы. Для этого в системе *Mathematica* имеется ряд команд, которые работают аналогично соответствующим командам в других языках программирования. Команда **Break[ ]** вызывает немедленное прекращение работы циклического оператора и переход к следующему выражению программы. Пример:

```
In[152]:= Do[ If[k > 3, Break[] ]; Print[k], {k, 2, 10}]
   From In[152]:=
      2
   From In[152]:=
      3
```

Команда **Return[expr]** также прерывает работу циклического оператора и выдает выражение *expr*.

```
In[153]:= Do[ If[k > 4, Return[k] ], {k, 2, 10}]

Out[153]= 5
```

С помощью команды **Goto[mark]** можно немедленно перейти к выполнению того оператора в программе, перед которым установлена метка **Label[mark]**. Пример:

```
In[154]:= (Do[If[k^2 > 10, Goto[mark1]], {k, 7}]; Print["Break"];
           Label[mark1]; Print["k^2>10"])

From In[154]:=
   k^2>10
```

Следует помнить, однако, что использовать команду **Goto** можно только в пределах одной и той же процедуры. Именно поэтому мы заключили все выражение в круглые скобки. Если скобки убрать, то получится ряд независимых выражений, и команда **Goto[mark]** не сможет найти метку *mark*.

Отметим еще одну команду **Continue[ ]**, которая позволяет сразу перейти к следующему шагу при выполнении циклической операции. Пример:

```
In[155]:= Do[ If[k < 4, Continue[] ]; Print[k], {k, 2, 5}]

From In[155]:=
   4

From In[155]:=
   5
```

Как видим, только после нарушения условия $k < 4$ команда **Continue[ ]**, которая является вторым аргументом оператора **If,** перестает работать, и последующие значения *k* выводятся на экран.

Часть II

Применение системы *Mathematica* к решению физических задач

Лабораторная работа № 1
Численные расчеты в системе *Mathematica*

Цель работы:
1) познакомиться с системой Mathematica и изучить особенности создаваемых в ней документов;
2) изучить основные особенности выполнения численных расчетов;
3) изучить встроенные функции для численного решения алгебраических и дифференциальных уравнений.

I. Загрузка системы, создание документа, сохранение результатов работы

1. С помощью мыши щелкните иконку *Mathematica* в меню "Программы". После загрузки системы на экране появляются два окна, одно из которых является основным и имеет заголовок **Mathematica 5.1 - [Untitled-1]**, где **5.1** соответствует номеру используемой версии системы, а выражение **Untitled-1** обозначает, что имя текущего документа еще не определено. Это окно размещается в верхней части экрана и содержит главное меню системы со стандартным набором разделов: **File**, **Edit**, **Cell**, **Format**, **Input**, **Kernel**, **Find**, **Window**, **Help**. Второе окно имеет заголовок **Untitled-1** и представляет собой бланк нового документа, в котором следует вводить свои данные. Обратите внимание на тонкую горизонтальную линию в верхней части рабочего поля этого окна. При вводе любого символа с помощью клавиатуры или мыши он будет изображаться под этой линией.

2. Введите какое-либо простое выражение, например,

```
2 + 2
```

Обратите внимание, что с появлением первого символа горизонтальная линия исчезает и появляется курсор в виде мигающего вертикального отрезка, который показывает место, в котором будет изображаться следующий вводимый символ. Кроме того, вводимые данные автоматически выделяются квадратной скобкой в правой части рабочего поля окна и составляют одну секцию или ячейку (**cell**) документа. Напомним, что секция является наименьшей частью данных, которые обрабатываются ядром системы *Mathematica*. Для обработки секции (т.е. выполнения содержащихся

в ней команд и вычислений) достаточно выделить ее, подведя указатель мыши к]-скобке и щелкнув левой кнопкой мыши (при этом скобка выделяется темным фоном), или просто поместить курсор в любое место этой секции и нажать клавиши: **Shift+Enter** (удерживая клавишу **Shift**, нажимаем клавишу **Enter**). В качестве альтернативы комбинации **Shift+Enter** можно использовать клавишу **Enter** на цифровой части клавиатуры.

3. Вычислите сумму **2+2**. Обратите внимание, что результат вычислений помещается в отдельную секцию, а у введенного выражения и полученного результата появляются метки *In[1]:=* и *Out[1]=*. Такие метки будут появляться при обработке каждого последующего выражения, причем число в квадратных скобках будет увеличиваться на единицу. Две секции, содержащие команду, вводимую пользователем, и получаемый результат, объединяются в группу с помощью еще одной]-скобки. Под этой группой секций появляется горизонтальная линия, показывающая, что вводимые символы следующего выражения будут изображаться под ней в новой секции.

Документ, получаемый в системе *Mathematica*, состоит из набора секций, каждая из которых имеет свой тип или стиль (**Style**), определяющий ее роль в документе. Например, секция, содержащая команды или выражения, предназначенные для обработки ядром, имеет тип **Input**. Результаты вычислений помещаются в секции, имеющие тип **Output**. Текст, поясняющий вычисления, можно поместить в отдельную секцию, которая имеет тип **Text** и не обрабатывается ядром. Секция, содержащая заголовок документа, имеет тип **Title**. Для определения подзаголовков имеются секции типа **Subtitle**, и **Subsubtitle**, а для выделения названий глав и параграфов – секции типа **Section**, **Subsection** и **Subsubsection**. В зависимости от стиля всего документа, который определяется опцией **Style Sheet** в разделе **Format** главного меню (по умолчанию каждый вновь создаваемый документ имеет стиль **Default**), количество различных типов секций может изменяться.

По умолчанию каждая новая секция в любом документе имеет тип **Input**. Для изменения типа секции необходимо выделить ее и затем с помощью мыши выбрать нужный тип в подразделе **Style** раздела **Format** главного меню (щелкнуть левой кнопкой мыши, поместив ее указатель на соответствующее название). Для удобства работы с документом рекомендуется вывести дополнительный ряд кнопок и выпадающее меню, в котором указывается тип текущей секции. Для этого необходимо выбрать опцию **Show ToolBar** в разделе **Format** главного меню. Тогда для изменения типа секции достаточно выделить ее и щелкнуть левой кнопкой мыши, поместив ее указатель на соответствующее название в выпадающем меню. Кроме того, выбирая в разделе **Format** опцию **Show Ruler**, мы помещаем в верхней части документа масштабную линейку, на которой отмечены границы рабочего поля окна и правая граница страницы при печати документа.

4. Перемещая мышь по коврику, обратите внимание на то, как изменяется форма указателя мыши в зависимости от его положения. Если указатель мыши находится в пределах какой-либо секции, то он имеет вид вертикального отрезка. Щелкнув в этот момент левой кнопкой мыши, мы перемещаем курсор в соответствующую секцию и можем редактировать введенное ранее выражение или продолжить ввод данных. Если же указатель мыши находится между секциями, то он принимает вид горизонтального отрезка. Если щелкнуть левой кнопкой мыши в этот момент, то между секциями появится тонкая горизонтальная линия, которая показывает, что следующее выражение будет вводиться под ней в новой секции. Следует

отметить, что точно так же будет изменяться форма курсора при его перемещении с помощью стрелок на клавиатуре.

5. Добавьте в свой документ название и номер лабораторной работы в качестве заголовка. Для этого переместите указатель мыши к верхней границе документа и щелкните ее левой кнопкой. В результате над первой секцией должна появиться горизонтальная линия. Это означает, что с началом ввода заголовка в верхней части документа откроется новая секция. Введите в этой секции текст:

Лабораторная работа № 1

Численные расчеты в системе Mathematica

После ввода заголовка установите для этой секции тип **Title**. При этом размер букв заголовка изменится. Если вы хотите изменить тип шрифта или размер букв в каком-либо выражении или во всей секции, с помощью мыши выделите это выражение или секцию и затем установите нужный шрифт или размер, выбирая последовательно соответствующие опции в разделе **Format** главного меню, например: **Format** ↦ **Font** ↦ **Arial** (выбираем шрифт **Arial**) или **Format** ↦ **Size** ↦ **18 Point** (устанавливаем размер символов 18 пикселей). Таким же образом можно изменить цвет какого-либо выражения (опция **Text Color**), выделить его с помощью фона (опция **Background Color**) и т.д.

6. В новой секции в качестве подзаголовка (тип **Subtitle**) добавьте фамилию и инициалы студента, выполняющего работу, и номер группы, например:

Выполнил: И.И.Иванов, гр. Т-55

Затем в следующей секции введите текст "Арифметические вычисления" и установите для нее тип **Section**. Все вычисления и пояснения, относящиеся к первому заданию, теперь нужно вводить ниже в последующих секциях, устанавливая для них подходящий стиль. Аналогично оформляются остальные задания.

7. Сохраните свой документ на диске, выбрав команду **Save** или **Save As** в разделе **File** главного меню. При этом открывается дополнительное окно, в котором нужно указать директорию или папку, куда следует поместить соответстующий файл, и имя файла (в строке **File name**). По умолчанию сохраняемый файл будет иметь тип *Mathematica* Notebook и расширение **.nb**. После сохранения имя файла будет автоматически внесено в заголовок окна вместо слова **Untitled-1**.

II. Арифметические вычисления

1. В отдельных секциях наберите последовательно следующие простые арифметические выражения, обращая внимание на дополнительные пробелы и точки в них, и произведите вычисления.

```
3 * 5

3 5

3   5

2 +   7
```

```
2 / 3
17^(1 / 2)
3 /7.
17.^(1 / 2)
```

Объясните полученные результаты и выполните следующее задание.

Задание 1

На основе проделанных выше вычислений подберите примеры, подтверждающие справедливость следующих утверждений:
а) Пробел выполняет роль знака умножения.
б) Введение дополнительных пробелов не изменяет результата вычислений.
в) Целые, рациональные и иррациональные числа считаются заданными точно. Поэтому в выражениях, содержащих только такие числа, при обработке производятся лишь возможные упрощения, но сами они остаются точными и преобразуются в числа, заданные приближенно, только по команде пользователя.
г) Действительные числа, содержащие десятичную точку, считаются заданными приближенно. Поэтому выражения, в которые входят действительные числа, вычисляются при обработке секции без специальной команды.
д) Для изменения стандартного порядка выполнения операций используются круглые скобки.

В отчете приведите примеры вместе с вычислениями и пояснениями.

2. Для вычисления приближенного значения числа или арифметического выражения в системе *Mathematica* используется функция **N**. Одним из способов получить информацию об этой функции (а также о любой встроенной функции) является следующий. После знака вопроса набираем имя функции и нажимаем клавиши **Shift+Enter**.

```
In[6]:= ? N

  N[expr] gives the numerical value of expr. N[expr, n]
     attempts to give a result with n-digit precision. More...
```

В результате получаем сообщение, что **N[expr]** выдает численное значение выражения **expr**, а **N[expr, n]** пытается получить численный результат с точностью до **n** значащих цифр. Таким образом, команда для вычисления $\sqrt{5}$, например, имеет вид:

```
In[7]:= N[Sqrt[5]]
Out[7]= 2.23607
```

Для вычисления квадратного корня используется функция **Sqrt[x]**. Чтобы выяснить, с какой точностью произведено вычисление, применим к полученному результату функцию **Precision[x]**, которая показывает, с точностью до скольких значащих цифр получен результат.

```
In[8]:= Precision[%]
Out[8]= MachinePrecision
```

Обратите внимание на аргумент у функции **Precision**. Напомним, что в системе *Mathematica* знак процента используется для ссылки на полученные ранее результаты. Так, один знак % означает последний полученный результат, два знака %% – предпоследний и т.д.. При этом знак %*n* соответствует результату, полученному в секции с меткой *Out[n]*.

Запомните правило: "**Имена всех встроенных функций в системе *Mathematica* пишутся с заглавной буквы, а их аргумент заключается в квадратные скобки**".

Задание 2

а) С помощью функции **N** вычислите $\sqrt{17}$ с точностью до 200 значащих цифр. Проверьте точность полученного результата с помощью функции **Precision**.

б) Найдите приближенное значение числа $e^{\pi\sqrt{163}}$ с точностью до 40 значащих цифр. Вычислите, на сколько полученный результат отличается от ближайшего к нему целого числа. Для стандартного округления чисел в системе *Mathematica* используется функция **Round[x]**.

в) Вычислите:

$$\left(10\times\left(\frac{10.8\times10^3}{300}\right)^{1/2}+4\right)^{1/3}; \qquad \frac{10^{-24}\times10^{12}}{10^{-14}}\sqrt{\frac{32768}{2^9}}.$$

г) Последовательно введите выражения

```
5 > 3
5 < 2
```

и посмотрите, что выдает *Mathematica* при их обработке. Затем определите, справедливо или нет неравенство: $\pi^e > e^\pi$. Найдите численные значения левой и правой частей неравенства и убедитесь в правильности полученного результата.

Замечание. Известные постоянные π и e в системе *Mathematica* обозначаются соответственно **Pi** и **E**. Кроме того, предусмотрен дополнительный ряд панелей с кнопками, позволяющими записать эти постоянные, буквы греческого алфавита и многие математические выражения в традиционном виде. Если с помощью мыши выбрать последовательно следующие опции в разделе **File** главного меню **File** ↦ **Palettes** ↦ **BasicInput**, например, то открывается дополнительное окно, в котором содержится ряд кнопок с греческими буквами и шаблонами для набора символов с верхним и нижним индексами, дробей, интегралов, сумм и т.д. При этом для основания натуральных логарифмов e используется обозначение вида $\mathbb{e}$, для мнимой единицы – i , а соответствующие строчные буквы латинского алфавита являются обычными символами, которые могут использоваться по усмотрению пользователя.

III. Численное решение уравнений

1. Для численного решения полиномиального уравнения или системы уравнений в системе *Mathematica* используется функция **NSolve**. Чтобы выяснить правила обращения к этой функции, воспользуемся справкой.

```
In[9]:= ? NSolve

NSolve[lhs==rhs, var] gives a list of numerical
   approximations to the roots of a polynomial equation.
   NSolve[{eqn1, eqn2, ... }, {var1, var2, ... }]
   solves a system of polynomial equations. More...
```

Как видим, у функции **NSolve** должно быть два аргумента, разделенных запятой. Первый аргумент представляет собой уравнение или список уравнений, заключенных в фигурные скобки и разделенных запятыми, а второй – имя переменной или список имен, относительно которых необходимо разрешить уравнение или систему. Запомните, что признаком уравнения в системе *Mathematica* является двойной знак равенства "==", причем символы "=" набираются без пробела. В качестве примера найдем корни уравнения $x^3 - 3x^2 + 1 = 0$.

```
In[10]:= NSolve[x^3 - 3 x^2 + 1 == 0, x]

Out[10]= {{x → -0.532089}, {x → 0.652704}, {x → 2.87939}}
```

Функция **NSolve** выдает список из трех корней, которые представлены в виде правил подстановки, т.е. конструкций вида: **имя → значение**, причем каждое решение заключено в фигурные скобки. Найденные корни могут быть легко подставлены в любое выражение. Пример:

```
In[11]:= x^3 - 3 x^2 /. %[[2]]

Out[11]= -1.
```

Напомним, что два символа "/" и ".", набираемые один за другим без пробела, обозначают оператор подстановки, после которого должно следовать правило замены или список правил замены. Напомним также, что знак % обозначает последний полученный результат. В данном случае этот результат представляет собой список из трех решений уравнения. Чтобы выделить второй элемент списка, сразу после знака процента в двойных квадратных скобках набирается его номер, т.е. 2.

Задание 3

а) Решить уравнения:

$$\sqrt{x+2} + 4x = 4; \qquad x^3 - 3x^2 + 5 = 0.$$

б) Решить системы уравнений:

$$\begin{cases} x^2 + xy + y^2 = 1, \\ x^3 + x^2 y + xy^2 + y^3 = 4; \end{cases}$$

$$\begin{cases} x - y^2 + 3z = 34, \\ x + y - z = -1, \\ -x + 2y + 3z = 4; \end{cases}$$

в) Придумайте уравнение или систему полиномиальных уравнений и найдите ее численное решение.

2. Для решения произвольных уравнений и их систем используется функция **FindRoot**, которая находит только одно численное решение при каждом обращении к ней. Правила обращения легко видеть из следующего сообщения.

```
In[12]:= ? FindRoot

  FindRoot[lhs==rhs, {x, x0}] searches for
     a numerical solution to the equation lhs==rhs,
     starting with x=x0. FindRoot[{eqn1, eqn2, ... },
     {{x, x0}, {y, y0}, ... }] searches for a numerical
     solution to the simultaneous equations eqni. More...
```

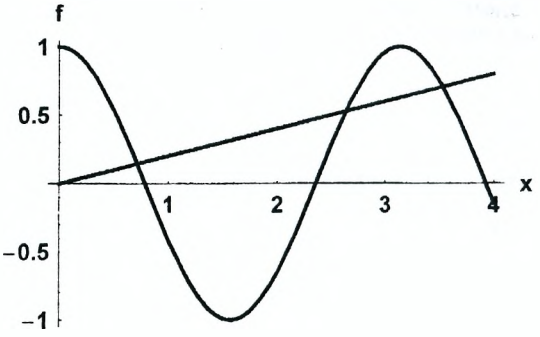
Подчеркнем, что при обращении к функции **FindRoot** необходимо указать приближенное значение корня, с которого начинается поиск его точного значения. Найти это приближенное значение проще всего графически. В качестве примера найдем корни уравнения

$$\cos 2x = 0.2x$$

на отрезке $x \in [0, 4]$.

Решение. Чтобы найти приближенные значения корней, построим графики левой и правой частей уравнения.

```
In[13]:= Plot[{Cos[2 x], 0.2 x}, {x, 0, 4},
           PlotStyle →
         {{Thickness[0.009], RGBColor[0, 0, 1]},
          {Thickness[0.009], RGBColor[1, 0, 0]}},
           AxesLabel → {"x", "f"}];
```



Как видим, на заданном отрезке есть три корня (значения x, соответствующие точкам пересечения кривой $y = \cos 2x$ и прямой $y = 0.2x$). Из графика находим приближенные значения корней.

```
In[14]:= x1 = .8; x2 = 2.6; x3 = 3.5;
```

Точные значения корней находим с помощью функции **FindRoot**.

```
In[15]:= FindRoot[Cos[2 x] == 0.2 x, {x, x1}]
Out[15]= {x → 0.713776}

In[16]:= FindRoot[Cos[2 x] == 0.2 x, {x, x2}]
Out[16]= {x → 2.63356}

In[17]:= FindRoot[Cos[2 x] == 0.2 x, {x, x3}]
Out[17]= {x → 3.53445}
```

Задание 4

Используя функцию **FindRoot**, найти решения следующих уравнений:

1. $x^4 - 5x + 2 = \exp(x - 1)$ на отрезке $x \in [0, 2]$.
2. $x^4 - 8x^2 + 3 = 2x^3$ на отрезке $x \in [-2, 1]$.
3. $\sqrt{x^4 - 5x + 5} = 3x^2$ на отрезке $x \in [-2, 2]$.
4. $x^5 + x^4 - 2x^3 - 5x - 2 = x^3$ на отрезке $x \in [-1, 2]$.
5. $e^x = 3\cos x$ на отрезке $x \in [-2, 2]$.
6. $x^2 + e^{-x} - 2 = \sin(2x)$ на отрезке $x \in [-1, 2]$.
7. $\ln x + 10 = \mathrm{tg}(2x)$ на отрезке $x \in [0, 3]$.
8. $8\cos(x) = x$ на отрезке $x \in [-2, 6]$.
9. $x^3 - 7x^2 + 14x - 8 = \ln x$ на отрезке $x \in [0, 5]$.
10. $\mathrm{sh}(x) = x^3 - 5x^2 + 2x + 8$ на отрезке $x \in [-2, 2]$.
11. $2\sin(x) + 1 = x^2 - 2x - 3$ на отрезке $x \in [-2, 4]$.
12. $e^{-x} = \sin(2x)$ на отрезке $x \in [0, 4]$.
13. $\mathrm{ch}\, x - 2 = \sin(x)$ на отрезке $x \in [-1, 2]$.
14. $\sin(x^2) = \cos(2x)$ на отрезке $x \in [-1, 3]$.
15. $2\cos^2(x) = \exp(x/2)$ на отрезке $x \in [-3, 1]$.
16. $2\cos(x) = \exp(-2x)$ на отрезке $x \in [-1, 2]$.
17. $3\cos(x - 1) = 2\,\mathrm{Sin}^2(x)$ на отрезке $x \in [-1, 3]$.
18. $\mathrm{ch}(x + 1) = \cos(x) + 1$ на отрезке $x \in [-2, 1]$.
19. $\mathrm{ch}(x + 1) = x^3 - 5x^2 + 3$ на отрезке $x \in [-1, 2]$.
20. $x^5 - 2x^2 - 3x + 5 = \exp(x)$ на отрезке $x \in [-2, 2]$.
21. $2x^4 - 3x^3 + 4x - 1 = \mathrm{th}(x)$ на отрезке $x \in [-2, 1]$.
22. $2x^3 - 14x + 4 = 2\ln(x + 1)$ на отрезке $x \in [0, 3]$.
23. $2x^2 - 9x - 5 = 7\,\mathrm{th}(x + 1)$ на отрезке $x \in [-2, 6]$.
24. $\sin(x^2 + 1) = \mathrm{ch}(x - 1) - 2$ на отрезке $x \in [-2, 3]$.
25. $\sin(2x) = x/8$ на отрезке $x \in [0., 2\pi]$.

IV. Численное решение дифференциальных уравнений

1. Для численного решения дифференциальных уравнений и их систем имеется функция **NDSolve**, которая вызывается по крайней мере с тремя аргументами: **NDSolve[eqns, y, {x, xmin, xmax}]**, где **eqns** – список, который включает дифференциальное уравнение и необходимое число граничных или начальных условий, **y** – имя функции, относительно которой решается уравнение, **x** – имя независимой переменной, причем решение ищется на

отрезке от **xmin** до **xmax**. В качестве примера найдем решение дифференциального уравнения

$$y'' - 2y' + 5xy = 0$$

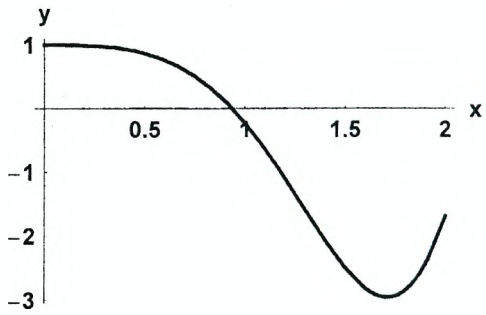
с начальными условиями $y(0) = 1$, $y'(0) = 0$ на отрезке $x \in [0, 2]$ и назовем его **sol1**.

```
In[18]:= sol1 = NDSolve[{y''[x] - 2 y'[x] + 5 x y[x] == 0,
           y[0] == 1, y'[0] == 0}, y, {x, 0, 2}]
Out[18]= {{y → InterpolatingFunction[{{0., 2.}}, <>]}}
```

Обратите внимание на наличие двойного знака равенства как в самом дифференциальном уравнении, так и в уравнениях, выражающих начальные условия. Это общее правило записи всех уравнений. Кроме того, функция и все ее производные должны быть записаны с аргументом, заключенным в квадратные скобки: **y[x]**, **y'[x]** и т.д. Для производных можно использовать стандартные обозначения, набирая после **y** необходимое число штрихов (для обозначения второй производной нельзя использовать кавычки). Решение получается в виде правила подстановки для искомой функции $y(x)$, в правой части которого находится интерполяционная функция.

Чтобы найти корни уравнения $y(x) = 0$, например, построим график найденной функции.

```
In[19]:= Plot[y[x] /. sol1, {x, 0, 2}, AxesLabel → {"x", "y"},
           PlotStyle → {RGBColor[0, 0, 1], Thickness[0.009]}];
```



Из графика видим, что на отрезке [0, 2] есть один корень уравнения $y(x) = 0$, приближенное значение корня $x = 0.9$.Точное значение находим с помощью функции **FindRoot**.

```
In[20]:= sol2 = FindRoot[(y[x] /. sol1[[1]]) == 0, {x, 0.9}]
Out[20]= {x → 0.935004}
```

Вычисляем значение производной в найденной точке.

```
In[21]:= D[y[x] /. sol1[[1]], x] /. sol2
Out[21]= -3.34414
```

Найденная функция имеет около точки $x = 1.7$ локальный минимум. Приравнивая производную этой функции к нулю и решая полученное уравнение, находим точное значение x.

```
In[22]:= sol3 = FindRoot[D[y[x] /. sol1[[1]], x] == 0, {x, 1.7}]

Out[22]= {x → 1.71173}
```

Соответствующее значение функции равно:

```
In[23]:= y[x] /. sol1[[1]] /. sol3

Out[23]= -2.95436
```

Минимальное значение функции $y(x)$ вместе с соответствующим значением x в окрестности точки $x = 1.7$ можно найти с помощью функции **FindMinimum**.

```
In[24]:= FindMinimum[y[x] /. sol1[[1]], {x, 1.7}]

Out[24]= {-2.95436, {x → 1.71173}}
```

Как видим, полученные значения совпадают с найденными ранее.

Задание 5

Найдите решение одного из следующих дифференциальных уравнений на заданном отрезке.

1. $y'' + 2y'\cos(x) + 5y = 0$, $y(0) = 0$, $y'(0) = 1$ на отрезке $x \in [0, 5]$.
2. $y'' + x^2 y - 1 = 0$, $y(0) = 0$, $y'(0) = 1$ на отрезке $x \in [0, 4]$.
3. $y'' + \cos(x)\, y + x^2 = 0$, $y(0) = 1$, $y'(0) = 1$ на отрезке $x \in [-2, 3]$.
4. $y'' + \cos(x)\, y' - x^2 = 0$, $y(0) = 1$, $y'(0) = 2$ на отрезке $x \in [-5, 2]$.
5. $y'' + x\, y - \cos(2x) = 0$, $y(0) = 0$, $y'(0) = 2$ на отрезке $x \in [0, 5]$.
6. $y'' + \mathrm{tg}(x)\, y' - \sin(x) = 0$, $y(0) = -1$, $y'(0) = 2$ на отрезке $x \in [0, 9]$.
7. $y'' + 3x^2 y' - 6\cos(2x)\, y^2 = 0$, $y(0) = -4$, $y'(0) = 3$ на отрезке $x \in [0, 5]$.
8. $y'' - x^2 y' + 5y^3 = 0$, $y(0) = 2$, $y'(0) = -1$ на отрезке $x \in [-2, 1]$.
9. $y'' + x^2 y' + 3\cos^2(x)\, y = 0$, $y(0) = 1$, $y'(0) = -1$ на отрезке $x \in [-2, 4]$.
10. $y'' + (1 + \cos x)\, y = 0$, $y(0) = 0$, $y'(0) = 1$ на отрезке $x \in [-4, 4]$.
11. $y'' + (5 + \cos^2(x))\, y = 0$, $y(0) = 1$, $y'(0) = 0$ на отрезке $x \in [0, 3]$.
12. $y'' + 0.5y' + (1 + \cos(x))\, y = 0$, $y(0) = 0$, $y'(0) = -1$ на отрезке $x \in [0, 8]$.
13. $y'' - 2y' + 3x\, y^3 = 0$, $y(0) = 0$, $y'(0) = 1$ на отрезке $x \in [0, 2]$.
14. $y'' + 7y^3 - 15y - 3x = 0$, $y(0) = 0$, $y'(0) = -1$ на отрезке $x \in [0, 4]$.
15. $y'' - 5x\, y^3 = 0$, $y(0) = -1$, $y'(0) = 1$ на отрезке $x \in [-3, 2]$.
16. $y'' + 2y'\,\mathrm{th}(x) + 3y = 0$, $y(0) = -1$, $y'(0) = 2$ на отрезке $x \in [-3, 2]$.
17. $y'' - y' + 2y\cos^2(x) = 0$, $y(0) = 1$, $y'(0) = -2$ на отрезке $x \in [-1, 5]$.
18. $y'' - y'\cos^2(x) + 12y = 0$, $y(0) = 1$, $y'(0) = 3$ на отрезке $x \in [0, 3]$.
19. $y'' + 2y' + (5x + \cos^2(x))\, y = 0$, $y(0) = 1$, $y'(0) = 1$ на отрезке $x \in [0, 3]$.
20. $y'' + y'\,\mathrm{tg}(x) + 3y = 0$, $y(0) = 0$, $y'(0) = 2$ на отрезке $x \in [0, 5]$.
21. $y'' - 4xy' + (4x^2 + 3x + 4)\, y = 0$, $y(0) = -10$, $y'(0) = -5$ на отрезке $x \in [-1, 2]$.
22. $y'' + 3y'\cos(x - 1) - (x^2 - 3)\, y = 0$, $y(0) = -1$, $y'(0) = 1$ на отрезке $x \in [-3, 3]$.
23. $y'' + 3xy' + (4x^2 + 3)\, y = 0$, $y(0) = -1$, $y'(0) = 2$ на отрезке $x \in [-2, 3]$.
24. $y'' - xy' + (x + 1)\, y = 0$, $y(0) = 1$, $y'(0) = -3$ на отрезке $x \in [-1, 3]$.
25. $y'' - y' + \sin^2(x)\, y = 0$, $y(0) = -1$, $y'(0) = 2$ на отрезке $x \in [-8, 2]$.

Используя найденное решение,
а) найдите на заданном отрезке корни уравнения $y(x) = 0$;
б) найдите значения производной $y'(x)$ в точках, где $y(x) = 0$;
в) выберите одну из точек, в которых производная функции $y(x)$ равна нулю, и вычислите значение функции в этой точке;
г) с помощью функции **FindMinimum** найдите точки, в которых функция $y(x)$ имеет локальный минимум или максимум, и сравните полученные результаты со значениями, найденными в пункте в).

V. Обработка численных данных

1. Предположим, что в ходе эксперимента производится ряд измерений некоторой величины y при заданных значениях x. Например, измеряется координата тела в заданные моменты времени. В результате получается таблица численных данных. Чтобы смоделировать такую ситуацию, рассмотрим, например, функцию вида:

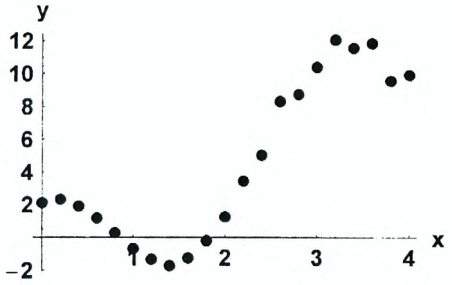
$$y = -2 + 3x + 4\cos(2x).$$

Вычислим значения этой функции, задавая значения x из интервала $0 \le x \le 4$ с шагом 0.2. Поскольку в реальном эксперименте измерения производятся с некоторой погрешностью, в каждой точке x введем случайное возмущение $|\Delta y| < 0.1|y|$, генерируемое с помощью функции **Random**. Используя функцию **Table**, получаем список, каждый элемент которого является списком из двух соответствующих значений (x, y).

```
In[25]:= dat1 = Table[{x, (-2 + 3 x + 4 Cos[2 x])
                (1 + Random[Real, {-0.1, 0.1}])}, {x, 0, 4, 0.2}]

Out[25]= {{0, 2.12834}, {0.2, 2.32993}, {0.4, 1.92634},
          {0.6, 1.19582}, {0.8, 0.286804}, {1., -0.665702},
          {1.2, -1.3162}, {1.4, -1.70866}, {1.6, -1.23162},
          {1.8, -0.203953}, {2., 1.26946}, {2.2, 3.4598},
          {2.4, 5.07342}, {2.6, 8.34072}, {2.8, 8.76947},
          {3., 10.4274}, {3.2, 12.1107}, {3.4, 11.6169},
          {3.6, 11.9038}, {3.8, 9.60057}, {4., 9.9647}}
```

Напомним, что в реальном эксперименте мы не знаем численные значения коэффициентов зависимости $y = y(x)$, а имеем только таблицу численных данных. Чтобы представить себе эту зависимость, отметим экспериментальные точки на координатной плоскости xOy. Для этого используем функцию **ListPlot**.

```
In[26]:= p1 = ListPlot[dat1, PlotStyle → PointSize[0.03],
            AxesLabel → {"x", "y"}];
```



Чтобы найти наилучшие, с точки зрения метода наименьших квадратов, значения коэффициентов *a*, *b*, *c* в выражении $y = a + b\,x + c\cos(2\,x)$, используем функцию **Fit**. При обращении к ней нужно указать три аргумента. Первым аргументом является список численных данных, в данном случае *dat1*. Второй аргумент представляет собой список функций, линейная комбинация которых должна аппроксимировать искомую зависимость. Третий аргумент указывает имя независимой переменной. Соответствующая команда имеет вид:

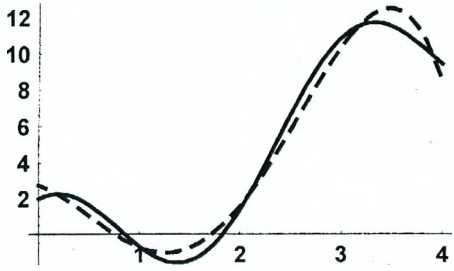
```
In[27]:= y1 = Fit[dat1, {1, x, Cos[2 x]}, x]

Out[27]= -2.03563 + 3.01166 x + 4.02323 Cos[2 x]
```

Как видим, найденные значения коэффициентов *a*, *b*, *c* близки к тем, которые мы использовали при генерации численных данных. Для сравнения попробуем провести через эксперимертальные точки наилучшую кривую, уравнение которой представляет собой, например, полином четвертой степени. Команда для поиска такого полинома имеет вид:

```
In[28]:= y2 = Fit[dat1, {1, x, x^2, x^3, x^4}, x]

Out[28]= 2.74496 - 1.70841 x - 5.59515 x^2 + 4.5576 x^3 - 0.740496 x^4
```

Теперь можно построить графики найденных функций.

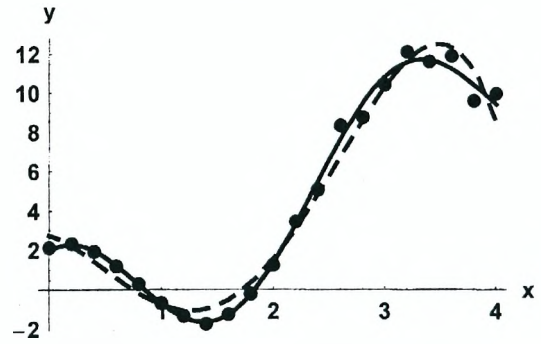
```
In[31]:= p2 = Plot[{y1, y2}, {x, 0, 4},
            PlotStyle → {{Thickness[0.01]},
              {Thickness[0.01], Dashing[{0.03}]}}];
```



Чтобы посмотреть, насколько хорошо найденные кривые проходят через

экспериментальные точки, совместим два графика, используя функцию **Show**.

In[32]:= `Show[p1, p2];`



Как видим, сплошная кривая, полученная при использовании реальной зависимости $y = y(x)$, лучше соответствует экспериментальным результатам, чем полиномиальная зависимость (кривая, изображаемая штриховой линией).

Задание 6

а) Задайте произвольные численные значения параметров *a*, *b*, *c*, *d*, *e* в следующих зависимостях:

$$y(x) = a + b\,x + c\,x^2 + d\,x^3 + e\,x^4,$$

$$y(x) = a + b\,x + c\sin x + d\cos x,$$

$$y(x) = a + b\,x + c\sin x + d\cos(2x),$$

$$y(x) = a\sin x + b\cos x + c\sin(2x) + d\cos(2x).$$

Используя функцию **Table**, определите список численных данных, каждый элемент которого содержит два соответствующих элемента *(x, y)*. Область изменения *x* и шаг выберите самостоятельно. С помощью функции **Random** внесите в каждой точке *x* случайное возмущение Δy.

б) Изобразите полученные точки на координатной плоскости.

в) С помощью функции **Fit** найдите наилучшую кривую, аппроксимирующую численные данные.

г) На одной координатной плоскости изобразите найденную кривую вместе с экспериментальными точками.

VI. Окончание работы

Для окончания работы с системой *Mathematica* достаточно с помощью мыши выбрать опцию **Exit** в разделе **File** главного меню. При следующем сеансе работы с системой *Mathematica* сохраненный файл может быть открыт путем выбора опции **Open** в разделе **File** главного меню для дальнейшей работы с ним.

Лабораторная работа № 2
Символьные вычисления в системе *MATHEMATICA*

Цель работы:

1) изучить основные встроенные функции для преобразования символьных выражений в системе Mathematica;

2) изучить правила дифференцирования и интегрирования выражений, а также основные функции для решения алгебраических и дифференциальных уравнений в символьном виде;

3) определить положение центра масс и вычислить моменты инерции тонкой пластинки относительно трех взаимно перпендикулярных осей.

Задание 1

1. Придумайте и определите символьное выражение, содержащее не менее трех слагаемых, причем хотя бы одно из них должно быть рациональным выражением. Пример:

$$In[33]:= t1 = \frac{6\,(x^3-8)\,(x^2-y^2)}{(2x^2-8)\,(x-y)} + (x+y)^2 + x^3 + y^3;$$

Последовательно примените к нему функции **Expand**, **ExpandAll**, **Factor**, **Together**, **Apart**, **Cancel**, **Simplify** и объясните их действие. Выражение *t1* должно быть таким, чтобы можно было проследить работу каждой их этих функций. Например, функция **Expand** раскрывает скобки в выражении и представляет результат в виде суммы.

```
In[34]:= Expand[t1]
```

$$Out[34]= x^2 + x^3 - \frac{48x^2}{(-8+2x^2)\,(x-y)} + \frac{6x^5}{(-8+2x^2)\,(x-y)} + 2xy + y^2 + \frac{48y^2}{(-8+2x^2)\,(x-y)} - \frac{6x^3y^2}{(-8+2x^2)\,(x-y)} + y^3$$

Каждое их трех слагаемых в *t1* содержит множитель $(x+y)$, который можно вынести за скобки и представить выражение в виде произведения множителей. Такую операцию выполняет функция **Factor**.

```
In[35]:= Factor[t1]
```

$$Out[35]= \frac{(x+y)\,(12+8x+6x^2+x^3+2y-xy-x^2y+2y^2+xy^2)}{2+x}$$

2. Приведите выражение *t1* к общему знаменателю и с помощью функции **Numerator** определите числитель полученного выражения, обозначив его *t2*. С помощью функции **Collect** представьте выражение *t2* в виде полинома по степеням *x* или *y*. Определите максимальную степень *n* одной из переменных, например, *x* в выражении *t2* и коэффициент при x^n. Используйте функции **Exponent** и **Coefficient**.

Задание 2

1. С помощью функции **D** вычислите производную функции $x^n \cos x$.
2. Вычислите следующие производные:

$$\frac{\partial}{\partial x}(a x^3 + 2 x^2); \quad \frac{\partial^3}{\partial x^3}\left(4 x^2 - 5 x + 8 - \frac{3}{x}\right);$$

$$\frac{\partial^5}{\partial x^3 \partial y^2}(x^3 y^2 + 3 x^2)$$

3. Используя функцию **Integrate**, вычислите интеграл

$$\int \frac{dx}{x^2 - 1}$$

4. Вычислите интегралы, затем продифференцируйте полученные выражения и убедитесь в том, что получаемые результаты совпадают с подынтегральными выражениями.

$$\int \frac{x^3}{x^2 + 1}\,dx \; ; \quad \int \frac{1}{x^3 + 1}\,dx \; .$$

При необходимости используйте функции для преобразования выражений.

5. Вычислите определенный интеграл:

$$\int_1^4 \left(5x - 2\sqrt{x} + \frac{32}{x^3}\right) dx \; .$$

6. Вычислите интегралы:

$$\int_0^1 (1 + x^4)^{1/3}\,dx; \quad \int_0^{1.} (1 + x^4)^{1/3}\,dx \; .$$

Обратите внимание, что в первом случае в качестве верхнего предела стоит целое число 1, а во втором – действительное число 1. (обратите внимание на точку после цифры 1). Объясните полученные результаты.

Задание 3

1. Используя функцию **Solve**, решить уравнение: $a x^4 + x^2 + 3 = 0$.
2. Найти точное решение системы уравнений:

$$\begin{cases} x^2 + y = 1 \\ y^2 - x^2 = 2 \end{cases}$$

3. Решить систему уравнений:

$$\begin{cases} x^2 - y^2 = 1 \\ y^3 + x = 5 \end{cases}$$

Если точное решение системы найти не удается, получите численное решение.

4. Найдите общие решения дифференциальных уравнений, используя функцию **DSolve**.

$y'(x) - y(x)\,\mathrm{tg}\,x = x$; $\quad y'(x) + y(x)\,\mathrm{tg}\,x = 1/\cos x$.

Самостоятельно задайте граничные условия и найдите соответствующие им частные решения. Постройте их графики.

5. Самостоятельно выбрав граничные условия, найдите решение дифференциального уравнения в символьном виде:

$$y''(x) - y'(x) + x\,y(x) = 0.$$

Затем с помощью функции **NDSolve** найдите численное решение этого уравнения. Интервал изменения x выберите самостоятельно.

На одной координатной плоскости постройте графики численного и символьного решений и покажите, что они совпадают.

Пример 1

Пусть тонкая пластинка массой m находится в плоскости xOy и ограничена кривыми $y = x^2 + x - 1$, $y = -x^2 + 3x + 4$ и прямой $y = 3 - x/2$. Требуется:

1. Найти координаты центра масс пластинки $(x_c,\ y_c)$.

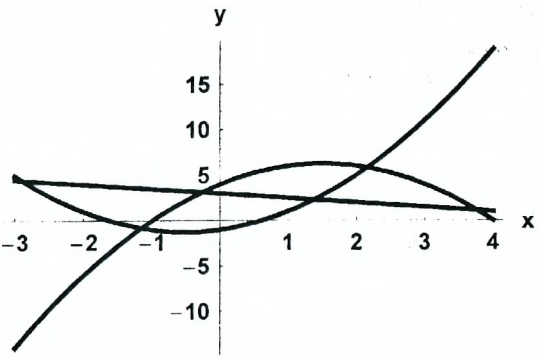
2. Изобразить пластинку на плоскости xOy и отметить положение ее центра масс.

3. Вычислить моменты инерции пластинки относительно осей Ox, Oy и Oz и показать, что справедливо соотношение:

$$I_x + I_y = I_z.$$

Решение. Чтобы представить себе форму пластинки, построим на координатной плоскости xOy графики всех заданных функций. Интервал изменения координаты x выбираем так, чтобы на рисунке были видны все точки пересечения кривых. Как обычно, начиная решение новой задачи, с помощью функции **Clear** удаляем все введенные ранее определения.

```
In[36]:= Clear["Global`*"];

In[37]:= Plot[{x^2 + x - 1, -x^2 + 3 x + 4, -x / 2 + 3}, {x, -3, 4},
           PlotStyle → Thickness[0.01],
         AxesLabel → {"x", "y"}];
```



Как видим, на плоскости *xOy* можно выделить несколько фигур, ограниченных заданными кривыми. Далее будем рассматривать фигуру, ограниченную снизу параболой $y = x^2 + x - 1$, а сверху – прямой $y = 3 - x/2$ в ее левой части и параболой $y = -x^2 + 3x + 4$ – в правой.

Чтобы определить соответствующие области изменения координаты *x* для каждого участка, найдем координаты точек пересечения кривых. Сначала зададим их уравнения.

In[38]:=
```
eq1 = y == x^2 + x - 1;
eq2 = y == -x^2 + 3 x + 4;
eq3 = y == -x / 2 + 3;
```

Очевидно, крайняя левая точка фигуры является точкой пересечения параболы $y = x^2 + x - 1$ и прямой $y = 3 - x/2$. Координаты этой точки находим как решение системы уравнений *eq1* и *eq3*.

In[41]:=
```
sol1 = Solve[{eq1, eq3}, {x, y}]
```

Out[41]=
$$\left\{\left\{y \to \frac{1}{8}\left(27 - \sqrt{73}\right),\ x \to \frac{1}{4}\left(-3 + \sqrt{73}\right)\right\},\right.$$
$$\left.\left\{y \to \frac{1}{8}\left(27 + \sqrt{73}\right),\ x \to \frac{1}{4}\left(-3 - \sqrt{73}\right)\right\}\right\}$$

Так как имеются две точки пересечения параболы и прямой, получили два решения. Интересующей нас точке соответствует отрицательное значение координаты *x*, т.е. второе решение системы. Поэтому первое граничное значение *x* задаем в виде:

In[42]:=
```
x1 = x /. sol1[[2]]
```

Out[42]=
$$\frac{1}{4}\left(-3 - \sqrt{73}\right)$$

Вторая граничная точка определяется из условия пересечения прямой $y = 3 - x/2$ и параболы $y = -x^2 + 3x + 4$. Решая соответствующую систему, находим:

In[43]:=
```
sol2 = Solve[{eq2, eq3}, {x, y}]
```

Out[43]=
$$\left\{\left\{y \to \frac{1}{8}\left(17 - \sqrt{65}\right),\ x \to \frac{1}{4}\left(7 + \sqrt{65}\right)\right\},\right.$$
$$\left.\left\{y \to \frac{1}{8}\left(17 + \sqrt{65}\right),\ x \to \frac{1}{4}\left(7 - \sqrt{65}\right)\right\}\right\}$$

Поскольку интересующая нас фигура располагается выше параболы $y = x^2 + x - 1$, в качестве второй граничной точки выбираем второй корень из *sol2*.

In[44]:=
```
x2 = x /. sol2[[2]]
```

Out[44]=
$$\frac{1}{4}\left(7 - \sqrt{65}\right)$$

Третья граничная точка является точкой пересечения двух парабол. Ее координаты находим, решая систему уравнений *eq1* и *eq2*.

In[45]:=
```
sol3 = Solve[{eq1, eq2}, {x, y}]
```

Out[45]=
$$\left\{\left\{y \to \frac{1}{2}\left(5 - 2\sqrt{11}\right),\ x \to \frac{1}{2}\left(1 - \sqrt{11}\right)\right\},\right.$$
$$\left.\left\{y \to \frac{1}{2}\left(5 + 2\sqrt{11}\right),\ x \to \frac{1}{2}\left(1 + \sqrt{11}\right)\right\}\right\}$$

Крайней правой точке рассматриваемой фигуры соответствует второй корень в *sol3*.

In[46]:=
```
x3 = x /. sol3[[2]]
```

Out[46]=
$$\frac{1}{2}\left(1 + \sqrt{11}\right)$$

Итак, при изменении координаты x в пределах отрезка $x_1 \le x \le x_2$ пластинка ограничивается параболой $y = x^2 + x - 1$ и прямой $y = 3 - x/2$, т.е. координата y точек пластинки изменяется в пределах

$$x^2 + x - 1 \le y \le 3 - x/2,$$

а при $x_2 \le x \le x_3$ имеем:

$$x^2 + x - 1 \le y \le -x^2 + 3x + 4,$$

т.е. соответствующие точки находятся между двумя параболами.

Разбиваем пластинку на маленькие кусочки площадью $dx\,dy$. Для определения площади всей пластинки нужно сложить площади всех кусочков, т.е. вычислить следующий двойной интеграл (обратите внимание на пределы интегрирования):

In[47]:=
```
S = Integrate[1, {x, x1, x2}, {y, x^2 + x - 1, 3 - x / 2}] +
    Integrate[1, {x, x2, x3},
     {y, x^2 + x - 1, -x^2 + 3 x + 4}] // Simplify
```

Out[47]=
$$\frac{1}{96}\left(60 + 176\sqrt{11} + 65\sqrt{65} + 73\sqrt{73}\right)$$

Так как координаты граничных точек заданы точно, для площади мы также получили точное выражение.

Положение центра масс определяется следующими формулами:

$$x_c = \frac{1}{m}\sum_{i=1}^{n} m_i x_i,\ \ y_c = \frac{1}{m}\sum_{i=1}^{n} m_i y_i.$$

Предполагая, что пластинка однородна, считаем, что масса каждого кусочка пропорциональна его площади, т.е. $m_i \to \frac{m}{S}\,\mathbf{dx\,dy}$. Тогда координата x центра масс пластинки равна:

```
In[48]:= xc = 1/m (Integrate[m/S x,
             {x, x1, x2}, {y, x^2 + x - 1, 3 - x/2}] +
           Integrate[m/S x, {x, x2, x3},
             {y, x^2 + x - 1, -x^2 + 3 x + 4}]) // Simplify
```

$$Out[48]= \frac{-4442 + 352\sqrt{11} + 455\sqrt{65} - 219\sqrt{73}}{4\left(60 + 176\sqrt{11} + 65\sqrt{65} + 73\sqrt{73}\right)}$$

Аналогично находим координату y_c.

```
In[49]:= yc = 1/m (Integrate[m/S y,
             {x, x1, x2}, {y, x^2 + x - 1, 3 - x/2}] +
           Integrate[m/S y, {x, x2, x3},
             {y, x^2 + x - 1, -x^2 + 3 x + 4}]) // Simplify
```

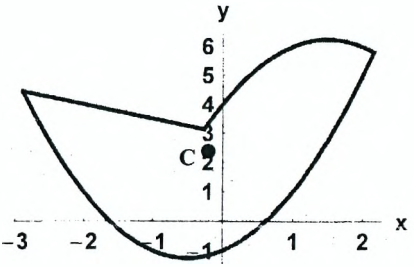
$$Out[49]= \frac{-2320 + 8800\sqrt{11} + 4875\sqrt{65} + 2263\sqrt{73}}{20\left(60 + 176\sqrt{11} + 65\sqrt{65} + 73\sqrt{73}\right)}$$

Чтобы изобразить пластинку и отметить на ней положение центра масс, генерируем три отдельных графика. На первом построим параболу $y = x^2 + x - 1$, на втором – прямую $y = 3 - x/2$, а на третьем – параболу $y = -x^2 + 3x + 4$. Необходимость построения трех графиков связана с тем, что пределы изменения записанных выше функций различны.

```
In[50]:= p1 = Plot[x^2 + x - 1, {x, x1, x2},
             PlotStyle → {Thickness[0.012], RGBColor[1, 0, 0]},
             DisplayFunction → Identity];

In[51]:= p2 = Plot[3 - x/2, {x, x1, x2},
             PlotStyle → {Thickness[0.012], RGBColor[1, 0, 0]},
             DisplayFunction → Identity];

In[52]:= p3 = Plot[-x^2 + 3 x + 4, {x, x2, x3},
             PlotStyle → {Thickness[0.012], RGBColor[1, 0, 0]},
             DisplayFunction → Identity];
```

Поскольку для опции **DisplayFunction** мы установили значение **Identity**, ни один из графиков не изображается. Теперь с помощью функции **Show** мы изобразим все три графика на одной координатной плоскости и укажем расположение центра масс **C**. Чтобы график был показан, у опции **DisplayFunction** следует установить значение **$DisplayFunction**.

```
In[53]:= Show[p1, p2, p3, AxesLabel → {"x", "y"},
          Epilog → {{PointSize[0.04],
             RGBColor[0, 0, 1], Point[{xc, yc}]},
            Text[FontForm["C", {"Times-Bold", 12}],
                                  {xc - 0.3, yc - 0.2}]},
          DisplayFunction → $DisplayFunction];
```



Момент инерции пластинки относительно оси определяется формулой:

$$I = \sum_{i=1}^{n} m_i r_i^2,$$

где r_i – расстояние от i-ого кусочка до оси. Для оси Ox это расстояние равно y, для оси Oy – x, а для оси Oz – $\sqrt{x^2 + y^2}$. В результате получаем:

```
In[54]:= Ix =
          Integrate[m/S y^2, {x, x1, x2}, {y, x^2 + x - 1, 3 - x/2}] +
            Integrate[m/S y^2, {x, x2, x3},
              {y, x^2 + x - 1, -x^2 + 3 x + 4}] // Simplify
```

$$Out[54]= \frac{\left(-197288 + 401280\sqrt{11} + 224185\sqrt{65} + 66357\sqrt{73}\right) m}{224\left(60 + 176\sqrt{11} + 65\sqrt{65} + 73\sqrt{73}\right)}$$

```
In[55]:= Iy =
          Integrate[m/S x^2, {x, x1, x2}, {y, x^2 + x - 1, 3 - x/2}] +
            Integrate[m/S x^2, {x, x2, x3},
              {y, x^2 + x - 1, -x^2 + 3 x + 4}] // Simplify
```

$$Out[55]= \frac{\left(-33420 + 5632\sqrt{11} + 10075\sqrt{65} + 4307\sqrt{73}\right) m}{40\left(60 + 176\sqrt{11} + 65\sqrt{65} + 73\sqrt{73}\right)}$$

```
In[56]:= Iz = Integrate[m/S (x^2 + y^2),
           {x, x1, x2}, {y, x^2 + x - 1, 3 - x/2}] +
         Integrate[m/S (x^2 + y^2), {x, x2, x3},
           {y, x^2 + x - 1, -x^2 + 3x + 4}] // Simplify
```

$$Out[56]= \frac{\left(-1922200 + 2164096\sqrt{11} + 1403025\sqrt{65} + 452381\sqrt{73}\right) m}{1120\left(60 + 176\sqrt{11} + 65\sqrt{65} + 73\sqrt{73}\right)}$$

Находим сумму $I_x + I_y$.

```
In[57]:= II = Ix + Iy // Simplify
```

$$Out[57]= \frac{\left(-1922200 + 2164096\sqrt{11} + 1403025\sqrt{65} + 452381\sqrt{73}\right) m}{1120\left(60 + 176\sqrt{11} + 65\sqrt{65} + 73\sqrt{73}\right)}$$

Чтобы сравнить полученное выражение с I_z, вычисляем разность $I_x + I_y - I_z$.

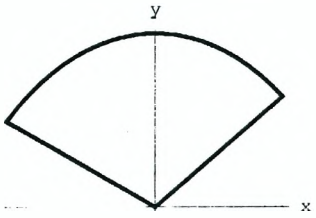
```
In[58]:= II - Iz // Simplify
Out[58]= 0
```

Таким образом, справедливость соотношения $I_x + I_y = I_z$ доказана, причем все вычисления выполнены точно.

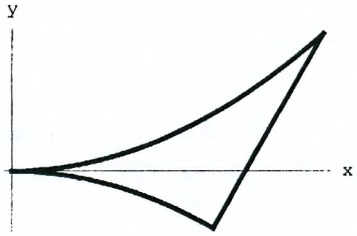
Задание 4

Выполните пункты 1, 2 и 3 из примера 1 в следующих случаях:

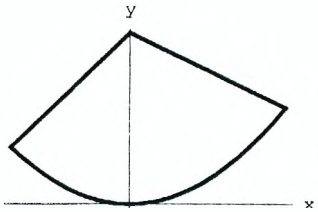
1. Пластинка ограничена кривой $x^2 + (y+1)^2 = 3$ и прямыми $y = x/2$, $y = -x/3$



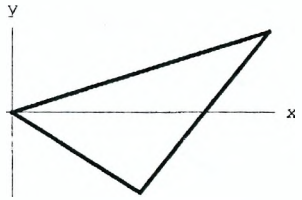
2. Пластинка ограничена кривыми $y = x^2$, $y = -x^2$ и прямой $y = 4x - 3$.



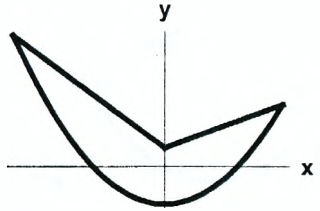
3. Пластинка ограничена кривой $y = x^2$ и прямыми $y = 2x + 3$, $y = -x + 3$.



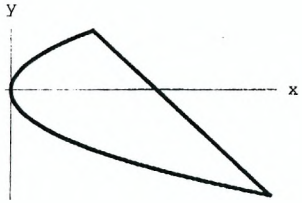
4. Пластинка ограничена прямыми $y = x/2$, $y = -x$, $y = 2x - 3$.



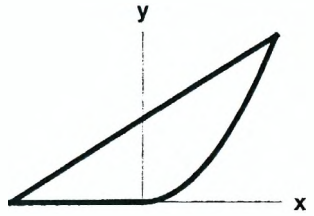
5. Пластинка ограничена кривой $y = x^2 - 2$ и прямыми $y = -2x + 1$, $y = x + 1$.



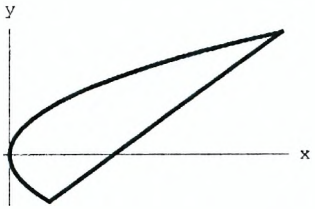
6. Пластинка ограничена кривой $y^2 = 2x$ и прямой $y = -3x + 2$.



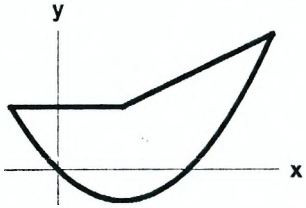
7. Пластинка ограничена прямыми $y = x + 2$, $y = 0$ и кривой $y = x^2$.



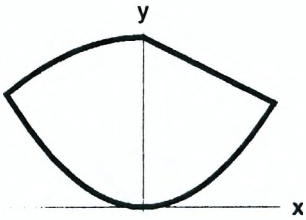
8. Пластинка ограничена кривой $y^2 = x$ и прямой $y = x - 1$.



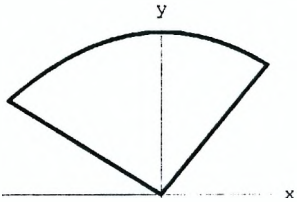
9. Пластинка ограничена кривой $y = x^2 - 2x$ и прямыми $y = x + 1$, $y = 2$.



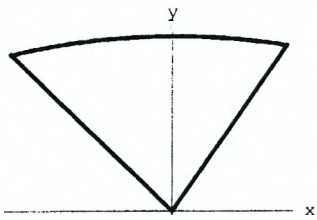
10. Пластинка ограничена кривыми $y = 2x^2$, $y = 2 - x^2$ и прямой $y = 2 - x$.



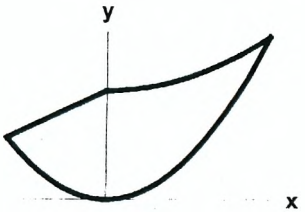
11. Пластинка ограничена кривой $y = 5 - x^2/4$ и прямыми $y = -x$, $y = 2x$.



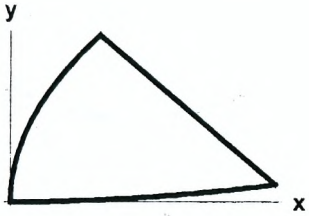
12. Пластинка ограничена кривой $x^2 + y^2 = 5$ и прямыми $y = 3x$, $y = -2x$.



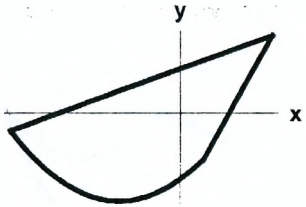
13. Пластинка ограничена кривыми $y = 3x^2$, $y = x^2 + 1$ и прямой $y = x + 1$.



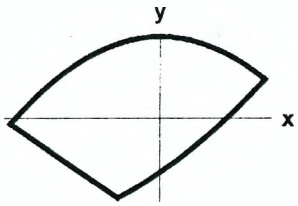
14. Пластинка ограничена кривыми $y = x^2/3$, $y^2 = 3x$ и прямой $y = -2x + 1$.



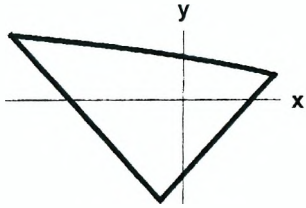
15. Пластинка ограничена кривой $y = x^2 + 2x - 3$ и прямыми $y = x + 2$, $y = 5x - 4$.



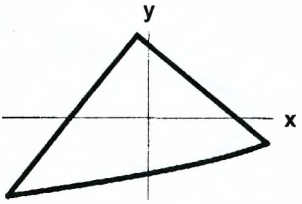
16. Пластинка ограничена кривыми $y = x^2 + 3x - 2$, $y = -2x^2 + 3$ и прямой $y = -3x - 4$.



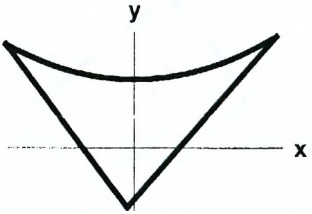
17. Пластинка ограничена кривой $y^2 = 3 - x$ и прямыми $y = 2x - 3$, $y = -2x - 5$.



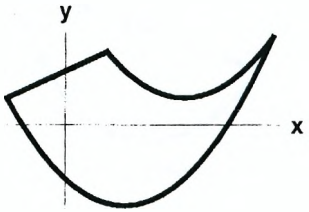
18. Пластинка ограничена кривой $y^2 = 5 - 2x$ и прямыми $y = -2x + 3$, $y = 3x + 4$.



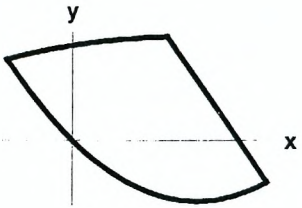
19. Пластинка ограничена кривой $y = x^2 + 4$ и прямыми $y = -7x - 4$, $y = 6x + 3$



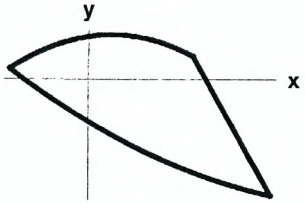
20. Пластинка ограничена кривыми $y = x^2 - 4x + 5$, $y = x^2 - 2x - 2$ и прямой $y = x + 2$.



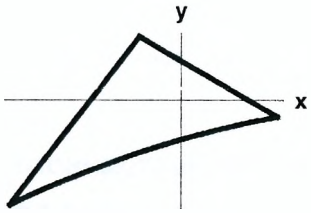
21. Пластинка ограничена кривыми $2y^2 + (x - 1)^2 = 6$, $y = x^2 - 2x$ и прямой $y = -3x + 4$.



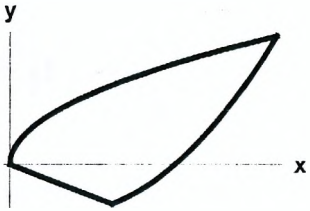
22. Пластинка ограничена кривыми $y = x^2 - 4x - 2$, $y = -3x^2 + x + 2$ и прямой $y = -12x + 10$.



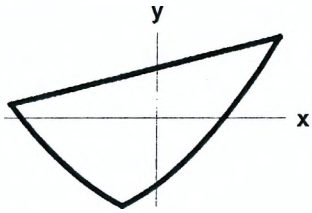
23. Пластинка ограничена кривой $y = -x^2 + 2x - 1$ и прямыми $y = -4x + 1$, $y = 9x + 3$.



24. Пластинка ограничена кривыми $y^2 = 4x$, $y = x^2 - x - 1$ и прямой $y = -x$.



25. Пластинка ограничена кривыми $y = x^2 - x - 6$, $y = x^2 + 3x - 4$ и прямой $y = x + 3$.



Лабораторная работа № 3
Анализ электрических цепей постоянного и переменного тока

Цель работы:

1) определить силы тока во всех участках разветвленной электрической цепи постоянного тока, используя правила Кирхгофа;

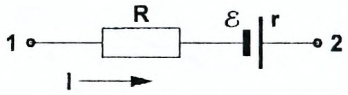
2) записать дифференциальные уравнения для определения силы тока во всех участках простейшей цепи переменного тока и решить их;

3) найти силы тока во всех участках цепи переменного тока методом комплексных амплитуд;

4) построить амплитудно-частотную и фазово-частотную характеристики цепи.

I. Электрические цепи постоянного тока

Рассмотрим участок цепи, состоящий из резистора сопротивлением R и гальванического элемента с ЭДС $\mathcal{E}$ и внутренним сопротивлением r. Стрелкой на рисунке обозначено направление тока I.



Закон Ома для участка цепи:

$$\pm I\,(R + r) = \varphi_1 - \varphi_2 \pm \mathcal{E}.$$

Знак "+" в левой части закона Ома соответствует направлению тока от точки 1 к точке 2, знак "–" – противоположному направлению тока. ЭДС считается положительной (знак "+" в правой части закона Ома), если, идя вдоль цепи из точки 1 в точку 2, мы переходим от отрицательного полюса элемента к положительному, и отрицательной (знак "–" в правой части закона Ома) – в противоположном случае.

Первое правило Кирхгофа: в каждом узле электрической цепи (точка, в которой сходится более двух проводников) алгебраическая сумма токов равна нулю:

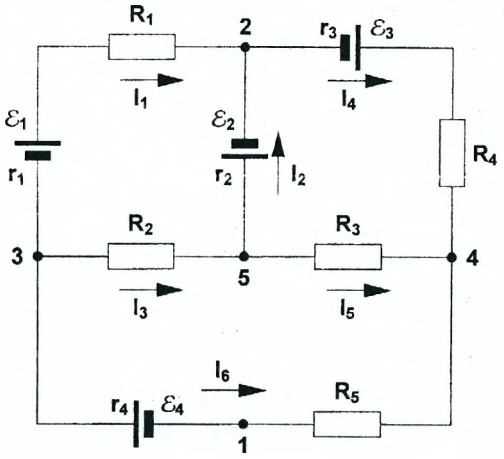
$$\sum I_i = 0\,.$$

Второе правило Кирхгофа: для любого замкнутого контура сумма произведений сил тока I_i в отдельных участках контура на их полные сопротивления $(R_i + r_i)$ равна сумме действующих в этом контуре ЭДС:

$$\sum I_i\,(R_i + r_i) = \sum \mathcal{E}_i\,.$$

Пример 1

Рассмотрим разветвленную электрическую цепь, изображенную на рисунке.



Обозначим токи в различных участках цепи через I_1, I_2, I_3, I_4, I_5, I_6 и укажем предполагаемые направления токов стрелками. Так как на данной схеме имеются четыре узла, для получения полной системы уравнений необходимо записать первое правило Кирхгофа для любых трех узлов и второе правило Кирхгофа для трех элементарных контуров. В результате получим систему уравнений *eq*.

In[5]:=
```
Clear["Global`*"];
eq = { i1 + i2 - i4 == 0,
       i3 - i2 - i5 == 0,
       i4 + i5 + i6 == 0,
       i1 (R1 + r1) - i2 r2 - i3 R2 == ℰ1 + ℰ2,
       i4 (R4 + r3) + i2 r2 - i5 R3 == ℰ3 - ℰ2,
       i3 R2 + i5 R3 - i6 (R5 + r4) == ℰ4};
```

В системе уравнений *eq* для обозначения токов в отдельных участках цепи использована строчная буква *i* с нижним индексом. Напомним, что заглавная буква *I* в системе *Mathematica* зарезервирована для обозначения мнимой единицы. Поэтому символ вида I_1, например, будет обозначать мнимую единицу с индексом, смысл которой не понятен. Чтобы не возникли проблемы

при вычислениях, рекомендуется не снабжать зарезервированные символы индексами и использовать их в качестве переменных.

Решение линейной системы может быть легко найдено в символьном виде. Однако, чтобы избежать громоздких символьных преобразований, дальнейшие вычисления проведем в численном виде. Генерируя численные значения параметров цепи с помощью функции **Random**, устанавливаем для них тип **Integer**, чтобы полученные числа были заданы точно. Это позволит в дальнейшем убедиться в точном выполнении некоторых соотношений, характерных для цепей постоянного тока.

```
In[7]:= r1 = Random[ Integer, {1, 5}];
        r2 = Random[ Integer, {1, 5}];
        r3 = Random[ Integer, {1, 5}];
        r4 = Random[ Integer, {1, 5}];
        ℰ1 = Random[ Integer, {1, 10}];
        ℰ2 = Random[ Integer, {1, 10}];
        ℰ3 = Random[ Integer, {1, 10}];
        ℰ4 = Random[ Integer, {1, 10}];
        R1 = Random[ Integer, {1, 20}];
        R2 = Random[ Integer, {1, 20}];
        R3 = Random[ Integer, {1, 20}];
        R4 = Random[ Integer, {1, 20}];
        R5 = Random[ Integer, {1, 20}];
```

Для решения системы используем функцию **Solve**.

```
In[14]:= sol = Solve[eq, {i1, i2, i3, i4, i5, i6}]
```

$$\text{Out[14]}= \left\{\left\{i_1 \to \frac{3269}{9559},\ i_2 \to -\frac{1960}{9559},\ i_3 \to -\frac{2745}{19118},\right.\right.$$
$$\left.\left. i_4 \to \frac{119}{869},\ i_5 \to \frac{1175}{19118},\ i_6 \to -\frac{3793}{19118}\right\}\right\}$$

В результате получаем точные значения силы тока в отдельных участках цепи. Вычислим теперь разность потенциалов между точками 1 и 2 двумя способами: сначала переходим из 1 в 2 через точку 3, а затем – через точки 4 и 5. Применяя закон Ома для каждого из участков цепи, получаем:

```
In[15]:= f1 = -i6 r4 - ℰ4 + i1 (R1 + r1) - ℰ1 /. sol[[1]]
```

$$\text{Out[15]}= \frac{50579}{9559}$$

```
In[16]:= f2 = i6 R5 - i5 R3 + i2 r2 + ℰ2 /. sol[[1]]
```

$$\text{Out[16]}= \frac{50579}{9559}$$

Теперь вычислим количество теплоты, выделяющееся во всех проводниках цепи в единицу времени.

```
In[17]:= q = i1^2 (R1 + r1) + i2^2 r2 + i3^2 R2 +
          i4^2 (R4 + r3) + i5^2 R3 + i6^2 (R5 + r4) /. sol[[1]]
```

$$Out[17]= \frac{67871}{19118}$$

Вычислим суммарную мощность всех источников ЭДС.

```
In[18]:= a = i1 ℰ1 - i2 ℰ2 + i4 ℰ3 - i6 ℰ4 /. sol[[1]]
```

$$Out[18]= \frac{67871}{19118}$$

Если уравнения системы *eq* записаны верно и все вычисления сделаны правильно, то значения f_1 и f_2 должны совпадать. Кроме того, должны совпасть *q* и *a*.

Задание 1

Придумайте и нарисуйте разветвленную электрическую цепь, содержащую не менее 4 узлов и 4 элементарных контуров. Для этой цепи выполните следующие задания:

1. Используя правила Кирхгофа, запишите полную систему уравнений для определения токов во всех участках цепи. Затем задайте численные значения параметров цепи и найдите точные значения токов, используя функцию **Solve**.
2. Выберите на схеме две точки 1 и 2 и вычислите разность потенциалов между ними при найденных значениях токов двумя способами. Убедитесь, что разность потенциалов в обоих случаях одинакова.
3. Проверьте выполнение баланса мощностей в цепи и сделайте вывод.

II. Анализ цепей переменного тока

Для понимания работы цепей переменного тока полезно исследовать зависимость силы тока в цепи от времени при приложении к ней некоторых стандартных сигналов, т.е. напряжений, которые изменяются во времени по заданному закону. Наиболее часто используются ступенчатое и синусоидальное напряжения. Первое особенно удобно для исследования переходных процессов в цепи. Главным достоинством второго является следующая его особенность: *если на вход линейной цепи подается синусоидальный сигнал определенной частоты, то установившиеся значения силы тока и напряжения на отдельных участках цепи, а также напряжение на выходе, являются синусоидальными функциями той же частоты, но с другими амплитудами и фазами.* Это дает возможность легко найти амплитуду и фазу силы тока и напряжения на любом элементе цепи без решения дифференциальных уравнений, используя метод комплексных амплитуд. Суть этого метода сводится к следующему:

1) конденсаторы и катушки индуктивности рассматриваются как резисторы с комплексными сопротивлениями $\frac{1}{i\omega C}$ и $i\omega L$ соответственно;

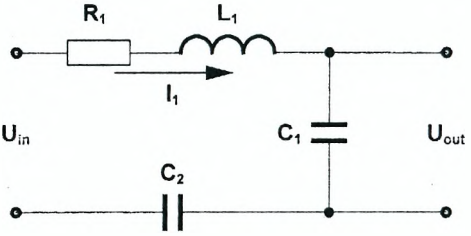
2) силы тока в отдельных участках цепи считаются постоянными комплексными числами;

3) входное синусоидальное напряжение $U_{in}(t) = A_0 \sin(\omega t)$ заменяется

постоянным напряжением величиной A_0;
4) система уравнений для определения комплексных амплитуд токов записывается на основе закона Ома и правил Кирхгофа так же, как и для цепей постоянного тока;
5) амплитуда и начальная фаза силы тока или напряжения на любом участке цепи определяются как модуль и аргумент соответствующей комплексной амплитуды, а их зависимость от времени находится как мнимая часть произведения комплексной амплитуды и функции $\mathrm{Exp}(i\,\omega\,t)$.

Пример 2

Рассмотрим электрическую цепь, изображенную на рисунке.



Обозначим силу тока в цепи через I_1, а напряжения на конденсаторах – U_{C1} и U_{C2}. Будем считать напряжение на конденсаторе C_1 положительным, когда заряд его верхней обкладки положителен. Тогда сила тока в цепи и напряжение U_{C1} связаны соотношением: $I_1 = C_1 \frac{dU_{C1}}{dt}$. Учитывая, что сила тока во всех элементах цепи одинакова, запишем систему уравнений, определяющую I_1, U_{C1}, U_{C2}, в виде:

```
In[19]:= Clear["Global`*"];
         eq1 = {R1 I1[t] + Uc1[t] + Uc2[t] == Uin - L1 I1'[t],
           I1[t] == C1 Uc1'[t], C1 Uc1'[t] == C2 Uc2'[t]};
```

Первое уравнение системы *eq1* представляет собой закон Ома для участка цепи, а второе и третье выражают связь между силой тока и напряжениями на конденсаторах. Напомним, что ЭДС самоиндукции, возникающая в катушке, равна: $\mathcal{E}_{ind} = -L_1 \frac{dI_1}{dt}$, где L_1 – индуктивность катушки. Считая, что в начальный момент времени сила тока в цепи и напряжения на конденсаторах равны нулю, задаем начальные условия в виде:

```
In[21]:= initial = {Uc1[0] == 0, Uc2[0] == 0, I1[0] == 0};
```

Найдем решение системы *eq1* в случае, когда $U_{in} = 1$. Система *eq1* содержит линейные дифференциальные уравнения и легко решается с помощью функции **DSolve**.

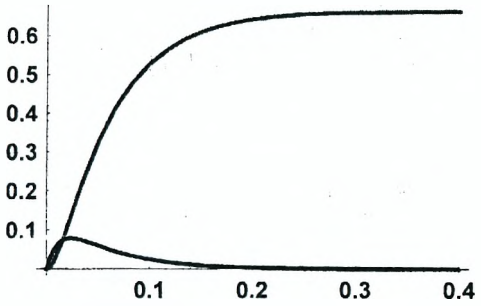
```
In[22]:= sol1 = DSolve[Join[eq1 /. Uin → 1, initial],
            {Uc1[t], Uc2[t], I1[t]}, t] // Simplify
```

Однако получаемое решение является довольно громоздким, и поэтому мы его не приводим. Чтобы построить график зависимости выходного напряжения и силы тока от времени, зададим параметры цепи.

In[23]:=
```
dat1 = {C1 → 0.01, C2 → 0.02, R1 → 10, L1 → 0.1};
```

Тогда найденные зависимости имеют вид:

In[24]:=
```
Plot[{Uc1[t] /. sol1[[1]] /. dat1,
    I1[t] /. sol1[[1]] /. dat1}, {t, 0, 0.4},
  PlotStyle → {{Thickness[0.01], RGBColor[1, 0, 0]},
    {Thickness[0.01], RGBColor[0, 0, 1]}}];
```



Как видим, обе функции с течением времени стремятся к постоянным значениям. Это означает, что при выбранных значениях параметров цепи переходные процессы в ней практически завершаются к моменту времени $t = 0.3$. Теперь приложим к цепи синусоидальное напряжение вида $U_{in}(t) = A_0 \sin(\omega t)$, выбрав частоту ω так, чтобы за время переходных процессов совершалось около пяти колебаний входного напряжения. Учитывая соотношение $\omega = \frac{2\pi}{T}$, где T – период колебаний, выбираем следующие значения параметров входного напряжения:

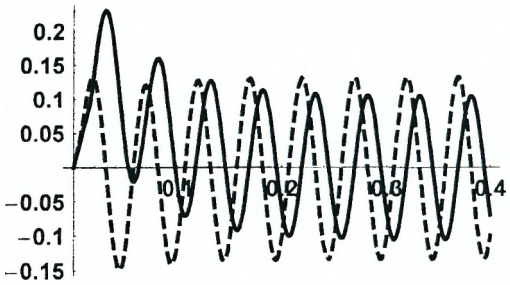
In[25]:=
```
dat2 = {A0 → 2, ω → 40 π};
```

Хотя решение системы *eq1* может быть найдено и в символьном виде, оно будет очень громоздким. Поэтому найдем ее численное решение, используя функцию **NDSolve**. Интервал изменения времени выберем несколько большим, чем время затухания переходных процессов в цепи, например, $t \in [0, 1]$. Напомним, что численное решение системы *eq1* будет найдено только в том случае, если значения всех параметров в ней заданы численно.

In[26]:=
```
sol2 =
  NDSolve[Join[eq1 /. Uin → A0 Sin[ω t] /. dat1 /. dat2,
    initial], {Uc1[t], Uc2[t], I1[t]}, {t, 0, 1}];
```

Построим графики зависимости выходного напряжения и силы тока в цепи от времени.

In[27]:=
```
Plot[{Uc1[t] /. sol2[[1]],
    I1[t] /. sol2[[1]]}, {t, 0, 0.4},
  PlotStyle → {{Thickness[0.01]},
    {Thickness[0.01], Dashing[{0.02}]}}];
```



Из графиков видно, что после завершения переходных процессов сила тока в цепи и выходное напряжение изменяются по синусоидальному закону. По такому же закону происходят колебания напряжения на всех элементах цепи, различными являются только амплитуды и начальные фазы колебаний. И если нас интересуют только установившиеся значения амплитуд силы тока и выходного напряжения и их начальные фазы, для их определения можно воспользоваться методом комплексных амплитуд. Для этого запишем закон Ома в комплексной форме, учитывая, что в цепи имеются резистор, катушка индуктивности и два конденсатора, соединенные последовательно.

In[28]:=
$$\text{eq3} = \text{A0} == \text{I1}\left(\text{R1} + \text{i}\,\omega\,\text{L1} + \frac{1}{\text{i}\,\omega\,\text{C1}} + \frac{1}{\text{i}\,\omega\,\text{C2}}\right);$$

Тогда комплексная амплитуда силы тока *I1* находится как решение уравнения *eq3*.

In[29]:=
```
sol3 = Solve[eq3, I1][[1]] // Simplify
```

Out[29]=
$$\left\{\text{I1} \to \frac{\text{i A0 C1 C2}\,\omega}{\text{C2} + \text{C1}\,(1 + \text{C2}\,\omega\,(\text{i R1} - \text{L1}\,\omega))}\right\}$$

Комплексную амплитуду выходного напряжения запишем в виде:

In[30]:=
$$\text{uc} = \frac{1}{\text{i}\,\omega\,\text{C1}}\ \text{I1}\ /.\ \text{sol3}\ //\ \text{Simplify}$$

Out[30]=
$$\frac{\text{A0 C2}}{\text{C2} + \text{C1}\,(1 + \text{C2}\,\omega\,(\text{i R1} - \text{L1}\,\omega))}$$

Зависимость *Uout(t)* получим, умножая амплитуду *uc* на $e^{i\omega t}$ и выделяя мнимую часть полученного выражения:

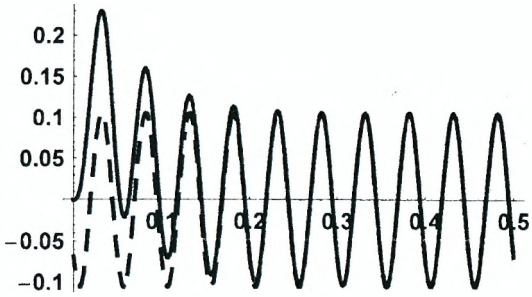
In[31]:=
```
Uout[t_] =
    Im[uc Exp[i ω t] // ExpToTrig ] // ComplexExpand //
     FullSimplify[#, {{C1, C2, R1, L1, ω} ∈ Reals}] &
```

Out[31]=
$$\begin{aligned}&(\text{A0 C2}\\&\quad(-\text{C1 C2 R1}\,\omega\,\text{Cos}[\text{t}\,\omega] + (\text{C1} + \text{C2} - \text{C1 C2 L1}\,\omega^2)\,\text{Sin}[\text{t}\,\omega]))\ /\\&((\text{C1} + \text{C2})^2 + \text{C1 C2}\,(-2\,(\text{C1} + \text{C2})\,\text{L1} + \text{C1 C2 R1}^2)\,\omega^2 +\\&\quad \text{C1}^2\,\text{C2}^2\,\text{L1}^2\,\omega^4)\end{aligned}$$

Теперь построим на одной координатной плоскости два графика зависимости *Uout(t)*. Одна из них найдена как численное решение системы дифферен-

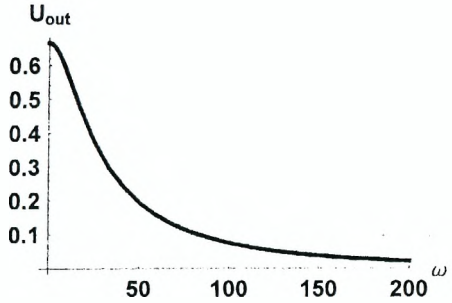
циальных уравнений *eq1*, а вторая – с помощью метода комплексных амплитуд.

```
In[32]:= Plot[{Uc1[t] /. sol2[[1]] , Uout[t] /. dat1 /. dat2},
          {t, 0, 0.5},
          PlotStyle → {{Thickness[0.01], RGBColor[1, 0, 0]},
                       {Thickness[0.01], Dashing[{0.04}]}}];
```



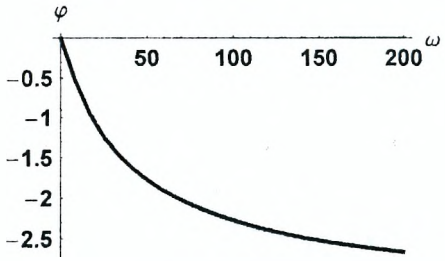
Поскольку второй график изображен штриховой линией, легко видеть, что после завершения переходных процессов в цепи оба графика совпадают. Таким образом, для определения установившегося значения выходного напряжения нет необходимости решать дифференциальные уравнения, а достаточно использовать метод комплексных амплитуд. Кроме того, метод комплексных амплитуд позволяет легко получить зависимость амплитуды выходного напряжения от частоты входного синусоидального напряжения. Для этого достаточно найти модуль комплексной амплитуды *uc* и построить ее график. В результате получаем:

```
In[33]:= Plot[Abs[uc] /. dat1 /. A0 → 1 // Evaluate, {ω, 0, 200},
          PlotStyle → {Thickness[0.011], Hue[0.7]},
          AxesLabel → {"ω", "Uout"}];
```



Полученная зависимость называется *амплитудно-частотной характеристикой цепи*. Аналогично можно получить зависимость разности фаз входного и выходного напряжений как функции частоты ω. Для этого достаточно выделить аргумент комплексного числа *uc* и построить соответствующий график.

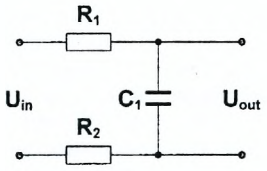
```
In[34]:= Plot[Arg[uc] /. dat1 /. A0 → 1 // Evaluate, {ω, 0, 200},
          PlotStyle → {Thickness[0.011], Hue[0.7]},
          AxesLabel → {"ω", "φ"}];
```



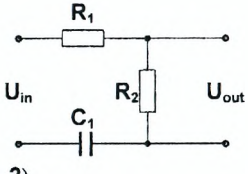
Полученная кривая называется *фазово-частотной характеристикой цепи*

Задание 2

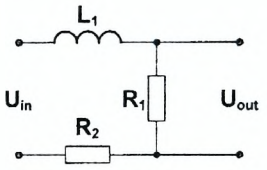
На рисунках изображены схемы электрических цепей.



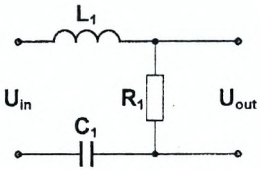
1)



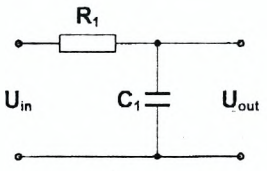
2)



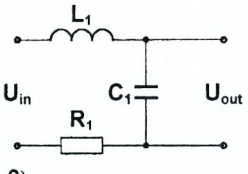
3)



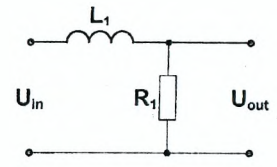
4)



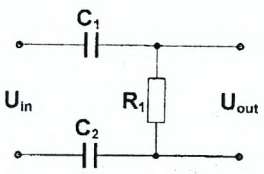
5)



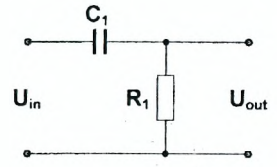
6)



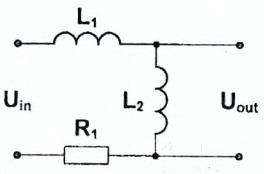
7)



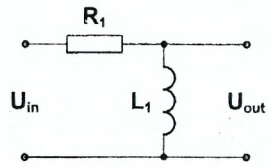
8)



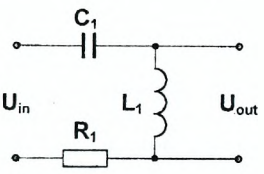
9)



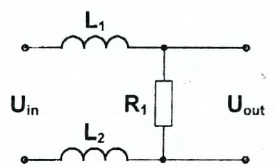
10)



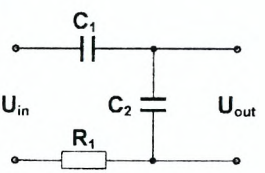
11)



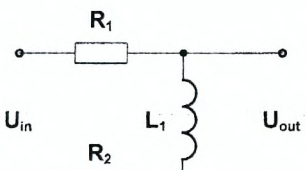
12)



13)



14)



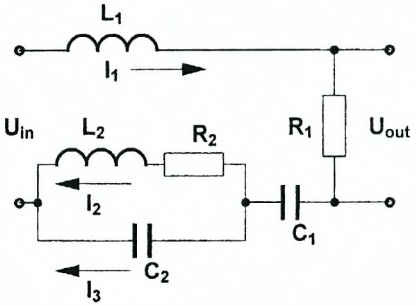
15)

Выберите одну из схем и выполните для нее следующие задания:
1) Запишите дифференциальное уравнение или систему дифференциальных уравнений для определения напряжения на конденсаторах и силы тока в катушках.
2. Задайте нулевые начальные условия и найдите решение полученного уравнения или системы в символьном виде, если на вход цепи подается

постоянное напряжение $U_{in} = 1$.
3. Задайте численные значения параметров цепи и постройте графики зависимостей силы тока в цепи и выходного напряжения от времени. Убедитесь в том, что с течением времени эти величины стремятся к некоторым постоянным значениям и оцените по графику время завершения переходных процессов.
4. Найдите численное решение уравнения или системы, если на вход цепи подается синусоидальное напряжение $U_{in} = A_0 \sin(\omega t)$. Предварительно задайте значения A_0 и ω. Временной интервал выберите таким образом, чтобы он превышал время завершения переходных процессов в цепи. Постройте графики найденных зависимостей силы тока в цепи и выходного напряжения от времени.
5. Определите полное комплексное сопротивление цепи (импеданс) и найдите амплитуды силы тока и выходного напряжения. Затем определите выходное напряжение *Uout* как функцию времени. На одной координатной плоскости постройте график полученной функции *Uout(t)* вместе с графиком зависимости выходного напряжения от времени, найденной в пункте 4.
6. Постройте амплитудно-частотную и фазово-частотную характеристики цепи.

Пример 3

Рассмотрим разветвленную электрическую цепь, изображенную на рисунке. Легко убедиться в том, что в этой цепи имеются три независимых тока, которые обозначены как I_1, I_2, I_3.



Чтобы исследовать зависимость токов и напряжений в цепи от времени, запишем полную систему уравнений, описывающую процессы в цепи, используя закон Ома для участка цепи и правила Кирхгофа.

```
In[35]:= Clear["Global`*"];
         eq1 = I1[t] == I2[t] + I3[t];
         eq2 = L1 I1'[t] + R1 I1[t] + Uc1[t] + Uc2[t] == Uin;
         eq3 = Uc2[t] == I2[t] R2 + L2 I2'[t];
```

Кроме того, запишем два уравнения, выражающие связь между токами *I1(t)* и *I3(t)* и напряжениями на конденсаторах C_1 и C_2.

```
In[39]:= eq4 = I1[t] == C1 Uc1'[t];
         eq5 = I3[t] == C2 Uc2'[t];
```

Получили пять уравнений относительно пяти неизвестных функций *I1(t)*, *I2(t)*, *I3(t)*, *Uc1(t)*, *Uc2(t)*. Поскольку в этих уравнениях отсутствует производная силы тока *I3(t)*, мы выражаем ее из уравнения *eq5* и подставляем в остальные уравнения. В результате получаем следующую систему из четырех дифференциальных уравнений:

```
In[42]:= eq = {eq1, eq2, eq3, eq4} /. Solve[eq5, I3[t]][[1]];
```

Выбираем нулевые начальные условия.

```
In[43]:= initial = {I1[0] == 0, I2[0] == 0, Uc1[0] == 0, Uc2[0] == 0};
```

Для исследования переходных процессов в цепи найдем решение системы *eq*, если на вход цепи подается постоянное напряжение $U_{in} = 1$. Попытка найти решение в символьной форме показывает, что оно чрезвычайно громоздкое.

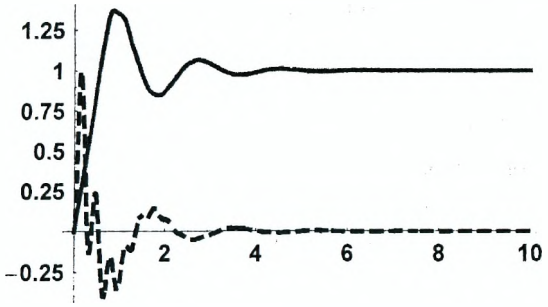
```
In[44]:= DSolve[Join[eq /. Uin → 1, initial],
           {I1[t], I2[t], Uc1[t], Uc2[t]}, t]
```

Поэтому мы его не приводим, а задаем численные данные и находим численное решение системы *eq*.

```
In[45]:= values1 = {R1 → 1, R2 → 0.5,
             C1 → 0.1, C2 → 0.02, L1 → 0.2, L2 → 0.5};
         sol1 = NDSolve[Join[eq /. Uin → 1 /. values1, initial],
           {I1[t], I2[t], Uc1[t], Uc2[t]}, {t, 0, 10}]
```

Построим графики зависимости напряжений на конденсаторах от времени.

```
In[49]:= Plot[{Uc1[t] /. sol1[[1]], Uc2[t] /. sol1[[1]]},
           {t, 0, 10}, PlotStyle → {{Thickness[0.01], Hue[0]},
             {Thickness[0.01], Dashing[{0.02}]}}];
```



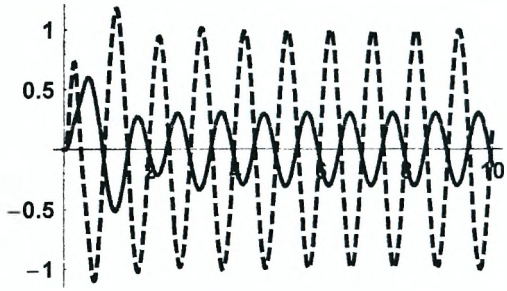
Как и должно быть, колебания в системе быстро затухают и напряжения *Uc1(t)* и *Uc2(t)* стремятся к значениям 1 и 0 соответственно. Этот результат, очевидно, соответствует нашим ожиданиям.

Теперь найдем решение системы *eq*, если на вход цепи подается синусоидальное напряжение $U_{in} = A_0 \sin(\omega t)$. Частоту ω задаем так, чтобы за время затухания переходных процессов происходило около пяти колебаний входного напряжения.

```
In[50]:= values2 = {A0 → 1, ω → 2 π};
         sol2 = NDSolve[
           Join[eq /. Uin → A0 Sin[ω t] /. values1 /. values2,
            initial], {I1[t], I2[t], Uc1[t], Uc2[t]}, {t, 0, 10}]
```

Графики зависимости напряжений на конденсаторах от времени в этом случае имеют вид:

```
In[52]:= Plot[{Uc1[t] /. sol2[[1]], Uc2[t] /. sol2[[1]]},
           {t, 0, 10}, PlotStyle → {{Thickness[0.01], Hue[0]},
             {Thickness[0.01], Dashing[{0.02}]}}];
```



Как видим, после затухания переходных процессов оба напряжения изменяются по синусоидальному закону с одинаковой частотой, но различными амплитудами.

Теперь найдем выходное напряжение, используя метод комплексных амплитуд. Сначала запишем уравнения для определения комплексных амплитуд токов, используя правила Кирхгофа.

$$In[55]:= eq6 = \left\{I1 == I2 + I3,\ i\,\omega\,L1\,I1 + R1\,I1 + \frac{I1}{i\,\omega\,C1} + \frac{I3}{i\,\omega\,C2} == 1,\right.$$

$$\left.\frac{1}{i\,\omega\,C2}\,I3 == I2\,R2 + i\,\omega\,L2\,I2\right\};$$

Решение этой системы легко находится.

```
In[56]:= sol3 = Solve[eq6, {I1, I2, I3}] // Simplify;
```

Используя полученное решение, можно определить выходное напряжение как функцию времени.

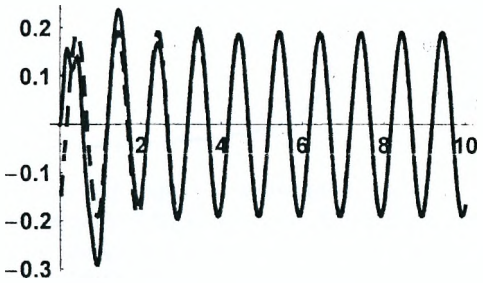
```
In[57]:= Uout1[t_] =
           Im[I1 R1 Exp[i ω t] /. sol3[[1]] // ExpToTrig ] //
             ComplexExpand //
           FullSimplify[#, {{C1, C2, R1, R2, L1, L2, ω} ∈ Reals}] &
```

Получаемые выражения довольно громоздки и мы их здесь не приводим. Вторую зависимость выходного напряжения от времени получаем, используя численное решение, найденное в *sol2*.

```
In[58]:= Uout2[t_] = R1 C1 D[Uc1[t] /. sol2[[1]], t] /. values1
```

Теперь построим на одной координатной плоскости два графика зависимости *Uout(t)*. Подчеркнем, что одна из них найдена как численное решение системы дифференциальных уравнений *eq*, а вторая – с помощью метода комплексных амплитуд.

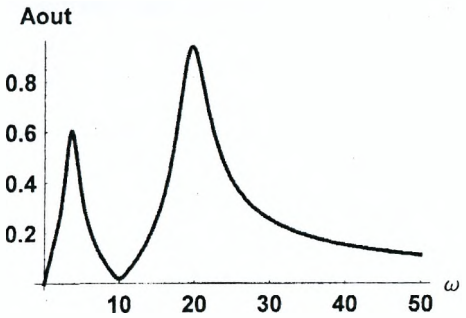
```
In[59]:= Plot[{Uout2[t] , Uout1[t] /. values1 /. values2},
          {t, 0, 10},
          PlotStyle → {{Thickness[0.01], RGBColor[1, 0, 0]},
                       {Thickness[0.01],
             Dashing[{0.04}], RGBColor[0, 0, 1]}}];
```



Как и ожидалось, после завершения переходных процессов в цепи обе функции совпадают.

Теперь можно построить амплитудно-частотную характеристику цепи.

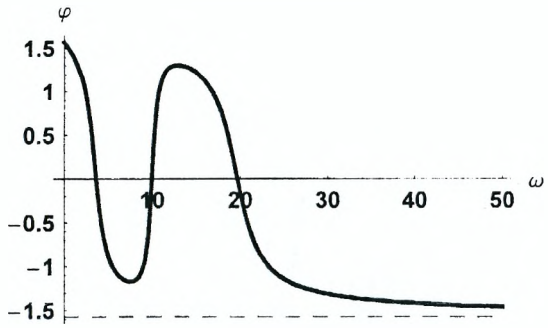
```
In[60]:= Plot[Abs[I1 R1 /. sol3[[1]] /. values1], {ω, 0, 50},
          PlotStyle → {{Thickness[0.01], Hue[0]}},
          AxesLabel → {"ω", "Aout"}];
```



Как видим, имеется две резонансные частоты, при которых амплитуда выходного напряжения имеет максимум. Так и должно быть, поскольку рассматриваемая цепь содержит два связанных колебательных контура.

Построим теперь фазово-частотную характеристику цепи.

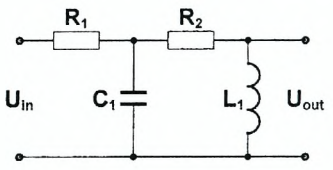
```
In[61]:= Plot[{Arg[I1 R1 /. sol3[[1]] /. values1], -π/2},
  {ω, 0, 50}, PlotStyle → {{Thickness[0.01], Hue[0.7]},
    {Dashing[{0.03}]}}, AxesLabel → {"ω", "φ"}];
```



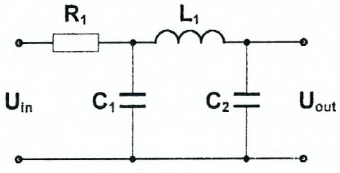
Из графика следует, что с ростом частоты ω сдвиг фаз между входным и выходным напряжениями стремится к $(-\frac{\pi}{2})$. Если учесть, что с ростом частоты сопротивление конденсаторов стремится к нулю, то при больших частотах цепь превращается в последовательно соединенные катушку индуктивности L_1 и резистор R_1, причем сопротивление катушки значительно превышает сопротивление резистора. Вспоминая, что сила тока в катушке отстает по фазе от напряжения на $\frac{\pi}{2}$, а в резисторе фазы силы тока и напряжения совпадают, приходим к выводу, что полученный результат соответствует действительности.

Задание 3

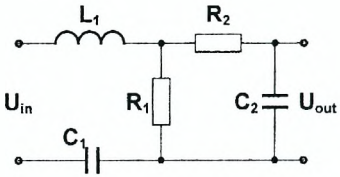
На рисунках изображены схемы разветвленных электрических цепей.



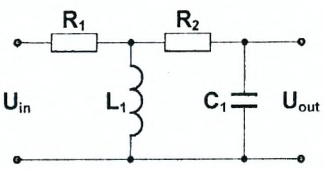
1)



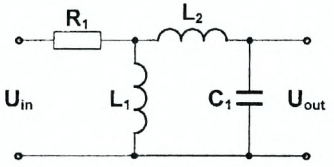
2)



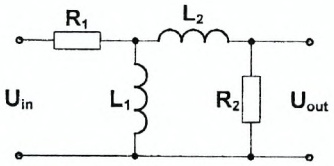
3)



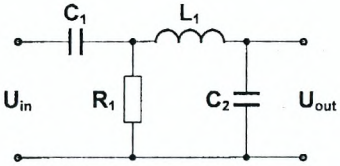
4)



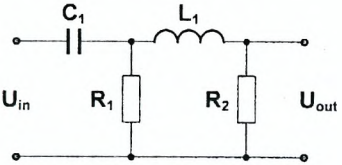
5)



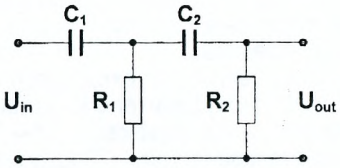
6)



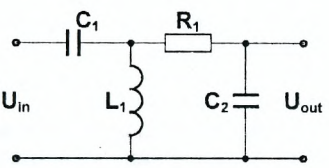
7)



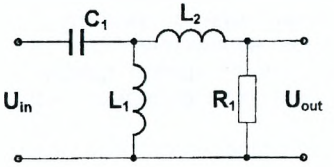
8)



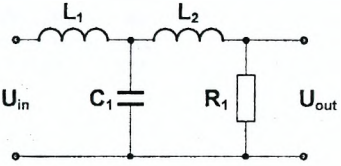
9)



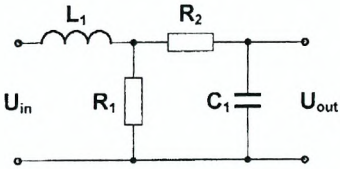
10)



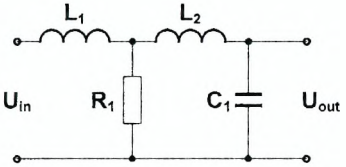
11)



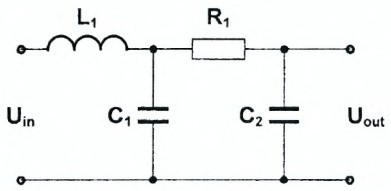
12)



13)



14)



15)

Выберите одну из схем и выполните для нее следующие задания:
1. Запишите полную систему дифференциальных уравнений для определения напряжений на конденсаторах и токов в катушках.
2. Задайте нулевые начальные условия, а также численные значения параметров цепи, и найдите решение полученной системы, если на вход цепи подается постоянное напряжение $U_{in} = 1$.
3. Постройте графики найденных зависимостей от времени. Убедитесь в том, что с течением времени токи в катушках прекращаются, а напряжения на конденсаторах стремятся к некоторым постоянным значениям. Оцените по графику время завершения переходных процессов.
4. Найдите численное решение системы, если на вход цепи подается синусоидальное напряжение $U_{in} = A_0 \sin(\omega t)$. Предварительно задайте значения A_0 и ω. Временной интервал выберите таким образом, чтобы он превышал время завершения переходных процессов в цепи. Постройте график зависимости выходного напряжения от времени.
5. Используя правила Кирхгофа, запишите полную систему уравнений для определения комплексных амплитут токов во все участках цепи. Решите полученную систему и определите комплексную амплитуду выходного напряжения. Затем определите выходное напряжение *Uout* как функцию времени. На одной координатной плоскости постройте график полученной функции *Uout(t)* вместе с графиком зависимости выходного напряжения от времени, найденной в пункте 4.
6. Постройте амплитудно-частотную и фазово-частотную характеристики цепи.

Лабораторная работа № 4
Изучение электрических цепей, содержащих нелинейные элементы

Цель работы:

1) изучить методы расчета простейших электрических цепей, содержащих полупроводниковые диоды;

2) исследовать разветвленные электрические цепи, содержащие полупроводниковые диоды, резисторы, конденсаторы и катушки индуктивности.

Одним из наиболее распространенных нелинейных элементов является полупроводниковый диод. Вольт-амперная характеристика идеального диода (т.е. зависимость силы тока I от приложенного к диоду напряжения U) имеет вид:

$$I = I_0\left(\mathrm{Exp}\left(\frac{U}{\varphi_T}\right) - 1\right),$$

где I_0 – величина обратного тока насыщения, φ_T – тепловой потенциал, который связан с зарядом электрона $q = 1.6\times10^{-19}$ Кл, постоянной Больцмана $k = 1.38\times10^{-23}$ Дж/К и температурой T соотношением:

$$\varphi_T = \frac{kT}{q}.$$

При комнатной температуре (T = 298 К) получаем:

$$\varphi_T = \frac{1.38\times10^{-23}\times298}{1.6\times10^{-19}} = 25.7\,mB.$$

Типичное значение обратного тока насыщения $I_0 = 10^{-9}\,A$. Поэтому вольт-амперную характеристику идеального диода можно представить в виде:

$$I(U) = 10^{-9}\left(\mathrm{Exp}\left(\frac{U}{0.0257}\right) - 1\right)(A).$$

Задание 1

Построить вольт-амперную характеристику идеального диода в интервале напряжений $U \in [-0.2, 0.15]$, используя функцию **Plot**. Убедиться в том, что обратный ток через диод является достаточно малым. Это означает, что при небольших обратных напряжениях током через диод можно пренебречь.

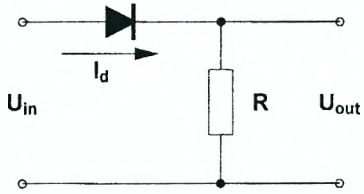
Замечание. Учитывая свойства диода, далее для уменьшения времени вычислений будем использовать для него следующую модельную вольт-амперную характеристику:

$$I(U) = \begin{cases} U/R_0, & U > 0 \\ 0, & U \le 0 \end{cases}$$

где R_0 – сопротивление диода при прямом токе. Таким образом, будем считать, что при положительном приложенном напряжении диод ведет себя как проводник сопротивлением R_0, а при отрицательном – диод закрыт и ток в нем отсутствует. При вычислениях сопротивление диода при прямом токе R_0 будем выбирать из интервала $R_0 \in [1, 5]$ Ом.

Пример 1

Рассмотрим простейшую электрическую цепь, в которой диод и резистор включены последовательно.



Обозначим направление тока I_d, соответствующее прямому току через диод. Соответствующее положительное напряжение на диоде обозначим U_d. Используя закон Ома для участка цепи, запишем уравнение для определения напряжения на диоде.

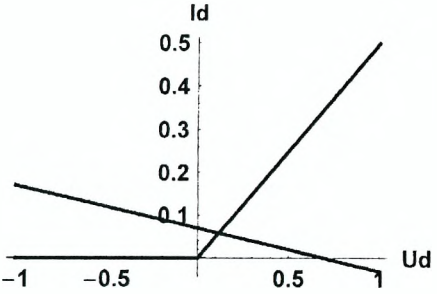
In[62]:=
```
Clear["Global`*"];
eq = Uin == Ud + R Id[Ud];
```

Определим вольт-амперную характеристику диода и параметры цепи.

In[64]:=
```
R0 = 2; R = 10;
Id[Ud_] := If[ Ud > 0, Ud / R0, 0];
```

Очевидно, уравнение *eq* является нелинейным. Найти его приближенное решение при заданном значении входного напряжения можно графически. Выберем для U_{in} значение $U_{in} = 0.7$, например, и построим на одной координатной плоскости графики функций *Id(Ud)* и *(Uin-Ud)/R*.

In[66]:=
```
Plot[{(Uin - Ud) / R /. Uin → 0.7, Id[Ud]}, {Ud, -1, 1},
  PlotStyle → {{Thickness[0.01], Hue[0]},
    {Thickness[0.01], Hue[0.65]}},
  AxesLabel → {"Ud", "Id"}];
```



Как видим, имеется одна точка пересечения линий, которая и соответствует корню уравнения *eq*. Точное значение корня можно найти с помощью функции **FindRoot**. Так как производная функции *Id(Ud)* не может быть найдена в символьном виде, для решения уравнения используем метод хорд, указав в фигурных скобках два стартовых значения для *Ud*. Поскольку уравнение *eq* имеет только один корень при любом значении *Ud*, в качестве стартовых значений можно выбрать любые два числа, например, {0.1, 0.3} (проверьте, так ли это).

```
In[67]:= FindRoot[eq /. Uin → 0.7 // Evaluate, {Ud, 0.1, 0.3}]

Out[67]= {Ud → 0.116667}
```

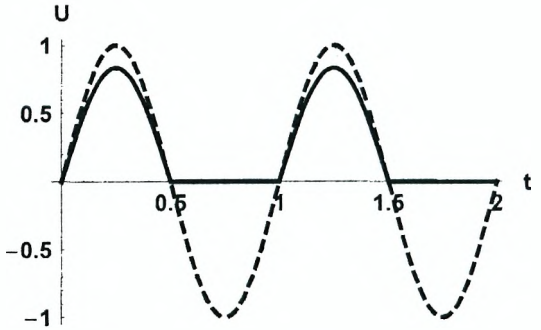
Ясно, что изменение значения *Uin* приведет к параллельному переносу прямой на графике и изменению положения точки пересечения линий. Поскольку рассматриваемая цепь не содержит инерционных элементов, таких как конденсаторы и катушки индуктивности, напряжение на диоде и ток в цепи в любой момент времени определяются только соответствующим значением входного напряжения. Для вычисления выходного напряжения при любом входном напряжении *U* введем следующую функцию:

```
In[68]:= Uout[U_?NumberQ] := R Id[Ud] /.
           FindRoot[eq /. Uin → U // Evaluate , {Ud, 0.1, 0.3}]
```

Следует отметить, что в определении функции *Uout* обязательно должен стоять знак отложенного присваивания ":=", так как в противном случае при обработке этой секции функция **FindRoot** сразу же будет пытаться найти корень уравнения *eq*, что невозможно сделать, пока не задано численное значение входного напряжения. По этой же причине на аргумент функции *Uout* наложено условие, что это должно быть число (команда **?NumberQ**, набираемая сразу после символа подчеркивания "_").

Теперь можно определить входное синусоидальное напряжение *U1(t)* и построить график зависимости выходного напряжения от времени вместе с графиком входного напряжения.

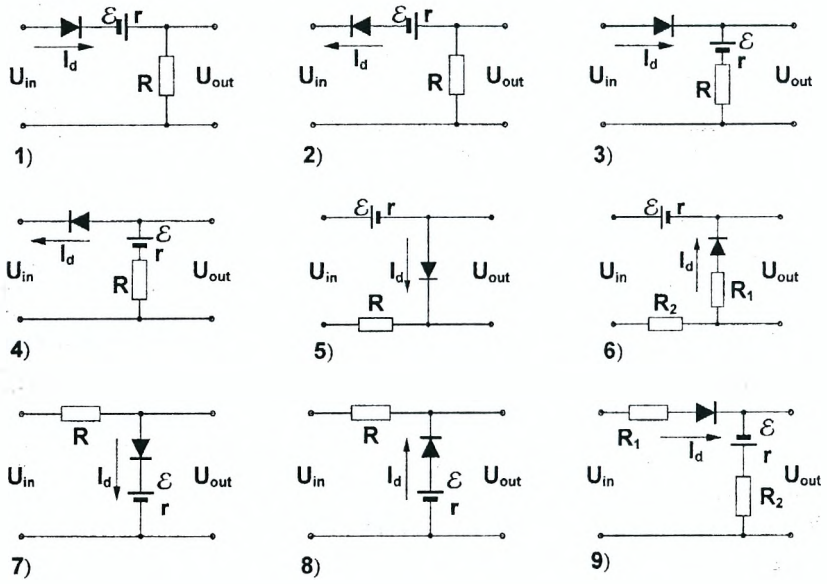
```
In[69]:= U1[t_] = Sin[2 π t];

In[70]:= Plot[{U1[t], Uout[U1[t]]}, {t, 0, 2},
           PlotStyle → {{Thickness[0.01], Dashing[{0.02}]},
                        {Thickness[0.01]}}, AxesLabel → {"t", "U"}];
```

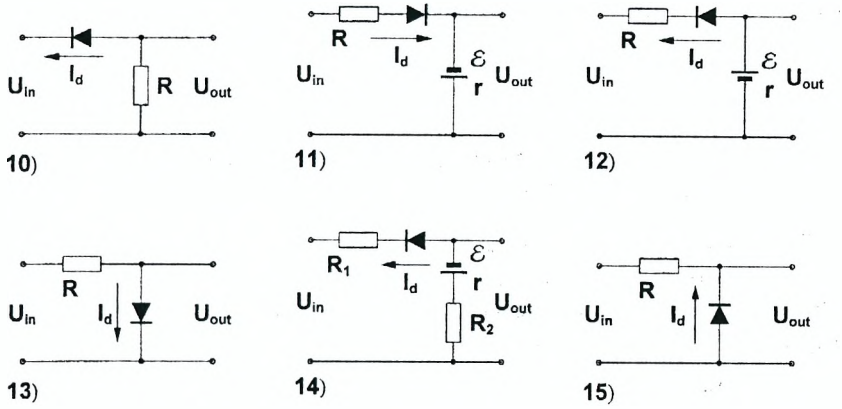


Как и следовало ожидать, во время положительного полупериода входного напряжения диод открыт и вместе с резистором составляет делитель напряжения. Поэтому выходное напряжение повторяет зависимость от времени входного напряжения, но имеет меньшую амплитуду. Во время отрицательного полупериода диод закрыт и напряжение на выходе равно нулю.

Задание 2

1. Выберите одну из схем, изображенных на рисунках. Определите вольт-амперную характеристику диода *Id*(*Ud*) и задайте численные значения параметров системы.

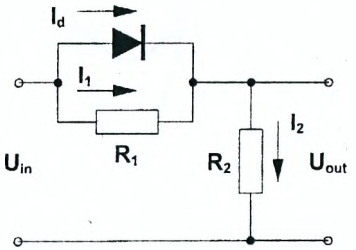




2. Используя закон Ома, запишите уравнение для определения напряжения на диоде *Ud* (см. Пример 1). Найдите графически приближенное решение этого уравнения при некотором значении входного напряжения U_{in} из интервала $U_{in} \in [0, 1]$. Затем найдите решение уравнения с помощью функции **FindRoot**.
3. Определите функцию для вычисления выходного напряжения U_{out} при произвольном входном. На одной координатной плоскости постройте графики зависимостей от времени входного синусоидального напряжения и выходного напряжения. Объясните работу схемы и сделайте вывод.

Пример 2

Рассмотрим простейшую разветвленную электрическую цепь.



Обозначим токи в различных звеньях цепи через I_d, I_1, I_2. Используя правила Кирхгофа, запишем систему уравнений для определения токов в виде:

```
In[71]:= Clear["Global`*"];
         eq1 = Id + I1 == I2;
         eq2 = Uin == I1 R1 + I2 R2;
         eq3 = Ud == I1 R1;
```

С помощью функции **Eliminate** исключим из системы уравнений *eq1*, *eq2*, *eq3* токи *I1* и *I2*. Полученное уравнение обозначим *eq4* и добавим у тока *Id* аргумент *Ud*.

```
In[75]:= eq4 = Eliminate[{eq1, eq2, eq3}, {I1, I2}] /. Id → Id[Ud]

Out[75]= -R1 Ud - R2 Ud + R1 Uin == R1 R2 Id[Ud]
```

Получили нелинейное уравнение, которое имеет один корень *Ud* при любом значении входного напряжения *Uin*. Определим теперь параметры цепи и вольт-амперную характеристику диода.

```
In[76]:= R0 = 1; R1 = 20; R2 = 10;
         Id[Ud_] := If[Ud > 0, Ud / R0, 0];
```

Тогда функцию, определяющую выходное напряжение, можно задать в виде:

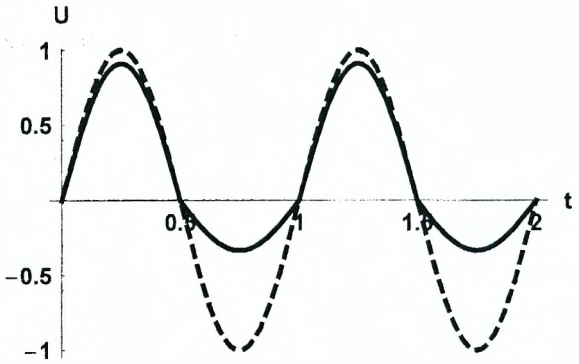
```
In[78]:= Uout[U_?NumberQ] := U - Ud /.
           FindRoot[eq4 /. Uin → U // Evaluate, {Ud, 0, 0.4}]
```

Определяем входное синусоидальное напряжение.

```
In[79]:= U1[t_] = Sin[2 π t];
```

Теперь можно построить графики входного и выходного напряжений как функций времени.

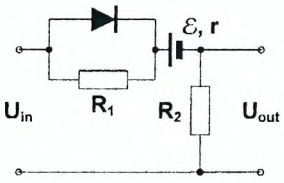
```
In[80]:= Plot[{U1[t], Uout[U1[t]]}, {t, 0, 2},
           PlotStyle → {{Thickness[0.01], Dashing[{0.02}]},
                        {Thickness[0.01]}}, AxesLabel → {"t", "U"}];
```



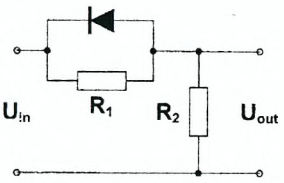
При положительном входном напряжении диод открыт. В этом случае сопротивление первого участка цепи мало, так как сопротивление диода мало, и практически все падение напряжения приходится на сопротивление *R2*. Поэтому амплитуда выходного напряжения мало отличается от амплитуды входного. При отрицательном входном напряжении диод закрыт и работает делитель напряжения *R1-R2*. Таким образом, полученный график правильно отражает работу цепи.

Задание 3

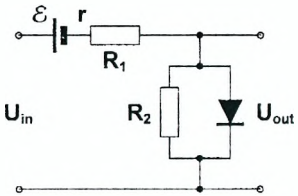
1. Выберите одну из схем, изображенных на рисунках.



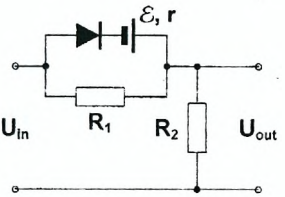
1)



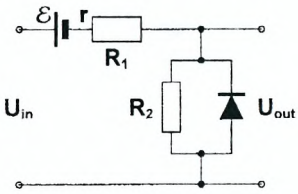
2)



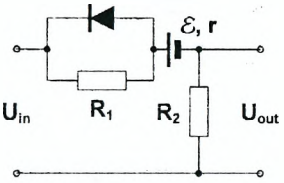
3)



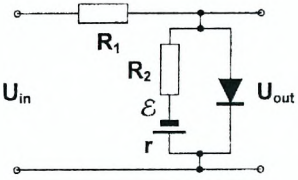
4)



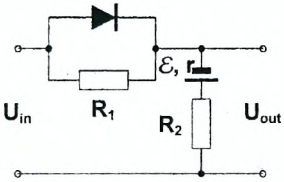
5)



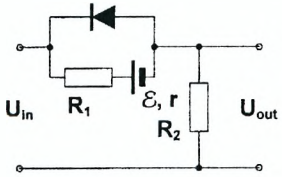
6)



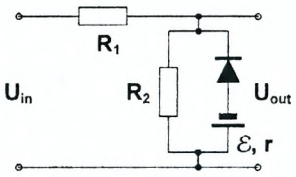
7)



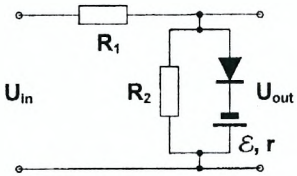
8)



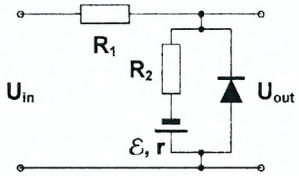
9)



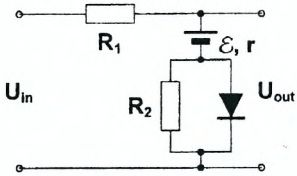
10)



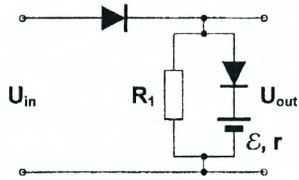
11)



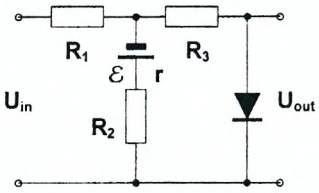
12)



13)



14)

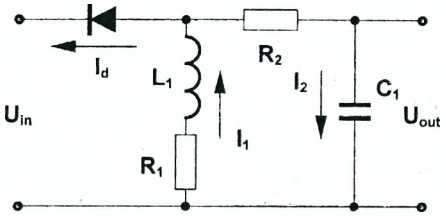


15)

2. Обозначьте токи в отдельных участках цепи и, используя правила Кирхгофа, запишите полную систему уравнений для их определения (см. Пример 2). Из этой системы получите одно уравнение, определяющее напряжение на диоде *Ud* при заданном входном напряжении *Uin*.
3. Определите вольт-амперную характеристику диода *Id*(*Ud*) и задайте численные значения параметров системы.
4. Определите функцию для вычисления выходного напряжения при произвольном входном. На одной координатной плоскости постройте графики зависимостей от времени входного синусоидального напряжения и выходного напряжения. Объясните работу схемы и сделайте вывод.

Пример 3

Рассмотрим разветвленную электрическую цепь, содержащую диод, резисторы, конденсатор и катушку индуктивности.



Обозначим токи в различных звеньях цепи через I_d, I_1, I_2. Напряжение на конденсаторе U_c считаем положительным, когда его верхняя обкладка имеет положительный заряд. Используя правила Кирхгофа, запишем систему уравнений для определения токов в виде:

$$I_1 = I_d + I_2,$$

$$U_{in} = -U_d - R_1 I_1 - L_1 \frac{d I_1}{d t},$$

$$U_{in} = -U_d + I_2 R_1 + U_c,$$

$$I_2 = C_1 \frac{d U_c}{d t}.$$

Так как в уравнениях содержатся производные функций I_1 и U_c, то для определения этих функций мы должны записать два дифференциальных уравнения. Силу тока I_2 выразим из третьего уравнения и подставим в первое и четвертое. В результате получаем два дифференциальных уравнения для функций I_1 и U_c

$$U_{in} = -U_d - R_1 I_1 - L_1 \frac{d I_1}{d t},$$

$$R_1 C_1 \frac{d U_c}{d t} = U_{in} - U_c + U_d,$$

и одно нелинейное уравнение для определения напряжения на диоде при заданных значениях входного напряжения U_{in}, напряжения на конденсаторе U_c и силы тока в катушке I_1:

$$R_1 (I_1 - I_d (U_d)) = U_{in} - U_c + U_d.$$

Для решения полученной системы уравнений определяем вольт-амперную характеристику диода, параметры цепи и входное синусоидальное напряжение.

```
In[81]:= Clear["Global`*"];
         R0 = 1;
         Id[Ud_] := If[Ud > 0, Ud / R0, 0];
         R1 = 20; R2 = 30; C1 = 0.2; L1 = 0.1;
         U1[t_] := Sin[2 π t];
```

Для решения нелинейного уравнения определяем следующую функцию *Ud*, зависящую от трех переменных.

```
In[86]:= Ud[Uin_?NumberQ, Uc_?NumberQ, I1_?NumberQ] :=
           Ud1 /. FindRoot[
             R1 (I1 - Id[Ud1]) == Uin - Uc + Ud1, {Ud1, 0, 0.3}];
```

Так как функция *Ud* решает нелинейное уравнение с помощью встроенной функции **FindRoot**, в ее определении используется знак отложенного присваивания ":=", а ее аргументы должны быть числами. Соответствующую систему дифференциальных уравнений определим как *eq*.

```
In[87]:= eq =
           {R1 C1 Uc'[t] == U1[t] - Uc[t] + Ud[U1[t], Uc[t], I1[t]],
                L1 I1'[t] + R1 I1[t] +
              U1[t] + Ud[U1[t], Uc[t], I1[t]] == 0}

Out[87]= {4. Uc'[t] ==
            Sin[2 π t] - Uc[t] + Ud[Sin[2 π t], Uc[t], I1[t]],
          20 I1[t] + Sin[2 π t] +
            Ud[Sin[2 π t], Uc[t], I1[t]] + 0.1 I1'[t] == 0}
```

Выберем нулевые начальные условия для функций I_1 и U_C.

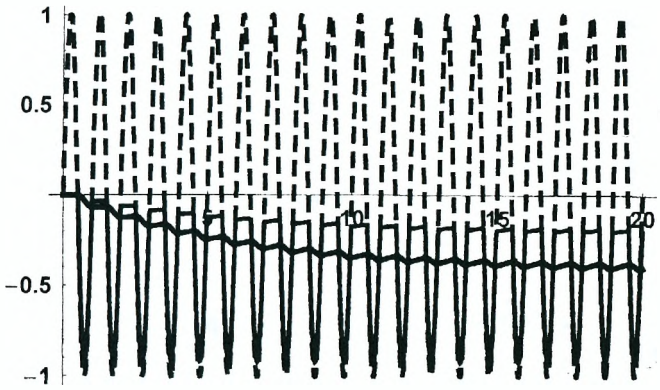
```
In[88]:= initial = {I1[0] == 0, Uc[0] == 0};
```

Находим численное решение системы дифференциальных уравнений.

```
In[89]:= sol = NDSolve[Join[eq, initial], {I1[t], Uc[t]},
             {t, 0, 20}, MaxSteps → 10000, StartingStepSize → 0.01]

Out[89]= {{I1[t] → InterpolatingFunction[{{0., 20.}}, <>][t],
           Uc[t] → InterpolatingFunction[{{0., 20.}}, <>][t]}}
```

Строим графики зависимости от времени входного и выходного напряжений, а также напряжения на резисторе R_1.

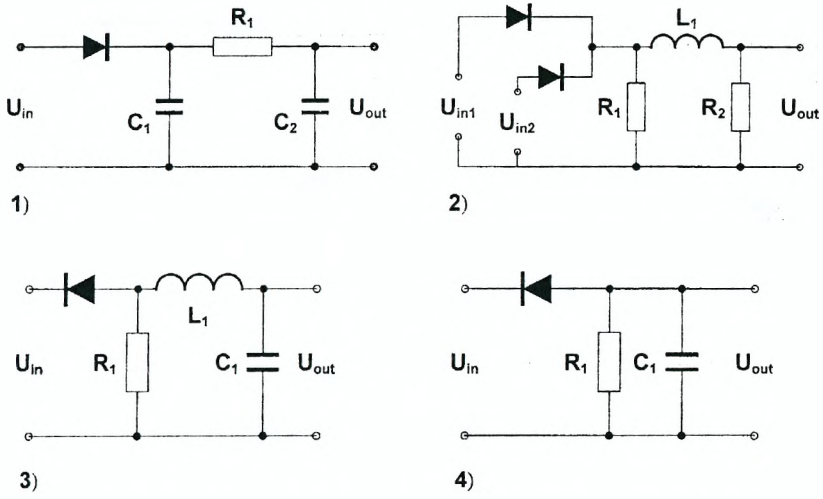
```
In[91]:= Plot[{U1[t] , -R1 I1[t] /. sol[[1]], Uc[t] /. sol[[1]] },
             {t, 0, 20}, PlotPoints → 50,
             PlotStyle → {{Thickness[0.008], Dashing[{0.02}]},
                          {Thickness[0.008]}, {Thickness[0.01]}}];
```

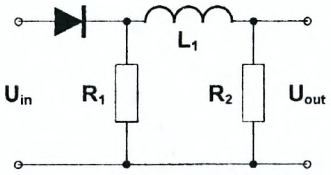


Так как диод пропускает только отрицательное напряжение, выходное напряжение является также отрицательным. Ток через резистор R_1 течет в одну сторону, но сильно пульсирует. За счет катушки индуктивности и конденсатора выходное напряжение существенно сглаживается. Таким образом, полученное решение правильно отражает работу цепи.

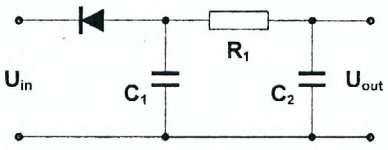
Задание 4

1. Выберите одну из схем, изображенных на рисунке. Затем обозначьте токи и запишите полную систему уравнений для их определения.

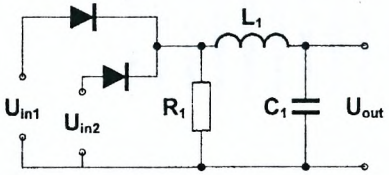




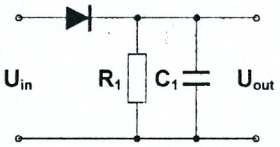
5)



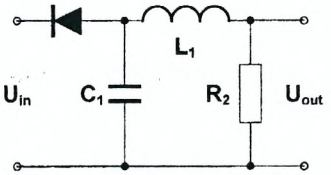
6)



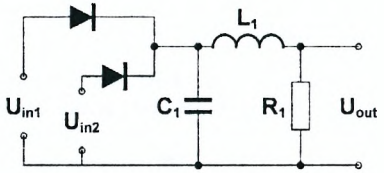
7)



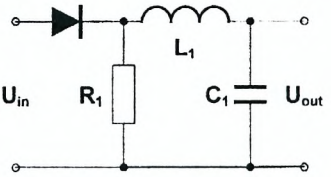
8)



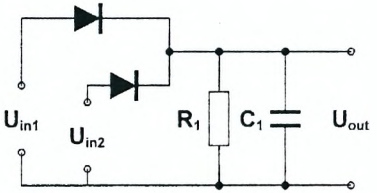
9)



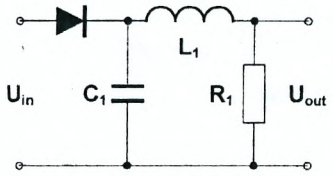
10)



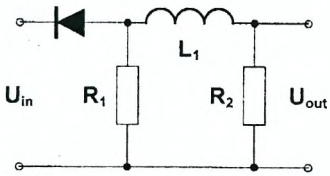
11)



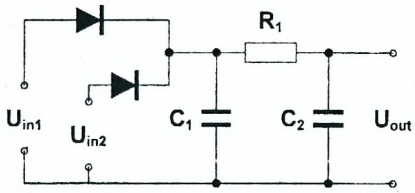
12)



13)



14)



15)

2. Определите вольт-амперную характеристику диода *Id(Ud)* и задайте численные значения параметров системы.
3. Найдите решение записанной системы уравнений при синусоидальном входном напряжении и определите выходное напряжение как функцию времени (см. Пример 3). На одной координатной плоскости постройте графики зависимостей от времени входного и выходного напряжений. Объясните работу схемы и сделайте вывод.

Лабораторная работа № 5

Интегрирование уравнений движения материальной точки

Цель работы:

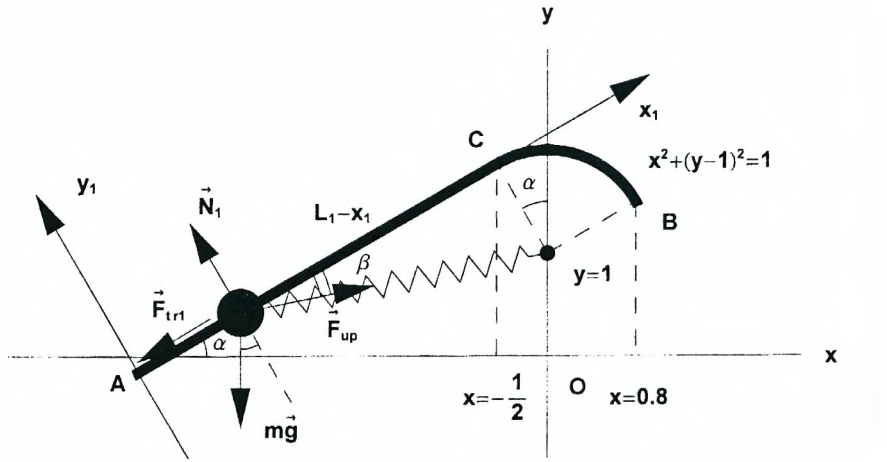
1) записать уравнение движения материальной точки вдоль заданной кривой, найти его решение, вычислить время движения и конечную скорость точки;

2) записать уравнение движения заряженной частицы в однородных электрическом и магнитном полях, найти его решение в символьном виде, построить траекторию частицы и проанализировать характер ее движения. Выяснить, что представляет собой электрический дрейф.

I. Движение материальной точки вдоль заданной кривой

Пример 1

Шарик массой m начинает движение из точки A с нулевой начальной скоростью вдоль стержня, который расположен в вертикальной плоскости xOy и состоит из двух частей: первая часть стержня представляет собой прямолинейный отрезок AC длиной L_1, а вторая – дугу окружности единичного радиуса, причем центр окружности находится на оси Oy в точке с координатой $y = 1$.



Прямолинейный отрезок является частью касательной, проведенной к окружности в точке C, в которой координата $x = -\frac{1}{2}$. При движении вдоль прямо-

линейного участка стержня на шарик действует сила упругости со стороны пружины, второй конец которой закреплен в центре окружности. В момент прохождения шариком точки *C* пружина, находящаяся в недеформированном состоянии, отсоединяется от него. Коэффициент жесткости пружины равен *k*. Коэффициент трения шарика о стержень равен μ. Требуется задать численные значения параметров системы и определить время движения шарика вдоль стержня и его скорость в точке *B*.

Сначала рассмотрим прямолинейный отрезок стержня. Угол α между стержнем и горизонтальной осью *Ox* найдем из условия, что стержень совпадает с касательной, проведенной к окружности в точке $x = -\frac{1}{2}$. Для этого определяем уравнение окружности и разрешаем его относительно *y*.

In[6]:=
```
Clear["Global`*"];
eq1 = x^2 + (y - 1)^2 == 1;
sol1 = Solve[eq1, y]
```

Out[8]= $\left\{\left\{y \to 1 - \sqrt{1 - x^2}\right\}, \left\{y \to 1 + \sqrt{1 - x^2}\right\}\right\}$

Очевидно, верхней половине окружности соответствует второй корень уравнения *eq1*. Вычисляя производную соответствующей функции *y* в точке $x = -\frac{1}{2}$, находим тангенс угла наклона касательной к оси *Ox*.

In[9]:=
```
tgα = D[y /. sol1[[2]], x] /. x → -1/2
```

Out[9]= $\frac{1}{\sqrt{3}}$

Поскольку в дальнейшем нам понадобятся значения $\sin\alpha$ и $\cos\alpha$, сразу вычислим их, используя известные соотношения между тригонометрическими функциями.

In[10]:=
```
sinα = tgα / Sqrt[1 + tgα^2]
```

Out[10]= $\frac{1}{2}$

In[11]:=
```
cosα = 1 / Sqrt[1 + tgα^2]
```

Out[11]= $\frac{\sqrt{3}}{2}$

Заметим, что мы не обращаемся к встроенным функциям Sin[α] и Cos[α], а используем символы $\sin\alpha$ и $\cos\alpha$ в качестве имен переменных.

Для описания движения шарика проведем вдоль стержня дополнительную ось O_1x_1, начало которой O_1 совпадает с точкой *A*. Отметим силы, действующие на шарик, находящийся в произвольной точке с координатой x_1. Поскольку касательная к окружности перпендикулярна радиусу, длина пружины *L* равна длине гипотенузы прямоугольного треугольника.

In[12]:=
```
L = Sqrt[1 + (L1 - x1)^2];
```

Так как по условию длина недеформированной пружины равняется радиусу окружности, сила упругости, действующая на шарик со стороны пружины, равна:

```
In[13]:= Fup = k (L - 1)
```

$$Out[13]= k\left(-1+\sqrt{1+(L1-x1)^2}\right)$$

Чтобы вычислить ее проекции на оси O_1x_1 и O_1y_1, определим $\cos\beta$ и $\sin\beta$, которые, очевидно, зависят от положения шарика.

```
In[14]:= cosβ = (L1 - x1)/L
```

$$Out[14]= \frac{L1-x1}{\sqrt{1+(L1-x1)^2}}$$

```
In[15]:= sinβ = 1/L
```

$$Out[15]= \frac{1}{\sqrt{1+(L1-x1)^2}}$$

Поскольку координата y_1 шарика не изменяется, сумма проекций на ось O_1y_1 всех сил, действующих на него, должна равняться нулю. Это условие дает следующее уравнение:

```
In[16]:= eq2 = N1 - m g cosα - Fup sinβ == 0
```

$$Out[16]= -\frac{1}{2}\sqrt{3}\,g\,m + N1 - \frac{k\left(-1+\sqrt{1+(L1-x1)^2}\right)}{\sqrt{1+(L1-x1)^2}} == 0$$

Отсюда легко найти силу нормальной реакции стержня *N1* и определить проекцию силы трения на ось O_1x_1.

```
In[17]:= Ftr1 = -Sign[v] μ N1 /. Solve[eq2, N1][[1]]
```

$$Out[17]= -\left(\left(-2k+2k\sqrt{1+L1^2-2\,L1\,x1+x1^2}+\sqrt{3}\,g\,m\sqrt{1+L1^2-2\,L1\,x1+x1^2}\right)\mu\,Sign[v]\right)\Big/\left(2\sqrt{1+L1^2-2\,L1\,x1+x1^2}\right)$$

Поскольку при движении шарика сила трения должна быть направлена против скорости, в выражении для *Ftr1* использована функция **Sign[v]**, которая принимает значения 1, 0, –1 при $v > 0$, $v = 0$, $v < 0$ соответственно. Следует отметить, что в общем случае при определении силы трения в выражение для *Ftr1* следует подставлять **Abs[N1]**, так как направление силы нормальной реакции стержня *N1* при движении шарика может измениться на противоположное. Легко видеть, что в рассматриваемом случае сила *N1* в любой момент времени направлена так, как указано на рисунке, и использование функции **Abs** не является обязательным.

Теперь определим уравнение движения шарика вдоль оси O_1x_1. Так как

координата шарика *x1* является функцией времени, записывая второй закон Ньютона в проекции на ось O_1x_1, в правой части уравнения *eq3* выполняем подстановки x1 → x1[*t*], *v* → x1'[*t*]. В результате получаем уравнение движения в виде:

In[18]:=
```
eq3 = { m x1''[t] == (Fup cosβ - m g sinα + Ftr1 /.
        {x1 → x1[t], v → x1'[t]})}
```

Out[18]=

$$\left\{m\,x1''[t] == -\frac{g\,m}{2} + \frac{k\left(-1+\sqrt{1+(L1-x1[t])^2}\right)(L1-x1[t])}{\sqrt{1+(L1-x1[t])^2}} - \left(\mu\,\text{Sign}[x1'[t]]\left(-2k+2k\sqrt{1+L1^2-2\,L1\,x1[t]+x1[t]^2}+\sqrt{3}\,g\,m\sqrt{1+L1^2-2\,L1\,x1[t]+x1[t]^2}\right)\right)\Big/\left(2\sqrt{1+L1^2-2\,L1\,x1[t]+x1[t]^2}\right)\right\}$$

Поскольку направление силы *Ftr1* учтено при ее определении, в уравнении движения шарика соответствующее слагаемое взято со знаком "+".

Очевидно, начальные условия для функции *x1(t)* имеют вид:

In[19]:=
```
initial1 = {x1[0] == 0, x1'[0] == 0};
```

Уравнение *eq3* является нелинейным, и получить его решение в символьной форме не представляется возможным. Чтобы найти его численное решение, задаем некоторые реалистичные значения параметров системы в виде списка правил замены.

In[20]:=
```
dat1 = {m → 1, L1 → 5, k → 100, g → 9.8, μ → 0.2};
```

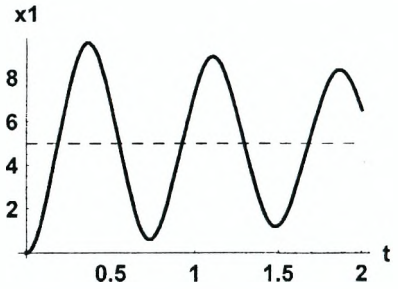
Теперь решение дифференциального уравнения *eq3* легко находится с помощью функции **NDSolve**.

In[21]:=
```
sol2 =
    NDSolve[Join[eq3 /. dat1, initial1], x1[t], {t, 0, 2}]
```

Out[21]=
```
{{x1[t] → InterpolatingFunction[{{0., 2.}}, <>][t]}}
```

График найденной зависимости *x1(t)* имеет вид:

In[22]:=
```
Plot[{x1[t] /. sol2[[1]], L1 /. dat1}, {t, 0, 2},
    PlotStyle →
  {{Thickness[0.011]}, {Dashing[{0.03}]}},
    AxesLabel → {"t", "x1"}];
```



Как видим, движение шарика вдоль прямолинейного отрезка представляет собой затухающие колебания. Этого и следовало ожидать, так как действие силы трения приводит к диссипации механической энергии системы. Очевидно, весь график соответствовал бы движению шарика в том случае, если бы в точке *C* прямолинейный участок не заканчивался. На самом деле в тот момент, когда шарик достигает точки *C*, т.е. его координата *x1* становится равной *L1* (на графике уровень $x1 = L1$ отмечен штриховой линией), начинается движение вдоль криволинейного участка. На графике видно, что это происходит в момент $t \approx 0.2$. Более точно этот момент времени может быть найден с помощью функции **FindRoot**. Выбирая $t = 0.2$ в качестве стартового значения, получаем:

In[23]:=
```
sol3 =
    FindRoot[(x1[t] /. sol2[[1]]) == (L1 /. dat1), {t, 0.2}]
```

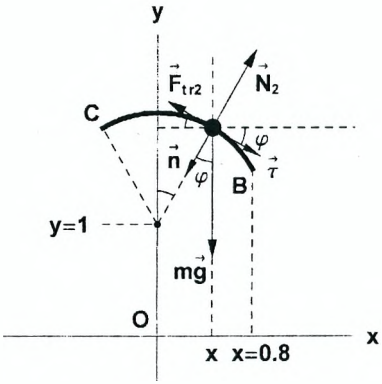
Out[23]= `{t → 0.187564}`

Скорость шарика в точке *C* равна:

In[24]:=
```
v20 = D[x1[t] /. sol2[[1]], t] /. sol3
```

Out[24]= `38.8165`

Очевидно, найденная скорость является начальной для криволинейного участка движения. Поскольку кривая, вдоль которой движется шарик, задана, для определения его положения достаточно одной координатой *x*. Чтобы получить уравнение для функции *x(t)*, изобразим шарик в произвольной точке на кривой и отметим все действующие на него силы.



На рисунке отмечены также два единичных вектора: $\vec{\tau}$ и $\vec{n}$. Вектор $\vec{\tau}$ направлен по касательной к кривой в сторону движения шарика, а вектор $\vec{n}$ – по нормали к центру ее кривизны (в данном случае, к центру окружности). Очевидно, тангенс угла φ между касательной и осью Ox равен производной функции $y = y(x)$, определяющей кривую. Используя второй корень уравнения *eq1*, находим:

In[25]:=
```
tgφ = D[y /. sol1[[2]], x]
```

Out[25]=
$$-\frac{x}{\sqrt{1-x^2}}$$

Зная $\mathrm{tg}\varphi$, можно легко найти $\sin\varphi$ и $\cos\varphi$. Чтобы записать результаты в более простом виде, добавим в качестве второго аргумента функции **Simplify** естественное в данном случае ограничение $-1 \le x \le 1$ на возможные значения координаты x.

In[26]:=
```
sinφ = tgφ 1/Sqrt[1 + tgφ^2] // Simplify[#, -1 ≤ x ≤ 1] &
```

Out[26]=
```
-x
```

In[27]:=
```
cosφ = 1/Sqrt[1 + tgφ^2] // Simplify[#, -1 ≤ x ≤ 1] &
```

Out[27]=
$$\sqrt{1-x^2}$$

Радиус кривизны R произвольной кривой $y = y(x)$ в каждой ее точке определяется соотношением:

$$\frac{1}{R} = \pm\frac{y''}{(1+y'^2)^{3/2}},$$

причем знак "+" или "–" выбирается из условия положительности R. Поскольку для рассматриваемой кривой вторая производная отрицательна, находим:

In[28]:=
```
R = -(1 + D[y /. sol1[[2]], x]^2)^(3/2)/D[y /. sol1[[2]], {x, 2}] //
    Simplify[#, -1 ≤ x ≤ 1] &
```

Out[28]=
```
1
```

Получили единицу, как и должно быть, так как шарик движется по дуге окружности единичного радиуса. Скорость шарика определяется соотношением:

In[29]:=
```
v2 = Sqrt[x'[t]^2 + D[y /. sol1[[2]] /. x → x[t], t]^2 //
     Simplify[#, x'[t] > 0] &
```

Out[29]=
$$\sqrt{\frac{1}{1-x[t]^2}}\,x'[t]$$

где учтено, что на рассматриваемом участке движения производная функции $x(t)$ положительна.

Используя выражение для нормального ускорения шарика $a_n = v^2/R$, запишем второй закон Ньютона в проекции на направление нормали $\vec{n}$.

```
In[30]:= eq5 = m v2^2/R == (m g Cos[φ] - N2 /. x → x[t])
```

$$Out[30]= \frac{m\,x'[t]^2}{1-x[t]^2} == -N2 + g\,m\sqrt{1-x[t]^2}$$

Решая уравнение *eq5*, находим силу нормальной реакции *N2* и определяем проекцию силы трения на тангенциальное направление, определяемое вектором $\vec{\tau}$.

```
In[31]:= Ftr2 = -μ Sign[x'[t]] Abs[N2] /. Solve[eq5, N2][[1]]
```

$$Out[31]= -\mu\,\text{Abs}\left[\frac{-g\,m\sqrt{1-x[t]^2} + g\,m\,x[t]^2\sqrt{1-x[t]^2} + m\,x'[t]^2}{-1+x[t]^2}\right]$$

$$\text{Sign}[x'[t]]$$

Так как при движении шарика по окружности сила *N2* может изменять свое направление на противоположное, в определении силы трения используется ее абсолютное значение.

Второй закон Ньютона в проекции на тангенциальное направление имеет вид:

```
In[32]:= eq6 = m D[v2, t] == (-m g Sin[φ] /. x → x[t]) + Ftr2
```

$$Out[32]= m\left(x[t]\left(\frac{1}{1-x[t]^2}\right)^{3/2}x'[t]^2 + \sqrt{\frac{1}{1-x[t]^2}}\,x''[t]\right) ==$$

$$-\mu\,\text{Abs}\left[\frac{-g\,m\sqrt{1-x[t]^2} + g\,m\,x[t]^2\sqrt{1-x[t]^2} + m\,x'[t]^2}{-1+x[t]^2}\right]$$

$$\text{Sign}[x'[t]] + g\,m\,x[t]$$

Здесь учтено, что тангенциальное ускорение определяет скорость изменения модуля скорости шарика, т.е. равняется производной скорости *v2* по времени.

Решение уравнения *eq6* должно удовлетворять следующим начальным условиям:

```
In[33]:= initial2 = {x[0] == -1/2, x'[0] == v20 Cos[α]}
```

$$Out[33]= \left\{x[0] == -\frac{1}{2},\ x'[0] == 33.6161\right\}$$

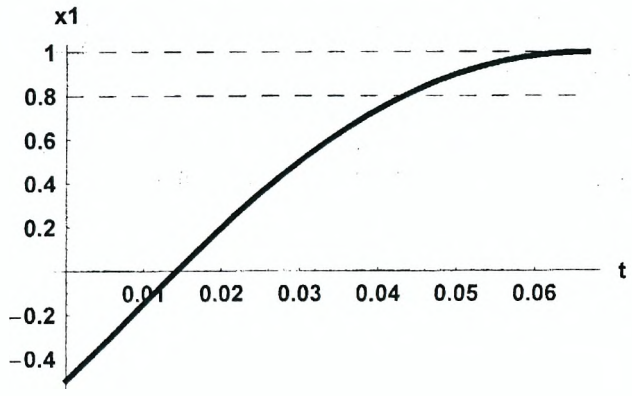
Следует отметить, что скорость шарика в точке *C* направлена под углом α к оси *Ox*. Поэтому начальное значение производной функции *x(t)* должно равняться проекции скорости *v20* на эту ось. Кроме того, предполагается, что в момент прохождения шариком точки *C* включается новый секундомер, и отсчет времени начинается с нуля. Если бы для измерения времени использовался один секундомер, то при определении начальных условий

initial2 необходимо было бы указать вместо $x(0)$ и $x'(0)$ значения $x(t0)$ и $x'(t0)$, где t0 равняется времени движения шарика на первом участке, найденному в *sol3*. Далее находим численное решение дифференциального уравнения *eq6*, используя функцию **NDSolve**.

```
In[34]:= sol6 =
          NDSolve[Join[{eq6 /. dat1}, initial2], x[t], {t, 0, 1}]
NDSolve::mconly :
 For the method NDSolve`LSODA, only machine real
   code is available. Unable to continue with complex
   values or beyond floating point exceptions. More...
Out[34]= {{x[t] →
            InterpolatingFunction[{{0., 0.0667342}}, <>][t]}}
```

Поскольку мы не знаем, в какой момент времени шарик достигает конечной точки *B*, мы пытаемся найти решение на произвольном отрезке, например, $t \in [0, 1]$. Напомним, однако, что определяя уравнения движения шарика, мы использовали уравнение только верхней половины окружности, что соответствует рисунку. При достижении положения $x = 1$ шарик должен переходить на нижнюю половину окружности, для которой используемые уравнения движения оказываются неправильными. Именно поэтому функция **NDSolve** находит решение не на всем заданном интервале, а только до того момента, когда шарик достиг бы положения $x = 1$. Это хорошо видно из графика найденной зависимости *x(t)*.

```
In[35]:= Plot[{x[t] /. sol6[[1]], 0.8 , 1}, {t, 0, 0.0667},
           PlotStyle → {{Thickness[0.011]},
                        {Dashing[{0.03}]},
          {Dashing[{0.03}]}},
           AxesLabel → {"t", "x1"},
           AxesOrigin → {0, 0}, PlotRange → All];
```



По условию конечной точке стержня B соответствует координата $x = 0.8$. Соответствующий момент времени может быть легко найден с помощью функции **FindRoot**.

```
In[36]:= sol7 = FindRoot[(x[t] /. sol6[[1]]) == 0.8, {t, 0.045}]

Out[36]= {t → 0.0433128}
```

Очевидно, полное время движения шарика равняется сумме времен его движения вдоль прямолинейного и криволинейного участков.

```
In[37]:= (t /. sol3) + (t /. sol7)

Out[37]= 0.230877
```

Скорость шарика в момент достижения конечной точки B равна:

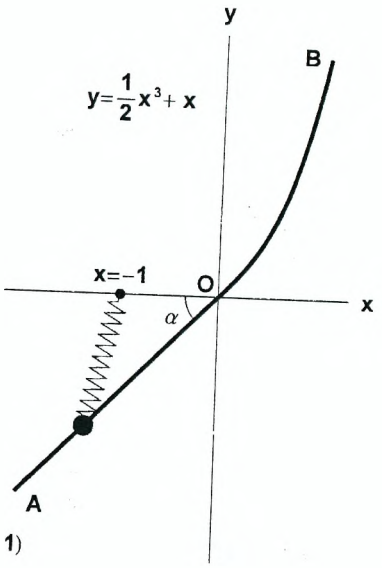
```
In[38]:= vB = v2 /. sol6[[1]] /.
            x'[t] → D[x[t] /. sol6[[1]], t] /. sol7

Out[38]= 29.197
```

Для ее определения мы подставили в выражение для скорости *v2* найденную зависимость *x(t)* и ее производную и выполнили вычисления в момент времени, найденный в решении *sol7*.

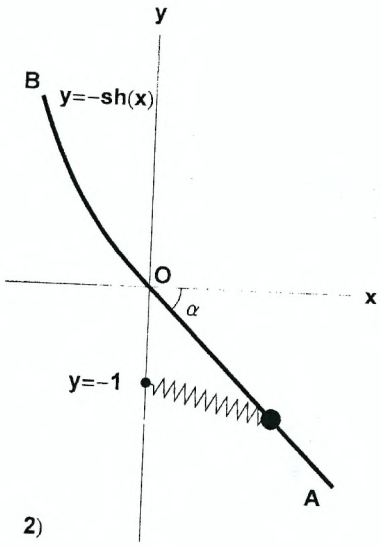
Задание 1

Шарик массой m начинает движение из точки A с нулевой начальной скоростью вдоль стержня, который расположен в вертикальной плоскости xOy и состоит из двух частей: первая часть стержня представляет собой прямолинейный отрезок AO длиной L_1, а вторая – отрезок кривой, уравнение которой $y = y(x)$ задано. Прямолинейный отрезок является частью касательной, проведенной к кривой в точке O. При движении вдоль прямолинейного участка стержня на шарик действует сила упругости со стороны пружины, второй конец которой закреплен в заданной точке. В момент прохождения шариком точки O пружина, находящаяся в недеформированном состоянии, отсоединяется от него. Коэффициент жесткости пружины равен k. Коэффициент трения шарика о стержень равен μ. Выберите один из рисунков и, задав численные значения параметров системы, определите время движения шарика вдоль стержня и его скорость в точке B. Координату x точки B задайте самостоятельно.

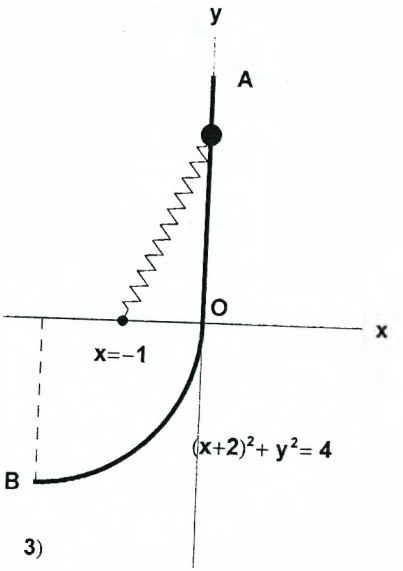
Замечание. Для систем, изображенных на рис. 3, 4, 9, 10, движение шарика на криволинейном участке следует описывать функцией *y(t)*. При этом зависимость *x(t)* получается из уравнения кривой, которое следует переписать в виде: $x = x(y)$.


y
B
$y=\frac{1}{2}x^3+x$
x=−1
O
x
α
A

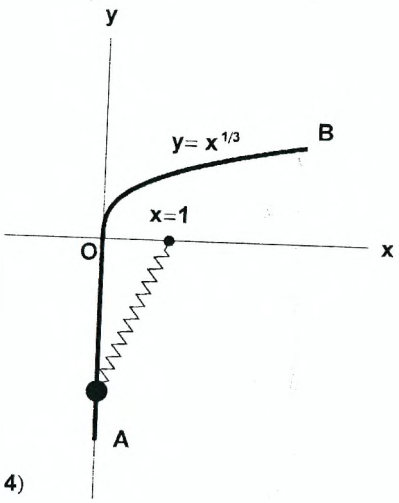
1)


y
B
y=−sh(x)
O
x
α
y=−1
A

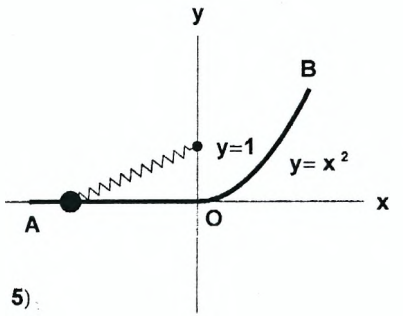
2)


y
A
O
x
x=−1
$(x+2)^2+y^2=4$
B

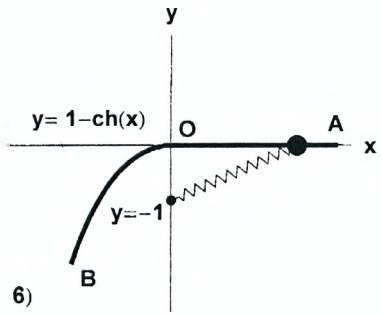
3)


y
$y=x^{1/3}$
B
x=1
O
x
A

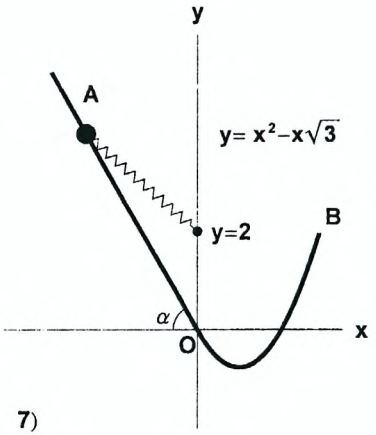
4)


y
B
y=1
y= x ²
A
O
x

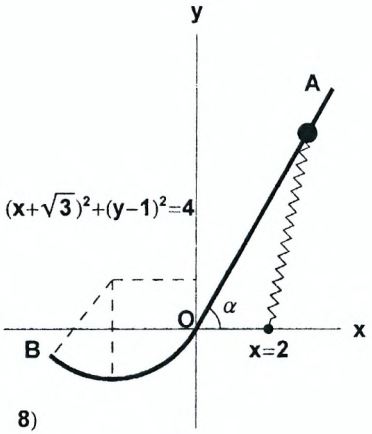
5)


y
y= 1−ch(x)
O
A
x
y=−1
B

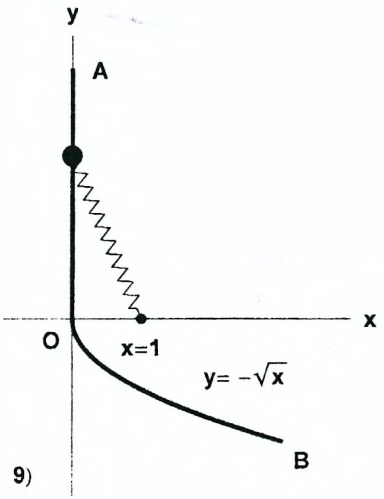
6)


y
A
y= x²−x√3
y=2
B
α
O
x

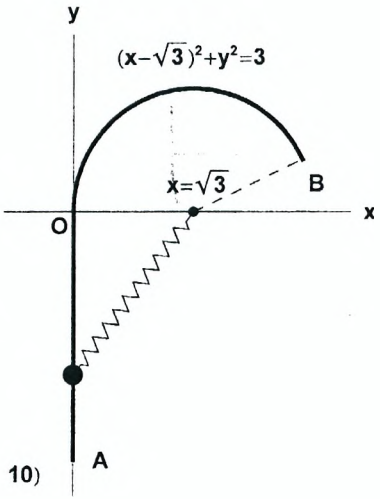
7)


y
A
(x+√3)²+(y−1)²=4
α
O
x
B
x=2

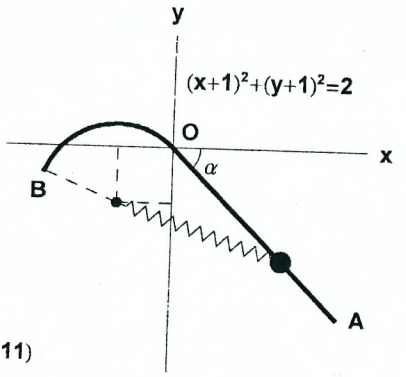
8)


y
A
x
O
x=1
y= −√x
B

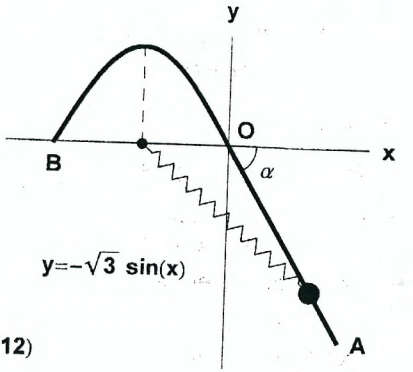
9)


y
(x−√3)²+y²=3
x=√3
B
O
x
A

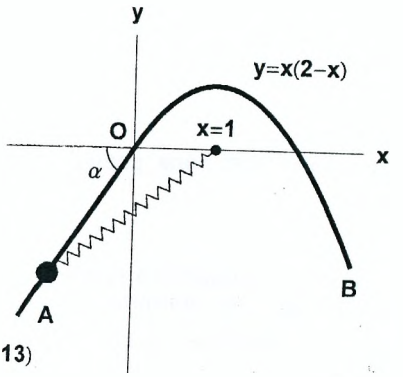
10)


y
(x+1)²+(y+1)²=2
O
x
α
B
A

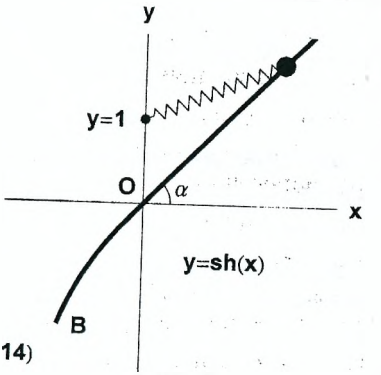
11)


y
O
x
B
α
y=−√3 sin(x)
A

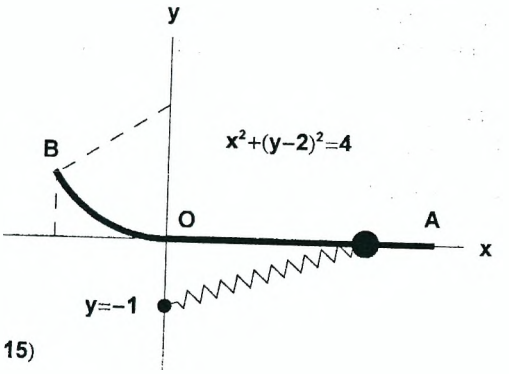
12)


y
y=x(2−x)
O
x=1
x
α
A
B

13)


y
y=1
O
α
x
y=sh(x)
B

14)


y
B
x²+(y−2)²=4
O
A
x
y=−1

15)

II. Движение заряженной частицы в электрическом и магнитном полях

Пример 2

Для описания движения частицы используем прямоугольную декартову систему координат. Пусть точечный заряд q начинает движение из начала координат со скоростью v_0, направленной вдоль оси Oz. Движение происходит в постоянных и однородных электрическом и магнитном полях, причем напряженность электрического поля $\vec{E}$ направлена вдоль оси Ox, а вектор магнитной индукции $\vec{B}$ параллелен плоскости yOz. Положение частицы в пространстве определяется радиусом-вектором $\vec{r}$, проекции которого на оси Ox, Oy, Oz зависят от времени. Поэтому радиус-вектор $\vec{r}$ зададим в виде следующего списка:

```
In[39]:= Clear["Global`*"];
         r = {x[t], y[t], z[t]};
```

Векторы $\vec{E}$ и $\vec{B}$ зададим в виде:

```
In[41]:= E0 = {Ex, 0, 0};
         B0 = {0, By, Bz};
```

Сила, действующая на заряд со стороны электрического поля, равна:

```
In[43]:= Fe = q E0

Out[43]= {Ex q, 0, 0}
```

Сила Лоренца, действующая на заряд со стороны магнитного поля, пропорциональна векторному произведению скорости заряда и магнитной индукции $\vec{B}$. Используя функцию **Cross** для вычисления векторного произведения, получаем для силы Лоренца следующее выражение:

```
In[44]:= Fm = q Cross[D[r, t], B0 ]

Out[44]= {q (Bz y'[t] - By z'[t]), -Bz q x'[t], By q x'[t]}
```

Записывая второй закон Ньютона, получаем уравнение движение заряда в виде:

```
In[45]:= eq = m D[r, {t, 2}] == Fe + Fm // Thread

Out[45]= {m x''[t] == Ex q + q (Bz y'[t] - By z'[t]),
           m y''[t] == -Bz q x'[t], m z''[t] == By q x'[t]}
```

Отметим, что применение функции **Thread** позволяет приравнять соответствующие элементы списков в левой и правой частях системы *eq* и получить список из трех уравнений. Для решения полученной системы определяем начальные условия в виде:

In[46]:=
```
initial = {x[0] == 0, y[0] == 0,
    z[0] == 0, x'[0] == 0, y'[0] == 0, z'[0] == v0};
```

Решение системы *eq* с начальными условиями *initial* может быть найдено в символьном виде с помощью функции **DSolve**. В результате получаем:

In[47]:=
```
sol = DSolve[Join[eq, initial],
    {x[t], y[t], z[t]}, t] // Simplify
```

Out[47]:=

$$\left\{\left\{x[t] \to \frac{e^{-\frac{\sqrt{-(By^2+Bz^2)\,q^2}\,t}{m}}\left(-1+e^{\frac{\sqrt{-(By^2+Bz^2)\,q^2}\,t}{m}}\right)^2 m\,(-Ex+By\,v0)}{2\,(By^2+Bz^2)\,q},\right.\right.$$

$$y[t] \to -\left(Bz\,e^{-\frac{\sqrt{-(By^2+Bz^2)\,q^2}\,t}{m}} q^2\left(m-e^{\frac{2\sqrt{-(By^2+Bz^2)\,q^2}\,t}{m}} m+\right.\right.$$

$$\left.2\,e^{\frac{\sqrt{-(By^2+Bz^2)\,q^2}\,t}{m}}\sqrt{-(By^2+Bz^2)\,q^2}\,t\right)$$

$$\left.(-Ex+By\,v0)\right) \Big/ \left(2\,(-(By^2+Bz^2)\,q^2)^{3/2}\right),$$

$$z[t] \to -\left(e^{-\frac{\sqrt{-(By^2+Bz^2)\,q^2}\,t}{m}} q^2\left(By\,Ex\left(m-e^{\frac{2\sqrt{-(By^2+Bz^2)\,q^2}\,t}{m}} m+\right.\right.\right.$$

$$\left.2\,e^{\frac{\sqrt{-(By^2+Bz^2)\,q^2}\,t}{m}}\sqrt{-(By^2+Bz^2)\,q^2}\,t\right)+$$

$$By^2\left(-1+e^{\frac{2\sqrt{-(By^2+Bz^2)\,q^2}\,t}{m}}\right) m\,v0+$$

$$\left.\left.2\,Bz^2\,e^{\frac{\sqrt{-(By^2+Bz^2)\,q^2}\,t}{m}}\sqrt{-(By^2+Bz^2)\,q^2}\,t\,v0\right)\right) \Big/$$

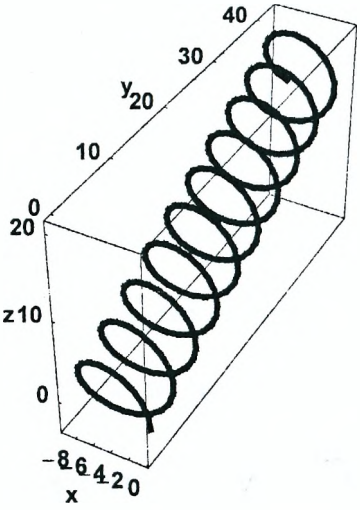
$$\left.\left.\left(2\,(-(By^2+Bz^2)\,q^2)^{3/2}\right)\right\}\right\}$$

Чтобы изобразить траекторию движения заряда, определяем численные значения параметров системы.

In[48]:=
```
dat1 = {m → 10^-6, q → 10^-9,
    Ex → 700, Bz → 300, By → 2000, v0 → 10};
```

Используя функцию **ParametricPlot3D**, получаем траекторию в виде:

In[49]:=
```
ParametricPlot3D[
  {x[t], y[t], z[t], {Hue[0], Thickness[0.02]}} /.
    sol[[1]] /. dat1 , {t, 0, 30},
  PlotPoints → 200, AxesLabel → {"x", "y", "z"}];
```



Как видим, движение заряда происходит вдоль кривой, похожей на спираль. Чтобы выяснить характер этого движения, введем три единичных взаимно перпендикулярных вектора *n1*, *n2*, *n3*. Вектор *n3*, направленный вдоль магнитной индукции $\vec{B}$, получим при делении вектора *B0* на его модуль.

In[50]:=
```
n3 = B0 / Sqrt[B0.B0]
```

Out[50]=
$$\left\{0,\ \frac{By}{\sqrt{By^2 + Bz^2}},\ \frac{Bz}{\sqrt{By^2 + Bz^2}}\right\}$$

Вектор *n1* получается как нормированное на единицу векторное произведение напряженности электрического поля *E0* на вектор *n3*.

In[51]:=
```
n1 = 1 / Sqrt[Cross[E0, n3].Cross[E0, n3]] Cross[E0, n3] //
     Simplify[#, Ex > 0] &
```

Out[51]=
$$\left\{0,\ -\frac{Bz}{\sqrt{By^2 + Bz^2}},\ \frac{By}{\sqrt{By^2 + Bz^2}}\right\}$$

Заметим,что единичный вектор *n1* перпендикулярен как вектору $\vec{E}$, так и вектору $\vec{B}$, и направлен вдоль векторного произведения $\vec{E} \times \vec{B}$. Чтобы векторы *n1*, *n2*, *n3* составляли правую тройку векторов, третий вектор *n2* определяем как векторное произведение *n3* и *n1*.

```
In[52]:= n2 = Cross[n3, n1] // Simplify
Out[52]= {1, 0, 0}
```

Чтобы выяснить характер движения заряда вдоль магнитного поля, вычислим проекцию радиус-вектора $\vec{r}$, найденного как решение уравнения движения, на направление *n3*.

```
In[53]:= x3 = r.n3 /. sol[[1]] // Simplify
```

$$Out[53]= \frac{Bz\ t\ v0}{\sqrt{By^2 + Bz^2}}$$

Поскольку *x3* является линейной функцией времени, происходит равномерное движение частицы вдоль магнитного поля со скоростью, равной составляющей начальной скорости v_0 вдоль вектора $\vec{B}$. Очевидно так и должно быть, потому что в рассматриваемом случае векторы $\vec{E}$ и $\vec{B}$ взаимно перпендикулярны, а магнитное поле не изменяет величину скорости частицы. Действие магнитного поля сводится лишь к изменению направления перпендикулярной по отношению к $\vec{B}$ составляющей вектора скорости частицы.

Вычислим теперь проекции радиус-вектора $\vec{r}$ на направления *n1* и *n2*. В полученных выражениях преобразуем экспоненциальные функции в тригонометрические, используя функцию **ExpToTrig**.

```
In[54]:= x1 = r.n1 /. sol[[1]] // ExpToTrig //
           Simplify[#, By > 0 && Bz > 0 && q > 0 && m > 0] &
```

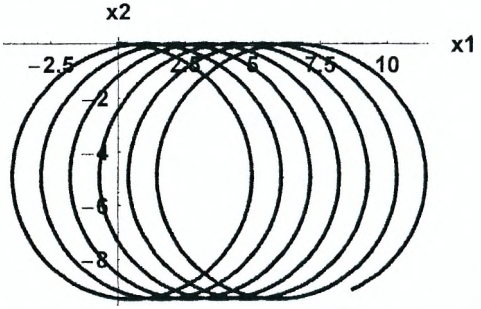
$$Out[54]= \frac{\sqrt{By^2 + Bz^2}\ Ex\ q\ t - m\ (Ex - By\ v0)\ Sin\left[\frac{\sqrt{By^2+Bz^2}\ q\ t}{m}\right]}{(By^2 + Bz^2)\ q}$$

```
In[55]:= x2 = r.n2 /. sol[[1]] // ExpToTrig //
           Simplify[#, By > 0 && Bz > 0 && q > 0 && m > 0] &
```

$$Out[55]= \frac{2\ m\ (Ex - By\ v0)\ Sin\left[\frac{\sqrt{By^2+Bz^2}\ q\ t}{2\ m}\right]^2}{(By^2 + Bz^2)\ q}$$

Теперь можно построить проекцию траектории частицы на плоскость, перпендикулярную к магнитному полю. Используя функцию **ParametricPlot**, получаем:

```
In[56]:= ParametricPlot[
           {x1, x2} /. sol[[1]] /. dat1 // Evaluate,
           {t, 0, 20}, PlotPoints → 200,
           AxesLabel → {"x1", "x2"}, AspectRatio → Automatic,
           PlotStyle → {Thickness[0.01], Hue[0]}];
```



Полученные выражения *x1*, *x2* и график показывают, что в рассматриваемой плоскости происходит вращение частицы с угловой скоростью $\omega = \frac{q|B|}{m}$, на которое накладывается равномерное движение вдоль вектора *n1*. Такое движение называется *электрическим дрейфом*, а величина скорости дрейфа равна:

In[57]:= **Coefficient[x1, t]**

$$Out[57]= \frac{Ex}{\sqrt{By^2 + Bz^2}}$$

Таким образом, в рассматриваемом случае движение заряженной частицы представляет собой наложение трех движений: равномерного движения в направлении вектора $\vec{B}$, равномерного вращения вокруг вектора $\vec{B}$ с угловой скоростью $\omega = \frac{q|B|}{m}$ и электрического дрейфа в направлении, которое определяется векторным произведением $\vec{E} \times \vec{B}$.

Задание 2

Точечный заряд *q* начинает движение из начала координат со скоростью v_0. Движение происходит в постоянных и однородных электрическом и магнитном полях. Выполните следующие задания:

1. Задайте векторы напряженности электрического поля $\vec{E}$, магнитной индукции $\vec{B}$ и начальной скорости $\vec{v}_0$ произвольным образом.
2. Определите уравнение движения заряженной частицы в электрическом и магнитном полях при заданных начальных условиях и найдите его решение в символьном виде.
3. Задайте численные значения параметров системы и постройте траекторию движения частицы.
4. Проанализируйте характер движения частицы вдоль направления магнитного поля и в плоскости, перпендикулярной к вектору $\vec{B}$, и найдите скорость электрического дрейфа.

Лабораторная работа № 6
Моделирование механических колебаний

Цель работы:

1) получить уравнение гармонических колебаний при движении тела вдоль заданной прямой и найти его решение;

2) изучить влияние нелинейных членов в уравнении движения на свободные и вынужденные колебания тела;

3) познакомиться с явлением параметрического резонанса.

Как известно, *дифференциальным уравнением гармонических колебаний* называется уравнение вида:

$$\frac{d^2x}{dt^2} + \omega^2 x = 0,$$

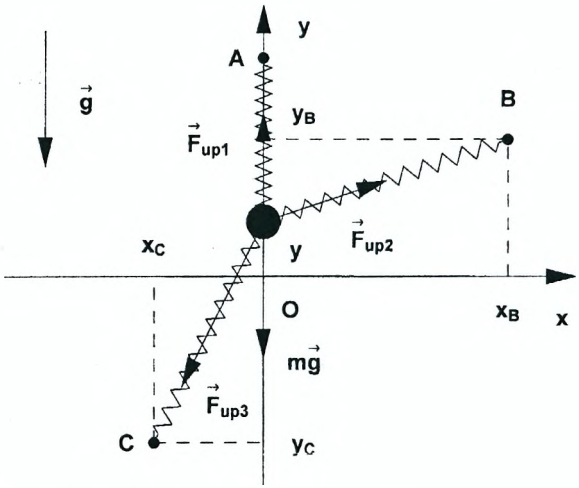
где ω – циклическая частота колебаний. Оно описывает, например, движение тела вблизи своего устойчивого положения равновесия и имеет общее решение вида:

$$x = A\cos(\omega t + \varphi),$$

где A и φ – соответственно амплитуда и начальная фаза колебаний. Чтобы тело могло совершать гармонические колебания, необходимо выполнение двух условий: во-первых, должно существовать положение равновесия тела, и во-вторых, отклонение тела от такого положения должно приводить к возникновению силы вида $F = -kx$, где k – положительный коэффициент, стремящейся вернуть тело обратно. Эти два условия означают, что устойчивому положению равновесия тела соответствует точка минимума его потенциальной энергии. Следует отметить, что в реальных системах зависимость возвращающей силы F от смещения тела из положения равновесия x является линейной только при малых значениях x. При возрастании амплитуды колебаний начинает проявляться нелинейность зависимости $F(x)$, приводящая к качественно новым особенностям движения. Чтобы исследовать эти особенности, рассмотрим следующую механическую систему.

Пример 1

Тело массой m может скользить без трения вдоль вертикальной оси Oy. К телу прикреплены три пружины, жесткости которых равны k_A, k_B, k_C, а вторые концы закреплены в точках A, B и C соответственно. Длины пружин в недеформированном состоянии L_{A0}, L_{B0}, L_{C0}, а также координаты точек A, B и C заданы. На рисунке тело изображено в произвольной точке с координатой y вместе с силами, действующими на него.



Сила тяжести $m\vec{g}$ направлена вертикально вниз. В зависимости от выбранных параметров системы каждая из пружин может быть как сжата, так и растянута. Направления сил упругости $\vec{F}_{up1}$, $\vec{F}_{up2}$, $\vec{F}_{up3}$ на рисунке указаны в предположении, что все пружины растянуты.

Обозначая координаты точек *A*, *B* и *C* через *yA*, *xB*, *yB*, *xC*, *yC*, определяем длины пружин в тот момент, когда тело находится в точке с координатой *y*.

In[6]:=
```
Clear["Global`*"];
La = yA - y;
Lb = Sqrt[xB^2 + (yB - y)^2];
Lc = Sqrt[xC^2 + (y - yC)^2];
```

Тогда потенциальные энергии пружин будут равны:

In[10]:=
```
Ua = kA/2 (La - La0)^2
```

Out[10]=
$$\frac{1}{2}\,(-La0 - y + yA)^2\,k_A$$

In[11]:=
```
Ub = kB/2 (Lb - Lb0)^2
```

Out[11]=
$$\frac{1}{2}\left(-Lb0 + \sqrt{xB^2 + (-y + yB)^2}\right)^2 k_B$$

In[12]:=
```
Uc = kC/2 (Lc - Lc0)^2
```

Out[12]=
$$\frac{1}{2}\left(-Lc0 + \sqrt{xC^2 + (y - yC)^2}\right)^2 k_C$$

Будем считать, что потенциальная энергия тела, находящегося в однородном поле силы тяжести, равна нулю на прямой $y = 0$. Тогда соответствующая потенциальная энергия тела равна:

```
In[13]:= Ug = m g y
```

$$\text{Out[13]= } g\,m\,y$$

Полная потенциальная энергия системы есть следующая сумма:

```
In[14]:= U = Ua + Ub + Uc + Ug
```

$$\text{Out[14]= } g\,m\,y + \frac{1}{2}\,(-La0 - y + yA)^2\,k_A + \frac{1}{2}\left(-Lb0 + \sqrt{xB^2 + (-y + yB)^2}\right)^2 k_B + \frac{1}{2}\left(-Lc0 + \sqrt{xC^2 + (y - yC)^2}\right)^2 k_C$$

Для определения положения устойчивого равновесия тела необходимо найти точку, в которой потенциальная энергия принимает минимальное значение. Параметры системы нужно выбрать так, чтобы такое положение существовало. Зададим их, например, в виде следующего списка правил замены.

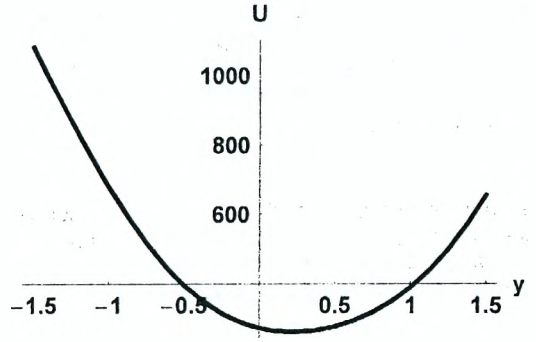
```
In[15]:= values1 = {m → 2, g → 9.8, kA → 250, kB → 300,
                    kC → 300, La0 → 0.7, Lb0 → 1.2, Lc0 → 0.8};
```

Определим также координаты точек *A*, *B* и *C* :

```
In[16]:= values2 =
                    {yA → 2, xB → 0.8, yB → 0.4, xC → -0.4, yC → -1.3};
```

Теперь можно построить график зависимости потенциальной энергии от координаты тела *y*.

```
In[17]:= Plot[U /. values1 /. values2, {y, -1.5, 1.5},
                    PlotStyle → Thickness[0.01], AxesLabel → {"y", "U"}];
```



Как видим, при заданных значениях параметров потенциальная энергия имеет минимум в окрестности точки $y = 0.2$. Найдем точное значение координаты этой точки *yr* из условия, что производная потенциальной энергии обращается в ней в ноль.

In[18]:=
```
eq1 = D[U, y] == 0
```

Out[18]=

$$g\,m - (-La0 - y + yA)\,k_A - \frac{(-y + yB)\left(-Lb0 + \sqrt{xB^2 + (-y + yB)^2}\right)k_B}{\sqrt{xB^2 + (-y + yB)^2}} + \frac{\left(-Lc0 + \sqrt{xC^2 + (y - yC)^2}\right)(y - yC)\,k_C}{\sqrt{xC^2 + (y - yC)^2}} == 0$$

In[19]:=
```
yr = y /. FindRoot[
        eq1 /. values1 /. values2 // Evaluate, {y, 0.2}]
```

Out[19]=
```
0.223494
```

Потенциальная энергия системы является довольно сложной функцией координаты *y*. Предположим, что тело может двигаться только в малой окрестности положения равновесия, так что выполнено условие: $|y - yr| << 1$. В этом случае энергию системы можно разложить в ряд по *y* в окрестности точки *yr* и ограничиться членами второго порядка. В результате получаем:

In[20]:=
```
U1 = Normal[
        Series[U /. values1 /. values2, {y, yr, 2}] ] // Chop
```

Out[20]=
```
261.109 + 210.57 (-0.223494 + y)^2
```

Суммарная сила, действующая на тело, равняется производной потенциальной энергии, взятой с обратным знаком. Поэтому уравнение движения тела запишем в виде:

In[21]:=
```
eq2 = m y''[t] == -(D[U1, y] /. y → y[t]) /. values1
```

Out[21]=
```
2 y''[t] == -421.14 (-0.223494 + y[t])
```

Так как в дифференциальном уравнении функция должна быть записана со своим аргументом, в правой части *eq2* выполнена соответствующая подстановка. Полученное уравнение является дифференциальным уравнением гармонических колебаний, а коэффициент при *y*(*t*) в правой части этого уравнения есть $(-m\,\omega_0^2)$, где ω_0 – циклическая частота колебаний.

In[22]:=
```
ω0 = (-1/m Coefficient[eq2[[2]], y[t]])^(1/2) /. values1
```

Out[22]=
```
14.511
```

Наличие постоянного слагаемого в правой части *eq2* объясняется тем, что колебания тела происходят около положения равновесия *yr*. Очевидно, постоянная функция *y* = *yr* является решением уравнения *eq2*.

In[23]:=
```
eq2 /. y → (yr &)
```

Out[23]=
```
True
```

Теперь определим точное уравнение движения *eq3*, используя выражение для потенциальной энергии *U*.

In[24]:=
```
eq3 =
  m y''[t] == -(D[U, y] /. y → y[t]) /. values1 /. values2
```
Out[24]=
$$2\,y''[t] == -19.6 + \frac{300\left(-1.2 + \sqrt{0.64 + (0.4 - y[t])^2}\right)(0.4 - y[t])}{\sqrt{0.64 + (0.4 - y[t])^2}} + 250\,(1.3 - y[t]) - \frac{300\,(1.3 + y[t])\left(-0.8 + \sqrt{0.16 + (1.3 + y[t])^2}\right)}{\sqrt{0.16 + (1.3 + y[t])^2}}$$

Ясно, что уравнение *eq3* может быть решено только численно.

Сначала рассмотрим случай малых колебаний. Для этого определяем начальные условия так, чтобы было выполнено соотношение: $|y_0 - yr| << 1$.

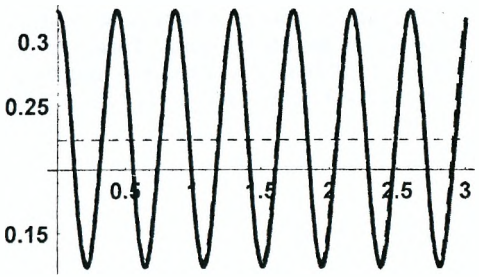
In[25]:=
```
initial1 = {y[0] == yr + 0.1, y'[0] == 0};
```
Находим решения обоих дифференциальных уравнений при этих начальных условиях.

In[26]:=
```
sol2 = DSolve[Join[{eq2}, initial1], y[t], t] // Chop
```
Out[26]=
```
{{y[t] → 0.223494 + 0.1 Cos[14.511 t]}}
```

In[27]:=
```
sol3 = NDSolve[Join[{eq3}, initial1], y[t], {t, 0, 5}]
```
Out[27]=
```
{{y[t] → InterpolatingFunction[{{0., 5.}}, <>][t]}}
```

Легко видеть, в первом случае *y(t)* испытывает гармонические колебания около равновесного положения *y = yr*. Для сравнения найденных решений изобразим их на одной координатной плоскости и отметим уровень *yr* штриховой линией.

In[28]:=
```
Plot[
  {y[t] /. sol2[[1]], y[t] /. sol3[[1]], yr}, {t, 0, 3},
  PlotStyle → {{Thickness[0.01], RGBColor[1, 0, 0]},
    {Dashing[{0.03}], Thickness[0.01],
     RGBColor[0, 0, 1]}, {Dashing[{0.02}]}}];
```

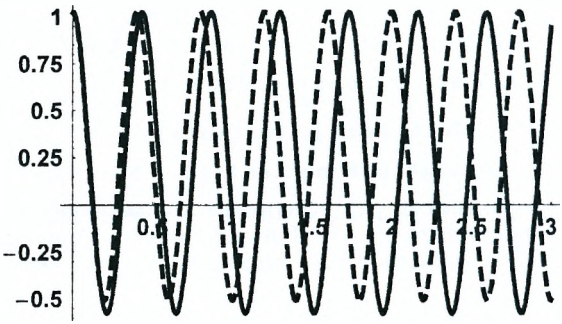


Как видим, в случае малых колебаний влияние нелинейностей на движение тела мало, и оба графика совпадают. Таким образом, малые колебания тела около положения равновесия являются гармоническими. Теперь увеличим начальное смещение тела и опять найдем решения уравнений движения.

```
In[29]:= initial2 = {y[0] == yr + 0.8, y'[0] == 0};

In[30]:= sol4 = DSolve[Join[{eq2}, initial2], y[t], t] // Chop

Out[30]= {{y[t] → 0.223494 + 0.8 Cos[14.511 t]}}

In[31]:= sol5 = NDSolve[Join[{eq3 }, initial2], y[t], {t, 0, 5}]

Out[31]= {{y[t] → InterpolatingFunction[{{0., 5.}}, <>][t]}}
```

Графики найденных решений имеют вид:

```
In[32]:= Plot[{y[t] /. sol4[[1]], y[t] /. sol5[[1]]},
           {t, 0, 3}, PlotStyle →
            {{Thickness[0.01], RGBColor[1, 0, 0]}, {Dashing[
               {0.02}], Thickness[0.01], RGBColor[0, 0, 1]}}];
```



Оба решения опять описывают некоторый колебательный процесс, но только решение *sol4* соответствует гармоническим колебаниям. Из графика видно, что учет нелинейных членов в уравнениях движения приводит к изменению периода колебаний. Найдя решение точного уравнения движения *eq3* при другом значении начального отклонения шарика от положения равновесия, например, при условии $y[0] = yr + 1$, легко убедиться в том, что период нелинейных колебаний зависит от их амплитуды. Таким образом, с увеличением амплитуды нелинейные члены в уравнении движения становятся весьма существенными, и колебания перестают быть гармоническими.

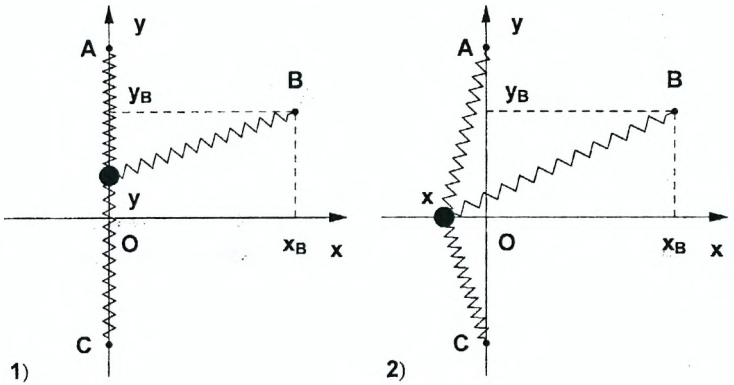
Задание 1

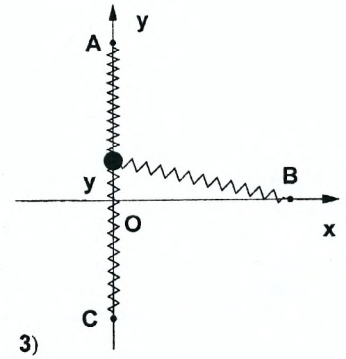
Тело массой m может скользить без трения вдоль одной из координатных осей в вертикальной плоскости xOy. К телу прикреплены три пружины, жесткости которых равны k_A, k_B, k_C, а вторые концы закреплены в точках A, B и C соответственно (см. рисунок).

1. Запишите выражение для потенциальной энергии системы как функции координаты x или y тела (см. Пример 1). Задайте численные значения параметров k_A, k_B, k_C, длины пружин в недеформированном состоянии L_{A0}, L_{B0}, L_{C0}, а также координаты точек A, B и C и постройте график потенциальной энергии как функции координаты. Убедитесь, что при выбранных значениях параметров системы у потенциальной энергии имеется точка минимума. Если это условие не выполняется, численные значения некоторых параметров следует изменить.

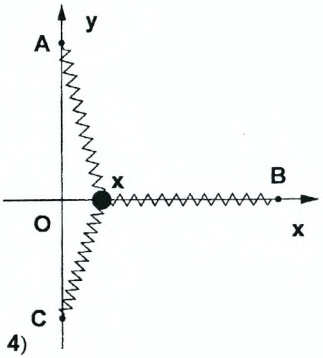
2. Разложите потенциальную энергию в ряд в окрестности точки минимума с точностью до второго порядка. Получите дифференциальное уравнение гармонических колебаний тела, а также точное уравнение движения. Определите циклическую частоту гармонических колебаний ω_0.

3. Задайте начальные условия и найдите решения дифференциального уравнения гармонических колебаний и точного уравнения движения тела. Постройте графики найденных решений. Повторите вычисления при других начальных условиях и покажите, что с ростом амплитуды колебания перестают быть гармоническими.

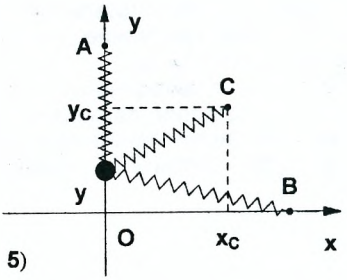



y
A
y
B
O
x
C

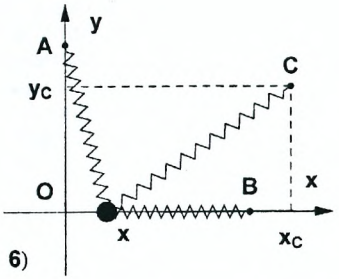
3)


y
A
x
B
O
x
C

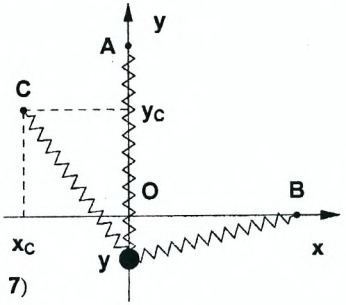
4)


y
A
C
yC
y
B
O
xC
x

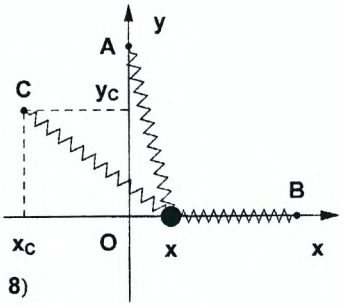
5)


y
A
C
yC
O
B
x
x
xC

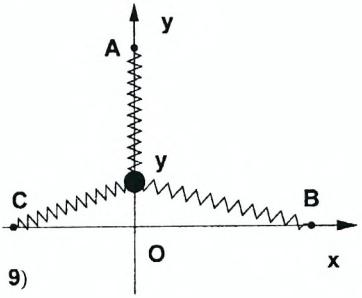
6)


y
A
C
yC
O
B
xC
y
x

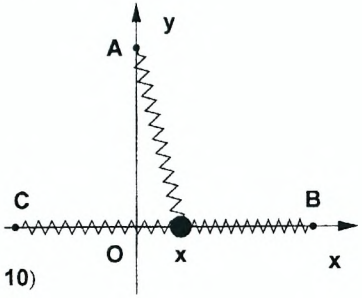
7)


y
A
C
yC
B
xC
O
x
x

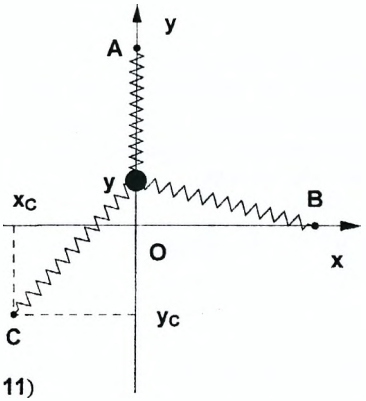
8)


y
A
y
C
B
O
x

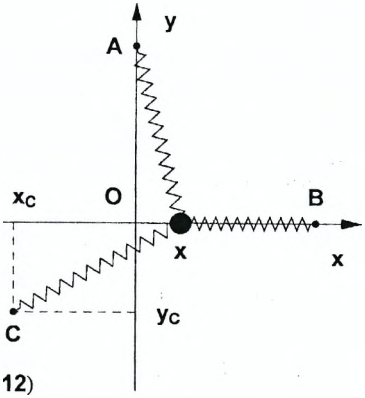
9)


y
A
C
B
O
x
x

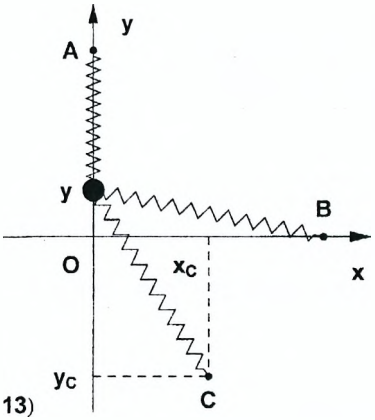
10)


y
A
y
x_C
B
O
x
C
y_C

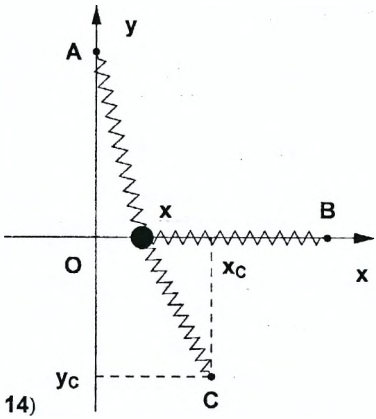
11)


y
A
x_C
O
B
x
x
C
y_C

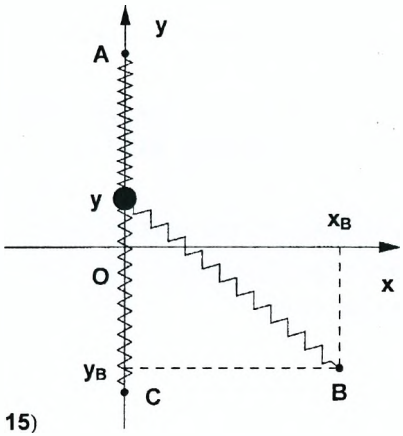
12)


y
A
y
B
O
x_C
x
y_C
C

13)


y
A
x
B
O
x_C
x
y_C
C

14)


y
A
y
x_B
O
x
y_B
C
B

15)

Пример 2

Предположим, что при движении тела на него действует дополнительная сила – сила вязкого трения, пропорциональная его скорости и направленная против нее: $\vec{F}_{tr} = -b\vec{v}$, где b – постоянная. Добавляя в правых частях уравнений *eq2* и *eq3* соответствующие слагаемые, получаем линейное и точное уравнения движения при наличии вязкого трения:

In[33]:=
```
eq4 =
  m y''[t] == -b y'[t] - (D[U1, y] /. y → y[t]) /. values1
```

Out[33]=
$$2\,y''[t] == -421.14\,(-0.223494 + y[t]) - b\,y'[t]$$

In[34]:=
```
eq5 =
  m y''[t] == -b y'[t] - (D[U, y] /. y → y[t]) /. values1 /.
   values2
```

Out[34]=
$$2\,y''[t] == -19.6 + \frac{300\left(-1.2 + \sqrt{0.64 + (0.4 - y[t])^2}\right)(0.4 - y[t])}{\sqrt{0.64 + (0.4 - y[t])^2}} + 250\,(1.3 - y[t]) - \frac{300\,(1.3 + y[t])\left(-0.8 + \sqrt{0.16 + (1.3 + y[t])^2}\right)}{\sqrt{0.16 + (1.3 + y[t])^2}} - b\,y'[t]$$

Коэффициент пропорциональности b в выражении для силы вязкого трения зададим следующим правилом замены:

In[35]:=
```
values3 = {b → 0.5};
```

Используя начальные условия *initial1*, находим решения обоих дифференциальных уравнений в случае малых колебаний.

In[36]:=
```
sol6 =
  DSolve[Join[{eq4 /. values3}, initial1], y[t], t] //
   Simplify
```

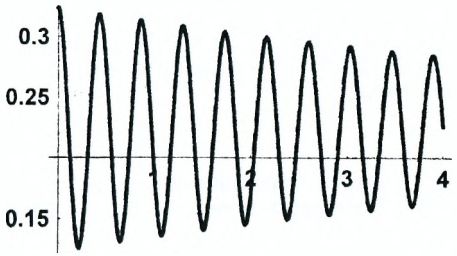
Out[36]=
$$\{\{y[t] \to 0.223494 + 0.1\,e^{-0.125\,t}\,\text{Cos}[14.5105\,t] + 0.000861445\,e^{-0.125\,t}\,\text{Sin}[14.5105\,t]\}\}$$

In[37]:=
```
sol7 = NDSolve[
   Join[{eq5 /. values3}, initial1], y[t], {t, 0, 10}]
```

Out[37]=
$$\{\{y[t] \to \text{InterpolatingFunction}[\{\{0.,\ 10.\}\},\ <>][t]\}\}$$

Строим графики полученных решений.

In[38]:=
```
Plot[{y[t] /. sol6[[1]], y[t] /. sol7[[1]]},
 {t, 0, 4}, PlotStyle →
  {{Thickness[0.01], RGBColor[1, 0, 0]}, {Thickness[
    0.01], Dashing[{0.04}], RGBColor[0, 0, 1]}}];
```



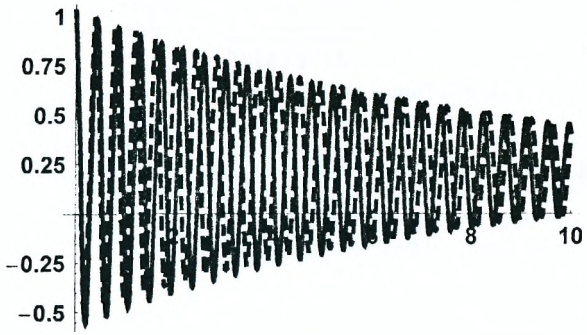
Графики показывают, что колебания являются затухающими. Так как в рассматриваемом случае начальное отклонение тела от положения равновесия мало, то графики обоих решений совпадают. Теперь найдем решения уравнений движения при начальных условиях *initial2*.

In[39]:=
```
sol8 =
 DSolve[Join[{eq4 /. values3}, initial2], y[t], t] //
  Simplify
```

In[40]:=
```
sol9 = NDSolve[Join[{eq5 /. values3}, initial2],
 y[t], {t, 0, 10}, MaxSteps → 5000]
```

Соответствующие графики имеют вид:

In[41]:=
```
Plot[{y[t] /. sol8[[1]], y[t] /. sol9[[1]]}, {t, 0, 10},
 PlotStyle → {{Thickness[0.01], RGBColor[1, 0, 0]},
   {Thickness[0.01], Dashing[{0.02}],
    RGBColor[0, 0, 1]}}, PlotPoints → 50];
```



Отметим, что на начальном этапе, когда амплитуда колебаний велика, наблюдается заметное отличие решений линейного и точного уравнений

движения. Однако, по мере затухания колебаний, это отличие постепенно пропадает.

Задание 2

1. Используя результаты из Задания 1, получите уравнение затухающих колебаний и точное уравнение движения тела, если на него в процессе движения действует сила вязкого трения: $\vec{F}_{tr} = -b\vec{v}$ (см. Пример 2).

2. Решите полученные уравнения при различных начальных условиях, постройте графики найденных решений и сделайте вывод в влиянии силы вязкого трения на колебания тела.

Пример 3

Предположим, что тело имеет электрический заряд q, а вся система находится в однородном электрическом поле, вектор напряженности которого $\vec{E}$ параллелен оси Oy, т.е. имеет вид: $\vec{E} = (0, E, 0)$, и изменяется во времени по закону: $E = E_0 \sin(\omega t)$. Тогда на тело будет действовать дополнительная синусоидальная сила $F = mA_0 \sin(\omega t)$, где $A_0 = qE_0/m$, направленная вдоль оси Oy. Поэтому линейное и точное уравнения движения примут вид:

In[42]:=
```
eq11 = m y''[t] ==
        -(D[U1, y] /. y → y[t]) + m A0 Sin[ω t] /. values1
```

Out[42]= $2\,y''[t] == 2\,\text{Sin}[t\,\omega]\,A_0 - 421.14\,(-0.223494 + y[t])$

In[43]:=
```
eq12 = m y''[t] == -(D[U, y] /. y → y[t]) + m A0 Sin[ω t] /.
       values1 /. values2
```

Out[43]=

$$2\,y''[t] == -19.6 + 2\,\text{Sin}[t\,\omega]\,A_0 + \frac{300\left(-1.2 + \sqrt{0.64 + (0.4 - y[t])^2}\right)(0.4 - y[t])}{\sqrt{0.64 + (0.4 - y[t])^2}} + 250\,(1.3 - y[t]) - \frac{300\,(1.3 + y[t])\left(-0.8 + \sqrt{0.16 + (1.3 + y[t])^2}\right)}{\sqrt{0.16 + (1.3 + y[t])^2}}$$

Как известно, при воздействии на колебательную систему внешней синусоидальной силы, частота которой совпадает с собственной частотой системы, происходит явление *резонанса*, когда амплитуда колебаний системы начинает возрастать. При этом линейная и нелинейная системы ведут себя по-разному. Чтобы убедиться в этом, найдем решения уравнений *eq11* и *eq12* при различных частотах внешней силы. Сначала предположим, что внешняя сила достаточно мала, и амплитуда A_0 также мала.

In[44]:=
```
values4 = {A0 → 0.8};
```

Пусть в начальный момент времени тело покоится в положении равновесия.

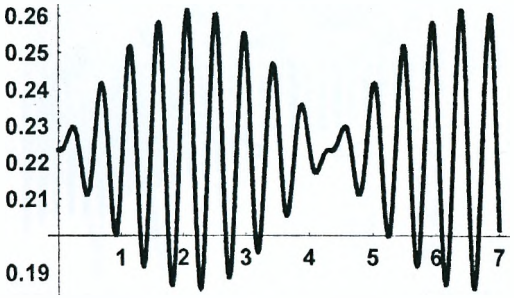
In[45]:=
```
initial4 = {y[0] == yr, y'[0] == 0};
```

Находим решения дифференциальных уравнений *eq11* и *eq12* при частоте внешней силы ω, близкой к собственной частоте ω_0, но не равной ей.

```
In[48]:= sol11 = NDSolve[Join[{eq11 /. ω → 0.9 ω0 /. values4 },
           initial4], y[t], {t, 0, 10}] ;

In[49]:= sol12 = NDSolve[
          Join[{eq12 /. ω → 0.9 ω0 /. values4}, initial4],
          y[t], {t, 0, 10}, MaxSteps → 3000];
```

Строим графики найденных решений.

```
In[50]:= Plot[
          {y[t] /. sol11[[1]], y[t] /. sol12[[1]]}, {t, 0, 7},
          PlotStyle → {{Thickness[0.01], RGBColor[1, 0, 0]},
            {Thickness[0.01], Dashing[{0.02}],
             RGBColor[0, 0, 1]}}, PlotPoints → 50];
```



Поскольку частота внешней силы ω не равна собственной частоте системы ω_0, резонанса не происходит. При этом колебания остаются малыми, и решения линейного и точного уравнений движения практически совпадают. Решение линейного уравнения, описывающего отклонение груза от положения равновесия, можно найти и в символьном виде:

```
In[51]:= DSolve[{y''[t] + ω0^2 y[t] == A0 Sin[ω t],
           y[0] == 0, y'[0] == 0}, y[t], t]  // Simplify
```

$$Out[51]= \left\{\left\{y[t] \to \frac{(\omega 0\, \mathrm{Sin}[t\,\omega] - \omega\, \mathrm{Sin}[t\,\omega 0])\, A_0}{-\omega^2\,\omega 0 + \omega 0^3}\right\}\right\}$$

Оно представляет собой наложение двух колебаний – с собственной частотой системы ω_0 и с частотой внешней силы ω. Если эти частоты достаточно близки, то результирующие колебания представляют собой биения, что и показано на полученном выше графике.

Теперь находим решения уравнений движения в резонансном случае, когда $\omega = \omega_0$, и строим соответствующие графики.

```
In[52]:= sol13 =
          NDSolve[Join[{eq11 /. ω → ω0 /. values4 }, initial4],
           y[t], {t, 0, 20}, MaxSteps → 5000];
```

```
In[53]:= sol14 =
          NDSolve[Join[{eq12 /. ω → ω0 /. values4}, initial4],
           y[t], {t, 0, 20}, MaxSteps → 5000];
In[54]:= Plot[
          {y[t] /. sol13[[1]], y[t] /. sol14[[1]]}, {t, 0, 17},
          PlotStyle → {{Thickness[0.009], RGBColor[1, 0, 0]},
            {Thickness[0.009], Dashing[{0.03}],
             RGBColor[0, 0, 1]}}, PlotPoints → 50];
```

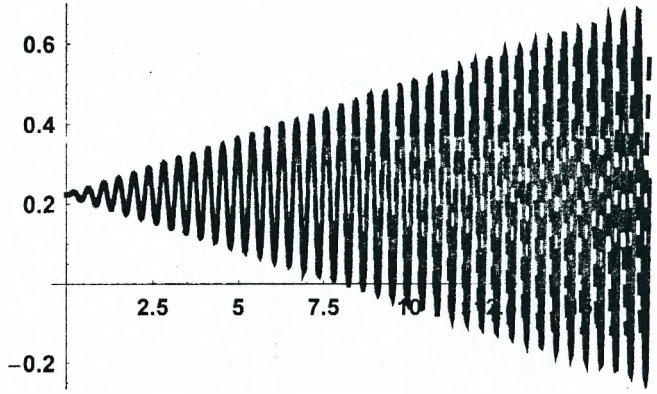


График показывает, что на начальном этапе движения, когда колебания малы, оба решения совпадают и демонстрируют линейное возрастание амплитуды колебаний. Однако если амплитуда колебаний линейной системы все время возрастает, то амплитуда нелинейных колебаний, достигнув некоторого максимума, начинает убывать. Это связано с тем, что собственная частота нелинейных колебаний зависит от их амплитуды. И когда амплитуда колебаний становится достаточно большой, условие резонанса для нелинейной системы перестает выполняться, и дальнейший рост амплитуды колебаний прекращается.

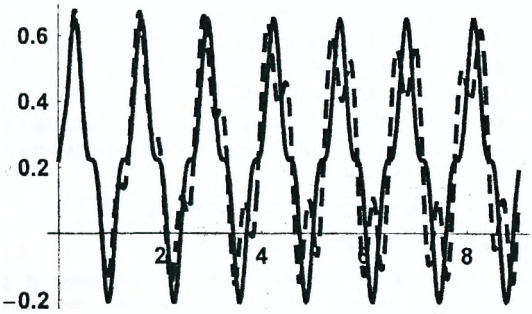
Наряду с изменением характера резонансных явлений при частоте $\omega = \omega_0$ нелинейность колебаний приводит также к появлению новых резонансов, в которых колебания с частотой, близкой к ω_0, возбуждаются синусоидальной силой с частотой $\omega = \frac{p}{q}\omega_0$, где p и q – целые числа. Однако при больших значениях p и q интенсивность резонансных явлений быстро убывает, так что реально могут наблюдаться лишь резонансы на частотах $\omega = \frac{p}{q}\omega_0$ с небольшими значениями p и q. В качестве примера рассмотрим случай $\omega = \frac{1}{3}\omega_0$. Чтобы нелинейные эффекты проявили себя более отчетливо, увеличим амплитуду внешней силы A_0, и решим уравнения *eq11* и *eq12* при нулевых начальных условиях.

```
In[55]:= values5 = {A0 → 60};
```

```
In[56]:= sol15 =
          NDSolve[Join[{eq11 /. ω → ω0 / 3 /. values5 }, initial4],
           y[t], {t, 0, 20}, MaxSteps → 5000];
In[57]:= sol16 =
          NDSolve[Join[{eq12 /. ω → ω0 / 3 /. values5}, initial4],
           y[t], {t, 0, 20}, MaxSteps → 7000];
```

Строим графики полученных решений.

```
In[58]:= Plot[
          {y[t] /. sol15[[1]], y[t] /. sol16[[1]]}, {t, 0, 9},
          PlotStyle → {{Thickness[0.01], RGBColor[1, 0, 0]},
            {Thickness[0.01], Dashing[{0.03}],
             RGBColor[0, 0, 1]}}, PlotPoints → 50];
```



Из графиков видно, что характер решения линейного уравнения *eq11* с течением времени не изменяется, и амплитуда линейных колебаний остается практически постоянной. У решения же нелинейного уравнения *eq12* наблюдается следующая особенность – на фоне колебаний частотой $\omega_0/3$, которая равняется частоте внешней синусоидальной силы, постепенно проявляются колебания с частотой ω_0. Таким образом, у нелинейной системы колебания с собственной частотой ω_0 могут возбуждаться внешней силой, частота которой существенно отличается от собственной частоты системы.

Задание 3

1. Предполагая, что трение в системе отсутствует, определите линейное и точное уравнения движения тела, когда на него действует дополнительная синусоидальная сила, направленная вдоль оси *Ox* или *Oy* (см. Пример 3).

2. Задайте численные значения параметров системы и начальные условия и найдите решения уравнений движения в линейном и нелинейном случае при частоте внешней силы, отличной от резонансной. Постройте графики полученных решений и сделайте вывод.

3. Найдите решения уравнений движения в резонансном случае, постройте их графики и сделайте вывод.

4*. Выполните пункты 2 и 3 при наличии в системе вязкого трения.

Пример 4

В предыдущем примере мы воздействовали на систему, непосредственно прикладывая синусоидальную силу к колеблющемуся телу, и убедились в том, что при определенных соотношениях между частотой силы и собственной частотой системы возникает резонанс. Однако воздействовать на систему можно и путем периодического изменения ее параметров. Например, если точка *B*, в которой закреплен конец второй пружины, совершает малые колебания вдоль оси *Ox*, то собственная частота системы ω_0, зависящая от величины *xB*, также будет испытывать колебания. При определенных частотах колебаний точки *B* амплитуда колебаний тела будет быстро возрастать со временем. Это явление называется *параметрическим резонансом*. По сравнению с обычным резонансом параметрический резонанс имеет две особенности. Во-первых, он не может происходить при нулевых начальных условиях, т.е. тело должно быть отклонено от равновесного положения хотя бы на очень малую величину. Во-вторых, амплитуда колебаний возрастает по экспоненциальному, а не линейному закону, т.е. существенно быстрее, чем при обычном резонансе. Наиболее ярко эти особенности параметрического резонанса проявляются в случае, когда частота колебаний параметров системы близка к удвоенной собственной частоте ω_0. Поэтому рассмотрим именно такой случай, когда частота колебаний точки *B* равна $2\omega_0$. Чтобы получить уравнение движения тела, определяем соответствующую подстановку, причем амплитуду колебаний точки *B* выбираем достаточно малой.

In[59]:=
```
values6 = {yA → 2, xB → 0.8 + 0.1 Sin[2 ω0 t],
           yB → 0.4, xC → -.4, yC → -1.3};
```

Выполняя подстановку и разлагая потенциальную энергию в ряд по *y* в окрестности положения равновесия с точностью до второго порядка, получаем:

In[60]:=
```
U3 = Normal[
     Series[U /. values1 /. values6, {y, yr, 2}] ] // Chop
```

Тогда линейное уравнение движения принимает вид:

In[61]:=
```
eq20 = m y''[t] == -(D[U3, y] /. y → y[t]) /. values1
```

Аналогично получаем точное уравнение движения тела при параметрическом воздействии на него.

In[62]:=
```
eq21 =
   m y''[t] == -(D[U, y] /. y → y[t]) /. values1 /. values6
```

Непосредственными расчетами легко убедиться в том, что при равновесных начальных условиях решения уравнений *eq20*, *eq21* тождественно равны нулю. Поэтому устанавливаем маленькое начальное смещение тела.

In[63]:=
```
initial5 = {y[0] == yr + 0.05, y'[0] == 0};
```

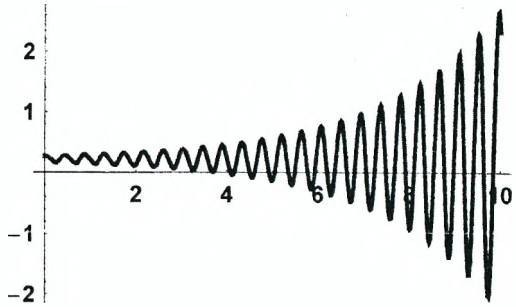
Находим численные решения уравнений *eq20* и *eq21*.

In[64]:=
```
sol20 = NDSolve[Join[{eq20}, initial5],
        y[t], {t, 0, 20}, MaxSteps → 6000];
```

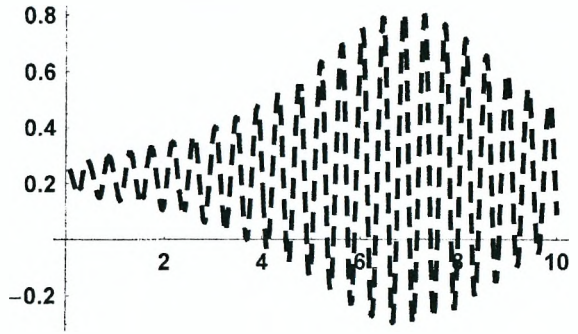
```
In[65]:= sol21 = NDSolve[Join[{eq21}, initial5],
           y[t], {t, 0, 20}, MaxSteps → 7000];
```

Строим графики полученных решений.

```
In[66]:= Plot[y[t] /. sol20[[1]], {t, 0, 10},
           PlotStyle → {{Thickness[0.01], RGBColor[1, 0, 0]}},
           PlotRange → All];
```




```
In[67]:= Plot[y[t] /. sol21[[1]],
           {t, 0, 10}, PlotStyle → {{Thickness[0.01],
              Dashing[{0.03}], RGBColor[0, 0, 1]}}];
```



Первый график показывает, что амплитуда колебаний решения линейного уравнения *eq20* неограниченно возрастает, причем скорость ее роста быстро увеличивается. В случае нелинейного уравнения *eq21* также наблюдается быстрый рост амплитуды колебаний тела. Однако, так как собственная частота ω_0 нелинейной системы зависит от амплитуды колебаний, как и при обычном резонансе, с ее ростом условие резонанса перестает выполняться и, достигнув максимума, амплитуда начинает уменьшаться. Затем процесс повторяется.

Задание 4

1. Предположим, что одна из точек A, B или C совершает малые колебания вдоль оси Ox или Oy. Направление колебаний этой точки должно быть перпендикулярным к направлению движения тела m, а частота равна удвоенной собственной частоте ω_0. Получите уравнения движения тела в линейном и нелинейном случаях при таком способе возбуждения колебаний (см. Пример 4).

2. Задайте подходящие начальные условия и решите оба уравнения движения. Постройте графики полученных решений и убедитесь в возникновении параметрического резонанса.

3. Найдите решения уравнений движения тела, изменив частоту параметрического воздействия на систему, и постройте их графики. Сделайте вывод.

Лабораторная работа № 7
Моделирование колебаний механических систем

Цель работы:

1) получить уравнения движения двух связанных грузов и найти их положения равновесия;

2) показать, что малые колебания грузов около положений равновесия являются гармоническими и найти собственные частоты колебаний системы;

3) убедиться в том, что произвольные движения грузов вблизи положений равновесия представляет собой суперпозицию собственных колебаний системы;

4) иссследовать затухающие и вынужденные колебания грузов.

Колебания механических систем происходят вблизи их устойчивых положений равновесия и описываются системами линейных однородных дифференциальных уравнений с постоянными коэффициентами вида:

$$m_i \frac{d^2 x_i}{d t^2} + \sum_{j=1}^{n} k_{ij} x_j = 0 \quad (i, j = 1, 2, \ldots, n), \tag{1}$$

где x_i – смещение i-ого тела из положения равновесия. Согласно общим правилам решения таких уравнений неизвестные функции $x_i(t)$ ищутся в виде:

$$x_i = A_i \exp(\lambda t), \tag{2}$$

где A_i и λ – некоторые постоянные, которые требуется определить. При подстановке решений (2) в уравнения движения (1) и сокращении множителя $e^{\lambda t}$, получаем систему линейных однородных алгебраических уравнений:

$$\lambda^2 m_i A_i + \sum_{j=1}^{n} k_{ij} A_j = 0 \quad (i, j = 1, 2, \ldots, n) \tag{3}$$

Для того, чтобы эта система имела отличные от нуля решения, должен обращаться в нуль определитель матрицы, составленной из коэффициентов A_j:

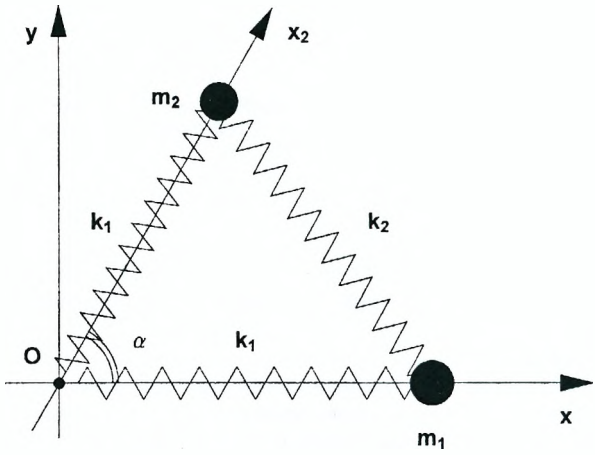
$$| \lambda^2 m_i \delta_{ij} + k_{ij} | = 0, \tag{4}$$

где $\delta_{ij} = 1$ при $i = j$ и $\delta_{ij} = 0$ при $i \neq j$. Это уравнение является уравнением степени n относительно λ^2 и называется *характеристическим уравнением*. В общем случае оно имеет n различных вещественных отрицательных корней λ_j^2 (при условии, что тела движутся в окрестностях их устойчивых положений равновесия). Для каждого корня получаем $\lambda_j = \pm i\omega_j$ ($j = 1, 2, \ldots, n$), где i – мнимая единица, а величины ω_j называются *собственными частотами системы*. После того как частоты найдены, подставляя каждое значение ω_j в уравнения (3), находим соответствующие значения коэффициент A_j. Тогда общее решение системы (1) получается в виде

линейной комбинации частных решений (2). Таким образом, изменение смещения каждого тела из положения равновесия $x_j(t)$ со временем представляет собой наложение n простых гармонических колебаний с частотами ω_j и произвольными амплитудами и начальными фазами, которые определяются условиями возбуждения колебаний.

Пример 1. Гармонические колебания связанных грузов

Два груза массами m_1 и m_2 могут скользить без трения соответственно вдоль оси Ox и прямой, составляющей с осью Ox угол α (на рисунке обозначена как ось Ox_2). Движение происходит в горизонтальной плоскости. Грузы соединены пружиной жесткостью k_2 и, кроме того, каждый из грузов соединен с другой пружиной жесткостью k_1. Другие концы пружин закреплены в точке O.



Пусть длины пружин жесткостью k_1 и k_2 в недеформированном состоянии равны соответственно L_{10}, L_{20}. Обозначим положения грузов на осях Ox и Ox_2 через x_1, x_2. Тогда потенциальную энергию системы можно записать в виде:

In[6]:=
```
Clear["Global`*"];
```

$$\text{Upot} = \frac{k_1}{2}\,(x1 - L10)^2 + \frac{k_1}{2}\,(x2 - L10)^2 + \frac{k_2}{2}\left(\sqrt{(x1 - x2\,\text{Cos}[\alpha])^2 + (x2\,\text{Sin}[\alpha])^2} - L20\right)^2$$

Out[7]=

$$\frac{1}{2}\,(-L10 + x1)^2\,k_1 + \frac{1}{2}\,(-L10 + x2)^2\,k_1 + \frac{1}{2}\left(-L20 + \sqrt{(x1 - x2\,\text{Cos}[\alpha])^2 + x2^2\,\text{Sin}[\alpha]^2}\right)^2 k_2$$

Силы, действующие на грузы вдоль осей Ox и Ox_2 связаны с потенциальной энергией следующими соотношениями:

```
In[8]:= F1x = -D[Upot, x1]
```

$$\text{Out[8]= } -(-L10+x1)\,k_1 - \left((x1-x2\,\text{Cos}[\alpha])\left(-L20+\sqrt{(x1-x2\,\text{Cos}[\alpha])^2+x2^2\,\text{Sin}[\alpha]^2}\right)k_2\right)\Big/\left(\sqrt{(x1-x2\,\text{Cos}[\alpha])^2+x2^2\,\text{Sin}[\alpha]^2}\right)$$

```
In[9]:= F2x = -D[Upot, x2]
```

$$\text{Out[9]= } -(-L10+x2)\,k_1 - \left(-2\,\text{Cos}[\alpha]\,(x1-x2\,\text{Cos}[\alpha])+2\,x2\,\text{Sin}[\alpha]^2\right)\left(-L20+\sqrt{(x1-x2\,\text{Cos}[\alpha])^2+x2^2\,\text{Sin}[\alpha]^2}\right)k_2\Big/\left(2\sqrt{(x1-x2\,\text{Cos}[\alpha])^2+x2^2\,\text{Sin}[\alpha]^2}\right)$$

В положении равновесия сила, действующая на каждый груз, должна равняться нулю. Это условие дает следующую систему уравнений для определения равновесных координат грузов:

```
In[10]:= eq = {F1x == 0, F2x == 0};
```

Поскольку эта система является существенно нелинейной относительно *x1*, *x2*, мы можем найти только ее численное решение. Для этого выберем некоторые реалистичные значения параметров системы и определим список соответствующих правил замены.

```
In[11]:= values1 = {k1 → 100, k2 → 200, L10 → 0.5,
           L20 → 0.8, α → π / 3, m1 → 1, m2 → 1.5};
```

Для решения системы используем функцию **NSolve**.

```
In[12]:= sol = NSolve[eq /. values1, {x1, x2}]
```

```
Out[12]= {{x1 → 0.75 - 0.259808 i, x2 → 0.75 + 0.259808 i},
          {x1 → 0.75 + 0.259808 i, x2 → 0.75 - 0.259808 i},
          {x1 → 0.65, x2 → 0.65}, {x1 → -0.15, x2 → -0.15}}
```

Очевидно, из четырех найденных решений только третье соответствует изображенной на рисунке конфигурации. Далее будем исследовать только малые колебания грузов около положений равновесия. Для этого в выражениях для сил, действующих на грузы, сделаем подстановку $x1 \to 0.65 + x1$, $x2 \to 0.65 + x2$ и разложим их в ряды Тейлора по $x1$ и $x2$ в окрестности нуля с точностью до первой степени $x1$ и $x2$.

```
In[13]:= F1xlin = (Series[F1x /. {x1 → (x1 /. sol[[3]]) + δ x1,
                                 x2 →
                    (x2 /. sol[[3]]) + δ x2} /. values1,
                     {δ, 0, 1}] // Normal // Chop // Simplify) /.
          δ → 1
Out[13]= -115.385 x1 - 84.6154 x2

In[14]:= F2xlin = (Series[F2x /. {x1 → (x1 /. sol[[3]]) + δ x1,
                                  x2 → (x2 /. sol[[3]]) + δ x2} /.
                values1,
                  {δ, 0, 1}] // Normal // Chop // Simplify) /.
          δ → 1
Out[14]= -84.6154 x1 - 115.385 x2
```

Заметим, что в полученных выражениях переменные *x1* и *x2* обозначают смещения грузов из положений равновесия. Теперь можно определить уравнения движения грузов вблизи этих положений.

```
In[15]:= eq1 = {m1 x1''[t] == (F1xlin /. {x1 → x1[t], x2 → x2[t]}),
                m2 x2''[t] ==
              (F2xlin /. {x1 → x1[t], x2 → x2[t]}) } /. values1
Out[15]= {x1''[t] == -115.385 x1[t] - 84.6154 x2[t],
          1.5 x2''[t] == -84.6154 x1[t] - 115.385 x2[t]}
```

Решение этой системы может быть легко найдено с помощью функции **DSolve**. Однако, чтобы выделить собственные частоты колебаний, мы подставим в нее решение вида: $x1 = u_1 e^{\lambda t}$, $x2 = u_2 e^{\lambda t}$, где u_1, u_2 и λ – постоянные. После сокращения множителей $e^{\lambda t}$ в левой и правой частях каждого уравнения, получаем систему двух линейных алгебраических уравнений относительно u_1, u_2.

```
In[16]:= eq2 = eq1 /. {x1 → (u1 Exp[λ #] &), x2 → (u2 Exp[λ #] &)} /.
          Exp[_] → 1
Out[16]= {λ^2 u1 == -115.385 u1 - 84.6154 u2,
          1.5 λ^2 u2 == -84.6154 u1 - 115.385 u2}
```

Выделяя в *eq2* коэффициенты при u_1, u_2, получаем матрицу:

```
In[17]:= A0 = Table[Coefficient[eq2[[i, 1]] - eq2[[i, 2]], uj],
            {i, 2}, {j, 2}]
Out[17]= {{115.385 + λ^2, 84.6154}, {84.6154, 115.385 + 1.5 λ^2}}
```

Приравнивая ее определитель к нулю, получаем уравнение четвертого порядка относительно λ:

```
In[18]:= eq3 = Det[A0] == 0
Out[18]= 6153.85 + 288.462 λ^2 + 1.5 λ^4 == 0
```

Его корни легко вычисляются с помощью функции **Solve**.

```
In[19]:= sol3 = Solve[eq3, λ] // Chop
Out[19]= {{λ → -4.9436 i}, {λ → 4.9436 i},
          {λ → -12.9564 i}, {λ → 12.9564 i}}
```

Как видим, имеются четыре чисто мнимых попарно комплексно сопряженных корня. Согласно формуле Эйлера

$$\exp(i\omega t) = \cos(\omega t) + i \sin(\omega t).$$

Следовательно, при значениях λ, найденных в *sol3*, решения вида $x1 = u_1 e^{\lambda t}$, $x2 = u_2 e^{\lambda t}$ уравнения *eq1* определяют гармонические колебания грузов около положений равновесия, причем система имеет две собственные частоты. Обозначим их через ω_1 и ω_2.

```
In[20]:= ω1 = i λ /. sol3[[1]]
Out[20]= 4.9436

In[21]:= ω2 = i λ /. sol3[[3]]
Out[21]= 12.9564
```

Чтобы найти решения, соответствующие каждой собственной частоте, подставляем в одно из уравнений системы *eq2*, например, в первое соответствующие значения λ и, полагая $u_1 = 1$, находим u_2. Так как λ входит в уравнения *eq2* только в виде λ^2, то имеется только два возможных значения u_2.

```
In[22]:= sol4 = Solve[eq2[[1]] /. u1 → 1 /. sol3[[1]], u2]
Out[22]= {{u2 → -1.07481}}

In[23]:= sol5 = Solve[eq2[[1]] /. u1 → 1 /. sol3[[3]], u2]
Out[23]= {{u2 → 0.620265}}
```

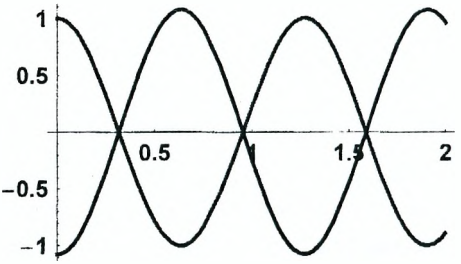
Общее решение исходной системы дифференциальных уравнений *eq1* запишем виде:

```
In[24]:= X1[t_] = C1 Re[Exp[i ω1 t]] + C2 Im[Exp[i ω1 t]] +
                  C3 Re[Exp[i ω2 t]] + C4 Im[Exp[i ω2 t]] //
          Simplify[#, t ∈ Reals] &
Out[24]= Cos[4.9436 t] C1 + Sin[4.9436 t] C2 +
         Cos[12.9564 t] C3 + Sin[12.9564 t] C4

In[25]:= X2[t_] =
         (C1 Re[u2 Exp[i ω1 t]] + C2 Im[u2 Exp[i ω1 t]] /. sol4[[1]]) +
           (C3 Re[u2 Exp[i ω2 t]] + C4 Im[u2 Exp[i ω2 t]] /.
             sol5[[1]]) // Simplify[#, t ∈ Reals] &
Out[25]= -1.07481 Cos[4.9436 t] C1 - 1.07481 Sin[4.9436 t] C2 +
         0.620265 Cos[12.9564 t] C3 + 0.620265 Sin[12.9564 t] C4
```

Оно содержит четыре произвольных постоянных C_1, C_2, C_3, C_4, которые находятся из начальных условий. Полагая одну из этих постоянных равной 1, а остальные – нулю, будем получать функции *x1(t)*, *x2(t)*, соответствующие собственным колебаниям системы. При этом случаи $C_1 = 1$, $C_2 = C_3 = C_4 = 0$ и $C_1 = C_3 = C_4 = 0$, $C_2 = 1$ соответствуют колебаниям с частотой ω_1 и отличаются только значением начальной фазы колебаний. Аналогичное утверждение справедливо для постоянных C_3 и C_4.

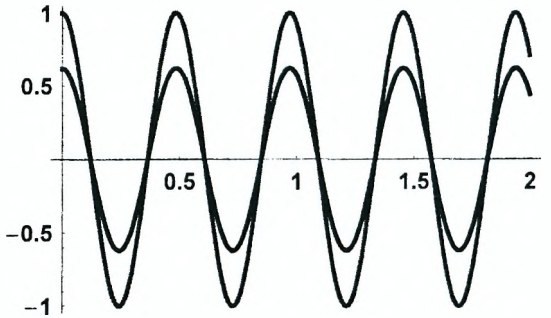
Полагая в общем решении $C_1 = 1$, $C_2 = C_3 = C_4 = 0$, получаем для колебаний с собственной частотой ω_1 следующий график:

In[26]:=
```
Plot[
  {X1[t] /. {C1 → 1, Cj_ → 0}, X2[t] /. {C1 → 1, Cj_ → 0} },
    {t, 0, 2}, PlotStyle →
  {{Thickness[0.01], RGBColor[1, 0, 0]},
   {Thickness[0.01], RGBColor[0, 0, 1]}}];
```



Для колебаний с собственной частотой ω_2 соответствующий график имеет вид (случай $C_3 = 1$, $C_1 = C_2 = C_4 = 0$):

In[27]:=
```
Plot[
  {X1[t] /. {C3 → 1, Cj_ → 0}, X2[t] /. {C3 → 1, Cj_ → 0} },
    {t, 0, 2}, PlotStyle →
  {{Thickness[0.01], RGBColor[1, 0, 0]},
   {Thickness[0.01], RGBColor[0, 0, 1]}}];
```



Теперь предположим, что в начальный момент времени грузы находятся в положениях равновесия, и первый груз получает толчок, т.е. некоторую начальную скорость. Начальные условия для этой ситуации зададим в виде:

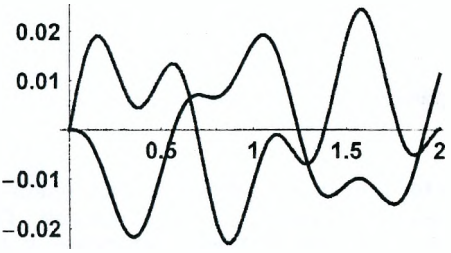
```
In[28]:= eq7 = {X1[0] == 0, X1'[0] == 0.2, X2[0] == 0, X2'[0] == 0}

Out[28]= {C1 + C3 == 0, 4.9436 C2 + 12.9564 C4 == 0.2,
          -1.07481 C1 + 0.620265 C3 == 0, -5.31343 C2 + 8.0364 C4 == 0}
```

Решая систему *eq7*, находим постоянные C_j ($j = 1, 2, 3, 4$).

```
In[29]:= sol7 = Solve[eq7, {C1, C2, C3, C4}]

Out[29]= {{C1 → 0., C2 → 0.0148039, C3 → 0., C4 → 0.00978787}}
```

Теперь можно построить графики функций *x1*(*t*), *x2*(*t*), соответствующие заданным начальным условиям.

```
In[30]:= Plot[
          {X1[t] /. sol7[[1]], X2[t] /. sol7[[1]]}, {t, 0, 2},
          PlotStyle → {{Thickness[0.01], RGBColor[1, 0, 0]},
            {Thickness[0.01], RGBColor[0, 0, 1]}}];
```



Чтобы сравнить решение, полученное в линейном приближении, с точным решением, определяем точные уравнения движения, которые являются существенно нелинейными.

```
In[32]:= eq8 =
          {m1 x1''[t] == (F1x /. {x1 → (x1 /. sol[[3]]) + x1[t],
                                  x2 → (x2 /. sol[[3]]) + x2[t]}),
            m2 x2''[t] ==
           (F2x /. {x1 → (x1 /. sol[[3]]) + x1[t],
                    x2 →
             (x2 /. sol[[3]]) + x2[t]}) } /. values1;
```

Соответствующие начальные условия для функций $x_1(t)$ и $x_2(t)$ имеют вид:

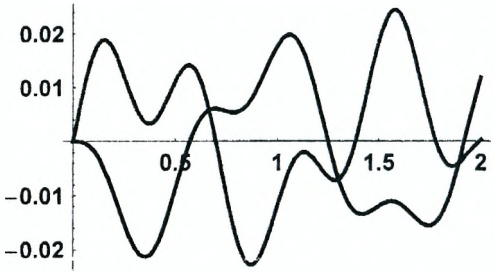
```
In[33]:= initial = {x1[0] == 0, x1'[0] == 0.2, x2[0] == 0, x2'[0] == 0};
```

Для решения уравнений движения используем функцию **NDSolve**.

```
In[34]:= sol8 =
          NDSolve[Join[eq8, initial], {x1[t], x2[t]}, {t, 0, 2}]
```

Строим графики найденных функций $x_1(t)$ и $x_2(t)$.

In[35]:=
```
Plot[
  {x1[t] /. sol8[[1]], x2[t] /. sol8[[1]]}, {t, 0, 2},
  PlotStyle → {{Thickness[0.01], RGBColor[1, 0, 0]},
    {Thickness[0.01], RGBColor[0, 0, 1]}}];
```

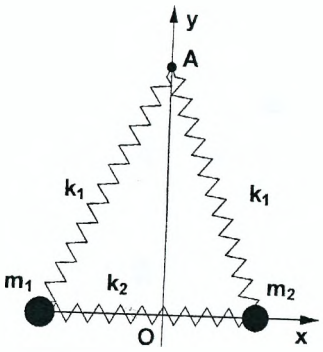


Как видим, графики решений точных и приближенных уравнений движения очень похожи. Построив их на одной координатной плоскости, можно убедиться в том, что они практически совпадают. Поэтому при малых смещениях грузов из положений равновесия для описания их движения можно использовать уравнения, полученные в линейном приближении.

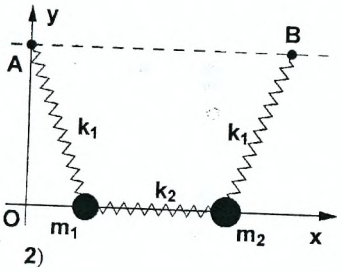
Задание 1

Два груза массами m_1 и m_2 могут скользить без трения вдоль некоторых прямых, расположенных в горизонтальной плоскости. Грузы соединены пружиной и, кроме того, каждый из грузов соединен с другой пружиной, второй конец которой закреплен. Выполните следующие задания:

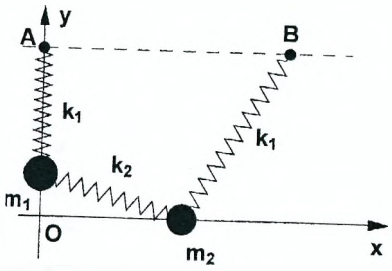
1. Выберите один из рисунков и установите возможные направления движения грузов в рассматриваемой системе. Определите потенциальную энергию системы и найдите силы, действующие на каждый груз в направлении движения (см. Пример 1).
2. Подберите численные значения параметров системы так, чтобы у грузов существовали устойчивые положения равновесия. Определите координаты грузов в этих положениях.
3. Разложите силы, действующие на грузы, в ряды около положений равновесия с точностью до первой степени по отклонениям грузов от этих положений. Получите систему дифференциальных уравнений, определяющую гармонические колебания грузов.
4. Найдите собственные частоты колебаний системы и запишите общее решение линейных уравнений движения. Постройте графики зависимостей найденных функций от времени для колебаний с собственными частотами, а также в общем случае, выбирая способ возбуждения колебаний самостоятельно.
5. Найдите решение точных уравнений движения при выбранных начальных условиях и сравните его с решением линейных уравнений движения. Сделайте вывод.


y
A
k1
k1
m1
k2
m2
O
x

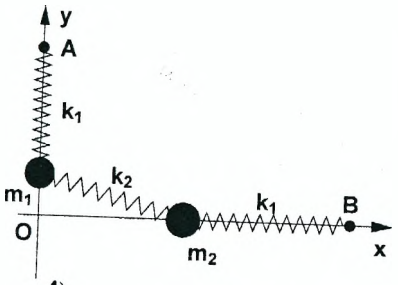
1)


y
B
A
k1
k1
k2
O
m1
m2
x

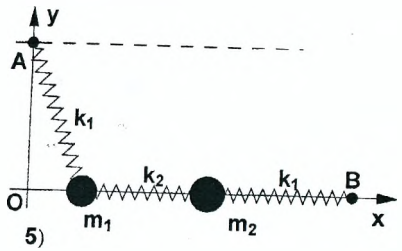
2)


y
A
B
k1
k1
k2
m1
O
m2
x

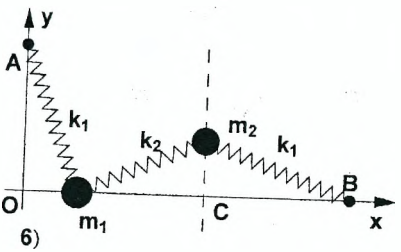
3)


y
A
k1
k2
m1
k1
B
O
m2
x

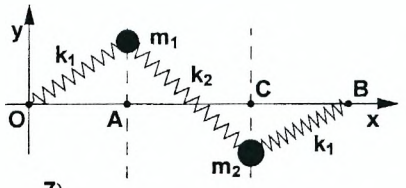
4)


y
A
k1
k2
k1
B
O
m1
m2
x

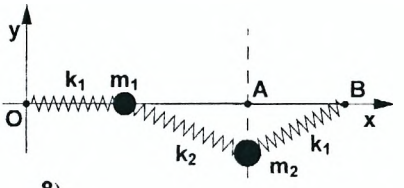
5)


y
A
k1
m2
k2
k1
B
O
m1
C
x

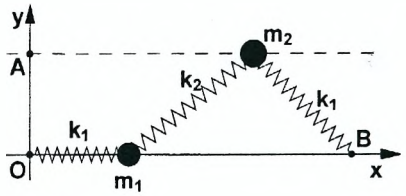
6)



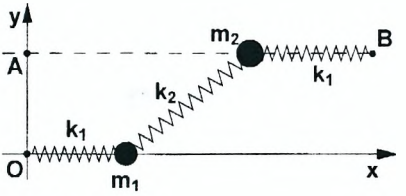
7)



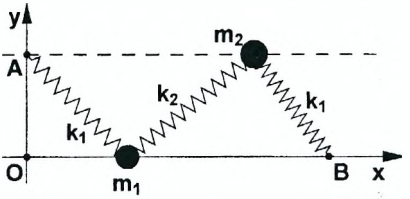
8)



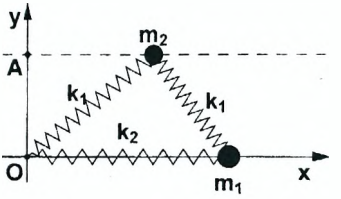
9)



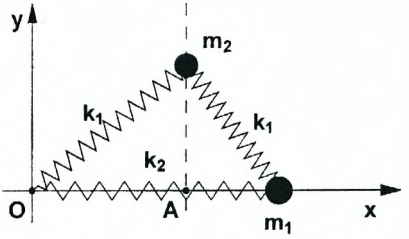
10)



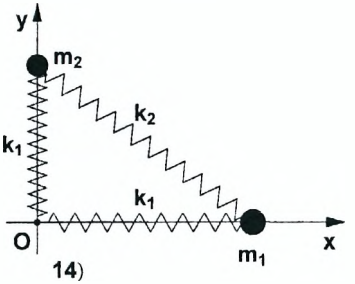
11)



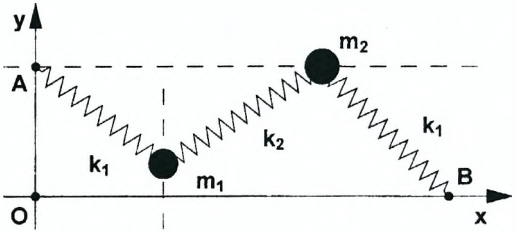
12)



13)



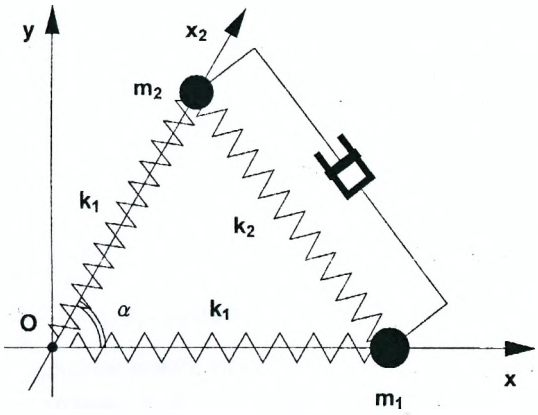
14)



15)

Пример 2. Затухающие колебания грузов

Предположим, что в системе из Примера 1 между грузами m_1 и m_2 имеется демпфер. Он обеспечивает появление силы вязкого трения, которая препятствует изменению расстояния между грузами и пропорциональна скорости изменения этого расстояния.



Проекцию на ось *Ox* силы трения, действующей на первый груз, можно записать в виде:

```
In[36]:= Ftr1 = -b (x1'[t] - x2'[t] Cos[α])

Out[36]= -b (x1'[t] - Cos[α] x2'[t])
```

Здесь *b* – положительный коэффициент. Легко видеть, что в скобках записана разность проекций скоростей грузов на ось *Ox*. Аналогично, проекция на ось Ox_2 силы трения, действующей на второй груз, имеет вид:

```
In[37]:= Ftr2 = -b (x2'[t] - x1'[t] Cos[α])

Out[37]= -b (-Cos[α] x1'[t] + x2'[t])
```

Задавая численное значение коэффициента *b* и добавляя силы трения в правых частях уравнений *eq1*, получаем систему уравнений движения грузов в линейном приближении в виде:

```
In[38]:= values3 = {b → 0.5};
         eq9 = {m1 x1''[t] ==
               (F1xlin /. {x1 → x1[t], x2 → x2[t]}) + Ftr1,
             m2 x2''[t] == (F2xlin /. {x1 → x1[t], x2 → x2[t]}) +
               Ftr2} /. values1 /. values3
```

Out[39]=
$$\left\{x1''[t] == -115.385\,x1[t] - 84.6154\,x2[t] - 0.5\left(x1'[t] - \frac{x2'[t]}{2}\right),\right.$$
$$\left. 1.5\,x2''[t] == -84.6154\,x1[t] - 115.385\,x2[t] - 0.5\left(-\frac{1}{2}\,x1'[t] + x2'[t]\right)\right\}$$

Согласно общему правилу, решение системы линейных дифференциальных уравнений с постоянными коэффициентами *eq9* ищем в виде: $x1 = u_1 e^{\lambda t}$, $x2 = u_2 e^{\lambda t}$, где u_1, u_2 и λ – постоянные. Выполняя соответствующую подстановку и сокращая общий множитель $e^{\lambda t}$, получаем систему двух алгебраических уравнений *eq10*.

In[40]:=
```
eq10 = eq9 /. {x1 → (u1 Exp[λ #] &), x2 → (u2 Exp[λ #] &)} /.
  Exp[_] → 1
```

Out[40]=
$$\left\{\lambda^2 u_1 == -115.385\,u_1 - 84.6154\,u_2 - 0.5\left(\lambda u_1 - \frac{\lambda u_2}{2}\right),\right.$$
$$\left. 1.5\,\lambda^2 u_2 == -84.6154\,u_1 - 115.385\,u_2 - 0.5\left(-\frac{\lambda u_1}{2} + \lambda u_2\right)\right\}$$

Выделяя в *eq10* коэффициенты при u_1, u_2, получаем матрицу:

In[41]:=
```
A1 = Table[Coefficient[eq10[[i, 1]] - eq10[[i, 2]], uj],
  {i, 2}, {j, 2}]
```

Out[41]=
$$\{\{115.385 + 0.5\,\lambda + \lambda^2,\ 84.6154 - 0.25\,\lambda\},$$
$$\{84.6154 - 0.25\,\lambda,\ 115.385 + 0.5\,\lambda + 1.5\,\lambda^2\}\}$$

Приравнивая ее определитель к нулю, получаем уравнение четвертого порядка относительно λ:

In[42]:=
```
eq11 = Det[A1] == 0
```

Out[42]=
$$6153.85 + 157.692\,\lambda + 288.649\,\lambda^2 + 1.25\,\lambda^3 + 1.5\,\lambda^4 == 0$$

Его корни легко вычисляются с помощью функции **Solve**.

In[43]:=
```
sol11 = Solve[eq11, λ]
```

Out[43]=
$$\{\{\lambda \to -0.295525 - 4.93507\,i\},\ \{\lambda \to -0.295525 + 4.93507\,i\},$$
$$\{\lambda \to -0.121141 - 12.955\,i\},\ \{\lambda \to -0.121141 + 12.955\,i\}\}$$

Как видим, имеются две пары комплексно сопряженных корней. Так как их действительные части отрицательны, решение системы *eq9* будет экспоненциально убывать с течением времени. Модули мнимых частей решений *sol11* равняются частотам колебаний грузов. В отсутствие трения мы получили для собственных частот следующие значения:

In[44]:=
```
{ω1, ω2}
```

Out[44]=
```
{4.9436, 12.9564}
```

Легко видеть, что учет трения приводит к небольшому уменьшению частот, т.е. к возрастанию периодов колебаний. Чтобы найти решения, соответствующие найденным значениям λ, подставляем каждое из них в первое уравнение *eq10* и, полагая $u_1 = 1$, находим u_2.

```
In[45]:= sol12 =
           Table[Solve[eq10[[1]] /. u1 → 1 /. sol11[[j]], u2],
             {j, 4}] // Flatten
Out[45]= {u2 → -1.074 + 0.0103405 i, u2 → -1.074 - 0.0103405 i,
          u2 → 0.62076 + 0.0156919 i, u2 → 0.62076 - 0.0156919 i}
```

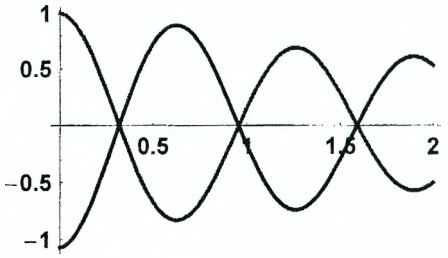
Как и ожидалось, комплексно сопряженным значениям λ соответствуют сопряженные значения u_2. Таким образом, имеются две пары комплексно сопряженных решений системы *eq9* вида: $x1 = u_1 e^{\lambda t}$, $x2 = u_2 e^{\lambda t}$, где $u_1 = 1$, а u_2 и λ принимают значения, найденные в *sol12* и *sol11* соответственно.

Общее решение исходной системы дифференциальных уравнений *eq9* можно представить в виде:

```
In[46]:= X1[t_] = C1 Re[Exp[λ t] /. sol11[[1]] ] +
                  C2 Im[Exp[λ t] /. sol11[[1]] ] +
                  C3 Re[Exp[λ t] /. sol11[[3]] ] +
                  C4 Im[Exp[λ t] /. sol11[[3]] ] // Simplify[#, t ∈ Reals] &
Out[46]= e^(-0.295525 t) Cos[4.93507 t] C1 - e^(-0.295525 t) Sin[4.93507 t] C2 +
         e^(-0.121141 t) (Cos[12.955 t] C3 - Sin[12.955 t] C4)

In[47]:= X2[t_] = C1 Re[u2 Exp[λ t] /. sol11[[1]] /. sol12[[1]] ] +
                  C2 Im[u2 Exp[λ t] /. sol11[[1]] /. sol12[[1]] ] +
                  C3 Re[u2 Exp[λ t] /. sol11[[3]] /. sol12[[3]] ] +
                  C4 Im[u2 Exp[λ t] /. sol11[[3]] /. sol12[[3]] ] //
                ComplexExpand // FullSimplify[#, t ∈ Reals] &
Out[47]= e^(-0.416667 t) (e^(0.121141 t)
           ((-1.074 Cos[4.93507 t] + 0.0103405 Sin[4.93507 t])
              C1 + (0.0103405 Cos[4.93507 t] +
                1.074 Sin[4.93507 t]) C2) + e^(0.295525 t)
           ((0.62076 Cos[12.955 t] + 0.0156919 Sin[12.955 t]) C3 +
             (0.0156919 Cos[12.955 t] - 0.62076 Sin[12.955 t])
              C4))
```

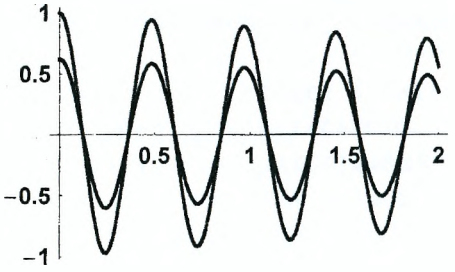
Как обычно, произвольные постоянные C_1, C_2, C_3, C_4 определяются из начальных условий. Полагая одну из этих постоянных равной 1, а остальные – нулю, будем получать функции $x1(t)$, $x2(t)$, соответствующие простейшим затухающим колебаниям системы. Как и в отсутствие трения, им соответствуют два типа движения грузов. В первом случае ($C_1 = 1$, $C_2 = C_3 = C_4 = 0$ или $C_2 = 1$, $C_1 = C_3 = C_4 = 0$) грузы движутся в противофазе. Соответствующие графики зависимостей $x1(t)$, $x2(t)$ имеют вид:

In[48]:=
```
Plot[
  {X1[t] /. {C1 → 1, Cj_ → 0}, X2[t] /. {C1 → 1, Cj_ → 0} },
                {t, 0, 2}, PlotStyle →
          {{Thickness[0.01], RGBColor[1, 0, 0]},
           {Thickness[0.01], RGBColor[0, 0, 1]}}];
```



Второй тип движения наблюдается при $C_3 = 1$, $C_1 = C_2 = C_4 = 0$ или $C_4 = 1$, $C_1 = C_2 = C_3 = 0$ и соответствует синфазному движению грузов.

In[49]:=
```
Plot[
  {X1[t] /. {C3 → 1, Cj_ → 0}, X2[t] /. {C3 → 1, Cj_ → 0} },
                {t, 0, 2}, PlotStyle →
          {{Thickness[0.01], RGBColor[1, 0, 0]},
           {Thickness[0.01], RGBColor[0, 0, 1]}}];
```



В общем случае наблюдается наложение двух указанных видов затухающих колебаний, причем их амплитуды и начальные фазы определяются способом возбуждения колебаний, т.е. начальными условиями.

Задание 2

1. Используя результаты из Задания 1, получите уравнения затухающих колебаний грузов m_1 и m_2, если между ними имеется демпфер. Предполагается, что сила вязкого трения действует вдоль прямой, проходящей через грузы, и пропорциональна скорости изменения расстояния между ними (см. Пример 2).

2. Найдите частоты колебаний системы и сравните их с собственными

частотами в отсутствие трения. Запишите общее решение линейных уравнений движения. Постройте графики зависимостей найденных функций от времени для простейших колебаний системы.

Пример 3. Вынужденные колебания грузов

Предположим, что на один из грузов системы из Примера 2, например, первый действует внешняя синусоидальная сила $F = \sin(\Omega t)$. В этом случае при исследовании движения системы в правую часть первого уравнения *eq9* следует добавить соответствующее слагаемое. В результате получится неоднородная система двух линейных дифференциальных уравнений, общее решение которой можно найти с помощью функции **DSolve**. Однако оно будет выглядеть очень громоздким и проанализировать его зависимость от частоты внешней силы Ω будет трудно. Чтобы убедиться в том, что при значениях Ω, близких к собственным частотам системы ω_1 и ω_2, происходит резонанс, воспользуемся следующим приемом.

Согласно теории дифференциальных уравнений общее решение неоднородной системы представляет собой сумму общего решения однородной системы, получаемой при $F = 0$, и частного решения неоднородной системы. В предыдущем примере мы видели, что общее решение однородной системы определяет затухающие колебания грузов. С течением времени такие колебания прекратятся и останутся только вынужденные колебания системы под действием внешней силы, которые определяются частным решением неоднородной системы. Напомним, что подобная ситуация имела место при рассмотрении электрических цепей переменного тока (см. Лабораторную работу № 3). Тогда для получения амплитуд токов в различных участках цепи мы использовали метод комплексных амплитуд. Этот же метод мы используем и сейчас.

Добавим в правой части первого уравнения системы *eq9* слагаемое $Exp(i\,\Omega\,t)$. Согласно формуле Эйлера, мнимая часть этого выражения совпадает с выражением для внешней силы *F*. В результате получим следующую систему:

In[50]:=
```
eq14 =
 {m1 x1''[t] == (F1xlin /. {x1 → x1[t], x2 → x2[t]}) +
                          Ftr1 + Exp[i Ω t],
  m2 x2''[t] == (F2xlin /. {x1 → x1[t],
                      x2 → x2[t]}) + Ftr2} /. values1 /. values3
```

Out[50]=

$$\left\{\mathbf{x1''[t]} == e^{i\,t\,\Omega} - 115.385\,\mathbf{x1[t]} - 84.6154\,\mathbf{x2[t]} - 0.5\left(\mathbf{x1'[t]} - \frac{\mathbf{x2'[t]}}{2}\right),\right.$$

$$\left.1.5\,\mathbf{x2''[t]} == -84.6154\,\mathbf{x1[t]} - 115.385\,\mathbf{x2[t]} - 0.5\left(-\frac{1}{2}\,\mathbf{x1'[t]} + \mathbf{x2'[t]}\right)\right\}$$

Решение *x1(t)*, *x2(t)*, описывающее вынужденные колебания грузов, будем искать в виде: $x1 = A_1 e^{i\Omega t}$, $x2 = A_2 e^{i\Omega t}$. Выполняя соответствующую подстановку и сокращая множитель $e^{i\Omega t}$, получаем систему двух алгебраических уравнений *eq15*.

```
In[51]:= eq15 =
           eq14 /. {x1 → (A1 Exp[i Ω #] &), x2 → (A2 Exp[i Ω #] &)} /.
            Exp[_] → 1
```

$$\text{Out[51]= } \left\{-\Omega^2 A_1 == 1 - 115.385 A_1 - 84.6154 A_2 - 0.5 \left(i \Omega A_1 - \frac{1}{2} i \Omega A_2\right),\right.$$

$$\left. -1.5 \Omega^2 A_2 == -84.6154 A_1 - 115.385 A_2 - 0.5 \left(-\frac{1}{2} i \Omega A_1 + i \Omega A_2\right)\right\}$$

Решая эту систему с помощью функции **Solve**, находим комплексные амплитуды колебаний грузов.

```
In[52]:= sol15 = Solve[eq15, {A1, A2}] // Simplify // Chop
```

$$\text{Out[52]= } \left\{\left\{A_1 \to \frac{26035.5 i - 35.8974 \Omega - 338.128 i \Omega^2 - 1. \Omega^3}{1.38856 \times 10^6 i - 31479.3 \Omega - 65025.9 i \Omega^2 + 89.6186 \Omega^3 + 337.628 i \Omega^4 + \Omega^5},\right.\right.$$

$$\left.\left. A_2 \to -\frac{56.4103 - 0.166667 i \Omega}{4102.56 + 105.128 i \Omega - 192.433 \Omega^2 - 0.833333 i \Omega^3 + \Omega^4}\right\}\right\}$$

Искомые зависимости *x1(t)*, *x2(t)*, описывающее вынужденные колебания грузов, определяются как мнимые части выражений $A_1 e^{i\Omega t}$ и $A_2 e^{i\Omega t}$ соответственно. Выполняя вычисления, получаем:

```
In[53]:= X1[t_] = Im[A1 Exp[i Ω t] /. sol15[[1]] // ExpToTrig ] //
           ComplexExpand // FullSimplify[#, {{t, Ω} ∈ Reals}] &
```

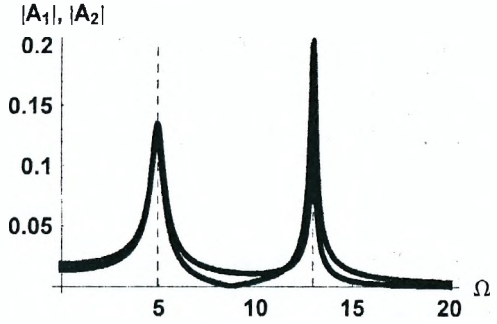
$$\text{Out[53]= } \frac{\Omega \left(-7.69733 \times 10^8 + 1.20316 \times 10^7 \Omega^2 - 57173. \Omega^4 - 0.5 \Omega^6\right) \text{Cos}[t\Omega] + \left(3.61519 \times 10^{10} - 2.16136 \times 10^9 \Omega^2 + 3.08057 \times 10^7 \Omega^4 - 114287. \Omega^6 - 1. \Omega^8\right) \text{Sin}[t\Omega]}{\left(1.9281 \times 10^{12} - 1.79594 \times 10^{11} \Omega^2 + 5.16036 \times 10^9 \Omega^4 - 4.39641 \times 10^7 \Omega^6 + 114172. \Omega^8 + \Omega^{10}\right)}$$

```
In[54]:= X2[t_] = Im[A2 Exp[i Ω t] /. sol15[[1]] // ExpToTrig ] //
           ComplexExpand // FullSimplify[#, {{t, Ω} ∈ Reals}] &
```

$$\text{Out[54]= } \frac{\Omega \left(6614.07 - 79.0807 \Omega^2 + 0.166667 \Omega^4\right) \text{Cos}[t\Omega] + \left(-231427. + 10872.7 \Omega^2 - 56.5491 \Omega^4\right) \text{Sin}[t\Omega]}{\left(1.6831 \times 10^7 - 1.56788 \times 10^6 \Omega^2 + 45060.3 \Omega^4 - 384.171 \Omega^6 + \Omega^8\right)}$$

Как видим, обе зависимости определяют гармонические колебания грузов с частотой внешней силы Ω. Амплитуды этих колебаний зависят от Ω и могут быть найдены как модули комплексных амплитуд A_1 и A_2. Их графики показаны на следующем рисунке. Штриховые линии на рисунке соответствуют значениям $\Omega = \omega_1$ и $\Omega = \omega_2$.

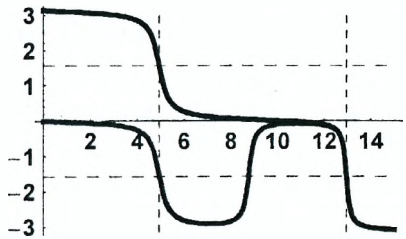
```
In[55]:= Plot[{Abs[A1 /. sol15[[1]]], Abs[A2 /. sol15[[1]]]},
          {Ω, 0, 20},
          PlotStyle → {{Thickness[0.015], RGBColor[1, 0, 0]},
            {Thickness[0.015], RGBColor[0, 0, 1]}}, Epilog →
           {{Dashing[{0.02}], Line[{{ω1, 0}, {ω1, 0.2}}]},
            {Dashing[{0.02}], Line[{{ω2, 0}, {ω2, 0.2}}]}},
          AxesLabel → {"Ω", "|A1|, |A2|"}];
```



Графики показывают, что при частотах внешней силы Ω, близких к собственным частотам системы, наблюдается резкое возрастание амплитуд колебаний грузов, что и соответствует явлению резонанса.

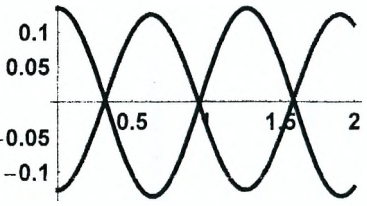
Следующие графики показывают зависимости начальных фаз вынужденных колебаний грузов от частоты внешней силы. Напомним, что они определяются как аргументы комплексных амплитуд A_1 и A_2.

```
In[56]:= Plot[{Arg[A1 /. sol15[[1]]],
           Arg[A2 /. sol15[[1]]], π / 2, -π / 2}, {Ω, 0, 15},
          PlotStyle → {{Thickness[0.015], RGBColor[1, 0, 0]},
            {Thickness[0.015], RGBColor[0, 0, 1]},
            {Dashing[{0.03}]}, {Dashing[{0.03}]}},
          Epilog → {{Dashing[{0.02}],
             Line[{{ω1, -π}, {ω1, π}}]},
            {Dashing[{0.02}], Line[{{ω2, -π}, {ω2, π}}]}}];
```



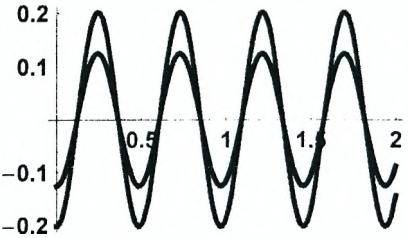
Как видим, при первой резонансной частоте $\Omega \approx \omega_1$ сдвиг фаз между колебаниями грузов и внешней силы составляет $\pm\pi/2$ (начальная фаза внешней силы равна нулю). При этом грузы совершают колебания в противофазе, что видно из следующего рисунка.

In[57]:=
```
Plot[{X1[t] /. {Ω → ω1}, X2[t] /. {Ω → ω1} },
                    {t, 0, 2}, PlotStyle →
          {{Thickness[0.015], RGBColor[1, 0, 0]},
           {Thickness[0.015], RGBColor[0, 0, 1]}}];
```



Напомним, что именно такой тип движения соответствует свободным колебаниям грузов с собственной частотой ω_1. При $\Omega \approx \omega_2$ грузы колеблются синфазно, отставая по фазе от внешней силы на $\pi/2$. Графики соответствующих зависимостей *x1(t)*, *x2(t)* показаны ниже.

In[58]:=
```
Plot[{X1[t] /. {Ω → ω2}, X2[t] /. {Ω → ω2} },
                    {t, 0, 2}, PlotStyle →
          {{Thickness[0.015], RGBColor[1, 0, 0]},
           {Thickness[0.015], RGBColor[0, 0, 1]}}];
```



Приведенные графики показывают, что в условиях обоих резонансов зависимость *x1(t)* имеет вид:

$$\mathbf{x1}\,(t) = \left| A_1 \right| \sin\left(\Omega\, t - \frac{\pi}{2}\right) = -\,|\,A_1\,|\cos\,(\Omega\, t)\,.$$

Очевидно, производная этой функции по времени, т.е. скорость первого груза, изменяется во времени по закону $\sin(\Omega t)$. Таким образом, фазы колебаний скорости первого груза и действующей на него силы совпадают. При этих условиях внешняя сила совершает максимальную работу и передает системе максимальную энергию, что и приводит к резонансу.

Задание 3

1. Используя результаты из Задания 2, получите уравнения вынужденных колебаний грузов m_1 и m_2, если на один из них действует внешняя синусоидальная сила $F = \sin(\Omega t)$. Найдите комплексные амплитуды колебаний и запишите зависимости *x1(t)*, *x2(t)*, описывающие вынужденные колебания грузов (см. Пример 3).

2. Постройте графики зависимостей амплитуд и начальных фаз вынужденных колебаний грузов от частоты внешней силы. Убедитесь в наличии резонансных частот. Проанализируйти фазовые соотношения между колебаниями и сделайте вывод.

Лабораторная работа № 8
Моделирование упругих деформаций стержней

Цель работы:

1) исследовать продольные деформации стержней под действием различных сил;

2) исследовать изгиб стержней под действием собственного веса и сосредоточенных сил.

Упругие деформации стержней

1. Растяжение и сжатие стержней

Возьмем однородный стержень длиной L, левое его основание закрепим, а к правому приложим растягивающую силу $\vec{F}$. Под действием этой силы длина стержня увеличивается на величину ΔL. Со стороны стержня на правое основание действует сила $\vec{F}_{уп}$, которая уравновешивает приложенную силу $\vec{F}$, т.е. $F = F_{уп}$. Если провести произвольное сечение стержня, перпендикулярное к его оси, то в условиях равновесия в этом сечении левый и правый отрезки стержня будут действовать друг на друга с равными по величине и противоположными по направлению силами $\vec{F}_1$ и $\vec{F}_2$, причем $F_1 = F_2 = F$.

Отношение силы F к площади поперечного сечения стержня S называется *напряжением:*

$$\sigma = F/S. \qquad (1)$$

В случае растяжения стержня напряжение называют *натяжением*. Если изменить направление силы F на противоположное, то длина стержня будет уменьшаться и соответствующее напряжение называют *давлением*. Очевидно, давление можно рассматривать как отрицательное натяжение. При однородной деформации стержня натяжение в любом его поперечном сечении одно и то же.

Отношение удлинения ΔL к длине недеформированного стержня L называется *относительным удлинением*

$$\varepsilon = \frac{\Delta L}{L}. \tag{2}$$

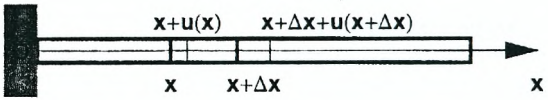
В случае растяжения стержня относительное удлинение положительно, а при сжатии – отрицательно.

Опыт показывает, что при небольших упругих деформациях натяжение σ пропорционально относительному удлинению ε:

$$\sigma = E\,\varepsilon, \tag{3}$$

где E – постоянная, зависящая только от материала стержня и его физического состояния. Она называется модулем Юнга, а само соотношение (3) выражает *закон Гука* для деформаций растяжения и сжатия. Закон Гука справедлив только при малых деформациях стержней ($\Delta L << L$) и может не выполняться при больших деформациях.

При неоднородных деформациях натяжение зависит от положения поперечного сечения. Пусть координатная ось OX совпадает с осью стержня.



Проведем два поперечных сечения стержня в точках с координатами x и $x + \Delta x$, выделяя отрезок стержня длиной Δx. Обозначим через $u(x)$ смещение из положения равновесия точек стержня, имеющих координату x. Тогда удлинение отрезка стержня длиной Δx будет равно:

$$(x + \Delta x + u(x + \Delta x)) - (x + u(x)) = u(x + \Delta x) - u(x) \approx \frac{du}{dx}\Delta x$$

а относительное удлинение:

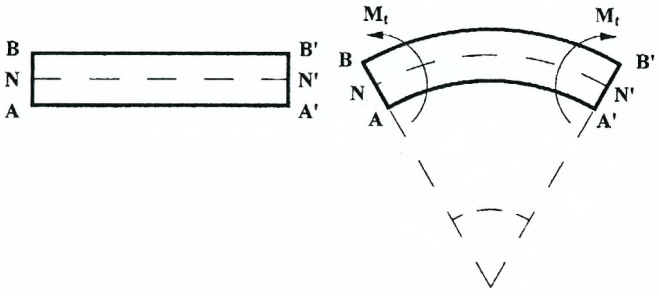
$$\varepsilon = \frac{du}{dx}.$$

Таким образом, сила упругости, действующая в поперечном сечении стержня, равна

$$F = SE\varepsilon = SE\frac{du}{dx}. \tag{4}$$

2. Изгиб стержней

При чистом изгибе продольные линии, нанесенные на поверхность стержня параллельно его оси, искривляются по дуге окружности, а поперечные сечения остаются плоскими и поворачиваются так, что остаются перпендикулярными к оси стержня.



Кривая NN', проходящая через центры масс поперечных сечений, при изгибе стержня не изменяет своей длины и называется *нейтральной линией*. При этом касательные напряжения в сечении отсутствуют, а сумма всех сил натяжения, действующих в каждом сечении стержня, равна нулю. Отличным от нуля является только момент сил натяжения M_t, причем его величина не зависит от оси, относительно которой он вычисляется, и равна:

$$M_t = \frac{E}{R} I,$$

где E – модуль Юнга, I – момент инерции поперечного сечения стержня, R – радиус кризизны нейтральной линии.

Пусть ось Ox направлена вдоль нейтральной линии недеформированного стержня, а ось Oy – перпендикулярна стержню, причем плоскость xOy является плоскостью изгиба. Если уравнение изогнутой нейтральной линии имеет вид: $y = u(x)$, то радиус ее кривизны вычисляется по формуле:

$$\frac{1}{R} = \frac{u''}{(1 + u'^2)^{3/2}}.$$

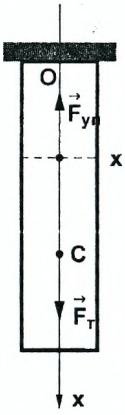
Если изгиб мал ($u' << 1$), то величиной u'^2 можно пренебречь. Тогда момент сил натяжения, действующих в сечении стержня, будет равен:

$$M_t = E I \frac{d^2 u}{d x^2}. \qquad (5)$$

Уравнение (5) называется *дифференциальным уравнением изогнутой оси стержня*.

Пример 1. Продольные деформации стержня под действием собственного веса

Пусть стержень расположен вдоль вертикальной оси *Ox*, причем для точек стержня $0 \le x \le L$. Пересечем его горизонтальной плоскостью в произвольной точке *x* и отметим силы, действующие на нижнюю часть стержня.



Сила тяжести, действующая на нижний кусок стержня, равна:

In[59]:=
```
Clear["Global`*"];
```

```
FT = ρ g (L - x);
```

Здесь ρ – линейная плотность стержня (т.е. масса единицы длины стержня), *g* – ускорение свободного падения.

Сила упругости находится из закона Гука (4).

In[61]:=
```
Fup = S E0 u'[x];
```

Здесь *S* – площадь поперечного сечения стержня, *E0* – модуль Юнга, а *u(x)* – функция, определяющая смещение относительно положения равновесия точек стержня, имеющих координату *x*. Напомним, что заглавная буква *E* в системе *Mathematica* используется для обозначения основания натурального логарифма, поэтому модуль Юнга мы обозначили через *E0*. Тогда условие равновесия нижнего отрезка стержня имеет вид:

In[62]:=
```
eq = FT == Fup
```

Out[62]=
```
g (L - x) ρ == E0 S u'[x]
```

Получили дифференциальное уравнение первого порядка относительно функции *u(x)*. Для его решения определяем граничное условие:

In[63]:=
```
initial = u[0] == 0;
```

Решаем уравнение *eq* с помощью функции **DSolve**.

In[64]:=
```
sol = DSolve[{eq, initial}, u[x], x][[1]]
```

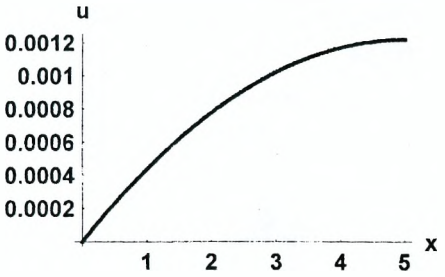
Out[64]= $\left\{u[x] \to \frac{2\,g\,L\,x\,\rho - g\,x^2\,\rho}{2\,E0\,S}\right\}$

Для построения графика найденной функции задаем численные значения параметров системы в виде списка *data*.

In[65]:=
```
data = {E0 → 10^10, L → 5, ρ → 20000, S → 0.2, g → 9.8};
```

Тогда график зависимости *u(x)* имеет вид:

In[67]:=
```
Plot[u[x] /. sol /. data, {x, 0, L /. data},
  PlotStyle → {Thickness[0.012], Hue[0.7]},
  AxesLabel → {"x", "u"}];
```



Вычисляя смещение точек стержня с координатами $x = L$, находим удлинение стержня.

In[68]:=
```
u[x] /. sol /. x → L /. data
```

Out[68]= `0.001225`

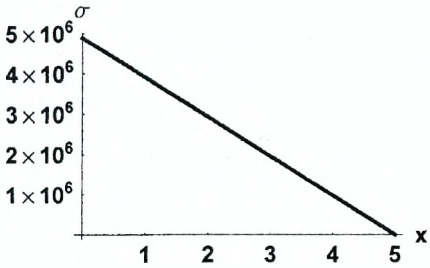
Напряжение как функцию координат определяем в виде:

In[69]:=
```
σ[x_] = E0 D[u[x] /. sol, x]
```

Out[69]= $\frac{2\,g\,L\,\rho - 2\,g\,x\,\rho}{2\,S}$

Соответствующий график имеет вид:

In[70]:=
```
Plot[σ[x] /. data, {x, 0, L /. data},
  PlotStyle → {Thickness[0.012], Hue[0.7]},
  AxesLabel → {"x", "σ"}];
```



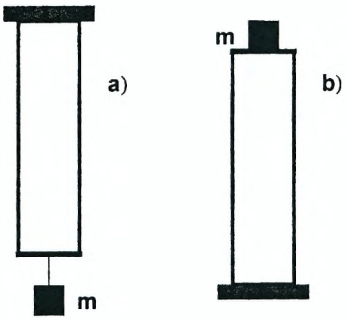
Напряжение стержня можно также изобразить с помощью цветовой функции. Например, следующая команда позволяет изобразить стержень так, что его

цвет изменяется от синего в точках с максмальным напряжением (верхняя точка стержня) к черному в точках с минимальным напряжением (нижняя точка стержня).

In[71]:=
```
DensityPlot[σ[-y] /. data,
  {x, 0, 0.5}, {y, 0, -L /. data},
    Mesh → False, AspectRatio → 3, PlotPoints → 100,
    ColorFunction → (RGBColor[0, 0, #] &),
  FrameTicks → None];
```

Задание 1. Исследование продольных деформаций стержня под действием собственного веса и дополнительного груза

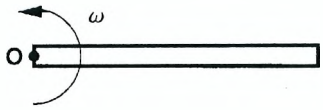
Однородный стержень подвешен вертикально и к его нижнему основанию прикреплен груз массой m (случай a)) или поставлен на горизонтальную плоскость, а на его верхнее основание положен груз массой m (случай b)).



1. Запишите дифференциальное уравнение, определяющее деформацию стержня $u(x)$, и найдите его решение.
2. Задайте численные значения параметров системы и постройте графики зависимостей $u(x)$ и $\sigma(x)$ при различных значениях массы груза m. Сделайте вывод о влиянии массы груза на деформацию стержня.
3. Постройте график зависимости удлинения стержня ΔL от массы груза m.

Задание 2. Исследование продольных деформаций стержня при его вращении

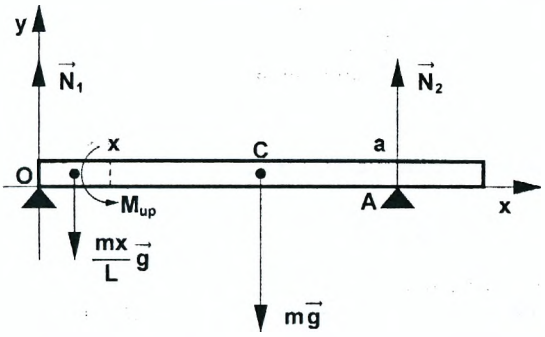
Тонкий стержень длиной L равномерно вращается вокруг перпендикулярной к нему оси, проходящей через конец стержня (точка O) с угловой скоростью ω.



1. Запишите дифференциальное уравнение, определяющее деформацию $u(r)$ стержня как функцию расстояния от оси вращения r, и найдите его решение.
2. Задайте численные значения параметров системы и постройте графики зависимостей $u(r)$ и $\sigma(r)$.
3. Постройте график зависимости удлинения стержня ΔL от угловой скорости его вращения ω.

Пример 2. Изгиб стержня, лежащего на двух опорах, под действием собственного веса

Пусть однородный стержень длиной L и массой m лежит на двух опорах, расположенных на горизонтальной оси Ox в точках O и A, причем $|OA| \equiv a$. Очевидно, стержень может находится в равновесии только при условии: $\frac{L}{2} \le a \le L$. На него действуют три силы: сила тяжести $m\vec{g}$, приложенная к центру масс C, и силы нормальной реакции опор $\vec{N}_1$ и $\vec{N}_2$.



Чтобы стержень находился в равновесии, необходимо, чтобы сумма действующих на него сил равнялась нулю, и сумма моментов этих сил относительно произвольной оси также равнялась нулю. Рассматривая в качестве оси прямую, проходящую перпендикулярно плоскости рисунка через точку O, запишем условия равновесия стежня.

In[72]:=
```
eq1 = {N1 + N2 - m g == 0, N2 a - m g L / 2 == 0};
```

Из уравнений *eq1* можно найти силы $\vec{N}_1$ и $\vec{N}_2$.

In[73]:=
```
sol1 = Solve[eq1, {N1, N2}][[1]]
```

Out[73]= $\left\{N1 \to -\frac{-2\,a\,g\,m + g\,L\,m}{2\,a},\ N2 \to \frac{g\,L\,m}{2\,a}\right\}$

Точка A, в которой приложена сосредоточенная сила $\vec{N}_2$, делит стержень на два отрезка: $0 \le x \le a$ и $a < x \le L$. Чтобы определить линию изгиба оси стержня, необходимо найти решения $u_1(x)$ и $u_2(x)$ дифференциального уравнения изогнутой оси (5) на каждом отрезке и потребовать выполнения условия отсутствия прогибов в точках опоры:

$$u_1(0) = 0,\ u_1(a) = 0,$$

и условия непрерывного и плавного сопряжения двух частей стержня:

$$u_1(a) = u_2(a),\ u_1'(a) = u_2'(a).$$

Проведем через произвольную точку с координатой $x < a$ сечение стержня. Естественно предположить, что стержень под действием силы тяжести будет прогибаться вниз. В этом случае вторая производная функции $u(x)$, определяющей смещение точек стержня вдоль оси Oy, будет положительной величиной. Поскольку момент упругих сил, действующих в сечении на левый отрезок стержня, положителен, то он может быть записан в виде:

```
In[74]:= Mup = E0 I0 u''[x];
```

Уравнение моментов, выражающее условие равновесия левого отрезка стержня, имеет вид:

```
In[75]:= eq2 = (m g x / L) x / 2 + Mup == N1 x /. sol1
```

$$\text{Out[75]= } \frac{g\,m\,x^2}{2\,L} + E0\,I0\,u''[x] == -\frac{(-2\,a\,g\,m + g\,L\,m)\,x}{2\,a}$$

Поскольку оба конца левой части стержня находятся на опорах и не смещаются, задаем следующие граничные условия:

```
In[76]:= initial1 = {u[0] == 0, u[a] == 0};
```

Тогда решение дифференциального уравнения *eq2*, определяющего изгиб левой части стежня, получаем в виде:

```
In[77]:= sol2 = DSolve[Join[{eq2}, initial1], u[x], x][[1]]
```

$$\text{Out[77]= } \left\{u[x] \to \frac{1}{24\,a\,E0\,I0\,L}\left(a^4\,g\,m\,x - 4\,a^3\,g\,L\,m\,x + 2\,a^2\,g\,L^2\,m\,x + 4\,a\,g\,L\,m\,x^3 - 2\,g\,L^2\,m\,x^3 - a\,g\,m\,x^4\right)\right\}$$

Теперь проведем сечение стержня в области $a < x < L$. В этом случае в правой части уравнения *eq2* следует добавить момент силы N_2. Тогда дифференциальное уравнение изогнутой оси (5) примет вид:

```
In[78]:= eq3 =
           (m g x / L) x / 2 + Mup == N1 x + N2 (x - a) /. sol1 // Simplify
```

$$\text{Out[78]= } \frac{g\,m\,(L-x)^2 + 2\,E0\,I0\,L\,u''[x]}{L} == 0$$

Легко убедиться в том, что полученное уравнение совпадает с условием равновесия правого отрезка стержня. Действительно, приравнивая момент силы тяжести, действующей на правый отрезок стержня, и момент упругих сил, действующих в сечении, получаем:

```
In[79]:= m g (L - x)/L (L - x)/2 == -Mup // Simplify
```

$$\text{Out[79]= } \frac{g\,m\,(L-x)^2 + 2\,E0\,I0\,L\,u''[x]}{L} == 0$$

Здесь мы учли, что под действием силы тяжести правый отрезок стержня должен изгибаться вниз. При этом вторая производная функции $u(x)$ отрицательна, и для получения положительного значения момента упругих сил мы взяли *Mup* со знаком минус.

Решение уравнения *eq3* должно удовлетворять следующему граничному условию:

In[80]:= `initial2 = u[a] == 0;`

В результате находим:

In[81]:= `sol3 = DSolve[{eq3, initial2}, u[x], x][[1]]`

Out[81]=
$$\left\{u[x] \to \frac{1}{24\,E0\,I0\,L}\left(a^4 g m - 4 a^3 g L m + 6 a^2 g L^2 m - 4 a g L^3 m + 4 g L^3 m x - 6 g L^2 m x^2 + 4 g L m x^3 - g m x^4 - 24 a E0\,I0\,L\,C[2] + 24 E0\,I0\,L\,x\,C[2]\right)\right\}$$

Поскольку уравнение *eq3* является дифференциальным уравнением второго порядка, то при одном заданном в *initial2* граничном условии найденное решение содержит произвольную постоянную *C[2]*. Уравнение для определения *C[2]* получаем, приравнивая производные функций, найденных в *sol2* и *sol3*, при $x = a$.

In[82]:= `eq4 = D[u[x] /. sol2, x] == D[u[x] /. sol3, x] /. x → a`

Out[82]=
$$\frac{-3 a^4 g m + 8 a^3 g L m - 4 a^2 g L^2 m}{24 a E0\,I0\,L} == \frac{-4 a^3 g m + 12 a^2 g L m - 12 a g L^2 m + 4 g L^3 m + 24 E0\,I0\,L\,C[2]}{24 E0\,I0\,L}$$

Соответствующее значение *C[2]* имеет вид:

In[83]:= `sol4 = Solve[eq4, C[2]][[1]] // Simplify`

Out[83]=
$$\left\{C[2] \to \frac{g\,(a^3 - 4 a^2 L + 8 a L^2 - 4 L^3)\,m}{24\,E0\,I0\,L}\right\}$$

Теперь можно определить функции $u_1(x)$ и $u_2(x)$. При этом удобно задать параметры *a* и *L* в виде двух других аргументов этих функций. В результате получаем:

In[84]:= `u1[x_, a_, L_] = u[x] /. sol2`

Out[84]=
$$\frac{1}{24 a E0\,I0\,L}\left(a^4 g m x - 4 a^3 g L m x + 2 a^2 g L^2 m x + 4 a g L m x^3 - 2 g L^2 m x^3 - a g m x^4\right)$$

In[85]:= `u2[x_, a_, L_] = u[x] /. sol3 /. sol4 // Simplify`

Out[85]=
$$\frac{g m (a - x)\,(a^2 x + a\,(-2 L^2 - 4 L x + x^2) + x\,(6 L^2 - 4 L x + x^2))}{24\,E0\,I0\,L}$$

Функцию, определяющую изгиб всего стержня, зададим в виде:

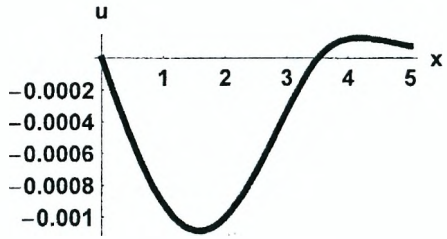
In[86]:= `uu[x_, a_, L_] := If[x ≤ a, u1[x, a, L], u2[x, a, L]];`

Чтобы построить график зависимости *uu(x)*, определяем численные значения параметров задачи.

In[87]:= `data = {E0 → 2 10^10, m → 1000, I0 → 0.0001, g → 9.8};`

Выбирая численные значения параметров *a* и *L* (с учетом $\frac{L}{2} < a < L$), получаем:

```
In[88]:= Plot[uu[x, 3.5, 5] /. data, {x, 0, 5},
          PlotStyle → {Thickness[0.02], Hue[0.7]},
          AxesLabel → {"x", "u"}];
```



Чтобы определить стрелу прогиба, найдем точку x, в которой производная функции $u_1(x)$ обращается в ноль.

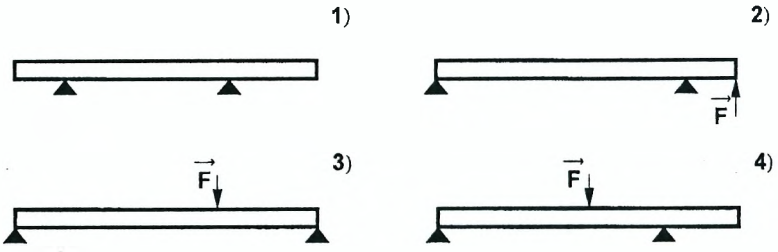
```
In[89]:= sol5 = Solve[D[u1[x, a, L], x] == 0, x] /. data /.
            {a → 3.5 , L → 5} // Chop
Out[89]= {{x → -1.12002}, {x → 3.82133}, {x → 1.58441}}
```

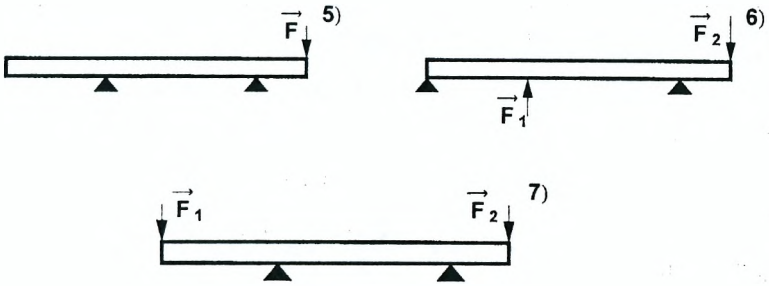
Очевидно, минимальному значению функции $u_1(x)$ соответствует третий корень. В этой точке значение функции $u_1(x)$ равно:

```
In[90]:= u1[x, a, L] /. data /. {a → 3.5 , L → 5} /. sol5[[3]]
Out[90]= -0.00108416
```

Задание 3. Исследование изгиба стержня под действием собственного веса и сосредоточенных сил

Однородный стержень длиной L и массой m лежит на двух опорах, расположенных на горизонтальной оси Ox. К стержню могут быть приложены одна или две сосредоточенные силы.

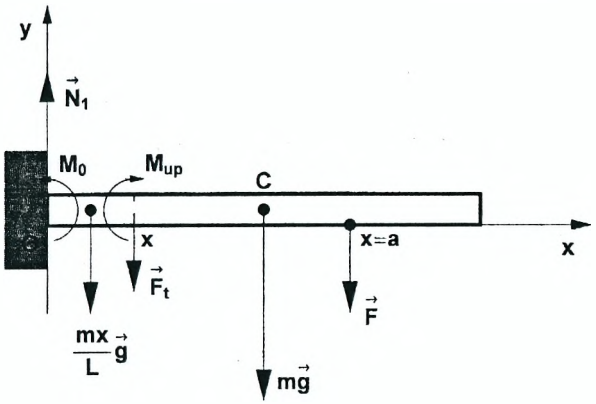




1. Запишите условия равновесия стержня на опорах и найдите силы реакции опор $\vec{N}_1$ и $\vec{N}_2$. Параметры, определяющие взаимное расположение стержня и опор, а также точки приложения сил, задайте самостоятельно.
2. Запишите дифференциальные уравнения, определяющие изгиб стержня на каждом участке между точками приложения сосредоточенных сил. Задайте граничные условия, а также условия непрерывного и плавного сопряжения соседних частей стержня, и найдите соответствующие решения дифференциальных уравнений.
3. Задайте численные значения параметров системы и постройте график зависимости $u(x)$.
4. Найдите координату точки, в которой прогиб стержня максимален, и вычислите стрелу прогиба.

Пример 3. Изгиб стержня, один конец которого закреплен в стене, под действием собственного веса и сосредоточенной силы

Пусть один конец однородного стержня длиной L заделан в стене, а второй является свободным. На расстоянии a от стены к стержню приложена сосредоточенная сила $\vec{F}$. Требуется определить линию изгиба оси стержня.



Со стороны стены на стержень в вертикальном направлении действует сила реакции $\vec{N}_1$, а также момент упругих сил M_0. Для равновесия стержня необходимо, чтобы сумма сил, действующих на стержень, равнялась нулю. Кроме того, момент упругих сил M_0 должен уравновесить моменты силы тяжести $m\vec{g}$ и силы $\vec{F}$. Соответствующая система уравнений имеет вид:

```
In[91]:= Clear["Global`*"];
         eq1 = {N1 - m g - F == 0, M0 - m g L / 2 - F a == 0}
```

$$\text{Out[92]= } \left\{-F - g\,m + N1 == 0,\ -a\,F - \frac{g\,L\,m}{2} + M0 == 0\right\}$$

Из уравнений *eq1* можно найти силу N_1 и момент M_0.

```
In[93]:= sol1 = Solve[eq1, {N1, M0}][[1]]
```

$$\text{Out[93]= } \left\{N1 \to F + g\,m,\ M0 \to \frac{1}{2}\,(2\,a\,F + g\,L\,m)\right\}$$

Проведем через точку с координатой $x < a$ сечение стержня и обозначим через F_t силу, действующую на левый отрезок стержня со стороны правого. Эта сила действует в плоскости сечения. Тогда первое условие равновесия этого отрезка запишется в виде:

```
In[94]:= eq2 = N1 - Ft - m g x / L == 0 /. sol1
```

$$\text{Out[94]= } F - Ft + g\,m - \frac{g\,m\,x}{L} == 0$$

Отсюда находим силу F_t.

```
In[95]:= sol2 = Solve[eq2, Ft][[1]]
```

$$\text{Out[95]= } \left\{Ft \to \frac{F\,L + g\,L\,m - g\,m\,x}{L}\right\}$$

Поскольку стержень под действием приложенных сил будет прогибаться вниз, момент упругих сил, действующих на левый отрезок стержня в сечении, равен:

```
In[96]:= Mup = -E0 I0 u''[x];
```

Уравнение моментов, выражающее второе условие равновесия левого отрезка стержня, имеет вид:

```
In[97]:= eq3 = (m g x / L) x / 2 + Ft x + Mup == M0 /. sol1 /. sol2
```

$$\text{Out[97]= } \frac{g\,m\,x^2}{2\,L} + \frac{x\,(F\,L + g\,L\,m - g\,m\,x)}{L} - E0\ I0\ u''[x] == \frac{1}{2}\,(2\,a\,F + g\,L\,m)$$

Поскольку левый конец стержня имеет заделку, граничные условия задаем в виде:

```
In[98]:= initial = {u[0] == 0, u'[0] == 0};
```

В результате получаем решение дифференциального уравнения *eq3*, определяющего изгиб левого отрезка стержня ($0 \le x \le a$).

```
In[99]:= sol3 = DSolve[Join[{eq3}, initial], u[x], x][[1]]
```

$$\text{Out[99]= } \left\{u[x] \to \frac{-12\,a\,F\,L\,x^2 - 6\,g\,L^2\,m\,x^2 + 4\,F\,L\,x^3 + 4\,g\,L\,m\,x^3 - g\,m\,x^4}{24\,E0\ I0\ L}\right\}$$

Теперь проведем сечение стержня через точку с координатой $a < x < L$. Силу F_t, действующую на левый отрезок стержня со стороны правого, найдем из условия равновесия этого отрезка:

```
In[100]:= eq4 = N1 - Ft - m g x / L - F == 0 /. sol1
```

$$\text{Out[100]}= -\text{Ft} + \text{g m} - \frac{\text{g m x}}{\text{L}} == 0$$

```
In[101]:= sol4 = Solve[eq4, Ft][[1]]
```

$$\text{Out[101]}= \left\{\text{Ft} \to \frac{\text{g L m} - \text{g m x}}{\text{L}}\right\}$$

Уравнение моментов для левого отрезка стержня запишем в виде:

```
In[102]:= eq5 = (m g x / L) x / 2 + Ft x + Mup + F a == M0 /. sol1 /. sol4 //
            Simplify
```

$$\text{Out[102]}= \frac{\text{g m } (\text{L} - \text{x})^2 + 2\,\text{E0 I0 L u}''[\text{x}]}{\text{L}} == 0$$

Очевидно, уравнение *eq5* совпадает с условием равновесия правого отрезка стержня длиной $(L - x)$. Его решение имеет вид:

```
In[103]:= sol5 = DSolve[eq5, u[x], x][[1]]
```

$$\text{Out[103]}= \left\{\text{u[x]} \to -\frac{\text{g m } (\text{L} - \text{x})^4}{24\,\text{E0 I0 L}} + \text{C[1]} + \text{x C[2]}\right\}$$

Поскольку это решение найдено для участка $a < x < L$, граничные условия *initial* для него не подходят. Неизвестные постоянные *C[1]* и *C[2]*, возникающие при интегрировании уравнения *eq5*, найдем из условия непрерывного и плавного сопряжения двух отрезков стержня. Соответствующая система уравнений определена в *eq6*.

```
In[104]:= eq6 = { (u[x] /. sol3) == (u[x] /. sol5) /. x → a,
                  D[u[x] /. sol3, x] == D[u[x] /. sol5, x] /. x → a}
```

$$\text{Out[104]}= \left\{\frac{-8\,\text{a}^3\,\text{F L} - \text{a}^4\,\text{g m} + 4\,\text{a}^3\,\text{g L m} - 6\,\text{a}^2\,\text{g L}^2\,\text{m}}{24\,\text{E0 I0 L}} == -\frac{\text{g } (-\text{a} + \text{L})^4\,\text{m}}{24\,\text{E0 I0 L}} + \text{C[1]} + \text{a C[2]},\right.$$

$$\frac{-12\,\text{a}^2\,\text{F L} - 4\,\text{a}^3\,\text{g m} + 12\,\text{a}^2\,\text{g L m} - 12\,\text{a g L}^2\,\text{m}}{24\,\text{E0 I0 L}} == \left.\frac{\text{g } (-\text{a} + \text{L})^3\,\text{m}}{6\,\text{E0 I0 L}} + \text{C[2]}\right\}$$

Решая систему, получаем:

```
In[105]:= sol6 = Solve[eq6, {C[1], C[2]}][[1]] // Simplify
```

$$\text{Out[105]}= \left\{\text{C[1]} \to \frac{4\,\text{a}^3\,\text{F} + \text{g L}^3\,\text{m}}{24\,\text{E0 I0}},\ \text{C[2]} \to -\frac{3\,\text{a}^2\,\text{F} + \text{g L}^2\,\text{m}}{6\,\text{E0 I0}}\right\}$$

Итак, на первом участке $0 \le x \le a$ изгиб стержня определяется функцией $u_1(x, a, L)$, найденной в *sol3*.

In[106]:=
```
u1[x_, a_, L_] = u[x] /. sol3 // Simplify
```

Out[106]=
$$-\frac{x^2\,(12\,a\,F\,L-4\,F\,L\,x+g\,m\,(6\,L^2-4\,L\,x+x^2))}{24\,E0\,I0\,L}$$

На втором участке $a < x < L$ соответствующая функция $u_2(x, a, L)$ имеет вид:

In[107]:=
```
u2[x_, a_, L_] = u[x] /. sol5 /. sol6 // Simplify
```

Out[107]=
$$-\frac{-4\,a^3\,F\,L+12\,a^2\,F\,L\,x+g\,m\,x^2\,(6\,L^2-4\,L\,x+x^2)}{24\,E0\,I0\,L}$$

Теперь можно определить функцию $uu(x, a, L)$, описывающую изгиб всего стержня.

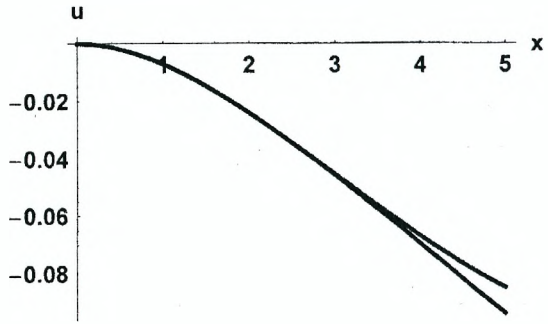
In[108]:=
```
uu[x_, a_, L_] := If[x < a, u1[x, a, L], u2[x, a, L]]
```

Чтобы построить график зависимости $uu(x)$, определяем численные значения параметров задачи.

In[109]:=
```
data =
    {E0 → 2 10^10, m → 1000, I0 → 0.0001, g → 9.8, F → 4000};
```

Для сравнения построим также график функции $u_1(x)$, которая описывает изгиб левого отрезка стержня.

In[110]:=
```
Plot[
    {u1[x, 2, 5] /. data, uu[x, 2, 5] /. data}, {x, 0, 5},
    PlotStyle → {{Thickness[0.01], Hue[0.]},
      {Thickness[0.01], Hue[0.7]}},
    AxesLabel → {"x", "u"}];
```



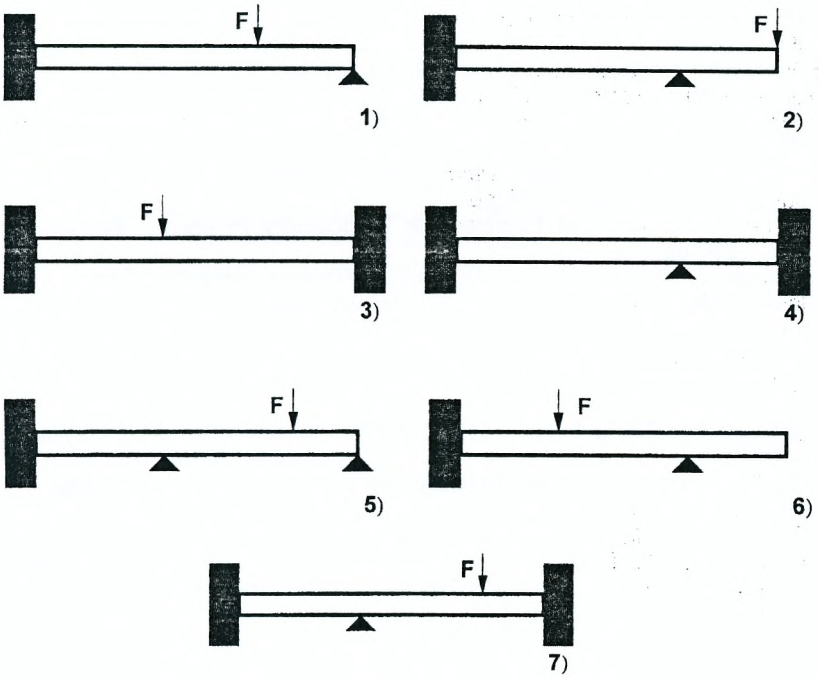
Очевидно, стрела прогиба равняется значению функции $u_2(x)$ в крайней правой точке стержня.

In[111]:=
```
u2[5, 2, 5] /. data
```

Out[111]=
```
-0.0938958
```

Задание 4. Исследование изгиба стержня под действием собственного веса и сосредоточенной силы при различных условиях его закрепления

Однородный стержень длиной L и массой m закреплен, как показано на рисунке.



1. Запишите условия равновесия стержня и разрешите их относительно двух сил реакции опор. Параметры, определяющие взаимное расположение стержня и опор, а также координату точки приложения силы F, задайте самостоятельно.
2. Запишите дифференциальные уравнения, определяющие изгиб стержня на каждом участке. Задайте граничные условия и найдите соответствующие решения дифференциальных уравнений.
3. Используя условия непрерывного и плавного сопряжения соседних частей стержня, запишите систему уравнений для определения неизвестных сил и моментов, а также постоянных, возникающих при решении дифференциальных уравнений.
4. Задайте численные значения параметров системы и постройте график зависимости $u(x)$.
5. Найдите координату точки, в которой прогиб стержня максимален, и вычислите стрелу прогиба.

Литература

1. Е.М.Воробьев. Введение в систему "Математика": Учебн. пособие.– М.: Финансы и статистика, 1998. – 262 с.
2. В.П.Дьяконов. *Mathematica* 4: Учебный курс. – СПб.:Питер, 2001. – 656 с.
3. Т.В.Капустина. Компьютерная система *Mathematica* 3.0 для пользователей: Справочное пособие. – М.: Солон-Р, 1999. – 240 с.
4. А.Н.Прокопеня, А.В.Чичурин. Применение системы *Mathematica* к решению обыкновенных дифференциальных уравнений: Учебн. пособие. – Мн.:БГУ, 1999. – 265 с.
5. N.Bellomo, L.Preziosi, A.Romano. Mechanics and Dynamical Systems with *Mathematica*. – Boston, Birkhauser, 2000. – 417 pp.
6. A.Marasco, A.Romano. Scientific Computing with *Mathematica*. – Boston, Birkhauser, 2001. – 270 pp.
7. R.L.Zimmerman, F.I.Olness. *Mathematica* for Physics. – Redwood City, Addison-Wesley, 1995. – 436 pp.
8. R.L.Varley. *Mathematica* Exercises in Introductory Physics. – Prentice Hall, 1996. – 207 pp.
9. G.Baumann. *Mathematica* in Theoretical Physics: Selected Examples from Classical Mechanics to Fractals. – TELOS/Springer-Verlag, 1996. – 348 pp.
10. S.Wolfram. The *Mathematica* book, 4th ed. – Wolfram Media/Cambridge University Press, 1999. – 1470 pp.
11. Д.В. Сивухин. Общий курс физики. Т. I. Механика. – М.: Наука,1979.– 520 с.
12. Д.В.Сивухин. Общий курс физики. Т. III. Электричество. – М.: Наука, 1977. – 688 с.
13. Л.Д.Ландау, Е.М.Лифшиц. Теоретическая физика: Учебн. пособие.– В 10-ти т. Т. I. Механика. – 4-е изд. – М.: Наука, 1988. – 216 с.
14. Л.Д.Ландау, Е.М.Лифшиц. Теоретическая физика: Учебн. пособие.– В 10-ти т. Т. VII. Теория упругости. – 4-е изд. – М.: Наука, 1987. – 248 с.
15. Сопротивление материалов / Под ред. Г.С.Писаренко. – 5-е изд. – К.: Вища шк. Головное изд-во, 1986. – 775 с.

СОДЕРЖАНИЕ

УЧЕБНОЕ ИЗДАНИЕ

Александр Николаевич Прокопеня

РЕШЕНИЕ ФИЗИЧЕСКИХ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ СИСТЕМЫ Matematica

Ответственный за выпуск: ***Прокопеня А.Н.***
Редактор: ***Строкач Т. В.***
Корректор: ***Никитчик Е.В.***

Лицензия № 02330/0148711.
Лицензия № 02330/0133017.
Подписано к печати 17.10.2005 г. Бумага «Снегурочка».
Формат 60x84 $^{1}/_{16}$. Усл. п. л. 15,0.
Уч. изд. л. 16,25. Тираж 250 экз. Заказ № 1003.
Отпечатано на ризографе учреждения образования
«Брестский государственный технический университет».
224017, г. Брест, ул. Московская, 267.