

## НАЗНАЧЕНИЕ MATLAB

Система Matlab (Matrix Laboratory – матричная лаборатория) разработана компанией MathWork. Это мощная вычислительная система, предназначенная для решения широкого круга математических, инженерных и экономических задач.

Работая в среде Matlab(ML), пользователь, даже не являющийся программистом, может легко и быстро решать вычислительные задачи в различных областях науки и техники (линейная алгебра, теория управления, обработка сигналов и т.п.). Простая командная среда позволяет вводить выражения в форме, близкой к естественной математической записи.

В наибольшей степени система ориентирована на выполнение инженерных расчетов. Математический аппарат оптимизирован для вычислений, проводимых с матрицами и комплексными числами. Matlab содержит множество встроенных функций, необходимых инженеру и научному работнику для выполнения сложных численных расчетов, а также моделирования поведения технических систем и физических процессов. Мощная графическая система Matlab позволяет визуализировать представление данных, что делает возможным графический анализ результатов. Все функциональные возможности объединены удобным пользовательским интерфейсом.

Большим плюсом системы является ее открытость и расширяемость. В ней могут быть написаны программы для многократного использования. Пользователь может не только использовать имеющиеся функции, но и создавать собственные специализированные функции. Пакет позволяет работать с программами, написанными на языках Фортран и Си. Большинство специализированных функций хранятся в виде текстовых файлов с расширением *m* (*m*-файлов). Файлы могут быть созданы в редакторе среды Matlab либо во внешнем редакторе, совместимом по кодировке с встроенным редактором. Созданные пользователем *m*-файлы могут использоваться так же, как и встроенные в Matlab функции.

Помимо работы с программами (функциями), вычисления можно выполнять в режиме ”калькулятора”, т.е. получать результат сразу после ввода очередной команды. Для решения специальных задач с помощью ML разработаны пакеты с дополнительными функциями, которые называются *Toolboxes*.

Предусмотрена возможность интегрирования системы ML с Microsoft Word и Microsoft Excel.

## ИНТЕРФЕЙС MATLAB 6.5

После запуска программы Matlab на экран выводится основное окно рабочей среды ML, которое называется «рабочий стол» (рис. 1).

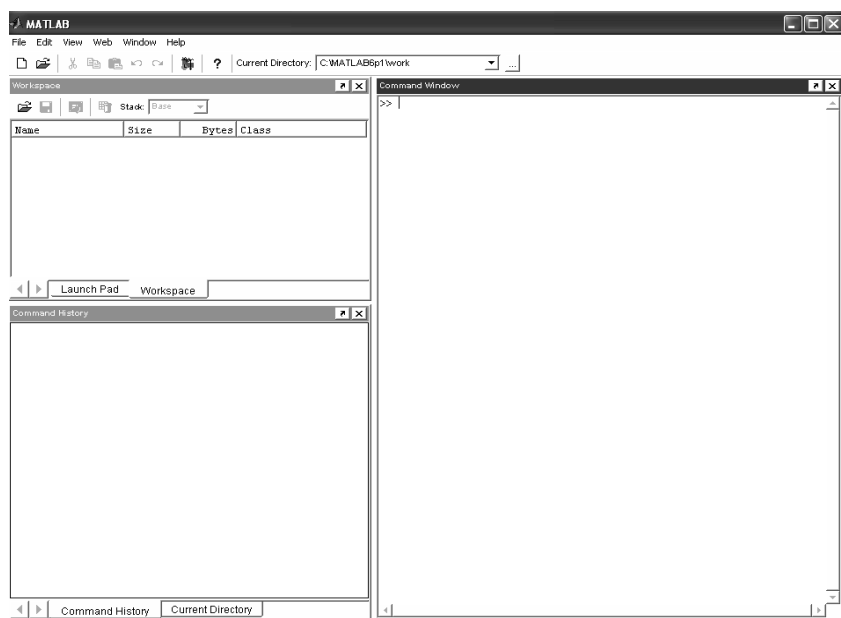


Рис. 1

Это окно содержит:

- строку заголовка;
- строку главного меню, в которой находятся пункты меню *File* (Файл), *Edit* (Редактирование), *View* (Вид), *Web*, *Window* (Окно), *Help* (Справка).

- панель инструментов с кнопками, позволяющими выполнить некоторые наиболее распространенные операции, которые можно выбрать и через меню. Рядом расположено окно *Current Directory*, позволяющее установить текущую папку;


- три внутренних окна:

- справа самое большое и важное окно – *Command Window* (командное окно), где отображаются вводимые пользователем данные, команды и результаты их выполнения, а также сообщения об ошибках;

- слева сверху окно с вкладками *Workspace* (рабочее пространство), позволяющее получить доступ к содержимому рабочей области, и *Launch Pad*, служащее для просмотра содержимого папки *ML* (например, чтобы просмотреть демонстрационные примеры или вызвать help);

- слева внизу окно с вкладками *Command History* (история команд), предназначенное для просмотра и повторного вызова ранее введенных команд, и окно *Current Directory* (текущая папка), служащее для установки текущего каталога;

- строку состояния, где отображаются сообщения системы.

В правом углу каждого окна две кнопки: X – закрыть;  – открепить от рабочего стола. Для возвращения в прежнее состояние необходимо выбрать пункты меню: *View, Desktop Layout, Default*.

Все используемые в текущем сеансе работы переменные хранятся в памяти компьютера в области, называемой *Workspace* (рабочее пространство). Полный перечень заданных переменных отображается в окне *Workspace*. Здесь содержится полный список переменных и их размеры, но не значения переменных. Двойной щелчок по имени переменной вызывает открытие окна *Array Editor*, которое можно использовать для редактирования отдельных элементов векторов и матриц.

Регулировать размеры окон можно с помощью мыши.

### **Основные команды главного меню ML**

Работа с меню в ML подобна работе с меню программ Microsoft Office. Многие элементы меню имеют для быстрого вызова команд горячие клавиши (их можно видеть справа от команды, например, *Copy*→*Ctrl+C* и т.п.).

Рассмотрим состав и назначение некоторых команд меню.

**Работа с файлами.** Пункт меню *File* содержит следующие команды (рис. 2):



Рис. 2

*New – m-file* – позволяет создать новый *m*-файл в редакторе ML;

*Open* – позволяет открыть имеющийся *m*-файл;

*Save Workspace* – позволяет сохранить переменные рабочего пространства;

*Set Path* – позволяет установить путь, указывающий расположение *m*-файлов;

*Preferences* – позволяет отображать диалоговое окно, в котором можно настроить различные параметры системы.

В окне пункта меню *File* отображаются имена четырех последних файлов, к которым обращался пользователь.

**Все файлы, которые будут созданы студентом в ML, должны храниться в предназначенной для этого папке, где разрешено сохранение файлов: на диске C: (C:\USERS\<папка>) или на сервере (N:\<папка>).** Всегда есть возможность установить нужный каталог в качестве текущего. Это можно сделать с помощью пункта меню *File*, *Set Path* или в окне *Current Directory* (Текущий каталог).

Команда *Exit* позволяет выйти из системы.

**Редактирование.** Пункт меню *Edit* содержит традиционные команды: *Undo* и *Redo* (отмена последнего действия и отказ от отмены соответственно), *Cut* (вырезать фрагмент), *Copy* (копировать), *Paste* (вставить), *Delete* (удалить), *Select All* (выделить все).

Используя команды *Clear Command Window*, *Clear Command History*, *Clear Workspace*, можно очистить содержимое командного окна, окна истории команд и рабочее пространство соответственно.

**Управление конфигурацией рабочего стола.** Пункт меню *View* (рис. 3) позволяет при необходимости управлять конфигурацией рабочего стола (наличием окон можно управлять, ставя или убирая галочки около выбранных команд меню).

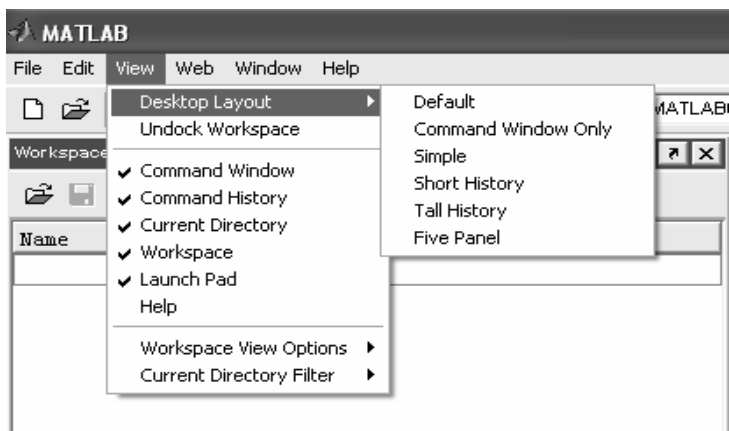


Рис. 3

**Работа с окнами.** Пункт меню *Window* служит для работы с окнами системы. Здесь отображаются команды, соответствующие окнам и программам, которые в данный момент открыты. Выбор одной из этих команд позволяет сделать активным требуемое окно или код программы в редакторе *m*-файлов.

Пункт меню *Help* предоставляет доступ к справочной системе программы *Matlab*.

**Командное окно *Command Window*.** Это окно предназначено для ввода чисел, переменных, выражений и команд. Здесь же выводятся результаты работы и сообщения об ошибках. О готовности системы к вводу свидетельствует знак `>>`, который располагается в текстовом поле командного окна (в командной строке). Для выполнения введенной команды следует нажать клавишу `<Enter>`. Пока она не нажата, вводимое выражение может быть отредактировано или удалено. Просмотр содержимого окна можно осуществить, используя полосы прокрутки, а также клавиши `<PgUp>`, `<PgDown>`, `<Ctrl+Home>`, `<Ctrl+End>`, `<Home>`, `<End>`.

Клавиши управления `↑` и `↓` в ML имеют совсем другое назначение, чем в текстовых редакторах. Использование этих клавиш позволяет отобразить в командной строке ранее введенные с клавиатуры команды и выражения для их повторного использования. Это возможно потому, что все выполненные команды сохраняются в специальной области памяти. Чтобы стереть содержимое командного окна, достаточно набрать и выполнить команду `clc`.

**Рабочая область *Workspace*.** В ML все переменные, используемые в текущем сеансе работы, хранятся в области, называемой рабочей областью или рабочим пространством. Полный перечень используемых переменных отображается в окне *Workspace*, где можно видеть список текущих переменных и их размеры. Эта информация представлена в виде таблицы, состоящей из следующих столбцов: *Name* – указывается имя переменной, *Size* – отображается размер переменной, *Bytes* – отображается количество выделенной памяти, *Class* – тип переменной.

Окно *Workspace* имеет панель инструментов, включающую команды для открытия файлов с данными, создания, сохранения и удаления переменных. Если дважды щелкнуть по строке, соответствующей какой-либо переменной, информация о ней отобразится в окне *Array Editor*.

**История команд. Окно *Command History*** – здесь отображаются дата и время сеанса работы в ML, а также содержится перечень команд, введенных в течение текущего сеанса. Их можно заново выполнить, дважды щелкнув по команде. Можно выполнить подряд несколько команд, находящихся в этом окне. Для этого надо выделить команды с помощью мыши, удерживая нажатой клавишу <Shift>, а затем нажать <Enter>. Если же нужные команды располагаются не подряд, то их надо выделить, удерживая клавишу <Ctrl>. Для повторного использования команд можно также воспользоваться стрелками ↑↓, помещая эти команды в командную строку. При нажатии правой клавиши мыши на какой-либо команде в этом окне отображается контекстное меню, содержащее команды копирования в буфер обмена выделенной команды, ее удаления, создания *m*-файла и т.п.

## ВЫЧИСЛЕНИЯ

### Особенности ввода команд и данных

Команды вводятся в окне *Command Window* в командной строке после приглашения системы >>. Для выполнения введенной команды надо нажать клавишу <Enter>. Для хранения выполненных команд предусмотрен кольцевой буфер. Для просмотра и выбора предыдущих команд используются клавиши управления: ↑ и ↓. Все выполненные команды можно видеть в окне *Command*

*History*. Любую предыдущую команду можно вызвать для исполнения, дважды щелкнув по ее отображению в этом окне.

В ML строчные и прописные буквы различаются. Это надо учитывать при записи команд. Принято обозначать все векторы и матрицы **прописными** (большими) буквами, функции – **строчными** (маленькими).

Если команда при наборе не помещается в строку, то в конце строки ставятся три точки (...) без пробелов, затем нажимается <Enter> и дальше набор продолжается с новой строки.

Результат выполнения каждой команды сразу отображается на экране. **Если в конце строки (после команды) стоит точка с запятой, то результат не будет выведен на экран.** При желании можно вводить несколько команд в одной строке, разделяя их точкой с запятой.

Простейший способ работы в ML – это **режим прямых вычислений (режим калькулятора)**. Команды выполняются сразу после их ввода, а результат выводится сразу после выполнения команды.

Например, вводим команду:

```
>> sin(0.5)
```

Нажимаем <Enter>

Получаем ответ:

```
ans =
```

```
0.4794
```

Вводим команду:

```
>> 3^2-(5+4)/2+6*3
```

Нажимаем <Enter> и получаем ответ:

```
ans =
```

```
22.5000
```

ML вычисляет выражение, помещает его в специальную переменную *ans* и выводит полученное значение в отдельной строке.

Текущий сеанс работы пользователя с системой Matlab называют *сессией*.

## Элементы данных в ML

Все данные в ML интерпретируются как массивы. Массив – это упорядоченный набор однородных данных. Элементами массива могут быть целые, вещественные, комплексные числа и символы. Каждый массив имеет имя и характеризуется числом измерений и

количеством элементов по каждому измерению. Доступ к элементу массива осуществляется указанием имени и индексов. Нумерация элементов начинается с единицы по каждому измерению.

Основной элемент данных в ML – матрица. Числа и векторы рассматриваются как вырожденные матрицы. Например, число – это матрица размером  $1 \times 1$ . Векторы – это матрицы с одним столбцом или с одной строкой.

Числа в ML могут быть целыми, вещественными, комплексными. Целые числа вводятся в обычной форме. Вещественные числа могут записываться в естественной форме с фиксированной точкой (2.5) и в экспоненциальной ( $3.4e-3$ ). При вводе между цифрами числа не допускаются пробелы.

### Форматы представления результатов вычислений

Вид результатов вычислений зависит от установленного формата вывода. Пользователь может задавать различные форматы представления чисел. По умолчанию установлен формат *short (4 цифры после десятичной точки)* – краткое представление числа.

Установить другой формат можно, если выбрать пункт меню *File, Preferences, Numeric format* (формат числа), *OK*. Этот формат будет использоваться для вывода результатов всех последующих вычислений, пока не будет изменен (рис. 4).

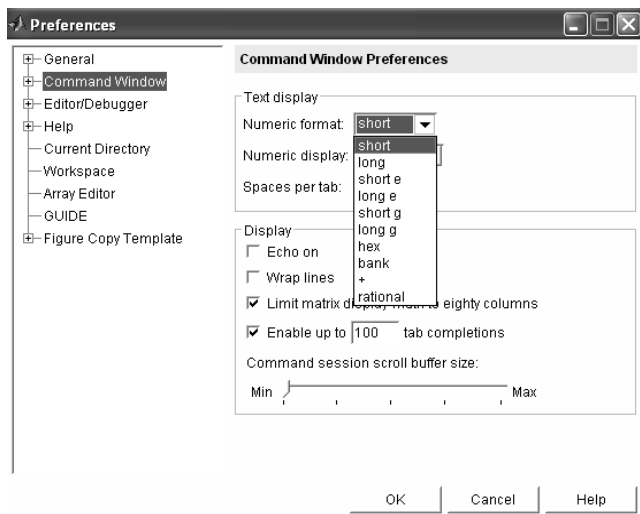


Рис. 4



Некоторые форматы: *long* (14 цифр) – длинное представление числа с фиксированной точкой, *short e* (четыре значащих цифры мантиссы и три знака порядка) – краткое представление числа в экспоненциальной форме, *long e* – длинное представление числа в экспоненциальной форме, *long g* –выбирает наиболее удачное представление для числа, *bank* (2 цифры после десятичной точки) – представление для денежных сумм и т.д.

Управлять форматом можно не только с помощью меню. Этого же результата можно достичь, если ввести в командную строку команду: ***format <формат>***.

Например,

```
>> format long
>> 7/8
ans =
    0.8750000000000000
>> format long g
>> 7/8
ans =
    0.875
>> format long e
>> 7/8
ans =
    8.750000000000000e-001
```

Для получения информации обо всех форматах необходимо ввести в командную строку команду: ***help format***.

## Переменные в ML

В ML, как и в других языках программирования, существует возможность работы с переменными. Любая переменная до использования в формулах должна быть определена. Для этого надо присвоить ей значение. **Типы переменных заранее не объявляются. Тип определяется значением, которое присваивается переменной.** В качестве оператора присваивания используется знак равенства(=):

```
>> n=5
n =
    5
```

```
>> k=0.5  
k =  
0.5000
```

По типу переменные могут быть числовыми (целые, вещественные, комплексные) или символьными.

Правила составления идентификаторов такие же, как и в ЯВУ (начинается с буквы и может содержать любые комбинации цифр, букв, символа подчеркивания, идентифицируются первые 63 символа, нельзя использовать специальные символы и пробелы, имя переменной должно быть уникальным и не совпадать с именами функций). **Следует помнить, что строчные и прописные буквы различаются** (Abc и abc – разные имена).

Существуют **встроенные системные переменные**, которые можно использовать, не задавая значения. Они при необходимости могут быть переопределены, т.е. им можно присвоить другие значения, но тогда значения, заданные по умолчанию, будут утеряны:

- *pi* – число  $\pi$  ( $\pi=3.141592653589793$ );
- *eps* – погрешность операций над числами с плавающей точкой; очень маленькое значение, равное  $2.2 \cdot 10^{-16}$ . Её используют, когда надо исключить деление на 0 (например,  $x/(y+eps)$ );
- *ans* – результат последней команды, если в ней нет операции присваивания. Например, если набрали команду 41/75 и не определили, какой переменной присвоить результат, на экране отобразится *ans* = 0.5467;
- *nan* – для обозначения неопределённости результата;
- *i*, *j* – мнимая единица ( $\sqrt{-1}$ ), используемая для задания мнимой части комплексного числа. Задать комплексные числа можно так:  $z = 3+4i$ .

Все эти переменные можно использовать в математических выражениях.

В ML все переменные занимают определенное место в памяти, называемой рабочим пространством системы – *Workspace*.

ML запоминает значения всех переменных, используемых в текущей сессии. Для того чтобы узнать, какие переменные были задействованы, используется команда *who*. Чтобы получить более подробную информацию о переменных: размеры, размерность, можно использовать команду *whos*. Эту информацию можно увидеть и в окне *Workspace*.

Для удаления из памяти всех переменных используется команда *clear*.

Выборочное удаление переменных можно сделать, применив команду *Clear* следующим образом: *Clear a b*.

После завершения сессии ML все переменные рабочего пространства, определенные в текущем сеансе работы, будут потеряны. При необходимости можно сохранить значения переменных в файле. Для этого нужно выбрать пункт меню *File, Save Workspace As*, далее в диалоговом окне задать имя файла для сохранения данных. Результаты будут сохранены в бинарном файле с расширением *mat*. Для восстановления значений переменных в следующем сеансе работы необходимо выбрать *File, Open, имя файла*. В результате все переменные станут доступны.

Другой способ сохранения и восстановления переменных рабочего пространства – набрать команды:

*Save <имя файла.mat>* и *Load <имя файла.mat>* соответственно.

Всю сессию (все, что отображается в командном окне) можно сохранить и поместить в текстовый файл. Для этого необходимо выполнить следующую команду:

*diary <имя>.txt*

Эта команда начинает запись сессии в текстовый файл. По команде *diary off* запись прекращается. Команда *diary on* возобновляет запись.

## ЗАДАНИЕ ВЕКТОРОВ И МАТРИЦ

### Способы задания векторов

Вектор – это одномерный массив данных. Вектор в ML – это матрица из одного столбца или одной строки. Соответственно вектор может быть вектором-столбцом или вектором-строкой.

Для задания вектора можно воспользоваться одним из приведенных ниже способов.

1. Можно задать значения вектора поэлементно:

*<имя пер.>=[<значение1> <значение2> ....<значениеN>]*

Например,

*>>P=[3 5 7 12]*

Значения элементов записываются через пробел или через запятую.

В результате выполнения этой команды создается вектор-строка:

```
P =  
    3    5    7   12
```

Если при задании вектора значения его элементов разделить точкой с запятой, то получим вектор-столбец:

```
<<A=[1; 2; 3]  
A=  
    1  
    2  
    3
```

2. Если элементы вектора являются арифметической прогрессией, то можно задать вектор так:

*<имя пер.>=<нач. значение>:<шаг>:<конечное значение>*

Например,

```
>>X=0 : 0.5: pi
```

В результате будет сформирован вектор со значениями:

```
X =  
    0    0.5000    1.0000    1.5000    2.0000    2.5000    3.0000
```

Шаг должен быть больше нуля. Если он равен единице, то его можно не указывать:

```
>>X = 1 : 10  
X =  
    1    2    3    4    5    6    7    8    9   10
```

Чтобы изменить форму вектора, надо записать X' – тогда вектор отобразится в виде столбца. Такая операция называется транспонированием.

Например,

```
>> X=1:1:5  
X =    1    2    3    4    5  
>> X'
```

```
ans =  
1  
2  
3  
4  
5
```

3. Также для формирования арифметической прогрессии можно использовать функцию ***linspace***:

***linspace*** (<нач. значение>, <кон. значение>, <кол. значений>)

Например,

```
>>B=linspace(0,pi,5)
```

B =

```
0 0.7854 1.5708 2.3562 3.1416
```

4. Вектор можно задать также путём объединения нескольких векторов.

Например,

```
>>A=[1 2 3]; B=[4 5 6]; C=[7 8 9]; D=[A B C]
```

D =

```
1 2 3 4 5 6 7 8 9
```

Для определения длины вектора предназначена функция ***length***:

```
>>l=length(D)
```

l =

```
9
```

Для доступа к элементу вектора необходимо указать его имя и в круглых скобках номер элемента. Например,  $D(3)$ . Для обращения к последнему элементу вектора можно записать:  $D(\text{length}(D))$  или  $D(\text{end})$ .

### Задание матриц

При задании матриц данные строк записываются через пробел или через запятую. Элементы разных строк разделяются знаком точка с запятой (;) или записываются с новой строки. В ML матрица хранится в порядке следования по столбцам.

Зададим матрицу:

```
>>A=[1 2 3; 4 5 6;7 8 9]
```

```
A =
```

```
1   2   3
4   5   6
7   8   9
```

```
или
```

```
>> A=[1 2 3
```

```
4 5 6
```

```
7 8 9]
```

```
A =
```

```
1   2   3
4   5   6
7   8   9
```

Обращение к элементу матрицы: <имя> (<индексы>), например,  $A(2,3)$ .

## ОПЕРАЦИИ

В ML можно не только выполнять обычные арифметические операции над числами и вычислять значения функций, но и производить операции над векторами и матрицами.

### Арифметические операции (АО)

К арифметическим операциям в ML относятся: сложение (+), вычитание (-), умножение (\*), деление (/), обратное деление (\), возведение в степень (^), транспонирование (').

Приоритеты АО:

- возведение в степень, транспонирование;
- умножение, деление;
- сложение, вычитание.

Все арифметические операции являются “матричными” и осуществляются по правилам линейной алгебры.

**При необходимости поэлементного выполнения операций над матрицами и векторами перед знаками операций ^, \*, /, \ следует ставить точку:**

```
>> X=[1 2 3 4];
```

```
>> X.^2
```

```
ans =
```

```
1   4   9  16
```

В данном случае каждый элемент исходного вектора возводится в квадрат. Выполнение команды  $X^2$  невозможно, так как это противоречит правилам матричной алгебры.

Транспонирование тоже бывает с точкой, тогда для комплексных чисел оно выполняется без комплексного сопряжения.

При выполнении арифметических операций с матрицами необходимо учитывать их размерность.

Каждой арифметической операции в ML соответствует определенная функция. Например, *plus(x,y)* – сложение массивов, *times(x,y)* – поэлементное умножение массивов, *mtimes(x,y)* – матричное умножение и т.д.:

```
>> 2+3
ans =
    5
>> plus(2,3)
ans =
    5
>> X=[1 2 3 4];
>> Y=[5 6 7 8];
>> times(X,Y)
ans =
    5    12    21    32
>> X.*Y
ans =
    5    12    21    32
```

Выполнение арифметических операций с матрицами будет подробно рассмотрено далее.

### Операции отношения

К операциям отношения в ML относятся: равно (`=`), не равно (`~ =`), меньше (`<`), меньше или равно (`<=`), больше (`>`), больше или равно (`>=`).

Операции отношения используются для поэлементного сравнения двух операндов (чисел, матриц, векторов одинакового размера). Результатом операции отношения может быть соответственно число, матрица или вектор, состоящие из элементов, обозначающих «истина» или «ложь». В ML это 1 и 0 соответственно:

```
>>A=[1 0 3; -2 5 -6];
>> B=[8 -9 1; 7 2 2];
>> A>B
ans =
    0    1    1
    0    1    0
```

В результате получили матрицу, каждый элемент которой имеет значение «истина» или «ложь»:

```
>> x=1; y=2; z=3;
>> ((x+y)==z)+(y<z)+(x<=y)
ans =
    3
```

### Логические операции

В ML существует возможность представления логических выражений с помощью логических операторов и логических операций. Логические операции предназначены для выполнения поэлементных логических операций над векторами и матрицами одинаковых размеров. К логическим операциям относятся логическое И (&), логическое ИЛИ (|), логическое НЕ (~).

Вместо логических операций можно пользоваться логическими операторами (функциями) *and*, *or*, *not* соответственно. Логические операторы определены над матрицами одинаковой размерности и выполняются над каждым из элементов.

В логических выражениях используются логические операции и операции отношения.

### Приоритет операций в ML

Приоритет операций определяет порядок действий в выражении. Его можно изменять с помощью круглых скобок. Далее представлены операции в порядке убывания приоритета:

- 1) логическая операция НЕ (~);
- 2) транспонирование(.' , '), возведение в степень(.^, ^);
- 3) унарный плюс (+), унарный минус (-);
- 4) умножение и деление(.\* , ./, .\, \*, /, \);
- 5) сложение и вычитание (+, -);
- 6) операции отношения (<, >, <=, >=, ==, ~=);



- 7) логическая операция И(&);
- 8) логическая операция ИЛИ(|).

## ЭЛЕМЕНТАРНЫЕ ФУНКЦИИ

В ML существует большое количество элементарных математических функций для выполнения действий с числами: тригонометрические, степенные, логарифмические, экспоненциальные и функции округления. Каждая функция обладает именем и списком аргументов, которые задаются в круглых скобках и, если их несколько, перечисляются через запятую.

Существуют встроенные тригонометрические и гиперболические функции:  $\sin(x)$ ,  $\cos(x)$ ,  $\tan(x)$ ,  $\cot(x)$ ,  $\text{asin}(x)$ ,  $\text{acos}(x)$ ,  $\text{atan}(x)$ ,  $\text{acot}(x)$ ,  $\sinh(x)$  и т.д. Аргументы этих функций могут задаваться в радианах и градусах (в следующих версиях, начиная с версии 7.0). У функций в градусной мере после названия добавляется буква *d*.  $\sin(x)$  – аргумент в радианах, а  $\text{sind}(x)$  – аргумент в градусах.

Некоторые часто используемые математические функции:

- $\exp(x)$  – экспонента числа  $x$ ;
- $\log(x)$  – натуральный логарифм;
- $\log_{10}(x)$  – десятичный логарифм;
- $\text{sqrt}(x)$  – квадратный корень;
- $\text{abs}(x)$  – абсолютное значение  $x$ ;
- $\text{real}(z)$  – вещественная часть комплексного числа;
- $\text{imag}(z)$  – мнимой часть комплексного числа;
- $\text{mod}(x, y)$  – остаток от целочисленного деления;
- $\text{round}(x)$  – округление до ближайшего целого.

Для работы с датами можно воспользоваться функциями:

- *calendar* – выводит календарь на текущий месяц;
- *date* – выводит текущую дату.

Все элементарные функции, приведенные выше, можно применять к векторам и матрицам. Если введен вектор  $X$ , то, задав функцию  $Y=\sin(X)$ , получим вектор, элементами которого будут значения синусов элементов исходного вектора:

```
>> X=[1 2 3 4];
>> Y=sin(X)
Y =
    0.8415    0.9093    0.1411   -0.7568
```

Информацию о любой функции ML можно получить, выполнив команду: `help <имя функции>`.

## ОСОБЫЕ МАТРИЦЫ

В ML можно не только задавать матрицы своими значениями или формировать их по формулам, но можно с помощью специальных функций получать особые матрицы:

1. Матрица случайных чисел:

`rand(n, m)` – формирует матрицу из  $n$  строк и  $m$  столбцов, заполненную случайными равномерно распределенными числами в интервале от 0 до 1.

`rand(n)` – формирует квадратную матрицу случайных чисел.

```
>>A=rand(3)
```

```
A =
```

```
0.9501 0.4860 0.4565
0.2311 0.8913 0.0185
0.6068 0.7621 0.8214
```

Можно задавать размер матрицы, используя вектор из двух элементов, равных числу строк и столбцов. Например,

```
>> B=rand([3 4])
```

```
B =
```

```
0.9501 0.4860 0.4565 0.4447
0.2311 0.8913 0.0185 0.6154
0.6068 0.7621 0.8214 0.7919
```

Если надо сгенерировать матрицу такого же размера, как и существующая матрица, то необходимо выполнить команду

```
>> rand(size(A))
```

```
ans =
```

```
0.4103 0.3529 0.1389
0.8936 0.8132 0.2028
0.0579 0.0099 0.1987
```

`size(A)` – функция, возвращающая размер матрицы  $A$  в виде вектора.

2. Магический квадрат (матрица, у которой суммы элементов в строках, столбцах и диагоналях одинаковы) – `magic(n,m)`:

```
>> magic(3)
```

```
ans =
```

```
8   1   6
3   5   7
4   9   2
```

3. Единичная матрица – `eye(n,m)`:

```
>> eye(3)
```

```
ans =
```

```
1   0   0
0   1   0
0   0   1
```

4. Матрица из 0 – `zeros(n,m)`:

```
>> zeros(3)
```

```
ans =
```

```
0   0   0
0   0   0
0   0   0
```

5. Матрица из 1 – `ones(n,m)`:

```
>> ones(3)
```

```
ans =
```

```
1   1   1
1   1   1
1   1   1
```

6. Диагональная матрица. Чтобы получить диагональную матрицу, необходимо задать вектор, количество элементов которого определит размерность матрицы. Значения вектора расположатся на главной диагонали:

```
>> V=[1 2 3 4 5];
```

```
>> diag(V)
```

```
ans =
```

```
1   0   0   0   0
0   2   0   0   0
0   0   3   0   0
0   0   0   4   0
0   0   0   0   5
```

## 7. Выделение диагонали из матрицы:

```
>> diag(A)
ans =
    0.9501
    0.8913
    0.8214
```

Результат – вектор-столбец, состоящий из элементов, расположенных на главной диагонали. Иными словами, при использовании этой функции, если параметром является матрица, результатом будет вектор, а если параметр – вектор, результат – матрица.

Для увеличения быстродействия работы программы бывает полезно заранее задать размерность матрицы, для этого создать матрицу с нулевыми элементами – `zeros(n,m)`, а затем заполнять ее значениями.

## ОПЕРАЦИИ С ВЕКТОРАМИ И МАТРИЦАМИ

В ML операции с векторами и матрицами подразделяются на два типа: *поэлементные преобразования и матричные (векторные) операции*, которые соответствуют правилам линейной алгебры.

### Выполнение операций с векторами

В ML можно выполнять поэлементные преобразования векторов с помощью арифметических операций. *Следует помнить, что операции поэлементного преобразования возможны только над векторами одного размера и структуры.* Сложение и вычитание векторов производится всегда поэлементно. Поэлементные операции обозначаются добавлением точки перед знаком операции. Например,

```
>>A=[2 5 7];
>>B=[1 2 3];
>>C=A.*B
C=
     2    10    21
```

Каждый элемент вектора А умножается на один соответствующий ему элемент вектора В, и таким образом получается каждый элемент вектора С.

Операторы поэлементного деления и возведения в степень записываются соответственно как  $(./)$ ,  $(.^)$ .

Векторное исчисление предусматривает следующие операции над векторами: сложение и вычитание векторов одного размера и типа, транспонирование векторов, умножение вектора на вектор (при условии, что они одного размера и один из них является вектор-столбцом, а другой – вектор-строкой или наоборот).

В соответствии с правилами векторного исчисления, если первый вектор – строка, а второй – столбец, то результат умножения – число:

```
>>A=[1 2 3];  
>>B=[1;2;3];  
>>A*B  
ans=  
44
```

Если первый вектор – столбец, а второй – строка, то результат умножения – квадратная матрица:

```
>>B*A  
ans=  
2    6    10  
4    12   20  
6    18   30
```

Для обозначения операции транспонирования векторов применяется апостроф:

```
>>A=[1 2 3];  
>>B=A'  
B=  
1  
2  
3
```

При транспонировании вектор-строка преобразуется в столбец и наоборот.

## Выполнение операций с матрицами

Операции с матрицами, как и с векторами, могут выполняться поэлементно, а могут – по правилам матричной алгебры.

**Транспонирование матрицы.** Операция обозначается символом апостроф ( ' ):

```
>> A=[1 2 3;4 5 6;7 8 9]
```

```
A =
```

```
1  2  3
4  5  6
7  8  9
```

```
>> B=A'
```

```
B =
```

```
1  4  7
2  5  8
3  6  9
```

**Сложение и вычитание матриц.** Размер матриц должен быть одинаковым. При выполнении этих операций производится поэлементное сложение и вычитание:

```
>> A=[1 2 3;4 5 6;7 8 9]
```

```
A =
```

```
1  2  3
4  5  6
7  8  9
```

```
>> B=[1 4 7;2 5 8;3 6 0]
```

```
B =
```

```
1  4  7
2  5  8
3  6  0
```

```
>> C=A+B
```

```
C =
```

```
2  6  10
6  10  14
10  14  9
```

```
>> D=A-B
```

```
D =
```

```
0  -2  -4
2   0  -2
4   2   9
```

**Умножение матриц.** При умножении  $A*B$  должно выполняться условие: число столбцов матрицы  $A$  равно числу строк матрицы  $B$ :

$$A(n, m) * B(m, k) \rightarrow C(n, k)$$

Элементы результирующей матрицы вычисляются по правилу: каждый элемент строки матрицы  $A$  умножается на соответствующий элемент столбца матрицы  $B$ , затем произведения складываются и получается один элемент матрицы  $C$ :

```
>> A=[1 2 3;4 5 6;7 8 9]
```

```
A =
```

```
1  2  3
4  5  6
7  8  9
```

```
>> B=[1 4 7;2 5 8;3 6 0]
```

```
B =
```

```
1  4  7
2  5  8
3  6  0
```

```
>> C=A*B
```

```
C =
```

```
14  32  23
32  77  68
50 122 113
```

Матрицу можно умножать на число. В результате получим матрицу, в которой каждый элемент получается умножением элемента исходной матрицы на это число:

```
>> A=[1 2 3;4 5 6;7 8 9];
```

```
>> C = A * 2
```

```
C =
```

```
2  4  6
8 10 12
14 16 18
```

Если надо, чтобы действия производились поэлементно, то перед знаком операции ставится точка.

Создадим две матрицы: одну единичную, вторую  $A=[1 2 3; 4 5 6; 7 8 9]$ :

```
>> A=[1 2 3;4 5 6;7 8 9]
```

```
A =
```

```
1 2 3
4 5 6
7 8 9
```

```
>> E=eye(3)
```

```
E =
```

```
1 0 0
0 1 0
0 0 1
```

```
>> B=A*E
```

```
B =
```

```
1 2 3
4 5 6
7 8 9
```

Матрица  $B$  получена в результате матричного умножения матриц  $A$  и  $E$ :

```
>> C=A.*E
```

```
C =
```

```
1 0 0
0 5 0
0 0 9
```

Матрица  $C$  получена в результате поэлементного умножения матриц  $A$  и  $E$ .

Пусть заданы две матрицы  $C$  и  $D$  одинакового размера:

```
>> C=[1 2 3; 4 5 6; 7 8 9];
```

```
>> D=[2 3 4; 1 2 3; 4 5 6];
```

```
>> Y=C.*D
```

```
Y =
```

```
2 6 12
4 10 18
28 40 54
```

**Деление (правое и левое).** В ML имеются две разновидности операции деления матриц: правое (/) и левое (\). В случае с числами  $2/3$  в ML трактуется как результат деления 2 на 3;

$2 \setminus 3$  соответствует результату деления 3 на 2.



С векторами и матрицами – иначе. Пусть  $A$  – матрица, а  $X$  – вектор.  $A * X = B$  и  $X * A = B$  – разные уравнения.

Для решения уравнения  $X * A = B$  используется обычное деление:

$$X = B / A = B * A^{-1}$$

Для решения уравнения  $A * X = B$  используется обратное деление:

$$X = A \setminus B = A^{-1} * B$$

Операция обратного деления используется для решения системы линейных уравнений. Например:

$$2x_1 + 3x_2 = 11$$

$$3x_1 - 4x_2 = 8$$

$A$  – матрица коэффициентов левой части.

$B$  – вектор правых частей.

Решается уравнение вида  $A * X = B$ :

```
>> A=[2 3;3 -4];
```

```
>> B=[11 8];
```

```
>> X=A\B'
```

```
X =
```

```
4.0000
```

```
1.0000
```

Для проверки можно выполнить умножение  $A * X$ :

```
>> A*X
```

```
ans =
```

```
11.0000
```

```
8.0000
```

В результате получили вектор правых частей, что доказывает правильность найденного решения.

### Специальные функции для матриц

В ML существует множество специальных функций линейной алгебры. Приведем некоторые из них:

$\det(A)$  – вычисляет определитель (детерминант) матрицы.

$\text{inv}(A)$  – вычисляет инверсную (обратную) матрицу.

*Обратной* для квадратной невырожденной (когда детерминант не равен нулю) матрицы  $A$  называется матрица  $A^{-1}$ , которая при умножении на матрицу  $A$  справа и слева даёт единичную матрицу.  $B$  называется обратной  $A$ , если выполняется  $AB=BA=E$  (единичная матрица).

$eig(A)$  – определение собственных чисел (характеристических чисел);

$norm(A, 1)$  – норма матрицы – наибольшая сумма модулей элементов столбцов;

$norm(A, inf)$  – наибольшая сумма модулей элементов строк;

$trace(A)$  – след матрицы (сумма элементов главной диагонали).

## ДЕЙСТВИЯ С ЭЛЕМЕНТАМИ МАТРИЦЫ

Обращение к элементу осуществляется заданием имени, за которым в круглых скобках через запятую указываются индексы – номера строки и столбца. Если указать  $A(2,1)$ , то выберется элемент второй строки первого столбца.

Можно выделить часть матрицы. Это делается с помощью символа двоеточие (:).

Пусть задана матрица:

```
>> A=[1 2 3;4 5 6;7 8 9]
```

A =

```
1  2  3
4  5  6
7  8  9
```

Если двоеточие стоит вместо номера строки или столбца, то соответственно выделяются столбец или строка.

Получим 3-й столбец матрицы A:

```
>> B=A(:,3)
```

B =

```
3
6
9
```

Получим 3-ю строку матрицы A:

```
>> X=A(3,:)
```

X =

```
7  8  9
```

Получим вектор-столбец из всех элементов матрицы:

```
>> C=A(:)
```

```
C =
```

```
1
```

```
4
```

```
7
```

```
2
```

```
5
```

```
8
```

```
3
```

```
6
```

```
9
```

Индексом в ML *может быть не только число, но и вектор*. Этот вектор-индекс удобно задавать с помощью двоеточия.

Для доступа к элементам последней строки или столбца можно использовать в качестве индекса *end*.

Получим матрицу *C*, состоящую из элементов матрицы *A*, начиная со второго столбца:

```
>> C=A(:, 2:end)
```

```
C =
```

```
2  3
```

```
5  6
```

```
8  9
```

Поменяем местами первую и последнюю строки матрицы *A*:

```
>> D=A(1,:);
```

```
>> A(1,:)=A(end,:);
```

```
>> A(end,:)=D;
```

```
>> A
```

```
A =
```

```
7  8  9
```

```
4  5  6
```

```
1  2  3
```

Получим вектор *C*, состоящий из элементов первых трех элементов 2-го столбца матрицы *A*.

Пусть задана матрица:

```
>> A=[1 2 3; 4 5 6; 7 8 9; 1 0 1]
```

```
A =
```

```
1  2  3
```

4	5	6
7	8	9
1	0	1

Запишем:

```
>> C=A(1:3, 2)
```

C=

2
5
8

Первый индекс представляет собой вектор из трех элементов [1:3], результат – вектор-столбец C.

Выделим из матрицы A квадратную матрицу C размером 2x2:

```
>> C=A(3:4,2:3)
```

Получим

C =

8	9
0	1

Выделим первые две строки матрицы A:

```
>>C=A(1:2,:)
```

C =

1	2	3
4	5	6

Можно заменить один фрагмент матрицы другим.

Пусть имеем матрицу A = [1 2 3;4 5 6; 7 8 9] и зададим матрицу C:

```
>> C=[20 30;10 15]
```

C =

20	30
10	15

Заменим правый верхний угол матрицы A матрицей C. Для этого запишем:

```
>> A(1:2,1:2)=C
```

Измененная матрица A

```
A =
    20    30     3
    10    15     6
     7     8     9
```

Поменяем местами 1-ю и 3-ю строки матрицы A:

Используем для этого возможность задания индекса вектором.

```
>>A=[1 2 3;4 5 6;7 8 9];
```

Для этого запишем:

```
>> A([1 3],:)=A([3 1],:)
```

Полученная матрица:

```
A =
     7     8     9
     4     5     6
     1     2     3
```

Обращение A([1 3], :) означает первую и третью строки матрицы; A([3 1], :) – аналогично третью и первую строки.

Индексация двоеточием упрощает формирование матриц по определенному закону. Пусть необходимо сформировать матрицу размером 5×5, в которой элементы первой и последней строк и первого и последнего столбцов равны единице. Остальные элементы матрицы равны нулю.

Сначала создадим матрицу из нулей размером 5×5. Затем заполним первую и последнюю строки и первый и последний столбцы единицами:

```
>> Z(1:5,1:5)=0;
>>Z(1, :)=1;
>>Z(end, :)=1;
>>Z(:, 1)=1;
>>Z(:, end)=1
```

В результате получим матрицу Z:

```
Z =
     1     1     1     1     1
     1     0     0     0     1
     1     0     0     0     1
     1     0     0     0     1
     1     1     1     1     1
```

или лучше:

```
>> Z([1 end], :)=1;  
>> Z(:, [1 end])=1
```

Можно выполнить объединение матриц:

➤ по 

X	Y
---	---

 горизонтали

```
>> X=[1 2;3 4];  
>> Y=[5 6;7 8];  
>> Z=[X,Y]
```

Z =  
1 2 5 6  
3 4 7 8

➤ по 

X
Y

 вертикали

```
>> Z=[X;Y]
```

Z =  
1 2  
3 4  
5 6  
7 8

Размеры матриц должны быть согласованы.

Добавим строку к матрице A:

```
>> A=[1 2 3;4 5 6;7 8 9];
```

Зададим вектор B:

```
>> B=[5 5 5];
```

Запишем:

```
>> A=[A;B]
```

Получим новую матрицу размером 4×3:

A =  
1 2 3  
4 5 6  
7 8 9  
5 5 5

Удалим из матрицы 2-ю и 3-ю строки:

```
>> A(2:3, :)=[]
```

A =  
1 2 3  
5 5 5

Удалим из матрицы 2 столбец

```
>> A(:, 2)=[]
```

A =

```
1  3
5  5
```

Можно удалить элемент из матрицы.

A(3)=[] – удаление 3-го элемента матрицы.

```
>> A=[1 2 3;4 5 6;7 8 9]
```

A =

```
1  2  3
4  5  6
7  8  9
```

```
>> A(3)=[]
```

Результат – вектор A, состоящий из элементов матрицы в порядке следования по столбцам без элемента A(3,1):

A =

```
1  4  2  5  8  3  6  9
```

Сделаем матрицу нулевой размерности

```
>>A=[];
```

Удаление матрицы из памяти

```
>>clear A
```

## ФУНКЦИИ, ИСПОЛЬЗУЕМЫЕ ДЛЯ РАБОТЫ С ВЕКТОРАМИ И МАТРИЦАМИ

Ранее мы познакомились с функциями для создания векторов и матриц. В пакете Matlab имеются функции, предоставляющие возможность построчной или постолбцовой обработки элементов матрицы. Это функции, позволяющие вычислить сумму либо произведение элементов, отсортировать данные, найти максимальное и минимальные значения элементов вектора или матрицы.

Пусть имеем **вектор X**

```
>> X=[9 2 3 4 15 6 1 7];
```

Рассмотрим некоторые функции.

***Сумма элементов вектора:***

```
>> sum(X)
ans =
    47
```

***Произведение элементов вектора:***

```
>> prod(X)
ans =
  136080
```

***Максимальное значение вектора:***

```
>> max(X)
ans =
    15
```

***Максимальное значение вектора и его номер:***

```
>> [m,k]=max(X)
m =
    15
k =
     5
```

***Минимальное значение вектора:***

```
>> min(X)
ans =
     1
```

***Минимальное значение вектора и его номер:***

```
>> [m,k]=min(X)
m =
     1
k =
     7
```

***Среднее арифметическое элементов вектора:***

```
>> mean(X)
ans =
   5.8750
```



### ***Сортировка элементов вектора по возрастанию:***

```
>> sort(X)
ans =
    1    2    3    4    6    7    9   15
```

Для сортировки элементов вектора по убыванию можно воспользоваться той же функцией:

```
>> X = -sort(-X)
X =
   15    9    7    6    4    3    2    1
```

### ***Разворот вектора на 90°***

```
>> rot90(X)
```

Эта функция разворачивает вектор против часовой стрелки. В результате получаем вектор-столбец:

```
ans =
   15
    9
    7
    6
    4
    3
    2
    1
```

Все эти функции можно применить к вектору, независимо от того, столбец это или строка. Для матриц используются те же функции, которые выполняют те же действия в каждом столбце матрицы.

Пусть имеем матрицу  $A$

```
>> A=[1 2 3;4 5 6;7 8 9];
```

Рассмотрим некоторые функции.

### ***Сумма элементов в столбцах матрицы:***

```
>> sum(A)
ans =
   12   15   18
```

или можно записать:

```
>> sum(A,1)
ans =
   12   15   18
```

Для получения суммы элементов в строках матрицы введем команду

```
>> sum(A,2)
ans =
     6
    15
    24
```

Для получения суммы всех элементов матрицы введем команду

```
>>sum(sum(A))
ans =
```

```
    45
```

***Произведение элементов матрицы по столбцам:***

```
>> prod(A)
ans =
    28    80   162
```

***Произведение элементов матрицы по строкам:***

```
>> prod(A')
ans =
     6   120   504
```

Произведение чисел от 1 до 4

```
>> prod(1:4)
ans =
    24
```

В функциях для вычисления суммы и произведения (*sum* и *prod*) элементов матрицы результатом будет вектор-строка, число элементов которой равно числу столбцов матрицы.

Есть функции, которые выдают наибольшее и наименьшее значения в столбцах матрицы. Результат – вектор-строка.

***Максимальное значение в каждом столбце:***

```
>> max(A)
ans =
     7     8     9
```

**Максимальное значение в каждой строке:**

```
>> max(A')
ans =
    3    6    9
```

**Наименьшее значение:**

```
>> min(A)
ans =
    1    2    3
>> min(A')
ans =
    1    4    7
```

Можно не только найти минимальное и максимальное значения, но и позиции этих элементов:

```
>> [m,k]=min(A)
```

Результат – два вектора, один – из минимального или максимального значений в столбцах матрицы, а второй – из позиций этих элементов в соответствующем столбце:

```
m =
    1    2    3
k =
    1    1    1
>> [m,k]=max(A)
m =
    7    8    9
k =
    3    3    3
```

**Среднее арифметическое в столбцах.** Результат – вектор-строка из средних арифметических в каждом столбце:

```
>> mean(A)
ans =
    4    5    6
>> mean(A')
ans =
    2    5    8
```

**Сортировка.** Сортировка элементов каждого столбца по возрастанию:

```
>>sort(A)
```

Сортировка элементов каждой строки по возрастанию:

```
>>sort(A,2)
```

### ***Разворот матрицы на 90°***

Эта функция разворачивает матрицу на 90° против часовой стрелки:

```
>> rot90(A)
```

```
ans =
```

```
3  6  9
2  5  8
1  4  7
```

***”Зеркальное” отображение матрицы относительно вертикальной оси:***

```
>> fliplr(A)
```

```
ans =
```

```
3  2  1
6  5  4
9  8  7
```

***”Зеркальное” отображение матрицы относительно горизонтальной оси:***

```
>> flipud(A)
```

```
ans =
```

```
7  8  9
4  5  6
1  2  3
```

## **ДЕЙСТВИЯ С ПОЛИНОМАМИ (МНОГОЧЛЕНАМИ)**

В МЛ предусмотрены функции для работы с полиномами. С помощью этих функций можно вычислить значение полинома, найти его корни, выполнить операции умножения и деления полиномов и т.д.

Полином (многочлен) – это выражение вида

$$P(x)=a_1x^n+a_2x^{n-1}+.....+a_nx+a_{n+1}$$

В системе Matlab полином задаётся и хранится в виде вектора, элементами которого являются коэффициенты полинома:

$$P=[a_1 \ a_2 \ \dots a_n \ a_{n+1}]$$

Число элементов вектора должно быть на единицу больше степени полинома. Если в полиноме отсутствует слагаемое, соответствующее какой-либо степени  $x$ , то в этом случае в векторе в качестве значения коэффициента записывается ноль.

Например, пусть задан полином  $2x^3+x^2-3x+5$ . Вектор коэффициентов полинома будет

$$p=[2 \ 1 \ -3 \ 5]$$

Для вычисления значения полинома при некотором значении аргумента предназначена функция *polyval(p,x)*, где  $p$  – вектор коэффициентов полинома;  $x$  – значение аргумента, при котором надо посчитать значение полинома.

```
>> P=[2 1 -3 5];  
>> Y=polyval(P,1)  
Y =  
    5
```

В результате будет вычислено значение полинома  $2x^3+x^2-3x+5$  при  $x=1$ .

В качестве аргумента  $x$  может быть указан вектор или матрица. В результате получится вектор или матрица того же размера, что и аргумент:

```
>> P=[2 1 -3 5];  
>> X=1:5;  
>> Y=polyval(P,X)  
Y =  
    5    19    59   137   265
```

Элементы вектора  $Y$  – значения полинома, вычисленные для каждого элемента вектора  $X$ .

Вычислить корни полинома можно с помощью функции *roots*. Число корней определяется степенью полинома:

```
>> P=[2 1 -3 5];  
>> roots(P)  
ans =  
-1.9388
```

$$0.7194 + 0.8786i$$

$$0.7194 - 0.8786i$$

В данном случае найдены три корня, из которых один вещественный и два комплексных.

Чтобы вычислить производную от полинома, следует использовать функцию *polyder*. Результатом этой функции является вектор, элементы которого представляют собой коэффициенты полинома-производной от исходного полинома:

```
>> P=[2 1 -3 5];
>> polyder(P)
ans =
    6    2   -3
```

Для выполнения умножения и деления полиномов предназначены функции *conv* и *deconv*:

$$Z=conv(P1,P2),$$

где  $P1, P2$  – полиномы, заданные векторами;  $Z$  – результирующий вектор коэффициентов полинома, полученного в результате перемножения полиномов, заданных векторами  $P1, P2$ .

$$[R1, R2]=deconv(P1,P2)$$

Результат работы этой функции – два вектора  $R1, R2$ , где  $R1, R2$  – векторы коэффициентов полинома-частного и полинома-остатка, полученного в результате деления полиномов, заданных векторами  $P1, P2$ .

## ГРАФИКА

Система ML обладает мощными графическими возможностями. Вывод графической информации ML осуществляет в отдельное окно, которое создается автоматически, когда используется какая-либо графическая функция. Для оформления и редактирования графиков предусмотрены специальные команды.

### Построение простейших графиков

Можно графически отобразить значение вектора  $Y$ .

Это делается командой *Plot(Y)*. По оси  $X$  в этом случае откладываются только номера элементов. Например,

```
>> Y=[1 3 7 12 86];  
>> plot(Y)
```

В результате будет получен график, изображенный на рис. 5

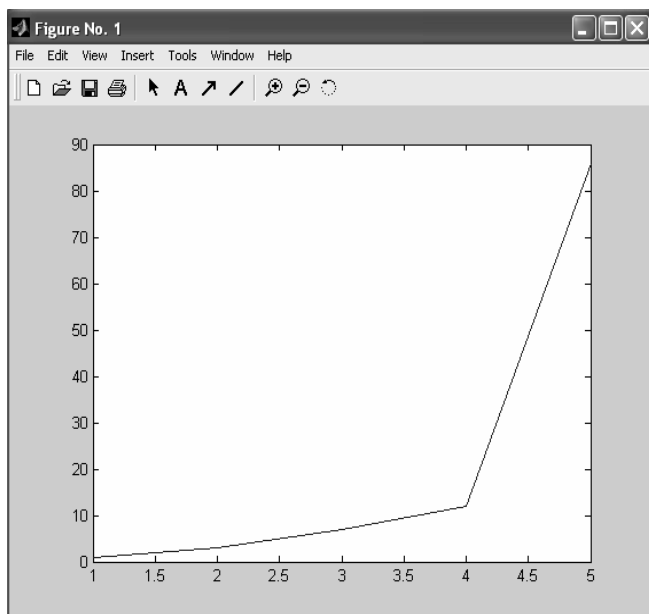


Рис. 5

Чтобы построить график какой-либо функции на определенном интервале  $x$  с отметками по оси  $X$ , надо получить вектор из значений аргумента, потом получить вектор значений функции, а затем графически отобразить его командой  $Plot(X, Y)$ .

Будем строить график функции  $\sin(x)$  на участке от 0 до  $2\pi$ .

Получим вектор значений аргумента:

```
>> X = linspace (0, 2*pi, 100);  
>> Y = sin(X);
```

Получим вектор из 100 элементов. Элементы вектора  $Y$  представляют значения синусов элементов вектора  $X$ .

```
>> Plot(X, Y)
```

Можно добавить к графику (рис. 6) сетку. Для этого надо воспользоваться командой *grid on*.

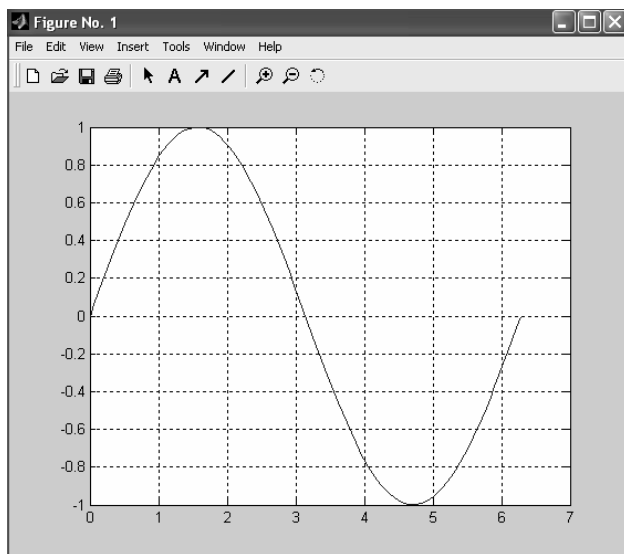


Рис. 6

Команда *grid on* включает сетку, а команда *grid off* отключает сетку.

Можно добавить общее название графика командой *title(<заголовок>)*

Названия осей задаются командами:

*xlabel(<название оси x>); ylabel(<название оси y>)*

Команда *text(<координата x>, <координата y>, <текст>)* выводит текст, начиная с позиции, заданной координатами *X* и *Y*.

*Заголовок, название оси x, название оси y – параметры строкового типа (записываются в апострофах).*

Например,

```
>> text(-1, 0.5, 'hhhhh')
```

Для указания цвета, типа линий и маркера в команде *Plot* надо дописать параметр:

*Plot(x, y, '<тип маркера, обозначение цвета и тип линий>')*

Этот дополнительный параметр может принимать следующие значения:



Тип линии	Цвет линии	Тип маркера
Сплошная(-)	Красный (r)	Точка (.)
Штрих(--)	Желтый (y)	Плюс (+)
Пунктир(:)	Зеленый (g)	Кружок (o)
Штрих-пунктир(-.)	Синий (b)	Крестик (x)
	Голубой (c)	Звездочка (*)
	Розовый (m)	Квадрат (s)
	Черный (k)	Ромб (d)
	Белый (n)	Треугольник с верш. вниз (v)
		Треугольник с верш. вверх (^)

Например,

```
plot(x, y, '* r -')
```

Порядок указания атрибутов приведен выше. Если указан только цвет, то график строится сплошной линией заданного цвета. Если указан символ, определяющий тип маркера, но не указан символ для типа линии, то на графике отображаются только маркеры, не соединённые линиями. По умолчанию цвет построения синий.

Командой *plot* можно на графике отметить какую-либо точку, указав координаты, тип и цвет маркера, если *a* и *b* – координаты точки:

```
plot(a, b, '* g')
```

В одних осях можно построить любое количество графиков. В команде *plot* для этого можно последовательно указать пары векторов и атрибуты для желаемого количества графиков. В этом случае с помощью команды *legend* на графике можно разместить легенду с информацией о линиях. Аргументы команды текстовые. Их число и порядок соответствуют линиям графика. В следующем примере в одних осях строятся три графика (рис. 7):

```
>> x=linspace(0,2*pi,100);
>> y=sin(x);
>> plot(x,y,'*g',x,cos(x),'r--',x,2*y,'+b')
>> legend('sin(x)','cos(x)','2sin(x)')
```

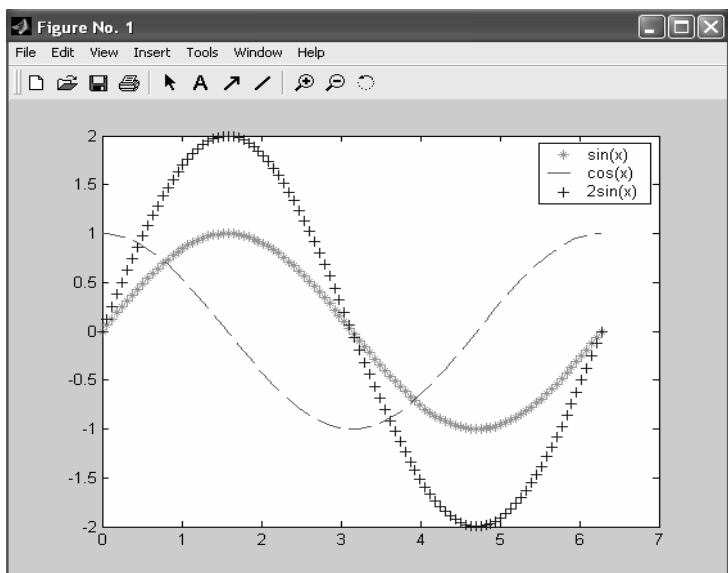


Рис. 7

Для последовательного построения графиков в одних осях надо установить режим сохранения текущего графика – *hold on*. Команда *hold off* снимает сохранение.

```
>> x=linspace(0,2*pi,100);
>> z=cos(x);
>> y=sin(x);
>> plot(x,y)
>> hold on;
>> plot(x,z)
>> hold off;
```

График, задаваемый значениями вектора  $Z$ , будет наложен на график, отображающий значения вектора  $Y$  (рис. 8). Если не использовать команду *hold on*, то при выводе каждый следующий график будет выведен в том же графическом окне вместо предыдущего. Чтобы новый график не накладывался на предыдущий, необходимо набрать команду *hold off*.

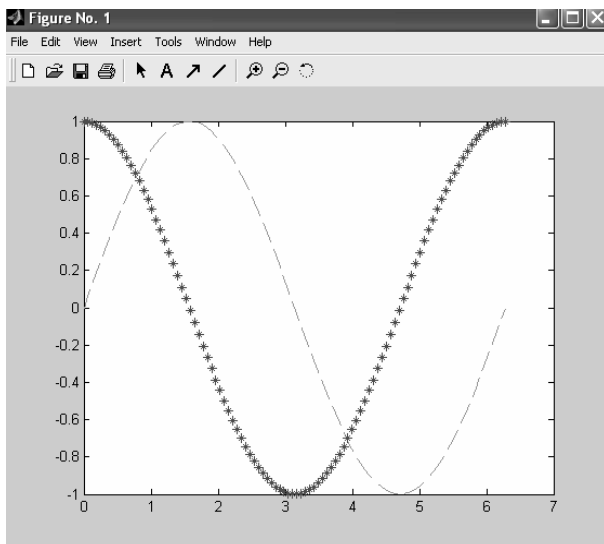


Рис. 8

Все графические окна автоматически нумеруются, начиная с №1. По умолчанию все построения сначала проводятся в этом окне. Имеется возможность при необходимости открыть новое окно, в котором будут производиться графические построения. Для этого надо набрать команду *figure*. При этом не произойдет наложение на прежнее графическое изображение, а откроется новое окно со следующим номером.

Можно построить график с использованием функции *ezplot* (<символьное выражение>, <интервал>).

Функция работает с символьным выражением, задающим формулу функции, которая заключается в апострофы.

Пусть требуется построить функцию  $x^2+x+1$  на интервале от -2 до 2, тогда

```
>> ezplot('x^2+x+1', [-2 2]) (рис. 9).
```

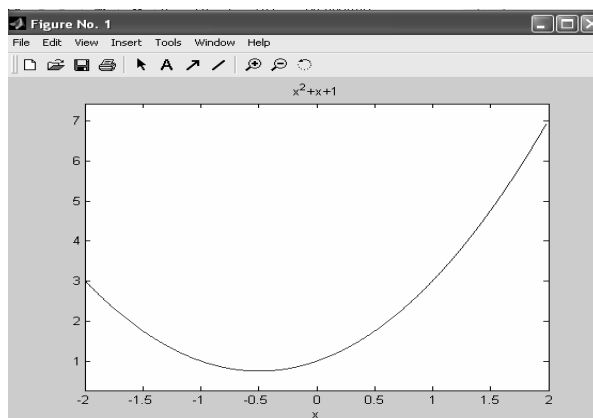


Рис. 9

### Вывод нескольких графиков в одном окне

Графическое окно можно разделить на несколько частей и в каждой из них вывести графики различных функций. Эта возможность реализуется командой:

*subplot(m, n, p),*

где *m*, *n* указывают, на сколько частей делится окно по вертикали (*m*) и горизонтали (*n*), а *p* – номер подокна, в котором будет отображаться график (нумерация слева направо и сверху вниз). Р=1, 2, 3, 4 при *m*, *n* = 2

1	2
3	4

Чтобы построить график во весь экран, необходимо задать команду: *Subplot(1, 1, 1)*.

Разделим экран на четыре части. В окнах 1 и 2 выведем графики функций  $\cos(x)$  и  $\sin(x)$  соответственно.

```
>>x = -2: 0.01: 2;
>>f1 = sin(x);
>>subplot(2, 2, 1);
>>plot(x, f1);
>>title('sin(x) ');
>>y2 = cos(x);
>>subplot(2, 2, 2);
>>plot(x, y2);
>>title('cos(x) ');
```

Результат изображен на рис. 10.

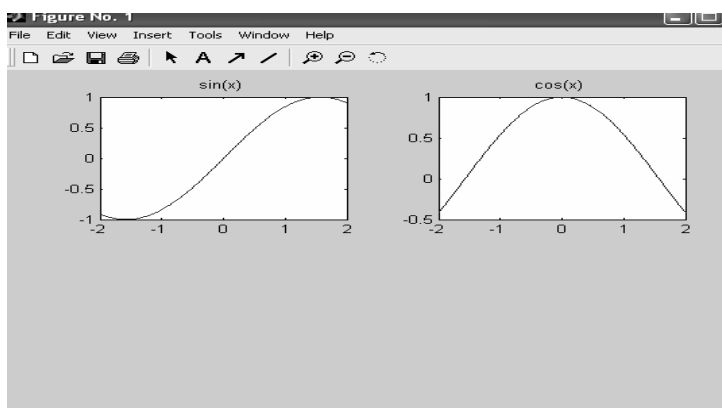


Рис. 10

Разделим экран на две части по вертикали (рис. 11):

```
>>subplot(2,1,1)
>>plot(x,y2)
>>title('cos(x) ');
>>subplot(2,1,2)
>>plot(x,f1)
>>title('sin(x) ');
```

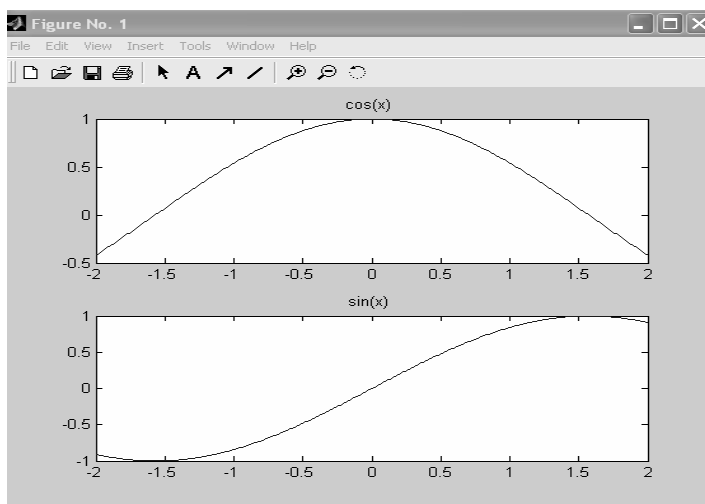


Рис. 11

Можно управлять осями, которые располагаются на экране. Их можно убрать с экрана:

*axis off* – убрать с экрана;

*axis on* – вернуть на экран.

Можно установить свой масштаб по осям:

*axis*([ $x_{\min}$   $x_{\max}$   $y_{\min}$   $y_{\max}$ ]),

где  $x_{\min}$ ,  $x_{\max}$  – минимальное и максимальное значения по оси  $x$ ;  $y_{\min}$ ,  $y_{\max}$  – минимальное и максимальное значения по оси  $y$ .

Если график трехмерный, то команда будет иметь вид:

*axis*([ $x_{\min}$   $x_{\max}$   $y_{\min}$   $y_{\max}$   $z_{\min}$   $z_{\max}$ ]).

*axis* ('i j') – центр координат будет расположен в левом верхнем углу экрана



*axis* ('xy') – обычное расположение осей



*axis* ('square') – график будет занимать область с одинаковыми диапазонами значений по осям  $x$  и  $y$ .

*axis* ('equal') – одинаковый масштаб по обеим осям.

*axis* ('auto') – устанавливает масштаб по умолчанию.

## Диаграммы

Диаграммы позволяют представить векторные и матричные данные в более наглядном виде. Для построения столбиковой диаграммы используется функция *bar*(<вектор>).

Пусть имеем вектор  $Y$ . Построим столбиковую диаграмму (рис. 12):

```
>>Y=[2 5 3 7];
```

```
>> bar(Y)
```

Каждый элемент представляется в виде столбца, высота которого пропорциональна его значению. На оси  $x$  – номера элементов.

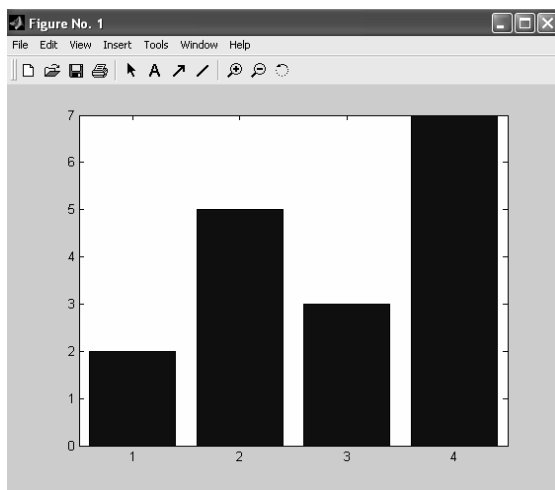


Рис. 12

Зададим два вектора:  $X$  и  $Y$ . Они должны быть одной длины:

```
>>Y=[2 5 3 7];  
>>X=[8 10 4 1];  
>> bar(X,Y)
```

В этом случае по оси  $y$  откладываются значения вектора  $y$ , а по оси  $x$  – не номера элементов, а значения вектора  $x$  (рис. 13).

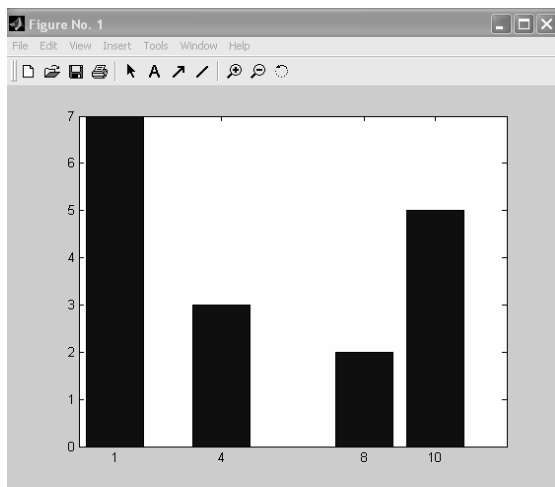


Рис. 13

В качестве аргумента функции *bar* можно задать матрицу (рис. 14):

```
>> A=[1 2 3; 4 5 6; 7 8 9];  
>> bar(A)
```

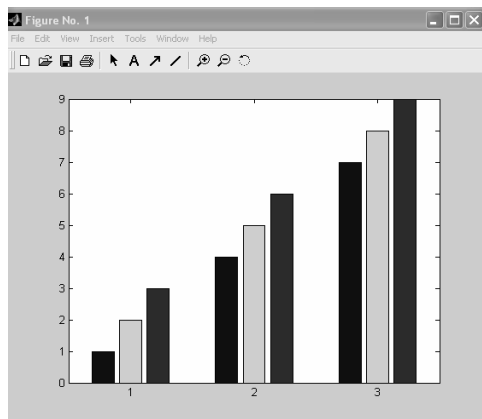


Рис. 14

Чтобы отобразить функцию в виде диаграммы, нужно получить два вектора (вектор значения аргумента и вектор значения функции) и применить к ним команду *bar* (рис. 15):

```
>> X=0: 0.1: 5;  
>> Y=exp(x);  
>> bar(X, Y)
```

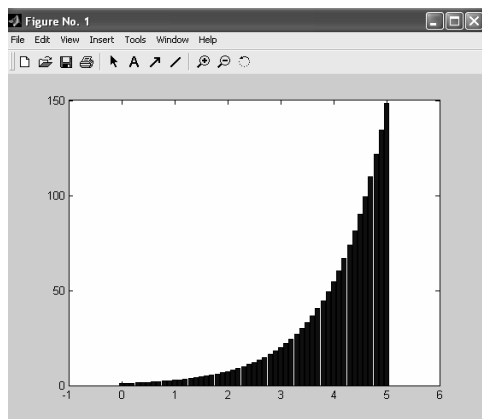


Рис. 15



Для получения объемной диаграммы (рис. 16) используется команда *bar3*:

```
>>A=[1 2 3; 4 5 6; 7 8 9];  
>> bar3(A)
```

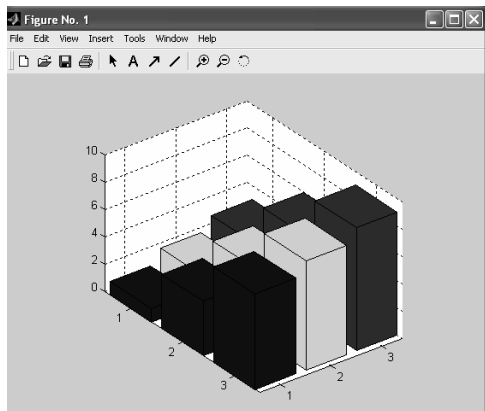


Рис. 16

При использовании функции *barh(A)* получим диаграмму с горизонтально расположенными столбцами.

Использование круговой диаграммы можно показать на примере определения доли (в процентах) каждого элемента вектора от общей суммы элементов.

Круговые диаграммы строятся с помощью функции *pie(<вектор>)*:

```
>>X=[3 8 10]  
>>pie(X)
```

Построение секторов ведется, начиная от вертикальной оси, против часовой стрелки (рис. 17).

Чтобы выделить какой-либо сектор, необходимо создать дополнительно вектор из 0 и 1 такого же размера, что и вектор, содержащий данные. Наличие 1 показывает, что надо отделить сектор с таким же номером, что и в этом векторе *Y*. Пусть заданы два вектора: *X* и *Y*.

```
>>X=[16 8 3 1 4];  
>>Y=[0 1 0 0 1];  
>> pie(X, Y)
```

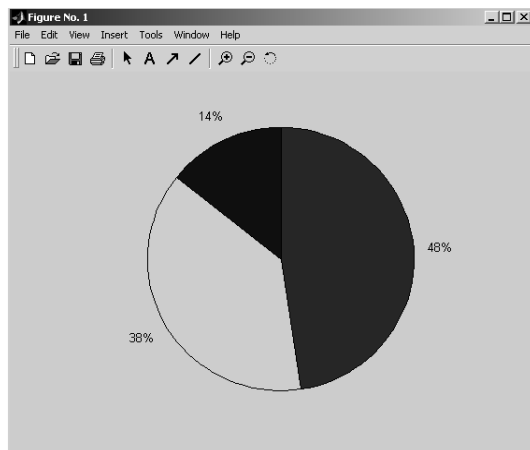


Рис. 17

Единица в векторе  $Y$  показывает, что второй и последний сектора должны быть отделены (рис. 18).

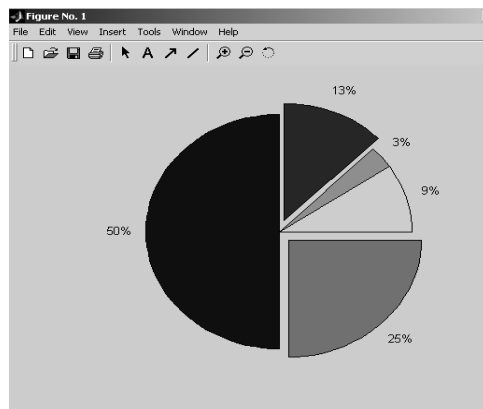


Рис. 18

Можно построить объемную диаграмму, применив функцию `pie3(<вектора>)`:

```
>>pie3(X, Y)
```

Результат изображен на рис. 19.

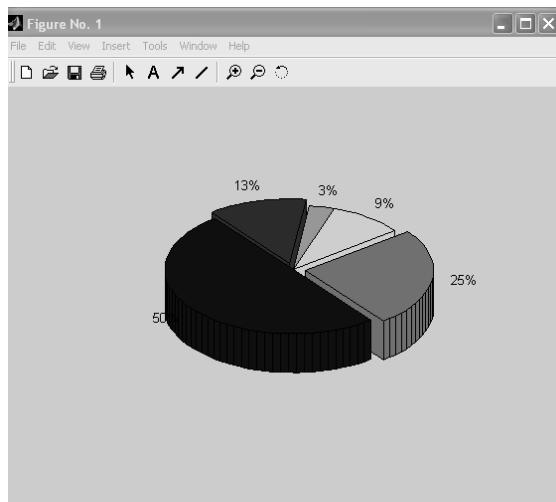


Рис. 19

## Построение графиков в полярных координатах

Полярная система координат состоит из заданной фиксированной точки  $O$ , называемой полюсом, концентрических окружностей с центром в полюсе и лучей, выходящих из точки  $O$ , один из которых  $OX$  – полярная ось.

Расположение любой точки  $M$  в полярных координатах можно задать положительным числом  $\rho = |OM|$ , равным расстоянию от полюса до точки, и числом  $\varphi$ , равным величине угла  $XOM$  (полярный угол);  $\rho$  и  $\varphi$  называют полярными координатами точки  $M$ , и точку обозначают  $M(\rho, \varphi)$ . Для формирования графика в полярных координатах используется функция  $polar(\varphi, \rho, s)$ , где  $\varphi$  – массив полярных углов,  $\rho$  – массив значений полярных радиусов точек, образующих графики,  $s$  – строка, состоящая из трех символов, которые определяют цвет линии, тип маркера и тип линии.

Пусть требуется построить график функции  $r(\varphi) = 5\cos(2-7\varphi)$ . При построении в полярных координатах будем изменять аргумент от 0 до  $2\pi$  (рис. 20).

Необходимо сформировать массивы значений полярного угла  $\varphi$  и полярного радиуса  $r$  (аргументом является угол):

```
>> fi=0:0.01:2*pi;
>> r=5*cos(2-7*fi);
>> polar(fi, r);
```

Можно также указать тип и цвет линий, тогда

```
>> fi=0:pi/200:2*pi;
>> r=5*cos(2-7*fi);
>> polar(fi, r, 'r*-');'
```

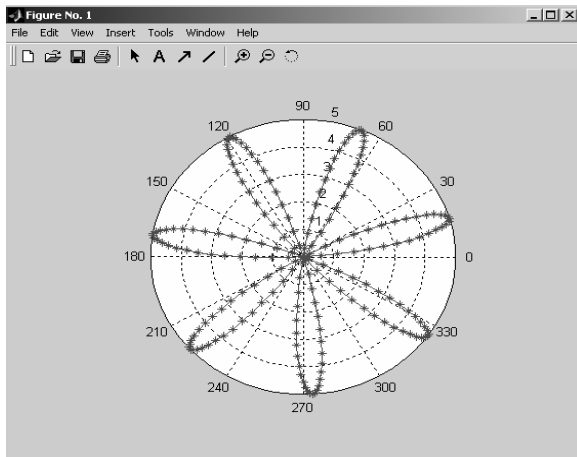


Рис. 20

Для построения траектории движения точки на плоскости (анимация) предусмотрена функция *comet*, а для построения траектории движения точки в трехмерном пространстве – функция *comet3*. Это простой способ создать анимированное изображение.

Изобразим движение точки по траектории, заданной параметрически (рис. 21):

```
>> t=0:0.005:2*pi;
>> x=cos(5*t).*(t+5);
>> y=sin(5*t).*(t+5);
>> comet(x,y)
```

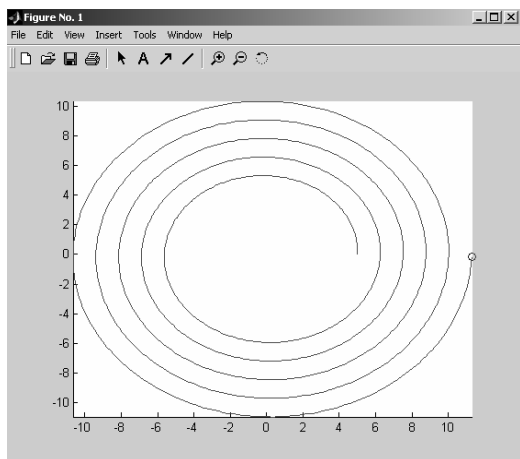


Рис. 21

Построим траекторию движения точки в пространстве (рис. 22):

```
>> t=-10*pi:pi/300:10*pi;
>> x=(sin(t).^3).*cos(t);
>> y=(cos(t).^3).*sin(t);
>> comet3(x,y,t)
```

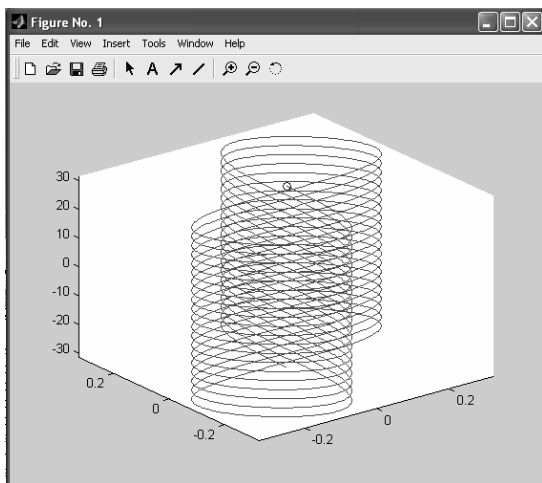


Рис. 22

Для более сложной анимации можно использовать команды *getframe* и *movieview*. Команда *getframe* захватывает активное окно изображения в один кадр фильма, а команда *movieview* воспроизводит результат в отдельном окне. Приведенные ниже команды воспроизводят кадры с вибрирующей струной:

```
>>x=0:0.01:1;  
>>for n=0:50  
>>plot(x,sin(n*pi/5)*sin(pi*x)),axis([0,1,-2,2])  
>>m(n+1)=getframe;  
>>end  
>>movieview(m)
```

В приведенном выше примере для организации цикла используется оператор *for*, работа которого будет описана далее.

### Трехмерная графика

График поверхности (трехмерный график) – это график, положение точки в котором определяется значениями трех координат. Трехмерным аналогом функции *plot* является функция *plot3*, которая позволяет создавать трехмерные линии. Если есть три вектора  $x$ ,  $y$ ,  $z$ , задающих координаты точек в трехмерном пространстве, то при выполнении функции *plot3(x,y,z)*, построится трехмерная линия на плоскости.

Пусть требуется построить график линии, заданной формулами

```
x=2sin(t+pi/2),  
y=2cos(t+pi/2),  
z=t, где t принадлежит диапазону [0, 8pi]  
>> t=0:pi/100:8*pi;  
>> x=2*sin(t+pi/2);  
>> y=2*cos(t+pi/2);  
>> z=t;  
>> plot3(x,y,z);  
>> axis square;  
>> grid on
```

Построилась винтовая линия (спираль) (рис. 23).

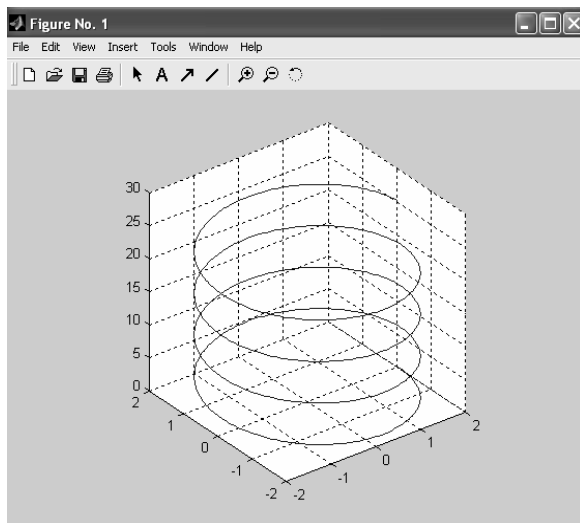


Рис. 23

Функция *plot3*, так же как и *plot*, может иметь дополнительный аргумент, задающий параметры линии (тип линии, цвет и тип маркера). Например, *plot3(x,y,z, 'g\*')*.

Можно также использовать команду *ezplot3*:

```
>> ezplot3('cos(2*pi*t)','sin(2*pi*t)','t',[-2,2])
```

Для вычерчивания каркасных поверхностей в трехмерном пространстве существуют две основные функции: *mesh* и *surf*.

Команда *mesh(Z)* строит прозрачную сетчатую поверхность, а команда *surf(Z)* – затененную, где *Z* – матрица, значения элементов которой определяют соответствующие координаты на графике. Построим прозрачную поверхность для единичной матрицы, задав команды:

```
>> Z=eye(8);
>> mesh(Z)
```

Результат показан на рис. 24.

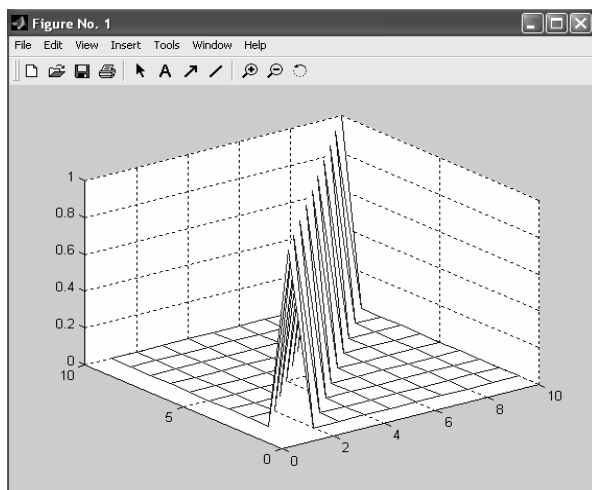


Рис. 24

По главной диагонали расположены “пики” – единицы. Аналогично, задав команду

`>>surf(Z)`

получим затененную поверхность (рис. 25).

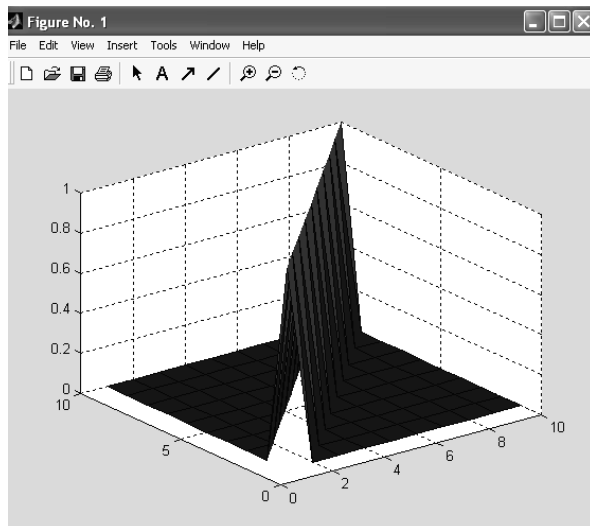


Рис. 25



Для построения в трехмерном пространстве функции от двух переменных  $Z(X, Y)$  необходимо сначала с помощью векторов  $x$  и  $y$  задать прямоугольную сетку, которая будет использоваться как основание для построения трехмерной поверхности. Для этого надо воспользоваться функцией *meshgrid*, которая создает эту сетку из точек в прямоугольной области с заданными интервалами. Диапазоны значений по осям  $x$  и  $y$  задаются векторами  $x$  и  $y$  соответственно. Имена векторов и матриц могут различаться.

Пусть значения по осям  $x$  и  $y$  заданы векторами  $x$  и  $y$ :

```
>> x=[1 2 3];
>> y=[4 5 6];
```

Сформируем матрицы  $X$  и  $Y$  с помощью функции *meshgrid*:

```
>> [X,Y]=meshgrid(x,y)
X =
    1    2    3
    1    2    3
    1    2    3
Y =
    4    4    4
    5    5    5
    6    6    6
```

На основе векторов  $x$  и  $y$  формируются две матрицы, в которые записываются координаты узлов сетки. Матрица  $X$  содержит одинаковые строки, в которых заданы координаты  $X$ . Матрица  $Y$  содержит одинаковые столбцы, в которых заданы координаты  $Y$ . Наложение матриц  $X$  и  $Y$  позволяет получить пары  $(x_i \text{ и } y_j)$ , для которых в дальнейшем вычисляется значение функции  $Z$ . Значения функции в узлах сетки записываются в матрицу  $Z$ , размерность которой равна размерности матриц  $X$  и  $Y$ .

Построим график функции, которая имеет вид  $z = 2 * x \cdot e^{-x^2 - y^2}$  на заданном интервале  $x$  от  $-2$  до  $2$  и от  $y$  от  $-4$  до  $4$  с шагом, равным  $0,1$ .

Зададим два вектора  $x$  и  $y$ , используя запись

```
>> x=-2 : 0.1 : 2;
>> y=-4 : 0.1 : 4;
```

Далее вызывается команда

```
>> [X, Y]=meshgrid(x, y);
```

Значения векторов можно указать прямо внутри команды `meshgrid(-2:0.1:2,-4:0.1:4)`. Если оба вектора одинаковые, то можно записать: `[X, Y]= meshgrid(x)`.

Создаётся прямоугольная сетка с шагом 0,1, которая используется для построения трехмерной поверхности. Каждому значению  $x$  ставится в соответствие каждое значение  $y$ . Для каждой пары  $x_i y_j$  (в узлах сетки) будет вычислено значение функции  $Z$ . В результате сформируется матрица из вычисленных значений функции. После этого можно вызвать команду для вывода изображения на экран.

Ниже приведен пример построения графика с использованием функции `plot3` (рис. 26):

```
>>x=-2:0.1:2;  
>>y=-4:0.1:4;  
>>[X, Y]=meshgrid(x, y);  
>>Z=2*X.*exp(-X.^2 - Y.^2);  
>>plot3(X, Y, Z)  
>>grid
```

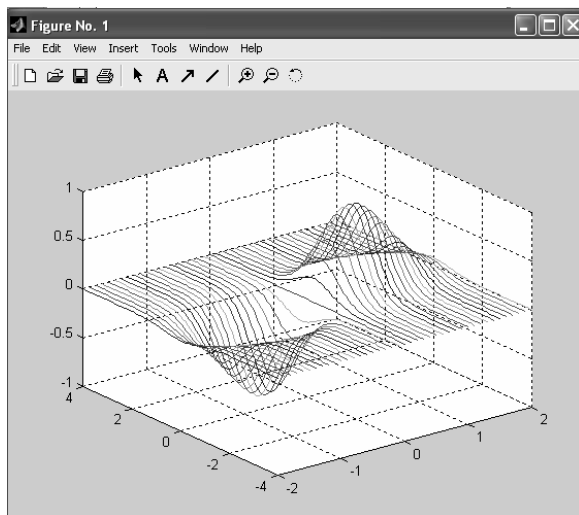


Рис. 26

Более наглядные графики получаются с использованием функции `mesh` (рис. 27).

```
>> mesh(X,Y,Z)
```

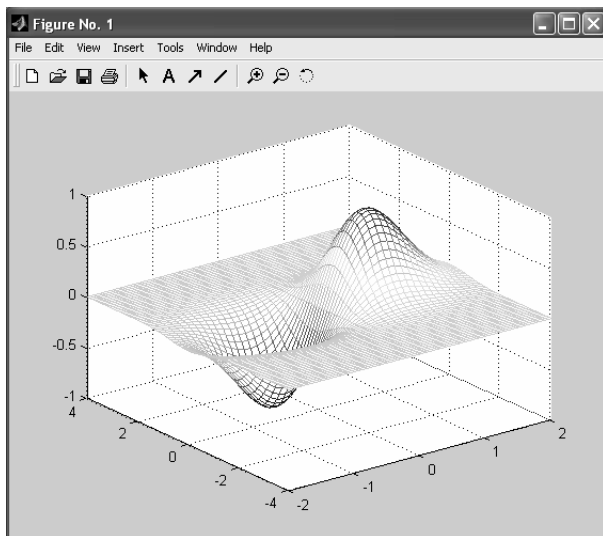


Рис. 27

Линии на разных участках графика окрашены в разные цвета. Эти цвета соответствуют значениям функции. По умолчанию оттенки красного цвета соответствуют большим значениям функции, а синего – меньшим.

Можно сделать “прозрачной” каркасную поверхность, отобразив ее скрытые части, применив команду *hidden off*. Отмена – *hidden on*.

Если использовать функцию *meshz(X,Y,Z)*, то будет видно основание, на котором построен график.

**Пример построения сферы.** Пусть необходимо начертить поверхность, которая не может быть представлена выражением  $z=f(x,y)$ , например сферу  $x^2+y^2+z^2=1$ . В этом случае параметры поверхности можно задать, используя подходящую координатную систему, например, сферические координаты. Можно взять в качестве параметров вертикальную координату  $z$  и полярную координату  $\theta$  на плоскости  $x-y$ .

Если  $r$  обозначает расстояние до оси  $z$ , то тогда выражение для сферы становится  $r^2 + z^2 = 1$ , или  $r = \sqrt{1 - z^2}$ , откуда  $x = \sqrt{1 - z^2} \cos \theta$ ,  $y = \sqrt{1 - z^2} \sin \theta$ .

Построение сферической поверхности можно выполнить командами

```
>>[z,t]=meshgrid(-1:0.1:1, (0:0.1:2)*pi);
```

Вектор значений  $Z$

Вектор значений  $t$  от 0 до  $2\pi$

```
>>x=sqrt(1-z.^2).*cos(t);
```

```
>>y=sqrt(1-z.^2).*sin(t);
```

```
>>surf(x,y,z)
```

```
>>axis square
```

Результат приведен на рис. 28.

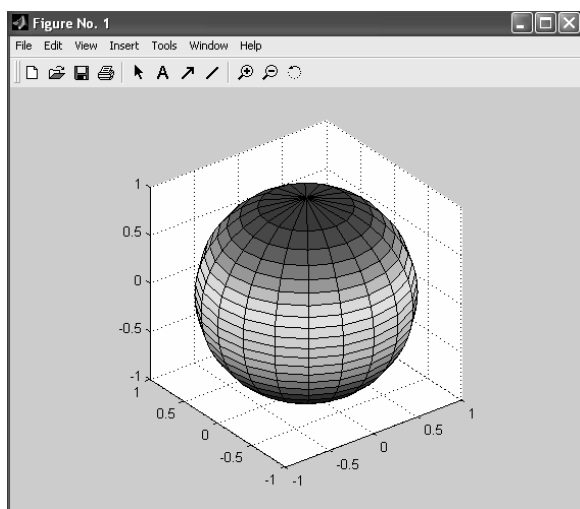


Рис. 28

Все трехмерные графики, которые мы строили, можно было видеть только из одной точки обзора, т.е. позиции, с которой мы видим трехмерное изображение. В ML можно изменять точку обзора. Эта точка характеризуется двумя параметрами: азимутом ( $Az$ ) и углом возвышения ( $El$ ). *Азимут* определяет угол поворота вокруг оси  $z$  и отсчитывается от оси, противоположной  $y$ , в направлении против часовой стрелки. *Угол возвышения* – это угол между отрезком, направленным из начала координат в точку обзора, и

плоскостью  $xz$ . Точку обзора можно менять программно, используя функцию  $view(Az, El)$  или более просто, используя специальную кнопку на панели инструментов графического окна Figure – Rotate 3D.

## ПРОГРАММИРОВАНИЕ

Система ML предоставляет пользователю для решения различных задач мощный язык программирования высокого уровня, понятный непрофессиональным программистам. До сих пор все вычисления и операции мы производили в режиме прямых вычислений. Для эффективной работы с большими наборами команд этого недостаточно. Гораздо лучше было бы записать этот набор команд в виде программы и сохранить ее на диске. Программа представляет собой последовательность команд, записанных на языке ML (программный код), и сохраняется на диске в виде *m-файла*.

Создать *m-файл* можно с помощью любого текстового редактора, но в ML предусмотрен собственный встроенный редактор *EDITOR*, который имеет удобные средства для создания и отладки программ. Написанный текст программы студенты должны сохранять в файле на диске, куда разрешена запись, задав ему имя (***имя файла и имя каталога не должны содержать русских букв***). Этот файл будет иметь расширение *.m*. Чтобы войти в текстовый редактор, надо выбрать пункт меню *File, New, m-file*. В редакторе *m-файлов* можно не только набрать текст программы, но и запустить ее на выполнение. Для сохранения программы на диске надо выбрать пункт меню *File, Save as, <имя m-файла>*.

*m-файлы* могут быть двух видов: файл-программа, или *Script-файл (Script m-File)*, и файл-функция (*Function m-File*).

Файлы-программы называют *Script-файлами*, или сценариями. Они состоят из последовательности команд и не имеют входных и выходных параметров. Они обычно используются для автоматизации выполнения большого набора команд. Вызов *Script-файла* осуществляется просто указанием его имени.

В файлах-функциях описываются функции, определяемые пользователем. Они могут иметь входные и выходные значения. Обращение к файлу-функции осуществляется указанием имени и в круглых скобках списка параметров.

## ОПЕРАТОРЫ ЯЗЫКА

Программа может иметь комментарии. Символ % означает, что далее следует поясняющий текст. Все комментарии начинаются с этого знака. Комментарий не является исполняемым оператором.

Вывод комментариев на экран при выполнении программы обеспечивает оператор *echo on*. Отменяет вывод комментариев на экран оператор *echo off*. Оператор *pause* приостанавливает выполнение программы и ожидает нажатия любой клавиши для продолжения. Оператор *pause(n)* создает паузу в *n* секунд.

Операторами языка можно пользоваться как при создании программ, так и в режиме прямых вычислений.

### Операторы ввода/вывода

Интерактивное взаимодействие пользователя с программой реализуется с помощью функций *input* и *display*.

Оператор *input(<текст>)* обеспечивает ввод данных с клавиатуры. Текст, указанный в качестве параметра, заключается в апострофы. Он отображается на экране при вводе. Обращение к этой функции имеет вид

```
<имя переменной>= input(<текст>)
```

Например, при выполнении команды *x=input('вв. x= ')* на экран выводится текст 'вв. x=' и ожидается ввод данных с клавиатуры. Введенное данное присваивается переменной *x*.

При наборе в командном окне

```
>> r=input('Введите радиус ');
```

на экране получим

```
<<Введите радиус>>
```

Далее ожидается ввод значения переменной *r*.

Если в текст выводимой строки ввести символы '\n', то курсор будет перемещен на следующую строку:

```
>> r=input('Введите \n радиус ');
```

```
<<Введите  
радиус>>
```

Оператор *display(<параметр>)* или *disp(<параметр>)* выводит значение переменной или константы.

*disp(A)* выведет матрицу A, причем выводятся только значения элементов, а текст 'A=', выводиться не будет.

*disp('Привет')* – на экране будет выведено слово «Привет».

## Операторы цикла и условные операторы

Язык ML имеет специальные управляющие конструкции, которые позволяют задавать последовательность выполнения команд в программах. Такие конструкции называются операторами управления и задаются с помощью ключевых слов. Это операторы цикла *for* и *while*, условный оператор *if* и оператор переключения (выбора) *switch...case*. Синтаксис и выполняемые действия этих операторов аналогичны соответствующим операторам известных языков программирования. Область действия каждой конструкции ограничивается словом *end*. *MATLAB* допускает вложенность операторов.

**Оператор цикла с параметром** позволяет описывать действия, которые выполняются фиксированное количество раз:

```
for <параметр цикла>=<начальное значение>:<шаг>:<конечное значение>  
    <операторы>  
end
```

Для досрочного прерывания цикла используется оператор *break*.

Если шаг изменения параметра цикла равен единице, то его можно не указывать. Параметры цикла, их начальное и конечное значения и шаг изменения могут быть целого и вещественного типа.

Например, требуется вычислить значения  $y=x^2$  для всех  $x$  на отрезке от 1 до 10 с шагом равным 3:

```
>>for x=1:3:10  
y=x^2  
end
```

Результат выполнения данного оператора:

```
y =  
1
```

```
y =  
    16  
y =  
    49  
y =  
   100
```

Или вычислим значения  $y$  для всех  $x$  на отрезке от 0,1 до 1 с шагом равным 0,5:

```
>> for x=0.1:0.5:1  
y=x^2  
end
```

В результате получили

```
y =  
    0.0100  
y =  
    0.3600
```

Везде, где возможно, в ML следует использовать не циклы, а векторные и матричные операции (так как это быстрее).

Пусть требуется составить вектор из пяти нечетных чисел натурального ряда. В ML решить эту задачу можно тремя способами:

Первый способ – использовать оператор цикла

```
>> for i =1:5  
    x(i)=2*i-1  
end
```

На экране получим

```
x =  
    1  
x =  
    1    3  
x =  
    1    3    5  
x =  
    1    3    5    7  
x =  
    1    3    5    7    9
```



Этот способ очень неэффективен по времени, так как память не выделяется сразу под весь массив, а при каждой итерации цикла заново создается вектор  $x$  с размерностью на единицу большей, чем на предыдущем шаге, что и приводит к большим затратам времени. Это показывает распечатка результатов.

Второй способ – сформировать вектор из нулей  $x=zeros(1,5)$ , а потом заполнить его значениями:

```
>> x=zeros(1,5)
x =
    0    0    0    0    0
>> for i=1:5
    x(i)=2*i-1
end
```

В результате получим

```
x =
    1    0    0    0    0
x =
    1    3    0    0    0
x =
    1    3    5    0    0
x =
    1    3    5    7    0
x =
    1    3    5    7    9
```

В данном случае мы заранее определили размер вектора, в который записываются результаты работы цикла. Этот способ несколько быстрее первого.

Третий способ – можно сформировать два вектора: вектор  $i=1:5$  и вектор  $x=2*i-1$

```
>> i=1:5
i =
    1    2    3    4    5
>> x=2*i-1
x =
    1    3    5    7    9
```

В данном случае применяются векторные операции вместо использования циклов, и этот способ самый быстрый.

Чтобы повысить эффективность работы программы, следует иметь в виду следующее правило: везде, где возможно, заменять фрагменты программы, содержащие циклы *for* и *while*, на матричные операции или матричные функции. Это объясняется тем, что скорость выполнения матричных и векторных операций выше.

Если нельзя обойтись без циклов, то можно заставить их выполняться быстрее, если следовать правилу: создавать заранее каждый массив, в который требуется записывать результат работы цикла.

**Оператор цикла с предусловием** позволяет запрограммировать цикл, в котором количество повторений зависит от выполнения некоторого условия:

```
while <логическое выражение, задающее условие продолжения цикла>  
    <операторы>  
end
```

Сформируем вектор нечетных чисел:

```
>>x=zeros(1, 5)  
>> i=1;  
>> while i<=5  
    x(i)=2*i-1;  
    i=i+1;  
end
```

В результате получим вектор

```
x =  
    1     3     5     7     9
```

Как видим, такой способ получения последовательности нечетных чисел тоже возможен, но он также является неэффективным, т. е. медленным.

Найти наибольшее число, факториал которого < 150:

```
>>echo on  
% нахождение наибольшего числа, факториал, которого меньше 150  
>>i=1;  
    f=1;  
while f<150  
    i=i+1;  
    f=f*i;  
end  
disp(i-1)
```

на экране получим

5

или можно воспользоваться встроенной функцией:

```
>> i=1;
while prod(1:i)<150
    i=i+1;
end;
disp(i-1)
```

на экране получим

5

**Условный оператор** позволяет организовать ветвления в программе и имеет следующую синтаксическую конструкцию:

```
if (<логическое выражение1>)
    <операторы>
elseif (<логическое выражение2>)
    <операторы>
elseif (<логическое выражение3>)
    <операторы>
...
else
    <операторы>
end
```

В этой конструкции может быть произвольное количество вложений ветвей *elseif*, или их может не быть вовсе.

Пусть необходимо по введенной с клавиатуры температуре воздуха охарактеризовать погоду. Введем следующий текст:

```
>> t=input('введи температуру')
if (t<0)
    s='мороз'
elseif and(t>0,t<=10)
    s='холодно'
elseif and(t>10,t<=20)
    s='тепло'
else
    s='жарко'
end
```

В результате выполнения этих команд получим на экране

введи температуру (например, ввели -4) -4

t =

-4

s =

мороз

Логическое выражение может быть записано и в такой форме:

(t>0)&(t<=10)

**Оператор переключения (выбора).** Если необходимо сделать множественный выбор, можно воспользоваться оператором переключения, синтаксическая конструкция которого приведена ниже:

*Switch* <ключевое выражение>

*Case* { <список значений1>

<операторы>

*Case* {<список значений2>}

<операторы>

.....

[*Otherwise*

<операторы>]

*End*

Сначала вычисляется значение ключевого выражения. Затем это значение ищется в списках значений. Если совпадение найдено, то выполняются операторы соответствующего блока, в противном случае выполняются операторы, стоящие после ключевого слова *otherwise*.

Пусть необходимо по введенному с клавиатуры номеру месяца вывести название времени года. Ниже приведены команды, позволяющие решить эту задачу:

```
>>n=input('введи номер месяца : ')
```

```
switch n
```

```
case {1,2,12}
```

```
    disp('зима')
```

```
case {3,4,5}
```

```
    disp('весна')
```

```
case {6,7,8}
```

```
    disp('лето')
```

```

case {9,10,11}
    disp('осень')
otherwise
    disp('ошибка')
end
На экране получим:
введи номер месяца : 11
n =
    11
осень

```

## ВСТРОЕННЫЕ ФУНКЦИИ EVAL И MENU

**Функция eval.** В ML имеется возможность сформировать требуемую команду в виде строки символов, а затем выполнить ее, как обычную команду, набранную в командном окне. Для этого предназначена встроенная функция *eval*.

Метод задания команд в виде текстовых строк и их выполнения с помощью этой функции используется при написании *m*-файлов, так как такой метод позволяет реализовывать более эффективные алгоритмы решения задач.

Функция *eval* (<текст>) предназначена для выполнения команды, заданной текстом, как команды ML.

Например, *eval* ('2\*2+3') выводит на экран 7. Аргументом функции может быть также строковая переменная.

Команда *eval* ('<имя скрипт-файла>') вызовет выполнение операторов, находящихся в файле с соответствующим именем.

**Функция menu.** Эта функция предназначена для вывода на экран окна с выпадающим меню, которое имеет заголовок и кнопки с названиями пунктов меню. Количество кнопок, равное количеству пунктов меню, определяется количеством параметров функции *menu*:

```

k=menu('<заголовок меню>','<назв. пункта 1>',...,'<назв. пункта n>')

```

Номер выбранной кнопки является результатом функции *menu*.

**П р и м е р.** Пусть необходимо построить график функции на заданном интервале от 0 до  $2\pi$ , цветом, выбранным из меню.

Напишем script-файл:

```
k=menu ('цвет','красный','зеленый','синий')
color=['r'
      'g'
      'b'];
t=0:0.1:2*pi;
s=sin(t);
plot(t,s,color(k))
```

При выполнении этого скрипт-файла функция *menu* выводит на экран окно, изображенное на рис. 29.



Рис. 29

При нажатии на одну из кнопок с названием цвета номер этой кнопки становится результатом функции *menu* и попадает в переменную *k*. Вектор-столбец *color* содержит атрибуты цвета для построения графика. По номеру *k* нужный атрибут выбирается из массива *color* и указывается в команде *plot*. В результате, выбрав нужный пункт меню, получим график функции, построенный выбранным цветом.

Для организации многократного выбора и выполнения нужных действий с помощью меню можно использовать цикл *while* и функцию *eval*. Для этого команды, реализующие необходимые действия, оформляются в виде скрипт-файлов: для каждого пункта меню – свой скрипт-файл. Затем необходимо задать символьную матрицу, например *D*, состоящую из названий файлов, соответствующих пунктам меню (заданий). Строки символьной матрицы должны иметь одинаковую длину. Команда *return* обеспечивает возврат из функции или скрипт-файла в точку вызова. В данном примере можно также использовать команду *break*, которая предназначена для досрочного прерывания цикла. Предположим, что необходимо выполнить по выбору три задания в любом порядке:

```
D=['f1      ',
   'f2      ',
   'f3      ',
   'return'];
```

```

n=1;
while n<4
n=menu('меню', 'пункт1', 'пункт2', 'пункт3', 'выход')
eval(D(n,:))
end

```

Каждая строка символьной матрицы  $D$  содержит имя одного из файлов, реализующих соответствующий пункт меню. В  $n$ -й строке матрицы находится название файла, выполняющего  $n$ -й пункт меню. Обращение  $D(n,:)$  позволяет извлечь из матрицы  $D$   $n$ -ю строку, содержащую название файла. В результате выполнения команды *menu* в  $n$  записывается номер выбранного пункта меню. Для выполнения файл-программы достаточно в командной строке указать имя этого скрипта. Функция *eval* с параметром  $D(n,:)$  запускает соответствующий скрипт-файл для выполнения.

## СОЗДАНИЕ И ИСПОЛЬЗОВАНИЕ *m*-ФАЙЛОВ

Эффективным способом выполнения команд в ML является применение *m*-файлов. Текстовый файл, содержащий набор инструкций на языке системы ML, называется *m*-файлом. Инструкции выполняются последовательно от начала до конца файла. Существует возможность досрочного выхода из *m*-файла, для этого используется оператор *return*. При необходимости прервать работу с ML используется оператор *exit*.

Как уже говорилось, *m*-файлы бывают двух видов: *script m*-файл и *function m*-файл.

### Script-файлы

Файлы-программы (их называют скриптами или сценариями) являются самым простым типом *m*-файлов. Script-файл состоит из последовательности команд, не содержит заголовка, а также входных и выходных параметров. Все объекты, используемые внутри script-файла, считаются глобальными. Если в рабочем пространстве есть данные, то внутри script-файла их можно использовать, а по окончании его выполнения использовать данные, созданные с его помощью. Такие файлы используются для автоматизации выполнения большого набора инструкций.

В одной строке программы можно размещать несколько операторов, тогда они разделяются либо запятой, либо точкой с запятой.

той. Если оператор длинный и не умещается на одной строке, то в конце строки ставятся три точки и дальше на другой строке продолжается набор этого оператора.

Для выполнения файла-программы достаточно в командной строке указать имя этого скрипта. Перед запуском программы на выполнение необходимо установить свой каталог в качестве текущего. Запуск файла на выполнение можно осуществить двумя способами: из окна редактора и из командного окна.

Для запуска программы на выполнение из окна редактора надо выбрать пункт меню *Debug, Run* или нажать функциональную клавишу *F5*, или выбрать соответствующую пиктограмму на панели инструментов. Выполнить программу, уже сохраненную на диске, можно из командного окна ML, просто набрав имя файла без расширения.

Создадим script-файл, в который поместим текст программы, позволяющей вычислить длину окружности по величине ее радиуса.

В текстовом редакторе наберем следующий текст:

```
% Вычисление длины окружности
r=input('Введите радиус окружности ')
l=2*pi*r;
disp('Длина окружности равна')
disp(l)
```

Сохраним его на диске с именем *dl\_okr* в своей папке. Затем вызовем его для выполнения, указав в окне *Command Window* в командной строке имя файла *dl\_okr*.

```
>> dl_okr
```

Получим на экране:

Введите радиус окружности 12

r =

12

Длина окружности равна

75.3982

## Файлы-функции

Система ML позволяет описать свои пользовательские функции и обращаться к ним, как к встроенным. В отличие от script-файла файл-функция имеет входные параметры и может переда-



вать результат в виде выходных параметров. Файлы-функции часто применяют при решении задач вычислительной математики.

Можно создавать файл-функцию с любым количеством входных и выходных параметров. Файл-функция всегда состоит из заголовка и тела функции.

Заголовок функции:

*function[<список вых. параметров>]=<имя функции>(<список вх. параметров>)*

Заголовок функции включает ключевое слово *function*, имя функции, входные и выходные параметры. После заголовка записываются операторы, реализующие алгоритм поставленной задачи. Они составляют тело функции.

Если функция содержит только один выходной параметр, то квадратные скобки не нужны, а если несколько, то они указываются через запятую в квадратных скобках, образуя вектор. Переменным из списка выходных параметров присваиваются значения, вычисленные в функции. Список входных параметров указывается через запятую в круглых скобках после имени функции. Имя *m*-файла, в котором сохраняется текст функции, должно совпадать с именем функции. Имя функции не несёт результата.

При вызове файла-функции нужно указать ее имя и список фактических параметров. Это можно осуществить либо в командном режиме, либо из других файлов-программ или файлов-функций.

*Вызов (активизация) файла-функции* с одним выходным параметром может иметь вид

*<имя функции>(<список входных фактических параметров>)*

и может использоваться в операторах присваивания или в выражении:

*<имя переменной>=<имя функции>(<список входных фактических параметров>)*

Выходных параметров может быть столько, сколько необходимо, тогда их список перечисляется через запятую в квадратных скобках:

*function[<z1, z2,...,zn>]=<имя функции>(<список входных параметров>)*

*<операторы, в результате которых выходные параметры получают значения>*

**Вызвать** такую функцию можно так:

$[y1, y2, \dots, yn] = \langle \text{имя функции} \rangle (\langle \text{список вх. факт. параметров} \rangle)$

Все переменные, используемые внутри файла-функции, а также входные и выходные параметры являются локальными. Они доступны только внутри функции.

Напишем файл-функцию, позволяющую вычислить факториал любого числа.

Не будем использовать оператор цикла, а используем функцию *prod(1:n)*. В редакторе наберем следующий текст:

```
% нахождение факториала любого числа
function f=factorial(x)
f=prod(1:x);
```

Сохраним файл на диске с именем *factorial*. Этот файл получит расширение *.m*. Обратимся к функции для вычисления факториала числа 5:

```
>> factorial(5)
ans =
    120
или
>> f=factorial(5)
f =
    120
```

### ***Отличия файла-функции от script-файла:***

1. Файл-функция имеет входные и выходные параметры, а *script*-файл – нет.

2. Все переменные, используемые внутри файла-функции, локальны. Файл-программа работает с переменными рабочего пространства, т.е. в *script*-файле можно использовать глобальные данные.

3. Функция перед выполнением компилируется. *Script*-файл не компилируется целиком, а выполняется в режиме интерпретации.

4. Имя файла, в котором записана функция, должно совпадать с именем функции, а имя *script*-файла может быть любым.

Программа будет выполняться быстрее, если вместо *script*-файла использовать файл-функцию (в случае, если это возможно).

## ИСПОЛЬЗОВАНИЕ ФАЙЛОВ-ФУНКЦИЙ

Файлы-функции широко используются в ML. Пользователь может с их помощью описывать свои функции для многократного выполнения, также с их помощью могут решаться следующие задачи:

- 1) построение графика функций;
- 2) численное интегрирование;
- 3) нахождение корней трансцендентного уравнения;
- 4) поиск экстремумов функции;
- 5) решение систем дифференциальных уравнений.

### Построение графика

График функции на заданном отрезке, символьное выражение (формула) которой записано в файле-функции, можно построить, используя команду

*fplot ( fun, interval),*

где *fun* – имя функции или символьное выражение, задающее формулу вычисления; *interval* – отрезок, который задается вектором из двух значений: начального и конечного значения отрезка.

В качестве примера построим график функции  $x^2-1$  на отрезке от 0 до  $\pi$ . Для функции  $x^2-1$  напомним файл-функцию. В текстовом редакторе наберем следующий текст:

```
function f=func(x)
    f=x.^2-1;
```

Сохраним его в файле с именем *func.m*.

Для построения графика (рис. 30) выполним команду

```
>> fplot('func',[0 pi])
```

Или используем вместо имени функции символьное выражение

```
>> fplot('sin(x)',[0 pi])
```

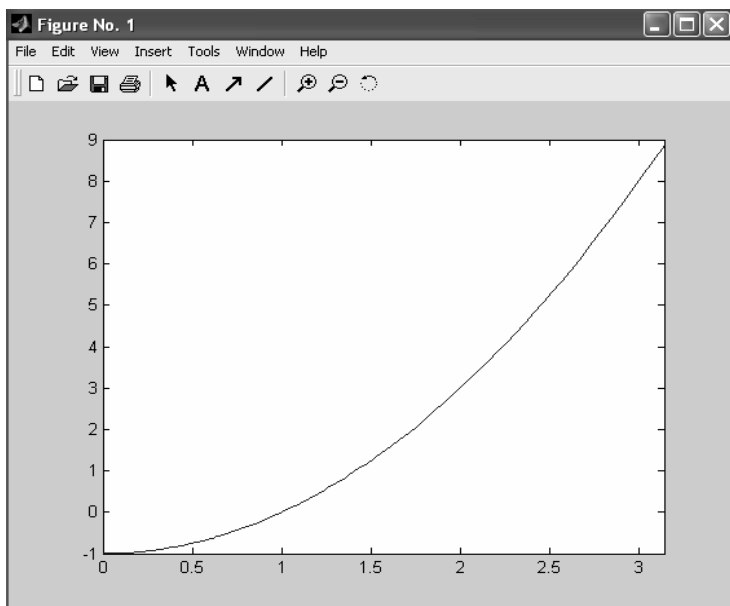


Рис. 30

### **Вычисление интеграла**

Самым простым численным методом нахождения определенных интегралов является метод трапеций, который реализован функцией *trapz* (*x*, *y*), где *x* и *y* – векторы, задающие значения аргумента и функции в пределах интегрирования. Для ее использования не требуется оформление подынтегральной функции в качестве файла-функции.

Вычислим интеграл  $\int_0^{\pi} \sin(x) dx$  :

```
>> x=0:pi/100:pi;
>> y=sin(x);
>> z=trapz(x,y);
z =
    1.9998
```

Получили приближенное значение интеграла. Увеличив шаг интегрирования до  $\pi/1000$ , получим более точное значение – 2.

Более точные методы реализованы в функциях *quad*, *quad8*, *quadl*.

Функция *quad* основана на квадратурной формуле Симпсона, а функция *quadl* предназначена для вычисления интеграла по более точным формулам Гаусса–Лейбница. Функция *quad8* реализует метод Ньютона–Котеса восьмого порядка. Все три функции имеют одинаковый набор параметров:

$[i, n] = \text{quad}(\text{fun}, a, b, \text{tol}, \text{trace}),$

где *fun* – имя файла-функции или символьное выражение, определяющее подынтегральную функцию (оно должно быть заключено в апострофы); *a*, *b* – пределы интегрирования; *tol* – погрешность вычислений; *trace* – признак; если его значение равно единице, то при выполнении функции строится график подынтегральной функции (рис. 31), если нулю, то не строится. Последние два параметра являются необязательными; *i* – значение интеграла; *n* – количество значений функции, которое потребовалось вычислить для нахождения значения интеграла.

```
>> [i, n]=quad8('func', 0, pi, 1e-8, 1)
```

```
i =
```

```
7.1938
```

```
n =
```

```
33
```

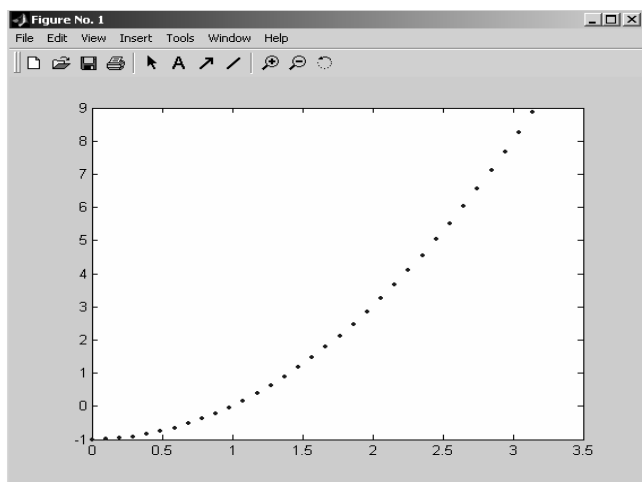


Рис. 31

При использовании этих численных методов задаётся требуемая точность вычислений, а шаг подбирается автоматически (в отличие от функции *trapz*). По умолчанию точность равна  $10^{-6}$ .

Функция может быть задана символьным выражением:

```
>> [i,n]=quad8('sin(x)', 0, pi, 1e-8, 0)
i =
    2.0000
n =
    33
```

Такой же результат получим, если не укажем два необязательных параметра:

```
>> [i, n]=quad8('sin(x)', 0, pi )
i =
    2.0000
n =
    33
```

Тот же интеграл можно вычислить с использованием функции *quadl*:

```
>> [i, n]=quadl('sin(x)',0,pi,1e-8,0)
i =
    2.0000
n =
    48
```

Для достижения одной и той же точности понадобилось разное количество вычислений. Можно при вызове функции не указывать выходной параметр *n*. Тогда команда обращения к функции будет такой:

```
>> i=quadl('sin(x)', 0, pi, 1e-8, 0)
i =
    2.0000
```

В этом случае не используется количество значений функции, которое потребовалось вычислить для нахождения значения интеграла.

## Решение трансцендентных уравнений

Для решения уравнений вида  $f(x)=0$  (нахождения нулей функции  $f(x)$ ) предназначена функция *fzero*, которая находит приближенное значение корня уравнения по заданному начальному приближению. Функция *fzero* находит значение аргумента, при котором функция обращается в ноль:

*fzero(fun, x0, tol, trace)*,

где *fun* – имя файл-функции или символьное выражение (формула), заключенные в апострофы; *x0* – начальное значение  $x$ , от которого начнётся процесс; *tol* – погрешность вычислений; *trace* – признак; если значение единица, то график строится, если нуль, то не строится; *tol* и *trace* являются необязательными параметрами.

Если корней несколько, то, задавая различные начальные значения, можно найти различные корни.

Например, построим график функции  $\sin(x)/(x^2)$  на отрезке  $[-10\ 1]$  (рис. 32):

```
>> s='sin(x)/(x*2)'  
>> fplot(s, [-10 1])
```

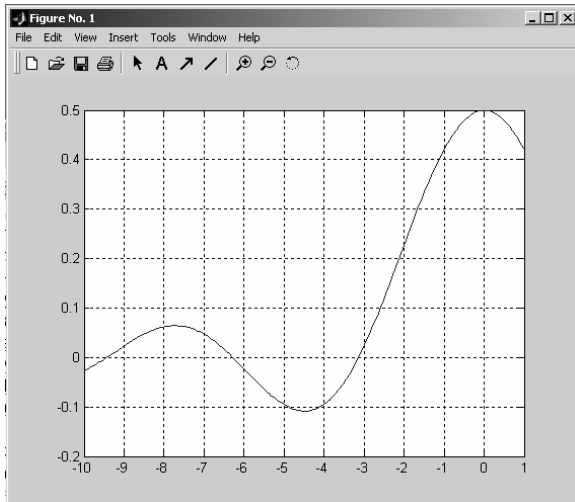


Рис. 32

При различных начальных значениях получаем различные корни:

```
>> fzero(s, -7)
ans =
    -6.2832
>> fzero(s, -4)
ans =
    -3.1416
```

### ***Нахождение минимума функции на заданном отрезке***

Вычисление локального минимума функции одной переменной на заданном отрезке выполняет функция *fminbnd*. При использовании функции поиска минимума рекомендуется предварительно построить график исследуемой функции, чтобы точнее определить отрезки поиска минимума:

$$[x, y] = \text{fminbnd}(\text{fun}, a, b),$$

где *fun* – имя функции или символьное выражение (формула), заключенное в апострофы; *a*, *b* – границы отрезка, на котором ищется минимум.

Результат – вектор, состоящий из двух значений: *x* – аргумент функции (вычисленное значение), при котором достигается минимум; *y* – значение функции в точке минимума;

**П р и м е р.** Для функции  $\sin(x)/(x^2)$  найти локальный минимум на отрезке  $[-10 \ 1]$ , вывести значение функции в этой точке и построить график:

```
>> [x,y]=fminbnd('sin(x)/(x*2)',-10,1)
x =
    -4.4934
y =
    -0.1086
```

Если хотим получить только значение аргумента, при котором функция будет иметь минимальное значение, то обращение к функции будет таким:

```
>> fminbnd('sin(x)/(x*2)',-10,1)
ans =
    -4.4934
или
```



```
>> x=fminbnd('sin(x)/(x*2)',-10,1)
x =
    -4.4934
>> ezplot('sin(x)/(x*2)',-10,1)
>> grid on
```

График приведен на рис. 33

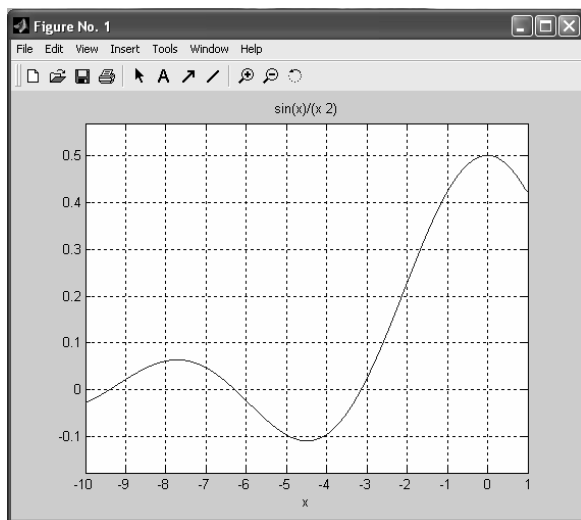


Рис. 33

Для нахождения локального максимума не предусмотрено специальной функции. Можно найти максимум, используя *fminbnd* для функции, имеющей обратный знак.

### ***Решение систем дифференциальных уравнений***

Система ML имеет встроенные средства для решения обыкновенных дифференциальных уравнений (ДУ) любого порядка или систем ДУ. Дифференциальным уравнением  $n$ -го порядка называется соотношение вида  $F(t, y, y', \dots, y^{(n)})=0$ . Наивысший порядок производной искомой функции в данном уравнении называется порядком дифференциального уравнения. Решением ДУ является такая функция, которая при подстановке в него обращает его в равенство. В случае системы дифференциальных уравнений  $n$ -го порядка решением являются  $n$  функций. При использовании численных методов решения искомые функции выражаются не аналити-

чески, а в табличном виде (в виде набора значений в определенных точках). Самое простое ДУ – это уравнение 1-го порядка, разрешенное относительно производной. Оно имеет вид  $y' = f(t, y)$ . ДУ и системы 2-го порядка и выше можно свести к системам 1-го порядка. Так, для того чтобы свести к системе 1-го порядка уравнение  $y'' = f(t, y, y')$ , нужно использовать замену:  $y_1 = y$  и  $y_2 = y'$ . В результате замены будем иметь систему

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= f(t, y_1, y_2) \end{aligned}$$

В ML имеется целый ряд встроенных функций, предназначенных для решения систем дифференциальных уравнений. Это функции *ode23*, *ode45*, *ode113*. В них реализованы различные методы решения обыкновенных ДУ. Наиболее часто используют функции *ode23* и *ode45*, в основе работы которых лежит метод Рунге–Кутты. Для систем уравнений 2-го и 3-го порядков используют функцию *ode23*, а для систем 4-го и 5-го порядков – функцию *ode45*. Обращение к функции, находящей решение системы ДУ, имеет вид

$$[t, y] = \text{<имя встроенной функции>}(fun, interval, y0),$$

где *fun* – имя *m*-файла, в котором вычисляются правые части системы ДУ; *interval* – вектор, определяющий отрезок, на котором ищется решение; *y0* – скаляр или вектор-столбец, в котором задаются начальные условия; *t* – вектор-столбец, содержащий значения независимой переменной на заданном интервале (обычно это время *t*); *y* – матрица решений. Количество столбцов в ней равно порядку системы ДУ. Каждый столбец содержит значения одной искомой функции на заданном отрезке. Каждая строка соответствует определенному значению *t*.

**П р и м е р.** Имеем ДУ 2-го порядка  $y'' + 4y' + 5y = 0$ , описывающее движение точки в зависимости от времени. Требуется его решить на отрезке  $[0, 5]$  с начальными условиями  $y(0) = 3$  и  $y'(0) = -5$ .

Выполним замену:  $y_1 = y$  и  $y_2 = y'$ , получим систему ДУ:

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= -5y_1 - 4y_2 \end{aligned}$$

при  $y_1(0) = 3$  и  $y_2(0) = -5$  (тогда  $y_1$  будет обозначать координаты точки, а  $y_2$  – ее скорость).

Составим файл-функцию для вычисления правых частей системы ДУ. Эта функция должна иметь два входных параметра: переменную  $t$ , вектор  $y$  с количеством элементов, равным числу неизвестных системы, и один выходной параметр (вектор правых частей системы):

```
function F=mydif(t,y)
F=[y(2); -5*y(1)-4*y(2)];
```

Сохраним файл с именем *mydif* в каталоге, разрешенном для записи. Обратимся к функции *ode45*:

```
>>[t,y]=ode45('mydif', [0 5],[3;-5])
```

В результате будут получены вектор-столбец  $t$  – вектор значений аргумента (в нашем примере это время) и матрица  $y$  из двух столбцов. Первый столбец содержит значения функции  $y_1$ , т.е. самой искомой функции при каждом значении  $t$ , а второй столбец – значения функции  $y_2$ , т.е. первой производной искомой функции. У нас в первом столбце будут получаться значения координат точки, а во втором – значение скорости ее движения в каждый момент времени  $t$ .

Отобразим график решения ДУ и график производной (рис. 34):

```
>> plot(t,y(:, 1), 'r', t, y(:, 2),'k--')
>> xlabel('t');
>> ylabel('y')
```

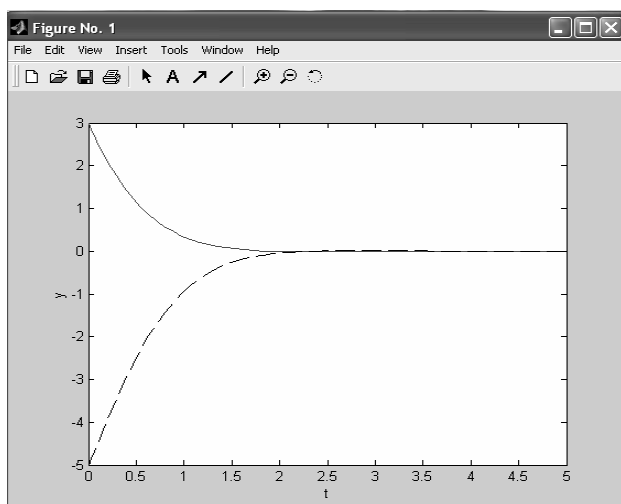


Рис. 34

## ЛАБОРАТОРНАЯ РАБОТА №1

### ЗНАКОМСТВО С ПАКЕТОМ МАТЛАВ, РЕШЕНИЕ ПРОСТЕЙШИХ ЗАДАЧ В СИСТЕМЕ МАТЛАВ

**Цель работы** – познакомиться с интерактивным пакетом Matlab 6.5, предназначенным для решения широкого круга инженерных и математических задач, научиться задавать векторы и матрицы, выполнять различные действия с ними, использовать встроенные функции для их обработки, а также изучить средства графического отображения результатов расчета и освоить создание скрипт-файлов.

#### Постановка задачи

1. Вычисление значений функции на заданном отрезке.
2. Выполнение действий с векторами.
3. Формирование и выполнение действия с матрицами.
4. Построение графиков функций одной переменной на заданном отрезке.
5. Построение графика функции двух переменных.
6. Создание скрипт-файла.

По результатам работы должен быть составлен отчет, содержащий тексты индивидуальных заданий, команды ML для решения задач, текст *script*-файла, а также графическое представление результатов работы.

#### Индивидуальные задания

**Задание №1.** Вычислить  $N$  значений функции на заданном отрезке. На экран вывести значения аргумента и значения функции.

#### Варианты

Функция	Отрезок	Количество разбиений
1. $y(x) = \frac{\sin x \cos x}{x + 1}$	$[0, 2\pi]$	$N=10$
2. $y(x) = \ln(x+1)\sqrt{e^x + e^{-x}}$	$[-0.2, 4]$	$N=9$
3. $y(x) = x^2 \operatorname{tg} \sqrt{\arcsin x}$	$[0, 0.3]$	$N=7$

- |   |               |        |
|---|---------------|--------|
| 4. $y(x) = x \sin x + x^3 \frac{e^x}{x+1}$  | $[0,1]$       | $N=10$ |
| 5. $y(x) = \frac{1}{1 + \frac{x}{\sqrt{1+x}}}$  | $[0,3]$       | $N=8$  |
| 6. $y(x) = \frac{e^{\sin x} + e^{\cos x}}{x^2}$   | $[\pi, 3\pi]$ | $N=8$  |
| 7. $y(x) = \operatorname{ctg}(x^2 + 1) \cdot (\sin 2x + \cos 2x)$                               | $[-1,1]$      | $N=7$  |
| 8. $y(x) = \log_2(x^2 + 1) \sin \frac{1}{x^2 + 1}$  | $[-1,1]$      | $N=10$ |
| 9. $y(x) =  x^3 + 2x^2 - 3  \sin \pi x$   | $[-2,2]$      | $N=7$  |
| 10. $y(x) = \frac{\sqrt[3]{x+1}}{\sqrt{ x  + \frac{1}{2}}} \cdot \frac{\sin x + 1}{\cos x + 2}$ | $[-2,2]$      | $N=9$  |

**Задание №2.** Для заданных векторов  $a$  и  $b$  длиной  $n$  (значения элементов векторов и их длину студент задает сам) выполнить преобразования и вычисления в соответствии с вариантом.

### ***Варианты***

1. В векторе  $a$  элементы с номерами от  $n1$  до  $n2$  удвоить, а в векторе  $b$  заменить их средним арифметическим.

2. Образовать новый вектор  $c = [a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_n]$ , определить его максимальный и минимальный элементы и поменять их местами.

3. Образовать вектор  $c = [a_1, a_2, a_3, b_4, b_5, \dots, b_n]$  и упорядочить его по возрастанию и убыванию.

4. Образовать вектор  $c = [a_3, a_4, \dots, a_n, b_1, b_2, b_3]$  и переставить элементы вектора  $c$  в обратном порядке. Результат записать в новый вектор.

5. Получить вектор  $x$ , содержащий удвоенные значения элементов вектора  $a$ , и вектор  $y$ , содержащий утроенные значения элементов вектора  $b$ . Определить среднее арифметическое каждого вектора.

6. Вычислить среднее арифметическое элементов двух векторов. Заменить минимальный элемент первого вектора на максимальный элемент второго.

7. Получить два новых вектора, состоящих из элементов исходных векторов, начиная с номера  $n1$  до номера  $n2$ . Найти сумму минимальных элементов новых векторов.

8. Заменить нулем минимальный элемент вектора  $a$  и максимальный элемент вектора  $b$ .

9. Вычислить произведение элементов векторов с номерами от  $n1$  до  $n2$ . Найти минимальные значения векторов и заменить последние элементы векторов их минимумами.

10. Образовать вектор  $c = [a_2, a_3, a_4, b_3, b_4, \dots, b_n]$ . Элементы с номерами от  $n1$  до  $n2$  заменить средним арифметическим этих элементов.

**Задание №3.** При помощи встроенных функций для заполнения стандартных матриц, индексации двоеточием и, возможно, объединения, поворота или транспонирования получить следующие матрицы. Применить функции обработки данных и поэлементные операции для нахождения заданных величин.

### Варианты

$$1. \quad A = \begin{bmatrix} 1 & 0 & 0 & 4 & 0 & 0 \\ 0 & 1 & 0 & 0 & 4 & 0 \\ 0 & 0 & 1 & 0 & 0 & 4 \\ 2 & 2 & 2 & 3 & 3 & 3 \\ 2 & 2 & 2 & 3 & 3 & 3 \\ 2 & 2 & 2 & 3 & 3 & 3 \end{bmatrix} \quad m = \max \left\{ \sum_{i=1}^6 a_{ij}^2 \right\}.$$

$$2. \quad A = \begin{bmatrix} 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \end{bmatrix} \quad s = \sum_{i=1}^6 \sum_{j=1}^6 |a_{ij}|.$$

$$3. \quad A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ -1 & -2 & -3 & -4 & -5 & -6 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 \end{bmatrix} \quad m = \min_{i,j=1,\dots,6} a_{ij}^3.$$

$$4. \quad A = \begin{bmatrix} -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 3 & 0 & 0 \\ 2 & 2 & 2 & 0 & 3 & 0 \\ 2 & 2 & 2 & 0 & 0 & 3 \end{bmatrix} \quad s = \sum_{k=1}^6 a_{kk}^3.$$

$$5. \quad A = \begin{bmatrix} 1 & 1 & -3 & -3 & -3 & -3 \\ 1 & 1 & -3 & -3 & -3 & -3 \\ -3 & -3 & 2 & 0 & 0 & 0 \\ -3 & -3 & 0 & 2 & 0 & 0 \\ -3 & -3 & 0 & 0 & 2 & 0 \\ -3 & -3 & 0 & 0 & 0 & 2 \end{bmatrix} \quad s = \sum_{i=1}^6 a_{ii} + \sum_{i=1}^6 a_{4i}.$$

$$6. \quad A = \begin{bmatrix} -1 & 2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 0 & 4 & 4 \end{bmatrix} \quad s = \sum_{i=1}^6 a_{ii} + \sum_{i=1}^6 a_{i6}.$$

$$7. \quad A = \begin{bmatrix} 0 & 0 & 0 & 0 & 3 & 4 \\ 0 & 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 3 & 0 & 0 & 4 \\ 0 & 3 & 0 & 0 & 0 & 4 \\ 3 & 0 & 0 & 0 & 0 & 4 \\ 1 & 1 & 1 & 1 & 1 & -2 \end{bmatrix}$$

$$s = \sum_{i=1}^6 \sum_{j=1}^6 \sin\left(\frac{\pi}{6} a_{ij}^2\right).$$

$$8. \quad A = \begin{bmatrix} 1 & 2 & 3 & -1 & 0 & 0 \\ 1 & 2 & 3 & 0 & -1 & 0 \\ 1 & 2 & 3 & 0 & 0 & -1 \\ 0 & 0 & 7 & 2 & 2 & 2 \\ 0 & 7 & 0 & 2 & 2 & 2 \\ 7 & 0 & 0 & 2 & 2 & 2 \end{bmatrix}$$

$$m = \max_{i=1,\dots,6} \min_{j=1,\dots,6} a_{ij}.$$

$$9. \quad A = \begin{bmatrix} 1 & 0 & 0 & 1 & -3 & -3 \\ 0 & 1 & 1 & 0 & -3 & -3 \\ 0 & 1 & 1 & 0 & -3 & -3 \\ 1 & 0 & 0 & 1 & -3 & -3 \\ 0 & 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 0 & 4 & 4 \end{bmatrix}$$

$$s = \sum_{i=1}^6 \min_{j=1,\dots,6} (a_{ij}).$$

$$10. \quad A = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 \\ -1 & -1 & -1 & 4 & 0 & 0 \\ -1 & -1 & -1 & 0 & 4 & 0 \\ -1 & -1 & -1 & 0 & 0 & 4 \end{bmatrix}$$

$$p = \prod_{i=1}^6 \sum_{j=1}^6 (a_{ij})^{a_{ij}}.$$



**Задание №4.** Построить графики двух функций на заданном отрезке. Вывести графики:

- в разных окнах,
- в одном окне в одних осях,
- в одном окне в разных осях.

Использовать различные цвета, стили, подписи, легенду. Наести сетку.

### ***Варианты***

Функция $f$	Функция $g$	Аргумент $x$
1. $f(x) = \sin x$ ;	$g(x) = \sin^2 x$ ;	$x \in [-2\pi, 3\pi]$
2. $f(x) = \sin x^2$ ;	$g(x) = \cos x^2$ ;	$x \in [-\pi, \pi]$
3. $f(x) = x^3 + 2x^2 + 1$ ;	$g(x) = (x-1)^4$ ;	$x \in [-1, 1]$
4. $f(x) = \ln x$ ;	$g(x) = x \ln x$ ;	$x \in [0.2, 10]$
5. $f(x) =  2x ^3$ ;	$g(x) =  2x ^5$ ;	$x \in [-0.5, 0.5]$
6. $f(x) = x^2$ ;	$g(x) = x^3$	$x \in [-1, 1]$
7. $f(x) = \arcsin x$	$g(x) = \arccos x$ ;	$x \in [-1, 1]$
8. $f(x) = \cos x$	$f(x) = \cos 2x$ ;	$x \in [-1, 1]$
9. $f(x) = \frac{\sin x}{x}$	$g(x) = e^{-x} \cos x$ ;	$x \in [-0.01, 2\pi]$
10. $f(x) = x^x$	$f(x) = x^{x^x}$ ;	$x \in [0.1, 1]$

Построить график кусочно-заданной функции, отобразить ветви разными цветами и маркерами.

### ***Варианты***

$$1 \quad f(x) = \begin{cases} -1, & -3 \leq x \leq -1 \\ x, & -1 < x < 1 \\ e^{1-x}, & 1 < x \leq 3 \end{cases}$$

$$2 \quad f(x) = \begin{cases} \sqrt{x}, & 0 \leq x \leq 1 \\ 1, & 1 < x \leq 3 \\ (x-4)^2, & 3 < x \leq 5 \end{cases}$$

$$3 \quad f(x) = \begin{cases} \ln x, & 1 \leq x \leq 5 \\ x/5, & 5 < x \leq 9 \\ 9e^{8-x}, & 9 < x \leq 12 \end{cases}$$

$$4 \quad f(x) = \begin{cases} \sin x, & -2\pi \leq x \leq 0 \\ -x^3, & 0 < x \leq 1 \\ \cos \pi x, & 1 < x \leq 3\pi \end{cases}$$

$$5 \quad f(x) = \begin{cases} \arcsin x - 1, & 0 \leq x \leq 1 \\ \frac{\pi}{2} - x, & 1 < x \leq \frac{\pi}{2} \\ \cos x, & \frac{\pi}{2} < x \leq \pi \end{cases}$$

$$6 \quad f(x) = \begin{cases} |x|, & -2 \leq x \leq 1 \\ \sin \frac{\pi}{2} x, & 1 < x \leq 2 \\ (2-x)^3, & 2 < x \leq 3 \end{cases}$$

$$7 \quad f(x) = \begin{cases} (x-1)^2, & -2 \leq x \leq 1 \\ \cos \frac{\pi}{2} x, & 1 < x \leq 3 \\ 1 - e^{3-x}, & 3 < x \leq 8 \end{cases}$$

$$8 \quad f(x) = \begin{cases} e^x, & -2 \leq x \leq -1 \\ \frac{|x|}{e^x}, & -1 < x \leq 1 \\ e^{-x}, & 1 < x \leq 2 \end{cases}$$

$$9 \quad f(x) = \begin{cases} e^{x+1}, & -2 \leq x \leq -1 \\ x^2, & -1 < x \leq 1 \\ (2-x)^3, & 1 < x \leq 2 \end{cases}$$

$$10 \quad f(x) = \begin{cases} x^2 \log_2 x, & 1 \leq x \leq 2 \\ x^3/2, & 2 < x \leq 3 \\ x^x/2, & 3 < x \leq 3.5 \end{cases}$$

**Задание №5.** Построить график функции двух переменных.

### *Варианты*

Функция $z$	Аргумент $x$	Аргумент $y$
1. $z(x, y) = \sin x \cdot e^{-3y}$	$x \in [0, 2\pi]$	$y \in [0, 1]$
2. $z(x, y) = \sin^2(x - 2y)e^{- y }$	$x \in [0, 2\pi]$	$y \in [1, 10]$
3. $z(x, y) = \sin^2(x - 2y)e^{- y }$	$x \in [0, \pi]$	$y \in [-1, 1]$
4. $z(x, y) = \frac{x^2 y^2 + 2xy - 3}{x^2 + y^2 + 1}$	$x \in [-2, 2]$	$y \in [-1, 1]$
5. $z(x, y) = \frac{\sin xy}{x}$	$x \in [0.1, 5]$	$y \in [-\pi, \pi]$
6. $z(x, y) = (\sin x^2 + \cos y^2)^{xy}$	$x \in [-1, 1]$	$y \in [-1, 1]$
7. $z(x, y) = \arctan(x + y)(\arccos x + \arcsin y)$	$x \in [-1, 1]$	$y \in [-1, 1]$
8. $z(x, y) = (1 + xy)(3 - x)(4 - y)$	$x \in [0, 3]$	$y \in [0, 4]$
9. $z(x, y) = e^{- x }(x^5 + y^4) \sin(xy)$	$x \in [-2, 2]$	$y \in [-3, 3]$
10. $z(x, y) = (y^2 - 3) \sin \frac{x}{ y  + 1}$	$x \in [-2\pi, 2\pi]$	$y \in [-3, 3]$

**Задание №6.** Написать скрипт-файл для решения следующих задач.

### ***Варианты***

1. По заданному вектору определить номер его элемента с наибольшим отклонением от среднего арифметического всех элементов векторов.

2. Найти среднее арифметическое элементов заданного вектора и заменить первый элемент этим значением.

3. Вычислить максимальное значение среди элементов главной диагонали заданной матрицы.

4. Переставить первый столбец квадратной матрицы со строкой, где находится наименьший элемент.

5. Сложить все элементы заданной матрицы, кроме элементов главной диагонали.

6. Заменить максимальный элемент вектора средним значением всех его элементов.

7. Заменить элемент матрицы с индексами 1,1 произведением всех элементов матрицы.

8. Заменить последний элемент вектора максимальным элементом.

9. Найти значение и номер максимального элемента. Графически отобразить элементы заданного вектора синими маркерами, а максимальный элемент – красным.

10. Упорядочить элементы вектора по убыванию, затем последний элемент заменить средним арифметическим всех элементов вектора.

### ***ЛАБОРАТОРНАЯ РАБОТА №2***

## **РЕШЕНИЕ ТИПОВЫХ ВЫЧИСЛИТЕЛЬНЫХ ЗАДАЧ В СИСТЕМЕ MATLAB**

***Цель работы*** – научиться использовать встроенные функции, реализующие различные численные методы решения, а также средства отображения результатов расчета; освоить создание файлов-сценариев и файлов-функций.

## Постановка задачи

1. Вычисление корней полинома.
2. Решение системы линейных уравнений.
3. Вычисление локального минимума и максимума функции.
4. Вычисление определенного интеграла.
5. Решение трансцендентного уравнения.

Согласно варианту индивидуального задания требуется написать пять программ, соблюдая приведенные ниже указания:

- для каждого пункта задания создать свой *скрипт-файл*;
- написать файл-сценарий (*script-файл*), в котором пользовательский интерфейс оформлен в виде меню. Выбранный пункт меню определяет выполнение файла, соответствующего пункту индивидуального задания. При написании программы для реализации меню использовать *встроенную функцию тегу()*, *оператор цикла с предусловием while* и *функцию eval()*;

- уравнения и функции для выполнения пунктов № 3–5 описать в виде файлов-функций;

- для пункта №1 при вычислении корней полинома построить график полинома и отобразить на нем найденные действительные корни. Ввод границ построения графика должен осуществляться с клавиатуры;

- для пункта №2 при решении системы линейных уравнений осуществить проверку полученного решения;

- для пункта №3 при поиске максимума и минимума функции построить график заданной функции в заданных границах и отобразить на нем полученные экстремумы маркерами разного цвета;

- для пункта №4 при вычислении интеграла построить график подынтегральной функции, границы графика вводить с клавиатуры и закрасить площадь, ограниченную функцией на заданном отрезке;

- для пункта №5 при решении трансцендентного уравнения построить график функции в границах, заданных пользователем в форме диалога, и отобразить на нем значение корня уравнения цветом, отличным от цвета графика;

- на всех графиках должны быть выведены заголовки, названия осей координат. Все графики должны располагаться в одном графическом окне.

### При написании программ обязательно:

- использовать комментарии, содержащие назначение программы и описание ее переменных;

- вывод результатов сопровождать пояснительным текстом.

По результатам работы должен быть составлен отчет, содержащий текст индивидуального задания, тексты *script*-файлов и файлов-функций, а также графическое представление результатов работы.

### Варианты индивидуальных заданий

#### Вариант 1

1. Вычислить корни полинома

$$f(x) = x^2 + 6e^{0,15x}$$

2. Найти решение системы линейных уравнений

$$\begin{cases} 5,4x_1 + 1,8x_2 - 3x_3 = 7 \\ 4,5x_1 - 2,8x_2 + 6,7x_3 = 2,6 \\ 5,1x_1 + 3,7x_2 - 1,4x_3 = -0,14 \end{cases}$$

3. Найти значение локального минимума и максимума функции

$$f(x) = \sqrt{x-1}(\sin(2x) + 3x^2) \\ 1 \leq x \leq 2,5$$

4. Вычислить значение определенного интеграла

$$\int_{0,8}^{1,6} \frac{dx}{\sqrt{x^2 + 1}}$$

5. Решить трансцендентное уравнение

$$x \cdot \lg(x) - 1,2 = 0$$

### **Вариант 2**

1. Вычислить корни полинома

$$x^3 + 0,1x^2 + 0,4x - 1,2$$

2. Найти решение системы линейных уравнений

$$\begin{cases} 3,8x_1 + 6,7x_2 - 1,2x_3 = 5,2 \\ 6,4x_1 + 1,3x_2 - 2,7x_3 = 3,8 \\ 2,4x_1 - 4,5x_2 + 3,5x_3 = -0,6 \end{cases}$$

3. Найти значение локального минимума и максимума функции

$$f(x) = \sqrt{x}(\sin(x) + \cos(x))$$
$$2 \leq x \leq 3$$

4. Вычислить значение определенного интеграла

$$\int_{1,2}^{2,7} \frac{dx}{\sqrt{x^2 + 3,3}}$$

5. Решить трансцендентное уравнение

$$x^2 - 4 \sin 10x = 0$$

### **Вариант 3**

1. Вычислить корни полинома.

$$9x^5 + x^2 - 2x - 1$$

2. Найти решение системы линейных уравнений

$$\begin{cases} 4x_1 - 3x_2 + 2x_3 = 9 \\ 2x_1 + 5x_2 - 3x_3 = 4 \\ 5x_1 + 6x_2 - 2x_3 = 18 \end{cases}$$

3. Найти значение локального минимума и максимума функции

$$f(x) = e^{-0,5x} * x^2 - 0,5 \cos x$$
$$0 \leq x \leq \pi/2$$

4. Вычислить значение определенного интеграла

$$\int_1^2 \frac{dx}{\sqrt{2x^2 + 1,3}}$$

5. Решить трансцендентное уравнение

$$\lg(2 + x) + 2x - 3 = 0$$

#### **Вариант 4**

1. Вычислить корни полинома.

$$x^4 + 2x^3 - x - 1$$

2. Найти решение системы линейных уравнений

$$\begin{cases} 5,92x_1 - 1,24x_2 - 1,84x_3 = 2,44 \\ 2,72x_1 - 9,71x_2 + 2,43x_3 = 2,4 \\ 1,76x_1 - 3,12x_2 + 9,38x_3 = 1,93 \end{cases}$$

3. Найти значение локального минимума и максимума функции

$$\begin{aligned} f(x) &= 2x^3 + 4x - 1 \\ 0 \leq x &\leq 0,5 \end{aligned}$$

4. Вычислить значение определенного интеграла

$$\int_{0,2}^{1,2} \frac{dx}{\sqrt{x^2 + 1}}$$

5. Решить трансцендентное уравнение

$$\cos x - x + 1 = 0$$

#### **Вариант 5**

1. Вычислить корни полинома

$$x^3 - 0,1x^2 + 1,5x - 1,5$$



2. Найти решение системы линейных уравнений

$$\begin{cases} 2,7x_1 + 0,9x_2 - 1,5x_3 = 3,5 \\ 4,5x_1 - 2,8x_2 + 6,7x_3 = 2,6 \\ 5,1x_1 + 3,7x_2 - 1,4x_3 = -0,14 \end{cases}$$

3. Найти значение локального минимума и максимума функции

$$f(x) = 0,5 \sin x - 0,2 \cos x \\ 0 \leq x \leq \pi/3$$

4. Вычислить значение определенного интеграла

$$\int_{0,8}^{1,4} \frac{dx}{\sqrt{2x^2 + 3}}$$

5. Решить трансцендентное уравнение

$$e^x + \sqrt{1 + e^{2x}} - 2 = 0$$

### **Вариант 6**

1. Найти корни полинома.

$$x^4 + 2x^3 - x - 1$$

2. Найти решение системы линейных уравнений

$$\begin{cases} 3x_1 + 2x_2 + x_3 = 5 \\ 2x_1 + 3x_2 + x_3 = 1 \\ 2x_1 + x_2 + 3x_3 = 11 \end{cases}$$

3. Найти значение локального минимума и максимума функции

$$f(x) = e^x \sin x^2 \\ \pi/2 \leq x \leq \pi$$

4. Вычислить значение определенного интеграла

$$\int_{0,4}^{1,2} \frac{dx}{\sqrt{2 + 0,5x^2}}$$

5. Решить трансцендентное уравнение

$$2\ln x - \frac{1}{x} + 0,5 = 0$$

### **Вариант 7**

1. Вычислить корни полинома

$$x^5 + 2x^2 - x + 11$$

2. Найти решение системы линейных уравнений

$$\begin{cases} x_1 + x_2 + 2x_3 = -1 \\ 2x_1 - x_2 + 2x_3 = -4 \\ 4x_1 + x_2 + 4x_3 = -2 \end{cases}$$

3. Найти значение локального минимума и максимума функции

$$\begin{aligned} f(x) &= 2\sin x^2 - x \\ 1 \leq x &\leq 3 \end{aligned}$$

4. Вычислить значение определенного интеграла

$$\int_{0,4}^{1,2} \frac{dx}{\sqrt{2 + 0,5x^2}}$$

5. Решить трансцендентное уравнение

$$x + \lg x - 0,5 = 0$$

### **Вариант 8**

1. Вычислить корни полинома

$$5x^3 + x - 4$$

2. Найти решение системы линейных уравнений

$$\begin{cases} 5x_1 + 8x_2 - x_3 = -7 \\ x_1 + 2x_2 + 3x_3 = 1 \\ 2x_1 - 3x_2 + 2x_3 = 9 \end{cases}$$

3. Найти значение локального минимума и максимума функции

$$f(x) = \sqrt{x+1}(\sin 2x + 3x^2)$$

$$1 \leq x \leq 2,5$$

4. Вычислить значение определенного интеграла

$$\int_{0,6}^{1,4} x^2 \cos x dx$$

5. Решить трансцендентное уравнение

$$e^x - e^{-x} - 2 = 0$$

### **Вариант 9**

1. Вычислить корни полинома

$$x^3 - 9x$$

2. Найти решение системы линейных уравнений

$$\begin{cases} 10x_1 + x_2 + x_3 = 12 \\ 2x_1 + 10x_2 + x_3 = 13 \\ 2x_1 + 2x_2 - 10x_3 = 14 \end{cases}$$

3. Найти значение локального минимума и максимума функции

$$f(x) = x^2 + 6e^{0,15x}$$

$$0 \leq x \leq 1$$

4. Вычислить значение определенного интеграла

$$\int_{0,4}^{1,2} \frac{\cos(x^2)}{x+1} dx$$

5. Решить трансцендентное уравнение

$$x^2 - \sin 5x = 0$$

### **Вариант 10**

1. Вычислить корни полинома

$$x^3 - 0,1x^2 + 0,4x - 1,5$$

2. Найти решение системы линейных уравнений

$$\begin{cases} 2,7x_1 + 0,9x_2 - 1,5x_3 = 3,5 \\ 4,5x_1 - 2,8x_2 + 6,7x_3 = 2,6 \\ 5,1x_1 + 3,7x_2 - 1,4x_3 = -0,14 \end{cases}$$

3. Найти значение локального минимума и максимума функции

$$f(x) = e^{-0,5x^2} - 0,5\cos x \\ 0,5 \leq x \leq 1,5$$

4. Вычислить значение определенного интеграла

$$\int_{1,6}^{2,4} (x+1)\sin x dx$$

5. Решить трансцендентное уравнение

$$1,8x^4 - \sin 10x = 0$$

### **ЛАБОРАТОРНАЯ РАБОТА №3**

#### **ПРОГРАММИРОВАНИЕ В СИСТЕМЕ MATLAB**

**Цель работы** – освоить операторы языка ML, научиться использовать их для решения ситуационных задач.

#### **Постановка задачи**

Написать скрипт-файл для решения задачи из области математики или физики.

## Варианты индивидуальных заданий

**Вариант 1.** В гелиоцентрической системе отсчета Земля движется по окружности радиуса  $R_1=1,496 \cdot 10^8$  км (период обращения  $T_1=3,156 \cdot 10^7$  с). Координаты Земли описываются зависимостями

$$X_3(t)=R_1 \cos\left(\frac{2\pi}{T_1}t+\varphi_0\right), \quad Y_3(t)=R_1 \sin\left(\frac{2\pi}{T_1}t+\varphi_0\right), \quad \varphi_0=0.$$

Луна, в свою очередь движется вокруг Земли по окружности радиусом  $R_2=3,844 \cdot 10^5$  км (период обращения  $T_2=2,36 \cdot 10^6$  с). Координаты Луны в геоцентрической системе координат:

$$X'_{\text{л}}(t)=R_2 \cos\left(\frac{2\pi}{T_2}t\right), \quad Y'_{\text{л}}(t)=R_2 \sin\left(\frac{2\pi}{T_2}t\right).$$

Построить орбиты Земли и Луны в гелиоцентрической системе координат. Промоделировать, как изменится картина при других значениях  $R_2$  и  $T_2$ , например при  $R_2=3,844 \cdot 10^7$  км,  $T_2=2,36 \cdot 10^5$  с.

**Вариант 2.** Получить амплитудно-частотную и фазочастотную характеристики (АЧХ и ФЧХ) цифрового рекурсивного фильтра  $N$ -го порядка:

$$A(\omega)=\left(\frac{A_{11}^2+A_{12}^2}{B_{11}^2+B_{12}^2}\right)^{1/2};$$

$$\varphi(\omega)=\arctg\left(\frac{A_{12}B_{11}-A_{11}B_{12}}{A_{11}B_{11}+A_{12}B_{12}}\right),$$

где  $A_{11}=\sum_{k=0}^N a_k \cos(k\omega T_d)$ ;  $A_{12}=\sum_{k=0}^N a_k \sin(k\omega T_d)$ ;  $B_{11}=\sum_{i=1}^N b_i \cos(i\omega T_d)$ ;

$B_{12}=\sum_{i=1}^N b_i \sin(i\omega T_d)$ ;  $a_k$  и  $b_i$  – весовые коэффициенты фильтра,

$T_d$  – период дискретизации. Принять  $T_d=0,001$  с,  $N=2$ ,  $\omega T_d$  – изменять от 0 до  $\pi$ .

Значения весовых коэффициентов вводить с клавиатуры:

$$a_0=1; \quad a_1=-2,208; \quad a_2=1,208; \quad b_1=-0,848; \quad b_2=0,36.$$

**Вариант 3.** В ультразвуковом дефектоскопе поисковый элемент (излучатель ультразвуковых импульсов и приемник отраженных сигналов) совершает движение из одного крайнего положения в другое и останавливается.

Математическая модель закона изменения ускорения:

$$g(t) = \begin{cases} g_m \sin(\pi \frac{t}{t_b}), & 0 \leq t \leq t_b, \\ 0, & t_b < t < t_n, \\ -g_m \sin(\pi \frac{t-t_n}{t_k-t_n}), & t_n \leq t \leq t_k. \end{cases}$$

Получить выражения для законов изменения скорости  $v(t)$  и пути  $S(t)$  и построить зависимости  $g(t)$ ,  $v(t)$ ,  $S(t)$  для следующих исходных данных:

$$t_b = 0,1 \text{ с}; \quad t_n = 0,9 \text{ с}; \quad t_k = 1,0 \text{ с}; \quad g_m = 10 \text{ м/с}^2.$$

**Вариант 4.** Построить траекторию спуска космического аппарата в трехмерном пространстве.

Законы изменения составляющих ускорения:

$$\begin{aligned} g_x(t) &= g_{xm} \cos(\pi \frac{t}{t_k}), \\ g_y(t) &= g_{ym} \cos(\pi \frac{t}{t_k}), \\ g_h(t) &= g_{hm} \cos(\pi \frac{t}{t_k}). \end{aligned}$$

Путем двойного интегрирования получить законы изменения координат  $x(t)$ ,  $y(t)$  и  $h(t)$  для следующих начальных условий:

$$\begin{aligned} x(0) &= y(0) = 0; & h(0) &= 250 \text{ км}; \\ v_x(0) &= v_y(0) = 8/\sqrt{2} \text{ км/с}; & v_h(0) &= 0. \end{aligned}$$

Исходные данные:

$g_{xm} = g_{ym} = g$ ;  $g_{hm} = -4g$ ;  $g = 9,8 \text{ м/с}^2$ ;  $t_k = 5 \text{ мин}$ ; шаг изменения времени  $\Delta t = 3 \text{ с}$ .

Воспользоваться оператором *plot3*, местоположение указывать зеленой звездочкой. С помощью операторов *line* показать проекции точек положения космического аппарата на горизонтальную и вертикальную плоскости.

**Вариант 5.** Построить траекторию стартующей ракеты в трехмерном пространстве. Законы изменения координат:

$$\left. \begin{aligned} x(t) &= x_0 + a_{x1} \frac{t^2}{2}, \\ y(t) &= y_0, \\ h(t) &= a_{h1} \frac{t^2}{2}, \end{aligned} \right\} \text{ при } 0 \leq t \leq 10;$$

$$\left. \begin{aligned} x(t) &= x(10) + a_{x2} \frac{t^2}{2}, \\ y(t) &= y_0 + a_y \frac{t^2}{2}, \\ h(t) &= a_{h2} \frac{t^2}{2} + h(10) \end{aligned} \right\} \text{ при } 10 < t \leq 50,$$

где  $x_0 = 5$  км;  $y_0 = 5$  км;  $a_{x1} = 20$  м/с<sup>2</sup>;  $a_{x2} = 10$  м/с<sup>2</sup>;  $a_{h1} = 40$  м/с<sup>2</sup>;  $a_y = 20$  м/с<sup>2</sup>;  $a_{h2} = 0,3a_y$ .

Воспользоваться оператором *plot3*, местоположение ракеты указывать красной звездочкой. С помощью операторов *line* показать проекции точек положения летательного аппарата на горизонтальную и вертикальную плоскости. Шаг изменения времени  $\Delta t = 1$  с.

**Вариант 6.** Изобразить интерференционную картину, получившуюся при освещении оранжевым светом с длиной волны  $\lambda = 0,6$  мкм плоской пластины с прижатыми к ней плосковыпуклыми линзами с радиусом кривизны выпуклой поверхности  $R = 5$  м.

Разность фаз интерферирующих волн:  $\varphi(r) = \frac{2\pi r^2}{\lambda R}$ , где  $r$  – расстояние до центра интерференционной картины.  $r = 0 \div 0,4 \cdot 10^{-2}$  м (шаг изменения  $\Delta r = 0,5 \cdot 10^{-4}$  м). Яркость  $I(r) = 2[1 + \cos(\varphi(r))]$ .

Построить график зависимости  $I(r)$  в декартовой и полярной системах координат. Для моделирования интерференционной картины ограничить  $r_{\max} = 7 \cdot 10^{-3}$  м и построить матрицу значений фазы  $\bar{F}$  и интенсивности отраженного света  $\bar{I}$  размером  $60 \times 60$ .

$$F_{ij} = \frac{2\pi(x_i^2 + y_j^2)}{\lambda R}; \quad I_{ij} = 2(1 + \cos(F_{ij})),$$

где координаты  $x_i$  и  $y_i$  изменяются так:

$$x_i = \frac{r_{\max}}{N} \cdot i - \frac{r_{\max}}{2}, \quad y_j = \frac{r_{\max}}{N} \cdot j - \frac{r_{\max}}{2}, \quad N=60.$$

Построить график зависимости интенсивности отраженного света от координат и карту линий одного уравнения (командой *Contour*).

**Вариант 7.** Биоритмы человека представляют собой синусоиды, выходящие из нуля в день рождения человека и имеющие периоды: интеллектуальный – 33 дня, эмоциональный – 28 дней, физический – 23 дня.

По введенной дате рождения человека построить графики его биоритмов на текущий месяц (или указанный). Выделить на нем текущий день (или указанный).

**Вариант 8.** Колесо электровоза, движущегося со скоростью  $v=10$  м/с, имеет радиус  $R=1$  м. Необходимо рассчитать и построить траекторию точки, лежащей на расстоянии  $r=0,5$  м от оси колеса. Считать, что в начальный момент времени точка находилась в самом нижнем положении. Кинематические уравнения движения точки:

$$x(t) = vt - r \sin \frac{vt}{R}, \quad y(t) = R - r \cos \frac{vt}{R}.$$

Построить график на интервале  $t=0 \div 2$  с. Указать на графике положения точки в моменты  $t_1=0,4$  с и  $t_2=0,8$  с. Изобразить вектора скорости движения точки для моментов  $t_1$  и  $t_2$ .

**Вариант 9.** Осуществить гармонический синтез пилообразного сигнала по первым 3, 6 и 15 гармоникам:

$$y(t) = \sum_i \frac{(-1)^{i+1}}{i} \sin(2\pi \frac{t}{T} i).$$

Для этого суммировать 3, 6 и 15 синусоидальных сигналов соответственно. Построить графики полученных сигналов при  $T=50$ ,  $t=0 \div T$ .



**Вариант 10.** В момент преодоления самолетом звукового барьера число Маха становится равным единице. Функция, определяющая число Маха:

$$M(v, H) = \sqrt{5 \left( \left( \left( \left( 1 + 0,2 \left( \frac{v}{a} \right)^2 \right)^{3,5} \right) - 1 \right) \left( 1 - b \frac{H}{c} \right)^d + 1 \right)^e - 1},$$

где  $v$  – скорость полета самолета;  $H$  – высота полета; коэффициенты:  $a = 1222,5$ ;  $b = 6,875 \cdot 10^{-6}$ ;  $c = 0,3048$ ;  $d = -5,2656$ ;  $e = 0,286$ .

Получить графики зависимости  $M(v)$  для  $H=500$ ;  $1000$ ;  $2000$ ;  $5000$ ; показать уровень  $M=1$ , соответствующий достижению скорости звука.

Из условия  $M(v, H)=1$  получить зависимость скорости преодоления звукового барьера  $v$  от высоты  $H$ . Для этого, изменяя  $H$  в диапазоне от 0 до  $2,5 \cdot 10^4$ , решать уравнение  $M(v, H) - 1 = 0$ . При решении уравнения передавать  $H$  в функцию, описывающую правую часть уравнения, как глобальное данное (командой *global H*). Построить график зависимости  $v(H)$ , при котором  $M(v, H) = 1$ .

(Для сведения: при  $H \approx 0$ ,  $v \approx 1200$  км/ч; при  $H \approx 2,5 \cdot 10^4$ ,  $v \approx 150$  км/ч).

По результатам работы должен быть составлен отчет, содержащий текст индивидуального задания, тексты *script*-файлов и файлов-функций, а также графическое представление результатов работы.

### Контрольные вопросы

1. Каково назначение системы Matlab?
2. В каких режимах может выполняться работа в системе ML?
3. В виде каких файлов хранится большинство команд и функций системы ML?
4. Опишите структуру окна рабочей среды ML.
5. Для чего в ML используют клавиши управления  $\uparrow$  и  $\downarrow$ ?
6. Что является элементарной единицей данных языка ML?
7. Как записываются действительные числа в ML?
8. Какими командами можно получить информацию о данных, хранящихся в рабочем пространстве?

9. Какие форматы вывода числовых данных в ML вы знаете?
10. Как изменить формат вывода результатов вычисления в ML?
11. Как в системе ML определяется тип переменных?
12. Назовите правила составления имен переменных.
13. Какие основные системные переменные существуют в ML?
14. Какие операции существуют в ML?
15. Назовите операции ML в порядке убывания приоритета.
16. Как представляются вектора и матрицы в ML?
17. Как записываются и чем различаются матричные и поэлементные операции в ML?
18. Назовите способы задания векторов в ML.
19. Как обратиться к элементам векторов и матриц?
20. Для чего используется функция `length()`?
21. Как обратиться к последнему элементу вектора?
22. Какими командами формируются особые матрицы?
23. Для чего используются символы двоеточие (`:`), точка с запятой (`;`) и многоточие (`...`)?
24. Для чего используются `[]`, `()`?
25. Какие функции используются для обработки векторов и матриц?
26. Какие команды используются для построения графиков функции одной переменной? В чем их различия?
27. Какие команды используются для построения графиков функции двух переменных? В чем их различия?
28. Каков порядок действий для построения графика функции вида  $y = f(x)$ ?
29. Каков порядок действий для построения графика функции вида  $z = f(x, y)$ ?
30. Как построить несколько графиков в одних координатных осях?
31. Как можно управлять внешним видом графика?
32. Каким образом можно вывести несколько графиков в разных координатных осях в одном окне?
33. Какие средства предоставляет система ML для построения диаграмм?
34. Как задается полином?
35. Чему равно число элементов в векторе, определяющем полином?
36. Что такое *m*-файлы?