

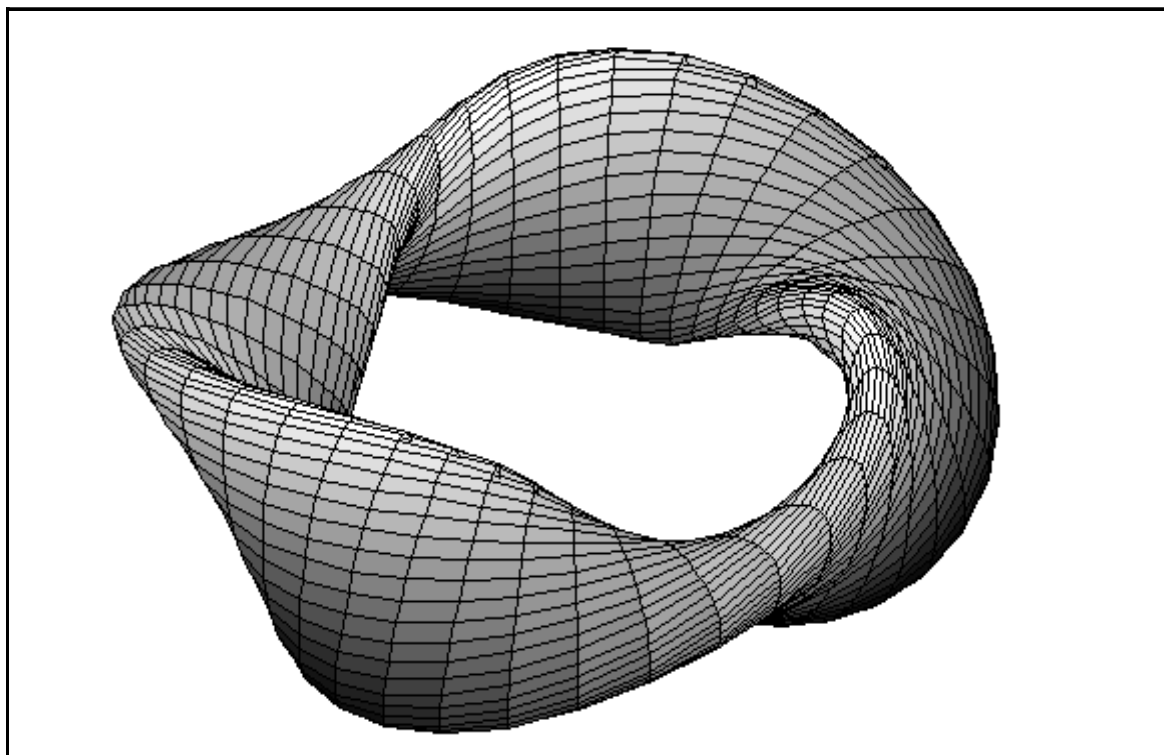


*Томский межвузовский центр  
дистанционного образования*

**М.В. Черкашин**

# **Система для математических и инженерных расчетов MATLAB**

**Учебное пособие**



**ТОМСК - 2002**

Министерство образования Российской Федерации

**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ  
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)**

**Кафедра компьютерные системы в управлении  
и проектировании (КСУП)**

**М.В. Черкашин**

# **Система для математических и инженерных расчетов MATLAB**

**Учебное пособие**

**2002**

Корректор: Красовская Е.Н.

**Черкашин М.В.**

Система для математических и инженерных расчетов MATLAB: Учебное пособие. – Томск: Томский межвузовский центр дистанционного образования, 2002. – 63 с.

© Черкашин М.В., 2002

© Томский межвузовский центр  
дистанционного образования, 2002

## СОДЕРЖАНИЕ

Введение .....	5
1 Основные сведения о системе MATLAB .....	5
2 Установка и состав системы MATLAB .....	6
3 Запуск системы, основные справочные и управляющие команды .....	7
4 Работа в режиме прямых вычислений .....	13
5 Рабочая область .....	21
6 Программирование в среде MATLAB .....	23
6.1 Введение .....	23
6.2 Создание М-файлов. М-сценарии. М-функции .....	24
6.3 Типы переменных .....	29
6.3 Операторы системы MATLAB. Объединение операторов в арифметические выражения .....	31
6.4 Ввод информации .....	41
6.5 Повышение эффективности обработки М-файлов .....	42
6.6 Работа с графическими средствами системы MATLAB .....	44
6.6.1 Двумерные графики .....	45
6.6.2 Оформление и комбинирование графиков .....	48
6.6.3 Специальные двумерные графики .....	53
6.6.4 Трехмерные графики функции двух переменных .....	56
7 Заключение .....	63
Список дополнительных источников .....	64

# СИСТЕМА ДЛЯ МАТЕМАТИЧЕСКИХ И ИНЖЕНЕРНЫХ РАСЧЕТОВ MATLAB

## Введение

Данное методическое руководство является кратким описанием известного пакета для математических и инженерных расчетов MATLAB. Цель – дать студентам навыки работы в этой системе, изучить основные приемы программирования на языке MATLAB. Кроме этого в данном руководстве содержится небольшой справочник по встроенным функциям системы.

Руководство предназначено для студентов специальностей 220300 и 210100, изучающих курсы «Вычислительные методы» и «Автоматизированное проектирование средств и систем управления».

## 1 Основные сведения о системе MATLAB

Система MATLAB (сокращение от MATrix LABoratory – МАТричная ЛАБоратория) разработана фирмой The MathWorks, Inc. (США, г. Нейтик, штат Массачусетс) и является интерактивной системой для выполнения инженерных и научных расчетов. Система MATLAB распространяется более 10 лет, она постоянно развивается: последняя версия – MATLAB 6.1 Release 12 for Windows.

MATLAB – это высокоэффективная среда для математических, инженерных, научных вычислений, ориентированная на работу с массивами данных (матрицами), чем и обязана своему названию. Система MATLAB может выполнять операции в режиме прямых вычислений (режим интерпретатора). Это позволяет использовать ее как мощный калькулятор, в котором наряду с обычными арифметическими и алгебраическими действиями, могут использоваться и операции матричной алгебры, т.е. операции по обращению матриц, нахождению собственных чисел и векторов, решению систем линейных уравнений и многое др.

Наиболее известные области применения системы MATLAB:

- математика и вычислительные методы;
- разработка алгоритмов;
- вычислительный эксперимент, математическое и имитационное моделирование, макетирование;
- анализ данных, визуализация и обработка данных;
- научная и инженерная графика;
- разработка приложений, связанных с вычислительными алгоритмами и визуализацией данных, включая разработку пользовательского графического интерфейса (GUI).

Система MATLAB – это одновременно и операционная среда, и язык программирования. Ее можно использовать как интерпретатор, в режиме непосредственных вычислений, и как компилятор – для написания отдельных

программ. Но главное достоинство среды MATLAB – это легкость ее модификации и адаптации к самым различным задачам, требующим математических вычислений, обработки данных и работы с графическими средствами. Поэтому ее с равным успехом можно применять для расчетов в математике, физике, биологии, электро- и радиотехнике, для статистических исследований и др.

## 2 Установка и состав системы MATLAB

Рассмотрим версию системы MATLAB 4.x for Windows, как наиболее удобную для использования в учебном процессе. Она обладает достаточно мощными средствами по обработке и визуализации данных, содержит большое число прикладных программ, для реализации самых разнообразных вычислительных алгоритмов. При этом она занимает не слишком много места на жестком диске (около 12 МБайт) и работает даже в среде Windows 3.x., что позволяет ставить ее даже на ПК с процессором класса x386. Более поздние версии системы (MATLAB 5.x, MATLAB 6.x), хотя и обладают гораздо лучшими показателями, как со стороны вычислительных методов, так и по возможностям пользователя при создании собственных программ, но в тоже время требуют серьезных ресурсов от ПК (не менее 85 Мбайт на жестком диске и процессор не ниже Pentium 166). В целом же MATLAB является совместимым сверху вниз, т.е. современные версии системы «понимают» программы написанные для более ранних версий.

Исходный дистрибутив системы занимает 5 дискет размером 3,5". Установка системы на жесткий диск не представляет трудности для пользователя, знакомого с операционной системой Windows. В ходе установки следует указать диск и имя корневого каталога для системы MATLAB (по умолчанию система будет установлена в каталог **c:\MATLAB**). Также при установке возможен запрос на подключение дополнительных пакетов прикладных программ (ППП) – например, SIMULINK или др. Различные ППП для MATLAB распространяются дополнительно и могут быть подключены к системе позже. В ходе инсталляции будет создана группа (в главном меню Windows доступном после нажатия на кнопку **ПУСК**) с ярлыками для запуска системы MATLAB и вызова системы помощи.

После установки на жесткий диск система MATLAB 4.2 имеет следующую структуру: в каталоге **c:\MATLAB\BIN** содержатся основные системные файлы; каталог **c:\MATLAB\TOOLBOX** содержит встроенные функции системы (script-файлы) и установленные ППП; каталог **c:\MATLAB\EXTERN** содержит исходные файлы и библиотеки на языках программирования C и FORTRAN для создания MEX-файлов; каталог **c:\MATLAB\GHOSTSR** содержит программу GhostScript и векторные ps-шрифты – данная программа представляет собой графический постпроцессор и используется для качественной печати графических изображений на внешних устройствах (лазерных принтерах, графопостроителях и т.д.) или в файл формата \*.ps (postscript).

В корневом каталоге **c:\MATLAB** должны быть два файла с системными установками: **matlabrc.m** – содержит основные установки для командного окна системы (MATLAB Command Window), которое является корневым объектом системы и **printopt.m** – содержит установки для текущего принтера. Кроме этого пользователь может создать в корневом каталоге файл **startup.m** с индивидуальными установками, который будет автоматически выполняться системой при каждой загрузке (аналог файла **autoexec.bat** в системе MS-DOS).

### 3 Запуск системы, основные справочные и управляющие команды

После запуска файла **matlab.exe** на экране ПК появляется главное окно MATLAB Command Window (см. рис. 1), которое является корневым (главным) объектом системы.

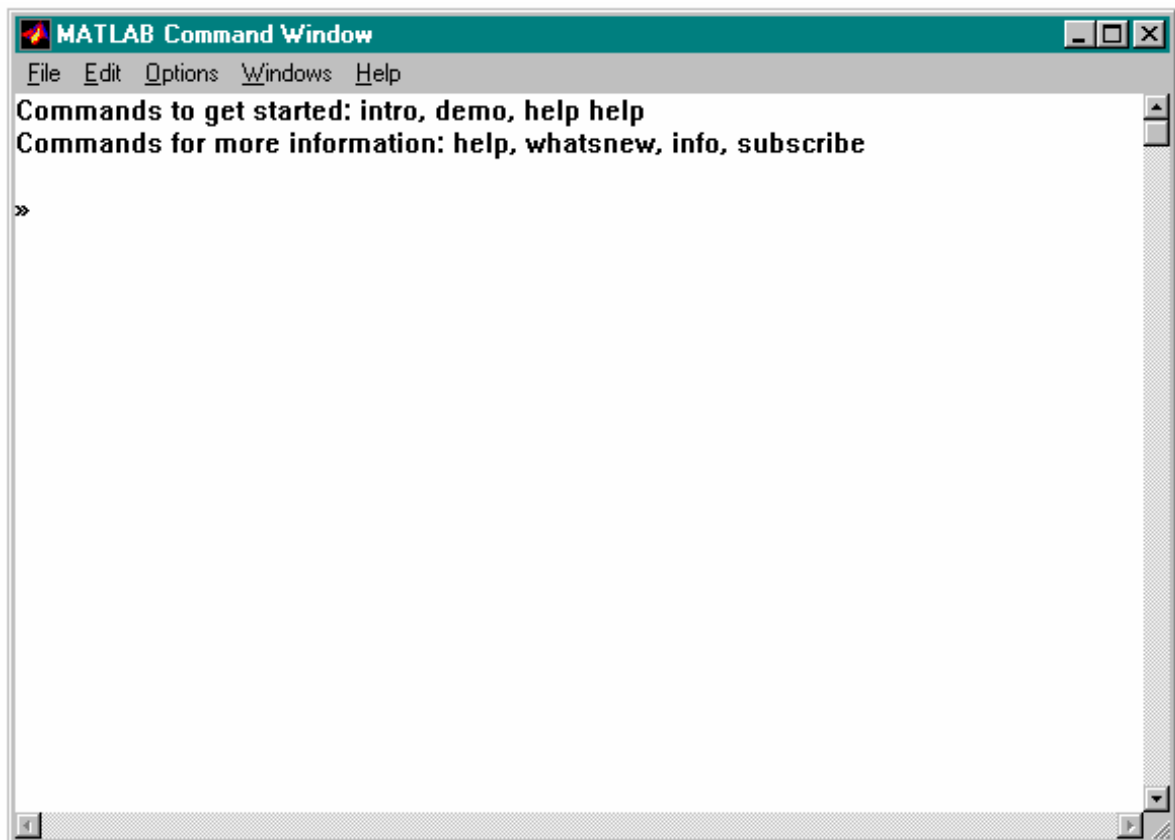


Рисунок 1 – Вид командного окна системы MATLAB 4.2

В командной строке главного окна можно непосредственно вводить команды, выполнять вычисления или вызывать программы (script-файлы), написанные на языке системы MATLAB.

При загрузке системы выводится подсказка вида  
 commands to get started: intro, demo, help help  
 commands for more information: help, whatsnew,  
 info, subscribe

которая содержит команды для первого знакомства с MATLAB.

Команда `intro`, вызывает программу, демонстрирующей основные возможности системы MATLAB. Команда `demo` – более полная демонстрация возможностей системы и подключенных прикладных пакетов с использованием средств GUI-интерфейса. Команда `help` – вызов встроенной справочной системы MATLAB по функциям и командам системы. Команды `whatsnew`, `info` предназначены для вывода более полной информации о текущей версии системы MATLAB. Команда `subscribe` служит для создания файла `subscribe.log` для подписки на услуги фирмы The MathWorks, Inc.

Вид окна, которое появляется после вызова команды `demo`, показан на рис. 2. Далее можно посмотреть примеры работы с самой системой MATLAB и прикладными пакетами, входящих в ее состав (если они установлены).



Рисунок 2 – Вид окна для демонстрации возможностей системы MATLAB и прикладных программ (программа `expo.m`)

В число демонстрационных примеров входят векторные и матричные операции, построение различных графиков (в том числе трехмерных фигур), реализация численных методов, спектральный анализ, расчет фильтров и т.д. Студентам рекомендуется перед началом работы с MATLAB внимательно просмотреть все примеры. Они прекрасно иллюстрируют возможности и разнообразие применений системы, ее высокую скорость вычислений.



Далее рассмотрим некоторые наиболее часто используемые управляющие команды и функции языка MATLAB.

**ver**

### Используемая версия системы MATLAB и ППП

*Синтаксис:* ver

ver < *имя ППП* >

Команда ver выводит на экран номера текущей версии системы MATLAB и подключенных ППП.

Команда ver < *имя ППП* > выводит на экран информацию о текущей версии запрошенного ППП.

*См. также:* version, readme, help, whatsnew, info

**version**

### Используемая версия системы MATLAB

*Синтаксис:* version

Команда version возвращает строку с указанием текущей версии системы MATLAB.

*См. также:* ver, readme, help, whatsnew, info

**help**

### Справка о командах и функциях системы MATLAB

*Синтаксис:* help

help < *раздел/функция* >

help < *спец. символ* >

Команда help без параметров выводит на экран список разделов системы MATLAB. Каждому разделу соответствует имя каталога в списке путей доступа MATLABPATH (команда path).

Команда help < *раздел/функция* > выводит перечень функций или описание самой функции. При вызове команды help следует указывать лишь имя раздела или функции.

Команда help < *спец. символ* > выводит список специальных символов системы.

*См. также:* lookfor, what, which, dir, pwd

**path**

### Управление списком путей доступа системы MATLAB

*Синтаксис:* path

p=path

path(p)

path(p1,p2)

Команда path выводит на экран список путей доступа в системе MATLAB. Этот список соответствует строковой переменной MATLABPATH, которая устанавливается в файлах matlabrc.m или/i startup.m.

Команда p=path возвращает строку, содержащую список путей доступа.

Команда `path(p)` заменяет текущий список (переменную `MATLABPATH`) списком `p`.

Команда `path(p1,p2)` объединяет списки `p1` и `p2` в один и заменяет им текущий список путей доступа.

*См. также:* `what`, `dir`, `cd`, `pwd`

`quit, exit`

### Завершение работы и выход из системы

*Синтаксис:* `quit`  
`exit`

Команда `quit` или `exit` завершает работу системы MATLAB и закрывает командное окно.

При завершении работы выводится информация о том, какое число операций с плавающей запятой было выполнено за сеанс работы. При выходе из системы все текущие переменные, находящиеся в рабочей памяти, стираются. При необходимости рабочую область памяти можно сохранить командой `save`.

`who, whos`

### Вывод списка текущих переменных

*Синтаксис:* `who`  
`whos`

Команда `who` выводит список переменных текущей рабочей области памяти.

Команда `whos` выводит подробную информацию относительно текущих переменных, включая имя, размер и число элементов используемых массивов, длину в байтах, тип матрицы (плотная/разряженная, комплексная/действительная).

`clear`

### Очистка рабочей области памяти

*Синтаксис:* `clear`  
`clear all`  
`clear <список имен переменных/функций>`  
`clear global`  
`clear global <имя глобальной переменной>`  
`clear functions`

Команда `clear` удаляет все переменные из рабочей области памяти.

Команда `clear all` удаляет все переменные, функции и ссылки на MEX-файлы из рабочей области памяти.

Команда `clear x1, x2` удаляет переменные или функции с именами `x1` и `x2` из текущей рабочей области памяти. Если `x1` глобальная переменная, то команда `clear x1` удаляет ее из текущей рабочей области памяти, но оставляет ее доступной для функций, где эта переменная объявлена глобальной.

Команда `clear global` удаляет все глобальные переменные из рабочей области памяти.

Команда `clear global X` удаляет глобальную переменную `X` из рабочей области памяти.

Команда `clear functions` удаляет все используемые М-функции из рабочей области памяти.

**type**

### Вывод на экран содержимого текстового файла

*Синтаксис:* `type <имя файла .расширение>`

Команда `type <имя файла .расширение>` выводит на экран командного окна содержимое текстового файла.

Команда `type <имя файла>` выводит на экран содержимое М-файла (с расширением `*.m`).

**disp**

### Вывод на экран переменных и текста

*Синтаксис:* `disp(<переменная> / '<текст>')`

Команда `disp(X)` выводит на экран командного окна содержимое переменной `X` без указания ее имени.

Команда `disp('<текст>')` выводит на экран символьную строку `'<текст>'`.

После каждой команды `disp` происходит переход на новую строку.

**matlabroot**

### Корневой каталог системы MATLAB

*Синтаксис:* `p=matlabroot`

Команда `matlabroot` возвращает имя корневого каталога системы MATLAB.

*См. также:* `cd`, `dir`, `path`, `pwd`

**pwd**

### Текущий каталог системы MATLAB

*Синтаксис:* `p=pwd`

Команда `pwd` возвращает имя текущего каталога при работе в системе MATLAB.

*См. также:* `cd`, `dir`, `path`, `matlabroot`

**cd**

### Просмотр и смена текущего каталога

*Синтаксис:* `cd`

`cd ..`

`cd <имя нового каталога>`

Команда `cd` выводит на экран путь доступа к текущему каталогу.

Команда `cd ..` переход на единицу вверх по дереву каталогов.

Команда `cd <имя нового каталога>` изменяет текущий каталог на новый, определенный строкой `<имя нового каталога>`.

*См. также:* matlabroot, dir, path, pwd, what

**dir**

### Просмотр содержимого каталога

*Синтаксис:* dir

dir <имя каталога>

Команда dir выводит листинг текущего каталога.

Команда dir <имя каталога> выводит список файлов указанного каталога. Можно указывать тип файлов или путь доступа к каталогу.

*См. также:* matlabroot, cd, path, pwd, what

**what**

### Вывод списка файлов с расширениями M, MAT и MEX

*Синтаксис:* what

what <имя каталога>

Команда what выводит на экран списки M-, MAT- и MEX-файлов текущего каталога.

Команда what <имя каталога> выводит списки файлов указанного каталога. Путь доступа к данному каталогу должен быть описан в переменной MATLABPATH. При вызове команды what <имя каталога> можно не указывать полный путь доступа к каталогу <имя каталога>.

*См. также:* matlabroot, cd, path, pwd, dir

**!**

### Вызов команды ОС

*Синтаксис:* ! <команда ОС>

Команда ! <команда ОС> позволяет выполнять команду ОС из командного окна системы MATLAB.

*Пример:* Команда ! notepad.exe запускает программу notepad.exe (стандартный блокнот ОС Windows 95/98) непосредственно из командного окна MATLAB.

**clc**

### Очистка командного окна

Команда clc очищает командное окно системы и устанавливает курсор в верхний левый угол.

*См. также:* home

**home**

### Возвращение курсора

Команда home устанавливает курсор в верхний левый угол.

*См. также:* clc

**echo**

### Переключение режима вывода на экран содержимого script-файлов

*Синтаксис:* echo

```
echo on/off
echo on all/off all
```

Команда `echo` управляет режимом вывода на экран (переключает) содержимого файла-сценария при выполнении.

Команда `echo on` включает режим вывода на экран текста script-файла.

Команда `echo on all` включает режим вывода на экран текста script-файла и всех М-функций, вызываемых в этом файле.

Команда `echo off` выключает режим вывода на экран текста script-файла.

Команда `echo off all` включает режим вывода на экран текста script-файла и М-функций.

#### 4 Работа в режиме прямых вычислений

Система MATLAB создана таким образом, что любые (подчас весьма сложные вычисления) можно выполнять в режиме прямых вычислений, т.е. без программы. Это превращает MATLAB в необычайно мощный калькулятор, который способен производить не только обычные для калькуляторов вычисления (например, выполнять арифметические операции и вычислять некоторые элементарные функции), но и операции с векторами и матрицами, комплексными числами, с рядами и многочленами. Можно почти мгновенно задать и вывести графики различных функций – от простой синусоиды до сложной трехмерной фигуры.

Работа с системой в режиме прямых вычислений носит диалоговый характер. Пользователь набирает на клавиатуре вычисляемое выражение, редактирует его (если нужно) и завершает ввод нажатием клавиши ENTER. При этом действует простейший строчный редактор. Перечислим его команды:

Комбинация клавиш	Назначение
→	перемещение курсора вправо на один символ
←	перемещение курсора влево на один символ
Ctrl →	перемещение курсора вправо на одно слово
Ctrl ←	перемещение курсора влево на одно слово
Home	перемещение курсора в начало строки
End	перемещение курсора в конец строки
↑ и ↓	перелистывание строк вверх или вниз
Del	стирание символа, на котором установлен курсор
Back_Space	стирание символа слева от курсора
Ins	включение/выключение режима вставки

Возможность перемещения курсора по словам в командной строке полезна при редактировании сложных выражений. Для вызова предыдущей команды следует использовать клавиши  $\uparrow$  или  $\downarrow$ , которые позволяют просматривать историю вводимых команд.

В некоторых случаях вводимое математическое выражение может оказаться настолько длинным, что для него не хватит одной строки - 80 символов. В этом случае часть выражения можно перенести на новую строку с помощью знака многоточия ... (три или более точек).

*Например:*  $s = 1 - 1/2 * 1/3 - 1/4 * 1/5 - 1/6 * 1/7 \dots$   
 $- 1/8 * 1/9 - 1/10 * 1/11 - 1/12;$

Иногда в ходе вывода данных вычислений появляется сокращение NaN (от слов Not a Number - не число). Оно обозначает операцию неопределенности, например вида 0/0 или Inf/Inf, где Inf—системная переменная со значением машинной бесконечности. Могут появляться и различные сообщения об ошибках (на английском языке). Например, при делении на 0 конечного числа появляется сообщение Warning: Devide by Zero. (Предупреждение: Деление на ноль). Весь диапазон представления чисел в системе лежит от  $10^{-308}$  до  $10^{308}$ .

Система MATLAB содержит несколько системных переменных: pi - число  $\pi \approx 3,1415926\dots$ ; inf - значение машинной бесконечности; ans - переменная, хранящая результат последней операции и обычно вызывающая его отображение на экране дисплея. Эти переменные можно использовать в математических выражениях.

Начнем с простейших вычислений. Для вычисления математических выражений достаточно набрать их по общепринятым правилам (как на Бейсике или Паскале) и завершить ввод нажатием клавиши ENTER. Значение вычисленного выражения будет присвоено переменной ans и выведено на экран дисплея.

*Пример:*

```
» 2*3-1
ans =
    5

» 2*sin(1)
ans =
    1.6829

» 10*(1-exp(-2))
ans =
    8.6466
```

В системе MATLAB можно задавать переменные. Для этого используется операция присваивания, вводимая знаком равенства (=) в следующем виде:

<Имя\_переменной> = <Выражение> [;]

Знак ; (точка с запятой) в конце выражения необязателен, однако играет важную роль. Он определяет открытые и закрытые команды системы. Если знак ; в конце выражения не стоит, то выражение является открытым и система MATLAB после вычисления (после нажатия на клавишу ENTER) присваивает результат вычислений переменной ans и выводит его на экран командного окна. В противном случае, когда мы поставим в конце выражения ;, вывода результата на экран не будет. При написании программ (script-файлов и функций) на языке MATLAB все выражения рекомендуется закрывать ;, чтобы предотвратить лишний вывод информации в командном окне.

Имя переменной может содержать сколько угодно символов, но запоминаются и идентифицируются только 19 первых символов. Имя переменной не должно совпадать с именами функций и процедур системы. Переменные могут быть обычными и индексированными, т.е. элементами векторов или матриц (см. далее). Могут использоваться и символьные переменные (строки), причем символьные значения заключаются в апострофы.

*Например:* `txt='Demo';`

Символьная переменная при этом рассматривается как вектор, состоящий из отдельных символов.

В арифметических выражениях можно использовать следующие знаки арифметических операции: + (сложение); - (вычитание); \* (умножение); / (деление слева направо); \ (деление справа налево); ^ (возведение в степень) и др. Полный список операторов можно получить, используя справочную систему MATLAB командой `help ops`.

В выражениях также можно использовать и доступные системе функции (алгебраические, тригонометрические, обратные тригонометрические, гиперболические, обратные гиперболические и др.). Перечень некоторых математических функций представлен в приложении.

Система MATLAB работает как с действительными, так и с комплексными числами вида  $z = \text{Re}(z) + i \cdot \text{Im}(z)$ ,

где  $i$  (или  $j$ ) - мнимая единица, т.е. квадратный корень из  $-1$ ,  $\text{Re}(z)$  - действительная часть комплексного числа  $z$ , а  $\text{Im}(z)$  - его мнимая часть. Тогда, если  $\text{Re}(z)=2$ , а  $\text{Im}(z) = 3.5$ , то число  $z$  можно задать в виде  $z = 2 + 3.5 \cdot i$  или  $z = 2 + 3.5 \cdot j$ .

Функция `real(z)` возвращает действительную часть комплексного числа  $\text{Re}(z)$ , а функция `imag(z)` - мнимую  $\text{Im}(z)$ . Для получения модуля комплексного числа используется функция `abs(z)`, а для вычисления аргумента - `angle(z)`.

MATLAB - система, специально предназначенная для проведения сложных вычислений с векторами, матрицами и многочленами. При этом она по умолчанию предполагает, что каждая заданная переменная - это вектор

или матрица. Все определяется конкретным значением переменной. Например, если задано  $X = 1$ , то это значит, что  $X$  есть вектор с единственным элементом, имеющим значение 1. Если надо задать вектор из трех элементов, то их значения надо перечислить в квадратных скобках, разделяя пробелами.

*Например, ввод* »  $V = [1 \ 2 \ 3]$

$V =$

1          2          3

задает вектор  $V$ , имеющий три элемента со значениями 1, 2 и 3. После ввода вектора система выводит его на экран дисплея.

Задание матрицы требует указания различных строк. Для различения строк также используется знак ; (точка с запятой).

*Например:*

»  $A = [ \ 1 \ 2 \ 3; \ 4 \ 5 \ 6; \ 7 \ 8 \ 9 ]$

$A =$

1	2	3
4	5	6
7	8	9

Мы задаем квадратную матрицу  $A$ , которую потом можно вывести на экран, набрав в командной строке ее имя  $A$  и нажав клавишу ввода ENTER. Матрицы можно вводить и по-другому (при вводе строки нажимается клавиша ENTER):

»  $A = [ \ 1 \ 2 \ 3$   
4 5 6  
7 8 9 ] ;

Такой способ может оказаться предпочтительным при вводе больших матриц.

Возможен ввод элементов матриц и векторов в виде арифметических выражений, содержащих любые доступные системе функции.

*Пример:*

$V = [2*2/(3*4) \ \exp(5) \ \sqrt{10}] ;$

»  $V$

$V =$

0.3333    148.4132    3.1623

Для указания отдельного элемента вектора или матрицы используются выражения вида  $V(i)$  или  $M(i, j)$ . Например, легко определить значение элемента матрицы  $A$ , стоящего на месте (2, 2):

»  $A(2, 2)$

ans =

5

он равен 5. Если нужно присвоить элементу  $A(i, j)$  новое значение  $x$ , то нужно использовать выражение



$$A(i, j) = x$$

Например, если элементу  $A(2, 2)$  надо присвоить значение 10, следует записать

$$A(2, 2) = 10$$

Выражение  $A(i)$  с одним индексом дает доступ к элементам матрицы, развернутым в один столбец. Такая матрица образуется из исходной, если подряд выписать ее столбцы. Следующий пример поясняет такой доступ к элементам матрицы  $A$ :

```
» A=[1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

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

```
» A(2)
```

```
ans =
```

```
    4
```

```
» A(8)
```

```
ans =
```

```
    6
```

```
» A(5)=100
```

```
A =
```

```
    1    2    3
    4   100    6
    7    8    9
```

Возможно задание векторов и матриц с комплексными элементами.

*Пример:*

```
» CM=[ [1 2; 3 4] + i*[5 6; 7 8] ]
```

```
CM =
```

```
1.0000 + 5.0000i    2.0000 + 6.0000i
3.0000 + 7.0000i    4.0000 + 8.0000i
```

или по-другому

```
» CM=[ 1+5i 2+6i; 3+7i 4+8i]
```

```
CM =
```

```
1.0000 + 5.0000i    2.0000 + 6.0000i
3.0000 + 7.0000i    4.0000 + 8.0000i
```

Наряду с операциями над отдельными элементами матриц векторов система позволяет производить операции умножения, деления и возведения в степень сразу над всеми элементами, т.е. над массивами. Для этого перед

операцией ставится точка. Например, оператор `*` означает умножение для векторов или матриц (по правилам матричной алгебры), а оператор `.*` - по-членное умножение всех элементов массива. Таким образом, если `M` - матрица, то `M.*2` даст матрицу, все элементы которой умножены на число 2. Необходимо также помнить, что при математических операциях с матрицами и векторами, их размеры должны соответствовать друг другу.

*Пример:*

```
» A = [1 2; 4 5]
```

```
A =
```

```
1    2
4    5
```

```
» B=[ 1 0; 0 1]
```

```
B =
```

```
1    0
0    1
```

```
» A*B
```

```
ans =
```

```
1    2
4    5
```

```
» A.*B
```

```
ans =
```

```
1    0
0    5
```

**Объединение малых матриц в большую.** Описанный способ задания матриц позволяет выполнить операцию объединения малых матриц в одну большую. Например, создадим вначале матрицу размером 3x3:

```
» A=magic(3)
```

```
A =
```

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

Теперь можно построить матрицу `B`, содержащую четыре матрицы:

```
» B=[A A+16; A+32 A.*3]
```

```
B =
```

```
8    1    6    24    17    22
3    5    7    19    21    23
4    9    2    20    25    18
40   33   38   24    3    18
35   37   39    9   15   21
36   41   34   12   27    6
```

Полученная матрица имеет уже размер 6x6.

**Применение оператора : (двоеточие).** Очень часто необходимо произвести формирование упорядоченных числовых последовательностей. Такие последовательности нужны например для создания векторов или значений абсциссы при построении графиков. Для этого в MATLAB используется оператор : (двоеточие):

*Синтаксис:* <Начальное\_значение>:[<Шаг>]:<Конечное\_значение>

Данная конструкция порождает возрастающую последовательность чисел, которая начинается с <Начального\_значения>, идет с заданным <Шагом> и завершается <Конечным\_значением>. Если <Шаг> не задан, по умолчанию он принимается равным 1. Примеры применения оператора : даны ниже.

*Пример:*

```
» 1:5
ans =
     1     2     3     4     5
» i=0:2:10
i =
     0     2     4     6     8    10
» j=10:-2:1
j =
    10     8     6     4     2
» V=0:pi/2:2*pi
V =
     0  1.5708  3.1416  4.7124  6.2832
» 5:-1:0
ans =
     5     4     3     2     1     0
```

Выражения с оператором : могут использоваться в качестве аргументов функций для получения множества их значений. Например, вычисление функции  $\sin(x)$  в диапазоне от 0 до 3 с шагом 0,5:

```
» sin(0:0.5:3)
ans =
     0  0.4794  0.8415  0.9975  0.9093  0.5985  0.1411
```

Оператор : часто используется для вывода только части матрицы на экран или применения в математических операциях.

*Пример:*

```
A=[1 2 3 4;5 6 7 8; 9 10 11 12 ; 13 14 15 16]
A =
     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16
```

вывод подматрицы, состоящей из элементов 1, 2 и 3 строки и 2, 3 и 4 столбцов матрицы A.

```
» A(1:3, 2:4)
```

```
ans =
```

```
     2     3     4
     6     7     8
    10    11    12
```

вывод только 1 и 3 строк матрицы A (вектор номеров строк указан с шагом 2)

```
» A(1:2:4, :) % -
```

```
ans =
```

```
     1     2     3     4
     9    10    11    12
```

Для формирования матриц и выполнения ряда матричных операций возникает необходимость удаления отдельных столбцов или строк матрицы. Для этого используется пустые квадратные скобки [] (так называемая «пустая матрица»). Проделаем это над матрицей A:

```
» A=[1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

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

Удалим второй столбец, используя оператор : (двоеточие):

```
» A(:, 2) = []
```

```
A =
```

```
     1     3
     4     6
     7     9
```

A теперь удалим вторую строку:

```
» A(2, :) = []
```

```
A =
```

```
     1     3
     7     9
```

Помимо встроенных в систему операторов и функций могут использоваться и внешние программы, т.е. написанные пользователем файлы с расширением \*.m. Можно как воспользоваться почти 150 встроенными функциями, уже включенными в поставку системы, так и создать свои собственные. В итоге MATLAB оказывается расширяемой системой, легко адаптируемой под решение конкретных математических задач, интересующих пользователя.

В системе MATLAB внешние программы (написанные пользователем) используются точно так же, как и встроенные функции и операторы. Никаких особых указаний по их применению делать не надо. Разумеется, скорость вычислений по внешним программам несколько ниже, чем по встроенным функциям или операторам. При этом вычисления происходят следующим об-

разом: вначале система быстро определяет, есть ли слово среди служебных слов системы. Если оно есть, то нужные вычисления выполняются сразу; если нет, то система ищет m-файл с соответствующим именем на диске. Если файла нет, то выдается сообщение об ошибке. Если файл найден, он загружается в память (компилируется) и выполняется. Список доступных m-файлов в текущем каталоге выводится с помощью команды `what`.

## 5 Рабочая область

*Рабочая область* системы MATLAB - это область памяти, в которой размещены переменные системы. Содержимое этой области можно просмотреть из командной строки с помощью команд `who` и `whos`. Команда `who` выводит только имена переменных, а команда `whos` – еще и дополнительную информацию о размерах массивов и типе переменной.

**Загрузка и сохранение рабочей области.** Команды `save` и `load` позволяют в любой момент времени сохранить содержимое рабочей области или загрузить новые данные в процессе сеанса работы. С помощью этих команд можно также осуществлять экспорт и импорт ASCII-файлов.

**Сохранение переменных рабочей области.** Команда `save` позволяет сохранить содержимое рабочей области в двоичном MAT-файле (он имеет расширение `*.mat`), который можно в дальнейшем вызвать командой `load`.

**Спецификация формата файла.** Для того чтобы управлять форматами файлов, следует в команде `save` в дополнение к имени файла и списку переменных использовать следующие флаги:

Флаг	Пояснение
<code>-mat</code>	Двоичный MAT-файл (по умолчанию)
<code>-ascii</code>	ASCII-формат (8 цифр)
<code>-ascii –double</code>	ASCII-формат (16 цифр)
<code>-ascii –double -tabs</code>	Формат с разделителями и метками табуляции
<code>-append</code>	Добавить данные к существующему MAT-файлу

*Синтаксис:* `save< имя_файла . [ расширение ] >список_переменных -флаг`

*Пример:*

`save` – сохранение всех переменных (содержимого рабочей области) в файл `matlab.mat` в текущем каталоге;

`save aaa` – сохранение содержимого рабочей области в файл `aaa.mat` в формате системы MATLAB;

`save my.txt a -ASCII` - сохранение переменной `a` в файл `my.txt` в виде ASCII-кодов (текстовый формат);

`save my a1, a2, a3 -ASCII` - сохранение переменных `a1`, `a2`, `a3` в файл `my.mat` в двоичном формате.

Когда содержимое рабочей области сохраняется в ASCII-формате, то рекомендуется единовременно сохранять только одну переменную. Если сохраняется более одной переменной, то система MATLAB создаст файл ASCII-файл, который нельзя будет в дальнейшем загрузить в MATLAB, используя команду `load`. По умолчанию, если пользователь не указывает расширение в имени файла и не устанавливает флаг, система присваивает файлу расширение `< имя_файла >.mat`.

**Загрузка рабочей области.** Команда `load` позволяет загрузить MAT-файл, который был ранее сохранен с помощью команды `save`. При загрузке MAT-файла новые значения одноименных переменных будут записаны взамен старых. Если MAT-файл имеет расширение, отличающееся от `*.mat`, то необходимо использовать флаг `-mat`; в противном случае MATLAB будет считать форматом файла ASCII-формат.

**Загрузка файлов данных в ASCII-формате.** Команда `load` позволяет также выполнять импорт файлов данных в ASCII-формате; она преобразует содержимое файла в переменную с именем файла только без расширения. Например, применение команды `load tides.dat` создает в рабочей области системы MATLAB переменную с именем `tides`. Если исходный файл в ASCII-формате имеет `m` строк с `n` значениями в каждой строке, то результатом будет массив чисел размера `mхn`.

*Синтаксис:* `load < имя_файла . [ расширение ] > -флаг`

*Пример:*

`load` – загрузка файла `matlab.mat` из текущего каталога;  
`load aaa` – загрузка файла `aaa.mat` в формате системы MATLAB;  
`load aaa.txt -mat` – загрузка файла `aaa.txt` в формате системы MATLAB;

`load my.txt` – загрузка файла `my.txt` в текстовом формате (при этом содержимое файла будет помещено в переменную (матрицу) с именем `my`)

*Замечание:* Для сохранения или загрузки последовательности файлов, имена которых имеют общий корень и дополнительный целочисленный суффикс, необходимо использовать структуру цикла.

Например, следующая конструкция позволяет сохранить квадраты чисел от 1 до 10 в файлах с именами `data1`, ..., `data10`:

```
file = 'data';
for i = 1:10,
    j = i.^2;
    save([file int2str(i)], 'j');
end
```

Команды `load` и `save` допускают использование группового символа `*` в качестве замены ряда символов в шаблоне имени переменной.

Например, команда `save rundate x*` сохраняет все переменные, имена которых начинаются с символа `x` в файле с именем `rundata.mat`.

Точно также команда `load testdata ex1*95` загружает все переменные, имена которых начинаются с символов 'ex1' и заканчиваются символами '95', независимо от того, какие символы размещены между ними.

**Удаление переменных из рабочей области.** В памяти компьютера введенные переменные занимают определенное место. Использование матриц и векторов больших размеров при сложных вычислениях может привести к нехватке памяти. Поэтому возникает необходимость удаления из памяти (рабочего пространства) MATLAB ненужных переменных. Для очистки рабочего пространства используется функция `clear` в разных формах (см. ранее гл.3).

Стертая из рабочего пространства переменная становится недоступной для использования. Если необходимо лишь очистить переменную от значений, не удаляя ее из рабочего пространства, следует использовать «пустую матрицу», т.е. использовать выражения вида `A=[]`. В этом случае имя переменной сохраняется и в дальнейшем ее можно использовать для вычислений.

## 6 Программирование в среде MATLAB

### 6.1 Введение

Файлы, которые содержат команды языка MATLAB, называются М-файлами. Для создания М-файла используется любой текстовый редактор; вызову М-файла предшествует присваивание значений входным аргументам; результатом является значение выходной переменной. Таким образом, вся процедура включает две операции:

- 1) Создать М-файл, используя текстовый редактор:

```
function c = myfile(a, b)
    c = sqrt((a.^2)+(b.^2))
```

- 2) Вызвать М-файл из командной строки или из другого М-файла:

```
» a=7.5;
» b=3.342;
» c=myfile(a,b)
c =
    8.2109
```

**Типы М-файлов.** Существует два типа М-файлов: М-сценарии (М-script) и М-функции (М-function) со следующими характеристиками

М-сценарий	М-функция
Не использует входных и выходных аргументов	Использует входные и выходные аргументы
Оперирует с данными из рабочей области	По умолчанию, внутренние переменные являются локальными по отношению к функции
Предназначен для автоматизации последовательности шагов, которые нужно выполнять много раз	Предназначена для расширения возможностей языка MATLAB (библиотеки функций, пакеты прикладных программ)

**Структура М-файла.** М-файл, оформленный в виде функции состоит из следующих компонентов

Строка определения функции  
Первая строка комментария (help-строка)

Комментарии

Тело функции

```
function f = fact (n)
% FACT Вычисление факториала.
%
% fact(n) возвращает n! – факториал
% числа n
f = prod(1:n);
```

Структура этой простейшей функции содержит компоненты, которые являются общими для любых функций системы MATLAB:

- **Строка определения функции** задает имя, количество и порядок следования входных и выходных аргументов.
- **Первая строка комментария (help-строка)** определяет назначение функции или сценария. Она выводится на экран с помощью команд `lookfor` или `help <имя_файла>`.
- **Комментарий** выводится на экран вместе с первой строкой при использовании команды `help <имя_файла>`.
- **Тело функции** - это программный код, который реализует вычисления и присваивает значения выходным аргументам.

## 6.2 Создание М-файлов. М-сценарии. М-функции

М-файлы являются обычными текстовыми файлами, которые создаются с помощью текстового редактора. Система MATLAB 5.x поддерживает специальный встроенный редактор/отладчик, хотя можно использовать и любой другой текстовый редактор работающий с ASCII-кодами (например, «Блокнот» или текстовый редактор файлового менеджера FAR, Norton Commander и др.).

### М-сценарии (М-script)

Сценарии являются самым простым типом М-файла – у них нет входных и выходных аргументов. Они используются для автоматизации много-



кратно выполняемых вычислений. Сценарии оперируют данными из рабочей области и могут генерировать новые данные для последующей обработки в этом же файле. Данные, которые используются в сценарии, сохраняются в рабочей области после завершения сценария и могут быть использованы для дальнейших вычислений.

*Пример:* Следующие операторы вычисляют радиус-вектор  $\rho$  для различных тригонометрических функций от угла  $\theta$  и строят последовательность графиков в полярных координатах.

Строка комментария (help-строка)	% M-file petals - сценарий
Вычисления	% построения лепесткового графика $\theta = -\pi:0.01:\pi;$ $\rho(1, :) = 2*\sin(5*\theta).^2;$ $\rho(2, :) = \cos(10*\theta).^3;$ $\rho(3, :) = \sin(\theta).^2;$ $\rho(4, :) = 5*\cos(3.5*\theta).^3;$
Команды графического вывода	for i = 1:4 polar (theta, rho(i, :)) pause end

Создайте М-файл `petals.m`, вводя указанные выше операторы. Этот файл является сценарием. Ввод команды `petals.m` в командной строке системы MATLAB вызывает выполнение операторов этого сценария.

После того, как сценарий отобразит первый график, нажмите любую клавишу, чтобы перейти к следующему графику. В сценарии отсутствуют входные и выходные аргументы; программа `petals.m` сама создает переменные, которые сохраняются в рабочей области системы MATLAB. Когда выполнение завершено, переменные ( $i$ ,  $\theta$  и  $\rho$ ) остаются в рабочей области. Для того чтобы увидеть этот список, следует воспользоваться командой `whos`.

### М-функции (M-function)

М-функции являются М-файлами, которые допускают наличие входных и выходных аргументов. Они работают с переменными в пределах собственной рабочей области, отличной от рабочей области системы MATLAB.

*Пример:* Функция `average` - это достаточно простой М-файл, который вычисляет среднее значение элементов вектора:

```
function y = average (x)
% AVERAGE Среднее значение элементов вектора.
```

```
% AVERAGE(X), где X - вектор. Вычисляет среднее значение
элементов вектора.
% Если входной аргумент не является вектором,
% генерируется ошибка.
[m,n] = size(x);
if (~(m == 1) | (n == 1)) | (m == 1 & n == 1))
    error('Входной массив должен быть вектором')
end
y =sum(x)/length(x); % -> Собственно вычисление
```

Попробуйте ввести эти команды в М-файл, именуемый `average.m`. Функция `average` допускает единственный входной и единственный выходной аргументы. Для того чтобы вызвать функцию `average`, надо ввести следующие операторы:

```
» z = 1:99;
» average(z)
» ans = 50
```

**Структура М-функции.** Любая М-функция состоит из:

- строки определения функции;
- первой строки комментария;
- собственно комментария;
- тела функции;
- строчных комментариев.

**Строка определения функции.** Строка определения функции сообщает системе MATLAB, что файл является М-функцией, а также определяет список входных аргументов.

*Пример:* Строка определения функции `average` имеет вид:

```
function y = average(x)
```

Здесь:

`function` - ключевое слово, определяющее М-функцию;  
`y` - выходной аргумент;  
`average` - имя функции;  
`x` - входной аргумент.

Каждая функция в системе MATLAB содержит строку определения функции, подобную приведенной.

Если функция имеет более одного выходного аргумента, список выходных аргументов помещается в квадратные скобки. Входные аргументы, если они присутствуют, помещаются в круглые скобки. Для отделения аргументов во входном и выходном списках применяются запятые.

*Пример:*

```
function [x, y, z] = sphere(theta, phi, rho)
```

Имена входных переменных могут, но не обязаны совпадать с именами, указанными в строке определения функции.

**Первая строка комментария.** Для функции `average` первая строка комментария выглядит так:

```
% AVERAGE Среднее значение элементов вектора
```

Это - первая строка текста, которая появляется, когда пользователь набирает команду `help <имя_функции>`. Кроме того, первая строка комментария выводится на экран по команде поиска `lookfor`. Поскольку эта строка содержит важную информацию об М-файле, она должна быть тщательно составлена.

**Тело функции.** Тело функции содержит код языка MATLAB, который выполняет вычисления и присваивает значения выходным аргументам. Операторы в теле функции могут состоять из вызовов функций, программных конструкций для управления потоком команд, интерактивного ввода/вывода, вычислений, присваиваний, комментариев и пустых строк.

*Пример:*

Тело функции `average` включает ряд простейших операторов программирования:

Оператор вызова функции `size`  
Начало оператора `if`  
Сообщение об ошибке  
Конец оператора `if`  
Вычисление и присваивание

```
[m, n] = size(x);  
if ~(m == 1) | (n==1) | (m ==1 & n==1)  
    Error('Input должно быть вектором')  
end  
y = sum(x)/length(x);
```

Как уже говорилось ранее, комментарии отмечаются знаком (%). Строка комментария может быть размещена в любом месте М-файла, в том числе и в конце строки.

*Пример:*

```
% -- Найти сумму всех элементов вектора x  
y = sum(x)      % -- Использована функция sum
```

Кроме строк комментариев в текст М-файла можно включать пустые строки. Однако надо помнить, что пустая строка может служить указателем окончания подсказки.

**Имена М-функций.** В системе MATLAB на имена М-функций налагаются те же ограничения, что и на имена переменных - их длина не должна превышать 31 символа. Более точно, имя может быть и длиннее, но система MATLAB принимает во внимание только первые 31 символ. Имена М-функций должны начинаться с буквы; остальные символы могут быть любой комбинацией букв, цифр и подчеркиваний.

Имя файла, содержащего М-функцию, составляется из имени функции и расширения \*.m : (например, average.m).

Если имя файла и имя функции в строке определения функции разные, то используется имя файла, а внутреннее имя игнорируется. Хотя имя функции, определенное в строке определения функции, может и не совпадать с именем файла, настоятельно рекомендуется использовать одинаковые имена.

**Двойственность функций и команд.** Команды системы MATLAB - это операторы вида:

```
load
help
```

Многие команды могут быть модифицированы добавлением операндов:

```
load August17.dat
help magic
type rank
```

Альтернативный метод задания модификаторов - определить их в качестве строковых аргументов функции:

```
load('August17.dat')
help('magic')
type('rank')
```

В этом заключается двойственность понятий команды и функции в системе MATLAB. Любая команда вида

```
command argument
```

может быть записана в форме функции

```
command('argument').
```

Преимущество функционального описания проявляется, когда строка аргументов формируется по частям. Следующий пример показывает, как может быть обработана последовательность файлов August1.dat, August2.dat, и т.д. Здесь используется функция int2str, которая переводит целое число в строку символов, что помогает сформировать последовательность имен файлов.

```
for d = 1:31
    s = ['August', int2str(d), '.dat']
    load(s) % - Загрузить файл с именем August'd'.dat
    % --- Операторы обработки файла ...
end
```

М-функцию можно вызвать из командной строки системы MATLAB или из других М-файлов, обязательно указав все необходимые атрибуты - входные аргументы в круглых скобках, выходные аргументы в квадратных скобках.

**Назначение имени.** Когда появляется новое имя, система MATLAB проверяет:

- Не является ли новое имя именем переменной.
- Не является ли это имя именем подфункции, то есть функции, которая размещена в этом же М-файле и является вызываемой.
- Не является ли оно именем частной функции, размещаемой в каталоге `private`. Этот каталог доступен только М-файлам, размещенным на один уровень выше.
- Не является ли оно именем функции в пути доступа системы MATLAB. В этом случае система использует тот М-файл, который встречается первым в пути доступа.

В случае дублирования имен система MATLAB использует первое имя в соответствии с вышеприведенной 4-уровневой иерархией. Следует отметить, что в системе MATLAB 5.x допускается переопределять функцию по правилам объектно-ориентированного программирования.

**Вызов функции.** При вызове М-функции, система MATLAB транслирует функцию в псевдокод и загружает в память. Это позволяет избежать повторного синтаксического анализа. Псевдокод остается в памяти до тех пор пока не будет использована команда `clear` или завершен сеанс работы.

### 6.3 Типы переменных

**Локальные и глобальные переменные.** Использование переменных в М-файле ничем не отличается от использования переменных в командной строке, а именно:

- переменные не требуют объявления; прежде чем переменной присвоить значение, необходимо убедиться, что всем переменным в правой части значения присвоены;
- любая операция присваивания создает переменную, если это необходимо, или изменяет значение существующей переменной;
- имена переменных начинаются с буквы, за которой следует любое количество букв, цифр и подчеркиваний; система MATLAB различает символы верхнего и нижнего регистров;
- имя переменной не должно превышать 31 символа. Более точно, имя может быть и длиннее, но система MATLAB принимает во внимание только первые 31 символ.

Обычно каждая М-функция, задаваемая в виде М-файла, имеет собственные локальные переменные, которые отличны от переменных других функций и переменных рабочей области. Однако, если несколько функций и рабочая область объявляют некоторую переменную глобальной, то все они используют единственную копию этой переменной. Любое присваивание этой переменной распространяется на все функции, где она объявлена глобальной.

*Пример:*

Допустим, требуется исследовать влияние коэффициентов  $\alpha$  и  $\beta$  для модели хищник-жертва, описываемой уравнениями Лотке-Вольтерра:

$$\begin{cases} \dot{y}_1 = y_1 - \alpha y_1 y_2 \\ \dot{y}_2 = -y_2 + \beta y_1 y_2 \end{cases}$$

Создадим М-файл `lotka.m`:

```
function yp = lotka(t, y)
% ЛОТКА уравнения Лотке-Вольтерра для модели хищник-жертва
global ALPHA BETA
yp = [y(1)-ALPHA*y(1)*y(2); -y(2)+BETA*y(1)*y(2)];
```

Затем через командную строку введем операторы:

```
» global ALPHA BETA
» ALPHA = 0.01;
» BETA = 0.02;
» [t,y] = ode23('lotka2',[0 10],[1; 1]);
» lot(t,y)
```

Команда `global` объявляет переменные `ALPHA` и `BETA` глобальными и следовательно, доступными в функции `lotka.m`. Таким образом, они могут быть изменены из командной строки, а новые решения будут получены без редактирования М-файла `lotka.m`.

Для работы с глобальными переменными необходимо:

- 1) объявить переменную как глобальную в каждой М-функции, которая необходима эта переменная. Для того чтобы переменная рабочей области была глобальной, необходимо объявить ее как глобальную из командной строки;
- 2) в каждой функции использовать команду `global` перед первым появлением переменной; рекомендуется указывать команду `global` в начале М-файла.

Имена глобальных переменных обычно более длинные и более содержательные, чем имена локальных переменных, и часто используют заглавные буквы. Это необязательно, но рекомендуется, чтобы обеспечить удобочитаемость кода языка MATLAB и уменьшить вероятность случайного переопределения глобальной переменной.

**Специальные переменные.** Некоторые М-функции возвращают специальные переменные, которые играют важную роль при работе в среде системы MATLAB:

<code>ans</code>	Последний результат; если выходная переменная не указана, то MATLAB использует переменную <code>ans</code> .
<code>eps</code>	Точность вычислений с плавающей точкой; определяется длиной мантиссы и для PC <code>eps = 2.220446049250313e-016</code>
<code>realmax</code>	Максимальное число с плавающей точкой, представимое в компьютере; для PC <code>realmax = 1.797693134862316e+308</code> .

realmin	Минимальное число с плавающей точкой, представимое в компьютере; для PC $\text{realmin} = 2.225073858507202\text{e-}308$ .
pi	Специальная переменная для числа $\pi$ : $\text{pi} = 3.141592653589793\text{e}+000$ .
i, j	Специальные переменные для обозначения мнимой единицы
inf	Специальная переменная для обозначения символа бесконечности
NaN	Специальная переменная для обозначения неопределенного значения - результата операций типа: $0/0$ , $\text{inf}/\text{inf}$ .
computer	Специальная переменная для обозначения типа используемого компьютера; для PC - PCWIN.
flops	Специальная переменная для обозначения количества операций с плавающей точкой.
version	Специальная переменная для хранения номера используемой версии системы MATLAB.

Соответствующие М-функции, генерирующие эти специальные переменные, находятся в каталоге `elmat` и поддержаны online-подсказкой.

### **6.3 Операторы системы MATLAB. Объединение операторов в арифметические выражения**

Операторы системы MATLAB делятся на три категории:

- Арифметические операторы позволяют конструировать арифметические выражения и выполнять числовые вычисления.
- Операторы отношения позволяют сравнивать числовые операнды.
- Логические операторы позволяют строить логические выражения.

Логические операторы имеют самый низкий приоритет относительно операторов отношения и арифметических операторов.

**Арифметические операторы.** При работе с массивом чисел установлены следующие уровни приоритета среди арифметических операций:

- уровень 1:* поэлементное транспонирование (`.'`), поэлементное возведение в степень (`.^`), эрмитово сопряженное транспонирование матрицы (`'`), возведение матрицы в степень (`^`);
- уровень 2:* унарное сложение (`+`), унарное вычитание (`-`);
- уровень 3:* умножение массивов (`.*`), правое деление (`./`), левое деление массивов (`.\`), умножение матриц (`*`), решение систем линейных уравнений - операция (`/`), операция (`\`);
- уровень 4:* сложение (`+`), вычитание (`-`);
- уровень 5:* оператор формирования массивов (`:`).

Внутри каждого уровня операторы имеют равный приоритет и вычисляются в порядке следования слева направо. Заданный по умолчанию порядок следования может быть изменен с помощью круглых скобок

Арифметические операторы допускают использование индексных выражений:

*Пример:*

```
b = sqrt(A(2))+2*B(1);
b = 7
```

Арифметические операторы системы MATLAB работают, как правило, с массивами одинаковых размеров. Для векторов и прямоугольных массивов оба операнда должны быть одинакового размера, за исключением единственного случая, когда один из них - скаляр. Если один из операндов скалярный, а другой нет, в системе MATLAB принято, что скаляр расширяется до размеров второго операнда и заданная операция применяется к каждому элементу. Такая операция называется расширением скаляра.

**Операторы отношения.** В системе MATLAB определено 6 следующих операторов отношения:

- < Меньше
- <= Меньше или равно
- > Больше
- >= Больше или равно
- == Равно тождественно
- ~= Не равно

Операторы отношения выполняют поэлементное сравнение двух массивов равных размерностей. Для векторов и прямоугольных массивов, оба операнда должны быть одинакового размера, за исключением случая, когда один из них скаляр. В этом случае MATLAB сравнивает скаляр с каждым элементом другого операнда. Позиции, где это соотношение истинно, получают значение 1, где ложно – 0.

Операторы отношения, как правило, применяется для изменения последовательности выполнения операторов программы. Поэтому они чаще всего используются в теле операторов if, for, while, switch. Операторы отношения всегда выполняются поэлементно

*Пример:*

Выполним сравнение двух массивов, используя условие  $A < B$ :

```
» A = [2 7 6; 9 0 -1; 3 0.5 6];
» B = [8 0.2 0; -3 2 5; 4 -1 7];
» A < B
ans =
     1     0     0
     0     1     1
     1     0     1
```

Полученная матрица указывает позиции, где элемент A меньше соответствующего элемента B.

При вычислении арифметических выражений операторы отношения имеют более низкий приоритет, чем арифметические, но более высокий, чем логические операторы.



Операторы отношения могут применяться к многомерным массивам, для которых одна из размерностей равна нулю, при условии, что оба массива - одинакового размера или один из них - скаляр. Однако выражения типа  $A == []$  применимы только к массивам размера  $0 \times 0$  или  $1 \times 1$ , а в других случаях вызывают ошибку.

Поэтому наиболее универсальный способ проверить, является ли массив пустым - это применить функцию `isempty(A)`.

**Логические операторы.** В состав логических операторов системы MATLAB входят следующие операторы:

&	И
	ИЛИ
~	НЕТ

Логические операторы реализуют поэлементное сравнение массивов одинаковых размерностей. Для векторов и прямоугольных массивов оба операнда должны быть одинакового размера, за исключением случая, когда один из них скаляр. В последнем случае MATLAB сравнивает скаляр с каждым элементом другого операнда. Позиции, где это соотношение истинно, получают значение 1, где ложно - 0.

Каждому логическому оператору соответствует некоторый набор условий, которые определяют результат логического выражения:

- Логическое выражение с оператором AND (&) является истинным, если оба операнда - истинны. Если элементами логического выражения являются числа, то выражение истинно, если оба операнда отличны от нуля.

*Пример:*

Пусть заданы два числовых вектора:

```
» u = [1 0 2 3 0 5];
```

```
» v = [5 6 1 0 0 7];
```

и логическое выражение с оператором AND (&):

```
» u & v
```

```
ans =
```

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

- Логическое выражение с оператором OR (|) является истинным, если один из операндов или оба операнда логически истинны. Выражение ложно, только если оба операнда логически ложны. Если элементами логического выражения являются числа, то выражение ложно, если оба операнда равны нулю.

*Пример:*

Используем векторы  $u$  и  $v$ , определенные выше, и выполним логическое выражение с оператором OR ( $|$ ):

```
» u|v
ans =
    1    1    1    1    0    1
```

▪ Логическое выражение с оператором NOT ( $\sim$ ) строит отрицание. Результат логически ложен, если операнд истинен, и истинен, если операнд ложен. Если элементами логического выражения являются числа, то любой операнд, отличный от нуля, становится нулем, и любой нулевой операнд становится единицей.

*Пример:*

```
» ~u
ans =
    0    1    0    0    1    0
```

**Логические функции.** В дополнение к логическим операторам в состав системы MATLAB включено ряд логических функций:

▪ Функция `xor(a,b)` реализует операцию ИСКЛЮЧИТЕЛЬНОЕ ИЛИ. Выражение, содержащее ИСКЛЮЧИТЕЛЬНОЕ ИЛИ истинно, если один из операндов имеет значение TRUE, а другой FALSE. Для числовых выражений, функция возвращает 1, если один из операндов отличен от нуля, а другой - нуль.

*Пример:*

Рассмотрим два числовых операнда  $a$  и  $b$ :

```
» a=1;
» b=1;
» xor(a,b)
ans =
    0
```

Функция `all` возвращает 1, если все элементы вектора истинны или отличны от нуля

*Пример:*

Пусть задан вектор  $u$  и требуется проверить его на условие «все ли элементы меньше 3». Если это условие выполняется, то выдается сообщение «Все элементы меньше 3»:

```
» u= [ 1 2 3 4 0];
» if all(u<3),
    disp(' Все элементы меньше 3 ');
end
```

В данном случае никакого сообщения не появится, но если в качестве вектора *u* взять

```
» u = [ 0 1 2 1 0];
```

то результатом будет сообщение: Все элементы меньше 3

В случае массивов функция *all* проверяет столбцы, то есть является ориентированной по столбцам.

- Функция *any* возвращает 1, если хотя бы один из элементов аргумента отличен от нуля; иначе, возвращается 0. В случае обработки массивов функция *any* является столбцовоориентированной.

- Функции *isnan* и *isinf* возвращают 1 для NaN и Inf, соответственно. Функция *isfinite* истинна только для величин, которые не имеют значения inf или NaN.

*Пример:*

Рассмотрим следующие два числовых массива A и B

```
A = [0 1 5; 2 NaN -inf];
```

```
B = [0 0 15; 2 5 inf];
```

Образует массив C и применим перечисленные выше функции

```
C = A./B
```

```
C =
```

```
NaN      Inf      0.3333
1.0000    NaN      NaN
```

Isfinite(C)	isnan(C)	Isinf(C)
ans =	ans =	ans =
0 0 1	1 0 0	0 1 0
1 0 0	0 1 1	0 0 0

- Функция *find* определяет индексы элементов массива, которые удовлетворяют заданному логическому условию. Как правило, она используется для создания шаблонов для сравнения и создания массивов индексов. В наиболее употребительной форме функция *txt=find(x<условие>)* возвращает вектор индексов тех элементов, которые удовлетворяет заданному условию.

*Пример:*

```
» A=magic(4)
```

```
A =
```

```
16      2      3      13
 5     11     10      8
 9      7      6     12
 4     14     15      1
```

```

» txt=find(A>8);
» A(txt)=100*ones(1,length(txt))
A =
    100     2     3    100
     5    100    100     8
    100     7     6    100
     4    100    100     1

```

Функция вида `[i,j]=find(x)` позволяет получить индексы ненулевых элементов прямоугольного массива. Функция вида `[i,j,s]=find(x)` возвращает кроме того и их значения в виде вектора `s`.

### **Объединение операторов в арифметические выражения**

Теперь вы имеете возможность строить выражения, которые используют любую комбинацию арифметических, логических операторов и операторов отношения.

**Управление последовательностью исполнения операторов.** Существуют четыре основных оператора управления последовательностью исполнения инструкций:

- оператор условия `if`, в сочетании с оператором `else` и `elseif` выполняет группу инструкций в соответствии с некоторыми логическими условиями;
- оператор переключения `switch`, в сочетании с операторами `case` и `otherwise` выполняет различные группы инструкций в зависимости от значения некоторого логического условия;
- оператор условия `while` выполняет группу инструкций неопределенное число раз, в соответствии с некоторым логическим условием завершения;
- оператор цикла `for` выполняет группу инструкций фиксированное число раз.

Все операторы управления включают оператор `end`, чтобы указать конец блока, в котором действует этот оператор управления.

`if...else...elseif...end`

#### **Оператор условия**

*Синтаксис:*

<pre> if &lt;логическое_выражение&gt;     инструкции end </pre>	<pre> if &lt;логическое_выражение&gt;     инструкции else     инструкции end </pre>	<pre> if &lt;логическое_выражение&gt;     инструкции elseif &lt;лог-кое_выраж-е&gt;     инструкции else     инструкции end </pre>
---	---	---

*Описание:*

Оператор условия `if . . . . end` вычисляет некоторое логическое выражение и выполняет соответствующую группу инструкций в зависимости от значения этого выражения. Если логическое выражение истинно, то MATLAB выполнит все инструкции между `if` и `end`, а затем продолжит выполнение программы в строке после `end`. Если условие ложно, то MATLAB пропускает все утверждения между `if` и `end` и продолжит выполнение в строке после `end`.

*Пример:*

```
if rem(a,2)==0,
    disp('a - четное число ')
    b = a/2;
end
```

Если логическое условие включает переменную, не являющуюся скаляром, то утверждение будет истинным, если все элементы отличны от нуля.

*Пример:*

Пусть задана матрица `X`; запишем следующий оператор условия:

```
if X
    инструкции
end
```

Этот оператор равносителен следующему:

```
if all(X(:))
    инструкции
end
```

Операторы `if...else...end` и `if...elseif...end` создают дополнительные ветвления внутри тела оператора `if`:

1. Оператор `else` не содержит логического условия. Инструкции, связанные с ним, выполняются, если предшествующий оператор `if` (и возможно `elseif`) ложны. Оператор `elseif` содержит логическое условие, которое вычисляется, если предшествующий оператор `if` (и возможно `elseif`) ложны. Инструкции, связанные с оператором `elseif` выполняются, если соответствующее логическое условие истинно.

2. Оператор `elseif` может многократно использоваться внутри оператора условия `if`.

*Пример:*

Рассмотрим фрагмент программы:

```
if n<0, % Если n<0, вывести сообщение об ошибке
    disp('Введенное число должно быть положительным');
elseif rem(n,2)==0, % Если n≥0 и четное, разделить на 2
    a=n/2;
```

```

else % Если  $n \geq 0$  и нечетное, увеличить на 1 и разделить
    a=(n+1)/2;
end

```

Если в операторе if условное выражение является пустым массивом, то такое условие ложно. То есть оператор условия вида

```

if A,
    S1
else
    S0
end

```

выполнит инструкции S0 только тогда, когда A - пустой массив.

```
switch...case...otherwise...end
```

## Оператор переключения

*Синтаксис:*

```

switch <выражение> % выражение - это обязательно скаляр или строка
case <значение1>
    инструкции % выполняются, если < выражение> =< значение1>
case <значение2>
    инструкции % выполняются, если <выражение> = < значение2>
otherwise
    инструкции % выполняются, если <выражение> не совпало ни с одним из
    значений
end

```

Оператор переключения switch работает только в системе MATLAB  
5.x и выше. Оператор switch...case\_1...case\_k...otherwise...end выполняет ветвления, в зависимости от значений некоторой переменной или выражения.

Оператор переключения включает:

- Заголовок switch, за которым следует вычисляемое выражение (скаляр или строка).
- Произвольное количество групп case; Заголовок группы состоит из слова case, за которым следует возможное значение выражения, расположенное на одной строке. Последующие строки содержат инструкции, которые выполняются для данного значения выражения. Выполнение продолжается до тех пор, пока не встретится следующий оператор case или оператор otherwise. На этом выполнение блока switch завершается
- Группа otherwise. Заголовок включает только слово otherwise, начиная со следующей строки размещаются инструкции, которые выполняются, если значение выражения оказалось не обработанным ни одной из групп case. Выполнение завершается оператором end.
- Оператор end является последним в блоке переключателя.

Оператор `switch` работает, сравнивая значение вычисленного выражения со значениями групп `case`. Для числовых выражений оператор `case` выполняется, если `<значение>==<выражение>`. Для строковых выражений, оператор `case` истинен, если `strcmp(значение, выражение)` истинно.

*Пример:*

Рассмотрим оператор `switch` со следующими условиями: он проверяет переменную `input_num`; если `input_num` равно -1, 0 или 1, то операторы `case` выводят на экран соответствующее сообщения. Если значения выражения `input_num` не равно ни одному из этих значений, то выполнение переходит к оператору `otherwise`.

```
switch input_num
    case -1, disp('минус один');
    case 0, disp('ноль');
    case 1, disp('один');
    otherwise, disp('другое значение');
end
```

while...end
-------------

## Оператор цикла с неопределенным числом операций

*Синтаксис:*

```
while <выражение>
    инструкции
end
```

*Описание:*

Оператор цикла с неопределенным числом операций `while ... end` многократно выполняет инструкцию или группу инструкций, пока управляющее выражение истинно.

Если выражение использует массив, то все его элементы должны быть истинны для продолжения выполнения. Чтобы привести матрицу к скалярному значению, следует использовать функции `any` и `all`.

*Пример:*

Этот цикл с неопределенным числом операций находит первое целое число `n`, для которого `n!` - записывается числом, содержащим 100 знаков:

```
n=1;
while prod(1:n)<1e100,
    n=n+1;
end
```

Досрочный выход из `while`-цикла может быть реализован с помощью оператора `break`. Если в операторе `while`, управляющее условие является пустым массивом, то такое условие ложно, то есть оператор вида

```
while A,
    S1,
```

end

никогда не выполнит инструкции S1, если A - пустой массив.

for...end
-----------

### Оператор цикла с определенным числом операций

*Синтаксис:*

```
for <перем-я_цикла>=<нач-ное_знач-е>:<приращение>:<конеч-ное_знач-е>
    инструкции
end
```

*Описание:*

Оператор цикла for .... end выполняет инструкцию или группу инструкций predeterminedное число раз. По умолчанию приращение равно 1. Можно задавать любое приращение, в том числе отрицательное. Для положительных индексов выполнение завершается, когда значение индекса превышает <конечное\_значение>; для отрицательных приращений выполнение завершается, когда индекс становится меньше чем <конечное\_значение>.

*Пример:*

Этот цикл выполняется пять раз:

```
for i=2:6,
    x(i)=2*x(i-1);
end
```

Допустимы вложенные циклы типа:

```
for i=1:m,
    for j=1:n,
        A(i,j)=1/(i + j - 1);
    end
end
```

В качестве переменной цикла for могут использоваться массивы. Например, рассмотрим массив A размера mхn. Оператор цикла

```
for i = A
    инструкции
end
```

определяет переменную цикла i как вектор A(:, k). Для первого шага цикла k равно 1; для второго k равно 2, и так далее, пока k не достигнет значения n. То есть цикл выполняется столько раз, сколько столбцов в матрице A. Для каждого шага i - это вектор, содержащий один из столбцов массива A.

Для досрочного выхода из цикла можно использовать оператор прерывания break. Он часто применяется совместно с условными выражениями. Когда этот оператор встречается в тексте программы, цикл досрочно прекращается и управление передается последующим (стоящим после слова end) операторам программы.



Следует помнить, что тело цикла выполняется столько раз, сколько раз меняется переменная цикла. Поэтому следует избегать применения циклов там, где MATLAB допускает альтернативные операции в векторной или матричной форме. К примеру, рассмотрим фрагмент программы

```
i=0;
for t=0:0.001:1,
    i=i+1;
    y(i)=sin(t);
end
```

его можно заменить на следующий:

```
t=0:0.001:1;
y=sin(y);
```

который не только проще, но и выполняется намного быстрее (примерно в 20-30 раз).

**Частные каталоги.** Они введены в систему MATLAB, начиная с версии 5.0. Частные каталоги представляют собой подкаталог с именем `private` родительского каталога. М-файлы частного каталога доступны только М-файлам родительского каталога. Поскольку файлы частного каталога не видимы вне родительского каталога, они могут иметь имена совпадающие, с именами файлов других каталогов системы MATLAB. Это удобно в тех случаях, когда пользователь создает собственные версии некоторой функции, сохраняя оригинал в другом каталоге. Поскольку MATLAB просматривает частный каталог раньше каталогов стандартных функций системы MATLAB, он в первую очередь находит функцию из частного каталога.

## 6.4 Ввод информации

В процессе выполнения М-файла пользователь может:

- вывести на экран запрос и ввести соответствующую информацию с клавиатуры;
- сделать паузу до нажатия клавиши;
- использовать графический интерфейс пользователя (GUI).

**Формирование запроса для ввода с клавиатуры.** Функция `input` выводит на экран запрос и ждет ответа пользователя. Ее синтаксис выглядит следующим образом:

```
n = input('запрос')
txt = input('запрос','s')
```

Функция `input` возвращает введенное с клавиатуры значение. Если пользователь вводит арифметическое выражение, функция вычисляет его и возвращает соответствующее значение. Функция полезна для реализации диалоговых прикладных программ.

Функция `input` может также возвращать не числовое, а строковое выражение, вводимое пользователем. Для ввода символьного выражения необходимо добавить строку `'s'` к списку параметров функции:

*Пример:*

```
name = input('Введите адрес ->', 's');
```

**Задание паузы.** В некоторых случаях целесообразно устанавливать паузу между отдельными шагами алгоритма, например, при выводе графиков. Команда `pause <без параметров>` останавливает выполнение до тех, пока не будет нажата какая-нибудь клавиша. Чтобы реализовать паузу в `n` секунд, необходимо применить оператор `pause(n)`.

**Выход в оболочку DOS.** Для обращения к программам, написанным на языках `C` или `Fortran`, можно использовать команду перехода в среду `DOS`, которая обозначается символ `(!)`. Это позволяет выполнять автономную внешнюю программу по аналогии с выполнением `M`-функции. Такая `M`-функция с вызовом внешней автономной программы равносильна `M`-файлу, который реализует следующие условия:

- Сохраняет переменные на диске.
- Выполняет внешнюю программу, которая читает файлы данных, обрабатывает их, и записывает результаты на диск.
- Загружает обрабатываемый файл в рабочую область.

## 6.5 Повышение эффективности обработки `M`-файлов

Этот раздел описывает методы повышения быстродействия при выполнении программы и управление памятью:

- векторизация циклов;
- предварительное размещение векторов.

`MATLAB` - это язык, специально разработанный для обработки массивов и выполнения матричных операций. Всюду, где это возможно, пользователь должен учитывать это обстоятельство.

**Векторизация циклов.** Под векторизацией понимается преобразование циклов `for` и `while` к эквивалентным векторным или матричным выражениям. При векторизации алгоритма ускоряется выполнение `M`-файла.

*Пример:*

Вот один из способов вычислить 1001 значение функции синуса на интервале `[0 10]`, используя оператор цикла:

```
i=0;
for t = 0:.01:10,
    i=i+1;
    y(i)=sin(t);
end
```

Эквивалентная векторизованная форма имеет вид

```
t = 0:.01:10;
y = sin(t);
```

В этом случае вычисления выполняются намного быстрее, и такой подход в системе MATLAB является предпочтительным. Время выполнения этих М-файлов можно оценить, используя команды `tic` и `toc`.

**Предварительное выделение памяти.** В системе MATLAB есть возможность для существенного сокращения времени выполнения программы за счет предварительного размещения массивов для выходных данных. Предварительное распределение избавляет от необходимости изменять массив при увеличении его размеров. Используйте соответствующие функции предварительного выделения памяти, как это показано в таблице для различных типов массивов

Тип массива	Функция	Примеры
Массив чисел	<code>zeros</code>	<pre>y = zeros(1, 100) for i = 1:100,     y(i) = det(X^i); end</pre>
Массив записей	<code>struct</code>	<pre>data = struct([1 3], 'x', [1 3], 'y', [5 6]) data(3).x = [9 0 2]; data(3).y = [5 6 7];</pre>
Массив ячеек	<code>cell</code>	<pre>B = cell(2, 3) B{1, 3} = 1:3; B{2, 2} = 'string'</pre>

Предварительное выделение памяти позволяет избежать фрагментации памяти при работе с большими матрицами. В ходе сеанса работы системы MATLAB, память может стать фрагментированной из-за работы механизмов динамического распределения и освобождения памяти. Это может привести к появлению большого количества фрагментов свободной памяти, и непрерывного пространства памяти может оказаться недостаточно для хранения какого-либо большого массива.

Предварительное выделение памяти позволяет также определить непрерывную область, достаточную для проведения всех вычислений

**Переменные и память.** Память выделяется для переменной всякий раз, когда происходит присваивание этой переменной какого-то значения (т.е. во время обращения к этой переменной).

Для экономии памяти надо:

- избегать использовать одни и те же переменные в качестве входных и выходных аргументов функции, поскольку они будут передаваться ссылкой;
- после использования переменной целесообразно либо присвоить ей пустой массив, либо удалить с помощью команды `clear <имя_переменной>`;
- стремиться использовать переменные повторно.

**Глобальные переменные.** При объявлении глобальной переменной в таблицу переменных просто помещается флаг. При этом не требуется дополнительной памяти.

Например, последовательность операторов `a=5; global a` определяет переменную `a` как глобальную и формируется дополнительная копия этой переменной.

Функция `clear a` удаляет переменную `a` из рабочей области системы MATLAB, но сохраняет ее в области глобальных переменных. Функция `clear global a` удаляет переменную `a` из области глобальных переменных.

## **6.6 Работа с графическими средствами системы MATLAB**

Одно из достоинств системы MATLAB – обилие средств графики, начиная от команд построения простых графиков функций одной переменной в декартовой системе координат и кончая комбинированными и демонстрационными графиками с элементами анимации. Особое внимание в системе уделено трехмерной графике с функциональной окраской отображаемых фигур.

Пользователь может на одном графике строить несколько кривых, выделяя их различными цветами или различными метками. Графики могут иметь линейную, логарифмическую или полулогарифмическую шкалу. По умолчанию устанавливается режим автомасштабирования графиков, так что они имеют максимально большие размеры. Отметим основные операторы графики в системе MATLAB:

- `plot` – построение графика в линейном масштабе;
- `loglog` – построение графика в логарифмическом масштабе;
- `semilogx` – построение графика в полулогарифмическом масштабе – логарифмический масштаб по оси  $x$ ;
- `semilogy` – построение графика в полулогарифмическом масштабе – логарифмический масштаб по оси  $y$ ;
- `polar` - построение графика в полярной системе координат;
- `mesh` - построение графика трехмерной поверхности в виде сетки;
- `surf` – построение графика функции двух переменных в виде гладкой («залитой») поверхности ;
- `contour` – построение графика с контурными линиями уровнями равных высот для функции двух переменных;

- `bar` или `hist` – построение графика столбцовой гистограммы.

Для окончательного оформления графиков необходимо предусмотреть нанесение надписей по осям координат, титульной надписи и надписей с произвольной ориентацией относительно графиков. Для этой цели служат следующие операторы:

- `text` – вывод надписи в заданное место графика;
- `title` – задание титульной надписи для графика;
- `xlabel` – задание надписи по оси  $x$ ;
- `ylabel` – задание надписи по оси  $y$ ;
- `grid on/off` – включение/выключение масштабной сетки.

Ниже рассмотрим применение графических операторов на ряде конкретных примеров.

### 6.6.1 Двумерные графики

Функции одной переменной  $y(x)$  находят широкое применение в практике математических и инженерных расчетов, а также при компьютерном математическом моделировании. Для отображения таких функций часто используют прямоугольную декартовую систему координат.

Команда `plot` служит для построения графиков функции одной переменной в декартовой системе координат. Эта команда имеет ряд параметров, рассматриваемых ниже

`plot`

**построение графика функции одной переменной**

*Синтаксис:* `plot(X, Y, S)`

Строит график функции  $y(x)$ , координаты точек  $(x, y)$  которой берутся из векторов одинакового размера  $X$  и  $Y$ . Если  $X$  или  $Y$  – матрица, то строится семейство графиков по данным, содержащимся в матрице. Тип линии графика можно задавать с помощью строковой константы  $S$ .

Значениями константы  $S$  могут быть следующие символы:

Константа $S$	Цвет линии
y	желтый
m	фиолетовый
b	синий
r	красный
g	зеленый
w	белый
k	черный
c	голубой

Константа S	Тип точки
.	точка
o	окружность
x	крест
+	плюс
*	звездочка
-	сплошная
:	пунктирная
-.	штрих-пунктирная
--	Штриховая

Таким образом, с помощью строковой константы S можно менять цвет линии, представлять узловые точки различными метками и менять тип линии графика. Если в команде `plot` отсутствует указание на цвет линий и точек, то он выбирается автоматически из таблицы цветов. Если линий больше шести, то выбор цветов повторяется.

*Пример:* Построение графика функции  $y=\sin(x)$

```
»% график функции sin(x)
» x=0:0.01:2*pi; % формирование вектора x
» plot(x,sin(x),'-w');
```

Команда `plot(Y)` строит график  $y(i)$ , где значения  $y$  берутся из вектора  $Y$ , а  $i$  представляет собой индекс соответствующего элемента. Если  $Y$  содержит комплексные числа, то выполняется команда `plot(real(Y),imag(Y))`. Во всех других случаях мнимая часть данных игнорируется.

*Пример:* построение фигуры Лиссажу для кратности частот 1/3.

```
% построение фигуры Лиссажу
x=0:0.02*pi:2*pi;
k1=1;
k2=3;
y=sin(k1*x)+i*cos(k2*x);
plot(y);
axis([-1.2 1.2 -1.2 1.2]); % задание масштаба осей
координат
```

Результат выполнения этой программки показан на рис. 3.

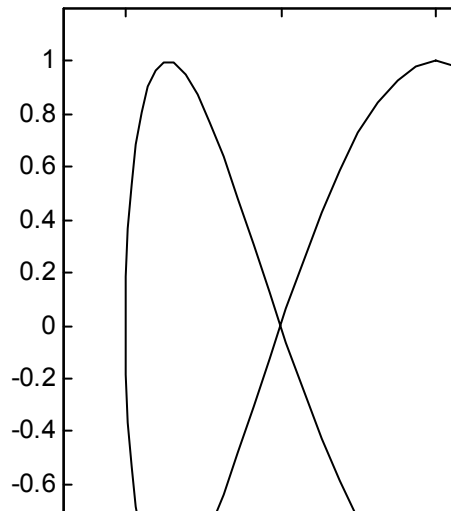


Рисунок 3 – Фигура Лиссажу для соотношения частот 1/3

Система MATLAB позволяет на одном графике строить несколько кривых одновременно:

`plot(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3,...)` – эта команда строит на одном графике ряд линий, представленных данными вида  $(X_i, Y_i, S_i)$ , где  $X_i, Y_i$  – векторы значений функций,  $S_i$  – строковые константы, определяющие вид графика.

*Пример:* рассмотрим пример построения графика нескольких функций с различным стилем представления каждой

```
% графики двух функций со спецификацией линий каждого графика
a=-2*pi; b=2*pi; % интервал по оси x построения графиков
N=100; % число точек линий
N1=10; % число меток
delx=(b-a)/(N-1); % расчет шага для расчета x
x=a:delx:b; % формирование вектора x
delx1=(b-a)/(N1-1);
x1=a:delx1:b; % формирование вектора x для меток
y1=sin(x); y1m=sin(x1); % расчет первой функции
y2=sin(x)./x; y2m=sin(x1)./x1; % расчет второй функции
plot(x,y1,'-w',x1,y1m,'ow',x,y2,'-b');
```

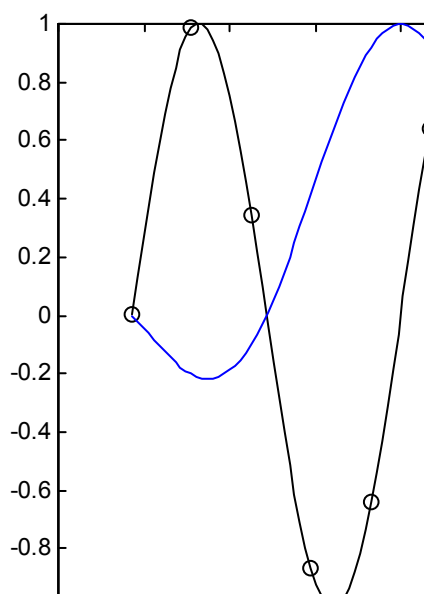


Рисунок 4 – Пример построения нескольких графиков в одном графическом окне

### 6.6.2 Оформление и комбинирование графиков

После того, как график уже построен, MATLAB позволяет выполнить его дополнительное форматирование или оформление в нужном виде. Соответствующие операторы рассмотрены ниже.

**Установка титульной надписи и подписей на осях.** Для установки над графиком титульной надписи используется команда `title`.

*Синтаксис:* `title('string')` – установка над графиком надписи, заданной строковой переменной `string`.

Для установки подписей возле осей  $x$ ,  $y$  или  $z$  используются команды:

```
xlabel('string')
ylabel('string')
zlabel('string')
```

подпись также определяется строковой переменной `string`.

**Ввод текста в графическом окне.** Часто возникает необходимость добавления текста в определенное место графика, например для обозначения той или иной кривой графика. Для этого используются команды `text` и `gtext`:

text

**ввод текста в поле графика**

*Синтаксис:* `text(X,Y,'string')`  
`text(X,Y,Z,'string')`



добавляет на график (двухмерный или трехмерный) текст, заданный строкой 'string', так что начало текста расположено в точке с координатами (X,Y,Z).

`gtext`

**ввод текста в поле графика с помощью мыши**

*Синтаксис:* `gtext('string')`

добавляет на график (двухмерный или трехмерный) текст, заданный строкой 'string'. Место вывода текста определяется с помощью мыши.

**Управление свойствами осей координат.** Обычно графики в системе MATLAB строятся в режиме автоматического масштабирования. Используя команду `axis` можно управлять формой осей координат и масштабом.

`axis`

**установка масштаба и типа осей координат**

*Синтаксис:*

```
axis([xmin xmax ymin ymax])
axis([xmin xmax ymin ymax zmin zmax])
axis('auto')
axis(axis)
V=axis
axis('square')
axis('equal')
axis('image')
axis('normal')
axis('off') или axis off
axis('on') или axis on
```

*Описание:*

▪ `axis([Xmin Xmax Ymin Ymax])` – ручная установка масштаба по осям x и y для текущего двухмерного графика;

▪ `axis([Xmin Xmax Ymin Ymax Zmin, Zmax])` – ручная установка масштаба по осям x и y для текущего трехмерного графика;

▪ `axis auto` - включение режима автоматического масштабирования (стоит по умолчанию);

▪ `axis('square')` – устанавливает текущие оси координат в виде квадрата (или куба в трехмерном случае);

▪ `axis('equal')` – включает масштаб с одинаковым расстоянием между метками по осям x, y и z;

▪ `axis('image')` – команда устанавливает масштаб, который обеспечивает квадратные размеры пикселей;

▪ команда `axis('normal')` восстанавливает полноразмерный масштаб, отменяя масштабы, установленные командами `axis('square')` и `axis('equal')` – данный режим стоит по умолчанию;

- `axis(axis)` – фиксирует текущие значения масштабов для последующих графиков, как если бы был включен режим `hold`;
- функция `V=axis` возвращает вектор-строку масштабов по осям для активного графика. Если график двумерный, то `V` имеет 4 компонента; если трехмерный – 6 компонентов;
- `axis('off')` – выключает отображение осей, их обозначения и маркеры на экране;
- `axis('on')` – включает отображение осей, их обозначения и маркеры на экране (стоит по умолчанию).

<code>grid</code>
-------------------

### нанесение масштабной сетки

#### *Синтаксис:*

```
grid on
grid off
grid
```

#### *Описание:*

Команда `grid on` наносит координатную сетку на текущие оси.

Команда `grid off` удаляет координатную сетку.

Команда `grid` выполняет роль переключателя с одной функции на другую.

<code>hold</code>
-------------------

### управление режимом сохранения текущего графического окна

#### *Синтаксис:*

```
hold on
hold off
hold
```

#### *Описание:*

Команда `hold on` включает режим сохранения текущего графика и свойств объекта `axes`, так что последующие команды приведут к добавлению новых графиков в графическом окне.

Команда `hold off` выключает режим сохранения графика.

Команда `hold` реализует переключение от одного режима к другому.

#### *Пояснение:*

Команды `hold` воздействуют на значения свойства `NextPlot` объектов `figure` и `axes`:

- `hold on` присваивает свойству `NextPlot` для текущих объектов `figure` и `axes` значение `add`;

▪ `hold off` присваивает свойству `NextPlot` для текущих объектов `figure` и `axes` значение `replace`.

**subplot**

**разбиение графического окна**

*Синтаксис:*

```
subplot(m, n, p)
subplot(h)
subplot(mnp)
```

*Описание:*

Данная команда выполняется перед обращением к функциям построения графиков для одновременной выдачи нескольких графиков в различных частях графического окна.

Команды `subplot(mnp)` или `subplot(m, n, p)`, где `mnp` - 3 цифры, производит разбику графического окна на несколько подокон, создавая при этом новые объекты `axes`; значение `m` указывает, на сколько частей разбивается окно по горизонтали, `n` - по вертикали, а `p` - номер подокна, куда будет выводиться очередной график. Эти же команды могут использоваться для перехода от одного подокна к другому.

Команды `clf`, `subplot(111)`, `subplot(1,1,1)` выполняют одну и ту же функцию - удаляют все подокна и возвращают графическое окно в штатное состояние.

*Пример:* В верхней части экрана строится функция  $y_1 = \sin(x)$ , в нижней -  $y_2 = \log(\text{abs}(y))$ .

```
x = -1:.1:1;
y1 = sin(x);
subplot(2, 1, 1), plot(x, y1)
y2 = log(abs(y1));
subplot(2, 1, 2), plot(x, y2)
```

**colormap**

**Указание палитры цветов**

*Синтаксис:*

```
colormap(C)
colormap('default')
C = colormap
caxis(caxis)
```

*Описание:*

Палитра цветов `C` - это матрица размера `mх3` действительных чисел из диапазона `[0.0 1.0]`. Строка `k` палитры сформирована из трех чисел, которые

указывают интенсивность красного, зеленого и синего цветов, то есть  $C(k,:)=[r(k) \ g(k) \ b(k)]$ .

Команда `colormap(C)` устанавливает палитру согласно матрице  $C$ . Если значение элемента матрицы выходит за пределы интервала  $[0 \ 1]$ , выдается сообщение об ошибке

`Colormap must have values in [0,1].`

Значения элементов палитры должны быть в интервале  $[0 \ 1]$ .

Команды `colormap('default')` или `colormap(hsv)` устанавливают штатную палитру, которая соответствует модели hue-saturation-value (оттенок-насыщенность-значение). Последовательность цветов этой палитры соответствует цветам радуги.

red	Красный	Каждый
-	-	охотник
yellow	Желтый	желает
green	Зеленый	знать,
cyan	Голубой	где
blue	Синий	сидит
magenta	Фиолетовый	фазан

В рамках версий 4.x системы MATLAB реализованы следующие палитры:

bone	Grey-scale with tinge of blue color map	Серая палитра с оттенком синего
cool	Shades of cyan and magenta color map	Палитра с оттенками голубого и фиолетового
copper	Linear copper-tone color map	Линейная палитра в оттенках меди
flag	Alternating red, white, blue, and black color map	Палитра с чередованием красного, синего и черного
gray	Linear grey-scale color map	Линейная палитра в оттенках серого
hot	Black-red-yellow-white color map	Палитра с чередованием черного, красного, желтого и белого
hsv	Hue-saturation-value color map	Палитра радуги
jet	Variant of hsv	Разновидность hsv-палитры
pink	Pastel shades of pink color map	Розовая палитра с оттенками пастели
prism	Prism (red-orange-yellow-green -blue-violet) color map	Палитра с чередованием красного, оранжевого, желтого, зеленого, синего и фиолетового
white	All white color map	Белая палитра

### 6.6.3 Специальные двумерные графики

Система MATLAB также позволяет выполнять построение графиков функции одной переменной в специфической форме: в полярной системе координат, в логарифмическом масштабе, построение гистограмм и столбцовых диаграмм, векторные диаграммы и пр. Рассмотрим несколько примеров.

loglog

**график функции в логарифмическом масштабе**

*Синтаксис:*

```
loglog(x, y)
loglog(x, y, s)
loglog(x1, y1, s1, x2, y2, s2, ...)
```

*Описание:*

Команды `loglog(...)` равносильны функциям `plot`, за исключением того, что они используют по обеим осям логарифмический масштаб вместо линейного.

*Пример:* Построим график  $y = \exp(x)$  в логарифмическом масштабе:

```
x = logspace(-1, 2);
loglog(x, exp(x))
grid on
```

semilogx semilogy

**график функции в полулогарифмическом масштабе**

*Синтаксис:*

<code>semilogx(x, y)</code>	<code>semilogy(x, y)</code>
<code>semilogx(x, y, s)</code>	<code>semilogy(x, y, s)</code>
<code>semilogx(x1, y1, s1, x2, y2, s2, ...)</code>	<code>semilogy(x1, y1, s1, x2, y2, s2, ...)</code>

*Описание:*

Команды `semilogx(...)` используют логарифмический масштаб по оси  $x$  и линейный масштаб по оси  $y$ .

Команды `semilogy(...)` используют логарифмический масштаб по оси  $y$  и линейный масштаб по оси  $x$ .

*Пример:* Построим график  $y = \exp(x)$  в полулогарифмическом масштабе по оси  $y$ :

```
x = 0:0.1:100;
semilogy(x, exp(x))
grid on
```

**polar****график функции в полярной системе координат***Синтаксис:*

```
polar(phi, rho)
polar(phi, rho, s)
```

*Описание:*

Команды `polar(...)` реализуют построение графиков в полярных координатах, задаваемых углом `phi` и радиусом `rho`.

*Пример:* Построим график функции  $\rho = \sin(2\phi) \cdot \cos(2\phi)$  в полярных координатах

```
phi = 0:0.01:2*pi;
polar(phi, sin(2*phi) .* cos(2*phi)) ;
```

Результат представлен на рис. 5.

Рисунок 5 – График функции  $\sin(x) \cdot \cos(x)$  в полярной системе координат

**bar****график функции в виде столбцовой диаграммы***Синтаксис:*

```
bar(y) bar(x, y)
[xb, yb]=bar(...)
bar(y, '<тип_линии>')
bar(x, y, '<тип_линии>')
```

*Описание:*

Команда `bar(y)` выводит график элементов одномерного массива `y` в виде столбцовой диаграммы.

Команда `bar(x, y)` выводит график элементов массива `y` в виде столбцов в позициях, определяемых массивом `x`, элементы которого должны быть упорядочены в порядке возрастания.

Если `X` и `Y` - двумерные массивы одинаковых размеров, то каждая диаграмма определяется соответствующей парой столбцов и они надстраиваются одна над другой.

Команды `bar(y, '<тип_линии>')`, `bar(x, y, '<тип_линии>')` позволяют задать тип линий, используемых для построения столбцовых диаграмм, по аналогии с командой `plot`.

Функция `[xb, yb] = bar(...)` не выводит графика, а формирует такие массивы `xb` и `yb`, которые позволяют построить столбцовую диаграмму с помощью команды `plot(xb, yb)`.

*Пример:* Построить график функции  $y = e^{-x^2}$  в виде столбцовой диаграммы.

```
x=-2.9:0.2:2.9;
bar(x, exp(-x.*x));
```

**hist****Построение гистограммы***Синтаксис:*

```
hist(y)
hist(y, n)
hist(y, x)
[y, x] = hist(y, ...)
```

*Описание:*

Команды `hist(...)` подсчитывают и отображают на графике количество элементов массива `y`, значения которых попадают в заданный интервал; для этого весь диапазон значений `y` делится на `n` интервалов (по умолчанию 10) и подсчитывается количество элементов в каждом интервале.

Команда `hist(y)` выводит гистограмму для 10 интервалов.

Команда `hist(y, n)` выводит гистограмму для `n` интервалов.

Команда `hist(y, x)` выводит гистограмму с учетом диапазона изменения переменной `x`. Здесь `x` должен быть вектором.

Функции `[y, x] = hist(y, ...)` формируют такие массивы `y` и `x`, что `bar(x, y)` является гистограммой.

*Пример:* Построить гистограмму для 10 000 случайных чисел, распределенных по нормальному и равномерному законам.

```

y1= randn(10000,1);
y2=rand(10000,1);
subplot(211);
hist(y1, 20);
subplot(212);
hist(y2,20)

```

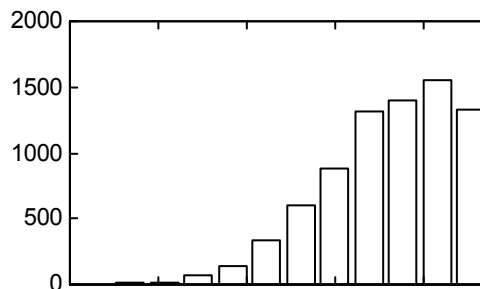


Рисунок 6 – Гистограммы нормального и равномерного распределений случайной величины

#### 6.6.4 Трехмерные графики функции двух переменных

В системе MATLAB предусмотрено несколько команд и функций для построения трехмерных графиков. Значения элементов числового массива рассматриваются как  $z$ -координаты точек над плоскостью, определяемой координатами  $x$  и  $y$ . Возможно несколько способов соединения этих точек. Первый из них - это соединение точек в сечении (функция `plot3`), второй - построение сетчатых поверхностей (функции `mesh` и `surf`). Поверхность, построенная с помощью функции `mesh`, - это сетчатая поверхность, ячейки которой имеют цвет фона, а их границы могут иметь цвет, который определяется свойством `EdgeColor` графического объекта `surface`. Поверхность, построенная с помощью функции `surf`, - это сетчатая поверхность, у которой может быть задан цвет не только границы, но и ячейки; последнее управляется свойством `FaceColor` графического объекта `surface`. Заинтересованному читателю рекомендуем обратиться к документации по системе MATLAB.



**Создание массивов данных для трехмерной графики.** Трехмерные поверхности обычно описываются функцией двух переменных  $z=f(x, y)$ . Специфика построения трехмерных графиков требует не просто задания ряда значений  $x$  и  $y$ , то есть векторов  $X$  и  $Y$ . Она требует определения для координат  $x$  и  $y$  специальных двумерных массивов – матриц. Для создания таких массивов служит функция `meshgrid`. Она используется в основном совместно с функциями построения графиков трехмерных поверхностей и фигур.

<code>meshgrid</code>
-----------------------

### Формирование двумерных массивов $X$ и $Y$ для построения 3-D графиков

*Синтаксис:*

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

*Описание:*

Функция `[X,Y]=meshgrid(x,y)` задает сетку на плоскости  $x$ - $y$  в виде двумерных массивов  $X$ ,  $Y$ , которые определяются одномерными массивами  $x$  и  $y$ . Строки массива  $X$  являются копиями вектора  $x$ , а столбцы - копиями вектора  $y$ . Формирование таких массивов упрощает вычисление функций двух переменных, позволяя применять операции над массивами.

Функция `[X,Y]=meshgrid(x)` представляет собой упрощенную форму записи для функции `[X,Y] = meshgrid(x, x)`.

*Пример:*

```
>> [X,Y]=meshgrid(1:4,-3:3)
```

$X =$

1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4

$Y =$

-3	-3	-3	-3
-2	-2	-2	-2
-1	-1	-1	-1
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3

Используя полученные массивы  $X$  и  $Y$ , далее осуществляют расчет функции  $z=f(x, y)$  для выполнения построения графика. График трехмерной функции по полученной матрице  $z$  можно строить разными операторами.

Plot3
-------

### Построение линий и точек в трехмерном пространстве

*Синтаксис:*

```
plot3(x, y, z)
plot3(x, y, z, s)
plot3(x1, y1, z1, s1, x2, y2, z2, s2, ...)
```

*Описание:*

Команды `plot3(...)` являются трехмерными аналогами функции `plot(...)`.

Команда `plot3(x, y, z)`, где  $x, y, z$  - одномерные массивы одинакового размера, строит точки с координатами  $x(i), y(i), z(i)$  и соединяет их прямыми линиями.

Команда `plot3(x, y, z, S)` позволяет выделить график функции  $z=f(x, y)$ , указав способ отображения линии, способ отображения точек, цвет линий и точек с помощью строковой переменной  $S$ .

Команда `plot3(x1, y1, z1, s1, x2, y2, z2, s2, ...)` позволяет объединить на одном графике несколько функций  $z1(x1, y1), z2(x2, y2), \dots$ , определив для каждой из них свой способ отображения.

Обращение к команде `plot3` вида `plot3(x, y, z, s1, x, y, z, s2)` позволяет для графика  $z=f(x, y)$  определить дополнительные свойства, для указания которых применения одной строковой переменной  $s1$  недостаточно, например при задании разных цветов для линии и для точек на ней.

*Пример:* Построим график функции  $z = xe^{(-x^2-y^2)}$  в трехмерном пространстве.

```
[X,Y]=meshgrid([-2:0.1:2]); % построение сетки значений для x и y
Z=X.*exp(-X.^2-Y.^2); % расчет функции
plot3(X, Y, Z)
```

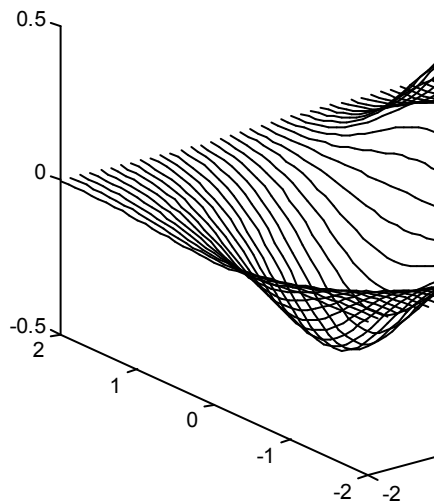


Рисунок 7- График функции  $z = xe^{(-x^2-y^2)}$  построенный с помощью оператора plot3

mesh,  
meshc

### Трехмерная сетчатая поверхность

#### Синтаксис:

```
mesh(X,Y,Z,C)   meshc(X,Y,Z,C)
mesh(x,y,Z,C)   meshc(x,y,Z,C)
mesh(Z,C)        meshc(Z,C)
mesh(X,Y,Z)      meshc(X,Y,Z)
mesh(x,y,Z)      meshc(x,y,Z)
mesh(Z)          meshc(Z)
```

#### Описание:

Команда `mesh(X,Y,Z,C)` выводит на экран сетчатую поверхность для значений массива `Z`, определенных на множестве значений массивов `X` и `Y`. Цвета узлов поверхности задаются массивом `C`. Цвета ребер определяются свойством `EdgeColor` объекта `surface`. Можно задать одинаковый цвет для всех ребер, определив его в виде вектора `[r g b]` интенсивности трех цветов - красного, зеленого, синего. Если определить спецификацию `none`, то ребра не будут прорисовываться. Если определить спецификацию `flat`, то цвет ребер ячейки определяется цветом того узла, который был первым при обходе этой ячейки. Поскольку одни и те же ребра обходятся несколько раз, то цвета будут замещаться. Если определить спецификацию `interp`, то будет реализована линейная интерполяция цвета между вершинами ребра.

Применение функции `shading` после обращения к функции `mesh` изменяет спецификации свойств `EdgeColor` и `FaceColor` согласно следующей таблице.

Свойство	Применяемая функция		
	mesh	shading flat	shading interp
EdgeColor	flat	flat	interp
FaceColor	Цвет фона	Цвет фона	Цвет фона

Команда `mesh(x, y, Z, C)` выполняет ту же функцию, но вместо двумерных массивов  $X$ ,  $Y$  использует одномерные вектора, такие что если `length(x)=n`, а `length(y)=m`, то `[m, n]=size(Z)`. В этом случае узлы сетчатой поверхности определяются тройками  $\{x(j), y(i), Z(i, j)\}$ , где вектор  $x$  определяет столбцы массива  $Z$ , а  $y$  - строки.

Команда `mesh(Z, C)` использует сетку, которая определяется одномерными массивами  $x = 1:n$  и  $y = 1:m$ .

Команды `mesh(X, Y, Z)`, `mesh(x, y, Z)`, `mesh(Z)` используют в качестве массива цвета  $C=Z$ , то есть цвет в этом случае пропорционален высоте поверхности.

Группа команд `meshc(...)` в дополнение к трехмерным поверхностям строит проекцию линий постоянного уровня.

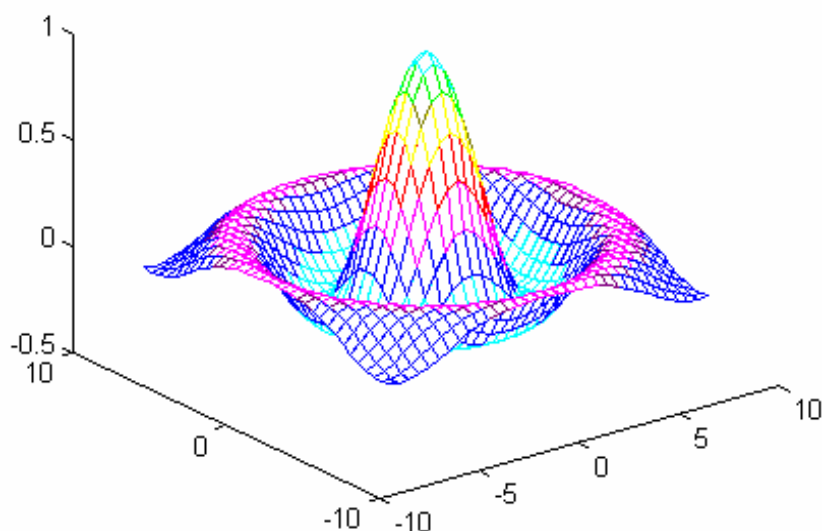


Рисунок 8 – График трехмерной поверхности, выполненный с помощью функции `mesh`

*Пример:* Построим трехмерную поверхность функции  $z = \sin(\sqrt{x^2 + y^2}) / (\sqrt{x^2 + y^2})$ .

```
[X, Y] = meshgrid([-8 : 0.5 : 8]);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R) ./ R;
mesh(X, Y, Z)
```

Результат выполнения данного фрагмента программы представлен на рис. 8.

**colorbar****Шкала палитры для трехмерного графика***Синтаксис:*

```
colorbar('vert')
colorbar('horiz')
colorbar(h)
colorbar
```

*Описание:*

Команда `colorbar('vert')` добавляет к текущему графику вертикальную шкалу палитры.

Команда `colorbar('horiz')` добавляет к текущему графику горизонтальную шкалу палитры.

Команда `colorbar(h)` добавляет к графику с дескриптором `h` шкалу палитры. Размещение шкалы реализуется автоматически в зависимости от соотношения ширины и высоты графика.

Команда `colorbar` без аргументов размещает на текущем графике новую вертикальную шкалу палитры или обновляет прежнюю

**contour****Изображение линий уровня для трехмерной поверхности***Синтаксис:*

<code>contour(Z)</code>	<code>contour(x, y, Z)</code>
<code>contour(Z, n)</code>	<code>contour(x, y, Z, n)</code>
<code>contour(Z, v)</code>	<code>contour(x, y, Z, v)</code>
<code>contour(..., 'тип линии')</code>	
<code>C = contour(...)</code>	
<code>[C, h] = contour(...)</code>	

*Описание:*

Команда `contour(Z)` рисует двумерные линии уровня для массива данных `Z`, определяющих поверхность в трехмерном пространстве без учета диапазона изменения координат `x` и `y`.

Команда `contour(x, y, Z)`, где `x` и `y` - векторы, рисует линии уровня для массива данных `Z` с учетом диапазона изменения координат `x` и `y`.

Команды `contour(Z, n)`, `contour(x, y, Z, n)` рисует `n` линий уровня для массива данных `Z`; по умолчанию, `n` равно 10.

Команды `contour(Z, V)`, `contour(x, y, Z, V)` рисуют линии уровня для заданных значений, которые указаны в векторе `V`.

Команда `contour(..., '<тип_линии>')` рисует линии уровня, тип и цвет которых определяются параметром `<тип_линии>` команды `plot`.

Функция `C=contour(...)` возвращает массив `C` описания линий уровней по аналогии с функцией `contourc` для последующего использования командой `clabel`.

Функция `[C, h]=contour(...)` возвращает массив `C` и вектор-столбец дескрипторов `h` графических объектов `line` для каждой линии уровня.

*Пример:* Построить линии уровня функции  $z = xe^{(-x^2-y^2)}$ .

```
x=-2:.2:2;
y=x;
[X,Y]=meshgrid(x);
Z=X.*exp(-X.^2-Y.^2);
contour(X, Y, Z)
```

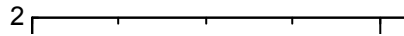


Рисунок 9 – Линии уровня функции  $z = xe^{(-x^2-y^2)}$

*Диагностические сообщения:* Если меньшая размерность массива `Z` оказывается меньше двух, то выдается сообщение

`Matrix must be 2-by-2 or larger.`

Матрица должна иметь размер 2x2 или больше.

*Ограничения:* При работе с командой и функцией `contour` предполагается, что элементы массивов `x` и `y` монотонно возрастают.

## **7 Заключение**

Данное руководство не является полным описанием системы MATLAB. Цель, которую преследовал автор – познакомить студентов с системой, показать основные приемы работы и программирования в ней. Для более глубокого изучения MATLAB следует обращаться к документации и дополнительной литературе. Следует отметить, что в последнее время MATLAB завоевывает все большую популярность среди пакетов для математических и инженерных расчетов в России. Это подтверждает появление большого числа публикация и книг о MATLAB и различных прикладных пакетах, входящих в его состав.

## СПИСОК ДОПОЛНИТЕЛЬНЫХ ИСТОЧНИКОВ

1. [www.mathworks.com](http://www.mathworks.com) - сайт фирмы-разработчика системы MATLAB. Содержит последние новости о системе и прикладных пакетах, имеет архив свободно распространяемых программ на языке системы MATLAB.
2. [www.matlab.ru](http://www.matlab.ru) - сайт пользователей системы MATLAB в нашей стране, имеет ссылки на другие ресурсы, посвященный MATLAB.
3. [www.exponenta.ru](http://www.exponenta.ru) - очень полезный сайт для начинающих и уже продвинутых пользователей MATLAB. Здесь можно найти книгу и справочник по операторам языка MATLAB Потемкина В.Г. Сайт содержит полезную информацию и по другим математическим пакетам Maple, Mathematica, MathCAD и пр. Также есть ссылки на другие полезные ресурсы.
4. Дьяконов В.П. Справочник по применению системы PC MathLAB. – М.: Наука, ФизМфалит, 1993.
5. Потемкин В.Г. MATLAB: справочное пособие – М.: ДИАЛОГ-МИФИ, 1997 – подробный справочник по основным функциям системы MATLAB.
6. Потемкин В.Г. MATLAB 5 для студентов – М.: ДИАЛОГ-МИФИ, 1998.
7. Потемкин В.Г. Система инженерных и научных расчетов MATLAB 5.x. В 2-х т. – М.: ДИАЛОГ-МИФИ, 1999.
8. Дьяконов В.П. MATLAB: учебный курс. – СПб: ПИТЕР, 2001. – полезная книга для начинающих пользователей. Курс изучения системы разбит на 20 уроков по разным темам.
9. Потемкин В.Г. Инструментальные средства MATLAB 5.x – М.: ДИАЛОГ-МИФИ, 2000. – Книга посвящена вопросам создания интерфейсов в среде MATLAB на основе технологии GUI и созданию самостоятельно выполняемых приложений. Рекомендуются для тех, кто уже знаком с основами программирования в среде пакета MATLAB.