МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Особенности системы MatLAB для решения задач вычислительной математики

Рекомендовано в качестве учебного пособия
Редакционно-издательским советом
Томского политехнического университета

Составитель
Е.А. Кочегурова

Издательство
Томского политехнического университета
2013

УДК 519.6 (075.8)

Особенности системы MatLAB для решения задач вычислительной математики: учебное пособие / сост. Е.А. Кочегурова; Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2013. – 110 с.

Учебное пособие предназначено для *самостоятельного* освоения приемов работы в системе проведения математических и инженерных расчетов MatLab. Особое внимание уделено изучению основ программирования собственных задач пользователя.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

За многие годы накоплены обширные библиотеки научных подпрограмм на различных алгоритмических языках, предназначенных для решения типовых задач вычислительной математики. Кроме того, имеется целый ряд различных математических пакетов, реализующих разнообразные численные методы и производящих аналитические математические преобразования. Наиболее известными сегодня являются пакеты прикладных программ (ППП) и математические библиотеки: MatLab (фирма The MathWorks), Maple (фирма Waterloo Maple Inc), Mathematica (фирма Wolfram Research), MathCAD (фирма MathSoft Inc).

MatLab – MATrix LABoratory (матричная лаборатория). Эта система предназначена для осуществления любых численных расчетов и моделирования технических и физических систем, а также выполнения научных и инженерных расчетов при работе с массивами данных;

MathCAD является интегрированной системой программирования, ориентированной на проведение математических и инженерно-технических расчетов.

Maple — программный пакет, система компьютерной алгебры. Система Maple предназначена для символьных вычислений, хотя имеет ряд средств и для численного решения дифференциальных уравнений и нахождения интегралов. Обладает развитыми графическими средствами. Имеет собственный язык программирования, напоминающий Паскаль.

Mathematica — система компьютерной алгебры компании Wolfram Research. Содержит множество функций, как для аналитических преобразований, так и для численных расчётов. Кроме того, программа поддерживает работу с графикой и звуком, включая построение двух- и трёхмерных графиков функций, рисование произвольных геометрических фигур, импорт и экспорт изображений и звука.

1. ОБЗОР MATLAB

1.1. Система математических расчетов MATLAB

Система использует математический сопроцессор и допускает обращения к программам, написанным на языках Fortran, C и C++.

Наиболее известные области применения системы MATLAB:

- математика и вычисления;
- разработка алгоритмов;
- вычислительный эксперимент, имитационное моделирование;
- анализ данных, исследование и визуализация результатов;
- научная и инженерная графика;
- разработка приложений, включая графический интерфейс пользователя.

MATLAB – это интерактивная система, основным объектом которой является массив, для которого не требуется указывать размерность явно. Это позволяет решать многие вычислительные задачи, связанные с векторно-матричными формулировками, существенно сокращая время, необходимое для программирования на скалярных языках типа Fortran или C. Будучи ориентированной на работу с реальными данными, эта система выполняет все вычисления в арифметике с плавающей точкой, в отличие от систем компьютерной алгебры REDUCE, MACSYMA, DERIVE, Maple, Mathematica, Theorist, где преобладает целочисленное представление и символьная обработка данных.

Система MATLAB – это одновременно и операционная среда и язык программирования. Одна из наиболее сильных сторон системы состоит в том, что на языке MATLAB могут быть написаны программы для многократного использования. Пользователь может сам написать специализированные функции и программы, которые оформляются в виде M-файлов. По мере увеличения количества созданных программ возникают проблемы их классификации и тогда можно попытаться собрать родственные функции в специальные папки. Это приводит к концепции пакетов прикладных программ (Application Toolboxes или просто Toolboxes), которые представляют собой коллекции M-файлов для решения определенной задачи или проблемы.

В действительности Toolboxes – это нечто большое, чем просто набор полезных функций; часто это результат работы многих исследователей по всему миру, которые объединяются в группы по самым различным интересам, начиная от нейтронных сетей, дифференциальных уравнений в частных производных, сплайн-аппроксимации, статистики и размытых множеств до проектирования робастных систем управления, теории сигналов, идентификации, а также моделирования линейных и нелинейных динамических систем с помощью исключительно эффективного пакета SIMULINK. Именно поэтому пакеты прикладных программ MATLAB Application Toolboxes, входящие в состав семейства продуктов MATLAB, позволяют находиться на уровне самых современных мировых достижений в разных областях науки и техники.

1.2. Рабочий стол (desktop) системы MATLAB

После запуска MATLAB на экране появляется основное окно системы MATLAB. Обычно это окно раскрыто не полностью и занимает часть рабочего стола. Вы можете раскрыть окно полностью, щелкнув на средней из трех кнопок, расположенных в конце титульной (верхней) строки окна. Левая кнопка сворачивает окно в кнопку с именем приложения, помещаемую в панель задач Windows, а правая закрывает окно и прекращает работу с MATLAB.

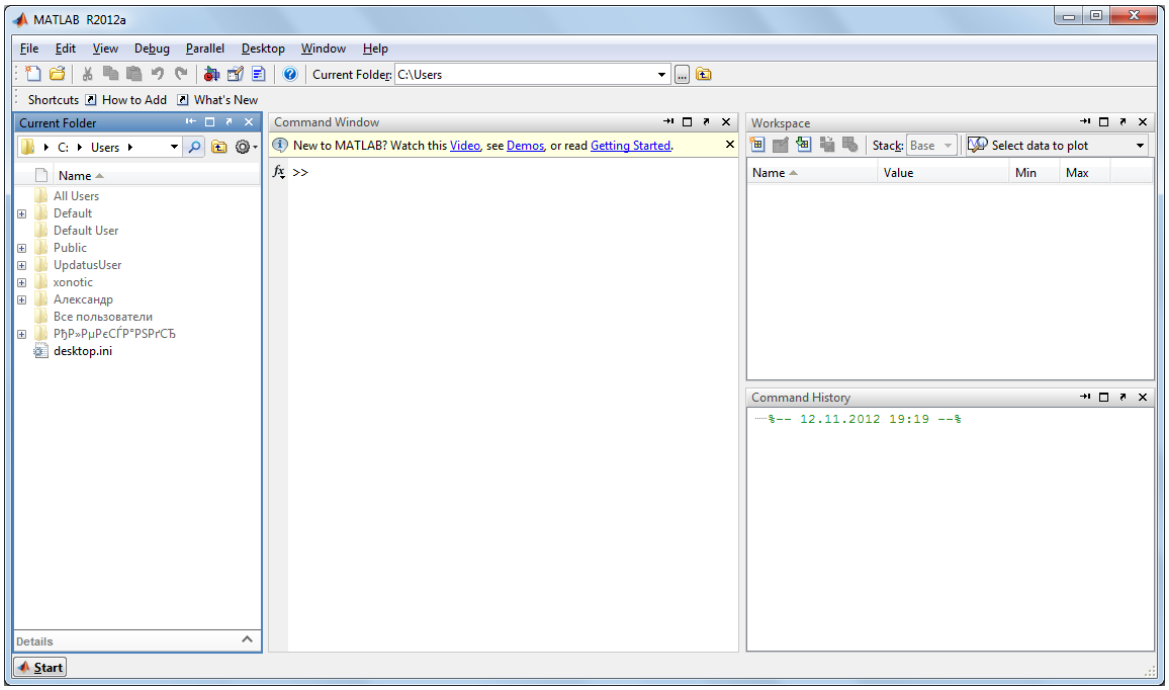


Рис. 1.1. Основное окно системы MATLAB

Сеанс работы с MATLAB принято именовать *сессией* (session). Сессия, в сущности, является текущим документом, отражающим работу пользователя с системой MATLAB. В ней имеются строки ввода, вывода и сообщений об ошибках.

Входящие в сессию определения переменных и функций, расположенные в рабочей области памяти, но не саму сессию, можно записать на диск (файлы формата .mat), используя команду save (Сохранить).

Команда load (Загрузить) позволяет считать с диска данные рабочей области. Фрагменты сессии можно оформить в виде дневника с помощью команды diary (Дневник).

Окна системы вполне отвечают канонам современного интерфейса. Пользовательский интерфейс многооконный и имеет ряд средств прямого доступа к различным компонентам системы.

Рабочий стол системы MATLAB содержит следующие инструментальные окна, часть из которых не появляется при начальном запуске:

- Command Window (Командное Окно) – Выполняет все функции и команды системы MATLAB.
- Command History (История Команд) – Просмотр функций, введенных ранее в Command Window, их копирование и выполнение.
- Launch Pad (Окно Запуска) – Запускает все инструменты и обеспечивает доступ ко всем пакетам системы MATLAB.
- Current Directory Browser (Окно Просмотра Текущего Каталога) – Просмотр файлов MATLAB, а также сопутствующих файлов, а также выполнение таких операций над файлами, как поиск и открытие файлов.
- Help Browser (Окно Просмотра Помощи) – Поиск и просмотр документации по всем функциям и средствам системы MATLAB.
- Workspace Browser (Окно Просмотра Рабочего Пространства) – Просмотр и изменение содержания рабочего пространства (workspace) системы MATLAB.
- Array Editor (Редактор Массивов Данных) – Просмотр содержимого массивов данных, записанных в виде таблицы и редактирование данных.
- Editor/Debugger (Редактор/Отладчик) – Для создания, редактирования и отладки М-файлов, т.е. файлов, содержащих функции системы MATLAB.

Полезно сразу усвоить некоторые команды управления окном командного режима:

- clc — очищает экран и размещает курсор в левом верхнем углу пустого экрана.
- home — возвращает курсор в левый верхний угол окна.
- echo <file_name> on — включает режим вывода на экран текста Script-файла (файла-сценария).
- echo <file_name> off — выключает режим вывода на экран текста Script-файла.
- echo <fi1e_name> — меняет режим вывода на противоположный.

- echo on all — включает режим вывода на экран текста всех m-файлов.
- echo off all — отключает режим вывода на экран текста всех m-файлов.
- more on — включает режим постраничного вывода (полезен при просмотре больших m-файлов).

more off — отключает режим постраничного вывода (в этом случае для просмотра больших фалов надо пользоваться линейкой прокрутки).

1.3. Конфигурация рабочего стола MATLAB

Полезно знать, что в начале запуска автоматически выполняется команда rnatl abrc, которая исполняет загрузочный файл matlabrc.m и файл startup.m, если таковой существует. Эти файлы выполняют начальную настройку терминала системы и задают ряд ее параметров. В частности, могут быть заданы пути доступа к другим файлам, необходимым для корректной работы системы MATLAB.

Таким образом, опытные пользователи могут выполнить настройку системы под свои запросы. Однако в большинстве случаев особой необходимости в этом нет. Поскольку указанные файлы имеют текстовый формат, их легко просмотреть с помощью какого-либо текстового редактора или с помощью команды type в командном режиме работы MATLAB.

Общий вид рабочего окна MATLAB представлен ниже (рис.1). Каждое из перечисленных окон может быть выведено из конфигурации рабочего стола нажатием кнопки со стрелкой в верхнем правом углу окна (см. рис. 1).

Обратная операция, то есть ввод в общую конфигурацию, осуществляется выбором опции Dock в меню View сответствующего окна. Можно также изменить конфигурацию рабочего стола путем перемещения любого открытого окна в новое положение. Для этого нужно просто нажать левой клавишей мыши на выбранное название окна (Title Bar) и «перетащить» его в желаемое положение.

Для восстановления стандартной конфигурации рабочего стола MATLAB необходимо выбрать опцию Default (По Умолчанию) в подменю Desktop Layout (План Рабочего Стола) в меню View (Вид)

любого открытого окна системы. Все окна MATLAB содержат также контекстное меню (context menu), которое вызывается нажатием правой кнопки мыши и содержит наиболее часто применяемые опции (функции), связанные с данным окном.

Таким образом, в системе MATLAB имеется возможность изменения вида рабочего стола путем открытия, закрытия, перемещения или изменения размеров каждого из индивидуальных окон.

1.4. Справка и текущая документация

MATLAB имеет интерактивную систему помощи, которая реализуется в командном режиме с помощью ряда команд. Одной из них является команда

```
» help
```

которая выводит весь список папок, содержащих m-файлы с определениями операторов, функций и иных объектов, присущих конкретной реализации системы MATLAB.

Следует отметить, что набор входящих в список средств зависит от набора пакетов расширения, которыми располагает версия системы MATLAB, заказанная или полученная конкретным пользователем. Этот набор может заметно отличаться от приведенного. Кроме того, в MATLAB для доступа к пакетам расширения может потребоваться указание пути к их файлам на диске в панели браузера файловой системы.

Для получения справки по какому-либо конкретному объекту используются команды

```
» help имя
```

или

```
» doc имя
```

где имя — имя объекта, для которого требуется вывод справочной информации. Ниже дается пример для функции вычисления гиперболического синуса, намеренно введенной с неверным указанием имени:

```
» help hsln hsin.m
not found.
```

Нетрудно заметить, что система помощи сообщает, что для функции с именем hsin соответствующий m-файл отсутствует. Введем имя верно:

```
» help slnh
SINH Hyperbolic sine.
SINH(X) Is the hyperbolic sine of the elements of
X.
Overloaded methods
help sym/slnh.m
```

Теперь полученное сообщение содержит информацию о функции slnh. Хотя имена функций в MATLAB задаются малыми (строчными) буквами, в сообщениях справочной системы имена функций и команд выделяются большими (прописными) буквами. Этот не слишком удачный прием использован для выделения заголовка текста справки в виде имени функции. В данной книге мы отказались от такого приема, вводящего начинающих пользователей в заблуждение.

Аналогичным образом можно получить справку по константам и другим объектам языка MATLAB. Ниже дан пример обращения к справке о числе *n:*

```
» help pi
PI 3.1415926535897
PI= 4*atan(1) = imagdog(-l)) = 3.1415926535897
```

При всей примитивности справки help надо отметить ее высокую эффективность. Особенно популярна интерактивная справка у пользователей, привыкших к работе с языками программирования, которые используются в среде операционной системы MS-DOS. Справка doc имя выводит более полную информацию в окне помощи в формате HTML.

Пользователя системы MATLAB часто интересует набор функций, команд или иных понятий, относящихся к определенной группе объектов. Выше были указаны имена основных групп объектов системы MATLAB.

Ниже дан пример вызова справки по группе объектов timefun:

```
» help timefun
Time and dates.
Current date and time.
Now - Current date and time as date num-
ber.
Date - Current date as date string.
clock - Current date and time as date v
tor.
Basic functions.
datenum - Serial date number.
datestr - String representation of date.
datevec - Date components.
Date functions.
calendar - Calendar.
weekday - Day of week.
eomday - End of month.
datetick - Date formatted tick labels.
Timing functions.
cputime - CPU time in seconds.
tic. toe - Stop watch timer.
etime - Elapsed time.
pause - Wait in seconds.
```

После уточнения состава определенной группы объектов можно получить детальную справку по любому выбранному объекту. Как это делается, было описано выше.

Ввиду обилия в системе MATLAB m-функций, число которых около 800, важное значение имеет поиск m-функций по *ключевым словам.* Для этого служит команда

```
lookfor Ключевое слово
```

или

```
lookfor 'Ключевые слова'
```

В первом случае ищутся все m-файлы, в заголовках которых встречается заданное ключевое слово, и заголовки обнаруженных файлов выводятся на экран. Следует отметить, что широкий поиск по одному ключевому слову может подчас привести к выводу многих десятков определений и длится довольно долго.

Для уточнения и сокращения зоны поиска следует использовать вторую форму команды lookfor. Вот пример ее применения:

```
» lookfor 'inverse sin'
ASIN Inverse sine.
ASIN Symbolic inverse sine.
```

В данном случае для поиска использованы слова ′ inverse sin', т. е. задан поиск арксинуса. Система поиска нашла только два вида арксинуса ASIN — обычного и в символьной форме. Число найденных определений зависит от того, с каким числом пакетов прикладных программ (пакетов расширений) поставляется версия системы MATLAB.

В командном режиме можно получить справочные данные с помощью ряда команд:

- computer — выводит сообщение о типе компьютера, на котором установлена текущая версия MATLAB;
- help script — выводит сообщение о назначении m-файлов сценариев (Script-файлов);
- helpf unction — выводит сообщение о назначении и структуре m-файлов функций;
- info — выводит информацию о фирме Math Works с указанием адресов электронной почты;
- subscribe — позволяет создать файл с бланком регистрации;
- ver — выводит информацию о версиях установленной системы MATLAB и ее компонентов;
- version — выводит краткую информацию об установленной версии MATLAB,
- version -Java — выводит информацию об установленной в составе MATLAB версии Ява (Java);
- what — выводит имена файлов текущего каталога;
- what name — выводит имена файлов каталога, заданного именем name;
- whatsnew name — выводит на экран содержимое файлов readme заданного именем name класса для знакомства с последними изменениями в системе и в пакетах прикладных программ;
- which name — выводит путь доступа к функции с данным именем.

2. ПРОГРАММИРОВАНИЕ MATLAB

До сих пор мы в основном использовали систему MATLAB в режиме непосредственного счета — в командном режиме. Однако при решении серьезных задач возникает необходимость сохранения используемых последовательностей вычислений, а также их дальнейшей модификации. Иными словами, существует необходимость программирования решения задач.

Это может показаться отходом от важной цели, которая преследуется разработчиками большинства математических систем, — выполнения математических вычислений без использования традиционного программирования. Однако это не так. Множество математических задач решается в системе MATLAB без программирования. С использованием языков высокого уровня для их решения потребовалось бы написать и оттестировать сотни программ.

Практически невозможно предусмотреть в одной, даже самой большой и мощной, математической системе возможность решения всех задач, которые могут интересовать пользователя. Программирование в системе MATLAB является эффективным средством ее расширения и адаптации к решению специфических проблем. Оно реализуется с помощью языка программирования системы.

Большинство объектов этого языка, в частности все команды, операторы и функции, одновременно являются объектами входного языка общения с системой в командном режиме работы.

В основном отличие входного языка от языка программирования в способе фиксации создаваемых ими кодов. Сессии в командном режиме работы не сохраняются в памяти компьютера. Хранятся только определения созданных в ходе их выполнения переменных и функций. А вот программы на языке программирования MATLAB сохраняются в виде текстовых m-файлов. При этом могут сохраняться как целые программы в виде файлов-сценариев, так и отдельные программные модули — функции. Кроме того, важно, что программа может менять структуру алгоритмов вычислений в зависимости от входных данных и данных, создаваемых в ходе вычислений.

С позиций программиста язык программирования системы является типичным проблемно-ориентированным языком программирования высокого уровня. Точнее говоря, это даже язык сверхвысокого уровня, содержащий сложные операторы и функции, реализация которых на обычных языках (например Бейсике, Паскале или Си) потребовала бы много усилий и времени.

2.1. Основы редактирования и отладки m-файлов

2.1.1. Интерфейс редактора/отладчика m-файлов

Для подготовки, редактирования и отладки m-файлов служит специальный многооконный редактор. Он выполнен как типичное приложение Windows. Редактор можно вызвать командой *Edit* из командной строки или командой *New -> M-file* из меню *File*. После этого в окне редактора можно создавать свой файл, пользоваться средствами его отладки и запуска. Перед запуском файла его необходимо записать на диск, используя команду *File -> Save as* в меню редактора.

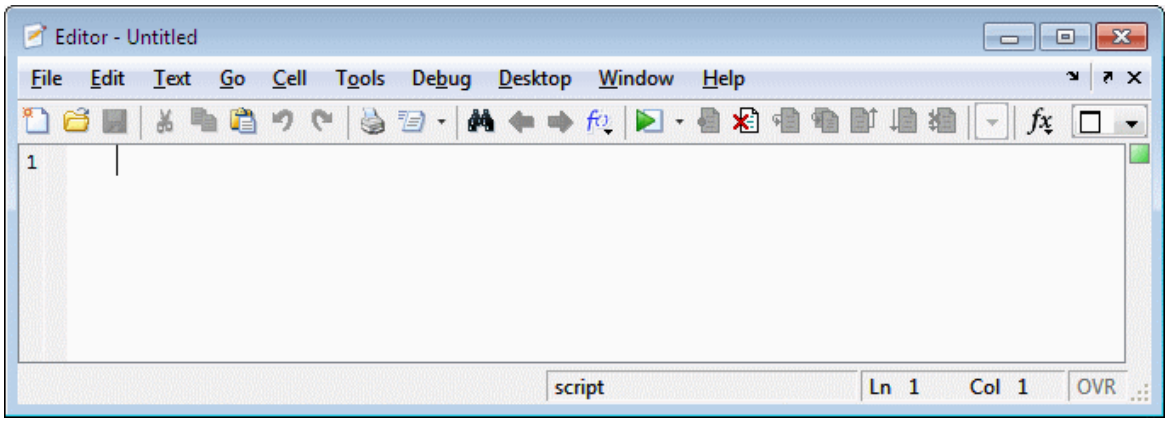


Рис. 2.1. Окно редактора/отладчика MATLAB

Выше показано окно редактора/отладчика MATLAB..

После записи файла на диск можно заметить, что команда *Run* в меню *Tools* (Инструменты) редактора становится активной (до записи файла на диск она пассивна) и позволяет произвести запуск файла. Запустив команду *Run*, можно наблюдать исполнение m-файла.

Редактор/отладчик позволяет создать m-файл (программу). Далее мы убедимся, что текст такого файла подвергается тщательной синтаксической проверке, в ходе которой выявляются и отсеиваются многие

ошибки пользователя. Таким образом, редактор обеспечивает синтаксический контроль файла.

Редактор имеет и другие важные отладочные средства — он позволяет устанавливать в тексте файла специальные метки, именуемые точками прерывания (breakpoints). При их достижении вычисления приостанавливаются, и пользователь может оценить промежуточные результаты вычислений (например, значения переменных), проверить правильность выполнения циклов и т. д.

Для удобства работы с редактором/отладчиком строки программы в нем нумеруются в последовательном порядке. Редактор является многооконным. Окно каждой программы оформляется как вкладка.

2.1.2. Цветовые выделения и синтаксический контроль

Редактор/отладчик m-файлов выполняет синтаксический контроль программного кода по мере ввода текста. При этом используются следующие цветовые выделения:

- ключевые слова языка программирования — синий цвет;
- операторы, константы и переменные — черный цвет;
- комментарии после знака % — зеленый цвет;
- символьные переменные (в апострофах) — зеленый цвет;
- синтаксические ошибки — красный цвет.

Благодаря цветовым выделениям вероятность синтаксических ошибок снижается.

Однако далеко не все ошибки диагностируются. Ошибки, связанные с неверным применением операторов или функций (например, применение оператора - вместо + или функции cos(x) вместо sin(x) и т. д.), не способна обнаружить ни одна система программирования. Устранение такого рода ошибок (их называют семантическими) — дело пользователя, отлаживающего свои алгоритмы и программы.

2.1.3. Панель инструментов редактора и отладчика

Редактор имеет свое меню и свою инструментальную панель. По стилю данная панель похожа на панель инструментов окна командного режима работы, но имеет несколько иной набор кнопок.

Рис. 2.2. Панель инструментов редактора и отладчика

Назначение кнопок панели инструментов редактора/отладчика следующее:

- *New* — создание нового m-файла;
- *Open* — вывод окна загрузки файла;
- *Save* — запись файла на диск;
- *Print* — печать содержимого текущего окна редактора;
- *Cut* — вырезание выделенного фрагмента и перенос его в буфер;
- *Copy* — копирование выделенного объекта в буфер;
- *Paste* — размещение фрагмента из буфера в позиции текстового курсора;
- *Undo* — отмена предшествующей операции;
- *Redo* — повтор отмененной операции;
- *Find text* — нахождение указанного текста;
- *Show function* — показ функции;
- *Set/Clear Breakpoint* — установка/сброс точки прерывания;
- *Clear All Breakpoints* — сброс всех точек прерывания;
- *Step* — выполнение шага трассировки;
- *Step In* — пошаговая трассировка с заходом в вызываемые m-файлы;
- *Step Out* — пошаговая трассировка без захода в вызываемые m-файлы;
- *Save and Run* — сохранение и запуск;
- *Exit Debug Mode* — выход из режима отладки.

2.1.4. Алфавит языка программирования

В MATLAB, как и в других системах, используются все буквы латинского алфавита от A до Z и арабские цифры от 0 до 9. Как и в C++, большие и малые буквы это разные символы. Кроме букв латинского алфавита используются все специальные символы клавиатуры компьютера.

2.2. Понятие о файлах-сценариях и файлах-функциях

Здесь полезно отметить, что m-файлы, создаваемые редактором/отладчиком, делятся на два класса:

- *файлы-сценарии*, не имеющие входных параметров;
- *файлы-функции*, имеющие входные параметры.

2.2.1. Файл-сценарий

Файл сценарий называется также *Script-файлом* или просто *скриптом*.

Файл-сценарий имеет весьма простую *структуру*:

1. % Основной комментарий, если необходимо.
2. % Дополнительный комментарий, если необходимо.
3. Тело программы с любыми допустимыми выражениями.

Важными являются следующие *свойства файлов-сценариев*:

1. Они не имеют входных и выходных аргументов.
2. Работают с данными из рабочей области.
3. В процессе выполнения не компилируются.
4. Представляют собой последовательность операций, аналогичную той, что используется в сеансе работы из командной строки.

Пример:

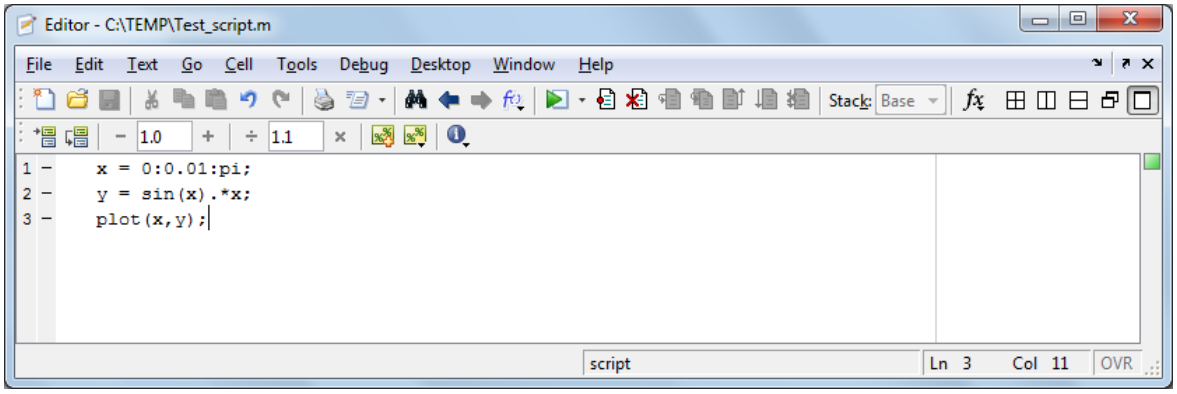


Рис. 2.3. Пример файла-сценария

2.2.2. Файл-функция

Отличие М-файла функции от сценария состоит в том, что он является аналогом подпрограммы типа function в языке Pascal.

М-файл функция обладает следующими свойствами:

1. Он начинается с ключевого слова **function**, после которого указывается имя переменной var – выходного параметра, имя самой функции f _ name и список ее входных параметров, отделенных запятой.
   Внимание: Имя М-файла функции должно совпадать с именем самой функции (f _ name). MATLAB автоматически присваивает данное имя при выполнении команды *Save as*.
2. Результат выполнения М-файла функции присваивается имени функции, которое может использоваться в математических выражениях подобно функциям sin(x), log(x) и т. п.
3. Все переменные, используемые в файле-функции, являются локальными, т.е. действуют только в пределах тела функции.
4. Последняя конструкция **var=выражение** вводится, если требуется, чтобы функция возвращала результат вычислений. Вместо имени var можно ипользовать любое другое имя.
5. Файл-функция является самостоятельным программным модулем, который связан с другими модулями и головной программой через входные и выходные параметры.
6. При вызове файла-функции он компилируется и затем исполняется.
7. m-файл функция должен сохраняться в ваш рабочий каталог.

Структура М-файла функции с одним выходным параметром имеет вид:

- function var = f _ name (Список параметров)
- % Основной комментарий, если необходимо.
- % Дополнительный комментарий, если необходимо.
- Тело программы с любыми выражениями.
- var = выражение

Структура М-файла функции с двумя выходными параметрами имеет вид:

- function [var1 var2] = f _ name (Список параметров)
- % Основной комментарий, если необходимо.
- % Дополнительный комментарий, если необходимо.
- Тело программы с любыми выражениями,
- где обязательно присутствуют операторы:
- var1 = выражение

- и
- var2 = выражение

В общем случае выходной параметр - матрица.

Пример:

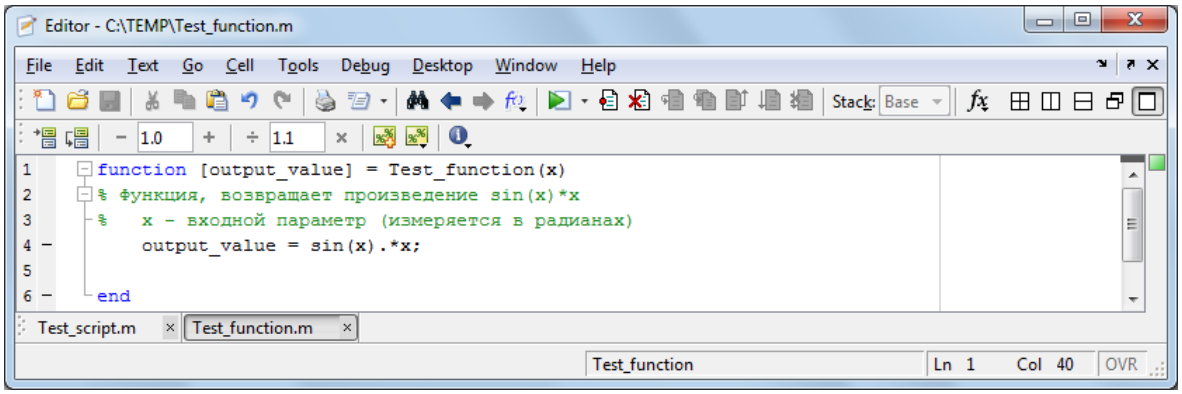


Рис. 2.4. Пример файла-функции

Данную функцию можно использовать в m-файле сценарии Test_script.m, например, следующим образом:

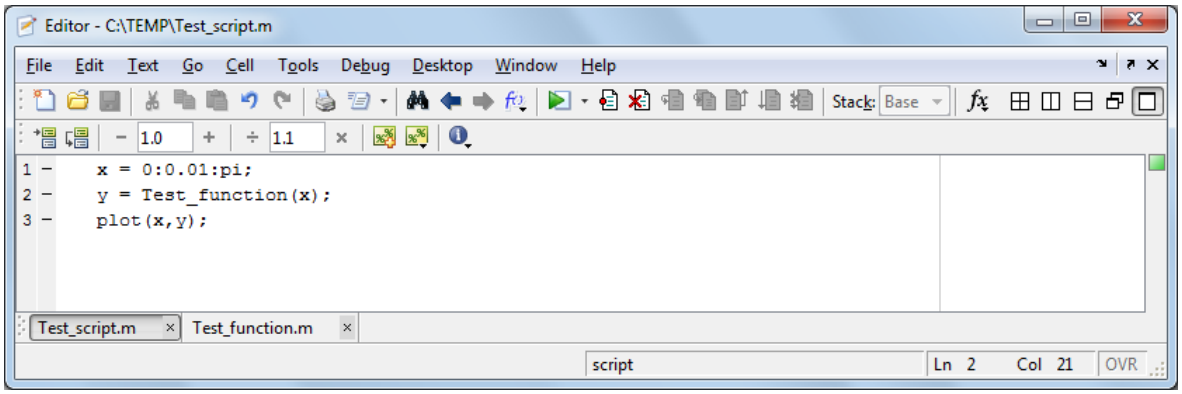


Рис. 2.5. Пример использования файла-сценария

2.3. Операторы

2.3.1. Арифметические и логические операторы

Список *арифметических операторов*.

Таблица 1. Список арифметических операторов

Функция	**Обозначение (синтаксис)**
Сложение	**+ (M1+M2)**
Вычитание	**- (M1-M2)**
Умножение чисел или матриц	*** (M1*M2)**
Поэлементное умножение матриц	**.* (M1.*M2)**
Возведение числа или матрицы в степень	**^ (M1^ *x*)**
Поэлементное возведение матрицы в степень	**.^ (M1.^ *x*)**
Деление чисел или деление матриц слева направо	**/ (M1 / M2)**
Поэлементное деление матриц слева направо	**./ (M1 ./ M2)**
Деление матриц справа налево	**\ (M1 \ M2)**
Поэлементное деление матриц справа налево	**.\ (M1 .\ M2)**

В математических выражениях операторы имеют определенный приоритет исполнения. В MATLAB приоритет логических операторов выше, чем арифметических, приоритет возведения в степень выше приоритетов умножения и деления, приоритет умножения и деления выше сложения и вычитания. Для повышения приоритета операций нужно использовать круглые скобки. Степень вложения скобок не ограничивается.

Операторы отношения служат для сравнения двух величин, векторов или матриц, все операторы отношения имеют две сравниваемые величины и записываются, как показано в таблице:

Таблица 2. Список операторов отношения

Функция	**Оператор (синтаксис)**
Равно	**= = (*x* = = *y*)**
Не равно	**~ = (*x* ~ = *y*)**
Меньше	**< (*x* < *y*)**
Больше	**> (*x* > *y*)**
Меньше или равно	**< = (*x* < = *y*)**
Больше или равно	**> = (*x* > = *y*)**

Данные операторы выполняют поэлементное сравнение векторов или матриц одинакового размера и логическое выражение принимает значение 1 (*True*), если элементы идентичны, и значение 0 (*False*) в противном случае.

Логические операторы служат для реализации поэлементных логических операций над элементами одинаковых по размеру массивов согласно таблице:

Таблица 3. Список логических операторов

Функция	**Оператор (синтаксис)**
Логическое И	**&; *and* (*and* (*a, b*))**
Логическое ИЛИ	**|; *or* (*or* (*a, b*))**
Логическое НЕ	**~ ; *not* (*not* (*a, b*))**
Исключающее ИЛИ	***xor* (*xor* (*a, b*))**
Верно, если все элементы вектора равны нулю	***any* (*any* (a))**
Верно, если все элементы вектора не равны нулю	***all* (*all* (*a*))**

2.3.2. Оператор присваивания

Программирование в системе MATLAB является средством ее расширения и использовании в решении специфических проблем. Некоторые вопросы программирования изложены выше, здесь рассмотрим

правила, дополняющие синтаксис языка MATLAB.

Программы оперируют с переменными и константами. Переменные – это имеющие имена объекты, способные хранить разные по значению данные. В зависимости от этих данных переменные могут быть числовыми или символьными, векторными или матричными.

Для задания переменным определенных значений используется *оператор присваивания*, вводимый знаком равенства =

Имя _ переменной = Выражение ;

Типы переменных заранее не объявляются. Они определяются выражением, значение которого присваивается переменной.

Имя переменной может содержать сколько угодно символов, но идентифицируется только 31 начальный символ. Имя любой переменной должно быть уникальным. Имя должно начинаться с буквы, может содержать буквы, цифры и символ подчеркивания _. Недопустимо включать в имена пробелы и специальные знаки.

2.3.3. Операторы циклов

Циклы типа *for … end* используется для организации цикла с фиксированным числом повторений. Он имеет вид:

```
for  var = Выражение
     Операторы
end ;
```

Здесь *var* – счетчик цикла – любая переменная, обычно это *i, j, k, l, m* и т. д.

Выражение записывается в виде *s : d : e*, где *s* – начальное значение счетчика цикла *var, d* – шаг изменения и *e* – конечное значение *var*. Возможна и запись в виде *s : e*, тогда *d* = 1.

Список операторов завершается ключевым словом *end*.

- Оператор *continue* передает управление в следующую итерацию цикла, пропуская операции, которые записаны за ним.

- Оператор *break* используется для досрочного прерывания цикла.

Возможны вложенные циклы

```
for  i = 1 : 3
        for  j = 1 : 3
           a (i, j) = i * j ;
        end ;
     end ;
```

Циклы типа *while … end* выполняются до тех пор, пока выполняется заданное условие. Оператор записывается в виде:

```
while  Логическое условие
     Операторы
end ;
```

2.3.4. Условный оператор

В некоторых вариантах индивидуального задания требуется вычислить значение переменной в зависимости от некоторого условия. Это можно сделать с помощью условного оператора.

Условный оператор в языке Matlab аналогичен условным операторам в других языках:

```
if <логическое выражение>
  <операторы>
elseif <логическое выражение>
  <операторы>
else
  <операторы>
end
```

Блоков типа elseif может быть более одного в пределах условного выражения. Блок else, а также и блоки elseif , могут отсутствовать вовсе. Т.е.:

```
if <логическое выражение>
  <операторы>
else
```

```
  <операторы>
end
```

или

```
if <логическое выражение>
  <операторы>
end
```

Следует отличать оператор elseif от пары операторов else if. В первой конструкции elseif начинает очередную ветвь условного выражения, во второй – оператор if начинает самостоятельное (вложенное) условное выражение, которое, конечно, должно заканчиваться отдельным оператором end.

Пример:

Написать функцию нахождения точки минимума методом деления отрезка пополам. Метод представить в m-файл функции, а вызов метода и m-файле сценарии.

Решение:

Шаг 1:

Создать m-файл, описывающий функцию, минимум которой необходимо найти

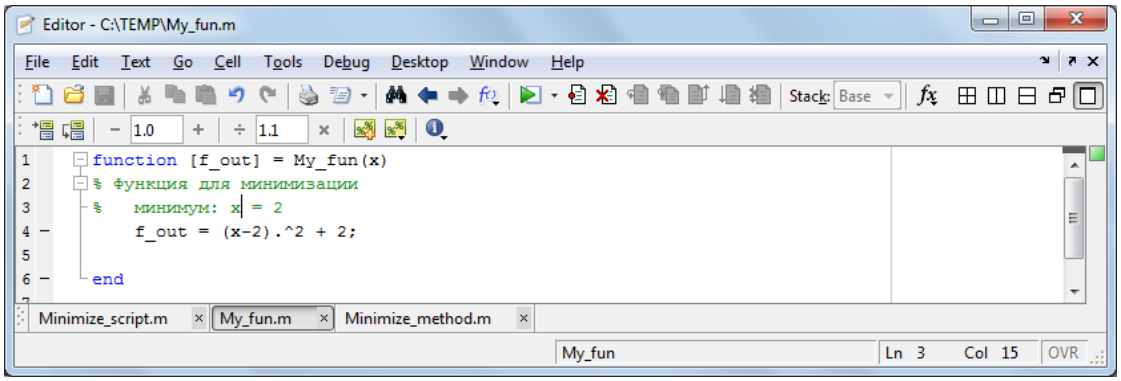


Рис. 2.6. Создание m-файла

Шаг 2:

Создать m-файл функцию, описывающую метод минимизации

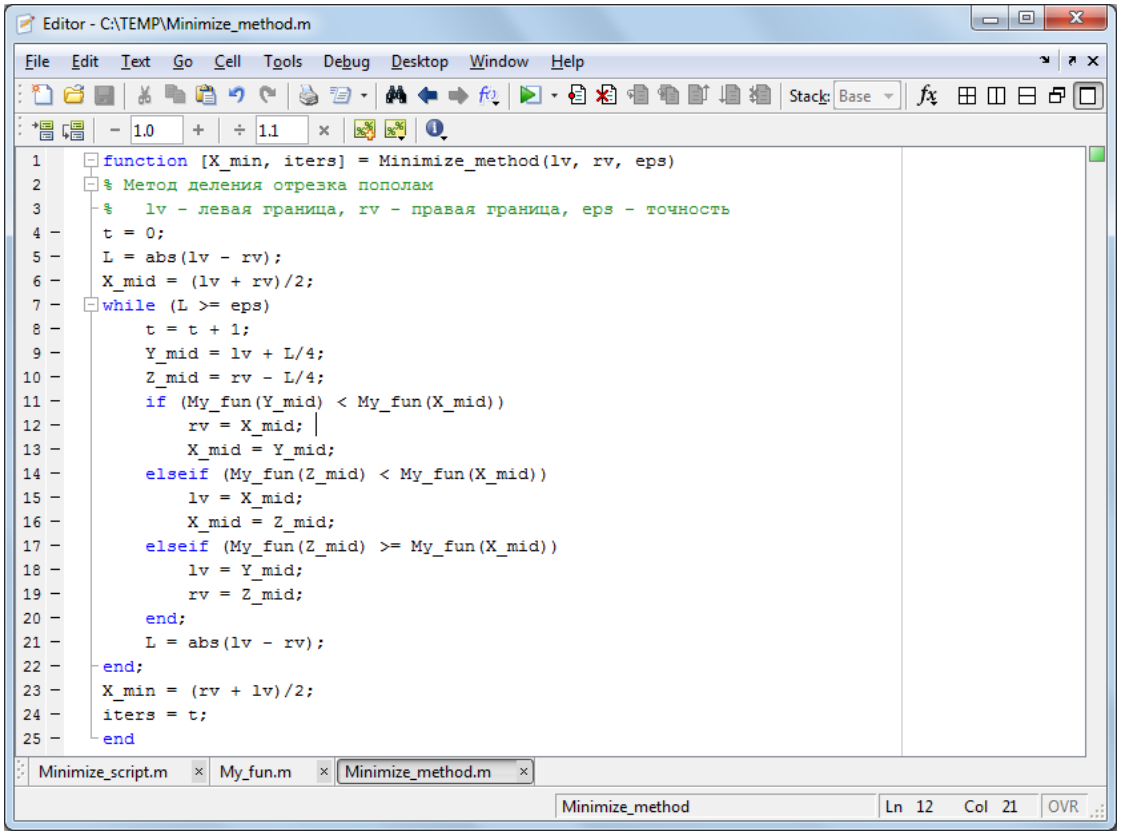


Рис. 2.7. Создание m-файла-функции

Шаг 3:

Создать m-файл сценарий, описывающий вызов метода минимизации

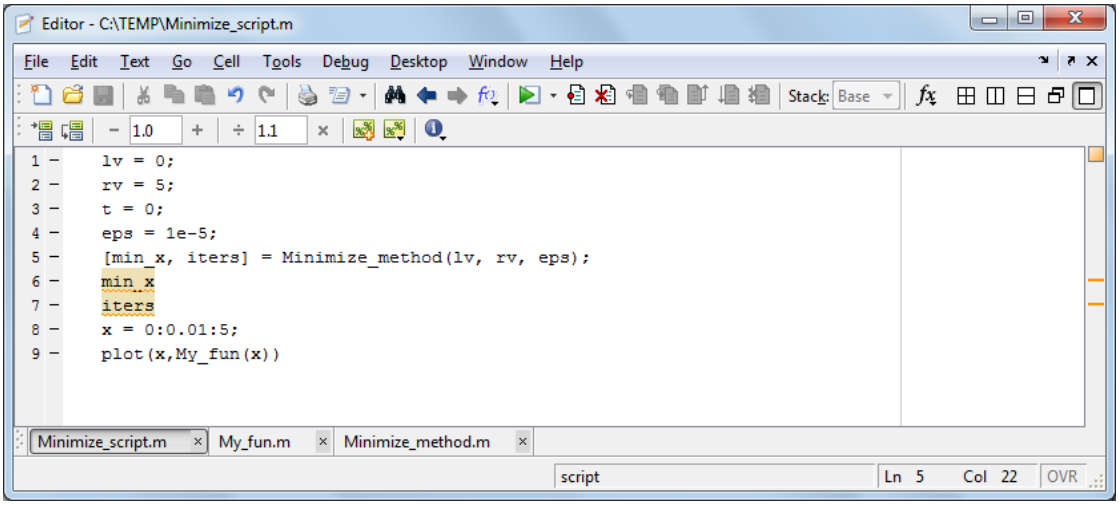


Рис. 2.8. Создание m-файла-сценария

Результатом работы файла сценария Minimize_script будут 2 значения в командном окне и график функции My_fun. Таким образом мы получим:

```
>> Minimize_script
min_x =
2.0000
iters =
19
```

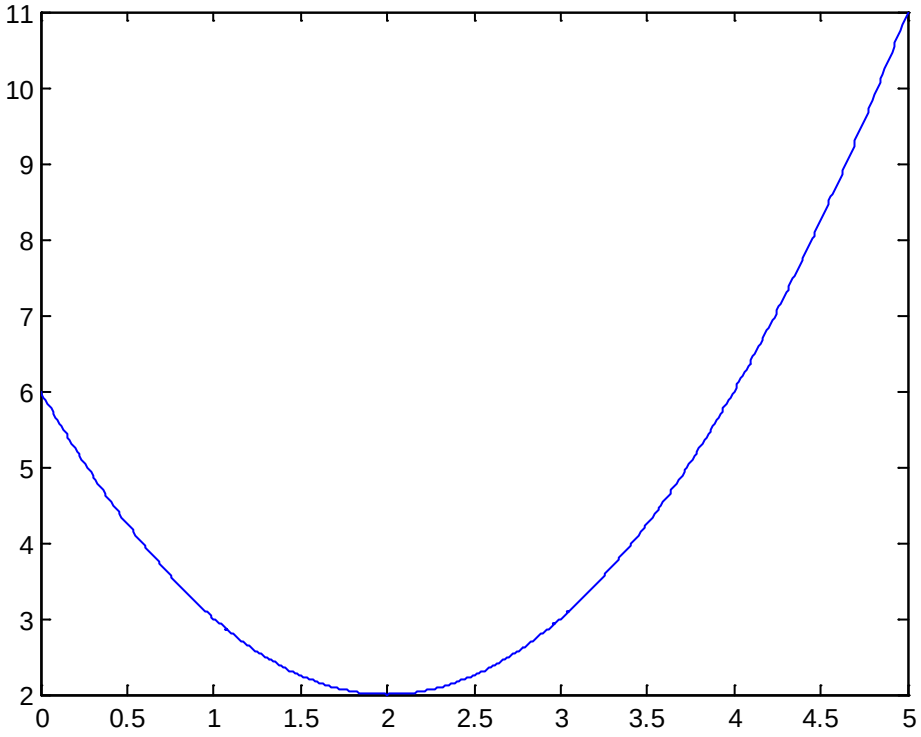


Рис. 2.9. График функции My_fun

Из графика легко убедиться, что минимум функции найден верно, задача решена.

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.

2.4. Формат вывода результата вычислений

По умолчанию MATLAB выдает числовые результаты в *нормализованной форме* с четырьмя цифрами после десятичной точки и одной до нее. Многих такая форма представления не всегда устраивает.

Поэтому при работе с числовыми данными можно задавать различные *форматы* представления чисел. Однако в любом случае все вычисления проводятся с предельной, так называемой *двойной,* точностью. Для установки формата представления чисел используется команда » format name где name — имя формата.

Для числовых данных name может быть следующим сообщением:

- short — короткое представление в фиксированном формате (5 знаков)
- short e — короткое представление в экспоненциальном формате (5 знаков мантиссы и 3 знака порядка)
- long — длинное представление в фиксированном формате (15 знаков)
- long e — длинное представление в экспоненциальном формате (15 знаков мантиссы и 3 знака порядка)
- hex — представление чисел в шестнадцатеричной форме
- bank — представление для денежных единиц.

Для иллюстрации различных форматов рассмотрим вектор, содержащий два элемента-числа: `x=[4/3 1.2345e-6]`

В различных форматах их представления будут иметь следующий вид:

format short e	1.3333	0.0000
format short e	1.3333E+000	1.2345E-006
format long	1.333333333333338	0.000001234500000
format long e	1.333333333333338E+000	1.234500000000000E-006

format bank 1.33 0.00

Задание формата сказывается только на форме вывода чисел. Вычисления все равно происходят в формате двойной точности, а ввод чисел возможен в любом удобном для пользователя виде.

2.5. Command Window Preferences

Если требуется получить результат вычисления более точный, то откроем это окно, File -> Preferences и из раскрывающегося списка выберем нужный формат, например long. Нажмем на кнопку ОК. Результат будет отображаться в формате long с плавающей точкой и с 14 цифрами после десятичной точки.

2.6. Вычисление элементарных функций

Функции — это имеющие уникальные имена объекты, выполняющие определенные преобразования своих аргументов и при этом возвращающие результаты этих преобразований. Возврат результата — отличительная черта функций. При этом результат вычисления функции с одним выходным параметром подставляется на место ее вызова, что позволяет использовать функции в математических выражениях, например функцию sin в 2*sin(pi/2).

Функции в общем случае имеют список аргументов (параметров), заключенный в круглые скобки. Например, функция Бесселя записывается как bessel(NU,X). В данном случае список параметров содержит два аргумента — NU в виде скаляра и X в виде вектора. Многие функции допускают ряд форм записи, отличающихся списком параметров.

Если функция возвращает несколько значений, то она записывается в виде [Yl, Y2....]=func(Xl. X2...), где Yl, Y2,... — список выходных параметров и XI, X2.... — список входных аргументов (параметров).

Со списком элементарных функций можно ознакомиться, выполнив команду hel p elfun, а со списком специальных функций — с помощью команды help specfun.

Функции могут быть встроенными (внутренними) и внешними, или т-функциями. Так, встроенными являются наиболее распространенные элементарные функции например, sin(x) и exp(y), тогда как функция sinh(x) является внешней функцией.

Внешние функции содержат свои определения в m-файлах. Задание таких функций с помощью специального редактора m-файлов. Встроенные функции хранятся в откомпилированном ядре системы

MATLAB, в силу чего они выполняются предельно быстро.

2.7. Присвоение переменных

Переменные — это имеющие имена объекты, способные хранить некоторые, обычно разные по значению, данные. В зависимости от этих данных переменные могут быть числовыми или символьными, векторными или матричными.

В системе MATLAB можно задавать переменным определенные значения. Для этого используется операция *присваивания,* вводимая знаком равенства =: `Имя_переменной = Выражение`

Типы переменных заранее не декларируются. Они определяются выражением, значение которого присваивается переменной. Так, если это выражение — вектор или матрица, то переменная будет векторной или матричной.

Имя переменной (ее идентификатор) может содержать сколько угодно символов, но запоминается и идентифицируется только 31 начальный символ. Имя любой переменной не должно совпадать с именами других переменных, функций и процедур системы, т. е. оно должно быть уникальным. Имя должно начинаться с буквы, может содержать буквы, цифры и символ подчеркивания _. Недопустимо включать в имена переменных пробелы и специальные знаки, например +,.-, *, / и т. д., поскольку в этом случае правильная интерпретация выражений становится невозможной.

Желательно использовать содержательные имена для обозначений переменных, например speed_1 для переменной, обозначающей скорость первого объекта. Переменные могут быть обычными и *индексированными,* то есть элементами векторов или матриц (см. выше). Могут использоваться и *символьные* переменные, причем символьные значения заключаются в апострофы, например s='Demo'.

В памяти компьютера переменные занимают определенное место, называемое *рабочей областью* (workspace). Для очистки рабочей области используется функция clear в разных формах, например:

- clear — уничтожение определений всех переменных;
- clear x — уничтожение определения переменной x;
- clear a, b, c — уничтожение определений нескольких переменных.

Уничтоженная (стертая в рабочей области) переменная становится неопределенной. Использовать неопределенные переменные нельзя, и такие попытки будут сопровождаться выдачей сообщений об ошибке. Приведем примеры задания и уничтожения переменных:

```
» x=2*pi
x = 6.2832
» V=[l 2 3 4 5]
V = 1 2 3 4 5
»MAT
??? Undefined function or variable 'MAT'.
» MAT=[1 2 3 4; 5 6 7 8]
MAT=
1234
5678
» clear V
» V
??? Undefined function or variable 'V'.
» clear
» x
??? Undefined function or variable 'x'.
» M
??? Undefined function or variable 'M'.
```

Обратите внимание на то, что сначала выборочно стерта переменная V, а затем командой clear без параметров стерты все остальные переменные.

2.8. Сохранение рабочей среды

Переменные и определения новых функций в системе MATLAB хранятся в особой области памяти, именуемой рабочей областью. MATLAB позволяет сохранять значения переменных в виде бинарных файлов с расширением .mat. Для этого служит команда save, которая может использоваться в ряде форм:

- save fname — записывается рабочая область всех переменных в файле бинарного формата с именем fname.mat;
- save fname X — записывает только значение переменной X;
- save fname X Y Z — записывает значения переменных X, Y и Z.

После этих параметров можно указать ключи, уточняющие формат

записи файлов:

- -mat — двоичный MAT-формат, используемый по умолчанию;
- -ascii — ASCII-формат единичной точности (8 цифр);
- -ascii -double — ASCII-формат двойной точности (16 цифр);
- -ascii -double -tabs — формат с разделителем и метками табуляции;
- V4 — запись MAT-файла в формате версии MATLAB 4;
- -append — добавление в существующий MAT-файл.

Возможно использование слова save и в формате функции, а не команды, например: `save ('fname', 'varT', 'var2').`

В этом случае имена файлов и переменных задаются строковыми константами.

Следует отметить, что возможности сохранения всего текста сессии, формируемой в командном режиме, команда save не дает. Дело в том, что сессия является результатом проб и ошибок, и ее текст наряду с правильными определениями содержит сообщения об ошибках, переопределения функций и переменных.

Необходимости сохранять такое, обычно нет. А если есть — для этого служит команда diary, описанная чуть ниже. Тем не менее это не значит, что вы не имеете возможности записать только то рациональное зерно, которое родилось в ходе попыток реализации ваших алгоритмов и методов решения задач. Надо просто воспользоваться редактором и отладчиком, которые позволяют (после отладки программы) получить документ в корректной форме без синтаксических и иных ошибок. Такой документ сохраняется в текстовом формате в виде файла с расширением .m.

2.9. Просмотр переменных

При работе с достаточно большим количеством переменных необходимо знать, какие переменные уже использованы, а какие нет.

Для этой цели служит команда who, выводящая в командное окно MatLab список используемых переменных:

```
>>who
Your variables are: al      a2      a3
```

Команда whos позволяет получить более подробную информацию о переменных в виде таблицы:

```
>> whos
Name Size Bytes Class
```

```
al 1X1 8 double array
a2 1X1 8 double array
a3 1X1 8 double array
Grand total is 3 elements using 24 bytes
```

Первый столбик Name состоит из имен используемых переменных. То, что содержится в столбике Size, по существу, определяется основным принципом работы MatLab. Программа MatLab все данные представляет в виде массивов.

Переменные al, a2 и a3 являются двумерными массивами размера один на один. Каждая из переменных занимает по восемь байтов, как указано в столбике Bytes. Наконец, в последнем столбике Class указан тип переменных - double array, т.е. массив, состоящий из чисел двойной точности.

В строке под таблицей написано, что в итоге три элемента, т.е. переменные, занимают двадцать четыре байта. Оказывается, что представление всех данных в MatLab в виде массивов дает определенные преимущества. Как было показано выше для освобождения из памяти всех переменных используется команда clear.

Начиная с версии 6.0, появилось удобное средство для просмотра переменных рабочей среды - окно Workspace, для перехода к которому следует активизировать одноименную закладку. Данное окно содержит таблицу, аналогичную той, что выводится командой whos. Двойной щелчок по строке, соответствующей каждой переменной, приводит к отображению ее содержимого в отдельном окне, что особенно полезно при работе с массивами. Панель инструментов окна Workspace позволяет удалить лишние переменные, сохранить и открыть рабочую среду.

3. ОПЕРАЦИИ С МАТРИЦАМИ

3.1. Вектор-столбцы и вектор-строки

Описанные выше простые правила вычислений распространяются и на гораздо более сложные вычисления, которые (при использовании обычных языков программирования) требуют составления специальных программ. MATLAB — система, специально предназначенная для проведения сложных вычислений с векторами, матрицами и массивами.

При этом она по умолчанию предполагает, что каждая заданная переменная — это вектор, матрица или массив. Все определяется конкретным значением переменной. Например, если задано X=1, то это значит, что X —это вектор с единственным элементом, имеющим значение 1.

Если надо задать вектор из трех элементов, то их значения следует перечислить в квадратных скобках, разделяя пробелами.

Так, например, присваивание:

```
» V=[l 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.

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

```
» V= [2+2/(3+4) exp(5) sqrt(l0)]
» V
V =
2.2857     148.4132     3.1623
```

Возможно задание векторов с комплексными элементами, например:

```
» i=sqrt(-l):
```

```
» CM =[1 2] + i*[5 6]
```

или

```
» CM = [1+5*i 2+6*i]
```

Это создает вектор:

```
CM=
1.0000 + 5.0000i
3.0000 + 7.0000i
```

3.1.1. Сцепление вектор-столбцов

Описанный ниже способ позволяет выполнить операцию объединения векторов в матрицу. Например, создадим вначале вектор 3x1:

```
» A=(8; 3; 4)
A=
8
3
4
```

Теперь можно построить матрицу, содержащую четыре вектора A:

```
» B = [A A+16 A+32 A+16]
B =
8   24   40   24
3   19   35   19
4   20   36   20
```

Полученная матрица имеет размер 3x4.

3.1.2. Сцепление вектор-строк

Описанный ниже способ позволяет выполнить операцию объединения векторов в матрицу. Например, создадим вначале строку 1x3:

```
» A= (8 3 4)
```

```
A=
8     3     4
```

Теперь можно построить матрицу, содержащую четыре вектора A:

```
» B = [A; A+16; A+32; A+16;]
B =
8   3   4
24  19  20
40  35  24
24  19  20
```

Полученная матрица имеет размер 3x4.

3.2. Элементы вектора

3.2.1. Обращение к элементам вектора

Для указания отдельного элемента вектора или матрицы используются выражения вида A(i) или M(i. j). Например, если задать

```
» A =  (24 19 20)
» A(2)
ans = 19
```

то результат будет равен 19.

Если нужно присвоить элементу A(i) новое значение x, следует использовать выражение `A(i)=x`

Например, если элементу A(1) надо присвоить значение 10, следует записать

```
» A(1)=10
```

3.2.2. Индексация при помощи вектора

Обычное индексирование в MATLAB заключается в обращении к элементам вектора по индексу или по массиву индексов. Например, если есть вектор

```
>> y=[1.1 2.2 3.3 4.4 5.5 6.6 7.7 8.8 9.9];
```

то для обращения к его пятому элементу следует писать

```
>> a=y(5)
a =
5.5000
```

Если в вектор b требуется занести его элементы с четвертого по седьмой, то применятся индексирование при помощи сечения:

```
>> b=y(5:7)
b =
5.5000    6.6000    7.7000
```

Для доступа к элементам вектора y с четными индексами следует произвести индексацию при помощи вектора индексов:

```
>> ind=2:2:length(y)
ind =
2     4     6     8
>> d=y(ind)
d =
2.2000    4.4000    6.6000    8.8000
```

Все эти способы относятся к обычной индексации.

Логическая индексация заключается в задании некоторого условия выбора элементов из массива.

Например, если требуется сформировать логический вектор той же длины, что и вектор y, в котором логическая единица соответствует элементам вектора y меньшим пяти, а логический ноль соответствует элементам вектора y большим или равным пяти, то следует записать логическое выражение для массива y с использованием операции сравнения (знака меньше <):

```
>> index = y<5
index =
1    1    1    1    0    0    0    0    0
```

3.2.3. Индексация при помощи знака двоеточия

: (двоеточие) — формирование подвекторов из векторов. Оператор : — один из наиболее часто используемых операторов в системе MATLAB.

Оператор : использует следующие правила для создания векторов:

- j:k — то же, что и [j.j+l,...,k];
- j:k — пустой вектор, если j>k;
- j:i :k — то же, что и [j, j+i ,j+2i .....k];
- j:i :k — пустой вектор, если i>0 и j>k или если i<0 и j<k, где 1, j и k — скалярные величины.

Ниже показано, как выбирать с помощью оператора : элементы из векторов:

- A(j) — это j-й элемент из A;
- A(:) — эквивалент одномерного массива. Для векторов это аналогично A;
- A(j:k) - это A(j), A(j+l),...,A(k);

3.3. Применение функций обработки данных к векторам.

3.3.1 Нахождение минимума и максимума из элементов вектора

Самый простой анализ данных, содержащихся в некотором массиве, заключается в поиске его элементов с максимальным и минимальным значениями. В системе MATLAB определены следующие быстрые функции для нахождения минимальных и максимальных элементов массива:

- max(A) — возвращает наибольший элемент
- [C, I] =max(A) — кроме максимального значения возвращает индекс I этого элемента.

Примеры:

```
» A=(30; 38; 46; 5; 13; 21; 22)
30
38
46
5
```

```
13
21
22
» C = max(A)
C=
46
»[C,I]=max(A)
C=
46
I=
3
```

Для быстрого нахождения элемента массива с минимальным значением служит следующая функция:

- min(A) — возвращает наименьший элемент

[C, I] =min(A) — кроме минимального значения возвращает индекс I этого элемента.

3.3.2. Сортировка элементов вектора по возрастанию, убыванию и упорядочение элементов в порядке возрастания их модулей

Многие операции статистической обработки данных выполняются быстрее и надежнее, если данные предварительно отсортированы. Кроме того, нередко представление данных в отсортированном виде более наглядно и ценно. Ряд функций служит для выполнения сортировки элементов массива. Они представлены ниже.

- sort (A) — в случае одномерного массива A сортирует и возвращает элементы по возрастанию их значений. Допустимы вещественные, комплексные и строковые элементы. Если A принимает комплексные значения, то элементы сначала сортируются по абсолютному значению, а затем, если абсолютные значения равны, по аргументу. Если A включает NaN-элементы, sort помещает их в конец;
- [B, INDEX] = sort(A) — наряду с отсортированным массивом возвращает массив индексов INDEX. Он имеет размер size(A), с помощью этого массива можно восстановить структуру исходного массива.

Примеры:

```
» A=(17 23 4 10 11)
A =
17
23
4
10
11
» [B, INDEX]= sort(A)
B=
4
10
11
17
23
index=
3
4
5
1
2
```

3.4. Поэлементные операции с векторами

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

Для этого перед знаком операции ставится точка. Например, оператор * означает умножение для векторов, а оператор .* —поэлементное умножение всех элементов массива. Так, если A — вектор, то A.*2 даст вектор, все элементы которой умножены на скаляр — число 2.

Выражения умножения вектора на скаляр — A*2 и A.*2 не являются эквивалентными.

3.5. Работа с массивами.

3.5.1 Построение таблицы значений функции

Отображение функции в виде таблицы удобно, если имеется сравнительно небольшое число значений функции.

Пусть требуется вывести в командное окно таблицу значений функции $y(x) = \frac{\sin^2 x}{1+\cos x} + e^{-x}\ln x$ в точках 0.2, 0.3, 0.5, 0.8, 1.3, 1.7, 2.5. Задача решается в два этапа.

1. Создается вектор-строка x, содержащая координаты заданных точек.
2. Вычисляются значения функции y(x)от каждого элемента вектора x и записываются полученные значения в вектор-строку y.

Значения функции необходимо найти для каждого из элементов вектор-строки x, поэтому операции в выражении для функции должны выполняться поэлементно.

```
» x = [0.2 0.3 0.5 0.8 1.3 1.7 2.5]
x =
0.2000   0.3000   0.5000   0.8000   1.3000   1.7000
2.5000
» y = sin(x).^2./(1+cos(x))+exp(-x).*log(x)
y =
-1.2978  -0.8473  -0.2980   0.2030   0.8040   1.2258
1.8764
```

Часто требуется вывести значение функции в точках отрезка, отстоящих друг от друга на равное расстояние (шаг). Предположим, что необходимо вывести таблицу значений функции y(x) на отрезке [1, 2] с шагом 0.2.

Можно, конечно, ввести вектор-строку значений аргумента x=[1, 1.2, 1.4, 1.6, 1.8, 2.0] из командной строки и вычислить все значения функции так, как описано выше.

Однако если шаг будет не 0.2, а например 0.01, то предстоит большая работа по вводу вектора x. В MatLab предусмотрено простое создание векторов, каждый элемент которых отличается от предшествующего на постоянную величину, т.е. на шаг.

Следующие два оператора приводят к формированию одинаковых вектор-строк.

```
» x = [1, 1.2, 1.4, 1.6, 1.8, 2.0]
```

```
x =
1.0000   1.2000   1.4000   1.6000   1.8000   2.0000
» x = [1:0.2:2]
x =
1.0000   1.2000   1.4000   1.6000   1.8000   2.0000
```

Таким образом вектор x можно записать как:
x = [начальное значение : шаг : конечное значение].

Обратите внимание, что элементы вектора, заполняемого при помощи двоеточия, могут быть только вещественными, поэтому для транспонирования можно использовать апостроф вместо точки с апострофом.

Шаг, равный единице, допускается не указывать при автоматическом заполнении:

```
» x = [1:5] x = 1  2  3  4  5
```

3.5.2. Построение графиков функции одной переменной

Функции одной переменной *y(x)* находят широкое применение в практике математических и других расчетов, а также в технике компьютерного математического моделирования. Для отображения таких функций используются графики в декартовой (прямоугольной) системе координат.

При этом обычно строятся две оси — горизонтальная *X* и вертикальная *Y,* и задаются координаты *x* и *y,* определяющие узловые точки функции *y(x)*. Эти точки соединяются друг с другом отрезками прямых, т. е. при построении графика осуществляется линейная интерполяция для промежуточных точек.

Поскольку MATLAB — матричная система, совокупность точек *y(x)* задается векторами X и Y одинакового размера.

Команда plot служит для построения графиков функций в декартовой системе координат.

- plot (X, Y) — строит график функции *y(x),* координаты точек *(x, y)* которой берутся из векторов одинакового размера Y и X. Если X или Y — матрица, то строится семейство графиков по данным, содержащимся в колонках матрицы.

Приведенный ниже пример иллюстрирует построение графиков двух функций — *sin(x)* и *cos(x)*, значения функции которых содержатся в матрице Y, а значения аргумента *x* хранятся в векторе X:

```
» x=[0 1 2 3 4 5];
» Y=[sin(x):cos(x)];
» plot(x,Y)
```

Ниже показан график функций из этого примера. В данном случае отчетливо видно, что график состоит из отрезков, и если вам нужно, чтобы отображаемая функция имела вид гладкой кривой, необходимо увеличить количество узловых точек. Расположение их может быть произвольным.

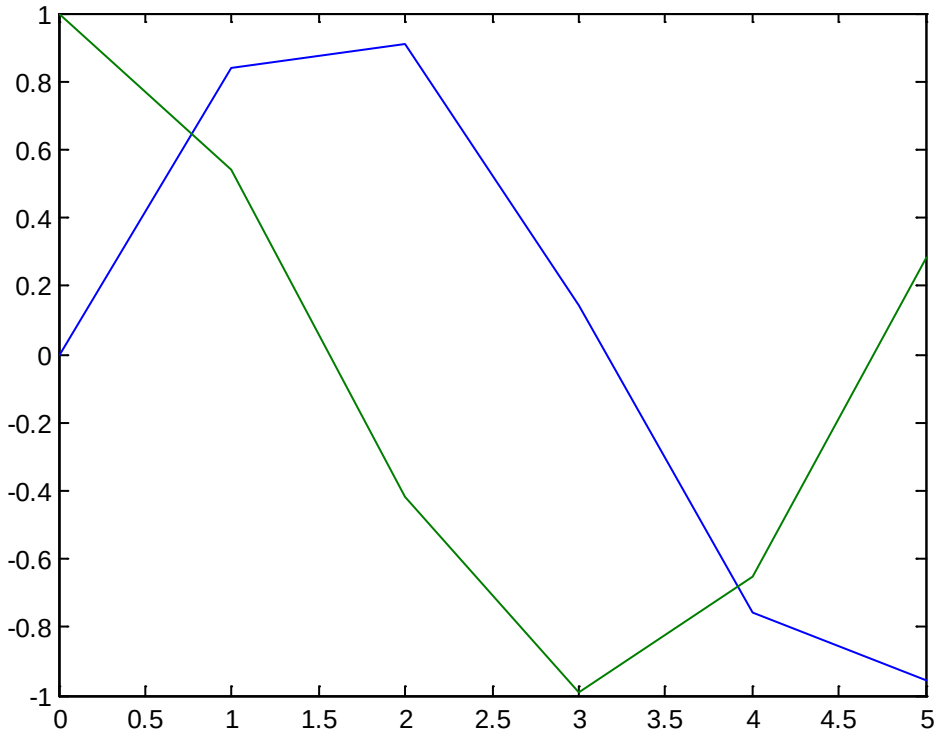


Рис. 3.1. График двух функции одной переменной

Уменьшим шаг для того, что бы получить более плавный график.

```
>> X=[0:0.05:5];
>> Y=[sin(X); cos(X)];
>> plot(X,Y)
```

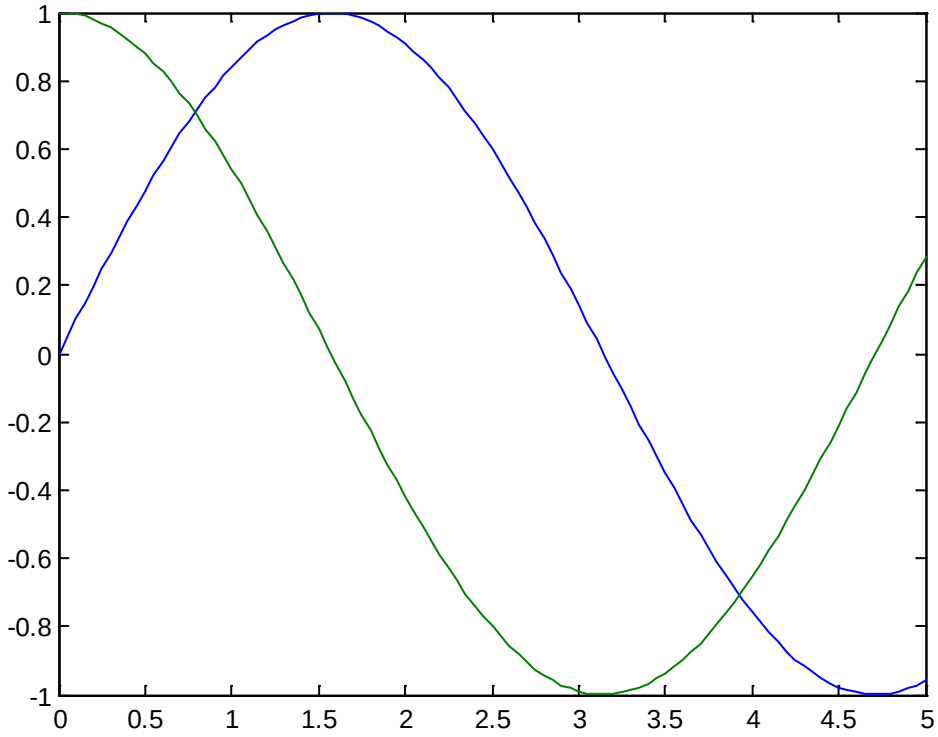


Рис. 3.2. Плавный график двух функции одной переменной

3.6. Операции произведения

3.6.1. Скалярное произведение

Несколько простых функций служат для перемножения элементов массивов:

- prod(A) — возвращает произведение элементов массива

Пример:

```
>> A = [1 2 3 4 5]
A =
1     2     3     4     5
>> B = prod(A)
B =
120
```

3.6.2. Векторное произведение

Аналогично существует несколько простых функций для векторного перемножения матриц.

- cross(U, V) — возвращает векторное произведение векторов U и V в трехмерном пространстве, т. е. W=UxV. U и V — обязательно векторы с тремя элементами;
- cross(U, V, dim) — возвращает векторное произведение U и V по размерности, определенной скаляром dim. U и V — многомерные массивы, которые должны иметь одну и ту же размерность, причем размер векторов в каждой размерности size(U, dim) и size(V, dim) должен быть равен 3.

Пример:

```
» a = [6 5 3]; b= [1 7 6];
» c = cross(a, b)
c =
9 -33 37
```

3.6.3. Матричное произведение

Матричное произведение реализуется оператором * и выполняется при выполнении условия перемножения матриц.

```
>> a = [1 3 2 8];
>> b = [4; 2; 3; 1];
>> c = b*a
c =
4    12     8    32
2     6     4    16
3     9     6    24
1     3     2     8
```

3.7. Двумерные массивы и матрицы

Задание матрицы требует указания нескольких строк. Для разграничения строк используется знак ; (точка с запятой). Этот же знак в конце ввода предотвращает вывод матрицы или вектора (и вообще результата любой операции) на экран дисплея.

Так, ввод

```
» M=[1 2 3; 4 5 6; 7 8 9];
```

задает квадратную матрицу, которую можно вывести:

```
» M
M =
1     2     3
4     5     6
7     8     9
```

3.8. Обращение к элементам матриц

Для указания отдельного элемента вектора или матрицы используются выражения вида V(i) или M(i, j). Например, если задать

```
» M =  (24 19; 20 11);
» M(2, 2)
ans = 11
```

то результат будет равен 11.

Если нужно присвоить элементу M(i, j) новое значение x, следует использовать выражение: `M(i,j)=x`

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

```
» M(2, 2)=10
```

3.9. Операции над матрицами

3.9.1. Сложение и вычитание матриц

При использовании матричных операций следует помнить, что для сложения или вычитания матрицы должны быть одного размера, а при перемножении число столбцов первой матрицы обязано равняться числу строк второй матрицы. Сложение и вычитание матриц, так же как чисел и векторов, осуществляется при помощи знаков плюс и минус.

```
>> a = [1 10 4; 1 4 7; 1 8 5]
a =
1    10     4
1     4     7
1     8     5
>> b = [3 4 2; 1 8 2; 3 2 5]
b =
3     4     2
1     8     2
3     2     5
>> c = a + b
c =
4    14     6
2    12     9
4    10    10
```

3.9.2 Умножение матриц

Пусть дана матрица $A_{m1,n1}$ и матрица $B_{m2,n2}$. Тогда для того, что бы умножить матрицу A на матрицу B необходимо, что бы m2 = n1. Матрица C, полученная путем перемножения матриц A и B будет иметь размер $C_{m1,n2}$.

```
>> a = [1 10 4 3; 1 4 7 3; 1 8 5 1]
a =
1    10     4     3
1     4     7     3
1     8     5     1
>> b = [3 4; 1 8; 3 2; 3 1]
b =
3     4
1     8
3     2
3     1
>> c = a*b
c =
34    95
37    53
29    79
```

3.9.3. Умножение матрицы на число

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

Для этого перед знаком операции ставится точка. Например, оператор * означает умножение для векторов или матриц, а оператор .* — поэлементное умножение всех элементов массива. Так, если M — матрица, то M.*2 даст матрицу, все элементы которой умножены на скаляр — число 2.

Впрочем, для умножения матрицы, на скаляр оба выражения — M*2 и M.*2 — оказываются эквивалентными.

```
>> a = [1 10 4 3; 1 4 7 3; 1 8 5 1]
a =
1    10     4     3
1     4     7     3
1     8     5     1
>> a.*2
ans =
2    20     8     6
2     8    14     6
2    16    10     2
```

3.9.4. Транспонирование вещественных матриц

Транспонирование матрицы, так же как и вектора, производится при помощи .', а символ ' означает комплексное сопряжение. Для вещественных матриц эти операции приводят к одинаковым результатам:

```
>> a = [1 10 4 3; 1 4 7 3; 1 8 5 1]
a =
1    10     4     3
1     4     7     3
1     8     5     1

>> a'
ans =
1     1     1
```

```
10     4     8
4      7     5
3      3     1
```

3.9.5. Транспонирование матриц, содержащих комплексные числа

Транспонирование матриц, содержащих комплексные числа, а эта операция и является комплексным сопряжением, выполняется командой ' (апостроф).

```
>> a = [1+2i 3-2i; 5+i 4-4i]
a =
1.0000 + 2.0000i   3.0000 - 2.0000i
5.0000 + 1.0000i   4.0000 - 4.0000i
>> a'
ans =
1.0000 - 2.0000i   5.0000 - 1.0000i
3.0000 + 2.0000i   4.0000 + 4.0000i
```

3.9.6. Возведение матрицы в степень

Операция возведения матрицы в степень осуществляется коман-). Только квадратные матрицы могут быть возведены в степень.

```
>> a = [1 10 4; 1 4 7; 1 8 5]
a =
1    10     4
1     4     7
1     8     5
>> a^2
ans =
15    82    94
12    82    67
14    82    85
```

3.9.7. Перемножение матрицы и вектора

Пусть дан вектор $A_{m1,1}$ и матрица $B_{m2,n2}$. Тогда для того, что бы умножить вектор A на матрицу B необходимо, что бы n2 = m1. Вектор C, полученный путем перемножения матрицы B и вектора A будет

иметь размер $C_{m2,1}$.

```
>> a = [1 10 4; 1 4 7; 1 8 5]
a =
1    10     4
1     4     7
1     8     5
>> b = [1; 3; 3]
b =
1
3
3
>> c = a*b
c =
43
34
40
```

3.10. Решение систем линейных уравнений

Простейшим способом решения систем является применение знака обратной косой черты. Предположим, что требуется решить систему

$$\begin{cases} 4x_1 + x_2 + 2x_3 = 7 \\ 3x_1 + 7x_2 + x_3 = 11 \\ 2x_1 + 2x_2 + 8x_3 = 12 \end{cases}$$

Для этого заполняем матрицу и вектор-столбец правой части (правая часть должна быть именно столбцом, иначе выведется ошибка о несовпадении размерностей)

```
A = [4 1 2;
     3 7 1;
     2 2 8];
f = [7; 11; 12];
```

и используем знак обратной косой черты

```
x = A\f
x =
1
1
1
```

Вместо знака обратной косой черты можно было вызвать функцию mldivide

```
x = mldivide(A, f)
```

Результат будет тем же самым

Вместо решения системы (или систем с несколькими правыми частями, заданными в матрице) линейных алгебраических уравнений при помощи знака обратной косой черты можно использовать функцию linsolve, которая не делает всех проверок матрицы, заложенных в алгоритме операции.

Функция linsolve, вызванная в самом простом варианте с двумя входными аргументами и одним выходным аргументом x = linsolve(A, b) решает систему Ax = b одним из способов, в зависимости от того, квадратная матрица, или нет.

3.11. Считывание и запись данных

Для считывания данных из файла в MATLAB существует несколько способов, которые можно разделить на две группы: с использованием функций MATLAB, или приложений с графическим интерфейсом. Предположим, что данные записаны в текстовом файле в два столбика и в качестве разделителя десятичных разрядов используется точка, т.е. файл с данными data.txt имеет следующий формат:

```
1.25 0.8973
1.44 1.2398
1.54 2.0019
```

Для записи столбцов файла в вектора MATLAB можно воспользоваться Мастером импорта (Import Wizard), для чего следует в меню File основного окна MATLAB выбрать пункт Import Data и в появившемся диалоговом окне открытия файла указать нужный файл.

После нажатия на кнопку Finish в рабочей среде окажется двумерный массив (с двумя столбцами) с выбранным именем.

Для последующей работы следует создать два вектора x и y, содержащие, соответственно, первый и второй столбец массива data. Для этого следует ввести в командной строке MATLAB две команды, в которых производится индексация: для обращения ко всем элементам первого и второго столбцов массива data использовано двоеточие (индексация двоеточием), а номер выделяемого в вектор столбца задается числом:

```
>> x=data(:,1);
>> y=data(:,2);
```

Массивы данных x и y созданы в рабочей среде MATLAB (см., например, содержимое окна Workspace).

Если данные находятся в книге Excel, содержащей один лист, то Мастер импорта также способен создать массив, в который будут записаны эти данные. Если между столбцами данных на листе Excel есть пустые столбцы, то в MATLAB получится массив, содержащий NaN (не числа - Not a Number) в ячейках этих столбцов.

Для чтения числовых данных из текстовых файлов, имеющих табличную структуру, подходит функция load. Символы-разделители (пробел, точка, точка с запятой, табуляция) интерпретируются как разделители элементов строки. Например, если текстовый файл data.txt содержит

```
1.25; 0.8973
1.44; 1.2398
1.54; 2.0019
```

то результатом выполнения команды

```
>> A=load('data.txt')
```

будет двумерный массив

```
A =
1.2500    0.8973
1.4400    1.2398
1.5400    2.0019
```

Запись содержимого его столбцов в вектора x и y мы уже рассматривали выше, она производится при помощи простых команд

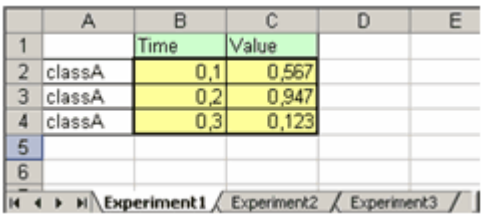
```
>> x=data(:,1);
>> y=data(:,2);
```

Для считывания данных из Excel в MATLAB есть специальная функция xlsread, которая допускает несколько способов вызова. Рассмотрим основные из них.

Указание имени книги Excel в апострофах в качестве входного аргумента функции xlsread приводит к считыванию данных с первого листа книги. Если первый лист не содержит данных, то функция xlsread вернет пустой массив. При считывании данных с листа книги Excel игнорируются верхние строки и правые столбцы, содержащие текст. Числовые данные, полученные в ячейках листа в результате вычисления

по формулам, считываются функцией xlsread как обычные числовые константы, введенные в ячейки.

Если, например, книга ExpData.xls содержит три листа, которые называются Experiment1, Experiment2, Experiment3 и первый лист такой, как на рисунке



	A	B	C	D	E
1		Time	Value		
2	classA	0,1	0,567		
3	classA	0,2	0,947		
4	classA	0,3	0,123		
5					
6					

Experiment1 | Experiment2 | Experiment3

Рис. 3.3. Книга ExpData.xls

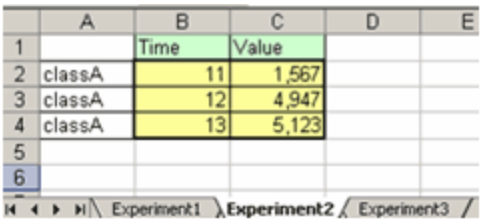
то команда

```
>> A=xlsread('ExpData.xls')
```

приведет к появлению в рабочей среде следующего массива

```
A =
0.1000    0.5670
0.2000    0.9470
0.3000    0.1230
```

Для считывания данных из книги Excel не только с первого, а с произвольного листа, достаточно указать номер листа или его имя (в апострофах) в качестве второго входного аргумента. Если в книге ExpData.xls на втором листе с именем Experiment2 находятся такие данные



	A	B	C	D	E
1		Time	Value		
2	classA	11	1,567		
3	classA	12	4,947		
4	classA	13	5,123		
5					
6					

Experiment1 | Experiment2 | Experiment3

Рис. 3.4. Книга ExpData.xls

то следующие команды приводят к одинаковым результатам

```
>> A=xlsread('ExpData.xls',2)
A =
11.0000    1.5670   12.5670
12.0000    4.9470   16.9470
```

```
13.0000    5.1230   18.1230
>> A=xlsread('ExpData.xls','Experiment2')
A =
11.0000    1.5670   12.5670
12.0000    4.9470   16.9470
13.0000    5.1230   18.1230
```

Функция xlsread позволяет указать диапазон данных, которые требуется считать. Если диапазон данных указан в качестве ее второго входного аргумента, то происходит считывание данных из этого диапазона на первом листе книги Ecxel, а если во втором входном аргументе xlsread указать имя листа, а диапазон данных - в третьем, то будет производиться считывание данных из заданного диапазона этого листа. В предыдущем примере данные на листе Experiment2 расположены в диапазоне B2:C4, поэтому для их считывания следует применить:

```
>> A=xlsread('ExpData.xls','Experiment2','B2:C4')
A =
11.0000    1.5670
12.0000    4.9470
13.0000    5.1230
```

3.12. Блочные матрицы

Очень часто в приложениях возникают так называемые блочные матрицы, т.е. матрицы, составленные из непересекающихся подматриц (блоков).

Система MATLAB позволяет выполнить операцию *конкатенации* — объединения малых матриц в большую. Например, создадим вначале 3x3. Функция magic(n) задает магическую матрицу размера *nxn, у* которой сумма всех столбцов, всех строк и даже диагоналей равна одному и тому же числу:

```
» A=magic(3)
A=
1
5
9
```

Теперь можно построить матрицу, содержащую четыре матрицы, т.е. блочную:

```
» B=[A A+16;A+32 A+16]
B =
8   1   6   24  17  22
3   5   7   19  21  23
4   9   2   20  25  18
40  33  38  24  17  22
35  37  39  19  21  23
36  41  34  20  25  18
```

Полученная матрица имеет уже размер 6x6.

3.12.1. Выделение блоков

Обратной задачей к конструированию блочных матриц является задача выделения блоков. Выделение блоков матриц осуществляется индексацией при помощи двоеточия.

Введите матрицу и затем выделите подматрицу с элементами $P=\begin{pmatrix}1 & 2 & 0 & 2\\ 4 & 10 & 12 & 5\\ 0 & 11 & 10 & 5\\ 9 & 2 & 3 & 5\end{pmatrix}$, задав номера строк и столбцов при помощи двоеточия:

```
»P1 = P(2:3,2:3)
P1 =
10   12
11   10
```

Для выделения из матрицы столбца или строки (то есть массива, у которого один из размеров равен единице) следует в качестве одного из индексов использовать номер столбца или строки матрицы, а другой индекс заменить двоеточием без указания пределов.

Например, запишите вторую строку матрицы P в вектор p

```
»p = P(2, :)
p =
4  10  12  5
```

При выделении блока до конца матрицы можно не указывать ее размеры, а использовать элемент end:

```
»p = P(2,  2:end)
p =
10  12  5
```

3.12.2. Выделение из матрицы строки

: (двоеточие) — формирование подвекторов и подматриц из векторов и матриц. Оператор : — один из наиболее часто используемых операторов в системе MATLAB.

Оператор : использует следующие правила для создания векторов:

- j:k — то же, что и [j.j+l,...,k];
- j:k — пустой вектор, если j>k;
- j:i :k — то же, что и [j, j+i ,j+2i .....k];
- j:i :k — пустой вектор, если i>0 и j>k или если i<0 и j<k, где 1, j и k — скалярные величины.

Ниже показано, как выбирать с помощью оператора : строки, столбцы и элементы из векторов, матриц и многомерных массивов:

- A(:, j) — это j-й столбец из A;
- A(i,:) — это i-я строка из A;
- A(:,:) — эквивалент двумерного массива. Для матриц это аналогично A;
- A(j:k) - это A(j), A(j+l),...,A(k);
- A(:J:k)-это A(:.j),A(:,j+l),...,A(:,k);
- A(:,: ,k) — это *k-я* страница трехмерного массива A;
- A(i.j.k.:) — вектор, выделенный из четырехмерного массива A. Вектор включает элементы A(1, j.k.l), A(i. j,k,2), A(i. j,k.3) и т. д.;
- A(:) — записывает все элементы массива A в виде столбца.

3.12.3. Удаление строк

Для формирования матриц и выполнения ряда матричных операций возникает необходимость удаления отдельных столбцов и строк матрицы. Для этого используются пустые квадратные скобки [ ]. Проделаем это с матрицей M:

```
» M=[1 2 3; 4 5 6; 7 8 9]
M =
2
5
8
```

Удалим второй столбец используя оператор : (двоеточие):

```
» M(:,2)=[ ]
1     3
4     6
7     9
```

А теперь, используя оператор : (двоеточие), удалим вторую строку:

```
» M(2,:)=[ ]
M =
1    3
7    9
```

3.12.4. Удаление столбцов

См. п.3.12.3.

3.13. Генерация матриц

3.13.1. Заполнение матриц при помощи индексации

Выше было описано несколько способов ввода матриц в MatLab. Однако часто бывает проще сгенерировать матрицу, чем вводить ее, особенно если она обладает простой структурой.

Рассмотрим пример такой матрицы: $T = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 \end{pmatrix}$. Генерация матрицы T осуществляется в три этапа:

1. Создание массива T размера пять на пять, состоящего из нулей.

2. Заполнение первой строки единицами.

3. Заполнение части последней строки минус единицами до последнего элемента. Соответствующие команды MatLab приведены ниже.

```
» A(1:5, 1:5) = 0
A=
0   0   0   0   0
0   0   0   0   0
0   0   0   0   0
0   0   0   0   0
0   0   0   0   0
» A(1, :) = 1
A=
1   1   1   1   1
0   0   0   0   0
0   0   0   0   0
0   0   0   0   0
0   0   0   0   0
» A(end, 3:end) = -1
A=
1   1   1   1   1
0   0   0   0   0
0   0   0   0   0
0   0   0   0   0
0   0  -1  -1  -1
```

Создание некоторых специальных матриц в MatLab осуществляется при помощи встроенных функций.

3.13.2. Создание матриц специального вида

Заполнение прямоугольной матрицы нулями производится встроенной функцией zeros, аргументами которой являются число строк и столбцов матрицы:

```
» A = zeros(3, 6)
A =
0  0  0  0  0  0
0  0  0  0  0  0
0  0  0  0  0  0
```

Один аргумент функции zeros приводит к образованию квадратной матрицы заданного размера:

```
» A = zeros(3)
A =
```

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

Единичная матрица инициализируется при помощи функции eye:

```
» I = eye(4)
I=
1  0  0  0
0  1  0  0
0  0  1  0
0  0  0  1
```

Функция eye с двумя аргументами создает прямоугольную матрицу, у которой на главной диагонали стоят единицы, а остальные элементы равны нулю:

```
» I = eye(4, 8)
I =
1  0  0  0  0  0  0  0
0  1  0  0  0  0  0  0
0  0  1  0  0  0  0  0
0  0  0  1  0  0  0  0
```

Матрица, состоящая из единиц, образуется в результате вызова функции ones:

```
» E = ones(3, 7)
E =
1  1  1  1  1  1  1
1  1  1  1  1  1  1
1  1  1  1  1  1  1
```

Использование одного аргумента в ones приводит к созданию квадратной матрицы, состоящей из единиц.

MatLab предоставляет возможность заполнения матриц случайными элементами. Результатом функции rand является матрица чисел, распределенных случайным образом между нулем и единицей, а функции randn — матрица чисел, распределенных по нормальному закону:

```
» R = rand(3,   5)
```

```
R =
0.9501  0.4860  0.4565  0.4447  0.9218
0.2311  0.8913  0.0185  0.6154  0.7382
0.6068  0.7621  0.8214  0.7919  0.1763
```

Один аргумент функций rand и randn приводит к формированию квадратных матриц.

Часто возникает необходимость создания диагональных матриц, т.е. матриц, у которых все недиагональные элементы равны нулю. Функция diag формирует диагональную матрицу из вектор-столбца или вектор-строки, располагая их элементы по диагонали матрицы:

```
» d = [1; 2; 3; 4];
» D = diag(d)
D =
1  0  0  0
0  2  0  0
0  0  3  0
0  0  0  4
```

Функция diag служит и для выделения диагонали матрицы в вектор, например

```
» A = [10 1 2; 1 20 3; 2 3 30];
» d = diag(A) d = 10 20 30
```

3.14. Поэлементные операции с матрицами

: (двоеточие) — формирование подвекторов и подматриц из векторов и матриц. Оператор : — один из наиболее часто используемых операторов в системе MATLAB.

Оператор : использует следующие правила для создания векторов:

- j:k — то же, что и [j.j+l,...,k];
- j:k — пустой вектор, если j>k;
- j:i :k — то же, что и [j, j+i ,j+2i .....k];
- j:i :k — пустой вектор, если i>0 и j>k или если i<0 и j<k, где 1, j и k — скалярные величины.

Ниже показано, как выбирать с помощью оператора : строки, столбцы и элементы из векторов, матриц и многомерных массивов:

- A(:, j) — это j-й столбец из A;
- A(i,:) — это i-я строка из A;
- A(:,:) — эквивалент двумерного массива. Для матриц это аналогично A;
- A(j:k) - это A(j), A(j+l),...,A(k);
- A(:J:k)-это A(:.j),A(:,j+l),...,A(:,k);
- A(:,: ,k) — это *k-я* страница трехмерного массива A;
- A(i.j.k.:) — вектор, выделенный из четырехмерного массива A. Вектор включает элементы A(1, j.k.l), A(i. j,k,2), A(i. j,k.3) и т. д.;
- A(:) — записывает все элементы массива A в виде столбца.

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

Для этого перед знаком операции ставится точка. Например, оператор * означает умножение для векторов или матриц, а оператор .* — поэлементное умножение всех элементов массива. Так, если M — матрица, то M.*2 даст матрицу, все элементы которой умножены на скаляр — число 2.

Впрочем, для умножения матрицы, на скаляр оба выражения — M*2 и M.*2 — оказываются эквивалентными.

3.15. Применение функций к матрицам.

3.15.1. Вычисление математических функций от элементов матриц

Поскольку система MATLAB представляет переменные в виде матриц, то все функции ориентированы на вычисление от элементов матриц.

```
>> a = [1 10 4; 1 4 7; 1 8 5]
a =
1    10     4
1     4     7
1     8     5
>> log(a)
ans =
0    2.3026    1.3863
```

```
0    1.3863    1.9459
0    2.0794    1.6094
```

3.15.2. Применение функций обработки данных к матрицам

Функция вычисления произведения

- prod(A) — возвращает вектор-строку, содержащую произведения элементов каждого столбца;
- prod (A, dim) — возвращает матрицу (массив размерности два) с произведением элементов массива A по столбцам (dim=l), по строкам(dim=2), по иным размерностям в зависимости от значения скаляра dim.

Определены следующие функции суммирования элементов массивов:

- sum(A) — возвращает вектор-строку, содержащую сумму элементов каждого столбца;
- sum(A, dim) — возвращает сумму элементов массива по столбцам (dim=1), строкам (dim=2) или иным размерностям в зависимости от значения скаляра dim.

```
>> a = [1 10 4; 1 4 7; 1 8 5]
a =
1    10     4
1     4     7
1     8     5
>> prod(a)
ans =
1   320   140
>> sum(a)
ans =
3    22    16
```

3.15.3. Сортировка элементов матрицы в порядке возрастания и убывания

Многие операции статистической обработки данных выполняются быстрее и надежнее, если данные предварительно отсортированы. Кроме того, нередко представление данных в отсортированном виде более наглядно и ценно. Ряд функций служит для выполнения сортировки

элементов массива. Они представлены ниже.

- sort (A) — в случае одномерного массива A сортирует и возвращает элементы по возрастанию их значений; в случае двумерного массива происходит сортировка и возврат элементов каждого столбца. Допустимы вещественные, комплексные и строковые элементы. Если A принимает комплексные значения, то элементы сначала сортируются по абсолютному значению, а затем, если абсолютные значения равны, по аргументу. Если A включает NaN-элементы, sort помещает их в конец;
- [B, INDEX] = sort(A) — наряду с отсортированным массивом возвращает массив индексов INDEX. Он имеет размер size(A), с помощью этого массива можно восстановить структуру исходного массива.
- sort(A, dim) — для матриц сортирует элементы по столбцам (dim=l) или по рядам в зависимости от значения переменной dim.

Примеры:

```
» A=magic(5)
17    24    1     8     15
23    5     7     4     6
4     6     13    0     2
10    12    19    21    3
11    18    25    2     9
» [B, INDEX] sort(A)
4     5     1     2     3
10    6     7     8     9
11    12    13    14    15
17    18    19    20    16
23    24    25    21    22
```

- sortrows(A) — выполняет сортировку рядов массива A по возрастанию и возвращает отсортированный массив, который может быть или матрицей, или вектором-столбцом;
- sortrows(A, column) — возвращает матрицу, отсортированную по столбцам, точно указанным в векторе column. Например, sortrows(A,[2 3]) сортирует строки матрицы A сначала по второму столбцу, и затем, если его элементы равны, по третьему;

- [B, index] = sort rows (A) — также возвращает вектор индексов index. Если A — вектор-столбец, то B=A(index). Если A — матрица размера *mxn,* то B=A(index, :).

Примеры:

```
» A=[2 3 5 6 8 9; 5 7 1 2 3 5; 1 3 2 1 5 1; 5 0 8 8
4 3]
A =
2     3     5     6     8     9
5     7     1     2     3     5
1     3     2     1     5     1
5     0     8     8     4     3
» b = sortrows(A)
1     3     2     1     5     1
2     3     5     6     3     9
5     0     8     8     4     3
5     7     1     2     3     5
» b = sortrows(A,3)
B =
5     7     1     2     3     5
1     3     2     1     5     1
2     3     5     6     8     9
5     0     8     8     4     3
```

3.15.4. Максимальные или минимальные элементы в соответствующих столбцах матрицы

Самый простой анализ данных, содержащихся в некотором массиве, заключается в поиске его элементов с максимальным и минимальным значениями. В системе MATLAB определены следующие быстрые функции для нахождения минимальных и максимальных элементов массива:

- max(A) —возвращает вектор-строку, содержащую максимальные элементы каждого столбца;
- max(A, B) — возвращает массив того же размера, что A и B, каждый элемент которого есть максимальный из соответствующих элементов этих массивов;
- max(A, [ ], dim) — возвращает наибольшие элементы по столбцам или по строкам матрицы в зависимости от значения скаляра dim.

Например, так(A, [ ], 1) возвращает максимальные элементы каждого столбца матрицы A;

- [C, I] =max(A) — кроме максимальных значений возвращает вектор индексов I этих элементов.

Примеры:

```
» A=magic(7)
30    39    48    1     10    19    28
38    47    7     9     18    27    29
46    6     8     17    26    35    37
5     14    16    25    34    36    45
13    15    24    33    42    44    4
21    23    32    41    43    3     12
22    31    40    49    2     11    20
» C = max(A)
C=
46    47    48    49    43    44    45
» C = max(A,[ ], l)
C =
46    47    48    49    43    44    45
» C = max(A, [ ], 2)
C =
48
47
46
45
44
43
49
»[C,I]=max(A)
C=
49    43    44    45
I=
7     6     5     4
```

Для быстрого нахождения элемента массива с минимальным значением служит следующая функция:

- min(A) — возвращает минимальный элемент, если A — вектор; или возвращает вектор-строку, содержащую минимальные элементы каждого столбца, если A — матрица;

- min(A.B) — возвращает массив того же размера, что A и B, каждый элемент которого есть минимальный из соответствующих элементов этих массивов;
- min(A,[ ],dim) — возвращает наименьший элемент по столбцам или по строкам матрицы в зависимости от значения скаляра dim. Например, max(A,[ ],1) возвращает минимальные элементы каждого столбца матрицы A;
- [C,I] = min(A) — кроме минимальных значений возвращает вектор индексов этих элементов.

Пример:

```
» A=magic(4)
A =
16    2     3     13
5     11    10    8
9     7     6     12
4     14    15    1
» [C,I] = min(A)
C =
4     2     3     1
I =
4     1     1     4
```

Работа указанных функций базируется на сравнении численных значений элементов массива A, что и обеспечивает высокую скорость выполнения операций.

4. ПОСТРОЕНИЕ ДИАГРАММ

4.1. Диаграммы векторных данных

Столбцовые диаграммы широко используются в литературе, посвященной финансам и экономике, а также в математической литературе. Ниже представлены команды для построения таких диаграмм.

- bar(x, Y) — строит столбцовый график элементов вектора Y (или группы столбцов для матрицы Y) со спецификацией положения столбцов, заданной значениями элементов вектора x, которые должны идти в монотонно возрастающем порядке;
- bar(Y) — строит график значений элементов матрицы Y так же, как указано выше, но фактически для построения графика используется вектор x=1:m;
- bar(x,Y,WIDTH) или BAR(Y,WIDTH) — команда аналогична ранее рассмотренным, но со спецификацией ширины столбцов (при WIDTH > 1 столбцы в одной и той же позиции перекрываются). По умолчанию задано WIDTH = 0.8.

Возможно применение этих команд и в следующем виде: bar(....,'Спецификация') для задания спецификации графиков, например типа линий, цвета и т. д., по аналогии с командой plot. Спецификация 'stacked' задает рисование всех n столбцов в позиции m друг на друге.

Пример построения столбцовой диаграммы матрицы размером 12x3 приводится ниже:

```
» % Столбцовая диаграмма с вертикальными столбцами
» bar(rand(12,3),'stacked')
» colormap(cool)
```

Ниже представлена полученная диаграмма

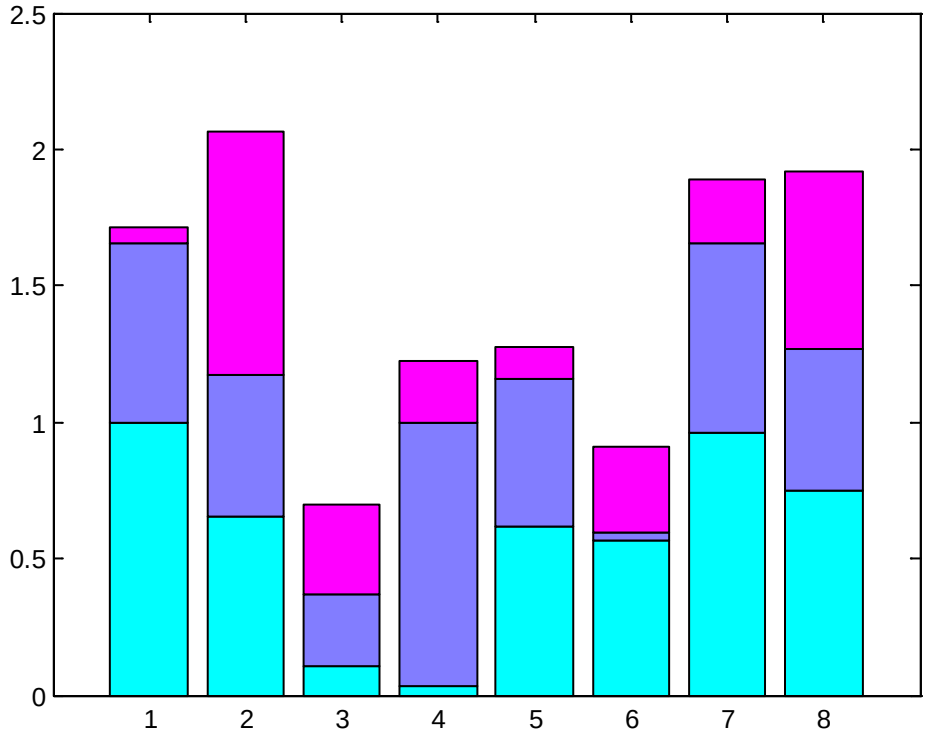


Рис. 4.1. Столбцовая диаграмма с вертикальными столбцами

Помимо команды bar(...) существует аналогичная ей по синтаксису команда barh(...), которая" строит столбцовые диаграммы с горизонтальным расположением столбцов. Пример, приведенный ниже, дает построение столбцовой диаграммы.

```
» barh(rand(5,3),'stacked')
» colormap(cool)
```

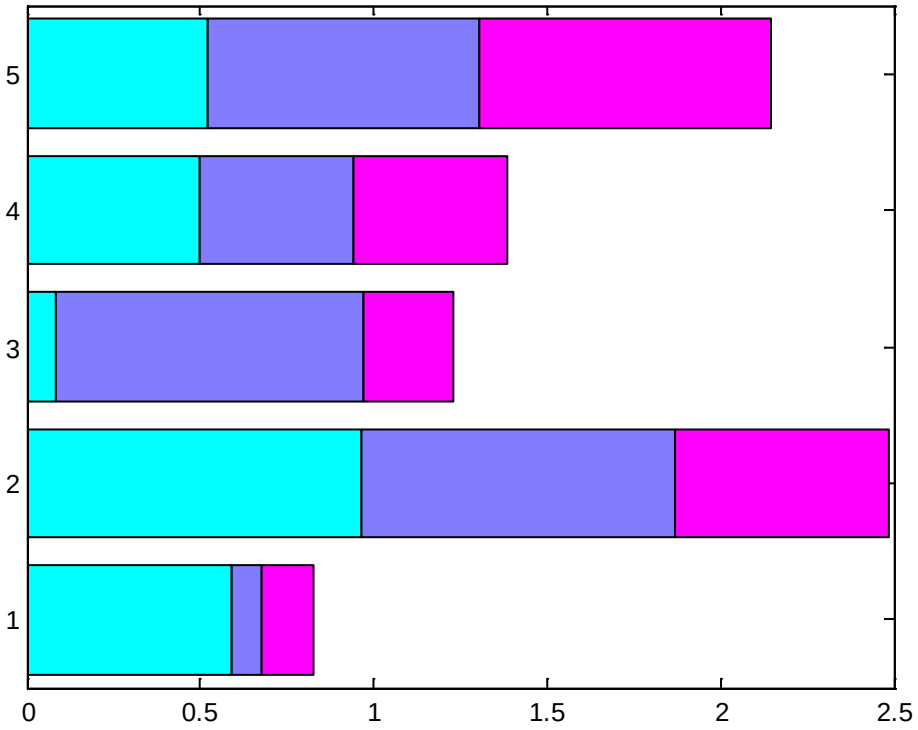


Рис. 4.2. Столбцовая диаграмма с горизонтальными столбцами

Какое именно расположение столбцов выбрать, зависит от пользователя, использующего эти команды для представления своих данных.

4.2. Круговая диаграмма

Круговые диаграммы, состоящие из плоских или объемных секторов(аналогов кусков пирога), строятся с помощью функций pie и pie3. В простейшем случае вектор q, содержащий k положительных компонентов, генерирует k секторов, центральный угол которых пропорционален вкладу каждого компонента в общую сумму. По умолчанию против каждого сектора помещается его процентный вклад.

Построение секторов в плоской диаграмме начинается с "севера" и осуществляется против часовой стрелки. Для объемных диаграмм начало отсчета располагается на "северо-западе".

Для того чтобы обратить внимание на отдельный сектор или группу секторов, их принято немного выдвигать из "пирога".

Создание таких выделяющихся элементов обеспечивается заданием еще од-ного аргумента такой же размерности, что и вектор q. Выдви-

гаемым секторам в новом векторе должны соответствовать ненулевые элементы. Например:

```
>> q = rand(1,3);
>> pie(q)
```

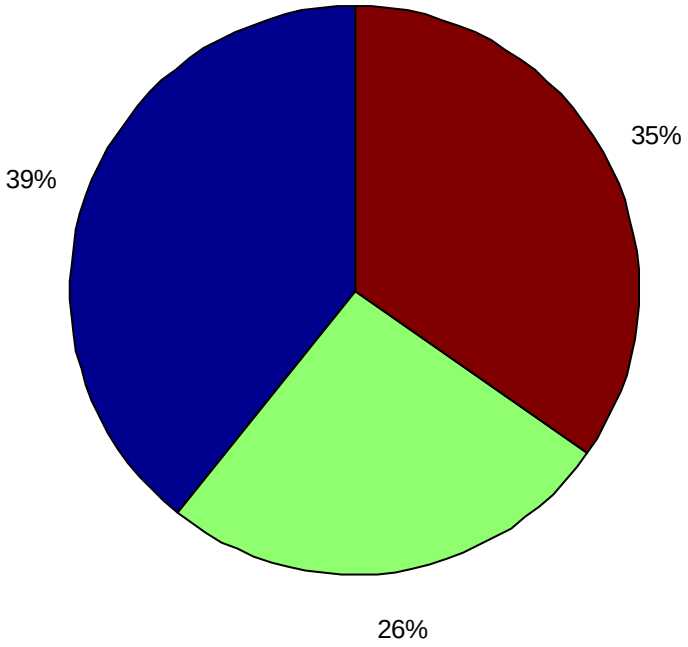


Рис. 4.3. Круговая диаграмма

Вместо процентных меток мы можем разместить свои надписи, предварительно сформированные в массиве ячеек.

4.2.1 Круговая диаграмма с отдельным сектором

Создание таких выделяющихся элементов обеспечивается заданием еще одного аргумента такой же размерности, что и вектор q. Выдвигаемым секторам в новом векторе должны соответствовать ненулевые элементы. Например:

```
>> q = rand(1,3);
>> v=[0,1,0];
>> pie(q, v)
```

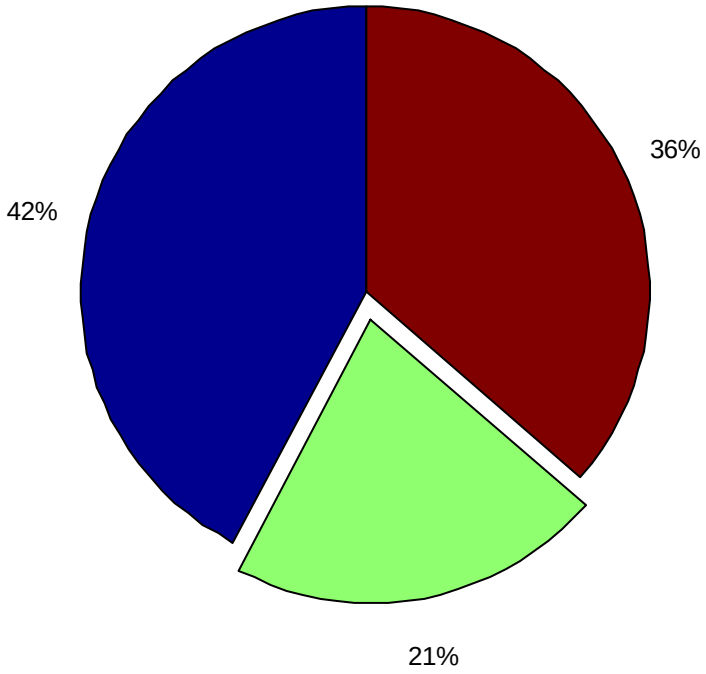


Рис. 4.4. Круговая диаграмма с отдельным сектором

4.2.2. Круговая диаграмма с максимальным сектором

Создание таких выделяющихся элементов обеспечивается заданием еще од-ного аргумента такой же размерности, что и вектор q. Выдвигаемым секторам в новом векторе должны соответствовать ненулевые элементы. Например:

```
»[q_max q_ind]=max(q);
»v=zeros(1,size(q));
»v(q_ind) = 1;
»pie(q,v);
```

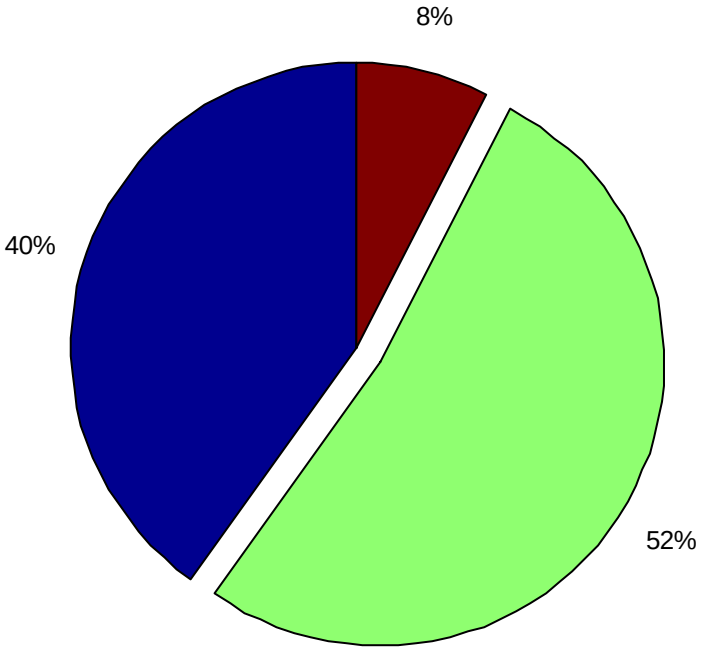


Рис. 4.5. Круговая диаграмма с максимальным сектором

4.2.3. Трехмерная круговая диаграмма с максимальным сектором

Создание таких выделяющихся элементов обеспечивается заданием еще од-ного аргумента такой же размерности, что и вектор q. Выдвигаемым секторам в новом векторе должны соответствовать ненулевые элементы. Например:

```
»[q_max q_ind]=max(q);
»v=zeros(1,size(q));
v(q_ind) = 1;
»pie3(q,v)
```

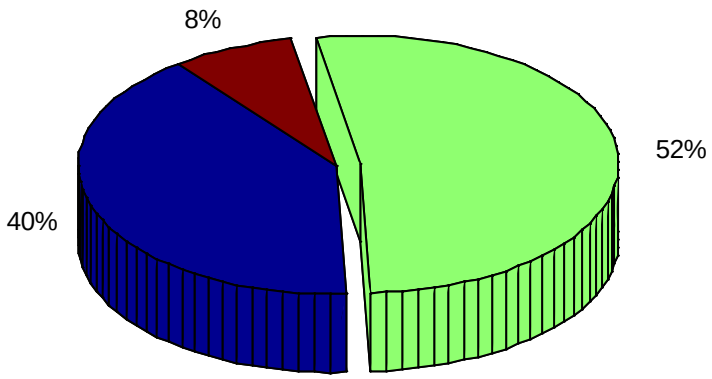


Рис. 4.6. Трехмерная круговая диаграмма с максимальным сектором

4.3. Гистограммы векторных данных

Классическая гистограмма характеризует числа попаданий значений элементов вектора Y в M интервалов с представлением этих чисел в виде столбцовой диаграммы. Для получения данных для гистограммы служит функция hist, записываемая в следующем виде:

- N=hist(Y) — возвращает вектор чисел попаданий для 10 интервалов, выбираемых автоматически. Если Y — матрица, то выдается массив данных о числе попаданий для каждого из ее столбцов;
- N=hist(Y,M) — аналогична вышерассмотренной, но используется M интервалов (M — скаляр);
- N=hist(Y,X) — возвращает числа попаданий элементов вектора Y в интервалы, центры которых заданы элементами вектора X;
- [N,X]=HIST(...) — возвращает числа попаданий в интервалы и данные о центрах интервалов.

Команда hist(...) с синтаксисом, аналогичным приведенному выше, строит график гистограммы.

В следующем примере строится гистограмма для 1000 случайных чисел и выводится вектор с данными о числах их попаданий в интервалы, заданные вектором x:

```
» y=randn(1000,1);
» hist(y)
```

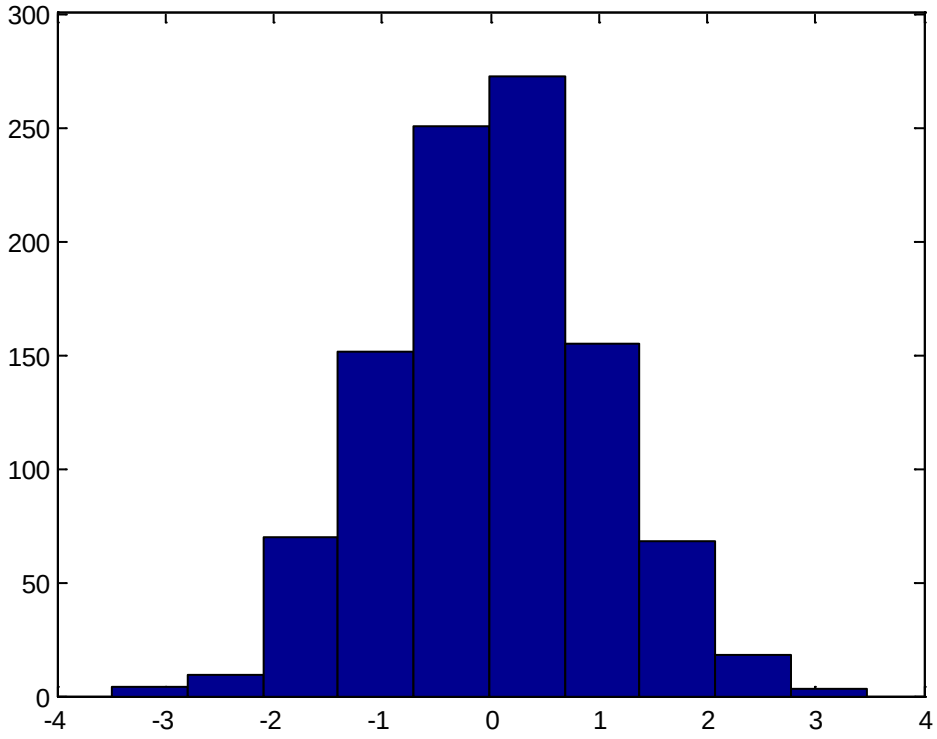


Рис. 4.7. Гистограммы векторных данных

4.3.1. Гистограмма векторных данных с центрами интервалов

См. п.4.6.

```
» x=-3:0.2:3;
» y=randn(1000,1);
» hist(y,x)
```

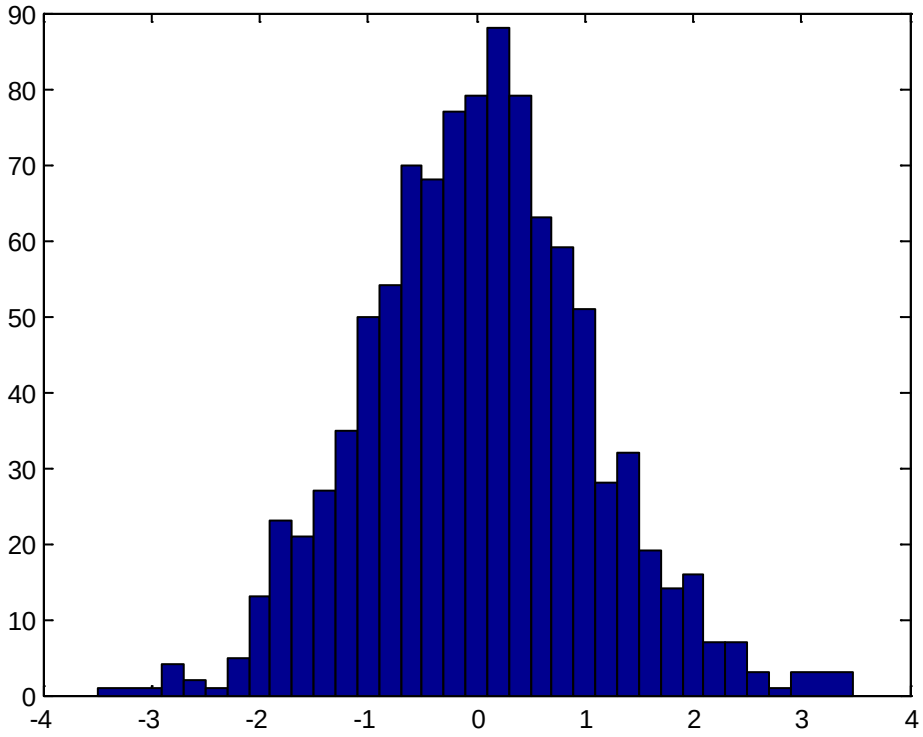


Рис. 4.8. Гистограммы векторных данных с центрами интервалов

4.4. Представление матричных данных

Система MATLAB может представить матрицу $A_{m,n}$ графически, в виде m диаграм с n столбцами на одном графике.

```
>> a = magic(6)
a =
35    1     6     26    19    24
3     32    7     21    23    25
31    9     2     22    27    20
8     28    33    17    10    15
30    5     34    12    14    16
4     36    29    13    18    11
>> bar(a)
```

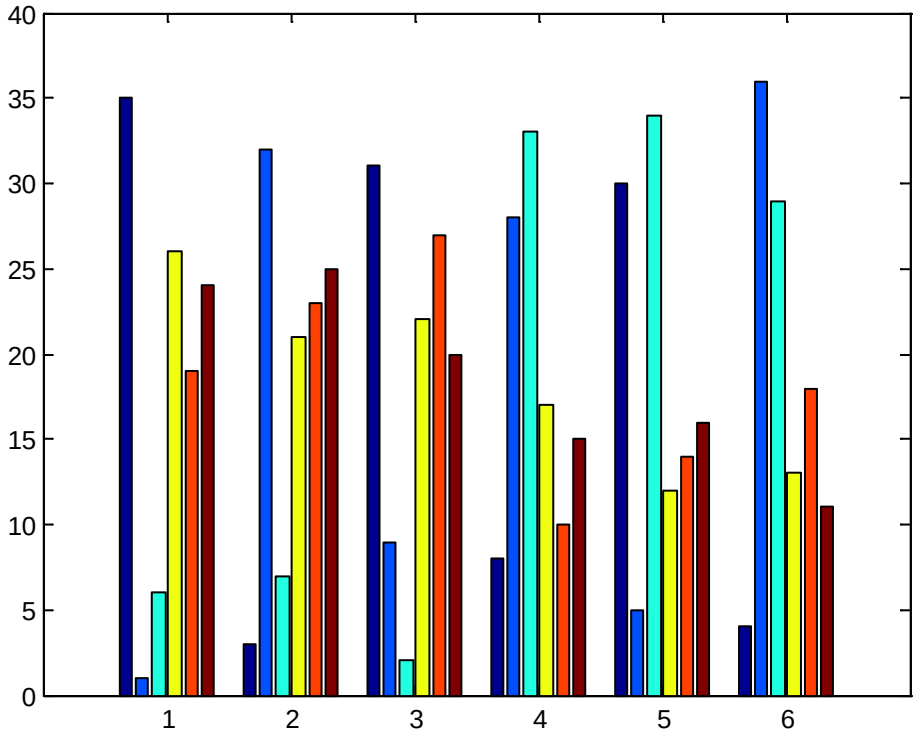


Рис. 4.9. Представление матричных данных

5. ПОСТРОЕНИЕ ГРАФИКОВ И ПОВЕРХНОСТЕЙ

5.1. Графики двух переменных

Построение графика функции двух переменных в MatLab на прямоугольной области определения переменных включает два предварительных этапа:

1. Разбиение области определения прямоугольной сеткой.
2. Вычисление значений функции в точках пересечения линий сетки и запись их в матрицу.

Построим график функции z(x, y)= x2 + y2 на области определения в виде квадрата x є [0, 1], y є [0, 1]. Необходимо разбить квадрат равномерной сеткой (например, с шагом 0.2) и вычислить значения функций в узлах, обозначенных точками.

Удобно использовать два двумерных массива x и y, размерностью шесть на шесть для хранения информации о координатах узлов. Массив x состоит из одинаковых строк, в которых записаны координаты x1, x2, ..., x6, а массив y содержит одинаковые столбцы с y1, y2, ..., y6.

Значения функции в узлах сетки запишем в массив z такой же размерности (6 x 6), причем для вычисления матрицы Z используем выражение для функции, но с поэлементными матричными операциями. Тогда, например z(3,4) как раз будет равно значению функции z(x,y)в точке (x3, y4).

Для генерации массивов сетки x и y по координатам узлов в MatLab предусмотрена функция meshgrid, для построения графика в виде каркасной поверхности - функция mesh. Следующие операторы приводят к появлению на экране окна с графиком функции (точка с запятой в конце операторов не ставится для того, чтобы проконтролировать генерацию массивов):

```
» [X, У] = meshgrid(0:0.2:1,0:0.2:1)
X =
0   0.2000   0.4000   0.6000   0.8000   1.0000
0   0.2000   0.4000   0.6000   0.8000   1.0000
```

```
0  0.2000  0.4000  0.6000  0.8000  1.0000
0  0.2000  0.4000  0.6000  0.8000  1.0000
0  0.2000  0.4000  0.6000  0.8000  1.0000
0  0.2000  0.4000  0.6000  0.8000  1.0000

y =
0       0       0       0       0       0
0.2000  0.2000  0.2000  0.2000  0.2000  0.2000
0.4000  0.4000  0.4000  0.4000  0.4000  0.4000
0.6000  0.6000  0.6000  0.6000  0.6000  0.6000
0.8000  0.8000  0.8000  0.8000  0.8000  0.8000
1.0000  1.0000  1.0000  1.0000  1.0000  1.0000

» Z = X.^2+Y.^2
Z =
0       0.0400  0.1600  0.3600  0.6400  1.0000
0.0400  0.0800  0.2000  0.4000  0.6800  1.0400
0.1600  0.2000  0.3200  0.5200  0.8000  1.1600
0.3600  0.4000  0.5200  0.7200  1.0000  1.3600
0.6400  0.6800  0.8000  1.0000  1.2800  1.6400
1.0000  1.0400  1.1600  1.3600  1.6400  2.0000

» mesh(X,Y,Z)
```

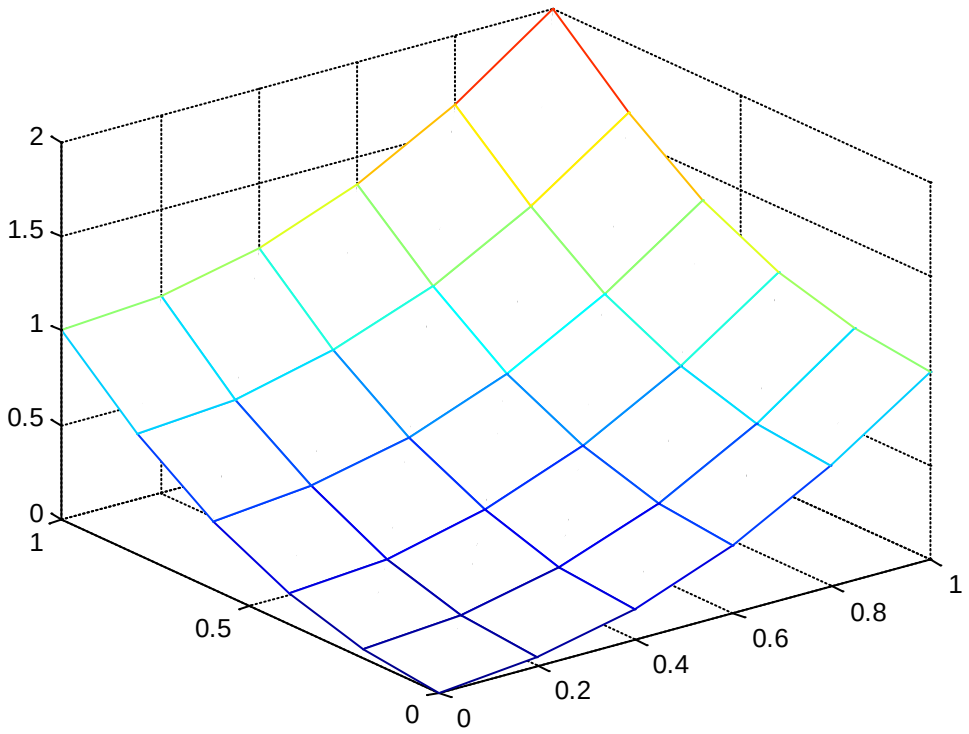


Рис. 5.1. График двух переменных на прямоугольной области определения переменных

5.2. Масштабы графиков

5.2.1. Графики в линейном масштабе

Графики в MATLAB могут строиться в линейном, логарифмическом и полулогарифмическом масштабе. В случае линейного масштаба система сама подбирает необходимый масштаб оси, но пользователю так же доступна возможность увеличивать некоторую область.

```
>> x = 0:0.01:10;
>> y = exp(-x).*sin(x);
>> plot(x,y)
```

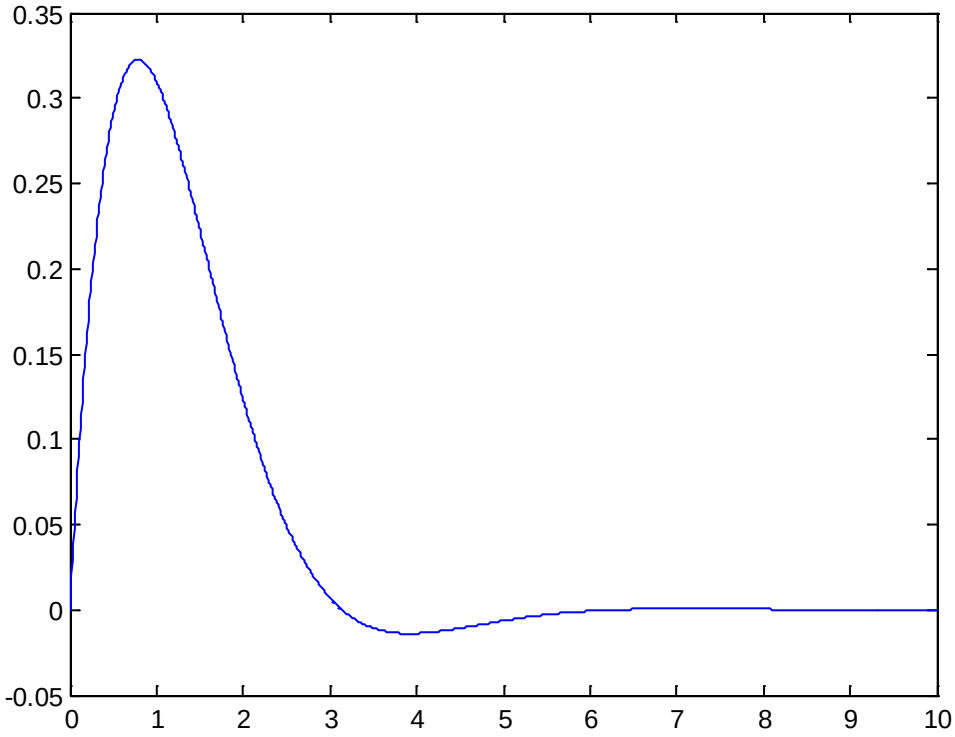


Рис. 5.2. График в линейном масштабе

5.2.2. Графики в логарифмических масштабах

Для построения графиков функций со значениями *x* и *y,* изменяющимися в широких пределах, нередко используются логарифмические масштабы. Рассмотрим команды, которые используются в таких случаях.

- loglog(...) — синтаксис команды аналогичен ранее рассмотренному для функции plot(...). Логарифмический масштаб используется для координатных осей *X* и У. Ниже дан пример применения данной команды:

```
» x=logspace(-1, 3);
» loglog(x,exp(x)./x)
» grid on
```

Ниже представлен график функции *exp(x)/x* в логарифмическом масштабе. Обратите внимание на то, что командой grid on строится координатная сетка.

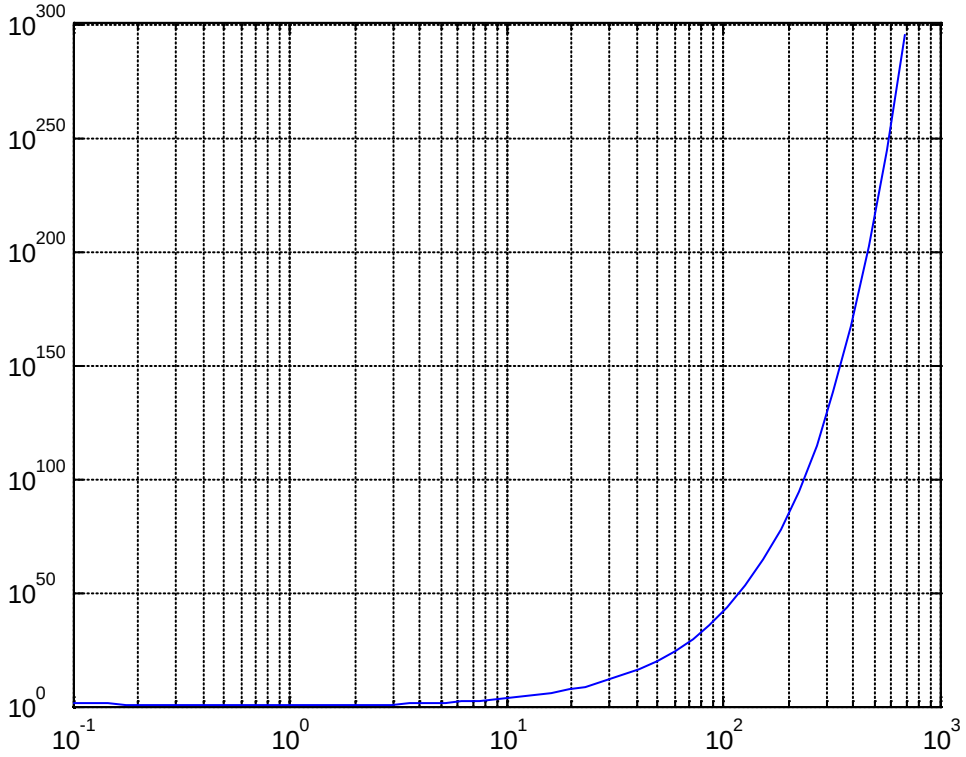


Рис. 5.3. График в логаритмических масштабах

Неравномерное расположение линий координатной сетки указывает на логарифмический масштаб осей.

В некоторых случаях предпочтителен *полулогарифмический* масштаб графиков, когда по одной оси задается логарифмический масштаб, а по другой — линейный.

Для построения графиков функций в полулогарифмическом масштабе используются следующие команды:

- semilogx(...) — строит график функции в логарифмическом масштабе (основание 10) по оси *X* и линейном по оси *Y;*
- semilogxy (...) — строит график функции в логарифмическом масштабе по оси *Y* и линейном по оси *X*.

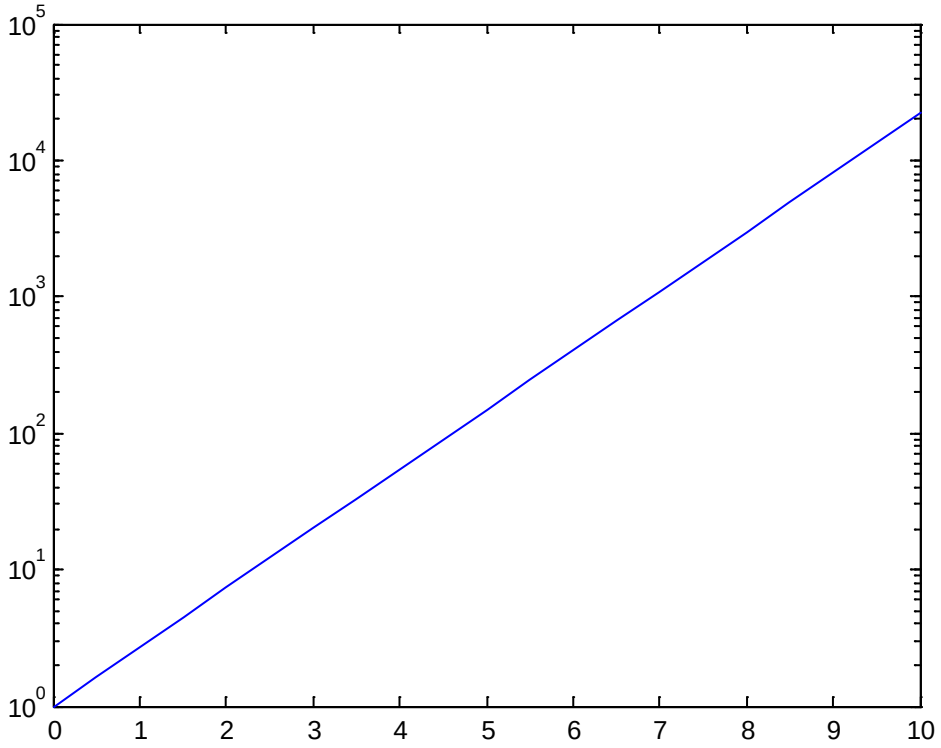


Рис. 5.4. График в полулогаритмических масштабах

Запись параметров (...) выполняется по аналогии с функцией plot(...). Ниже приводится пример построения графика экспоненциальной функции:

```
» x=0:0.5:10;
» semilogy(x,exp(x))
```

5.3. Графики параметрических и кусочно-заданных функций

Для построения параметрических графиков достаточно указать зависимость переменных, которые будут выражать ось абсцисс и ординат. Например

```
>> t = -pi:0.01:pi;
>> x = 2*sin(t);
>> y = 5*cos(t);
>> plot(x,y)
```

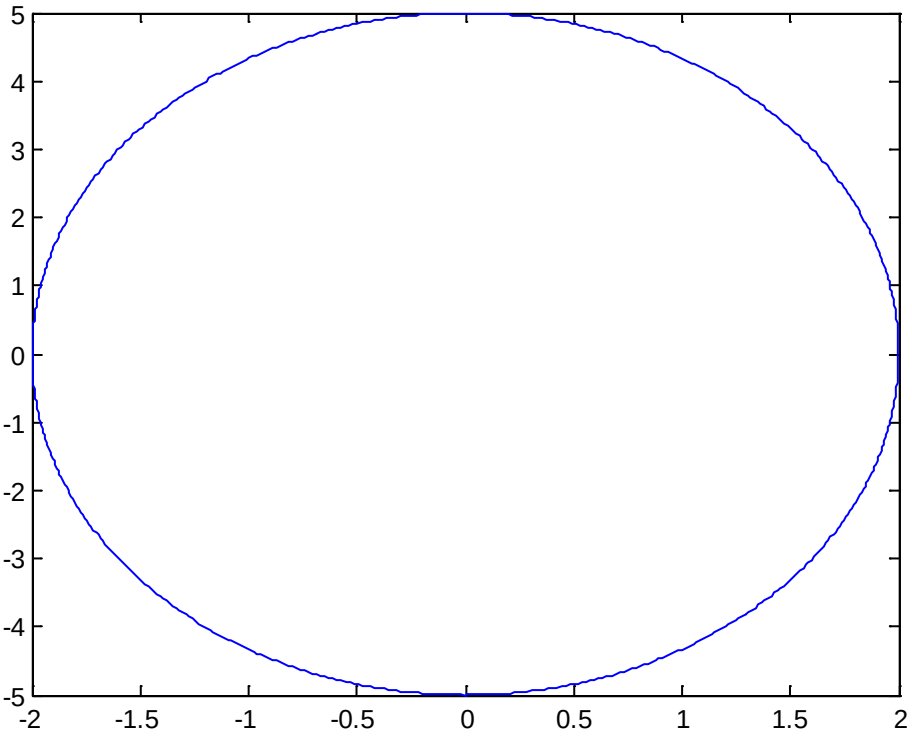


Рис. 5.5. Параметрический график

5.4. Трехмерные графики функций

Особенно наглядное представление о поверхностях дают сетчатые графики, использующие функциональную закраску ячеек.

Например, цвет окраски поверхности *z(x, y)* может быть поставлен в соответствие с высотой *z* поверхности с выбором для малых высот темных тонов, а для больших — светлых.

Для построения таких поверхностей используются команды класса surf (...):

- surf (X, Y, Z. C) — строит цветную параметрическую поверхность по данным матриц X, Y и Z с цветом, задаваемым массивом C;
- surf(X.Y.Z) — аналогична предшествующей команде, где C=Z, так что цвет задается высотой той или иной ячейки поверхности;
- surf(x.y.Z) и surf(x.y.Z.C) с двумя векторными аргументами x и y — векторы x и y заменяют первых два матричных аргумента и

должны иметь длины length(x)=n и length(y)=m, где [m.n]=size(Z). В этом случае вершины областей поверхности представлены тройками координат (x(j), yd), Z(1,j)). Заметим, что x соответствует столбцам Z, а y соответствует строкам;

- surf(Z) и surf(Z.C) используют x = 1:n и y = 1:m. В этом случае высота Z — однозначно определенная функция, заданная геометрически прямоугольной сеткой;
- h=surf (...) —строит поверхность и возвращает дескриптор объекта класса surface.

Команды axis, caxis, color-map, hold, shading и view задают координатные оси и свойства поверхности, которые могут использоваться для большей эффектности показа поверхности или фигуры.

Ниже приведен простой пример построения поверхности — параболоида:

```
» [X,Y]=meshgrid([-3:0.15:3]);
» Z=X.^2+Y.^2;
» surf(X,Y,Z)
```

Соответствующий этому примеру график показан ниже.

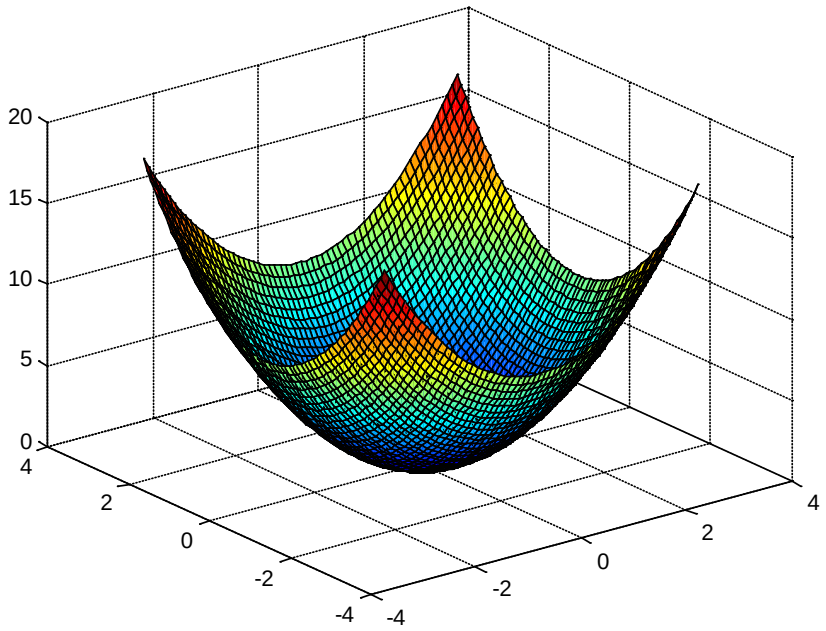


Рис. 5.6. График параболоида

Можно заметить, что благодаря функциональной окраске график поверхности гораздо более выразителен, чем при построениях без такой окраски, представленных ранее (причем даже в том случае, когда цветной график печатается в черно-белом виде).

В следующем примере используется функциональная окраска оттенками серого цвета с выводом шкалы цветовых оттенков:

```
» [X,Y]=meshgrid([-3:0.1:3]);
» Z=sin(X)./(X.^2+Y.^2+0.3);
» surf(X,Y,Z)
» colormap(gray)
» shading interp
» colorbar
```

В этом примере команда colormap(gray) задает окраску тонами серого цвета, а команда shading Interp обеспечивает устранение изображения сетки и задает интерполяцию для оттенков цвета объемной поверхности.

Ниже показан вид графика, построенного в этом примере.

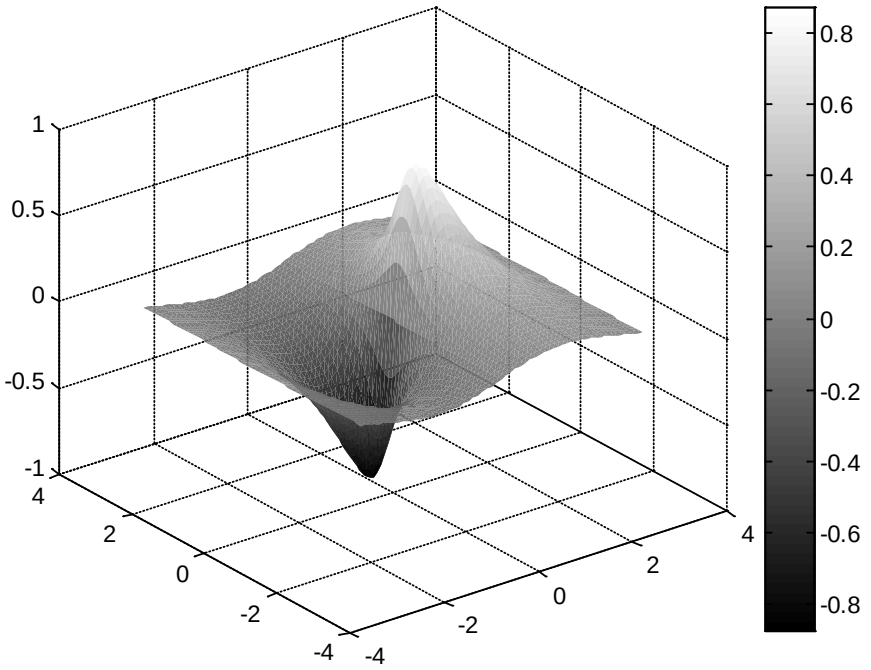


Рис. 5.7. График, окрашенный в оттенки серого

Обычно применение интерполяции для окраски придает поверхностям и фигурам более реалистичный вид, но фигуры каркасного вида дают более точные количественные данные о каждой точке.

Для повышения наглядности представления поверхностей можно использовать дополнительный график линий равного уровня, получаемый путем проецирования поверхности на опорную плоскость графика (под поверхностью). Для этого используется команда surf c:

- surfc(...) — аналогична команде surf, но обеспечивает дополнительное построение контурного графика проекции фигуры на опорную плоскость.

Пример применения команды surf c приводится ниже:

```
» [X,Y]=meshgrid([-3:0.1:3]);
» Z=sin(X)./(X.^2+Y.^2+0.3);
» surfc(X,Y,Z)
```

Ниже показаны графики, построенные в данном примере.

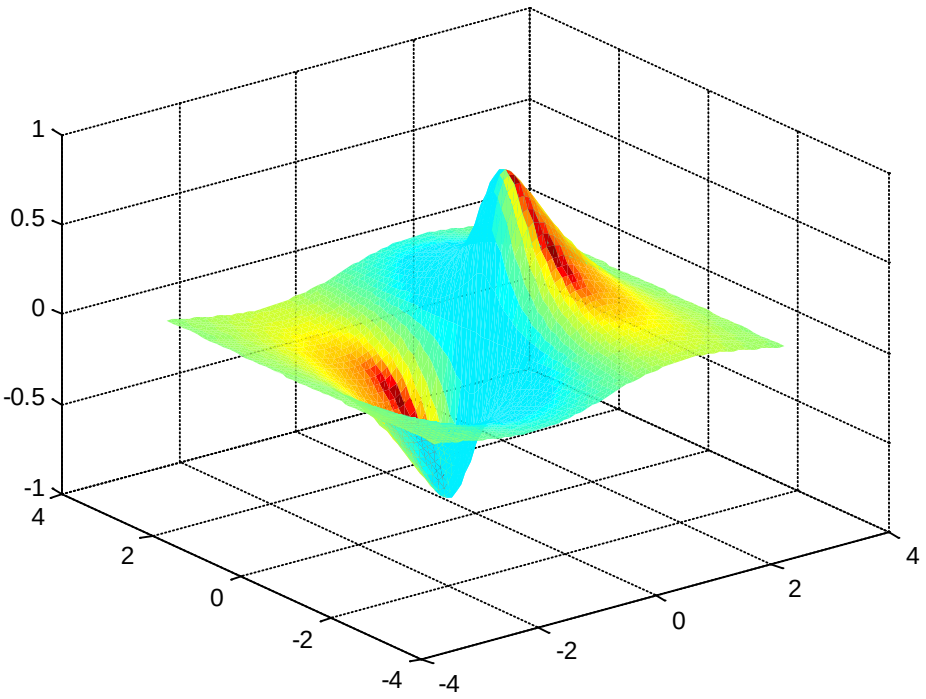


Рис. 5.8. График линий равного уровня

Рассмотрим еще один пример применения команды surfc, на этот раз для построения поверхности, описываемой функцией peaks, с применением интерполяции цветов и построением цветовой шкалы:

```
» [X,Y]=meshgrid([-3:0.1:3]);
» Z=peaks(X,Y);
» surfc(X,Y,Z)
» shading interp
» colorbar
```

И здесь нетрудно заметить, что графики сложных поверхностей с интерполяцией цветовых оттенков выглядят более реалистичными, чем графики сетчатого вида и графики без интерполяции цветов.

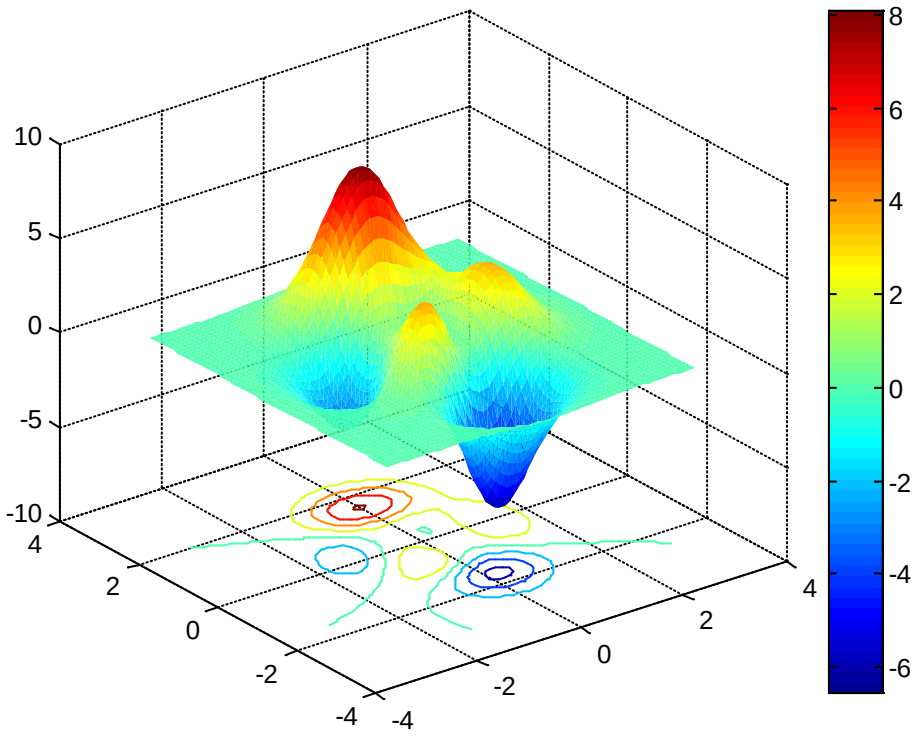


Рис. 5.9. График сложной поверхности с интерполяцией световых оттенков

5.4.1. Построение параметрически заданных поверхностей и линий

Для этого используется команда plot3(...), которая является аналогом команды plot(...), но относится к функции двух переменных *z(x, y)*. Она строит аксонометрическое изображение трехмерных поверхно-

стей и представлена следующими формами:

- plot3(x.y,z) — строит массив точек, представленных векторами x, y и z, соединяя их отрезками прямых. Эта команда имеет ограниченное применение;
- plot3(X,Y,Z), где X, Y и Z — три матрицы одинакового размера, строит точки с координатами X(i.:), Y(i,:) и Z(i,:) и соединяет их отрезками прямых.

Ниже дан пример построения трехмерной поверхности, описываемой функцией $z(x,y)=x^2+y^2$:

```
» [X,Y]=meshgrid([-3:0.15:3]);
» Z=X.^2+Y.^2;
» plot3(X,Y,Z)
```

График этой поверхности показан на рис. 5.13.

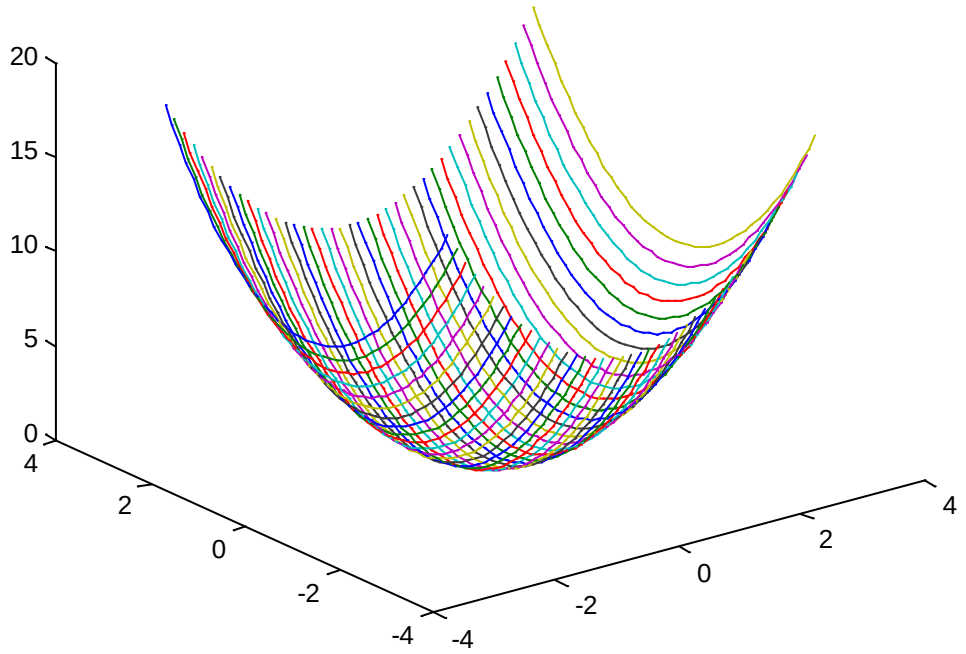


Рис. 5.10. График $z(x,y)=x^2+y^2$

- plot3 (X. Y, Z, S) — обеспечивает построения, аналогичные рассмотренным ранее, но со спецификацией стиля линий и точек, соответствующей спецификации команды plot. Ниже дан пример применения этой команды для построения поверхности кружками:

```
» [X,Y]-meshgrid([-3:0.15:3]);
» Z=X.^2+Y.^2;
» plot3(X,Y,Z,'o')
```

График поверхности, построенный кружками, показан ниже.

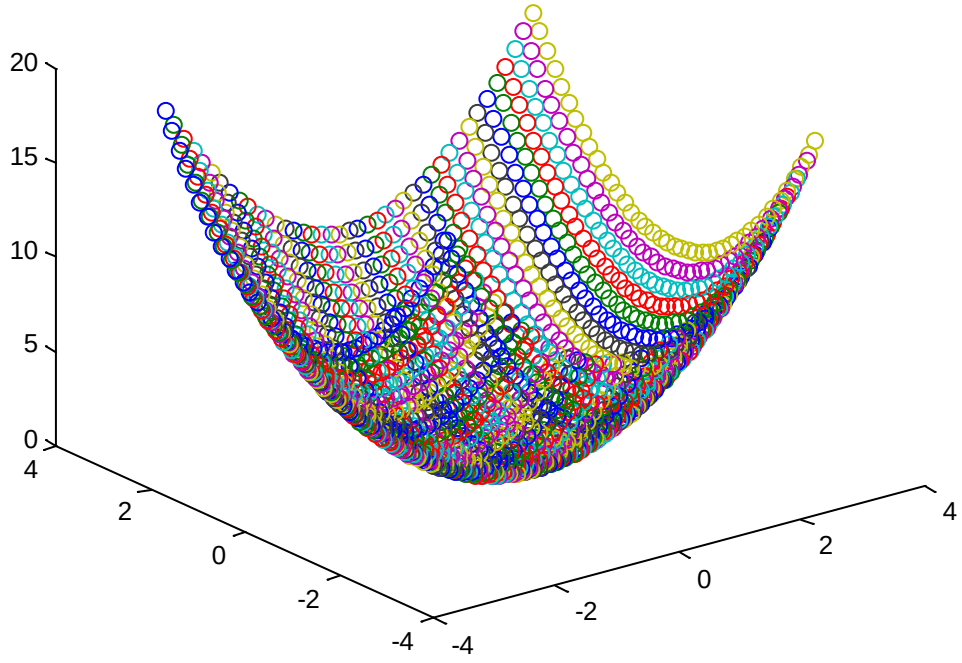


Рис. 5.11. График поверхности, построенный кружками

- plot3(xl ,yl, zl, sl, x2, y2, z2, s2, x3, y3, z3, s3, ...)— строит на одном рисунке графики нескольких функций zl(xl ,yl), z2(x2,y2) и т. д. со спецификацией линий и маркеров каждой из них.

Пример применения последней команды дан ниже:

```
» [X,Y]=meshgrid([-3:0.15:3]);
```

```
» Z=X.^2+Y.^2;
» plot3(X,Y,Z,'-k',Y,X,Z,'-k')
```

График функции, соответствующей последнему примеру, представлен ниже.

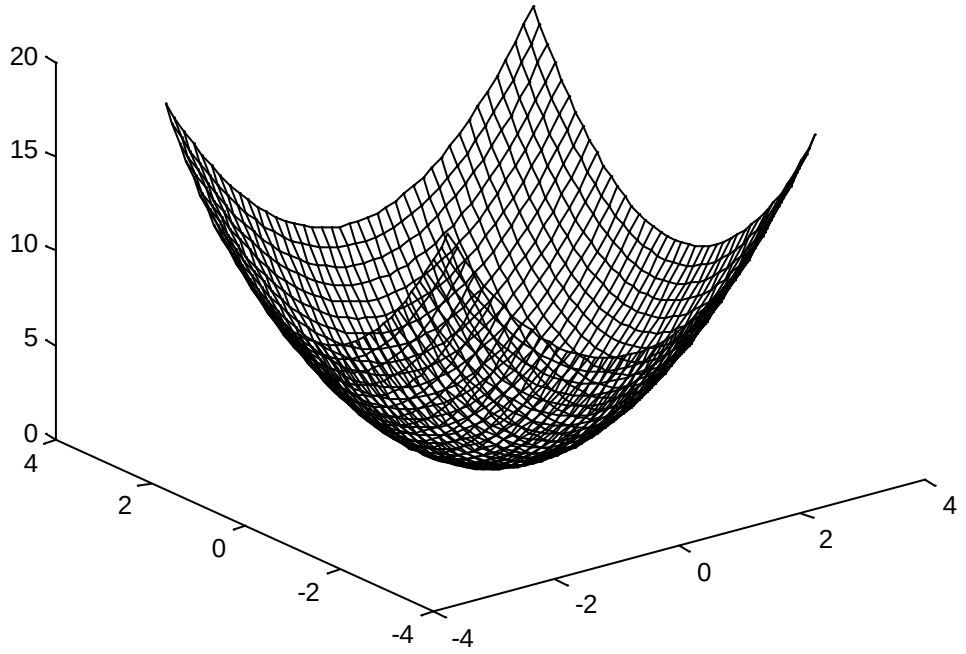


Рис. 5.12. График поверхности, имеющий сеточный вид

В данном случае строятся два графика одной и той же функции с взаимно перпендикулярными образующими линиями. Поэтому график имеет вид сетки без окраски ее ячеек (напоминает проволочный каркас фигуры).

Для наглядности удобно пользоваться командами добавления градаций цвета. Например:

```
>> [X,Y]=meshgrid([-3:0.15:3]);
>> Z=X.^2+Y.^2;
>> surf(X,Y,Z)
>> colormap(copper)
```

Ниже показан результат выполнения команд.

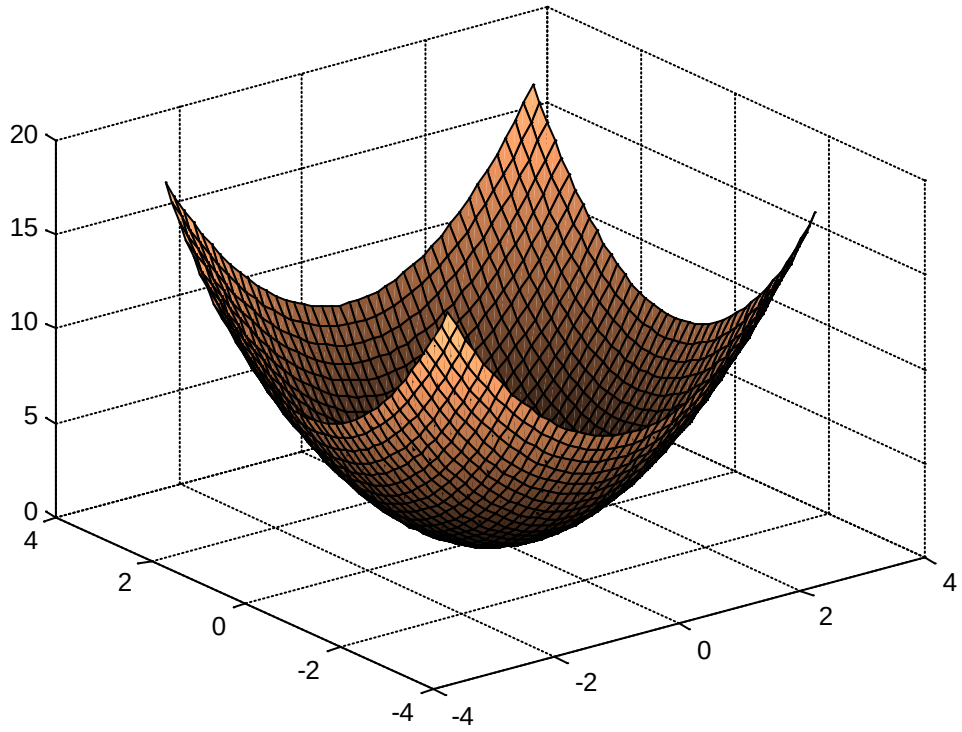


Рис. 5.13. Цветной график поверхности, имеющий сеточный вид

5.4.2. Построение освещенной поверхности

Пожалуй, наиболее реалистичный вид имеют графики поверхностей, в которых имитируется освещение от точечного источника света, расположенного в заданном месте координатной системы. Графики имитируют оптические эффекты рассеивания, отражения и зеркального отражения света.

Для получения таких графиков используется команда surfl:

- surfl(...) — аналогична команде surf (...), но строит график поверхности с подсветкой от источника света;
- surfl (Z.S) или surfl(X,Y,Z,S) — строит графики поверхности с подсветкой от источника света, положение которого в системе декартовых координат задается вектором S=[Sx,Sy.Sz], а в системе сферических координат — вектором S=[AZ.EL];
- surfl (..., 'light') — позволяет при построении задать цвет подсветки с помощью объекта Light;
- surfl (..., 'cdata') — при построении имитирует эффект отражения;

- surfl(X,Y,Z,S.K) — задает построение поверхности с параметрами, заданными вектором K=[ka,kd,ks.spread], где ka — коэффициент фоновой подсветки, kd — коэффициент диффузного отражения, ks — коэффициент зеркального отражения и spread — коэффициент глянцевитости;
- H=surfl (...) — строит поверхность и возвращает дескрипторы поверхности и источников света.

По умолчанию вектор S задает углы азимута и возвышения в 45°. Используя команды cla, hold on, view(AZ,EL), surfl (...) и hold off, можно получить дополнительные возможности управления освещением.

Надо полагаться на упорядочение точек в X, Y, и Z матрицах, чтобы определить внутреннюю и внешнюю стороны параметрических поверхностей.

Попробуйте транспонировать матрицы и использовать surfl(X' ,Y' ,Z'), если вам не понравился результат работы этой команды. Для вычисления векторов нормалей поверхности surfl требует в качестве аргументов матрицы с размером по крайней мере 3x3.

Ниже представлен пример применения команды surfl:

```
» [X.Y]=meshgrid([-3:0.1:3]);
» Z=sin(X)./(X.^2+Y.^2+0.3);
» surfl(X.Y.Z)
» colormap(gray)
» shading interp
» colorbar
```

Построенная в этом примере освещенная поверхность представлена ниже.

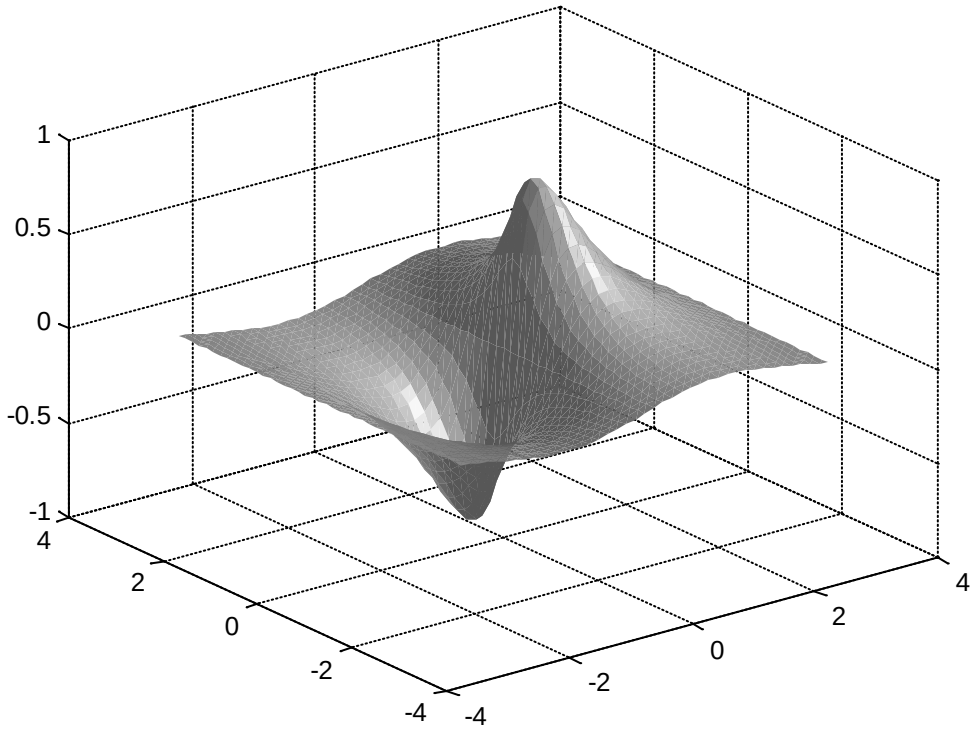


Рис. 5.14. График освещенной поверхности

5.4.3. Построение освещенной поверхности и изменение азимута источника на –90° по отношению к наблюдателю

Покажем на примере как изменять изменение азимута источника света:

```
» [X.Y]=meshgrid([-3:0.1:3]);
» Z=sin(X)./(X.^2+Y.^2+0.3);
» [Az, El] = view;
» surfl(X,Y,Z,[Az-90, 0])
» colormap(copper)
» shading interp
```

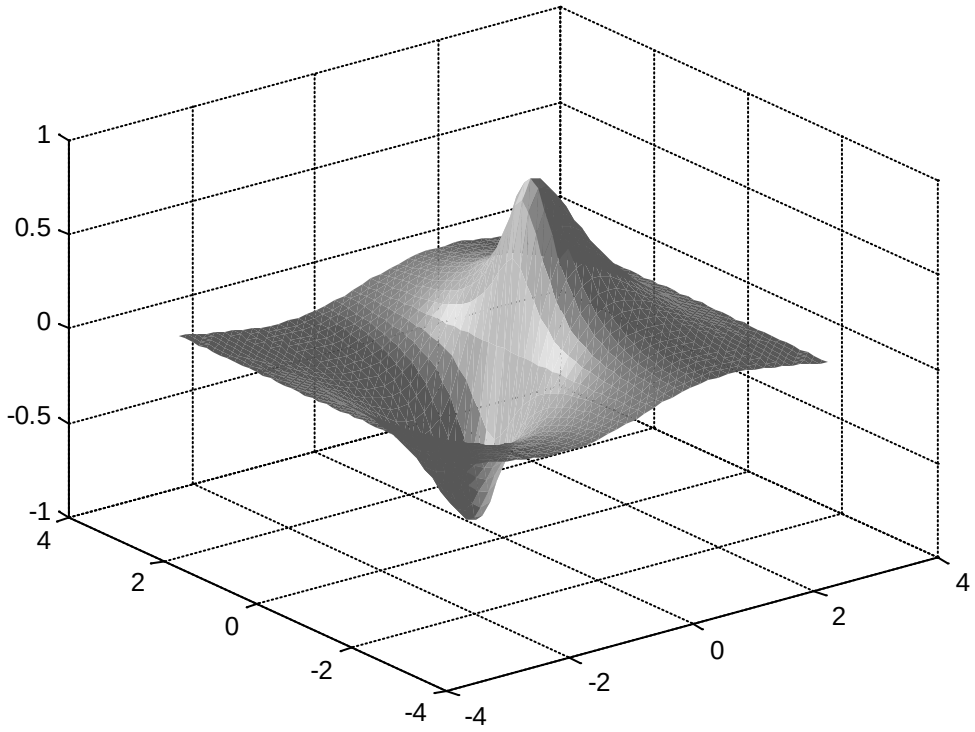


Рис. 5.15. График освещенной поверхности с измененным азимутом

5.5. Анимированные графики

Для отображения движения точки по траектории используется команда comet. При этом движущаяся точка напоминает ядро кометы с хвостом. Используются следующие формы представления этой команды:

- comet (Y) — отображает движение «кометы» по траектории, заданной вектором Y;
- comet (X, Y) — отображает движение «кометы» по траектории, заданной парой векторов Y и X;
- comet (X, Y, p) — аналогична предшествующей команде, но позволяет задавать длину хвоста кометы (отрезка траектории, выделенного цветом) как p*length(Y), где length(Y) - размер вектора Y, а p<1. По умолчанию p = 0.1

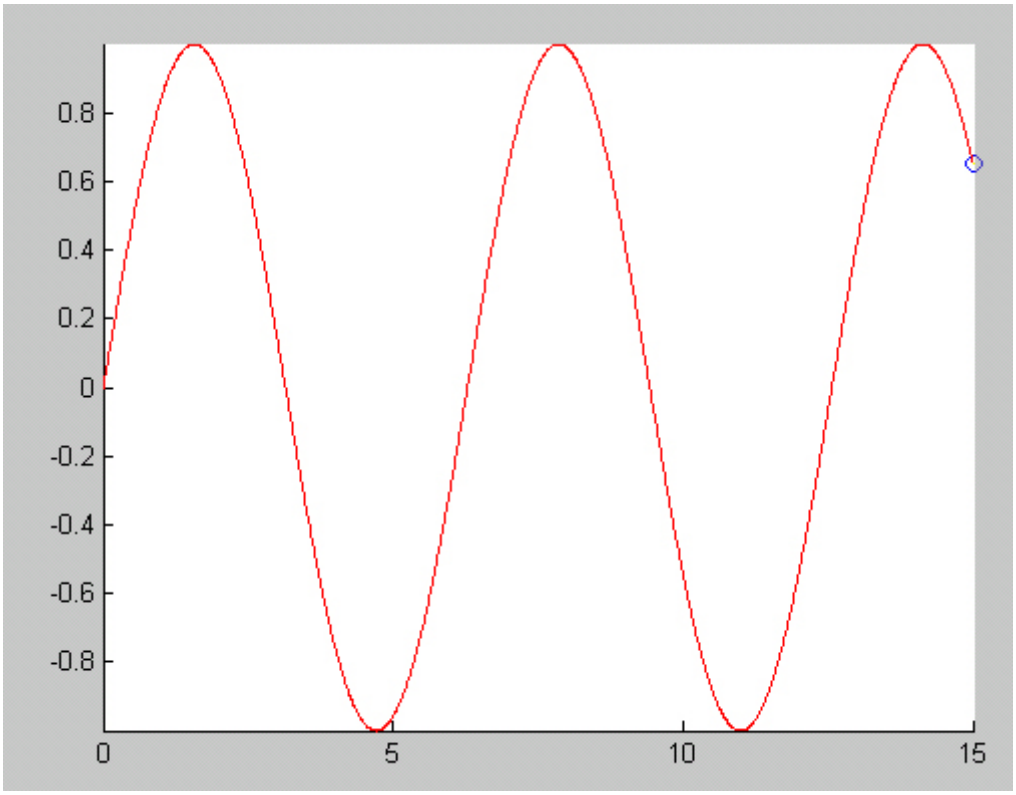


Рис. 5.16. Анимированный график

Следующий пример иллюстрирует применение команды comet:

```
» X=0:0.01:15;
» comet(X.sin(X).0.15)
```

Стоп-кадр изображения показан на рисунке выше. «Хвост кометы» на черно-белом рисунке заметить трудно, поскольку он представляет собой отрезок линии с цветом, отличающимся от цвета линии основной части графика.

5.5.1. Траектория движения точки, перемещающейся в пространстве

Есть еще одна команда, которая позволяет наблюдать движение точки, но уже в трехмерном пространстве. Это команда comet3:

- comet3(Z) — отображает движение точки с цветным «хвостом» по трехмерной кривой, определенной массивом Z;
- comet3 (X.Y.Z) — отображает движение точки «кометы» по кривой в пространстве, заданной точками [X(i),Y(i),Z(i)];

- comet3(X,Y,Z,p) — аналогична предшествующей команде с заданием длины «хвоста кометы» как p*1ength(Z). По умолчанию параметр p равен 0.1.

Ниже представлен пример применения команды comet3:

```
» W=0:pi/500:10*pi;
» comet3(cos(W).sin(W)+W/10.W)
```

Ниже показан стоп-кадр изображения, созданного командой comet3.

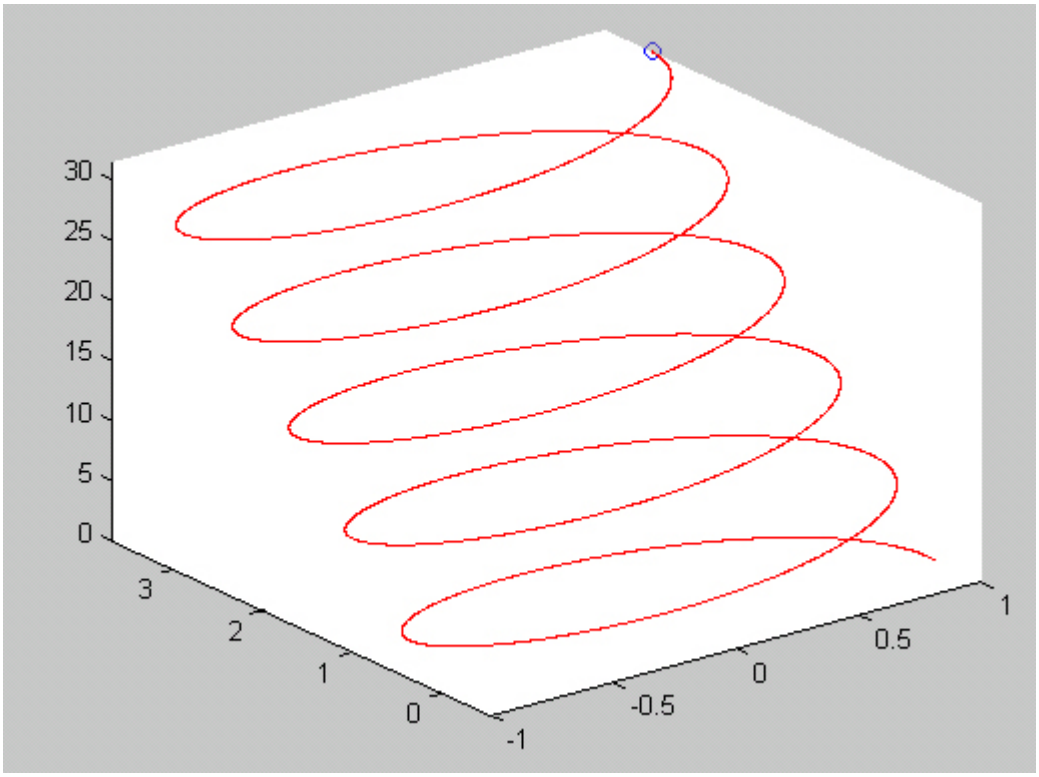


Рис. 5.17. График движения точки, движущейся в трехмерном пространстве

Разумеется, движение точки по заданной траектории, как в двумерном, так и в трехмерном пространстве является самым простейшим примером анимации. Тем не менее эти средства существенно расширяют возможности графической визуализации при решении ряда задач динамики.

5.6. Вывод графиков

5.6.1. Два графика на одних осях

Во многих случаях желательно построение многих наложенных друг на друга графиков в одном и том же окне. Для этого служит команда продолжения графических построений hold. Она используется в следующих формах:

- hold on — обеспечивает продолжение вывода графиков в текущее окно, что позволяет добавлять последующие графики к уже существующим;
- hold off — отменяет режим продолжения графических построений;
- hold — работает как переключатель, последовательно включая режим продолжения графических построений и отменяя его.

Команда hold on устанавливает значение add для свойства NextPlot объектов figure и axes, a hold off устанавливает для этого свойства значение replace.

Приведенный ниже пример показывает, как с помощью команды hold on на график синусоиды накладываются еще три графика параметрически заданных функций:

```
» x=-5:0.1:5;
» plot(x,sin(x))
» hold on
» plot(sin(x),cos(x))
» plot(2*sin(x),cos(x))
» plot(4*sin(x),cos(x))
» hold off
```

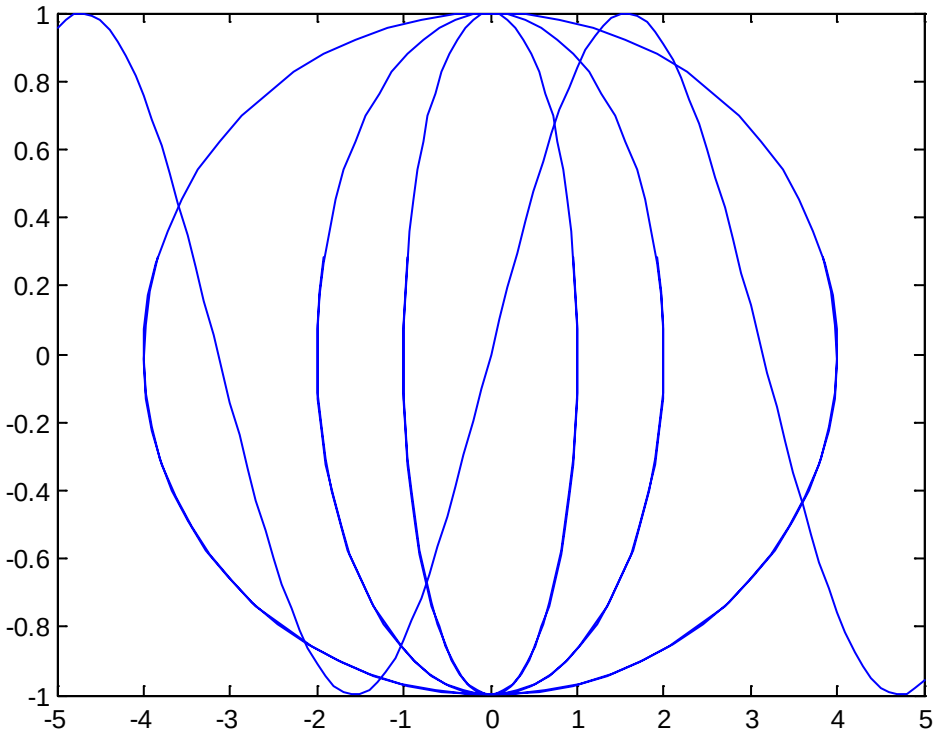


Рис. 5.18. Два графика на одних осях

5.6.2. Вывод графика в окно с двумя вертикальными осями. Функция *plotyy*.

```
>> x = 0:0.1:3;
>> y1 = x.^-3;
>> y2 = 10^4*(x+1).^2;
>> plotyy(x,y1,x,y2)
```

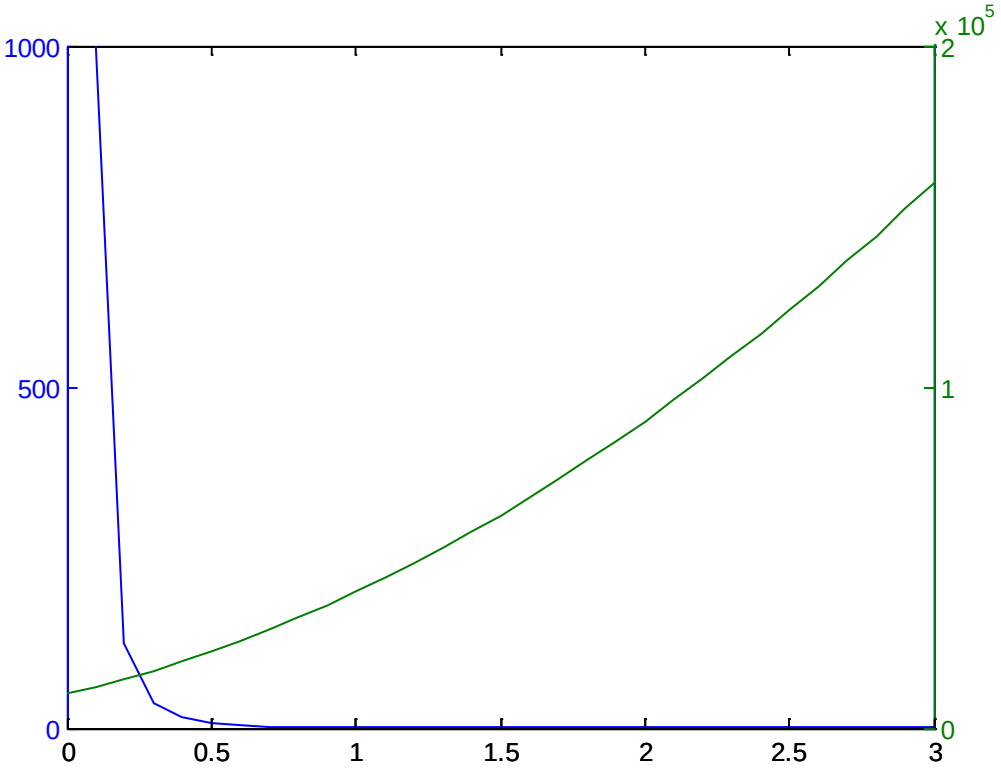


Рис. 5.19. График с двумя вертикальными осями

5.6.3. Оформление графиков

MATLAB позволяет пользователю задавать тип линий графика.

- plot(X,Y,S) — аналогична команде plot(X,Y), но тип линии графика можно задавать с помощью строковой константы S.

Значениями константы S могут быть следующие символы.

Цвет линии	
Y	Желтый
M	Фиолетовый
C	Голубой
R	Красный
G	Зеленый
B	Синий
W	Белый
K	Черный

Тип точки

.	Точка
0	Окружность
X	Крест
+	Плюс
*	Звездочка
S	Квадрат
D	Ромб
V	Треугольник (вниз)
A	Треугольник (вверх)
<	Треугольник (влево)
>	Треугольник (вправо)
P	Пятиугольник
H	Шестиугольник

Тип линии

-	Сплошная
;	Двойной пунктир
-.	Штрих-пунктир
--	Штриховая

Таким образом, с помощью строковой константы S можно изменять цвет линии, представлять узловые точки различными отметками (точка, окружность, крест, треугольник с разной ориентацией вершины и т. д.) и менять тип линии графика.

- 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 и Y_i — векторы или матрицы, а S_i — строки. С помощью такой конструкции возможно построение, например, графика функции линией, цвет которой отличается от цвета узловых точек. Так, если надо построить график функции линией синего цвета с красными точками, то вначале надо задать построение графика с точками красного цвета (без линии), а затем графика только линии синего цвета (без точек).

При отсутствии указания на цвет линий и точек он выбирается ав-

томатически из таблицы цветов (белый исключается). Если линий больше шести, то выбор цветов повторяется. Для монохромных систем линии выделяются стилем.

Рассмотрим пример построения графиков трех функций с различным стилем представления каждой из них:

```
» x=-2*pi:0.1*pi:2*pi;
» yl=sin(x);
» y2=sin(x).^2;
» y3=sin(x).^3:
» plot(x,yl,'-m',x,y2,'-.+r',x,y3,'--ok')
```

Нарисуем графики функций для этого примера.

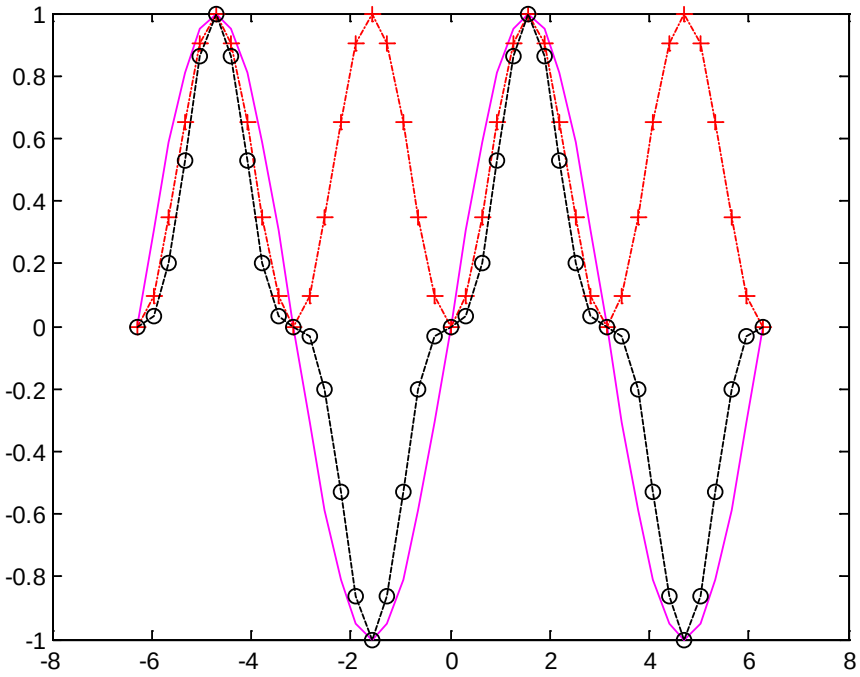


Рис. 5.20. Графики трех функций

Здесь график функции yl строится сплошной фиолетовой линией, график y2 строится штрих пунктирной линией с точками в виде знака «плюс» красного цвета, а график y3 строится штриховой линией с кружками черного цвета. К сожалению, на черно-белых рисунках этой книги вместо разных цветов видны разные градации серого цвета.

После того как график уже построен, MATLAB позволяет выполнить его форматирование или оформление в нужном виде. Соответствующие этому средства описаны ниже. Так, для установки над графиком титульной надписи используется следующая команда:

- title('string') — установка на двумерных и трехмерных графиках титульной надписи, заданной строковой константой 'string'.

Для установки надписей возле осей *x, y* и z используются следующие команды:

```
xlabel('String')
ylabel('String')
zlabel('String')
```

Пояснение в виде отрезков линий со справочными надписями, размещаемое внутри графика или около него, называется *легендой.* Для создания легенды используются различные варианты команды legend:

- legend(stringl,string2. strings,...) — добавляет к текущему графику легенду в виде строк, указанных в списке параметров;
- legend (H.stringl,string2. strings,...) — помещает легенду на график, содержащий объекты с дескрипторами H, используя заданные строки как метки для соответствующих дескрипторов;
- legend(AX . ...) — помещает легенду в осях (объект класса axes) с дескриптором AX;
- legend (M) — размещает легенду, используя данные из строковой матрицы M;
- legend OFF — устраняет ранее выведенную легенду;
- legend — перерисовывает текущую легенду, если таковая имеется;
- legend(I egendhandl e) — перерисовывает легенду, указанную дескриптором legendhandle;
- legend (....Pos) — помещает легенду в точно определенное место, специфицированное параметром Pos:

5.6.4. Вывод графиков в отдельные окна

Вывод графиков в отдельные окна осуществляется командой `figure`. Таким образом для каждого графика, который необходимо вывести в новое окно, нужно указать команду `figure` перед вызовом

функции построения этого графика.

```
>> x = 0:0.1:4;
>> y1 = sin(x);
>> y2 = exp(x);
>> plot(x,y1);
>> figure;
>> plot(x,y2);
```

В результате появится 2 графических окна, с графиками функций y1 и y2 соответственно.

5.6.5. Вывод нескольких графиков на одни оси

Во многих случаях желательно построение многих наложенных друг на друга графиков в одном и том же окне. Для этого служит команда продолжения графических построений hold. Она используется в следующих формах:

- hold on — обеспечивает продолжение вывода графиков в текущее окно, что позволяет добавлять последующие графики к уже существующим;
- hold off — отменяет режим продолжения графических построений;
- hold — работает как переключатель, последовательно включая режим продолжения графических построений и отменяя его.

Команда hold on устанавливает значение add для свойства NextPlot объектов figure и axes, а hold off устанавливает для этого свойства значение replace.

```
>> [X,Y]=meshgrid([-5:0.15:5]);
>> Z = peaks(X, Y);
>> surf(X, Y, Z)
>> Z1 = -X.^2 - Y.^2;
>> hold on;
>> surf(X, Y, Z1)
```

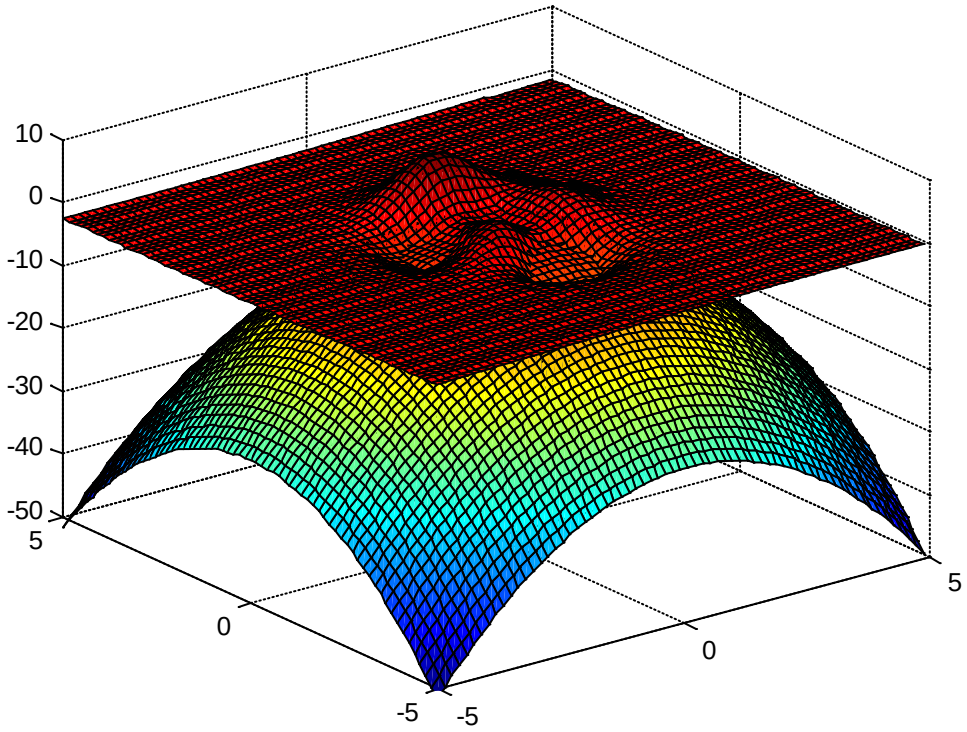


Рис. 5.21. Несколько графиков в одном окне

5.6.6. Несколько графиков в одном графическом окне

Бывает, что в одном окне надо расположить несколько координатных осей с различными графиками без наложения их друг на друга. Для этого используются команды subplot, применяемые перед построением графиков:

- subplot — создает новые объекты класса axes (подокна);
- subplot(m.n.p) или subplot(mnp) — разбивает графическое окно на *mxn*подокон, при этом m — число подокон по горизонтали, n — число подокон по вертикали, а p— номер подокна, в которое будет выводиться текущий график (подокна отсчитываются последовательно по строкам);
- subplot(H), где H — дескриптор для объекта axes, дает альтернативный способ задания подокна для текущего графика;
- subplot('position',[left bottom width height]) — создает подокно с заданными нормализованными координатами (в пределах от 0.0 до 1.0);

- subplot(lll) Hclf reset — удаляют все подокна и возвращают графическое окно в обычное состояние.

Следующий пример иллюстрирует применение команды subplot:

```
» x=-5:0.1:5;
subplot(2,2,1),plot(x, sin(x))
subplot(2,2,2),plot(sin(5*x), cos(2*x+0.2))
subplot(2,2,3),contour(peaks)
subplot(2,2,4),surf(peaks)
```

В этом примере последовательно строятся четыре графика различного типа, размещаемых в разных подокнах.

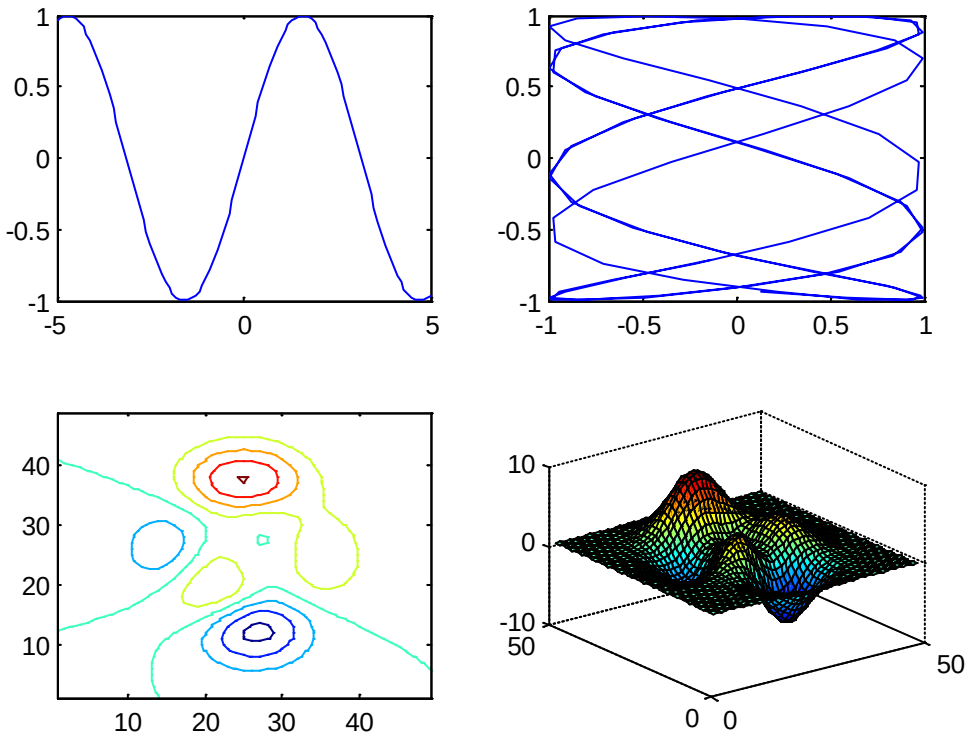


Рис. 5.22. Несколько графиков в одном графическом окне

Следует отметить, что для всех графиков возможна индивидуальная установка дополнительных объектов, например титульных надписей, надписей по осям и т. д.

УЧЕБНО-МЕТОДИЧЕСКАЯ ЛИТЕРАТУРА

1. Мироновский Л.А., Петрова К.Ю. Введение в MATLAB [учебное пособие]. – СПб, СПбГУАП, 2005.
2. Поршев, С.В. MATLAB. Основы работы и программирования: учебное пособие для вузов [учебное пособие]. – Москва, Бином, 2006.
3. Кетков, Ю.Л., Шульц М.М. MATLAB 7. Программирование, численные методы [учебное пособие]. – СПб, БХВ-Петербург, 2005.
4. Higham N..J.MATLAB guide [учебное пособие]. – SIAM, Philadelphia, 2002.
5. Селезнев А.В. Эффективное программирование в среде MATLAB [электронный ресурс] – Режим доступа: http://vmg.pp.ua/books/КопьютерыИсети - Загл. с экрана.
6. Язык программирования MATLAB [электронный ресурс] – Режим доступа: http://life-prog.ru/view_cat.php?cat=5 – Загл.с экрана.
7. The MathWorks. Object-oriented programming in MATLAB [электронный ресурс] – Режим доступа: http://www.mathworks.com/discovery/object-oriented-programming.html – Загл.с экрана.

Учебное издание

Особенности системы MatLAB для решения задач вычислительной математики

Учебное пособие

Составитель
КОЧЕГУРОВА ЕЛЕНА АЛЕКСЕЕВНА

Научный редактор
кандидат технических наук, доцент кафедры АиКС ИК
И.В. Цапко