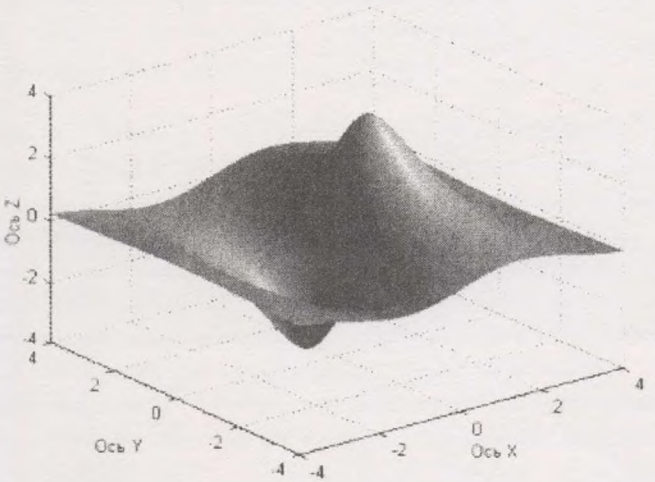
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Томский государственный университет

А.В. Иванов, А.В. Клоков, В.П. Якубов

MATLAB

Неограниченные возможности физика



Учебно-методическое пособие

Томск – 2004

УТВЕРЖДАЮ
Зав. кафедрой радиофизики, профессор
В.П. Якубов
"14" 12 2004 г.



Указания РАССМОТРЕНЫ И УТВЕРЖДЕНЫ методической комиссией радиофизического факультета
Протокол № 46 от "14" декабря 2004 г.

Председатель методкомиссии радиофизического факультета,
Доцент Г.М. Дейкова

Пособие написано с целью показать студентам Радиофизического Факультета широкие возможности и привлечь внимание к системе компьютерной математики MATLAB.

MATLAB – это система автоматизации математических расчётов, сочетающая в себе реализацию современных численных методов, возможности визуализации и программирования с достижениями в области математики за всю историю человечества.

Авторы не стремились написать руководство или, тем более, самоучитель, а пытались лишь дать импульс посредством данного пособия для дальнейшего глубокого изучения системы MATLAB.

Авторы: **Иванов Антон Валериевич** – студент 731 группы РФФ ТГУ

Клоков Андрей Владимирович – студент 701 группы РФФ ТГУ

Якубов Владимир Петрович – зав. кафедрой радиофизики РФФ ТГУ

Введение 4
1. Матричная лаборатория MATLAB и выгода от её использования 6
2. Запуск и интерфейс MATLAB 9
2.1. Запуск системы MATLAB 9
2.2. Характеристика пользовательского интерфейса 9
3. Алгебра матриц на MATLAB 13
3.1. Задание и операции над матрицами в MATLAB 13
3.2. Типы переменных. 19
3.3. Формат переменных. 19
4. Лёгкость построения 2-х и 3-хмерных графиков 21
4.1. Функции одной переменной 21
4.2. Окна изображений 25
4.3. Функции двух переменных 25
4.4. Анимационная графика 29
4.5. Печать графики 30
5. Как работать с файлами? 31
6. Основы программирования 34
6.1. Структуры управления потоками 34
6.2. M – файлы – функции 37
Заключение 41
Литература 41

Введение

Методическое пособие, предлагаемое Вашему вниманию, отличается от большинства известных прежде всего тем, что оно малообъемное и написано для студентов самими студентами 2-го и 5-го курсов.

Использование многотрудного собственного опыта способствовало выбору оптимального способа построения материала, обеспечивающего максимально доступный и краткий способ начала практического использования мощного современного инструмента физика – системы MATLAB.

Известные книги, посвященные MATLAB, конечно же, более фундаментальны и универсальны. Авторы настоятельно рекомендуют читателям обратиться к ним для углубления знаний. Как показал собственный опыт авторов, для первого шага в систему MATLAB нужно краткое, но ёмкое введение-пособие.

ЦЕЛЬЮ предлагаемой работы ставится максимально сократить первый шаг Вам – студентам к системе MATLAB и тем самым сделать её Вашим рабочим инструментом.

В настоящее время система MATLAB является настольным инструментом большинства зарубежных ведущих учебных и научных центров. Этому способствует широкая система дополнений и расширений системы MATLAB, позволяющая легко интегрировать её с другими средствами, например, с системой графического программирования LabView.

На радиофизическом факультете ТГУ система MATLAB пока не входит в перечень обязательных для изучения компьютерных систем. Пособие будет способствовать самостоятельному изучению MATLAB.

Авторы надеются, что это, прежде всего, облегчит процесс выполнения курсовых и дипломных работ студентами при

моделировании и анализе процессов и, кроме того, станет хорошим подспорьем в дальнейшей работе.

Весь материал разбит на 6 разделов, из которых Вы узнаете:

1. Краткую историю и цель создания системы MATLAB, какое место занимает MATLAB среди других математических компьютерных систем, а также о предоставляемых возможностях высокопродуктивных исследований, моделирования физических процессов, разработки и анализа данных.

2. О том, как запустить систему MATLAB, и об общей характеристике пользовательского интерфейса.

3. Как работать с матрицей – одним из основных объектов MATLAB.

4. Что одно из больших достоинств MATLAB – это лёгкость построения и обилие средств машинной графики функций одной и многих переменных.

5. Как сохранять полученные данные или считывать их из файлов. Как показывает практика, эта операция представляет основную трудность для начинающих изучение MATLAB.

6. Что программирование в системе MATLAB, включая логические переходы, циклы и функции, – это достаточно простое и эффективное средство её адаптации к решению практически неограниченного класса задач.

Методическое пособие предназначено, прежде всего, для студентов - радиофизиков, но будет полезно для студентов других специальностей, а также всех решившихся сделать первый шаг к современному вычислительному инструменту – системе MATLAB.

1. Матричная лаборатория MATLAB и выгода от её использования

Одной из первостепенных целей создания системы MATLAB было предоставление пользователям языка программирования, ориентированного на математические расчёты и способного превзойти традиционные языки программирования как по скорости вычислений и реализации численных методов, так и по адаптации системы к решению многочисленных и разнообразных задач.

Система MATLAB была разработана Молером (C.B.Moler) и с конца 70-х годов широко использовалась на ЭВМ. В начале 80-х годов Дж. Литл (John Little) из фирмы MathWork, Inc. разработал версию системы PC MATLAB для компьютеров класса IBM PC, VAX и Macintosh.

Возможности MATLAB весьма обширны, а по скорости выполнения задач система превосходит своих конкурентов. Система применима для вычислений в любой области науки и техники и широко используется при математическом моделировании физических устройств и систем, относящихся к механике, в частности, к динамике, гидродинамике и аэродинамике, акустике и т.д.

Этому способствует не только расширенный набор матричных и иных операций и функций, но и наличие расширения *Simulink*, предназначенного для решения задач блочного моделирования динамических систем и устройств, а также десятков других расширений.

Типичное использование системы MATLAB - это:

- математические расчёты и вычисления,
- создание алгоритмов,
- моделирование,

- анализ данных, их исследование и визуализация,
- научная и инженерная графика,
- разработка приложений, включая создание графического интерфейса.

Важными достоинствами системы являются её *открытость* и *расширяемость*. Большинство команд и функций системы реализованы в виде текстовых m – файлов (с расширением .m) и файлов на языке *C*, причем все файлы доступны для модификации.

Пользователю дана возможность создавать не только отдельные файлы, но и библиотеки файлов для реализации специфических задач.

Система MATLAB выполняет сложные и трудоёмкие операции над векторами и матрицами в режиме прямых вычислений без какого-либо программирования.

Поэтому системой можно пользоваться как мощнейшим калькулятором, в котором наряду с обычными арифметическими и алгебраическими действиями могут использоваться такие сложные операции как инвертирование матрицы, вычисление её собственных значений, решение системы линейных уравнений, вывод графиков функций одной и двух переменных и многое-многое другой.

Широкому распространению MATLAB способствуют её пакеты расширения. Пакеты расширения – это основные средства профессионального расширения системы и её адаптации под решение определённых классов математических и научно-технических задач. Среди этих расширений следует назвать:

Simulink.

Этот пакет служит для имитационного создания моделей, состоящих из графических блоков с заданными свойствами. *Simulink* составляет и решает уравнения состояния модели и позволяет

подключать в нужные точки разнообразные виртуальные измерительные приборы. Действительно поражает наглядность представления результатов моделирования.

Real Time Windows Target и ***WorkShop***.

Это мощная подсистема имитационного моделирования в реальном масштабе времени, средство управления реальными объектами и системами. Достоинством такого моделирования является его математическая и физическая наглядность.

В MATLAB важная роль отводится специализированным группам программ, называемых *toolboxes*. *Toolboxes* – это всесторонняя коллекция функций MATLAB, которые позволяют решать частные классы задач. *Toolboxes* применяют для обработки сигналов, систем контроля, нейронных сетей, нечёткой логики, вэйвлетов, моделирования и т.д. Здесь следует назвать:

Statistics Toolbox.

Этот пакет прикладных программ по описательной статистике, распределению вероятностей, проверке гипотез, статистическому управлению процессами, статистической графики и т.д.

Spline Toolbox.

Пакет программ для работы со сплайнами, позволяющий выполнять сплайн-интерполяцию и аппроксимацию, а также производить такие операции над сплайнами как вычисление производной и интеграла.

Partial Differential Equations Toolbox.

Весьма важный пакет для решения систем дифференциальных уравнений в частных производных. Даёт эффективные средства для решения таких систем уравнений.

2. Запуск и интерфейс MATLAB

2.1. Запуск системы MATLAB

Запустить систему MATLAB очень просто, для этого на PC и Mac дважды щёлкните на иконку MATLAB. Для запуска в системе UNIX напишите matlab в строке операционной системы.

2.2. Характеристика пользовательского интерфейса

Система MATLAB (в данном пособии используется версия MATLAB 6.5) работает в командном интерактивном режиме, так как такой режим является одним из наиболее удобных для работы с системой.

Имеются и типовые средства Windows – меню и панель инструментов. Но они выглядят намного скромнее, чем у большинства современных приложений Windows.

Скорее всего, так и должно быть, так как, чем серьёзнее математическая система, тем меньше она нуждается в использовании всевозможных кнопок на панели инструментов и тем скромнее может быть её главное меню.

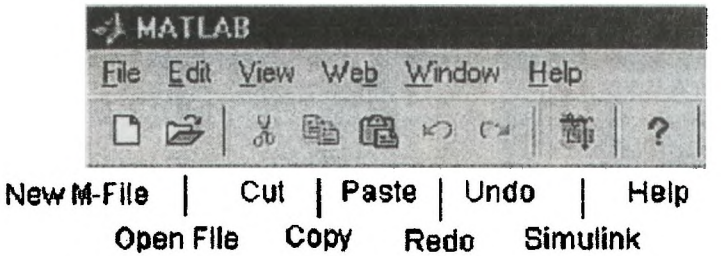
Командное окно

При запуске системы MATLAB открываются три окна: рабочая область (Workspace), окно истории команд (History Window) и командное окно (Command Window).

В рабочей области находится информация об используемых переменных. В командном окне возможен набор функций и команд (без использования m – файлов), которые выполняют прямые расчёты и действия.

Панель инструментов

На рисунке приведена часть окна системы MATLAB, содержащая меню и панель инструментов:



Перечислим функции и назначение всех кнопок панели инструментов:

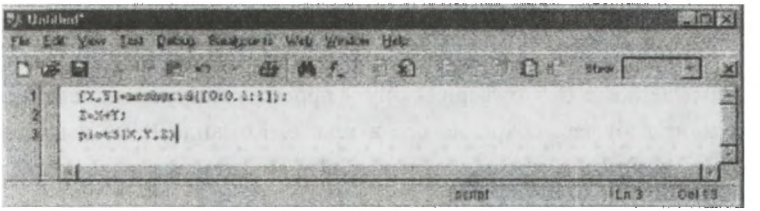
- New M-File – выводит пустое окно редактора m – файлов;
- Open File – открывает окно для загрузки m – файлов;
- Cut – вырезает выделенный фрагмент и помещает его в буфер;
- Copy – копирует выделенный фрагмент в буфер;
- Paste – переносит фрагмент из буфера в текущую строку ввода;
- Undo – отменяет предшествующую операцию;
- Redo – возвращает отменённую операцию;
- Simulink – создаёт пустое окно модели Simulink;
- Help – открывает окно справки.

Набор кнопок панели инструментов обеспечивает выполнение наиболее необходимых команд и вполне достаточен для повседневной работы с системой.

О назначении кнопок говорят и всплывающие подсказки, которые появляются при наведении курсора мыши на соответствующую кнопку.

Окно открытия нового файла

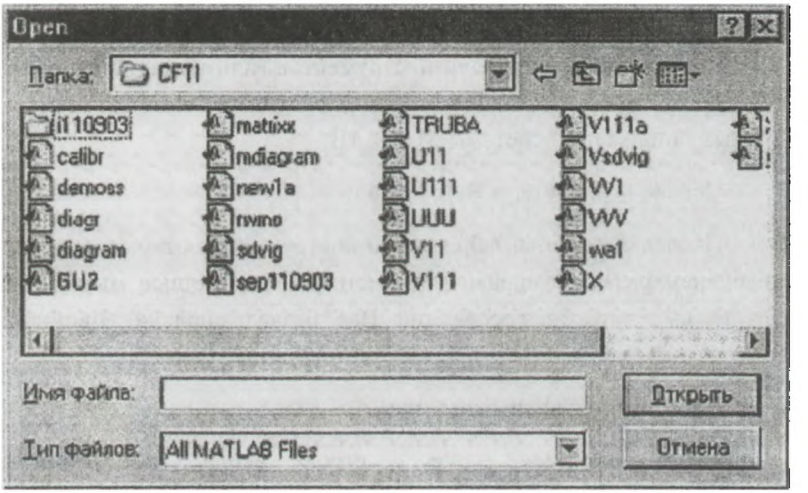
Кнопка New M-File открывает окно редактора m – файлов. Это окно показано на рисунке:



По умолчанию файлу даётся имя Untitled (безымянный), которое затем, при сохранении файла можно изменить.

Окно загрузки имеющегося файла

Кнопка Open File служит для загрузки в редактор ранее созданных m – файлов. Кнопка открывает окно, которое является типичным элементом интерфейса Windows-приложений:



После загрузки файла из документов его текст появляется в текущем окне. Его можно начинать редактировать и использовать.

Операции с буфером обмена

Кнопки Cut, Copy и Paste реализуют наиболее характерные команды над буфером обмена. Первые две операции относятся к выделенным фрагментам текста m – файлов.

Команда Cut осуществляет вырезание выделенного фрагмента и размещение его в буфере. Copy - просто копирует выделенный фрагмент в буфер, сохраняя его в тексте. Команда Paste вызывает объект из буфера, сохраняя его в нём, и помещает на место в документе, указанное текстовым курсором.

Отмена и возврат предшествующей операции.

Часто, при выполнении какого-либо действия, может оказаться, что та или иная операция оказалась ошибочной. При работе в MATLAB такую операцию можно отменить посредством нажатия кнопки Undo. Но, если выйдет, что отменённую операцию всё-таки надо вернуть, тогда необходимо нажать кнопку Redo.

Запуск приложения Simulink

Кнопка Simulink панели инструментов (или команда simulink из строки ввода в командном окне) запускает одно из самых мощных приложений системы MATLAB.

Справочная система Help Window

Последняя кнопка help панели инструментов открывает окно с перечнем разделов справочной системы. С помощью мыши Вы можете выделить интересующий Вас раздел справки. Двойной щелчок открывает окно справки соответствующего раздела.

Интерфейс графического окна

Набрав команду figure в строке ввода, Вы откроете графическое окно.

В данное окно графики выводятся графические команды и изображаются графики функций.

3. Алгебра матриц на MATLAB

MATLAB – это матричная лаборатория, и все операции в этой системе носят матричный характер. Поэтому, для правильного понимания принципов работы с MATLAB, сначала необходимо будет научиться задавать матричные переменные и выполнять над ними простейшие операции.

Напомним, что матрицей называется двумерный массив чисел или математических выражений, а вектором – одномерный массив. Одной из основных характеристик матрицы является её размерность, т.е. количество элементов в строках и столбцах матрицы. Следует обратить внимание на матрицу размерностью 1x1, так как такие матрицы в MATLAB являются обычными числами, скалярами.

3.1. Задание и операции над матрицами в MATLAB

Пусть нам даны две матрицы:

$$A = \begin{bmatrix} 1 & 8 & 9 & 16 \\ 2 & 7 & 10 & 15 \\ 3 & 6 & 11 & 14 \\ 4 & 5 & 12 & 13 \end{bmatrix} \text{ и } B = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{bmatrix}$$

Сперва, введём наши матрицы в память системы. Для этого в командном окне необходимо набрать:

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

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

Как видно, для задания матриц используются квадратные скобки, элементы матрицы вводятся построчно, причём каждая строка отделяется от другой точкой с запятой.

В итоге, будут созданы две переменные A и B, в которых и

будут записаны значения матриц. В командном окне MATLAB отобразит матрицу:

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

В то же время, матрица *B* не отобразится. Не трудно увидеть, что в конце строки ввода матрицы *B* стоит знак «;», а у матрицы *A* - нет. Так, если Вам нужно только ввести переменную, а отображать её значение не надо, то необходимо поставить символ «;» в конце ввода переменной, тогда MATLAB запомнит переменную, но не покажет её.

Теперь выполним стандартные операции над матрицами.

Давайте найдём сумму этих матриц *A* и *B*:

A + B

Получим результат:

```
ans =
    17    10    12    29
     7    18    20    23
    12    13    17    26
     8    19    27    14
```

Новая матрица, являющаяся суммой предыдущих двух, названа *ans*. Дело в том, что, когда выходная переменная не определена, MATLAB использует переменную *ans* для хранения результатов вычисления. Если необходимо поместить результат операции в конкретную переменную, то необходимо набрать:

C = A + B

Тогда результат сложения будет сохранён в переменной *C*.

Теперь поработаем с матрицей B. Найдём сумму элементов в каждом столбце, для чего используем функцию:

sum(B)

MATLAB выдаст ответ:

```
ans =
  34   34   34   34
```

MATLAB предпочитает работать со столбцами матрицы, таким образом, для нахождения суммы не в столбцах, а в строках, лучшим способом является транспонировать нашу матрицу, подсчитать сумму в столбцах транспонированной матрицы, а потом транспонировать результат. Напомним, что транспонированная матрица – это матрица, столбцами которой являются строки исходной матрицы. Операция транспонирования обозначается апострофом:

B'

Результат транспонирования выглядит так:

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

Следующее выражение считает сумму элементов в каждой строке транспонированной матрицы:

sum(B')'

```
ans =
  34
  34
  34
  34
```

Теперь найдём сумму элементов главной и побочной диагоналей матрицы B.

Функция **diag(B)** выбирает элементы матрицы, стоящие на главной диагонали.

sum(diag B) - эта функция считает сумму элементов главной диагонали.

```
ans =
   34
```

А для побочной или антидиагонали в MATLAB нет специальной функции, поэтому мы сначала получим зеркальную матрицу, при помощи функции *fliplr*:

fliplr(B)

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

sum(diag(fliplr(B)))

```
ans =
   34
```

Видно, что сумма элементов в столбцах, строках, главной и побочной диагоналях есть одно и тоже число. Матрицы, обладающими такими свойствами, называется магическими квадратами.

Для магических квадратов в MATLAB есть специальная функция *magic(n)*, где n – это размерность магического квадрата. Так матрицу B можно было задать по-другому:

B = magic(4);

Вообще, над матрицами выполнимы следующие операции:

«+» - сложение, «-» - вычитание, «*» - умножение, «.*» - поэлементное умножение, «./» - поэлементное деление и «.^» - поэлементное возведение в степень.

Очень важно сразу понять разницу между умножением и поэлементным умножением матриц.

Например:

A*B

```
ans =
  201  377  377  201
  217  361  361  217
  233  345  345  233
  249  329  329  249
```

A.*B

```
ans =
   16   16   27  208
   10   77  100  120
   27   42   66  168
   16   70  180   13
```

Итак, очевидно, что эти две операции дают различные результаты. Операция «*» определяется как произведение матриц, «.*» даёт другой тип произведения, а именно произведение соответствующих элементов матриц. Это важный момент, показывающий, что необходимо учитывать матричный характер операций в MATLAB, для того чтобы получать корректные результаты.

Далее, приведём список самых распространенных операций

над матрицами:

- **rref(C)** - команда, приводящая матрицу к ступенчатому виду;
- **det(C)** - функция подсчёта определителя матрицы;
- **eig(C)** - нахождение собственных значений матрицы.

C = [1 2 3 4; 5 6 7 8; 10 9 8 7; 11 12 12 15]

```
C =
    1    2    3    4
    5    6    7    8
   10    9    8    7
   11   12   12   15
```

rref(C)

```
ans =
    1    0    0   -3
    0    1    0    5
    0    0    1   -1
    0    0    0    0
```

det(C)

```
ans =
   65
```

eig(C)

```
ans =

   31.1760
   -2.5068
    1.3307
    0.0000
```

Сделаем некоторые замечания.

Во-первых, для обращения к конкретному элементу матрицы, а не к матрице в целом, необходимо указывать индекс элемента C(n,m), где n – номер строки, в которой расположен элемент, m – номер столбца.

```
C(1,1)+C(2,2)

    ans =
       7
```

Во-вторых, отметим оператор «:». Двоеточие, «:», - один из важнейших операторов. Он проявляется в различных формах. Так выражение 1:10 – это вектор-строка, содержащий целые числа от 1 до 10. Или, например, если необходимо получить последовательность от 100 до 0, уменьшая последующие члены на 5, тогда последовательность можно задать так: 100:-5:0, где 100 – начальное значение, 0 – конечное, а -5 – шаг.

Далее перейдём к характеристике типов и форматов переменных.

3.2. Типы переменных.

В MATLAB нет необходимости объявлять тип переменных. Когда MATLAB встречает новое имя переменной, он автоматически создаёт переменную и выделяет соответствующий объём памяти. Если переменная уже существует, MATLAB изменяет её состав и, если это необходимо, выделяет дополнительную память.

Имена переменных состоят из букв, цифр или символов подчеркивания. MATLAB использует только первые 31 символ имени переменной. MATLAB чувствителен к регистрам, он различает заглавные и строчные буквы, так переменные A и a – это разные переменные.

3.3. Формат переменных.

По умолчанию MATLAB выдаёт числовые результаты в

форме с четырьмя цифрами после десятичной точки. При работе с числовыми данными можно задавать различные форматы представления чисел, что не влияет на тип или точность представления переменной. Для установки формата представления чисел используется команда *format name*, где *name* – это имя формата. Имена формата бывают следующие:

- short – короткое представление в фиксированном формате (5 знаков);
- short e – короткое представление в экспоненциальной форме (5 знаков мантиссы и 3 порядка);
- long – длинное представление в фиксированном формате (15 знаков)
- long e – длинное представление в экспоненциальной форме (15 знаков – мантисса, 3 - порядок);
- hex – представление числа в шестнадцатеричной форме;
- bank – представление для денежных единиц;
- rational – в виде рациональной дроби.

Например, число 100/3

100/3	ans = 33.3333
format short e	ans = 3.3333e+001
format long	ans = 33.33333333333334
format long e	ans = 3.333333333333334e+001
format hex	ans = 4040aaaaaaaaaab
format bank	ans = 33.33
format rational	ans = 100/3

4. Лёгкость построения 2-х и 3-хмерных графиков

При работе с любыми данными необходимо не только уметь вводить их в память компьютера и выполнять над ними многочисленные операции, но так же уметь представлять полученные данные в наглядной форме.

Одним из наиболее наглядных способов представления данных является график зависимости одной переменной от другой переменной (или переменных). MATLAB имеет обширные возможности визуального представления функций, что делает графику в MATLAB одной из лучших среди математических пакетов.

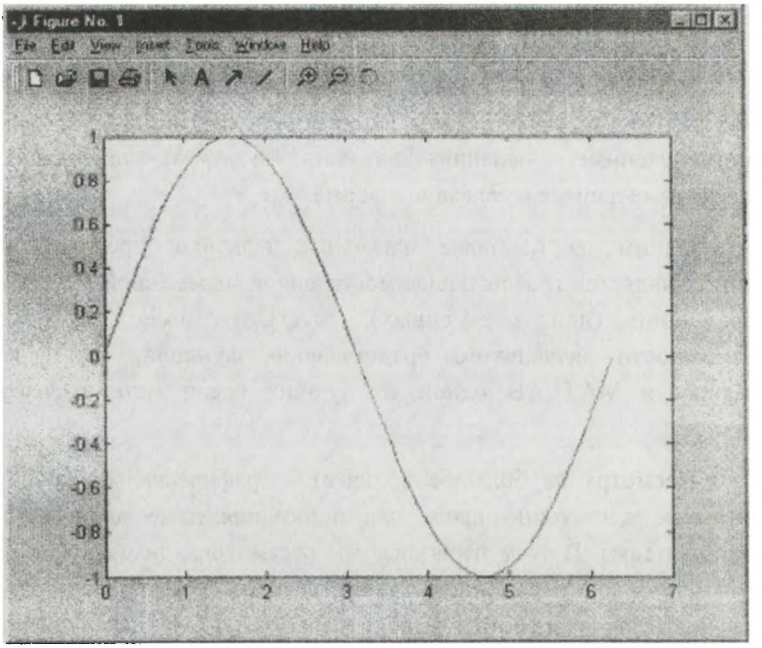
Несмотря на большое количество графических команд, их синтаксис достаточно прост для понимания даже начинающим пользователям. В этом параграфе мы рассмотрим сначала графику функции одной переменной, затем трёхмерную и анимационную графику, а так же научимся выводить полученные данные на печать.

4.1. Функции одной переменной

Для начала рассмотрим самый простой пример – построение графика синусоиды. С самого начала следует обратить внимание на то, что MATLAB строит графики функций по ряду из точек, соединяя их отрезками прямых. Зададим интервал области задания функции от 0 до 2π с шагом $\pi/100$. Зададим вектор аргумента $x = 0 : \pi/100 : 2\pi$, а затем используем функцию *plot* для построения графика:

```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
```

Всего три строчки и график готов:



Функция *plot* имеет параметры, благодаря которым возможно управлять цветом графика, стилем линий и маркера следующим образом:

plot(x,y,'цвет_стиль_маркер'), где *x* – независимая переменная, *y* – зависимая, а «цвет_стиль_маркер» - это 1-,2-,3-символьная строка, составленная из типов цвета, стиля линий и маркеров (в нашем примере эта символьная строка отсутствовала):

- Символы цвета: ‘Y’, ‘M’, ‘C’, ‘R’, ‘G’, ‘B’, ‘W’, ‘K’ обозначающие соответственно: желтый, фиолетовый, голубой, красный, зеленый, синий, белый, чёрный.
- Типы линий: ‘-’, ‘:’, ‘-.’, ‘--’ для: сплошной, двойной пунктирной, штрих-пунктирной, штриховой линий.
- Типы маркера: ‘.’, ‘O’, ‘X’, ‘+’, ‘*’, ‘S’, ‘D’, ‘V’, ‘^’, ‘<’, ‘>’,

'P', 'H' для: точки, окружности, креста, плюса, звёздочки, квадрата, ромба, треугольника (вниз), треугольника (вверх), треугольника (влево), треугольника (вправо), пяти- и шестиугольника.

Построим, например, два графика синусоиды, находящихся в противофазе, так чтобы один из графиков был зеленый, пунктирный с маркером «+», а другой – чёрный, сплошной, с маркером «*»:

```
x = 0:pi/10:2*pi;
y = sin(x);
y2 = sin(x-pi);
plot(x,y,'G:+',x,y2,'K-*')
```

При этом добавим на график сетку координат с помощью команды:

```
grid on
```

Команда **grid off** наоборот отключает сетку. Также мы подпишем оси, заголовок и вставим поясняющий текст.

Обозначим ось *Ox*:

```
xlabel(' 0 \leq \itx \leq 2*\pi')
```

Затем подпишем ось *Oy*:

```
ylabel(' sin(x), sin(x-\pi) ')
```

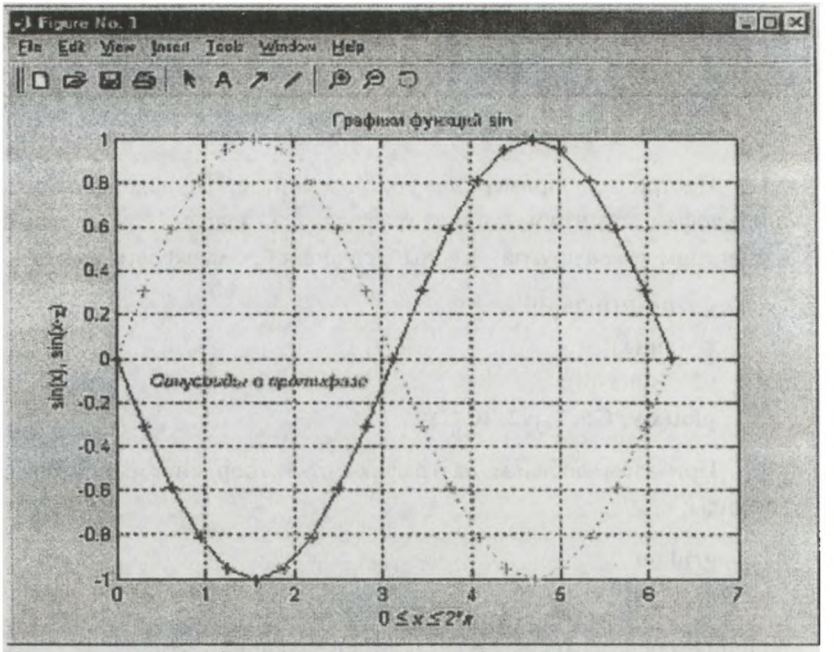
Обозначим заголовок:

```
title('Графики функций sin')
```

Вставим текст с первоначальным положением текстового маркера в точке (1/3,–0.1):

```
text(1/3, -0.1, ' \it{Синусоиды в противофазе}')
```

В итоге получим (рисунок чёрно-белый, хотя на самом деле он цветной):



Отметим также функцию *axis*, эта функция имеет несколько возможностей для настройки масштаба, ориентации и коэффициента сжатия. Функция:

axis([xmin xmax ymin ymax])

позволит Вам выбирать соответствующий масштаб по осям, т.е. минимальные и максимальные значения по соответствующим осям;

axis squre – сделает оси равной длины;

axis equal – сделает отметки приращений осей одинаковой длины;

axis on – включает обозначение осей и метки промежуточных делений;

axis off – выключает обозначение осей и метки промежуточных делений.

4.2. Окна изображений

Функция *plot* автоматически открывает новое окно изображений, если до этого его не было на экране. Если оно уже существует, то *plot* использует его по умолчанию. Для открытия нового окна изображений и выбора его по умолчанию, наберите:

figure

При двух и более открытых окнах необходимо выбрать то, которое будет текущим, т.е. в которое будут выводиться результаты построения графиков. Следующая функция позволяет это сделать:

figure(n), где n – это номер в заголовке окна. В этом случае результаты всех последующих команд будут выводиться в это окно.

А команда *hold* позволяет добавлять кривые на уже существующий график. Когда вы наберёте:

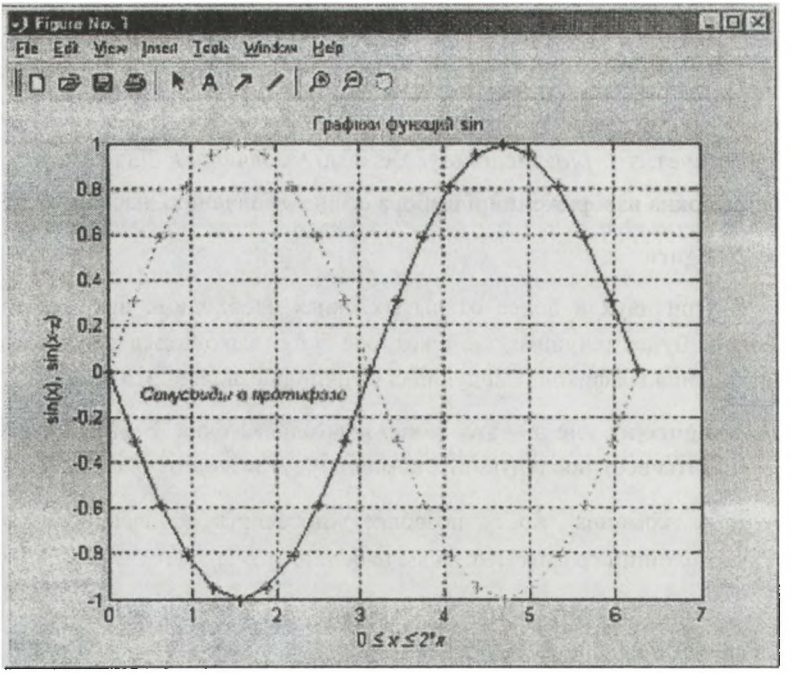
hold on MATLAB не стирает существующий график, а добавляет в него новые данные, изменяя оси, если это необходимо.

4.3. Функции двух переменных

Одной из самых замечательных возможностей системы MATLAB является то, что графики функций двух переменных строятся столь же просто, как и функций одной переменной. Например, построим поверхность, задаваемую функцией двух переменных: $z = \sin x + \sin y + \sin(x - y)$. Анализ и построение такой функции заняли бы продолжительное время, а в MATLAB всё намного проще:

```
[X,Y] = meshgrid([-3:0.15:3]);
Z = sin(X)+sin(Y)+sin(X-Y);
plot3(X,Y,Z)
grid on
```

Сразу же получаем:



Команда *plot*3 является аналогом команды *plot*, но относится к функциям двух переменных. Также как и функция *plot* может содержать параметры стиля линий, точек и маркера, так и при помощи *plot*3 возможно управлять стилем графика.

Наиболее представительными и наглядными являются сетчатые графики с заданной или функциональной окраской. Построение таких графиков достигается командами:

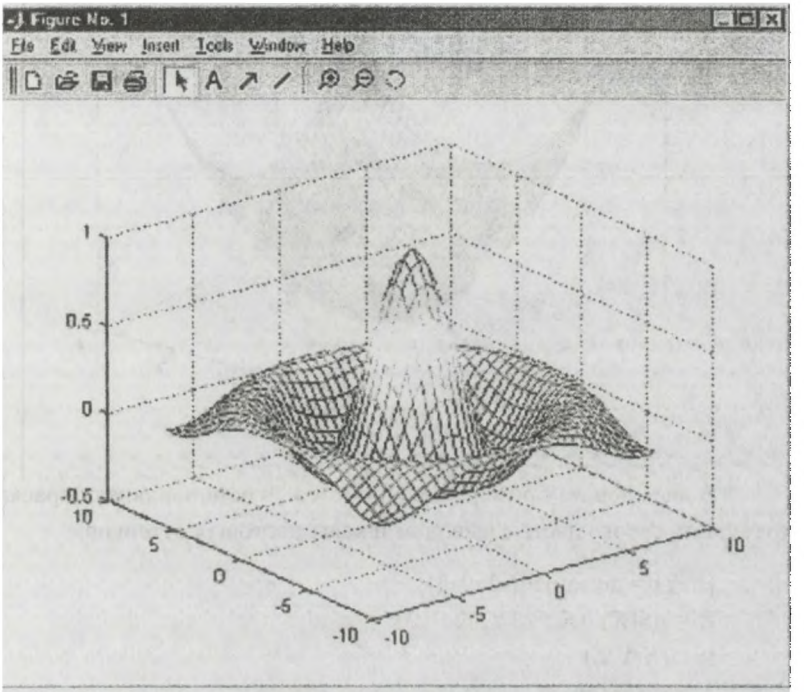
mesh(X, Y, Z, C) - выводит в графическое окно сетчатую поверхность с цветами узлов поверхности, заданным массивом C.

mesh(X, Y, Z) - аналог предыдущей команды при $C = Z$. В данном случае используется функциональная окраска, при которой цвет задаётся высотой поверхности.

Приведём пример применения команды *mesh* для функции $z = \sin\left(\sqrt{x^2 + y^2}\right) \Big/ \sqrt{x^2 + y^2}$:

```
[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2+Y.^2)+eps;
Z = sin(R)./R;
mesh(X,Y,Z)
```

В итоге имеем:

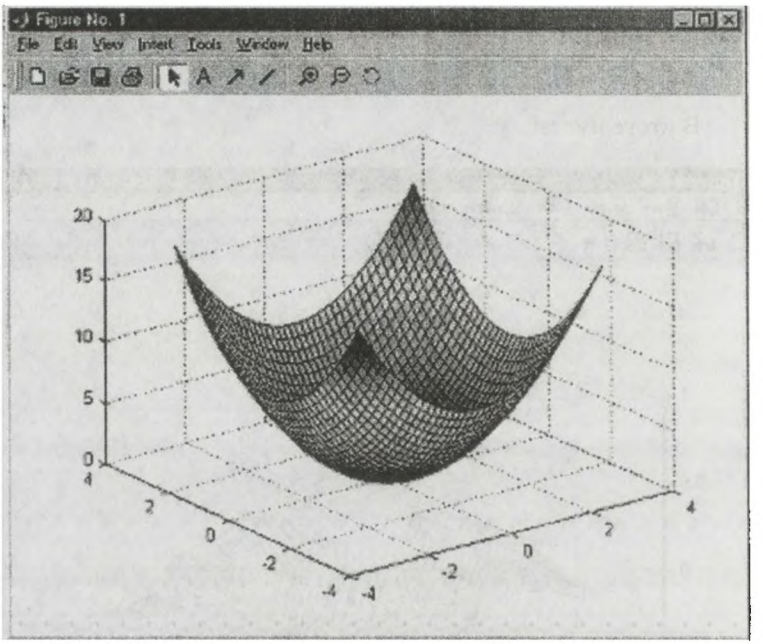


Особенно наглядное представление имеют графики, использующие функциональную закраску ячеек. Для построения таких поверхностей используется команда *surf* .

Ниже приведён простой пример построения параболоида:

```
[X,Y] = meshgrid(-3:.15:3);
Z = X.^2+Y.^2;
surf(X,Y,Z)
```

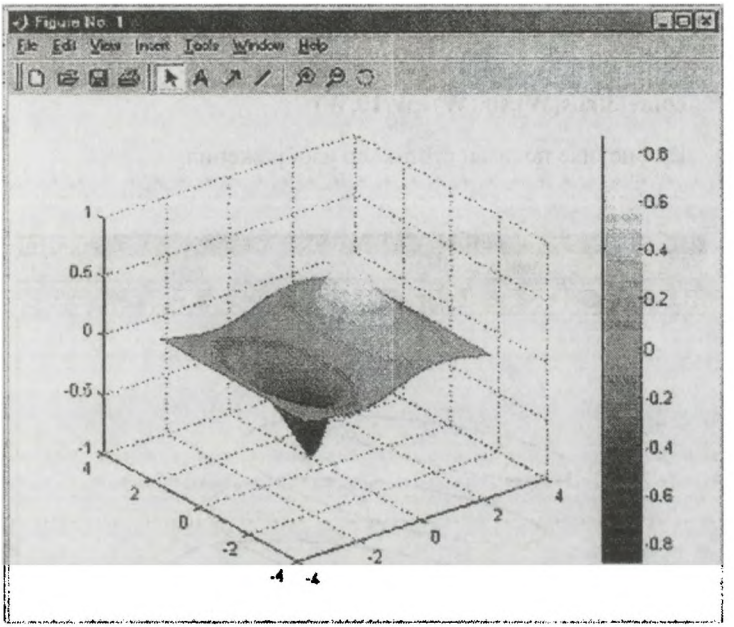
MATLAB построит поверхность параболоида:



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

```
[X,Y] = meshgrid(-3:.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*) задаёт окраску тонами серого цвета, а команда *shading interp* обеспечивает устранение изображения сетки и задаёт интерполяцию для оттенков объемной поверхности. На рисунке показан вид графика, построенного в этом примере:



4.4. Анимационная графика

Для отображения движения точки по траектории используется команда *comet*3. При этом движущаяся точка напоминает ядро кометы с хвостом. Используются следующие формы представления этой команды:

*comet*3(Z) отображает движение точки с цветным «хвостом» по трёхмерной кривой, определённой массивом Z.

*comet*3(X, Y, Z) отображает движение «кометы» по кривой в

пространстве, заданной соответствующими точками.

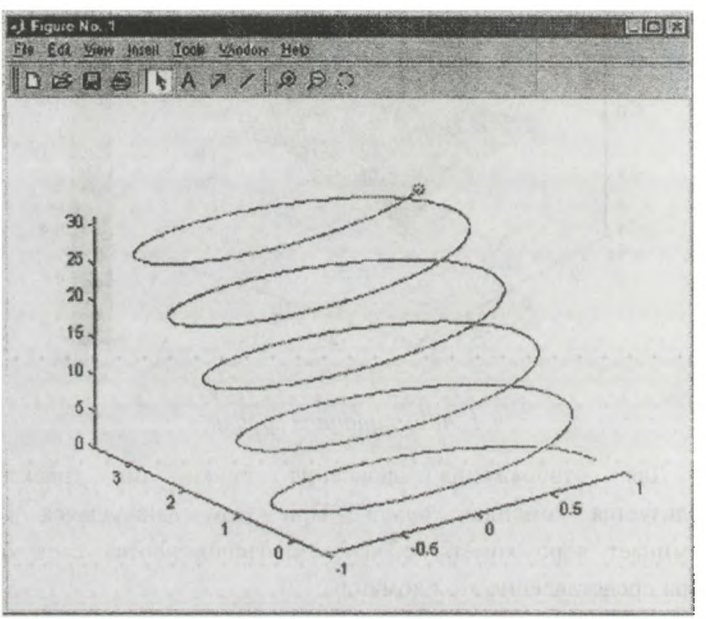
*comet*3(X, Y, Z, p) аналогична предыдущей команде с заданием длины «хвоста» «кометы» как $p * length(Z)$. По умолчанию параметр p равен 0.1.

Ниже представлен пример применения команды *comet*3 :

W = 0:pi/500:10*pi;

comet3(cos(W),sin(W)+W/10,W)

На рисунке показан стоп-кадр изображения:



4.5. Печать графики

Опция Print в меню File и команда *print* печатают графику MATLAB. Меню Print вызывает диалоговое окно, которое позволяет выбирать общие стандартные варианты печати.

5. Как работать с файлами?

В этом параграфе мы познакомимся с очень важной операцией в MATLAB – созданием и использованием m – файлов. M – файлы представляют собой текстовые файлы, содержание код MATLAB.

Как было отмечено ранее, для работы с m – файлами в MATLAB предусмотрен специальный редактор. Откроем его нажатием кнопки New M-File на панели инструментов.

Наберём в теле документа следующую программу:

```
A = [1 3 5; 3 5 7; 5 8 9];
B = rref(A)
C = det(A)
D = inv(A)
```

Затем, сохраним наш файл под именем My_M_File, для чего необходимо выбрать **Save** из меню **File**.

После того как мы создали m – файл My_M_File.m, прочитаем его через командное окно в MATLAB. Набёрем в командной строке название файла (без указания расширения) – My_M_File. В итоге, MATLAB прочитает файл и создаст четыре переменные: A – исходная матрица, B – матрица, приведённая к ступенчатому виду, C – определитель матрицы A и D – матрица, обратная A. Причём последние три переменные будут отображены в командном окне.

Прежде чем приступать к более сложному примеру, необходимо сделать важное замечание. Ещё до попытки прочитать тот или иной m – файл следует убедиться в том, что текущей директорией MATLAB выбрана именно та директория, в которой записан необходимый m – файл.

Поле изменения текущей директории находится на панели

инструментов правее функциональных кнопок:



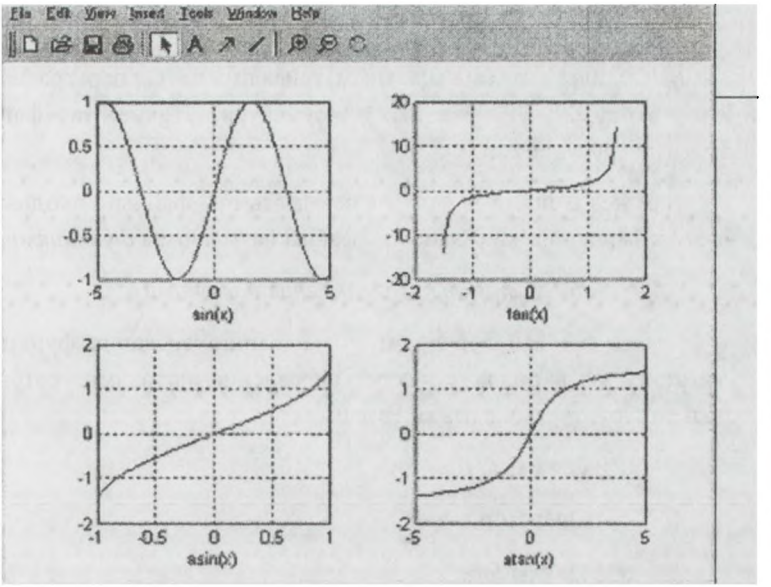
Далее, создадим m – файл TrigFun.m, в котором запрограммируем построение графиков синуса, тангенса, арксинуса и арктангенса:

```
%Графики тригонометрических функций
x = -5:0.01:5;
subplot(2,2,1)
plot(x,sin(x)); grid on; xlabel('sin(x)');
x = -1.5:0.01:1.5;
subplot(2,2,2)
plot(x,tan(x)); grid on; xlabel('tan(x)');
x = -1:0.01:1;
subplot(2,2,3)
plot(x,asin(x)); grid on; xlabel('asin(x)');
x = -5:0.01:5;
subplot(2,2,4)
plot(x,atan(x)); grid on; xlabel('atan(x)');
```

Команда *subplot(m,n,p)* разбивает графическое окно на *m* подокон по горизонтали, и *n* подокон по вертикали. Причём параметр *p* – это номер подокна, в которое будет выводиться текущий график (подокна отсчитываются последовательно по строкам).

Запустим m – файл TrigFun.m, набрав в командной строке TrigFun. Вы увидите, что MATLAB построит четыре графика тригонометрических функций в графическом окне:

5. К



Теперь наберём в командной строке:

help TrigFun

В командном окне появится поясняющий текст:

Графики тригонометрических функций

Этот же текст был набран в самом начале m – файла после символа «%». Данный текст – это комментарий к файлу, и он должен идти именно после символа «%» (если комментарии находятся на нескольких строках, то «%» необходимо ставить в начале каждой строки). Комментарии не влияют на выполнение файла, их вообще может не быть в документе. Но хорошим стилем программирования считается наличие подробных комментариев, т. к. даже в небольших файлах бывает трудно проследить логику и методы реализации той или иной задачи.

6. Основы программирования

M – файлы, которые мы рассматривали в пятом параграфе, не имели входных и выходных аргументов. Такие m – файлы называются *сценариями*.

Нам необходимо научиться создавать m – файлы с входными и выходными данными. Такие m – файлы называются *функциями*.

6.1. Структуры управления потоками

Перед тем как переходить к описанию создания функций, рассмотрим пять видов структур управления потоками, которые схожи со структурами в языке Pascal:

- Оператор *if*
- Оператор *switch*
- Циклы *for*
- Циклы *while*
- Оператор *break*

Оператор if

Оператор *if* вычисляет логическое выражение и выполняет один блок операторов, если условие истинно, и другой блок, если ложно. Необязательные команды *elseif* и *else* служат для выполнения альтернативных групп операторов.

Ключевое слово *end* завершает последнюю группу операторов.

Таким образом, блоки операторов располагаются между четырьмя ключевыми словами, без использования фигурных и обычных скобок.

К примеру, сравним два скаляра A и B:

```
if A>B
```

```
disp('greater')
elseif A<B
  disp('less')
elseif A==B
  disp('equal')
else
  error('Непредвиденная ситуация')
end
```

Для скаляров одно из трёх условий выполнится обязательно и мы не попадём в непредвиденную ситуацию, но ситуация может измениться при сравнении, к примеру, пар матриц.

Операторы switch и case

Оператор *switch* выполняет группу операторов, базируясь на значении переменной или выражения. Ключевые слова *case* и *otherwise* разделяют эти группы.

Выполняется только первый соответствующий случай. Группу операторов необходимо завершать ключевым словом *end*.

Логика оператора приведена в примере. Пусть дальнейшие действия зависят от значения переменной A+B:

```
switch A+B
 case 0          % выполняемые действия при A+B=0
  M=A.^2
 case 1                    % при A+B=1
  M=B.^2
 case 2                    % при A+B=2
  M=A+B
 otherwise                 % действия в иных случаях
  error('Это невозможно')
end
```

Таким образом, значение переменной M зависит от значения

переменной A+B. Приведённую выше конструкцию можно было реализовать и с помощь оператора *if*, но применение оператора *switch* более логично.

Цикл for

Цикл *for* повторяет группу операторов определённое число раз. Завершает тело цикла слово *end*.

```
for n=2:2:10
  M = M*n;
end
```

Точка с запятой после выражения в теле цикла предотвращает повторение вывода результатов на экран. Цикл повторится пять раз.

Цикл while

Цикл *while* повторяет группу операторов определённое количество раз, пока выполняется логическое условие. Слово *end* очерчивает тело цикла.

Оператор break

Оператор *break* позволяет досрочно выходить из циклов *for* и *while*. Во вложенных циклах *break* осуществляет выход только из внутреннего цикла.

Рассмотрим пример с применением цикла *while*. Найдем корень уравнения $x^3 - 2x - 5 = 0$ на отрезке $[0,3]$, используя метод дихотомии, т.е. метод деления отрезка пополам. Создадим m – файл Dihot.m:

```
a = 0;          % задаём левую границу отрезка
b = 3;          % задаём правую границу отрезка
fa = Inf;       % fa присвоили значение плюс
                % бесконечности
fb = - Inf;     % fb присвоили значение минус
```

```
                % бесконечности
while b-a > eps*b  % выполняем цикл пока b-a>eps*b
x = (a+b)/2;       % делим отрезок пополам
fx = x^3-2*x-5;    % вычисляем значение функции

if sign(fx) == sign(fa)      % вычисляем знак функции
    a = x; fa = fx;          % вычисляем знак функции в
                             % левой границе отрезка
else
    b = x; fb = fx;
  end
end
x
```

MATLAB выдаст результат:

```
x  = 2.0946
```

6.2. M – файлы – функции

А теперь создадим m – файл – функцию. Путь необходимо исследовать на экстремум и нули функцию:

$$y = \frac{1}{(x-0.3)^2+0.01} + \frac{1}{(x-0.9)^2+0.4} - 6$$

на отрезке $[0, 1]$.

Для этого создадим два m – файла, в первом будет задана наша функция:

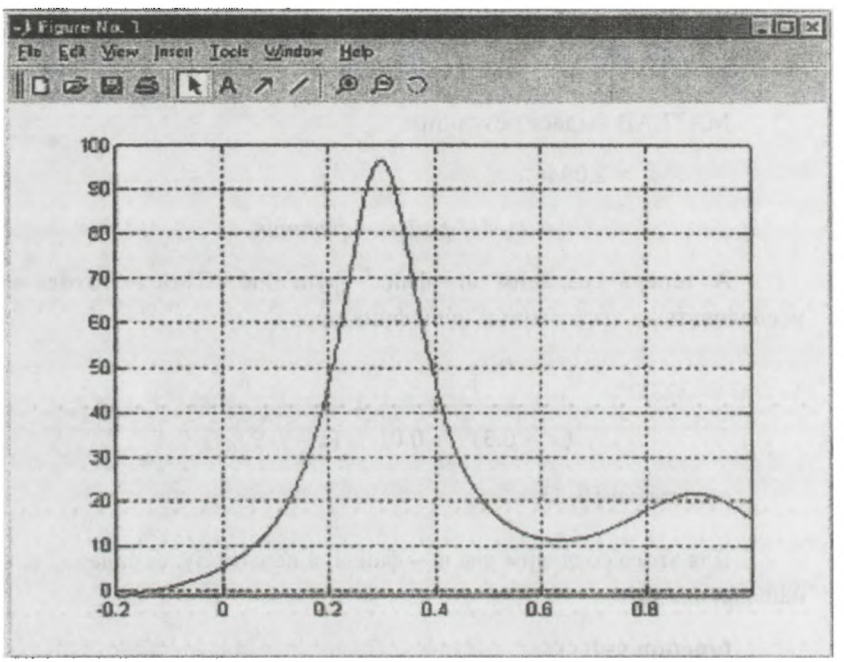
```
function y=f(x)
y=1./((x-.3).^2+.01)+1./((x-.9).^2+.04)-6;
```

Сохраним этот m – файл под именем f.m.

Теперь создадим второй файл, в котором реализуется построение и исследование функции:

```
x=-0.2:.002:1;
y=f(x);
plot(x,y)
grid on
axis ([-0.2 1 -1 100])
min=fmins('f',.5)
fmin=f(min)
zer=fzero('f',.5)
```

Сохраним его под именем my_func.m. После прочтения файла MATLAB выдаст результат:




```
min =
   0.6370
fmin =
```

```
    11.2528

zer =
   -0.1316
```

Поясним принципы построения функции и полученные результаты.

Функция *fmins* находит точки минимума функции. Первым из аргументов *fmins* является имя функции, минимум которого ищется, а вторым - приближённое положение минимума.

Функция fzero ищет нули функции.

Структура функции с одним выходным параметром выглядит следующим образом:

```
Function var = f_name(Список_параметров)
%Основной комментарий
%Дополнительный комментарий
Тело файла с любыми выражениями
var = выражение
```

M – файл – функция имеет следующие свойства:

- Он начинается с объявления function, после которого указывается имя переменной var – выходного параметра, имя самой функции и список её входных параметров;
- Функция возвращает своё значение и может использоваться в виде f_name(Список_параметров);
- Все переменные, имеющиеся в теле файла-функции, являются локальными, т.е. действуют только в пределах тела функции;

Файл-функция является самостоятельным программным модулем, который общается с другими модулями через свои

входные и выходные параметры;

- Файл-функция служит средством расширения системы MATLAB;

Необходимо помнить, что переменные в файлах-сценариях являются *глобальными*, а в файлах-функциях – *локальными*.

Хотя возможно использование глобальных переменных в файлах-функциях. Для этого её надо объявить как global var, где var – некоторая переменная. Определение global должно быть до самой переменной.

В заключение приведём пример обращения к функции, созданной пользователем.

Создадим файл-функцию fun.m:

```
function z=fun(x,y)
z=x^2+y^2;
```

Вызовем теперь функцию *fun* через командное окно:

```
>>y = fun(2,3)    %x = 2, y = 3
```

В итоге:

```
y =
   13
```

Заканчивая, отметим, что программирование требует особой точности и педантичности, именно поэтому для того, чтобы стать хорошим программистом, необходимо вырабатывать в себе эти качества.

Заключение

Дорогой читатель, открыв эту страницу, Вы, как надеются авторы, скорее всего, прочитали данное пособие. Мы можем Вас и себя с этим поздравить! Теперь Вы знакомы с мощной математической системой MATLAB. Если при этом MATLAB вызвал у Вас интерес, то авторы считают, что они достигли своей цели. Дальше Вы уже сможете самостоятельно изучать систему, для этого авторы рекомендуют приведенную в конце литературу.

Авторы методического пособия выражают свою благодарность рецензентам: доцентам радиофизического факультета Беличенко Виктору Петровичу и Жукову Андрею Александровичу.

Литература

1. Потемкин В.Г. Control System Toolbox. MATLAB 5 для студентов. М.: ДИАЛОГ – МИФИ. – 1999.
2. Медведев В.С., Потемкин В.Г. система инженерных и научных расчетов MATLAB 5.X. В 2-х т. М.: ДИАЛОГ – МИФИ. – 1999.
3. Гультяев А. Визуальное моделирование в среде MATLAB: учебный курс. СПб.: ПИТЕР. – 2001.
4. Дьяконов В.П. MATLAB. Учебный курс. СПб.: ПИТЕР. – 2001.
5. Дьяконов В.П. MATLAB 6/6.1/6.5 + Simulink 4/5 в математике и моделировании. Полное руководство пользователя. М.: СОЛОН-Пресс. – 2003.