MATLAB

Язык технических вычислений

Вычисление
Визуализация
Программирование

Getting Started with MATLAB
Начало работы с MATLAB

Содержание

Введение

Что такое MATLAB?

MATLAB – это высокопроизводительный язык для технических расчетов. Он включает в себя вычисления, визуализацию и программирование в удобной среде, где задачи и решения выражаются в форме, близкой к математической. Типичное использование MATLAB – это:

- математические вычисления
- создание алгоритмов
- моделирование
- анализ данных, исследования и визуализация
- научная и инженерная графика
- разработка приложений, включая создание графического интерфейса

MATLAB – это интерактивная система, в которой основным элементом данных является массив. Это позволяет решать различные задачи, связанные с техническими вычислениями, особенно в которых используются матрицы и вектора, в несколько раз быстрее, чем при написании программ с использованием "скалярных" языков программирования, таких как Си или Фортран.

Слово MATLAB означает матричная лаборатория (matrix laboratory). MATLAB был специально написан для обеспечения легкого доступа к LINPACK и EISPACK, которые представляют собой современные программные средства для матричных вычислений.

MATLAB развивался в течении нескольких лет, ориентируясь на различных пользователей. В университетской среде, он представлял собой стандартный инструмент для работы в различных областях математики, машиностроении и науки. В промышленности, MATLAB – это инструмент для высокопродуктивных исследований, разработок и анализа данных.

В MATLAB важная роль отводится специализированным группам программ, называемых toolboxes. Они очень важны для большинства пользователей MATLAB, так как позволяют изучать и применять специализированные методы. Toolboxes – это всесторонняя коллекция функций MATLAB (M-файлов), которые позволяют решать частные классы задач. Toolboxes применяются для обработки сигналов, систем контроля, нейронных сетей, нечеткой логики, вэйвлетов, моделирования и т.д.

Система MATLAB

Система MATLAB состоит из пяти основных частей:

Язык MATLAB. Это язык матриц и массивов высокого уровня с управлением потоками, функциями, структурами данных, вводом-выводом и особенностями объектно-ориентированного программирования. Это позволяет как программировать в "небольшом масштабе" для быстрого создания черновых программ, так и в "большом" для создания больших и сложных приложений.

Среда MATLAB. Это набор инструментов и приспособлений, с которыми работает пользователь или программист MATLAB. Она включает в себя средства для управления переменными в рабочем пространстве MATLAB, вводом и выводом данных, а также создания, контроля и отладки M-файлов и приложений MATLAB.

Управляемая графика. Это графическая система MATLAB, которая включает в себя команды высокого уровня для визуализации двух- и трехмерных данных, обработки изображений, анимации и иллюстрированной графики. Она также включает в себя команды низкого уровня, позволяющие полностью редактировать внешний вид графики, также как при создании Графического Пользовательского Интерфейса (GUI) для MATLAB приложений.

Библиотека математических функций. Это обширная коллекция вычислительных алгоритмов от элементарных функций, таких как сумма, синус, косинус, комплексная арифметика, до более сложных, таких как обращение матриц, нахождение собственных значений, функции Бесселя, быстрое преобразование Фурье.

Программный интерфейс. Это библиотека, которая позволяет писать программы на Си и Фортране, которые взаимодействуют с MATLAB. Она включает средства для вызова программ из MATLAB (динамическая связь), вызывая MATLAB как вычислительный инструмент и для чтения-записи MAT-файлов.

О Simulink

Simulink, сопутствующая MATLAB программа, - это интерактивная система для моделирования нелинейных динамических систем. Она представляет собой среду, управляемую мышью, которая позволяет моделировать процесс путем перетаскивания блоков диаграмм на экране и их манипуляцией. Simulink работает с линейными, нелинейными, непрерывными, дискретными, многомерными системами.

Blocksets – это дополнения к Simulink, которые обеспечивают библиотеки блоков для специализированных приложений, таких как связь, обработка сигналов, энергетические системы.

Real-Time Workshop – это программа, которая позволяет генерировать C код из блоков диаграмм и запускать их на выполнение на различных системах реального времени.

Запуск MATLAB

Эта книга предназначена для начального освоения и изучения MATLAB. Она содержит некоторое количество примеров, которые могут быть запущены и отслежены в MATLAB.

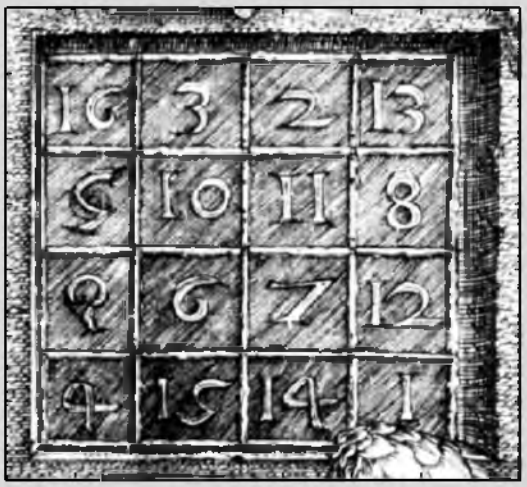
Чтобы запустить MATLAB на PC или Mac, дважды щелкните на иконку MATLAB. Для запуска в системе UNIX напишите *matlab* в строке операционной системы. Для выхода из MATLAB необходимо набрать *quit* в строке MATLAB.

Если вам необходимо получить дополнительную информацию, наберите *help* в строке MATLAB или выберите **Help** в меню на PC или Mac. Мы более подробно расскажем вам об этом позднее.

Матрицы и магические квадраты

Лучший способ начать работу с MATLAB — это научиться обращаться с матрицами. В этой главе мы покажем вам, как надо это делать. В MATLAB матрица – это прямоугольный массив чисел. Особое значение придается матрицам 1x1, которые являются скалярами, и матрицам, имеющим один столбец или одну строку, — векторам. MATLAB использует различные способы для хранения численных и не численных данных, однако вначале лучше всего рассматривать все данные как матрицы. MATLAB организован так, чтобы все операции в нем были как можно более естественными. В то время как другие программные языки работают с числами как элементами языка, MATLAB позволяет вам быстро и легко оперировать с целыми матрицами.

Хороший пример матрицы, которая используется во всей этой книге, можно найти на гравюре времен Ренессанса художника и любителя математики Альбрехта Дюрера. Это изображение содержит много математических символов, и если хорошо присмотреться, то в верхнем правом углу можно заметить квадратную матрицу. Это матрица известна как магический квадрат и во времена Дюрера считалось, что она обладает магическими свойствами. Она и на самом деле обладает замечательными свойствами, стоящими изучения.



Ввод матриц

Вы можете вводить матрицы в MATLAB несколькими способами:

- вводить полный список элементов
- загружать матрицы из внешних файлов
- генерировать матрицы, используя встроенные функции
- создавать матрицы с помощью ваших собственных функций в M–файлах

Начтем с введения матрицы Дюрера как списка элементов. Вы должны следовать нескольким основным условиям:

- отделять элементы строки пробелами или запятыми
- использовать точку с запятой, ; , для обозначения окончания каждой строки
- окружать весь список элементов квадратными скобками, [ ].

Чтобы ввести матрицу Дюрера просто напишите:

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

MATLAB отобразит матрицу, которую мы ввели,

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

Это точно соответствует числам на гравюре. Если мы ввели матрицу, то она автоматически запоминается средой MATLAB. И мы можем к ней легко обратиться как к A. Сейчас, когда мы имеем A в рабочем пространстве MATLAB,

посмотрим, что делает её такой интересной. Почему она называется магической?

Операции суммирования элементов, транспонирования и диагонализации матрицы

Вы возможно уже знаете, что особые свойства магического квадрата связаны с различными способами суммирования его элементов. Если вы берёте сумму элементов вдоль какой-либо строки или столбца, или вдоль какой-либо из двух главных диагоналей, вы всегда получите одно и тоже число. Давайте проверим это, используя MATLAB. Первое утверждение, которое мы проверим –

```
sum(A)
```

MATLAB выдаст ответ

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

Когда выходная переменная не определена, MATLAB использует переменную *ans*, коротко от answer – ответ, для хранения результатов вычисления. Мы подсчитали вектор-строку, содержащую сумму элементов столбцов матрицы A. Действительно, каждый столбец имеет одинаковую сумму, магическую сумму, равную 34.

А как насчет сумм в строках? MATLAB предпочитает работать со столбцами матрицы, таким образом, лучший способ получить сумму в строках – это транспонировать нашу матрицу, подсчитать сумму в столбцах, а потом транспонировать результат. Операция транспонирования обозначается апострофом или одинарной кавычкой. Она зеркально отображает матрицу относительно главной диагонали и меняет строки на столбцы. Таким образом

```
A'
```

вызывает

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

А выражение

```
sum(A')'
```

вызывает результат вектор-столбец, содержащий суммы в строках

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

Сумму элементов на главной диагонали можно легко получить с помощью функции *diag*, которая выбирает эту диагональ.

```
diag(A)
```

```
ans =
  16
  10
   7
   1
```

А функция

```
sum(diag(A))
```

вызывает

```
ans =
  34
```

Другая диагональ, называемая антидиагональю, не так важна математически, поэтому MATLAB не имеет специальной функции для неё. Но функция, которая вначале предполагалась для использования в графике, *fliplr*, зеркально отображает матрицу слева направо.

```
sum(diag(fliplr(A)))
```

```
ans =
     34
```

Таким образом, мы проверили, что матрица на гравюре Дюрера действительно магическая, и научились использовать некоторые матричные операции MATLAB. В последующих разделах мы продолжим использовать эту матрицу для демонстрации дополнительных возможностей MATLAB.

Индексы

Элемент в строке i и столбце j матрицы A обозначается A(i,j). Например, A(4,2) – это число в четвертой строке и втором столбце. Для нашего магического квадрата A(4,2) = 15. Таким образом, можно вычислить сумму элементов в четвертом столбце матрицы A, набрав

```
A(1,4) + A(2,4) + A(3,4) + A(4,4)
```

получим

```
ans =
     34
```

Однако это не самый лучший способ суммирования отдельной строки.

Также возможно обращаться к элементам матрицы через один индекс, A(k). Это обычный способ ссылаться на строки и столбцы матрицы. Но его можно использовать только с двумерными матрицами. В этом случае массив рассматривается как длинный вектор, сформированный из столбцов исходной матрицы.

Так, для нашего магического квадрата, A(8) – это другой способ ссылаться на значение 15, хранящееся в A(4,2).
Если вы пытаетесь использовать значение элемента вне матрицы, MATLAB выдаст ошибку:

```
t=A(4,5)

???  Index exceeds matrix dimensions.
```

С другой стороны, если вы сохраняете значение вне матрицы, то размер матрицы увеличивается.

```
X=A;
X(4,5) = 17

X =
    16     3     2    13     0
     5    10    11     8     0
     9     6     7    12     0
     4    15    14     1    17
```

Оператор двоеточия

Двоеточие, : , - это один из наиболее важных операторов MATLAB. Он проявляется в различных формах. Выражение

```
1:10
```

- это вектор-строка, содержащая целые числа от 1 до 10

```
 1     2     3     4     5     6     7     8     9    10
```

Для получения обратного интервала, опишем приращение. Например

```
100:-7:50
```

что дает

```
100    93    86    79    72    65    58    51
```

или

```
0:pi/4:pi
```

что приводит к

```
0    0.7854    1.5708    2.3562    3.1416
```

Индексное выражение, включая двоеточие, относится к части матрицы.

A(1:k, j)

это первые k элементов j-го столбца матрицы A. Так

sum(A(1:4,4))

вычисляет сумму четвертой строки. Но есть и лучший способ. Двоеточие, само по себе, обращается ко всем элементам в строке и столбце матрицы, а слово *end* – к последней строке или столбцу. Так

```
sum(A(:,end))
```

вычисляет сумму элементов в последнем столбце матрицы A

```
ans =
    34
```

Почему магическая сумма квадрата 4x4 равна 34? Если целые числа от 1 до 16 отсортированы в четыре группы с равными суммами, эта сумма должна быть

```
sum(1:16)/4
```

которая, конечно, равна

```
ans =
    34
```

Функция magic

MATLAB на самом деле обладает встроенной функцией, которая создает магический квадрат почти любого размера. Не удивительно, что эта функция называется *magic*.

```
B=magic(4)

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

Эта матрица почти та же матрица, что и на гравюре Дюрера, и она имеет все те же магические свойства. Единственное отличие заключается в том, что два средних столбца поменялись местами. Для того чтобы преобразовать B в матрицу Дюрера A, переставим их местами.

```
A=B(:,[1 3 2 4])
```

Это означает, что для каждой строки матрицы B элементы переписываются в порядке 1, 3, 2, 4

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

Почему Дюрер переупорядочил столбцы, по сравнению с тем, что использует MATLAB? Без сомнения, он хотел включить дату гравюры, 1514, в нижнюю часть магического квадрата.

Выражения

Как и большинство других языков программирования, MATLAB предоставляет возможность использования математических выражений, но в отличие от многих из них, эти выражения в MATLAB включают матрицы. Основные составляющие выражения:

- переменные
- числа
- операторы
- функции

Переменные

В MATLAB нет необходимости в определении типа переменных или размерности. Когда MATLAB встречает новое имя переменной, он автоматически создает переменную и выделяет соответствующий объем памяти. Если переменная уже существует, MATLAB изменяет ее состав и если это необходимо выделяет дополнительную память. Например,

```
num_students = 25
```

создает матрицу 1x1 с именем num_students и сохраняет значение 25 в ее единственном элементе.

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. Поэтому A и a – не одна и та же переменная. Чтобы увидеть матрицу связанную с переменной, просто введите название переменной.

Числа

MATLAB использует принятую десятичную систему счисления, с необязательной десятичной точкой и знаками плюс-минус для чисел. Научная система счисления использует букву e для определения множителя степени десяти. Мнимые числа используют i или j как суффикс. Некоторые примеры правильных чисел приведены ниже

3	-99	0.0001
9.6397238	1.60210e-20	6.02252e23
1i	-3.14159j	3e5i

Все числа для хранения используют формат long, определенный стандартом плавающей точки IEEE. Числа с плавающей точкой обладают ограниченной точностью - приблизительно 16 значащих цифр и ограниченным диапазоном – приблизительно от 10^{-308} до 10^{308} (Компьютер VAX использует другой формат чисел с плавающей точкой, но их точность и диапазон приблизительно те же).

Операторы

Выражения используют обычные арифметические операции и правила старшинства.

+	сложение
-	вычитание
*	умножение
/	деление
\	левое деление(описано в разделе Матрицы и Линейная Алгебра в книге "*Using MATLAB*")
^	степень
'	комплексно сопряженное транспонирование
()	определение порядка вычисления

Функции

MATLAB предоставляет большое количество элементарных математических функций, таких как *abs*, *sqrt*, *exp*, *sin*. Вычисление квадратного корня или логарифма отрицательного числа не является ошибкой: в этом случае результатом является соответствующее комплексное число. MATLAB также предоставляет и более сложные функции, включая Гамма функцию и функции Бесселя. Большинство из этих функций имеют комплексные аргументы. Чтобы вывести список всех элементарных математических функций, наберите

```
help elfun
```

Для вывода более сложных математических и матричных функций, наберите

```
help specfun
help elmat
```

соответственно.

Некоторые функции, такие как *sqrt* и *sin*, - встроенные. Они являются частью MATLAB, поэтому они очень эффективны, но их вычислительные детали трудно доступны. В то время как другие функции, такие как *gamma* и *sinh*, реализованы в M-файлах. Поэтому вы можете легко увидеть их код и, в случае необходимости, даже модифицировать его.

Несколько специальных функций предоставляют значения часто используемых констант.

pi	3.14159265…
i	мнимая единица, $\sqrt{-1}$
j	то же самое, что и i
eps	относительная точность числа с плавающей точкой, 2^{-52}
realmin	наименьшее число с плавающей точкой, 2^{-1022}
realmax	наибольшее число с плавающей точкой, $(2-\epsilon)2^{1023}$
Inf	бесконечность
NaN	не число

Бесконечность появляется при делении на нуль или при выполнении математического выражения, приводящего к переполнению, т.е. к превышению *realmax*. Не число (*NaN*) генерируется при вычислении выражений типа 0/0 или *Inf – Inf*, которые не имеют определенного математического значения.

Имена функций не являются зарезервированными, поэтому возможно изменять их значения на новые, например

```
eps = 1.e-6
```

и далее использовать это значение в последующих вычислениях. Начальное значение может быть восстановлено следующим образом

```
clear eps
```

Выражения

Вы уже познакомились с некоторыми примерами использования выражений в MATLAB. Ниже приведено еще несколько примеров с результатами.

```
rho = (1+sqrt(5))/2

rho =
    1.6180

a = abs(3+4i)

a =
     5

z = sqrt(besselk(4/3,rho-i))

z =
   0.3730 + 0.3214i

huge = exp(log(realmax))

huge =
1.7977e+308

toobig = pi*huge

toobig =
 Inf
```

Работа с матрицами

Этот раздел расскажет вам о различных способах создания матриц.

Генерирование матриц

MATLAB имеет четыре функции, которые создают основные матрицы:

zeros	все нули
ones	все единицы
rand	равномерное распределение случайных элементов
randn	нормальное распределение случайных элементов

Некоторые примеры:

```
Z = zeros(2,4)

Z =
     0     0     0     0
     0     0     0     0

F = 5*ones(3,3)

F =
     5     5     5
     5     5     5
     5     5     5

N = fix(10*rand(1,10))

N =
     9     2     6     4     8     7     4     0     8     4

R = randn(4,4)

R =
   -0.4326   -1.1465    0.3273   -0.5883
   -1.6656    1.1909    0.1746    2.1832
    0.1253    1.1892   -0.1867   -0.1364
    0.2877   -0.0376    0.7258    0.1139
```

Загрузка матриц

Команда *load* считывает двоичные файлы, содержащие матрицы, созданные в MATLAB ранее, или текстовые файлы, содержащие численные данные. Текстовые файлы должны быть сформированы в виде прямоугольной таблицы чисел, отделенных пробелами, с равным количеством элементов в каждой строке. Например, создадим вне MATLAB текстовой файл, содержащий 4 строки:

```
16.0    3.0    2.0   13.0
 5.0   10.0   11.0    8.0
 9.0    6.0    7.0   12.0
 4.0   15.0   14.0    1.0
```

Сохраним этот файл под именем magik.dat. Тогда команда

load magik.dat

прочитает этот файл и создаст переменную magik, содержащую нашу матрицу.

M-файлы

Вы можете создавать свои собственные матрицы, используя M-файлы, которые представляют собой текстовые файлы, содержащие код MATLAB. Просто создайте файл с выражением, которое вы хотите написать в командной строке MATLAB. Сохраните его под именем, заканчивающимся на .m.

Замечание. Для вызова текстового редактора на PC или Mac, выберите **Open** или **New** из меню **File** или нажмите соответствующую кнопку на панели инструментов. Для обращения к текстовому редактору на UNIX используйте символ ! сразу за командой, которую вы используете в строке операционной системы.

Например, создадим файл, включающий следующие 5 строк:

```
A = [ ...
16.0    3.0     2.0     13.0
5.0     10.0    11.0    8.0
9.0     6.0     7.0     12.0
4.0     15.0    14.0    1.0   ];
```

Сохраним его под именем magik.m. Тогда выражение

magik

прочитает файл и создаст переменную A, содержащую исходную матрицу.

Объединение

Объединение – это процесс соединения маленьких матриц для создания больших. Фактически, вы создали вашу первую матрицу объединением её отдельных элементов. Пара квадратных скобок – это оператор объединения. Например, начнем с матрицы A (магического квадрата 4x4) и сформируем

```
B = [A A+32; A+48 A+16]
```

Результатом будет матрица 8x8, получаемая соединением четырех подматриц

```
B =
    16     2     3    13    48    34    35    45
     5    11    10     8    37    43    42    40
     9     7     6    12    41    39    38    44
     4    14    15     1    36    46    47    33
    64    50    51    61    32    18    19    29
    53    59    58    56    21    27    26    24
    57    55    54    60    25    23    22    28
    52    62    63    49    20    30    31    17
```

Это матрица лишь наполовину является магической. Её элементы представляют собой комбинацию целых чисел от 1 до 64, а суммы в столбцах точно равны значению для магического квадрата 8x8.

```
sum(B)

ans =
   260   260   260   260   260   260   260   260
```

Однако, суммы в строках этой матрицы (*sum(B')'*) не все одинаковы. Необходимо провести дополнительные операции, чтобы сделать эту матрицу действительно магическим квадратом 8x8.

Удаление строк и столбцов

Вы можете удалять строки и столбцы матрицы, используя просто пару квадратных скобок. Рассмотрим

```
X = A;
```

Теперь удалим второй столбец матрицы X.

```
X(:,2) = []
```

Эта операция изменит X следующим образом

```
X =
    16     3    13
     5    10     8
     9     6    12
     4    15     1
```

Если вы удаляете один элемент матрицы, то результат уже не будет матрицей. Так выражение

```
X(1,2) = []
```

результатом вычисления выдаст ошибку. Однако использование одного индекса удаляет отдельный элемент или последовательность элементов и преобразует оставшиеся элементы в вектор-строку. Так

```
X(2:2:10) = []
```

выдаст результат

```
X =
    16     9     3     6    13    12     1
```

Командное окно

До сих пор, мы использовали только командную строку MATLAB, печатая команды и выражения и наблюдая результаты. В этой главе описано несколько способов изменения внешнего вида командного окна. Если ваша система позволяет вам выбирать шрифт, то мы рекомендуем использовать шрифты с фиксированной шириной, такие как Fixedsys или Courier, для обеспечения правильного межстрочного интервала.

Команда format

Команда format управляет численным форматом значений, выводимых MATLAB. Эта операция влияет только на то, как числа изображаются на экране, но не влияет на то, как их вычисляет и сохраняет MATLAB. Ниже представлены различные форматы чисел, используемые для отображения вектора x с компонентами различных величин.

```
x = [4/3  1.2345e-6]

format short

1.3333    0.0000

format short e

1.3333e+000  1.2345e-006

format short g

1.3333  1.2345e-006

format long

1.33333333333333   0.00000123450000

format long e

1.333333333333333e+000    1.234500000000000e-006

format long g

1.33333333333333               1.2345e-006

format bank

1.33          0.00

format rat

4/3          1/810045

format hex

3ff5555555555555   3eb4b6231abfd271
```

Если самый большой элемент матрицы больше 10^3 или самый маленький меньше 10^{-3}, MATLAB применяет общий масштабный коэффициент для форматов *short* и *long*.
В добавление к командам *format*, рассмотренным выше

```
format compact
```

убирает много пустых линий, появляющихся на выходе. Это позволяет вам видеть больше информации на экране. Если вы хотите изменить контроль над форматом выходных данных, используйте функции *sprintf* и *fprintf*.

Сокращение выходных данных

Если вы наберете выражение и нажмете **Return** или **Enter**, MATLAB автоматически выведет результат на экран. Однако если в конце строки вы поставите точку с запятой, MATLAB проведет вычисления, но не отобразит их. Это часто бывает нужно при создании больших матриц. Например,

```
A = magic(100);
```

Длинные командные строки

Если выражение не умещается на одной строке, используйте троеточие, а за ним **Return** или **Enter**, для обозначения того, что выражение продолжается на следующей строке. Например

```
s = 1 -1/2 + 1/3 -1/4 + 1/5 - 1/6 + 1/7 ...
  -1/8 + 1/9 - 1/10 + 1/11 - 1/12;
```

Пробелы вокруг знаков =, +, - не обязательны, но улучшают читаемость текста.

Редактор командной строки

Различные стрелки и управляющие клавиши на вашей клавиатуре позволяют вам вызывать, редактировать и многократно использовать команды, набранные ранее. Например, предположим, что вы допустили ошибку при вводе

```
rho = (1 + sqt(5))/2
```

Вы ошиблись в написании *sqrt*. MATLAB ответит вам предупреждением

```
Undefined function or variable 'sqt'.
```

Вместо того, чтобы заново набирать всю строку, просто нажмите клавишу ↑. Тогда на экране изобразится ошибочная команда. Используйте клавишу ← для перемещения курсора и вставки пропущенной буквы r. Повторное использование клавиши ↑ вызовет предыдущие строки. Наберите несколько символов, и тогда клавиша ↑ найдет предыдущую строку, которая начинается с них.

Список доступных клавиш редактирования в командной строке отличается у разных компьютеров. Поэкспериментируйте, чтобы узнать, какие из нижесле-

дующих клавиш доступны на вашей машине. (Многие из этих клавиш будут знакомы пользователям редактора EMACS.)

↑	ctrl-p	Вызов предыдущей строки
↓	ctrl-n	Вызов последующей строки
←	ctrl-b	Движение назад на один символ
→	ctrl-f	Движение вперед на один символ
ctrl-→	ctrl-r	Движение вправо на одно слово
ctrl-←	ctrl-l	Движение влево на одно слово
home	ctrl-a	Переход на начало строки
end	ctrl-e	Переход на конец строки
esc	ctrl-u	Очистка строки
del	ctrl-d	Удаление символа за курсором
backspace	ctrl-h	Удаление символа перед курсором
	ctrl-k	Удаление до конца строки

Графика

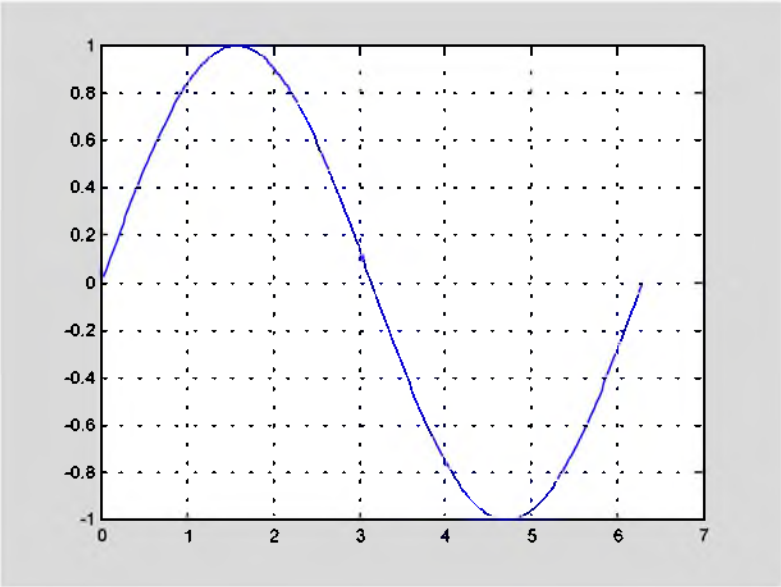
MATLAB имеет широкие возможности для графического изображения векторов и матриц, а также для создания комментариев и печати графики. Эта глава описывает несколько наиболее важных графических функций и дает примеры их применения.

Создание графика

Функция *plot* имеет различные формы, связанные с входными параметрами, например *plot(y)* создает кусочно-линейный график зависимости элементов y от их индексов. Если вы задаете два вектора в качестве аргументов, *plot(x,y)* создаст график зависимости y от x.

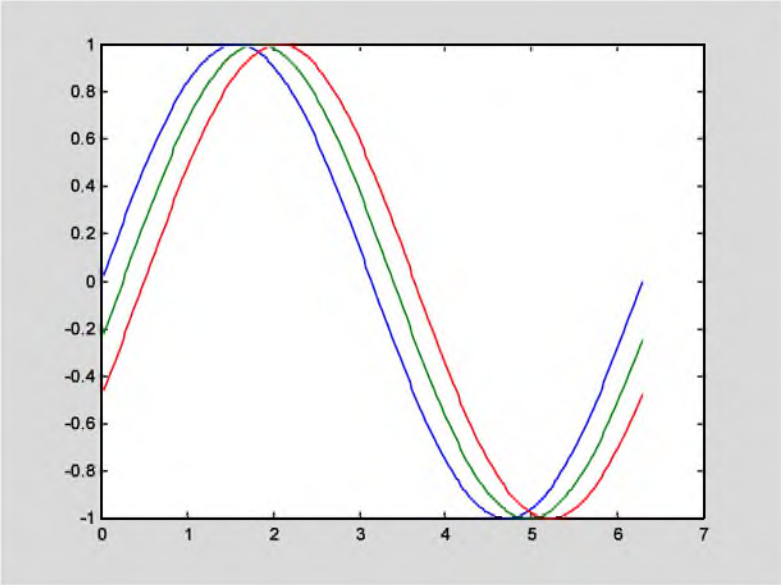
Например, для построения графика значений функции *sin* от нуля до 2π, сделаем следующее

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



Вызов функции *plot* с многочисленными парами x-y создает многочисленные графики. MATLAB автоматически присваивает каждому графику свой цвет (исключая случаи, когда это делает пользователь), что позволяет различать заданные наборы данных. Например, следующие три строки отображают график близких функций, и каждой кривой соответствует свой цвет:

```
y2 = sin(t-.25);
y3 = sin(t-.5);
plot( t, y, t, y2, t, y3)
```



Возможно изменение цвета, стиля линий и маркеров, таких как знаки плюс или кружки, следующим образом

plot(x, y, 'цвет_стиль_маркер')

цвет_стиль_маркер это 1-, 2-, 3- символьная строка (заключенная в одинарные кавычки), составленная из типов цвета, стиля линий и маркеров:

• Символы, относящие к цвету: *'c'*, *'m'*, *'y'*, *'r'*, *'g'*, *'b'*, *'w'* и *'k'*. Они обозначают голубой, малиновый, желтый, красный, зеленый, синий, белый и черный цвета соответственно.

• Символы, относящиеся к типу линий: *' – '* для сплошной, *' – – '* для разрывной, *' : '* для пунктирной, *' –. '* для штрихпунктирной линий и ' none ' для её отсутствия.

• Наиболее часто встречающиеся маркеры *' + '*, *' o '*, *' * '* и *' x '*.

Например, выражение

```
plot(x,y,'y:+')
```

строит желтый пунктирный график и помещает маркеры ' + ' в каждую точку данных. Если вы определяете только тип маркера, но не определяете тип стиля линий, то MATLAB выведет только маркеры.

Окна изображений

Функция *plot* автоматически открывает новое окно изображения (далее окно), если до этого его не было на экране. Если же оно существует, то *plot* использует его по умолчанию. Для открытия нового окна и выбора его по умолчанию, наберите

figure

Для того, чтобы сделать существующеё окно текущим -

figure(n)

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

Добавление кривых на существующий график

Команда *hold* позволяет добавлять кривые на существующий график. Когда вы набираете

hold on

MATLAB не стирает существующий график, а добавляет в него новые данные, изменяя оси, если это необходимо. Например, следующий элемент кода вначале создает контурные линии функции *peaks*, а затем накладывает псевдоцветной график той же функции:

```
[x,y,z] = peaks;
contour(x,y,z,20,'k')
hold on
pcolor(x,y,z)
shading interp
```

Команда *hold on* является причиной того, что график *pcolor* комбинируется с графиком *contour* в одном окне

Подграфики

Функция *subplot* позволяет выводить множество графиков в одном окне или распечатывать их на одном листе бумаги.

subplot(m,n,p)

разбивает окно изображений на матрицу m на n подграфиков и выбирает p-ый подграфик текущим. Графики нумеруются вдоль первого в верхней строке, потом во второй и т.д. Например, для того, чтобы представить графические данные в четырех разных подобластях окна необходимо выполнить следующее:

```
t = 0:pi/10:2*pi;
[X,Y,Z] = cylinder(4*cos(t));
subplot(2,2,1)
mesh(X)
subplot(2,2,2); mesh(Y)
subplot(2,2,3); mesh(Z)
subplot(2,2,4); mesh(X,Y,Z)
```

Мнимые и комплексные данные

Если аргумент функции *plot* комплексное число, то мнимая часть игнорируется, за исключением случая, когда комплексный аргумент один. Для этого специального случая происходит построение графика зависимости реальной части аргумента от мнимой. Поэтому

plot(Z)

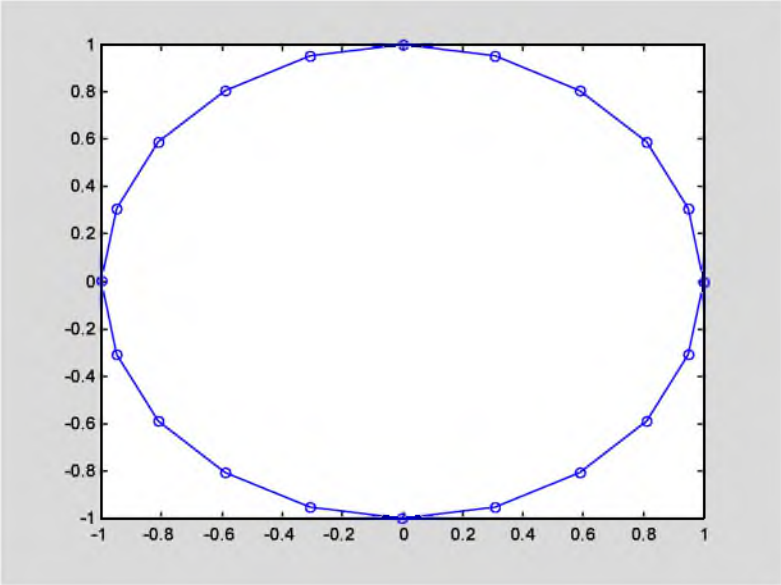
где Z комплексный вектор или матрица, эквивалентно

plot(real(Z),imag(Z))

Например,

```
t = 0:pi/10:2*pi;
plot(exp(i*t),'-o')
```

отобразит двадцатисторонний многоугольник с маленькими кружками на вершинах.



Управление осями

Функция *axis* имеет несколько возможностей для настройки масштаба, ориентации и коэффициента сжатия.

Обычно MATLAB находит максимальное и минимальное значение и выбирает соответствующий масштаб и маркитирование осей. Функция *axis* заменяет значения по умолчанию предельными значения, вводимыми пользователем.

```
axis( [xmin xmax ymin ymax] )
```

В функции *axis* можно также использовать ключевые слова для управления внешним видом осей. Например

```
axis square
```

создает x и y оси равной длины, а

```
axis equal
```

создает отдельные отметки приращений для x и y осей одинаковой длины. Так функция

```
plot(exp(i*t))
```

следующая либо за *axis square*, либо за *axis equal* превращает овал в правильный круг.

```
axis auto
```

возвращает значения по умолчанию и переходит в автоматический режим.

```
axis on
```

включает обозначения осей и метки промежуточных делений.

axis off

выключает обозначения осей и метки промежуточных делений.

grid off

выключает сетку координат, а

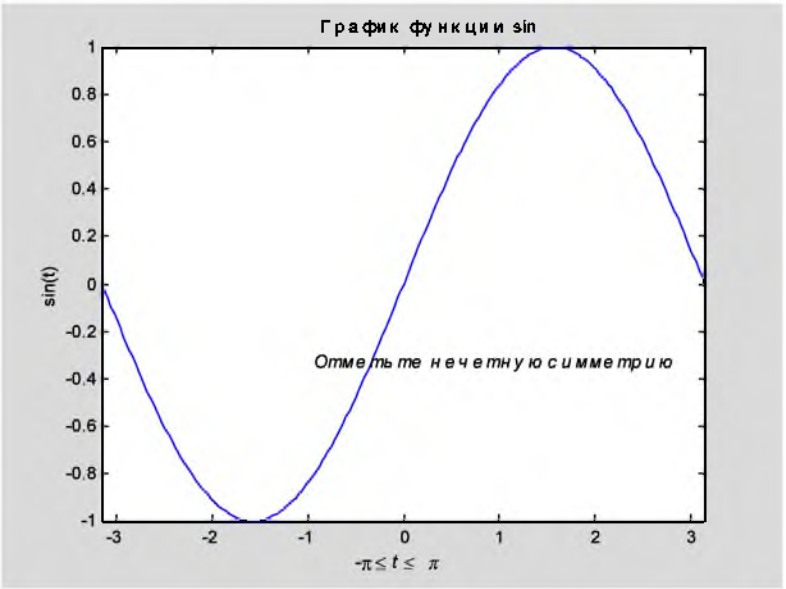
grid on

включает её заново.

Подписи к осям и заголовки

Функции *xlabel*, *ylabel*, *zlabel* добавляют подписи к соответствующим осям, функция title добавляет заголовок в верхнюю часть окна, а функция text вставляет текст в любое место графика. Использование TEX представления позволяет применять греческие буквы, математические символы и различные шрифты. Следующий пример демонстрирует эту возможность.

```
t  = -pi:pi/100:pi;
y = sin(t);
plot(t,y)
axis([-pi pi -1 1])
xlabel( ' -\pi \leq \itt \leq \pi ' )
ylabel( ' sin(t) ' )
title( ' График функции sin ' )
text(-1, -1/3, ' \it{Отметьте нечетную симметрию} ' )
```



Функции mesh и surface

MATLAB определяет поверхность как z координаты точек над координатной сеткой плоскости x-y, используя прямые линии для соединения соседних точек. Функции *mesh* и *surface* отображают поверхность в трех измерениях. При этом

mesh создает каркасную поверхность, где цветные линии соединяют только заданные точки, а функция *surface* вместе с линиями отображает в цвете и саму поверхность.

Визуализация функций двух переменных

Для отображения функции двух переменных, z = f (x,y), создаются матрицы X и Y, состоящие из повторяющихся строк и столбцов соответственно, перед использованием функции. Затем используют эти матрицы для вычисления и отображения функции. Функция *meshgrid* преобразует область определения, заданную через один вектор или два вектора x и y, в матрицы X и Y для использования при вычислении функции двух переменных. Строки матрицы X дублируют вектор x, а столбцы Y – вектор y.

Для вычисления двумерной функции *sinc* , *sin(r)/r*, в области x-y поступают следующим образом

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

В этом примере R – это рассояние от начала координат, которому соответствует центр матрицы. Добавление *eps* позволяет избежать неопределенности 0/0 в начале координат.

Изображения

Двумерные массивы могут отображать как изображения, где элементы массива определяют их яркость и цвет. Например

```
load durer
whos
```

покажет, что файл durer.mat в директории demo состоит из матрицы размером 648 на 509 (матрицы X) и матрицы размером 128 на 3 (матрицы map). Элементы матрицы X – это целые числа от 1 до 128, которые служат индикаторами в цветном отображении, map. Следующие строки

```
imag(X)
colormap(map)
axis image
```

воспроизводят гравюру Дюрера, показанную в начале этой книги. Высокое разрешение магического квадрата, находящегося в правом верхнем углу, доступно в другом файле. Наберите

```
load detail
```

и используйте стрелку 'вверх' на клавиатуре для повторного запуска команд *image, colormap* и *axis*.

```
colormap(hot)
```

добавит цветовую гамму двадцатого века на гравюру шестнадцатого.

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. Команда *print* обеспечивает большую гибкость при выводе выходных данных и позволяет контролировать печать из M-файлов. Результат может быть послан прямо на принтер, выбранный по умолчанию, или сохранен в заданном файле. Возможно широкое варьирование формата выходных данных, включая использование PostScript.

Например, следующая команда сохранит текущее окно изображения как цветной PostScript Level 2 Encapsulated в файле magicsquare.eps:

```
print –depsc2 magicsquare.eps
```

Важно знать возможности вашего принтера перед использованием команды *print*. Например, файлы Postscript Level 2 обычно меньше и воспроизводятся намного быстрее, чем Postscript Level 1. Однако, не все Postscript принтеры поддерживают Level 2, таким образом вам необходимо узнать, что может обрабатывать ваше устройство вывода. MATLAB использует дифференцированный подход для вывода графики и текста, даже для черно-белых устройств.

Справка и текущая документация

Есть несколько способов получить текущую документацию по функциям MATLAB.

- Команда help
- Окно справки
- MATLAB Help Desk
- Текущие справочные страницы
- Связь с The MathWorks, Inc.

Команда help

Команда *help* – это самый основной способ определения синтаксиса и поведения отдельных функций. Информация отображается прямо в командном окне. Например

```
help magic
```

выдаст

```
MAGIC  Magic square.
   MAGIC(N) is an N-by-N matrix constructed from the integers
   1 through N^2 with equal row, column, and diagonal sums.
   Produces valid magic squares for N = 1,3,4,5,...
```

Замечание MATLAB в текущей справке использует заглавные буквы для функций и переменных для того, чтобы выделить их из текста. Однако, при наборе имен функций всегда используйте соответствующие строчные буквы, так как MATLAB чувствителен к регистрам, а все имена функции строчные.

Все функции MATLAB организованы в логические группы и структура директорий MATLAB базируется на этом группировании. Например, все функции линейной алгебры находятся в директории matfun. Чтобы вывести имена всех функций в этой директории с кратким описанием, надо набрать

```
help matfun

Matrix functions - numerical linear algebra.

Matrix analysis.
  norm        - Matrix or vector norm.
  normest     - Estimate the matrix 2-norm.
...
```

Команда

```
help
HELP topics:

matlab\general       -  General purpose commands.
matlab\ops           -  Operators and special characters.
...
```

Окно справки

Окно справки MATLAB появляется на PC или Mac после выбора опции **Help Window** в меню **Help** или нажатием кнопки вопроса на панели инструментов. Эта же операция может быть выполнена при наборе команды

```
helpwin
```

Для вывода окна справки по отдельным разделам, наберите

```
helpwin topic
```

Окно справки предоставляет вам такую же информацию, как и команда *help*, но оконный интерфейс обеспечивает более удобную связь с другими разделами справки.

Команда lookfor

Команда *lookfor* позволяет искать функции по ключевому слову. Она просматривает первую строку текста справки, называемую строкой **H1**, для каждой функции MATLAB и возвращает строки H1, содержащие заданное ключевое слово. Например, MATLAB не имеет функции с именем inverse. Поэтому ответ на запрос

```
help inverse
```

будет

```
inverse.m not found.
```

В то время как

```
lookfor inverse
```

найдет множество согласованных ответов. В зависимости от того, какие toolboxes вы установили, вы получите соответствующие записи. Например

```
INVHILB Inverse Hilbert matrix.
ACOS   Inverse cosine.
ACOSH  Inverse hyperbolic cosine.
ACOT   Inverse cotangent.
ACOTH  Inverse hyperbolic cotangent.
ACSC   Inverse cosecant.
ACSCH  Inverse hyperbolic cosecant.
ASEC   Inverse secant.
ASECH  Inverse hyperbolic secant.
ASIN   Inverse sine.
ASINH  Inverse hyperbolic sine.
ATAN   Inverse tangent.
...
```

Добавление ключа –all в команду *lookfor*, как, например,

```
lookfor –all
```

позволяет искать ключевое слово во всех записях справки, а не только в строке H1.

Help Desk

MATLAB Help Desk обеспечивает доступ к широкому диапазону справочной информации, хранящейся на диске или на CD вашей системы. Большая часть важной документации использует HTML (HyperText Markup Language) формат и доступна через браузеры Интернет, такие как Netscape или Microsoft Explorer. Запустить Help Desk можно на Pc или Mac, выбрав опцию **Help Desk** в меню **Help** или, на всех компьютерах, просто набрав команду

```
helpdesk
```

Все операторы и функции MATLAB имеют текущие справочные страницы в формате HTML, к которым можно обратиться через Help Desk. Эти страницы предоставляют больше деталей и примеров, чем обычные записи справки. Также доступны HTML версии и других документов, включая это справочное пособие. Средство поиска, запущенное на вашей машине, может найти все текущие справочные материалы.

Команда doc

Если вы знаете имя конкретной функции, вы можете обратиться к её текущей справочной странице напрямую. Например, для получения справочной страницы для функции *eval*, наберите

```
doc eval
```

Команда *doc* сама запустит ваш Web браузер, если он еще не был запущен ранее.

Печать текущих справочных страниц

Версии текущих справочных страниц, как и большинство документации, также доступно в формате PDF (Portable Document Format) через Help Desk. Эти страницы обрабатываются с помощью Adobe's Acrobat reader. Они воспроизводят внешний вид страниц после печати, полностью с шрифтами, графикой, с заданным форматом и рисунками. Это лучший способ получить печатные копии справочных материалов.

Связь с MathWorks

Если ваш компьютер подключен в Интернет, Help Desk обеспечивает вам связь с MathWorks – домом MATLAB. вы можете использовать электронную почту для вопросов, предложений и сообщений о возможных ошибках. Вы можете также использовать Solution Search Engine на сайте MathWorks Web для запроса в новейшую базу данных по технической поддержке.

Среда MATLAB

Среда MATLAB включает в себя как совокупность переменных, созданных за время сеанса работы MATLAB, так и набор файлов, содержащих программы и данные, которые продолжают существовать между сеансами работы.

Рабочее пространство

Рабочее пространство – это область памяти, доступная из командной строки MATLAB. Две команды, *who* и *whos*, показывают текущее содержание рабочего пространства. Команда who выдает краткий список, а команда *whos* размер и используемую память.

Ниже представлен вывод, осуществленный командой *whos*, на рабочем пространстве, содержащем результаты из некоторых примеров этой книги. Он показывает различия в структуре данных MATLAB. В качестве упражнения попробуйте найти сегмент кода, который соответствует каждой из ниже приведенных переменных.

```
whos
```

Name	Size	Bytes	Class
A	4x4	128	double array
D	5x3	120	double array
M	10x1	3816	cell array
S	1x3	442	struct array
h	1x11	22	char array
n	1x1	8	double array
s	1x5	10	char array
v	2x5	20	char array

Grand total is 471 elements using 4566 bytes.

Для удаления всех существующих переменных из рабочего пространства MATLAB, введите

```
clear
```

Команда save

Команда *save* сохраняет содержание рабочего пространства в MAT-файле, который может быть прочитан командой *load* в последующих сеансах работы MATLAB. Например,

```
save August17th
```

сохраняет содержание всего рабочего пространства в файле August17th.mat. Если нужно, вы можете сохранить только определенные переменные, указывая их имена после имени файла.

Обычно, переменные сохраняются в двоичном формате, который может быть быстро (и точно) прочитан MATLAB. Если же вы хотите использовать эти файлы вне MATLAB, вы можете указать другой формат.

-ascii	Использует 8-значный текстовой формат.
-ascii -double	Использует 16-значный текстовой формат.
-ascii -double -tabs	Разделяет элементы массива табуляцией.
-v4	Создает файл для MATLAB 4.
-append	Добавляет данные в существующий MAT-файл.

Когда вы сохраняете содержание рабочего пространства в текстовом файле, вы должны сохранять только одну переменную в данный момент. Если вы сохраняете более одной переменной, MATLAB создает текстовой файл, но вы не сможете загрузить его обратно.

Маршрут поиска

MATLAB использует маршрут поиска, упорядоченный список директорий, для того, чтобы определить как выполнять функции, которые вы вызываете. Когда вы вызываете стандартную функцию, MATLAB исполняет первый M-файл на своем пути, который имеет заданное имя. Вы можете заменить поведение использованием специальных директорий и поддиректорий.

Команда

```
path
```

показывает маршрут поиска на всех платформах. На Pc и Mac выберите опцию **Set Path** из меню **File** для просмотра и изменения маршрута.

Операции над дисковыми файлами

Команды *dir, type, delete* и *cd* осуществляют комплекс групповых операционных системных команд для манипуляций над файлами. Нижеприведенная таблица показывает, как эти команды соответствуют различным операционным системам.

MATLAB	MS-DOS	UNIX	VAX/VMS
dir	dir	ls	dir
type	type	cat	type
delete	del, erase	rm	delete
cd	chdir	cd	set defalt

Для большинства из этих команд вы можете использовать полные пути, шаблоны и указатели дисков в обычной форме.

Команда diary

Команда *diary* создает дневник сеанса MATLAB в дисковом файле. Вы можете просмотреть и отредактировать конечный текстовой файл, используя любой текстовой процессор. Для создания файла под именем diary, который содержит все команды, которые вы используете, включая вывод на печать (кроме графического вывода), введите

```
diary
```

Для сохранения сеанса MATLAB в файле с определенным именем, используйте

```
diary filename
```

При остановке записи сеанса работы, наберите

```
diary off
```

Запуск внешних программ

Восклицательный знак означает выход из оболочки MATLAB. Он также означает, что последующая строка ввода будет командой к операционной системе (или Finder на Mac). Это очень полезно для вызова утилит или запуска других программ без выхода из MATLAB. На VMS, например,

```
!edt magik.m
```

вызовет редактор, называемый edt, для файла magik.m. Когда вы выходите из внешней программы, операционная система возвращает управление MATLAB.

Подробнее о матрицах и массивах

Этот раздел расскажет вам больше о работе с матрицами и массивами, уделяя особое внимание

- Линейной алгебре
- Массивам
- Многомерным данным

Линейная алгебра

Термины матрица и массив часто неправильно используют как взаимозаменяемые. Более точно, матрица – это двумерный численный массив, используемый в линейных преобразованиях. Математические операции, определенные на матрицах являются объектом линейной алгебры.

Магический квадрат Дюрера

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

используется в некоторых примерах, которые позволяют почувствовать операции над матрицами в MATLAB. Вы уже видели операцию транспонирования матрицы, A'. Добавление матрицы к её транспонированной дает симметричную матрицу.

```
A + A'

ans =
    32     8    11    17
     8    20    17    23
    11    17    14    26
    17    23    26     2
```

Символ умножения, *, обозначает матричное произведение, включающее внутреннее произведение между строками и столбцами. Умножение матрицы на её транспонированную также дает симметричную матрицу.

```
A'*A

ans =
   378   212   206   360
   212   370   368   206
   206   368   370   212
   360   206   212   378
```

Определитель этой частной матрицы оказывается равным нулю, означая, что эта матрица является сингулярной.

```
d = det(A)
```

```
d =
     0
```

Приведенная к строкам ступенчатая форма матрицы A выглядит следующим образом

```
R = rref(A)

R =
     1     0     0     1
     0     1     0    -3
     0     0     1     3
     0     0     0     0
```

Поскольку заданная матрица является сингулярной, то она не имеет обратной. Если вы все-таки попытаетесь её определить

```
X = inv(A)
```

то получите предупреждение

```
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.175530e-017.
```

Ошибка округления препятствует алгоритму обращения матрицы при определении точной сингулярности. Но значение *rcond*, которое устанавливает условие оценки для обратной матрицы, имеет порядок *eps*, относительной точности числа с плавающей точкой, поэтому вычисление обратной матрицы не очень желательно.

Интересно найти собственные значения магического квадрата

```
e = eig(A)

e =
   34.0000
    8.0000
   -0.0000
   -8.0000
```

Одно из собственных значений равно нулю, что является следствием сингулярности. Самое большое собственное значение равно 34, магической сумме. Это происходит потому, что вектор, состоящий из всех единиц, является собственным вектором.

```
v = ones(4,1)

v =
     1
     1
     1
     1

A*v
```

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

Когда магический квадрат измеряется его магической суммой

```
P = A/34
```

результат будет бистохастической матрицей, у которой суммы в строках и столбцах все равны единицам.

```
P =
    0.4706    0.0882    0.0588    0.3824
    0.1471    0.2941    0.3235    0.2353
    0.2647    0.1765    0.2059    0.3529
    0.1176    0.4412    0.4118    0.0294
```

Такие матрицы представляют собой вероятность перехода в Марковском процессе. Повторные возведения матрицы в степень являются повторными шагами в этом процессе. Для нашего примера, пятая степень

```
P^5
```

равна

```
ans =
    0.2507    0.2495    0.2494    0.2504
    0.2497    0.2501    0.2502    0.2500
    0.2500    0.2498    0.2499    0.2503
    0.2496    0.2506    0.2505    0.2493
```

Из этого следует, что если k стремится к бесконечности, тогда все элементы в k-ой степени, P^k, стремятся к ¼.

В заключение, рассмотрим коэффициенты характеристического полинома

```
poly(A)
```

```
ans =
  1.0e+003 *
    0.0010   -0.0340   -0.0640    2.1760    0.0000
```

Из чего следует, что характеристический полином

$\det(A-\lambda I)$

равен

$$\lambda^4 - 34\lambda^3 - 64\lambda^2 + 2176\lambda$$

Константа равна нулю, так как матрица является сингулярной, а коэффициент при третьей степени равен 34, так как матрица магическая!

Массивы

Когда мы выходим из мира линейной алгебры, матрицы становятся двумерными численными массивами. Арифметические операции на массивах производятся поэлементно. Это означает, что суммирование и вычитание являются одинаковыми операциями для матриц и массивов, а умножение для них различно. MATLAB использует точку, или десятичную точку, как часть записи для операции умножения массивов.

Список операторов включает в себя:

+	суммирование
-	вычитание
.*	поэлементное умножение
./	поэлементное деление
.\	поэлементное левое деление
.^	поэлементное возведение в степень
.'	несопряженное матричное транспонирование

Если магический квадрат Дюрера умножить на себя по правилам умножения массивов

```
A.*A
```

результатом будет массив, содержащий квадраты целых чисел от 1 до 16 в необычном порядке

```
ans =
   256     9     4   169
    25   100   121    64
    81    36    49   144
    16   225   196     1
```

Операции над массивами полезны для создания таблиц. Пусть n – это вектор-столбец

```
n = (0:9)';
```

Тогда

```
pows = [n  n.^2  2.^n]
```

создает таблицу квадратов и степеней двойки

```
pows =
     0     0     1
     1     1     2
     2     4     4
     3     9     8
     4    16    16
     5    25    32
     6    36    64
     7    49   128
     8    64   256
     9    81   512
```

Элементарные математические функции работают с массивами поэлементно. Так

```
format short g
x = (1:0.1:2)';
logs = [x log10(x)]
```

создает таблицу логарифмов

```
logs =
           1            0
         1.1     0.041393
         1.2     0.079181
         1.3      0.11394
         1.4      0.14613
         1.5      0.17609
         1.6      0.20412
         1.7      0.23045
         1.8      0.25527
         1.9      0.27875
           2      0.30103
```

Многомерные данные

MATLAB использует метод ориентации столбцов для многомерных статистических данных. Каждый столбец в наборе данных представляет переменную, а каждая строка – результаты наблюдений. Таким образом, элемент (i,j) – это i-ое наблюдение j-ой переменной.

В качестве примера рассмотрим набор данных с тремя переменными

- частота сердечных сокращений
- вес
- часы упражнений за неделю

Для пяти наблюдений, результирующий массив может выглядеть следующим образом

```
D =
          72          134          3.2
          81          201          3.5
          69          156          7.1
          82          148          2.4
          75          170          1.2
```

Первая строка содержит частоту сердечных сокращений, вес и часы упражнений для первого пациента, вторая строка содержит аналогичные данные для второго и т.д. Вы можете применить многие функции MATLAB для анализа данных к этому набору информации. Например, для того, чтобы получить среднее и среднеквадратичное отклонение для каждого столбца, надо

```
mu = mean(D), sigma = std(D)
```

```
mu =
         75.8         161.8          3.48
sigma =
       5.6303       25.499       2.2107
```

Чтобы посмотреть список всех функций MATLAB для анализа данных, наберите

```
help datafun
```

Если вам нужно узнать о Statistics Toolbox, введите

```
help stats
```

Скалярное расширение

Матрицы и скаляры могут комбинироваться различными путями. Например, скаляр вычитается из матрицы путем вычитания из каждого элемента. Среднее значение элементов для нашего магического квадрата равно 8.5, поэтому

```
B = A - 8.5
```

формирует матрицу, у которой суммы в столбцах равны нулю

```
B =
          7.5         -5.5         -6.5          4.5
         -3.5          1.5          2.5         -0.5
          0.5         -2.5         -1.5          3.5
         -4.5          6.5          5.5         -7.5

sum(B)

ans =
     0     0     0     0
```

Используя скалярное расширение, MATLAB указывает заданный скаляр всем индексам в диапазоне. Например,

```
B(1:2,2:3)=0
```

обнуляет часть матрицы B

```
B =
          7.5            0            0          4.5
         -3.5            0            0         -0.5
          0.5         -2.5         -1.5          3.5
         -4.5          6.5          5.5         -7.5
```

Логическая индексация

Логические вектора, созданные из логических операторов и операторов сравнения, могут быть использованы для ссылки на подмассивы. Предположим, что X обыкновенная матрица и L матрица того же размера, но содержащая логические операции. Тогда X(L) задает элементы X, в которых элементы L ненулевые.

Этот вид индексации может быть осуществлен за один шаг указанием логической операции, такой как индексация выражения. Пусть вы имеете следующий набор данных.

```
x =
  2.1  1.7  1.6  1.5  NaN  1.9  1.8  1.5  5.1  1.8  1.4  2.2  1.6  1.8
```

NaN – это метка для недостающего наблюдения, как например ошибка при ответе на вопрос анкеты. Для того, чтобы убрать данные с логической индексацией, используйте *finite(x)*, которая является истиной для всех конечных численных значений и ложью для *NaN* и *Inf*.

```
x = x(finite(x))

x =
2.1  1.7  1.6  1.5  1.9  1.8  1.5  5.1  1.8  1.4  2.2  1.6  1.8
```

Сейчас осталась одна наблюдаемая величина, 5.1, заметно отличающаяся от остальных, – это выброс. Последующие действия устраняют выбросы, в данном случае те элементы, для которых среднеквадратичное отклонение более, чем в три раза уклоняется от среднего.

```
x = x(abs(x-mean(x)) <= 3*std(x))

x =
2.1  1.7  1.6  1.5  1.9  1.8  1.5  1.8  1.4  2.2  1.6  1.8
```

В качестве другого примера найдем положение простых чисел в магическом квадрате Дюрера, используя логическое индексирование и скалярное расширение, что обнулить все не простые числа.

```
A(~isprime(A))=0

A =
     0     3     2    13
     5     0    11     0
     0     0     7     0
     0     0     0     0
```

Функция find

Функция *find* определяет индексы массива элементов, с заданными логическими условиями. В наиболее простой форме, *find* возвращает вектор-столбец индексов. Транспонируем его для получения вектора-строки. Например,

```
k = find(isprime(A))'
```

выбирает положения, используя одномерное индексирование, простых чисел магического квадрата

```
k =
     2     5     9    10    11    13
```

Покажем эти числа, как вектор-строку в порядке определенном функцией.

```
A(k)

ans =
     5     3     2    11     7    13
```

Если вы используете k как индекс с левой стороны в операторе присваивания, то матричная структура сохраняется.

```
A(k) = NaN

A =
    16   NaN   NaN   NaN
   NaN    10   NaN     8
     9     6   NaN    12
     4    15    14     1
```

Управление потоками

MATLAB имеет пять видов структур управления потоками:

- оператор *if*
- оператор *switch*
- циклы *for*
- циклы *while*
- оператор *break*

if

Оператор *if* вычисляет логическое выражение и выполняет группу операторов, если выражение истинно. Необязательные ключевые слова *elseif* и *else* служат для выполнения альтернативных групп операторов. Ключевое слово *end*, которое согласуется с *if*, завершает последнюю группу операторов. Таким образом, все группы операторов заключены между четырех ключевых слов, без использования фигурных или обычных скобок.

Алгоритм MATLAB для создания магического квадрата порядка n включает три разных случая : n нечетное, n четное, но не делится на 4, и n четное и делится на 4. Ниже приведен пример соответствующего кода.

```
if rem(n,2) ~= 0
   M = odd_magic(n)
elseif rem(n,4) ~= 0
   M = single_even_magic(n)
else
   M = double_even_magic(n)
end
```

В этом примере три случая являются взаимно исключающими, но если бы это было не так, то выполнялось бы первое истинное условие.

Важно понять, как оператор отношения и оператор *if* работают с матрицами. Когда вы хотите узнать, равны ли две переменные, нужно использовать следующую конструкцию

```
if A == B, . . .
```

Это правильный код MATLAB и он осуществляет то, что вы ожидаете, если A и B являются скалярами. Но когда A и B – матрицы, A == B не работает, если они не равны. Равенство матриц означает поэлементное равенство. Фактически, если A и B имеют различные размеры, MATLAB выдаст ошибку.

Правильный способ определения равенства между двумя переменными – это использование функции *isequal*

```
if isequal(A,B), . . .
```

Далее приведен другой пример, который исследует этот вопрос. Если A и B являются скалярами, то нижеприведенная программа никогда не приведет к неожиданной ситуации. Но для большинства пар используемых матриц, включая наши магические квадраты с переставленными столбцами, ни одно из условий A > B, A < B или A ==B не является истинным для всех элементов и поэтому выполняется случай *else*.

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

Некоторые функции могут быть полезны для матричного сравнения при использовании с оператором *if*, например

```
isequal
isempty
all
any
```

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. Необходимо использовать *end* для согласования с *switch*.

Логика алгоритма магических квадратов может быть описана на следующем примере

```
switch (rem(n,4)==0) + (rem(n,2)==0)
   case 0
      M = odd_magic(n)
   case 1
      M = single_even_magic(n)
   case 2
      M = double_even_magic(n)
   otherwise
      error( ' Это невозможно ' )
end
```

Замечание для программистов Си. В отличие от языка Си, оператор switch в MATLAB не "проваливается". Если первый случай является истинным, другие

случаи не выполняются. Таким образом, нет необходимости в использовании оператора break.

for

Цикл *for* повторяет группу операторов фиксированное, предопределенное число раз. Ключевое слово *end* очерчивает тело цикла.

```
for n = 3:32
   r(n) = rank(magic(n));
end
r
```

Точка с запятой после выражения в теле цикла предотвращает повторения вывода результатов на экран, а r после цикла выводит окончательный результат.

Хорошим стилем являются отступы при использовании циклов для лучшей читаемости, особенно, когда они вложенные.

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

while

Цикл *while* повторяет группу операторов определенное число раз, пока выполняется логическое условие. Ключевое слово *end* очерчивает используемые операторы.

Ниже приведена полная программа, иллюстрирующая работу операторов *while*, *if*, *else* и *end*, которая использует метод деления отрезка пополам для нахождения нулей полинома.

```
a = 0; fa = -Inf;
b = 3; fb = Inf;
while b-a > eps*b
   x = (a+b)/2;
   fx = x^3-2*x-5;
   if sign(fx) == sign(fa)
      a = x; fa = fx;
   else
      b = x; fb = fx;
   end
end
x
```

Результатом будет корень полинома x^3-2x-5

```
x =
   2.09455148154233
```

Для оператора while верны те же предостережения относительно матричного сравнения, что и для оператора if, которые обсуждались ранее.

break

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

Ниже представлен улучшенный вариант предыдущего примера. Почему использование оператора *break* в данном случае выгодно?

```
a = 0; fa = -Inf;
b = 3; fb = Inf;
while b-a > eps*b
   x = (a+b)/2;
   fx = x^3-2*x-5;
   if fx ==0
      break
   elseif sign(fx) == sign(fa)
      a = x; fa = fx;
   else
      b = x; fb = fx;
   end
end
x
```

Другие структуры данных

Этот раздел познакомит вас с некоторыми структурами данных в MATLAB, включая

- Многомерные массивы
- Массивы ячеек
- Символы и текст
- Структуры

Многомерные массивы

Многомерные массивы в MATLAB - это массивы более чем с двумя индексами. Они могут быть созданы вызовом функций *zeros*, *ones*, *rand* или *randn* с более чем двумя аргументами. Например

R = randn(3,4,5)

создает 3x4x5 массив с 3*4*5=60 нормально распределенными случайными элементами.

Трехмерные массивы могут представлять трехмерные физические данные, например, температуру в комнате. Результат представляется последовательностью матриц $A^{(k)}$ или зависящей от времени матрицей A(t). В этих случаях, элемент (i,j) k-ой матрицы или t_k-ой обозначается A(i,j,k).

Версии магического квадрата MATLAB и Дюрера отличаются переставленными столбцами. Оператор

```
p = perms(1:4);
```

генерирует 4! = 24 перестановки чисел от 1 до 4. k-ая перестановка – это вектор-строка, p(k,:). Тогда

```
A = magic(4);
M = zeros(4,4,24);
for k = 1:24
  M(:,:,k) = A(:,p(k,:));
end
```

хранит последовательность 24 магических квадратов в трехмерном массиве M. Размер M равен

```
size(M)

ans =
     4     4    24
```

Оказывается, что 22-ая матрица в этой последовательности – матрица Дюрера:

```
M(:,:,22)
```

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

Функция

```
sum(M,d)
```

вычисляет суммы, изменяя индекс d. Так

```
sum(M,1)
```

- это 1x4x24 массив, содержащий 24 копии вектора-строки

```
34    34    34    34
```

А

```
sum(M,2)
```

является массивом 4x1x24, содержащим 24 копии вектора-столбца

```
34
34
34
34
```

И наконец,

```
S = sum(M,3)
```

добавляет 24 матрицы в последовательность. Результат имеет размерность 4x4x1, поэтому он выглядит как массив 4x4,

```
S =
   204   204   204   204
   204   204   204   204
   204   204   204   204
   204   204   204   204
```

Массивы ячеек

Массивы ячеек в MATLAB – это многомерные массивы, элементы которых являются копиями других массивов. Массив ячеек пустых матриц может быть создан с использованием функции *cell*. Но более часто они создаются путем заключения разнообразной группы объектов в круглые скобки. Круглые скобки также используются с индексами для получения доступа к содержанию различных ячеек. Например,

```
C = { A sum(A) prod(prod(A)) }
```

дает массив ячеек 1x3. Эти три клетки содержат: магический квадрат, вектор-строку с суммами в столбцах и произведение его элементов. Если отобразить C на экране, то вы увидите следующее

```
C =
    [4x4 double]    [1x4 double]    [2.0923e+013]
```

Это происходит потому, что первые две ячейки слишком большие для вывода в этом ограниченном пространстве, а третья ячейка содержит только отдельное число, 16!, и для него есть необходимая область вывода.

Очень важно запомнить два важных момента. Первое, для получения содержания одной ячейки, используйте индекс в круглых скобках. Например, C{1} возвращает магический квадрат, а C{3} – 16!. Второе, массивы ячеек содержат копии других массивов, а не их указатели. Поэтому, если вы впоследствии измените матрицу A, с C ничего не произойдет.

Трехмерные массивы могут быть использованы для хранения последовательности матриц одинакового размера. А массивы ячеек могут применяться для хранения последовательности матриц различного размера. Например,

```
M = cells(8,1);
for n =1:8
  M{n} = magic(n);
end
M
```

создает последовательность магических квадратов в определенной последовательности

```
M =
    [          1]
    [2x2 double]
    [3x3 double]
    [4x4 double]
    [5x5 double]
    [6x6 double]
    [7x7 double]
    [8x8 double]
```

Вы можете найти нашего старого друга, просто набрав

```
M{4}
```

Символы и текст

Введите текст в MATLAB, используя одинарные кавычки. Например,

```
s = 'Hello'
```

Результатом не будет являться численная матрица или массив, которые мы рассматривали ранее. Это будет символьный массив 1x5.

Внутренне, символы хранятся как числа, но не в формате с плавающей точкой. Оператор

```
a = double(s)
```

преобразует массив символов в числовую матрицу, содержащую представление с плавающей точкой ASCII кода для каждого символа. Результатом будет

```
a =
    72   101   108   108   111
```

А выражение

```
s = char(a)
```

осуществляет обратное превращение.

Преобразование чисел в символы делает возможным присутствие различных шрифтов на вашем компьютере. Печатаемые символы в ASCII коде представляются целыми числами от 32 до 127. (Целые числа меньше 32 представляют непечатаемые символы.) Эти числа расположены в соответствующем массиве 6x16

```
F = reshape(32:127,16,6)';
```

Печатаемые символы в расширенном ASCII наборе представлены F+128. Когда эти числа интерпретируются как символы, результат зависит от того, какой шрифт в данный момент используется. Наберите следующее

```
char(F)
char(F+128)
```

и потом поизменяйте шрифты в командном окне MATLAB. Ниже представлен один из примеров того, что может получиться.

```
ans =
 !"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~·
ans =
 ЎўJ¤Ґ¦§Ё©Є«¬-®Ї
°±Iiґµ¶·ё№є»jSsї
АБВГДЕЖЗИЙКЛМНОП
РСТУФХЦЧШЩЪЫЬЭЮЯ
абвгдежзийклмноп
рстуфхцчшщъыьэюя
```

Соединение квадратными скобками конкатенирует текстовые переменные вместе в большую строку.

```
h = [s, ' world']
```

объединяет строки по горизонтали и дает

```
h =
Hello world
```

Оператор

```
v = [s; 'world']
```

объединяет строки вертикаль, что приводит к

```
v =
Hello
world
```

Заметьте, что перед символом w в переменной h необходимо поставить пробел, а оба слова в переменной v должны быть равной длины. Результирующие массивы являются снова массивами символов: переменная h – 1x11, а переменная v – 2x5.

Есть два способа, чтобы управлять группой текста, содержащей строки разной длины: формировать заполненный массив символов или клеточный массив строк. Функция *char* принимает любое число строк, добавляет пробелы в каждую строку, чтобы все они были равной длины, и формирует массив строк с символьной строкой в каждой строке. Например,

```
S = char('A' , 'rolling' , 'stone' , 'gathers' , 'momentum.')
```

выдает

```
S =
A
rolling
stone
gathers
momentum.
```

Присутствует достаточное количество пробелов в первых четырех строках, чтобы все строки были равной длины. Другой способ – это сохранить текст в массиве ячеек.

```
C = {'A' ; 'rolling' ; 'stone' ; 'gathers' ; 'momentum.'  }
```

будет массив ячеек 1x5

```
C =
    'A'
    'rolling'
    'stone'
    'gathers'
    'momentum.'
```

Вы можете преобразовать заполненный символьный массив в массив ячеек из строк следующим образом

```
C = cellstr(S)
```

Обратное преобразование

```
S = char(C)
```

Структуры

Структуры – это многомерные массивы MATLAB с элементами, доступ к которым осуществляется через поля. Например,

```
S.name = 'Ed Plum';
S.score = 83;
S.grade = 'B+';
```

создает скалярную структуру с тремя полями

```
S =
     name: 'Ed Plum'
    score: 83
    grade: 'B+'
```

Как и всё в MATLAB, структуры являются массивами, поэтому вы можете добавлять в них элементы. В этом случае каждый элемент массива является структурой с несколькими полями. Поля могут добавляться либо по одному

```
S(2).name = 'Toni Miller';
S(2).score = 91;
S(2).grade = 'A-' ;
```

либо полностью

```
S(3) = struct( 'name', 'Jerry Garcia', . . .
                 'score', 70, 'grade', 'C' )
```

Сейчас структура стала достаточной большой, поэтому печатается лишь её сводка.

```
S =
1x3 struct array with fields:
    name
    score
    grade
```

Есть несколько способов перетранслировать различные поля в другие массивы MATLAB. Все они базируются на записи списка, разделенного запятыми. Если вы наберете

S.score

- это будет равносильно следующему

S(1).score, S(2).score, S(3).score

Это и есть список, разделенный запятыми. Правда, без другой пунктуации он не очень полезен. В этой строке происходит присваивание трех счетов (score) переменной по умолчанию *ans* и вывод результатов каждого присваивания. Но если вы включаете выражение к квадратные скобки,

```
[S.score]
```

то же самое, что

```
[S(1).score, S(2).score, S(3).score]
```

результатом будет численный вектор-строка, содержащий все счета (score)

```
ans =
    83    91    70
```

Аналогично,

```
S.name
```

просто присваивает имена (names), по одному, переменной *ans*. Однако, заключение этого выражения в круглые скобки

```
{S.name}
```

создает массив ячеек 1x3, содержащий три имени (names)

```
ans =
    'Ed Plum'    'Toni Miller'    'Jerry Garcia'
```

И функция

```
char(S.name)
```

с тремя аргументами создает массив символов из поля name.

```
ans =
Ed Plum
Toni Miller
Jerry Garcia
```

Сценарии и функции

MATLAB – это мощный язык программирования, также как и интерактивная вычислительная среда. Файлы, которые содержат код на языке MATLAB, называются М-файлами. Вы создаете М-файлы, используя текстовой редактор, а затем используете их как любую функцию или команду MATLAB.

Существует два вида М-файлов

- Сценарии, которые не имеют входных и выходных аргументов. Они оперируют с данными из рабочего пространства.
- Функции, которые имеют входные и выходные аргументы. Они оперируют с локальными переменными.

Если вы являетесь новичком в MATLAB программировании, просто создавайте М-файлы, которые вы хотите использовать, в текущей директории. Если же вы разработали много М-файлов, вы захотите сгруппировать их в отдельные директории и персональные пакеты программ (toolboxes). Для этого вам необходимо добавить их маршрут поиска MATLAB.

Если вы повторяете имя функции, то MATLAB вызывают только ту, которая встречается первой.

Чтобы увидеть содержание М-файла, например, myfunction.m необходимо набрать

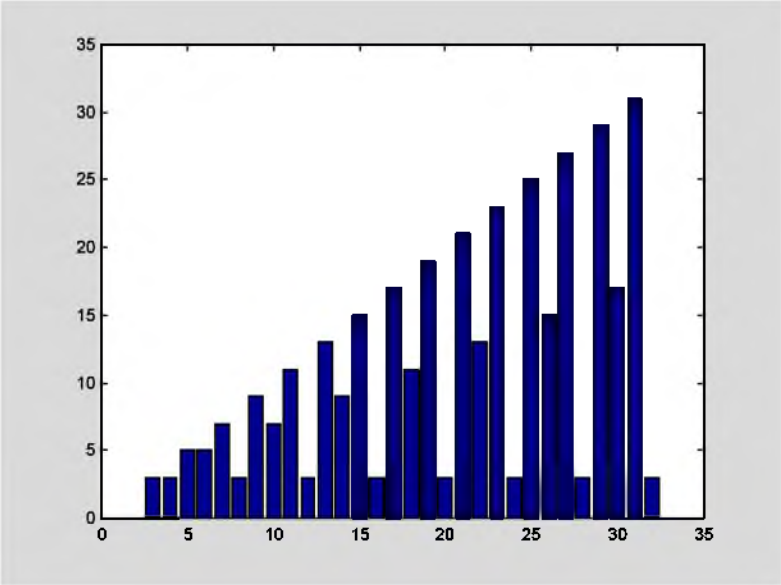
```
type myfunction
```

Сценарии

Кода вы вызываете сценарий, MATLAB просто вызывает команды, содержащиеся в файле. Сценарии могут оперировать существующими данными в рабочем пространстве или они могут сами создавать эти данные. Хотя сценарии не возвращают значений, все переменные, которые они создают, остаются в рабочем пространстве для использования в последующих вычислениях. В добавление к сказанному, сценарии могут осуществлять графический вывод, используя такие функции как *plot*.

В качестве примера, создадим файл magicrank.m, который содержит эти команды MATLAB:

```
% Investigate the rank of magic squares
r = zeros(1,32);
for n = 3:32
  r(n) = rank(magic(n));
end
r
bar(r)
```



Ввод строки

magicrank.m

повлечет за собой исполнение команд, вычисление ранга первых 30 магических квадратов и отображения столбиковой диаграммы результатов. После полного выполнения файла переменные n и r остаются в рабочем пространстве.

Функции

Функции – это М-файлы, которые могут иметь входные и выходные возвращать. Имя М-файла и функции должно быть одним и тем же. Функции работают с переменными в пределах их собственного рабочего пространства, отделенного от рабочего пространства, с которым вы оперируете в командной строке MATLAB.

Хорошим примером является функции *rank*. М-файл *rank.m* находится в директории

toolbox/matlab/matfun

Вы можете просмотреть его содержание, введя

```
type rank
```

```
function r = rank(A,tol)
%RANK   Matrix rank.
%   RANK(A) provides an estimate of the number of linearly
%   independent rows or columns of a matrix A.
%   RANK(A,tol) is the number of singular values of A
%   that are larger than tol.
%   RANK(A) uses the default tol = max(size(A)) * norm(A) * eps.

%   Copyright (c) 1984-98 by The MathWorks, Inc.
%   $Revision: 5.7 $  $Date: 1997/11/21 23:38:49 $
```

```
s = svd(A);
if nargin==1
   tol = max(size(A)') * max(s) * eps;
end
r = sum(s > tol);
```

Первая строка функции М-файла начинается со слова *function*. Здесь происходит задание имени со списком аргументов. В нашем случае, используется до двух входных аргументов и один выходной.

Следующие несколько строк, до первой пустой или выполняемой строки, являются комментариями, которые предоставляют справочную информацию. Эти строки будут выведены на экран, если вы наберете

help rank

Первая строка справочного текста – это H1 строка, которую MATLAB отображает при использовании команды *lookfor* или при запросе *help* по всей директории.

Остальное содержание файла составляет исполняемый код MATLAB. Переменная s, представленная в теле функции, также как и переменные в первой строке, r, A и tol, все являются локальными. Они отделены от других переменных в рабочем пространстве MATLAB.

Этот пример показывает важную особенность функций MATLAB, которая обычно не встечается в других языках програмирования, - переменное число аргументов. Функция *rank* может быть использована в нескольких различных формах:

rank(A)
r = rank(A)
r = rank(A, 1.e-6)

Многие функции MATLAB работают таким образом. Если нет выходного аргумента, то результат сохраняется в переменной *ans*. Если нет второго входного аргумента, то функция вычисляет значение по умолчанию. Внутри тела функции присутствуют две величины *nargin* и *nargout*, которые выдают число входных и выходных аргументов при каждом использовании функции. Функция *rank* использует переменную *nargin* , но не использует *nargout*.

Глобальные переменные

Если вы хотите, чтобы более одной функции использовали отдельную копию переменной, просто объявите её как *global* во всех функциях. Делайте то же самое в командной строке, если вы хотите, чтобы основное рабочее пространство получило доступ к переменной. Определение *global* должно быть до самой переменной, используемой в функции. Хотя это не обязательно, использование больших букв для имени глобальной переменной поможет отличить их от других переменных. Например, создадим М-файл falling.m:

```
function h = falling(t)
global GRAVITY
h = ½*GRAVITY*t.^2;
```

Затем введем следующие строки

```
global GRAVITY
GRAVITY = 32;
y = falling((0: .1: 5)' );
```

Таким образом, строки определения GRAVITY в командной строке делают её доступной внутри функции. Вы можете после изменить GRAVITY и получить новое решение, не редактируя какие-либо файлы.

Командно-функциональная двойственность.

Примеры команд MATLAB – это

```
load
help
```

Многие команды имеют управляющий параметр, который определяет последующее действие.

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

Другой метод использования командных параметров – это создание строки аргументов функций.

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

Это и есть "командно-функциональная двойственность". Любая команда типа

```
command argument
```

также может быть переписана в функциональной форме

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

Преимущество функционального подхода проявляется, когда строка аргумента создается из отдельных частей. Следующий пример обрабатывает многочисленные файлы с данными: August1.dat, August2.dat и т.д. Он использует функцию *int2str*, которая преобразует целые числа в строку символов для создания имени файла.

```
for d = 1:31
   s = [ 'August'  int2str(n)  '.dat']
   load(s)
   % Обработка содержания d-го файла
end
```

Функция eval

Функция *eval* работает с текстовыми переменными для вычисления и реализации текстовых строк.

```
eval(s)
```

использует интерпретатор MATLAB для вычисления и выполнения выражения, содержащегося в текстовой строке s.

Пример из предыдущего раздела может быть также реализован следующим образом (хотя это будет менее эффективно, т.к. используется полный интерпретатор, а не вызов функции).

```
for d = 1:31
   s = [ 'load August'  int2char(n)  '.dat' ]
   eval(s)
   % Обработка содержания d-го файла
end
```

Векторизация

Чтобы добиться максимальной скорости вне MATLAB, очень важно векторизовывать алгоритм в M-файлах. Там где другие языки программирования могут использовать циклы for или do, MATLAB может применять векторные или матричные операции. Простым примером является создание таблицы логарифмов.

```
x = 0
for k = 1:1001
   y(k) = log10(x);
   x = x + .01;
end
```

(Опытные пользователи MATLAB любят говорить "Жизнь слишком коротка чтобы тратить время на запись циклов")

А векторизованная версия этого кода выглядит следующим образом

```
x = 0: .10:10;
y = log10(x);
```

Для более сложных программ возможности векторизации не так очевидны. Однако, когда важна скорость, вы должны всегда искать способы векторизации вашего алгоритма.

Предварительное выделение

Если вы не можете векторизовать часть кода, вы можете заставить ваш цикл *for* работать быстрее. Для этого нужно предварительно выделить вектора или массивы, в которых будут храниться выходные результаты. Например, следующий код использует функция *zeros* для предварительного выделения вектора, создаваемого в цикле *for*. Это позволяет циклу *for* работать заметно быстрее.

```
r = zeros(32,1)
for n = 1:32
   r(n) = rank(magic(n));
end
```

Без предварительного выделения в предыдущем примере интерпретатор MATLAB увеличивает вектор r по одному элементу каждый раз внутри цикла. Предварительное выделение вектора устраняет это действие, и результат получается быстрее.

Функция от функций

Класс функций, называемый "функция от функций", работает с нелинейными функциями скалярных переменных. То есть одна функция работает с другой . Функции от функций включают в себя

- Нахождения нуля
- Оптимизация
- Интегрирование
- Обыкновенные дифференциальные уравнения

MATLAB представляет нелинейные функции через М-файлы. Например, ниже приведена упрощенная версия функции *humps* из директории *matlab/demos*

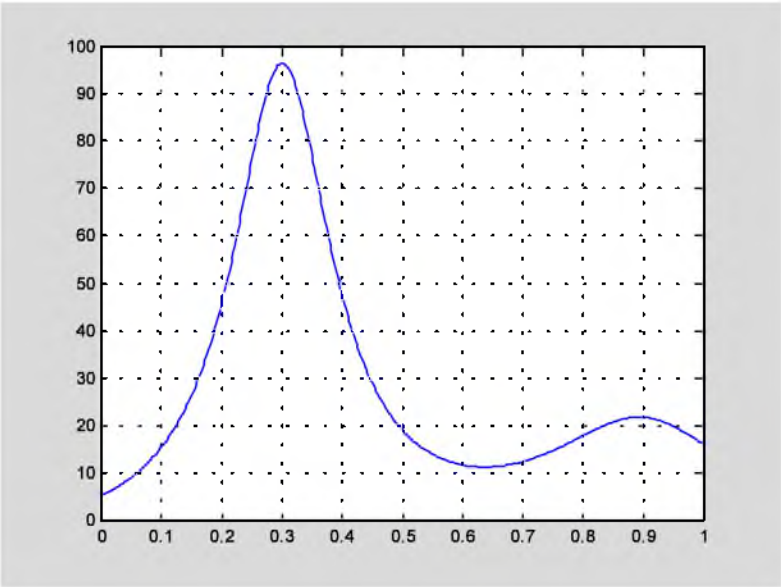
```
function y = humps(x)
y = 1. / ( (x - .3). ^2 + .01) + 1. / ( (x - .9) .^2 + .04) - 6;
```

Вычислим эту функцию в нескольких точках интервала [0,1]

```
x = 0: .002:1;
y = humps(x);
```

Затем построим график

```
plot(x,y)
```



На графике видно, что эта функция имеет локальный минимум около x=0.6. Функция *fmins* найдет минимумы, т.е. те значения x, в которых функция достигает своего минимума. Первым аргументом *fmins* является имя функции, минимум которой ищется, а вторым – приближенное положение минимума

```
p = fmins( 'humps', .5)

p =
    0.6370
```

Чтобы найти значение функции в минимуме,

```
humps(p)

ans =
   11.2528
```

Специалисты используют термины квадратура и интегрирование, чтобы различать численную аппроксимацию определенных интегралов и численное интегрирование обыкновенных дифференциальных уравнений. *quad* и *quad8* –это подпрограммы MATLAB по вычислению квадратуры.

```
Q = quad8( 'humps', 0, 1)
```

вычисляет площадь под кривой на графике и выдает

```
Q =
   29.8583
```

В заключение, на графике видно, что заданная функция не имеет нулей на этом интервале. Поэтому, если мы попытаемся отыскать нуль

```
z = fzero( 'humps', .5)
```

то мы найдем его вне нашего интервала

```
z =
   -0.1316
```

Управляемая графика

MATLAB предоставляет комплекс функций низкого уровня, которые позволяют создавать и обрабатывать линии, поверхности и другие графические объекты. Эта система называется управляемая графика (Handle Graphics®).

Графические объекты

Графические объекты – это базисные элементы системы управляемой графики в MATLAB. Они сформированы в дерево структурной иерархии. Этим отражается связь графических объектов. Например, объекты *Line* (линия) нуждаются в объектах *Axes* (оси) как в системе отсчета. В свою очередь объекты *Axes* существуют только с объектами *Figure*.

Есть одиннадцать видов объектов управляемой графики:

- Объекты *Root* являются вершиной иерархии. Они соответствуют экрану компьютера. MATLAB автоматически их создает вначале сеанса работы.
- Объекты *Figure* – это окна на экране, кроме командного окна.
- Объекты *Uicontrol* – это пользовательское управление интерфейсом. Когда пользователь активирует объект, вызывается соответствующая функция. Они включают в себя *pushbutton*, *radio button* и *slider*.
- Объекты *Axes* определяют область в окне *Figure* и ориентацию дочерних объектов в этой области.
- Объекты *Uimenu* представляют собой меню пользовательского интерфейса, которое расположено в верхней части окна *Figure*.
- Объекты *Image* – это двумерные объекты, которые выводит MATLAB, используя элементы прямоугольного массива как индексы в палитре.
- Объекты *Line* являются основными графическими базисными элементами для большинства двумерных графиков.
- Объекты *Surface* – это трехмерное представление данных матрицы, созданное путем графического отображения данных как высот над плоскостью x-y.
- Объекты *Text* – это строки символов.
- Объекты *Light* определяют источник света, действующий на все объекты в пределах *Axes*.

Управление объектами

Каждый отдельный графический объект имеет свой уникальный идентификатор, называемый handle (манипулятор), который MATLAB присваивает объекту при создании. Некоторые графики, например с несколькими кривыми, состоят из многих объектов, каждый из которых имеет свой собственный идентификатор (handle). Чем пытаться прочитать их с экрана и повторно вводить, вы увидите, что всегда лучше хранить значение в переменной и использовать его по необходимости.

Идентификатор объекта *root* всегда нуль. Идентификатор *figure* – это целое число, которое по умолчанию отображается в заголовке окна. Идентификаторы других объектов являются числа с плавающей точкой, которые содержат ин-

формацию, используемую MATLAB. Например, если A – это магический квадрат Дюрера, то

```
h = plot(A)
```

создаст линейный график с четырьмя линиями, для каждого столбца A, а также возвратит вектор-идентификатор, такой как

```
h =
    3.0022
    1.0038
    4.0020
    5.0016
```

Данные численные значения являются несущественными и могут изменяться от одной системы к другой. Какие бы числа не были, важным является то, что h(1) – это идентификатор первой кривой на графике, h(2) – второй и т.д.

MATLAB имеет несколько функций для получения доступа к часто используемым объектам:

- gcf
- gca
- gco

Вы можете использовать эти функции в качестве входных аргументов других функций, оперирующих с идентификаторами *figure* и *axes*. Получить идентификаторы других объектов можно во время их создания. Все функции MATLAB, которые создают объекты, возвращают идентификатор (или вектор идентификаторов) созданного объекта.

Для удаления объекта служит функция *delete*, использующая идентификатор объекта в качестве аргумента. Например, удалим текущие оси (*axes*), а вместе с ними и все дочерние объекты.

```
delete(gca)
```

Функции создания объектов

Вызов функции с именем какого-либо объекта создает этот объект. Например, функция *text* создает объект *text*, функция *figure* создает объект *figure* и т.д. Графические функции MATLAB высокого уровня (такие как *plot* и *surface*) вызывают необходимые функции низкого уровня для отображения соответствующих графиков.

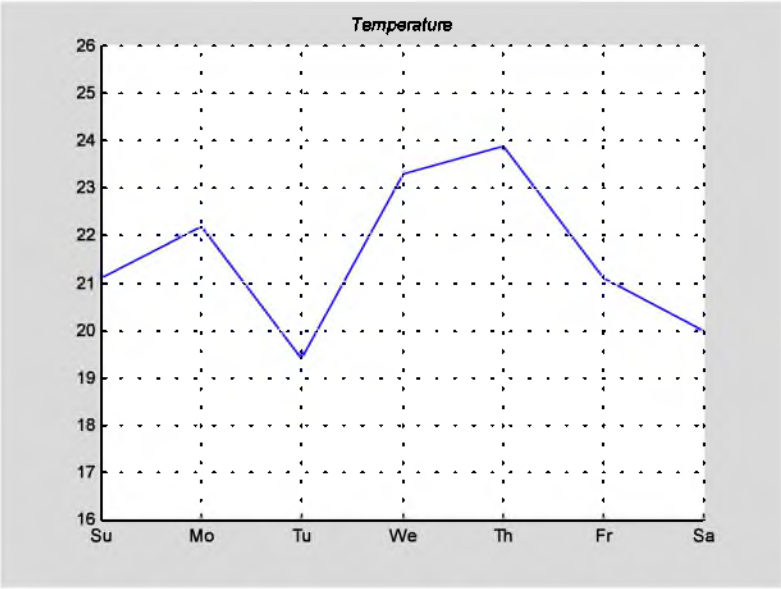
Функции низкого уровня просто создают один из одиннадцати графических объектов определенных MATLAB, за исключением корневого объекта (root), который создает только MATLAB. Например,

```
line( [1 3 6], [8 –2 0], 'Color', 'red' )
```

Свойства объекта

Все объекты обладают свойствами, которые определяют, как они выводятся на экран. MATLAB предоставляет два механизма для задания свойств. Свойства объекта могут быть установлены функцией создания объекта или могут быть изменены функцией *set*, когда объект уже существует. Например, следующий элемент кода создает три объекта и заменяет некоторые из их свойств по умолчанию.

```
days = [ 'Su' ; 'Mo' ; 'Tu' ; 'We' ; 'Th' ; 'Fr' ; 'Sa' ]
temp = [ 21.1  22.2  19.4  23.3  23.9  21.1  20.0 ];
f = figure;
a = axes( 'YLim' , [16 26] , 'Xtick' , 1:7 , 'XtickLabel' , days )
h = line(1:7, temp)
```



days – это массив символов, содержащий сокращения дней недели, а temp – это численный массив с типичной температурой. Окно изображений создается после вызова *figure* без аргументов, т.е. со значениями по умолчанию. Оси существуют внутри *figure* и имеют заданный диапазон по y и заданные метки для приращений по x. Линии существуют внутри осей и имеют заданные значения для данных по x и y. Три идентификатора объектов f, a и h сохранены для дальнейшего использования.

set и get

Свойства объекта задаются обращением к нему после его создания. Для этого используйте идентификатор, возвращаемый создаваемой функцией.

Функция *set* позволяет устанавливать свойства объектов, указанием идентификатора объекта и совокупности пар название свойства / значение. В качестве упражнения изменим цвет и ширину линии из предыдущего примера.

```
set( h , 'Color' , [0 .8 .8] , 'LineWidth' , 3)
```

Для того, чтобы увидеть список всех доступных свойств заданного объекта, вызовете *set* с идентификатором объекта.

```
set(h)

   Color
   EraseMode: [ {normal} | background | xor | none ]
   LineStyle: [ {-} | -- | : | -. | none ]
   LineWidth
   Marker: [ + | o | * | . | x | square | diamond | v | ^ | > | < |
pentagram | hexagram | {none} ]
   MarkerSize
   . . .
   XData
   YData
   Zdata
   . . .
   .
```

Чтобы вывести список всех текущих установленных свойств заданного объекта, вызовете get с идентификатором объекта

```
get(h)

   Color = [0 0.8 0.8]
   EraseMode = normal
   LineStyle = -
   LineWidth = [3]
   Marker = none
   MarkerSize = [6]
   . . .
   XData = [ (1 by 7) double array]
   YData = [ (1 by 7) double array]
   ZData = []
   . . .
```

Для запроса значения отдельного свойства используйте get с именем свойства.

```
get(h , 'Color')

ans =
         0    0.8000    0.8000
```

Объекты axes имеют много детальных свойств для всего графика. Например, заголовок - title.

```
t = get(a , 'title' );
set(t , 'String' , 'Temperature' , 'FontAngle' , 'oblique')
```

Функция *title* обеспечивает другой интерфейс к этим же свойствам.

Графический Пользовательский Интерфейс (GUI)

Ниже приведен пример иллюстрирующий использование управляемой графики (Handle Graphics) для создания пользовательского интерфейса.

```
b = uicontrol( 'Style' , 'pushbutton' , . . .
          'Units' , 'normalized' , . . .
          'Position' , [.5 .5 .2 .1] , . . .
```

'String' , 'click here');

создает *pushbutton* в центре окна изображения (*figure*) и возвращает идентификатор нового объекта. Однако пока, нажатие на эту кнопку ни к чему не приводит.

s = ' set(b , ' 'Position' ' , [.8*rand .9*rand .2 .1]) ';

создает строку, содержащую команду, которая меняет положение кнопки. Повторное использование

eval(s)

будет передвигать кнопку в случайные места. Окончательно,

set(b , 'Callback' , s)

установит s в качестве обработки нажатия кнопки. Поэтому каждый раз, когда вы её нажимаете, она перемещается на новое место.

Анимация

MATLAB предоставляет несколько способов для создания двигающейся, анимационной графики. Использование свойства *EraseMode* предназначено для длинной последовательности простых графиков, где изменение от кадра к кадру минимально. Ниже представлен пример, моделирующий броуновское движение. Определим количество точек

```
n = 20;
```

Температуру или скорость как

```
s = .02;
```

Лучшие значения этих параметров зависят от скорости вашей машины. Сгенерируем n случайных точек с координатами (x,y) между -½ и ½.

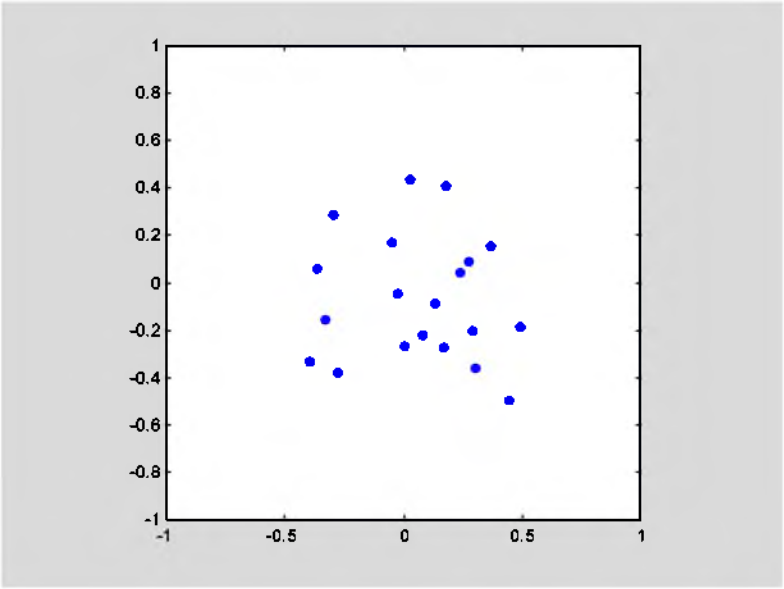
```
x = rand(n,1)-0.5;
y = rand(n,1)-0.5;
```

Отобразим точки в квадрате со сторонами в пределах от –1 до 1. Сохраним идентификатор для вектора точек и установим свойство *EraseMode* равным *xor*. Это укажет графической системе MATLAB не перерисовывать весь график, когда изменяется координата одной точки, а восстанавливать цвет фона в окрестности точки, используя операцию исключающего ИЛИ.

```
h = plot(x , y , ' . ' );
axis([-1 1 -1 1])
axis square
grid off
set(h , 'EraseMode' , 'xor' , 'MarkerSize' , 18 );
```

```
while 1
   x = x + s*randn(n,1);
   y = y + s*randn(n,1);
   set(h, 'Xdata' , x , 'Ydata' , y)
end
```

Сколько нужно времени, чтобы одна из точек вышла за пределы квадрата? Через какое время все точки окажутся вне квадрата?



Movie

Если вы увеличиваете число точек в броуновском движении, например, n = 300, то в этом случае движение не будет очень подвижным, так как потребуется слишком много времени на каждом шаге. Становится более эффективным сохранить определенное число кадров как bitmap и проигрывать их как кино.

Во-первых, пусть число кадров

```
nframes = 50;
```

Во-вторых, установим первый график как и ранее, за исключением использования *EraseMode*

```
x = rand(n,1)-0.5;
y = rand(n,1)-0.5;
h = plot(x, y, ' . ' )
set(h, 'MarkerSize' , 18)
axis([-1 1 -1 1])
axis square
grid off
```

Теперь выделим достаточно памяти для сохранения нашего фильма.

```
M = moviein(nframes);
```

Это установит большую матрицу с nframes столбцами. Каждый столбец достаточно длинный, чтобы сохранить один кадр. Полное количество требуемой памяти пропорционально количеству кадров и области текущих осей, но она не зависит от сложности отдельных графиков. Для 50 кадров и осей по умолчанию необходимо около 7.5 мегабайт памяти. В нашем примере используются квадратные оси, которые немного меньше, поэтому требуется около 6 мегабайт.

Сгенерируем полученный фильм и используя getframe, сохраним каждый кадр.

```
for k = 1:nframes
   x = x + s*rand(n,1);
   y = y + s*rand(n,1);
   set(h , 'XData' , x , 'YData' , y)
   M(: , k) = getframe;
end
```

Теперь проиграем фильм 30 раз

```
movie(30)
```

Дополнение

Чтобы увидеть дополнительные примеры MATLAB, выберите **Examples and Demos** из меню, или наберите

```
demo
```

в командной строке MATLAB. Выберите из меню примеры, которые вас заинтересовали, и следуйте инструкциям на экране.

Для более детального объяснения отдельных тем, затронутых в этой книге, обращайтесь к следующей литературе:

- *MATLAB Installation Guide*, описывает, как установить MATLAB на вашу платформу
- *Using MATLAB* представляет более глубокий материал по языку MATLAB, рабочей среде и математическим темам.
- *Using MATLAB Graphics*, описывает использование графики и инструментов для визуализации MATLAB.
- *MATLAB Application Program Interface Guide* объясняет, как писать программы на Си и Фортране, которые взаимодействуют с MATLAB.
- *MATLAB 5.1 New Features Guide* предоставляет полезную информацию для перехода от MATLAB 4.x к MATLAB 5.1.

MATLAB Toolboxes – это группы M-файлов, которые расширяют возможности MATLAB в различных технических областях. Доступны отдельные руководства по каждой из них. Вот некоторые из тем, которые они охватывают:

Связь

Системы контроля

Финансовые вычисления

Идентификация в частотной области

Нечеткая логика

Спектральный анализ высокого порядка

Обработка изображений

Линейные матричные неравенства

Управление с эталонной моделью

μ – анализ и синтез

Численные алгоритмы

Нейронные сети

Уравнения в частных производных

Проектирование робастных систем с обратной связью

Робастное управление

Обработка сигналов

Моделирование (Simulink)

Сплайны

Статистика

Символьная математика

Идентификация систем

Вэйвлеты

Для получения последней информации о MATLAB и других продуктах MathWorks обращайтесь к

http://www.mathworks.com

и используйте ваш Internet New reader для доступа к группе новостей

comp.soft-sys.matlab

Книги о MATLAB выпускают различные издания. Брошюра, озаглавленная *MATLAB-Based Books,* может быть получена от компании MathWorks, а последний список доступен на Web сайте.

Если вы прочитали всю эту книгу и запустили все примеры, поздравляем, вы начали работу с MATLAB просто великолепно. Если вы пропустили некоторые разделы или не попробовали отдельные примеры, мы советуем вам потратить еще немного времени на изучение *Getting Started.* В любом случае добро пожаловать в мир MATLAB !

Предметный указатель

Перевод с английского Конюшенко В.В.

Конюшенко Валерий Вячеславович
18/05/1977
тел. 460-30-82
e-mail: konushenko@afrodita.phys.msu.su