МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Курбатова Н.В., Пустовалова О.Г.

Основы MatLab в примерах и задачах

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

К ПРАКТИКУМУ ПО КУРСУ

«Пакеты компьютерной алгебры»

Ростов-на-Дону

2017

Учебно-методическое пособие «Основы MatLab в примерах и задачах» разработано кандидатом физико-математических наук, доцентом кафедры математического моделирования Н. В. Курбатовой и кандидатом физико-математических наук, доцентом кафедры математического моделирования О. Г. Пустоваловой.

Печатается в соответствии с решением кафедры математического моделирования института математики, механики и компьютерных наук ЮФУ (протокол № 6 от 13 июня 2017 г.).

Учебно-методическое пособие предназначено для поддержки практикума курса «Пакеты компьютерной по специальности «Прикладная математика и информатика» алгебры» с использованием пакета MatLab. Данные курс и практикум являются центральным учебным материалом, который позволяет системно усвоить теоретические положения курсов линейной алгебры, математического анализа, расширить их практические приложения и научиться применять возможности MatLab (ML) для оперативного решения прикладных задач.

В пособии предлагается подход от *простого к сложному*, который не исчерпывается сквозным эволюционным изложением материала, а предполагает реструктурирование сложных задач на простые в рамках тематического раздела. Такие *простые задачи* составляют содержание серии примеров, поддержанных программной реализацией средствами языка ML.

Оглавление

Введение 5

Первое знакомство с MatLab (ML) 7

Интерфейс MatLab. Command Windows (CW) 7

Интерфейс MatLab. Workspace 9

Интерфейс MatLab. Help 10

Простые операции с векторами и матрицами 12

Ввод векторов и матриц 12

Обращение к элементам матрицы 13

Удаление элементов матрицы 14

Некоторые специальные матрицы 15

Создание матриц, заполненных случайными числами 16

Поэлементные операции 17

Матричные операции 18

Часто используемые матричные функции 22

Логические операции с матрицами 26

Задачи для самостоятельного решения 27

Графика в MatLab 31

Построение графиков функций 31

Несколько графиков в одном графическом окне 32

Установка параметров графиков 33

Построение графика неявно заданной функции 35

Создание нескольких графических окон 37

Использование логарифмической шкалы 39

Задания для самостоятельного решения 41
Основные типы данных 46
Методы класса Array 46
Типы данных Numeric и Double 47
Способы создания объектов Double 48
Задания для самостоятельного решения 53
Объекты класса Char. Функции и свойства 56
Объекты класса Cell. Функции и свойства 61
Создание функций в Matlab 63
Функции и процедуры 63
Аноним и функция-строка 66
Подпроцедуры 67
Литература 69

Введение

Предлагаемое учебно-методическое пособие ориентировано на поддержку практикума по курсу «Пакеты компьютерной алгебры» по специальности «Прикладная математика и информатика» в части освоения инструментария пакета MatLab для решения широкого спектра задач, являющихся основой компьютерного моделирования.

Предполагается владение базовыми знаниями по курсам программирования, математического анализа и линейной алгебры.

Данные курс и практикум относятся к обязательным дисциплинам (цикла - вариативная часть) и являются центральным учебным материалом, который дает возможность научиться оперировать функционалом предлагаемого ресурса для эффективного решения задач, в том числе смежных предметных областей. Многомодульная структура MatLab (ML) имеет все предпосылки стать основным инструментом исследования, как в учебном, так и научно-исследовательском процессе, а предлагаемое пособие обеспечивает освоение его основ. Оперативный и интерактивный характер взаимодействия пользователя и системы ML позволяет экстраполировать функционал пакета на решение прикладных, оптимизационных задач и других задач более высокого уровня.

Авторы предлагают эволюционный подход от *простого к сложному*, и он не сводится к сквозному усложнению предлагаемого материала, а опирается на решения простых тематических задач и заданий-тренажеров. Такие *простые задачи* составляют содержание серии примеров, поддержанных программной реализацией средствами языка ML.

Предлагаемый подход обеспечивает базу для решения более сложных задач и индивидуальных работ, охватывают темы или даже разделы курса.

Пакет MatLab является замечательным примером взаимодействия пользователя (студента) и компьютерной системы. Облегченный синтаксис,

нестрогая типизация, проработанные библиотеки с функциями эффективного программирования облегчают использование ML для решения задач, возникающих при изучении таких курсов, например, как численные методы, математические модели естественных наук, стохастическое моделирование. Данное учебно-методическое пособие может быть весьма полезно для решения научно-технических задач, возникающих при работе над курсовыми и дипломными проектами.

Для более глубокого и всестороннего освоения методик интерактивного и программного моделирования средствами пакета ML предлагаем следующую литературу [1-6].

Материал пособия ориентирован на использование программного пакета MatLab версии 7.x -11.x.

Первое знакомство с MatLab (ML)

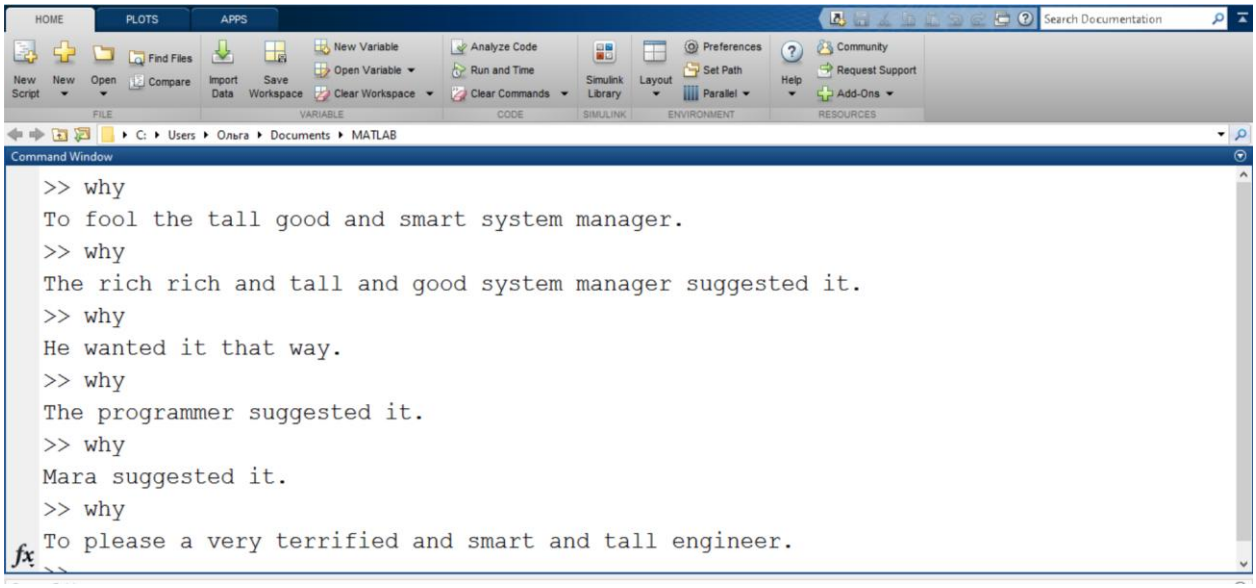
Пакет ориентирован на интерактивное (суперкалькулятор) и программное функционирование (MatLab – высокоуровневый язык на базе FORTRAN с оптимизацией на C, C++).

В пакете по умолчанию реализована комплексная арифметика, вычисления производятся с двойной точностью, базовый элемент – массив.

Пакет снабжен удобным интерфейсом - окнами, отличающимися своей функциональностью. Конфигурирование необходимых для пользователя окон осуществляется в меню команд так: Desktop (с выбором необходимых окон) или Desktop Layout →Default (по умолчанию). Остановимся на некоторых из них.

Интерфейс MatLab. Command Windows (CW)

При интерактивной работе в командном окне все команды и их последовательности помещаются в строку ввода, она начинается символом >>. Исполняются команды после нажатия клавиши Enter. А отделяются команды друг от друга запятой или точкой запятой. Если использовать разделитель точку с запятой, то результат выполнения команды не отображается.



На рисунке, представленном выше, представлены результаты исполнения команды `why` в командном окне. Читателю предлагается поставить собственный опыт - выполнить несколько раз данную команду и сравнить результаты ее исполнения.

Выполненные команды помещаются в стек и могут быть извлечены в строку ввода перебором исполненных команд с помощью стрелок ↑↓ и при необходимости редактируются при повторном исполнении. Строка вывода не доступна для редактирования.

Все переменные среды – глобальные. Это может стать причиной ошибок, если какие-то переменные уже ранее были определены и их используют повторно. Поэтому необходимо внимательно контролировать процесс идентификации и использования переменных. Пример, приведенный ниже, показывает, как можно отобразить на экране переменные и очистить некоторые из них или сразу все.

Пример 1. Контроль переменных, сохранение и очистка CW

```
who% Идентификаторы всех переменных
whos% Идентификаторы и типы всех переменных

% Удаление всех переменных
clear
% Удаление конкретных переменных, например, x и y
clear x и y
% Сохранить все переменные оперативной памяти в системном двоичном файле matlab.mat
save
% Сохранить переменные x,y,z  в двоичном файле variables.mat
save variables  x y z;
% Очистить содержимое оперативной памяти (все переменные - глобальные)
Clc
```

Заметим, что командное окно является с одной стороны средой для вычислений, а с другой стороны графическим объектом, тип которого -

структура. Такой дуализм CW, двойственность, сохраняется и в управлении его свойствами.

Пример 2. Свойства CW как функции высокого уровня

```
% Задание формата, который поддерживает 15 цифр после запятой
format long % short – по умолчанию  и 4 цифры после запятой
% Задание формата рациональных чисел
format rational
% Выполнение скрипт-файла с отображением каждой исполняемой %строки
echo on % echo off – по умолчанию
```

Пример 3. Свойства CW как графического объекта Root с нулевым дескриптором

```
% Определить текущие свойства CW
get(0)

% Определить свойства CW текущие и возможные
set(0)

%Задание формата, который поддерживает 15 цифр после запятой
set(0,'format','long')

% Выполнение скрипт-файла с отображением каждой исполняемой строки
set(0,' echo',' on')
```

Интерфейс MatLab. Workspace

Workspace – рабочее пространство; окно, содержащее информацию обо всех переменных, типе, значениях. Щелчком по пиктограмме переменной активируется редактор переменных (VE), позволяющий изменять их зна-

чения в интерактивном режиме. Этот прием работы отображен на рисунке, приведенном ниже.

Интерфейс MatLab. Help

Справочная система ML содержит информацию об имеющихся модулях – Toolbox (Help→Product Help), алфавитный (Index) и содержательный (Contents) поиск по имеющемуся программному функционалу, а также демонстрационные тематические программы (Demos). Следует отметить, что политика ML обусловила такое структурирование ML, при котором каждый модуль, по сути, является обособленным и определяет исследовательскую среду для выделенной предметной области. Все они ориентированы на преимущества ML: высокую точность, векторно-матричную природу, простой синтаксис и нестрогую типизацию.

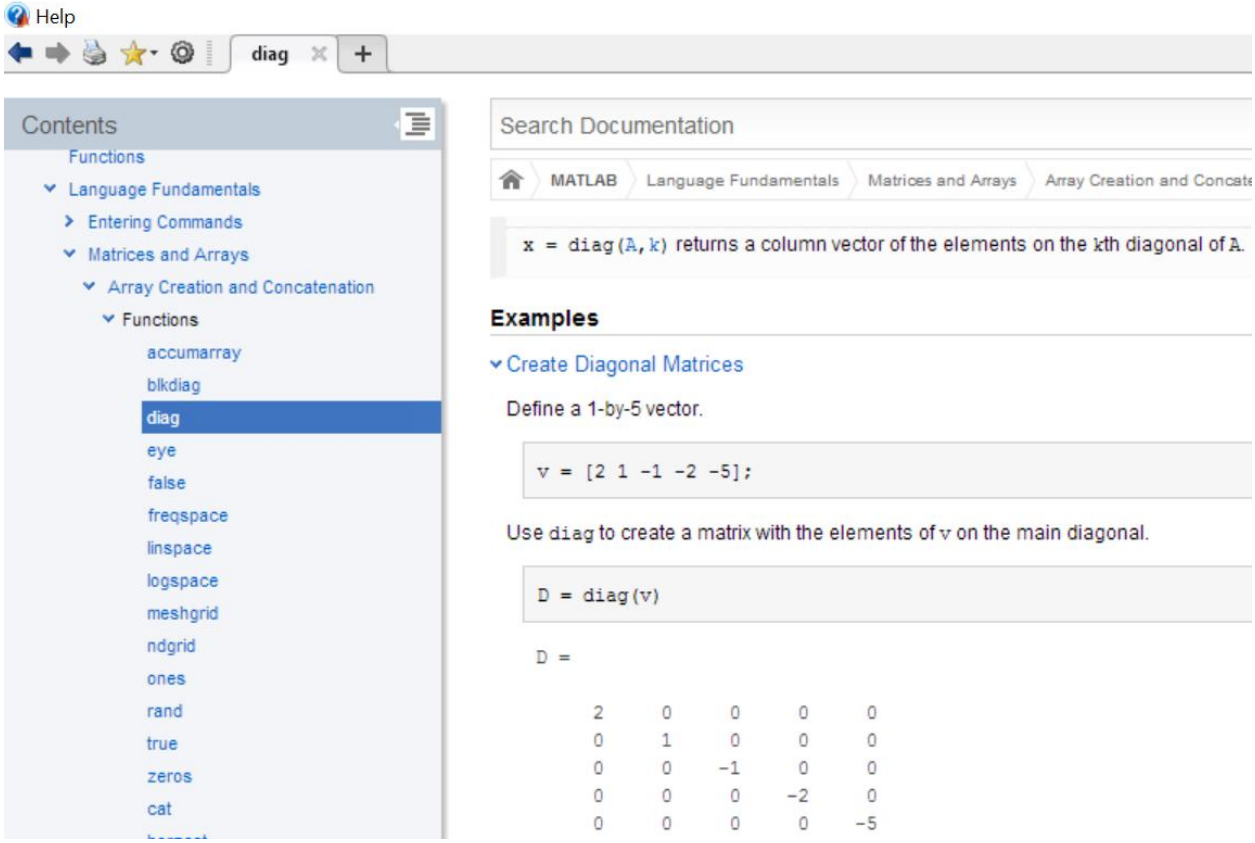
Пример 4. Тематические справочные материалы

```
% Справка по элементарным  функциям
help elfun
% Справка по элементарным  операциям (арифметическим,
операциям отношения, логическим, над  множествами
см.help)
help > %знак больше
% Справка по элементарным, специальным матрицам и сис-
темным переменным
help elmat
```

В контекстном поиске (Help→Product Help→Contents) статья Program Control Statements описывает все элементы программирования, используе-

мые в ML. В алфавитном поиске (Help→Product Help→ Index), набрав is*, можно получить справку по контролю возможных типов данных.

На рисунке, представленном ниже, отображено окно справочной системы ML. Как и во многих других приложениях, примеры справки доступны для копирования с последующим выполнением в рабочей среде.



Простые операции с векторами и матрицами

Основными объектами, с которыми начинает работать пользователь, знакомящийся с MATLAB, являются матрицы. Если проверить с помощью команды **`size`** размер числа 5, или символа 'A', то мы получим два числа - количество строк и количество столбцов, в данном случае - это две единицы. Лозунг, которым призывают руководствоваться создатели языка – 'Think vectorized', или 'Мысли векторно'.

Ввод векторов и матриц

Для ввода векторов и матриц используются квадратные скобки [ ]. Разделителями данных в векторах и матрицах служат пробел и запятая в строке, и точка с запятой - в столбце.

Пример 1. Задание векторов

```
% Вектор-строка
a1=[1 2 3]
% Вектор-строка
a2=[1,2,3]
% Вектор-столбец
a3=[1;2;3]
```

Пример 2. Задание матриц

```
% Матрица, размера 2x3
b1=[1 2 3; 4 5 6]
% Матрица, размера 3x2
b2=[1 2; 3 4; 5 6]
```

Значения вектора можно задать с помощью следующей конструкции:

`[начальное значение : шаг : конечное значение],`

или

`[начальное значение : : конечное значение],`

тогда шаг по умолчанию равен единице. Квадратные скобки в этом выражении можно опустить.

Пример 3. Задание вектора и вычисление вектора

```
% Задаем вектор x
x=0:0.01:6
% Вычисляем вектор y
y=sin(x)
```

Функцией

x = linspace(начальное значение, конечное значение),

также можно пользоваться при создании линейного массива. При этом вектор x по умолчанию будет содержать сто компонент. Возможен и другой способ вызова функции **linspace** - с тремя входными параметрами, последним из которых является количество компонент вектора.

Пример 4. Функция linspace

```
% Вектор из 100 компонент
c1=linspace(1,100)
% Вектор из 20 компонент
c2=linspace(1,100,20)
```

Обращение к элементам матрицы

Для обращения к элементам матрицы используют круглые скобки ( ). Первый индекс – номер строки, второй – номер столбца. Возможно задавать диапазон строк – столбцов.

Пример 5. Задание матрицы и обращение к ее элементам

```
% Очистка экрана
clc
% Очистка переменных
clear
% Задание матрицы
A=[1 2 3; 4 5 6; 7 8 9]
% Изменение 1-го элемента матрицы
A(1,1)=100
```

```
% Изменение 3-й строки матрицы
A(:,3)=50
% Изменение 2-го столбца матрицы
A(2,:)=33
```

В примере, представленном выше, знак двоеточие обозначает, что в рассмотрение берутся все элементы.

Пример 6. Изменение фрагмента матрицы

```
clc
% Очистка переменной A
clear A
% Задание квадратной матрицы 7-го порядка из единиц
A=ones(7)
% Изменение фрагмента матрицы
A(2:6,2:6)=55
```

Пример 7. Использование ключевого слова end

```
clear A
A=ones(7)
% Изменение фрагмента матрицы
A(4:end,4:end)=-21
```

В примере #6 значения элементов строк и столбцов со 2 по 6 заменяются на 55. В примере #7 использовано ключевое слово end для обозначения конца диапазона.

Удаление элементов матрицы

Удалить из матрицы можно строку или столбец целиком. Для удаления строки или столбца необходимо присвоить удаляемому элементу пустой массив.

Пример 8. Удаление элементов матрицы

```
% Задание матрицы
A=[5 5 5; 2 10 2; 2 10 2]
```

```
% Удаление 1-й строки матрицы
A(1,:)=[]
% Удаление 2-го столбца матрицы
A(:,2)=[]
```

Пример 9. Удаление нескольких строк матрицы

```
% Задание матрицы
A=[1 1 1; 2 2 2; 7 3 3]
% Удаление 2-х строк матрицы
A(1:2,:)=[]
% Удаление двух последних элементов матрицы
A(2:end)=[]
```

Некоторые специальные матрицы

Приведем примеры некоторых специальных и часто используемых матриц.

Пример 10. Матрица из единиц

```
% Матрица из единиц 5-го порядка
A=ones(5)
% Матрица из единиц, в которой 2 строки и 3 столбца
B=ones(2,3)
```

Пример 11. Матрица из нулей

```
% Матрица из нулей 3-го порядка
C=zeros(3)
% Матрица из нулей, в которой 2 строки и 6 столбцов
D=zeros(2,6)
```

Обычно команду zeros используют для инициализации матриц.

Пример 12. Единичная матрица

```
% Единичная матрица 3-го порядка
E=eye(3)
% Единичная матрица, в которой 3 строки и 4 столбца
F=eye(3,4)
```

Следующий пример демонстрирует команду **magic**, которая позволяет формировать матрицу Альбрехта Дюрера, или магический квадрат. Данная матрица знаменита тем, что суммы элементов в строках, столбцах и диагоналях одинаковы.

Пример 13. Магический квадрат

```
% Матрица Дюрера 3-го порядка
M3=magic(3)
% Матрица Дюрера 5-го порядка
M5=magic(5)
```

Создание матриц, заполненных случайными числами

Существует несколько функций, позволяющих заполнять матрицы случайными числами. Рассмотрим несколько примеров, реализующих это.

Пример 14. Функция rand

```
% Матрица 5-го порядка,
% заполненная вещественными случайными числами
% с равномерным распределением из открытого интервала
(0,1)
A=rand(5)
% Матрица 2x3,
A=rand([2 3])
```

Пример 15. Заполнение матрицы случайными целыми числами

```
% Используем функцию округления round
% Заполняем матрицу случайными целыми числами от 0 до
10
A=round(rand(8)*10)
% Заполняем матрицу случайными целыми числами от -5 до
5
B=round(rand(8)*10-5)
```

Пример 16.
Заполнение матрицы случайными целыми с помощью функции randi

```
% Матрица 15-го порядка с элементами в диапазоне от -20
до 20
A=randi([-20 20],15)
% Матрица 5x7 с элементами в диапазоне от -3 до 3
B=randi([-3 3],5,7)
```

Поэлементные операции

Для выполнения поэлементной арифметической операции необходимо поставить точку перед знаком операции:

A.+B A.-B A.*B A./B A.\B A.^B

Пример 17. Поэлементные операции с векторами

```
% Заполнение векторов. Вектора одинаковой длины
v1=10:10:50, v2=1:5
% Поэлементное умножение векторов
r_1=v1.*v2
% Поэлементное деление векторов
r_2=v1./v2
% Поэлементное суммирование векторов и умножение на
число
% Точку в данном случае ставить необязательно
r_3=0.1*v1+100*v2
```

Пример 18. Поэлементные операции с матрицами

```
% Заполнение матриц. Матрицы одинаковой размерности
m1=[2 4 6; 3 7 9], m2=[6 4 2; 9 7 3]
% Поэлементное умножение матриц
z_1=m1.*m2
% Поэлементное деление матриц
z_2=m2./m2
% Поэлементное суммирование матриц и умножение на число
% Точку в данном случае ставить необязательно
z_3=m1+10*m2
```

Пример 19. Поэлементное деление прямое и обратное

```
% Заполнение векторов. Векторы одинаковой размерности
q1=[1 2 3 4], q2=[10 20 30 40]
% Поэлементное прямое деление векторов
p_1=q1./q2
% Поэлементное обратное деление векторов
p_2=q1.\q2
% Заполнение матриц. Матрицы одинаковой размерности
h1=[10 20; 30 40], h2=[5 10; 15 20]
% Поэлементное прямое деление матриц
w_1=h1./h2
% Поэлементное обратное деление матриц
w_2=h1.\h2
```

Матричные операции

В MatLab определены матричные операции по правилам линейной алгебры: при сложении и вычитании должны совпадать размерности, при умножении и делении число столбцов первого матричного сомножителя и число строк второго должны совпадать. К матричным же операциям относится возведение в степень и транспонирование матрицы.

Пример 20. Матричное умножение

```
% Задание матриц
M1=[1 1 1; 2 2 2]
M2=[3 4; 3 5;3 6]
% Матричное умножение
M1*M2
% Вектор-строка из 5 элементов
M3=[1 2 3 4 5]
% Вектор-столбец из 5 элементов
M4=[1; 2; 3; 4; 5]
% Матричное умножение
M3*M4
% Задание квадратных матриц
```

```
M5=[1 2; 3 4]
M6=[1 2; 2 1]
% Матричное умножение
M5*M6
```

Обратное матричное деление используется для отыскания решения систем алгебраических линейных уравнений (СЛАУ). Если задана система вида Ax=b, где A – квадратная матрица, b – столбец свободных членов, а x – разыскиваемое решение, то в том случае, когда система совместна, x можно найти с помощью операции обратного деления.

Пример 21. Матричное обратное деление

```
clear, clc
% Задание матрицы A и столбца свободных членов b
A=[1 0 0; 0 2 0; 0 0 3]
b=[10; 40; 150]
% Решение системы Ax=b
x=A\b
% или
x=A^(-1)*b
% или
x=inv(A)*b
```

Условия для системы, приведенной в примере #21, подобраны с расчетом, что читатель найдет решение устно и проверит совпадение с решением, полученным с помощью MatLab. Описание методов решения СЛАУ в ML читатель может найти в источниках [1-6].

Для возведения квадратной матрицы в целую положительную степень, используется операция ^.

Пример 22. Возведение матрицы в степень

```
clear, clc
% Задание матрицы A
A=[1 2 3; 0 2 0; 0 0 3]
% Возведение матрицы в степень
A^2
```

Пример 23. Транспонирование вещественной матрицы

```
clear, clc
A=[1 1 1; 2 2 2; 4 5 6]
% Транспонирование матрицы
A'
% Транспонирование матрицы
A.'
```

Знак ' – обозначает операцию транспонирования с взятием комплексного сопряжения, очевидно, что для вещественных матриц эта операция сводится к обычному транспонированию, а .’ обеспечивает *простое* транспонирование, даже в случае комплексных матриц.

Пример 24. Транспонирование матрицы, содержащей комплексные элементы

```
clear, clc
% Задание матрицы A
A=[1-i 1+i; 2+3i 2-3i]
% Транспонирование матрицы с комплексными значениями
A.'
% Транспонирование матрицы и комплексное сопряжение
A'
```

При проведении операций с матрицами нужно помнить приоритет операций. Он следующий: сначала выполняется операция транспонирования, затем возведения в степень, потом умножение и деление, а в последнюю очередь – сложение.

Пример 25. Приоритет матричных операций. Транспонирование и умножение

```
clear
clc
A=[1 1; 2 2]
% Вычисление значения выражения без скобок
A*A'
```

```
% Вычисление значения выражения со скобками
(A*A)'
```

Пример 26. Приоритет матричных операций. Возведение в степень и деление

```
clear, clc
A=[1 3; 0 5]
% Вычисление значения выражения без скобок
A/A^2
% Вычисление значения выражения со скобками
(A/A)^2
```

Рассмотрим операцию объединения матриц. Она может выполняться по горизонтали для матриц, количество строк которых одинаково, и по вертикали, для матриц с одинаковым количеством столбцов, также можно объединять матрицы одинаковой размерности вдоль третьей оси. Для плоского объединения матриц используют квадратные скобки, функция **cat** объединяет матрицы вдоль трех направлений:

cat(направление, матрица_1, матрица_2,…,матрица_n)

Параметр направление может принимать значение 1, что соответствует объединению по вертикали, 2 – горизонтали, 3 – объединить вдоль третьей оси.

Пример 27. Объединение матриц по горизонтали

```
clear, clc
% Задание матриц
M1=[1 2; 3 4], M2=[5 6 7; 8 9 10]
% Объединение по горизонтали с помощью
% квадратных скобок
[M1 M2]
% Объединение по горизонтали с помощью функции cat
cat(2,M1,M2)
```

Пример 28. Объединение матриц по вертикали

```
clear, clc
% Задание матриц
```

```
M3=[1 2 3], M4=[5 6 7; 8 9 10]
% Объединение по горизонтали с помощью
% квадратных скобок
[M3; M4]
% Объединение по горизонтали с помощью функции cat
cat(1,M3,M4)
```

Пример 29. Объединение матриц вдоль третьей оси

```
clear, clc
% Задание матриц
M5=[1 2; 3 4], M6=[5 6; 8 9]
% Сложение в «стопку» с помощью функции cat
cat(3,M5,M6)
```

С помощью функции **inv** и операции возведения в степень **-1** можно найти обратную матрицу.

Пример 30. Нахождение обратной матрицы

```
A=[1 2; 0 2]
inv(A)
A^(-1)
```

Часто используемые матричные функции

Рассмотрим некоторые часто применяемые матричные функции, такие как **sum, prod, diag, fliplr, rot90, reshape, repmat, blkdiag.**

Пример 31. Сумма по столбцам

```
clear
clc
A=[1 2; 3 4]
sum(A)
```

При исполнении примера #31, получим результат – два числа 4 и 6, что соответствует суммам элементов в столбцах. Чтобы получить суммирование

по строкам, необходимо указать второй параметр в функции **sum**, а именно 2.

Пример 32. Сумма по строкам

```
clear, clc
A=[1 2; 3 4]
sum(A,2)
```

Пример 33. Сумма всех элементов матрицы

```
clear, clc
A=[1 2; 3 4]
sum(sum(A))
```

Чтобы найти произведение элементов матрицы, используйте функцию prod.

Пример 34. Произведение элементов матрицы

```
clear, clc
A=[1 2; 3 4]
prod(A)
prod(A,2)
prod(prod(A))
```

Функция **diag** позволяет выделить диагонали матрицы, если аргумент функции – матрица, либо построить матрицу с заданной диагональю, если аргумент - вектор.

Пример 35. Выделение диагоналей матрицы

```
clear, clc
A=[1 2 3; 1 2 3; 1 2 3]
% Выделение главной диагонали
diag(A)
% Выделение побочной диагонали, расположенной ниже
главной
diag(A,-1)
% Выделение побочной диагонали, расположенной выше
главной
diag(A,1)
```

Пример 36. Построение матрицы на основе заданной диагонали

```
clear, clc
d1=[1 2 3]
% элементы d1 будут располагаться на главной диагонали
diag(d1)
% элементы d1 будут располагаться ниже
% главной диагонали
diag(d1(2:3),-1)
% элементы d1 будут располагаться выше
% главной диагонали
diag(d1(2:3),1)
```

Функции **fliplr** и **rot90** позволяют отражать и поворачивать векторы и матрицы. Покажем их работу на примере.

Пример 37. Функции fliplr и rot90

```
clc, clear
d1=[1 2 3],d2=[11 12 13]
rot90(d1)
fliplr(d2)

a1=[10 2 3; 40 5 6; 70 8 9]
fliplr(a1)
a2=[10 20 30; 4 5 6; 7 8 9]
rot90(a2)
```

С помощью функции **reshape** можно изменить форму – размерность массива, количество элементов массива при этом остается неизменным.

Пример 38. Функция reshape

```
clc, clear
d=1:12
size(d)
d=reshape(d,3,4)
size(d)
d=reshape(d,4,[])
size(d)
```

```
d=reshape(d,12,1)
size(d)
```

Функция **repmat** позволяет задавать новую матрицу с помощью реплицирования (повторения) исходной матрицы в соответствии с заданной размерностью.

Пример 39. Функция repmat

```
d=1:3
d1=repmat(d,2), d2=repmat(d,2,3)
```

С помощью функции **blkdiag** выполняют построение блочно-диагональных матриц, а с помощью функции spy можно отобразить структуру матрицы, её ненулевые элементы.

Пример 40. Функция blkdiag

```
clc, clear
m1=[1 2; 3 4], m2=[10 20 30; 40 50 60],
m3=[2 4 6; 1 3 7; 5 4 3]
% Формирование блочно-диагональной матрицы
m=blkdiag(m1,m2,m3)
% Визуализация структуры матрицы
spy(m)
```

В некоторых примерах, приведенных выше, использовалась функция **size**, возвращающая количество строк и количество столбцов объекта. Функция **numel** возвращает общее количество элементов массива, а функция **length** – количество элементов вектора или строки матрицы.

Пример 41. Функции size, numel, length

```
clc, clear
v1=[1; 2; 3; 4], v2=[1 2 3 4]
size(v1), size(v2)
length(v1),length(v2)
numel(v1),numel(v2)
M=[1 2 3; 4 5 6]
size(M), length(M), numel(M)
```

Логические операции с матрицами

Для матриц определены логические операции:

Операция	Знак операции
Равно	==
Не равно	~=
Больше	>
Больше или равно	>=
Меньше	<
Меньше или равно	<=
Логическое И	&
Логическое ИЛИ	|

Пример 42. Подсчет количества элементов матрицы, равных двум

```
clc, clear
A=[1 2; 3 2]
sum(sum(A==2))
```

Пример 43. Подсчет количества одинаковых элементов в матрицах A и B и стоящих на одинаковых местах

```
clc, clear
A=[1 2; 3 2]
B=[1 0; 3 5]
sum(sum(A==B))
```

Задачи для самостоятельного решения

Вариант 1

1. Задана матрица A=randi([-5 5],3,3)
   a) Определить количество ненулевых элементов.
   b) Найти A+A', показать, что полученная матрица симметричная.
   c) Определить количество элементов, равных двум.
   d) Переставить верхнюю и нижнюю строки матрицы.
   e) Найти сумму элементов главной диагонали матрицы.

2. Заданы 5 матриц различного порядка. Создать блочно-диагональную матрицу, состоящую из заданных матриц-блоков. Отобразить структуру полученной матрицы с помощью spy.

3. Заданы матрицы одинаковой размерности:
A= randi([-5 5],3,3), B= randi([-5 5],3,3)
   a) Определить количество позиций, на которых стоят ненулевые элементы в обеих матрицах.
   b) Определить количество позиций, на которых, хотя бы в одной из матриц стоят ненулевые элементы.

4. Задан вектор x=1:9. Получить из него матрицу 3-го порядка, в каждой строке которой записаны последовательно элементы вектора.

5. Задан вектор x=1:4. Создать матрицу 4-го порядка, элементы каждой строки (столбца) матрицы являются элементами вектора.

6. Задано целое число n и целочисленный вектор. Повторить каждый элемент вектора n раз.

7. Задан вектор, в котором есть нулевые элементы. Каждый нулевой элемент заменить средним арифметическим элементов вектора.

Вариант 2

1. Задана матрица A=randi([-5 5],3,3)
   a) Определить количество нулевых элементов.
   b) Проверить – является ли матрица симметричной.
   c) Определить количество элементов, неравных двум.
   d) Переставить правый и левый столбцы матрицы.
   e) Найти сумму элементов побочной диагонали матрицы.

2. Заданы 5 матриц различного порядка. Создать блочно-диагональную матрицу из заданных матриц-блоков, блоки расположить вдоль побочной диагонали. Отобразить структуру полученной матрицы с помощью spy.

3. Заданы матрицы одинаковой размерности:
A= randi([-5 5],3,3), B= randi([-5 5],3,3)
   a) Определить количество позиций, на которых стоят элементы, равные двум в обеих матрицах.
   b) Определить количество позиций, на которых, хотя бы в одной из матриц стоят элементы, равные двум.

4. Задан вектор x=1:9. Получить из него матрицу 3-го порядка, в каждом столбце которого записаны последовательно элементы вектора.

5. Задан вектор x=1:4. Создать матрицу 4-го порядка, на диагоналях которой стояли бы элементы вектора. Если диагональ короче size(x), то заполнение начинать 1-го элемента вектора x.

6. Задано целое число n и целочисленный вектор. Повторить каждый элемент вектора n раз.

7. Задан вектор, в котором есть нулевые элементы. Каждый нулевой элемент заменить максимальным среди элементов вектора.

Вариант 3

1. Задана матрица A=randi([-5 5],3,3)
   a) Определить количество отрицательных элементов.
   b) Проверить – является ли матрица кососимметричной.
   c) Определить количество элементов, равных минус единице.
   d) Переставить местами главную и побочную диагонали матрицы.
   e) Найти сумму элементов, стоящих выше главной диагонали.

2. Заданы 5 матриц различного порядка. Создать матрицу из заданных матриц-блоков, блоки расположить в углах матрицы. Отобразить структуру полученной матрицы с помощью spy.

3. Заданы матрицы одинаковой размерности:
A= randi([-5 5],3,3), B= randi([-5 5],3,3)
   a) Определить количество позиций, на которых стоят элементы, равные максимальному значению матрицы, в обеих матрицах.
   b) Определить количество позиций, на которых, хотя бы в одной из матриц стоят максимальные элементы.

4. Задан вектор x=1:16. Получить из него матрицу 4-го порядка, в каждом столбце которого записаны последовательно элементы вектора.

5. Задан вектор x=1:3. Создать матрицу 3-го порядка, на диагоналях которой стояли бы элементы вектора. Если диагональ короче size(x), то заполнение начинать 1-го элемента вектора x.

6. Задано целое число n и целочисленный вектор. Повторить каждый элемент вектора n раз.

7. Задан вектор, в котором есть ненулевые элементы. Каждый ненулевой элемент заменить минимальным среди элементов вектора.

Вариант 4

1. Задана матрица A=randi([-5 5],3,3)
   a) Определить количество элементов, равных 3.
   b) Проверить – является ли матрица положительноопрделенной.
   c) Определить количество неотрицательных элементов.
   d) Переставить угловые элементы матрицы.
   e) Найти сумму элементов, стоящих ниже главной диагонали.

2. Заданы 5 матриц различного порядка. Создать матрицу из заданных матриц-блоков, блоки расположить вдоль 1-й строки матрицы. Отобразить структуру полученной матрицы с помощью spy.

3. Заданы матрицы одинаковой размерности:
A= randi([-5 5],3,3), B= randi([-5 5],3,3)
   a) Определить количество позиций, на которых стоят элементы, равные максимальному отрицательному элементу матрицы, в обеих матрицах.
   b) Определить количество позиций, на которых, хотя бы в одной из матриц стоят максимальные отрицательные элементы.

4. Задан вектор x=1:16. Получить из него матрицу 4-го порядка, в каждой строке которой записаны последовательно элементы вектора.

5. Задан вектор x=1:3. Создать матрицу 3-го порядка, на диагоналях которой стояли бы элементы вектора. Если диагональ короче size(x), то заполнение начинать 1-го элемента вектора x.

6. Задано целое число n и целочисленный вектор. Повторить каждый элемент вектора n раз.

7. Задан вектор, в котором есть ненулевые элементы. Каждый ненулевой элемент заменить средним значение среди элементов вектора.

Графика в MatLab

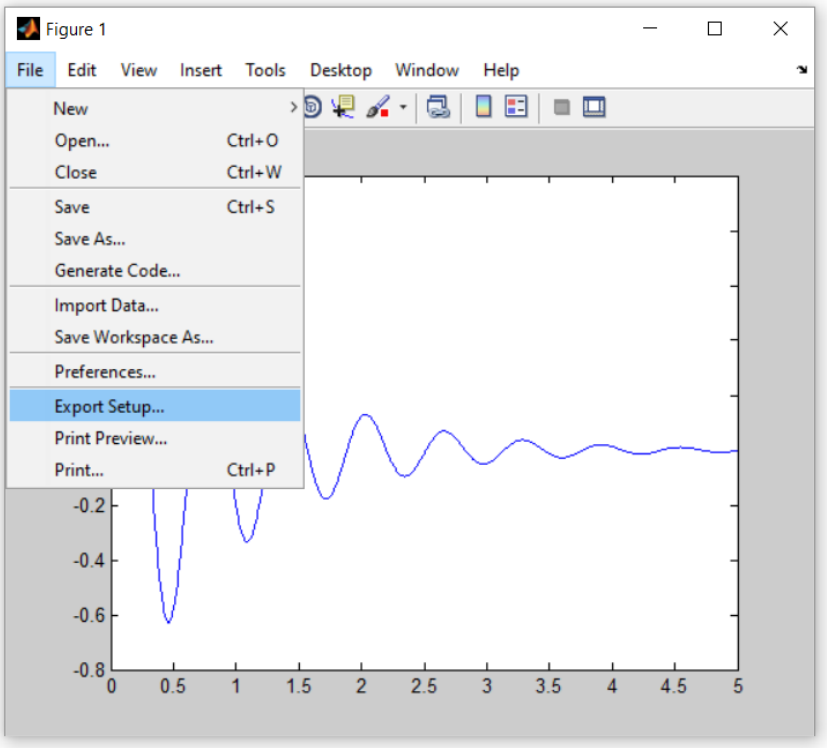
Система MATLAB предоставляет огромное количество графических средств. К ним относятся команды построения простых графиков функций, комбинированные и презентационные графики, элементы анимации и средства проектирования графического пользовательского интерфейса (GUI).

Построение графиков функций

Рассмотрим на примерах способы построения графиков в ML. Если заданы два вектора одинакового размера, хранящие координаты функции, то построить график функции можно с помощью команды **`plot`**.

Пример 1. Построение графика функции

```
% задание вектора x
x = [0:0.005:5];
% расчет значений функции
y = exp(-x).*sin(10*x);
% построение графика функции
plot(x, y)
```



Графическое окно пакета MatLab

После выполнения скрипта из примера #1 на экране возникнет графическое окно, выполняя команды меню этого графического окна File→Save as, File→Export Setup, можно сохранить построенный график в файл MatLab с расширением fig, либо экспортировать в графические форматы - png, eps, gif. Файл с расширением fig можно открыть в MatLab. Результатом открытия файла будет графическое окно.

Несколько графиков в одном графическом окне

Для того, чтобы построить еще один график в этом же графическом окне, то можно действовать двумя путями – воспользоваться командой hold on, либо добавить еще два аргумента в команду plot. С помощью функции delete производят удаление графика.

Пример 2. Два графика функции в одних осях с помощью hold on

```
x = [0:0.005:5];
y1 = exp(-x).*sin(10*x);
y2 = exp(-x).*cos(10*x);

% построение первого графика функции
plot(x, y1)
% продолжать построение в этом же окне
hold on
% построение второго графика функции
plot(x, y2)
```

Пример 3. Два графика функции в одних осях с помощью plot

```
x = [0:0.005:5];
y1 = exp(-x).*sin(10*x);
y2 = exp(-x).*cos(10*x);

% построение сразу двух графиков функций
plot(x, y1, x, y2)
```

Установка параметров графиков

Задать цвет и тип линии для графиков можно несколькими способами. Рассмотрим один из этих способов на примере.

Пример 4. Задание цвета и типа линии для графика

```
x = [0:0.005:5];
y = exp(-x).*sin(10*x);
plot(x, y,'r:')
```

Обозначения для типа линий, цветов и маркеров приведены в таблице ниже.

Цвет	
Y	Желтый
M	Розовый
C	Голубой
R	Красный
G	Зеленый
B	Синий
W	Белый
K	Черный
Линия	
-	Сплошная
:	Пунктирная
-.	штрих-пунктирная
--	Штриховая

Маркер	
.	Точка
o	Кружок
x	Крестик
+	знак "плюс"
*	звездочка
s	Квадрат
d	Ромб
v	Треугольник вершиной вниз
^	Треугольник вершиной вверх
<	Треугольник вершиной влево
>	треугольник вершиной вправо
P	пятиконечная звезда
H	шестиконечная звезда

Иногда необходимо создать несколько графических окон одновременно. В этом случае нужно использовать команду **`figure`** без параметров, или задать в качестве параметра целое число, соответствующее номеру окна.

Заголовок для графического окна задается с помощью функции **`title`**, ее параметром является текстовая строка.

Если заголовок используется не так часто - обычно его заменяет подрисуночная подпись, то использование легенды - **legend** всегда целесообразно, когда в одних осях отображается более одного графика.

Легенда позволяет определить взаимное соответствие графиков и функций. Для каждого из графиков в качестве параметров функции **legend** используют текстовую строку, описывающую соответствующий график. Последний необязательный аргумент определяет положение легенды в графическом окне. Также положение легенды можно менять уже после построения графика с помощью перетаскивания.

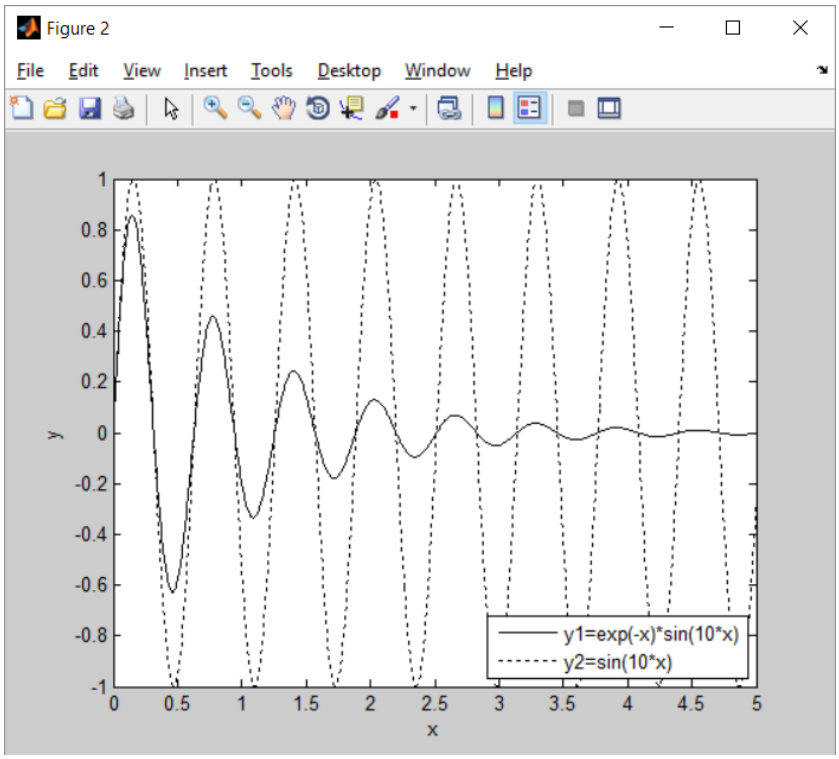
Расположение легенды	
-1	вне графика в правом верхнем углу графического окна
0	лучшее положение в пределах графика так, чтобы как можно меньше перекрывать сами графики
1	в верхнем правом углу графика (это положение используется по умолчанию)
2	в верхнем левом углу графика
3	в нижнем левом углу графика
4	в нижнем правом углу графика

Поименование осей осуществляется посредством функций **xlabel** и **ylabel**. Обратите внимание на то, что дополнительные параметры для графических осей устанавливаются, когда оси уже созданы, то есть после команды **plot**.

Пример 5. Использование легенды и подписи осей

```
% создание графического окна
figure
x = [0:0.005:5];
y1 = exp(-x).*sin(10*x);
y2 = sin(10*x);
% построение графиков двух функций
plot(x, y1,'k-',x,y2, 'k:')
```

```
% создание легенды
legend('y1=exp(-x)*sin(10*x)','y2=sin(10*x)',4)

% создание подписей к осям
xlabel('x')
ylabel('y')
```



Использование легенды и подписи осей.

Построение графика неявно заданной функции

Кроме рассмотренной функции plot есть и другие, выполняющие построение графиков. Например, с помощью **ezplot** можно построить график, как явной функции, так и неявно заданной функции. По умолчанию функция **ezplot** выполняет построение графика на отрезке [-2Pi, 2Pi]. В качестве первого аргумента этой функции необходимо задать текстовую строку – выражение для функции. Вторым аргументом может быть отрезок, на котором выполняется построение функции.

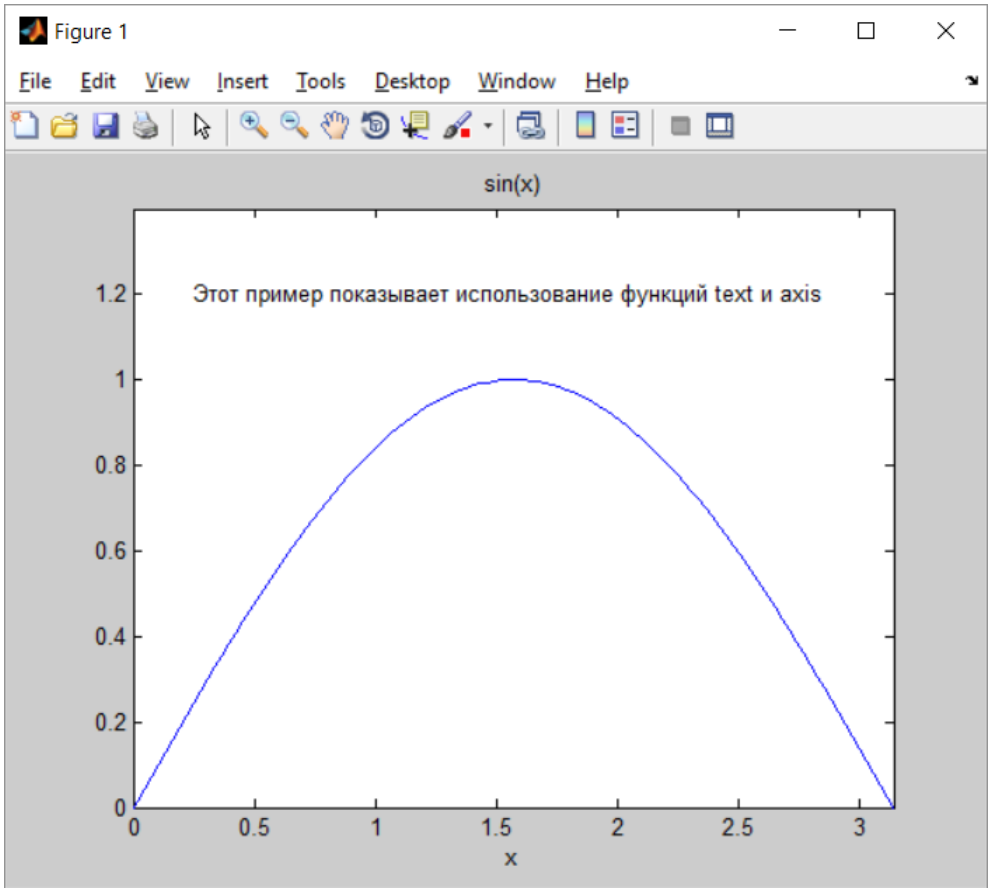
Пример 6. Использование функции ezplot

```
% построение графика сердечной функции, заданной неявно
ezplot('x^2+(y-abs(x)^(1/2))^2=1')
% построение графика функции, заданной явно
% с заданием отрезка, на котором выполняется построение
ezplot('sin(x)',[0 pi])
```

Задать максимальное и минимальное значение для осей можно с помощью функции

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

Это бывает необходимо, когда нужно расширить область осей, например, для расположения поясняющего текста. В примере 7 можно увидеть использование функций **`axis`** и **`text`**. Последняя Функция позволяет поместить в графические оси текстовую надпись. Ее аргументами являются координаты начала размещения текстовой строки и сама надпись.



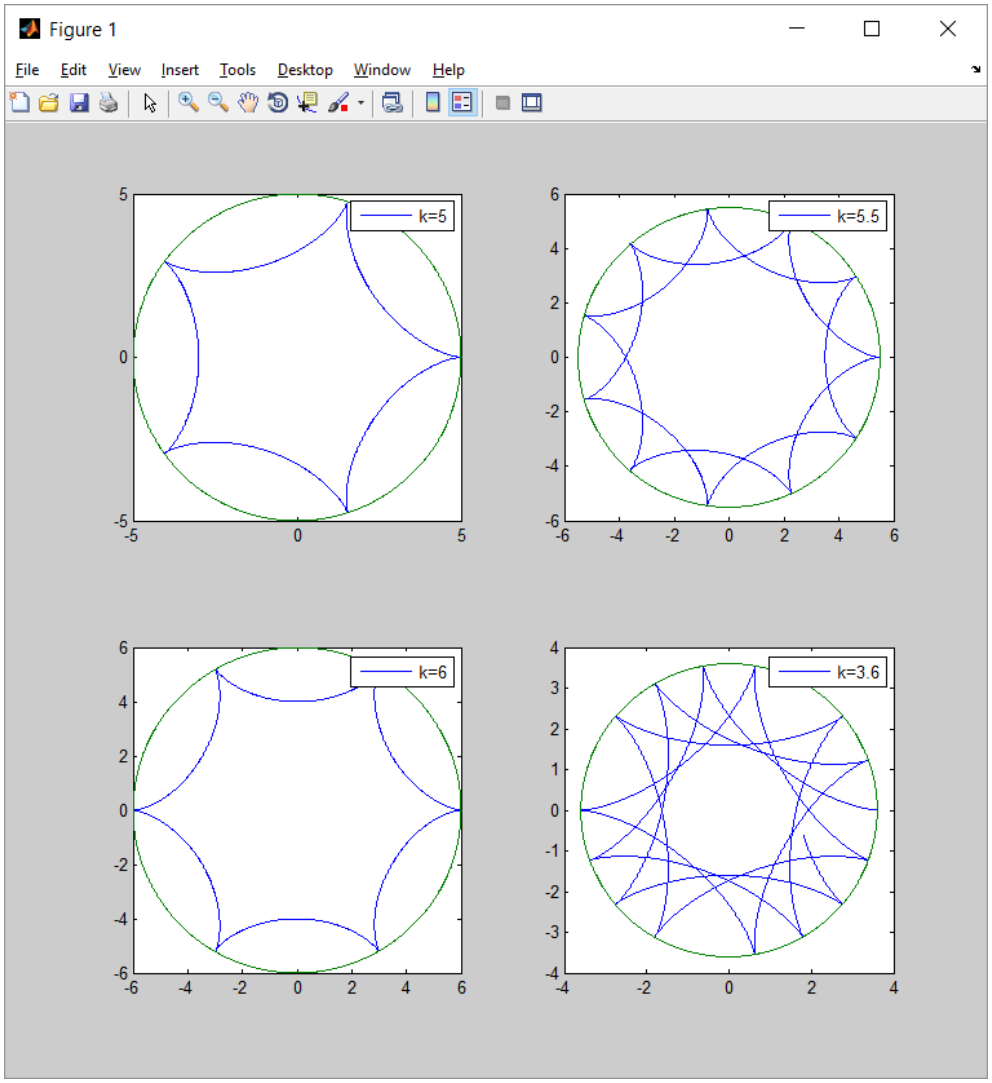
Использование функций axis и text.

Пример 7. Использование функций axis и text

```
ezplot('sin(x)')
axis([0 pi 0 1.4])
text(0.25, 1.2,…
'Этот пример показывает использование функций text и
axis')
```

Создание нескольких графических окон

Пример #8 соответствует случаю, когда в одном окне **figure** размещается несколько графических окон для визуализации графиков функций (а, вообще говоря, любой графической информации).



Использование функции subplot.

Структура таких окон и выбор активного окна осуществляются процедурой **subplot**. Первый и второй аргументы **subplot** задают матричную структуру окон-осей, их количество по строкам и столбцам, а третий аргумент – номер активных осей. Здесь выбираются графики функций, заданных параметрически.

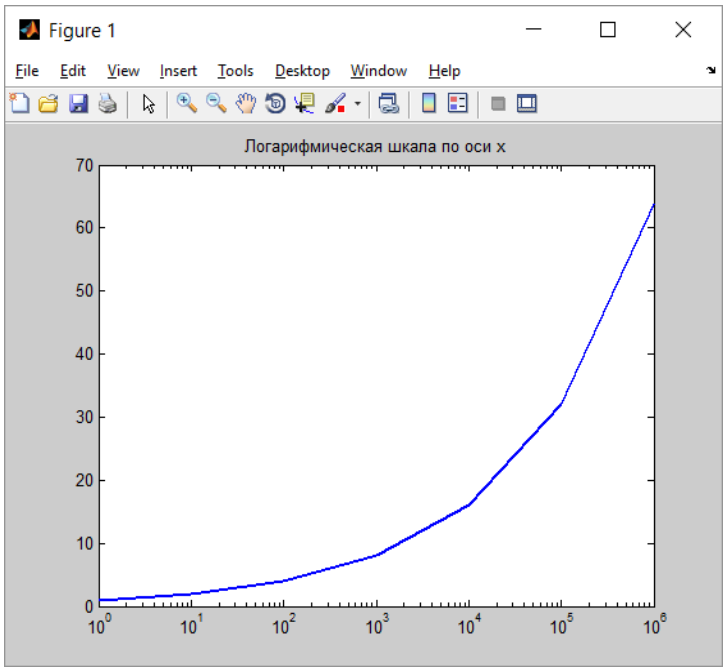
Пример 8. Использование функции subplot

```
% Гипоциклоиды
clear, clc
% Задание вектора-параметра t
t=0:0.001:8*pi;
% Данные и вектора для 1-го окна subplot
k=5;
x11=(k-1)*(cos(t)+cos((k-1)*t)/(k-1));
y11=(k-1)*(sin(t)-sin((k-1)*t)/(k-1));
x12=k*cos(t);
y12=k*sin(t);
% Данные и вектора для 2-го окна subplot
k=5.5;
x21=(k-1)*(cos(t)+cos((k-1)*t)/(k-1));
y21=(k-1)*(sin(t)-sin((k-1)*t)/(k-1));
x22=k*cos(t);
y22=k*sin(t);
% Данные и вектора для 3-го окна subplot
k=6;
x31=(k-1)*(cos(t)+cos((k-1)*t)/(k-1));
y31=(k-1)*(sin(t)-sin((k-1)*t)/(k-1));
x32=k*cos(t);
y32=k*sin(t);
% Данные и вектора для 4-го окна subplot
k=3.6;
x41=(k-1)*(cos(t)+cos((k-1)*t)/(k-1));
y41=(k-1)*(sin(t)-sin((k-1)*t)/(k-1));
x42=k*cos(t);
y42=k*sin(t);

% Построение графиков в 1-м окне subplot
```

```
subplot(2,2,1)
plot(x11,y11,x12,y12)
legend('k=5')
% Построение графиков во 2-м окне subplot
subplot(2,2,2)
plot(x21,y21,x22,y22)
legend('k=5.5')
% Построение графиков в 3-м окне subplot
subplot(2,2,3)
plot(x31,y31,x32,y32)
legend('k=6')
% Построение графиков в 4-м окне subplot
subplot(2,2,4)
plot(x41,y41,x42,y42)
legend('k=3.6')
```

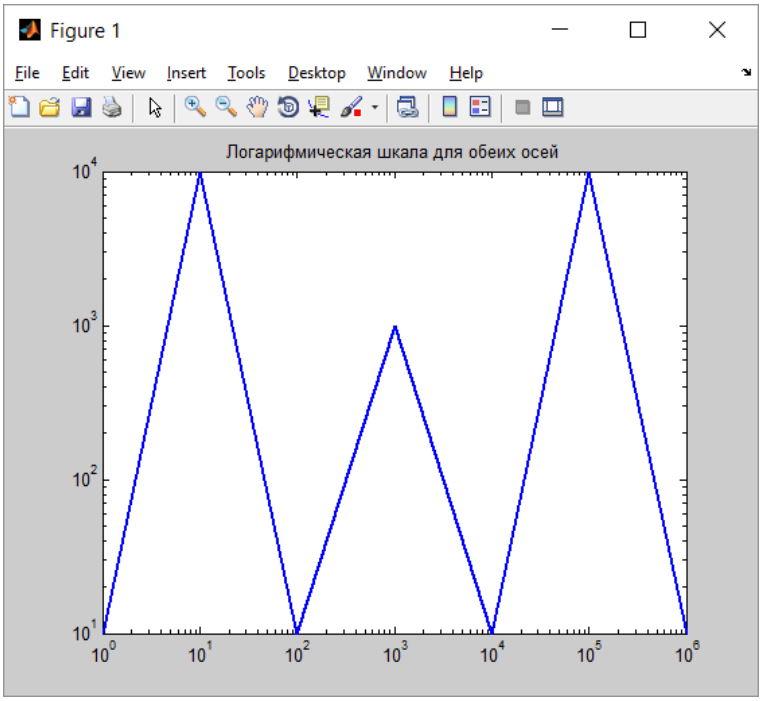
Использование логарифмической шкалы

Следующие примеры охватывают случаи, когда для одной или обеих осей стоит выбрать логарифмический масштаб, для больших диапазонов значений или для зависимостей логарифмического типа. Функции, которые строят графики в логарифмическом масштабе, это: **semilogx, semilogy и loglog**.



Пример 9. Использование логарифмической шкалы для оси x

```
clear, clc
x=[1e+0,1e+1,1e+2,1e+3,1e+4,1e+5,1e+6]
y=[1,2,4,8,16,32,64];
p=semilogx(x,y), set(p,'LineWidth',2);
title('Логарифмическая шкала по оси x')
```



Пример 10. Использование логарифмической шкалы для обеих осей

```
clear, clc
x=[1e+0,1e+1,1e+2,1e+3,1e+4,1e+5,1e+6]
y=[1e+1,1e+4,1e+1,1e+3,1e+1,1e+4,1e+1];
p=loglog(x,y)
set(p,'LineWidth',2);
title('Логарифмическая шкала для обеих осей')
grid on % задание сетки
```

Задания для самостоятельного решения

1. Постройте график функции y = exp(-x)*sin(10*x). Задайте тип и цвет линии. Добавьте заголовок.

2. Постройте графики функций в одних осях.
y1 = exp(-x).*sin(10*x)
y2 = sin(10*x)
Добавьте легенду. Попробуйте перенести легенду с помощью мыши в другое место окна figure. Задайте параметр расположения легенды вне графика - в правом верхнем углу графического окна.

3. Постройте графики функций:
y1 = sin(x)
y2 = cos(x)
Подпишите оси. Добавьте сетку.

4. Постройте графики функций в одном графическом окне, один под другим.
y1 = sin(x)
y2 = cos(x)

5. Постройте четыре графика функций, каждый в своем окне, в одном figure
y1=cos(x)
y2=sin(x)
y3=x^(1/2)
y4=x^2

6. Постройте график функции y = sin(x). Добавьте подписи координатных осей, сетку, команду задания границ для осей. Добавьте подпись '← sin(x)' в точке (3.05,0.16).

7. Для функции n! постройте график функции в логарифмическом масштабе по оси y.

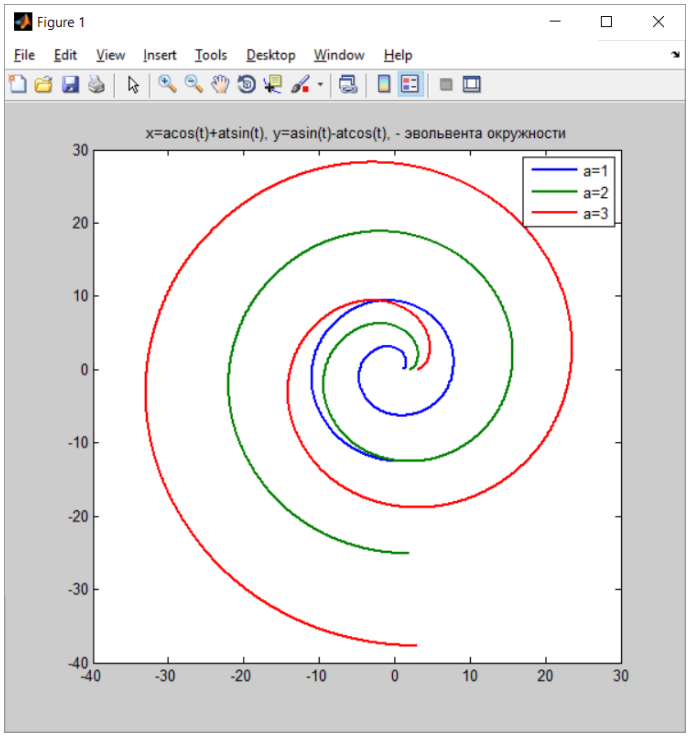
8. Постройте графики функций
f = log(0.5*x)
g = sin(log(x))
в логарифмическом масштабе по оси x. Добавьте легенду на графики.

9. Постройте графики функции y=exp(x) в обычном и логарифмическом масштабе в одном figure, но в разных осях (используйте subplot). Добавьте легенду на графики.

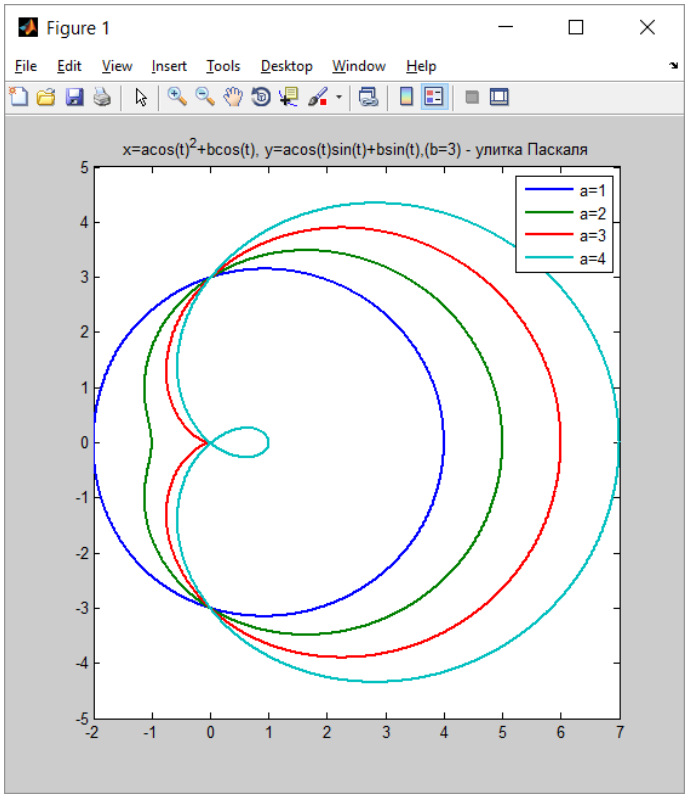
10. Постройте график функции в полярных координатах:
x=[0:0.01:2*pi]
f =8*sin(x)
r = cos(2*x)

11. Постройте в полярных координатах следующие кривые: окружность, спираль Архимеда, сердце, бабочка, полярная роза.

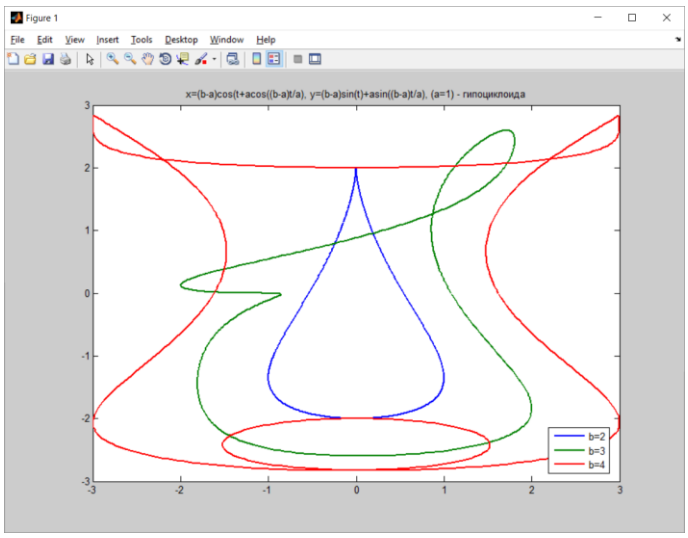
12. Постройте графики функций, заданных параметрически. Диапазон изменения параметра подберите таким образом, чтобы построенные графики были похожи на графики, приведенные на рисунках ниже. Установите толщину линии графиков в 2 пикселя. Добавьте заголовок и легенду. Сохраните построенные графики в различных графических форматах - png, jpg, eps. Сравните качество и размер получившихся графических файлов. Сделайте вывод о том, какой графический формат предпочтительнее для графиков и почему.



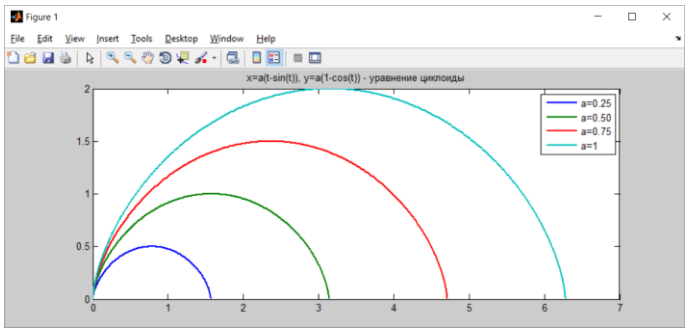
Эвольвента окружности



Улитка Паскаля



Гипоциклоида



Циклоида

13. Постройте кривую «локон Аньези» для нескольких значений параметра.

14. Постройте график функции x*sin(1/x).

15. Задайте три вектора. Первый вектор содержит номера дней месяца. Второй и третий список содержат значения курса акций в эти дни, для двух разных месяцев. Построить графики зависимости курса акций от номера дня. Добавить заголовок, подписи осей, сетку и легенду, цвета для линий графиков.

16. Дана таблица. Отобразите данные в графическом виде.

Количество	**Эксперимент 1**	**Эксперимент 2**	**Эксперимент 3**
10^3	0.02755529361166	0.04623873215974	0.06307710110520
10^4	0.28666151628344	0.67483306464728	0.85628458136542
10^5	2.84742674641796	4.92785226951551	6.29249360982561
10^6	28.5048610591205	48.4353558734875	60.7404539333518

17. Постройте график зависимости времени возведения матрицы в квадрат от ее порядка. Матрицу наполнять случайными целыми числами в диапазоне от 1 до 10. Построить такие же графики для верхней и нижней треугольных матриц. Все графики строить в одних осях. Добавить сетку и легенду. Сделать вывод.

18. Постройте график функции sin(x). С помощью панели инструментов окна figure измените толщину линии графика, добавьте текстовую надпись. Выполните экспорт графика в файл png.

19. Постройте график функции sin(x)/x. С помощью функции gtext добавьте текстовую надпись.

Контрольные вопросы

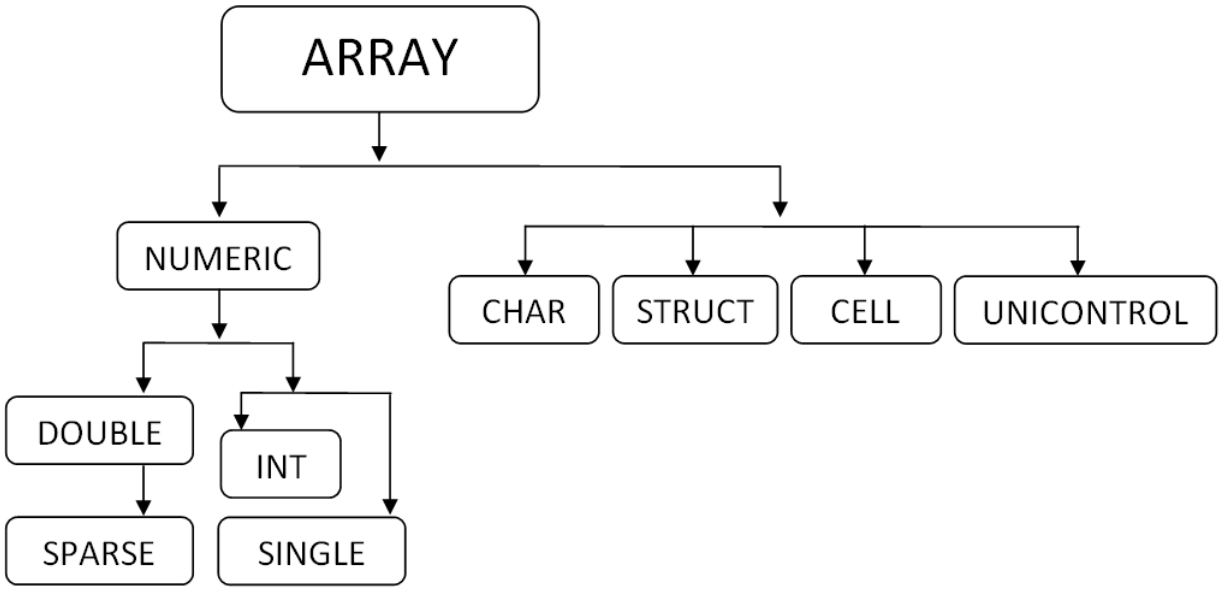
1. Перечислите способы построения графиков функций.
2. Как построить график функции, заданной параметрически?
3. С помощью какой команды можно отобразить сетку в координатных осях?
4. Как построить несколько графиков в одних осях с помощью одной команды (с помощью нескольких команд)?
5. С помощью какой команды можно вывести текстовую надпись в графическое окно?
6. Для чего используют логарифмический масштаб для графиков. Приведите примеры.
7. Что такое легенда и когда ее используют?
8. Перечислите параметры функции subplot. Для чего используют эту функцию?
9. Чем отличаются команды plot и fplot?
10. Для чего строят графики?

Основные типы данных

Любой объект в ML, в том числе скаляр, является массивом. Класс ARRAY – законодатель класса массивов, в котором разработаны методы функционирования всех дочерних объектов.

Хранение массивов в ML осуществляется в векторной форме последовательно по столбцам, поэтому поддерживается как двойная (матричная) нумерация, так одинарная (векторная). Основой операций с массивами является согласование размерностей. Заметим, что класс ARRAY является родительским для всех потомков и определяет все объекты более низкого уровня как массивы (матрицы) так и правила (методы).

На рисунке представлена схема иерархии классов (типов данных) в ML.



Методы класса Array

`ndims` – возвращает количество размерностей многомерного массива (любой природы), `n = ndims(a);`

`size` – определяет вектор `v = size(a)` размерностей массива;

так, для n = 2, `[m, p] = size(a);`

`length(a)` – определяет длину вдоль большей размерности;

`length(a(:))` определяет длину массива, записанного вектором;

`numel(a)` – количество элементов массива;

`disp(display)` – визуализация объектов, не подавляется знаком (;)

Типы данных Numeric и Double

Все объекты в ML делятся на числовые, если `isnumeric(a) = 1` равно логической единице, и нечисловые, если `isnumeric = 0`.

В момент создания числового объекта идентифицируется класс потомка Numeric, которому он принадлежит. ML ориентирован на матричные вычисления двойной точности с элементами класса Double.

Пример 1. Логическая единица не является числовым объектом

```
% Справка по элементарным  функциям
clear
a=rand(2,3);
% генерирование матрицы из двух строк, трех столбцов, элементы которой равномерно распределены на отрезке [0,1]
b=isnumeric(a)
% результат: 1 , т.е. a – числовой объект
islogical(b)
% результат: 1, т.е. b – логический объект
isnumeric(ans)
% результат: 0, т.е. логический объект - нечисловой
```

В силу свойств наследования имеем:

• скаляр – тоже массив, минимальный элемент размера (1,1);

• в памяти матрица хранится как вектор, записанный последовательно по столбцам и поддерживается одинарная и двойная нумерация;

Заметим, что диапазон вещественных чисел `[realmin, realmax]`, здесь `realmin, realmax` системные переменные минимальное и максимальное вещественные значения.

Способы создания объектов Double

Пример 2. Задание объектов перечислением:

```
% вектор-столбец
a = [1; 2; 3];
% задание вектор-строки
a = [1 2 3]
% задание вектор-строки, иной синтаксис
a = [1,2,3]
% задание вектора с элементами  арифметической прогрессии (первый член, шаг, последний)
a = [a1 : aStep:aEnd]
```

Пример 3. Задание объектов с помощью специальных матриц:

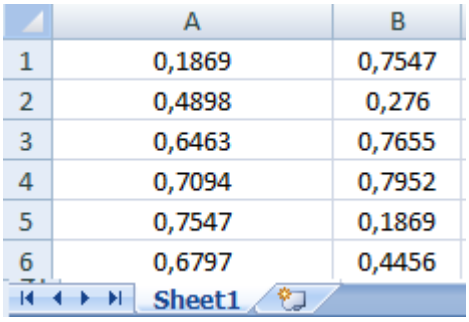
```
a = rand(size(b)) % - матрица  с элементами, полученными генератором случайных чисел,  равномерно распределенных на отрезке [0,1], такого же размера, как некоторая матрица  b

a = randn(m, n) % - размер генерируемой матрицы  m на n; используется генератор нормального распределения с нулевым средним и единичной дисперсией.
a = ones(n)% - создается квадратная матрица из единиц размера  n
b = eye(n) % -    единичная матрица
a = zeros(n) % -   нулевая матрица
```

Задание объектов импортированием можно производить с помощью непосредственного импортирования и чтением из внешних файлов:

- команды File →ImportData,
- чтение экселевского файла:
  `namemtrix=xlsread(pathname.filename)` , здесь `pathname.filename` – строка (путь и имя файла);

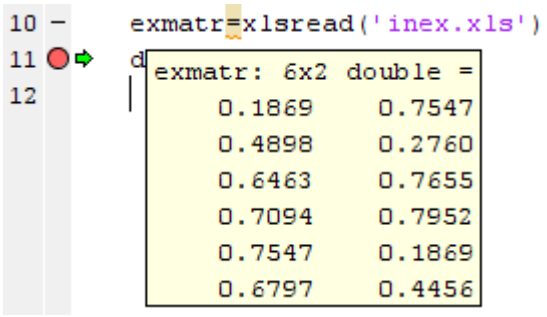
Заметим, что имена листов следует писать на английском языке, файлы не должны нарушать структуру матриц; создается файл по имени выходной переменной `namematrix`.

Ниже, на рисунке, приведен пример чтения данных из файла Excel и результат чтения в ML:



	A	B
1	0,1869	0,7547
2	0,4898	0,276
3	0,6463	0,7655
4	0,7094	0,7952
5	0,7547	0,1869
6	0,6797	0,4456

Sheet1



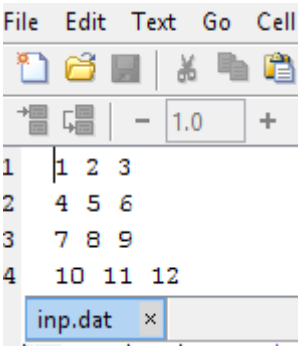
```
10 –    exmatr=xlsread('inex.xls')
11      d
12
```

```
exmatr: 6x2 double =
    0.1869    0.7547
    0.4898    0.2760
    0.6463    0.7655
    0.7094    0.7952
    0.7547    0.1869
    0.6797    0.4456
```

Файл `inex.xls`
Имя листа Sheet1

- импортирование текстового файла

`load filename` (текстовый файл, см. пример, пусть `filename` это `inp.dat`)

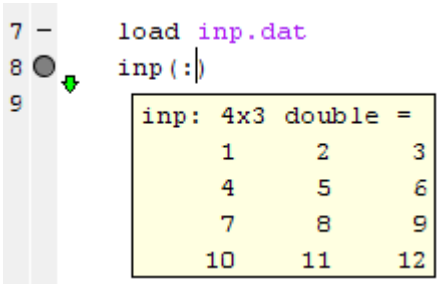


File Edit Text Go Cell

1.0

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

inp.dat ×



```
7 –     load inp.dat
8       inp(:)
9
```

```
inp: 4x3 double =
     1     2     3
     4     5     6
     7     8     9
    10    11    12
```

- форматированный ввод

сначала открывается файл `pathname.filename`, записанный строкой в качестве первого аргумента:

```
fid = fopen('pathname.filename', 'permission')
```

второй аргумент, `'permission'` – указывает на форму доступа (`'r'` – чтение из файла; `'w'` – запись в новый файл, `'a'` – добавление в существующий файл, подробнее см. help); `fid` – файловый идентификатор, который система связывает с файлом (здесь подразумевается, что файл расположен в рабочей папке) или указывает на ошибку (например, отрицательное значение `fid`).

Затем происходит считывание данных из файла:

`a = fscanf(fid, format, size),`

здесь `format = [' %g  %f  %e  %s %c  %d \n'],`

указывает, что в каждой строке файла записано шесть чисел , указанного формата:

`%g` – с плавающей точкой машинного представления

`%f` – с фиксированной точкой

`%e` – с плавающей точкой

`%s` – строка, пробелы в которой не учитываются

`%c` – строка, учитываются пробелы

`%d` – целые десятичные

символ `\n` – переход считывания на следующую строку (перевод каретки);

`size` - размер массива (матрицы), обусловлен машинным представлением, т.е. колонки матрицы, последовательно записаны в вектор-строку

`size = [m, n]` m - количество элементов в строке; `n` – количество элементов в столбце, `n = inf`, если неизвестно количество строк в файле (длина столбца), из которого производится чтение, т.е. `size = [m, inf]` (см. далее рисунок с примерами, случай `size= [2,inf]`).

Обратите внимание, что после считывания получаем матрицу, транспонированную по отношению к заданной в файле, но операция транспонирования A.' поставит все на свои места.

```
clear
fid1=fopen('inpnum.txt','r')
A=fscanf(fid1,'n=%d eps=%g x0=%f \n', [3,4])
```

```
A: 3x4 double =
    1.0000    2.0000    3.0000    4.0000
    0.2000    0.0500    0.0001    0.0000
    3.2564    3.1419    3.1417    3.1415
```

d1)

```
1   n=1   eps=2.0e-01 x0=3.2564
2   n=2   eps=5.0e-02 x0=3.1419
3   n=3   eps=1.0e-04 x0=3.1417
4   n=4   eps=1.0e-05 x0=3.1415
```

inpnum.txt

```
clear
fid=fopen('inpinf.txt','r')
A=fscanf(fid,'%d %g \n', [2,inf])
```

```
A: 2x4 double =
    1.0000    2.0000    3.0000    4.0000
    0.2000    0.0500    0.0001    0.0000
```

```
1   1   2.0e-01
2   2   5.0e-02
3   3   1.0e-04
4   4   1.0e-05
```

inpinf.txt

Операцию транспонирования A.' необходимо применить после считывания в обоих примерах, это обусловлено машинным представлением – хранением матриц вектором, записанным по столбцам.

- форматированный вывод

`fprintf(fid, format, namevar)`

Запись в файл `namevar` (записанный строкой с необходимым расширением); `fid` – файловый идентификатор файла, который уже открыт и пустой или ранее создан;

`format` – строка по аналогии с форматированным вводом, здесь далее приведен пример формата, который еще имеет поясняющие надписи

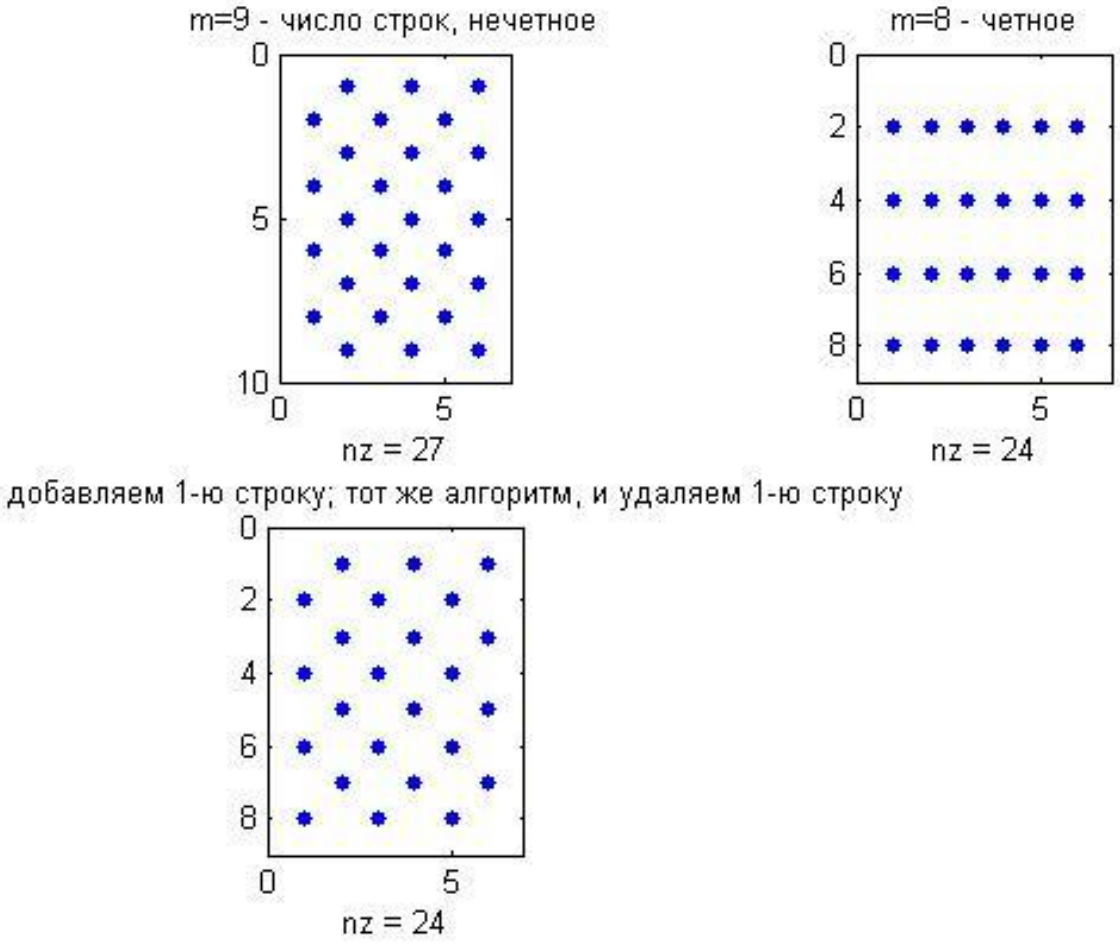
`['points number= %d  x= %g  y=%e  surf=%f \n' ]`

`fclose(fid)` – команда, закрывающая созданный файл.

Пример 4. Создание шахматной структуры, без использования циклов

```
%% шахматный порядок
figure
%m - нечетное; n-любое
m=9
n=6
A=rand(m,n)
A(1:2:end)=0

mytitle=['m=',num2str(m),' - число строк, нечетное']
subplot(2,2,1),spy(A),title(mytitle)

m=8 %m - четное;
n=6
A=rand(m,n)
A(1:2:end)=0
mytitle=['m=',num2str(m),' - четное']
subplot(2,2,2),spy(A), title(mytitle)
A=rand(m,n)
AA=A
A=[rand(1,n);A]
```

```
A(2:2:end)=0
A(1,:)=[]

mytitle=['добавляем 1-ю строку; тот же алгоритм, и уда-
ляем 1-ю строку']
subplot(2,2,3),spy(A), title(mytitle)
```



Задания для самостоятельного решения

Задание 1

- Построить блочно-диагональную матрицу, которая состоит из n блоков (`n` – целое, генерируется случайным образом на отрезке `[4,8]` с помощью `randi` или `randint` – зависит от версии MatLab).
- Блоки строятся генератором равномерно распределенных чисел на отрезке `[0,1]`, размер каждого из n блоков определяется арифметической прогрессией `n:1:2n-1`.
- Вывести на экран структуру матрицы командой `spy`.

Задание 2

- Построить матрицу `A` пятого порядка с помощью генератора `rand`.
- Вычислить `A' A` и `A'+A` и доказать, что полученные матрицы симметричны. При выполнении задания циклы использовать нельзя.

Задание 3

- Второй и предпоследний блоки блочно-диагональной матрицы (см. задание 1) определяют подматрицу блочной матрицы, начинающейся с `n+1` –й строки и столбца и до `end-(2n-1)` строки и столбца.
- Требуется передвинуть выбранные блоки так, чтобы они разместились в вершинах побочной диагонали подматрицы.
- Структуру матрицы отобразить на экране с помощью команды `spy`.

Задание 4

- Постройте заполненную матрицу размера (`m`, `n`).
- Расставьте нули в матрице в шахматном порядке, не используя операторов цикла.
- Рассмотрите случаи для четного и нечетного значений `m`.
- Отобразите на экране структуру исходной матрицы и результата в смежных осях. Для этого используйте команды `spy` и `subplot`.

Задание 5

- Предложите два способа суммирования элементов вектора длины 10000000 (сгенерируйте случайным образом с помощью `rand`), без использования цикла.
- Оцените скорость выполнения операций в обоих случаях. Команды оценивания времени работы процессора: `tic, toc`; -секундомер; и затраты на процесс как разница времени начала и конца:
- `t1=clock`, инструкции алгоритма, `t2=etime(clock,t1)`

Задание 6

- Задайте `n` - размер матриц `A=ones(n), B=zeros(n); n=size(C), C` – произвольная матрица, состоящая из нулей и единиц, сгенерированная `randi (randint)`.
- Объясните смысл и результат следующих операций:
  `B&C`
  `A|C`

Замечание. Любую матрицу, элементы которой нули, или единицы, можно конвертировать в логическую. Например, матрица `A=logical(ones(n))` – логическая; здесь `logical` – конвертер.

Контрольные задания и вопросы

Выполните и объясните следующие команды. Полагайте переменные заданий 1-4 глобальными.

Задание 1. help elmat

```
ones(2,5)
A=[1:3;4:6]
B=ones(size(A))
eye(5)
rand(5)
C=zeros(3)
```

Задание 2. help >

```
D=[1-i 2; 0.5i -1+2*i]
real(D)
imag(D)
D1=D'
D2=D.'  % объясните отличие D1 D2
A' % матрица A из задания 1
```

Задание 3. Выполнить, объяснить особенности

```
10/3
format rational
1/3
format short
1/3
% обратите внимание на системные переменные,
% которые являются результатом выполнения последних двух команд
1/0
0/0
realmin
realmax
```

Задание 4. Выполнить, определить размер результирующих массивов. Объяснить.

```
A=[A; sin(0.5*pi*(1:3))]
diag(A)
diag(diag(A))
S=sum(A)
S=sum(sum(A))
abs(A)
norm(A) %определите по справке, какую норму вы определили
```

Объекты класса Char. Функции и свойства

Объекты типа Char являются потомком ARRAY, поэтому для них справедливы все принципы, которые поддерживаются для массивов.

Объекты типа Char – строки; строки состоят из цифр, букв и символов таблицы ASCII, каждый элемент строки занимает два байта, это является нетипичным для других языков программирования, в которых одному элементу отводится один байт памяти, но MatLab ориентирован на матричные вычисления, в основе которых лежит комплексная арифметика (переход от операций комплексной арифметики к вещественной достигается автоматически с нулевой мнимой частью, обратный переход автоматически невозможен), это и обусловливает резервирование еще одного байта на комплексную часть.

Справка о создании, контроле типа и редактировании строковых переменных

```
% Задание строки:
% один или серия символов заключается в апостроф
A='ученье'
% кавычка задается четырьмя апострофами
A=''''ученье'''' % выводится  “ученье”

% контроль типа:
% является ли аргумент функции ischar строкой,
ischar(A)  % если да, то результат - логическая единица
length(A)% количество элементов строки равно 10

% Конкатенация строк (по правилом матричной алгебры):
% строки записываются матрицей размера 2×3
['tip'; 'top']
% аналогично функцией strvcat (важно: согласование размерностей по столбцам не требуется!)

% две строки последовательно записываются строкой %из шести букв,аналогично функцией strcat
['tip', 'top']
```

```
% Поиск букв или лексем:
S= 'sin(x)*cos(y)', s='x'
%  ищем в большей строке меньшую
k = findstr(s, S)
%  ищем в строке первого аргумента строку второго
r = strfind(S, s)
isempty(k), isempty(r) %  проверка хотя бы
% одного совпадения лексем (успех - логическая единица)

%% Сравнение строковых переменных v1 и v2
% v1 сравнилось с v2, если все буквы совпали с учетом
% регистра
strcmp(v1,v2)
% v1 сравнилось с v2, если все буквы совпали без учета
% регистра
strcmpi(v1,v2)
% сравнилось n букв, с учетом регистра
strncmp(v1,v2,n)
% сравниваются  n  букв без учета регистра
strncmpi(v1,v2,n)
```

Пример 1. Конкатенация (соединение) строк

```
% Конкатенация строк с пробелами и без в конце,
% первый способ:
strcat('С ', 'Новым ', 'Годом!')
% пробел – элемент строки!
strcat('С', 'Новым', 'Годом!')
% второй способ:
['С ', 'Новым ', 'Годом!']
```

Пример 2. Сравнение строк

```
% с учетом регистра и без:
s1='ABCDEFGH'
s2='ABCDefgh'
s3='ABCabc'
strcmp(s1,s2)
strcmpi(s1,s2)
strncmp(s1,s2,3)
```

Пример 3. Сравнение матриц, элементы которых - строки

```
% с учетом регистра и без:
sm1=['1234567'; 'ABCDEFG'], sm2=['1234567'; 'ABCDefg']
strcmp(sm1, sm2)
strncmp(sm1, sm2, 4)
strcmpi(sm1, sm2)
sm3=['1234567'; 'ABCDEFG']
sm4=['1234567'; 'ABCdefg']
strcmpi(sm3, sm4)
```

Пример 4. Преобразование регистра строки

```
% строчные буквы:
lower('Happy Birthday - С Днем Рождения!')
% прописные буквы:
upper('Happy New Year - С Новым Годом!')
```

Пример 5. Выделение лексем

```
% выделение слов, составляющих выражение,
% разделенных пробелами
s='С Новым Годом!'
[t1, r1] = strtok(s)
[t2, r2] = strtok(r1)
[t3, r3] = strtok(r2)
% выбраны три лексемы в переменных: t1, t2, t3

% выделение слов, составляющих выражение, разделенных
%нестандартными разделителями:
s='a+b*c'
[t1, r1] = strtok(s, '+*')
[t2, r2] = strtok(s, '+*')
[t3, r3] = strtok(s, '+*')
% второй аргумент –строка, должна содержать весь набор
%разделителей
```

Пример 6. Поиск элементов строки (подстрок в строке)

```
% результат – номер элемента в строке поиска, с которо-
го, начинаются совпадения
s1='00', s2='2003'; s3='100002'
```

```
findstr(s1,s2)
findstr(s2,s1)
strfind(s1,s2)
strfind(s2,s1)
findstr(s3,s1)
```

Пример 7. Поиск элементов (подстрок) в многомерных строках

```
sm=strvcat('com', 'compare', 'computer')
strmatch('com', sm)
strmatch('com', sm, 'exact')
s='123com'
strmatch('com',s)
sc={'com';'compare';'computer'}
strmatch('com', sc)
```

Пример 8. Поиск и замена элементов строки

```
s='12341234'
s1=strrep(s, '123', 'ABCD')
s2=strrep(s, '124', 'ABCD')
s3=strrep(s, '123', '')
```

Пример 9. Вывод элементов таблицы ASCII

```
% вся таблица
char(1:255)
% xyz
char(120:122)
```

Пример 10. Заполнение многомерной строки элементами

```
% второй и третий аргументы – размерности %массива
repmat('=', 1,4)
repmat('*-', 3,4)
```

Пример 11. Выравнивание строки

```
s=' 123456 '
sl=strjust(s, 'left')
sc=strjust(s, 'center')
sr=strjust(s, 'right')
```

Пример 12. Выполнение фрагментов строки

```
s='С Новым Годом!';
s(1:2)
s(3:8)
s(9:end)
```

Пример 13. Выявление позиций элемента в строке

```
s='С Новым Годом!';
r=eq(s, 'о')
% или равносильно:
q=r=='о';
if any(r), disp('есть совпадение'),end %
sum(r) % количество совпадений
index=find(r) % порядковые номера позиций совпадений
```

Объекты класса Cell. Функции и свойства

Cell – конструктор класса, массива разнородных объектов – ячеек. Однако с его помощью только задается размер массива. Фигурные скобки используются для перечисления его элементов, а также для указания индексов при оперировании его объектами.

Способы создания: делятся на декларативные, описательные, конвертационные и создаваемые системой. К числу создаваемых системой относятся массивы ячеек, получаемые при формировании выходных параметров переменной длины, массивы ячеек, которые используются системой и пользователем при обработке событий пользовательского интерфейса и т.д.

В отличие от ранее рассмотренных объектов, содержание массива ячеек даже при отсутствии подавления вывода точкой с запятой (;) будет невидимым.

Функция `celldisp(c)` – визуализации элементов, решит эту проблему, также как и команда c{:}

Пример 1. Создание массивов ячеек

```
A=ones(6)
% резервирование
C=cell(size(A))
b = {'sin(x.^2)/(3 * pi* x.^2) ',[1:2:pi],rand(5)}
celldisp(b)
% каждый элемент полученной матрицы –
% ячейка, состоит из одного элемента, обращение к (i,j)
% элементу %g{i,j};
g=num2cell(randn(3))
% r –массив ячеек, состоящий из одного элемента, и
% этот  элемент есть матрица класса double 4-го порядка
% – и  обращение к (i,j) элементу r{1}(i,j)
r=mat2cell(rand(4))
%понять адресацию к элементам d
d = {[1] [2 3 4]; [5; 9] [6 7 8; 10 11 12]}
iscell(d) % контроль типов
```

Пример 2. Поиск совпадающих лексем с использованием массивов ячеек

```
sc1=[{'1234'}; 'ABCDEFGH']
sc2=[{'1235'}; 'ABCDefgh']
strcmp(sc1, sc2) % поиск совпадений  без учета регистра
strcmpi(sc1, sc2)
strncmp(sc1, sc2, 3) % поиск первого совпадения трех
% подряд элементов строки
```

Пример 3. Эффективного построения блочно-диагональной матрицы - blkdiag

```
% матрицы для блоков – массив % ячеек
Blocks={rand(3);randn(5);ones(4)}
% B -  блочно-диагональная матрица
B=blkdiag(Blocks{:})
```

Пример 4. Конвертирования в char

```
str = { 'Goodbye', 'cruel', 'world' }
char(str{:})
```

Пример 5. Конкатенации

```
c = { [3 4], [5 6] };
cat(1, [1 2], c{:} ) % добавление строк
cat(2, [1 2], c{:} ) % добавление столбцов
e = {}; cat(2, [1 2], e{:} )
```

Пример 6. Создания массива ячеек

```
T = cell(1,9); % резервирование
T(1:2) = { [1], [1 0] };
for n=2:8, T{n+1}=[2*T{n} 0] - [0 0 T{n-1}]; end
T{4}
```

Создание функций в Matlab

В ML для эффективного программирования используются процедуры и процедуры-функции. Каждая процедура записывается в отдельном файле с расширением *.m и имя процедуры должно совпадать с именем этого файла.

Функции и процедуры

Для создания процедур и процедур-функций используется одинаковый заголовок, но в процедуре может быть один или несколько выходных параметров

```
function [out1,out2] = myproc(in1,in2,in3)
```

а в функции только один, который вычисляется в последнем исполняемом операторе процедуры.

```
function resfunc = myfun(in1,in2,in3)
. . . . . .
resfunc=sin(in1)*in2^in3 % некоторое выражение
```

Пример 1. Процедуры

```
function [x1,x2] = quadform(a,b,c)
d = sqrt(b^2 - 4*a*c);
x1 = (-b + d) / (2*a);
x2 = (-b - d) / (2*a);
```

Обратиться к процедуре можно `[r1,r2]=quadform(1,1,1)`, используя конкретные значения входных параметров.

В MATLAB имеются встроенные функции, которые могут иметь меняющееся число входных аргументов и меняющееся число выходных параметров. Например, функция S=svd(A) вычисления сингулярных чисел матрицы A. Она может применяться в виде [U,S,V]=svd(A), когда требуется большее чис-

ло выходных параметров. Другим примером такой функции может служить функция cat(A,B) горизонтального объединения массивов A и B . Она может иметь произвольное число входных массивов, cat(A1,A2,A3,A4).

При написании собственных функций в ML существует возможность указывать переменное количество входных и выходных аргументов. Для этого предназначен массив ячеек переменной длины varargin для входных параметров и varargout для выходных. В этом случае заголовок процедуры будет иметь вид:

```
function [out1,out2,varargout] = myproc(in1,in2,in3,
varargin)
```

Такие ситуации обусловлены тем, что пользователь сам решает в каждом конкретном случае, что ему нужно на выходе, например, кроме постоянного выходного параметра вектора-решения, точность и или номер итерации.

При обращении к такой процедуре будут заданы конкретные параметры `varin1, varin2,…` и идентификаторы `varout1, varout2,…`

Неопределенность длин этих массивов ячеек накладывает дополнительную ответственность на программиста при программировании процедур. Так в момент обращения все переменные аргументы помещаются системой в `varargin`, их следует оттуда извлечь и присвоить соответствующим сущностям-переменным.

Длину массива `varargin` определяем по формуле: количество всех входных переменных (определяет функция `nargin`) минус количество постоянных входных аргументов, так же как и длину `varargout`; количество всех выходных переменных определяет функция `nargout`.

Пример 2. Тип файла – функция. Имя файла – varlist.m

```
function varlist(varargin)
   fprintf('Number of arguments: %d\n',nargin);
% nargin – количество входных аргументов в функции
   celldisp(varargin)
```

Вызов функции:

```
varlist(ones(2),'some text',pi)
```

Результат:

```
Number of arguments: 3
varargin{1} =
     1     1
     1     1
varargin{2} =  some text
varargin{3} = 3.1416
```

Пример 3. Тип файла – функция. Имя файла – sizeout.m

```
function [s,varargout] = sizeout(x)
nout = max(nargout,1) - 1;
% nargout – количество выходных аргументов функции
s = size(x);
for k=1:nout
   varargout{k} = s(k);
end
```

Вызов функции:

```
[s,rows,cols] = sizeout(rand(4,5,2))
```

Результат:

```
s =      4     5     2
rows =     4
cols =       5
```

Пример 4. Количество входных параметров.
Тип файла – функция. Имя файла – testarg1.m

```
function c = testarg1(a,b)
if (nargin == 1)
    c = a.^2;
elseif (nargin == 2)
```

```
    c = a + b;
end
```

Вызовы функции:

```
estarg1([1 2])
testarg1([1 2],[3 4])
```

Результат выполнения:

```
ans =     1     4
ans =     4     6
```

Пример 5. Суммирование объектов double в массиве ячеек `varargin`

```
function s = add(s,varargin)
for n = 1:nargin-1
s = s + varargin{n};
end
```

Пример 6.
О массиве ячеек `varargin` входных параметров переменной длины

```
function b = blue(varargin)
if nargin < 1
varargin = {'rgb'};
end
switch(varargin{1})
case 'rgb'
b = [0 0 1];
case 'hsv'
b = [2/3 1 1];
otherwise
error('Цветовая модель не определена')
end
```

Аноним и функция-строка

Помимо описанных конструкций в ML используются анонимы. Это не-поименованные процедуры-функции одного или нескольких аргументов.

Синтаксис анонимов сводится к выражению, левая часть которого является именем процедуры, правая состоит из определяющего символа @, после которого в круглых скобках перечисляются один или несколько аргументов функции, а затем приводится её аналитическое представление, зависящее от этих аргументов, например,

```
sincos = @(x) sin(x) + cos(x);
w = @(x,t,c) cos(x-c*t);
```

Заметим, что анонимы могут быть аргументами функций, например, fzero

```
fzero( @(x) sin(x)+cos(x), 0 ).
```

Анонимную функцию можно определять прямо в командной строке ML или в пределах функции или скрипта. То есть, можно создать простые функции без необходимости создания файла специально для них.

Конструкция `inline` также обеспечивает быстрое создание функции одной или нескольких переменных в соответствие с предлагаемым синтаксисом:
`Namefunction=inline(expression_string)`

Пример 5. Процедура `inline`

```
g=inline('2*cos(x)-sin(y)')
g(pi/8,pi/12)
symvar(g) % массив ячеек, содержит аргументы функции
g{1},g{2} % аргументы
```

Подпроцедуры

Помимо функций и процедур иногда целесообразно определить функцию, которая нужна только для выполнения конкретной процедуры, тогда она должна быть записана в том же файле, что и головная процедура, и является подпроцедурой (подфункцией). Подпроцедура «невидима» для остальных программ или процедур.

Пример 6. Процедуры и подфункции

```
function [x1,x2] = quadform(a,b,c)
d = discrim(a,b,c);
x1 = (-b + d) / (2*a);
x2 = (-b - d) / (2*a);
end % quadform()

function D = discrim(A,B,C)
D = sqrt(B^2 - 4*A*C);
end % discrim()
```

Литература

1. Говорухин, В. Компьютер в математическом исследовании: Учеб.курс - СПб. [и др.]: Питер, 2001. - 624 с.
2. А. М. Половко, П. Н. Бутусов. MATLAB для студента Санкт-Петербург БХВ-Петербург 2005 320 с. (19 экз)
3. Таранчук В.Б. Основные функции систем компьютерной алгебры. , 2013. — 59 р.
4. Дьяконов В.П. MatLab:Учебный курс. СПб..: 2001
5. Джон Г. Мэтьюз,КуртисД.Финк.Численные методы. Использование MatLab.И. «Вильямс »Москва, Санкт-Петербург, Киев.2001, 713 с.
6. Дьяконов В. П. Энциклопедия компьютерной алгебры. ДМК-Пресс, 2009. — С. 1264. — ISBN 978-5-94074-490-0.