Первое знакомство с MatLab (ML)

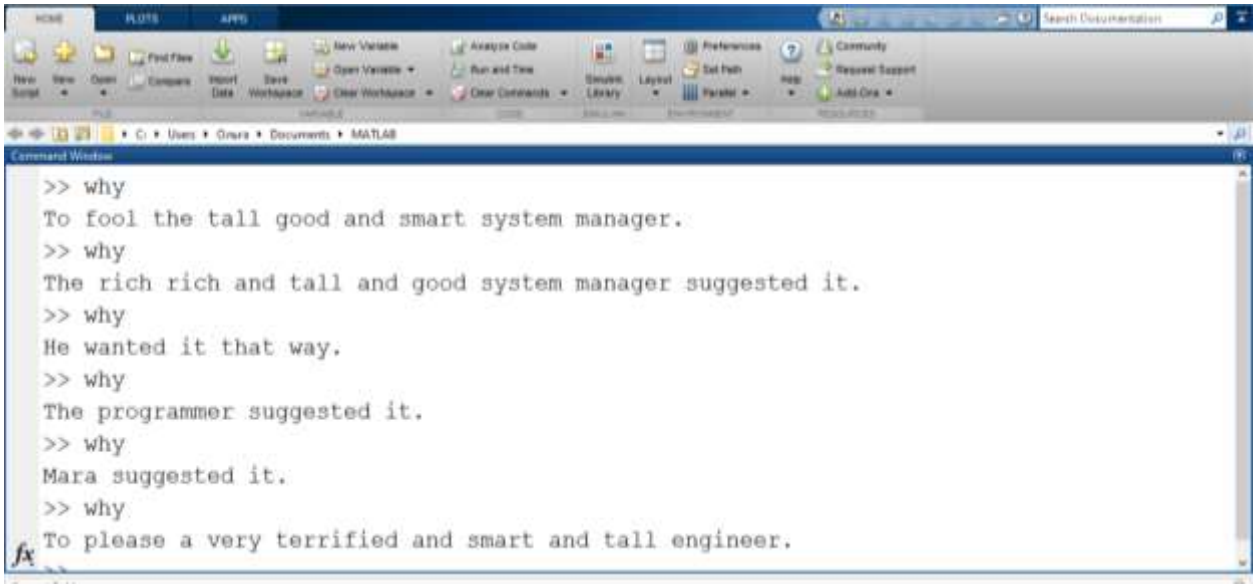
Пакет ориентирован на интерактивное (суперкалькулятор) и программное функционирование (MatLab – высокоуровневый язык на базе FORTRAN с оптимизацией на C, C++).

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
```

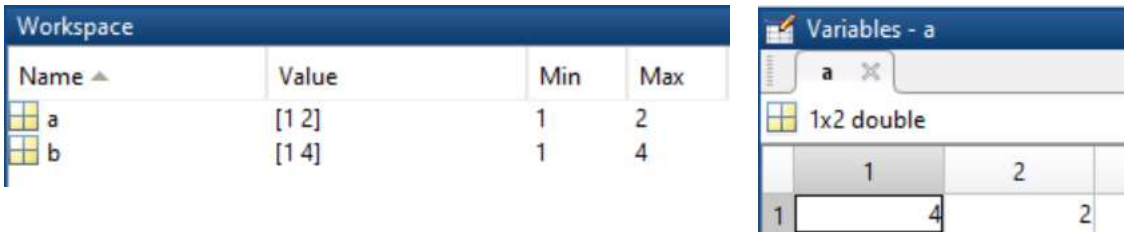
Пример 3. Свойства CW как графического объекта Root с нулевым дескриптором

```
% 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.

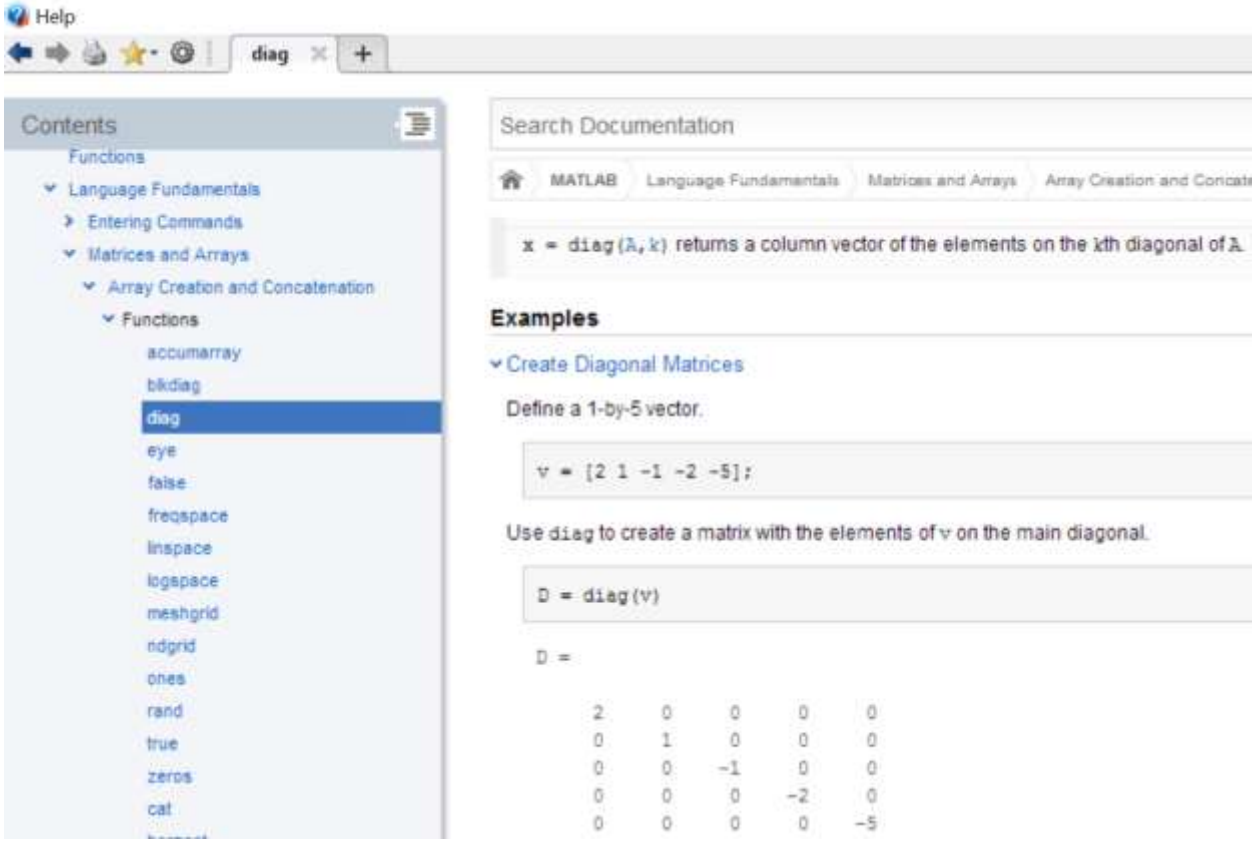
Пример 4. Тематические справочные материалы

```
% 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 си-
стемным 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 число столбцов первого матричного сомножителя и число строк второго должны совпадать. Матричными операциями являются возведение в степень и транспонирование.

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.