**Функция diag()**

Данная функция используется для:

1. построения диагональной матрицы из вектора,
2. выделения диагонали из матрицы,
3. заполнения побочных диагоналей. Для построения побочных диагоналей используется функция с двумя аргументами: первый аргумент – вектор, второй аргумент число – смещение вправо (+), влево (-), число – насколько смещаться.

**Пример:** % построение диагональной матрицы

**a=[1 2 3]**

**A=diag(a)**

**A =**

**1 0 0**

**0 2 0**

**0 0 3**

**Пример:** % выделение диагонали

**B=[5 6 7; 2 4 7; 2 6 5]**

**b=diag(B)**

**b =**

**5**

**4**

**5**

**Пример:** % построение побочной диагонали

**a=[1 1 1]**

**A=diag(a)**

**b=[2 2]**

**B=diag(b,1)** % верхняядиагональ

**C=A+B**

**D= diag(b,-1)** % нижняядиагональ

**a =**

**1 1 1**

**A =**

**1 0 0**

**0 1 0**

**0 0 1**

**b =**

**2 2**

**B =**

**0 2 0**

**0 0 2**

**0 0 0**

**C =**

**1 2 0**

**0 1 2**

**0 0 1**

**D =**

**0 0 0**

**2 0 0**

**0 2 0**

**Пример: Построить матрицу, используя функции MATLAB**

**A =**

**1 1 -3 -3 -3 -3**

**1 1 -3 -3 -3 -3**

**-3 -3 2 0 0 0**

**-3 -3 0 2 0 0**

**-3 -3 0 0 2 0**

**-3 -3 0 0 0 2**

**Функцияrepmat**

Создание блочной матрицы, состоящей из одинаковых блоков

**X=[1 2;3 4]**

**repmat(X,2,3)**

**1 2 1 2 1 2**

**3 4 3 4 3 4**

**1 2 1 2 1 2**

**3 4 3 4 3 4**

**Функцияblkdiag()**

Матрицы на главной диагонали

**Пример:**

**A=[2 5 4; 3 4 2]**

**B=[2 3; 7 4]**

**C=[7 8 9; 2 4 7]**

**D= blkdiag(A, B, C)**

**D =**

**2 5 4 0 0 0 0 0**

**3 4 2 0 0 0 0 0**

**0 0 0 2 3 0 0 0**

**0 0 0 7 4 0 0 0**

**0 0 0 0 0 7 8 9**

**0 0 0 0 0 2 4 7**

**Функция cat**

Сцепление массивов

**A=[1 2; 3 4];**

**B=[5 6; 7 8];**

**M=cat(1,A,B)** % сцепление идет по столбцу

**M =**

**1 2**

**3 4**

**5 6**

**7 8**

**M=cat(2,A,B)** % сцепление идет по строке

**M =**

**1 2 5 6**

**3 4 7 8**

**M=cat(3,A,B)** % создание трехмерной матрицы

**M(:,:,1) =**

**1 2**

**3 4**

**M(:,:,2) =**

**5 6**

**7 8**

**Функции работы смассивами**

**prod(x)** – перемножение элементов массива;

**sum(x)** – суммирование элементов массива;

**max(x)** – нахождение максимального элемента массива;

**min(x)** – нахождениеминимального элемента массива;

**mean(x)** – нахождениесреднего значения массива;

**sort(x)** – сортировка массива.

**Пример:**% проверить работу функций

**x=[2 7 5 1 8];**

**y=prod(x)**

**y=sum(x)**

**y=max(x)**

**y=min(x)**

**y=mean(x)**

**y=sort(x)**

Рассмотрим работу функций с несколькими векторами (многомерными массивами).

Проанализировать работу и выводы записать в тетрадь.

Для примера возьмем функцию **min()**.

Пусть

**x=[2 7 5 1 8];**

**[u,v]=min(x)** % u =1,v =4; u= минимальное значение, v= индекс (номер) минимального элемента

**y=[4 5 3 2 3]**

Тогда

z=min(x,y) % 2 5 3 1 3

z1=min([x,y]) % 1

z2=min([x;y]) % 2 5 3 1 3

z3=min([x,y],[],1) % 2 7 5 1 8 4 5 3 2 3

z4=min([x,y],[],2) % 1

**Рассмотрим матрицу**

**A=[4 1 7; 3 5 2; 1 4 3]**

**z=min(A)** % **1 1 2**

**z1=min(A,[],1)** % **1 1 2**

**z2=min(A,[],2)** % (**1 2 1)’**

**Проверить работу:**

**[n,m]=min(A)**

**[n,m]=min(A,[],1)**

**[n,m]=min(A,[],2)**

**Функции преобразования**

**rot90(A) –** поворачивает массив на 90 градусов против часовой стрелки.

**rot90(A,n) –** поворачиваетмассивнаn\*90 градусовn = +-1,+-2,...

**fliplr(A) –** перестановка столбцов матрицы слева направо относительно оси середины

**flipud(A) –** перестановка строк матрицы сверху вниз

**reshape(x,m,n)** – изменяет форму матрицы или массива (из вектора в матрицу)

**tril(A) –** выделение нижнего треугольника из матрицы относительно оси середины

**triu(A) –** выделение верхнего треугольника из матрицы

**Пример:** Проверить работу функций преобразования

A=[1 2 3; 4 5 6;7 8 9]

**Пример**: Сформировать из вектора**x** длины**m×n**матицу **A**размерностью **m×n**

x=[1:12];

**A=reshape(x,3,4)** % формирование идет по столбцам

**1 4 7 10**

**2 5 8 11**

**3 6 9 12**

**Создание трехмерного массива**

**B= reshape(x,2,3,2)**

**B= reshape(x,2,3,2**) % по столбцам

**B(:,:,1) =**

**1 3 5**

**2 4 6**

**B(:,:,2) =**

**7 9 11**

**8 10 12**

**Логическое индексирование (ЛИ)**

ЛИ предназначено для создания массива с выбором элементов массива, отвечающих некоторому логическому выражению. Решение данной задачи состоит из двух этапов (их можно совместить в одном операторе).

1. Получение логического массива.
2. Выбор элементов массива, отвечающих логическому условию, и запись их в вектор.

**Пример: %** последовательно и, используя один оператор

>>**A=[2 3 5; 4 2 6; 7 2 2]**

**A** =

2 3 5

4 2 6

7 2 2

>>I=A==2 % создание логического массива

I =

1 0 0

0 1 0

0 1 1

>>**b=A(I**) % выбор элементов из заданного массива и запись его в вектор

**b** =

2

2

2

2

Используя один оператор:

>>**b=A(A==2)**

**b** =

2

2

2

2

**Пример:**Заменить все элементы матрицы **A**,равные **2**, на **5.**

**A** =

2 3 5

4 2 6

7 2 2

**Решение:**A(A==2)=5

Рассмотрим задачу получения индексов элементов массива, удовлетворяющих заданному логическому выражению.

Для решения такой задачи используется функция **find()**cодним или двумя выходными параметрами.

**Пример:**

**n=find(A==2)** % вектор, элементы которого указывают на «место» в матрице (по столбцам)

**[i,j]=find(A==2)** % два вектора – индексы (строка, столбец)

**n =**

**1**

**5**

**6**

**9**

**i =**

**1**

**2**

**3**

**3**

**j =**

**1**

**2**

**2**

**3**

В первом случае вектор **n**будет состоять из номеров элементов, если идти по столбцам матрицы.

Во втором случае получим индексы элементов.

**Пример**: Дана матрица **A**. Найти индексы максимальных элементов матрицы**A**.

A =[ 2 3 5; 4 7 6; 7 2 2]

# §3. Работа в редакторе. М-файлы. Функции

**Сценарии, функции, переменные**

**Определение**: Файлы, содержащие команды и операторы MATLAB, называются М-файлами.

Существует два типа М-файлов:

**М-сценарии и М-функции.**

М-сценарий (файл-программа) – файл, который содержит последовательность команд.

М-функция (файл-функция) – файл, в который записывается функция.

Имя М-файла не должно начинаться с цифры, **'**+**'**, **'**-**'**.

В отличие от файлов сценариев, которые могут называться как угодно (согласно соглашению об имени), **имя m-файла, содержащего функцию, должно совпадать с именем функции.**

**НастройкаредактораМ-фалов**

Preferences ->Editor/Debugger

**Создание m-файлов**

**Создание М-сценария**

**HOME ->FILE->NewScript**

Бывает полезно “закрепить” редактор в командном окне (DockEditor).

Выполнение программы:**Run**или **F5**

**Создание М-функции**

1. **Создание простого m-файла** (функция не принимает никаких данных и ничего не возвращает)

HOME ->New ->Function

% **Форматкомандыdisp()**:

**% disp**(<перменная>|<**'**текст**'**>)

**functionfun0()**

**% функциябезпараметров**

**% disp('Hello, World!!! ');**

**disp('Hello, World!!!');**

**end**

**Вызовфункции:**

**>>fun0**

**Hello, World!!!**

Строки с комментариями, расположенными сразу же после определения функции, выводятся при запросе команды**help** с именем функции.

**>> help fun0**

**функция без параметров**

**disp('Hello, World!!! ');**

1. **Создание m-файла** (функция принимает несколько параметров и возвращает один):

Создадим новый файл:

**function [a] = fun1( c,d )**

**% программа - функция**

**% два входных параметра и один выходной**

**a=c\*d;**

**end**

Сохранить файл:

EDITOR->SAVE

**Вызовфункции:**

Вокне**Command Window**:

**>> x=5; y=7;**

**>> z=fun1(x,y)**

**z =**

**35**

1. Создание m-файла (функция принимает несколько параметров и возвращает несколько). Выходные параметры необходимо перечислить в квадратных скобках:

**function [s,d]=sum\_and\_difference(a,b)**

**% function [s,d]=sum\_and\_difference(a,b)**

**s=a+b;**

**d=a-b;**

**end**

**Вызовфункции:**

Вокне**Command Window**:

**>> [x,y]= sum\_and\_difference(7,5)**

**x =**

**12**

**y =**

**2**

**Редактирование М-файла**

Вокне**CommandWindow**:

**edit<имя\_M-файла>**

**Выполнение части М-файла:**

**Способ 1: Используя выделение части файла**

1. Выделить часть файла
2. <F9>

**Способ 2:Используя комментарии** (закомментировать отдельные блоки программы).

Комментарии (вводятся в отдельные строки) вначале строки

1. %{

начало комментариев

1. конец комментариев

%}

**Способ 3:Разбиение М-файла на ячейки**

Можно разбить М-файл на части (ячейки) и выполнять их независимо. Текст разбивается на ячейки с помощью строк комментариев, начинающихся с двух идущих подряд **%%**.

Выполнение кода секции:

**RunSection**

Одновременно можно редактировать несколько M-файлов.

**Глобальные переменные**

Все переменные по умолчанию в функции являются локальными.

Введем в корневом пространстве (**CommandWindow**) переменную и попытаемся использовать ее в вызываемой функции

**function a2=test\_local()**

**% использование локальных переменных**

**a2=a^2;**

**end**

**>> a=5**

**a =**

**5**

**>> test\_local**

**Undefined function or variable 'a'.**

**Error in test\_local (line 5)**

**a2=a^2;**

Есть возможность объявить переменную глобальной при помощи команды **global**:

**function a2=test\_global()**

**% использование глобальных переменных**

**global a;**

**a2=a^2;**

**end**

Прежде чем использовать глобальную переменную, ее необходимо объявить таковой в корневом рабочем пространстве:

**globala**

Потом присвоить значение:

**a=5**

Далее вызвать функцию:

**test\_global**

Проверить работу функции test\_global()

Осуществить проверку того, является переменная глобальной или нет, можно при помощи команды **isglobal**:

**b=9;**

**isglobal(b)**

**isglobal(a)**

**Подфункции**

**В M-файле может быть одна основная функция и несколько подфункций.**

**Имя основной функции совпадает с именем М-файла. Заголовок новой подфункции одновременно является признаком конца предыдущей подфункции.**

**Пример:**

**functionsbfun1(x,y)**

**% иллюстрация основной функции**

**% и нескольких подфункций в одном файле**

**global s d**

**s=10;**

**d=20;**

**fsum(x,y)**

**fdif(x,y)**

**fmul(x,y)**

**function fsum(a,b)**

**global s d**

**fsum=(a+b)+(s+d)**

**function fdif(a,b)**

**global s d**

**fdif=(a-b)+(s-d)**

**function fmul(a,b)**

**global s d**

**fmul=a\*b+s\*d**

Вокне**Command Window**:

x=5; y=17;

sbfun1(x,y)

**Вложенные функции**

Есть главная функция и в ней находятся вложенные функции.

При написании вложенных функций для обозначения конца функции используется оператор **end**.

**function sbfun2(x,y)**

**% иллюстрация главной функции**

**% и нескольких вложенных функций**

**% не надо использовать глобальные переменные**

**s=10;**

**d=20;**

**fsum(x,y)**

**fdif(x,y)**

**fmul(x,y)**

**function fsum(a,b)**

**fsum=(a+b)+(s+d)**

**end**

**function fdif(a,b)**

**fdif=(a-b)+(s-d)**

**end**

**function fmul(a,b)**

**fmul=a\*b+s\*d**

**end**

**end**

Вокне**Command Window**:

sbfun2(x,y)