# Основы MATLAB

## Функции, ячейки, строки, структуры

Юдинцев В. В.

Кафедра теоретической механики

17 февраля 2026 г.

САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

# Содержание

# Функции MATLAB

# Математические функции

`sin, cos, tan, cot, acos, asin, ...` – тригонометрические
`log, log10, log2, exp, sqrt, nthroot(x,n), ...`
`sign(a)` – знак числа `a`: -1, 0, +1.

# Функции округления

- К ближайшему к нулю целому:
  `fix(1.8)=1` но `fix(-1.8)=-1`
- К меньшему целому:
  `floor(-1.9)=-2` но `floor(1.9)=1`
- К большему целому:
  `ceil(1.3)=2`
- К ближайшему целому:
  `round(1.4)=1`, `round(-1.5)=-2`

# Функции комплексных чисел

- `abs(z)` – модуль.
- `angle(z)` – аргумент.
- `conj(z)` – комплексно-сопряженное число.
- `imag(z)` – мнимая часть.
- `real(z)` – вещественная часть.
- `isreal(z)` – 1, если z – вещественное число, 0 – мнимое.

# Специальные матрицы

- `eye(n), eye(n,m)` – единичные матрицы $n \times n$.

```
>> eye(3,4)
ans =
     1     0     0     0
     0     1     0     0
     0     0     1     0
```

- `rand(n), rand(n,m)` – матрица псевдослучайных чисел
- `ones(n,m)` – матрица $n \times m$, заполненная единицами
- `zeros(n,m)` – матрица $n \times m$, заполненная нулями

# Специальные матрицы

`magic(n)` – квадратная матрица $n \times n$,с одинаковыми суммами элементов по строкам и столбцам и диагоналям.

```
>> magic(3)
ans =
     8     1     6
     3     5     7
     4     9     2
```

# Фунция size

Количество элементов по каждой размерности (результат – массив)

```
>> a = [1 2 3; 4 5 6];
>> size(a)

ans =
     2     3
```

Количество элементов по 1 и 2 размерности

```
>> a = [1 2 3; 4 5 6];
>> size(a,1)

ans =
     2
>> size(a,2)

ans =
     3
```

# Фунция numel

Количество элементов:

```
>> a = [1 2 3; 4 5 6];
>> numel(a)

ans =
     6
```

# Функция reshape

Преобразование матрицы-строки 1x6 в матрицу 3x2

```
>> a = [1 2 3 4 5 6];
>> b = reshape(a, 3, 2)

b =
     1     4
     2     5
     3     6
```

Преобразование матрицы-строки 1x6 в матрицу 2x3

```
>> a = [1 2 3 4 5 6];
>> b = reshape(a, 2, 3)

b =
     1     3     5
     2     4     6
```

# Функция reshape

Преобразование матрицы 3x2 в матрицу 2x3

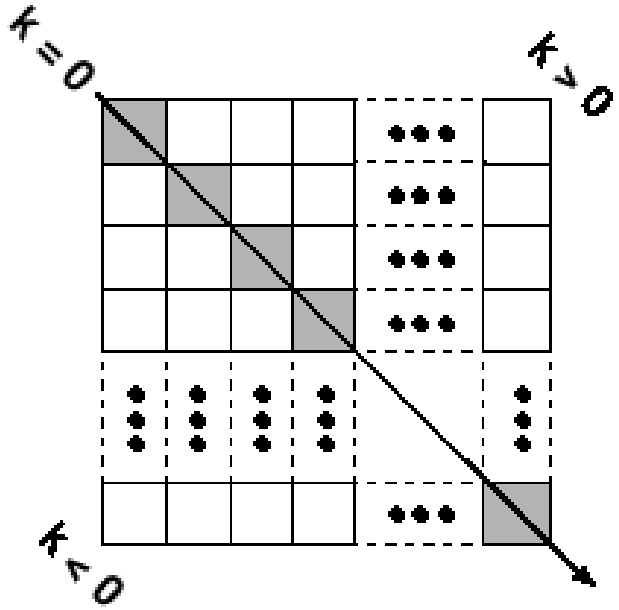
```
>> a = [1   4
        2   5
        3   6];
>> b = reshape(a, 2, 3)

b =
     1    3    5
     2    4    6
```

# Функции над матрицами

- `diag(A), diag(A,k)` – строка, содержащая диагональные элементы матрицы $\mathbf{A}$, если $\mathbf{A}$ - матрица (k=0 для главной диагонали).
- `diag(V)` – диагональная матрица с элементами вектора $V$ на диагонали, если $V$ - вектор.
- `tril(A), tril(A,k)` – нижняя треугольная матрица, построенная по матрице $A$.
- `triu(A), triu(A,k)` – верхняя треугольная матрица, построенная по матрице $A$.

# Функции над матрицами

Параметр $k$ в функциях `diag(A,k)`, `tril(A,k)`, `triu(A,k)`



# Функция repmat

repeat matrix – создать матрицу, используя 2 копии матрицы $a$ по первой размерности и восемь копий матрицы $a$ во второй размерности:

```
>> a = [1   8];
>> b = repmat(a, 2, 4)

b =
     1     8     1     8     1     8     1     8
     1     8     1     8     1     8     1     8
```

# Векторное произведение

## Векторное произведение строк

```
>> cross([1,2,3],[4,5,6])
ans =
    -3     6    -3
```

## Векторное произведение столбцов

```
>> cross([1;2;3],[4;5;6])

ans =
    -3
     6
    -3
```

# Векторное произведение аргументов-матриц

Для аргументов-матриц векторное произведение выполняется для векторов, которые расположены по первой размерности, равной 3:

```
>> a = [1  2  3;
        2  5  6;
        3  8  9];

>> b = [1  1  3;
        2  1  3;
        3  1  2];

>> cross(a,b)
ans =
     0    -3   -15
     0     6    21
     0    -3    -9
```

$$c(:, i) = a(:, i) \times b(:, i)$$

# Векторное произведение аргументов-матриц

Третьим аргументом указывается размерность в которой расположены векторы.

```
>> a = [1  2  3;
        4  5  6;
        7  8  9];

>> b = [1  2  3;
        1  2  3;
        1  2  2];

>> cross(a,b,2)
ans =
     0     0     0
     3    -6     3
    -2    -5     6
```

$$c(i,:) = a(i,:) \times b(i,:)$$

# Скалярное произведение

Векторы-строки

```
>> a = [1  2  3];
>> b = [2  2  2];
>> dot(a,b)
ans =
    12
```

Столбцы и строки

```
>> a = [1; 2; 3];
>> b = [2  2  2];
>> dot(a,b)
ans =
    12
```

# Скалярное произведение

При скалярном произведении матриц матрицы перемножаются по первой размерности не равной 1 (матрицы должны быть одинаковых размеров)

```
>> a = [1  2  3;
        4  5  6;
        7  8  9];

>> b = [1  2  3;
        1  2  3;
        1  2  2];

>> dot(a,b)
ans =
    12    30    45
```

$$c(:, i) = a(:, i) \cdot b(:, i)$$

# Скалярное произведение

Третьим аргументом можно указать явно размерность, в которой расположены векторы в матрице

```
>> a = [1 2 3;
        4 5 6;
        7 8 9];

>> b = [1 2 3;
        1 2 3;
        1 2 2];

>> dot(a,b,2)
ans =
    14
    32
    41
```

$$c(i,:) = a(i,:) \cdot b(i,:)$$

# Сумма элементов вектора или матрицы

`sum(a)`

Сумма элементов вектора:

```
>> a=[1  2  3  4];
>> sum(a)
ans=
    10

```

Сумма элементов матрицы:

```
>> a=[1  2;
      3  4];
>> sum(a)
ans=
    4  6

```

# Сумма элементов матрицы

Сумма элементов матрицы по заданной размерности:

```
>> a=[1  2;
      3  4];
>> sum(a,2)
ans =
    3
    7

```

# Сортировка

- `sort(a)` – сортировка вектора или матрицы.
- Если `a` – матрица, то производится сортировка элементов в столбцах (по первой размерности)

```
>> a = magic(3)
a =
   8     1     6
   3     5     7
   4     9     2
>> sort(a)
ans =
   3     1     2
   4     5     6
   8     9     7
```

# Сортировка

Вторым аргументом функции `sort` может быть номер измерения, по которому должна выполняться сортировка:

```
>> a = magic(3)
a =

     8     1     6
     3     5     7
     4     9     2

>> sort(a,2)
ans =

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

# Сортировка

`[res,i]=sort(a)`. `i` содержит индексы элементов исходного вектора `a`, расположенные по порядку сортировки.

```
>> a = magic(3)
a =
     8     1     6
     3     5     7
     4     9     2
>> [res,i] = sort(a)

res =
     3     1     2
     4     5     6
     8     9     7
i =
     2     1     3
     3     2     1
     1     3     2
```

# Среднее значение

```
>> a = [1  2  6]
>> mean(a)
ans =
      3
```

Среднее в столбцах

```
>> a = [1  2  6;
        7  8  9];
>> mean(a)

ans =
      4    5    6
```

Среднее в строках

```
>> mean(a, 2)

ans =
      2
      8
```

# Максимум

```
>> a = [1  9  6]
>> max(a)
ans =
     9
```

Максимум из двух массивов

```
>> a = [1  9  6]
>> max(a,2)
ans =
     [2   9    6]
```

# Максимум

```
>> a = [1  9  10;
        4  6  1];
>> max(a)
ans =
     4   9   10
```

Размерность, по которой определяется максимум задается третьим аргументом

```
>> max(a,[],2)
ans =
     10
     6
```

# Индексы максимальных значений

```
a = [8      1      6;
     3      5      7;
     4      9      2];
>> [~,i] = max(a)

i =
     1      3      2
```

Аналогичным образом работает функция min

```
>> a = [1  9  10;
        4  6  1];
>> min(a)
ans =
     1   6   1
```

# Множества

# Работа с множествами

`intersect(A, B)` – пересечение векторов A и B как множеств:

```
>> a = [1 2 3 7 8 7 3 1];
>> b = [7 1 5 6];
>> intersect(a,b)
ans =
     1     7
```

`ismember(A, S)` – содержит ли A элементы из S:

```
>> a = [1 2 3 7 8 7 3 1];
>> b = [7 1 5 6];
>> ismember(a,b)
ans =
     1     0     0     1     0     1     0     1
```

# Работа с множествами

`setdiff(A, B)` – элементы, входящие в A, но отсутствующие в B (A-B):

```
>> a = [1 2 3 7 8 7 3 1];
>> b = [7 1 5 6];
>> setdiff(a,b)
ans =
     2     3     8
```

`setxor(A, B)` – элементы, не входящие в результат пересечения множеств A и B:

```
>> a = [1 2 3 7 8 7 3 1];
>> b = [7 1 5 6];
>> setdiff(a,b)
ans =
     2     3     5     6     8
```

# Работа с множествами

`union(A, B)` – объединение множеств:

```
>> a = [1 2 3 7 8 7 3 1];
>> b = [7 1 5 6];
>> union(a,b)
ans =
     1     2     3     5     6     7     8
```

`unique(A)` – элементы вектора A без повторений:

```
>> a = [1 2 3 7 8 7 3 1];
>> unique(a)
ans =
     1     2     3     7     8
```

# Битовые функции

# Битовые функции

`bitand(a,b)` – поразрядное И

```
>> bitand(5,1)
ans =
     1
```

`bitor(a,b)` – поразрядное ИЛИ

```
>> bitor(4,2)
ans =
     6
```

`bitset(a,bit,v)` – установка бита в позиции bit числа a; v = 1 или 0

```
>> bitset(5,3,0)
ans =
     1
```

# Преобразование числа к другому основанию

`dec2bin(a)` – строка с двоичным представлением положительного числа $a$

```
>> dec2bin(5)
ans =
     101

```

`dec2hex(a)` – строка с шестнадцатеричным представлением положительного числа $a$

```
>> dec2hex(255)
ans =
     FF

```

# Преобразование числа к другому основанию

`bin2dec(a)` – преобразование строки с двоичным представлением числа к десятичной системе $a$

```
>> bin2dec('111')
ans =
     7

```

`hex2dec(a)` – преобразование строки с шестнадцатеричным представлением числа к десятичной системе $a$

```
>> hex2dec('F1')
ans =
     241

```

# Ячейки

# Ячейки

Массивы ячеек (cells) могут хранить данные различных типов: числа, строки, структуры, массивы ячеек.

```
>> data{1} = 'Текст'
data =
    'Текст'
>> data{2} = 1
data =
    'Текст'    [1]
>> data{5} = [1 2 3]
data =
    'Текст'    [1]    []    []    [1x3 double]
```

# Доступ к элементам массива ячеек

```
>> data{1}
ans =
      Текст

>> data{2}
ans =
      1

>> data{5}
ans =
      1       2       3
```

# Функция cell

Формирование пустой матрицы ячеек

```
>> a = cell(3,4)
a =
     []     []     []     []
     []     []     []     []
     []     []     []     []
```

Присвоение значения элементу новой матрицы

```
>> a{2,3} = 'элемент 2,3'
a =
     []     []                    []     []
     []     []     'элемент 2,3'       []
     []     []                    []     []
```

# Функции для работы с ячейками

Преобразование числовой матрицы в матрицу ячеек

```
>> a=magic(5)
a =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

```
>> b = mat2cell(a,[2 3],[2 1 2])
b =
    [2x2 double]    [2x1 double]    [2x2 double]
    [3x2 double]    [3x1 double]    [3x2 double]
```

# Функции для работы с ячейками

Преобразование числовой матрицы в матрицу ячеек

```
>> a=magic(5)
a =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

```
>> b = mat2cell(a,[2  3],[2  1  2])
b =
    [2x2 double]    [2x1 double]    [2x2 double]
    [3x2 double]    [3x1 double]    [3x2 double]
```

# Преобразование матрицы ячеек в числовую матрицу

Функция `mat2cell`:

```
>> b = mat2cell(a,[2  3],[2  1  2])
b =
    [2x2 double]    [2x1 double]    [2x2 double]
    [3x2 double]    [3x1 double]    [3x2 double]
>> cell2mat(b)
ans =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

# Структуры

# Структура

Структура – это элемент данных, который может содержать разнотипные поля:

```
имя = struct('поле1',значение,'поле2',значение,...)
```

Структура, описывающая параллелепипед:

```
box=struct('height', 100, 'width', 10, 'depth', 5)
box=
    height: 100
     width: 10
     depth: 5
```

# Доступ к полям структуры

- Считывание значения поля

```
>> box.height
ans =
    100
```

- Имя поля может быть задано строкой

```
>> box.('height')
ans =
    100
```

# Список полей структуры

```
>> names = fieldnames(box)
names =
    'height'
    'width'
    'depth'
    'mass'
```

# Изменение значения поля

```
>> box.height = 20
box =
     height: 20
      width: 10
      depth: 5
```

# Добавление поля

Для добавления поля к структуре необходимо присвоить значение новому полю, как существующему:

```
>> box.mass=5
box =
     height: 20
      width: 10
      depth: 5
       mass: 5
```

# Создание структуры

Создание новой структуры без использования функции `struct`:

```
>> sphere2.radius=2
sphere2 =
    radius: 2
```

# Строки

# Строковые переменные

Строки создаются при помощи одинарных кавычек

```
>> a = 'To be or not to be?'
```

Строка это массив символов

```
>> whos
  Name          Size            Bytes  Class    Attributes

  a             1x19               38  char
```

# Склейка строк

Для склейки двух строк их можно объединить в матрицу-строку:

```
>> S1 = 'String 1 ';
>> S2 = 'String 2';
>> [S1 S2]
ans =

    'String 1 String 2'
```

или использовать функцию `strcat`

```
>> strcat(S1, S2)
ans =

    'String 1String 2'
```

Во втором случае пробелы в конце строк игнорируются.

# Вывод результатов

# Функция disp

Функция disp выводит результат с форматированием по умолчанию.

```
>> a = [1.0  2.5  3.0];
>> disp(a);
    1.0000    2.5000    3.0000

>> disp('Result')
Result
```

Такой же результат можно получить, записав выражение без точки с запятой в конце.

```
>> a = [1.0  2.5  3.0]

a =
    1.0000    2.5000    3.0000
```

# Функция fprintf

Функция fprintf используется для форматированного вывода:

```
>> a = [1.0  2.5  3.0];
>> fprintf('%3.1f  %3.1f  %4.2f\n', a);
1.0  2.5  3.00

>> fprintf('%+6.3f\n', 1.5);
+1.500

>> fprintf('%+6.3f\n', -1.5);
-1.500

>> fprintf('%+08.3f\n', -1.5);
-001.500
```

```
>> fprintf('%3.1f\n', a);
1.0
2.5
3.0
```

# Функция fprintf

```
>> fprintf('%+3.3e\n', -1.5);
-1.500e+00

>> fprintf('Square root of %i is %3.1f\n', 4, sqrt(4));
Square root of 4 is 2.0
```

Функция fprintf может использоваться для записи в файл

```
a = [1 2 3; 4 5 6];
f = fopen('res.txt','w');
fprintf(f, '%5.2f %5.2f %5.2f\n', a(1,:));
fprintf(f, '%5.2f %5.2f %5.2f\n', a(2,:));
fclose(f);
```