

## 2. Операции с вектори и матрици. Работа с масиви.

Основен обект в MATLAB е матрицата. В общия случай, матрицата има m реда и n стълба, т.е размерност m x n. Когато броят на редовете и стълбовете е еднакъв, матрицата е квадратна, с размерност n x n. Вектор-стълб се разглежда като матрица с размерност m x 1. Вектор-ред се разглежда като матрица с размерност 1 x n. Една скаларна стойност (единична стойност) се разглежда като матрица с размерност 1 x 1. Примери за различни матрици са представени на Фиг.1.

$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 1 \end{bmatrix}$
Матрица 3x4	Матрица 3x3	Матрица 1x4	Матрица 4x1	Матрица 1x1 (скаларна стойност)

Фиг.1. Примери за матрици

Командата `>> help elmat` извежда всички функции за действия с матрици.

### 2.1. Въвеждане на вектори и матрици

За правилно поелементно въвеждане на вектори и матрици трябва да се спазват следните правила:

- Въвеждането на елементите започва с отваряща квадратна скоба “ [ “ и завършва със затваряща квадратна скоба “ ] ”;
- Отделните елементи на един вектор ред или в един ред на матрица се разделят помежду си с интервал или със запетайка;
- Отделните редове на матрица или елементите на вектор стълб се разделят помежду си със символа точка и запетайка “ ; ”;
- Вместо да се използва разделител “ ; ” , всеки ред на матрица или всеки елемент на вектор-стълб може да се разполага на нов ред. Този начин на представяне съответства на естественото записване на матрица и вектор и е най-нагледен.

Следващите примери показват различни начини за въвеждане на стойности на матрица а с размерност 3 x 3; на вектор-стълб v1 с размерност 3 x 1 и на вектор-ред v2 с размерност 1 x 7.

```
>> A = [1 1 1 ; 2 2 2 ; 3 3 3] % разделител между елементите е интервал
```

A =

```
1     1     1
2     2     2
3     3     3
```

```
>> A = [1,1,1 ; 2,2,2 ; 3,3,3] % разделител между елементите е запетая
```

A =

```
1 1 1
2 2 2
3 3 3
```

```
>> A = [ 1 1 1
2 2 2
3 3 3 ] % разполагане на всеки ред на нов ред
```

A =

```
1 1 1
2 2 2
3 3 3
```

```
>> v2 = [ 7 6 5 4 3 2 1 ] % задаване на вектор-ред
```

v2 =

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

```
>> v1=[7 ; 8 ; 9] % задаване на вектор-стълб
```

v1 =

```
7
8
9
```

```
>> v3=[9 8 7]' % задаване на вектор-стълб чрез транспониране на вектор-ред
```

v3 =

```
9
8
7
```

Възможно е стойностите на елементите на матрица да се въведат еднократно или на порции (многократно), след което да се обединят по подходящ начин.

### Примери:

```
>>v = [1.2 -0.3 1.2e-5] % еднократно въвеждане
```

v =

```
1.2000 -0.3000 0.0000
```

```
>> v1 = [1 2 3];
>> v2 = [5 6 7];           % многократно въвеждане

>> v = [v1, v2]           % конкатенация

v =

     1     2     3     5     6     7
```

Стойности на вектори, чиито елементи са аритметични прогресии, се въвеждат по точно определен начин:

*име на променлива = начална стойност :стъпка на нарастване : крайна стойност*

```
>> z = 1 : 2 : 9

z =

     1     3     5     7     9
```

Ако липсва един параметър, то се подразбира, че стъпката на нарастване е 1, а зададените стойности са началната и крайната.

```
>> y = 2 : 5

y =

     2     3     4     5
```

## 2.2. Генериране на вектори и матрици

В MATLAB са включени специални функции за автоматично генериране на вектори и матрици от специален вид. Най-често използвани са следните функции:

- **zeros(m, n)** – матрица с размерност  $m \times n$ , запълнена с нули;
- **ones(m, n)** - матрица с размерност  $m \times n$ , запълнена с единици;
- **eye(m, n)** - матрица с размерност  $m \times n$ , с единици по главния диагонал, а останалите елементи са нули;
- **rand(m, n)** - матрица с размерност  $m \times n$ , запълнена със случайни числа, равномерно разпределени в интервала от 0 до 1;
- **randn(m, n)** - матрица с размерност  $m \times n$ , запълнена със случайни числа, разпределени по нормален закон;

Когато се генерира квадратна матрица може вместо размерност  $n \times n$  да се зададе само един аргумент  $n$ , например **zeros(n)**.

### Примери:

```
>>zeros(3, 4) % създава матрица с размери 3 x 4 само с нулеви елементи
```

```
ans =  
    0 0 0 0  
    0 0 0 0  
    0 0 0 0
```

```
>>ones(3, 4) % създава матрица с размери 3 x 4 само с единични  
елементи
```

```
ans =  
    1 1 1 1  
    1 1 1 1  
    1 1 1 1
```

```
>>eye(3) % създава квадратна матрица с размери 3 x 3 с единици по  
главния диагонал, а останалите елементи са нулеви
```

```
ans =  
    1 0 0  
    0 1 0  
    0 0 1
```

```
>>randn(5, 3) % създава матрица с размерност 5 x 3, запълнена със  
случайни числа, разпределени по нормален закон.
```

```
ans =  
 -0.43256481152822    1.19091546564300   -0.18670857768144  
 -1.66558437823810    1.18916420165210    0.72579054829330  
  0.12533230647483   -0.03763327659332   -0.58831654301419  
  0.28767642035855    0.32729236140865    2.18318581819710  
 -1.14647135068146    0.17463914282092   -0.13639588308660
```

### 2.3. Определяне на размера на вектори и матрици

В MATLAB са включени две специални функции за определяне на размера на вектори и матрици.

- **length(vector1)** - определяне на дължината на вектора vector1;
- **size(matrix1)** - определяне на размера на матрица matrix1.

**Пример:**

```
>> vector1=[0 9 8 7 6 5 4 3 2 1];  
>> matrix1=[1 1 1;2 2 2; 3 3 3];  
>> length(vector1)
```

```
ans =  
    10
```

```
>> size(matrix1)
```

```
ans =  
     3     3
```

## 2.4. Преобразуване на вектори и матрици

В MATLAB са включени функции за преобразуване на вектори и матрици от един вид в друг вид. Някои от тези функции са:

- **flip1r(A)** – огледален образ спрямо вертикална ос (left right);
- **flipud(A)** – огледален образ спрямо хоризонтална ос (up down);
- **rot90(A)** – завъртане на  $90^\circ$  обратно на часовата стрелка;
- **reshape(A, m, n)** – преобразува матрицата A в матрица с размери  $m \times n$ , като избира и подрежда елементите по стълбове; броят на елементите на матрицата A трябва да е равен на  $m*n$ ;
- **tril(A)** – връща долната триъгълна част на матрицата A (lower);
- **triu(A)** – връща горната триъгълна част на матрицата A (upper);
- **diag(v)** – образува диагонална матрица от елементите на вектора v, разположени върху главния диагонал или извлича главния диагонал на матрица.

### Примери:

```
>> s=[3 3 3; 4 4 4; 5 5 5]
```

```
s =  
    3     3     3  
    4     4     4  
    5     5     5
```

```
>> flipud(s)
```

```
ans =  
    5     5     5  
    4     4     4  
    3     3     3
```

```
>> diag(s)
```

```
ans =  
    3  
    4  
    5
```

## 2.5. Извличане, вмъкване и премахване на части от матрица. Работа с отделен елемент.

В MATLAB е възможно обръщане към отделен елемент на матрица. Това става чрез записване на името на матрицата и в кръгли скоби индекса на реда i и индекса на стълба j на който се намира елементът, разделени със запетая – A(i, j).

### Пример:

- **A(2, 5)** - елемент от матрица A, който се намира на ред 2 и стълб 5;
- **c(3)** – третият елемент от вектор c ;

Възможно е да се присвои стойност на отделен елемент на матрица. В този случай се записва елементът по показания начин и на него се присвоява желаната стойност.

```
>>A(2, 3) = 5.5555; A
```

```
A =  
1.0000 2.0000 3.0000 4.0000  
5.0000 6.0000 5.5555 8.0000  
9.0000 10.0000 11.0000 12.0000
```

От този пример се вижда, че в матрица A се е променила стойността само на елемент A(2, 3).

В MATLAB е възможно също обръщане към отделен ред, стълб или блок от матрица. В тези случаи се използва символ двоеточие " : ". Ако вместо индекс се постави символът " : ", това означава целия диапазон от възможни стойности на индекса. Това записване е еквивалентно на записването " 1:end ". Записването " i1:i2 " означава всички елементи с индекс на ред от i1 до i2; записването " j1:j2 " означава всички елементи с индекс на стълб от j1 до j2.

- $\mathbf{v1} = \mathbf{A}(:, j)$  – вектор-стълб, състоящ се от елементите на j-тия стълб на матрицата A;
- $\mathbf{v2} = \mathbf{A}(i, :)$  – вектор-ред, състоящ се от елементите на i-тия ред на матрицата A;
- $\mathbf{B} = \mathbf{A}(i1:i2, j1:j2)$  – матрица с елементи, разположени на местата на пресичане на редове с номера от i1 до i2 със стълбове с номера от j1 до j2 на матрицата A;
- $\mathbf{A}(2:4, 3:5) = \mathbf{B}$  – копиране на матрицата B с размерност 3x3 в матрица A на пресечното място на редове от 2 до 4 и стълбове от 3 до 5. Старите елементи на A се заменят с новите стойности.

```
>> s1=s(2, :)
```

```
s1 =  
4 4 4
```

```
>> s2=s(:, 2)
```

```
s2 =  
3  
4  
5
```

```
>> s3=s(1:2, 1:2)
```

```
s3 =  
3 3  
4 4
```

Преобразуване на матрица във вектор стълб:

```
>> sv=s(:)
```

sv =

3  
4  
5  
3  
4  
5  
3  
4  
5

Премахването на части от вектор или матрица става, като се присвои на съответната част символът за празен масив " [] ". При това "изрязване" размерът на вектора или матрицата се намалява автоматично.

- $v(1:3) = []$  – премахва първите три елемента на вектора  $v$ ;
- $A(i, :) = []$  – премахва  $i$ -тия ред на матрицата  $A$ ;
- $A(:, j) = []$  – премахва  $j$ -тия стълб на матрицата  $A$ ;
- $A(:, 3:7) = []$  – премахва от 3-ти до 7-и стълб;
- $A(:, [1 5 9]) = []$  – премахва 1-ви, 5-и и 9-и стълб.

Тъй като в паметта на компютъра матриците се съхраняват като вектор, в който елементите се разполагат последователно по стълбове, при обръщение към дадена част от матрицата може да се използва само един индекс:

- $A(n)$  – извличане на  $n$ -тия елемент от вектора, представляващ матрицата  $A$ ;
- $A(3:9)$  – извличане на елементите от 3-ти до 9-и;
- $A(:)$  – извличане на цялата матрица като вектор-стълб.

### Пример:

```
>> s(2, :) = []
```

s =

3      3      3  
5      5      5

## 2.6. Логическо индексване

Много често в практически задачи се налага да се обработват само тези елементи на матрица, които отговарят на определено условие. Извличане на елементи на вектори и матрици, удовлетворяващи определени условия се извършва чрез така нареченото логическо индексване. Това се постига като вместо индекси, които определят извличаните части според тяхното разположение в матрицата, се използват логическа функция или логическо отношение, в които влиза името на същия масив.

Общият вид на логическо индексване е:

$$B = A(\text{logical\_expression} ( A ))$$

### Пример:

```
s =  
    3    3    3  
    5    5    5
```

```
>> s (s>4)
```

```
ans =  
    5  
    5  
    5
```

```
>> s (s<5)
```

```
ans =  
    3  
    3  
    3
```

Подробности за работа с елементи на матрици, които отговарят на определено условие, ще се разгледат при темата за управляващи оператори в MATLAB.

## 2.7. Действия с вектори и матрици

Когато се обработват матрици в MATLAB се разграничават две групи действия. В първата група са включени действия с вектори и матрици по правилата на линейната алгебра. Във втората група са включени действия, които се изпълняват с всеки елемент на матрицата и като резултат я преобразуват в друга матрица със същия размер. Тъй като в този случай се обработва матрицата елемент по елемент тези операции се наричат поелементни операции. Поелементните операции се наричат още и операции за работа с масиви.

### 2.7.1. Действия с вектори и матрици от линейната алгебра

Основните операции с вектори и матрици от линейната алгебра се извършват в MATLAB с помощта на аритметични оператори +, -, \*, ^, ':

- $A + B$  – събиране;
- $A - B$  – изваждане;
- $A * B$  – умножение;
- $A^n$  – степенуване ( $n$  – цяло число,  $A^2$  е еквивалентно на  $A * A$ );
- $A'$  – пресмятане на комплексно спрегната матрица;

При прилагането на тези действия е важно да се спазват правилата, при които те са възможни:

- При събиране или изваждане векторите или матриците трябва да са с еднакъв размер.
- При умножение броят на стълбовете на първия множител трябва да е равен на броя на редовете на втория.
- На степен могат да се повдигат само квадратни матрици.

### Примери за операции с вектори:

#### Събиране и изваждане

```
>>x = [1 2 3]; y = [4 5 6];
>>v = x + y
v =
    5    7    9
```

```
>>v = x - y
v =
   -3   -3   -3
```

### Транспониране

```
>>x'
ans =
     1
     2
     3
```

### Умножаване на вектор със скалар

```
>>v = 2*x
v =
     2     4     6
```

### 2.7.2. Поелементни операции с вектори и матрици

В MATLAB съществуват специфични операции с матрици, които се изпълняват над всеки елемент на матрицата и я преобразуват в друга матрица със същия размер. Това са така наречените поелементни операции. Такива операции извършват всички елементарни функции. Например, ако  $\mathbf{v} = [\mathbf{v1}, \mathbf{v2}, \dots, \mathbf{vn}]$ , то  $\mathbf{sin}(\mathbf{v})$  ще представлява векторът  $[\mathbf{sin}(\mathbf{v1}), \mathbf{sin}(\mathbf{v2}), \dots, \mathbf{sin}(\mathbf{vn})]$ .

Като се добави една точка "." пред обикновените математически оператори \*, /, \, ^ и ', се получават следните поелементни оператори:

- .\* – поелементно умножение;
- ./ – поелементно деление отдясно;
- .\ – поелементно деление отляво;
).^ – поелементно степенуване;
- .' – транспониране на вектор или матрица.

Важно е да се спазват следните правила:

- Обикновените оператори "+" и "-" също имат поведение на поелементни, без да е необходимо да добавяме точка пред тях.
- При поелементното умножение, деление и степенуване единият от операндите може да бъде и скалар.
- Ако и двата операнда са вектори или матрици, те трябва да са с еднакъв размер.
- Ако елементите на вектора или матрицата са реални числа, то A' е еквивалентно на A.!

### Примери:

$C = A + b$  – прибавяне на скалара b към всеки елемент на матрицата A:  $c_{ij} = a_{ij} + b$ ;  
 $C = A .* B$  – поелементно умножение на две матрици:  $c_{ij} = a_{ij} * b_{ij}$ ;  
 $C = A ./ B$  – поелементно деление на две матрици:  $c_{ij} = a_{ij} / b_{ij}$ ;  
 $C = A .\ B$  – поелементно деление на две матрици:  $c_{ij} = b_{ij} / a_{ij}$ ;

$C = A.^B$  – поелементно степенуване:  $c_{ij} = a_{ij}^{b_{ij}}$ ;  
 $C = A.^b$  – поелементно степенуване ( $b$  е скалар):  $c_{ij} = a_{ij}^b$ ;  
 $C = b.^A$  – поелементно степенуване:  $c_{ij} = b^{a_{ij}}$ .

Поелементните операции на системата MATLAB определят нейната изключителна мощ. Те позволяват много лесно и бързо да се пресмятат стойностите на сложни изрази (функции) за множество от стойности на параметрите (аргументите), без необходимост от цикли.

```
>> b=[2 2 2; 3 3 3; 4 4 4]
```

```
b =  
    2     2     2  
    3     3     3  
    4     4     4
```

```
>> b*2
```

```
ans =  
    4     4     4  
    6     6     6  
    8     8     8
```

```
>> b^2
```

```
ans =  
   18    18    18  
   27    27    27  
   36    36    36
```

```
>> b.^2
```

```
ans =  
    4     4     4  
    9     9     9  
   16    16    16
```

## 2.8. Обработка на експериментални данни

MATLAB предлага допълнителни функции, които са подходящи за обработка на експериментални данни, представени чрез вектори и матрици. Пълен списък на тези функции се получава с командата **>> help datafun**.

Тези функции могат да се прилагат и за вектори, и за матрици. При работа с вектори действието на тези функции е следното:

- **max(v)** - изчисляване на максималния елемент;
- **min(v)** - изчисляване на минималния елемент;
- **mean(v)** – изчисляване на средната стойност;

- **std(v)** – изчисляване на средното квадратично отклонение;
- **sort(v)** – подреждане елементите на масива във възходящ ред;
- **sum(v)** – пресмята сумата от всички елементи;
- **prod(v)** – изчисляване произведението от елементите на масива;

Функциите **min** и **max** могат да се използват и с два изходни аргумента – първият, **M**, ще бъде съответно минималният или максималният елемент, а вторият – неговият индекс **n**:

- **[M, n] = min(v)**
- **[M, n] = max(v)**

Всички описани дотук функции могат да се прилагат и към матрици. В този случай всяка от функциите обработва всеки стълб на матрицата така, както обработва един вектор, т.е. матрицата се обработва по стълбове. Например функцията **min(A)** ще върне вектор с минималните елементи от всеки стълб на матрицата **A**. При функцията **sum** е възможно сумиране по редове и по стълбове.

- **sum(A)** - сума на елементите по стълбове;
- **sum(A, 2)** - сума на елементите по редове.

### Пример:

```
>> z=[1 1 1; 2 2 2; 3 3 3]
```

```
z =
     1     1     1
     2     2     2
     3     3     3
```

```
>> sum(z)
```

```
ans =
     6     6     6
```

```
>> sum(z,2)
```

```
ans =
     3
     6
     9
```

MATLAB включва и функции за закръгляне към цяло число и остатъци:

- **fix** – закръгляне в посока към нулата;
- **floor** – закръгляне надолу;
- **ceil** – закръгляне нагоре;
- **round** – закръгляне към най-близкото цяло число;
- **rem** – остатъка от деленето на две реални числа;
- **mod** – модул при деленето на две реални числа;
- **sign** – връща знака на аргумента.

**Пример:**

```
>> z1=-0.789
```

```
z1 =
```

```
    -0.79
```

```
>> z2=0.789
```

```
z2 =
```

```
    0.79
```

```
>> fix(z1)
```

```
ans =
```

```
    0
```

```
>> fix(z2)
```

```
ans =
```

```
    0
```

```
>> ceil(z1)
```

```
ans =
```

```
    0
```

```
>> ceil(z2)
```

```
ans =
```

```
    1.00
```

```
>> round(z1)
```

```
ans =
```

```
   -1.00
```

```
>> round(z2)
```

```
ans =
```

1.00