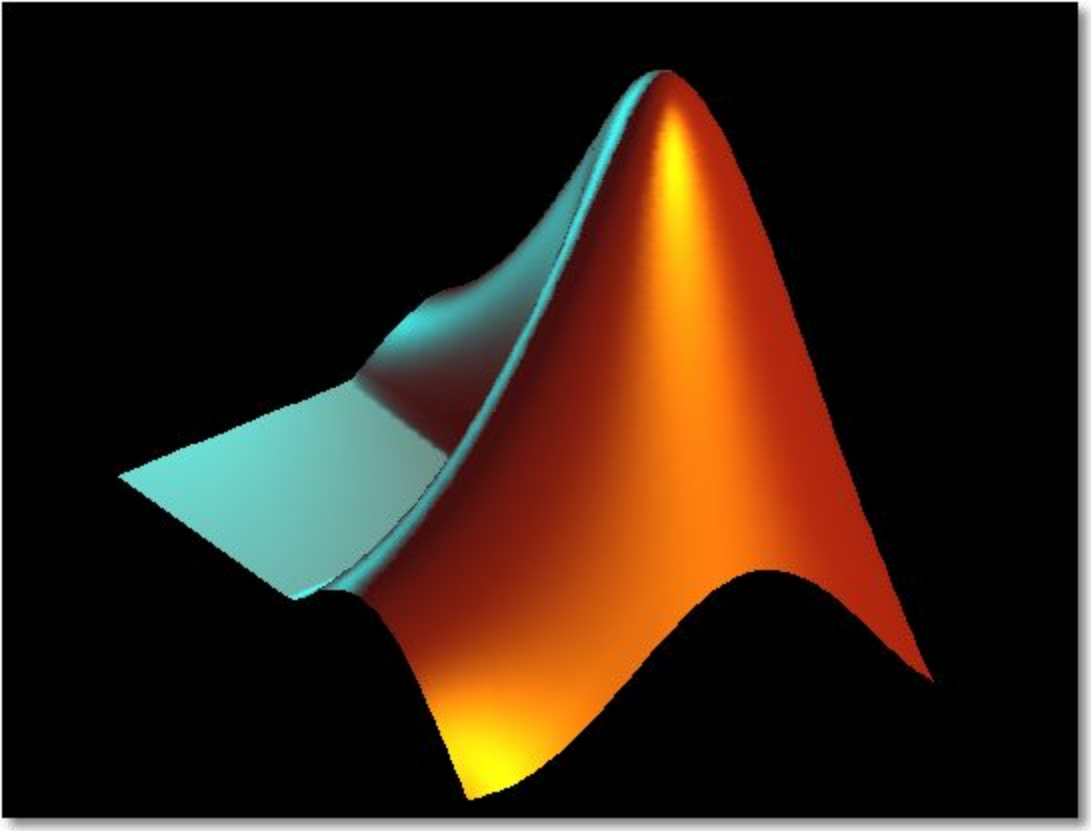


Лекция 3

Функции для работы с массивами в Matlab

Умножение матриц

```
>> a = [1 10 4 3; 1 4 7 3; 1 8 5 1]
a =
1    10     4     3
1     4     7     3
1     8     5     1
>> b = [3 4; 1 8; 3 2; 3 1]
b =
3     4
1     8
3     2
3     1
>> c = a*b
c =
34    95
37    53
29    79
```

Умножение матрицы на число

```
>> a = [1 10 4 3; 1 4 7 3; 1 8 5 1]
a =
1    10     4     3
1     4     7     3
1     8     5     1
>> a.*2
ans =
2    20     8     6
2     8    14     6
2    16    10     2
```

Транспонирование вещественных матриц

Транспонирование матрицы, так же как и вектора, производится с помощью символа ‘ (апостроф).

Пример.

```
>> a = [1 10 4 3; 1 4 7 3; 1 8 5 1]
a =
1    10     4     3
1     4     7     3
1     8     5     1

>> a'
ans =
1     1     1
10    4     8
4     7     5
3     3     1
```

Транспонирование матриц, содержащих комплексные числа

Эта операция выполняется командой ‘ (апостроф). При транспонировании комплексные числа заменяются на комплексно сопряженные.

Пример.

```
>> a = [1+2i 3-2i; 5+i 4-4i]
a =
1.0000 + 2.0000i   3.0000 - 2.0000i
5.0000 + 1.0000i   4.0000 - 4.0000i

>> a'
ans =
1.0000 - 2.0000i   5.0000 - 1.0000i
3.0000 + 2.0000i   4.0000 + 4.0000i
```

Возведение матрицы в степень

Операция возведения матрицы в степень осуществляется командой ^. Только квадратные матрицы могут быть возведены в степень.

Пример.

```
>> a = [1 10 4; 1 4 7; 1 8 5]
a =
1    10     4
1     4     7
1     8     5

>> a^2
ans =
15    82    94
12    82    67
14    82    85
```

Поэлементные операции с матрицами

$$A = \begin{pmatrix} 2 & 5 & -1 \\ 3 & 4 & 9 \end{pmatrix}$$

$$B = \begin{pmatrix} -1 & 2 & 8 \\ 7 & -3 & -5 \end{pmatrix}$$

```
>> C=A.*B

C=

    -2   10   -8

    21  -12  -45
```

Создание матриц специального вида

- Для работы с матрицами удобно пользоваться следующими функциями
  - **`ones`** – формирование массива из единиц
  - **`zeros`** – формирование массива из нулей
  - **`eye`** – формирование единичной матрицы
  - **`rand`** – формирование массива из чисел, случайно распределённых на отрезке [0, 1]
  - **`randn`** – формирование массива из чисел, нормально распределённых на отрезке [-1, 1]
  - **`magic`** – формирование магического квадрата
  - **`pascal`** – формирование квадрата Паскаля
  - **`diag`** – диагональная матрица
  - и др.

Матрицы специального вида

- Рассмотрим основной синтаксис на примере функции создания единичной матрицы (`eye`)
- **`eye(m)`** – создание единичной матрицы размера [m, m]
- **`eye(m,n)`** – создание единичной матрицы размера [*m, n*]
  - «лишние» строки или столбцы дополняются нулями

Матрицы специального вида

```
>> a = eye(4)

a =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1

>> a = eye(4, 6)

a =

     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0
     0     0     0     1     0     0
```

Матрицы специального вида

```
>> z = zeros(4)

z =

     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0

>> z = zeros(3, 4)

z =

     0     0     0     0
     0     0     0     0
     0     0     0     0
```

```
>> z = ones(5)

z =

     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1

>> z = ones(5, 3) * 7

z =

     7     7     7
     7     7     7
     7     7     7
     7     7     7
     7     7     7
```

Матрицы специального вида

```
>> rand(4)

ans =

    0.9355    0.0579    0.1389    0.2722
    0.9169    0.3529    0.2028    0.1988
    0.4103    0.8132    0.1987    0.0153
    0.8936    0.0099    0.6038    0.7468

>> randn(4)

ans =

   -0.4326   -1.1465    0.3273   -0.5883
   -1.6656    1.1909    0.1746    2.1832
    0.1253    1.1892   -0.1867   -0.1364
    0.2877   -0.0376    0.7258    0.1139
```

Матрицы специального вида

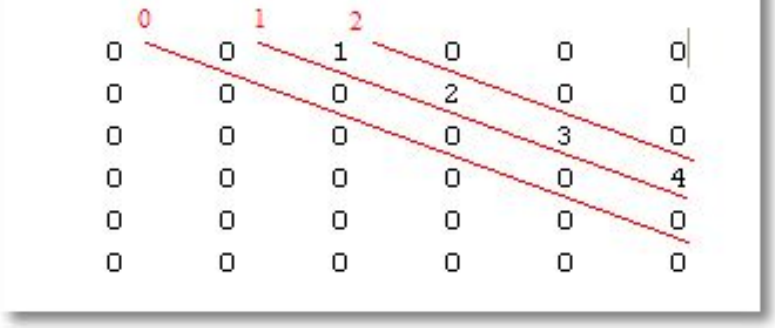
```
>> magic(3)

ans =

     8     1     6
     3     5     7
     4     9     2

>> pascal(6)

ans =

     1     1     1     1     1     1
     1     2     3     4     5     6
     1     3     6    10    15    21
     1     4    10    20    35    56
     1     5    15    35    70   126
     1     6    21    56   126   252
```

Матрицы специального вида

- Функция **`diag`**: работа с диагональными матрицами
  - у которых ненулевые элементы расположены на диагоналях
- Синтаксис:
  - **`X = diag(v)`** – на главной диагонали матрицы **`X`** расположены элементы вектора **`v`**
  - **`X = diag(v,k)`** – на **`k`**-ой диагонали матрицы **`X`** расположены элементы вектора **`v`** (по умолчанию k=0)
  - **`v = diag(X,k)`** – извлечь из матрицы X k-ую диагональ и сохранить её в векторе v

Матрицы специального вида

```
>> v = 1:4

v =

     1     2     3     4

>> A = diag(v)

A =

     1     0     0     0
     0     2     0     0
     0     0     3     0
     0     0     0     4

>> B = diag(v, 2)

B =
```




```
>> B = diag(v, -1)

B =

     0     0     0     0     0
     1     0     0     0     0
     0     2     0     0     0
     0     0     3     0     0
     0     0     0     4     0

>> M = magic(6)

M =

    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11

>> u = diag(M, 1)

u =

     1
     7
    22
    10
    16
```

Матрицы специального вида

```
>> m = 3

m =

     3

>> A = diag(-m:m)+diag(ones(2*m,1),1)+diag(ones(2*m,1),-1)

A =

    -3     1     0     0     0     0     0
     1    -2     1     0     0     0     0
     0     1    -1     1     0     0     0
     0     0     1     0     1     0     0
     0     0     0     1     1     1     0
     0     0     0     0     1     2     1
     0     0     0     0     0     1     3
```

Вычисления с элементами массивов

- Простейшие операции над элементами массивов:
  - **`sum`**: сумма элементов
  - **`prod`**: произведение элементов
  - **`cumsum`**: кумулятивная сумма элементов
  - **`cumprod`**: кумулятивное произведение элементов
  - **`max`**: нахождение максимального элемента
  - **`min`**: нахождение минимального элемента
  - **`sort`**: сортировка элементов

Вычисления с элементами массивов

- Рассмотрим работу некоторых из этих функций на примере **`sum`**
- Для векторов эта функция возвращает сумму элементов.
- Для массивов – сумму элементов по каждому из столбцов
  - *результат – вектор-строка*

Суммирование элементов массива можно проводить командой:

sum(A,[ ],dim) – возвращает сумму элементов массива по столбцам (dim=1), по строкам (dim=2).

Вычисления с элементами массивов

```
>> A = round(10*rand(4,5)-3)

A =

     3     1     2     5     5
     1    -1     3     4    -2
     2     3    -1     2     3
     0     5     1     3    -2

>> v = sum(A)

v =

     6     8     5    14     4
```

```
>> sum(v)

ans =

    37

>> sum(sum(A))

ans =

    37
```

Здесь **round**(X) возвращает значения, округленные до ближайшего целого.

Произведение элементов массива

Синтаксис: **prod(X), cumprod(X)**

Описание:

Функция **prod(X)** в случае одномерного массива возвращает произведение элементов массива; в случае двумерного массива - это вектор-строка, содержащая произведения элементов каждого столбца.

Функция **cumprod(X)**, кроме того, возвращает все промежуточные результаты.

Пример:

Рассмотрим массив M = magic(3).

M = 8 1 6

3 5 7

4 9 2

cumprod(M)	prod(M)
8 1 6 24 5 42 96 45 84	96 45 84

Произведение элементов массива можно проводить командой:

prod(A,[ ],dim) – возвращает матрицу (массив размерности два) с произведение элементов массива A по столбцам (dim=1), по строкам (dim=2).

Поворот матрицы

rot90(A) - осуществляет поворот матрицы A на 90° против часовой стрелки;

rot90(A,k) - осуществляет поворот матрицы A на величину 90**k* градусов, где *k* — целое число.

Пример:

Рассмотрим массив M = magic(3).

M = 8 1 6
3 5 7
4 9 2

>>rot90(M)

M=

6 7 2
1 5 9
8 3 4

Вычисления с элементами массивов

- Кумулятивная сумма вычисляется так же, только происходит накопление вычисленных значений в элементах массива:

```
>> 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

>> cumsum(A)

ans =

    17    24     1     8    15
    40    29     8    22    31
    44    35    21    42    53
    54    47    40    63    56
    65    65    65    65    65
```

Вычисления с элементами массивов

- Максимальный и минимальный элементы:

```
>> A = round(100*rand(6,5)-40)

A =

    41   -23    47   -15    21
    21    43    37    -5   -33
    30    44     4   -21    -9
   -31     5    22     9    21
     2    56    55     1   -22
    -2   -25    24     6    22

>> max(A)

ans =

    41    56    55     9    22

>> [max(min(A)), min(max(A))]

ans =

     4     9
```

Вычисления с элементами массивов

- Вызов функций `max/min` с двумя выходными параметрами позволяет определить и индекс найденного элемента:

```
A =

   -15    -6    10   -39    38
    19     0    32    26    59
    11    -9    -9    32     7
     6     1   -29   -12    50
    14   -11     4   -14     5
    54    -1     7    31    40

>> [a,i]=max(A)

a =

    54     1    32    32    59


i =

     6     4     2     3     2
```

Вычисления с элементами массивов

- Функция

`sort(имя массива)`

производит сортировку элементов матрицы по возрастанию.

Сортировка производится по столбцам:

```
a =
    41   -12    56    39    28
    51    15     9    56    36
   -27    56    40    26    34
    51    56   -26   -36    -1
    23   -24     2    45    26
   -30    57    52    53   -23
>> sort(a)
ans =
   -30   -24   -26   -36   -23
   -27   -12     2    26    -1
    23    15     9    39    26
    41    56    40    45    28
    51    56    52    53    34
    51    57    56    56    36
```

Вычисления с элементами массивов

Сортировка элементов массива по убыванию осуществляется с помощью искусственного приема:

-sort(-имя массива)

```
a =
    41   -12    56    39    28
    51    15     9    56    36
   -27    56    40    26    34
    51    56   -26   -36    -1
    23   -24     2    45    26
   -30    57    52    53   -23
>> -sort(-a)
ans =
    51    57    56    56    36
    51    56    52    53    34
    41    56    40    45    28
    23    15     9    39    26
   -27   -12     2    26    -1
   -30   -24   -26   -36   -23
```

Логические функции

- **`All(v)`** – возвращает истину, если все элементы вектора v отличны от нуля. Для матриц выдаёт вектор-строку с аналогичным результатом для каждого столбца

```
>> M = magic(4)-7

M =

     9    -5    -4     6
    -2     4     3     1
     2     0    -1     5
    -3     7     8    -6

>> all(M)

ans =

     1     0     1     1
```

Логические функции

- **`Any(v)`** – возвращает истину, если хотя бы один элемент вектора v отличен от нуля. Для матриц выдаёт вектор-строку с аналогичным результатом для каждого столбца

```
>> M=zeros(5); M(1,1)=3; M(2,3)=5

M =

     3     0     0     0     0
     0     0     5     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0

>> any(M)

ans =

     1     0     1     0     0
```

Логические функции

```
>> v=round(20*rand(1,8)-5)

v =

     5    15     2     6    -1     5     3     8

>> v>7

ans =

     0     1     0     0     0     0     0     1

>> v(v>7)

ans =

    15     8

>> v(v>5 & v<=8)

ans =

     6     8
```

Поиск в массиве

- **`find`**: определяет индексы элементов, удовлетворяющих заданному условию

```
>> v

v =

     5    15     2     6    -1     5     3     8

>> find(v>7)

ans =

     2     8

>> v(find(v>7))

ans =

    15     8
```

Поиск в массиве

- Пример применения команды **`find`** к массивам:

```
>> a = magic(3)

a =

     8     1     6
     3     5     7
     4     9     2

>> k = find(a>5);
>> [w, i] = find(a>5);
>> [k w i]

ans =

     1     1     1
     6     3     2
     7     1     3
     8     2     3
```

Поиск в массиве

Пример. Найти индексы максимального элемента в массиве.

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

>> m=max(max(a));
>> [i,j]=find(a==m)

i =
     3

j =
     2
```

Математические матричные операции

- **det** – вычисление определителя квадратной матрицы

```
>> format long
>> A = round(100*rand(6)-40)

A =

   -21    -7    22    57    40   -12
     5    48    29    42    27   -33
   -39     8    11    -8   -39     8
    -9    16    31    19    16    58
    48    22    12   -27     5    52
    44    26    21   -15    50    16

>> det(A)

ans =

   -5.225668154000000e+009
```

Матричные и поэлементные операции

- При работе с матрицами можно использовать два вида операторов:
  - *матричные:* производят действия по правилам матричной алгебры.
  - *поэлементные:* производят действия над соответствующими элементами матриц
    - размеры матриц должны быть одинаковыми;
    - от матричных операций отличаются точкой перед знаком операции.

Матричные и поэлементные операции

- ‘ транспонирование
- + матричное (и поэлементное) сложение
- - матричное (и поэлементное) вычитание
- * матричное умножение
- / матричное деление
- ^ матричное возведение в степень
- \ матричное деление «слева»
- .* поэлементное умножение
- ./ поэлементное деление
- .^ поэлементное возведение в степень
- .\ поэлементное деление «слева»

Матричные и поэлементные операции

```
>> a = [1 2; 3 4]

a =

     1     2
     3     4

>> b = [3 1; 2 0]

b =

     3     1
     2     0
```

```
>> a*b

ans =

     7     1
    17     3

>> a.*b

ans =

     3     2
     6     0
```

Матричные и поэлементные операции

- Такие операции часто используются, если нужно применить какую либо функцию ко всем элементам матрицы.

```
>> x = [1 2 3 4 5]

x =

     1     2     3     4     5

>> 1/x^2
??? Error using ==> mpower
Matrix must be square.

>> 1./x.^2

ans =

    1.0000    0.2500    0.1111    0.0625    0.0400

>> format rat
>> 1./x.^2

ans =

       1          1/4        1/9        1/16       1/25
```

Матричные и поэлементные операции

- Некоторые операции по умолчанию считаются поэлементными:

```
>> a = magic(3)

a =

     8     1     6
     3     5     7
     4     9     2

>> a+1

ans =

     9     2     7
     4     6     8
     5    10     3

>> sin(a)

ans =

    0.9894    0.8415   -0.2794
    0.1411   -0.9589    0.6570
   -0.7568    0.4121    0.9093
```