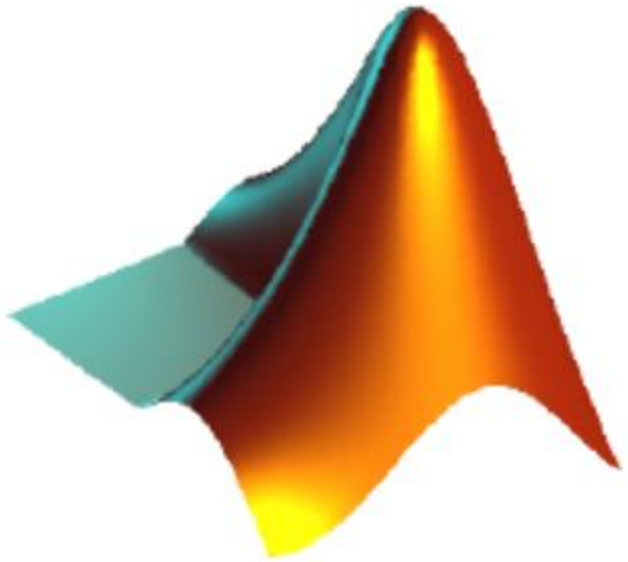
Arrays
MatLab
(2)



Variables

- In MATLAB environment, **every variable is an array or matrix**
- **Rules for declaring a variable:**
  - **Don't have** to declare **type** but just assign it
  - Matlab is **case sensitive**; it distinguishes between uppercase and lowercase letters. **A and a are not the same variable**

```
>>a=6, A=0
   a=6
   A=0
```

  - Variable name consist of a letter, followed by any number of letters, digits, or underscores
  - **Avoid Matlab Keywords**, for instance if, end. For a complete list, run the **iskeyword** command.
  - **Avoid conflicts with Function Names**, for instance max, min, count.

valid variable names

```
>>X6=3, lastValue =9, n_factorial =0;
```

Invalid variable names:

```
>>6x=0, end=7, n!=9, max=8;
Error: Unexpected MATLAB expression.
```

Variables Examples

>> a=5 % **define variable** 'a' by assigning it the value 5

a = 5

>> sqrt(16)

ans=??

Comment are written using %

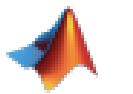
>> ans *2

ans=??

>> b=2+a;

Strings are defined using single quote '

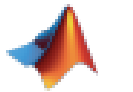
>> c=cos(b); d=c ^5; s='hello'



More in Variables

- **To display information** about variables in workspace, 'whos' command is used.

```
>> whos  % Variables 'a, ans, b, c, d, s' are matrices.
  Name      Size            Bytes  Class     Attributes

   a         1x1                 8  double
 ans         1x1                 8  double
   b         1x1                 8  double
   c         1x1                 8  double
   d         1x1                 8  double
   s         1x5                10  char
```



1-D Array
Vectors (Row Vector)

- **Two ways for creating row vectors:**

  1- i. Enclose the set of elements in square brackets

  ii. Delimit element using space or comma. For example:

```
>> rv=[-1 sin(3)  7]       %  equivalent to rv=[-1, sin(3), 7]
   rv= -1.0000  0.1411   7.0000
```

  2- Using the colon notation **start: step: end**. The default value of step is 1; which is used when step is omitted

```
>>v=2:10      % use the colon notation (:) with step =1, start : end
   v= 2  3  4  5  6  7  8  9 10

>>v = 20:-1:16    % use the colon notation, start:step:end
   v= 20  19  18  17  16
```

- **Concatenating two row vectors using comma or space:**

```
>>u=[v 2 3, v]    % horizontal concatenation using comma or space
   u= 20  19  18  17  2  3  20  19  18  17  16
```

1-D Array (Vectors)
Column Vector

- **Creating a column vector:**
  1. Enclose the set of elements in square brackets
  2. Delimit element using (;)or enter. For example:

```
>> cv=[20
       30
       40]          %  equivalent to cv=[20; 30; 40]
cv= 20
    30
    40
```

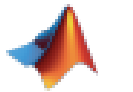
- **Concatenating two column vectors using semicolon**

```
>>u=[cv; 4; 3]        % vertical concatenation using ;
u=  20
    30
    40
    4
    3
```

2-D Array Creation (Matrices)

- **2-D array creation:**
  - **All elements** must be of **the same datatype** and **size.**
  - **Commas or spaces** are used to **separate elements** in a **row**.
  - **Semicolons** are used to separate individual rows.

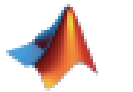
```
>> A = [1 2 3; 4 5 6; 7 8 9]     % row by row input
    A =  1 2 3
         4 5 6
         7 8 9

>> A2 = [1:4; -1:2:5]    % using colon notation
   A2 = 1 2 3 4
       -1 1 3 5
```



2-D Array Concatenation (Matrices)

- **2-D array concatenation**

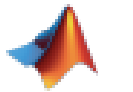
```
>> x = [4; -1]; y = [-1 3];
>> X = [x y']  % concatenation
    X =  4 -1
         -1 3

>> T = [-1 3 4; 4 5 6]; t = 1:3;
>> T = [T; t]    % concatenation
    T =  -1 3 4
          4 5 6
          1 2 3

>> G = [1 5; 4 5; 0 2];
>> T2 = [T G]   % concatenation
    T2 =  -1 3 4 1 5
           4 5 6 4 5
           1 2 3 0 2
```



Array Arithmetic Operations

V: vector, RV: Row Vector, CV: Column Vector, M: Matrix, S: Scalar

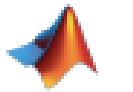
+(-)	V+V	V+S= S+V	M+M	M+S= S+M		
(.*) element by element	V.*V	V*S= S*V= V.*S= S.*V		M.*S=S.* M=M*S= S*M		
(*)	CV*RV RV*CV		M*M (n*m ** m*k = n*k)		RV*M (1*n ** n*m= 1*m)	
/& ./(element by element)	V./V	V./S=V/S S./V	M/M (n*m/ k*m= n*k)	M/S=M./S S./M		M/V(n*m/ 1*m =n*1)



Array Arithmetic Operations

V: vector, RV: Row Vector, CV: Column Vector, M: Matrix, S: Scalar

.\&		V\S=V.\S S\V=S.\V	M\M (n*m \n*k= m*k)		V\M (n*1 \n*K = 1*k)
.^ (element-by-element)	V.^V	V.^S	M.^M		
^				M^S=M*M*M…S times	



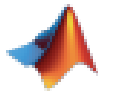
Adding vector to vector

- Two vectors must have the same type
- Two vectors must be of the same length
- The result will be element- by- element addition

```
>> u =[1 2 3];  v=[4, 5, 6];
>> v+u
    ans =  5     7     9
```

Another example:

```
>> u =[1 2 3];  v=[4, 5, 6, 5];
>> v+u
    ??? Error using ==> plus
    Matrix dimensions must agree.
```

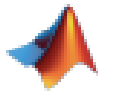


Adding row vector to column vector

```
>> u =[1 2 3]; v=[4; 5; 6];
>> v+u
??? Error using ==> plus
Matrix dimensions must agree.
```

1. Transpose one of them
2. Add it to the second

```
>> u =[1 2 3];  v=[4; 5; 6];
>> v+u‘
     ans =    5
              7
              9
>> v'+u
      ans =    5    7    9
```



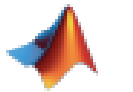
Vector element by element product(.*)

- The two vectors must be of the same length
- The two vectors must be of the same type

```
>> u=[1 2 3]; f=[4, 5, 7];
>> u .* f
    ans = 4    10    21
```

- Another Example:

```
>> v=[6, 4, 5, 7]; u=[1 2 3];
>> v .* u
??? Error using ==> times
Matrix dimensions must agree.
```

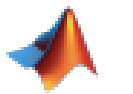


Row Vector, Column vector element by element product (.*)

```
>> u =[1 2 3]; v=[4; 5; 6];
>> u.*v
??? Error using ==> times
Matrix dimensions must agree.
```

1. Transpose one of them
2. Multiply it to the second

```
>> u =[1 2 3];  v=[4; 5; 6];
>> u'.*v
ans =     4
         10
         18
>> u.*v'
ans =     4    10    18
```



Vector inner product using (*) operator

Constraints:

- **Vectors much be of the same length.**
- **one of the two vectors is row, while the other is column vector**

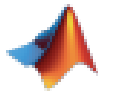
Two row vectors

```
>> u =[1 2 3];  m=[1 2 3];
>> u*m
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

```
>> m=[1 2 3]; u =[1 2 3];
>> u' * m     % transpose one of them
                using (‘) operator
ans =
    1    2    3
    2    4    6
    3    6    9

>> u * m'
ans =    14
```

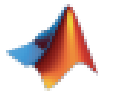
Row & column vectors

```
>> u =[1 2 3];  v=[4; 5; 6];
>> u*v
ans =     32

>> v * u
ans =     4    8    12
          5    10   15
          6    12   18
```



Multiply 2 Vector (.*) operator (element-by-element)

```
>> u = [-1; 3; 5];   v = [-1; 2; 7];
>> u .* v          % this is an element-by-element multiplication
    ans=     1
             6
             35
>> sum(u.*v)   % this is an another way to compute the inner product
    ans = 42
 >> z = [4 3 1];      % z is a row vector
>> sum(u'.*z)       % this is the inner product
    ans = 10
```



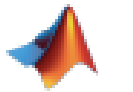
Divide two vectors in Matlab

- In Matlab, the operator **./** is defined to perform an element-by-element division.
- It is, therefore, defined for vectors of the same size and type.

```
>> x = 2:2:10;  y = 6:10;
>> x./y
   ans =
   0.3333 0.5714 0.7500 0.8889 1.0000
```

```
>> x = 2:2:10; z = -1:3;
>> x./z      % division 4/0, resulting in Inf
Warning: Divide by zero.
ans =
-2.0000   Inf   6.0000   4.0000   3.3333
```

```
>> z = -1:3
>> z./z          % division 0/0, resulting in NaN
Warning: Divide by zero.
ans =
1   NaN  1   1   1
```



Scalar-Vector operations

- **Adding scalar to vector:**

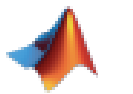
```
>> u =[1 2 3];
>> 2+u        % is equivalent to 'u+2'
   ans =    3    4    5
```

- **Scalar by vector**

```
>> u =[1 2 3];
>> 2.*u       % is equivalent to 'u.*2'
    ans =   2    4    6
>> 2*u        % is equivalent to 'u*2'
    ans =   2    4    6
```

- **Divide a scalar by a vector using (./) operator**

```
>> x=1:5; 2/x                % this is not possible
     ??? Error using ==> /
     Matrix dimensions must agree.
>> 2./x
   ans = 2.0000 1.0000 0.6667 0.5000 0.4000
```



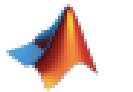
Arithmetic operators precedence

- **Precedence (priority)** form **highest to lowest** is shown below including the **arithmetic, element-wise arithmetic and transpose** operators, within each precedence level, **operators have equal precedence** are **evaluated from left to right**.

()
^ , .^ , Transpose (')
Unary plus (+) , Unary minus (-)
*** , / , \ , .* , ./ , .**
Addition (+) , Subtraction (-)

Example:

```
>> u=[1 2 3]; v=[4; 5; 6];
>> u .* v'+2
    ans=??
```

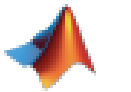


Quiz 1

Q1: What are the outputs of the following expressions:

```
>> v = [6  1  2] , w = [5  3  2] , v*w‘

v = [6  1  2] , w = [5  3  2] , v.*w

v = [0  9  0] ; w = [2 ; 1 ; 5]; v.* w' > 2
```

Q2: Use Matlab to create a raw vector with size 1*100 called T such that T contains only even numbers starting from 350 to 550



Matrix Functions

Transpose operation (‘)

•Transposing interchanges rows with the corresponding columns

```
>> A2 = [1:4; -1:2:5];
>> A2’      % transpose of A2
   ans =  1 -1
          2 1
          3 3
          4 5
>> A2’’
    ans=??
```

Matrix Random

•**rand**: creates a matrix of uniformly distributed random numbers

```
>> D = rand(2,3) ;
```

•**randn:** creates a matrix of normal random numbers

```
>> D = randn(2) ;
```

Matrix size

•**Size**: Returns the matrix size (dimensions)

```
>> A2 = [1:4; -1:2:5];
>>size(A2)
ans = 2   4
```

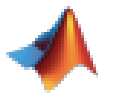
Matrix rank

•**Rank**: Returns the matrix # of rows

```
>> a=[10 0 25 4 70; 1 3 3 4 2]
>> e= rank(a)
    e =  2
```

linspace

```
>> v = linspace(1,2,4)
   v = 1.0000   1.3333   1.6667   2.0000
```



Matrices functions (Cont.)

Matrix find

Find: returns indices and values of nonzero elements in an array

```
>> v = [6 9 0 5 0 12];
>>find(v)  % or find(v ~= 0)
   ans =    1  2  4   6

>> find(v > 8)
    ans =   2   6

>> find(v >= 5, 2)     % or find(v>=5,2,'first')
    ans =   1     2

>> find(v >= 5, 2, 'last')
    ans =   4             6
```

Matrix/vector # of ~0 elements

Nnz: returns the #of nonzero elements.

```
>> a = [1 0 3 ; 0 5 0];
>> nnz(a)
    ans =   3
```

Matrix find (Cont.)

```
>> a = [1,0,3 ; 0,5,0];
>> [r,c,y] = find(a)   % or [r,c,y] = find(a ~= 0)
     r =  1
          2
          1
     c =  1
          2
          3
     y =  1
          5
          3
```

Matrix/vectors ~0 elements

•**nonzeros**: returns a full column vector of the nonzero elements ordered by columns.

```
>> a = [1,0,3 ; 0,5,0]; nonzeros(a)
    ans =   1
            5
            3
```

Matrix functions (Cont.)

Flip matrix

•**Fliplr:** reverse matrix left-right around its vertical access

```
>> a=[1 2 3 4 5; 6 7 8 9 10];
>>fliplr(a)
    ans =  5   4   3   2   1
           10  9   8   7   6
```

•**Flipud:** reverse matrix up-down around its horizontal access

```
>>flipud(a)
   ans=    6   7   8   9   10
           1   2   3   4   5
```

Matrix inverse

```
>> a=[1 2; 3 4];
>> inv(a)
ans =   -2.0000   1.0000
         1.5000  -0.5000
```

Matrix Sum

```
>> A=[1 2 3; 4 5 6];
>>sum(A)    %equivalent to sum(A, 1)
   ans= 5 7 9
>> sum(A, 2)
    ans= 6
         15
```

Matrix magic

•**Magic:** creates a square matrix , such that the summation of its columns, rows and diagonal are all equal.

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

Matrix length

Length: returns the length of the largest dimension

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

Quiz 2

Q1: Write the instructions that do the following:

- Create a 2*3 matrix M of uniformly distributed random numbers in the open interval (0,1).
- Create a magic square matrix M of size 4*4.
- Create a 3*2 matrix M of uniformly distributed random numbers in the interval (0,10).

  Note: To generate N random numbers in the interval [a,b], the following formula is used

  r=a + (b-a).*rand(N)
- Get the number of rows and columns of a matrix, M.
- Get the square root of each element of a matrix, M.
- Get the number of the nonzero elements of a matrix M.
- Get the rows and columns indices of the elements of matrix M that are between 6 and 12.

Quiz 2 (Cont.)

Q2: Use Matlab to create a new **4*4 matrix** called T such that T 's value is the same **first** and **third rows** from **R** while the **second** and **fourth rows** are all **ones**
Given R is a 5*4 matrix and defined as: **R=[1:4; 5:8;9:12; 13:16; 17:20]**

Solution

```
>>R=[1:4; 5:8;9:12; 13:16; 17:20];
>> t([1, 3],:)=R([1, 3],:)
t = 1    2    3    4
    0    0    0    0
    9   10   11   12
```

```
>> t([2, 4],:)=ones(2, 4)
t = 1    2    3    4
    1    1    1    1
    9   10   11   12
    1    1    1    1
```

Another Solution

```
>>R=[1:4; 5:8;9:12; 13:16; 17:20];
>> t=[R(1,:); ones(1, 4); R(3, :); ones(1, 4)]
t = 1    2    3    4
    1    1    1    1
    9   10   11   12
    1    1    1    1
```

Special Matrices

Identity Matrix

•**eye**: creates an identity matrix

>> I = eye(3);

I = 1 0 0
0 1 0
0 0 1

Identity diagonal

•**Diag**: Returns the matrix diagonal

>> A = [1 2 3; 4 5 6; 7 8 9]; diag(A)

ans = 1
5
9

•**Diag**: creates a diagonal matrix with a vector on the diagonal

>> r = [1 3 -2]; R = diag(r)

R = 1 0 0
0 3 0
0 0 -2

Matrix of Ones

•**ones**: creates a matrix of all ones

>> B = ones(3)

B = 1 1 1
1 1 1
1 1 1

>> T =[-1, 3, 4; 4 5 6 ; 1 2 3]; G = [1 5; 4 5; 0 2];

>> T3 = [T; G ones(3,1)]

T3 = ??

>> T3 = [T; G'];

>> [G' diag(5:6); ones(3,2) T]

ans=??

Matrix of Zeros

•**zeros**: creates a matrix of all zeros

>> C = zeros (size(G'));

>> C=zeros(3, 4);

Quiz 3

Q1: What are the outputs of the following expressions:

```
>>t = [1 ; 2 ;  3 ;  4 ; 5]
q = [1:2:10]'
s = [t  , q]
z = [t' ; q']
a = 2 : 3 : 8
w = ones(3)
r = zeros([3 2])
r = zeros(3, 2)
x = eye(3,5)
67 / 0
v = [6  1  2] , w = [5  3  2] , max(v,w)
```

Q2: Let **r=randn(3, 5),** Use Matlab to create a new **3*5 matrix** called **T** such that **T(i, j) =**

$$T(i,j) = \begin{cases} r(i,j) & ; r(i,j)>0 \\ 0 & ; \text{otherwise} \end{cases}$$

Accessing matrix element

```
>> A = [1:3; 4:6; 7:9];
>> A(1,2); A(2,3); A(3,1);A(4);
>> A(3,:);     % extract the 3rd row of A
>> A(:,2);     % extract the 2nd column of A
>> A(1:2,:) ; % extract the rows 1st and 2nd of A
>> A(5,4) = 5;    % assign 5 to the position (5,2); the uninitialized elements become
    zeros
>> A([2,4],1:2); % extract a part of A (the 1st & 2nd columns from the 1st & 4th rows)
>> A(2,3) = A(2,3) + 2*A(1,1);   % change the value of A(2,3)
>> A(4,:) = [2, 1, 2]; % assign vector [2, 1, 2] to the 4th row of A
>> A(5,[1,3]) = [4, 4];     % assign: A(5,1) = 4 and A(5,3) = 4
>> A;       % how does the matrix A look like now?


>> A(6,3) % this is not possible: A is a 5-by-4 matrix!
??? Index exceeds matrix dimensions.
```

Removing column from a matrix

The concept of an empty matrix []

- Removing a number of rows from a certain matrix
- Removing a number of column from a matrix

```
>> C = [1 2 3 4; 5 6 7 8; 1 1 1 1];
>> D = C;
>>D(:,2) = [];   % remove the second column using empty matrix []
>> C ([1,3],:) = []  % remove the 1st and the 3rd rows
>> C(1,2) = []
    Subscripted assignment dimension mismatch.
```

Operations on matrices (+, -, ^)

Adding/Subtraction matrices

- They must be of the same size

```
>> B = [1 -1 3; 4 0 7]; B2 = [1 2; 5 1; 5 6];
>> B+B2
??? Error using ==> plus
Matrix dimensions must agree.

>> B' + B2
ans =   2    6
        4    1
        8   13
>> B - B2'
ans =   0   -6   -2
        3   -1    1
```

Multiplying scalar to matrix

```
>> B *2    %equivalent to 2*B, 2.*B, B.*2
    ans = 2   -2    6
          8    0   14
```

Adding/Subtraction scalar to matrix

```
>> B = [1 -1 3; 4 0 7];
>> B + 2    %equivalent to 2+B
    ans =   3    1    5
            6    2    9
>> 2 - B
    ans =   1    3   -1
           -2    2   -5
```

Power scalar to matrix (^)

```
>> C =[3 1; 1 -3];
>> C^3      % this is equivalent to C*C*C
    ans = 30   10
          10  -30
>>D=[6 8; 4 6];
>>C^D
Error using  ^
Inputs must be a scalar and a square matrix.
```

Operations on matrices (./, .*, .^)

Element-by element division
Scalar ./ matrix

```
>> B = [1 -1 3; 4 0 7];
>> ans = B./4
   ans =  0.2500   -0.2500    0.7500
          1.0000         0    1.7500
>> 4./B     %equivalent to  4.*ones(size(B)) ./ B
    ans = 4.0000   -4.0000    1.3333
          1.0000       Inf    0.5714

>>B/4        %equivalent to 4\B
    ans=   0.2500   -0.2500    0.7500
           1.0000         0    1.7500
>> 4/B        %equivalent to B\4
??? Error using ==> /
Matrix dimensions must agree.
```

Element-by element multiplication
matrix .* matrix

```
>> C = [1 -1 4; 7 0 -1];
>> B = [1 -1 3; 4 0 7];
>> B .* C   % multiply element-by-element
   ans = 1     1    12
         28    0    -7
```

Matrix-(scalar/Matrix) power (.^)

```
>> B = [1 -1 3; 4 0 7];
>> B.^3     % element-by-element power
    ans =    1   -1   27
             64   -0  343

>> b = [1,3,2;2,3,2]; a = [1 2 3; 4 5 6];
>> a.^b
    ans= 1     8        9
         16   125    36
```

Matrix-vector& Matrix-matrix product

Row vector * Matrix

Rule:

Vector: R(1 × n) * Matrix :M(n × m) = Vector: Y(1 × m)

```
>> b = [1 3 -2];  B = [1 -1 3; 4 0 7];
>> b * B
??? Error using ==> *
Inner matrix dimensions must agree.
>> b * B'
    ans = -8        -10
```

Matrix * Matrix

Rule:

Matrix: M1(n × m)* Matrix:M2(m × k) = Matrix: M3(n × k)

```
>> B = [1 -1 3; 4 0 7];
>> B' *ones(2,1)
    ans = 5
          -1
           10
>> C = [3 1; 1 -3];
>> C * B
    ans = 7      -3     16
          -11    -1     -18
```

Matrix-vector& Matrix-matrix division

Matrix \Matrix

```
>> a = [1 2 3; 4 5 6]; i=[9 8 7 6; 9 6 0 56] ;
 >> i\a          %(2*4) \(2*3) =(4*3)
    ans =    0.0711    0.1822    0.2933
                  0         0         0
                  0         0         0
             0.0600    0.0600    0.0600
```

Matrix / Matrix

```
>> a = [1 2 3; 4 5 6];   b = [1,3,2;2,3,2];
 >> a/b          %(2*3) /(2*3) =(2*2)
    ans =    0.8462    0.0769
             0.1538    1.9231
```

Vector\Matrix

```
>> a = [1 2 3; 4 5 6]; >> k=[9 8];
>> k'\a     %(2*1) \ (2*3)  =(1*3)
    ans =    0.2828    0.4000    0.5172
```

Vector/ Matrix

```
>> a = [1 2 3; 4 5 6];  v=[1 2 3]
>> v/a      %(1*3) /(2*3) =(1*2)
    ans =    1.0000    0.0000
```

Arithmetic operators precedence

- **Precedence (priority)** form **highest to lowest** is shown below. **Operators have equal precedence** are **evaluated from left to right**.

  ()

  ^ , .^ , Transpose (')

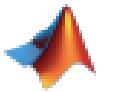
  Unary plus (+) , Unary minus (-)

  *** , / , \ , .* , ./ , .**

  Addition (+) , Subtraction (-)

Example:

```
>> u=[1 2 3; 4 5 6]; v=[1 2; 3 4; 5 6];
>> u .* v'+2                          % transpose then .* then addition
    ans= 3     8     17
         10    22    38

 >> ones(3,4)./4 * diag(1:4)          %division then multiplication
    ans = 0.2500   0.5000   0.7500   1.0000
          0.2500   0.5000   0.7500   1.0000
          0.2500   0.5000   0.7500   1.0000
```



Quiz 4

- Let x is a row vector, compute f(x) as follows:

$$f(x) = \begin{cases} 1 + \cos(2\pi x), & |x| \leq \frac{1}{2} \\ 0, & |x| > \frac{1}{2}. \end{cases}$$

Given x=[1, 3, 5, -.25, -.5]

Note:

abs(x) is a function that calculate the absolute value of x

Quiz 4: Solution

```
>>x=[1, 3, 5, -.25, -.5]
```

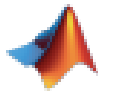
$$f(x) = \begin{cases} 1 + \cos(2\pi x), & |x| \le \frac{1}{2} \\ 0, & |x| > \frac{1}{2}. \end{cases}$$

```
>>r=find(abs(x)<=0.5)
  r=  4   5
>>f(r)=1+cos(2*pi*x(r))      % f([4,5])=1+cos(2*pi*x([4, 5]))
  f=  0   0   0   1   0
```

Comparison (Relational) Operators

- **Comparison Operators: <, >, <=, >=, ==, ~=(Not-Equal)**

```
>> v = [6 9 0 5 0 12];
>> v == 0
    ans =     0    0    1    0    1    0
>> v(v == 0) = 10    %equivalent to v(ans)=10
    v =       6    9   10    5   10   12
>> a = [1 0 3 ; 0 5 0];
>> a ~= 0
    ans =     1    0    1
              0    1    0
```

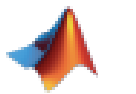
Quiz 5

Let **r=randn(3, 5),** Use Matlab to create a new 3*5 matrix called T such that

$$T(i, j) = \begin{cases} r(i, j) & ; r(i, j)>0 \\ 0 & ; \text{otherwise} \end{cases}$$

Solution

```
>> r=randn(3, 5)
r =
   -1.7947    0.1001   -0.6003    1.7119   -0.8396
    0.8404   -0.5445    0.4900   -0.1941    1.3546
   -0.8880    0.3035    0.7394   -2.1384   -1.0722

>> t=r;
>> t(r<0)=0
t =
         0    0.1001         0    1.7119         0
    0.8404         0    0.4900         0    1.3546
         0    0.3035    0.7394         0         0
```



Operators Precedence

- **Precedence (priority)** form **highest to lowest i**s shown below. **Operators have equal precedence** are **evaluated from left to right**.

()

^ , .^ , Transpose (')

Unary plus (+) , Unary minus (-) , NOT(~)

*** , / , \ , .* , ./ , .**

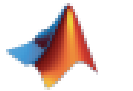
Addition (+) , Subtraction (-)

< , > , <= , >= , == , ~=

&

|

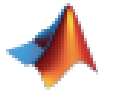
```
>> A = [0 9 0]; B = [2 ; 1 ; 5]; C = ~A .* B' > 2
C =
     0     0     1
```

Element-Wise Logical Operators

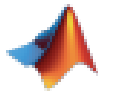
- **NOT(~), AND (&), OR (|) for more complex conditions**

```
>> v = [6 9 0 5 0 12];
 >> ~v
ans =        0    0    1    0    1    0
>> v  > 5 & v < 8
v =          1    0    0    0    0    0
>> v  == 0 | v < 9
ans =        1    0    1    1    1    0
>> ~v | v >= 9 & v <= 12
ans =        0    1    1    0    1    1
```

Element-Wise Logical Operators (2)

```
>> a = [1 0 3 ; 0 5 0];
>> b = [1 3 2 ; 2 3 2];
>> a & b
ans =       1    0    1
            0    1    0
>> a == 0 & b ~=2
ans =   0    1    0
        0    0    0
>> a( a | ~b ) = 9
a =   9    0    9
      0    9    0
```

Multi-dimensional array Creation

- **Creating a multi-dimensional array:**

  1- create a 2-D array

```
>> x = [7 0 5; 6 1 3; 4 0 1] ;
```

  2- extend it to be multidimensional (add a third dimension to it)

```
>> x(:, :, 2)= [ 9 11 5; 3 6 8; 7 0 1];
```

- **Creating special 3-D arrays using ones(), zeros(), rand(), randn()**

```
>> x = ones(2, 5, 3)
   ans=??
```

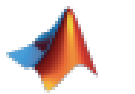
- **Creating a multi-dimensional array from 2-D arrays**

```
>> a = [1 2 ; 4 5]; b=[7 8 ; 10 11];
>> c= cat(3, a, b, [13 14; 16 17])
```

c(:, :, 1)= 1 2
4 5

c(:, :, 2)= 7 8
10 11

c(:, :, 3)= 13 14
16 17



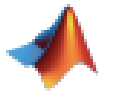
Multi-dimensional array
Accessing its elements

- **Accessing the multidimensional array elements**

```
>> x (:,:,1)
   ans=  7    0    5
         6    1    3
         4    0    1
>>x([1, 3],1:3,2)   %from the 2nd view, the 1st, 2nd, 3th clumns in the 1st and 3rd rows
   ans=??
```

- **Multidimensional arrays functions: 'sum'**

```
>> sum(x)    %sum(x, 1)
 ans(:,:,1) =   17    1    9
>>sum(x, 2)
ans(:,:,1) =   12
               10          ans(:,:,2) =    19    17    14
               5
>> sum(x, 3)              ans(:,:,2) =   25
ans =    16    11    10                   17
          9     7    11                   8
          11    0     2
```

Multi-dimensional array Concatenation

- **Concatenating a multidimensional arrays**

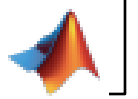
```
>> x = [7 0 5; 6 1 3; 4 0 1];
   x(:,:,1) =   7   0   5
                6   1   3
                4   0   1
```

```
x(:,:,2)=[9 11 5; 3 6 8; 7 0 1]
x(:,:,2) =   9   11   5
             3   6    8
             7   0    1
```

```
>> y=[99 11 5; 73 6 68; 17 1 10; 6 7 9];
   y(:,:,1) =   99   11   5
                73   6    68
                17   1    10
                6    7    9
```

```
y(:, :,2)=[9 1 5; 3 6 8; 70 1 0; 16 17 19]
y(:,:,2) =   9   1   5
             3   6   8
             70  1   0
             16  17  19
```

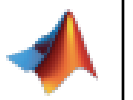
```
>> s=[ [x; ones(1, 3, 2) ], y]
   s(:,:,1) =   7   0   5   99   11   5
                6   1   3   73   6    68
                4   0   1   17   1    10
                1   1   1   6    7    9

   s(:,:,2) =   9   11   5   9   1   5
                3   6    8   3   6   8
                7   0    1   70  1   0
                1   1    1   16  17  19
```



Quiz 6

Create the following matrix, g, and write the outputs of the following instructions in order:

```
g =
                16      2       3      13
                 5      11      10      8
                 9      7       6       12
                 4      14      15      1
>>g(3)
>>max(g)
min(g,[],2)
>>sum(g)
>>diag(g)
>>prod(g,2)
size(g)
>>g(:, :, 2)=[16 , 6, 0  8;9, 7  56,5;    2  88, 9,  13; 7 8 6 43]
>>Size(g)
>>S=[g  eye(4, 4, 2)]
```



Multi-dimensional array programming

- the following code produces **24 different magic square matrices by interchanging columns** and store them in a multidimensional array that is 3D

```
>> A = magic(4);
>> p = perms(1:4);
>> M = zeros(4,4,24);
>> for k = 1:24
M(:,:,k) = A(:,p(k,:));
end
>> for i=1:24
x(i)=sum(diag(M(:, :, 1)));
end
>> x
x =
     34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
     34  34  34  34  34  34  34
```