2.5 adding element to the matrix

Example 1:

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

>> x(:,4)=[8 8 8]
x =
   1   2   3   8
   4   5   6   8
   7   8   9   8
```

>> x(4,:)=[8 8 8 8]

x =

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

Example2 :

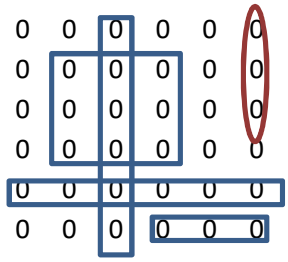
```
>> x=[2 3 5; 4 5 1];
>> x=[x;3 7 6]
x =
   2   3   5
   4   5   1
   3   7   6
>> x=[8 6 9 ;x]
x =
   8   6   9
   2   3   5
   4   5   1
   3   7   6
>> x(:,4)=[2;3;7;6]
x =
   8   6   9   2
   2   3   5   3
   4   5   1   7
   3   7   6   6
```

```
>> x(:,7)=[7;8;2;5]
x =
   8   6   9   2   0   0   7
   2   3   5   3   0   0   8
   4   5   1   7   0   0   2
   3   7   6   6   0   0   5
```

2.6 Replacing vector elements

Example1:

```
>> f=zeros(6,6)
f =
   0   0   0   0   0   0
   0   0   0   0   0   0
   0   0   0   0   0   0
   0   0   0   0   0   0
   0   0   0   0   0   0
   0   0   0   0   0   0
```



```
>> f(1:3,6)=[1 1 1]
f =
   0   0   0   0   0   1
   0   0   0   0   0   1
   0   0   0   0   0   1
   0   0   0   0   0   0
   0   0   0   0   0   0
   0   0   0   0   0   0

>> f(5,:)=[1 1 1 1 1 1 ]
f =
   0   0   0   0   0   1
   0   0   0   0   0   1
   0   0   0   0   0   1
   0   0   0   0   0   0
   1   1   1   1   1   1
   0   0   0   0   0   0
```

```
>> f(6,4:6)=[1 1 1]
f =
   0   0   0   0   0   1
   0   0   0   0   0   1
   0   0   0   0   0   1
   0   0   0   0   0   0
   1   1   1   1   1   1
   0   0   0   1   1   1


>> f(:,3)=[1 1 1 1 1 1]
f =
   0   0   1   0   0   1
   0   0   1   0   0   1
   0   0   1   0   0   1
   0   0   1   0   0   0
   1   1   1   1   1   1
   0   0   1   1   1   1

>> f(2:4,2:4)=[1 1 1 ;1 1 1;1 1 1]
f =
   0   0   1   0   0   1
   0   1   1   1   0   1
   0   1   1   1   0   1
   0   1   1   1   0   0
   1   1   1   1   1   1
   0   0   1   1   1   1
```

20/11/2016

Example:

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

To insert new row to the end of matrix:

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

Example:

```
>> x=[2 3 5; 4 5 1];
>> x=[x;3 7 6]
x =
   2   3   5
   4   5   1
   3   7   6

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

Transposing a matrix

The transpose operation is denoted by an apostrophe or a single quote ('). It flips a matrix about its main diagonal and it turns a row vector into a column vector. Thus,

example 1:

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

By using linear algebra notation, the transpose of m x n real matrix a is the n x m matrix that results from interchanging the rows and columns of A.

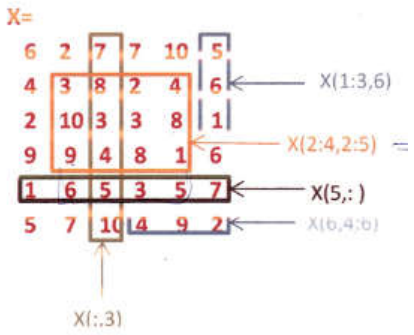
Example :

Let x be a matrix of size 6x6 as shown below then

X(5,:) denotes the 5^{th} row .

X(:,3) denotes the 3^{th} column.

X(1:3,6) first three elements of the column 6.
X(6,4:6) denotes the 4th to the 6th elements of row 6.
X(2:4,2:5) denotes the sub matrix at the intersection of 2th to 4th row and 2th to 5th column of the matrix x.



Dimension

To determine the dimensions of a matrix or vector, use the command size

Example:

```
>> a=[2 1 0;5 1 4]
a =
   2   1   0
   5   1   4

>> size(a)
ans =
   2   3
```

If we want to display number of rows only

```
>> size(a,1)
ans =
   2
```

If we want to display number of column only

```
>> size(a,2)
ans =
   3
```

1.2 Array arithmetic operations

Array arithmetic operations or array operations for short, are done element-by-element. The period character, . , distinguishes the array operations from the matrix operations. However, since the matrix and array operations are the same for addition (+) and subtraction (-), the character pairs (.+) and (.-) are not used. The list of array operators is shown below in Table. If A and B are two matrices of the same size with elements A = $[a_{ij}]$ and B = $[b_{ij}]$, then the command

.*	Element-by-element multiplication
./	Element-by-element division
.^	Element-by-element exponentiation

Table :Array operator

>> C = a.*b

produces another matrix C of the same size with elements $c_{ij} = a_{ij}\, b_{ij}$, using the same 3 x 3 matrices.(a and b must be same size)

example:

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

>> b=[4 1 3 ; 3 2 8 ;2 8 1]
b =
   4   1   3
   3   2   8
   2   8   1

>> c = a.*b
c =
   4   4  15
   3   6  72
   8  64   1
```

Example:

```
>> a=[1 4 1 ;2 6 1 ;4 6 7]
a =
   1    4    1
   2    6    1
   4    6    7

>> a.^2
ans =
   1   16    1
   4   36    1
  16   36   49
```

Example :

```
>> x=[5 6 2;4 6 9]
x =
   5    6    2
   4    6    9

>> x./2
ans =
   2.5000    3.0000    1.0000
   2.0000    3.0000    4.5000
```

Array operation:

Product of tow array a and b by

p=a*b

We can see clearly that the tow resulted matrixes p and c is different

If

$$a=\begin{matrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{matrix}$$

and

$$b = \begin{matrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,1} & b_{3,2} & b_{3,3} \end{matrix}$$

then

p=

$$\begin{matrix} a_{1,1} \times b_{1,1} + a_{1,2} \times b_{2,1} + a_{3,1} \times b_{1,3} & a_{1,1} \times b_{1,2} + a_{1,2} \times b_{2,2} + a_{1,3} \times b_{3,2} & a_{1,1} \times b_{1,3} + a_{1,2} \times b_{3,2} + a_{1,3} \times b_{3,3} \\ a_{2,1} \times b_{1,1} + a_{2,2} \times b_{2,1} + a_{2,3} \times b_{3,1} & a_{2,1} \times b_{1,2} + a_{2,2} \times b_{2,2} + a_{2,3} \times b_{3,2} & a_{2,1} \times b_{1,3} + a_{2,2} \times b_{2,3} + a_{2,3} \times b_{3,3} \\ a_{3,1} \times b_{1,1} + a_{3,2} \times b_{2,1} + a_{3,3} \times b_{3,1} & a_{3,1} \times b_{1,2} + a_{3,2} \times b_{2,2} + a_{3,3} \times b_{3,2} & a_{3,1} \times b_{1,3} + a_{3,2} \times b_{2,3} + a_{3,3} \times b_{33} \end{matrix}$$

if first matrix(e) has two rows and three column and matrix f has three rows and three columns , the product e*f will have two rows and three column

example :

```
>>  e=[1 2 3; 1 0 2; 4 2 1]
e =
    1    2    3
    1    0    2
    4    2    1

>> f=[1 3 2 5 ; 1 1 1 0; 4 1 3 2]
f =
    1    3    2    5
    1    1    1    0
    4    1    3    2

>> n=e*f
n =
   15    8   13   11
    9    5    8    9
   10   15   13   22
```

Note : We must know that the inner number are both 3 so there are no error appear

1 x 3 , 3 x 4

then n= 3x4

example:

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

>> h^2
ans =
    7   10
   15   22
```

Note: h^2 will be (h*h) without dot (.).

X=a/b will be equal to ($a*b^{-1}$)

Array merging:

By using to array we can create one array (merging tow arrays)

If a= 3X3

b=3X3

then

c=[a,b] is array with size 3X6

c=[a;b] is array with size 6X3

example :

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

>> b=[4 7 8;1 1 2 ;6 8 3]
b =
    4    7    8
    1    1    2
    6    8    3
```

```
>> c=[a,b]
c =
   1   2   6   4   7   8
   5   1   8   1   1   2
   7   3   2   6   8   3

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

Homework: if you have matrix a=[1 4;2 7;1 2] add column to matrix a by using merging

Array functions

- **length** it is used with vectors to determine the length of vectors (number of vector elements)

example:

```
>> h=[7 5 9 1]
h =
   7   5   9   1
>> length(h)
ans =
   4
```

Example :

```
>> x=[4;2;6;1;2;4]
x =
   4
   2
   6
   1
   2
   4
```

```
>> length(x)
ans =
   6
```

- **Sum**: the sum of array or vector elements

  Sum(vector) the result will be one number.
  Sum(array) the result will be vector.

  Example :

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

>> sum(a)
ans =
   6

>> b= [1 4 0;0 5 10;3 1 1]
b =
   1    4    0
   0    5   10
   3    1    1

>> sum(b)
ans =
   4   10   11
```

  Note: where the function find the summation of each column .

  If we want to find the summation of each row we must use sum function as below

  Sum (array name, 2) were number 2 refer to the rows

Example: if we use the array in the previous example

```
>> b= [1 4 0;0 5 10;3 1 1]
b =
    1    4    0
    0    5   10
    3    1    1

>> sum (b,2)
ans =
    5
   15
    5

>> sum(b,1)
ans =
    4   10   11
```

Where number 1 refer to the column .

27-11-2016

Example :

```
>> b=[1 4 0;0 5 10; 3 1 1]
b =
    1    4    0
    0    5   10
    3    1    1
```

Home worke :create the matrix and find the average of matrix elements.

- **dot** it find the dot product of vector elements

example:

```
>> a=[1 4 2]
a =
    1    4    2
>> b=[5 1 0]
b =
    5    1    0
```

```
>> dot(a,b)
ans =
   9
```

Example:

```
>> v=[1;4;5]
v =
   1
   4
   5

>> h=[4 1 0]
h =
   4    1    0

>> dot(v,h)
ans =
   8
```

- prod

  It find the product of array elements

  Example :

```
>> a=[2 1 3 ;6 3 4 ;4 6 1]
a =
   2    1    3
   6    3    4
   4    6    1

>> prod(a)
ans =
  48   18   12

>> prod(a,2)
ans =
   6
  72
```

24

```
>> prod(a,1)
ans =
  48   18   12

>> prod(prod(a))
ans =
    10368

>> prod(a(:))
ans =
    10368
```

Homework: in the previous homework find the average by using **prod** function in the (المقام)

- max
  to extract largest element in array

example:

```
>> a=[ 0 8 4 3 0 2]
a =
   0    8    4    3    0    2
>> max(a)
ans =
   8
```

Example :

```
>> f=[2 0 4;5 1 4; 7 0 1]
f =
   2    0    4
   5    1    4
   7    0    1

>> max(f)
ans =
   7    1    4

>> max(max(f))
```

```
ans =
   7

>> max(f(:))
ans =
        7
```

- **min**
  to extract minimal element in array

  example :
```
>> a=[ 0 8 4 3 0 2]
a =
   0    8    4    3    0    2

>> min(a)
ans =
   0
```

  Example :
```
>> f=[2 0 4;5 1 4; 7 0 1]
f =
   2    0    4
   5    1    4
   7    0    1

>> min(f)
ans =
   2    0    1

>> min(min(f))
ans =
   0
```
  Or
```
>> min(f(:))
ans =
   0
```

- **sort**
  sort the elements of vector and array ascending

  example:
  >> a=[0 8 4 3 0 2]
  a =
  0 8 4 3 0 2

  >> sort(a)
  ans =
  0 0 2 3 4 8

  Example :
  >> c=[2 5 1; 6 1 9;5 2 3]

  c =

  2 5 1
  6 1 9
  5 2 3

  >> sort(c)
  ans =
  2 1 1
  5 2 3
  6 5 9

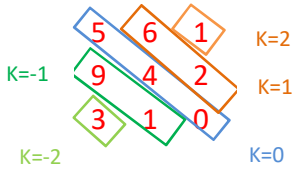
  If we want to sort elements descending

  Example :
  >> c=[2 5 1; 6 1 9;5 2 3]
  c =
  2 5 1
  6 1 9
  5 2 3

>> sort(c,'descend')
ans =
6 5 9
5 2 3
2 1 1

- diag
  diag is abbreviated of diagonal

  example :
  a =



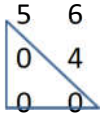
>> diag(a)
ans =
5
4
0

>> diag(a,1)
ans =
6
2

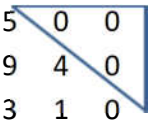
>> diag(a,-2)
ans =
3

- triu
  example:
  >> a=[5 6 1;9 4 2;3 1 0]
  a =
  5 6 1

9 4 2
3 1 0

>> triu(a)
ans =
5 6 1

0 4 2
0 0 0

- **tril**
  example :
  >> a=[5 6 1;9 4 2;3 1 0]
  a =
  5 6 1
  9 4 2
  3 1 0

  >> tril(a)
  ans =
  
  5 0 0
  9 4 0
  3 1 0

  >> triu(tril(a))
  ans =
  5 0 0
  0 4 0
  0 0 0

- **diff**

example :
>> a=[5 6 1;9 4 2;3 1 0]
a =
5 6 1
9 4 2
3 1 0

```
>> diff(a)
ans =
    4   -2    1
   -6   -3   -2

>> diff(diff(a))
ans =
  -10    -1    -3
```

Example :

```
>> c=[1 2 7 3; 2 5 0 6; 1 9 2 7; 2 1 5 3]
c =
    1    2    7    3
    2    5    0    6
    1    9    2    7
    2    1    5    3

>> diff(c)
ans =
    1    3   -7    3
   -1    4    2    1
    1   -8    3   -4

>> diff(diff(c))
ans =
   -2    1    9   -2
    2  -12    1   -5
```

- **Identity matrix**

  An identity matrix is a matrix with one on the main diagonal and zeros everywhere

  Example :

```
>> a= eye(3)
a =
    1    0    0
    0    1    0
    0    0    1
```

```
>> b= eye(3,2)
b =
   1   0
   0   1
   0   0
```

Use the size function to determine the correct number of rows and columns.

Example :

```
>> p=[1 2 3;4 5 6]
p =
   1   2   3
   4   5   6

>> d= eye(size(p))
d =
   1   0   0
   0   1   0
```

- **Fliplr**

  The MATLAB function filplr, flip an array or a matrix in the left-right direction.

  Example :

```
>> f=[2 3 3;1 1 2;1 7 5;3 4 6 ]
f =
   2   3   3
   1   1   2
   1   7   5
   3   4   6

>>  fliplr(f)
ans =
   3   3   2
   2   1   1
   5   7   1
   6   4   3
```

Example:

```
>> h=[3 5 1 2; 1 0 3 8; 6 7 8 1 ; 3 9 2 1]
h =
   3   5   1   2
   1   0   3   8
   6   7   8   1
   3   9   2   1

>> fliplr(h)
ans =
   2   1   5   3
   8   3   0   1
   1   8   7   6
   1   2   9   3
```

- **Flipud**

flipud is MATLAB function to perform flipping in the up-down direction.

Example :

```
>> c=[1 5 9 ;4 3 2; 0 1 4]
c =
   1   5   9
   4   3   2
   0   1   4

>> flipud(c)
ans =
   0   1   4
   4   3   2
   1   5   9
```

Example :

```
>> f=[2 3 3;1 1 2;1 7 5;3 4 6 ]
f =
   2   3   3
   1   1   2
   1   7   5
   3   4   6
```

```
>> flipud(f)
ans =
   3   4   6
   1   7   5
   1   1   2
   2   3   3
```

- rot90

The function **rot90** rotates matrix 90 degree. This is equivalent to flipping matrix in left-right direction and then transposing it. Or it is equivalent to transposing a matrix and then flipping it in up-down direction.

Example:

```
>> f=[2 3 3;1 1 2;1 7 5;3 4 6 ]
f =
   2   3   3
   1   1   2
   1   7   5
   3   4   6

>> rot90(f)
ans =
   3   2   5   6
   3   1   7   4
   2   1   1   3
```