Review of Lecture 5

- **Create a vector in MATLAB**

```
>> % Creating a row vector V
>> V = [1, 5, 8, 7, 10]
>> V = [1 5 8 7 10]
>> % Creating a column vector V
>> V = [1; 5; 8; 7; 10]
```

  - Vector with elements **sequentially arranged**

```
V = [first elmnt : last elmnt]
```

  - Vector with elements **arranged with consecutive values but with a step**

```
V = [first elmnt : step : last elmnt]
```

  - Vector with `n` values **evenly spaced between two defined boundaries (`S` and `E`)**

```
linspace (S,E,n)
```

  - Create a vector based on existing vectors

```
V = [V1 V2]
```

- **Operations on vectors**

```
V(x), V(x) = [] , V(x) = a,
V(X) = A, V = [V, e]
```

- **Important functions for vectors**

```
length (V), sum (v), prod (v),
max (v),  min(v), mean(v), sort(v)
```

- **Algebraic operations**

```
V + U,  V – U,   V .* U,  V ./ U,  V'
```

Info 3

Introduction to MATLAB

M. Bouzenita
2nd year Engineer - University of Jijel

Lecture 6

Vectors and matrices in MATLAB(2/2)

1. Matrix in MATLAB

Matrix in MATLAB is used to store same type of data. A matrix can be defined as a table of values with r rows and c columns where $r \times c$ represents the dimension of the matrix.

$$\begin{matrix} & c = 3 \\ r = 4 & \end{matrix} \begin{pmatrix} 2 & 1 & 5 \\ 8 & 0 & 3 \\ 5 & 2 & 9 \\ 3 & 0 & 7 \end{pmatrix}$$

2. Create a matrix i

For creating a matrix in MATLAB, we use square brackets **[]** to define the matrix's elements, separating rows with **semicolons** and columns with **spaces or commas**.

The number of elements in each row must be always the same.

```
% Create a 2x3 matrix
>> M = [7, 2, 3; 9, 5, 1] % or M=[7 2 3; 9 5 1]

M =

     7     2     3
     9     5     1
% Create a 3x3 matrix using enter key
>> M = [7 2 3
9 5 1
7 4 9]

M =

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

2. Create a matrix i

If we need **arranged raw** elements, we use iterators

```
>> M = [1:3; 4:6; 7:2:11]

M =

     1     2     3
     4     5     6
     7     9    11
```

2. Create a matrix i

To create a **zero matrix** (a matrix containing only zeros), we use the `zeros` function

To generate a matrix filled with **ones** in MATLAB, we employ the `ones` function.

To generate an **identity matrix**, we use `eye` function.

```
>> null_matrix = zeros(3)

null_matrix =

     0     0     0
     0     0     0
     0     0     0

>> null_matrix1 = zeros(2,3)

null_matrix1 =

     0     0     0
     0     0     0
```

2. Create a matrix i

To create a **zero matrix** (a matrix containing only zeros), we use the **zeros** function

To generate a matrix filled with **ones** in MATLAB, we employ the **ones** function.

To generate an **identity matrix**, we use **eye** function.

```
>> ones_matrix = ones(3)

ones_matrix =

     1     1     1
     1     1     1
     1     1     1

>> ones_matrix1 = ones(2,3)

ones_matrix1 =

     1     1     1
     1     1     1
```

2. Create a matrix i

To create a **zero** matrix (a matrix containing only zeros), we use the `zeros` function

To generate a matrix filled with **ones** in MATLAB, we employ the `ones` function.

To generate an **identity matrix**, we use `eye` function.

```
>> identity_matrix = eye(3)

identity_matrix =

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

2. Create a matrix i

In MATLAB, we can create a matrix with random numbers using **`rand()`** function.

`rand` returns a single uniformly distributed random number in the interval (0,1).

`rand(n)` returns an n-by-n matrix of random numbers.

`rand(v, u)` returns a v-by-u array of random numbers where v and u indicate the size of each dimension.

```
>> rand
ans =
    0.8147

>> rand(2) % 2 by 2 matrix of random numbers.
ans =
    0.9058    0.9134
    0.1270    0.6324

>> rand(2,3) % 2 by 3 matrix of random numbers.
ans =
    0.0975    0.5469    0.9649
    0.2785    0.9575    0.15761
```

2. Create a matrix i

In MATLAB, we can create a matrix with random numbers using **rand()** function.

randi(imax) returns a pseudorandom scalar integer between 1 and **imax**.

randi(imax,n) returns an n-by-m matrix of pseudorandom integers drawn from the discrete uniform distribution on the interval [**1,imax**].

randi(imax,n,m) returns an n-by-m array where **n, m** indicates the size of each dimension.

More specification of the **rand** function can be found in MATLAB help.

```
>> % random integer between 1 and 30.
>> randi(30)
ans =
    18
>> % 2 by 2 array of random integers between 1
   % and 30
>> randi(30,2)
ans =
    15    11
     1     5
>> % 2 by 3 array of random integers between
   % 1 and 30
>> randi(30,2,3)
ans =
    24    16    19
    10     5     8
```

2. Create a matrix in MATLAB

In addition to creating matrices, MATLAB have the flexibility to perform a range of operations. This includes the ability to **modify**, **add**, or **remove** elements within the matrices.

`M(i)`: This is called **linear indexing** which returns the value located at position **`'i'`** within the matrix where the **`i`** index is defined as provided below in the matrix.

$$M\begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$$

2. Create a matrix in MATLAB

In addition to creating matrices, MATLAB have the flexibility to perform a range of operations. This includes the ability to **modify**, **add**, or **remove** elements within the matrices.

`M(i,j)`: Returns the values located at position `(i,j)` within the matrix, where `i` and `j` indicate the row and the column positions respectively.

`M(i:ii,j:jj)`: Refers to a subset of matrix with `i` to `ii` rows and columns `j` to `jj`.

`M(i,:); M(:,j)`: Return the row `i` or the column `j` respectively.

2. Create a matrix i

In addition to creating matrices, MATLAB
operations. This includes the ability to **mo**
matrices.

M(i,j): Returns the values located at position (**i,j**) within the matrix, where **i** and **j** indicate the row and the column positions respectively.

M(i:ii,j:jj): Refers to a subset of matrix with **i** to **ii** rows and columns **j** to **jj**.

M(i,:); M(:,j): Return the row **i** or the column **j** respectively.

```
>> M = [7 2 3; 9 5 1; 7 4 9]

M =

     7     2     3
     9     5     1
     7     4     9

>> M(2)

ans =

     9

>> M(2,3)

ans =

     1
```

2. Create a matrix i

In addition to creating matrices, MATLAB
operations. This includes the ability to **mo**
matrices.

M(i,j): Returns the values located at position (**i,j**) within the matrix, where **i** and **j** indicate the row and the column positions respectively.

M(i:ii,j:jj): Refers to a subset of matrix with **i** to **ii** rows and columns **j** to **jj**.

M(i,:); M(:,j): Return the row **i** or the column **j** respectively.


```
>> M = [7 2 3; 9 5 1; 7 4 9]

M =

     7     2     3
     9     5     1
     7     4     9

>> M(2,:)

ans =

     9     5     1

>> M(:,3)

ans =

     3
     1
     9
>> M(2:3,1:3)

ans =

     9     5     1
     7     4     9
```

2. Create a matrix in I

Important functions for matrices

`size()`: This function determines the dimensions (number of rows and columns) of a matrix.

`numel()` : Returns the number of elements in a matrix.

`det()`: This function computes and returns the determinant of a square matrix.

`trace()`: This function computes the trace or the sum of the diagonal elements

`inv()`: This function generates the inverse of a square matrix.

```
>> M = [7 2 3; 9 5 1]

M =

     7     2     3
     9     5     1

>> size(M)

ans =

     2     3

>> [r, c] = size(M)

r =

     2

c =

     3

>> numel(M)

ans =

     6
```

2. Create a matrix in I

Important functions for matrices

`size()`: This function determines the dimensions (number of rows and columns) of a matrix.

`numel()` : Returns the number of elements in a matrix.

`det()`: This function computes and returns the determinant of a square matrix.

`trace()`: This function computes the trace or the sum of the diagonal elements

`inv()`: This function generates the inverse of a square matrix.

```
>> M = [7 2 3; 9 5 1; 7 4 9]

M =

     7     2     3
     9     5     1
     7     4     9

>> det(M)

ans =

  142.0000

>> trace(M)

ans =

    21

>> inv(M)

ans =

    0.2887   -0.0423   -0.0915
   -0.5211    0.2958    0.1408
    0.0070   -0.0986    0.1197
```

2. Create a matrix i

Important functions for matrice

MATLAB offers other functions which can be applied to matrices such as: **`reshape()`, `fliplr()`, `flipud()` and `rot90()`**.
The indicated functions are used to **reshape**, **flip** or **rotate** the matrix.

The reader can check the MATLAB **help** to know how these functions work.

```
>> help rot90()

--- help for rot90 ---

 rot90  Rotate array 90 degrees.
    B = rot90(A) is the 90 degree
counterclockwise rotation of matrix A. If
    A is an N-D array, rot90(A) rotates in the
plane formed by the first and
    second dimensions.

    rot90(A,K) is the K*90 degree rotation of A,
K = +-1,+-2,...

    Example,
        A = [1 2 3       B = rot90(A) = [ 3 6
             4 5 6 ]                      2 5
                                          1 4 ]

    See also flipud, fliplr, flip.

    Reference page for rot90
```

2. Create a matrix in M

Moreover, in the MATLAB environment, we have the capability to perform additional operations on matrices such as **inserting**, **deleting**, and **modifying** elements within matrices.

`M(i,:) = []`: This command removes the row **`i`** from the matrix

`M(:,j) =[]`: This commands removes the column **`j`** from the matrix

`M(i,j) = a` : This command replaces the element located in the position **`(i,j)`** with the element **`a`**

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

2. Create a matrix in M

Moreover, in the MATLAB environment, we have the capability to perform additional operations on matrices such as **inserting**, **deleting**, and **modifying** elements within matrices.

`M(i,:) = []`: This command removes the row **`i`** from the matrix

`M(:,j) =[]`: This commands removes the column **`j`** from the matrix

`M(i,j) = a` : This command replaces the element located in the position **`(i,j)`** with the element **`a`**

```
>> M = [7 2 3; 9 5 1; 7 4 9]

M =

     7     2     3

     9     5     1

     7     4     9


>> M(:,2) = []


M =


     7     3

     9     1

     7     9
```

2. Create a matrix in M

Moreover, in the MATLAB environment, we have the capability to perform additional operations on matrices such as **inserting**, **deleting**, and **modifying** elements within matrices.

`M(i,:) = []`: This command removes the row **`i`** from the matrix

`M(:,j) =[]`: This commands removes the column **`j`** from the matrix

`M(i,j) = a` : This command replaces the element located in the position **`(i,j)`** with the element **`a`**

```
>> M = [7 2 3; 9 5 1; 7 4 9]

M =

     7     2     3

     9     5     1

     7     4     9


>> M(3,2) = 0


M =

     7     2     3

     9     5     1

     7     0     9
```

2. Create a matrix in M

Moreover, in the MATLAB environment, we have the capability to perform additional operations on matrices such as **inserting**, **deleting**, and **modifying** elements within matrices.

`M(i,:) = V` : This command replaces the row i with the vector **`V`**.

`M(:,j) = V` : This command replaces the column j with the vector **`V`**.

`M=[M, v]`: Writing this expression allows adding a new column where **`v`** is a column vector.

`M=[M; v]`: Writing this expression allows adding a new row where **`v`** is a row vector.

```
>> M = [7 2 3; 9 5 1; 7 4 9]

M =

     7     2     3
     9     5     1
     7     4     9

>> M(:,2) = [10 11 12] % M(:,2) = [10 ;11; 12]

M =

     7    10     3
     9    11     1
     7    12     9
```

2. Create a matrix in M

Moreover, in the MATLAB environment, we have the capability to perform additional operations on matrices such as **inserting**, **deleting**, and **modifying** elements within matrices.

`M(i,:) = V` : This command replaces the row i with the vector **`V`**.

`M(:,j) = V` : This command replaces the column j with the vector **`V`**.

`M=[M, v]`: Writing this expression allows adding a new column where **`v`** is a column vector.

`M=[M; v]`: Writing this expression allows adding a new row where **`v`** is a row vector.

```
>> M = [7 2 3; 9 5 1; 7 4 9]

M =

     7     2     3
     9     5     1
     7     4     9

>> M(3,:) = [10 11 12]

M =

     7     2     3
     9     5     1
    10    11    12
```

2. Create a matrix in

Moreover, in the MATLAB environment, we have the capability to perform additional operations on matrices such as **inserting**, **deleting**, and **modifying** elements within matrices.

`M(i,:) = V` : This command replaces the row i with the vector **`V`**.

`M(:,j) = V` : This command replaces the column j with the vector **`V`**.

`M=[M, v]`: Writing this expression allows adding a new column where **`v`** is a column vector.

`M=[M; v]`: Writing this expression allows adding a new row where **`v`** is a row vector.


```
>> M = [7 2 3; 9 5 1; 7 4 9]

M =

     7     2     3
     9     5     1
     7     4     9

>> v = [10 15 12];

>> M1 = [M; v]

M1 =

     7     2     3
     9     5     1
     7     4     9
    10    15    12
```

2. Create a matrix in

Moreover, in the MATLAB environment, we have the capability to perform additional operations on matrices such as **inserting**, **deleting**, and **modifying** elements within matrices.

`M(i,:) = V` : This command replaces the row i with the vector **`V`**.

`M(:,j) = V` : This command replaces the column j with the vector **`V`**.

`M=[M, v]`: Writing this expression allows adding a new column where **`v`** is a column vector.

`M=[M; v]`: Writing this expression allows adding a new row where **`v`** is a row vector.

```
>> M = [7 2 3; 9 5 1; 7 4 9]

M =

     7     2     3
     9     5     1
     7     4     9

>> v = [10; 15; 12];

>> M1 = [M, v]

M1 =

     7     2     3    10
     9     5     1    15
     7     4     9    12
```

2. Create a matrix in MATLAB

Moreover, in the MATLAB environment, we have the capability to perform additional operations on matrices such as **inserting**, **deleting**, and **modifying** elements within matrices.

To add a new row at the beginning of the matrix we use the notation **`M = [v; M]`**, where **`v`** is a row vector.
`M = [u, M]` is used in MATLAB to add a new column at the beginning of the matrix M where u is a column vector.

2. Create a matrix in I

Algebraic operations

MATLAB provides a wide range of algebraic operations that we can perform on matrices in MATLAB environment.

```
>> A = [5 2 4; 9 2 8]

A =

     5     2     4
     9     2     8

>> B = [6 1 3; 7 2 9]

B =

     6     1     3
     7     2     9

% Adds matrices A and B element-wise
>> M1 = A + B

M1 =

    11     3     7
    16     4    17
```

2. Create a matrix in I

Algebraic operations

MATLAB provides a wide range of algebraic operations that we can perform on matrices in MATLAB environment.

```
>> A = [5 2 4; 9 2 8]

A =

     5     2     4
     9     2     8

>> B = [6 1 3; 7 2 9]

B =

     6     1     3
     7     2     9

% Subtracts matrices B from A element-wise

>> M2 = A - B

M2 =

    -1     1     1
     2     0    -1
```

2. Create a matrix in I

Algebraic operations

MATLAB provides a wide range of algebraic operations that we can perform on matrices in MATLAB environment.

```
>> A = [5 2 4; 9 2 8]

A =

     5     2     4
     9     2     8

>> B = [6 1 3; 7 2 9]

B =

     6     1     3
     7     2     9

% Matrix multiplication element by element
>> M5 = A.*B

M5 =

    30     2    12
    63     4    72
```

2. Create a matrix in

Algebraic operations

MATLAB provides a wide range of algebraic operations that we can perform on matrices in MATLAB environment.

```
>> A = [5 2 4; 9 2 8]

A =

     5     2     4
     9     2     8

>> B = [6 1 3; 7 2 9]

B =

     6     1     3
     7     2     9

% Matrix division  element by element
>> M4 = A./B

M4 =

    0.8333    2.0000    1.3333
    1.2857    1.0000    0.8889
```

2. Create a matrix in M

Algebraic operations

MATLAB provides a wide range of algebraic operations that we can perform on matrices in MATLAB environment.

```
>> A = [5 2 4; 9 2 8]

A =

     5     2     4
     9     2     8

>> B = [6 1 3; 7 2 9]

B =

     6     1     3
     7     2     9

% Matrix multiplication
>> M7 = A*B
Error using  *
Inner matrix dimensions must agree.
```

2. Create a matrix in M

Algebraic operations

MATLAB provides a wide range of algebraic operations that we can perform on matrices in MATLAB environment.

```
>> A = [5 2 4; 9 2 8]

A =

     5     2     4
     9     2     8


>> C = [2 7; 1 3; 8 4]

C =
     2     7
     1     3
     8     4

>> M7 = A*C

M7 =

    44    57
    84   101
```

2. Create a matrix in

Algebraic operations

MATLAB provides a wide range of algebraic operations that we can perform on matrices in MATLAB environment.

A/B = A * inv(B)
Inv(B) = ? ... B is not invertible
In this case MATLAB uses the **least-square** solution to solve the equation :
X = A*inv(B)

```
>> A = [5 2 4; 9 2 8]

A =

     5     2     4
     9     2     8

>> B = [6 1 3; 7 2 9]

B =

     6     1     3
     7     2     9

% Matrix division
>> M6 = A/B

M6 =

    0.5085    0.2903
    0.7578    0.6358
```

Practice