ARRAY AND MATRIX

Academic Year 2017 – 2018

By Ahmed Wael

ONE DIMENSIONAL ARRAY

- One dimensional array is a list of number arranged in a row or a column.
- To create a one dimensional:

Variable_name = [x1 x2 x3 x4 .... xn]

- **EXAMPLE:**

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

ONE DIMENSIONAL ARRAY

- Vector can be created with a constant spacing as follow:

Variable_Name = [m:q:n] or *variable_Name = m:q:n*

- **EXAMPLE:**

```
>> X_1 = [2:2:12]
      X_1 =   2   4   6   8   10   12
```

OR

```
>> X_1 = 2:2:12
      X_1 =   2   4   6   8   10   12
```

ONE DIMENSIONAL ARRAY

- A vector with ***n*** elements that are linearly (equally) spaced in which the first element is x_i and the last element x_n can be created using *linspace* command as follow:

$$Variable\_Name = linspace(x_i , x_n , m)$$

- **EXAMPLE:**

```
>> m  = linspace(1,4,5)
      m =    1.0000    1.7500    2.5000    3.2500    4.0000
```

TWO DIMENSIONAL ARRAY

- A two dimensional array is known also as a MATRIX. Matrix differ from one-dimensional array from the fact that it has numbers of rows and columns.
- Matrix can be created as follow:

Variable_name = [1st row elements; 2nd row elements, …, nth row elements]

- **EXAMPLE:**

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

m =   1   2   3
      4   6   8
      3   9   5
```

TWO DIMENSIONAL ARRAY

- Another method to write the matrix:

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

ZERO, ONE AND EYE MATRIX

- To create a matrix with elements all are equal to ZERO, we use:

Variable_name = zeros(x1,n)

- **EXAMPLE:**

```
>> x = zeros(4,5)
     x =     0     0     0     0     0
             0     0     0     0     0
             0     0     0     0     0
             0     0     0     0     0
```

ZERO, ONE AND EYE MATRIX

- To create a matrix with elements all are equal to ONE, we use:

Variable_name = ones(x1,n)

- **EXAMPLE:**

```
>> x = ones(4,5)
     x =     1     1     1     1     1
             1     1     1     1     1
             1     1     1     1     1
             1     1     1     1     1
```

ZERO, ONE AND EYE MATRIX

- To create a matrix with digonal elements all are equal to ONES, we use:

Variable_name = eye(n)

- **EXAMPLE:**

```
>> x = eye(4)
     x =     1     0     0     0
             0     1     0     0
             0     0     1     0
             0     0     0     1
```

THE TRANSPORT OPERATOR

- **EXAMPLE:**

```
>> x = [3 8 5]
     x =     3     8     5

>> y = x'
     y =
             3
             8
             5
```

- **EXAMPLE:**

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

>> y = x'
     y =     1     4     7
             2     5     8
             3     6     9
```

ADDRESSING A ARRAY

- We can addressing any element of a defined vector using:

  Variable_Name (element we would like to address)

- If we would like to change an element (or elements) in any vector, we follow:

  Variable_Name (element to be changed) = New element

- For a Matrix:

  Variable_Name (row number, column number)

  Variable_Name (row number, column number) = New element

ADDRESSING AN ARRAY

- **EXAMPLE:**

```
>> A = [33 65 88 12 90 41 47 83]
      A =   33   65   88   12   90   41   47   83

>> A(5)
      ans =   90   % The 5th element of the vector

>> A(3) = 244        % Changing the 3rd element of a vector A with 244
      A =   33   65   244   12   90   41   47   83
```

ADDRESSING AN ARRAY

- **EXAMPLE:**

```
>> M = [1 13  55  23;  33  91  16  75;  11  28  44  37]
        M =      1     13     55     23
                33     91     16     75
                11     28     44     37
>> M(3,2)
        ans =    28

>> M(3,3) = 97
        M =      1     13     55     23
                33     91     16     75
                11     28     97     37
```

ADDRESSING OF ARRAY

- *va*(:) Refers to all the elements of the vector *va* (either a row or a column vector).
- *A*(:,*n*) Refers to the elements in all the rows of column *n* of the matrix *A*.
- *A*(*n*,:) Refers to the elements in all the columns of row *n* of the matrix *A*.
- *A*(:,*m*:*n*) Refers to the elements in all the rows between columns *m* and *n* of the matrix *A*.
- *A*(*m*:*n*,:) Refers to the elements in all the columns between rows *m* and *n* of the matrix *A*.
- A(*m*:*n*,*p*:*q*) Refers to the elements in rows *m* through *n* and columns *p* through *q* of the matrix *A*.

ADDRESSING OF ARRAY

- **EXAMPLES:**

>> V = [1:3:15]

V = 1 4 7 10 13

>> U = V(2:5) %Create a new vector (U) with elements taken from vector (V)

U = 4 7 10 13

FUNCTION DEALING WITH ARRAYS

Function	Description	Example

`length(A)`	Returns the number of elements in the vector A.	`>> A=[5 9 2 4];`
		`>> length(A)`
		`ans =`
		`4`

`size(A)`	Returns a row vector `[m,n]`, where `m` and `n` are the size $m \times n$ of the array A.	`>> A=[6 1 4 0 12; 5 19 6 8 2]`
		`A =`
		`6 1 4 0 12`
		`5 19 6 8 2`
		`>> size(A)`
		`ans =`
		`2 5`

`reshape(A, m,n)`	Creates a `m` by `n` matrix from the elements of matrix A. The elements are taken column after column. Matrix A must have `m` times `n` elements.	`>> A=[5 1 6; 8 0 2]`
		`A =`
		`5 1 6`
		`8 0 2`
		`>> B = reshape(A,3,2)`
		`B =`
		`5 0`
		`8 6`
		`1 2`

Function	Description	Example

`diag(v)`	When `v` is a vector, creates a square matrix with the elements of v in the diagonal.	`>> v=[7 4 2];`
		`>> A=diag(v)`
		`A =`
		`7 0 0`
		`0 4 0`
		`0 0 2`

`diag(A)`	When A is a matrix, creates a vector from the diagonal elements of A.	`>> A=[1 2 3; 4 5 6; 7 8 9]`
		`A =`
		`1 2 3`
		`4 5 6`
		`7 8 9`
		`>> vec=diag(A)`
		`vec =`
		`1`
		`5`
		`9`