العلوم	الكلية
الرياضيات	القسم
Programming (MATLAB)	المادة باللغة الانجليزية
البرمجة بلغة (ماتلاب)	المادة باللغة العربية
الثانية	المرحلة الدراسية
صفوت عبدالقادر حمد	اسم التدريسي
Matrix in MATLAB	عنوان المحاضرة باللغة الانجليزية
المصفوفات	عنوان المحاضرة باللغة العربية
الخامسة	رقم المحاضرة
MATLAB A Practical Introduction to Programming and Problem Solving	المصادر والمراجع
MATLAB The Language of Technical Computing	
MATLAB numerical computing	

University Of Anbar

College Of Science

Math Department

MATRIX - LABORATORY (MATLAB)

Lecture 5

Matrix in MATLAB

By:

Safwat A. Hamad

For the 2nd stage Math Department

1. Column Vectors

One way to create a column vector is to explicitly put the values in square brackets, separated by semicolons (rather than commas or spaces):

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

There is no direct way to use the colon operator to get a column vector. However, any row vector created using any method can be transposed to result in a column vector. In general, the transpose of a matrix is a new matrix in which the rows and columns are interchanged. For vectors, transposing a row vector results in a column vector, and transposing a column vector results in a row vector. In MATLAB, the apostrophe (or single quote) is built-in as the transpose operator.

```
>> r = 1:3;
>> c = r'
    c =
    1
    2
    3
```

Example: create Colum vector named (degree_of_student) with degree 3 4 7 2 7 in two ways?

2. Creating Matrix Variables

Creating a matrix variable is simply a generalization of creating row and column vector variables. That is, the values within a row are separated by either spaces or commas, and the different rows are separated by semicolons. For example, the matrix variable mat is created by explicitly entering values:

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

There must always be the same number of values in each row and each column of a matrix. For example, if you attempt to create a matrix in which there are different numbers of values in the rows, the result will be an error message, such as in the following:

```
>> mat = [3 5 7; 1 2]
Dimensions of arrays being concatenated are not consistent.
```

Iterators can be used for the values in the rows using the colon operator. For example:

```
>> mat = [2:4; 3:5]
mat =
     2 3 4
     3 4 5
```

The separate rows in a matrix can also be specified by hitting the Enter key after each row, instead of typing a semicolon when entering the matrix values, as in:

```
>> newmat = [2 6 88
33 5 2]
newmat =
     2 6 88
     33 5 2
```

In addition, can also create a matrix by using the following functions:

<u>Rand and Randi Function</u>

Matrix of random numbers can be created using the **rand** function. If a single value n is passed to rand, an (n * n) matrix will be created; this is called a square matrix (same number of rows and columns).

```
>> rand(2)
ans =
      0.2311    0.4860
      0.6068    0.8913
```

If, instead, two arguments are passed, they specify the number of rows and columns in that order.

```
>> rand(1,3)
ans =
      0.7621    0.4565    0.0185
```

Matrix of random integers can be generated using **randi**; after the range is passed, the dimensions of the matrix are passed (again, using one value n for an (n * n) matrix, or two values for the dimensions):

```
>> randi([5 10], 2)
ans =
      8   10
      9   5
>> randi([10 30], 2, 3)
ans =
      21   10   13
      19   17   26


>>randi ([20 40],4,5)
ans =
     31     28     22     30     39
     34     29     23     27     34
     40     36     27     23     29
     26     37     21     24     39
```

Note: that the range can be specified for **randi**, but not for **rand**. The format for calling these functions is different.

Zeros and Ones Function

MATLAB also has several functions that create special matrix. For example, the zeros function creates a matrix of all zeros and the ones function creates a matrix of all ones. Like rand, either one argument can be passed (which will be both the number of rows and columns) or two arguments (first the number of rows and then the number of columns).

```
>> zeros(3)
ans =
      0  0  0
      0  0  0
      0  0  0


>> zeros (3,5)
ans =
        0        0        0        0        0
        0        0        0        0        0
        0        0        0        0        0
>> ones(2,4)
ans =
      1    1    1    1
      1    1    1    1
```

Note: that there is no twos function, or tens, or fifty-threes—just zeros and ones!

H.W. - Explain what the eye function for matrix, and how to use it in MATLAB, and, Are there any functions used for matrix?

3. Referring to and Modifying Matrix Elements

To refer to matrix elements, the row and then the column subscripts are given in parentheses (always the row first and then the column). For example, this creates a matrix variable *mat* and then refers to the value in the second row, third column of mat:

```
>> mat = [2:4; 3:5]
mat =
     2    3    4
     3    4    5
>> mat(2,3)
ans =
     5
```

This is called **subscripted indexing**; it uses the row and column subscripts. It is also possible to refer to a subset of a matrix. For example, this refers to the first to third rows, and third to fifth columns:

- `>> mat = randi ([20 40], 4, 5)`

```
mat = 40     40     28     33     34

      40     30     39     20     35

      23     36     36     37     35

      40     22     40     39     28
```

- `>> mat (1:3 , 3:5)`

```
ans = 28     33     34

      39     20     35

      36     37     35
```

Using just one colon by itself for the row subscript means all rows, regardless of how many, and using a colon for the column subscript means all columns. For example, this refers to all columns within the first row or, in other words, the entire first row:

- `>> mat (1, :)`

```
ans =
      40     40     28     33     34
>> mat (:, 1)

ans =

     40

      40

     23

     40
```

If a single index is used with a matrix, MATLAB unwinds the matrix column by column. For example, for the matrix *intmat* created here, the first two elements are from the first column, and the last two are from the second column:

```
>> intmat = [100 77; 28 14]
intmat =
    100   77
    28    14
>> intmat(1)
ans =
    100
>> intmat(2)
ans =
    28
>> intmat(3)
ans =
  77
>> intmat(4)
ans =
  14
```

This is called **linear indexing**. Note that it is usually much better style when working with matrix to use subscripted indexing.

MATLAB stores matrix in memory in column major order, or column wise, which is why linear indexing refers to the elements in order by columns. An individual element in a matrix can be modified by assigning a new value to it.

```
>> mat = [2:4; 3:5];
>> mat(1,2) = 11
mat =
   2   11   4
   3   4    5
```

An entire row or column could also be changed. For example, the following replaces the entire second row with values from a vector obtained using the colon operator.

```
>> mat(2,:) = 5:7
mat =
     2   11   4
     5   6    7
```

Notice that as the entire row is being modified, a vector with the correct length must be assigned (although that vector could be either a row or a column). Any subset of a matrix can be modified, as long as what is being assigned has the same number of rows and columns as the subset being modified.

```
>> mat(1:2, 2:3) = 4:5
Unable to perform assignment because the size of the left side is 2-by-2 and the size of the right side is 1-by-2.
>> mat(1:2, 2:3) = zeros(2)
mat =
2 0 0
5 0 0
```

The exception to this rule is that a scalar can be assigned to any size subset of a vector or matrix; what happens is that the same scalar is assigned to every element referenced. For example,

```
>> m = randi([10 50], 3,5)
m =
   38   11   38   11   41
   11   13   23   27   42
   21   43   48   25   17
>> m(2:3,3:5) = 3
m =
   38    11   8   11   41
  11    13   3    3    3
  21    43   3    3    3
```

To extend a matrix, an individual element could not be added as that would mean there would no longer be the same number of values in every row. However, an entire row or column could be added. For example, the following would add a fourth column to the matrix mat created previously.

```
>> mat(:,4) = [9 2]'
mat =
2 0 0 9
5 0 0 2
```

Just as we saw with vectors, if there is a gap between the current matrix and the row or column being added, MATLAB will fill in with zeros.

```
>> mat(4,:) = 2:2:8
mat =
2 0 0 9
5 0 0 2
0 0 0 0
2 4 6 8
```

Exercises:

1. Generate the following matrix:

$$A = \begin{bmatrix} 12 & 34 & 7 & 9 & -2 \\ 22 & 18 & -33 & 8 & 45 \\ 1 & 43 & -3 & -77 & 98 \end{bmatrix}$$

2. Generate the following vector Colum b = (22 44 55 66 77)
3. Extract the sub-matrix in rows 2 to 3 and columns 1 to 2 of the matrix A in Exercise 1 above.
4. Extract the second column of the matrix A in Exercise 1 above.
5. Extract the first row of the matrix A in Exercise 1 above.