كلية المستقبل الجامعة

قسم الفيزياء الطبية

MATLAB

Lec - 5

اعداد

م.م علا علي عبيد

Introduction:

Matrices are the basic elements of the MATLAB environment. A matrix is a two-dimensional array consisting of m rows and n columns. Special cases are column vectors (n = 1) and row vectors (m = 1). In this section we will illustrate how to apply different operations on matrices. The following topics are discussed: vectors and matrices in MATLAB, the inverse of a matrix, determinants, and matrix manipulation. MATLAB supports two types of operations, known as matrix operations and array operations. Matrix operations will be discussed first.

Matrix generation :

Matrices are fundamental to MATLAB. Therefore, we need to become familiar with matrix generation and manipulation. Matrices can be generated in several ways

Entering a vector:

A vector is a special case of a matrix. The purpose of this section is to show how to create vectors and matrices in MATLAB. As discussed earlier, an array of dimension 1 ×n is called a row vector, whereas an array of dimension m × 1 is called a column vector. The elements of vectors in MATLAB are enclosed by square brackets and are separated by spaces or by commas. For example, to enter a row vector, v, type

```
>> v = [1 4 7 10 13]

v =

1    4    7    10    13
```

Column vectors are created in a similar way, however, semicolon (;) must separate the components of a column vector,

```
>> w = [1;4;7;10;13]

w =
```

1

4

7

10

13

On the other hand, a row vector is converted to a column vector using the transpose operator. The transpose operation is denoted by an apostrophe or a single quote (’).

```
>> w = v’
w =
1
4
7
10
13
```

Thus, v(1) is the first element of vector v, v(2) its second element, and so forth. Furthermore, to access blocks of elements, we use MATLAB’s colon notation (:). For example, to access the first three elements of v, we write,

```
>> v(1:3)
ans =
    1   4  7
```

Or, all elements from the third through the last elements,

```
 >> v(3,end)
```

Ans=

7 10 13

where end signifies the last element in the vector. If v is a vector, writing

>> v(:)

produces a column vector, whereas writing

>> v(1:end)

produces a row vector

Entering a matrix

A matrix is an array of numbers. To type a matrix into MATLAB you must

• begin with a square bracket, [

• separate elements in a row with spaces or commas (,)

• use a semicolon (;) to separate rows

- end the matrix with another square bracket,].

Here is a typical example. To enter a matrix `A`, such as,

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \tag{2.1}$$

type,

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

MATLAB then displays the 3×3 matrix as follows,

```
A   =
    1   2   3
    4   5   6
    7   8   9
```

Note that the use of semicolons (`;`) here is different from their use mentioned earlier to suppress output or to write multiple commands in a single line.

Once we have entered the matrix, it is automatically stored and remembered in the *Workspace*. We can refer to it simply as matrix `A`. We can then view a particular element in a matrix by specifying its location. We write,

```
>> A(2,1)
ans  =
     4
```

`A(2,1)` is an element located in the second row and first column. Its value is 4.

2.5.3 Matrix indexing

We select elements in a matrix just as we did for vectors, but now we need two indices. The element of row i and column j of the matrix `A` is denoted by `A(i,j)`. Thus, `A(i,j)` in MATLAB refers to the element A_{ij} of matrix `A`. The *first* index is the *row* number and the *second* index is the *column* number. For example, `A(1,3)` is an element of *first* row and *third* column. Here, `A(1,3)=3`.

Correcting any entry is easy through indexing. Here we substitute `A(3,3)=9` by `A(3,3)=0`. The result is

```
>> A(3,3) = 0
A   =
    1    2    3
    4    5    6
    7    8    0
```