MATLAB

Stage2

Lec5

MATRIX

MS.c Ola Ali

MS.c Haneen Alharir

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 opera tions. Matrix operations will be discussion.

Matrix generation:

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

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.

Matrix index:

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
```

Colon operator in a matrix :

The colon operator can also be used to pick out a certain row or column. For example, the statement A(m:n,k:l specifies rows m to n and column k to l. Subscript expressions refer to portions of a matrix.

For example,

```
>> A(2,:)
ans  =
     4   5   6
```

is the second row elements of `A`.

The colon operator can also be used to extract a sub-matrix from a matrix `A`.

```
>> A(:,2:3)
ans  =
     2   3
     5   6
     8   0
```

`A(:,2:3)` is a sub-matrix with the last two columns of `A`.

A row or a column of a matrix can be deleted by setting it to a *null* vector, [ ].

```
>> A(:,2)=[]
ans  =
     1   3
     4   6
     7   0
```

To extract a *submatrix* `B` consisting of rows 2 and 3 and columns 1 and 2 of the matrix `A`, do the following

```
>> B = A([2 3],[1 2])
B    =
     4     5
     7     8
```

To interchange rows 1 and 2 of `A`, use the vector of row indices together with the colon operator.

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

It is important to note that the *colon operator* (`:`) stands for *all columns* or *all rows*. To create a vector version of matrix `A`, do the following