Al-Mustaqbal University
College of Engineering and Technology
Department of Medical Instrumentation Techniques Engineering
Class: First Class
Subject: Computer Programming and application I
Lecturer: Dr.Myasar Mundher Adnan
Lecture Address: Matrices
2024 - 2025

Matrices

2.1 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.

2.2 Matrix generation:

2.2.1 Entering a vector

A vector is a special case of a matrix. 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.

2.2.2 Entering a matrix

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

1. begin with a square bracket, [
2. separate elements in a row with spaces or commas (,)
3. use a semicolon (;) to separate rows
4. end the matrix with another square bracket,].

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

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

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

```
>> A (2,1)
```

2.4.3 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 *Aij* 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.

A=??

Hence can be correct entry of each element in matrix by :
A(i,j)=any number

For example

A(3,3)=9 to change 9 to 0 type

```
>> A (3,3) = 0
```
The result as:

```
A   =
     1     2     3
     4     5     6
     7     8     0
```

Single elements of a matrix are accessed as A(i,j), where $i \geq 1$ and $j \geq 1$. Zero or negative subscripts are not supported in MATLAB.

2.2.4 Colon operator:

The colon operator will prove very useful and understanding how it works is the key to efficient and convenient usage of MATLAB. It occurs in several different forms. Often we must deal with matrices or vectors that are too large to enter one element at a time. For example, suppose we want to enter a vector *x* consisting of points (0,0.1,0.2,0.3,…..5). We can use the command

```
>> x = 0:0.1:5;
```

The row vector has 51 elements.

Another example g from (1 to100 step 1)

2.4.5 Linear spacing

there is a command to generate linearly spaced vectors: linspace. It is similar to the colon operator (:), but gives direct control over the number of points. For example,

y = linspace (a,b)

generates a row vector y of 100 points linearly spaced between and including a and b.

y = linspace (a, b, n)

generates a row vector y of n points linearly spaced between and including a and b. This is useful when we want to divide an interval into a number of subintervals of the same length.

For example,

>> theta = linspace(0,2*pi,101)

divides the interval $[0, 2\pi]$ into 100 equal subintervals, then creating a vector of 101 elements.

2.4.6 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
```

2.2.7 Creating a sub-matrix

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

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

2.2.8 Deleting row or column

To delete a row or column of a matrix, use the *empty vector* operator, [ ].

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

Third row of matrix A is now deleted. To restore the third row, we use a technique for creating a matrix

```
>> A = [A(1,:);A(2,:);[7 8 0]]

A =
     1 2 3
     4 5 6
     7 8 0
```

Matrix A is now restored to its original form.

2.2.9 Dimension

To determine the *dimensions* of a matrix or vector, use the command size. For example,

```
>> size(A)
ans =
     3 3
```

means 3 rows and 3 columns.
Or more explicitly with,

```
>> [m,n]=size(A)
```

2.2.10 Transposing a matrix

The *transpose* operation is denoted by an apostrophe or a single quote ('). It flips a matrix
about its main diagonal and it turns a row vector into a column vector. Thus, By using linear algebra notation, the transpose of *m £ n* real matrix A is the *n £ m* matrix that results from interchanging the rows and columns of A. The transpose matrix is denoted *AT*.

```
>> A'
ans =
     1 4 7
     2 5 8
     3 6 0
```