Computer Science II

Environmental Engineering
Second Level
2024-2025
1st Course

COLLEGE OF ENGINEERING - كلية الهندسة
Tikrit University - جامعة تكريت

Lecturer Assist. Aalaa Ahmed

Lecture #4

Programming with MATLAB (Cont.)

COLLEGE OF ENGINEERING - كلية الهندسة
Tikrit University - جامعة تكريت

Lecturer Assist. Aalaa Ahmed

13. MATRICES

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.

13. MATRICES (Cont.)

MATLAB supports two types of operations, known as matrix operations and array operations. Matrix operations will be discussed first.

End as a subscript to access the last element of a matrix along a given dimension use end as a subscript.

13. MATRICES (Cont.)

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.

13. MATRICES (Cont.)

Entering a vector (Cont.):

For example, to enter a row vector, v, type:

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

13. MATRICES (Cont.)

Entering a vector (Cont.): 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
```

13. MATRICES (Cont.)

Entering a vector (Cont.):

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
```

13. MATRICES (Cont.)

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

is the second row elements of A.

13. MATRICES (Cont.)

Colon operator in a matrix (Cont.) :

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
```

13. MATRICES (Cont.)

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.

13. MATRICES (Cont.)

Creating a sub-matrix (Cont.):

```
>> 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:

13. MATRICES (Cont.)

Creating a sub-matrix (Cont.):

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

13. MATRICES (Cont.)

Creating a sub-matrix (Cont.):

The submatrix comprising the intersection of rows p to q and columns r to s is denoted by A(p:q,r:s).

As a special case, a colon (:) as the row or column specifier covers all entries in that row or column; thus:

- A(:,j) is the jth column of A, while
- A(i,:) is the ith row, and
- A(end,:) picks out the last row of A.

13. MATRICES (Cont.)

Creating a sub-matrix (Cont.):

The keyword end, used in A(end,:), denotes the last index in the specified dimension. Here are some examples.

```
>> A(2:3, 2:3)  →   ans = 5   6
                          8   9

>> A(end:-1:1, end)
ans = 9
      6
      3
```

13. MATRICES (Cont.)

Creating a sub-matrix (Cont.):

```
>> A( [1 3] , [2 3] )

ans = 2   3

      8   9
```

Ex:-

```
>> q = 4:10

q = 4  5  6  7  8  9  10

>> q (end)

ans = 10
```

13. MATRICES (Cont.)

Creating a sub-matrix (Cont.):

```
>> q(end-2 : end)
```

ans = 8 9 10

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

13. MATRICES (Cont.)

Creating a sub-matrix (Cont.):

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.

13. MATRICES (Cont.)

Creating a sub-matrix (Cont.):

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.