Arrays and Matrices in MATLAB

EE 201

Session Agenda

- Contact before work 5 min.
- Arrays and Matrices in MATLAB 70 min.

Class Learning Objectives

- Achieve Comprehension LOL of Arrays and Matrices in MATLAB.

Creating Numeric Matrix

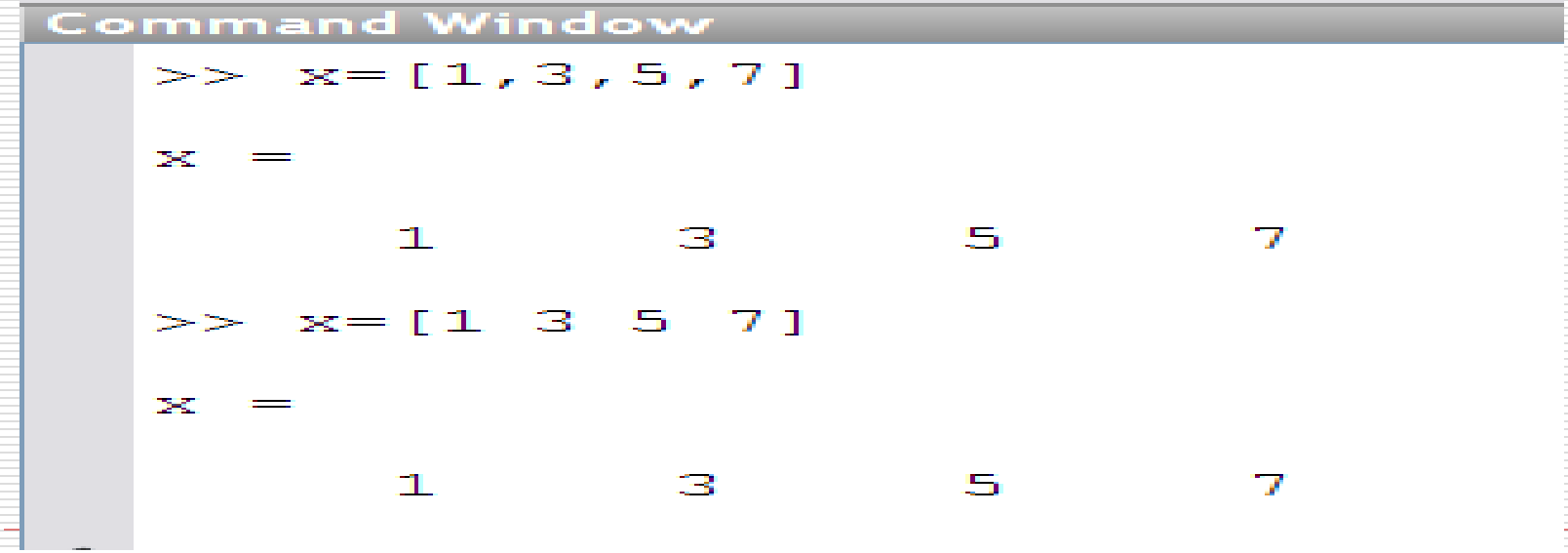
- We can create numeric array using:

a- Square bracket

b- Colon operator

Square bracket([ ])

- Row Vetor:The elements of the row must be separated by commas or spaces.
- Example:



Column Vector:

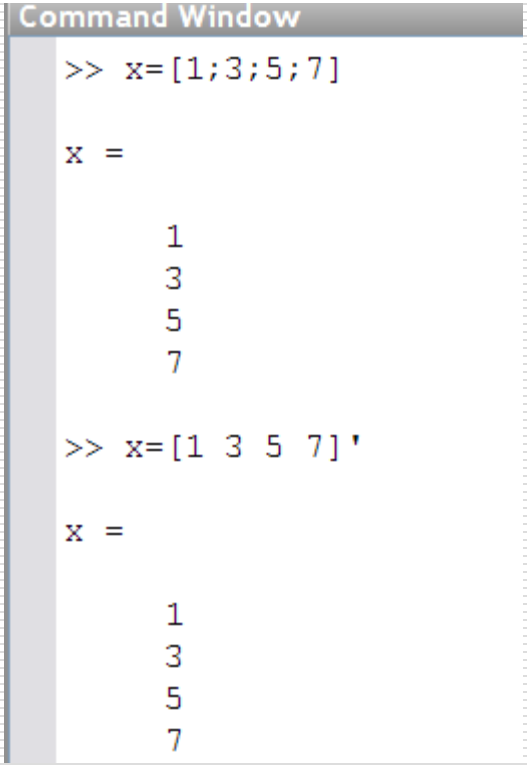
The elements of the Column must be separated by:

-semicolon or use the

-transpose notation(')

which converts a row vector into a column vector or vice versa.

For example:



Colon Operator(:)

- The colon operator **generates a sequence** of numbers that you can use in **creating** or **indexing** into Matrices.
- **Numeric Sequence Range**

Generate a sequential series of regularly spaced numbers from first to last using the syntax **first:last**. For an incremental sequence from 6 to 17, use:

```
N = 6:17
```

Example:

Command Window

```
>> N=6:17

N =

  Columns 1 through 11

     6     7     8     9    10    11    12    13    14    15    16

  Column 12

    17

>> N=[6:17]

N =

  Columns 1 through 11

     6     7     8     9    10    11    12    13    14    15    16

  Column 12

    17
```

Colon Operator(:)

- **Numeric Sequence Step**

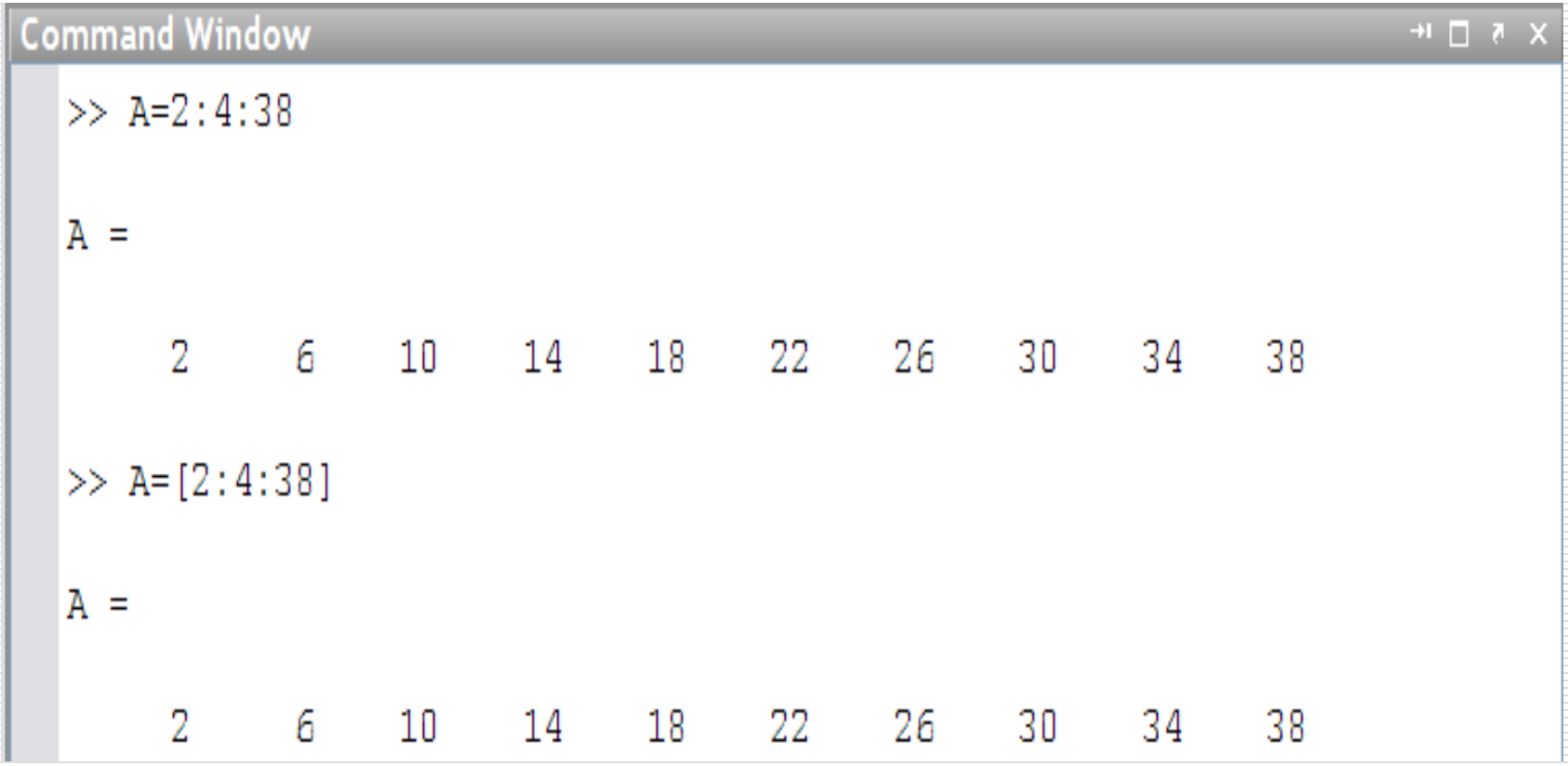
Generate a sequential series of numbers, each number separated by a step value, using the syntax :

first:step:last.

For a sequence from 2 through 38, stepping by 4 between each entry, use:

A = 2:4:38

Example:


Command Window
>> A=2:4:38
A =
2 6 10 14 18 22 26 30 34 38
>> A=[2:4:38]
A =
2 6 10 14 18 22 26 30 34 38

linspace command

- The **linspace** command also creates a **linearly spaced row vector**, but instead you specify the number of values rather than the increment.

The syntax is **linspace (x_1,x_2,n)**, where x_1 and x_2 are the lower and upper limits and n is the number of points.

-For example, **linspace (5,8,31)** is equivalent to [5:0.1:8].

-If n is omitted, generates a row vector of 100 linearly equally spaced points between x_1 and x_2.

logspace command

- The **logspace** command creates an array of *logarithmically* spaced elements.
- Its syntax is logspace(a,b,n), where n is the number of points between 10^a and 10^b.

For example, x=logspace(-1,1,4) is

x=[0.1000,0.4642, 2.1544,10.000].

If n is omitted, the number of points defaults to 50.

Vector Index

- Vector index, points to a particular element in the array.
- It uses to know the value of element in the vector .

Example:

Use MATLAB to compute w=5sinu for u=0,0.1,0.2…..10 and determine the value of the seventh element in the vector u and w.

Solution of example

```
>>  u=[0:0.1:10];
>>  w=5*sin(u);
>>  u(7)

ans  =

      0.6000

>>  w(7)

ans  =

      2.8232
```

Matrices

- A matrix has multiple rows and columns. For example, the matrix

$$M = \begin{bmatrix} 2 & 4 & 10 \\ 16 & 3 & 7 \\ 4 & 5 & 9 \\ 11 & 21 & 1 \end{bmatrix}$$

has four rows and three columns.

- Vectors are special cases of matrices having one row or one column.

Creating Matrices

- If the matrix is small you can type it row by row, separating the *elements* in a given row with *spaces* or *commas* and separating the *rows* with semicolons. For example, typing:

```
>>A=[2,4,10;16,3,7];
```

- creates the following matrix:

$$A=\begin{bmatrix} 2 & 4 & 10 \\ 16 & 3 & 7 \end{bmatrix}$$

- Remember, spaces or commas separate elements in different *columns*, whereas semicolons separate elements in different *rows*.

Creating Matrices from Vectors

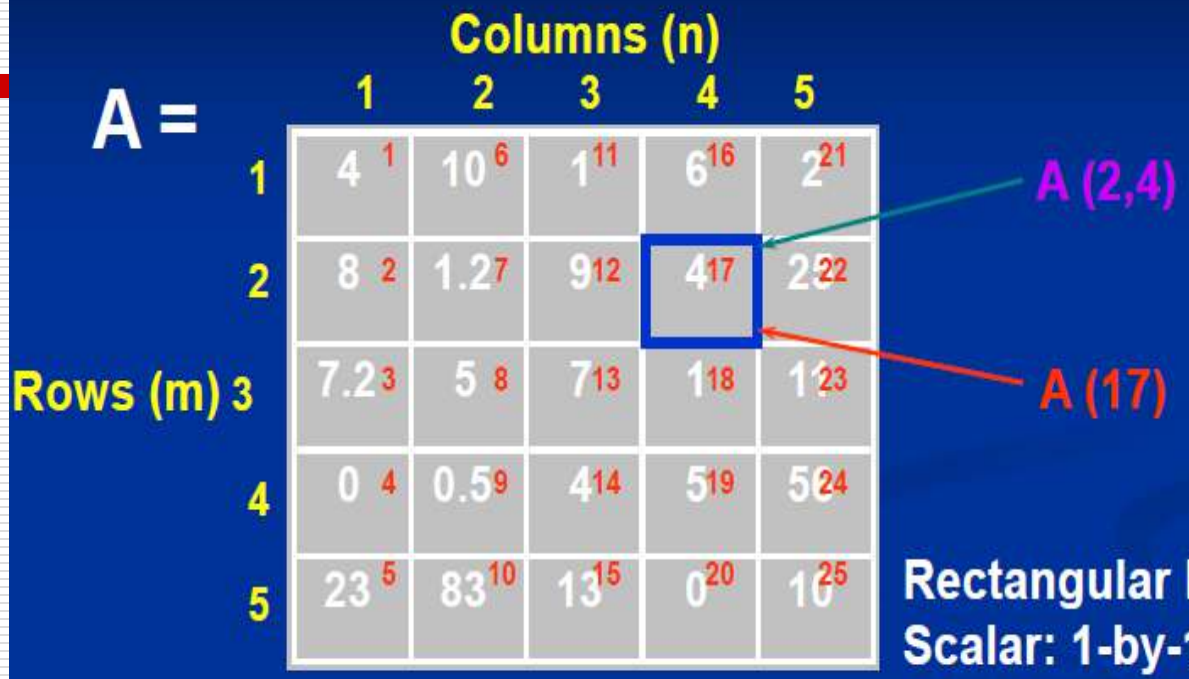
- Suppose a= [1,3,5] and b=[7,9,11] (row vectors). Note the difference between the results given by:

[a b] and [a ; b] in the following session:

```
>>c=[a  b]
c=
    1 3 5 7 9 11
```

```
>>d=[a;b]
d=
    1 3 5
    7 9 11
```

The Matrix in MATLAB



A = 5 x 5 matrix.

Rectangular Matrix:
Scalar: 1-by-1 array
Vector: m-by-1 array
1-by-n array
Matrix: m-by-n array

Array Addressing

- v(:) represents all the row or column elements of the vector v.
- v(2:5) represents the second through fifth elements; that is v(2), v(3), v(4), v(5)
- A(3,:) denotes all the elements in the third Row of the matrix A
- A(:,2) denotes all the elements in the Second column of the matrix A
- A(:,2:5) denotes all the elements in the second through fifth columns of A.
- A(2:3,1:3) denotes all the elements in the second and third rows that are also in the first through third columns.

- You can use array indices to extract a smaller array from another array. For example, if you first create the array **B.**

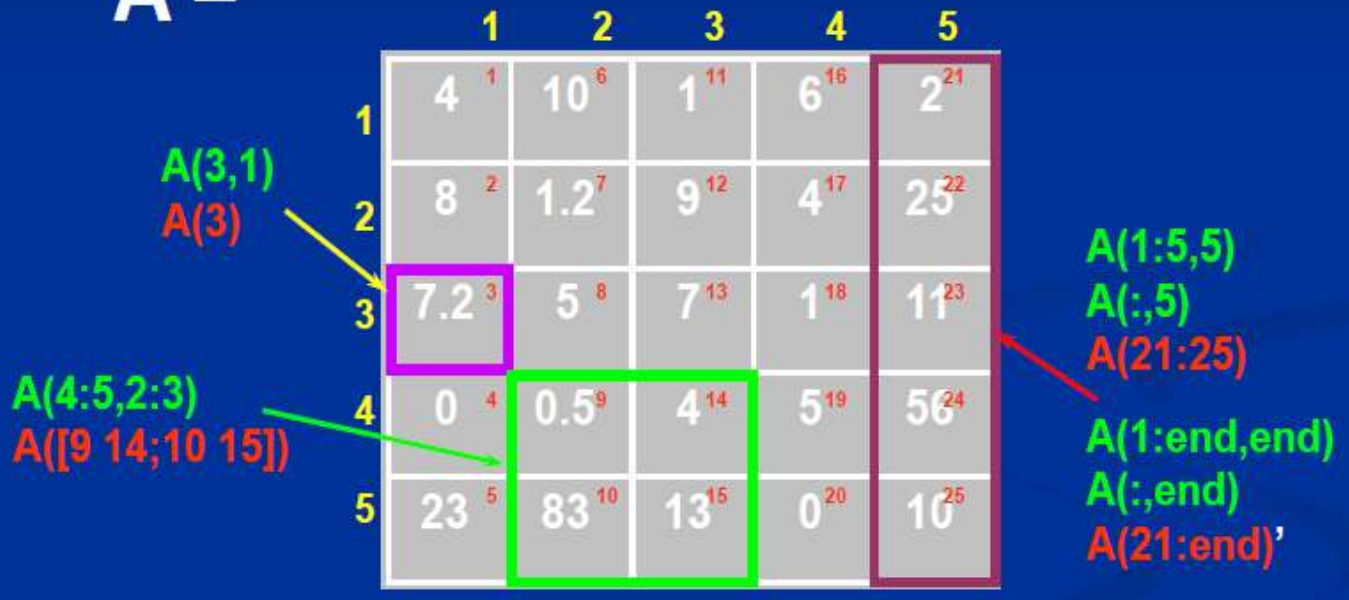
$$\mathbf{B}=\begin{bmatrix} 2 & 4 & 10 & 13 \\ 16 & 3 & 7 & 18 \\ 8 & 4 & 9 & 25 \\ 3 & 12 & 15 & 17 \end{bmatrix}$$

- then type **C**=**B**(2:3,1:3),you can produce the following array:

$$\mathbf{C}=\begin{bmatrix} 16 & 3 & 7 \\ 8 & 4 & 9 \end{bmatrix}$$

Array Subscripting / Indexing

A =



EMPTY ARRAY

- The empty(null) array contains no elements and is expressed as [].
- Rows and columns can be deleted by setting the selected row or column equal to the null array, for example:

-A(3,:)=[ ] deletes the third row in A

A(1:4,:)=[] delete the first 4 rows in A

-A([1 4],:)=[ ] deletes the first row and fourth rows of A

A(:,[1 4])=[ ] deletes the first column and fourth column of A

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

-A(1,5)=3 changes matrix to: A= $\begin{bmatrix} 6 & 9 & 4 & 0 & 3 \\ 1 & 5 & 7 & 0 & 0 \end{bmatrix}$

A(1,4:5)=3 -- 4th and 5th Elements in 1st row of A is 3

A(1:2,3)=5--- 3rd Element of 1st ,2nd row of A is 5

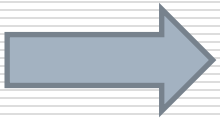
Extract the last two rows and colums

-B=A(:,5:-1:1)  B= $\begin{bmatrix} 3 & 0 & 4 & 9 & 6 \\ 0 & 0 & 7 & 5 & 1 \end{bmatrix}$

-suppose C=[-4,12,3,5,8] , B(2,:)=C

 B= $\begin{bmatrix} 3 & 0 & 4 & 9 & 6 \\ -4 & 12 & 3 & 5 & 8 \end{bmatrix}$

-suppose D=[3,8,5;2,-6,9] , E=D([2,2,2],:)

 E= $\begin{bmatrix} 2 & -6 & 9 \\ 2 & -6 & 9 \\ 2 & -6 & 9 \end{bmatrix}$