Introduction to Matlab
Matlab Screen

- **Command Window**
  - type commands

- **Current Directory**
  - View folders and m-files

- **Workspace**
  - View program variables
  - Double click on a variable to see it in the Array Editor

- **Command History**
  - view past commands
  - save a whole session using diary
Extracting a Sub-Matrix

- A portion of a matrix can be extracted and stored in a smaller matrix by specifying the names of both matrices and the rows and columns to extract. The syntax is:

  \[ \text{sub\_matrix} = \text{matrix} ( r1 : r2 , c1 : c2 ) ; \]

  where \( r1 \) and \( r2 \) specify the beginning and ending rows and \( c1 \) and \( c2 \) specify the beginning and ending columns to be extracted to make the new matrix.
# MATLAB Matrices

- A column vector can be extracted from a matrix. As an example, we create a matrix below:

```matlab
matrix = [1,2,3;4,5,6;7,8,9]
```

```matlab
» col_two = matrix(:, 2)
col_two =
    2
    5
    8
```

- Here we extract column 2 of the matrix and make a column vector:

```matlab
» col_two = matrix(:, 2)
col_two =
    2
    5
    8
```
## MATLAB Matrices

- A row vector can be extracted from a matrix. As an example we create a matrix below:

  ```matlab
  » matrix=[1,2,3;4,5,6;7,8,9]
  matrix =
  1     2     3
  4     5     6
  7     8     9
  
  » rowvec=matrix(2 : 2 , 1 : 3)
  rowvec =
  4     5     6
  ```

- Here we extract row 2 of the matrix and make a row vector. Note that the 2:2 specifies the second row and the 1:3 specifies which columns of the row.

  ```matlab
  » rowvec=matrix(2 : 2 , 1 : 3)
  rowvec =
  4     5     6
  ```
Concatenation of Matrices

- \( \mathbf{x} = \begin{bmatrix} 1 & 2 \end{bmatrix}, \, \mathbf{y} = \begin{bmatrix} 4 & 5 \end{bmatrix}, \, \mathbf{z} = \begin{bmatrix} 0 & 0 \end{bmatrix} \)

\[
\mathbf{A} = \begin{bmatrix} \mathbf{x} & \mathbf{y} \end{bmatrix}
\begin{bmatrix} 1 & 2 & 4 & 5 
\end{bmatrix}
\]

\[
\mathbf{B} = \begin{bmatrix} \mathbf{x} ; \mathbf{y} \end{bmatrix}
\begin{bmatrix} 1 & 2 
4 & 5 
\end{bmatrix}
\]
Reading Data from files

- MATLAB supports reading an entire file and creating a matrix of the data with one statement.

```matlab
>> load mydata.dat; % loads file into matrix.
% The matrix may be a scalar, a vector, or a matrix with multiple rows and columns. The matrix will be named mydata.

>> size (mydata) % size will return the number of rows and number of columns in the matrix
>> length (myvector) % length will return the total no. of elements in myvector
```
Plotting with MATLAB

• MATLAB will plot one vector vs. another. The first one will be treated as the abscissa (or x) vector and the second as the ordinate (or y) vector. The vectors have to be the same length.

• MATLAB will also plot a vector vs. its own index. The index will be treated as the abscissa vector. Given a vector “time” and a vector “dist” we could say:

```matlab
>> plot (time, dist) % plotting versus time
>> plot (dist) % plotting versus index
```
Plotting with MATLAB

• There are commands in MATLAB to "annotate" a plot to put on axis labels, titles, and legends. For example:

```matlab
>> % To put a label on the axes we would use:
>> xlabel ('X-axis label')
>> ylabel ('Y-axis label')
```

```matlab
>> % To put a title on the plot, we would use:
>> title ('Title of my plot')
```
Plotting with MATLAB

- Vectors may be extracted from matrices. Normally, we wish to plot one column vs. another. If we have a matrix “mydata” with two columns, we can obtain the columns as a vector with the assignments as follows:

```matlab
>> first_vector = mydata (: , 1) ;       % First column
>> second_vector = mydata (: , 2) ;    % Second one
>> % and we can plot the data
>> plot ( first_vector , second_vector )
```
Specifying Line Styles and Colors

It is possible to specify color, line styles, and markers (such as plus signs or circles) when you plot your data using the plot command:

\[
\text{plot}(x,y, 'color\_style\_marker')
\]

For example: \[
\text{plot}(x,y,'r:+')
\] plots a red-dotted line and places plus sign markers at each data point.
Some Useful MATLAB commands

• who       List known variables
• whos      List known variables plus their size
• help      >> help sqrt       Help on using sqrt
• lookfor   >> lookfor sqrt    Search for
             keyword sqrt in m-files
• what      >> what a:         List MATLAB files in a:
• clear     Clear all variables from work space
• clear x y Clear variables x and y from work space
• clc       Clear the command window
Some Useful MATLAB commands

• what List all m-files in current directory
• dir List all files in current directory
• ls Same as dir
• type test Display test.m in command window
• delete test Delete test.m
• cd a: Change directory to a:
• chdir a: Same as cd
• pwd Show current directory
• which test Display directory path to ‘closest’ test.m
MATLAB Relational Operators

- MATLAB supports six relational operators.

  - Less Than: `<`
  - Less Than or Equal: `<=`
  - Greater Than: `>`
  - Greater Than or Equal: `>=`
  - Equal To: `==`
  - Not Equal To: `~=`
MATLAB Logical Operators

• MATLAB supports three logical operators.

  not ~ % highest precedence
  and & % equal precedence with or
  or | % equal precedence with and
MATLAB Logical Functions

- MATLAB also supports some logical functions.
  xor (exclusive or) Ex: xor (a, b)
  Where a and b are logical expressions. The xor operator evaluates to true if and only if one expression is true and the other is false. True is returned as 1, false as 0.
  any (x) returns 1 if any element of x is nonzero
  all (x) returns 1 if all elements of x are nonzero
  isnan (x) returns 1 at each NaN in x
  isinf (x) returns 1 at each infinity in x
  finite (x) returns 1 at each finite value in x
Matlab Selection Structures

• An if - elseif - else structure in MATLAB. Note that elseif is **one** word.

```matlab
if expression1 % is true
    % execute these commands
elseif expression2 % is true
    % execute these commands
else % the default
    % execute these commands
end
```
MATLAB Repetition Structures

- A for loop in MATLAB
  ```matlab
  for ind = 1:100
    b(ind)=sin(ind/10)
  end
  
  Alternative:
  ```
  ```matlab
  x=0.1:0.1:10; b=sin(x);
  ```  
  - Most of the loops can be avoided!!!

- A while loop in MATLAB
  ```matlab
  while x <= 10
    % execute these commands
  end
  ```
Scalar - Matrix Addition

```matlab
» a=3;
» b=[1, 2, 3;4, 5, 6]
b =
1     2     3
4     5     6
» c= b+a  % Add a to each element of b
c =
4     5     6
7     8     9
```

Slides prepared by : Farzana Rahman
Scalar - Matrix Subtraction

```matlab
» a=3;
» b=[1, 2, 3;4, 5, 6]
b =
    1     2     3
    4     5     6
» c = b - a  %Subtract a from each element of b
c =
   -2    -1     0
    1     2     3
```
Scalar - Matrix Multiplication

```plaintext
» a=3;
» b=[1, 2, 3; 4, 5, 6]
b =

1  2  3
4  5  6

» c = a * b  % Multiply each element of b by a
c =

3  6  9
12 15 18
```
Scalar - Matrix Division

```matlab
>> a=3;
>> b=[1, 2, 3; 4, 5, 6]
b =
     1     2     3
     4     5     6
>> c = b / a  % Divide each element of b by a
c =
     0.3333   0.6667   1.0000
     1.3333   1.6667   2.0000
```
END