Working with MATLAB

Welcome and Goodluck

Syed Khaleel Ahmed

Dept. of Electronics and Communication Engg.,
Universiti Tenaga Nasional

April 14, 2016

Objectives

On the successful completion of this workshop, you should be able to

- use MATLAB to perform mathematical calculations,
- use MATLAB to define and manipulate scalars, vectors, and matrices,
- use pre-defined functions in MATLAB,
- effectively use the in-built help to accelerate the pace of learning,
- use MATLAB to visualize data by producing appropriate plots, and
- write simple MATLAB programs.

Introduction

1. What is MATLAB?
2. What is available in MATLAB?
3. Why MATLAB?
4. Where MATLAB?

Introduction (contd.)

What is MATLAB?

A high-level software for

- *numerical computation*,
- *data analysis*, and
- *visualization*.

Introduction (contd.)

What is available in MATLAB?

- Pre-defined functions.
- Toolboxes.
- SIMULINK.
- Blocksets.

Introduction (contd.)

Why MATLAB?

- de-facto industry standard, especially in engineering,
- easy to use, and
- availability of toolboxes and blocksets.

Introduction (contd.)

Where MATLAB?

- Automotive,
- Signal Processing,
- Communication,
- Aerospace,
- Finance and Economics,
- Computer, and
- *many more*.

Contents

1. MATLAB User Interface Layout.
2. Working with Variables.
3. Visualizing Data.
4. Programming.
5. Case Study.

Contents (contd.)

1. *MATLAB User Interface Layout*.
2. Working with Variables.
3. Visualizing Data.
4. Programming.
5. Case Study.

User Interface

- Basic Components of MATLAB
- MATLAB as a Calculator
- Operators and Operator Precedence
- Pre-defined Functions

User Interface (contd.)

Basic Components of MATLAB

- To start
  - *Double-click MATLAB on desktop* or
  - *click start menu*.
- MATLAB user interface or desktop environment
  - *Command Window*
  - *Command History*
  - *Current Directory browser*
  - *Workspace Browser*

User Interface (contd.)

MATLAB as a Calculator

```
>> 2*2 - 4/3

ans =
    2.6667


>> 16^(1/4) + 3*sin( pi/ 4 )

ans =
    4.1213


>> sqrt( 4 ) + exp( j*pi/6 )

ans =
  2.8660 + 0.5000i
```

User Interface (contd.)

Operators and operator precedence

Mathematical operations in MATLAB in order of precedence

1. ( ) parenthesis
2. ´ complex conjugate transpose
3. ˆ power
4. * multiplication; / division; \ left division
5. + addition; – subtraction

```
>> 2*3+2
>> 2*(3+2)
>> 2/3
>> 2\3
>> 2*3^4
>> (2*3)^4
```

User Interface (contd.)

Pre-defined Functions

- Trigonometric – COS, ACOS, EXP, SIN, ASIN
- Exponential – LOG, EXP, SQRT
- Complex – ABS, ANGLE, CONJ
- Discrete Maths – FACTOR, PRIMES, GCD

Want more information? Type

```
>> help elfun
>>
>> doc elfun
>>
>> help angle
```

Outline

1. *MATLAB User Interface Layout*.
2. *Working with Variables*.
3. Visualizing Data.
4. Programming.
5. Case Study.

Working with Variables

- Creating and Manipulating Variables.
- Accessing and Manipulating Elements in a Matrix.
- Computations with Matrices.

Working with Variables

Creating and Manipulating Variables

All variables in MATLAB are arrays (matrices).

- A *scalar* is a 1 × 1 array.

```
>> a = 1
a =
     1
```

- A *(column) vector* is an $n \times 1$ array.

```
>> b = [ 1; 2 ] % or [ 1 2 ]’
b =
     1
     2
```

- A *row vector* is a $1 \times m$ array.

```
>> c = [ 1 2 3 ] % or [ 1, 2, 3 ]
c =
     1     2     3
```

Working with Variables (contd.)

Creating and Manipulating Variables (contd.)

All variables in MATLAB are arrays (matrices).

- A *matrix* is an $n \times m$ array.

```
>> d = [ 1 2; 3 4 ]

d =

     1     2
     3     4
```

- *Strings* are arrays of characters.

```
>> str = ’Hello World!’

str =

Hello World!
```

Working with Variables (contd.)

Creating and Manipulating Variables (contd.)

```
>> x = 1

x =
     1

>> y = 4;
>> r = sqrt( x^2 + y^2 )

r =
    4.1231

>> fx = cos( 2*x + pi/4 )

fx =
   -0.9372
```

Working with Variables (contd.)

Creating and Manipulating Variables (contd.)

```
>> x = [ 1 2 3 4 5 ]
>> x = 1:5
>> x = 0:0.25:1
>> x = linspace( 0, 1, 5 )
>> x = logspace( -1, 2, 4 )
```

Exercises:

- Create the vector $x = \begin{bmatrix} 10 & \pi & \sin(30^\circ) & \sqrt{2} \end{bmatrix}$.
- Create the vector

$$y = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 10 \end{bmatrix}.$$

- Grid the interval from 1 to 5 using 11 points.
- Create a vector w with first element 0, last element 4 & increments of 0.5

Working with Variables (contd.)

Creating and Manipulating Variables (contd.)

Special Matrices

```
>> A = zeros( 3 )

>> B = ones( 2, 4 )

>> C = rand( 1, 4 )

>> D = magic( 4 )

>> E = eye( 2 )

>> X = [ ones( 2 ) zeros( 2, 3 ) rand( 2, 1 ) ]

>> sparse( X )
```

Working with Variables (contd.)

Creating and Manipulating Variables (contd.)

Special Matrices (contd.)

Example

Write MATLAB commands to obtain the following matrices

1. B_1: *A* 3×2 *matrix with all elements equal to 3.*
2. C_1: *A* 2×4 *matrix whose elements are random values between* -1 *and* 3.
3. D_1: *A* 5×5 *magic square. What is the sum of each row?*

Working with Variables (contd.)

Accessing and Manipulating Elements in a Matrix

Array elements are accessed through indices.

- A single matrix element.
- A sub-matrix.
- Re-order elements.

Working with Variables (contd.)

Accessing and Manipulating Elements in a Matrix (contd.)

```
>> A = rand( 4, 5 )

>> A( 2, 3 )

>> A( 2, 3 ) = 5

>> A( 1, : )

>> A( :, 2 )
```

```
>> A( 2:3, 3:4 )

>> A( 2:end, : )

>> A( end:-1:1, : )

>> A( [ 1 3 ], : )

>> A( 3, [ 2 4 ] )

>> A( [ 1 3 ], [ 4 2 ] )
```

Example

- *Create a random* 2×3 *matrix A.*
- *Modify* a_{23} *to* π.
- *Invert the order of the columns of A.*

Working with Variables (contd.)

Computations with Matrices

Two types of computations

Suppose

A_1 and A_2 are two matrices of order $m \times n$,
B is of order $n \times p$,
C is $n \times n$, and
α is a scalar.

- Matrix computations – *Mathematically defined*.

  Examples: $A_1 + A_2$, $A_1 B$, C^{-1}, αA, $\ldots$
- Element-wise computations – *useful for speeding up computations*.

  Examples: $a_{k\ell} b_{k\ell}$, $\ldots$

Working with Variables (contd.)

Computations with Matrices (contd.)

Two types of computations

Example

Create the matrices

- *a random* 2×3 *matrix* A, *and*
- *a random* 3×2 *matrix* B.

Hence, calculate

- $C = AB$,
- $D = C^{-1}$
- $E = \varepsilon D$
- $F = E + 2I$
- H *where* $h_{k\ell} = a^2_{k\ell}$.

Want to know more matrix functions, type

```
>> help matfun
>>
>> doc matfun
```

Outline

1. *MATLAB User Interface Layout*.
2. *Working with Variables*.
3. *Visualizing Data*.
4. Programming.
5. Case Study.

Visualizing Data

- Basic plotting commands.
- Customizing plots.
- 2-D and 3-D plots.
- Importing data into MATLAB.
- Saving and loading data.
- Basic curve fitting.

VISUALIZING DATA (CONTD.)

Basic plotting commands

- Plotting a sinusoidal function $y = \sin(x)$.

```
>> x = 0:0.2:2*pi;
>> y = sin( x );
>> plot( x, y )
```

- Different looks

```
>> plot( x, y, ’r’ )
>> plot( x, y, ’:’ )
>> plot( x, y, ’x’ )
```

- Standard form:
  plot(xdata, ydata, ' <color><linestyle><marker>') .

```
>> plot( x, y, ’g-.o’ )
```

- For more information

```
>> help plot
```

VISUALIZING DATA (CONTD.)

Basic plotting commands (contd.)

- Drawing multiple plots on the same graph:

$$y = \sin(x) \text{ and } z = \cos(x).$$

```
>> x = 0:0.2:2*pi;
>> y = sin( x );
>> z = cos( x );
```

- Does this work?

```
>> plot( x, y )
>> plot( x, z )
```

- What about this?

```
>> plot( x, y, x, z )
>> plot( x, y, ’r:o’, x, z, ’m--s’ )
```

- Or this?

```
>> plot( x, y, ’r:o’ ), hold on
>> plot( x, z, ’m--s’ ), hold off
```

VISUALIZING DATA (CONTD.)

Customizing Plots

```
>> x = 0:0.2:2*pi;
>> y = sin( x );
>> z = cos( x );
>> plot( x, y, ’r:o’, x, z, ’m--s’ )
```

- Adding a grid

```
>> grid
```

- Label the axes

```
>> xlabel(’Angle (rad)’)
>> ylabel(’Amplitude’)
```

- A title

```
>> title(’Plot of sinusoidal functions’)
```

- Legend for multiple graphs

```
>> legend(’sin(x)’, ’cos(x)’)
```

VISUALIZING DATA (CONTD.)

Customizing Plots (contd.)

- Drawing multiple graphs in the same window:

$$y = \sin(x) \text{ and } z = \cos(x).$$

```
>> x = 0:0.2:2*pi;

>> y = sin( x );
>> z = cos( x );

>> subplot(211), plot( x, y, ’r:o’ )
>> subplot(212), plot( x, z, ’m--s’ )
```

EXAMPLE

Plot

$$y = e^{\alpha x}\cos\left(\omega x + \beta\right); \alpha = -0.1, \omega = \frac{\pi}{6}, \beta = \pi.$$

Visualizing Data (contd.)

2−D and 3−D Plots

Types of 2−D Plots:

```
>> x = 0:0.2:2*pi;

>> y = sin( x );

>> plot( x, y, 'r:o' )

>> stem( x, y )

>> bar( x, y )

>> stairs( x, y )

>> area( x, y )
```

Visualizing Data (contd.)

2−D and 3−D Plots (contd.)

Using the *plot tool* in *workspace* window.

In the workspace window,

- select the data to be plotted,
- click Plots tab,
- choose desired plot type.

Visualizing Data (contd.)

2−D and 3−D Plots (contd.)

Exporting to other applications.

In Figure window,

- choose *File* → *Save As . . .*, and
- select desired file type,

or

- choose *Edit* → *Copy Figure*, then
- paste in desired file.

Visualizing Data (contd.)

2−D and 3−D Plots (contd.)

Types of 3-D Plots

```
>> X = membrane;

>> surf( X )

>> imagesc( X )

>> contour( X )

>> mesh( X )
```

```
>> surfl( X )

>> waterfall( X )

>> ribbon( X )

>> spy( X )

>> help spy
```

Visualizing Data (contd.)

Importing Data into MATLAB

- Using the *Import Wizard*.
  - Click *Import Data* .
  - Choose *studmarks.txt*.
- Using MATLAB commands – importdata.

```
>> mydata = importdata( 'studmarks.txt' );
```

- For more info, type

```
>> help importdata
>>
>> doc importdata
```

- Alternate commands xlsread, csvread, dlmread.

Visualizing Data (contd.)

Saving and Loading Data

Saving Data.

```
>> x = 0:0.2:2*pi;
>> y = sin( x );
>> z = cos( x );
>> save

Saving to: matlab.mat

>> save mydata
>> save mydata2 x y
>> save mydat3 x y -ascii
```

Loading Data.

```
>> clear
>> load mydata2
>>
>> clear
>> load mydata
```

Visualizing Data (contd.)

Basic Curve Fitting

- In Figure window, choose *Tools* → *Basic Fitting*
- Choose type of curve desired
- Choose other parameters
- Analyze
- Explore

Outline

1. *MATLAB User Interface Layout*.
2. *Working with Variables*.
3. *Visualizing Data*.
4. *Programming*.
5. Case Study.

PROGRAMMING

- The MATLAB editor.
- Script m-files.
- Function m-files.

PROGRAMMING (CONTD.)

The MATLAB Editor

- For writing MATLAB programs.
- Works like any normal text editor.
- Two types of programs
  - Script m-files.
  - Function m-files.
- To open the editor,
  - Choose *Home* tab,
  - Click *New Script* or *New* → *Script*.

PROGRAMMING (CONTD.)

Script (m-)files Type the following in the editor.

```
%% This is my first MATLAB program
% First clear the mess
close all % Closes all figure windows
clear % clear the workspace
clc % clear the command window

%% Display message
disp('Now I can write script m-files!!!')

%% Determine the variables
x = 0:0.2:2*pi; y = sin( x ); z = cos( x );

%% Plot the figures
plot( x, y, 'r:o', x, z, 'm--s' ), grid
xlabel('x values'), ylabel('y values')
title('Plot of sinusoidal functions')
legend('sin(x)', 'cos(x)')
```

PROGRAMMING (CONTD.)

Script (m-)files (contd.)

- To save the program (use *myfirst*.m as the name).
  - click on *Save*, or
  - type *Ctrl-s*.
- To run the program
  - Type the file name (*myfirst*) in the command window, or
  - click the *Run myfirst.m* icon.
- For clarity of the program, add the following (comments) at the top.

```
%% MYFIRST
% This program plots the sinusoidal functions sin(x)
% and cos(x) for x = 0 to pi in steps of 0.2 radians.
%
% Written by ...
% June 22, 2011
% Last modified ...
%
```

PROGRAMMING (CONTD.)

Script (m-)files (contd.)

- To publish a MATLAB program, use *Cells* in the Editor.
  - To define a Cell, use % %.
- To publish the file, click *PUBLISH* → *Publish*.
- By default, html files are produced.
- To publish to other file types,
  - click *Publish* → *Edit Publishing Options*.
  - In the dialog box, make changes as desired.

EXAMPLE

Write a MATLAB script file to calculate and plot

$$y = e^{\alpha x} \sin\left(\frac{\pi}{3} x\right)$$

for $\alpha = 0.1, 0.2, 0.5, 1$. *Choose range of* x *to display at least two cycles. Finally publish a report.*

PROGRAMMING (CONTD.)

Function (m-)files

- Type the following in the editor,

```
function y = mymean( x )
% Calculates the average of the numbers in a vector
[ m, n ] = size( x ); % No of rows & columns in x.
if m==1 | n==1
    k = max( m, n ); % Number of elements
    y = sum( x )/k;
else
    disp('x must be a vector.')
end
```

- The first line of a function must be

  function [outputs] = function_name(inputs)

  .
- Save the file as *mymean*.m

PROGRAMMING (CONTD.)

Function (m-)files (contd.)

- Type the following in the editor,

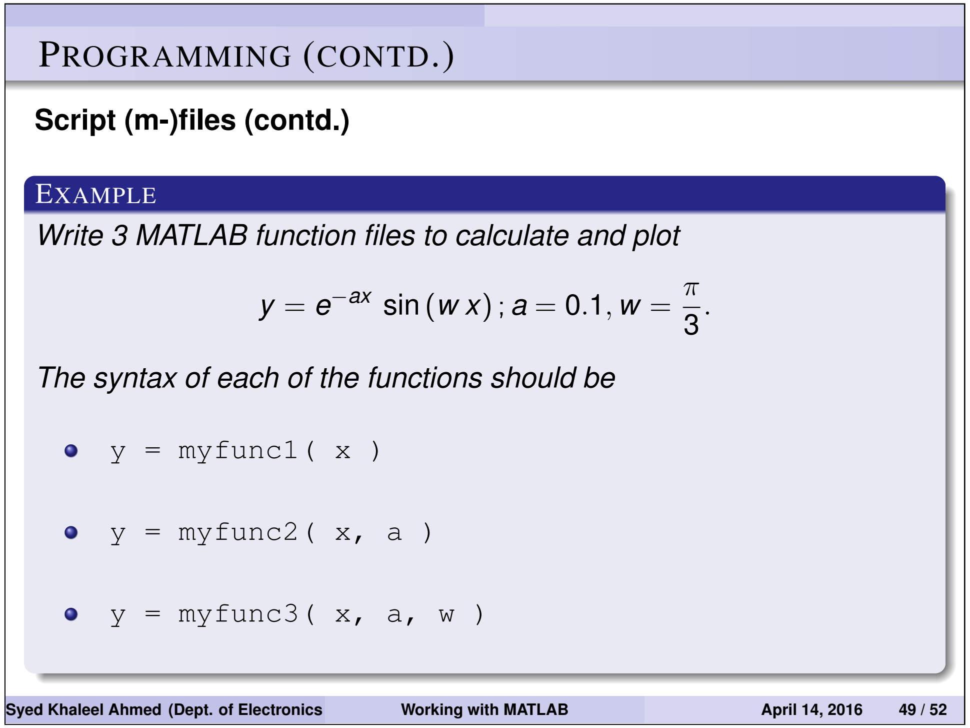
```
function [ yavg ymin ymax ] = mymean2( x )
% This program calculates the average, minimum, and
% maximum of the numbers in a vector.
%
[ m, n ] = size( x ); % No of rows & columns in x.
if m==1 | n==1
    k = max( m, n ); yavg = sum( x )/k;
    [ ymin ymax ] = myminmax( x );
else
    disp('x must be a vector.')
end

function [ ymin ymax ] = myminmax( x )  % Subfunction
w = sort( x );
ymin = w( 1 ); ymax = w( end );
end
```

PROGRAMMING (CONTD.)

Function (m-)files (contd.)

- To execute the program, type

```
>> x = rand( 1, 4 )

>> mymean( x )

>> y = rand( 4, 1 )

>> mymean( y )

>> z = rand( 2, 4 )

>> mymean( z )
```



PROGRAMMING (CONTD.)

Script (m-)files (contd.)

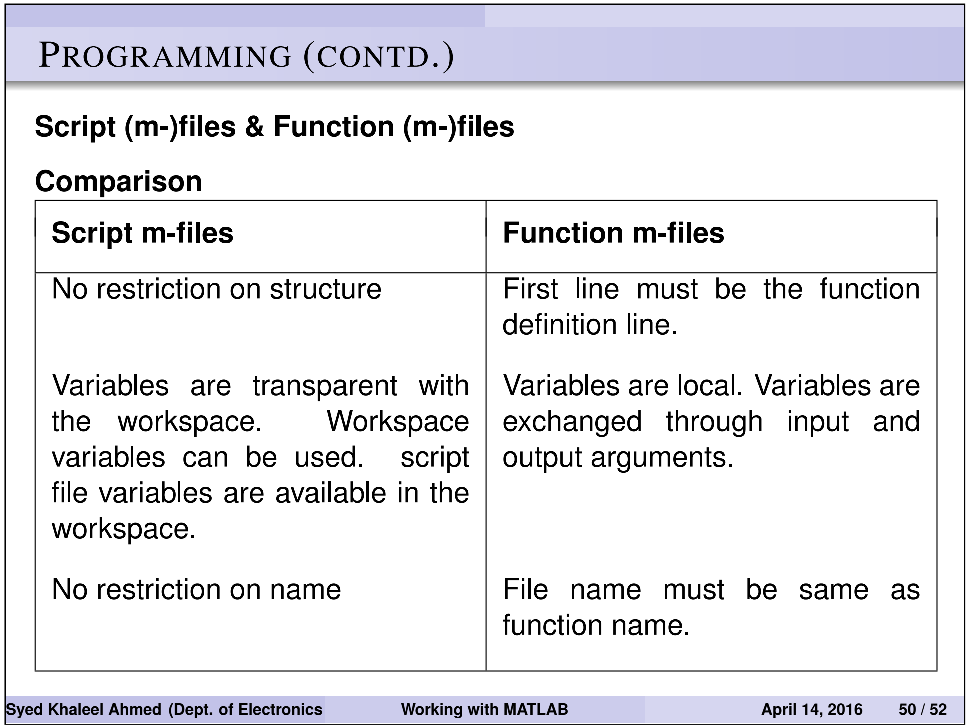
EXAMPLE

Write 3 MATLAB function files to calculate and plot

$$y = e^{-ax}\,\sin(w\,x)\,;\, a = 0.1,\, w = \frac{\pi}{3}.$$

The syntax of each of the functions should be

- `y = myfunc1( x )`
- `y = myfunc2( x, a )`
- `y = myfunc3( x, a, w )`

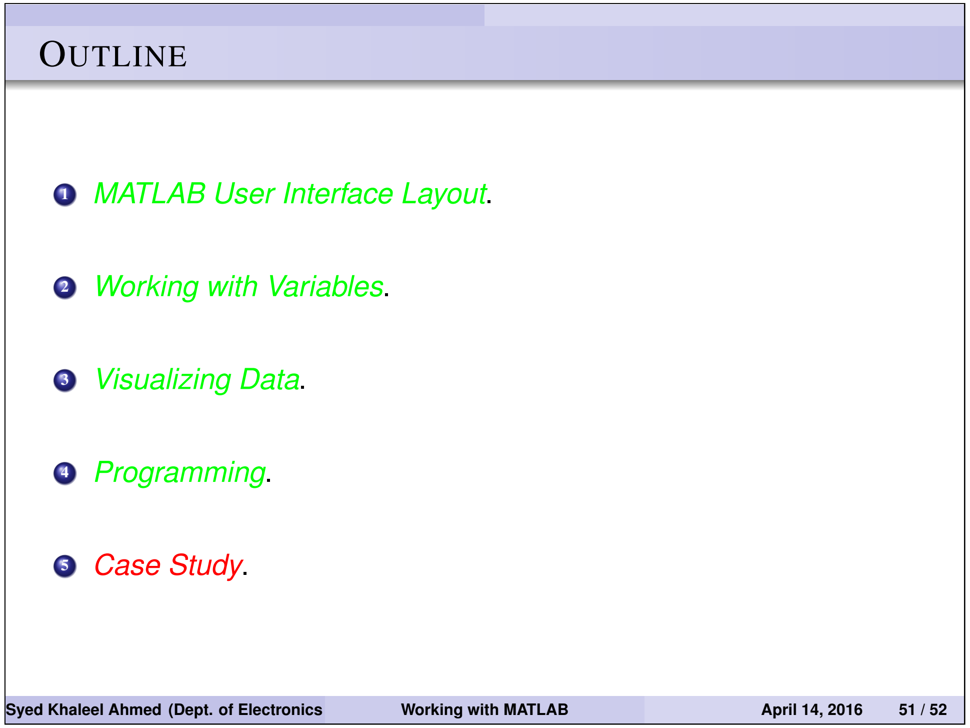


PROGRAMMING (CONTD.)

Script (m-)files & Function (m-)files

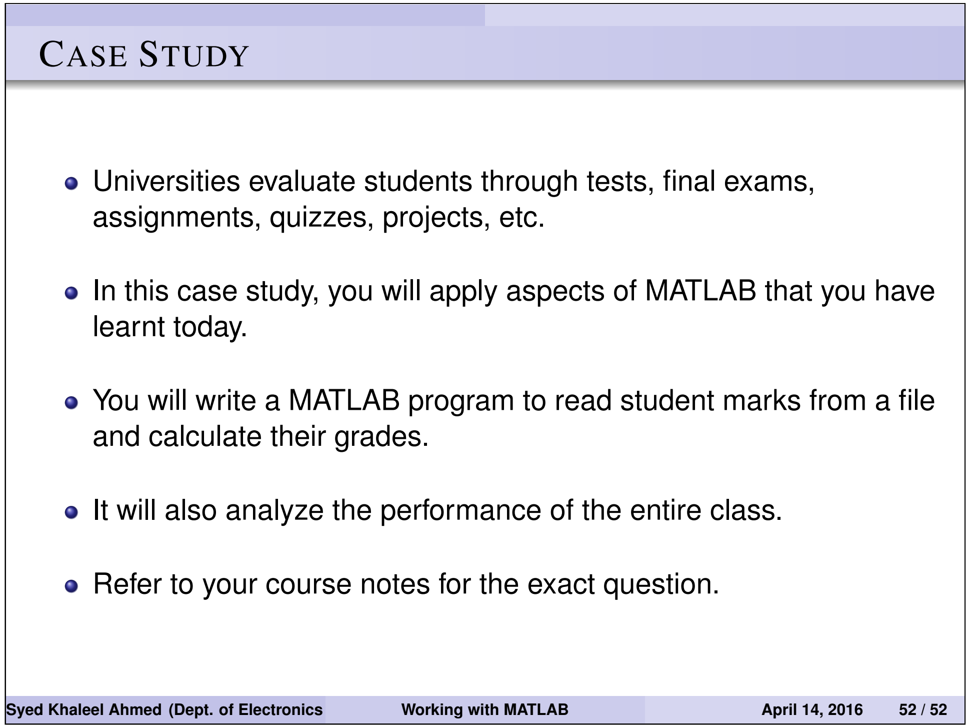
Comparison

Script m-files	**Function m-files**
No restriction on structure	First line must be the function definition line.
Variables are transparent with the workspace. Workspace variables can be used. script file variables are available in the workspace.	Variables are local. Variables are exchanged through input and output arguments.
No restriction on name	File name must be same as function name.



OUTLINE

1. *MATLAB User Interface Layout*.
2. *Working with Variables*.
3. *Visualizing Data*.
4. *Programming*.
5. *Case Study*.



CASE STUDY

- Universities evaluate students through tests, final exams, assignments, quizzes, projects, etc.
- In this case study, you will apply aspects of MATLAB that you have learnt today.
- You will write a MATLAB program to read student marks from a file and calculate their grades.
- It will also analyze the performance of the entire class.
- Refer to your course notes for the exact question.