ENGINEERING MATHEMATICS-I
MATLAB

Department of Science and Humanities

Introduction to MATLAB

- MATLAB is a programming language developed by MathWorks.
- MATLAB stands for **MAT**rix **LAB**oratory.
- MATLAB is a program for doing numerical computation. It was originally designed for solving linear algebra type problems using matrices.
- While other programming languages mostly work with numbers one at a time, MATLAB is designed to operate primarily on whole matrices and arrays.

Introduction to MATLAB, Continued…

- Using MATLAB, an image (or any other data like sound, etc.) can be converted to a matrix and then various operations can be performed on it to get the desired results and values.
- MATLAB is a fourth-generation high-level programming language and interactive environment for numerical computation, visualization and programming.
- It has numerous built-in commands and math functions that help in mathematical calculations, generating plots, and performing numerical methods.

MATLAB's Power of Computational Mathematics

- MATLAB is used in every fact of computational mathematics. Following are some commonly used mathematical calculations where MATLAB is used:

- Dealing with Matrices and Arrays
- 2-D and 3-D Plotting and graphics
- Linear Algebra
- Algebraic Equations
- Statistics

MATLAB's Power of Computational Mathematics, Continued…

- Data Analysis
- Calculus and Differential Equations
- Numerical Calculations
- Integration
- Transforms
- Curve Fitting
- Special Functions

Uses of MATLAB

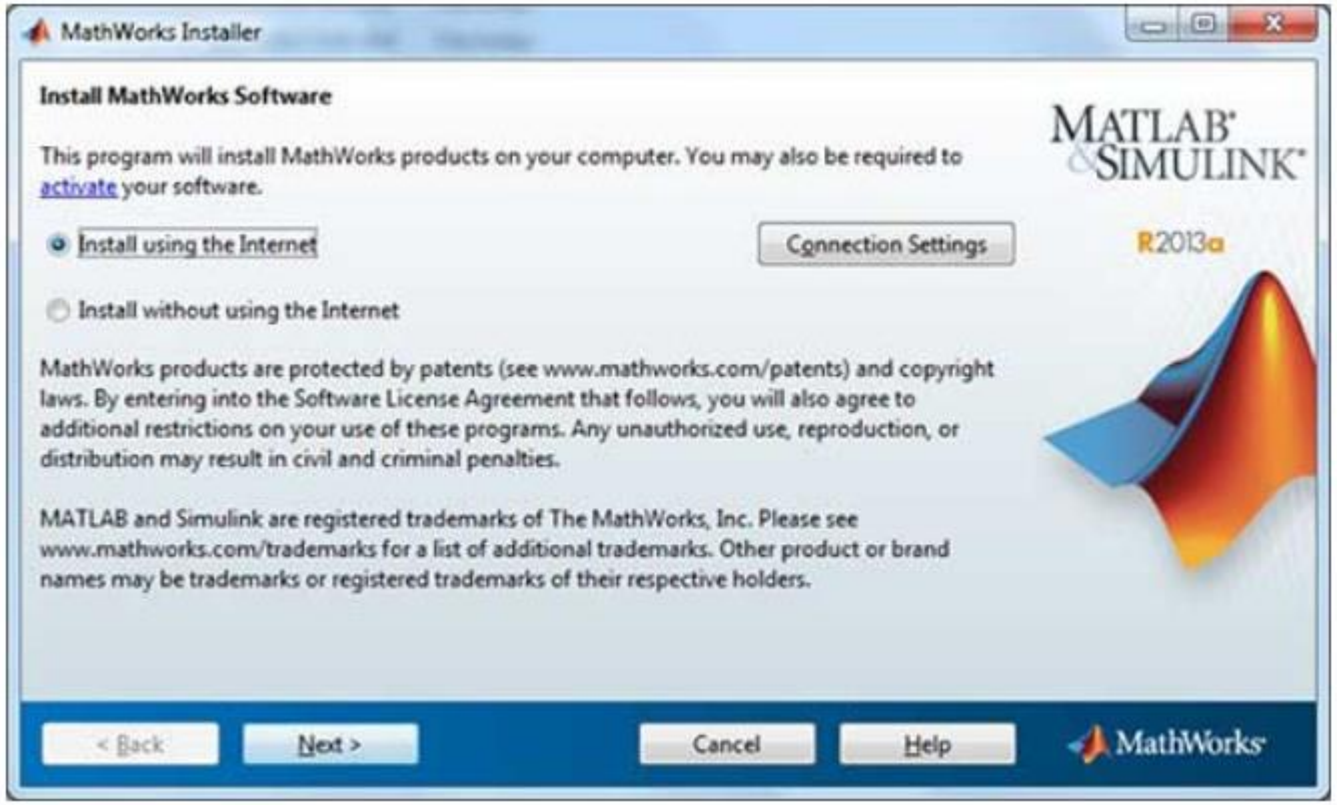
- MATLAB is widely used as a computational tool in Science and Engineering covering the fields of Physics, Chemistry, Mathematics, and all engineering streams.
- It is used in a range of applications including:

- Signal Processing and Communications
- Algorithm development
- Control Systems
- Computational Finance; Computational Biology

Local Environment Setup

- Setting up MATLAB environment is a matter of few clicks.
- MathWorks provides the licensed product, a trial version, and a student version as well.
- We need to log into the site and wait a little for their approval.
- After downloading the installer, the software can be installed through few clicks.

Local Environment Setup, Continued...


MathWorks Installer

Install MathWorks Software

This program will install MathWorks products on your computer. You may also be required to activate your software.

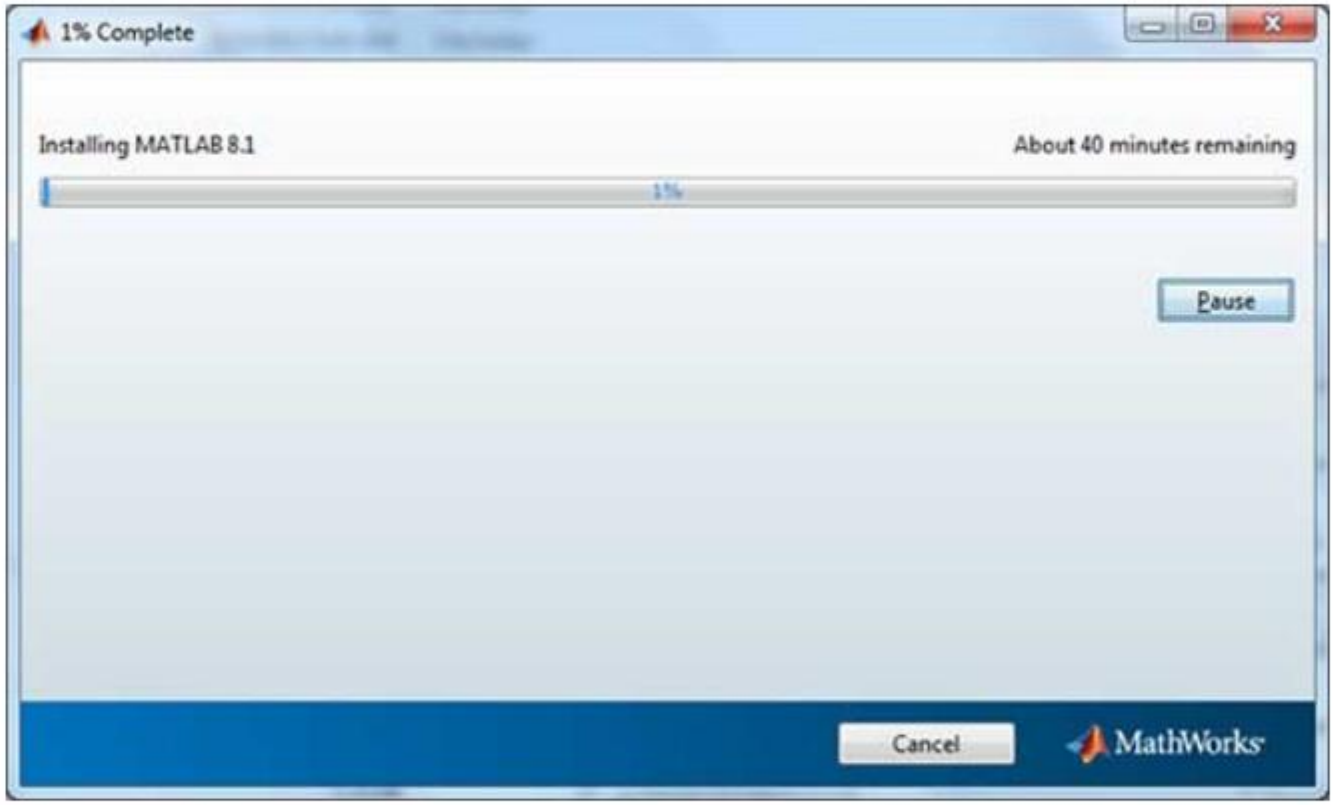
[✓ Install using the Internet]

Install without using the Internet

MathWorks products are protected by patents (see www.mathworks.com/patents) and copyright laws. By entering into the Software License Agreement that follows, you will also agree to additional restrictions on your use of these programs. Any unauthorized use, reproduction, or distribution may result in civil and criminal penalties.

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. Please see www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Local Environment Setup, Continued...

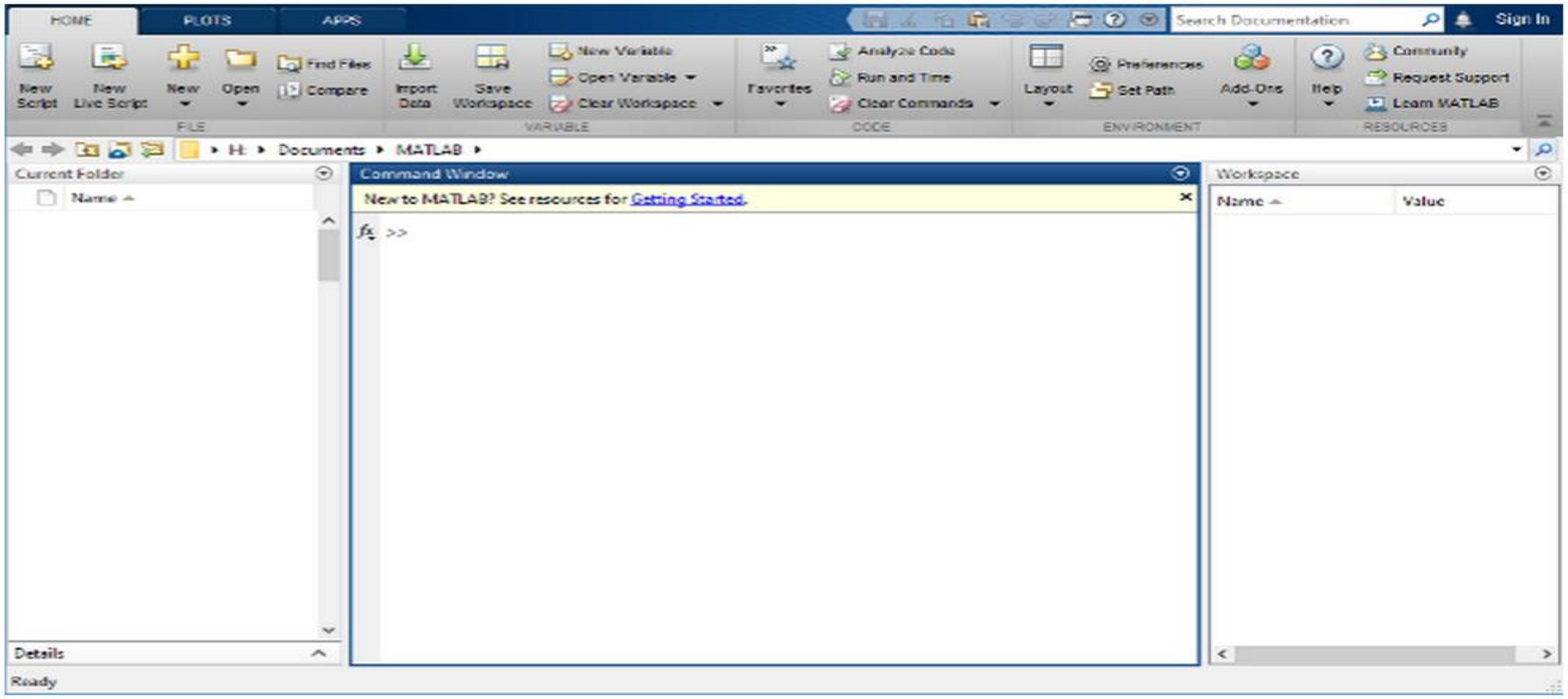

1% Complete

Installing MATLAB 8.1

About 40 minutes remaining

Understanding the MATLAB Environment

When you start MATLAB®, the desktop appears in its default layout.



New to MATLAB? See resources for Getting Started.

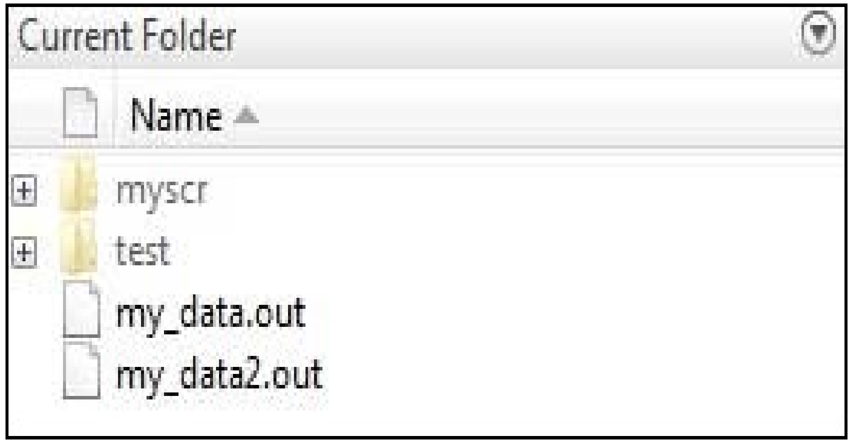
```
fx >>
```

The desktop includes these panels:

- **Current Folder** — Access your files.
- **Command Window** — Enter commands at the command line, indicated by the prompt (>>).
- **Workspace** — Explore data that you create or import from files.

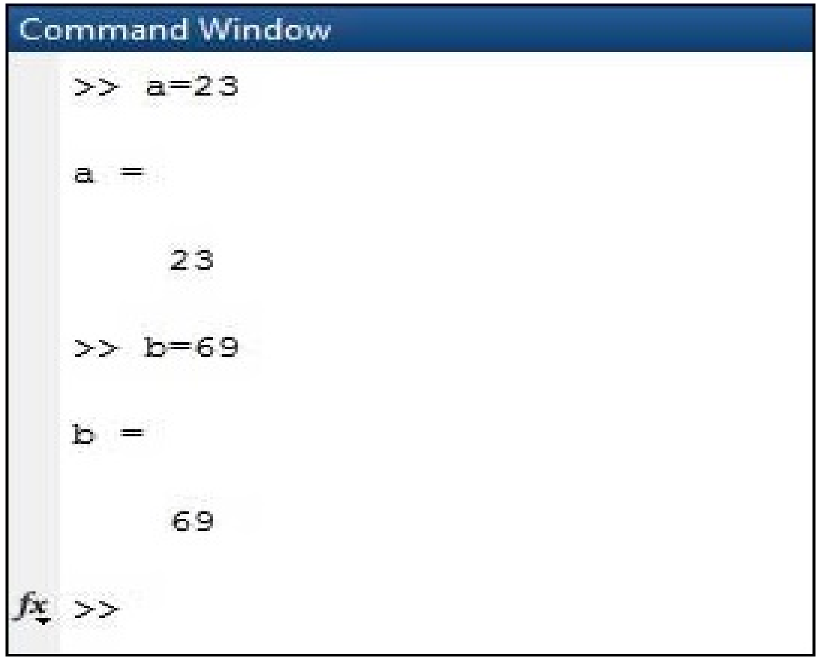
Understanding the MATLAB Environment, Continued…

- The desktop has the following panels:
- **Current Folder:** This panel allows us to access the project folders and files.



Understanding the MATLAB Environment, Continued…

- **Command Window:** This is the main area where commands can be entered at the command line. It is indicated by the command prompt (>>).

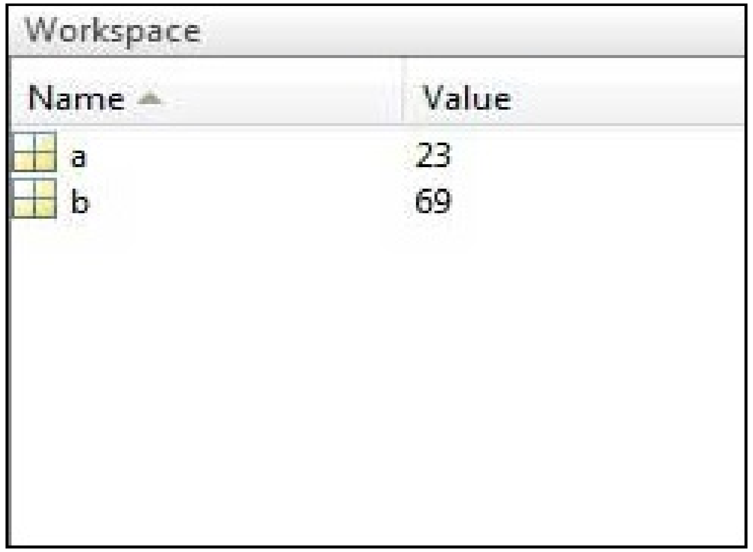


Command Window

```
>> a=23

a =

    23

>> b=69

b =

    69

>>
```

Understanding the MATLAB Environment, Continued…

- **Workspace:** The workspace shows all the variables created and/or imported from files.

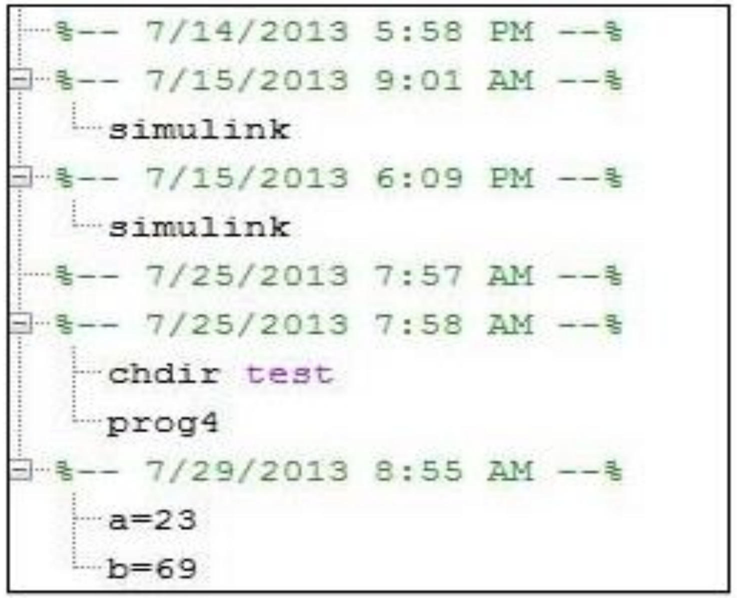


Workspace

Name	Value
a	23
b	69

Understanding the MATLAB Environment, Continued…

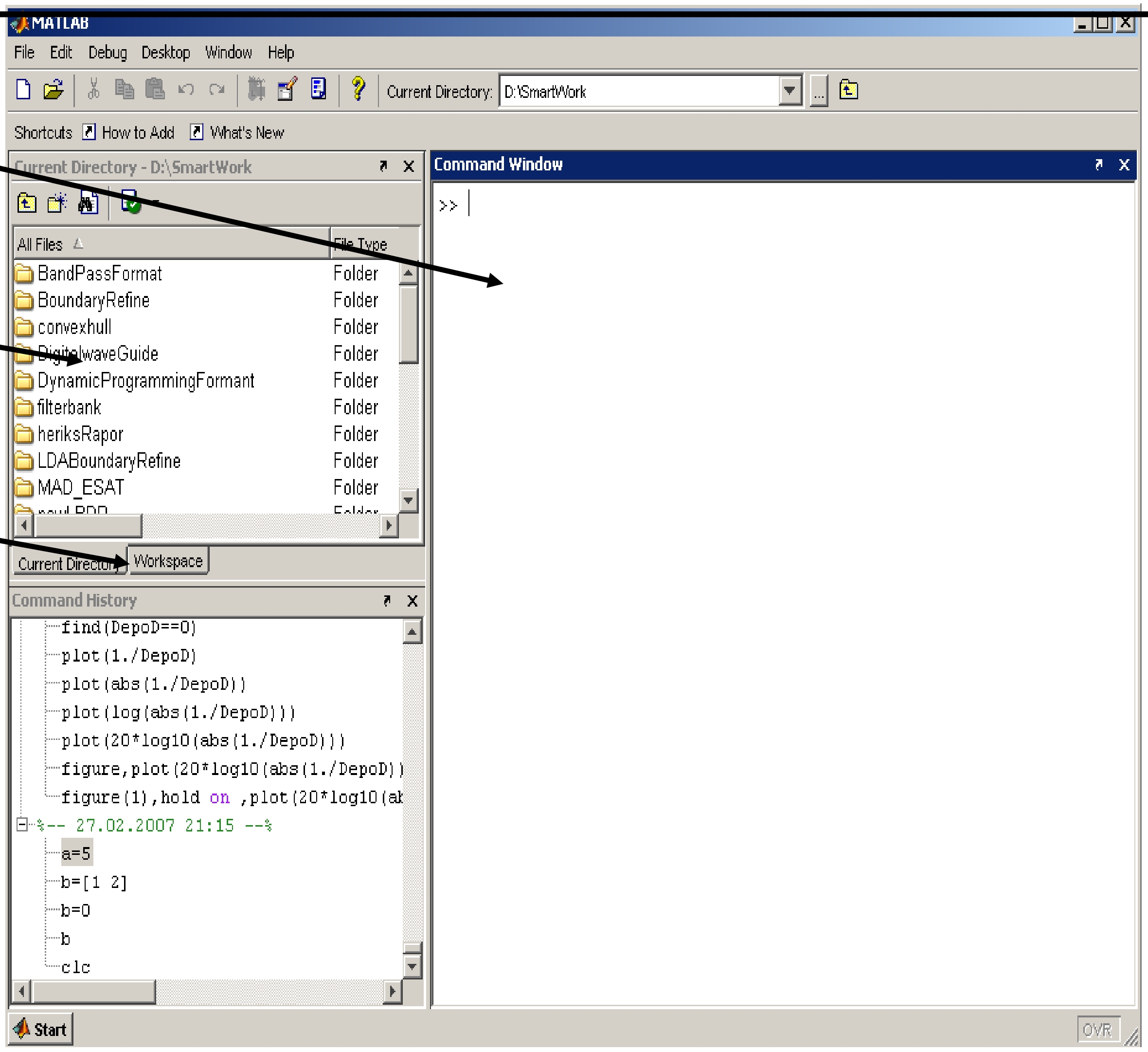
- **Command History:** This panel shows or return commands that are entered at the command line.



```
%-- 7/14/2013 5:58 PM --%
%-- 7/15/2013 9:01 AM --%
    simulink
%-- 7/15/2013 6:09 PM --%
    simulink
%-- 7/25/2013 7:57 AM --%
%-- 7/25/2013 7:58 AM --%
    chdir test
    prog4
%-- 7/29/2013 8:55 AM --%
    a=23
    b=69
```

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


MATLAB
File Edit Debug Desktop Window Help
Current Directory: D:\SmartWork
Shortcuts How to Add What's New
Current Directory - D:\SmartWork
Command Window
All Files
File Type
Current Directory
Workspace
Command History
Start
OVR

```
>> |
```

All Files	File Type
BandPassFormat	Folder
BoundaryRefine	Folder
convexhull	Folder
DigitalwaveGuide	Folder
DynamicProgrammingFormant	Folder
filterbank	Folder
heriksRapor	Folder
LDABoundaryRefine	Folder
MAD_ESAT	Folder

```
find(DepoD==0)
plot(1./DepoD)
plot(abs(1./DepoD))
plot(log(abs(1./DepoD)))
plot(20*log10(abs(1./DepoD)))
figure,plot(20*log10(abs(1./DepoD))
figure(1),hold on ,plot(20*log10(ak
%-- 27.02.2007 21:15 --%
a=5
b=[1 2]
b=0
b
clc
```

Hands on Practice

- MATLAB environment behaves like a super-complex calculator. We can enter commands at the >> command prompt.
- In MATLAB, we give a command and it executes the command right away.
- Type a valid expression, for example, >>20 + 21; and press ENTER.
- When we click the Execute button, MATLAB executes it immediately and the result returned is ans=41.

Hands on Practice, Continued…

- Let us take up few more examples:
- >>3 ^ 2 (%3 raised to the power of 2). When we click the Execute button, MATLAB executes it immediately and the result returned is ans=9.
- >>sin(pi/2). (% sine of angle 90). When we click the Execute button, MATLAB executes it immediately and the result returned is ans=1.
- >>7/0 (% Divide by zero). When we click the Execute button, MATLAB executes it immediately and the result returned is ans=Inf.

Hands on Practice, Continued…

- >>732 * 20.3. When we click the Execute button, MATLAB executes it immediately and the result returned is ans=1.4860e+04.
- MATLAB provides some special expressions for some mathematical symbols, like pi for π, Inf for ∞, i (and j) for $\sqrt{-1}$.
- In MATLAB, **NaN** stands for 'not a number'.

  For example, >> 0/0; -inf/inf

Use of Semicolon (;) in MATLAB

- Semicolon (;) indicates end of statement. However, if we want to suppress and hide the MATLAB output for an expression, add a semicolon after the expression.
- For example, >>x = 5; y = x + 5. When we click the Execute button, MATLAB executes it immediately and the result returned is y=8.

Adding Comments in MATLAB

- To add comments to MATLAB code, we use the percent (%) symbol. Comment lines can appear anywhere in a program file, and we can add comments to the end of a line of code.

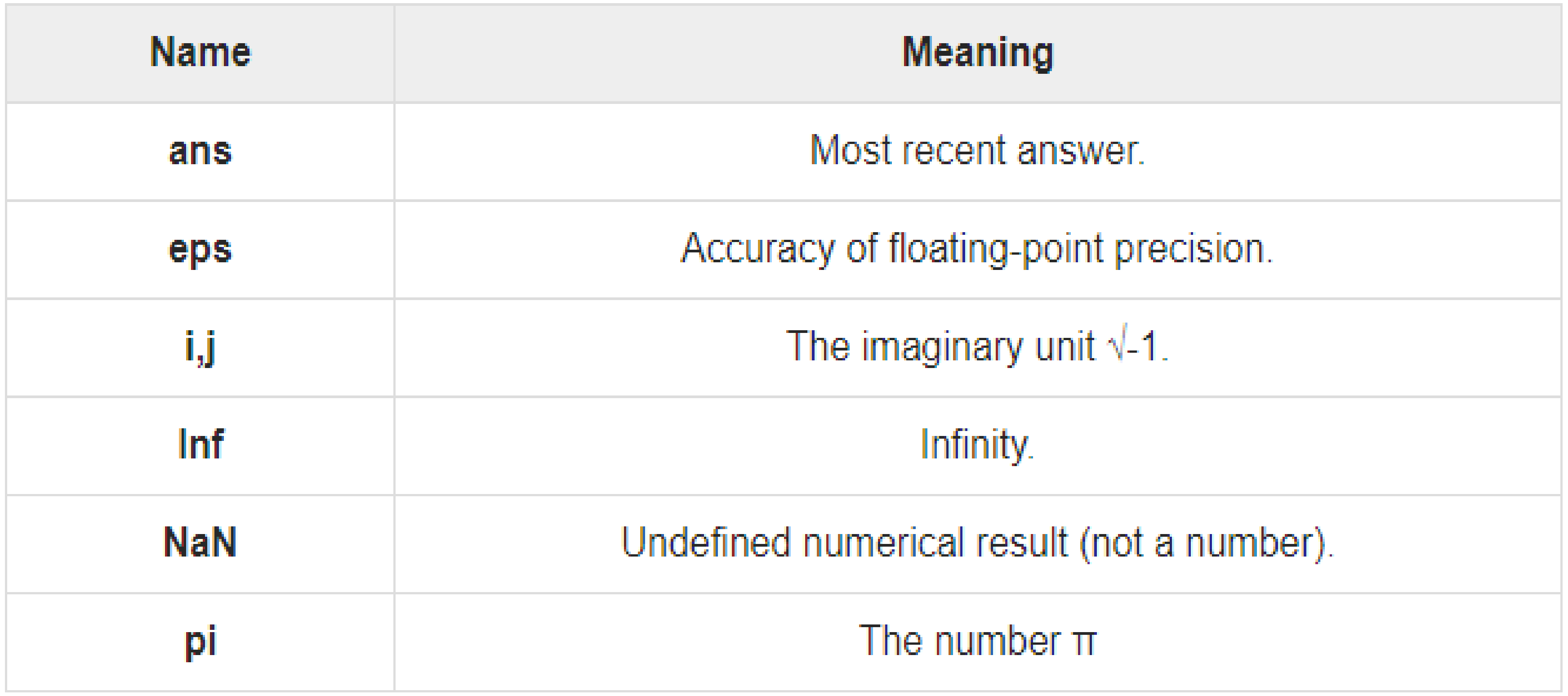
  For example, y = sum(x)   % Use the sum function

Commonly used Operators and Special Characters

Operation	Symbol	Example
Addition	+	5+3=8
Subtraction	-	5-3=2
Multiplication	*	5*3=15
Right Division	/	5/3=1.6667
Left Division	\	5/3=3\5=1.667
Exponentiation	^	5^3=125

Special Variables and Constants

MATLAB supports the following special variables and constants:



Name	Meaning
ans	Most recent answer.
eps	Accuracy of floating-point precision.
i,j	The imaginary unit √-1.
Inf	Infinity.
NaN	Undefined numerical result (not a number).
pi	The number π

Naming Variables

- Variable names consist of a letter followed by any number of letters, digits or underscore.
- MATLAB is **case-sensitive**.
- For example, >>x = 10 (% defining x and initializing it with a value). MATLAB will execute the above statement and return the following result x = 10.
- >>x = sqrt(25) (% defining x and initializing it with an expression) MATLAB will execute the above statement and return the result as x = 5.

Naming Variables, Continued…

- Note that once a variable is entered into the system, we can refer to it later. Variables must have values before they are used. When an expression returns a result that is not assigned to any variable, the system assigns it to a variable named answer, which can be used later.
- For example, >>x = 7 * 8; y = x * 7.89. MATLAB will execute the above statement and return the following result y = 441.8400.

Multiple assignments

- We can have multiple assignments on the same line.
- For example, >> a = 2; b = 7; c = a * b
- MATLAB will execute the above statement and return the result c = 14.

I have forgotten the Variables

- The who command displays all the variable names we have used.
- MATLAB will execute the above statement and return the result: Your variables are: ans - - - …
- The whos command. MATLAB will execute the above statement and return the following result.

```
Name      Size           Bytes  Class     Attributes
 ans      1x1               8    double
  x        1x1               8    double
```

I have forgotten the Variables

>> clear x. It will delete x, won't display anything.

>> clear. It will delete all variables in the workspace.

>> clc. It clears all the text from the Command Window, resulting in a clear screen.

Long Assignments

- Long assignments can be extended to another line as follows:

```
>> initial_velocity = 0;
acceleration = 9.8;
time = 20;
>> Final_velocity = initial_velocity + acceleration * time
```

- MATLAB will execute the above statement and return the following result: final velocity = 196.

The format Command

- By default, MATLAB displays numbers with four decimal place values. This is known as **short format**.
- However, if we want more precision, then we need to use the **format** command.
- The **format long** command displays 15 digits after the decimal point.
- For example: >> format long

  >> x = 7 + 10/3 + 5 ^ 1.2
- MATLAB will execute the above statement and return the following result: x = 17.231981640639408.

The format Command, Continued…

Another example,

```
>> format short
>> x = 7 + 10/3 + 5 ^ 1.2
```

- MATLAB will execute the above statement and return the following result: x = 17.232
- The **format bank** command rounds numbers to two decimal places.

The format Command, Continued…

For example,

```
>> format bank
>> daily_wage = 177.45; weekly_wage = daily_wage * 6
```

- MATLAB will execute the above statement and return the following result: weekly_wage = 1064.70
- MATLAB displays large numbers using exponential notation.
- The **format short e** command allows displaying in exponential form with four decimal places plus the exponent.

The format Command, Continued…

- For example,

```
>> format short e
4.678 * 4.9
```

- MATLAB will execute the above statement and return the following result: ans = 2.2922e+01
- The **format long e** command allows displaying in exponential form with four decimal places plus the exponent.

The format Command, Continued…

- For example, >> format long e
  >> x = pi
- MATLAB will execute the above statement and return the following result: x = 3.141592653589793e+00
- The **format rat** command gives the closest rational expression resulting from a calculation.
- For example, >> format rat
  4.678 * 4.9
- MATLAB will execute the above statement and return the following result: ans = 34177/1491

Creating Vectors

- A vector is a one-dimensional array of numbers.
- MATLAB allows creating two types of vectors:
- Row vectors and Column vectors.
- **Row vectors** are created by enclosing the set of elements in square brackets, using space or comma.
- For example, >> X = [7 8 9 10 11]. MATLAB will execute the above statement and return the following result:

```
X =

    7         8         9         10         11
```

Creating Vectors, Continued…

- Another example,

  ```
  >> X = [7 8 9 10 11]; >> Y = [2, 3, 4, 5, 6]; >> Z = X +Y
  ```

- MATLAB will execute the above statement and return the following result:

  ```
  Z =

    9        11        13        15        17
  ```

Creating Vectors, Continued…

- **Column vectors** are created by enclosing the set of elements in square brackets, using semicolon(;).
- For example, >> X = [7; 8; 9; 10]
- MATLAB will execute the above statement and return the following result:

```
X =

     7

     8

     9

    10
```

Creating matrices

- A matrix is a two-dimensional array of numbers.
- In MATLAB, a matrix is created by entering each row as a sequence of space or comma separated elements, and end of a row is terminated by a semicolon.
- For example, let us create a 3-by-3 matrix as

```
>>A = [1 2 3 4; 4 5 6 7; 7 8 9 10]
```

Creating matrices, Continued…

- MATLAB will execute the above statement and return the following result:

```
A =

     1          2          3          4

     4          5          6          7

    7          8          9         10
```

Elementary Math Built – In Functions

Function	Description	Examples

sqrt(x)	Square root	>>sqrt(81)
		ans=9

nthroot(x,n)	Real nth root of a real number x. (If x is negative n must be an odd integer).	>>nthroot(8,3)
		ans=2

exp(x)	Exponential(e^x)	>>exp(5)
		ans=1.484131591025766e+02

abs(x)	Absolute value	>>abs(-24)
		ans=24

log(x)	Natural logarithm. Base e logarithm (ln)	>>log(1000)
		ans=6.907755278982137e+00

log10(x)	Base 10 logarithm	>>log10(1000)
		ans=3

factorial(x)	The factorial function x!
	(x must be a psitive integer)
		>>factorial(5)
		ans=120

Trigonometric Math Functions

Function	Description	Examples	

sin(x)
sind(x)
	Sine of angle x (x in radians)
	Sine of angle x (x in degrees)
		>>sin(pi/6)
		>>sind(30)
			ans=0.5000
			ans=0.5000

cos(x)
cosd(x)
	Cosine of angle x (x in radians)
	Cosine of angle x (x in degrees)
		>>cos(0.5)
		>>cosd(30)
			ans=0.8776
			ans=0.8660

tan(x)
tand(x)
	Tangent of angle x (x in radians)
	Tangent of angle x (x in degrees)
		>>tan(pi/6)
		>>tand(45)
			ans=0.5774
			ans=1

cotx)
cotd(x)
	Cotangent of angle x (x in radians)
	Cotangent of angle x (x in degrees)
		>>cot(0.5)
		>>cotd(90)
			ans=1.8305
			ans=1

asin(x)
asind(x)
	Inverse of sine of angle x (x in radians)
	Inverse of sine angle x (x in degrees)
		>>asin(0.5)
		>>asin(0.8660)
			ans=0.5236
			ans=59.9971

acos(x)
acosd(x)
	Inverse of cosine of angle x (x in radians)
	Inverse of cosine angle x (x in degrees)
		>>acos(0.3)
		>>acosd(0.5)
			ans=1.2661
			ans=60

Rounding Functions

Function	Description	Examples
round(x)	Round to the nearest integer	>>round(25/3) ans=8 >>round(4.49) ans=4 >>round(4.5) ans=5 >>round(4.9999) ans=5
fix(x)	Round towards zero. In other words, chops off the fraction part.	>>fix(17/5) ans=3 >>fix(4.49) ans=4 >>fix(4.9999) ans=4 >>fix(-4.6674) ans=-4
ceil(x)	Round towards positive infinity.	>>ceil(27/2) ans=14 >>ceil(2.1) ans=3 >>ceil(2.9) ans=3 >>ceil(-4.99) ans=-4
floor(x)	Round towards minus infinity	>>floor(-9/4) ans=-3 >>floor(2.1) ans=2 >>floor(2.99) ans=2
rem(x,y)	Returns the remainder after x is divided by y	>>rem(13,5) ans=3

Examples for Arithmetic operators

1) Calculate the following using MATLAB.

$$\frac{1}{2+3^2}+\frac{4}{5}*\frac{6}{7}$$

```
>>a=1/(2+3^2)
a=
    0.0909
>>b=4/5
b=
     0.8000
>>c=6/7
c=
    0.8571
>>a+b*c
ans=
       0.7766
```

2. Find the value of $y = e^{-a}\sin(\mathrm{x}) + 10\sqrt{y}\ if\ a = 5, x = 2, y = 8.$

```
>>a=5;  x=2;   y=8;
y=exp(-a)*sin(x)+10*sqrt(y)
y=
     2.829039804533026e+013.
```

3. Compute sin$\left(\frac{\pi}{4}\right), e^{10}$.

```
>>sin(pi/4)
ans=0.7071
>>exp(10)
ans=2.2026e+004
```

Reference:

1. Getting started with MATLAB, Rudra Pratap, Oxford University Press, 7th Edition, 2016.
2. https://www.tutorialspoint.com/matlab/matlab_data_import.htm

PES UNIVERSITY

THANK YOU

ENGINEERING MATHEMATICS-I MATLAB

Department of Science and Humanities

Data Types

- Data types are those which define the type of data that we are using.
- Some common data types are:
  - Integers
  - Floating point numbers
  - Scalar
  - Character
  - Strings
  - Arrays

Integers

- An integer is a whole number (not a fraction) that can be positive, negative, or zero.
- Integers are a commonly used data type in computer programming.
- For example, the numbers 10, 0, and -25 are integers.
- When two integers are added, subtracted, or multiplied, the result is also an integer.

Integers, Continued…

For Example:

```
>> 2+3

ans =

       5

>> 4-5

ans =

     -1
```

Integers, Continued…

```
>> 2*8
ans =
      16
```

Note that when one integer is divided by another integer, the result may be an integer or a fraction.

For example:

```
>> 6/4
ans =
     3/2
```

Integers, Continued…

```
>> 6/3

ans =

      2
```

Floating Point Numbers

- As the name indicates, floating point numbers are numbers that contain floating decimal points.
- For example, the numbers 5.5, 0.001, and -2,345.6789 are floating point numbers.
- When a calculation includes a floating point number, it is called a "floating point calculation."

Scalar

- Any number which is used to represent a quantity.
- This includes integers, complex numbers , floating point numbers.
- Examples of scalar data types are: 3, 4+6i, -20.45.

Character

- Single alphanumeric symbol enclosed in a single quote is a character constant.
- Example, ‘B’ and ‘6’.
- 6 and ‘6’ are different. Here, 6 is a character constant and ‘6’ is a character constant.

Strings

- Any two or more alphanumeric symbols enclosed in a single quote.
- Example, ‘INDIA’=[‘I’, ‘N’, ‘D’, ‘I’, ‘A’]

Arrays

- List of similar data in a single row or a column.
- Elements can be numerical or character or strings.
- Examples, [1 2 3 4]; [a b c d].

Special Types of Arrays

- The four types of arrays are:
- zeros() function
- eye() function
- ones() function
- rand() function

Special Types of Arrays, Continued…

- **Zeros() Function :** It creates an array of all zeros.
- For example: >> zeros(5)
- MATLAB will execute the above statement and return the result:

```
ans =                                      .

    0    0    0    0    0

    0    0    0    0    0

    0    0    0    0    0

    0    0    0    0    0

    0    0    0    0    0
```

Special Types of Arrays, Continued…

- eye() function: It creates an identity matrix.
- For example: >> eye(4)
- MATLAB will execute the above statement and return the result:

```
ans =

    1    0    0    0

    0    1    0    0

    0    0    1    0

    0    0    0    1
```

Special Types of Arrays, Continued…

- **ones() function:** It creates an array of all ones.
- For example: >> ones(4,3)
- MATLAB will execute the above statement and return the result:

```
ans =

   1   1   1

   1   1   1

   1   1   1

   1   1   1
```

Special Types of Arrays, Continued…

- rand() function: It creates an array of uniformly distributed random numbers on (0,1).
- For example: >> rand(3, 5)
- MATLAB will execute the above statement and return the result

```
ans =
  0.8147   0.9134   0.2785   0.9649   0.9572
  0.9058   0.6324   0.5469   0.1576   0.4854
  0.1270   0.0975   0.9575   0.9706   0.8003
```

Relational Operators:

- Relational operators compare the elements in two arrays and return logical true or false values to indicate where the relation holds.

==	**Determine the equality**
>=	Determine greater than or equal to
>	Determine greater than
<=	Determine less than or equal to
<	Determine less than
~=	Determine inequality

Relational Operators continu…

- For example:

```
a=10; b=10; a==b    &    a=10; b=12; a==b

ans =                    ans =

Logical                   Logical

  1                      0
```

Relational Operators continu…

- For example:

```
a=15; b=20; a<=b    &    a=12; b=115; a>=b

ans =                    ans =

Logical                   Logical

   1                     0
```

Relational Operators continu…

- For example:

```
a=15; b=20; a~=b    &    a=115; b=115;  a~=b

ans =                    ans =

Logical                   Logical

   1                     0
```

Logical Operators:

- The logical data type represents true or false states using the numbers 1 and 0, respectively.
- The three logical operators are **&; |;** and **~**
- The meaning of **&** operator is **AND**
- The meaning of **|** operator is **OR**
- The meaning of **~** operator is **NOT**

Logical Operator & truth table:

operand	operand	AND operand
1	1	1
1	0	0
0	1	0
0	0	0

Logical Operator &, Continued…

For example,
```
>> a=[1 1 1 0 0 0];
>> b=[0 0 0 1 1 1];
>> a&b
ans =
      1×6 logical array
      0   0   0   0   0   0
```

Logical Operator &, Continued…

Consider,

```
>> a=1; b=1;

>> a&b

ans =

    logical

    1
```

Logical Operator &, Continued…

Consider,

```
>> a=0; b=1;
>> a&b
ans =
logical
0
```

Logical Operator |, truth table:

- If the two operands evaluate to true (1) or false (0), then the operator OR has the following effect.

operand	operand	operand OR operand
1	1	1
1	0	1
0	1	1
0	0	0

Logical Operator |, Continued…

Consider,

```
>>a=[1 1 1 0 0 0]; b=[0 0 0 1 1 1];

>> a|b

ans =

    1×6 logical array

    1   1   1   1   1   1
```

Logical Operator ~ truth table:

operand	NOT operand
1	0
0	1

Logical Operator ~, Continued…

- For example,
```
>>a=10;
>> ~a
ans =
logical
0
```

Consider,
```
>> a=0;
>> ~a
ans =
logical
1
```

PES UNIVERSITY

THANK YOU

ENGINEERING MATHEMATICS-I MATLAB

Department of Science and Humanities

The M-Files:

- In MATLAB, we write programs in M-files.
- They are called M-files because they must have a .m at the end of their name, (for example, myfunction.m).
- M-files are ordinary ASCII text files written in MATLAB’s Language.
- M-files can be created using any editor or word processing applications.
- There are two types of M-files: **Script files and Function files**.

Script Files; Creating and Running Script Files

- A script file is an M-file with a set of valid MATLAB commands in it.
- To create scripts files, we need to use a text editor. We can open the MATLAB editor using the command prompt.
- If we use the command prompt, then we type **edit** and then the filename (with .m extension). This will open the editor.
- The above command will create the file in default MATLAB directory.
- Alternatively, we can choose NEW -> Script. This also opens the editor and creates a file named untitled. We can name and save the file after typing the code.

Creating and Running Script Files, Continued…

- Consider an example:
- Type the following code in the editor:

```
>> a=25; b=26; c=27; d=a+b+c
```

- After creating and saving the file, you can run it by clicking the Run button on the editor window.
- The command window prompt displays the result: d=78

Function Files

- A function is a group of statements that together perform a task.
- In MATLAB, functions are defined in separate files.
- Functions can accept more than one input arguments and may return more than one output arguments.
- The syntax of the function definition line is as follows:

  function [output variables] = function_name(input variables)
- Note that the function_name must be the same as the file name (without the m.extension) in which the function is written.

Function Files, Continued…

- Consider the following example:
- First, create a script file as myname.m for the following:

  >> a=20; b=21; c=22;

  sum=a+b+c   (Press Run icon)

  prod=a*b*c   (Press Run icon)
- Then the following result is displayed on the command window:

  sum = 63; prod =9240.

Function Files, Continued…

Create a function file:

```
    >> function [sum,prod] = myname(a,b,c)
    sum = a+b+c;
     prod=a*b*c;
     end
```

On command window, type the following:

For, >> myname(20,21,22), the output is: ans =63

For, >> [s,p]=myname(20,21,22), the output is: s = 63, p =9240

Loop Control Statements

- With loop control statements, we can repeatedly execute a block of code. There are two types of loops: for loops and while loops.
- for loop: A for loop is used to repeat a statement or a group of statements for a fixed number of times.
- The syntax is:

```
for index = values
<program statements>
………………………………
end
```

for loop, Continued…

- For example, consider the following code:

```
for m=1:3
num=1/(m+1)
end
```

- When we run the file, it displays the following result:

```
num = 0.5000
num = 0.3333
num = 0.2500
```

for loop, Continued…

- For example, consider the following code:

```
for n=100:-2:96
k=1/(exp(n))
end
```

- When we run the file, it displays the following result:

```
k =3.7201e-44
k =2.7488e-43
k =2.0311e-42
```

for loop, Continued…

- For example, consider the following code:

```
  >> for a = 1:-0.1: 0
 disp(a)
   end
```

- When we run the file, it displays the following result:

```
1 0.9000 0.8000 0.7000 0.6000 0.5000 0.4000 0.3000
0.2000 0.1000
```

(These numbers will be displayed as a column matrix)

while loop

- The while loop repeatedly executes statements while condition is true.
- The while loop repeatedly executes program statement(s) as long as the expression remains true.
- The syntax is:

```
while <expression>
<program statements>
………………………………
end
```

while loop, Continued…

- Consider the following example: The code is as follows:

```
>>  a = 10;
 while(a < 15)
fprintf('value of a: %d\n', a);
 a = a + 1;
 end
```

while loop, Continued…

- When we run the file, it displays the following result:

```
value of a: 10

value of a: 11

value of a: 12

value of a: 13

value of a: 14
```

Conditional Statements

- There will be some situation where a program or a particular block has to be executed only when a specific condition is true. The conditional statements will be useful in such situations.
- The following are the conditional statements that we can use in MATLAB. if-end; if-else-end; if-elseif-elseif-else-end; switch case.

if-end statement

- if-end statement: It decides whether a particular block of code has to be executed or not, based on the given Boolean condition. When the given condition is true, only then it executes the statements inside the block otherwise not.

if-end statement, Continued…

- For example, consider the following:

```
>> number = 25;
if number > 20
fprintf('The number is greater than 20');
end
```

- When we run the file, it displays the following result:

```
The number is greater than 20.
```

if-end statement, Continued…

- For example, consider the following:

```
>> number = 50;
 if number > 60
 fprintf('The number is greater than 60');
 end
```

- When we run the file, no result will be displayed on the screen.

if-else-end statement

- An if statement can be followed by an optional else statement, which executes when the expression is false. Consider the following:

```
>> number = 25
 if number<20
 fprintf('The number is greater than 20')
else
 fprintf('The number is not less than 20')
 end
```

- When we run the file, it displays the following result: The number is not less than 20.

if-elseif-elseif-else-end

- We can use chain if-else-end statements with more than one condition. For example, consider the following:

```
>> number = 50;
if number<20
          fprintf('The number is less than 20\n');
 else
         if number<30
         fprintf('The number is less than 30\n');
else
         fprintf('The number is less than 60\n');
 end
 end
```

- Output: The number is less than 60.

Switch statement

- The switch statement is used to test for the equality against a set of known values.
- The syntax for switch statement is:

```
switch <switch_expression>
case <case_expression>
          <statements>
case <case_expression>
           <statements>
   ………………………………………….
 otherwise
           <statements>
 end
```

Switch statement, Continued…

- For example, consider the following:

```
>> grade = 'B';
     switch(grade)
     case 'A'
          fprintf('Excellent!\n' );
     case 'B'
           fprintf('Well done\n' );
     case 'C'
           fprintf('Well done\n' );
     case 'D'
           fprintf('You passed\n' );
```

Switch statement, Continued…

```
        case 'F'
              fprintf('Better try again\n' );
        otherwise
              fprintf('Invalid grade\n' );
end
```

- When we run the file, it displays the following result: Well done

PES UNIVERSITY

THANK YOU

ENGINEERING MATHEMATICS-I MATLAB

Department of Science and Humanities

MATLAB – Plotting

- To plot the graph of a function, we need to take the following steps:
- Define **x**, by specifying the **range of values** for the variable **x**, for which the function is to be plotted.
- Define the function, **y = f(x).** Call the **plot** command, as **plot(x, y).**
- The **plot** command is used to create two-dimensional plots.
- Following example would demonstrate the concept.
- Let us plot the function $y = 3.5^{0.5x}\cos(6x)$ for $-2 \le x \le 4$.

MATLAB – Plotting, Continued…

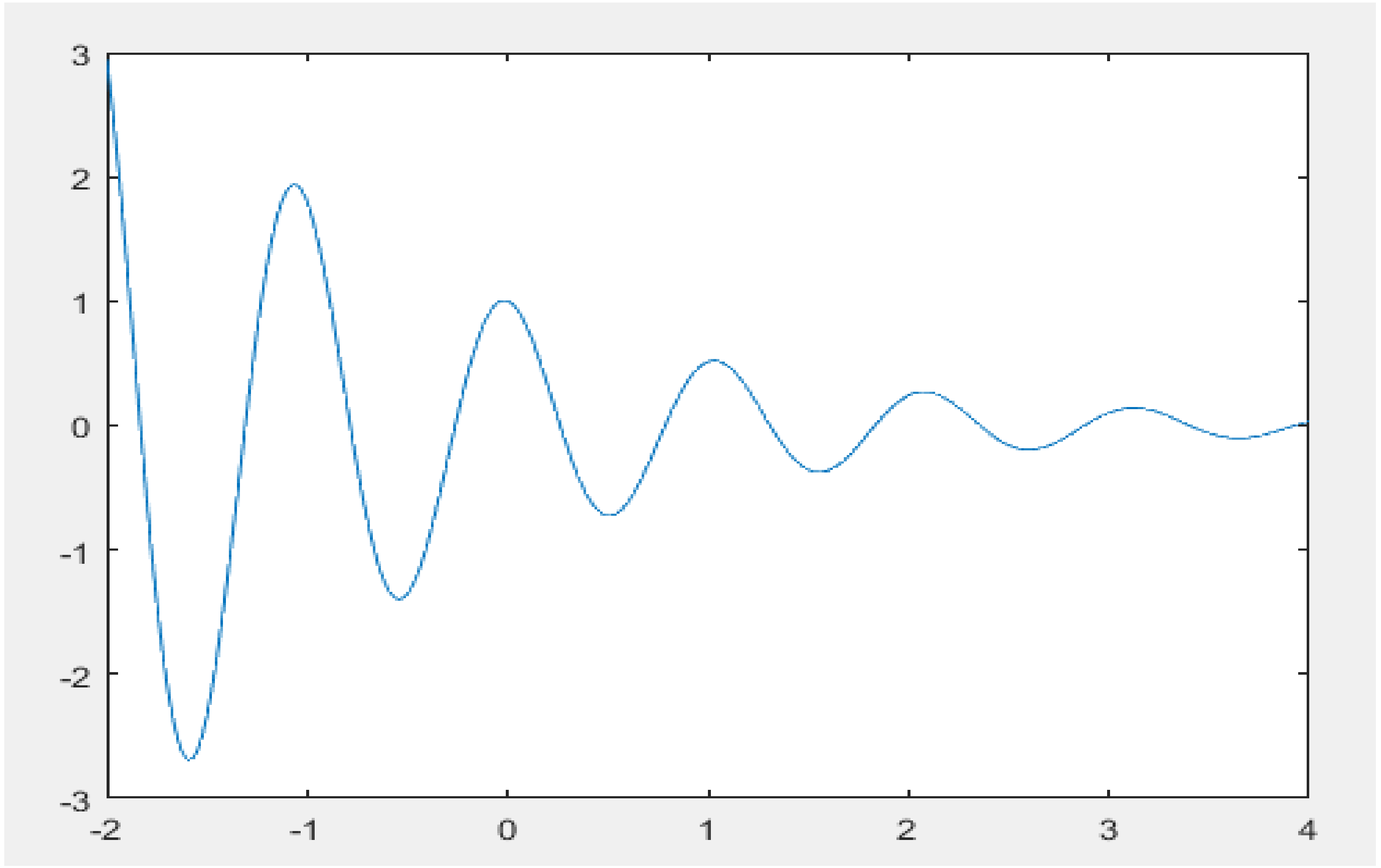
- The script file is as follows:

```
>> x=[-2:0.01:4];     (Here 0.01 is the spacing value)

>> y=3.5.^(-0.5*x).*cos(6*x);

>> plot(x,y)
```

- Here y is plotted as a function of x.

MATLAB – Plotting, Continued…

- When we run the file, MATLAB displays the following plot:



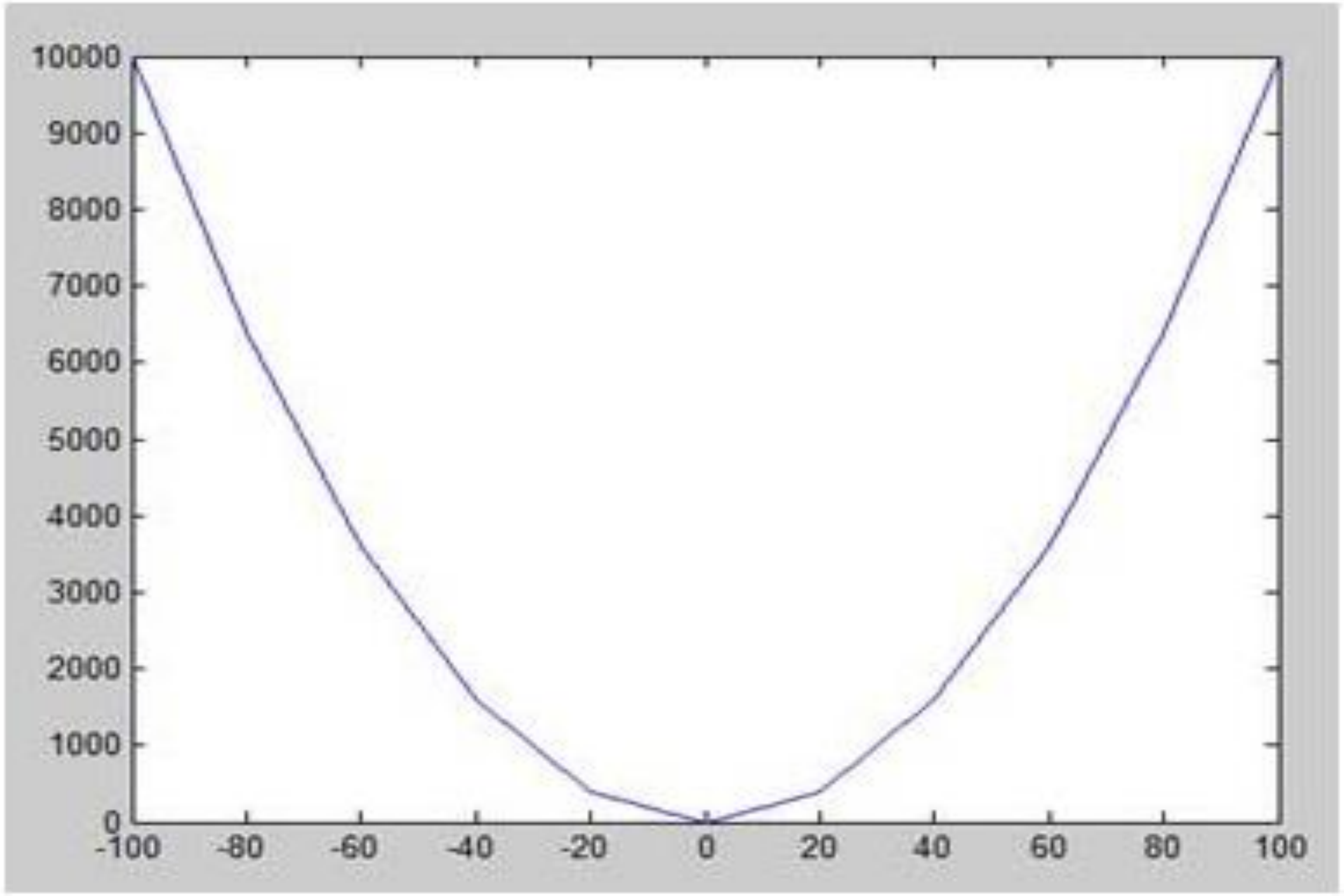
MATLAB – Plotting, Continued…

- Let us take another example to plot the function $y = x^2$.
- In this example, we will draw two graphs with the same function, but in second time, we will reduce the value of increment.
- Note that as we decrease the increment, the graph becomes smoother.
- The script file is as follows:

```
>> x = [-100:20:100];
>> y = x.^2;    [Here the “dot” is necessary]
>> plot(x, y)
```

MATLAB – Plotting, Continued…

- When we run the file, MATLAB displays the following plot:

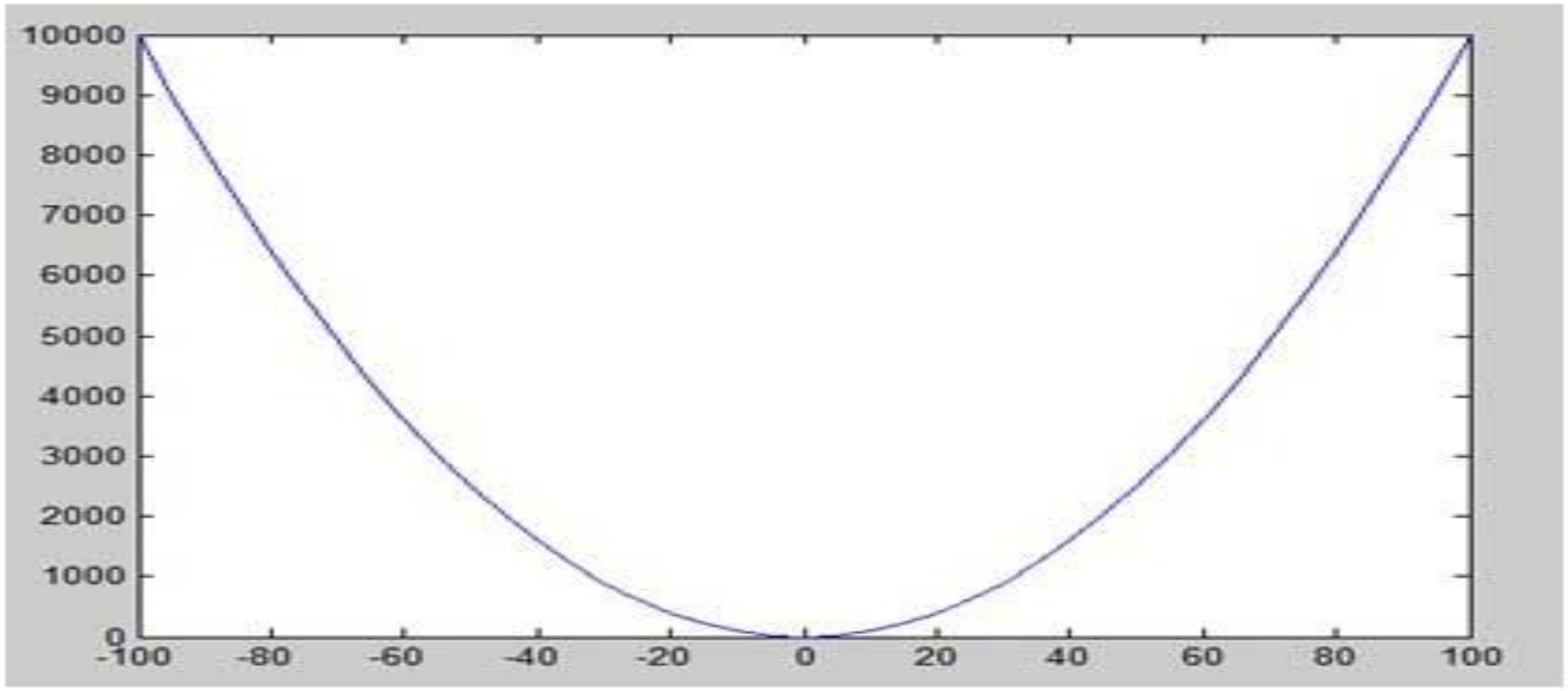


MATLAB – Plotting, Continued…

Change the code file a little, reduce the increment to 5.

```
>> x = [-100:5:100];

>> y = x.^2;

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

When you run the file, MATLAB displays the following



MATLAB - Three Dimensional Plots

- Three-dimensional plots display a surface defined by a function in two variables, $g = f(x, y)$.
- To define the function $g = f(x, y)$, we first create a set of (x, y) points over the domain of the function using the **meshgrid** command. Next, we assign the function itself. Finally, we use the **surf** command to create a surface plot.

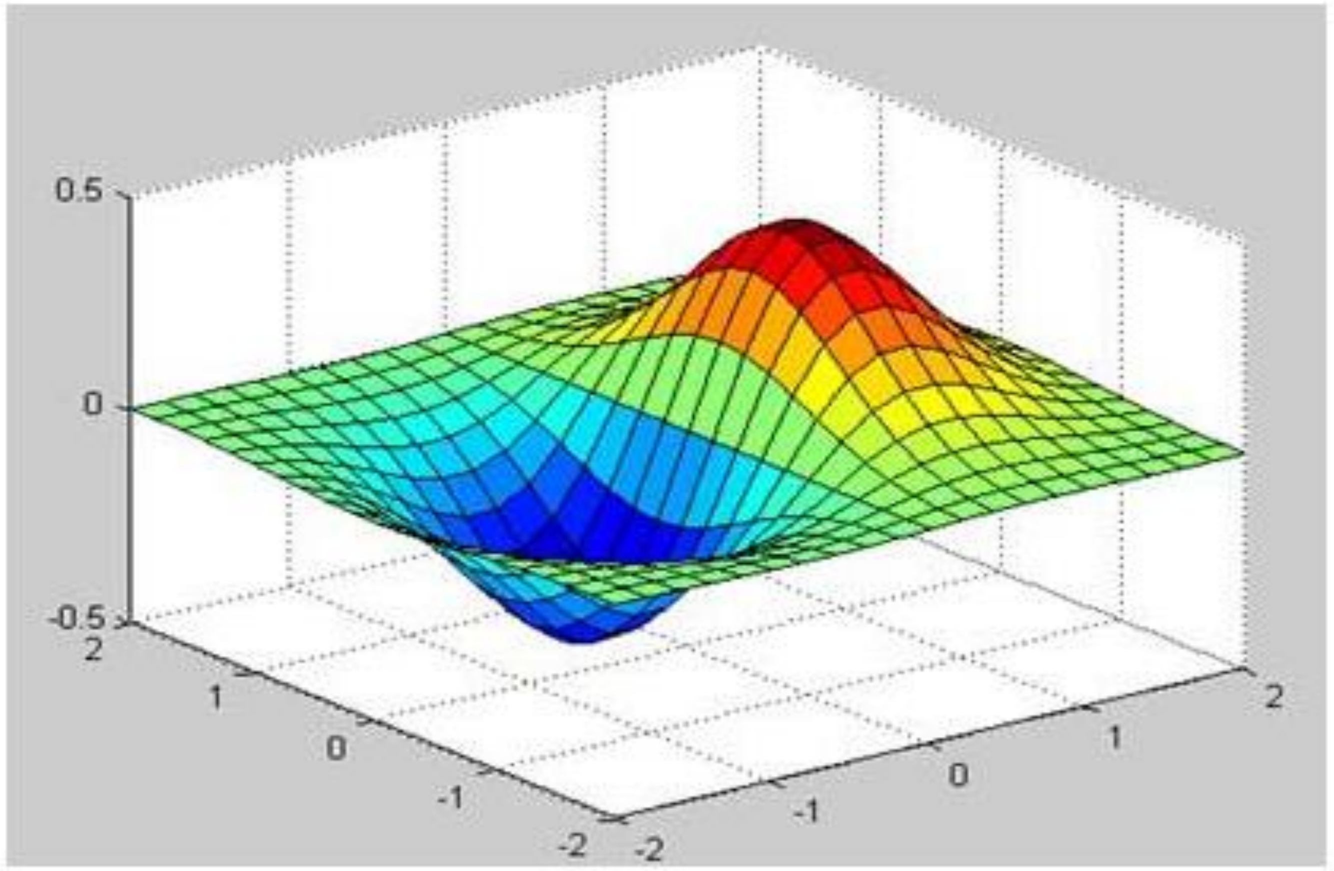
MATLAB - Three Dimensional Plots, Continued…

- The following example demonstrates the concept:
- Let us create a 3D surface map for the function $z = xe^{-(x^2+y^2)}$
- The script file is as follows:

```
>> [x,y] = meshgrid(-2:.2:2);
 >> z = x.* exp(-x.^2 - y.^2);
 >> surf(x,y,z)
```

MATLAB - Three Dimensional Plots, Continued…

- When we run the file, MATLAB displays the following 3-D map:



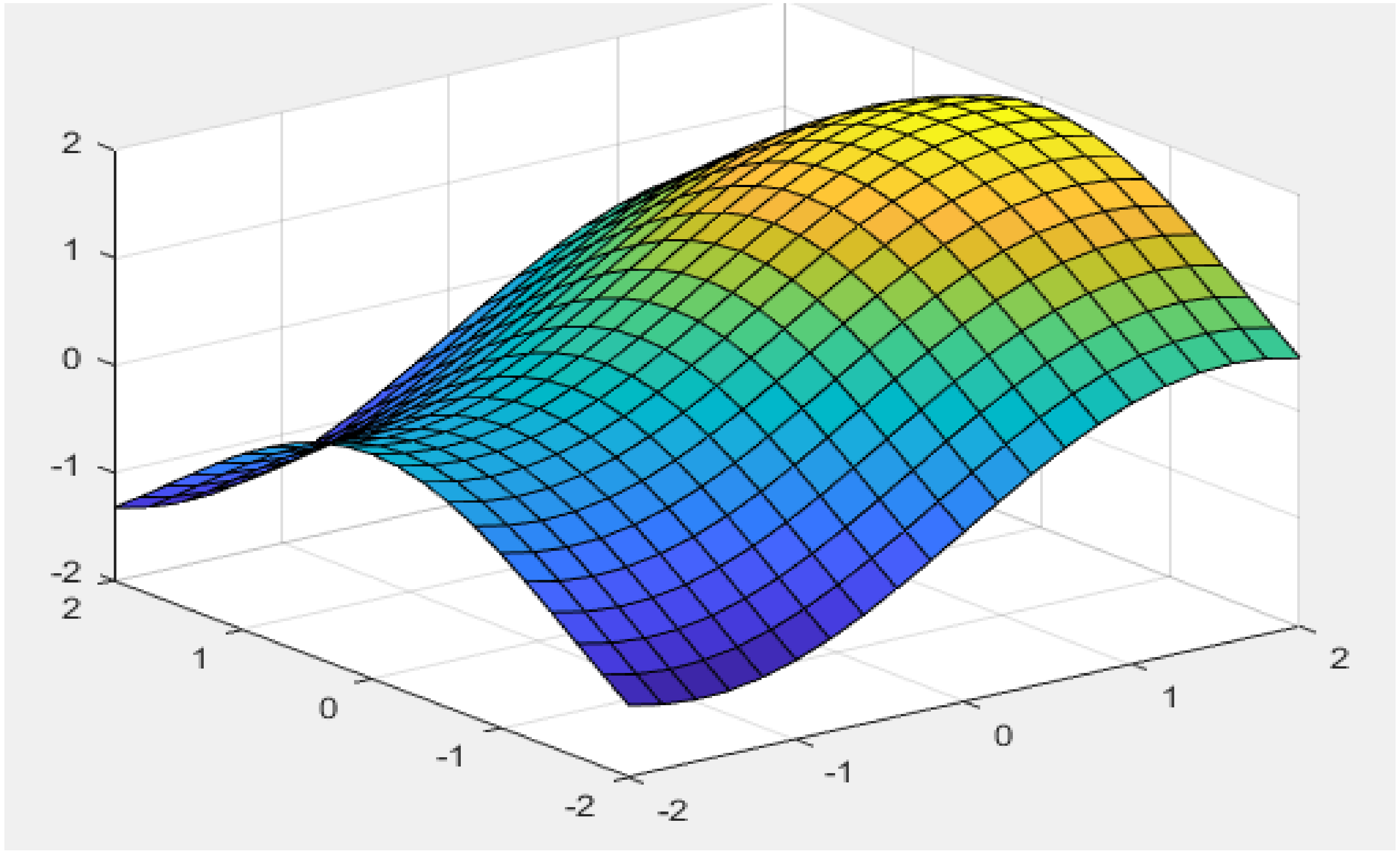
MATLAB - Three Dimensional Plots, Continued…

- Let us consider another example to create a 3D surface map for the function $z = \sin x + \cos y$
- The script file is as follows:

```
>> [x,y] = meshgrid(-2:.2:2);
>> z=sin(x)+cos(y);
 >> surf(x,y,z)
```

MATLAB - Three Dimensional Plots, Continued…

- When we run the file, MATLAB displays the following 3-D map:



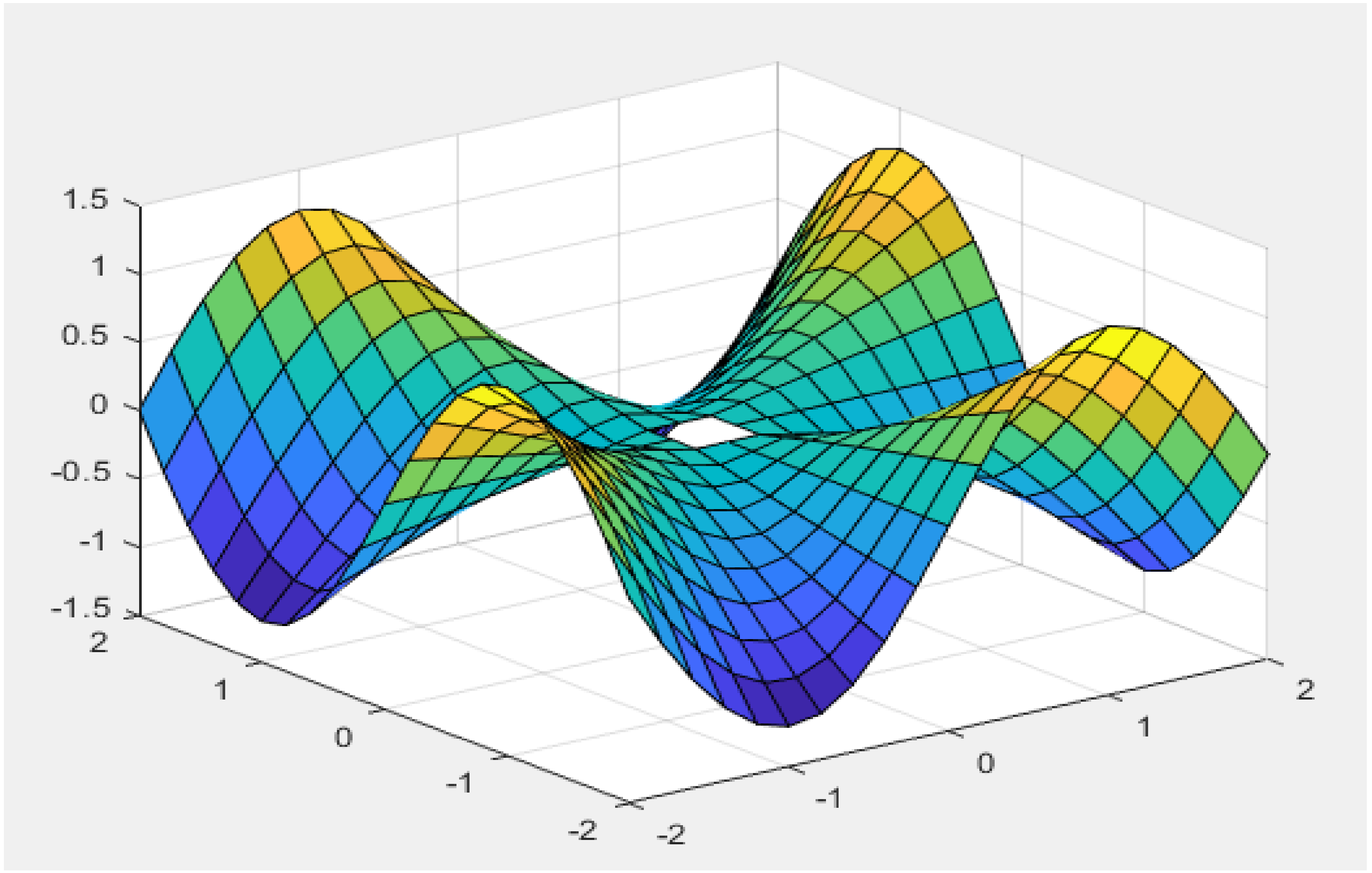
MATLAB - Three Dimensional Plots, Continued…

- Let us consider another example to create a 3D surface map for the function $z = \frac{xy(x^2-y^2)}{x^2+y^2}$.
- The script file is as follows:

```
>> [x,y] = meshgrid(-2:.2:2);
>> z=x.*y.*(x.^2-y.^2)./(x.^2+y.^2);
>> surf(x,y,z)
```

MATLAB - Three Dimensional Plots, Continued…

- When we run the file, MATLAB displays the following 3-D map:



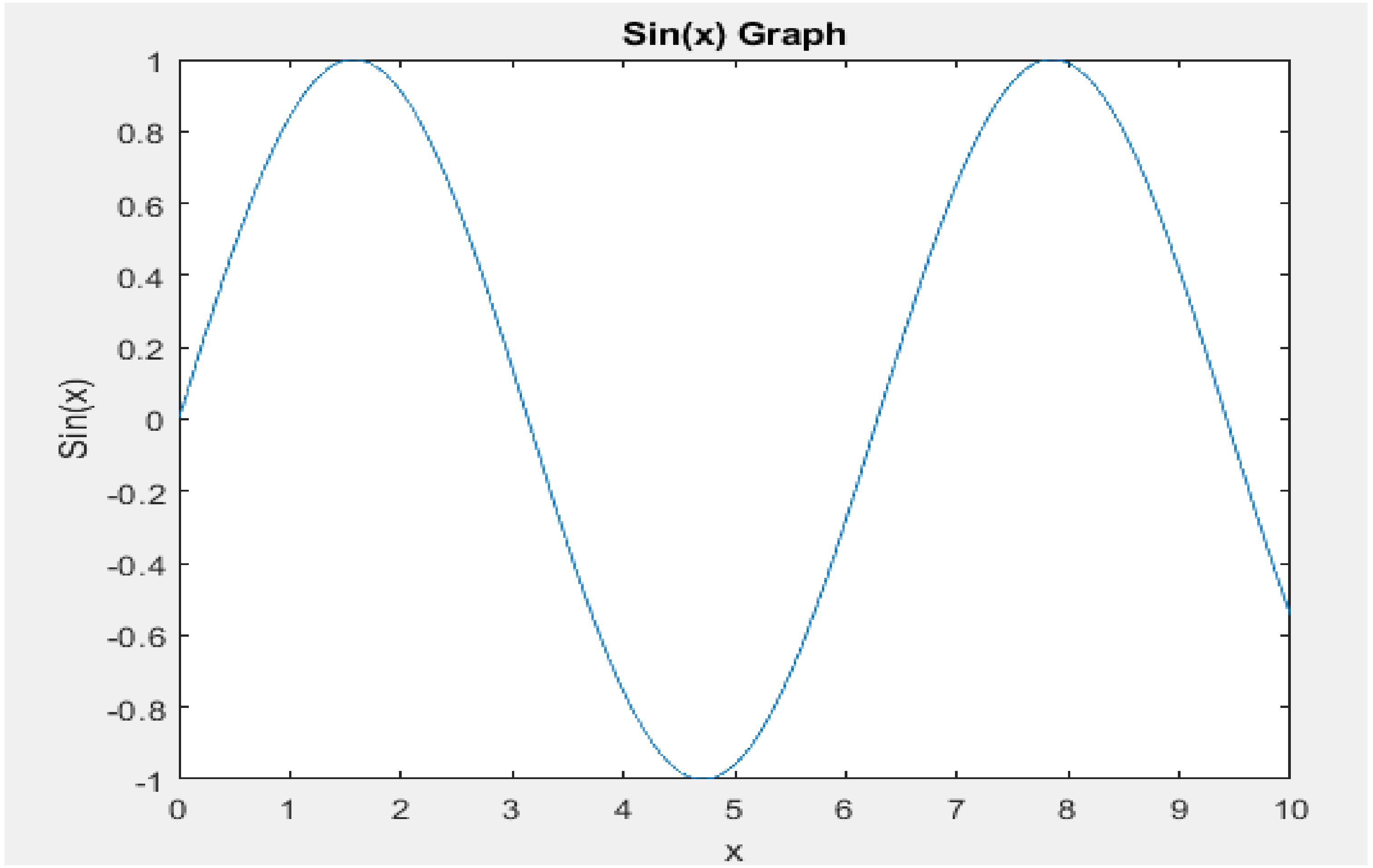
Adding Title and Labels on the Graph

- Using MATLAB we can add title, labels along the x-axis and y-axis of the graph.
- The **xlabel** and **ylabel** commands generate labels along x-axis and y-axis.
- The **title** command allows you to put a title on the graph.
- The script file is as follows:

```
>> x = [0:0.01:10];
>> y = sin(x);
>> plot(x, y), xlabel('x'), ylabel('Sin(x)'), title('Sin(x) Graph')
```

Adding Title and Labels on the Graph

- When we run the file, MATLAB displays the following plot:



MATLAB – Polar Plots

- To plot a curve in polar coordinates, we use the following command:
- **polarplot(theta,rho)**. Here **theta** is the angle in radians and rho is the radius value for each point.
- Following example would demonstrate the concept.
- Let us plot the curve $r = sin2\theta cos2\theta$.

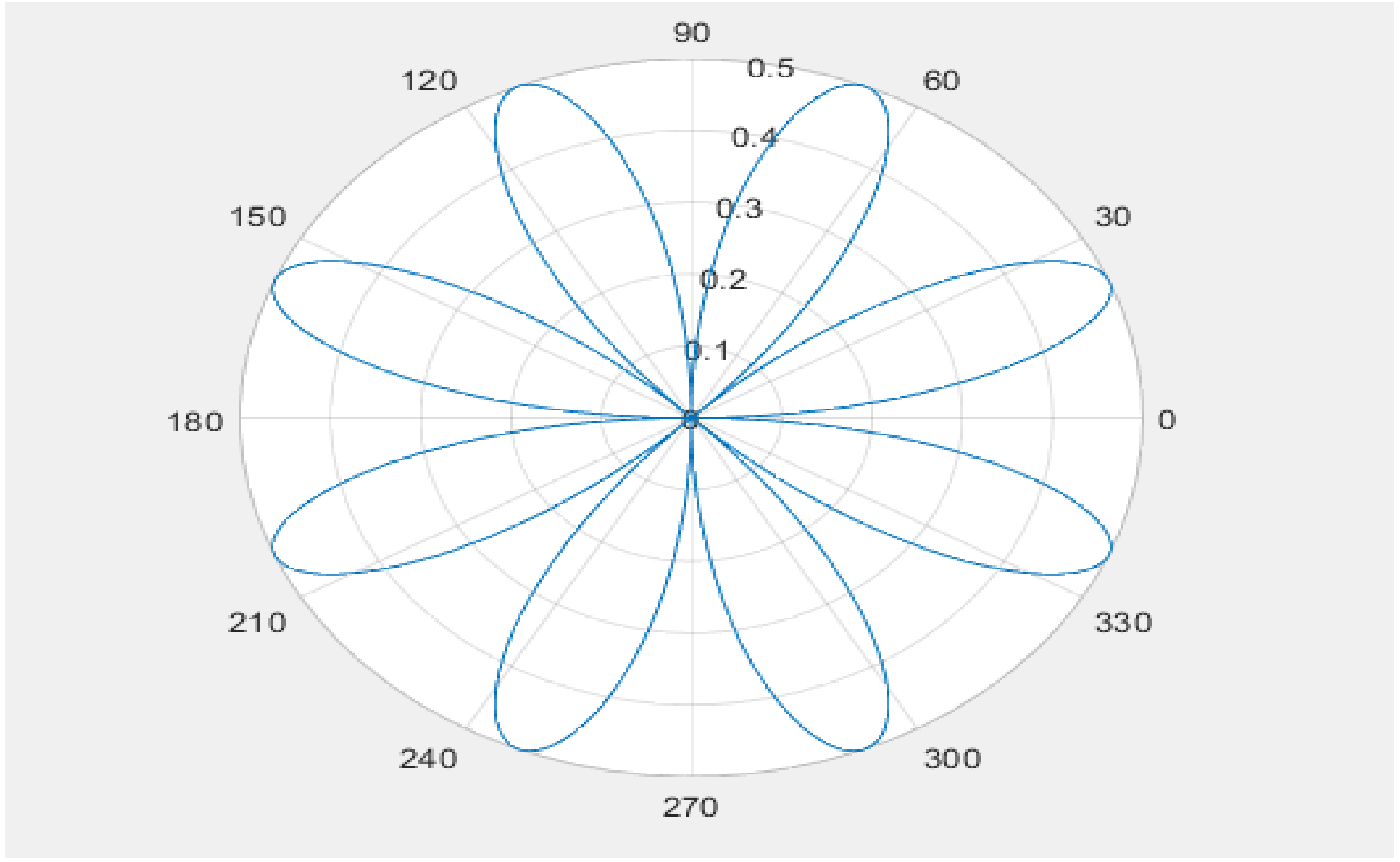
MATLAB – Polar Plots, Continued…

- The script file is as follows:

```
>> theta = 0:0.01:2*pi;

>> rho = sin(2*theta).*cos(2*theta);

>> polarplot(theta,rho)
```

MATLAB – Polar Plots, Continued…

- When we run the file, MATLAB displays the following plot:



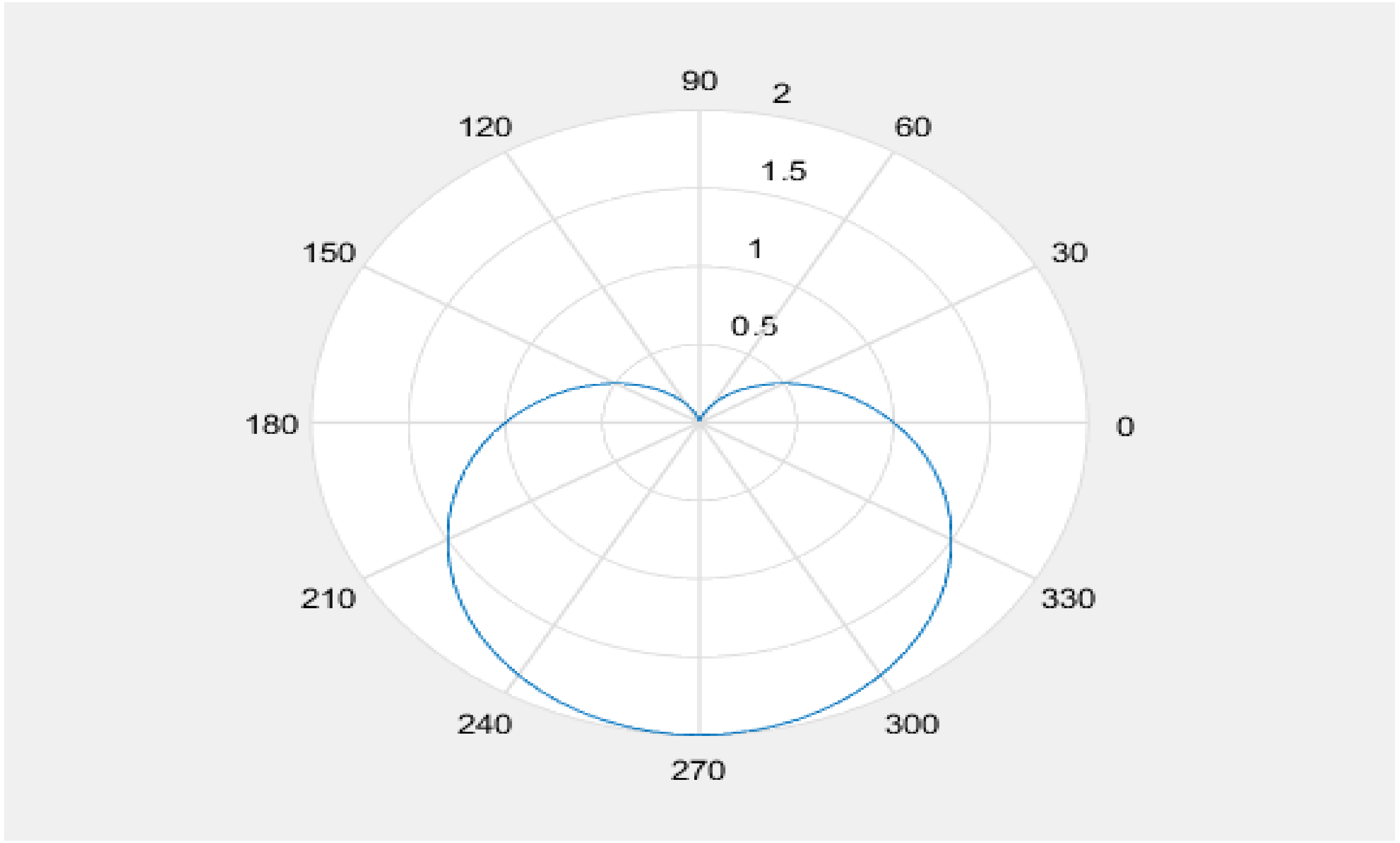
MATLAB – Polar Plots, Continued…

- Consider another example:
- Plot the curve $r = 1 - sin\theta$
- The script file is as follows:

```
>> theta = 0:0.01:2*pi;

>> rho = 1 - sin(theta);

>> polar(theta, rho)
```

MATLAB – Polar Plots, Continued…

- When we run the file, MATLAB displays the following plot:



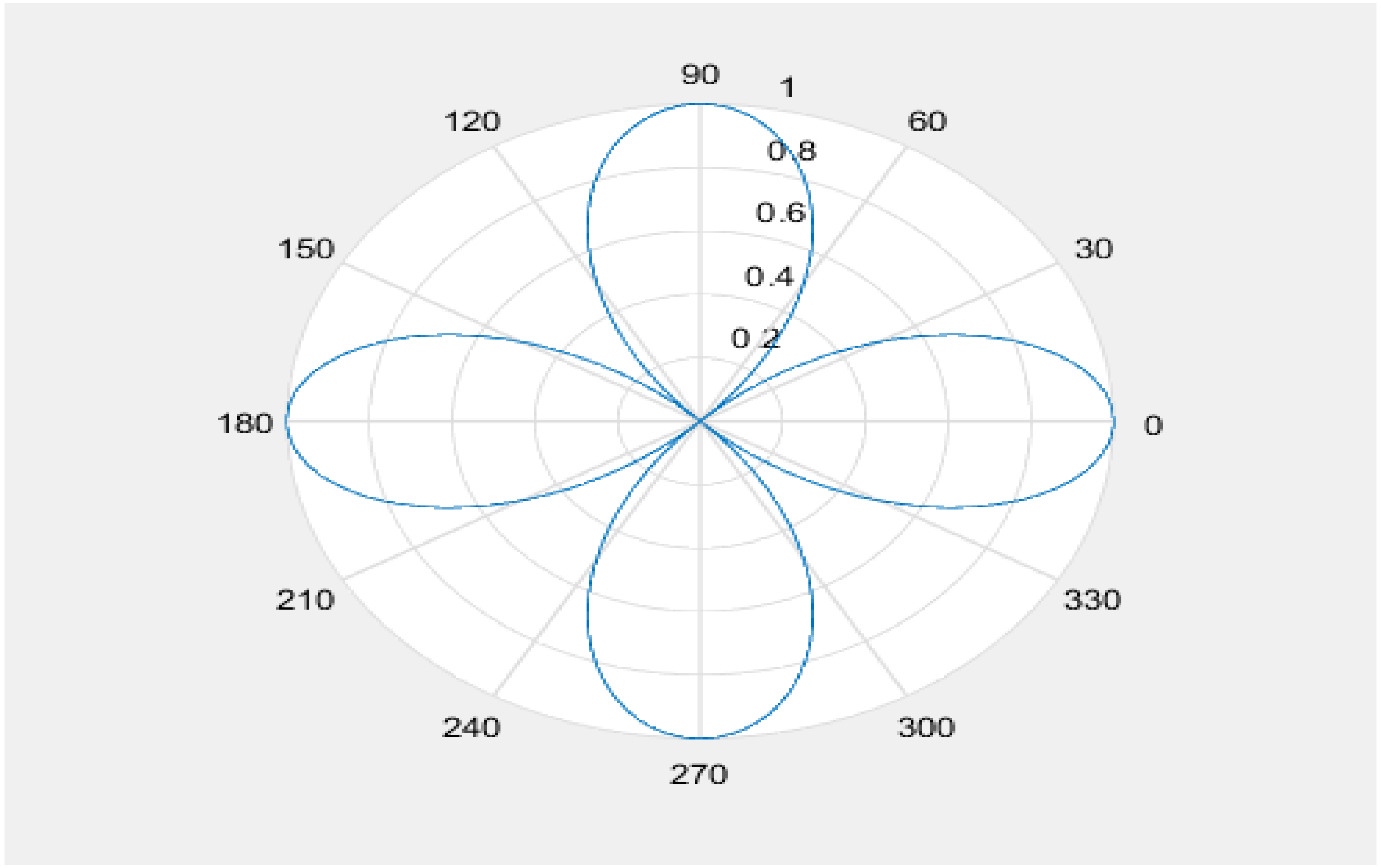
MATLAB – Polar Plots, Continued…

- Consider another example:
- Plot the curve $r = cos2\theta$
- The script file is as follows:

```
>> theta = 0:0.01:2*pi;
>> rho = cos(2*theta);
>> polar(theta, rho)
```

MATLAB – Polar Plots, Continued…

- When we run the file, MATLAB displays the following plot:



MATLAB – Parametric Plots

- To plot a parametric curve, we use the following command:
- **fplot3(xt,yt,zt)**, which plots the parametric curve xt=x(t), yt=y(t), and zt=z(t) over the default interval $-5 < t < 5$.
- Following example would demonstrate the concept.
- Let us plot a parametric curve: *x*=sin(*t*); *y*=cos(*t*); *z*=*t* over the default parameter range [-5 5].

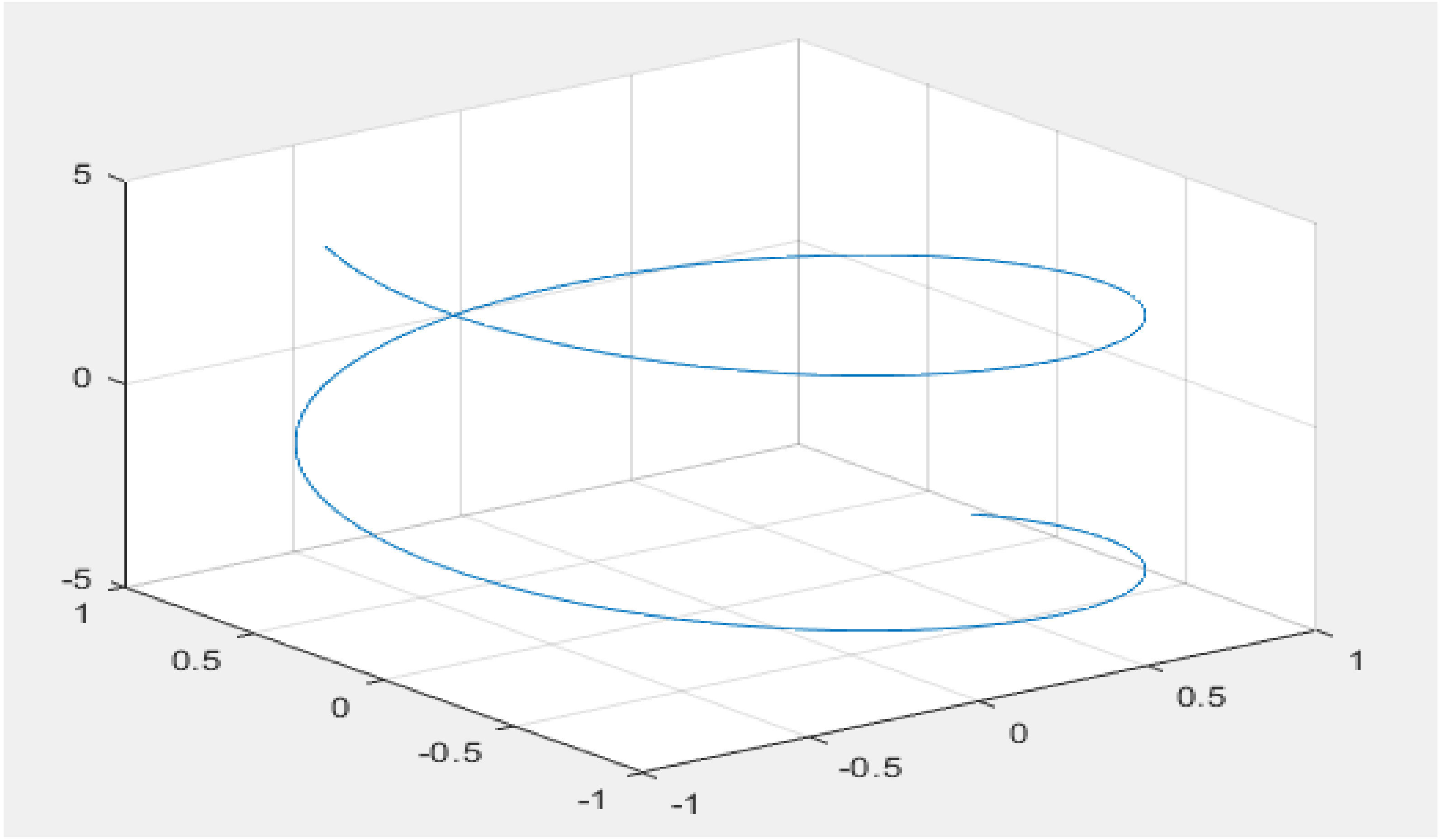
MATLAB – Parametric Plots, Continued…

- The script file as follows:

```
>> syms t
>> xt = sin(t);
>> yt = cos(t);
>> zt = t;
>> fplot3(xt,yt,zt)
```

MATLAB – Parametric Plots, Continued…

➢ When we run the file, MATLAB displays the following plot:



MATLAB – Polar Plots, Continued…

- Consider another example:
- Plot the parametric curve $x = e^{-\frac{t}{10}}\sin(5t)\,;y = e^{-\frac{t}{10}}\cos(5t); z = t$

  over the parameter range [-10 10] by specifying the fourth argument of fplot3.

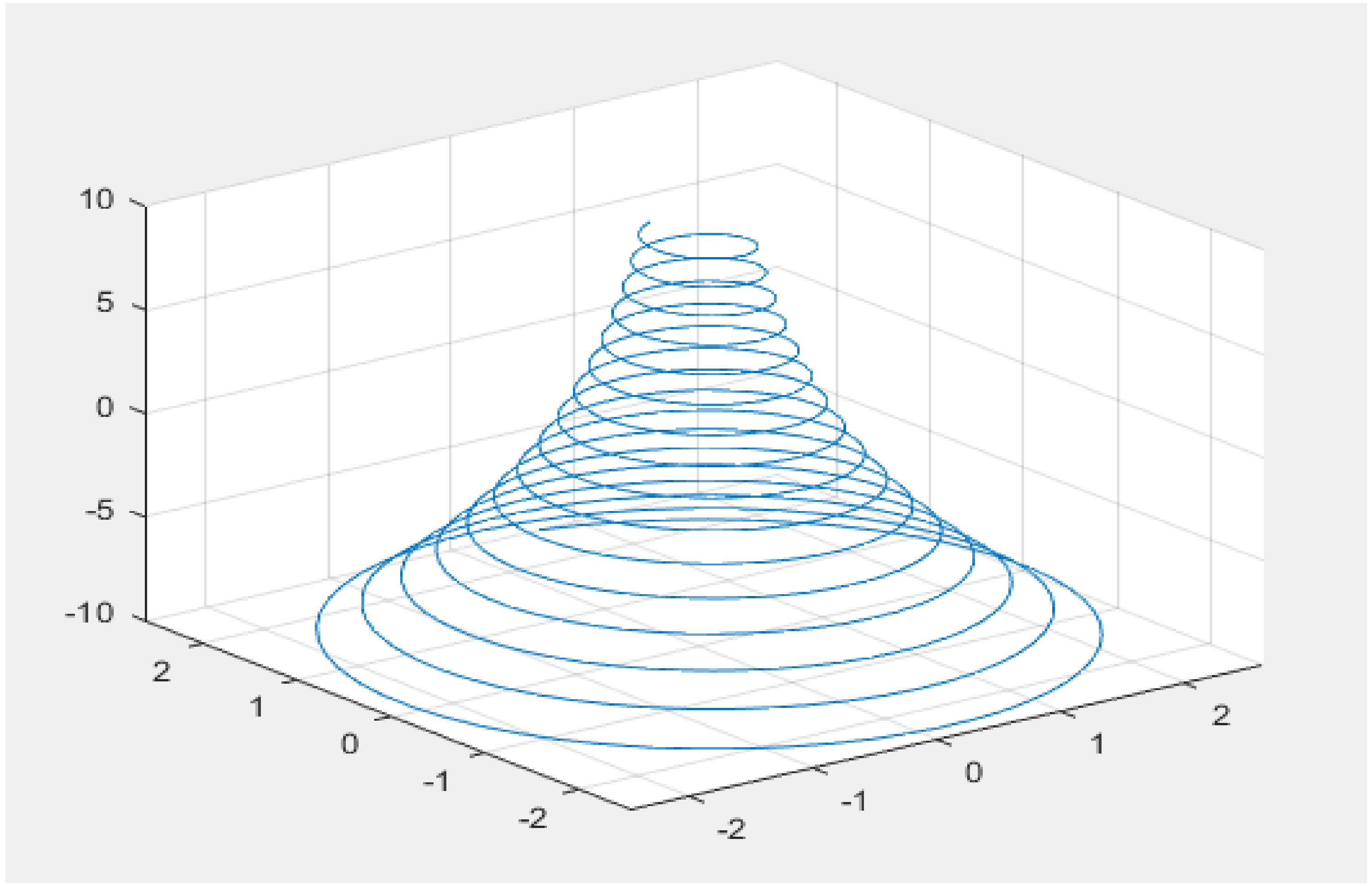
MATLAB – Parametric Plots, Continued…

- The script file as follows:

```
>> syms t
>> xt = exp(-t/10).*sin(5*t);
>> yt = exp(-t/10).*cos(5*t);
>> zt = t;
>> fplot3(xt,yt,zt,[-10 10])
```

MATLAB – Parametric Plots, Continued…

- When we run the file, MATLAB displays the following plot:



THANK YOU

ENGINEERING MATHEMATICS-I MATLAB

Department of Science and Humanities

Radius Of Curvature:

Find the radius of curvature of the curve $x^{\frac{2}{3}} + y^{\frac{2}{3}} = a^{\frac{2}{3}}$ at any point (x,y) of the curve.

```
>> syms x y a

>> F(x,y)=x^(2/3)+y^(2/3)-a^(2/3);

>> dy_dx = - diff(F,x)/diff(F,y)

  Out put: dy_dx(x, y) =-y^(1/3)/x^(1/3)

>> G(x,y)=-y^(1/3)/x^(1/3);
>> a=diff(G,x);
>> b=diff(G,y);
>> c=a+b*G(x,y)
 Out put: c(x, y) =1/(3*x^(2/3)*y^(1/3)) + y^(1/3)/(3*x^(4/3))
```

Radius Of Curvature Continued…

```
>> simplify(c)

Out put: (x, y) =(x^(2/3) + y^(2/3))/(3*x^(4/3)*y^(1/3))

>> d=(1+G(x,y)^2)^(3/2)

Out put: d =(y^(2/3)/x^(2/3) + 1)^(3/2)

>> rho=d/c

Out put: rho(x, y) =(y^(2/3)/x^(2/3) + 1)^(3/2)/(1/(3*x^(2/3)*y^(1/3)) +
y^(1/3)/(3*x^(4/3)))

>> simplify(rho(x,y))

Out put: (y^(2/3)/x^(2/3) + 1)^(3/2)/(1/(3*x^(2/3)*y^(1/3)) +
y^(1/3)/(3*x^(4/3)))
```

Radius Of Curvature Continued…

Find the radius of curvature of the curve $xy = c^2$at any point (x,y) of the curve.

```
>> syms x y a
 F(x,y)=x*y-a^2;
 dy_dx = - diff(F,x)/diff(F,y)
```

Out put: dy_dx(x, y) =-y/x

```
 G(x,y)=-y/x;
a=diff(G,x);
 b=diff(G,y);
 c=a+b*G(x,y)
```

Out put: c(x, y) =(2*y)/x^2

Radius Of Curvature Continued…

```
>> d=(1+G(x,y)^2)^(3/2)
```

Out put: d =(y^2/x^2 + 1)^(3/2)

```
>> rho=d/c
```

Out put: rho(x, y) =(x^2*(y^2/x^2 + 1)^(3/2))/(2*y)

```
>> simplify(rho)
```

Out put: (x, y) =(x^2*(y^2/x^2 + 1)^(3/2))/(2*y)

Radius Of Curvature Continued…

Find the radius of curvature of the curve $r = e^{2\theta}$ at any point on the curve.

```
>> syms theta
>> r=exp(2*theta);
>> r1=diff(r,theta);
>> r2=diff(diff(r,theta));
>> a=(r^2+r1^2)^(3/2)
```

Out put: a =(5*exp(4*theta))^(3/2)

```
>> b=r^2+2*r1^2-r*r2
```

Out put: b =5*exp(4*theta)

```
>> rho=a/b
```

Out put: rho =(exp(-4*theta)*(5*exp(4*theta))^(3/2))/5

Radius Of Curvature Continued…

```
>> simplify(rho)
```

Out put:
```
5^(1/2)*exp(4*theta)^(1/2)
```

Radius Of Curvature Continued…

Find the radius of curvature of the parametric curve $x = 6t^2 - 3t^4$, $y = 8t^3$.

```
>> syms t
>> x=6*t^2-3*t^4;
>> x1=diff(x,t);
>> x2=diff(diff(x,t));
>> y=8*t^3;
>> y1=diff(y,t);
>> y2=diff(diff(y,t));
>> a=(x1^2+y1^2)
```

Out put: a =(- 12*t^3 + 12*t)^2 + 576*t^4

```
>> b=simplify(a)
```

Out put: b =144*t^2*(t^2 + 1)^2

Radius Of Curvature Continued…

```
>> c=(x1*y2)-(y1*x2)
Out put: c =48*t*(- 12*t^3 + 12*t) + 24*t^2*(36*t^2 - 12)
>> d=simplify(c)
Out put: d =288*t^2*(t^2 + 1)
>> e=b^(3/2)
Out put: e =(144*t^2*(t^2 + 1)^2)^(3/2)
>>rho=e/d
Out put: rho =(144*t^2*(t^2 + 1)^2)^(3/2)/(288*t^2*(t^2 + 1))
>>simplify(rho)
Out put: rho= (6*(t^2*(t^2 + 1)^2)^(3/2))/(t^2*(t^2 + 1))
```

PES UNIVERSITY

THANK YOU

ENGINEERING MATHEMATICS-I
MATLAB

Department of Science and Humanities

Finding partial derivative of a function:

Find the partial derivative of the following functions:

a) If $f = \sin(x) + y^3 + x^{10} - y^2 + \log(x)$, then find f_x & f_y.

b) If $f = x^2 + 2 * y^2 - 22$, then find ${f_x}^2$ & ${f_y}^2$.

c) If $f = xy^3 + tanx + cos\sqrt{logx}$, then find f_x.

d) If $f = {}^{xy^3}/_{x+y}$, then find f_x; f_y; ${f_x}^2$; f_{xy}; f_{yx}.

Finding partial derivative of a function, Continued…

If $f = \sin(x) + y^3 + x^{10} - y^2 + \log(x)$, then find f_x & f_y.

```
>> syms x y

>> f=sin(x)+y^3+x^10-y^2+log(x);

>> diff(f,x)

>> diff(f,y)
```

Out put:

```
f =log(x) + sin(x) + x^10 - y^2 + y^3

ans =cos(x) + 1/x + 10*x^9

ans =3*y^2 - 2*y
```

Finding partial derivative of a function, Continued…

If f $= x^2 + 2 * y^2 - 22$, then find f_{xx} & f_{yy}.

```
>> syms x y

>> f=x^2+2*y^2-22

>> diff(f,x,2)

>> diff(f,y,2)
```

Out put:

```
f = x^2 + 2*y^2 – 22

ans =2

ans=4
```

Finding partial derivative of a function, Continued…

If $f = xy^3 + tanx + cos\sqrt{logx}$, then find f_x.

```
>> syms x y;

>> f=x*y^3+tan(x)+cos(sqrt(log(x)))

>> diff(f,x)
```

Out put:

```
f = cos(log(x)^(1/2)) + tan(x) + x*y^3

ans =tan(x)^2 + y^3 - sin(log(x)^(1/2))/(2*x*log(x)^(1/2)) + 1
```

Finding partial derivative of a function, Continued…

If $f = {}^{xy^3}/_{x+y}$, then find $f_x; f_y;\ f_x{}^2; f_{xy}; f_{yx}$.

```
>> syms x y

>> f=(x*y^3)/(x+y)

>> diff(f,x)

>> diff(f,y)

>> diff(f,x,2)

>> diff(f, x, y)

>> diff(f, y, x)
```

Out put: `f = (x*y^3)/(x + y);  ans = y^3/(x + y) - (x*y^3)/(x + y)^2`

Finding partial derivative of a function, Continued…

```
ans =(3*x*y^2)/(x + y) - (x*y^3)/(x + y)^2

ans =(2*x*y^3)/(x + y)^3 - (2*y^3)/(x + y)^2

ans =

(3*y^2)/(x + y) - y^3/(x + y)^2 - (3*x*y^2)/(x + y)^2 + (2*x*y^3)/(x + y)^3

ans =

(3*y^2)/(x + y) - y^3/(x + y)^2 - (3*x*y^2)/(x + y)^2 + (2*x*y^3)/(x + y)^3
```

Note that $f_{xy} = f_{yx}$.

Taylor’s and Macluarin’s series expansion of a function of single variable:

Expand f(x)=$e^{x sin x}$ about the point $\mathrm{x} = 2$ up to third degree terms.

```
>> syms x

>> f = exp(x*sin(x));

>> t= taylor(f, 'ExpansionPoint', 2, 'Order', 3)
```

Out put:

```
 t=exp(2*sin(2))  +  exp(2*sin(2))*(2*cos(2)  +  sin(2))*(x  -  2)  +  exp(2*sin(2))*(x  -
2)^2*(cos(2) - sin(2) + (2*cos(2) + sin(2))*(cos(2) + sin(2)/2))
```

Taylor’s and Macluarin’s series expansion of a function of single variable:

Expand f(x)=log(cosx) about the point $x = \frac{\pi}{3}$ up to fifth degree terms.

```
>> syms x

>> f = log(cos(x));

>> t= taylor(f, 'ExpansionPoint', pi/3, 'Order', 5)
```

Out put:

```
t = - log(2) - 3^(1/2)*(x - pi/3) - (4*3^(1/2)*(x - pi/3)^3)/3 - 2*(x - pi/3)^2 - (10*(x -
pi/3)^4)/3
```

Taylor’s and Macluarin’s series expansion of a function of single variable:

Expand f(x)=log(secx) about the origin up to six degree terms.

```
>> syms x

>> f = log(sec(x));

>> T= taylor(f, 'Order', 7)
```

Out put:

```
T = x^6/45 + x^4/12 + x^2/2
```

Taylor’s and Macluarin’s series expansion of a function of single variable:

Expand f(x)=sin(log(x^2+2x+1)) about the origin up to six degree terms.

```
>> syms x

>> f = sin(log(x^2+2*x+1));

>> T= taylor(f, 'Order', 7)
```

Out put:

```
T= (3*x^6)/2 - (5*x^5)/3 + (3*x^4)/2 - (2*x^3)/3 - x^2 + 2*x
```

Taylor’s and Macluarin’s series expansion of a function of single variable:

Plot the graph of the following:

1. sinx:

```
>> t = [0:0.1:2*pi]

>> a = sin(t);

>> plot(t,a)
```

2. cosx:

```
>> t = [0:0.1:2*pi]

>> a = cos(t);

>> plot(t,a)
```

Taylor’s and Macluarin’s series expansion of a function of two variables:

Expand $f(x, y) = e^{x} cos y$ about the point $x = 1, y = \frac{\pi}{4}$ up to three degree terms.

```
>> syms x y
>> f=exp(x)*cos(y);
>> t = taylor(f, [x, y], [1, pi/4], 'Order', 3)
```

Out put:

```
T = (2^(1/2)*exp(1))/2 - (2^(1/2)*exp(1)*(y - pi/4)^2)/4 + (2^(1/2)*exp(1)*(x -
1)^2)/4 - (2^(1/2)*exp(1)*(y - pi/4))/2 + (2^(1/2)*exp(1)*(x - 1))/2 -
(2^(1/2)*exp(1)*(y - pi/4)*(x - 1))/2
```

Taylor’s and Macluarin’s series expansion of a function of two variables:

Expand $f(x, y) = x^3 + y^3 + xy^2$ about $x = 1, y = 2$ up to fourth degree terms.

```
  >> syms x y

 >> f=x^3+y^3+x*y^2;

 >> t = taylor(f, [x, y], [1, 2], 'Order', 4)
```

Out put:

```
t=7*x + 16*y + 4*(x - 1)*(y - 2) + 3*(x - 1)^2 + (x - 1)^3 + 7*(y - 2)^2 + (y -
2)^3 + (x - 1)*(y - 2)^2 - 26
```

Taylor’s and Macluarin’s series expansion of a function of two variables:

Expand $f(x, y) = e^y \log(1+x)$ about the origin up to fourth degree terms.

```
>> syms x y

>> f=exp(y)*log(1+x);

>> T= taylor(f, [x, y], 'Order', 4)
```

Out put:

```
T= x^3/3 - (x^2*y)/2 - x^2/2 + (x*y^2)/2 + x*y + x
```

Taylor’s and Macluarin’s series expansion of a function of two variables:

Expand $f(x, y) = e^{x}\tan y$ about the origin up to fifth degree

terms.

```
>> syms x y

>> f=exp(x)*tan(y);

>> T= taylor(f, [x, y], 'Order', 5)
```

Out put:

```
T =(x^3*y)/6 + (x^2*y)/2 + (x*y^3)/3 + x*y + y^3/3 + y
```

PES UNIVERSITY

THANK YOU