CSE111 Introduction to Computer Applications

Lecture 13

Introduction to Matlab - Part 1

Prepared by Asst. Prof. Dr. Mohamed KURDI

Revised and presented by Asst. Prof. Dr. Samsun M. BAŞARICI

Summary of the Last Week

- Humans & Computer Algorithms
- Variables
- Pseudocode
  - Basic Pseudocode Statements
- Flowcharts
  - Flowchart Building Units
  - Control Structures in Flowcharts
    - The Sequence Structure.
    - The Decision Structure.
    - The Repetition Structure.

T.C. Adnan Menderes University Faculty of Engineering

Learning Objectives

- Explore and identify Matlab desktop panels.
  - Apply and understand how to enter commands and expressions.
  - Apply and understand how to perform arithmetic operations.
  - Apply and understand how to manage a work session.
- Apply and understand how to perform:
  - Array creation.
  - Array operations.
  - Concatenation.
  - Array indexing.
- Apply and understand how to save and retrieve workspace variables.
- Apply and understand how to create character strings.
- Apply and understand how to use and call Matlab functions.
- Apply and understand how to construct 2-D and 3-D plots.

T.C. Adnan Menderes University Faculty of Engineering

Outlines

- Introducing Matlab Desktop Panels.
  - Entering Commands and Expressions.
  - Performing Arithmetic Operations.
  - Managing a Work Session.
- Working on Matrices and Arrays.
  - Array Creation.
  - Array Operations.
  - Concatenation.
  - Complex Numbers.
  - Array Indexing.
- Workspace Variables.
- Character Strings.
- Calling Functions.
- 2-D Plots
- 3-D Plots
- Subplots

T.C. Adnan Menderes University Faculty of Engineering

What is Matlab?

- Matlab is a high-level language and interactive environment used by millions of engineers and scientists worldwide.
- It lets you explore and visualize ideas and collaborate across disciplines including signal and image processing, communications, control systems, and computational finance.
- Capabilities:
  - Numeric Computation
  - Data Analysis and Visualization
  - Programming and Algorithm Development
  - Application Development and Deployment

T.C. Adnan Menderes University Faculty of Engineering

Desktop Basics

To start Matlab, Double-click on the Matlab shortcut icon on the Windows desktop, the desktop appears in its default layout. The desktop includes these panels:

Command Window

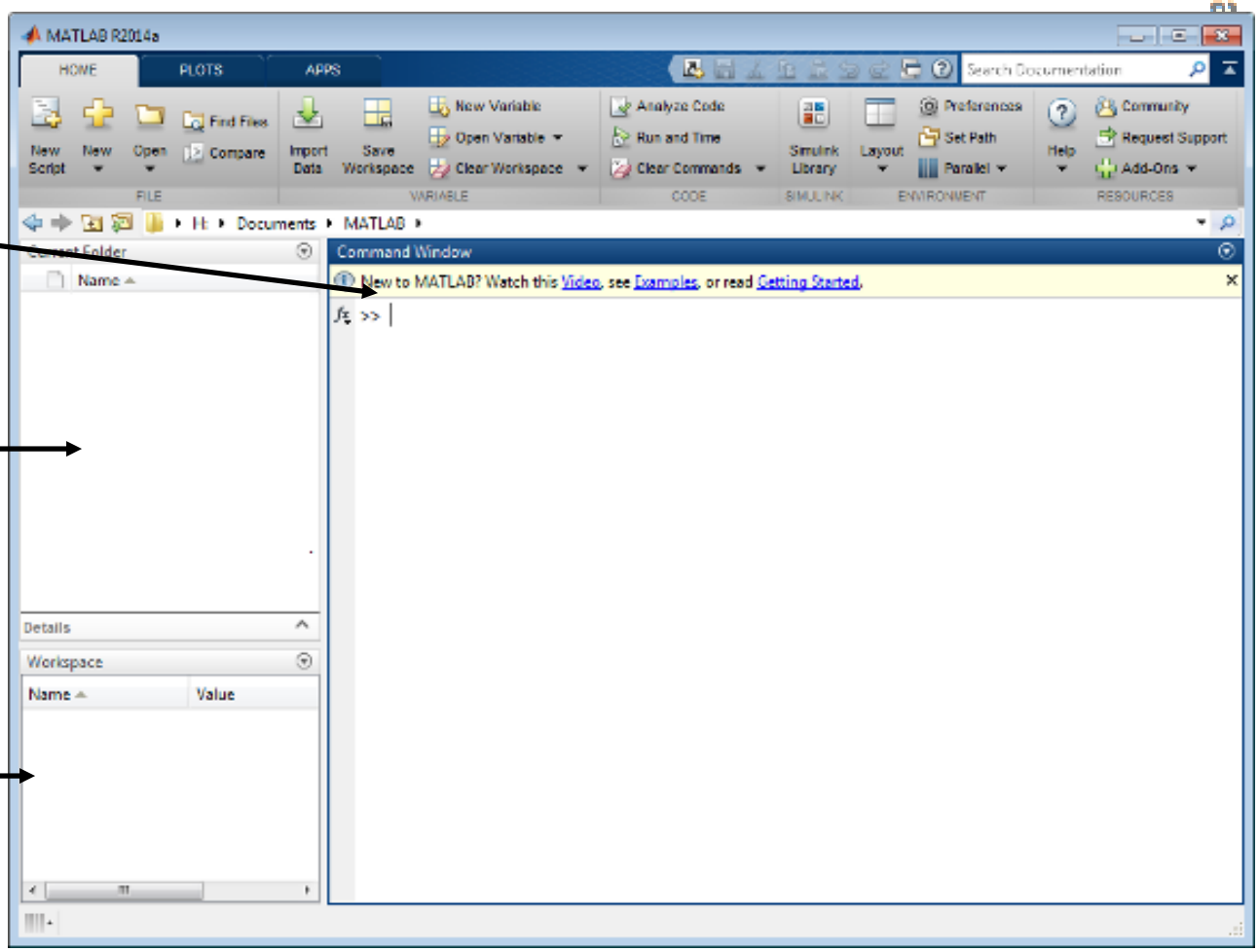
Enter commands at the command line, indicated by the prompt (>>).

Current Folder

Access your files

Workspace

Explore data that you create or import from files.



Entering Commands and Expressions

- Create a variable named **a** by typing this statement at the command line:

  a = 1

  Matlab adds variable a to the workspace and displays the result in the command window.

  a =

  1

- Create a few more variables.

  b = 2

  b =

  2

  c = a + b

  c =

  3

  d = cos(a)

  d =

  0.5403

T.C. Adnan Menderes University Faculty of Engineering

Entering Commands and Expressions

- When you do not specify an output variable, Matlab uses the variable *ans*, short for *answer*, to store the result of your calculation.

  sin(a)

  ans =

  0.8415

- If you end a statement with a semicolon, Matlab performs the computation, but suppresses the display of output in the command window.

  e = a*b;

- You can recall previous commands by pressing the up- and down-arrow keys, ↑ and ↓.

  Press the arrow keys either at an **empty command line** or after you **type the first few characters** of a command.

  For example, to recall the command b = 2, type **b**, and then press the up-arrow key.

T.C. Adnan Menderes University Faculty of Engineering

Arithmetic Operations

Symbol	Operation	MATLAB form
^	exponentiation: a^b	a^b
*	multiplication: ab	a*b
/	right division: $a/b = \frac{a}{b}$	a/b
\	left division: $a\backslash b = \frac{b}{a}$	a\ b
+	addition: $a + b$	a+b
-	subtraction: $a - b$	a-b

Table 1.1–1, page 8

Introduction to MATLAB for Engineers, Third Edition William J. Palm III

Copyright © 2010. The McGraw-Hill Companies, Inc.

T.C. Adnan Menderes University Faculty of Engineering

Order of Precedence

Precedence	Operation
First	Parentheses, evaluated starting with the innermost pair.
Second	Exponentiation, evaluated from left to right.
Third	Multiplication and division with equal precedence, evaluated from left to right.
Fourth	Addition and subtraction with equal precedence, evaluated from left to right.

Table 1.1–2, page 9

Introduction to MATLAB for Engineers, Third Edition William J. Palm III

Copyright © 2010. The McGraw-Hill Companies, Inc.

T.C. Adnan Menderes University Faculty of Engineering

Examples of Precedence

```
8 + 3*5
ans =
        23
8 + (3*5)
ans =
        23
(8 + 3)*5
ans =
        55
4^2/4*2
ans =
        8
4^2/(4*2)
ans =
        2
```

T.C. Adnan Menderes University Faculty of Engineering

Managing a Work Session

Command	Description
`clc`	Clears the Command window.
`clear`	Removes all variables from memory.
`clear var1 var2`	Removes the variables `var1` and `var2` from memory.
`exist('name')`	Determines if a file or variable exists having the name 'name'.
`quit`	Stops MATLAB.
`who`	Lists the variables currently in memory.
`whos`	Lists the current variables and sizes, and indicates if they have imaginary parts.
`:`	Colon; generates an array having regularly spaced elements.
`,`	Comma; separates elements of an array.
`;`	Semicolon; suppresses screen printing; also denotes a new row in an array.
`...`	Ellipsis; continues a line.

Table 1.1–3, Page 12

Introduction to MATLAB for Engineers, Third Edition William J. Palm III

Copyright © 2010. The McGraw-Hill Companies, Inc.

T.C. Adnan Menderes University Faculty of Engineering

Special Variables and Constants

Command	Description
`ans`	Temporary variable containing the most recent answer.
`eps`	Specifies the accuracy of floating point precision.
`i,j`	The imaginary unit $\sqrt{-1}$.
`Inf`	Infinity.
`NaN`	Indicates an undefined numerical result.
`pi`	The number π.

Table 1.1–4, Page 14

Introduction to MATLAB for Engineers, Third Edition William J. Palm III

Copyright © 2010. The McGraw-Hill Companies, Inc.

T.C. Adnan Menderes University Faculty of Engineering

Matrices and Arrays

- Matlab is an abbreviation for "matrix laboratory".
- While other programming languages mostly work with numbers, one at a time, Matlab is designed to operate primarily on whole matrices and arrays.
- All MATLAB **variables** are ***multidimensional arrays***, no matter what type of data.
- ***A matrix is a two-dimensional*** array often used for linear algebra.

T.C. Adnan Menderes University Faculty of Engineering

Matrices and Arrays - Array Creation

- To create an array with four elements in a single row, separate the elements with either a comma (,) or a space, and put them inside square brackets [ ]

a = [1 2 3 4] or a = [1,2,3,4]

a =

1 2 3 4

T.C. Adnan Menderes University Faculty of Engineering

Matrices and Arrays - Array Creation

- To create an array with four elements in a single row, separate the elements with either a comma (,) or a space, and put them inside square brackets [ ]

a = [1 2 3 4] or a = [1,2,3,4]

```
a =
    1    2    3    4
```

- To create a matrix that has **multiple rows**, separate the rows with semicolons.

a = [1 2 3; 4 5 6; 7 8 10]

```
a =
     1    2    3
     4    5    6
     7    8   10
```

T.C. Adnan Menderes University Faculty of Engineering

Matrices and Arrays - Array Creation

❖ To create an array with four elements in a single row, separate the elements with either a comma (,) or a space, and put them inside square brackets [ ]

a = [1 2 3 4] or a = [1,2,3,4]

```
a =
     1     2     3     4
```

❖ To create a matrix that has **multiple rows**, separate the rows with semicolons.

a = [1 2 3; 4 5 6; 7 8 10]

```
a =
     1     2     3
     4     5     6
     7     8    10
```

❖ You can also create a matrix using a function, such as ones, zeros, or rand. For example, create a 3-by-1 column vector of zeros.

z = zeros(3,1)

```
z =
     0
     0
     0
```

T.C. Adnan Menderes University Faculty of Engineering

Matrices and Arrays - Array Operations

- Matlab allows you to process all of the values in a matrix using a single arithmetic operator or function.

a + 10

ans =

11	**12**	**13**
14	**15**	**16**
17	**18**	**20**

sin(a)

ans =

0.8415	**0.9093**	**0.1411**
-0.7568	**-0.9589**	**-0.2794**
0.6570	**0.9894**	**-0.5440**

- To transpose a matrix, use a single quote (').

a'

ans =

1	**4**	**7**
2	**5**	**8**
3	**6**	**10**

T.C. Adnan Menderes University Faculty of Engineering

Matrices and Arrays - Array Operations

- Matrix multiplication.

p = a*inv(a)

p =

1.0000	**0**	**-0.0000**
0	**1.0000**	**0**
0	**0**	**1.0000**

T.C. Adnan Menderes University Faculty of Engineering

Matrices and Arrays - Array Operations

- Matrix multiplication.

```
p = a*inv(a)
p =
     1.0000         0   -0.0000
          0    1.0000         0
          0         0    1.0000
```

- Matlab stores numbers as floating-point values, you can display more decimal digits using the **format** command:

```
format long
p = a*inv(a)
p =
     1.000000000000000                   0   -0.000000000000000
                     0   1.000000000000000                   0
                     0                   0   0.999999999999998
```

You can reset the display to the shorter format using:

```
format short
```

T.C. Adnan Menderes University Faculty of Engineering

Matrices and Arrays - Array Operations

- You should know that **format** affects only the display of numbers, not the way Matlab computes or saves them.
- To perform **element-wise** multiplication rather than matrix multiplication, use the **.*** operator:

```
p = a.*a
p=
     1     4     9
    16    25    36
    49    64   100
```

- The matrix operators for multiplication, division, and power each have a corresponding array operator that operates element-wise, for example, raise each element of a to the third power:

```
a.^3
ans =  1     8    27
      64   125   216
     343   512  1000
```

T.C. Adnan Menderes University Faculty of Engineering

Matrices and Arrays - Arrays Concatenation

- ***Concatenation*** is the process of joining arrays to make larger ones. In fact, you made your first array by concatenating its individual elements. The pair of square brackets [] is the concatenation operator.

```
>> A = [a,a]

A =

     1     2     3     1     2     3
     4     5     6     4     5     6
     7     8    10     7     8    10
```

Concatenating arrays next to one another using commas is called ***horizontal*** **concatenation**. Each array must have the same number of rows.

T.C. Adnan Menderes University Faculty of Engineering

Matrices and Arrays - Arrays Concatenation

- ***Concatenation*** is the process of joining arrays to make larger ones. In fact, you made your first array by concatenating its individual elements. The pair of square brackets [] is the concatenation operator.

```
>> A = [a,a]
A =
     1     2     3     1     2     3
     4     5     6     4     5     6
     7     8    10     7     8    10
```

Concatenating arrays next to one another using commas is called ***horizontal*** **concatenation**. Each array must have the same number of rows.

- Similarly, when the arrays have the same number of columns, you can concatenate them ***vertically*** using semicolons.

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

T.C. Adnan Menderes University Faculty of Engineering

Matrices and Arrays- Array Indexing

- Every variable in Matlab is an array that can hold many numbers.
- When you want to access selected elements of an array, use **indexing**.
  For example, consider the 4-by-4 magic square A:

A = magic(4)

A =

16	**2**	**3**	**13**
5	**11**	**10**	**8**
9	**7**	**6**	**12**
4	**14**	**15**	**1**

T.C. Adnan Menderes University Faculty of Engineering

Matrices and Arrays- Array Indexing

- Every variable in Matlab is an array that can hold many numbers.
- When you want to access selected elements of an array, use **indexing**.
  For example, consider the 4-by-4 magic square A:

A = magic(4)

A =

16	**2**	**3**	**13**
5	**11**	**10**	**8**
9	**7**	**6**	**12**
4	**14**	**15**	**1**

- There are 2 ways to refer to a particular element in an array.

a) Normal Indexing: using row and column subscripts, such as

A(4,2)

ans =

14

b) linear indexing: using a single subscript that traverses down each column in order such as

A(8)

ans =

14

T.C. Adnan Menderes University Faculty of Engineering

Matrices and Arrays- Array Indexing

- If you try to refer to elements outside an array on the right side of an assignment statement, Matlab throws an error.

test = A(4,5)

Attempted to access A(4,5); index out of bounds because size(A)=[4,4].

T.C. Adnan Menderes University Faculty of Engineering

Matrices and Arrays- Array Indexing

- If you try to refer to elements outside an array on the right side of an assignment statement, Matlab throws an error.

test = A(4,5)

Attempted to access A(4,5); index out of bounds because size(A)=[4,4].

You can increase the size of an array to accommodate the newcomers, by this way

A(4,5) = 17

A =

16	2	3	13	0
5	11	10	8	0
9	7	6	12	0
4	14	15	1	17

T.C. Adnan Menderes University Faculty of Engineering

Matrices and Arrays- Array Indexing

- If you try to refer to elements outside an array on the right side of an assignment statement, Matlab throws an error.

test = A(4,5)

Attempted to access A(4,5); index out of bounds because size(A)=[4,4].

You can increase the size of an array to accommodate the newcomers, by this way

A(4,5) = 17

```
A =
    16     2     3    13     0
     5    11    10     8     0
     9     7     6    12     0
     4    14    15     1    17
```

- To refer to multiple elements of an array, use the colon operator, which allows you to specify a range of the form start:end.

  For example, list the elements in the first three rows and the second column of A:

A(1:3,2)

```
ans =
     2
    11
     7
```

T.C. Adnan Menderes University Faculty of Engineering

Matrices and Arrays - Complex Numbers

- Complex numbers have both real and imaginary parts, where the imaginary unit is the square root of –1.

```
sqrt(-1)
ans =
        0.0000 + 1.0000i
```

T.C. Adnan Menderes University Faculty of Engineering

Matrices and Arrays - Complex Numbers

- Complex numbers have both real and imaginary parts, where the imaginary unit is the square root of –1.

```
sqrt(-1)
ans =
        0.0000 + 1.0000i
```

- To represent the imaginary part of complex numbers, use either i or j.

```
c = [3+4i, 4+3j; -i, 10j]
c =
   3.0000 + 4.0000i   4.0000 + 3.0000i
   0.0000 - 1.0000i   0.0000 +10.0000i
```

T.C. Adnan Menderes University Faculty of Engineering

Workspace Variables

- The workspace contains variables that you create within or import into Matlab from data files or other programs.
- For example, these statements create variables **A** and **B** in the workspace.

  A = magic(4);

  B = rand(3,5,2);

T.C. Adnan Menderes University Faculty of Engineering

Workspace Variables

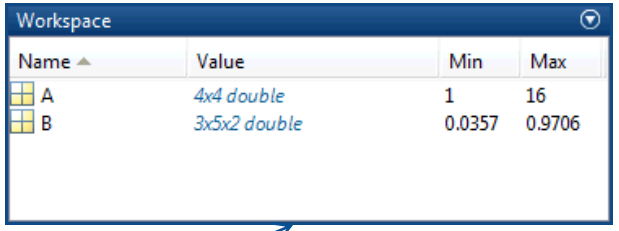
- The workspace contains variables that you create within or import into Matlab from data files or other programs.
- For example, these statements create variables **A** and **B** in the workspace.

  A = magic(4);

  B = rand(3,5,2);
- You can view the contents of the workspace using whos.

  whos

Name	Size	Bytes	Class	Attributes
A	4x4	128	double	
B	3x5x2	240	double	



- The variables also appear in the workspace panel on the desktop

T.C. Adnan Menderes University Faculty of Engineering

Workspace Variables

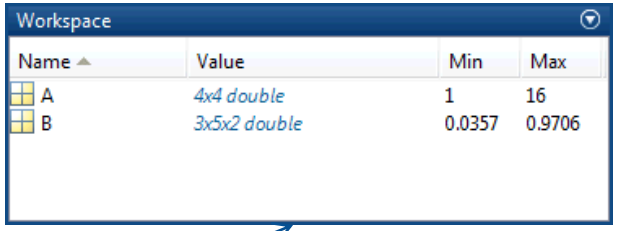
- The workspace contains variables that you create within or import into Matlab from data files or other programs.
- For example, these statements create variables **A** and **B** in the workspace.

  A = magic(4);

  B = rand(3,5,2);
- You can view the contents of the workspace using whos.

  whos

Name	Size	Bytes	Class	Attributes
A	4x4	128	double	
B	3x5x2	240	double	



- The variables also appear in the workspace panel on the desktop
- Workspace variables do not persist after you exit Matlab,but you can save your data for later use using the **save** command

  save myfile.mat
- Saving preserves the workspace in your current working folder in a compressed file with a **.mat** extension, called a **MAT-file.**
- To clear all the variables from the workspace, use the **clear** command.
- Restore data from a MAT-file into the workspace using the **load** command.

  load myfile.mat

T.C. Adnan Menderes University Faculty of Engineering

Character Strings

- A ***character string*** is a sequence of any number of characters enclosed in single quotes, you can assign a string to a variable:

  myText = 'Hello, world'

  myText =

  Hello, world

T.C. Adnan Menderes University Faculty of Engineering

Character Strings

- A ***character string*** is a sequence of any number of characters enclosed in single quotes, you can assign a string to a variable:

  myText = 'Hello, world'

  myText =

  Hello, world

- If the text includes a single quote, use two single quotes within the definition.

  otherText = 'You''re right'

  otherText =

  You're right

T.C. Adnan Menderes University Faculty of Engineering

Character Strings

- A ***character string*** is a sequence of any number of characters enclosed in single quotes, you can assign a string to a variable:

```
myText = 'Hello, world'
myText  =
            Hello, world
```

- If the text includes a single quote, use two single quotes within the definition.

```
otherText = 'You''re right'
otherText =
             You're right
```

- myText and otherText are arrays, like all Matlab variables. Their *class* or data type is *char*, which is short for *character*.

```
whos myText
Name        Size            Bytes  Class    Attributes
myText      1x12               24  char
```

T.C. Adnan Menderes University Faculty of Engineering

Character Strings

- A ***character string*** is a sequence of any number of characters enclosed in single quotes, you can assign a string to a variable:

```
myText = 'Hello, world'
myText  =
            Hello, world
```

- If the text includes a single quote, use two single quotes within the definition.

```
otherText = 'You''re right'
otherText =
             You're right
```

- myText and otherText are arrays, like all Matlab variables. Their *class* or data type is *char*, which is short for *character*.

```
whos myText
Name        Size            Bytes    Class    Attributes
myText      1x12               24    char
```

- You can concatenate strings with square brackets, just as you concatenate numeric arrays.

```
longText = [myText,' - ',otherText]
longText =
            Hello, world - You're right
```

T.C. Adnan Menderes University Faculty of Engineering

Character Strings

- A ***character string*** is a sequence of any number of characters enclosed in single quotes, you can assign a string to a variable:

```
myText = 'Hello, world'
myText  =
            Hello, world
```

- If the text includes a single quote, use two single quotes within the definition.

```
otherText = 'You''re right'
otherText =
             You're right
```

- myText and otherText are arrays, like all Matlab variables. Their *class* or data type is *char*, which is short for *character*.

```
whos myText
Name        Size            Bytes  Class    Attributes
myText      1x12               24  char
```

- You can concatenate strings with square brackets, just as you concatenate numeric arrays.

```
longText = [myText,' - ',otherText]
longText =
            Hello, world - You're right
```

- To convert numeric values to strings, use functions, such as *num2str* or *int2str*.

```
f = 71; c = (f-32)/1.8; %  (converting from Fahrenheit to Celsius)
tempText = ['Temperature is ',num2str(c),'C']
tempText = Temperature is 21.6667C
```

T.C. Adnan Menderes University Faculty of Engineering

Matlab Built in Functions

- Matlab functions are equivalent to *subroutines* or *methods* in other programming languages (implementation of flowcharts).
- We have two types of functions in matlab.
  1. Built-in functions (already written for you).
  2. User-defined Functions (you need to write them by yourself)

Elementary math built-in functions

Function	Description	Function	Description
cos(x)	Cosine	abs(x)	Absolute value
sin(x)	Sine	sign(x)	Signum function
tan(x)	Tangent	max(x)	Maximum value
acos(x)	Arc cosine	min(x)	Minimum value
asin(x)	Arc sine	ceil(x)	Round towards $+\infty$
atan(x)	Arc tangent	floor(x)	Round towards $-\infty$
exp(x)	Exponential	round(x)	Round to nearest integer
sqrt(x)	Square root	rem(x,y)	Remainder after division
log(x)	Natural logarithm	angle(x)	Phase angle
log10(x)	Common logarithm	conj(x)	Complex conjugate

http://www.slideshare.net/M3wney/matlab-hti-summer-training-courselecture2

T.C. Adnan Menderes University Faculty of Engineering

Calling Functions

- To call a function, such as max, enclose its input arguments in parentheses:

```
A = [1 3 5]; max(A)
ans =
     5
```

T.C. Adnan Menderes University Faculty of Engineering

Calling Functions

- To call a function, such as max, enclose its input arguments in parentheses:

```
A = [1 3 5]; max(A)
ans =
     5
```

- If there are multiple input arguments, separate them with **comma**:

```
B = [10 6 4]; max(A,B)
ans =
    10     6     5
```

T.C. Adnan Menderes University Faculty of Engineering

Calling Functions

- To call a function, such as max, enclose its input arguments in parentheses:

```
A = [1 3 5]; max(A)
ans =
     5
```

- If there are multiple input arguments, separate them with **comma**:

```
B = [10 6 4]; max(A,B)
ans =
    10     6     5
```

- Return output from a function by assigning it to a variable:

```
maxA = max(A);
```

- When there are multiple output arguments, enclose them in square brackets:

```
[maxA,location] = max(A);
```

T.C. Adnan Menderes University Faculty of Engineering

Calling Functions

- To call a function, such as max, enclose its input arguments in parentheses:

  A = [1 3 5]; max(A)

  ans =

  5

- If there are multiple input arguments, separate them with **comma**:

  B = [10 6 4]; max(A,B)

  ans =

  10 6 5

- Return output from a function by assigning it to a variable:

  maxA = max(A);

- When there are multiple output arguments, enclose them in square brackets:

  [maxA,location] = max(A);

- Enclose any character string input in single quotes, when you want to display it:

  disp ('hello world')

  hello world

- To call a function that does not require any inputs and does not return any outputs, type only the function name:

  clc

  The clc function clears the command window.

T.C. Adnan Menderes University Faculty of Engineering

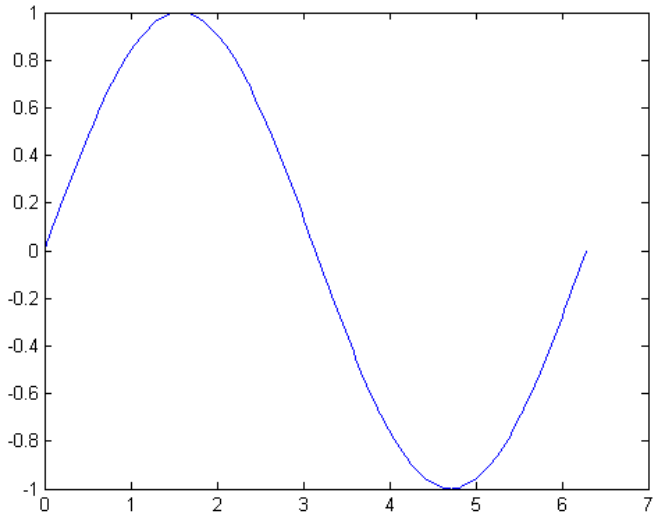
2-D Plots

- To create two-dimensional line plots, use the **plot** function. For example, plot the value of the sine function from 0 to 2π:

  x = 0:pi/100:2*pi;

  y = sin(x);

  plot(x,y)



T.C. Adnan Menderes University Faculty of Engineering

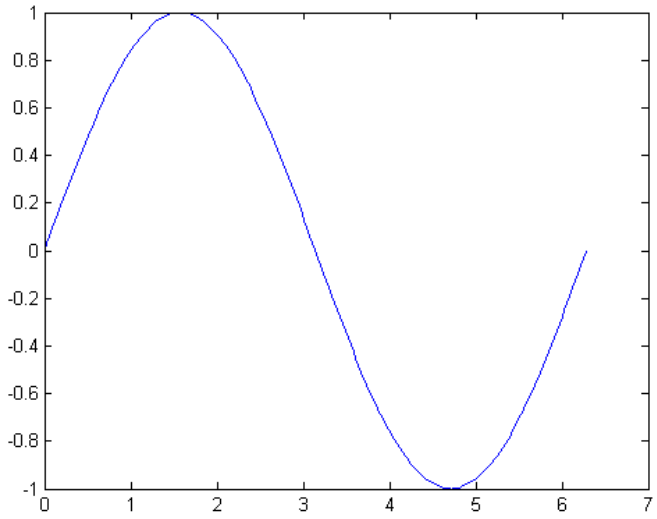
2-D Plots

- To create two-dimensional line plots, use the **plot** function. For example, plot the value of the sine function from 0 to 2π:

  x = 0:pi/100:2*pi;

  y = sin(x);

  plot(x,y)



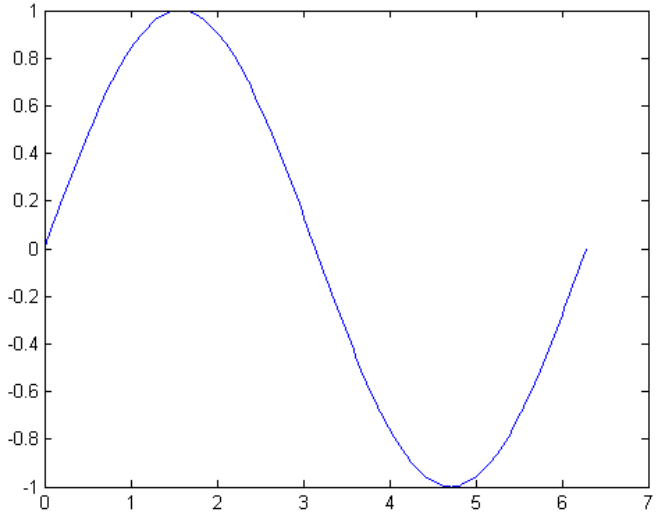
- The function domain usually has this form: X=Initial value:Step size:Final value
- If the Step size is not mentioned, that means it is equal to 1.

TC Adnan Menderes University Faculty of Engineering

2-D Plots

- To create two-dimensional line plots, use the **plot** function. For example, plot the value of the sine function from 0 to 2π:

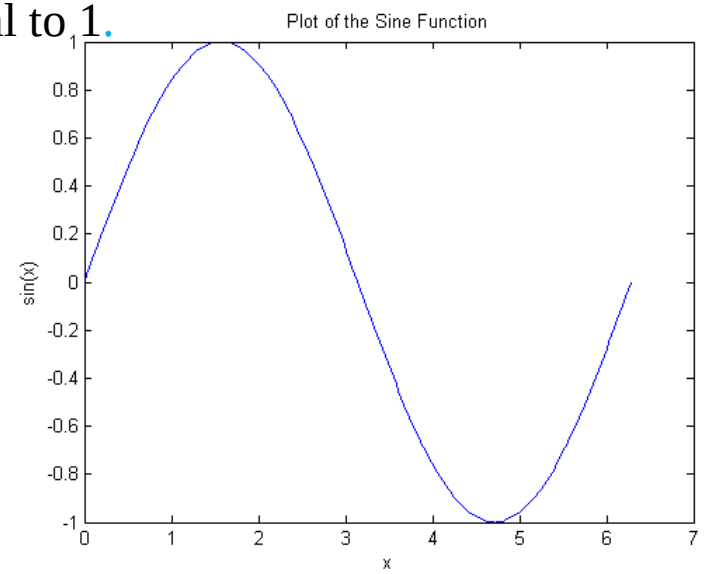
```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
```



- The function domain usually has this form: X=Initial value:Step size:Final value
- If the Step size is not mentioned, that means it is equal to 1.

- You can label the axes and add a title.

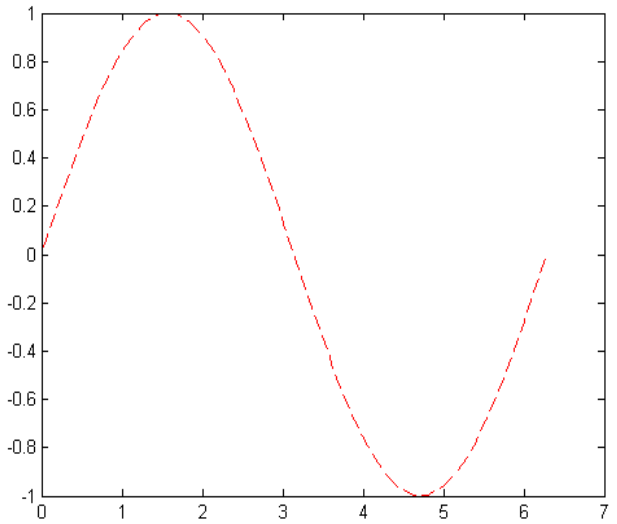
```
xlabel('x')
ylabel('sin(x)')
title('Plot of the Sine Function')
```



2-D Plots -Line Specification

- You can choose the color and the format of the drawing line by adding **a third input argument** to the plot function:

  plot(x,y,'r--')



T.C. Adnan Menderes University Faculty of Engineering

2-D Plots -Line Specification

- You can choose the color and the format of the drawing line by adding **a third input argument** to the plot function:

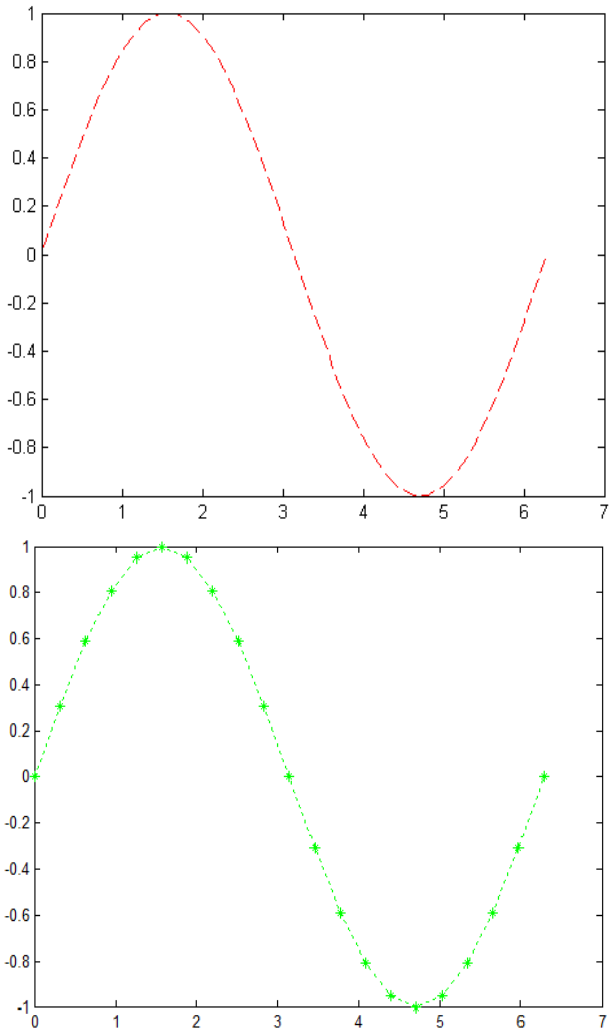
  plot(x,y,'r--')

- The '**r--**' string is a ***line specification***.

  Each specification can include **characters** for the line **color**, **style**, and **marker**.

- A **marker** is a symbol that appears at each plotted data point, such as a **+**, **o**, or *****, For example, "request a dotted green line with * markers.

  plot(x,y, 'g:*')



T.C. Adnan Menderes University Faculty of Engineering

2-D Plots -Line Specification

- You can choose the color and the format of the drawing line by adding **a third input argument** to the plot function:

  plot(x,y,'r--')

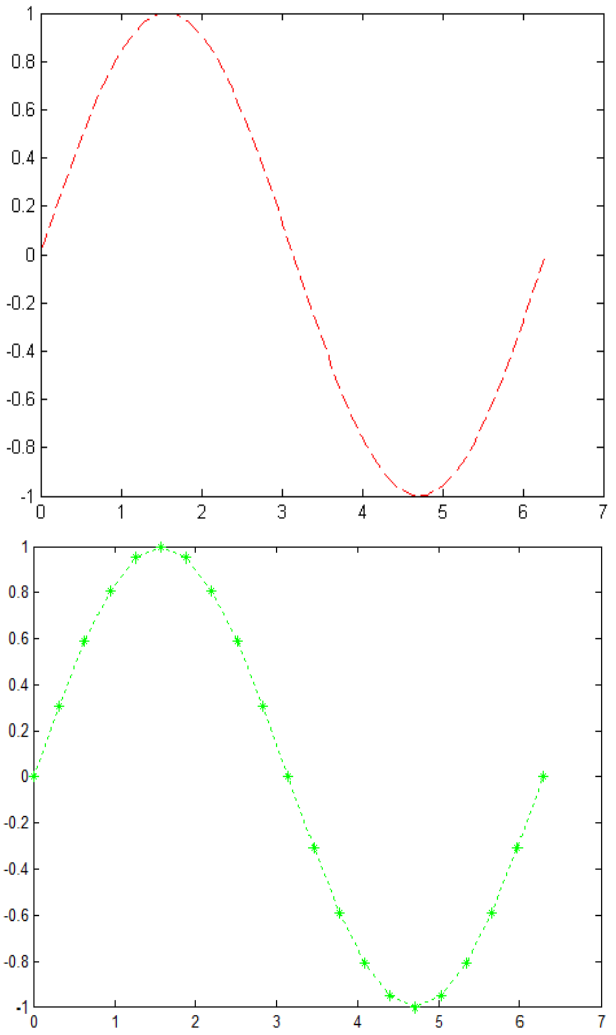
- The '**r--**' string is a ***line specification***.

  Each specification can include **characters** for the line **color**, **style**, and **marker**.

- A **marker** is a symbol that appears at each plotted data point, such as a **+**, **o**, or *****, For example, "request a dotted green line with * markers.

  plot(x,y, 'g:*')

- Notice that the titles and labels that you defined for the first plot are no longer in the current figure window.
- By default, Matlab clears the figure each time you call a plotting function, resetting the axes and other elements to prepare the new plot.
- To add plots to an existing figure, use the **hold** command.

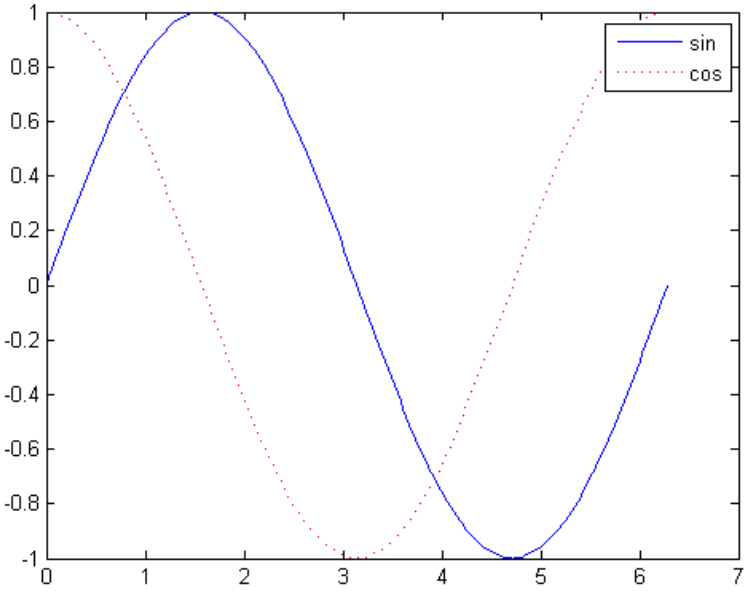


T.C. Adnan Menderes University Faculty of Engineering

2-D Plots – the Hold Command

- To add plots to an existing figure, use **hold**.

```
x = 0:pi/100:2*pi;
y = sin(x);
plot (x,y)
hold on
y2 = cos(x);
plot(x,y2,'r:')
legend('sin','cos')
```



- Until you use hold off or close the window, all plots appear in the current figure window.

T.C. Adnan Menderes University Faculty of Engineering

3-D Plots

- Three-dimensional plots typically display a surface defined by a function in two variables z = f(x,y).
- To plot z:

1.Create the domain (set of (x,y) points) using **meshgrid**.

```
[X,Y] = meshgrid(-2:.2:2);
```

2. Compute the range (set of corresponding Z points) using **the function formula**.

```
Z = X .* exp(-X.^2 - Y.^2);
```

3. Create a surface plot using surf(X,Y,Z) or mesh(X,Y,Z)

T.C. Adnan Menderes University Faculty of Engineering

3-D Plots

- Three-dimensional plots typically display a surface defined by a function in two variables z = f(x,y).

T.C. Adnan Menderes University Faculty of Engineering

3-D Plots

- Three-dimensional plots typically display a surface defined by a function in two variables z = f(x,y).
- For example,to plot Z = X .* exp(-X.^2 - Y.^2):

T.C. Adnan Menderes University Faculty of Engineering

3-D Plots

- Three-dimensional plots typically display a surface defined by a function in two variables z = f(x,y).
- For example,to plot Z = X .* exp(-X.^2 - Y.^2):

1. Create the domain (set of (x,y) points) using **meshgrid**.

```
[X,Y] = meshgrid(-2:.2:2);
```

T.C. Adnan Menderes University Faculty of Engineering

3-D Plots

- Three-dimensional plots typically display a surface defined by a function in two variables z = f(x,y).
- For example,to plot Z = X .* exp(-X.^2 - Y.^2):

1.Create the domain (set of (x,y) points) using **meshgrid**.

```
[X,Y] = meshgrid(-2:.2:2);
```

2. Compute the range (set of corresponding Z points) using **the function formula**.

```
Z = X .* exp(-X.^2 - Y.^2);
```

T.C. Adnan Menderes University Faculty of Engineering

3-D Plots

- Three-dimensional plots typically display a surface defined by a function in two variables z = f(x,y).
- For example,to plot Z = X .* exp(-X.^2 - Y.^2):

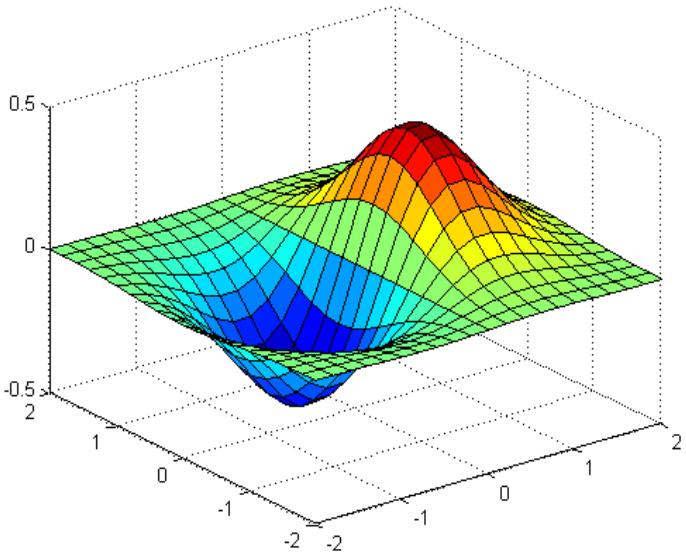
1.Create the domain (set of (x,y) points) using **meshgrid**.

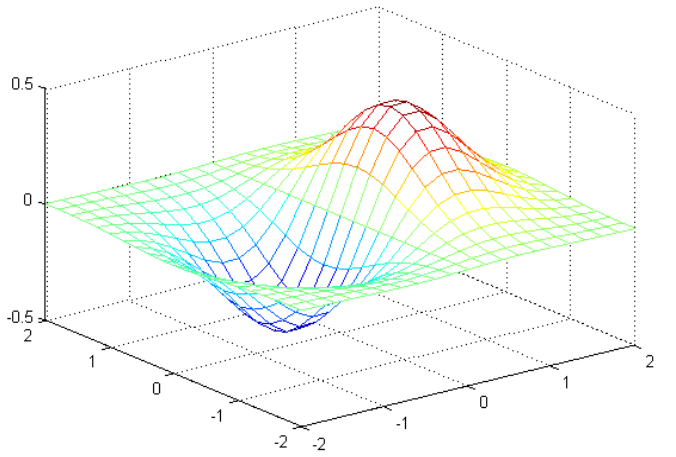
[X,Y] = meshgrid(-2:.2:2);

2. Compute the range (set of corresponding Z points) using **the function formula**.

Z = X .* exp(-X.^2 - Y.^2);

3. Create a surface plot using surf(X,Y,Z) or mesh(X,Y,Z)





T.C. Adnan Menderes University Faculty of Engineering

3-D Plots

- Three-dimensional plots typically display a surface defined by a function in two variables z = f(x,y).
- For example,to plot Z = X .* exp(-X.^2 - Y.^2):

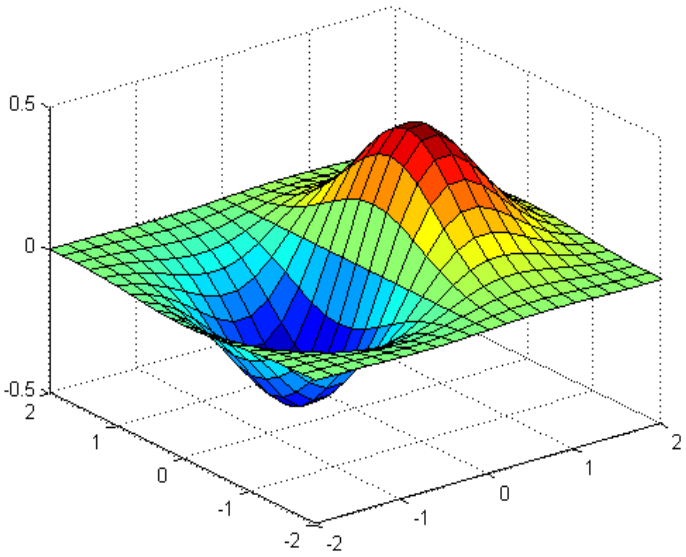
1.Create the domain (set of (x,y) points) using **meshgrid**.

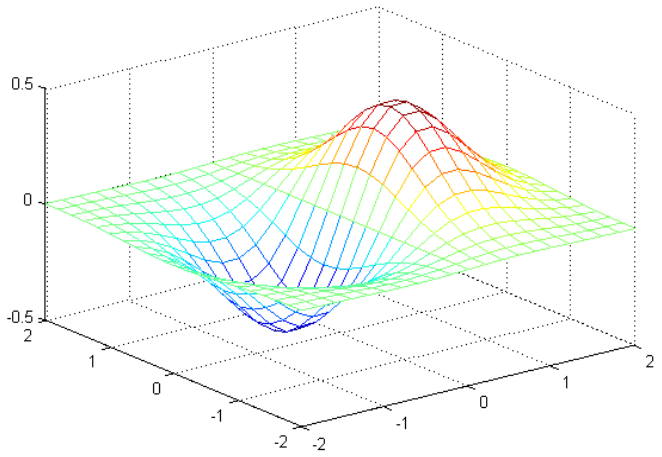
[X,Y] = meshgrid(-2:.2:2);

2. Compute the range (set of corresponding Z points) using **the function formula**.

Z = X .* exp(-X.^2 - Y.^2);

3. Create a surface plot using surf(X,Y,Z) or mesh(X,Y,Z)





- Both the **surf** function and its companion **mesh** display surfaces in three dimensions.
- **surf** displays both the connecting lines and the faces of the surface in color.
- **mesh** produces **wireframe** surfaces that color only the lines connecting the defining points.

T.C. Adnan Menderes University Faculty of Engineering

3-D Plots Subplots

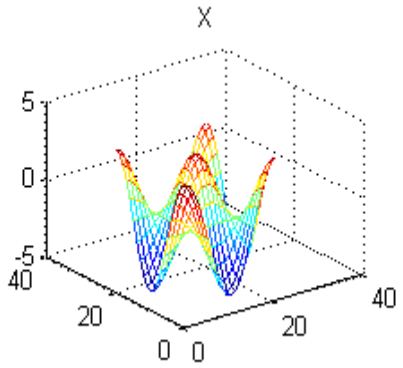
- To display multiple plots in different sub-regions of the same window, use the **subplot** function.

  For example, create four plots in a 2-by-2 grid within a figure window.

  t = 0:pi/10:2*pi;

  [X,Y,Z] = cylinder(4*cos(t));

  subplot(2,2,1); mesh(X); title('X');



- The first two inputs to the subplot function indicate the number of plots in each row and column. The third input specifies which plot is active

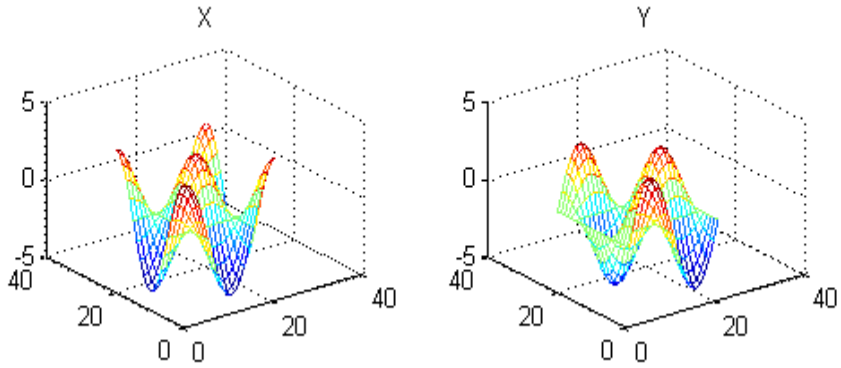
T.C. Adnan Menderes University Faculty of Engineering

3-D Plots Subplots

- To display multiple plots in different subregions of the same window, use the **subplot** function.

  For example, create four plots in a 2-by-2 grid within a figure window.

```
t = 0:pi/10:2*pi;
[X,Y,Z] = cylinder(4*cos(t));
subplot(2,2,1); mesh(X); title('X');
subplot(2,2,2); mesh(Y); title('Y');
```



- The first two inputs to the subplot function indicate the number of plots in each row and column. The third input specifies which plot is active

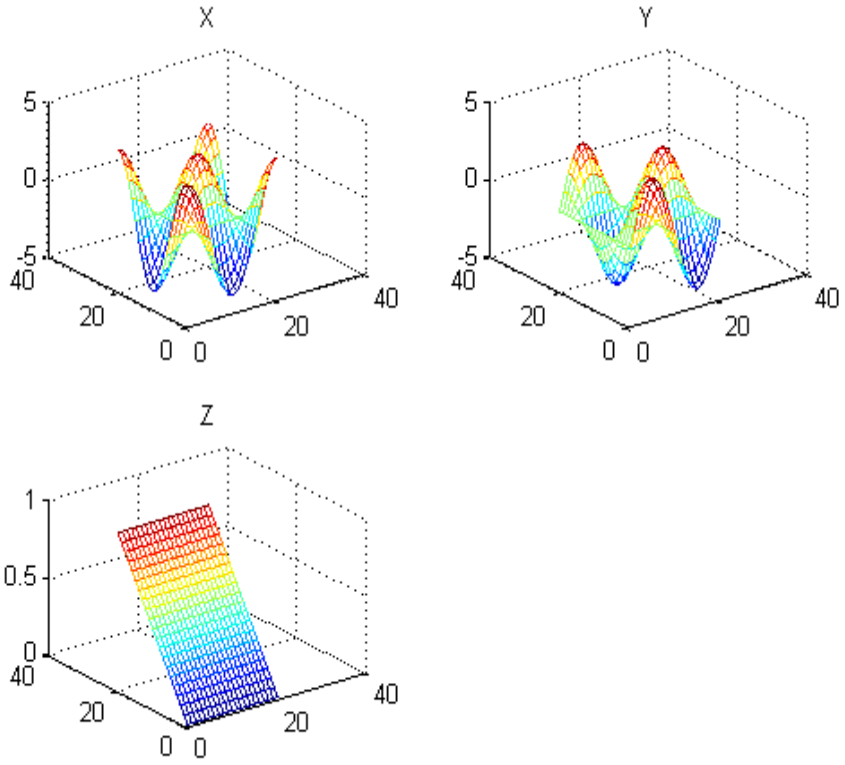
T.C. Adnan Menderes University Faculty of Engineering

3-D Plots Subplots

- To display multiple plots in different subregions of the same window, use the **subplot** function.

  For example, create four plots in a 2-by-2 grid within a figure window.

```
t = 0:pi/10:2*pi;
[X,Y,Z] = cylinder(4*cos(t));
subplot(2,2,1); mesh(X); title('X');
subplot(2,2,2); mesh(Y); title('Y');
subplot(2,2,3); mesh(Z); title('Z');
```



- The first two inputs to the subplot function indicate the number of plots in each row and column. The third input specifies which plot is active

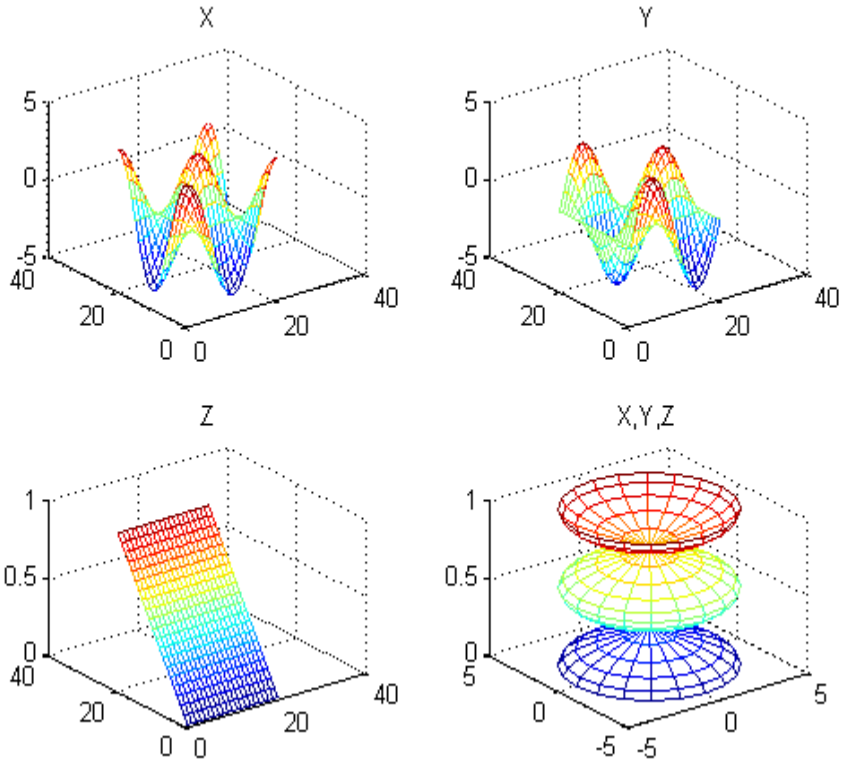
T.C. Adnan Menderes University Faculty of Engineering

3-D Plots Subplots

- To display multiple plots in different subregions of the same window, use the **subplot** function.

  For example, create four plots in a 2-by-2 grid within a figure window.

```
t = 0:pi/10:2*pi;
[X,Y,Z] = cylinder(4*cos(t));
subplot(2,2,1); mesh(X); title('X');
subplot(2,2,2); mesh(Y); title('Y');
subplot(2,2,3); mesh(Z); title('Z');
subplot(2,2,4); mesh(X,Y,Z); title('X,Y,Z');
```



- The first two inputs to the subplot function indicate the number of plots in each row and column. The third input specifies which plot is active

Examples

1. A real-valued exponential sequence can be generated by this formu $x(n) = a^n, \forall n;\ a \in \mathbb{R}$

In MATLAB, an array operator ". " is required to implement a real exponential sequence. For example, to generate x(n) = $(0.9)^n$, 0 ≤n≤ 10, we will need the following MATLAB script:

```
n=0:10;
x=(0.9).^n;
plot(n,x);
```

2. A complex-valued exponential sequence can be generated by this formula $x(n) = e^{(\sigma+j\omega_0)n}, \forall n$

For example, to generate x(n) = $e^{(2+3j)^n}$, 0≤n≤10, we will need the following MATLAB script:

```
n=0:10;
x=exp(2+3i).^n;
plot(n,x);
```

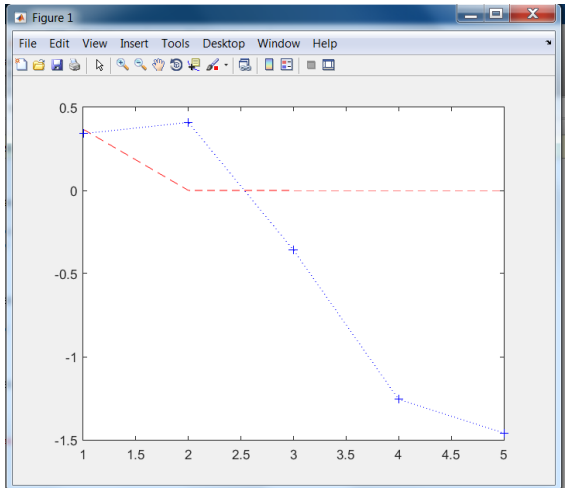
T.C. Adnan Menderes University Faculty of Engineering

Copyrights © 2011.Cengage Learning

Examples

3. Plot the two curves y=e^{-x^3} and z=sin(x)-0.5 in the interval [0 , 5] on the same figure.

```
x=1:1:5; % generate the domain of the  two functions
y1=exp(-(x.^3)); % generate the range of the first function
plot(x,y1,'r--');% plot the first function
hold on;% keep the current figure
y2=sin(x)-0.5;% generate the range of the second function
plot(x,y2,'b:+');% plot the second function on the same figure
```

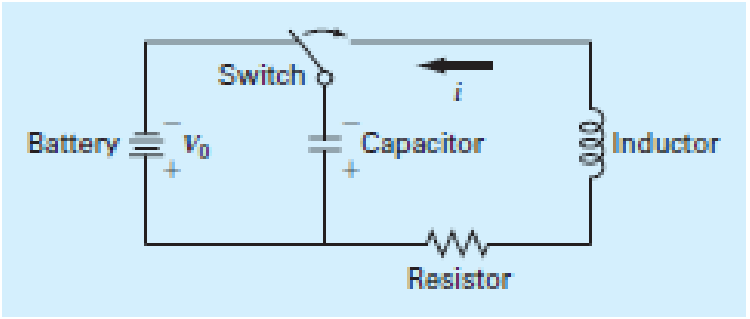


T.C. Adnan Menderes University Faculty of Engineering

Exercise

3. A simple electric circuit consisting of a resistor, a capacitor, and an inductor is depicted in the given Figure. The charge on the capacitor q(t) as a function of time can be computed where t = time, $q0$ = the initial charge, R = the resistance, L = inductance, and C = capacitance. Use MATLAB to generate a plot of this function from $t = 0$ to 0.8, given that $q0 = 10$, $R = 60$, $L = 9$, and $C = 0.00005$.

$$q(t) = q_0 e^{-Rt/(2L)} \cos\left[\sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2}\, t\right]$$



T.C. Adnan Menderes University Faculty of Engineering

Copyright © 2011. The McGraw-Hill Companies, Inc.

References

- http://www.mathworks.com/help/matlab/getting-started-with-matlab.html
- William J. Palm III, “Introduction to MATLAB for Engineers”, 3rd Edition, McGraw-Hil, 2010.

T.C. Adnan Menderes University Faculty of Engineering