Holy Angel University

Electrical Engineering Program

NUMETHODS Laboratory Activity no. 3: **Two-Dimensional Plots**

OBJECTIVES:

1. Create two-dimensional plots in MATLAB
2. Customize plots
3. Add titles, axis labels, and annotations

1. The `plot` Command

Plots are a very useful tool for presenting information. One of the most powerful features of MATLAB is that it allows engineers to easily create plots to visualize engineering data.

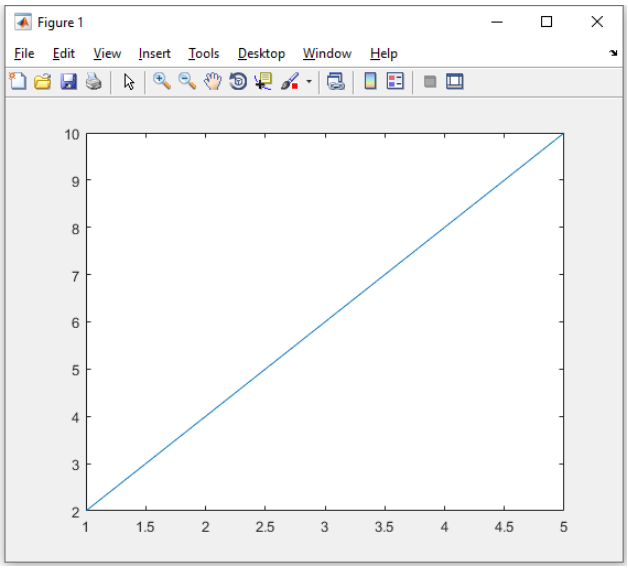
The *plot* command is used to create a two-dimensional plot.
Syntax:

```
plot(x,y)
```

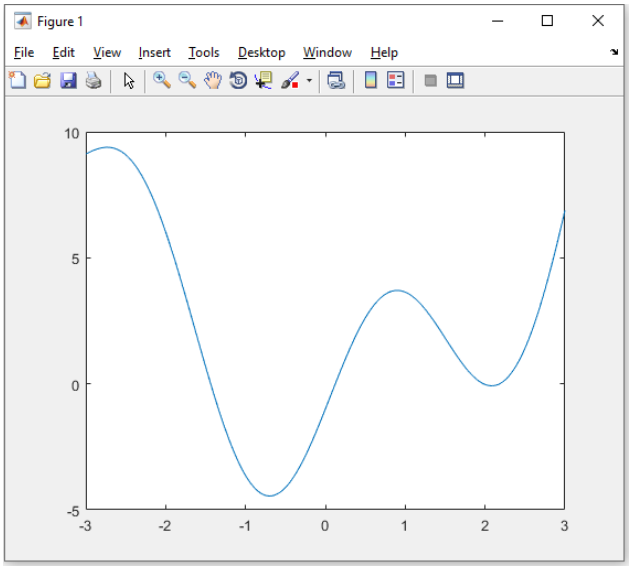
where `x` and `y` are vectors with the same number of elements.

Example: Type the following and execute.

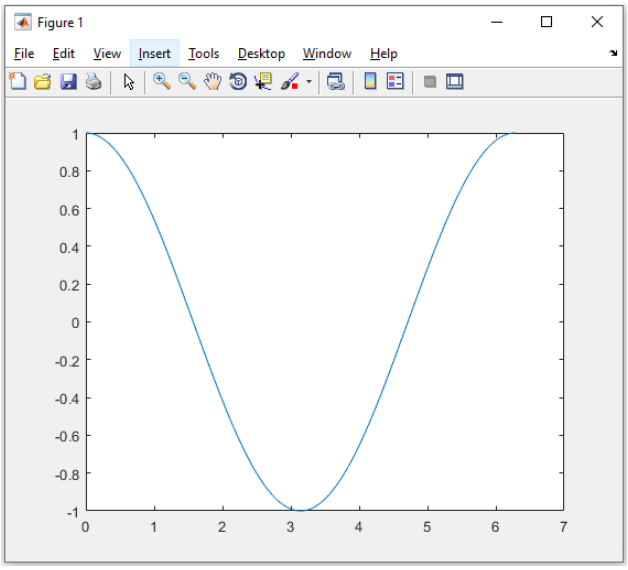
```
>> x = [1 2 3 4 5];
>> y = [2 4 6 8 10];
>> plot(x,y)
```




```
>> x = [-3:0.01:3];
>> y = x.^2 + 4*sin(2*x) - 1;
>> plot(x,y)
```




```
>> t = [0:pi/100:2*pi];
>> v = cos(t);
>> plot(t,v)
```



1.2 Line Specifiers: Line Styles, Colors, and Markers

Syntax:

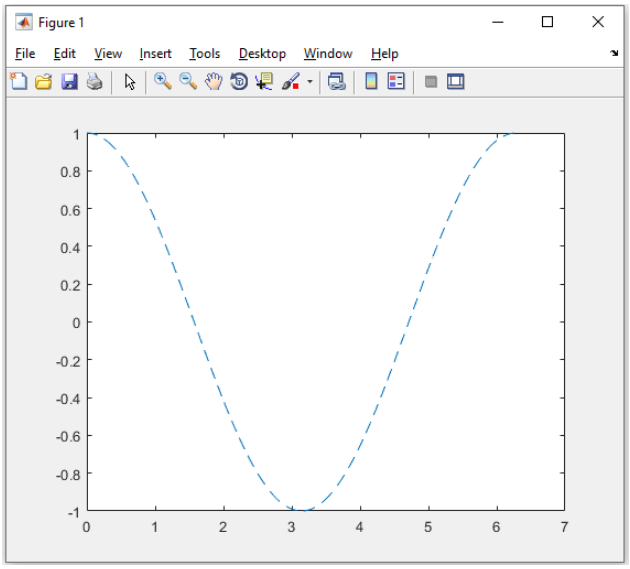
`plot(x,y,` 'line specifiers')

The line specifiers are optional and can be used to define the *style* and *color* of the line and the type of *markers*.

Line Style	Specifier
solid (default)	-
dashed	--
dotted	:
dash-dot	-.

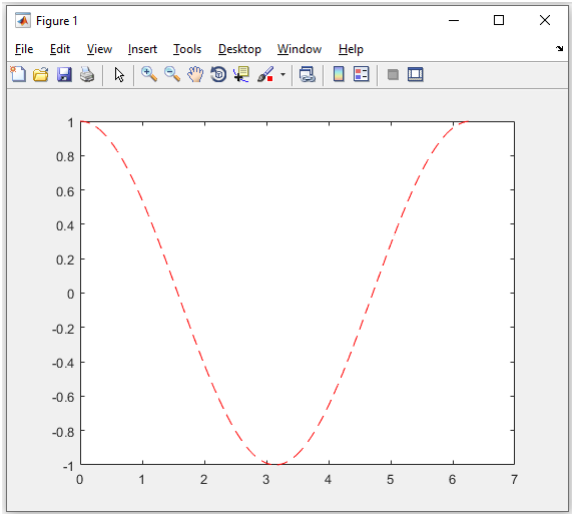
Continuing the previous example

```
>> plot(t,v,'--')
```



Line Color	Specifier
red	r
green	g
blue	b
cyan	c
magenta	m
yellow	y
black	k
white	w

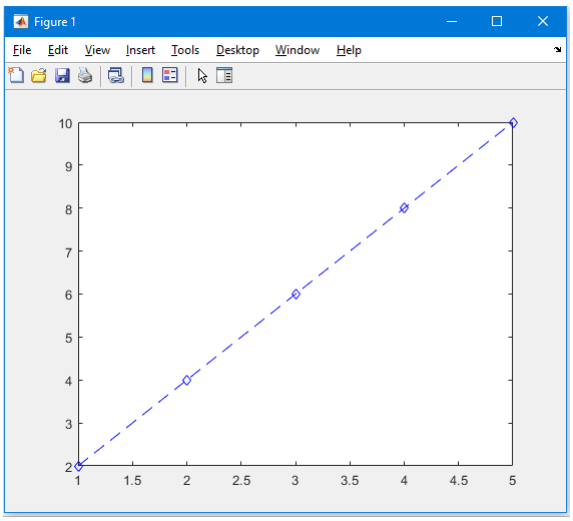
```
>> plot(t,v,'--r')
```



Marker Type	Specifier
plus sign	+
circle	o
asterisk	*
point	.
cross	x

square	s
diamond	d
five-pointed star	p
six-pointed star	h
triangle (pointed up)	^
triangle (pointed down)	v
triangle (pointed left)	<
triangle (pointed right)	>

```
>> x = [1 2 3 4 5];
>> y = [2 4 6 8 10];
>> plot(x,y,'--bd')
```



Notes on the use of line specifiers:

- The specifiers are typed inside the plot command as strings.
- Within the string the specifiers can be typed in any order.
- The specifiers are optional. This means that none, one, two, or all three types can be included in a command.

1.3 Property Name and Property Value

Properties are optional and can be used to specify the thickness of the line, the size of the marker, and the colors of the marker's edge line and fill. The Property Name is typed as a string, followed by a comma and a value for the property, all inside the plot command.

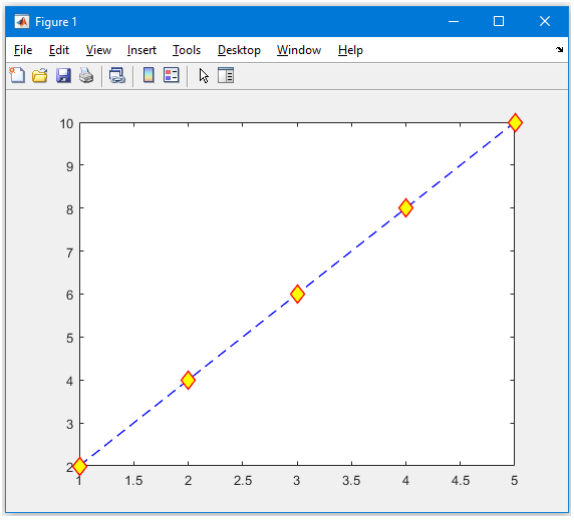
Syntax:

`plot(x,y,` 'line specifiers', 'PropertyName', PropertyValue)

Property name	Description	Possible property values
`LineWidth` `(or linewidth)`	Specifies the width of the line.	A number in units of points (default 0.5).
`MarkerSize` `(or markersize)`	Specifies the size of the marker.	A number in units of points.
`MarkerEdgeColor` `(or markeredgecolor)`	Specifies the color of the marker, or the color of the edge line for filled markers.	Color specifiers from the table above, typed as a string.

`MarkerFaceColor` `(or markerfacecolor)`	Specifies the color of the filling for filled markers.	Color specifiers from the table above, typed as a string.

```
>> plot(x,y,'--bd', 'linewidth', 1, 'markeredgecolor', 'r','markerfacecolor', 'y', 'markersize',10)
```



2. The `fplot` Command

The `fplot` command plots a function with the form $y = f(x)$ between specified limits.

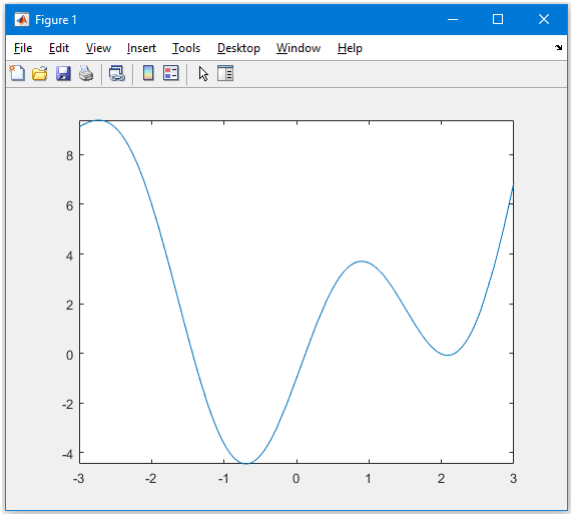
Syntax:

```
fplot(function, limits, 'line specifiers')
```

where:

- The `function` should be typed in the form of an anonymous function. The form of an anonymous function is: `@(x) f(x)`. For example, if the function $f(x) = x^2 + 4\sin 2x - 1$ is to be plotted, it is typed as: `f = @(x) x.^2 + 4*sin(2*x) - 1`.
- The limits argument is a vector with two elements that specify the domain of x [`xmin, xmax`], or a vector with four elements that specifies the domain of x and the limits of the y-axis [`xmin, xmax, ymin, ymax`].
- The line specifiers are the same as in the `plot` command.

```
>> f = @(x) x.^2 + 4*sin(2*x) - 1;
>> fplot(f, [-3,3])
```



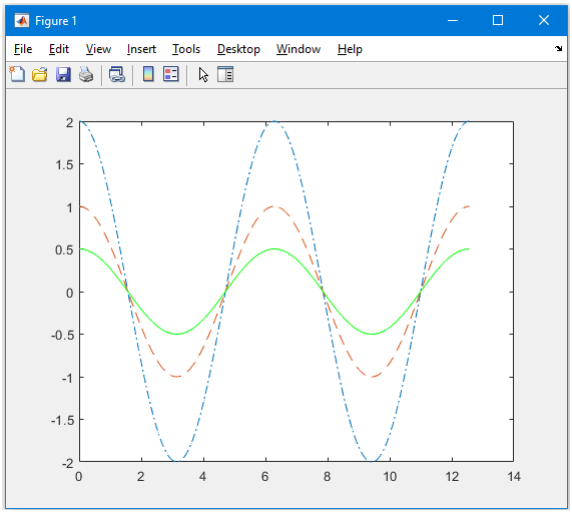
3. Plotting Multiple Graphs in the Same Plot

In many situations, there is a need to make several graphs in the same plot. There are three methods to plot multiple graphs in one figure. One is by using the `plot` command, the second is by using the `hold on` and `hold off` commands, and the third is by using the `line` command.

3.1 Using the `plot` Command

Two or more graphs can be created in the same plot by typing pairs of vectors inside the `plot` command.

```
>> wt = 0:pi/100:4*pi;
>> y1 = 2*cos(wt);
>> y2 = cos(wt);
>> y3 = 0.5*cos(wt);
>> plot(wt,y1,'-.',wt,y2,'--',wt,y3,'g')
```



3.2 Using the `hold on` and `hold off` Commands

To plot several graphs using the `hold on` and `hold off` commands, one graph is plotted first with the `plot` command. Then the `hold on` command is typed. Additional graphs can be added with `plot` commands that are typed next. The `hold off` command stops the process.

```
>> plot(wt,y1,'-.')
>> hold on
>> plot(wt,y2,'--')
>> plot(wt,y3,'g')
>> hold off
```

3.3 Using the `line` Command

With the `line` command additional graphs (lines) can be added to a plot that already exists.

Syntax:

```
line(x,y,'PropertyName',PropertyValue)
```

where: Properties with values that can be used to specify the line style, color, and width, marker type, size, and edge and fill colors.

```
>> plot(wt,y1, 'LineStyle', '-.', 'Color', 'b')
>> line(wt,y2, 'linestyle', '--', 'color', 'r')
>> line(wt,y3, 'linestyle', '-', 'color', 'g')
```

4. Formatting a Plot

The formatting commands are entered after the `plot` or the `fplot` command.

The `xlabel` and `ylabel` commands:

Labels can be placed next to the axes with the `xlabel` and `ylabel` commands.

Syntax:

```
xlabel('text as string')
ylabel('text as string')
```

The `title` command:

A title can be added to the plot. The text is placed at the top of the figure as a title.

Syntax:

```
title('text as string')
```

The `text` command:

A text label can be placed in the plot with the `text` or `gtext` commands:

Syntax:

```
text(xcoordinate,ycoordinate, 'text as string')
gtext('text as string')
```

The `text` command places the text in the figure such that the first character is positioned at the point with the coordinates `x`, `y` (according to the axes of the figure). The `gtext` command places the text at a position specified by the user. When the command is executed, the Figure Window opens and the user specifies the position with the mouse.

The `legend` command:

The `legend` command places a legend on the plot. The legend shows a sample of the line type of each graph that is plotted, and places a label, specified by the user, beside the line sample.

Syntax:

```
legend('string1', 'string2',…)
```

The strings are the labels that are placed next to the line sample. Their order corresponds to the order in which the graphs were created.

Formatting the text within the `xlabel, ylabel, title, text` and `legend` commands:

The text in the string that is included in the command and is displayed when the command is executed can be formatted. The formatting can be used to define the font, size, position (superscript, subscript), style (italic, bold, etc.), and color of the characters, the color of the background, and many other details of the display. The formatting can be done either by adding modifiers inside the string, or by adding to the command optional `PropertyName` and `PropertyValue` arguments following the string.

The modifiers are characters that are inserted within the string. Some of the modifiers that can be added are:

Modifier	**Effect**
`\bf`	bold font
`\it`	italic style
`\rm`	normal font
`\fontname{fontname}`	specified font is used
`\fontsize{fontsize}`	specified font size is used

These modifiers affect the text from the point at which they are inserted until the end of the string. It is also possible to have the modifiers applied to only a section of the string by typing the modifier and the text to be affected inside braces { }.

Subscript and superscript:
A single character can be displayed as a subscript or a superscript by typing _ (the underscore character) or ^ (caret symbol) in front of the character, respectively. Several consecutive characters can be displayed as a subscript or a superscript by typing the characters inside braces { } following the underscore or the caret.

_{xxx}
^{xxx}

Greek characters and mathematical symbols

Special Greek and mathematical symbols may also be used in text strings. They are created by embedding *escape sequences* into the text string.

Character Sequence	Symbol	Character Sequence	Symbol	Character Sequence	Symbol
\alpha	α			\int	$\int$
\beta	β			\cong	$\cong$
\gamma	γ	\Gamma	Γ	\sim	$\sim$
\delta	δ	\Delta	Δ	\infty	∞
\epsilon	ϵ			\pm	$\pm$
\eta	η			\leq	$\leq$
\theta	θ			\geq	$\geq$
\lambda	λ	\Lambda	Λ	\neq	$\neq$
\mu	μ			\propto	$\propto$
\nu	ν			\div	$\div$
\pi	π	\Pi	Π	\circ	$\circ$
\phi	ϕ			\leftrightarrow	$\leftrightarrow$
\rho	ρ			\leftarrow	$\leftarrow$
\sigma	σ	\Sigma	Σ	\rightarrow	$\rightarrow$
\tau	τ			\uparrow	$\uparrow$
\omega	ω	\Omega	Ω	\downarrow	$\downarrow$

The `axis` command:

When the `plot(x,y)` command is executed, MATLAB creates axes with limits that are based on the minimum and maximum values of the elements of `x` and `y`. The `axis` command can be used to change the range and the appearance of the axes.

Syntax:

```
axis([xmin,xmax,ymin,ymax])   -Sets the limits of both the x and y axes
axis equal                    -Sets the same scale for both axes
axis square                   -Sets the axes region to be square
axis tight                    -Sets the axis limits to the range of the data
```

The `grid` command:

```
grid on          -Adds grid lines to the plot
grid off         -Removes grid lines from the plot
```

```
>> wt = 0:pi/100:4*pi;
>> y1 = 2*cos(wt);
>> y2 = cos(wt);
>> y3 = 0.5*cos(wt);
>> plot(wt,y1,'-.',wt,y2,'--',wt,y3,'g')
>> xlabel('0 \leq \omegat \leq 4\pi')
>> ylabel('Cosine functions')
>> legend('2*cos(\omegat)','cos(\omegat)', '0.5*cos(\omegat)')
>> title('\bfTypical example of multiple plots')
```

```
>> axis([0 4*pi -3 3])
>> grid on
>> text(4.5, 0.5, '\bfPlot of \itCosines')
```