Plotting Graphs

MATLAB has many commands that can be used for creating various kinds of 2D plots.

Easy plots: The simplest built-in function for plotting an explicit function is `ezplot` command. For example, we want to plot a parabola on the interval `[-1,1]`

$$f(x)=x^2,\ -1\le x\le 1.$$

We can simply use the following command:

```
ezplot('x^2',[-1,1])
```

The first input of `ezplot` command is a string describing the function. The second input (which is optional) is the interval of interest in the graph. Note that the function description is not necessarily a function in variable `x`. We can use any variable we like:

```
ezplot('sin(b)',[-2,3])
```

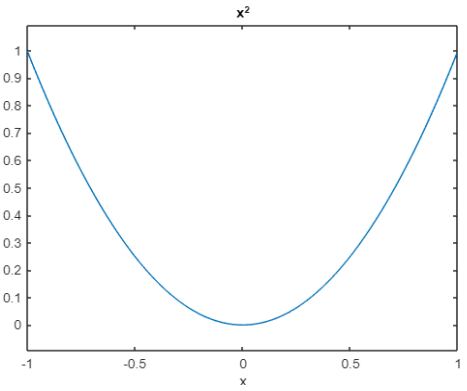
If the function definition involves some constant parameter, write it explicitly:

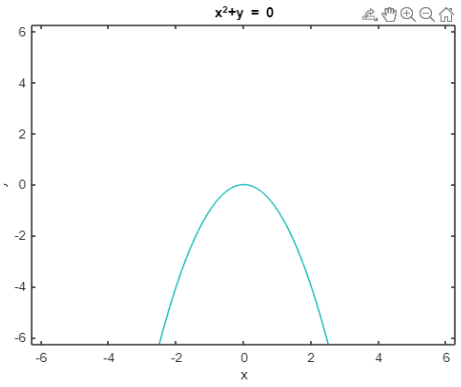
```
ezplot('x^2+2')
```

In this case, if we assign a value 2 to a variable first, says `y=2,` when executing:

```
y = 2;
ezplot('x^2+y')
```

MATLAB will interpret y as a variable (not a value of 2). Hence, it will create a contour plot of the function $x^2+y=0$





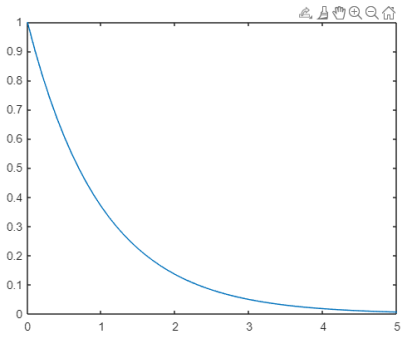
Plot command

The plot command is used to create a two-dimensional plot. The simplest form of the command is:

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

where x and y are each a vector. Both vector **must** have the same number of elements. For example:

```
x = 0:0.1:5;
y = exp(-x);
plot(x,y)
```



Once the plot command is executed, the figure Window opens and the plot is displayed. The plot appears on the screen in blue which is the default line color.
The plot command has additional arguments that can be used to specify the color and style of the line and the color and type of markers, if any are desired. With these options the command has the form:

```
plot(x,y,'linespec','PropertyName',PropertyValue)
```

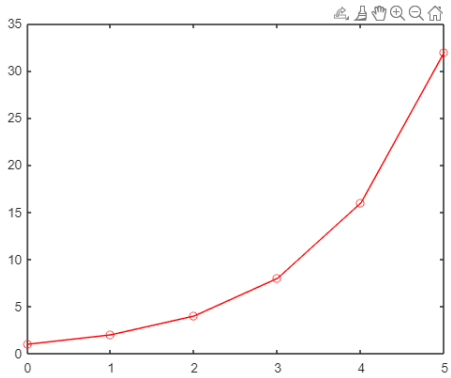
Line Specifiers (linespec): Line specifiers are optional and can be used to define the style and color of the line and the type of markers (if markers are desired).

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

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

Marker Type	Specifier
plus sign	+
circle	o
asterisk	*
point	.
cross	x
triangle (pointed up)	^
triangle (pointed down)	v
square	s
diamond	d
five-pointed star	p
sixed-pointed star	h
triangle (pointed left)	<
triangle (pointed right)	>

The specifiers are typed inside the plot command as strings. Within the string, the specifiers can be typed in any order and the specifiers are optional.



For example:

```
x=0:5;y=2.^x; plot(x,y,'-or')
```

This command creates a plot with solid red line and the marker is a circle.

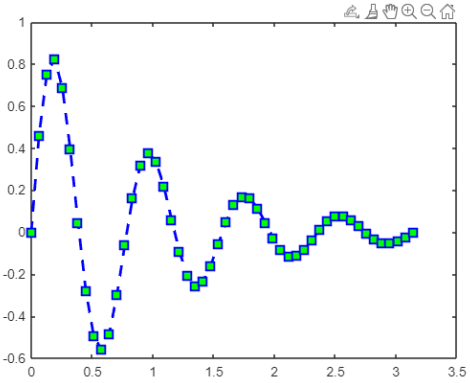
Property Name and PropertyValue: Properties are optional and can be used to specify the thickness of the line, the size of the marker, and the colors of the marker's edge line and fill. The Property Name is typed as a string, followed by a comma and a value for the property, all inside the plot command.

Property Name	Description
linewidth	the width of the line (default 0.5, possible values are 1,2,3,...)
markersize	the size of the marker (e.g., 5,6,...)
markeredgecolor	the color of the edge line for filled marker (e.g., r, b)
markerfacecolor	the color of the filling for filled markers (e.g, r, b)

For example, the command:

```
x=linspace(0,pi,50);
y=exp(-x).*sin(8*x);
plot(x,y,'--sb', 'linewidth', 2,
'markersize',8, 'markerfacecolor', 'g')
```

creates a plot with a blue dashed line and squares as markers. The linewidth is 2 points and the size of the square markers is 8 points. The marker has green filling.



Formatting a plot

Plots can be formatted by using MATLAB command that follow the `plot` or `fplot` commands, or interactively by using the plot editor in the Figure Window. Here we will explain the the first method which is more useful since a formatted plot can be created automatically every time the program is executed.

The xlabel and ylabel command

`xlabel` and `ylabel` create description on the x- and y-axes, respectively after we have plotted a graph. The usages are:

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

The title command: A title can be added to the plot with the command:

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

The text will appear on the top of the figure as a title.

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

```
text(x,y,'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 (with the mouse).

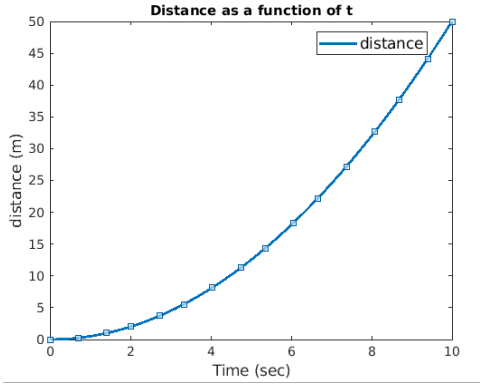
The legend command: The legend command places a legend on the plot. The legend shows a sample of the line type of each graph that is plotted, and places a label, specified by the user, beside the line sample. The usage is:

```
legend('string 1','string 2',... )
```

For example, the command:

```
t=linspace(0,10,100)
s=t.^2/2;
plot(t,s);
xlabel('Time (sec)');
ylabel('distance (m)');
title('Distance as a function of t');
legend('distance')
```

creates a plot as shown here:



Formatting the texts: The texts in the `xlabel, ylabel, title, text` and `legend` commands can be formatted to customize the font, size, style (italic, bold, etc.), and color. Formatting can be done by adding optional `PropertyName` and `PropertyValue` arguments following the string inside the command. For example:

```
text(x,y,'text as string', 'PropertyName', PropertyValue)
```

Some of the PropertyName are:

Property Name	Description
Rotation	the orientation of the text (in degree)
FontSize	the size of the font (in points)
FontWeight	the weight of the characters (light, normal, bold)
Color	the color of the text (e.g., r, b, etc.)

Some formatting can also be done by adding modifiers inside the string. For example, adding \bf, \it, or \rm, will create a text in bold font, italic style, and with normal font, respectively. A single character can be displayed as a subscript or a superscript by typing _ (the underscore character) or ^ in front of the character, respectively. A long superscript or subscript can be typed inside { } following the _or the ^. For example:

```
title('\bf The values of X_{ij}', 'FontSize',18)
```

Greek characters: Greek characters can be included in the text by typing \ (back slash) before the name of the letter.

Character	Letter
\alpha	α
\beta	β
\gamma	γ
\theta	θ
\pi	π
\sigma	σ

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 of the axes. Here are some possible forms of the axis command:

Commands	Description
axis([xmin, xmax, ymin, ymax])	sets the limits of both x and y
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

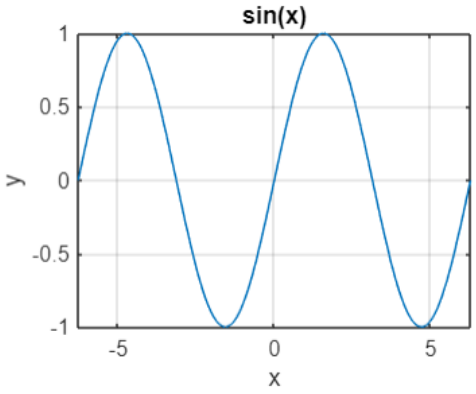
The grid command: 'grid on' adds grid lines to the plot. grid off removes grid lines from the plot.
For example:

```
fplot(@(x) x.^2+4.*sin(2.*x)-1,[-3,3])
grid on
axis([-2 2 -5 5])
```

Plotting with fplot:

`fplot(f)` plots the curve defined by the function $y = f(x)$ over the default interval $[-5, 5]$ for x. `fplot(f, [xmin, xmax])` plots over the specified interval `[xmin, xmax]`.
For example, to plot $f(x) = \sin x$ curve in the interval $[-2\pi, 2\pi]$.

```
syms x
f(x)=sin(x)
fplot(f(x),[-2*pi, 2*pi])
grid on
title('sin(x)')
xlabel('x');
ylabel('y');
```



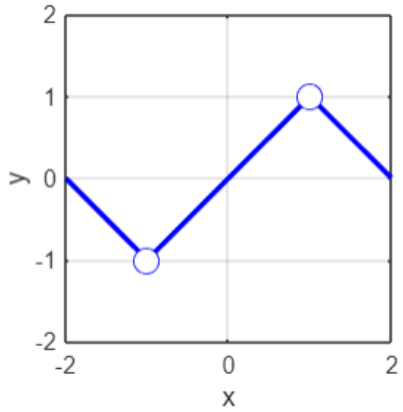
Specify Plotting Interval and Plot Piecewise Functions

`f(x)=piecewise(I1,f1,I2,f2,...)` can be used to define a piecewise function $f(x)$ whose value is `f1` in the interval `I1` and `f2` in the interval `I2`.

For example to plot a piecewise function $f(x) = \begin{cases} -x-2 & ; -2 \le x \le -1 \\ x & ; -1 < x \le 1 \\ -x+2 & ; 1 < x \le 2 \end{cases}$.

```
clear
clc
syms f(x)
f(x)=piecewise( -2<=x<=-1,-x-2,-1<x<=1,x,1<x<=2,-x+2);
figure
fplot(f(x), [-2,-1], 'b', 'LineWidth', 2);
grid on; hold on
```

```
fplot(f(x), [-0.999, 1], 'b', 'LineWidth', 2);
fplot(f(x), [1.001, 2], 'b', 'LineWidth', 2);
plot(-1, f(-1), 'bo', 'MarkerFaceColor', 'w', 'MarkerSize', 10);
plot(1, f(1), 'bo', 'MarkerFaceColor', 'w', 'MarkerSize', 10);
axis equal; axis([-2 2 -2 2]);
xlabel('x'); ylabel('y');
```



Plotting multiple graphs

In many situations there is a need to make several graphs in the same plot. There are two methods to plot multiple graphs in one figure. One is by using the plot command, the other is by using the hold on, hold off commands. For example:

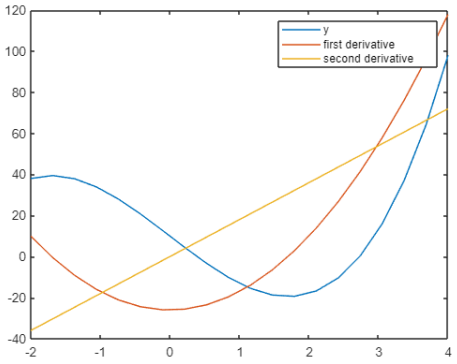
```
plot(x,y,u,v,s,t)
```

creates 3 graphs: y vs x, v vs u, and t vs s, all in the same plot. The vector of each pair must have the same length. MATLAB automatically plots the graphs in different colors so that they can be identified. It is also possible to add line specifiers following each pair. For example:

```
plot(x,y,'-bo',u,v,'--rx',s,t,'g:s')
```

plots y vs x with a solid blue line and circles, v vs u with a dashed red lines with cross signs, t vs s with a dotted green line and square markers.

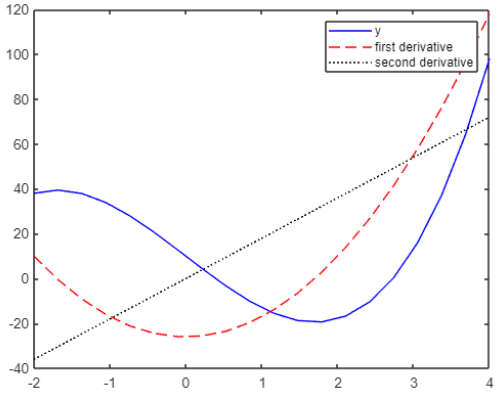
For example, the following script plots the graphs of the functions $y = 3x^3 - 26x + 10$ together with its first derivative $y' = 9x^2 - 26$ and second derivative $y'' = 18x$, for $-2 \le x \le 4$.




```
x = linspace(-2,4,20);
y = 3*x.^3-26*x+10;
dy = 9*x.^2-26;
ddy = 18*x;
ddy = 18*x;
plot(x,y,x,dy,x,ddy);
legend('y','first derivative','second derivative');
```

Using the hold on, hold off command

To plot several graphs using the hold on, hold off commands, one graph is plotted first with the plot command. Then the hold on command is typed. This keeps the Figure Window with the first plot open, including the axis properties and the formatting. Additional graphs can be added with plot commands that are typed next. The hold off command stops this process. It returns MATLAB to the default mode in which the plot command erases the previous plot and resets the axis properties.



With the data from the previous example, we can plot y and its derivatives by typing commands shown in the script:

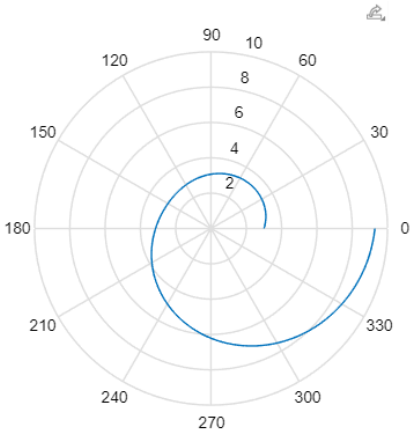
```
plot(x,y,'-b');
hold on
plot(x,dy,'--r');
plot(x,ddy,':k');
hold off
```

Polar plots

The polar command is used to plot functions in polar coordinates. The command has the form:

```
polar(theta,radius,'linespec')
```

where theta and radius are vectors whose elements define the coordinates of the points to be plotted. The line specifiers are the same as in the plot command. To plot a function $r = f(\theta)$ in a certain domain, a vector for values of θ is created first, and then a vector r with the corresponding values of $f(\theta)$ is created using element-wise calculation.



For example, a plot of the function $r = 3\cos^2(\theta/2) + \theta$, for $0 \le \theta \le 2\pi$ is done by:

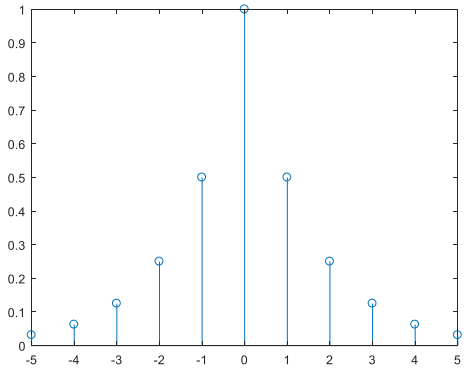
```
theta = linspace(0,2*pi,200);
r = 3*cos(theta/2).^2+theta;
polar(theta,r)
```

Stem plots: A two-dimensional stem plot displays data as lines extending from a baseline along the *x*-axis. A circle (the default) or other marker whose *y*-position represents the data value terminates each stem.

`stem(Y)` plots the data sequence `Y` as stems that extend from equally spaced and automatically generated values along the *x*-axis. When `Y` is a matrix, `stem` plots all elements in a row against the same *x* value. `stem(X,Y)` plots `X` versus the columns of `Y`. `X` and `Y` are vectors or matrices of the same size.

For example, the discrete plot of $y = \left(\frac{1}{2}\right)^{|x|}$, $-5 \le x \le 5$.

```
x=-5:5;
y= 0.5.^abs(x);
stem(x,y);
```



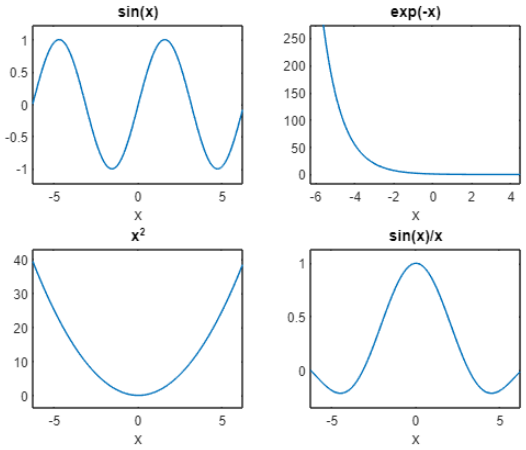
Multiple plots on the same window: Multiple plots on the same page can be created with the subplot command, which has the form:

```
subplot(m,n,p)
```

The command divides the Figure Window into mxn rectangular subplots where plots will be created. The subplots are arranged like elements in a mxn matrix where each element is a subplot. The subplots are numbered from 1 through mn. The number increases from left to right within a row, from the first row to the last. The command subplot(m,n,p) makes the subplot p current. This means that the next plot command will create a plot in this subplot.

For example, we create a plot that has 2 rows and 2 columns:

```
subplot(2,2,1);ezplot('sin(x)');
subplot(2,2,2);ezplot('exp(-x)');
subplot(2,2,3);ezplot('x^2');
subplot(2,2,4);ezplot('sin(x)/x');
```



Exercise:

1. Plot the graph of the function $f(x) = x^3 + 3x^2 + 2$ in the domain $-10 \le x \le 10$. Indicate the $x-$label, $y-$label, title of the graph. Set the thickness of the line as 2 pt.

2. In the domain $-3\pi \le x \le 3\pi$, plot $y = \sin x$. On the same graph, superimpose the curve $y = \cos x$ with a different colour. Indicate the $x-$label, $y-$label, title of the graph and legend.