2D Scilab Graphics

There are numerous Scilab graphics commands. We will first introduce some of the most basic graphics commands that are regularly used in simulation and modeling.

One of the first things that someone interested in modeling and simulation wants to do is to generate plots. These plots can be graphs of functions, surfaces, curves, or data.

Plot

See help: http://www.mathworks.com/help/matlab/ref/plot.html?refresh=true

The **plot** command generates two-dimensional graphs. Given vectors

$$x=[x_1,...,x_n] \text{ and } y=[y_1,...,y_n]$$

The Plot (x,y) will plot the points $\{(x_1,y_1),...,(x_n,y_n)\}$.

to insert markers the following record is used: **plot(x,y,s),** where ***s*** – a string of three characters that defines the line color, type of marker, and bar types, respectively (Tab 3.1-3.3).

Table 3.1. Color line graph

Symbol	Description
y	Yellow
m	Magenta
c	Cyan
r	Red
g	Green
b	Blue
w	White
k	Black

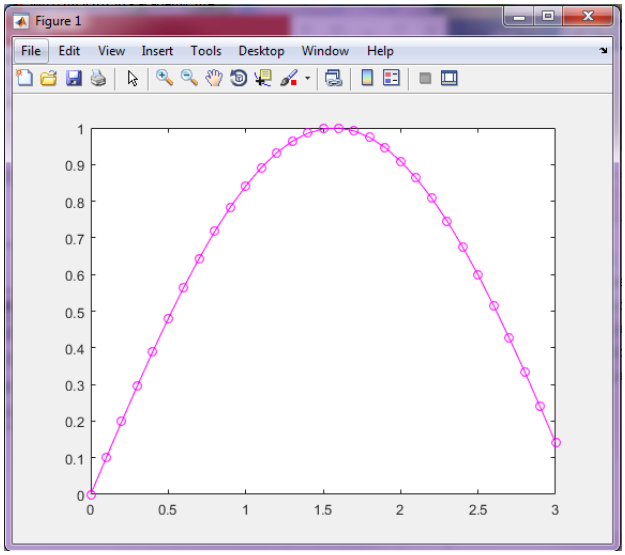
Table 3.2. Type of marker

Symbol	Description
.	Point
o	Circle
x	Cross
+	Plus sign
*	Asterisk
s	Square
d	Diamond
v	Downward-pointing triangle
^	Upward-pointing triangle
<	Left-pointing triangle
>	Right-pointing triangle Five-
P	Pentagram
h	Hexagram

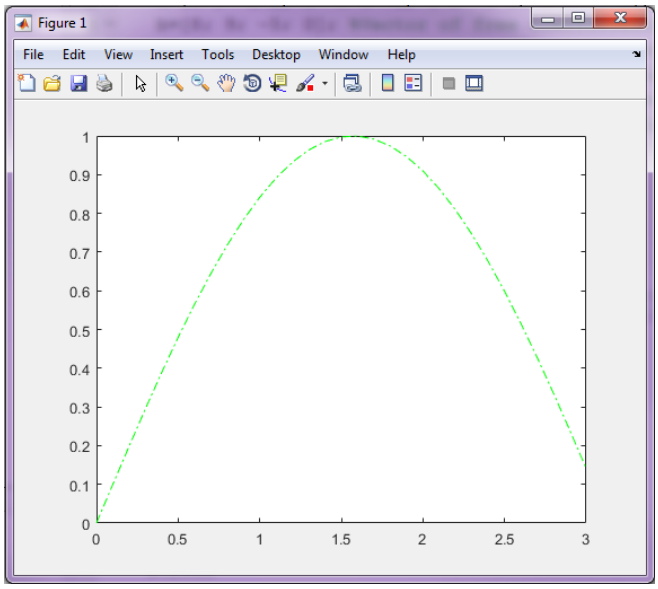
Table 3.3. Bar types of graph

Symbol	Description
-	Solid line (default)
:	Dotted line
-.	Dash-dot line
>	Dashed line

>> tt=0:0.1:3; plot(tt, sin(tt),'mo-');



>> tt=0:0.1:3; plot(tt, sin(tt),'g-.');



To construct several graphs in one coordinate system can refer to the function **plot** as follows:

plot(x1,y1,x2,y2,…,xn,yn)

where x1, y1- arrays abscissa, ordinate the first graph;

x2, y2 - arrays abscissa, ordinate of the second graph;

…

xn, yn - Arrays abscissa, ordinate of the n graph.

Or other form:

plot(x1,y1,s1,x2,y2,s2,…,xn,yn,sn)

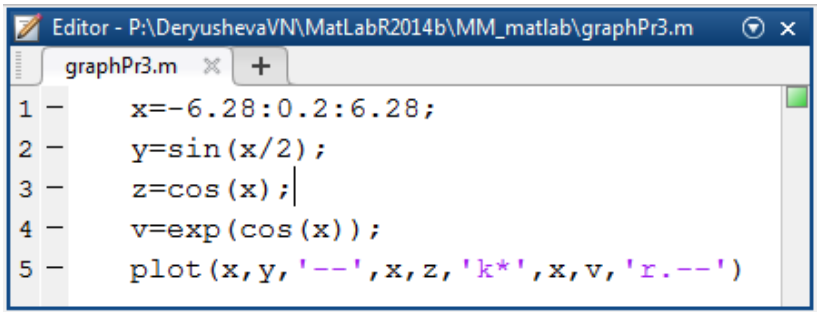
where s1 – parameters of line of the first graph;

s2 – parameters of line of the second graph;

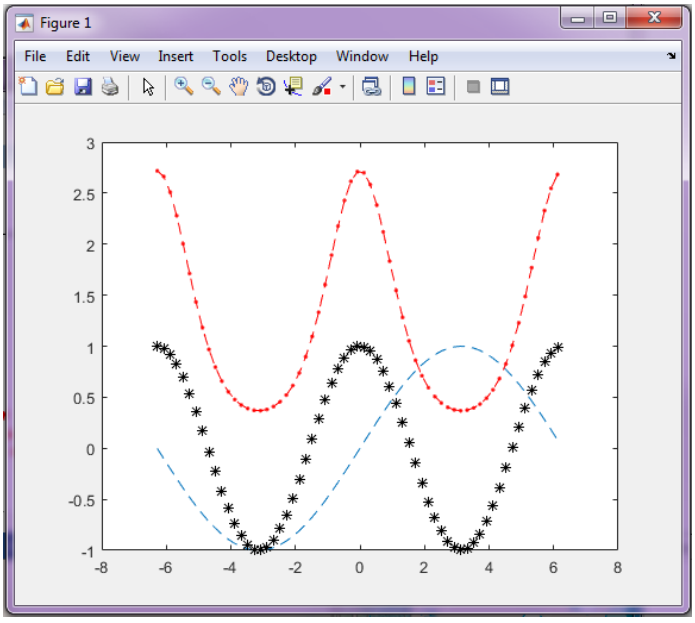
..

sn - parameters of line of the n graph.

For example, to construct graphs of functions $y=\sin\left(\frac{x}{2}\right)$, $z=\cos(x)$; $v=e^{\cos(x)}$ in the same coordinate axes. $y=\sin\left(\frac{x}{2}\right)$ - dashed; $z=\cos(x)$ - black color with a marker in the form of stars; $v=e^{\cos(x)}$ - dashed, red color, with a marker in form of a point:



Results:



Grid in the graph can be done with the command xgrid(color), where color is id color of the line of the grid. If parentheses of command **grid on** (default grid off) are empty than color of grid is black. We build sinusoids of red and blue in the same coordinate axes

The Command **grid** is used for the chart title:

title('title')

where title – the name of the graph;

xlabel('x') – X axis title;

ylabel('y') – Y axis title.

When we show the graphs of several functions in a coordinate plane, we need a "legend". We use for this command **legend**:

legend(s1,s2,…,sN)

where **s1** - the name of the first graph, **s2** - the name of the second graph, **sN** - the name of the n graph;

legend('boxon') **–** displays the box around the legend.

legend('boxoff') – removes the box around the legend.

legend(s1,...,sN,'Location',Value) – specifies the legend location. Set Value to a location string such as 'northoutside'. For a list of strings, see the Location property (Table 3.4).

legend(___,'Orientation',Value) – specifies the legend orientation. This option can be used with either of the input argument combinations in the previous syntaxes. Set Value as either 'vertical' or 'horizontal'. For more information, see the Orientation property.

Orientation, specified as one of these values:

'vertical' — Stacks the legend entries vertically. This is the default.

'horizontal' — Lists the legend entries side-by-side.

Table 3.4. Location with respect to the axes, specified as one of the location strings listed in this table.

Supported String	Resulting Location
'north'	Inside top of axes
'south'	Inside bottom of axes
'east'	Inside right of axes
'west'	Inside left of axes
'northeast'	Inside top-right of axes (default for 2-D axes)
'northwest'	Inside top-left of axes
'southeast'	Inside bottom-right of axes
'southwest'	Inside bottom-left of axes
'northoutside'	Above the axes
'southoutside'	Below the axes
'eastoutside'	To the right of the axes
'westoutside'	To the left of the axes
'northeastoutside'	Outside top-right corner of the axes (default for 3-D axes)
'northwestoutside'	Outside top-left corner of the axes
'southeastoutside'	Outside bottom-right corner of the axes
'southwestoutside'	Outside bottom-left corner of the axes

'best'	Inside axes where least conflict with data in plot
'bestoutside'	To the right of the axes
'none'	Determined by Position property

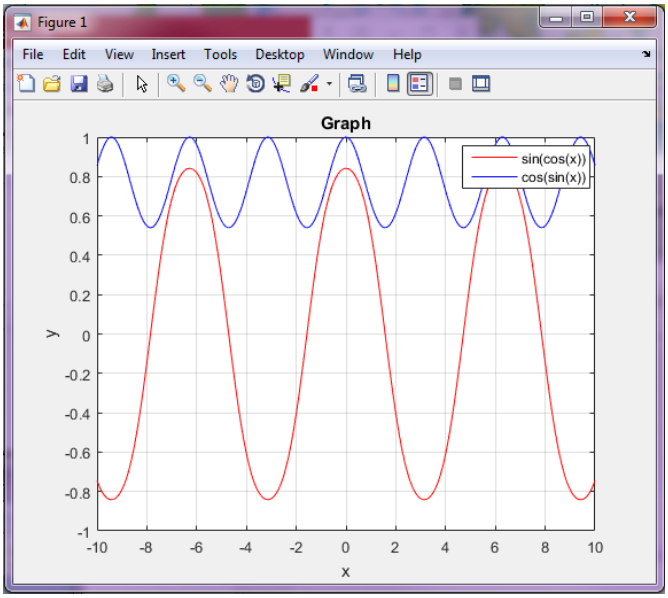
Table 3.5. This table lists the supported special characters with the Interpreter property set to 'tex'.

Character Sequence	Symbol	Character Sequence	Symbol	Character Sequence	Symbol
\alpha	α	\upsilon	υ	\sim	~
\angle	∠	\phi	Φ	\leq	≤
\ast	*	\chi	χ	\infty	∞
\beta	β	\psi	ψ	\clubsuit	♣
\gamma	γ	\omega	ω	\diamondsuit	♦
\delta	δ	\Gamma	Γ	\heartsuit	♥
\epsilon	ε	\Delta	Δ	\spadesuit	♠
\zeta	ζ	\Theta	Θ	\leftrightarrow	↔
\eta	η	\Lambda	Λ	\leftarrow	←
\theta	Θ	\Xi	Ξ	\Leftarrow	⇐
\vartheta	ϑ	\Pi	Π	\uparrow	↑
\iota	ι	\Sigma	Σ	\rightarrow	→
\kappa	κ	\Upsilon	Υ	\Rightarrow	⇒
\lambda	λ	\Phi	Φ	\downarrow	↓
\mu	μ	\Psi	Ψ	\circ	°
\nu	ν	\Omega	Ω	\pm	±
\xi	ξ	\forall	∀	\geq	≥
\pi	π	\exists	∃	\propto	∝
\rho	ρ	\ni	∋	\partial	∂
\sigma	σ	\cong	≅	\bullet	•
\varsigma	ς	\approx	≈	\div	÷
\tau	τ	\Re	ℜ	\neq	≠
\equiv	≡	\oplus	⊕	\aleph	ℵ
\Im	ℑ	\cup	∪	\wp	℘
\otimes	⊗	\subseteq	⊆	\oslash	∅
\cap	∩	\in	∈	\supseteq	⊇
\supset	⊃	\lceil	⌈	\subset	⊂
\int	∫	\cdot	·	\o	o
\rfloor	⌋	\neg	¬	\nabla	∇
\lfloor	⌊	\times	x	\ldots	...
\perp	⊥	\surd	√	\prime	′
\wedge	∧	\varpi	ϖ	\0	∅
\rceil	⌉	\rangle	〉	\mid	|
\vee	∨	\langle	〈	\copyright	©

See also: http://www.mathworks.com/help/matlab/ref/legend.html?searchHighlight=legend

For example:

```
Editor - P:\DeryushevaVN\MatLabR2014b\MM_matlab\graphPr32.m
graphPr32.m
1 —  x=-10:0.01:10;
2 —  y=sin(cos(x));
3 —  z=cos(sin(x));
4 —  plot(x,y,'r',x,z,'b')
5 —  grid on;
6 —  title('Graph');
7 —  xlabel('x');
8 —  ylabel('y');
9 —  legend('sin(cos(x))', 'cos(sin(x))');
```



Construction of several graphs in one graphics window

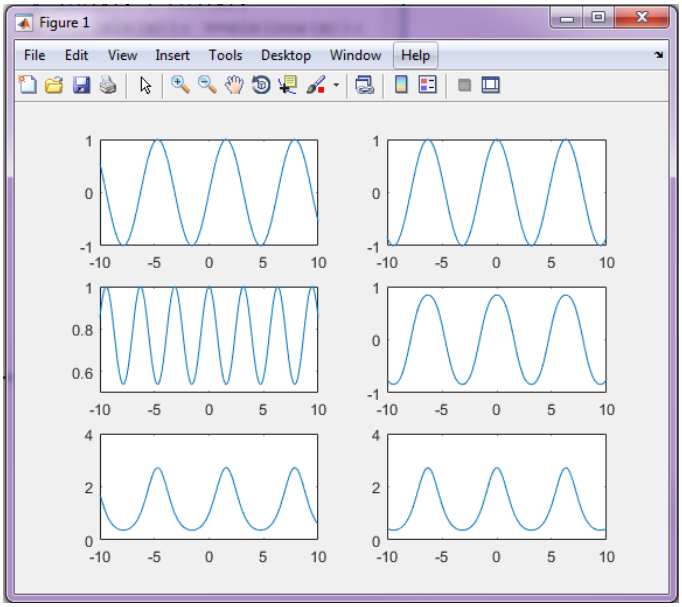
Subplot separates a graphical window into several areas:

subplot(m,n,p)

subplot separates the graphics window on **m** windows and vertical windows **n** horizontal window becomes the current window number **p.**

For example:

```
Editor - P:\DeryushevaVN\MatLabR2014b\MM_matlab\graph...
graphPr33.m
1  x=-10:0.01:10;
2  y=sin(x); z=cos(x);
3  u=cos(sin(x)); v=sin(cos(x));
4  w=exp(sin(x)); r=exp(cos(x));
5  subplot(3,2,1);
6  plot(x,y);
7  subplot(3,2,2);
8  plot(x,z);
9  subplot(3,2,3);
10 plot(x,u);
11 subplot(3,2,4);
12 plot(x,v);
13 subplot(3,2,5);
14 plot(x,w);
15 subplot(3,2,6);
16 plot(x,r);
```



2-D line plots with y-axes on both left and right side

plotyy(X1,Y1,X2,Y2) – plots Y1 versus X1 with y-axis labeling on the left and plots Y2 versus X2 with y-axis labeling on the right.

plotyy(X1,Y1,X2,Y2,function) –uses the specified plotting function to produce the graph.

function can be either a function handle or a string specifying plot, semilogx, semilogy, loglog, stem, or any MATLAB® function that accepts the syntax

h = function(x,y)

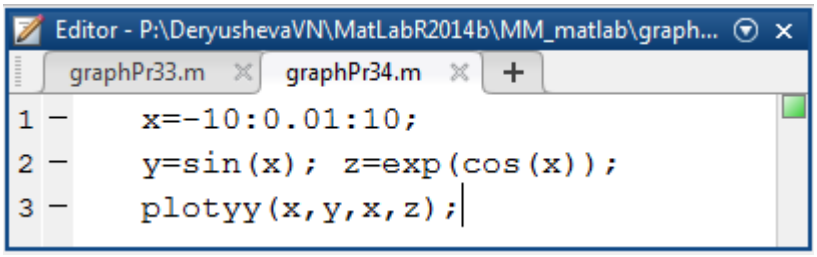
For example,

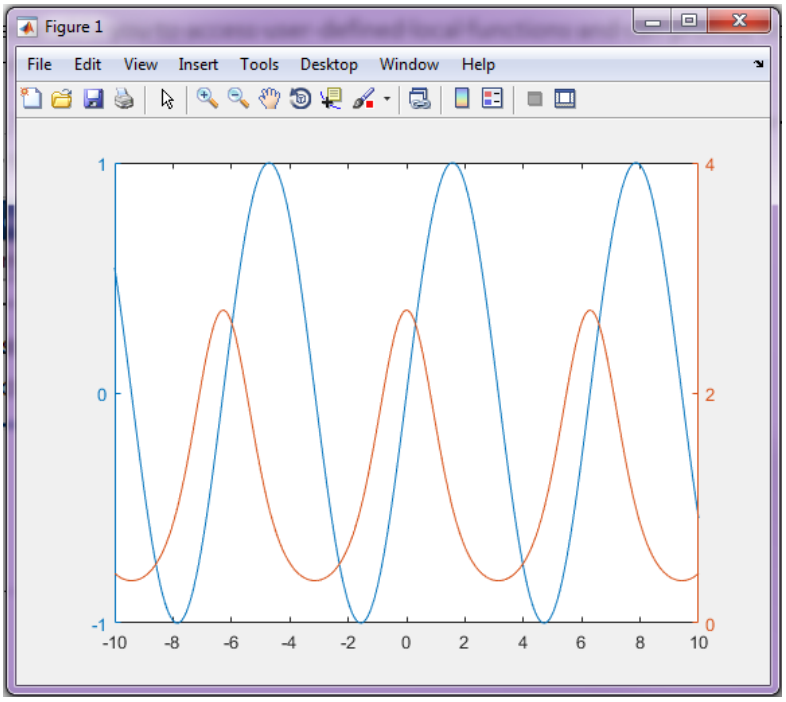
plotyy(x1,y1,x2,y2,@loglog) % function handle

plotyy(x1,y1,x2,y2,'loglog') % string

Function handles enable you to access user-defined local functions and can provide other advantages. For more information on using function handles, see Create Function Handle.

plotyy(X1,Y1,X2,Y2,'function1','function2') – uses function1(X1,Y1) to plot the data for the left axis and function2(X2,Y2) to plot the data for the right axis.





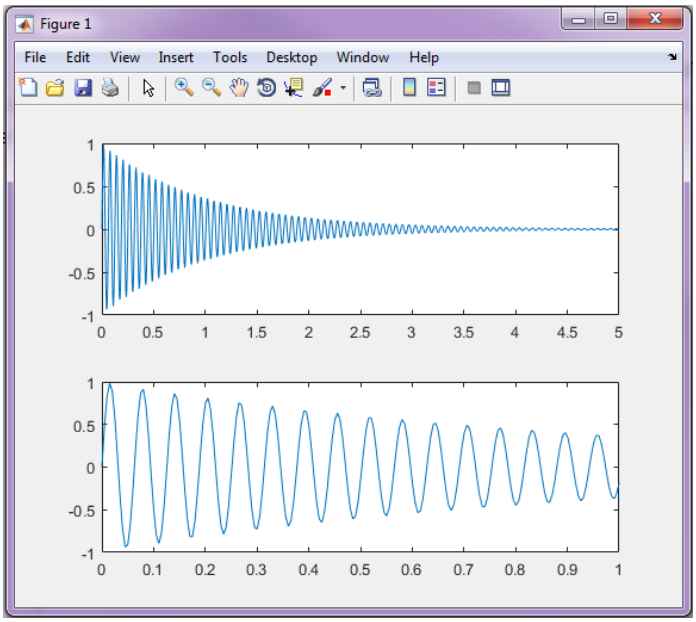
Set or query x-axis limits

xlim(limits) – specifies the x-axis limits for the current axes. Specify limits as a two-element vector of the form [xmin xmax], where xmax is a numeric value greater than xmin.

xlim auto lets the axes choose the x-axis limits. The axes chooses limits that span the range of the plotted data. This command sets the XLimMode property for the axes to 'auto'.

xlim manual freezes the limits at the current values. Use this option if you want to retain the current limits when adding new data to the axes using the hold on command. This command sets the XLimMode property for the axes to 'manual'.

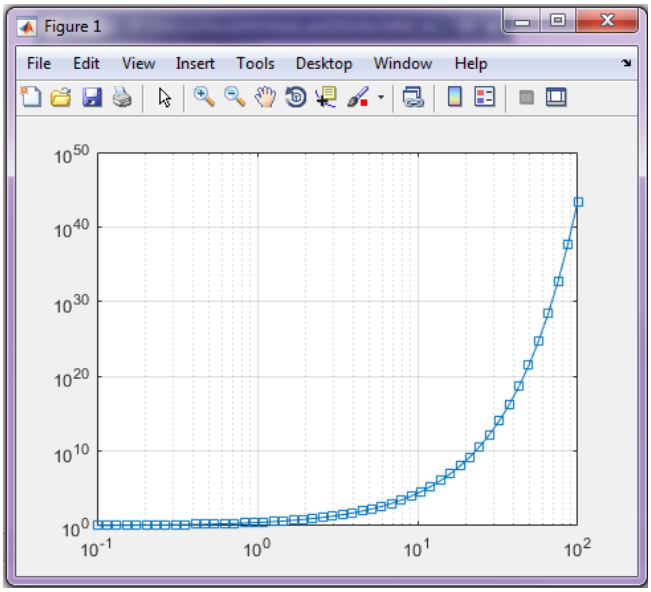
```
Editor - P:\DeryushevaVN\MatLabR2014b\MM_m...
graphPr35.m
1 —    x = linspace(0,5,1000);
2 —    y = sin(100*x)./exp(x);
3 —    ax1 = subplot(2,1,1);
4 —    plot(x,y)
5
6 —    ax2 = subplot(2,1,2);
7 —    plot(x,y)
8 —    xlim(ax2,[0 1])
```



Log-log scale plot

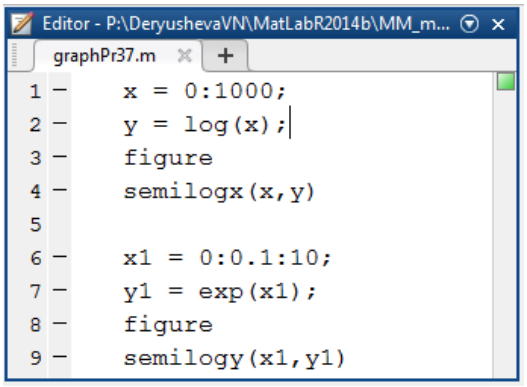
loglog(X1,Y1,LineSpec,...) – plots all lines defined by the Xn,Yn,LineSpec triples, where LineSpec determines line type, marker symbol, and color of the plotted lines. You can mix Xn,Yn,LineSpec triples with Xn,Yn pairs, for example,

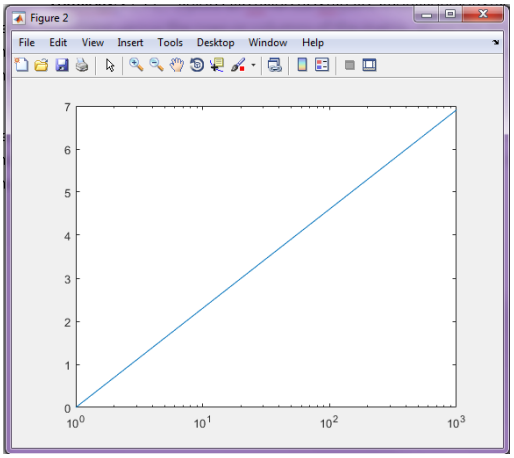
```
Editor - P:\DeryushevaVN\MatLabR2014b\MM_m...
graphPr36.m
1 —    x = logspace(-1,2);
2 —    y = exp(x);
3
4 —    figure
5 —    loglog(x,y,'-s')
6 —    grid on
```

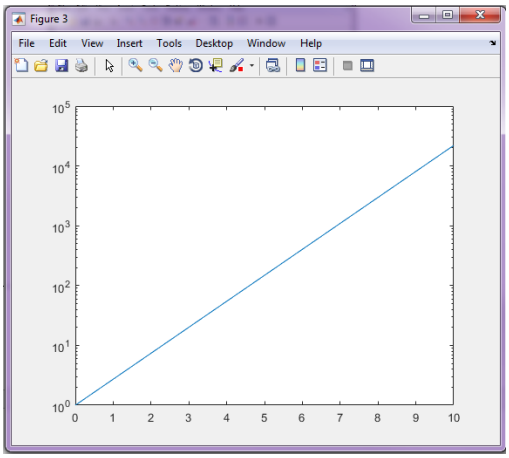


semilogx(X1,Y1,...) plots all Yn versus Xn pairs. If only one of Xn or Yn is a matrix, semilogx plots the vector argument versus the rows or columns of the matrix, along the dimension of the matrix whose length matches the length of the vector. If the matrix is square, its columns plot against the vector if their lengths match.

semilogy(X1,Y1,...) plots all Yn versus Xn pairs. If only one of Xn or Yn is a matrix, semilogy plots the vector argument versus the rows or columns of the matrix, along the dimension of the matrix whose length matches the length of the vector. If the matrix is square, its columns plot against the vector if their lengths match.







Curve in polar coordinates.

The polar function accepts polar coordinates, plots them in a Cartesian plane, and draws the polar grid on the plane.

polar(theta,rho) – creates a polar coordinate plot of the angle theta versus the radius rho. theta is the angle from the x-axis to the radius vector specified in radians; rho is the length of the radius vector specified in dataspace units.

polar(theta,rho,LineSpec) – LineSpec specifies the line type, plot symbol, and color for the lines drawn in the polar plot.

polar(axes_handle,...) – plots into the axes with the handle axes_handle instead of into the current axes (gca).

h = **polar**(...) – returns the handle of a line object in h.

The plot in polar coordinates of $\rho = \sin 2\theta \cdot \cos 2\theta$ for $0 \le \theta \le 2\pi$ is created by the following commands: