MATLAB misc plotting functions

This is Chapter/Section 2.10 onward of the textbook and Chapter/Section 3.5

Disclaimer: *These abbreviated notes DO NOT substitute the textbook for this class. They should be used IN CONJUNCTION with the textbook and the material presented in class. If there is a discrepancy between these notes and the textbook, ALWAYS consider the textbook to be correct. Report such a discrepancy to the instructor so that he resolves it. These notes are only distributed to the students taking this class with A. Gerbessiotis in Fall 2011 ; distribution outside this group of students is NOT allowed.*

E0 Plotting in MATLAB. MATLAB has a variety of plotting capabilities. The basic function `plot` allows for a 2D line plot. Function `plot3` allows for 3D line plots. Functions `semilogx, semilogy, loglog` allow for log-scale plots along the x, y and both axes. For plotting, one has to create the x and y coordinates of the function $f(x)$ that will be plotted. Thus for $y_i = f(x_i)$, one can create two vectors x and y that hold respectively the x_i and y_i coordinates. Then `plot(x,y)` will create a line plot of the (x_i, y_i) points. On the same figure multiple curves can be plotted through either one `plot` invocation or repeated `plot` invocations; in this latter case one has to issue a MATLAB `hold on` function call between consecutive invocations so that one plot does not overwrite the one.

Plots can have attributes. Thus functions `xlabel, ylabel, title` can assign titles on the x and y axes and the figure plotted. The titles are strings of characters. Thus
`xlabel('This is the x-axis label','FontSize',14,'FontName','Times','FontWeight','b');` can assign a title and change the default font size (10), font (Helvetica) and weight (regular) used. A plot function can also include attributes that include a color for the line that connects the points (a one character option such as `y,m,c,r,g,b,w,k` for yellow, magenta, cyan, red,green, blue, white and black), a marker style that can be `., o, x, +, *, s, d, v, ^, <, >, p, h` or `nothing at all` to indicate either the corresponding mark or a square, diamond, triangle down or up, or left, or right, a pentagram or hexagram and a line-style which can be a solid line, dotted, dash-dotted and dashed or no line at all respectively (`-,:, -.  , --`). Each plot can have a handle. Thus `h1=plot(x,y);` would create a handle `h1` for this plot. Handles can be used in defining a `legend` for the various plot curves. If no handle is created, then the first plot corresponds to the first legend, the second to the second, and so on. The last two arguments of a legend invocation might also specify with two strings the location of the legend: the first argument is the string `'Location'` followed by some location information such as `'NorthWest'` or `'NorthWestOutside'` depending on whether the legend will be located inside or outside of the plotting area. For a plot with handle `h1` we can get information about the handle through a sequence `f1=gcf;get(f1);`
We can then set additional attributes of the figure associated with the handle by typing for example `set(f1,'position',[ 40 40 100 100]);` For the example above,

```
legend('Linear function','Location','North');
```

or

```
legend(h1,'Linear function','Location','North');
```

can achieve the same objective.

E1. Plotting in MATLAB (line 2D plot)

E1.1 Defining the set of points. In its simplest form a `point` is described by its x and y coordinates on the Euclidean plane. (MATLAB can also plot in alternative spaces.) The x-axis and y-axis meet at $(0, 0)$; x grows to the right, y upwards. The set of x coordinates of the points can be specified in the form of a vector and so can the y coordinates or they can be expressed as a function (through the use of array operators) of the x coordinates. (One can use the colon operator : or function `linspace` to generate a sequence of points of specified step or length in an interval.)

Example of a simple 2D plot: $y = \log(x)$ **from** $x = 1$ **to** $x = 100$**.** A vector x is defined with the colon operator and then y in terms of x. The `plot` function is then issued. The code is given below and Figure 1 depicts the end-result. MATLAB draws the points of a 2d line plot on a separate window. In Figure 2. grid-lines are drawn, the x- and y-axis are labelled x and y respectively, and `Our plot` appears as the title of the drawing.

```
Figure 1:    x = 1:1:100;
             y = log(x);
             plot(x,y);

Figure 2:    x= 1:1:100;
             y= log(x);
             plot(x,y); title('Our plot'); xlabel('x');ylabel('y'); grid on
```

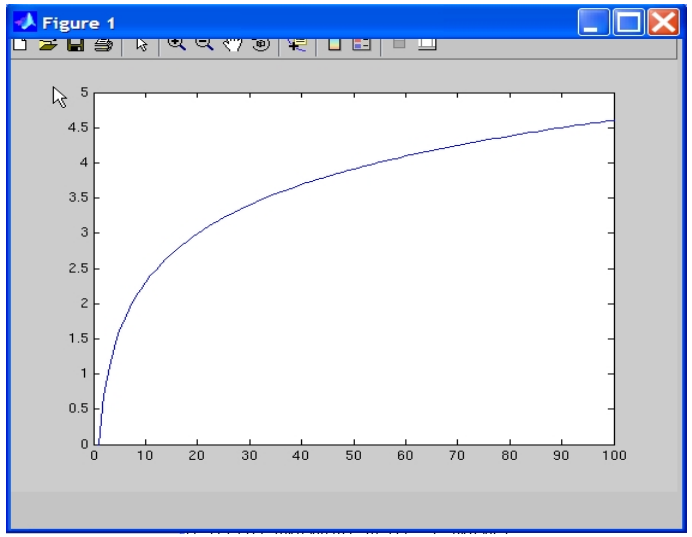


Figure 1: Plot for $y = \log(x)$

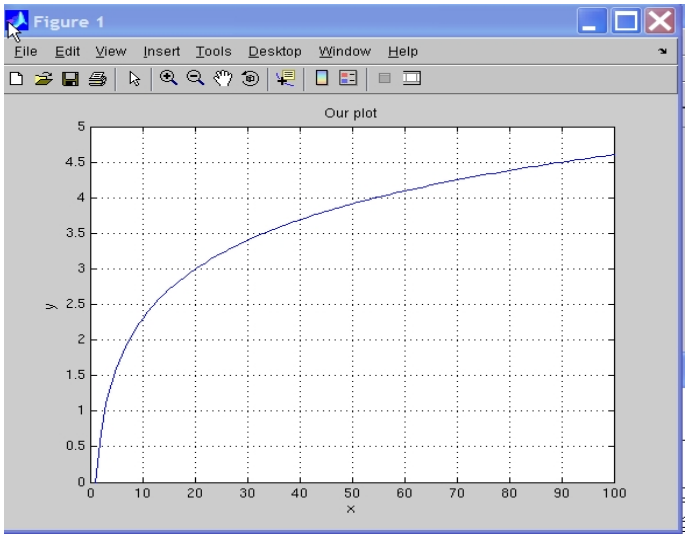


Figure 2: More elaborate plot for $y = \log(x)$

(c) Copyright A. Gerbessiotis. All rights reserved.

E1. Plotting in MATLAB (continued)

E1.2 Plotting and Printing. In MATLAB we can save/print a plot into a file in a variety of file formats for subsequent viewing or printing. If in the command prompt one types `print options filename`
MATLAB will generate a file named `filename` in the current directory as indicated by a `pwd` command (the whole process might take several seconds). The `options` can be one of

```
-deps    Creates a monochrome Encapsulated Postscript image        -djpeg   JPEG image
-depsc   Similar to -deps except that it is colored                -dpng    PNG  image
-dtiff   compressed TIFF image
```

E1.3 Multiple Plots. In MATLAB we can plot multiple functions at the same time. For each function plotted, a separate `plot` command needs not to be issued; one `plot` can be issued instead. This is shown in Figure 3, with the default options modified for Figure 4.

```
x = 1:1:100;        x = 1:1:100;
y = log(x);         y = log(x);
y1= sin(x);         y1= sin(x);
plot(x,y,x,y1);     plot(x,y,'r--',x,y1,'b-');
```

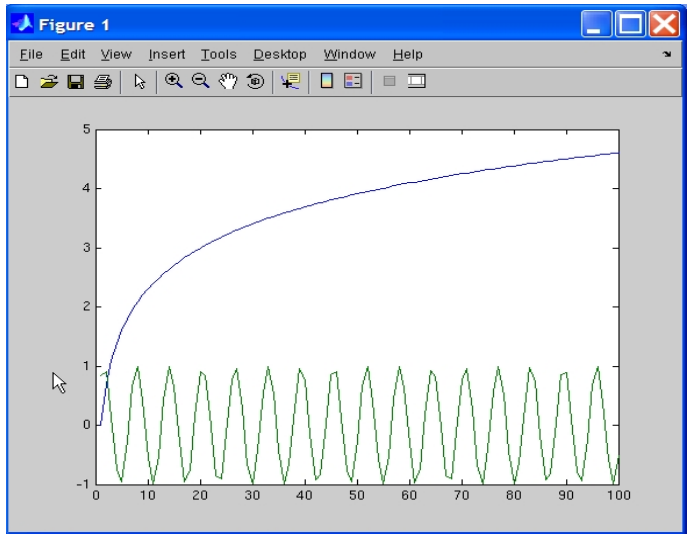


Figure 3: Plot for $y = \log(x)$, $y_1 = \sin(x)$

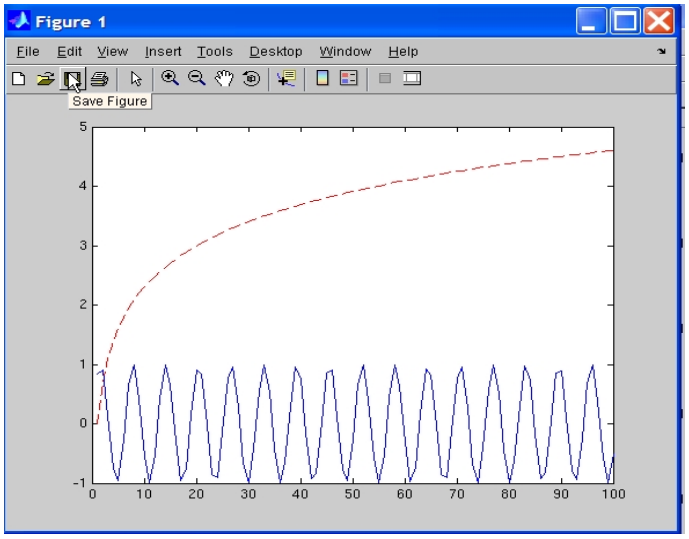


Figure 4: Different attributes for $y = \log(x)$, $y_1 = \sin(x)$

(c) Copyright A. Gerbessiotis. All rights reserved.

E1.4 `axis on`, `axis off`, `grid on`, `grid off`. The x and y axes can be removed or reapper with an `axis off` or `axis on` following the plot invocation. Similarly a `grid on` will incorporate a grid system in the plot, and a `grid off` will remove it (default). In addition `axis([ xmin xmax ymin ymax]);` sets the limits for the x and y axes. The default is `axis auto` that automatically determines the limits. For 3D plots, an `axis 'auto xz'` would determine the x and z limits automatically but not the y axis. An `axis ij` places the coordinate system origin at the upper left corner with i being vertical and increasing top to bottom and j increasing left to right horizontally.

E1.5 `xlabel`, `ylabel`, `title`. They introduce titles for the x, y axes and also the plot itself. The titles are provided in the form of MATLAB strings. The string of the title can contain however certain Latex (a typesetting language) commands. For example `string1` {\$bf `string2`} will print `string1` in normal-face (font) but `string2` in bold-face as in **string2** since the combination of a backslash a lower-case b and lower-case f indicates bold-face in Latex. Similarly instead of `bf` one could use `em` for emphasized, `sl` for slanted and `tt` for teletype and `rm` for roman. Greek letters can be typed in individually. Thus `\Gamma \Delta \gamma \delta` will print $\Gamma\ \Delta\ \gamma\ \delta$ respectively. More information is available in E8.1 (page 18).

The plot figure/window will remain open until the user decides to close it by `killing it`.

E1.6 `text`. A `text` operation can add text anywhere in the plot. Thus `text(20,log(20),'\rightarrow this is a note');` will post a text at location $x = 20$ and $y = \log(20)$ that will include a rightarrow followed by the string `this is a note`. See also E8.1. on or around page 18.

(c) Copyright A. Gerbessiotis. All rights reserved.

E2. Plotting in MATLAB (continued)

E2.1. Options in `plot`. Function `plot` plots sequences of points defined through their x and y coordinates and provided as the first two arguments of `plot`. A third argument is a string that describes the attributes of the plot. In the example of the previous page the attributes were `'r--'` and `'b-'`. The `color` attribute, a single character color indicator, specifies the color used for the plot. The `marker-style` determines whether the points plotted are individually shown. If `line-style` is used, then the points are connected with lines of the indicated style. Other options are listed in section E7 (page 16).

Color		Marker-Style		Line-Style	
y	yellow	.	point	-	solid
m	magenta	o	circle	:	dotted
c	cyan	x	x-mark	-.	dash-dot
r	red	+	plus	--	dashed
g	green	*	star		no line
b	blue	s	square		
w	white	d	diamong		
k	black	v	triangle down		
		^	triangle up		
		<	triangle left		
		>	triangle right		
		p	pentagram		
		h	hexagram		
			no marker		

E2.2 Legend positioning. Legends can be assigned to each plot drawn. Legend information is printed on or outside the plot, by assigning a value to parameter `'Location'` associated with the plot (see next page).

'North'	inside plot box near top
'South'	inside bottom
'East'	inside right
'West'	inside left
'NorthEast'	inside top right (default)
'NorthWest	inside top left
'SouthEast'	inside bottom right
'SouthWest'	inside bottom left
'NorthOutside'	outside plot box near top
'SouthOutside'	outside bottom
'EastOutside'	outside right
'WestOutside'	outside left
'NorthEastOutside'	outside top right
'NorthWestOutside'	outside top left
'SouthEastOutside'	outside bottom right
'SouthWestOutside'	outside bottom left
'Best'	least conflict with data in plot
'BestOutside'	least unused space outside plot

(c) Copyright A. Gerbessiotis. All rights reserved.

The following three examples show options of `plot`. The last one (with `NorthWest` and `Best`) are depicted in the figures below.

```
x = 1:1:100;        x = 1:1:100;
y = log(x);         y = log(x);
y1= sin(x);         y1= sin(x);
plot(x,y,x,y1);     plot(x,y,'r.',x,y1,'g-');
```

```
x = 1:1:100;                                        % This is a comment
y = log(x);
y1= sin(x);
plot(x,y,'r--',x,y1,'g--');
legend('log(x)','sin(x)','Location','NorthWest');  % Change NorthWest to Best
grid on';
```

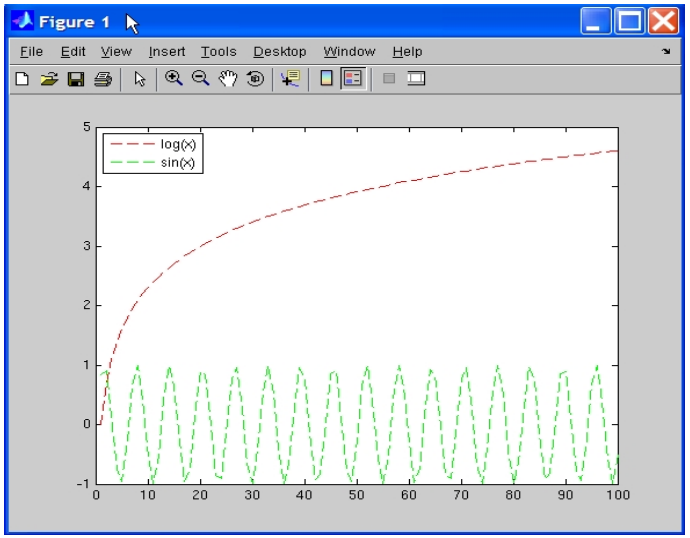


Figure 5: Plot for $y = \log(x), y_1 = \sin(x)$ with legend `NorthWest`

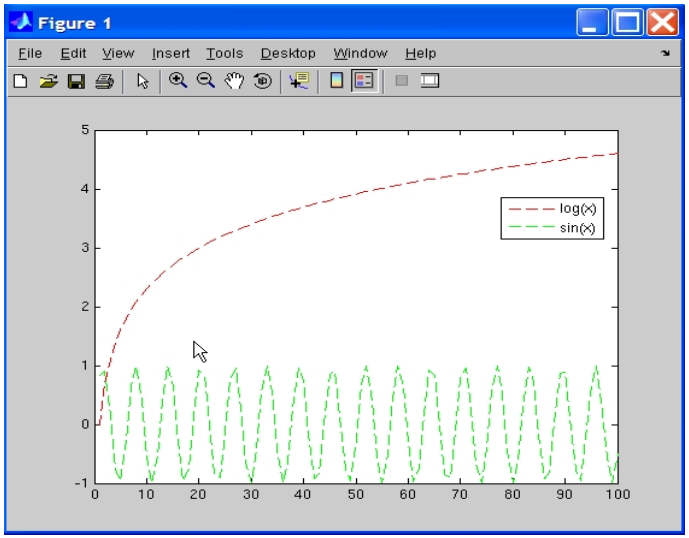


Figure 6: Plot for $y = \log(x), y_1 = \sin(x)$ with legend `Best`

(c) Copyright A. Gerbessiotis. All rights reserved.

E3.1 Scales. The default `plot` function uses linear axis scales for plotting. One can use `semilogx`, `semilogy`, or `loglog` instead of `plot` to mix linear and logarithmic plot axes. In addition one could employ `plot3` to draw 3D-plots. In addition to an x- and y- vector, one needs to provide a z-vector as well.

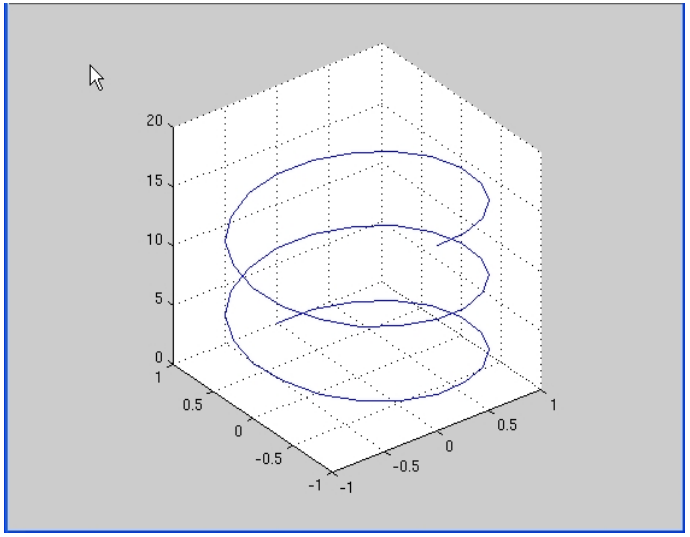


Figure 7: An example of plot3 usage

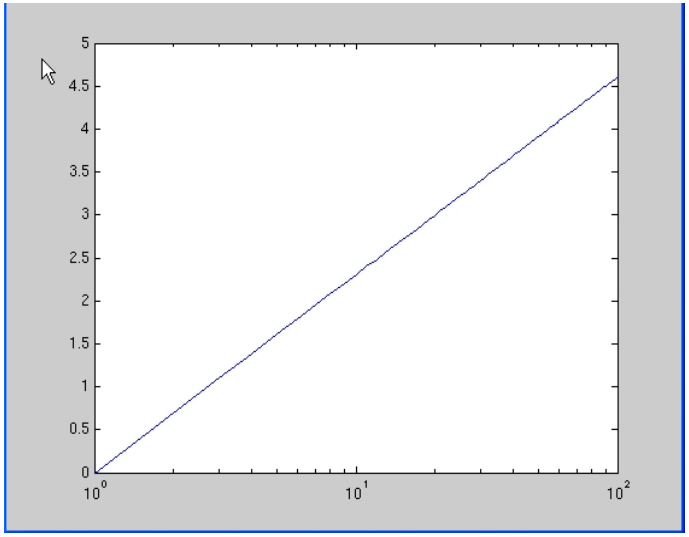


Figure 8: An example of semilogx axis usage

The former of the two figs provides an example of a 3D-plot and the latter a semilog plot for the logarithmic function that is now plotted as a linear function since $y = \log(x)$ with $X = \log x$ and $Y = y$ becomes a $Y = X$ function, i.e. a linear function. The horizontal axis is X and the vertical axis is Y in Figure 8.

```
Figure 7
  x= 0: pi/10:5*pi;
  plot3(sin(x),cos(x),x);
  grid on; axis square;
```

```
Figure 8
x=1:100
y=log(x);
semilogx(x,y); grid on;
```

E4. Small MATLAB program

E4.1 Writing MATLAB programs using ASCII files. We can write into an ASCII file a sequence of MATLAB commands such as those shown below. Then we save them into a file named `temp_conversion.m`. Note the suffix of the file-name; the file name must end in `.m`. The file ought to be saved in the directory which is accessible (and listable) in the top-left window of the MATLAB environment. Then, from within MATLAB we can run this tiny program by calling `temp_conversion` without the extension. MATLAB is intelligent enough to add to this file name the proper suffix extension `.m`. The three lines below will be executed, with input read, and output written on screen. Therefore, if you do not like the MATLAB **interactive** environment you can alternately use the **batch** processing capabilities of MATLAB under which you write all commands in a file, and you invoke that file to execute them. Or you can interactively edit/save the contents of the file in one Windows/Linux/Solaris/MATLAB window and from the Command Window you can invoke the commands of that file.

```
temp_F = input(' Enter temp in degrees Fahreneit ');
temp_C = ((temp_F -32)*5.0)/9.0 ;
fprintf('%6.2f F = %6.2f C \n',temp_F,temp_C);
```

Exercise. What does the following program plot? Explain

```
theta=0.0:pi/200:2*pi
x= 0.5 +0.5*cos(theta);
y= 0.5 +0.5*sin(theta);
plot(x,y);
```

(c) Copyright A. Gerbessiotis. All rights reserved.

```
axis                                  :  returns  [xmin , xmax , ymin , ymax ] x/y-axis limits
axis( [xmin , xmax , ymin , ymax ]);  :  set the min-max x/y-axis plotting limits
axis equal                            :  the inrements are equal for both axes
axis square                           :  axis box is a square
axis normal                           :  cancels the effects of equal and square
axis off                              :  no axis labelling,background,tickmarks
axis on                               :  turns on axis labelling,background,tickmarks [default]
```

E5.1 `axis` **command.** It returns a row vector with the limits of the xaxis and yaxis (default). Thus `xmin,xmax` means that the smallest x-value and largest x-value that can be plotted by default are `xmin, xmax` respectively. Similarly for the y-axis. Those limits can be reset if a 4-element row vector is passed as an argument to `axis`. For examples `axis([0 1000 0 10])` can change the x-axis scale to a 0 to 1000, and the y-axis scale to a 0 to 10.

E5.2 Example. Suppose we run the following MATLAB program. If we invoke `axis`, then the vector `[-8 8 -1 1]` is returned with the limits for the x-axis and y-axis. The first figure on the next page results for default setting for axis, whereas the second (on the right) is for `axis off`.

```
% File sub5p8.m
x=-2*pi:pi/10:2*pi
y=sin(x);
z=cos(x);
plot(x,y,'r--',x,z,'g--');
```

(c) Copyright A. Gerbessiotis.

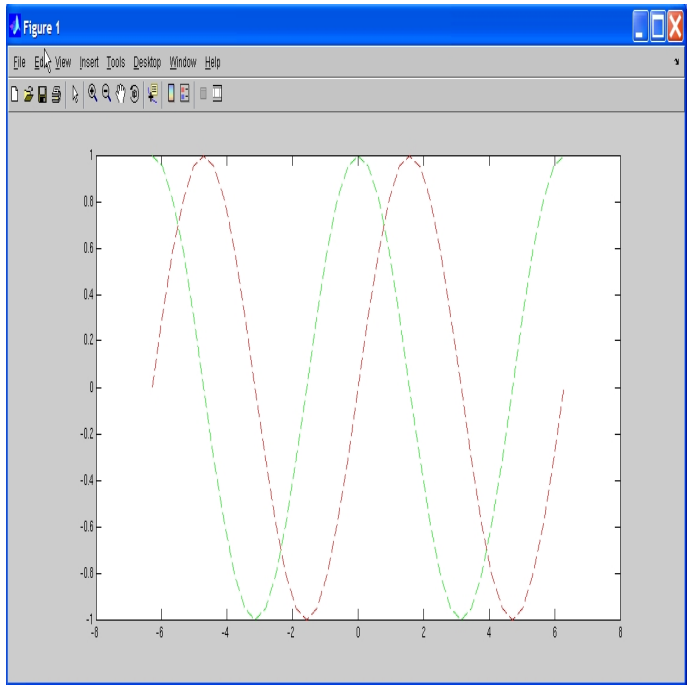


Figure 9: Plot for $y = \sin(x), z = \cos(x)$ with default `axis`

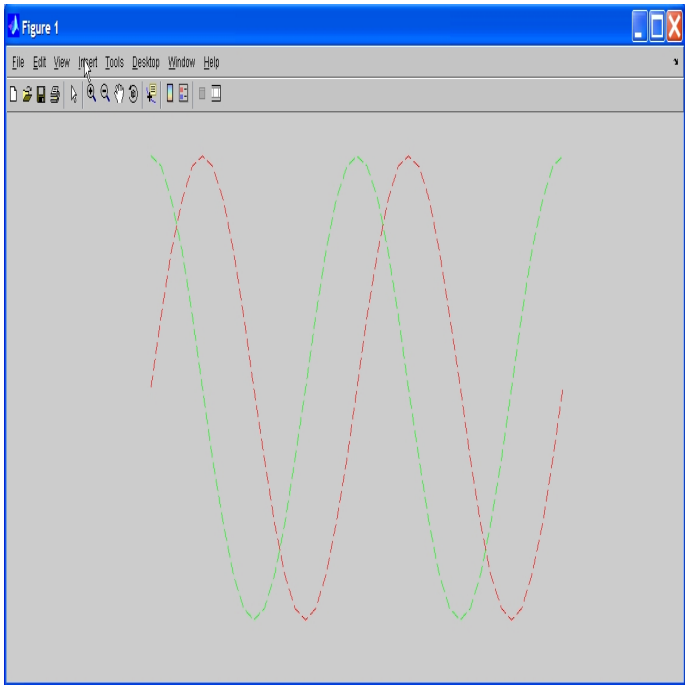


Figure 10: Plot for $y = \sin(x), z = \cos(x)$ with `axis off`

E5.3 `hold on, hold off.` When two `plot` commands are issued, the second one overwrites/destroys the first one. The only way to plot both, say one on top of the other, is to insert a `hold on` between the two. A `hold off` however switches back to the default (overwriting) behavior. Observe what happens if one tried to plot Case 1, or Case 2. Of course the effect of using a `hold on` between two `plot` commands can also be effected by the use of a single `plot` with multiple argument vectors as it was discussed in section E3.

```
% File sub5p10a.m : Case 1
x=-2*pi:pi/10:2*pi
y=sin(x);
z=cos(x);
plot(x,y,'r-');
plot(x,z,'b-');

% File sub5p10b.m : Case 2
x=-2*pi:pi/10:2*pi
y=sin(x);
z=cos(x);
plot(x,y,'r-');
hold on;
plot(x,z,'b-');
hold off;
```

(c) Copyright A. Gerbessiotis. All rights reserved.

E5. Multiple plots with multiple `plot` commands continued.

E5.4 Multiple plots on different windows. If you prefer separate windows rather than plotting one plot over the other, one can use the `figure(n)` command, where n is a number that is used for referencing the window and is also printed in the title portion of the window. For the example shown below, we call the first window `Figure 1` and the second one `Figure 9` (for no apparent reason). The second invocation makes Figure 9 the current window. Thus inserting a `figure` statement with the appropriate parameter n, makes `figure(n)` the current window used for plotting; an alternative is to click on a window with a mouse to make it the current one. Now that `Figure(9)` is the current window, if one types an `axis off` this will affect only that window while the other one will still have the default axis labelings,tickmarks,background. One can go back and forth or switch between figures in the command window by typing in the corresponding `figure()` command. Thus for the example belo, after the last line we are in Figure 9. We can go back to Figure 1 by issuing in the command window a `figure(1)` MATLAB function call. We can then for example type `grid on;` to reveal a grid. Issuing a plot again will overwrite the previous one unless we specify `hold on`.

```
% File sub5p11.m : Case 3
x=-2*pi:pi/10:2*pi
y=sin(x);
z=cos(x);
figure(1);
plot(x,y,'r-');
figure(9);
plot(x,z,'b-');
```

(c) Copyright A. Gerbessiotis.

E6.1 `subplot.` The subplot command `subplot(m,n,p)` splits the screen into an $m \times n$ matrix of figure regions, numbered left-to-right top-to-bottom in row-major order, and inserts the plot into region indexed in row-major p. In Figure 12 the two subplots have different axes as well, since the range of x is different. If in two consecutive `sublot` commands, the geometry $m \times n$ differs from one `subplot` to the other, the dimensions of the last subplot overwrite the previous ones. The MATLAB program below will generate two subplots as shown in Figure 11.

```
% File sub5p12a.m : Subplots (same axes)
x=-2*pi:pi/10:2*pi
y=sin(x);
z=cos(x);
figure(1);
subplot(2,1,1);
plot(x,y,'r-');
title('subplot 1 title');
subplot(2,1,2);
plot(x,z,'b-');
title('subplot 2 title');

% File sub5p12b.m : Subplots (different axes)
x=-2*pi:pi/10:2*pi
y=sin(x);
figure(1);
subplot(2,1,1);
plot(x,y,'r-');
title('subplot 1 title');
x=-pi:pi/10:2*pi
z=cos(x);
subplot(2,1,2);
plot(x,z,'b-');
title('subplot 2 title');
```

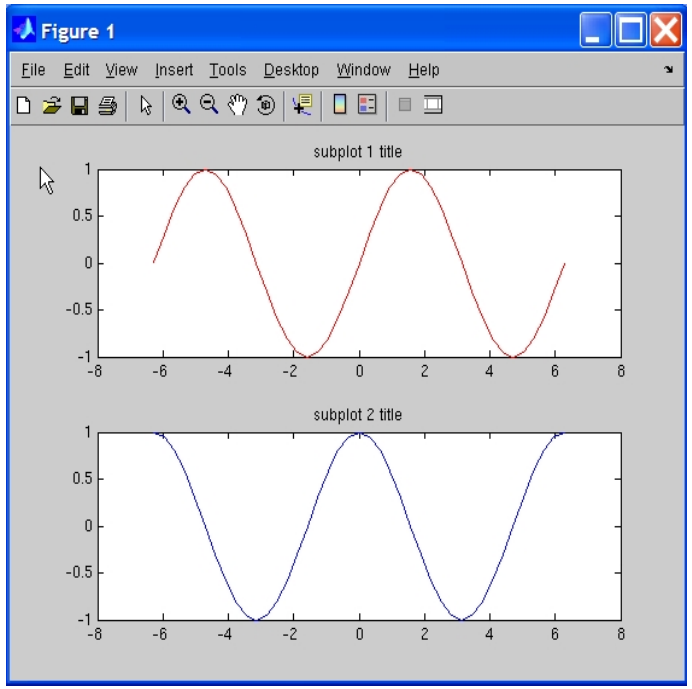


Figure 11: Subplots for $y = \sin(x), z = \cos(x)$ with same axes

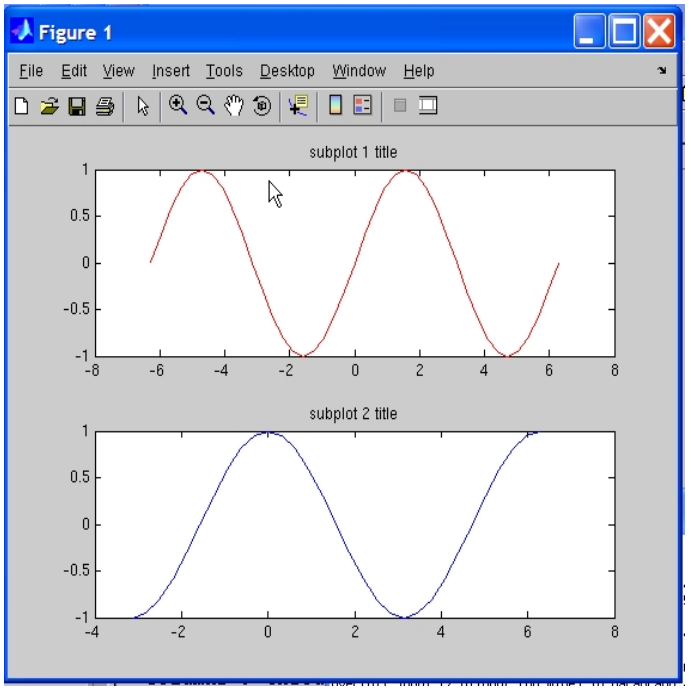


Figure 12: Subplots for $y = \sin(x), z = \cos(x)$ with different axes

(c) Copyright A. Gerbessiotis. All rights reserved.

E7.1 Plotted lines. Besides Marker-Style and Line-Style (see E2.1) other Marker and Line properties can be changed when one uses a plot command. Some of them are specified below and can be added to the plot command by first describing the string identifying the property, and then adding that property's value. The first subplot below with property `'b-o'` uses a 3-point solid-blue line, with the plot points marked with circles (the markers) of size 6-points, each marker having a red edge with a green face/center. The second subplot uses a solid black line of the same width (as the previous subplot's) with triangles pointing up of blue color.

```
LineWidth               : (pts)   Specifies width of a line in points.
MarkerEdgeColor         : (color) Specifies the color of a marker or edge color for filled
                                  markers
MarkerFaceColor         : (color) Color of face of filled markers
MarkerSize              : (pts)   Size of marker in points.
```

```
% File sub5p14.m : Plotted lines
x=-2*pi:pi/10:2*pi
y=sin(x);
figure(1);
subplot(2,1,1);
plot(x,y,'b-o','LineWidth',3,'MarkerSize',6,'MarkerEdgeColor','r','MarkerFaceColor','g');
title('subplot 1 title');
x=-pi:pi/10:2*pi
z=cos(x);
subplot(2,1,2);
plot(x,z,'k-^','LineWidth',3,'MarkerSize',6,'MarkerEdgeColor','b','MarkerFaceColor','r');
title('subplot 2 title');
```

(c) Copyright A. Gerbessiotis. All rights reserved.

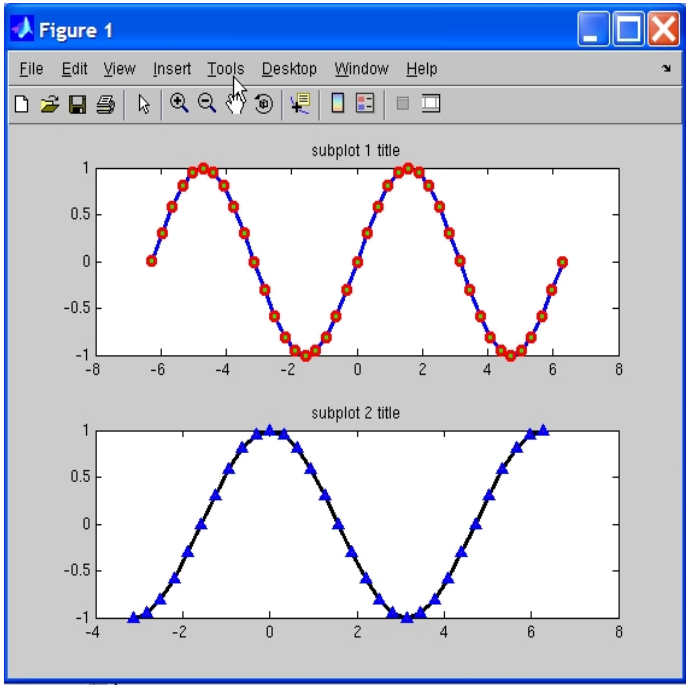


Figure 13: Subplots for $y = \sin(x), z = \cos(x)$ with different plotted line properties

(c) Copyright A. Gerbessiotis. All rights reserved.

E8.1 Text strings. The title of a subplot and the axis labels can be written with formatted text in a variety of fonts and sizes. One can also include Greek letters and mathematical symbols. This requires the use of `stream modifiers`. Thus one can generate also subscripts and superscripts such as A^{yyy}, B_{xxx}. Letters of the greek alphabet indicated below are printed as: $\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi o\pi\rho\sigma\tau\upsilon\phi\chi\psi\omega$. Note that all the symbols described here are borrowed from the text processing system known as LaTeX. Some additional characters/symbols include $\leq\neq\geq\ \infty\pm\ \sim\ \hat{}$.

```
\bf                : Boldface
\it                : Italics
\rm                : Roman font [default font]
\fontname{FontName}: Specify fontname as FontName
\fontsize{FontSize}: Specify font size (eg 10,11 12pt.)
B_{xxx}            : B Subscript xxx
A^{yyy}            : A Superscript yyy

\alpha \beta \gamma \delta \epsilon \zeta \eta \theta \iota \kappa \lambda
\mu \nu \xi o \pi \rho \sigma \tau \upsilon \phi \chi \psi \omega

\leq \neq \geq \infty \pm \sim \char`\^
```

```
% File sub5p16.m : Plotted lines
x=-2*pi:pi/10:2*pi
y=sin(x);
figure(1);
subplot(2,1,1);
plot(x,y,'b-o','LineWidth',3,'MarkerSize',6,'MarkerEdgeColor','r','MarkerFaceColor','g');
title('subplot 1 title');
text(pi,sin(pi),'\rightarrow\equiv Here i am!');
x=-pi:pi/10:2*pi
z=cos(x);
subplot(2,1,2);
plot(x,z,'k-^','LineWidth',3,'MarkerSize',6,'MarkerEdgeColor','b','MarkerFaceColor','r');
title('subplot 2 title');
```

(c) Copyright A. Gerbessiotis. All rights reserved.

E9.1 Polar plots. A plot that uses polar coordinates can be plotted using `polar(theta,r)`, where `theta` becomes an array of angles and `r` is an array of distances.

```
% sub5p17circle.m
subplot(1,1,1);
theta=0.0:pi/20:2*pi
r=ones(1,41);
polar(theta,r,'b-o');
```

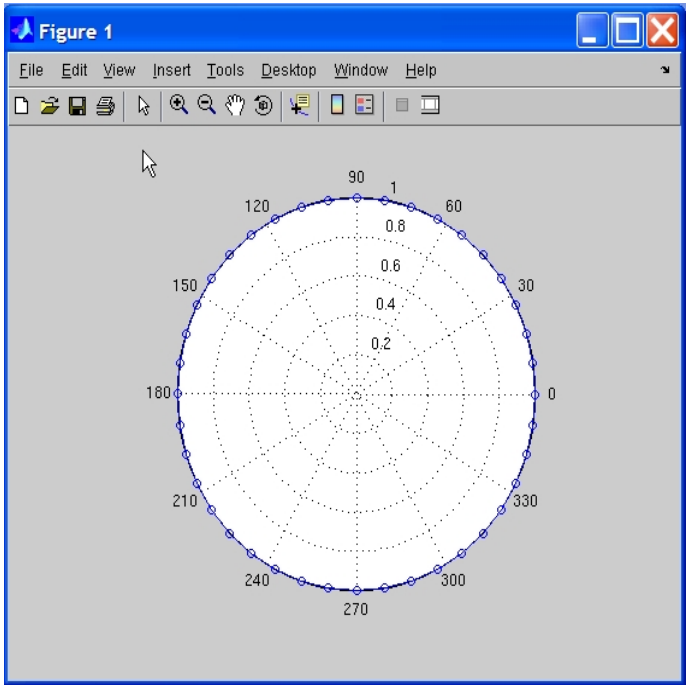


Figure 14: Polar plot