
output examples

Table of Contents

formatted display of results in command window	1
plotting 01 - basic plotting	1
plotting 02 - double y plots	4
plotting 03 - more plot options	5
plotting 04 - subplots	6
plotting 05 - put variable values into plot titles	7
user-written function	8
load and save to data files	10

formatted display of results in command window

```
clear all
clc

% fprintf is "file printing formatted" (print formatted text to a
% file)
% where a "file" is a device such as the command window (default)
% or a hard disk, tape, etc.

A = 12345.67890;
fprintf('This is my value of A \n') % note \n to start new line
fprintf('    A = %0.3f \n\n', A)

% the % starts a variable output format specification
% the 0 or other value is minimum width of text in which value is
% displayed
% the .3 or other value means round to 3 digits to right of decimal
% point
% the f means floating point (decimal number)

% as usual, enter "help fprintf" for more options

This is my value of A
    A = 12345.679
```

plotting 01 - basic plotting

```
x = linspace(-pi,pi,10);
y = sin(x);

figure(1)
plot(x,y)
title('my first plot','FontSize',14)
```

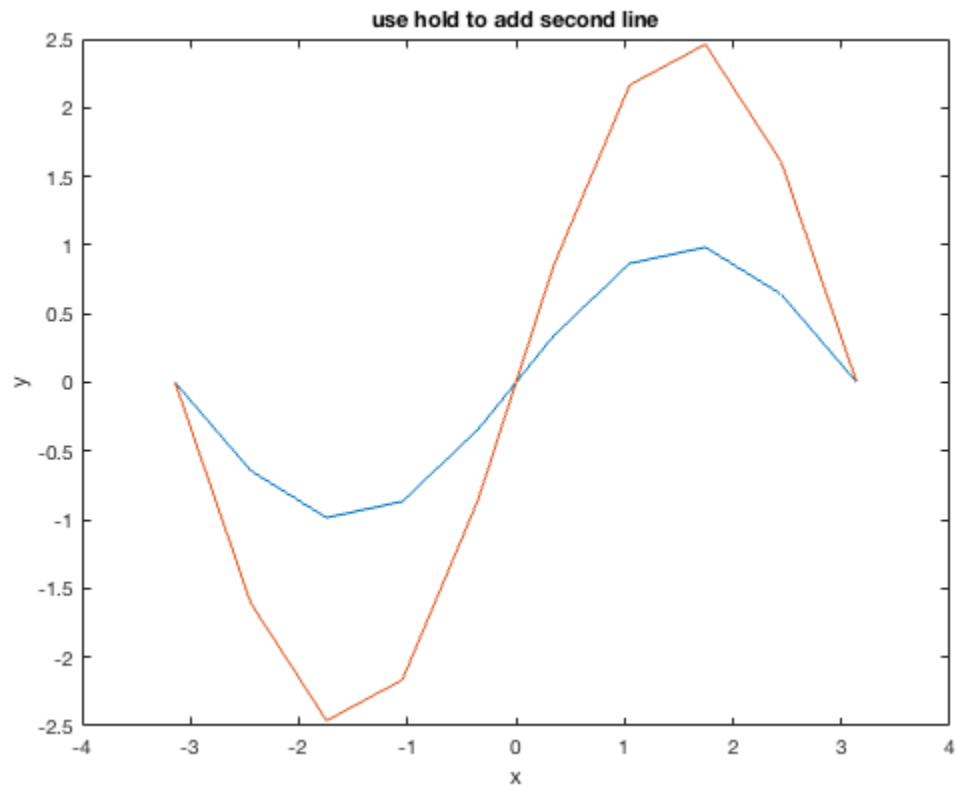
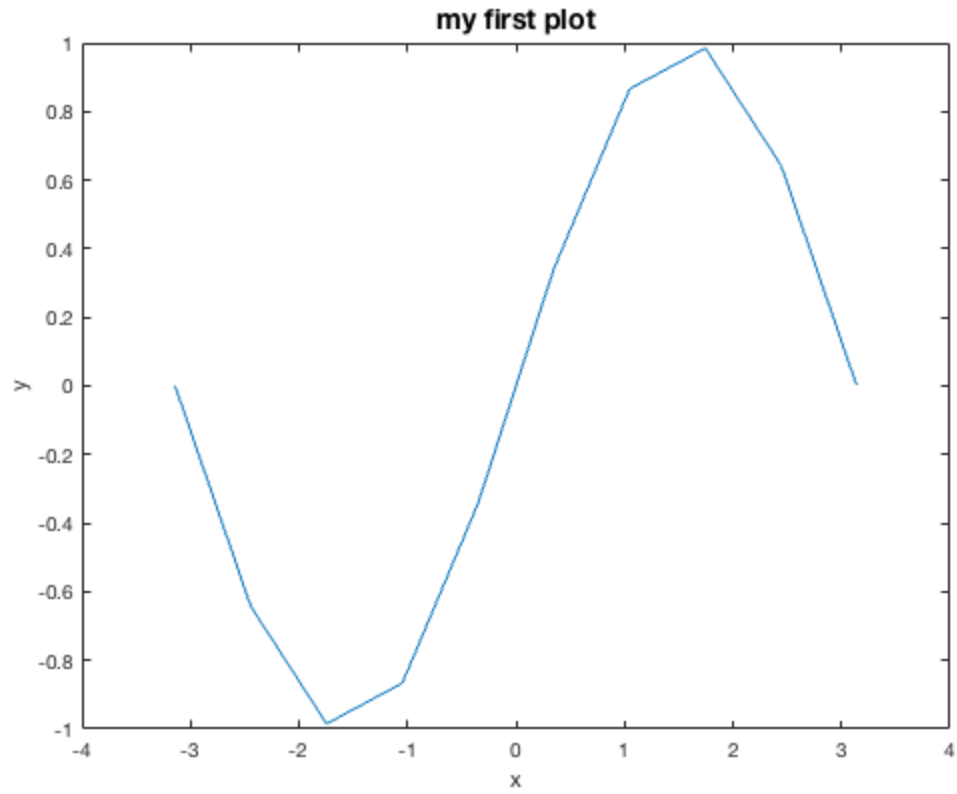
```
ylabel('y')
xlabel('x')

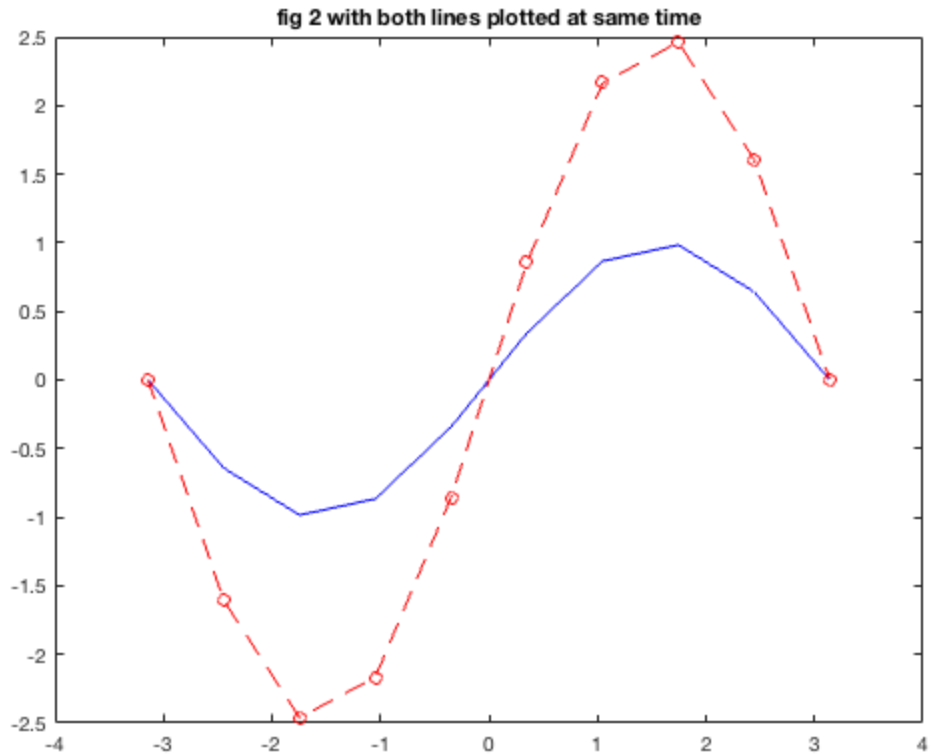
% compare size of title and axis label text on the plot
% could also change the FontSize of axis labels

% now we create a second y variable
y2 = 2.5*y;

% two ways to add y2 to the plot
% (1) use hold command
figure(2)
plot(x,y)
title('use hold to add second line')
ylabel('y')
xlabel('x')
hold on
plot(x,y2)
hold off

% (2) plot both at once, e.g., in a new figure window
% this also illustrates adding a plot format string,
% which can be done after any x,y pair, here as examples,
% b = blue, r = red, - = solid line, -- = dashed line, o = circles
figure(3)
plot(x,y,'b-',x,y2,'ro--')
title('fig 2 with both lines plotted at same time')
```





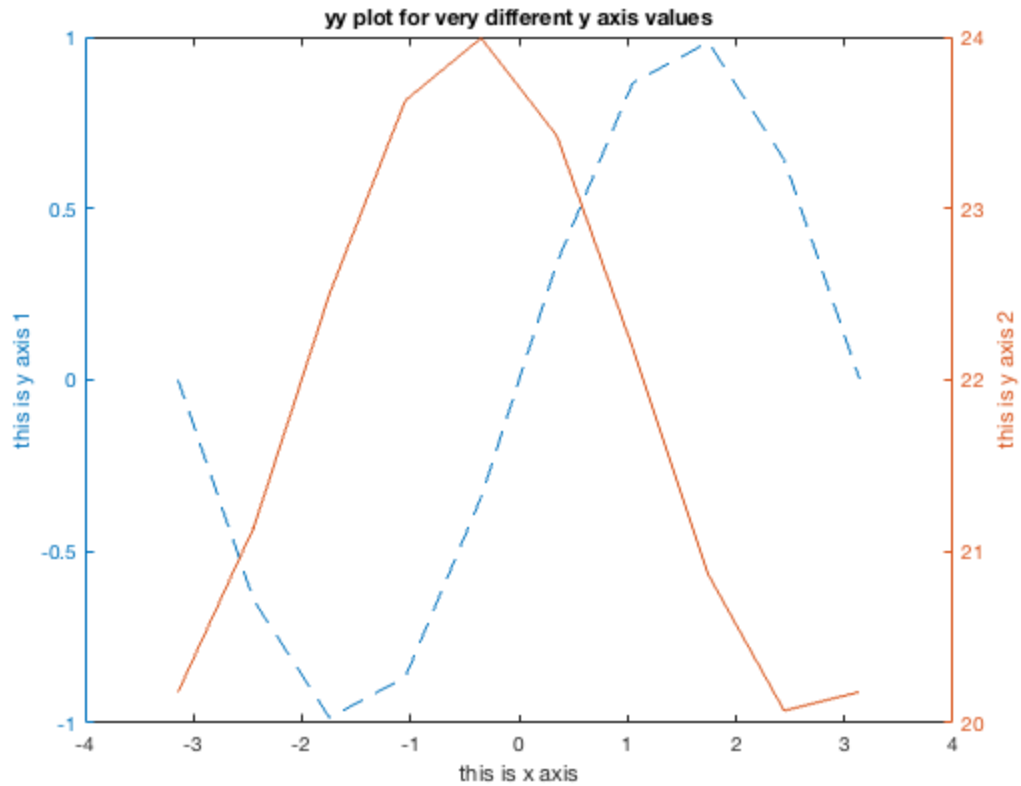
plotting 02 - double y plots

```
% you can plot data of very different scale on a double y plot
% on the same plot using plotyy

y2 = 22 + 2*sin(x+2);

% get "handles" (references) to axis, line1 and line2 returned
% if you want to add labels or change line color or style
% axis handle is 2-element array, (1) for right axis, (2) for left
% in Command window, enter >> doc axes properties
[haxis,hline1,hline2] = plotyy(x,y,x,y2);
hline1.LineStyle = '--';
haxis(1).YLabel.String = 'this is y axis 1';
haxis(2).YLabel.String = 'this is y axis 2';
title('yy plot for very different y axis values')
xlabel('this is x axis')

% as usual, enter "help plotyy" for more options
```



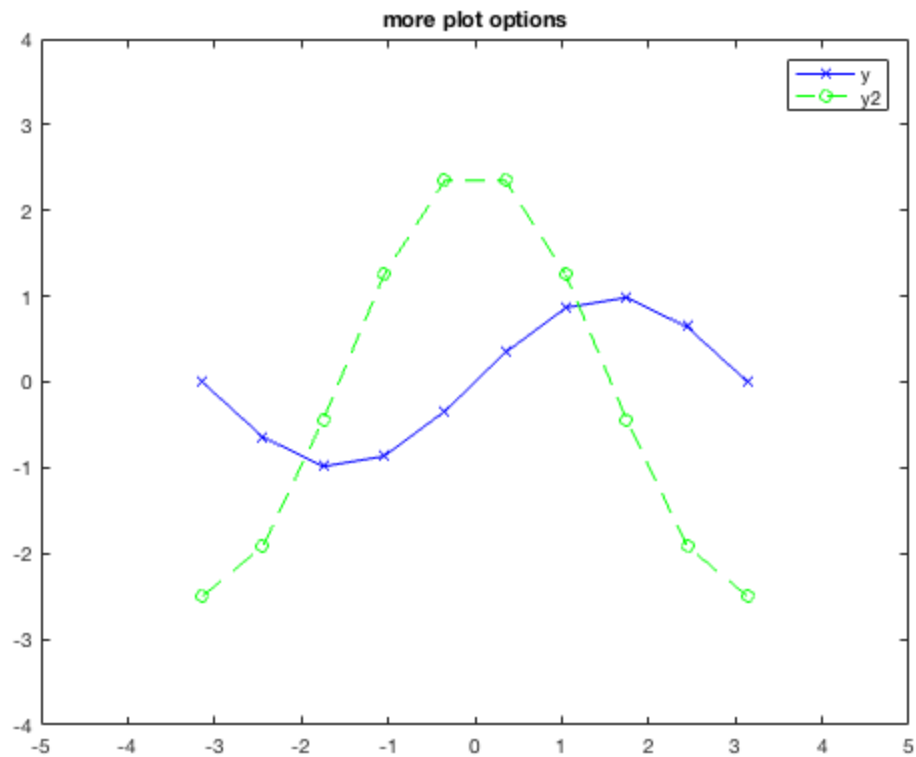
plotting 03 - more plot options

```
y2 = 2.5*cos(x);
```

```
% you can optionally specify plot color, symbols and lines  
plot(x,y, 'bx-',x,y2, 'go--')  
title('more plot options')
```

```
% the axis is set automatically to include all data on the plot  
% BUT you can change the axis ranges  
axis([-5 5 -4 4])
```

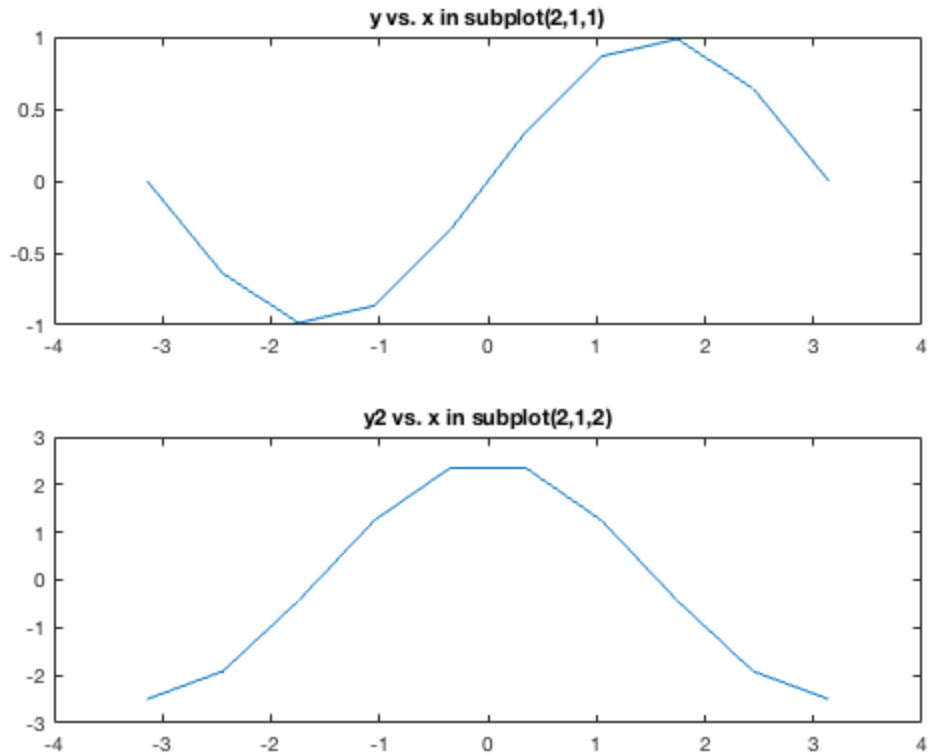
```
% you can add a legend, as usual, enter "help legend" for more options  
legend('y', 'y2')
```



plotting 04 - subplots

`% you can put more than one plot into one plot window with subplots`

```
subplot(2,1,1) % subplot with 2 rows, 1 column, put plot in location 1
plot(x,y)
title('y vs. x in subplot(2,1,1)')
subplot(2,1,2)
plot(x,y2)
title('y2 vs. x in subplot(2,1,2)')
```



plotting 05 - put variable values into plot titles

```

% let's say we want to study the effect of A, omega and phi on our
plot
% and we want to see in the plot title the current values

A = 2.1; % amplitude
w = 2; % frequency
phi = pi/2; % phase shift

t = linspace(-pi,pi,1000);
y = A*sin(w*t + phi);

clf % clear figures - otherwise will appear in subplot from above

plot(t,y)

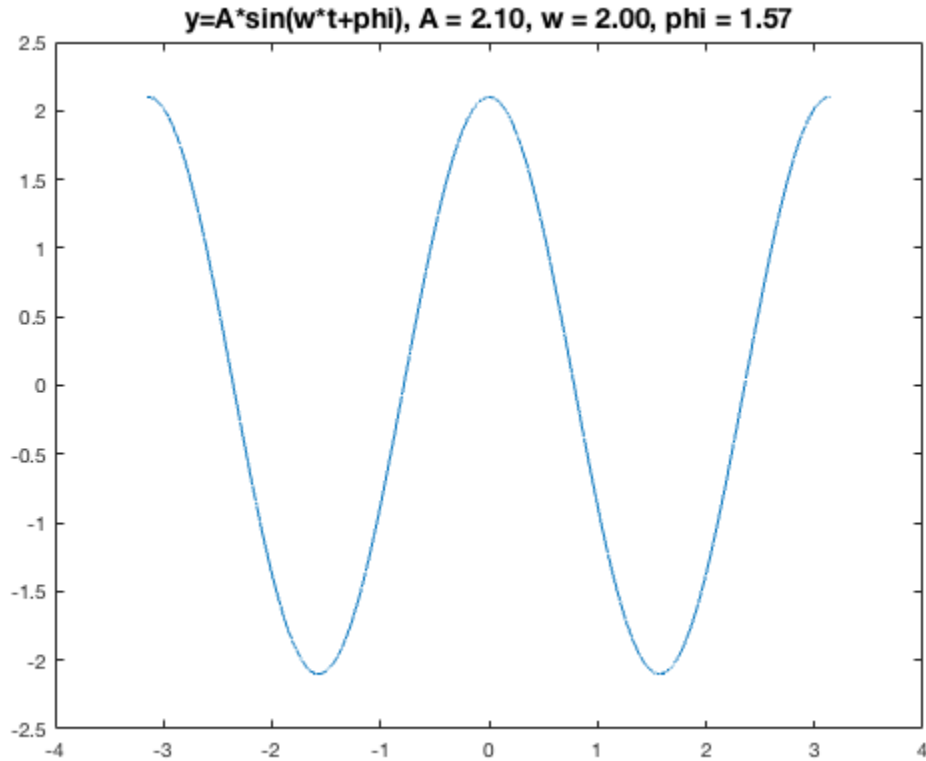
% assign a string variable using sprintf function
% where sprintf is string print formatted

tt = sprintf('y=A*sin(w*t+phi), A = %0.2f, w = %0.2f, phi =
%0.2f', ...
A,w,phi);

```

```
% note line split and continued with ... but can't split string
between ''

title(tt, 'FontSize', 14)
```



user-written function

```
% functions allow you to re-use code simply by "calling" the function
% a function might have 1 to 1000's of lines of code
% without functions, you might have to copy and paste the same code
% many times in a program
% but what if you wanted to make a change?
% you would have to search and replace many instances
% and hope you found everything
% with functions, you simply make the change in one place

% built-in Matlab functions are defined in m-files located
% in function "libraries" in your Matlab distribution
% Matlab calls function libraries "toolboxes"

x = linspace(-pi,pi,10);
a = 2;

% call user-written function in file myfunc.m
y = myfunc(x,a);
```



```
plot(x,y)

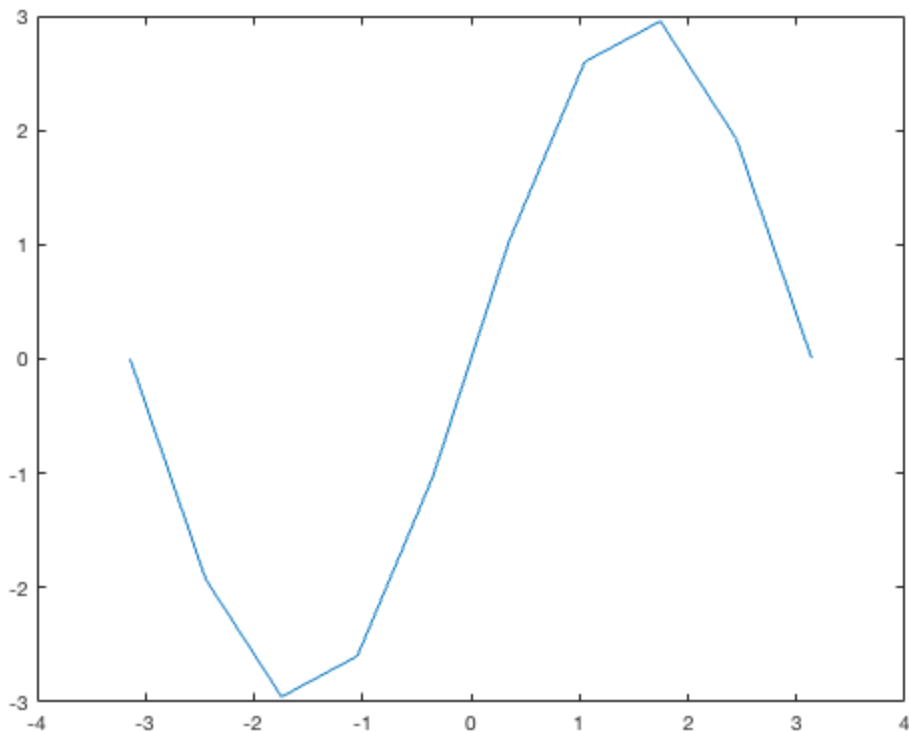
% for PUBLISH type function so I can see it in PDF for grading
type myfunc.m

function z = myfunc(t,w)
    % comment as 2nd line starts documentation of function
    % which can be read from command line with, e.g., help myfunc

% first line of a function file is, from left to right
% keyword "function"
% variable value returned to calling script (scalar or matrix)
% = sign
% name of function, which must be name of .m file
% list of inputs (scalar or array)

% note variable names can differ between main script and function
% values in function are "local" in "scope" and
% values are passed by location in argument list, not by name

w = 1.5 * w;
z = w * sin(t);
```



load and save to data files

```
% some data
a = [1 2 3; 4 5 6];

% save to file 'adata.txt' variable a with option -ascii
save adata.txt a -ascii

% where ascii = American Standard Code for Information Interchange
% without this option the file will be in binary format
% and load would put data into a 'structure' which
% we haven't learned about yet
% using .txt filename extension for ascii data
% allows you to double-click on file on desktop to
% open and inspect with text editor, e.g., Windows Notepad

% now check to make sure we can read the data
anew = load('adata.txt')

% optionally, for PUBLISH, we can "type" the file
% but ONLY if saved with the -ascii option
% PUBLISH to PDF fails when file is in binary format
type adata.txt
```

```
anew =
```

```
     1     2     3
     4     5     6
```

```
1.0000000e+00  2.0000000e+00  3.0000000e+00
4.0000000e+00  5.0000000e+00  6.0000000e+00
```

Published with MATLAB® R2016a