Plotting Continued:

You can change the line style by adding some information in the **plot** command within single quotation marks.

```
x = (0:1:10);
y = x.^2;
plot(x,y, '-xr')
xlabel('x, meters')
ylabel('y, meters squared')
title('A simple plot');
grid on;
```

- changes the *line type* to solid
x puts x-marks at the *data points*
r changes the *line color* to red

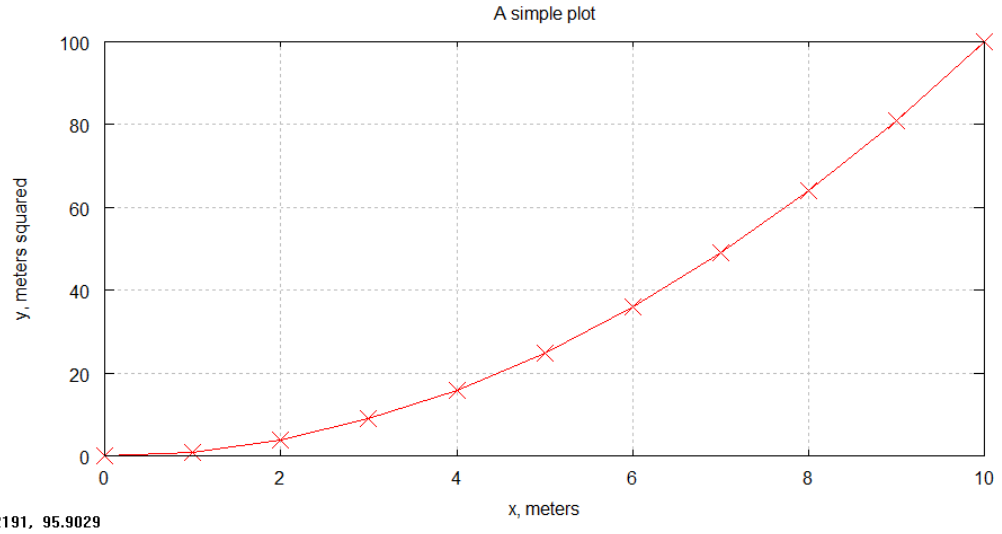


Table 5.2 (page 147) in your book shows you the various line type, point type, and color options. Unfortunately, some of the line types may not be available in Octave.

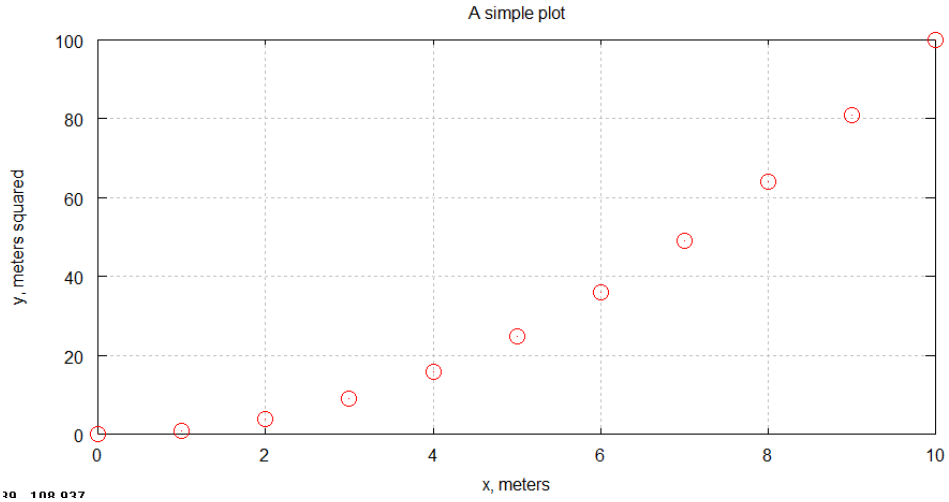
Line Type:	Point Type:	Color:
- solid	. point	b blue
: dotted	x x-mark	g green
-. dash-dot	+ plus	r red
-- dashed	* star	k black

Plotting Points (no line):

Sometimes you just want to plot data points without a line connecting them. Simply omit the line type in the format string.

In m-file:

```
x = (0:1:10);
y = x.^2;
plot(x,y,'or')   % NOTICE: The dashes have been omitted.
xlabel('x, meters')
ylabel('y, meters squared')
title('A simple plot');
grid on;
```



Multiple plots on same figure:

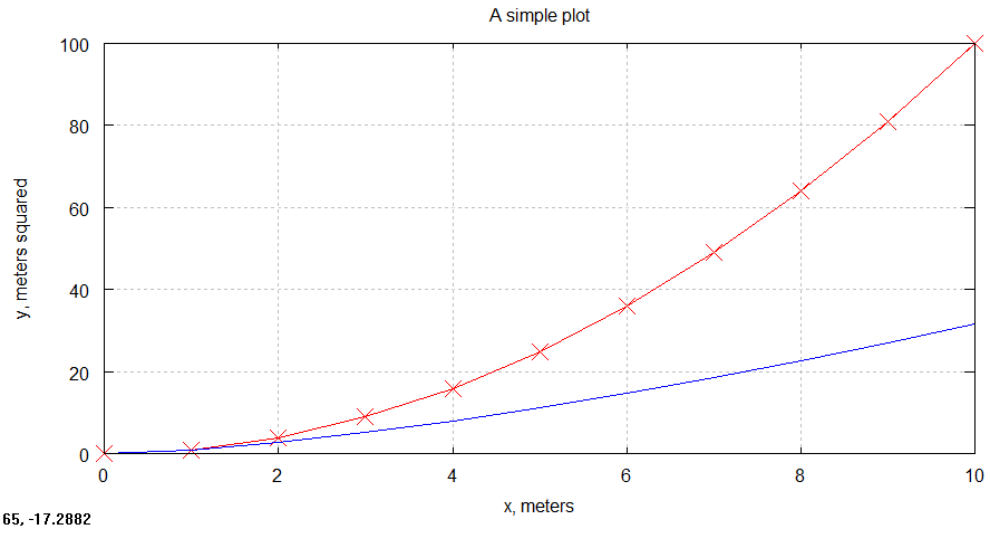
Often you want to compare data on the same figure. There are a couple ways to do this.

(1) Using the **hold on** command.

Every time you call the **plot( )** command, a new figure is created. The **hold on** command will prevent a new figure from overwriting the previous figure. More figures can be layered until the **hold off** command is issued.

In main.m:

```
x = (0:1:10);
y = x.^2;
z = x.^(3/2)
plot(x,y,'-xr')
xlabel('x, meters')
ylabel('y, meters squared')
title('A simple plot');
grid on;
hold on;
plot(x,z);
hold off;
```



If I did not include the **hold on** command, only the blue line would be seen.

There is a way to plot multiple sets of data on the same graph without using the **hold on/hold off** commands.

(2A) Put multiple output in a single **plot( )** command – General form.

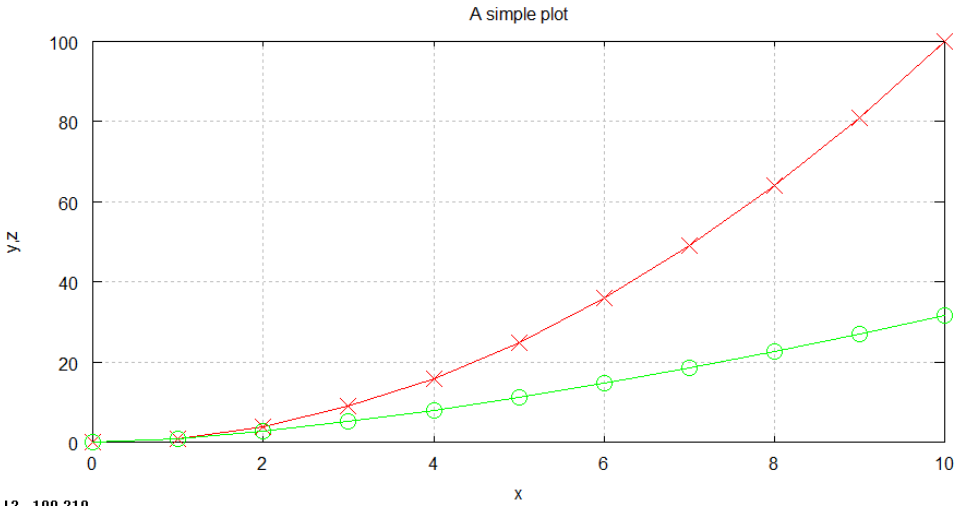
You can use a single **plot( )** command to do the same task. The general form is,

plot(x1, y1, 'formatting for plot1' , x2, y2, 'formatting for plot2' , etc…)

Returning to our previous example,

In main.m:

```
x = (0:1:10);
y = x.^2
z = x.^(3/2)
plot(x,y, ':xr' , x, z, '-og')
xlabel('x')
ylabel('y,z')
title('A simple plot');
grid on;
```

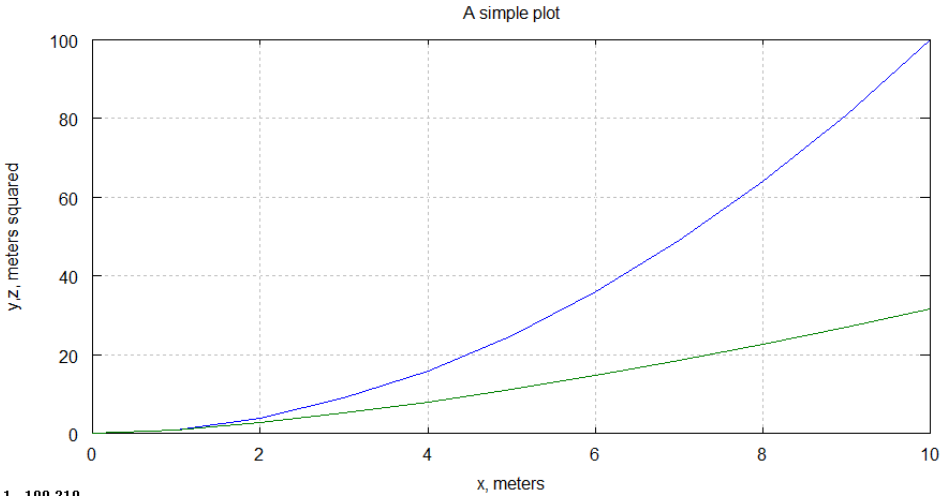


(2B) Put multiple output in a single **plot( )** command – Using same x values

If your x values are the same for multiple sets of data, such as in the example above, you can simplify the **plot( )** command by storing all the different output in a single matrix.

In main.m:

```
x = (0:1:10);
y = x.^2
z = x.^(3/2)
OUTPUT = [y ; z];  % The different data must be placed in different rows.
plot(x,OUTPUT)
xlabel('x')
ylabel('y,z')
title('A simple plot');
grid on;
```



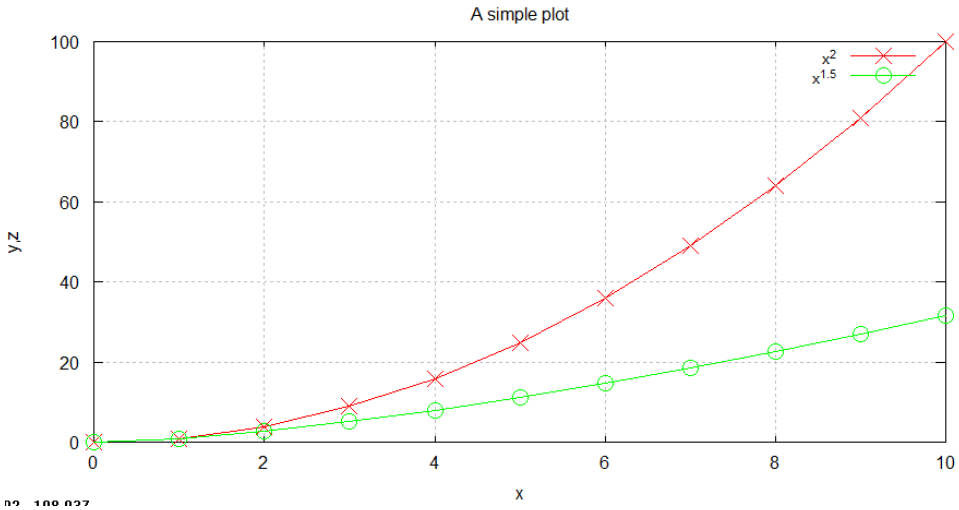
Legends:

When you put multiple sets of data on the same figure, you often want to include a legend.

legend('data1 info' , 'data2 info' , etc…)

Example:

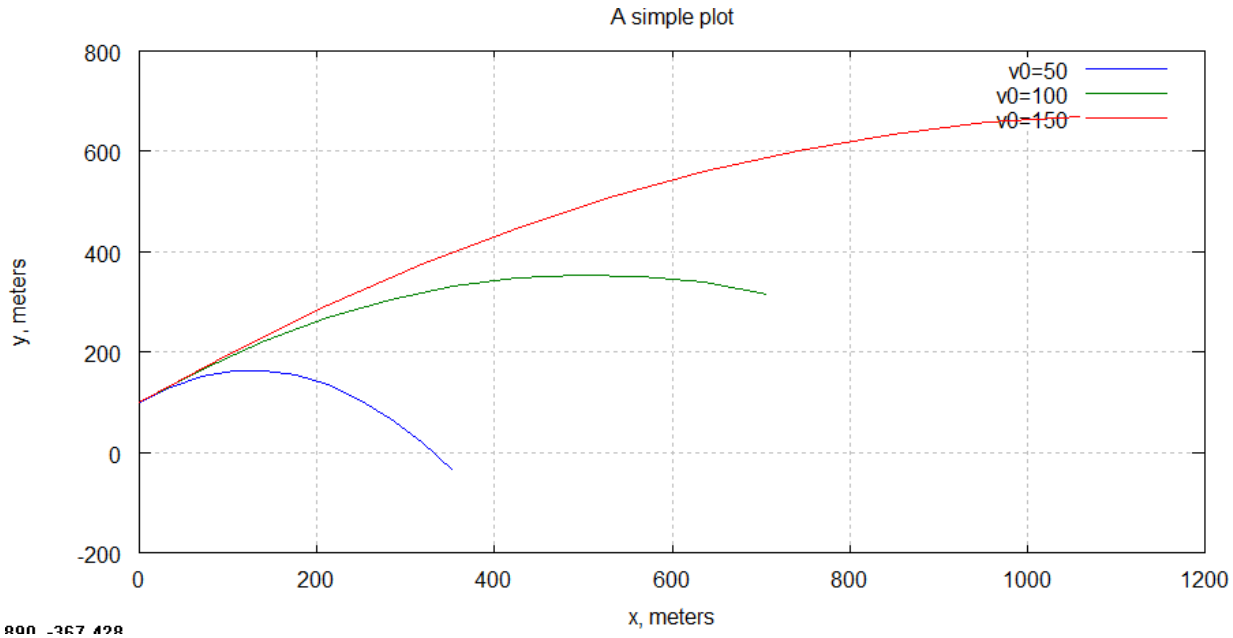
```
x = (0:1:10);
y = x.^2
z = x.^(3/2)
plot(x,y, ':xr' , x, z, ':og')
xlabel('x')
ylabel('y,z')
title('A simple plot');
legend('x^{2}' , 'x^{1.5}' )
grid on;
```



Example: Plot the trajectory of cannon balls shot off a cliff with different v0. The cliff is at (0,100).

In m-file:

```
x0 = 0;  y0 = 100;
g = -9.81;
angle = 45;
t = (0:1:10);

v0 = 50;
v0x = v0*cos(angle*pi/180);
v0y = v0*sin(angle*pi/180);
x1 = x0 + v0x.*t ;
y1 = y0 + v0y.*t + 0.5*g*t.^2 ;

v0 = 100;
v0x = v0*cos(angle*pi/180);
v0y = v0*sin(angle*pi/180);
x2 = x0 + v0x.*t ;
y2 = y0 + v0y.*t + 0.5*g*t.^2 ;

v0 = 150;
v0x = v0*cos(angle*pi/180);
v0y = v0*sin(angle*pi/180);
x3 = x0 + v0x.*t ;
y3 = y0 + v0y.*t + 0.5*g*t.^2 ;

plot(x1,y1,x2,y2,x3,y3)
xlabel('x, meters');
ylabel('y, meters');
title('A simple plot');
legend('v0=50', 'v0=100', 'v0=150');
grid on;
```


A simple plot
v0=50
v0=100
v0=150
y, meters
x, meters
800
600
400
200
0
-200
0
200
400
600
800
1000
1200

890 -367.428

Subplots:

Another way to present multiple sets of data on the same figure is using subplots. This command allows you to subdivide the graphing window into a grid of M rows and N columns.

subplot(M,N,p)

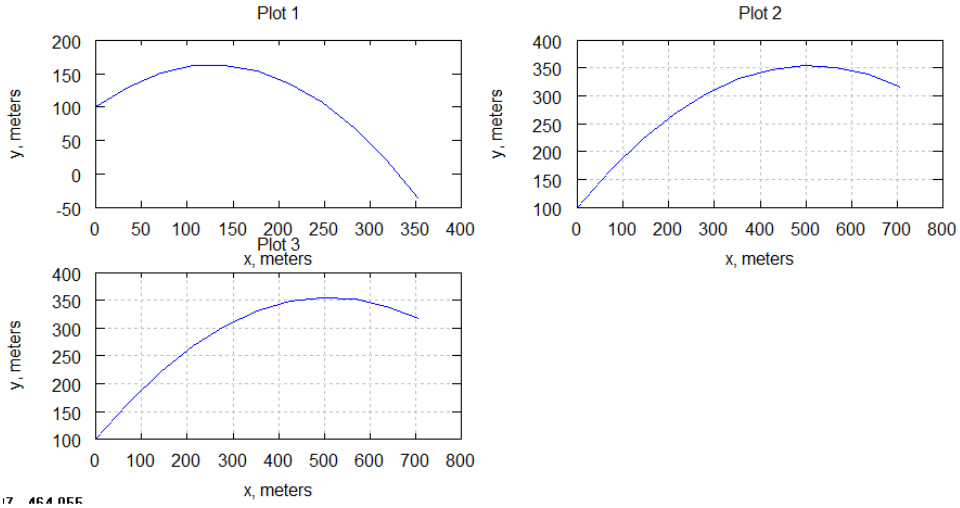
The variable **p** identifies the position where the subplot will be drawn. The value of **p** is increased in row order. For example, if M and N are both 2, then there will be 4 locations for subplots and then the value of **p** at each of those locations is,

p=1	p=2
p=3	p=4

In m-file:

```
clear;clc;
x0 = 0;  y0 = 100;
g = -9.81;
angle = 45;
t = (0:1:10);

v0 = 50;
v0x = v0*cos(angle*pi/180);
v0y = v0*sin(angle*pi/180);
x1 = x0 + v0x.*t ;
y1 = y0 + v0y.*t + 0.5*g*t.^2 ;

v0 = 100;
v0x = v0*cos(angle*pi/180);
v0y = v0*sin(angle*pi/180);
x2 = x0 + v0x.*t ;
y2 = y0 + v0y.*t + 0.5*g*t.^2 ;

v0 = 150;
v0x = v0*cos(angle*pi/180);
v0y = v0*sin(angle*pi/180);
x3 = x0 + v0x.*t ;
y3 = y0 + v0y.*t + 0.5*g*t.^2 ;

subplot(2,2,1)
plot(x1,y1)
xlabel('x, meters');
ylabel('y, meters');
title('Plot 1');
grid off;
```

```
subplot(2,2,2)
plot(x2,y2)
xlabel('x, meters');
ylabel('y, meters');
title('Plot 2');
grid on;

subplot(2,2,3)
plot(x2,y2)
xlabel('x, meters');
ylabel('y, meters');
title('Plot 3');
grid on;
```



We could have used a **for** loop to do the same task:

In m-file:

```
x0 = 0;  y0 = 100;
g = -9.81;
angle = 45;
t = (0:1:10);

v0 = 50;
for i=1:3
   v0x = v0*cos(angle*pi/180);
   v0y = v0*sin(angle*pi/180);
   x = x0 + v0x.*t ;
   y = y0 + v0y.*t + 0.5*g*t.^2 ;
   subplot(2,2,i)
   plot(x,y)
   xlabel('x, meters');
   ylabel('y, meters');
   titletxt = ['Plot ' , num2str(i)];
   title(titletxt);
   grid on;
   v0 = v0 + 50;
end
```