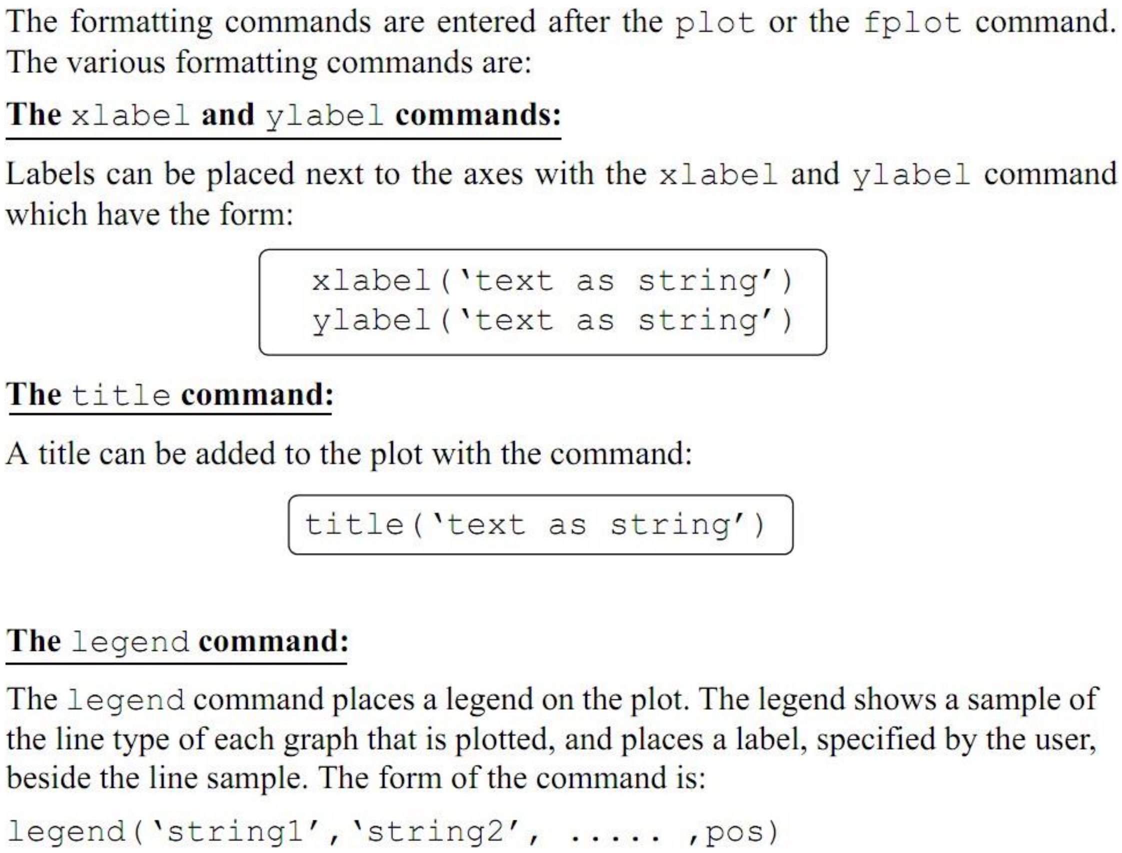
4.5 Formatting a Plot Using Commands



The formatting commands are entered after the plot or the fplot command. The various formatting commands are:

The xlabel and ylabel commands:

Labels can be placed next to the axes with the xlabel and ylabel command which have the form:

```
xlabel(‘text as string’)
ylabel(‘text as string’)
```

The title command:

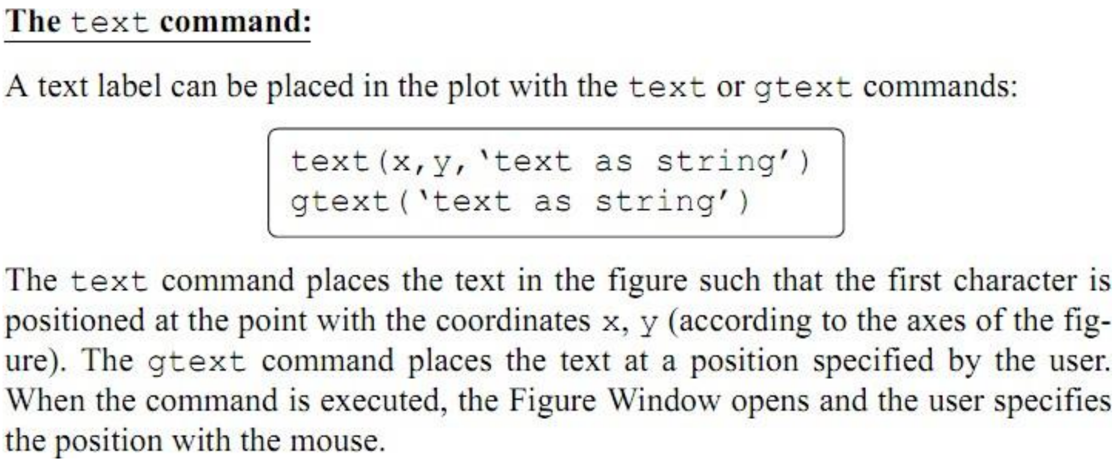
A title can be added to the plot with the command:

```
title(‘text as string’)
```

The 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 form of the command is:

```
legend(‘string1’,‘string2’, ..... ,pos)
```



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. When the command is executed, the Figure Window opens and the user specifies the position with the mouse.

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 and the appearance of the axes. In many situations a graph looks better if the range of the axes extend beyond the range of the data. The following are some of the possible forms of the `axis` command:

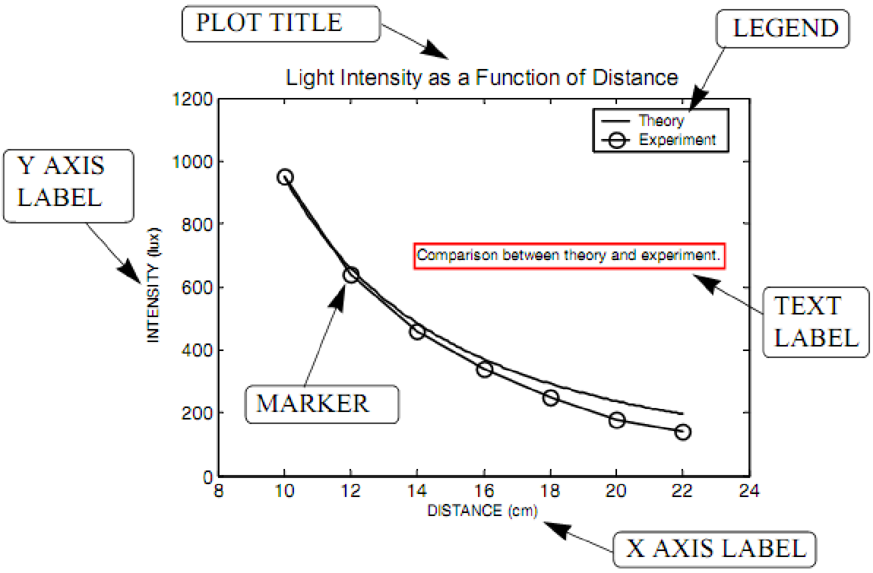
`axis([xmin,xmax,ymin,ymax])` Sets the limits of both the *x* and *y* axes (`xmin`, `xmax`, `ymin`, and `ymax` are numbers).

`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.

An example of formatting a plot by using commands is given in the following script file which was used to generate the formatted plot:

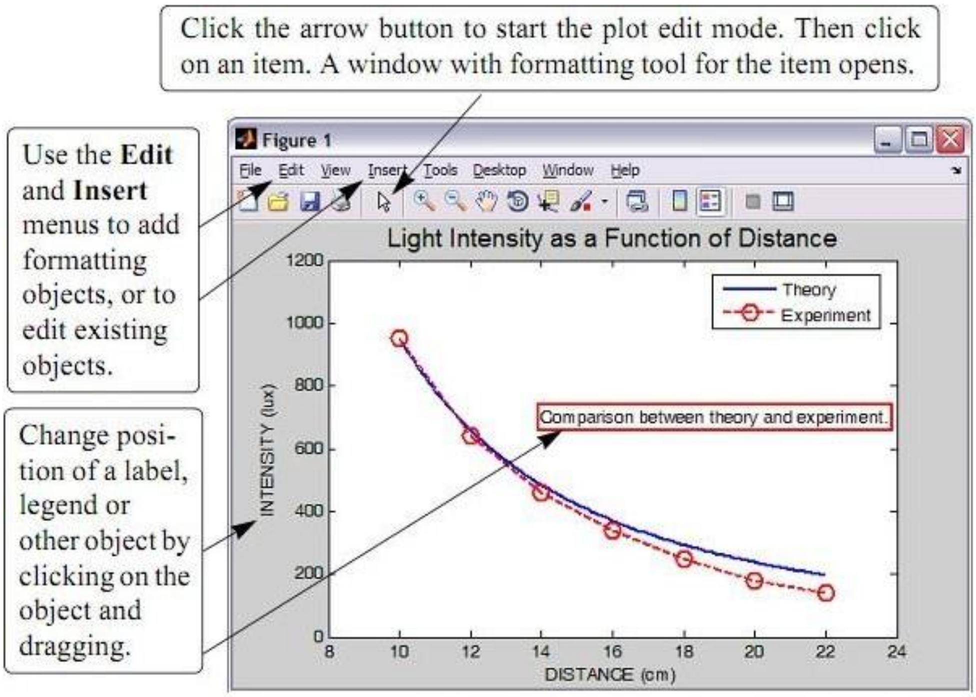


The code:

```
x=[10:0.1:22];
y=95000./x.^2;
xd=[10:2:22];
yd=[950  640  460  340  250  180  140];
plot(x,y,'-','LineWidth',1.0)
xlabel('DISTANCE (cm)')
ylabel('INTENSITY (lux)')
title('\fontname{Arial}Light  Intensity  as  a Function of
Distance','FontSize',14)
axis([8 24 0 1200])
text(14,700,'Comparison between theory and
experiment.','EdgeColor','r','LineWidth',2)
hold on
plot(xd,yd,'ro--','linewidth',1.0,'markersize',10)
legend('Theory','Experiment',0)
hold off
```

4.6 Formatting a Plot Using the Plot Editor

A plot can be formatted interactively in the Figure Window by clicking on the plot and/or using the menus. Figure 5-8 shows the Figure Window with the plot of Figure. The Plot Editor can be used to introduce new formatting items or to modify formatting that was initially introduced with the formatting commands.



Click the arrow button to start the plot edit mode. Then click on an item. A window with formatting tool for the item opens.

Use the **Edit** and **Insert** menus to add formatting objects, or to edit existing objects.

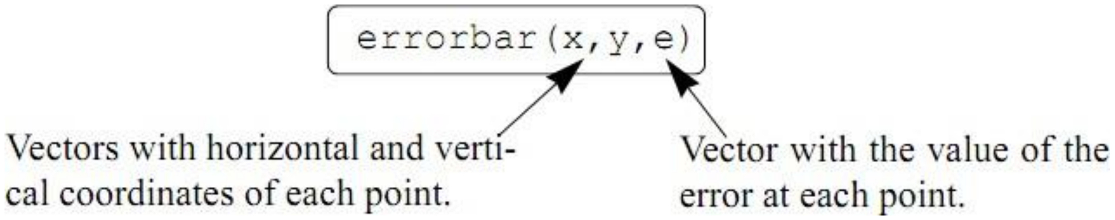
Change position of a label, legend or other object by clicking on the object and dragging.

Formatting a plot using the Plot Editor.

4.7 PLOTS WITH SPECIAL GRAPHICS

1. PLOTS WITH ERROR BARS

Plots with error bars can be done in MATLAB with the error bar command. Two forms of the command, one for making plots with symmetric error, the error bar extends the same length above and below the data point and the command has the form:



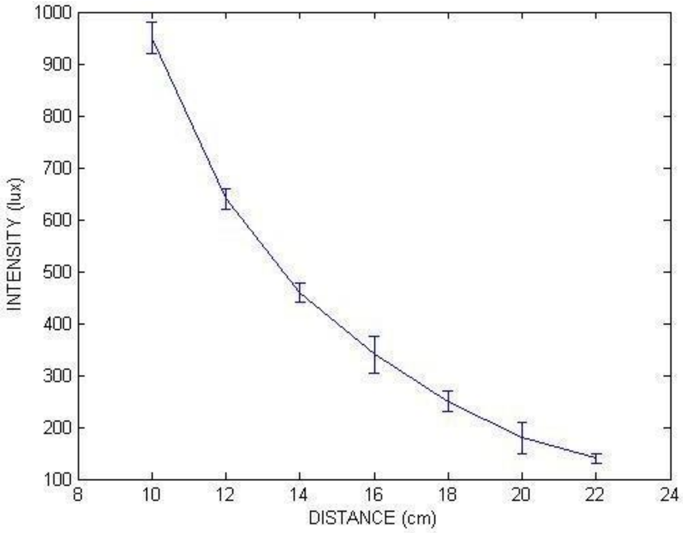
```
errorbar(x,y,e)
```

Vectors with horizontal and vertical coordinates of each point.

Vector with the value of the error at each point.

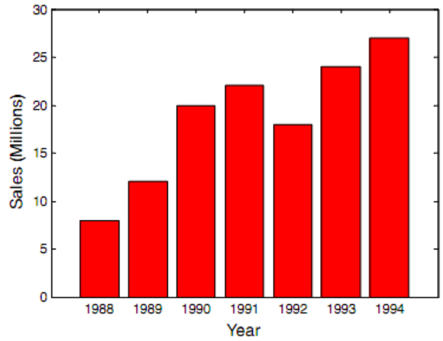
The plot in Figure, which has symmetric error bars, was done by executing the following code:

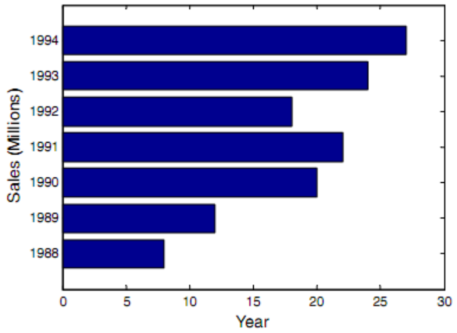
```
xd=[10:2:22];
yd=[950 640 460 340 250 180 140];
ydErr=[30 20 18 35 20 30 10]
errorbar(xd,yd,ydErr)
xlabel('DISTANCE (cm)')
ylabel('INTENSITY (lux)')
```



2. Bar (vertical and horizontal) plot

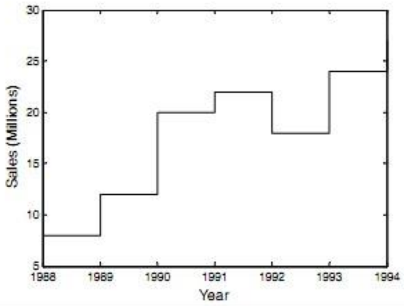
Bar (vertical and horizontal), are presented in the following charts using the sales data**:**

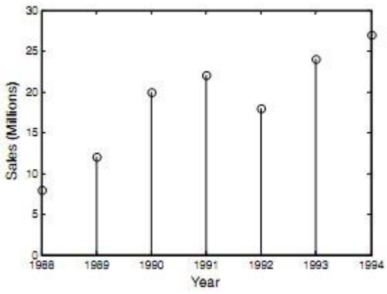
Vertical Bar Plot
Function format:
`bar(x,y)`
		`yr=[1988:1994];`
		`sle=[8 12 20 22 18 24 27];`
		`bar(yr,sle,'r')` ← The bars are in red.
		`xlabel('Year')`
		`ylabel('Sales (Millions)')`

Horizontal Bar Plot
Function format:
`barh(x,y)`
		`yr=[1988:1994];`
		`sle=[8 12 20 22 18 24 27];`
		`barh(yr,sle)`
		`xlabel('Sales (Millions)')`
		`ylabel('Year')`

3. Stairs and stem plot

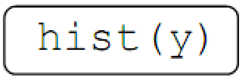
As following:

Stairs Plot
Function format:
`stairs(x,y)`
		`yr=[1988:1994];`
		`sle=[8 12 20 22 18 24 27];`
		`stairs(yr,sle)`

Stem Plot
Function Format
`stem(x,y)`
		`yr=[1988:1994];`
		`sle=[8 12 20 22 18 24 27];`
		`stem(yr,sle)`

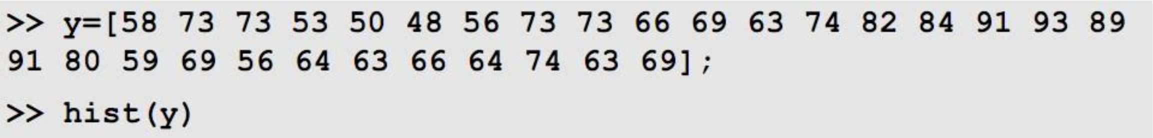
4. Histogram plot

Histograms are created in MATLAB with the hist command. The simplest form of the command is:

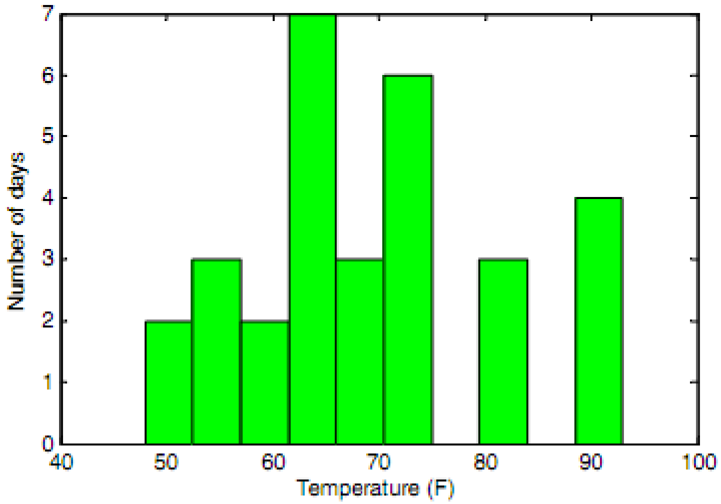


```
hist(y)
```

y is a vector with the data points. MATLAB divides the range of the data points into 10 equally spaced sub ranges (bins) and then plots the number of data points in each bin. For example, the following data points are the daily maximum temperature (in F) in Washington, DC, during the month of April 2002: 58 73 73 53 50 48 5673 73 66 69 63 74 82 84 91 93 89 91 80 59 69 56 64 63 66 64 74 63 69 (data from the U.S. National Oceanic and Atmospheric Administration). A histogram of this data is obtained with the commands:



```
>> y=[58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 91 93 89
91 80 59 69 56 64 63 66 64 74 63 69];
>> hist(y)
```


Number of days
Temperature (F)