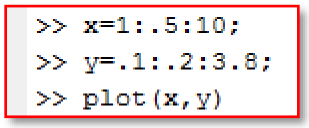
# MATLAB: *Plots*

In this tutorial, the reader will learn about obtaining graphical output. Creating a proper engineering plot is not an easy task. It takes lots of practice, because the engineer is trying to describe the system and its behavior on one plot. That means enough information must be presented on the plot, such as title, axes labels, units, etc., but not too much data that makes the plot cluttered and hard to interpret. Proper font size, grids, markers, line type are just few examples of decision you are faced with while create a proper plot. Unfortunately, the purpose of this tutorial is just to show you how to create a plot using MATLAB.

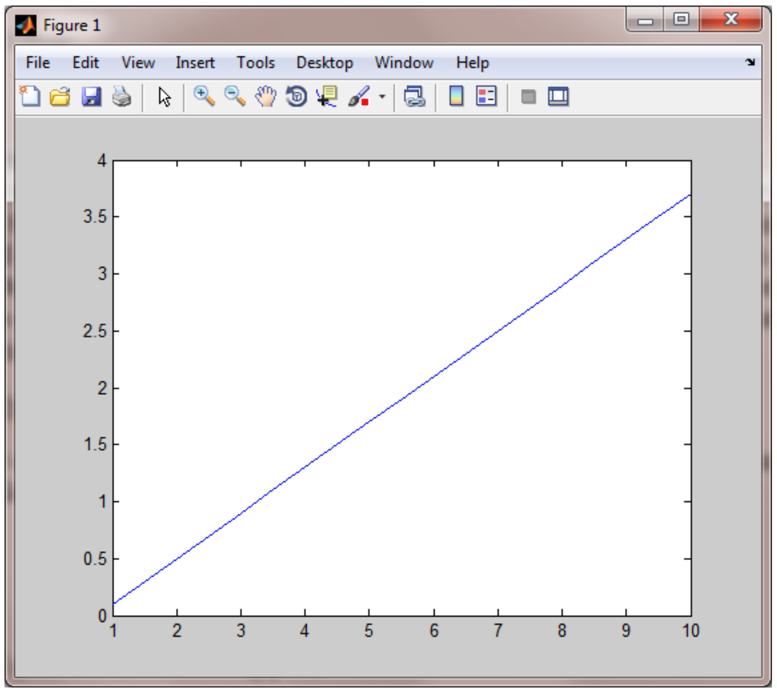
The ***plot(x,y)*** command

***Plot(x,y)*** command creates linear x-y plots, where x is the array that contains all data for the x-axis, while y is the array containing all y-axis data. For example:

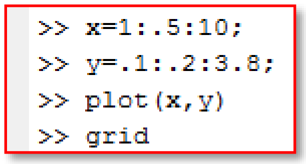


```
>> x=1:.5:10;
>> y=.1:.2:3.8;
>> plot(x,y)
```

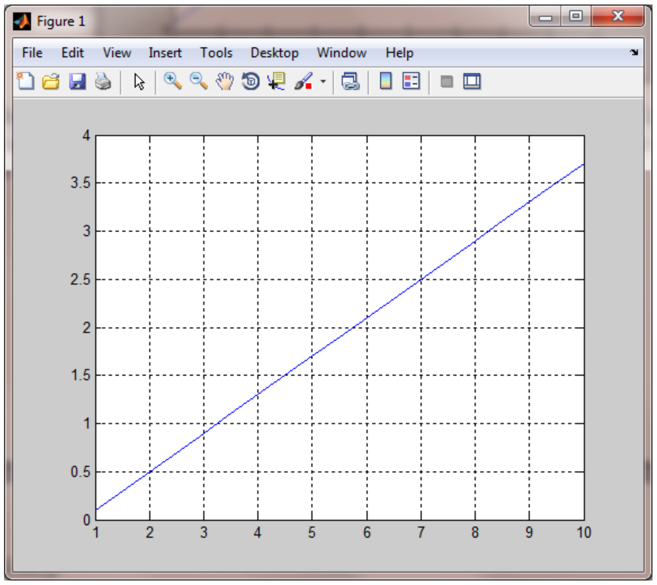
This will generate the following plot:



A seen above, the plot does not have any title, x-axis label or units, y-axis label or units, grids, and no markers at each point. To add ***grid*** to the plot in the command window, use the grid plot and press **ENTER**.



```
>> x=1:.5:10;
>> y=.1:.2:3.8;
>> plot(x,y)
>> grid
```

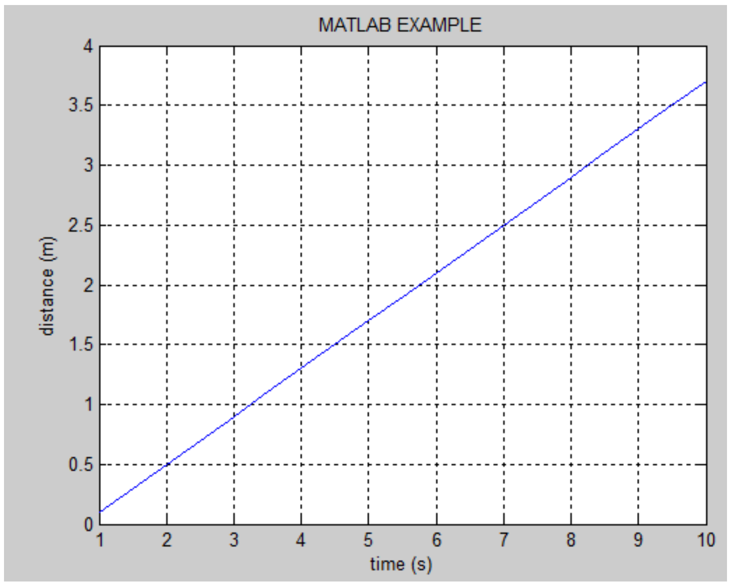


After displaying the grids, if it does not fit with your presentation style (like me), you can use ***grid off*** and press **ENTER** to remove the grid.

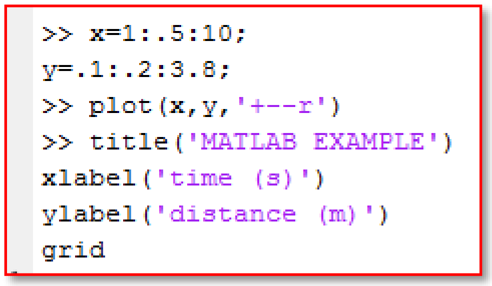
```
>> grid off
```

To add a plot title, use ***title('text')*** command, where text would be replaced with plot title, such as:

```
>> x=1:.5:10;
>> y=.1:.2:3.8;
>> grid
>> title('MATLAB EXAMPLE')
>> xlabel('time (s)')
>> ylabel('distance (m)')
```



To show the marker (points), line color and change the line style to dashed line instead of solid, use additional argument in ***plot()*** command, such as:



```
>> x=1:.5:10;
y=.1:.2:3.8;
>> plot(x,y,'+--r')
>> title('MATLAB EXAMPLE')
xlabel('time (s)')
ylabel('distance (m)')
grid
```

Note that the third argument in ***plot()*** command has been added, this argument is decoded such as: + indicate plus point style (refer to Table 2 below for different styles), **--** indicates dashed line style (see Table 1 below for other Line Styles), then **r** indicates the line color (refer to Table 3 below for some of the standard line colors). Once executing the code above, MATLAB will generate the following plot:

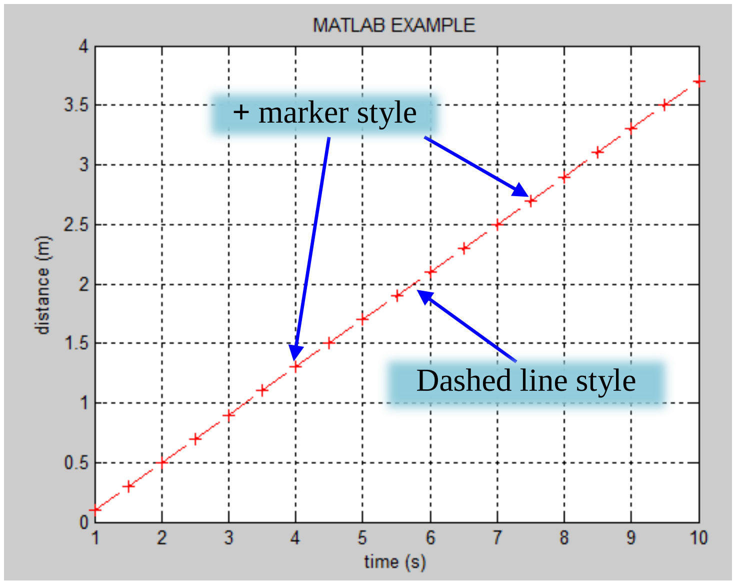


Table 1: Line Style

| Line Style | Symbol |
|---|---|
| Solid | - |
| Dashed | -- |
| Dotted | : |
| Dash-dot | -. |
| No line | none |

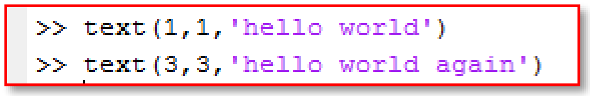
Table 2: Marker Style

| Marker Style | Symbol |
|---|---|
| Point | . |
| Plus | + |
| Star | * |
| Circle | O |
| X-mark | x |

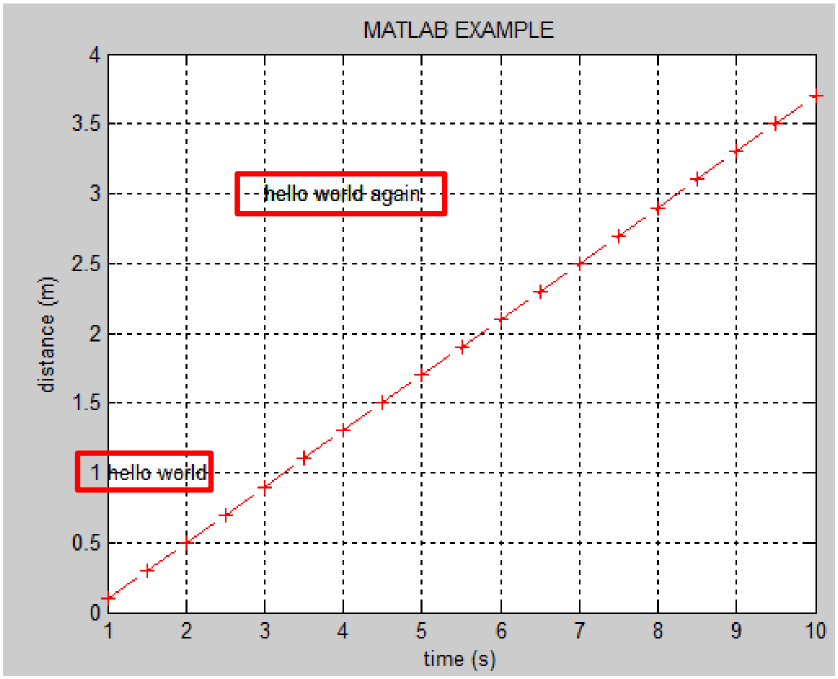
Table 3: Line Color modifiers

| Line Color | Modifier |
|---|---|
| Red | r |
| Green | g |
| Blue | b |
| While | w |
| Yellow | y |
| Magenta | m |
| Cyan | c |
| Black | black |

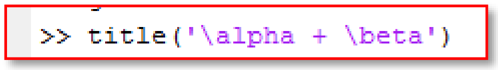
To insert text on the plot, use the *text(X,Y,'string'),* where (X,Y) refers to the location on the current axes in the same units of the current plot and 'string' is the text to be entered, for example:



```
>> text(1,1,'hello world')
>> text(3,3,'hello world again')
```



You also can use Greek characters and special symbols (Tables 4 and 5) on your plot. Let's say that title of the plot is "α+β." Here how would you display that:



```
>> title('\alpha + \beta')
```

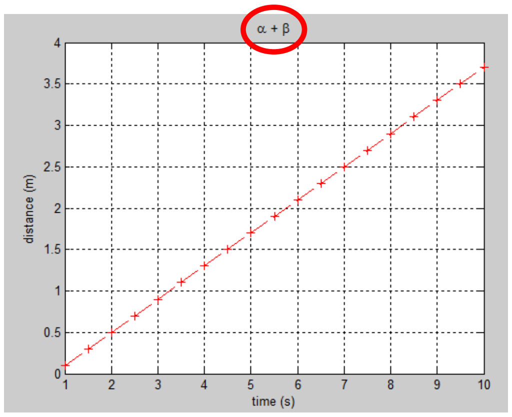


Table 4: Greek Characters

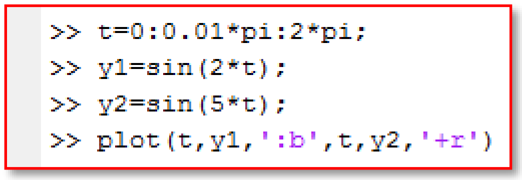
| Greek Letter* | Command |
|---|---|
| α | \alpha |
| β | \beta |
| γ | \gamma |
| δ | \delta |
| ω | \omega |
| σ | \sigma |
| φ | \phi |
| ψ | \psi |
| θ | \theta |
| ζ | \zeta |

* if you capitalize the first letter in the command, MATLAB will display capital Greek letter, for example \alpha will show α while \Alpha will show A.

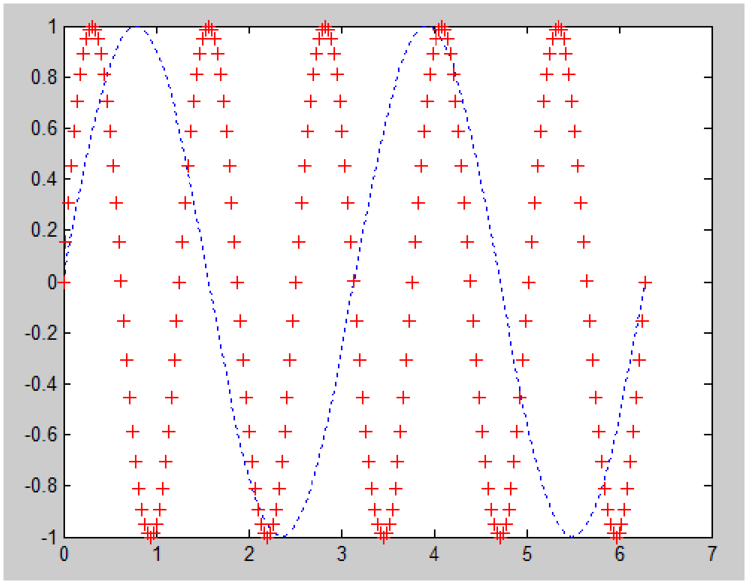
Table 5: Special Symbols

| Symbol | Command |
|---|---|
| ∞ | \infinity |
| ← | \leftarrow |
| → | \rightarrow |
| ↑ | \uparrow |
| ↓ | \downarrow |
| ± | \pm |
| ° | \circ |

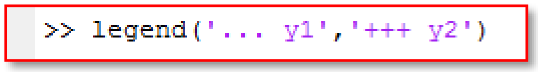
Next, let's say you would like to plot multiple series on the same plot, where the first curve uses **dotted line** and second curve uses **plus** for the points. To accomplish this, uses plot(X1,Y1,'line_specs',X2,Y2,'line_specs'), such as:



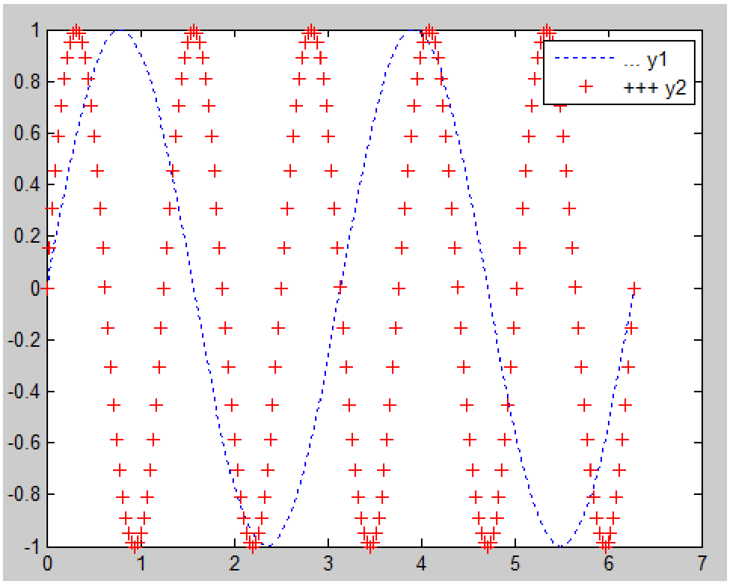
```
>> t=0:0.01*pi:2*pi;
>> y1=sin(2*t);
>> y2=sin(5*t);
>> plot(t,y1,':b',t,y2,'+r')
```



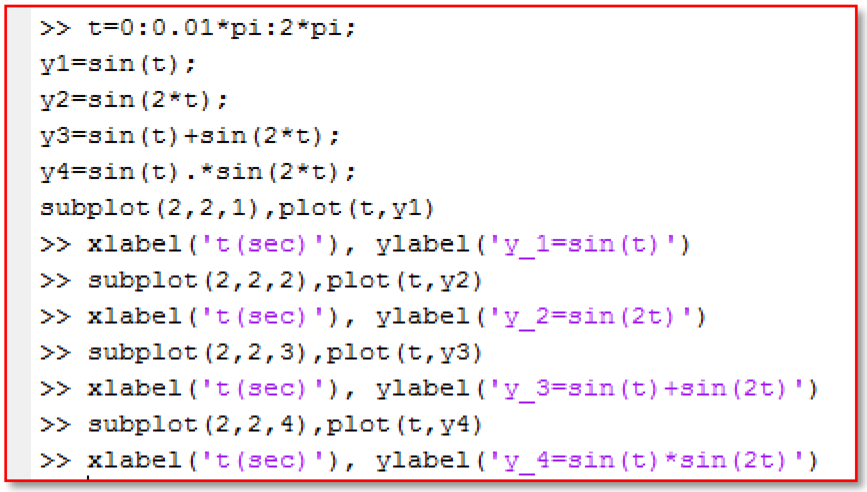
Now you have multiple curves on the same plot, you can display the legends using legend command, such as:



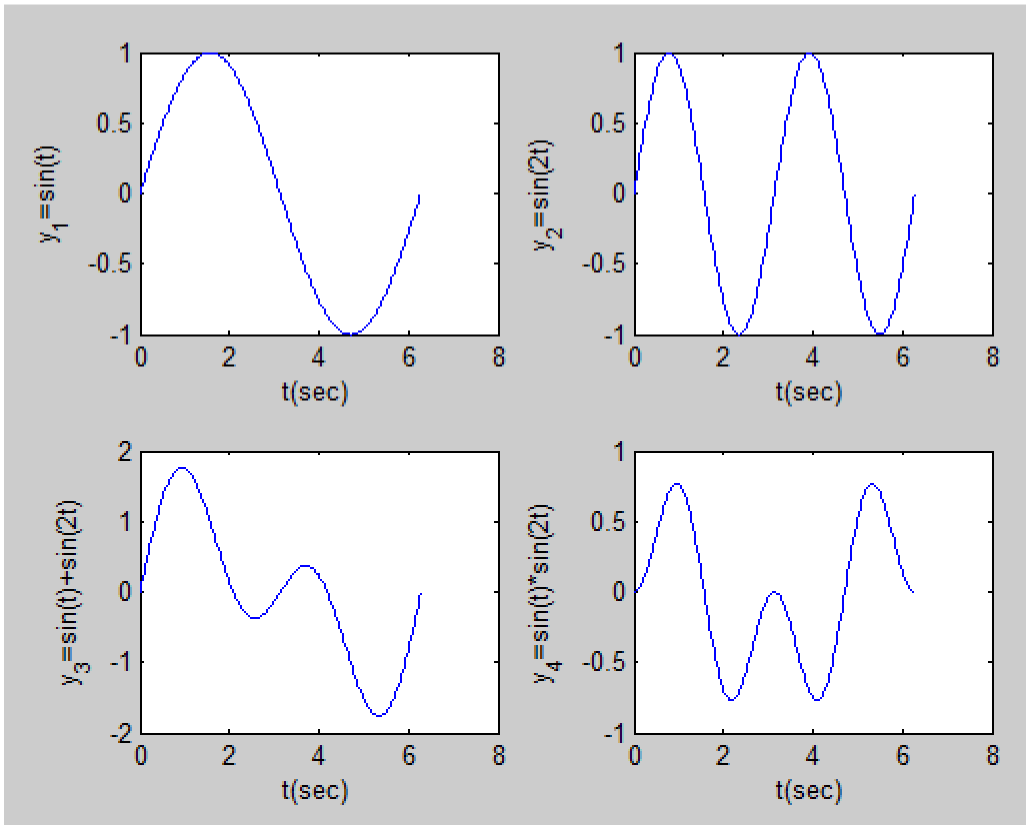
```
>> legend('... y1','+++ y2')
```



Use of the subplot command: multiple curves on one screen may be split into multiple windows with the use of ***subplot(m,n,p)*** command, where the graph display is subdivided into *mxn* smaller subwindows numbered from the ***left to right and top to bottom***. The integer *p* specifies the subwindow number. Consider the following example:



```
>> t=0:0.01*pi:2*pi;
y1=sin(t);
y2=sin(2*t);
y3=sin(t)+sin(2*t);
y4=sin(t).*sin(2*t);
subplot(2,2,1),plot(t,y1)
>> xlabel('t(sec)'), ylabel('y_1=sin(t)')
>> subplot(2,2,2),plot(t,y2)
>> xlabel('t(sec)'), ylabel('y_2=sin(2t)')
>> subplot(2,2,3),plot(t,y3)
>> xlabel('t(sec)'), ylabel('y_3=sin(t)+sin(2t)')
>> subplot(2,2,4),plot(t,y4)
>> xlabel('t(sec)'), ylabel('y_4=sin(t)*sin(2t)')
```



***Trickery:***

If you have too many figures open and wish to close all of them use ***close all*** command

If you have a figure open and wish to plot on the same figure use ***hold on*** command