Introduction to Computing for Engineers 050IU

Plots and Graphs

Dr. Nguyen Ngoc Truong Minh

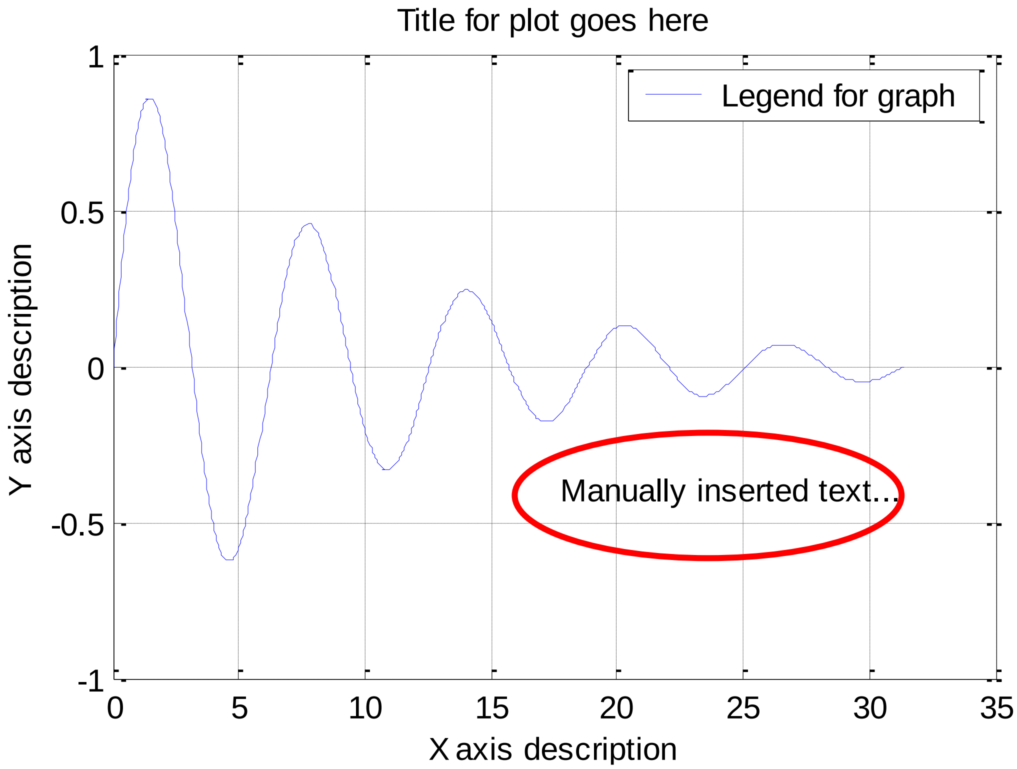
Basic 2D Plotting

- **The simplest kind of plot is a cartesian plot of (x,y) pairs defined by symbols or connected with lines**

```
>> x=0:0.05:10*pi;
>> y=exp(-.1.*x).*sin(x);
```

NOTE #1:
Reversing the x,y order (y,x) simply rotates the plot 90 degrees!

NOTE #2:
`line(x,y)` is similar to `plot(x,y)` but does not have additional options

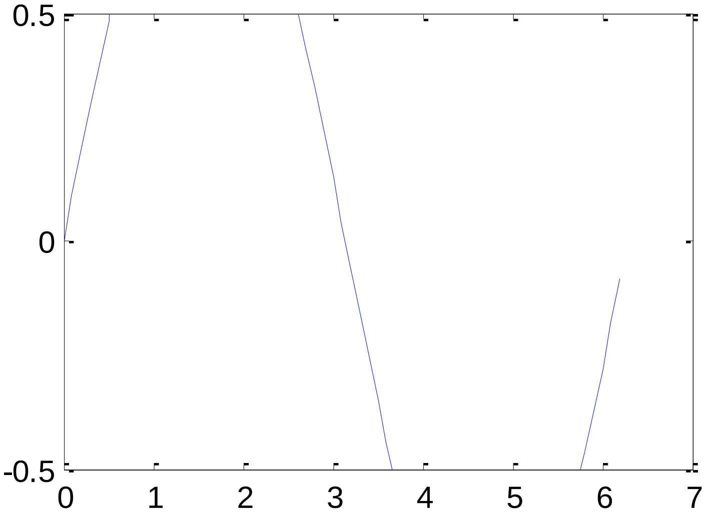


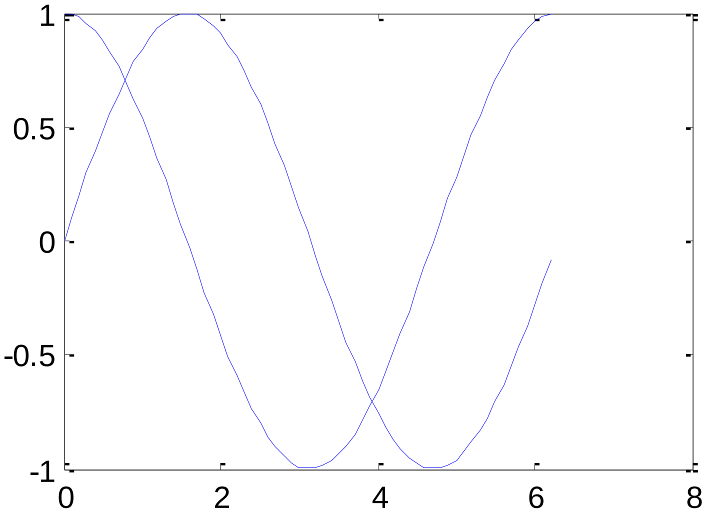
Supporting Commands

- **Several functions let you control the plot appearance**
  - ***axis():*** ***determines the axis scaling (see help for options)***
    - ***hold on/off:*** ***controls whether the plot is erased before another plot is drawn (toggles if no argument given)***

```
>> x=0:0.1:2*pi;
>> plot(x,sin(x));
>> axis
ans =
     0      7     -1      1
>> axis([0 7 -.5 .5]);
```

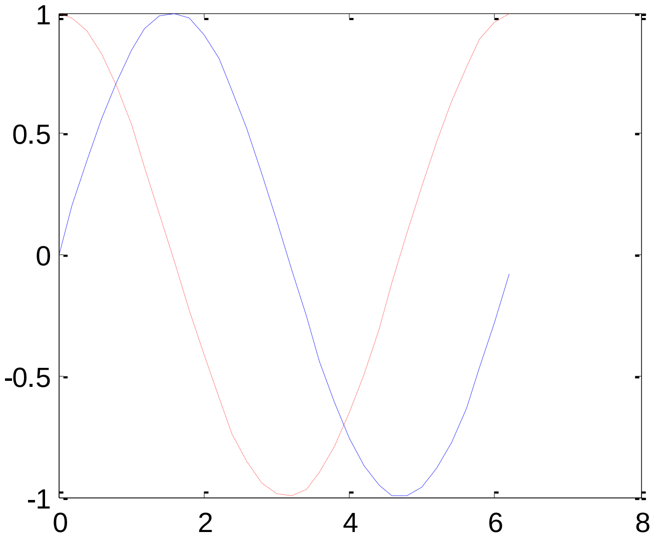
```
>> x=0:0.1:2*pi;
>> plot(x,sin(x));
>> hold on;
>> plot(x,cos(x));
```

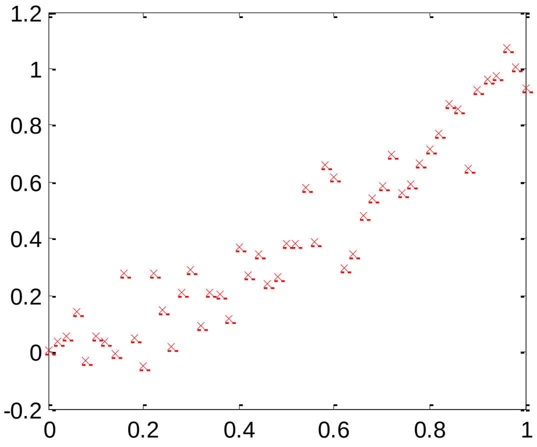


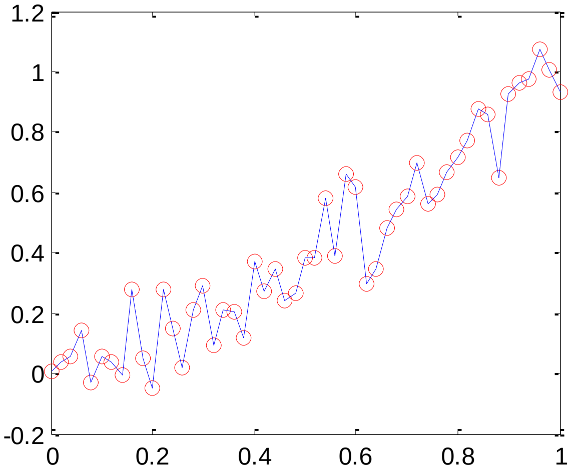


Using Lines or Markers or Both…

- **Plots <u>must</u> follow the following logic:**
  - *Lines*: ***whenever plotting analytical functions like sin(x) where you can compute y for any value of x***
  - *Markers*: ***whenever plotting discrete experimental data or whenever the data are known only discretely***
  - *Both*: ***connecting markers with straight lines is appropriate when you want to show a sequence***



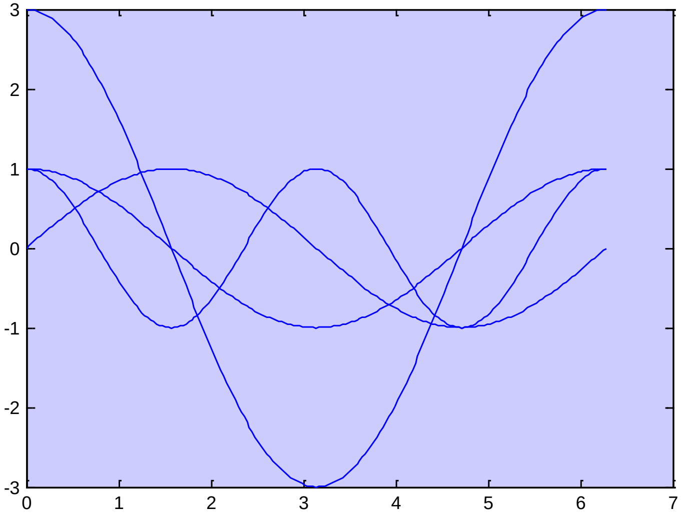




Plotting Multiple Curves

- **Problem: How can you compare several curves?**
- **Let's start with the following:**

```
>> X = 0.0:pi/100:2*pi;
>> Y1 = cos(X);
>> plot(X, Y1); hold on;
>> Y2 = 3*cos(X);
>> plot(X, Y2); hold on;
>> Y3 = cos(2*X);
>> plot(X, Y3); hold on;
>> Y4 = sin(X);
>> plot(X, Y4);
```



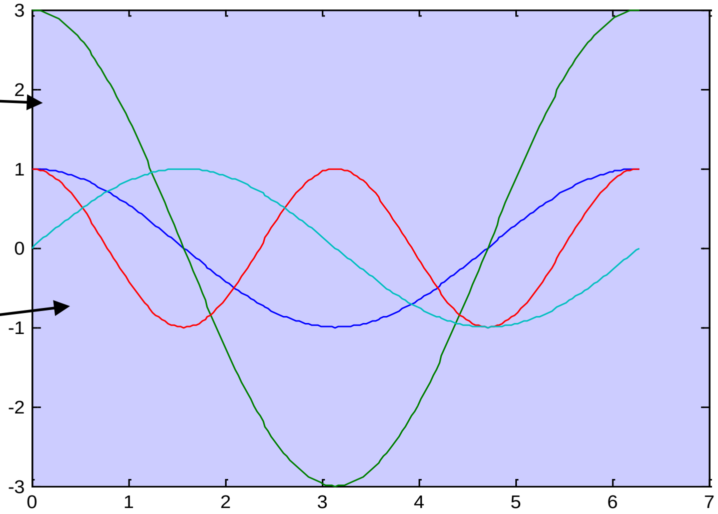
Plotting Multiple Curves *(cont’d)*

- **Or we could do:**

```
>> plot(X,Y1,X,Y2,X,Y3,X,Y4)
```

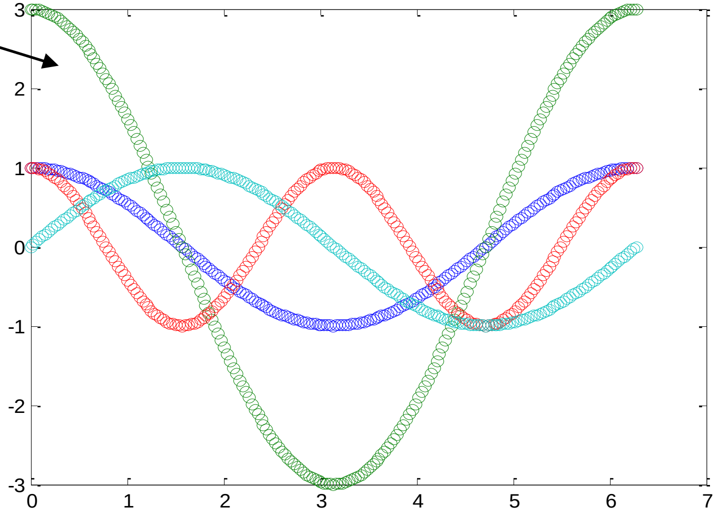
- **Or we could do this:**

```
>> Z = [Y1;Y2;Y3;Y4];
>> plot(X,Z)
```



- **What if we did this?**

```
>> plot(X, Z, 'o')
```



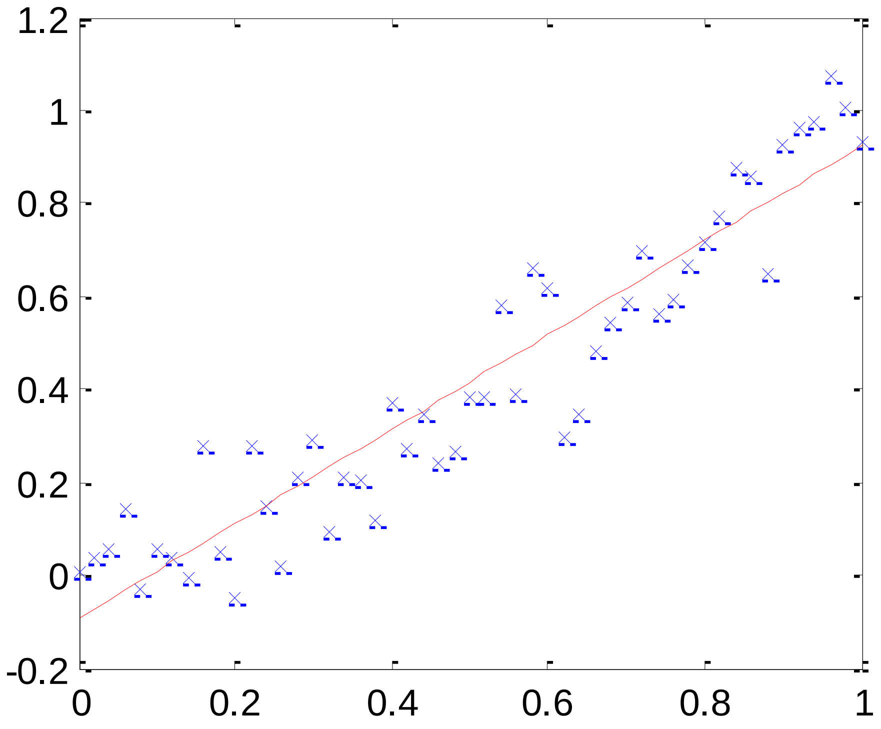
- **Do a “`help plot`” for more markers.**
- **How could we see the data points more distinctly?**

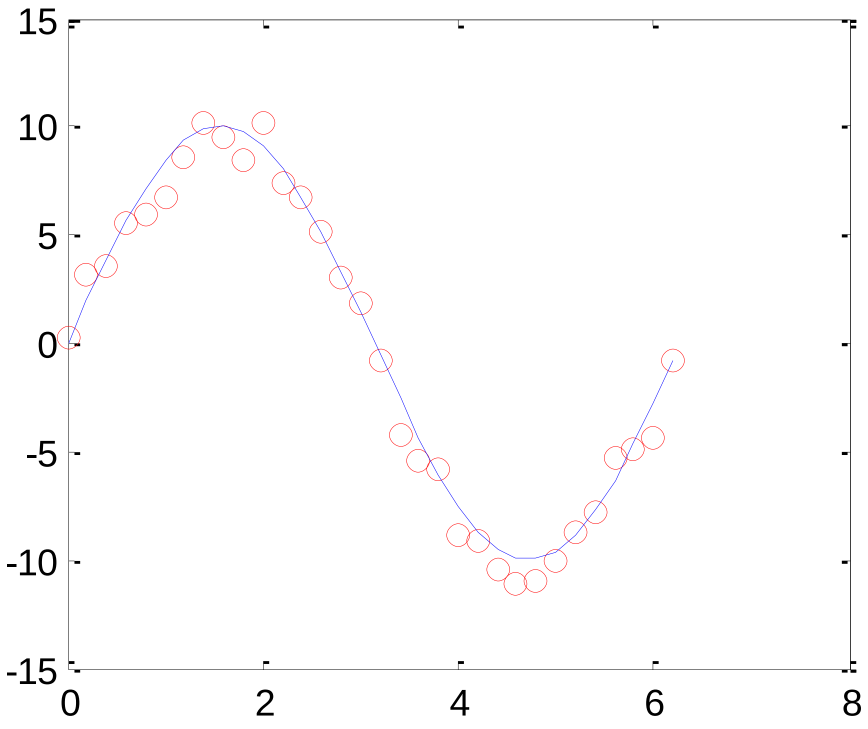
Using Both Markers & Lines

- **Use lines to show analytical fit through discrete data**

```
>> x=0:.02:1;
>> y=x.^1.5;
>> plot(x,y); hold on;
>> yr=randn(size(x));
>> yy=y+0.1.*yr;
>> plot(x,yy,’xb’);
```

```
>> x=0:0.2:2.*pi;
>> y=10*sin(x);
>> plot(x,y); hold on;
>> yr=y+rand(size(x));
>> plot(x,yr,’or’);
```



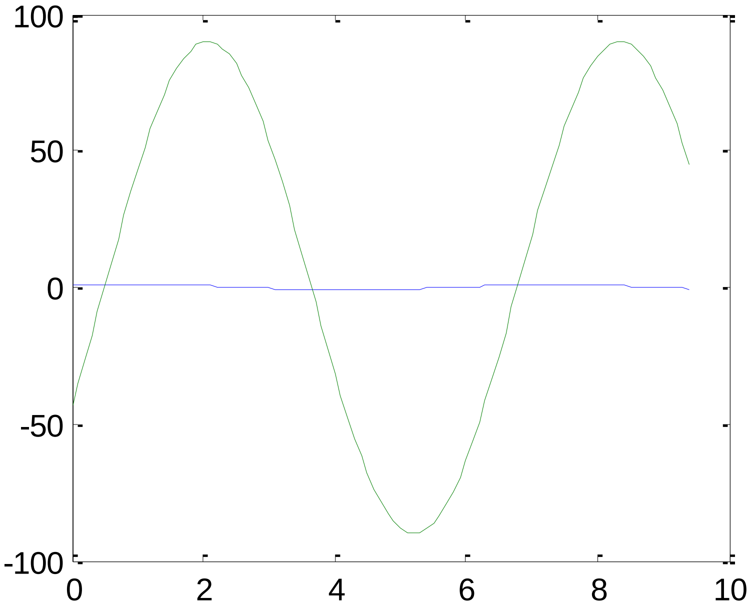


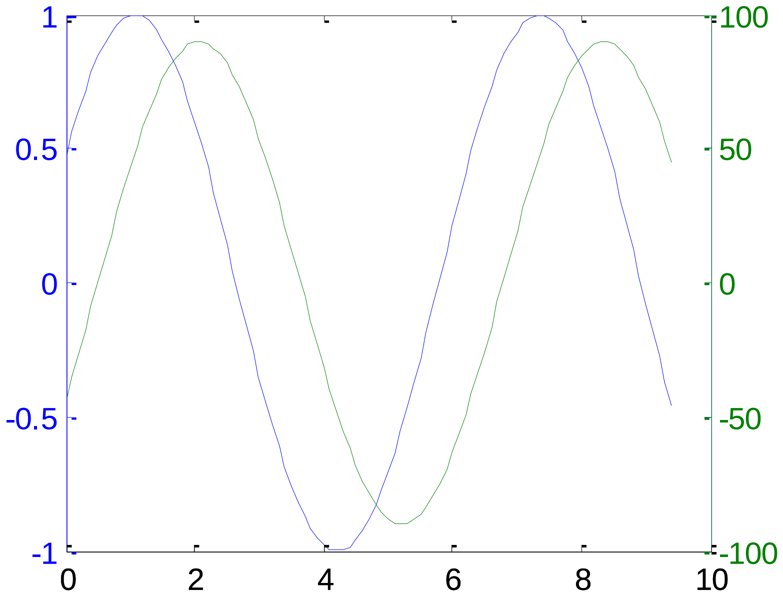
Using 2 Y-axis Scales

- **Sometimes it is useful to plot two curves with widely different y-axis scales**

```
>> x=0:0.1:3.*pi;
>> y1=sin(x+0.5);
>> y2=90.*sin(x-0.5);
>> plot(x,y1,x,y2);
```

```
>> x=0:0.1:3.*pi;
>> y1=sin(x+0.5);
>> y2=90.*sin(x-0.5);
>> plotyy(x,y1,x,y2);
```





NOTE: it is complicated to label the 2nd axis…

Basic Plot Commands

- `axis([xmin, xmax, ymin, ymax])` – sets axis limit values (note use of [ ])
- `axis off` – turns off display of axes (plot unchanged)
- `axis on` – turns on display of axes
- `grid on/off` – turns on/off display of a grid
- `text(x,y,‘string’)` - places horizontal text starting at (x,y)
- `gtext(‘string’)` – places horizontal text starting wherever user clicks with mouse
- `figure` – Create figure window

Example of Log Plots

- **Using a log scale can reveal large dynamic ranges**

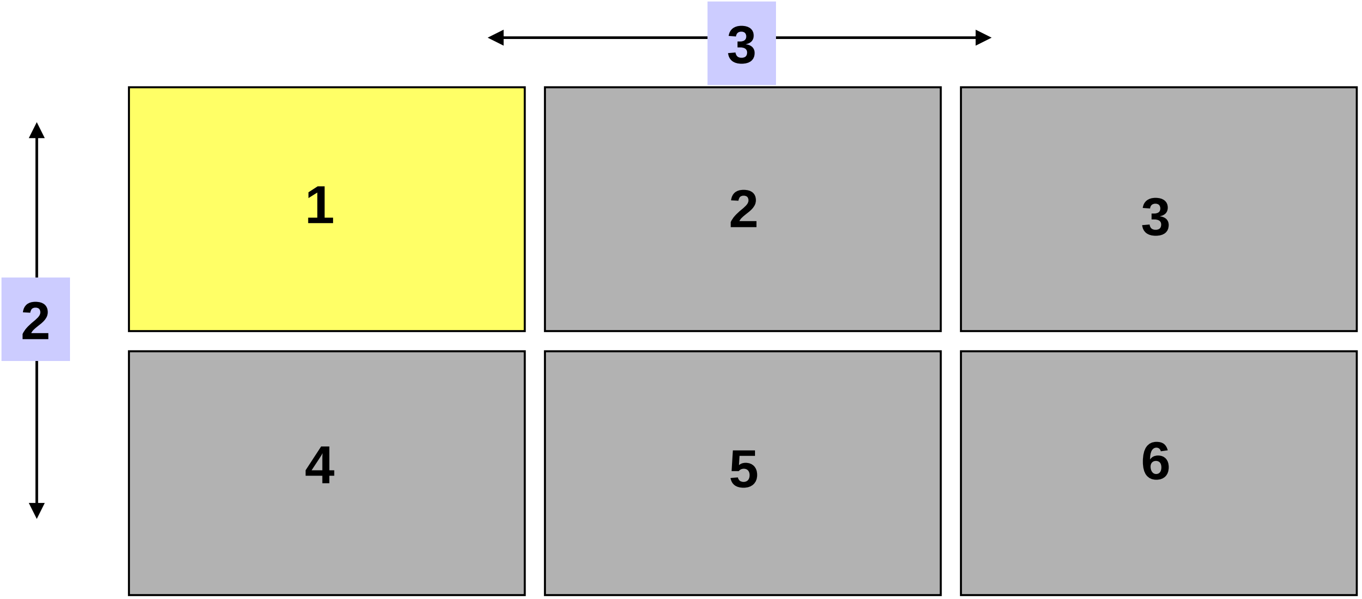
```
>> x=linspace(.1,10,1000);
>> damp=0.05;
>> y=1./sqrt((1-x.^2).^2 + (2.*damp.*x).^2);
```

Describes the behavior of vibrating systems

$$y = \frac{1}{\left[(1-x^2)^2 + (2\zeta x)^2\right]^{1/2}}$$

Subplot Command

- There are times when it is better to create several smaller plots arranged in grid; `subplot(m,n,k)` does this…
  - *m=rows, n=columns in the grid*
  - *k=current focus (numbered row-wise)*
- Let’s define a 2x3 subplot grid for: `subplot(2,3,1)` with the focus on the first plot.



3

2

1 2 3

4 5 6

On Your Own:

- **Putting it all together…**

```
X=0:0.5:50;
Y=5*X.^2;
subplot(2,2,1), plot(X,Y), title(‘Polynomial – Linear/Linear’), ...
   ylabel(‘y’), grid
subplot(2,2,2), semilogx(X,Y), title(‘Polynomial – Log/Linear’), ...
   ylabel(‘y’), grid
subplot(2,2,3), semilogy(X,Y), title(‘Polynomial – Linear/Log’), ...
   ylabel(‘y’), grid
subplot(2,2,4), loglog(X,Y), title(‘Polynomial – Log/Log’), ...
   ylabel(‘y’), grid
```

- **What does `grid` do?**
- **What’s the quickest way to execute this code?**

Specialized 2D Plots

- There are a number of other specialized 2D plots
  - `area(x,y)` : *builds a stacked area plot*
  - `pie()` : *creates a pie chart (with options)*
  - `bar(x,y)` : *creates a vertical bar chart (with many options)*
  - `stairs(x,y)` : *similar to bar() but shows only outline*
  - `errorbar(x,y,e)` : *plots x vs y with error bars defined by e*
  - `scatter(x,y)` : *creates a scatter plot with options for markers*
  - `semilogx(x,y)` : *plots x vs y with x using a log scaling*
  - `semilogy(x,y)` : *plots x vs y with y using a log scaling*
  - `loglog(x,y)` : *plots x vs y using log scale for both axes*

  - *And many others…* (explore these yourself; you may find a good use in a later course)

Questions?

Quiz 2

- Create function find largest elements in array **A** and **I** is the index of A(:) containing the largest element.

  Name of function must be **yourfirstname_max.**
- Write script generate random integer array that length is 50.

  Find the maximum value and index by **your own function**.

  And test with the **max** function of matlab.

The output on comment windows should be:

>> yourname_ID
The maximum value by my function is :….
The index of maximum value by my function is :…..
The maximum value by MATLAB function is: ….
The index of maximum value by MATLAB function is :…..