SECTION 3: TWO-DIMENSIONAL PLOTTING

ENGR 112 – Introduction to Engineering Computing

Data Visualization

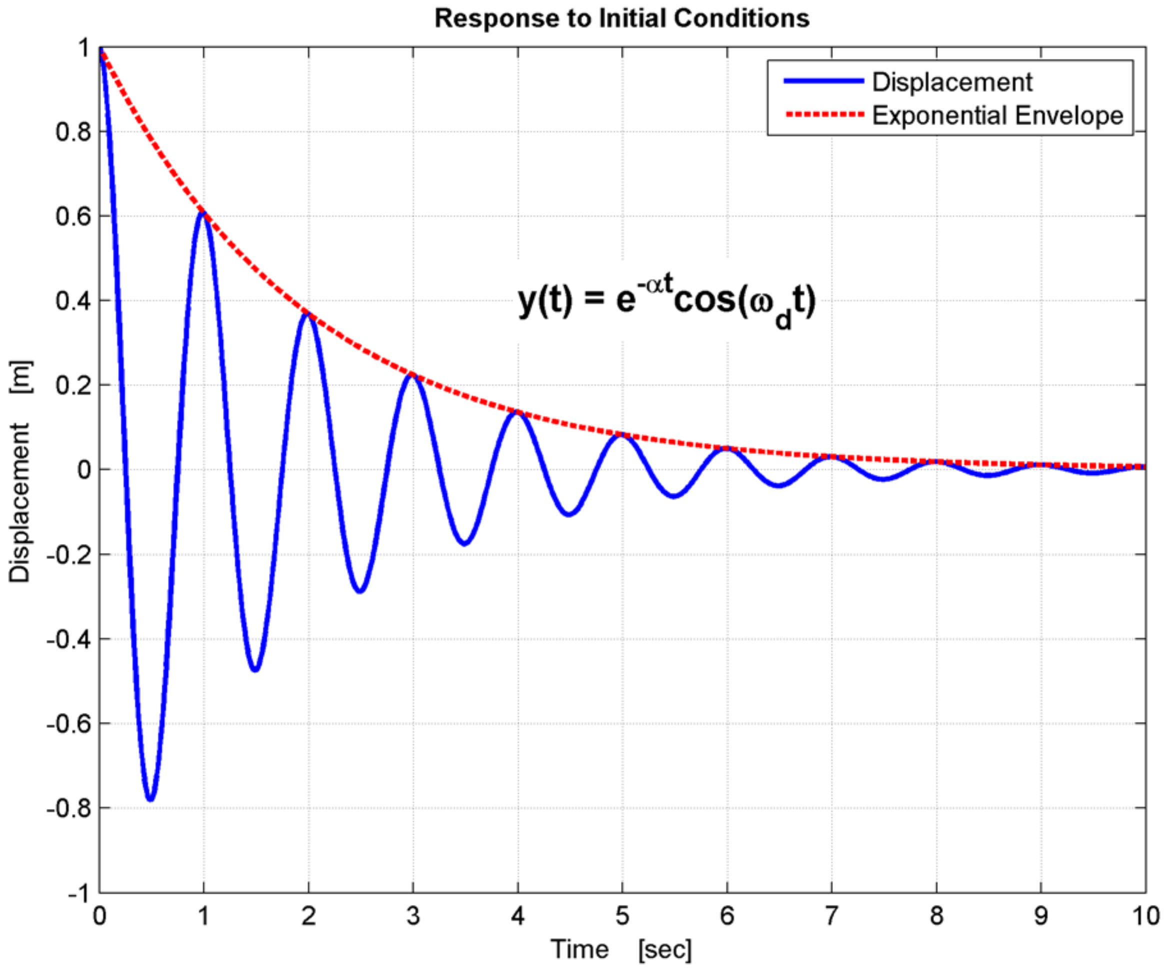
- Like it or not, the ability to ***communicate effectively*** is an important aspect of being a successful engineer
  - Coworkers, managers, marketing, customers, etc.

- As engineers, effective communication often means ***effective communication of data***
  - Technical writing
  - ***Graphical presentation of data***: plots, graphs, charts, etc.

- MATLAB has a variety of data-visualization tools – these fall into two main categories:
  - **2-D plotting**
  - **3-D plotting**

2-D Plots

Basic 2-D Plotting – `plot(…)`



Basic 2-D Plotting – `plot(…)`

- Syntax:

```
plot(x,y,’LineSpec’,’PropName’,PropValue)
```

- `x` and `y` are ***equal-length vectors*** of data
  - `x` data is the abscissa – plotted on the horizontal axis
  - `y` data is the ordinate – plotted on the vertical axis
- `LineSpec` defines the type and color of the line used to plot and the shape of the marker placed at each data point – (optional)
- `PropName` may be any number of properties, such as the width of the line, and is followed by its value – (optional)
  - Multiple property/value pairs may be specified in succession

LineSpec – Line Style

```
plot(x,y,’LineSpec’,’PropName’,PropValue)
```

- Three components – ***line style***, ***marker***, ***color***
  - Specify some or all
- ***Line Style*** specifiers:

Specifier	Line Style
`'-'`	Solid
`'--'`	Dashed
`':'`	Dotted
`'-.'`	Dash-dot

- Default is a solid line

LineSpec – Marker

- ***Marker*** specifiers:

Specifier	Marker
`'+'`	Plus sign
`'o'`	Circle
`'*'`	Asterisk
`'.'`	Point
`'x'`	Cross
`'s'`	Square
`'d'`	Diamond

Specifier	Marker
`'^'`	Upward-pointing triangle
`'v'`	Downward-pointing triangle
`'>'`	Right-pointing triangle
`'<'`	Left-pointing triangle
`'p'`	pentagram
`'h'`	hexagram

- Default is no marker
  - Markers are placed at every data point – can get crowded for closely spaced data

LineSpec – Line/Marker Color

- ***Color*** specifiers:

Specifier	Color
`'r'`	Red
`'g'`	Green
`'b'`	Blue
`'c'`	Cyan

Specifier	Color
`'m'`	Magenta
`'y'`	Yellow
`'k'`	Black
`'w'`	White

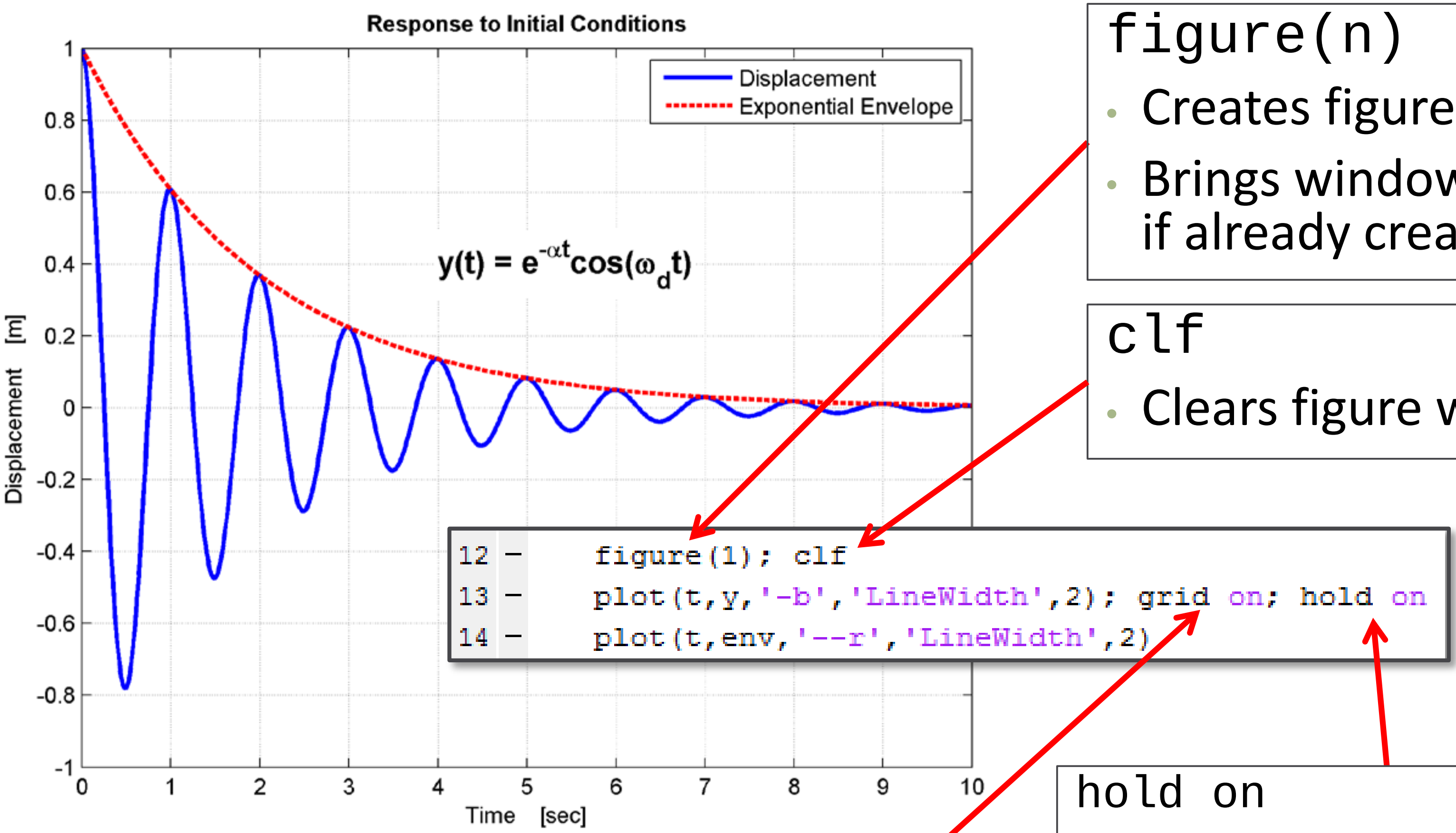
- Default color is blue
  - If multiple x,y pairs are specified in a single plot command, line/marker colors will cycle through automatically (white is skipped for white background)

Line Properties

```
plot(x,y,’LineSpec’,’PropName’,PropValue)
```

- Property name and value specified in pairs
  - ***LineWidth*** — numeric value (points) – 2 is good for most plots
  - ***MarkerEdgeColor*** —color of the marker or edge color for filled markers
  - ***MarkerFaceColor*** —face color of filled markers
  - ***MarkerSize*** — numeric value (points)

Using `plot(…)`



```
    figure(1); clf
    plot(t,y,'-b','LineWidth',2); grid on; hold on
    plot(t,env,'--r','LineWidth',2)
```

`figure(n)`

- Creates figure window
- Brings window to front if already created

`clf`

- Clears figure window

`hold on`

- Superimpose multiple traces
- Plot command won’t erase existing traces

`grid on`

- Turns on grid lines

Plot Annotation

- ***Title***

```
title(‘string’,‘PropName’,‘PropValue’)
```

- ***Axis labels***

```
xlabel(‘string’…)
ylabel(‘string’…)
```

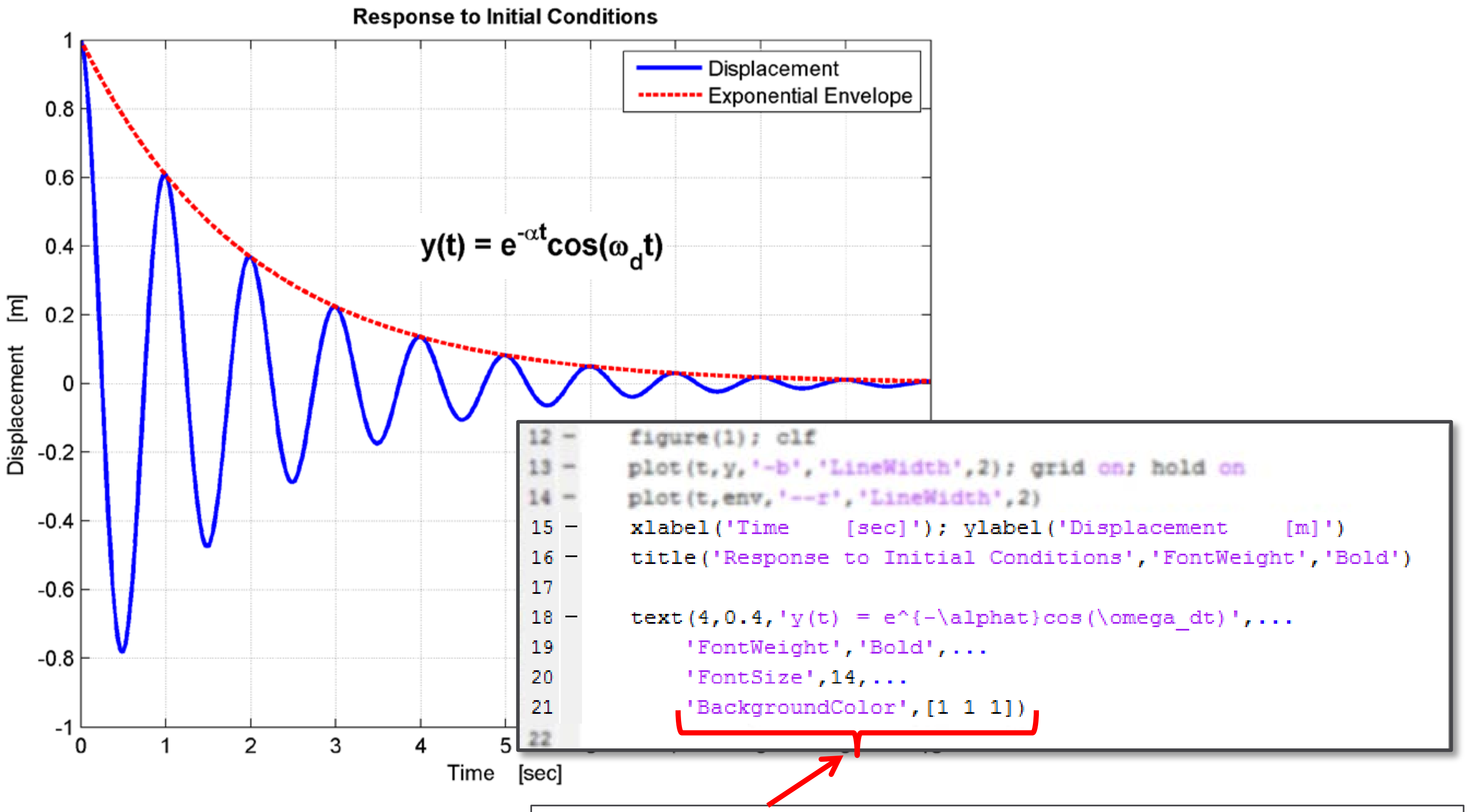
- ***Text***

```
text(x,y,‘string’…)
```

  - Text string printed at location (x,y) on the current figure axes

- ***Special characters***
  - *Most* MATLAB annotation functions can interpret TeX character sequences
  - E.g. `\beta`, `\mu`, `\it`, `\div`, etc.
  - Search help for ‘Text Properties’ for a table of TeX characters

Plot Annotation



```
figure(1); clf
plot(t,y,'-b','LineWidth',2); grid on; hold on
plot(t,env,'--r','LineWidth',2)
xlabel('Time     [sec]'); ylabel('Displacement     [m]')
title('Response to Initial Conditions','FontWeight','Bold')

text(4,0.4,'y(t) = e^{-\alphat}cos(\omega_dt)',...
    'FontWeight','Bold',...
    'FontSize',14,...
    'BackgroundColor',[1 1 1])
```

White background – good for printing over gridlines

Plot Legend

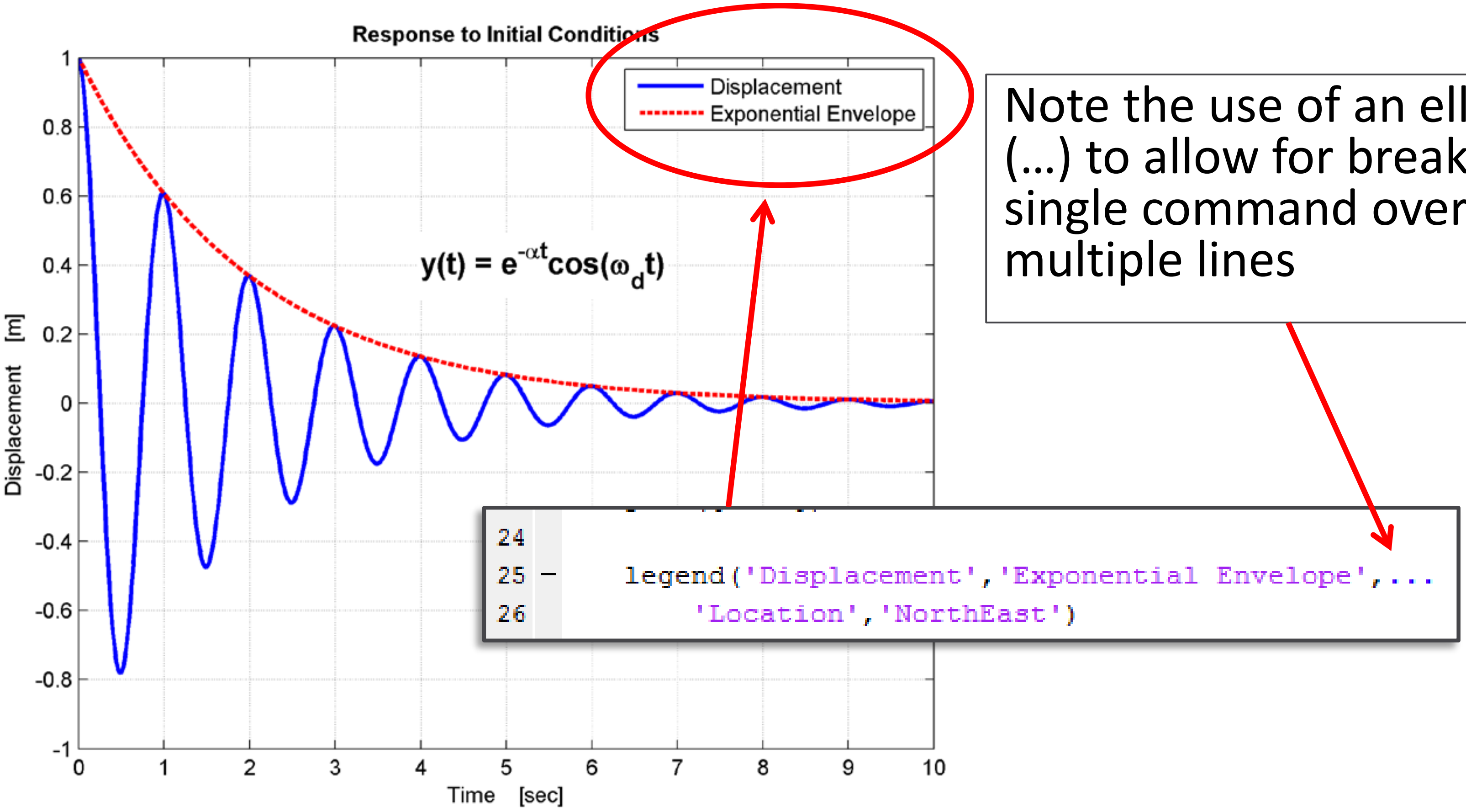
- Add a legend to a figure to identify multiple traces

```
legend(‘string_1’,‘string_2’,…,‘string_n’…
        ‘Location’,‘location’)
```

- Strings assigned to curves in the order they were plotted
- *location* is specified using cardinal points
  - E.g. ‘NorthEast’, ‘West’, etc.
  - MATLAB can also choose the location with the least interference with traces:

```
legend(……’Location’,‘Best’)
```

Plot Legend



```

    legend('Displacement','Exponential Envelope',...
        'Location','NorthEast')
```

Note the use of an ellipsis (…) to allow for breaking a single command over multiple lines

Axis Scaling

- Specify the range of all axes at once

```
axis([xmin,xmax,ymin,ymax])
```

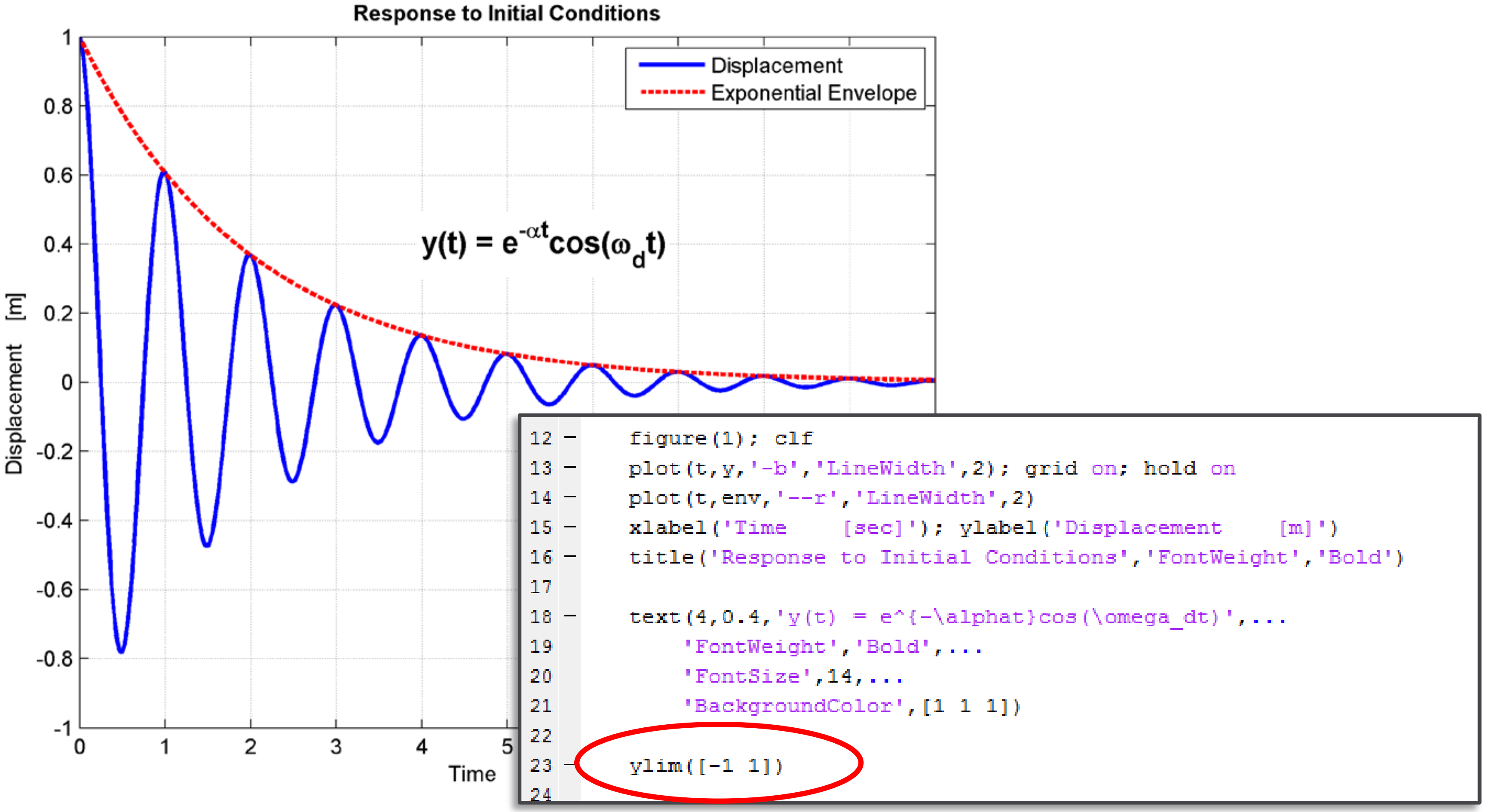
- Or, specify x and y axes individually

```
xlim([xmin,xmax])
ylim([ymin,ymax])
```

- Use `inf` to allow for ***autoscaling***, e.g.:

```
axis([-inf,1e4,0,40])
```

Axis Scaling



```
    figure(1); clf
    plot(t,y,'-b','LineWidth',2); grid on; hold on
    plot(t,env,'--r','LineWidth',2)
    xlabel('Time     [sec]'); ylabel('Displacement     [m]')
    title('Response to Initial Conditions','FontWeight','Bold')

    text(4,0.4,'y(t) = e^{-\alphat}cos(\omega_dt)',...
        'FontWeight','Bold',...
        'FontSize',14,...
        'BackgroundColor',[1 1 1])

    ylim([-1 1])

```

Subplots

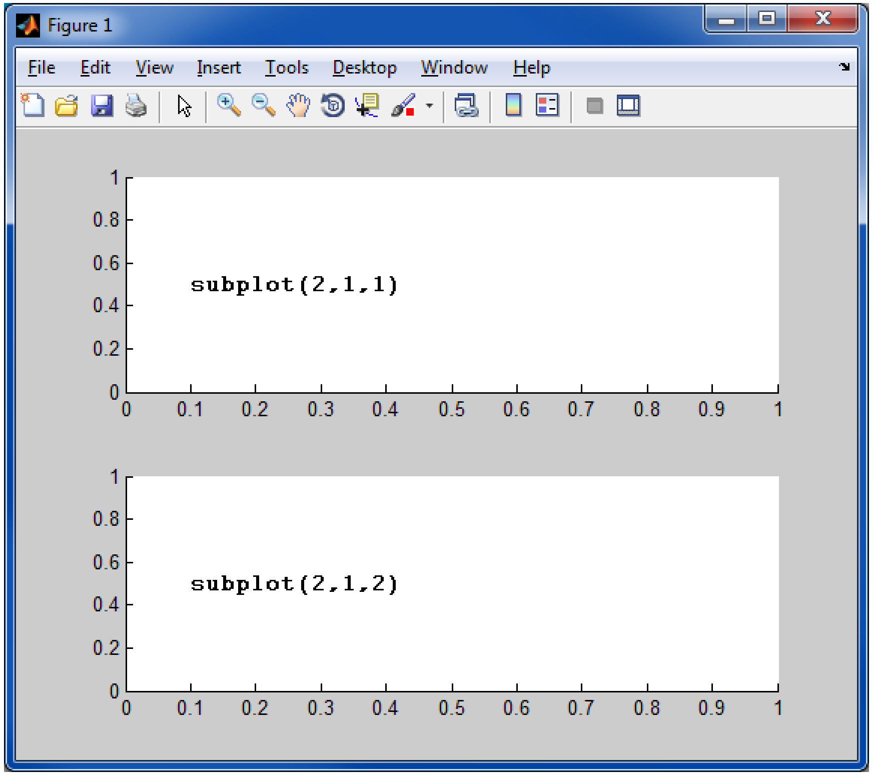
- Plot ***multiple sets of axes*** on a single figure

```
subplot(m,n,p)
```

- Figure window divided into `m` rows and `n` columns
- `p` is the current subplot index
  - Counted from left to right, top to bottom
- `subplot` command activates the `p`th subplot
  - All subsequent plotting/annotation commands issued to the active subplot
  - To plot to another subplot, issue `subplot` with a new value for `p`

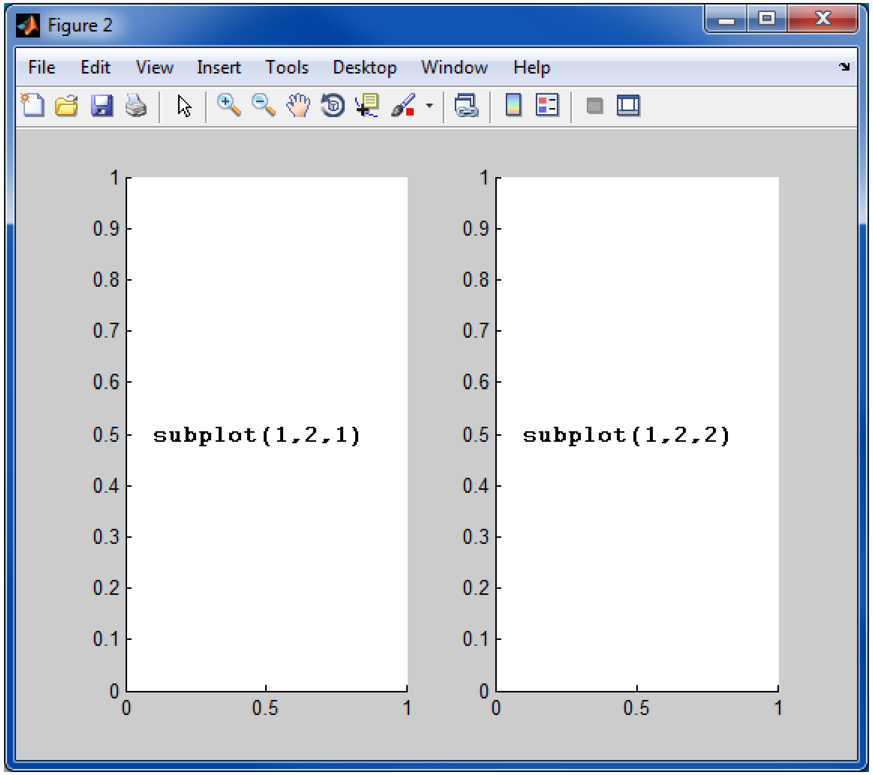
Subplot Numbering

- 2 rows, 1 column



```
subplot(2,1,1)
subplot(2,1,2)
```

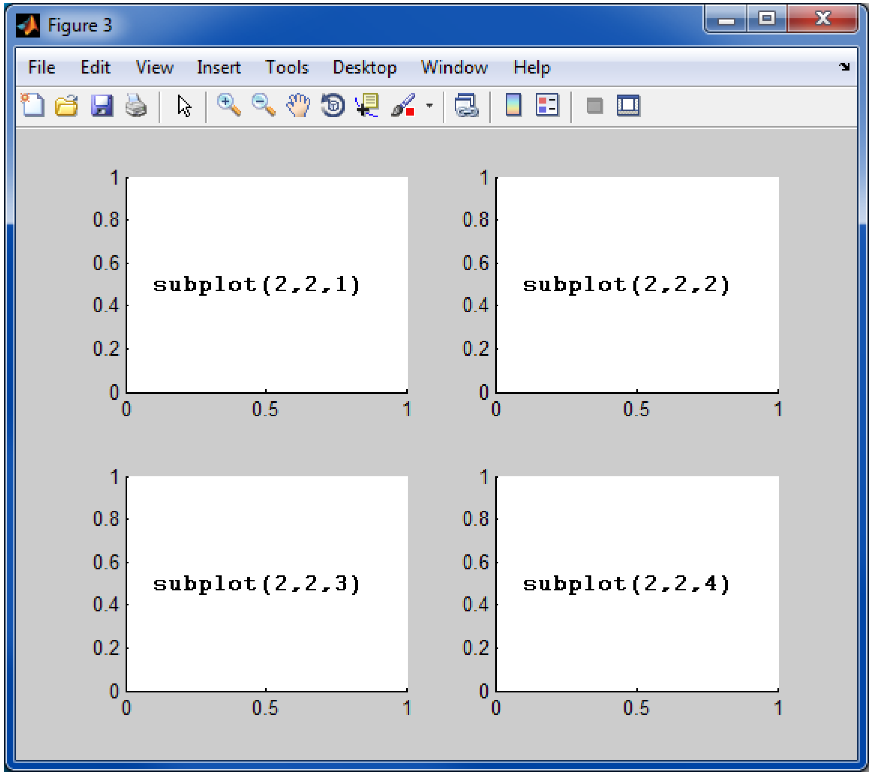
- 1 row, 2 columns

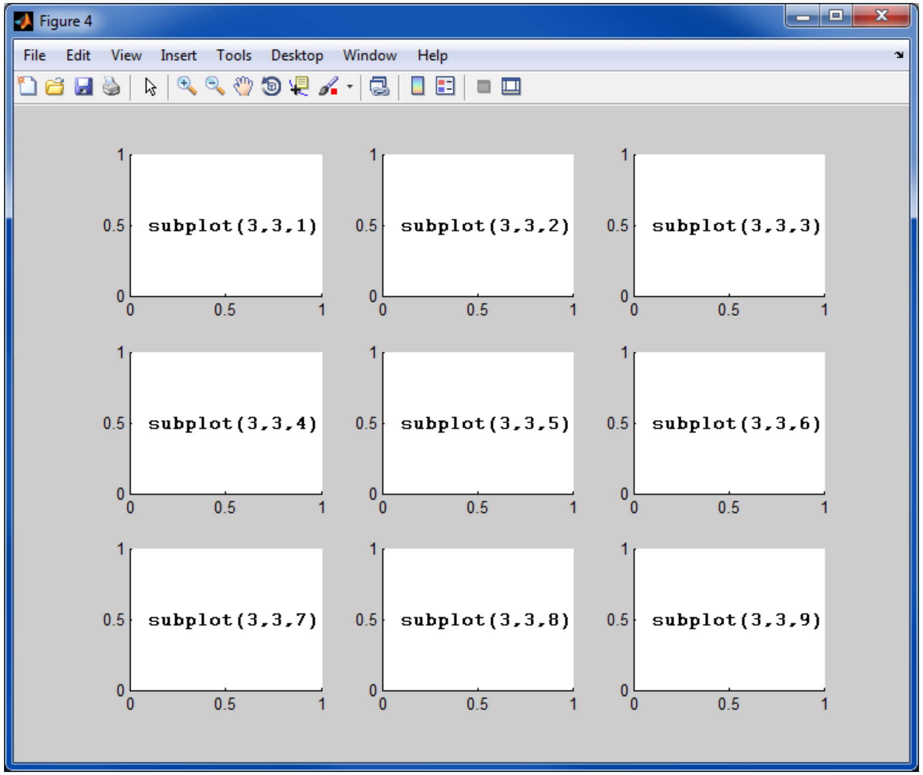


```
subplot(1,2,1)
subplot(1,2,2)
```

Subplot Numbering

- 2 rows, 2 columns
- 3 rows, 3 columns



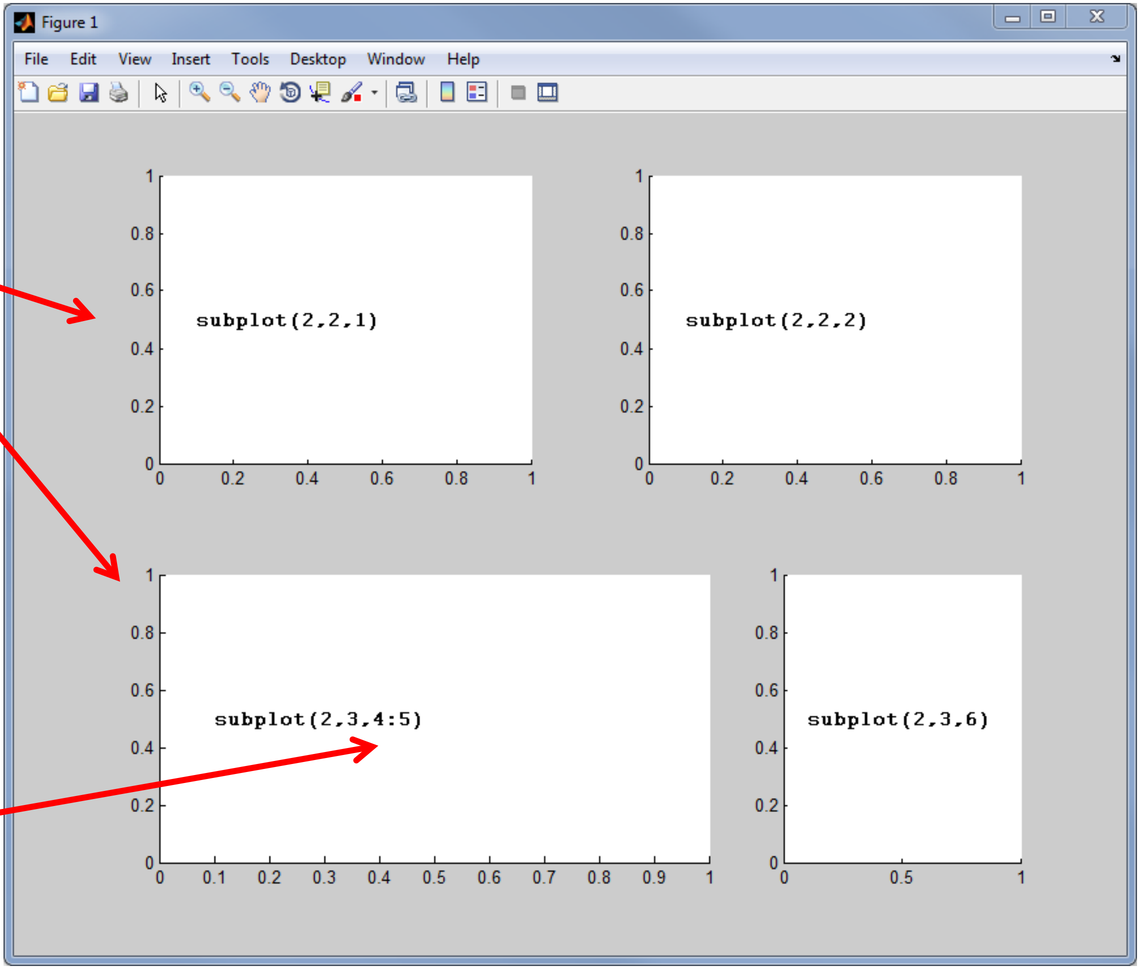


- Can have an arbitrary number of rows and columns

Subplot Numbering

`subplot(m,n,p)`

- m and n can vary within a figure window
- p can be specified as a range using the colon operator



subplot(2,2,1)

subplot(2,2,2)

subplot(2,3,4:5)

subplot(2,3,6)

Including Variable Values in Annotation

- Often, we want to include a variable value in a title or text annotation
- A couple of options:
  - `sprintf(…)` – use to create the string input for `title`, `text`, `xlabel`, etc. – does not recognize TeX character sequences – no special characters or Greek letters
  - `num2str(…)` – converts a variable to a string using the specified format
- In either case, if TeX formatting is required, use as ***part of a string array***

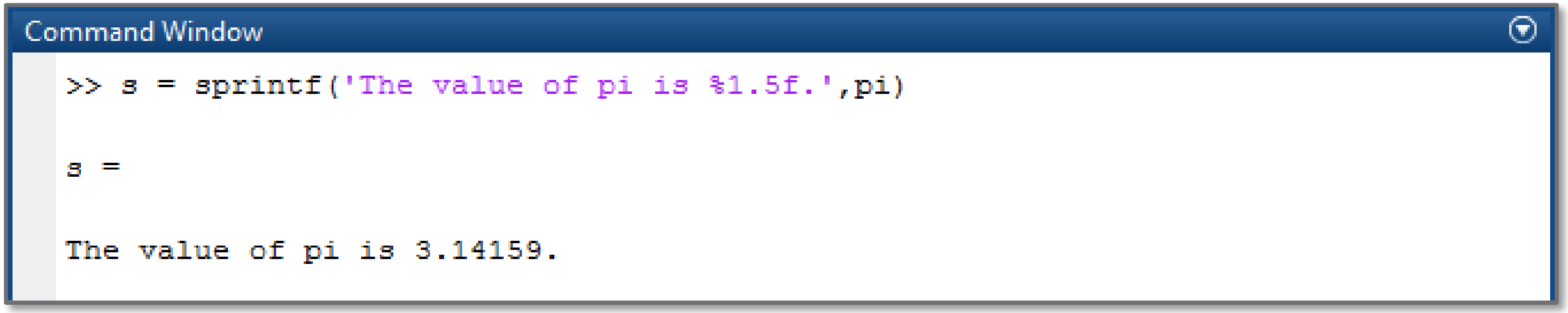
sprintf.m

- Write formatted data to an output string

```
str = sprintf(formatSpec,A1,A2,…,An)
```

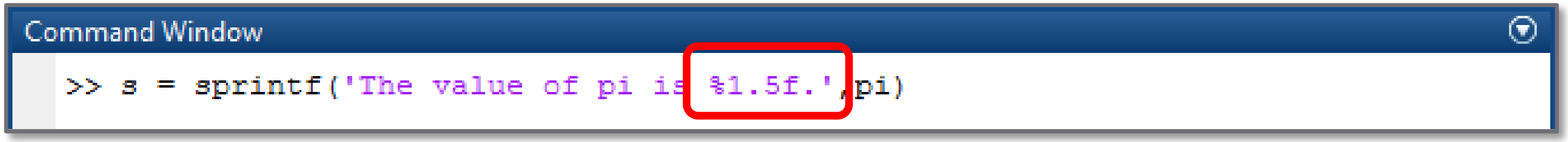
  - `formatSpec`: a *string* – may contain ***formatting sequences*** for insertion of variable values
  - `A1,A2,…,An`: variables whose values are to be inserted into the string – one for each formatting sequence in `formatSpec`
  - `str`: variable to which the created string is stored

- For example:



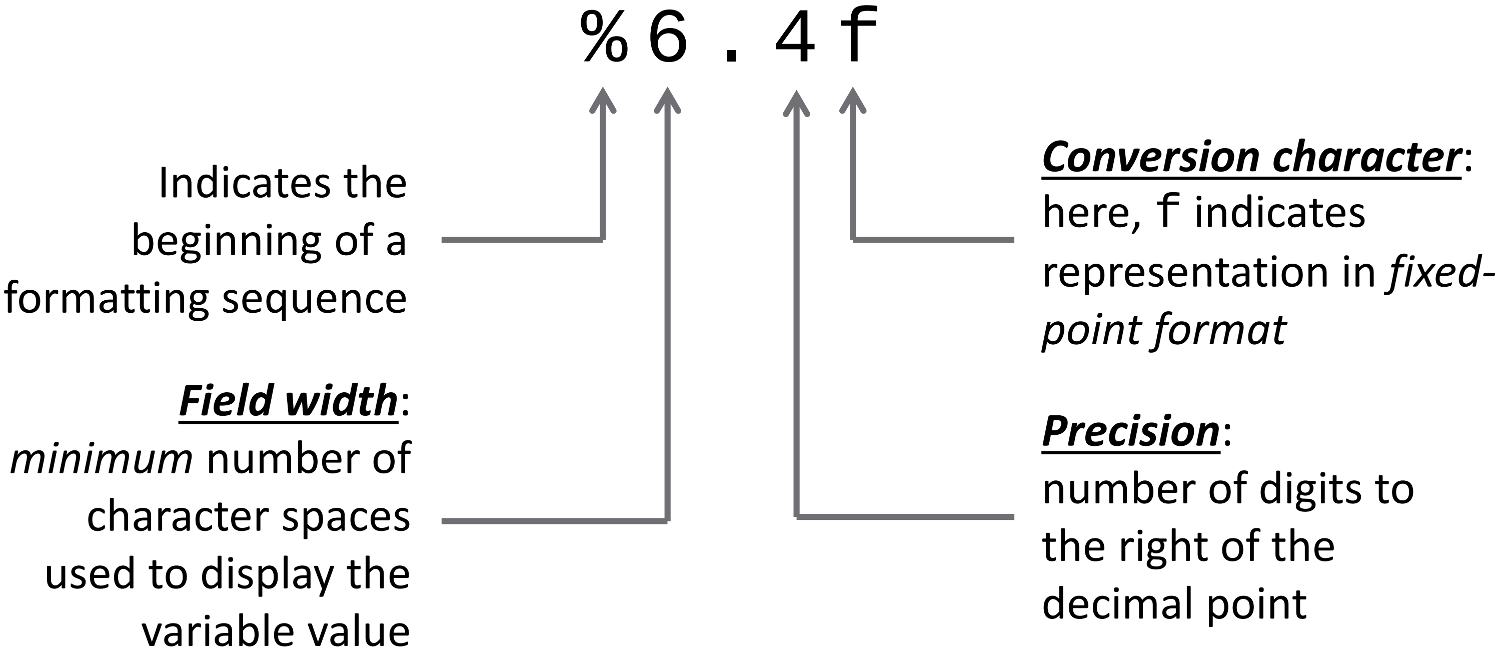
```
>> s = sprintf('The value of pi is %1.5f.',pi)

s =

The value of pi is 3.14159.
```

Formatting Sequences



```
>> s = sprintf('The value of pi is %1.5f.',pi)
```

- String may contain number ***formatting sequences***
  - Percent character (%) followed by conversion sequence



`%6.4f`

Indicates the beginning of a formatting sequence

Field width: *minimum* number of character spaces used to display the variable value

Conversion character: here, `f` indicates representation in *fixed-point format*

Precision: number of digits to the right of the decimal point

Conversion Characters

- □ Conversion characters specify how to format variable values within a string

Value Type	Conversion Character
Signed integer	%d
Unsigned integer	%u
Fixed-point notation	%f
Exponential notation (e.g., 1.6e-19)	%e
Exponential notation (e.g., 1.6e-19)	%E
Single character	%c
String	%s

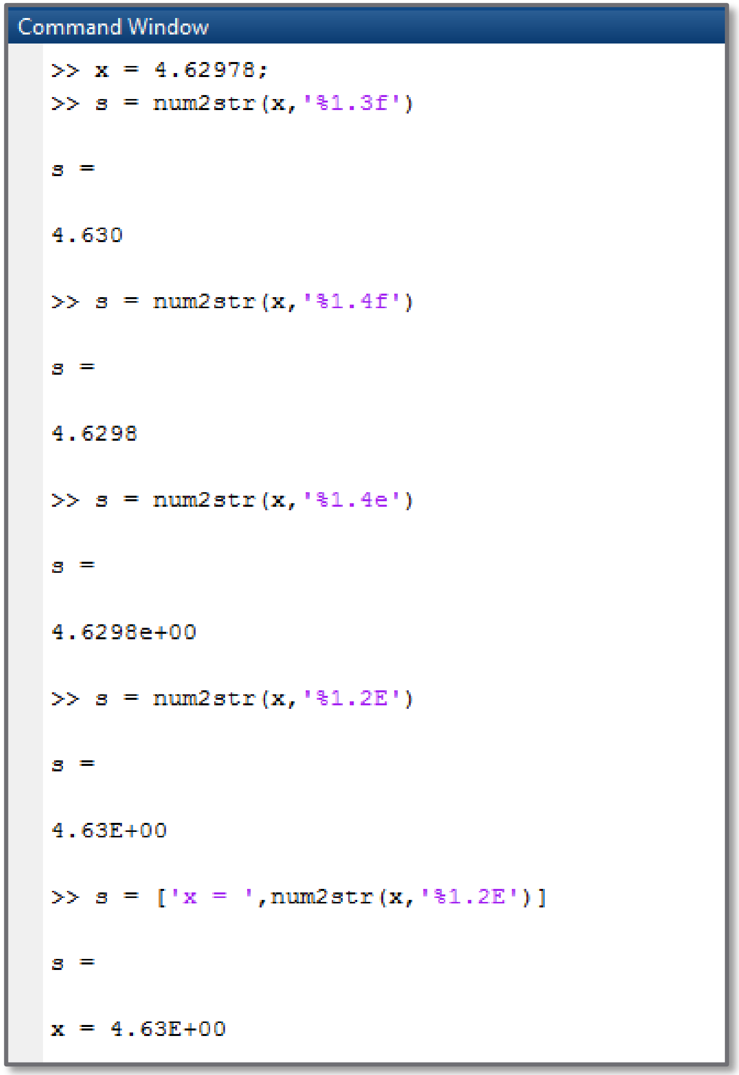
Annotations Using `num2str(…)`

```
num2str(A,’FormatSpec’)
```

- Converts the value of the variable A to a string according to `FormatSpec`
- `FormatSpec` specifies
  - ***Type of number*** (e.g. fixed-point, integer, etc.)
  - ***Field width*** and ***precision***
- The string created from the variable value can then be placed in a ***string array*** that is passed to the annotation function

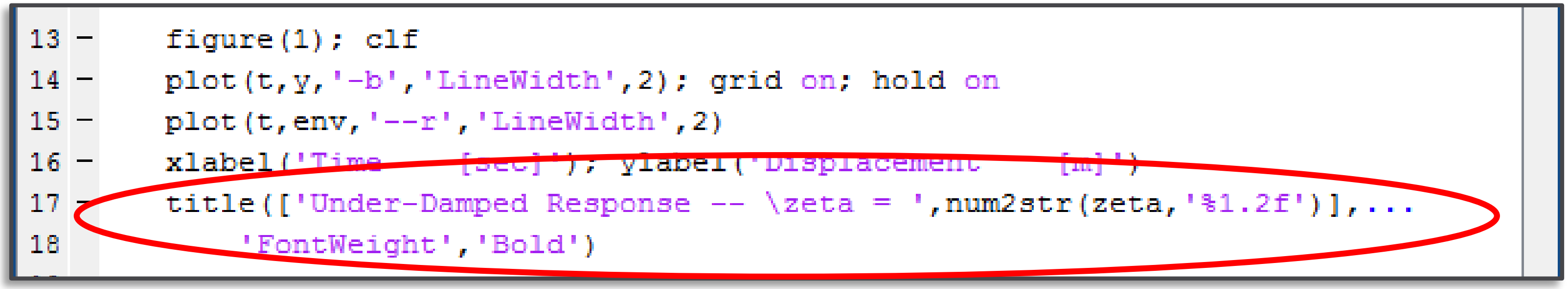
Creating Strings Using `num2str(…)`

- `x` is a ***double***
- `s` is a ***string*** representation of `x`
- `FormatSpec` controls how the numeric value is represented in the string
- `num2str` converts a single number to a string
- Insert numbers into strings by using `num2str` in ***string arrays***

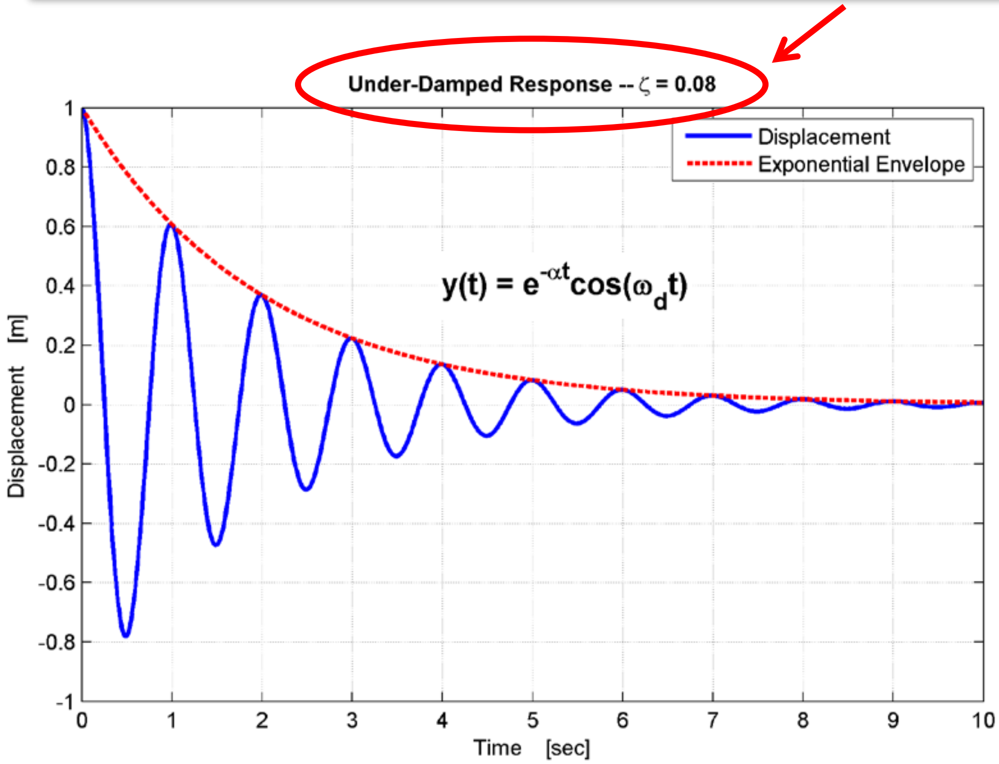


```
>> x = 4.62978;
>> s = num2str(x,'%1.3f')

s =

4.630

>> s = num2str(x,'%1.4f')

s =

4.6298

>> s = num2str(x,'%1.4e')

s =

4.6298e+00

>> s = num2str(x,'%1.2E')

s =

4.63E+00

>> s = ['x = ',num2str(x,'%1.2E')]

s =

x = 4.63E+00
```

Annotations Using `num2str(…)`



```
    figure(1); clf
    plot(t,y,'-b','LineWidth',2); grid on; hold on
    plot(t,env,'--r','LineWidth',2)
    xlabel('Time     [sec]'); ylabel('Displacement     [m]')
    title(['Under-Damped Response -- \zeta = ',num2str(zeta,'%1.2f')],...
        'FontWeight','Bold')
```



- □ `num2str(…)` used to generate one element – the value of ζ – of the ***string array*** passed to the `title` function
- □ String array enclosed in square brackets, [ ]

Annotations Using `num2str(…)`

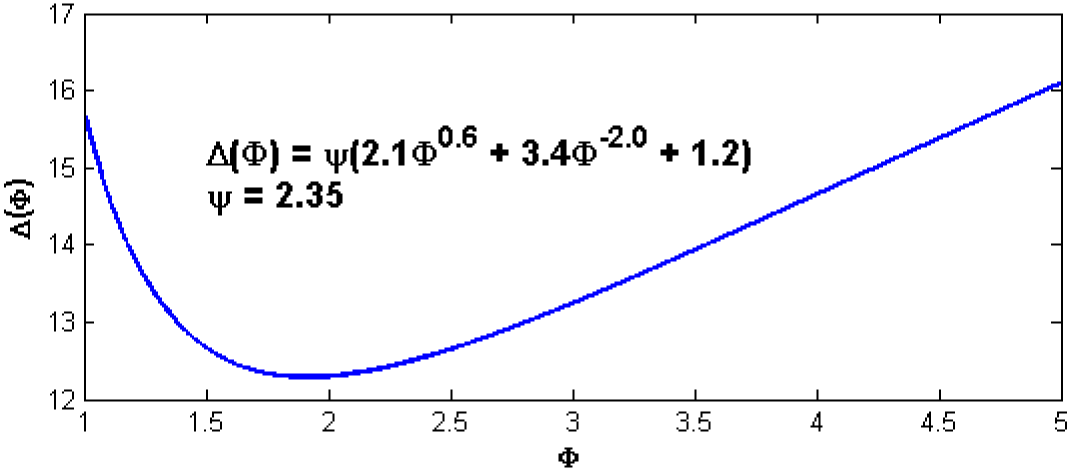
- □ Plot of : $\Delta(\phi) = \psi\left(a_1\phi^{b_1} + a_2\phi^{b_2} + a_3\right)$

  where: $\psi = 2.35$

  $a_1 = 2.1\,,\, a_2 = 3.4\,,\, a_3 = 1.2$

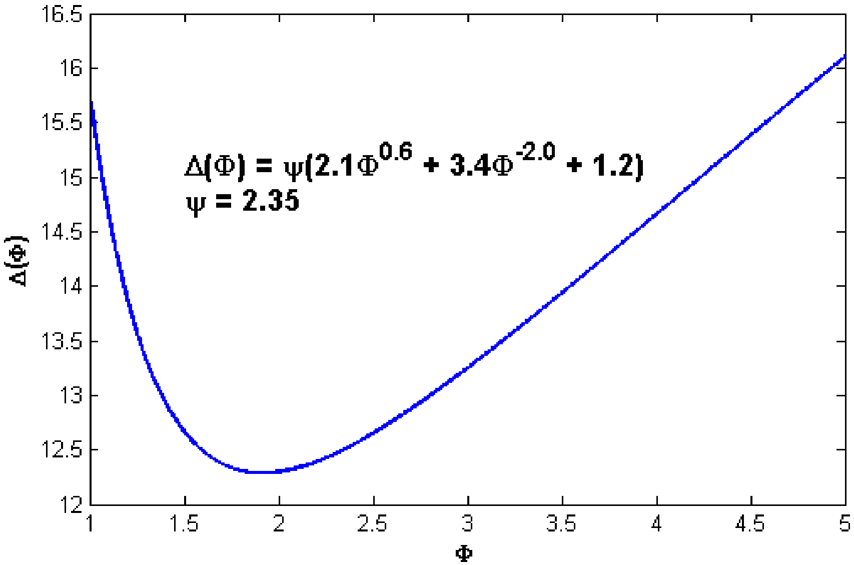
  $b_1 = 0.6\,,\, b_2 = -2.0$

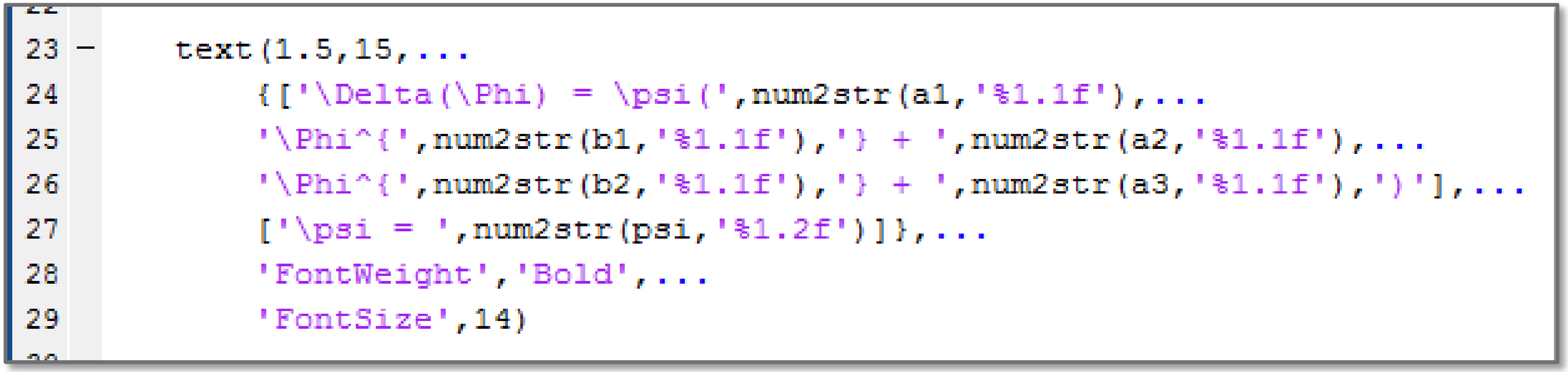
- □ Annotate the plot with the above function, substituting in a and b values
- □ Leave ψ as a variable, but annotate its value ***on a separate line***



Annotations Using `num2str(…)`

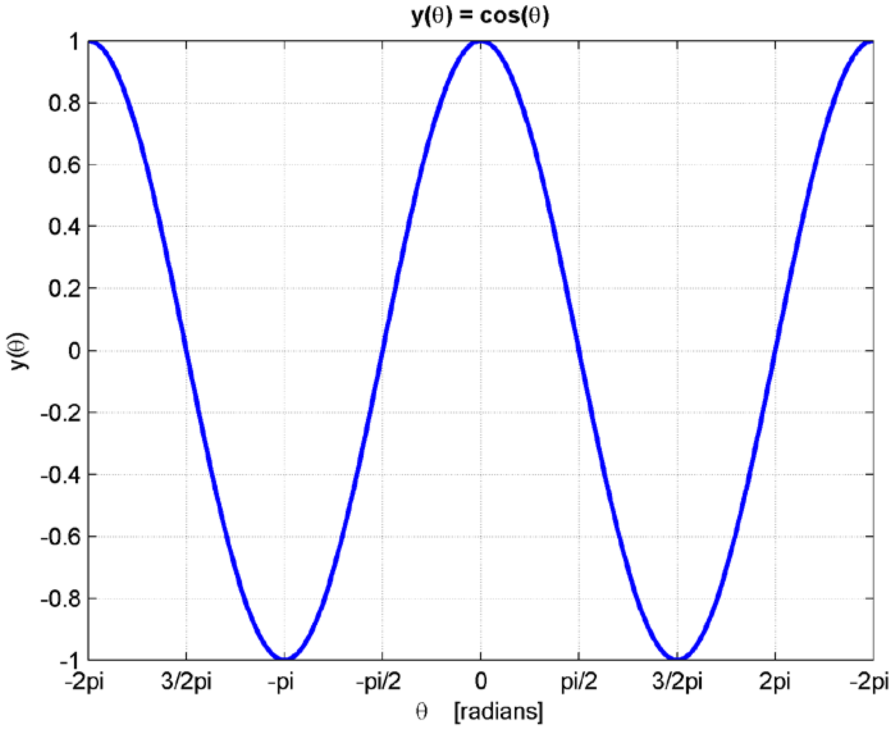
- Use text.m with a ***cell array*** input – enclose in { }
- One ***string array*** for each line of annotation text – enclose each in [ ]
- TeX character sequences can be included

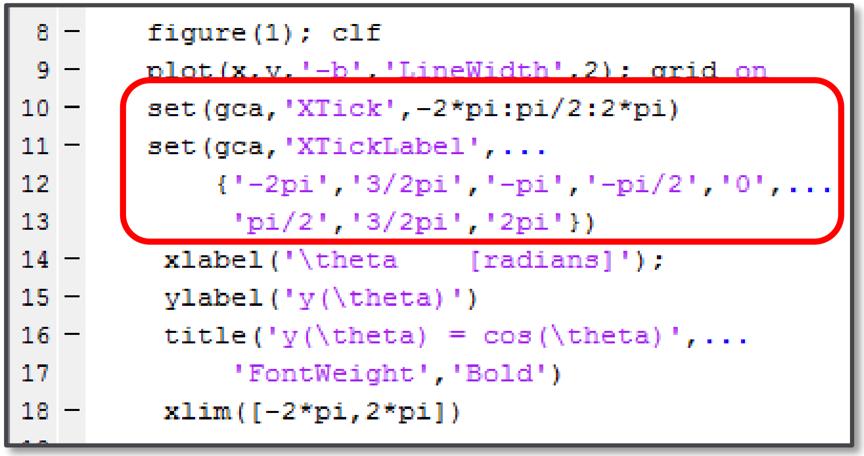




```
text(1.5,15,...
    {['\Delta(\Phi) = \psi(',num2str(a1,'%1.1f'),...
    '\Phi^{',num2str(b1,'%1.1f'),'} + ',num2str(a2,'%1.1f'),...
    '\Phi^{',num2str(b2,'%1.1f'),'} + ',num2str(a3,'%1.1f'),')'],...
    ['\psi = ',num2str(psi,'%1.2f')]},...
    'FontWeight','Bold',...
    'FontSize',14)
```

Controlling Axis Tick Marks – `XTickLabel`



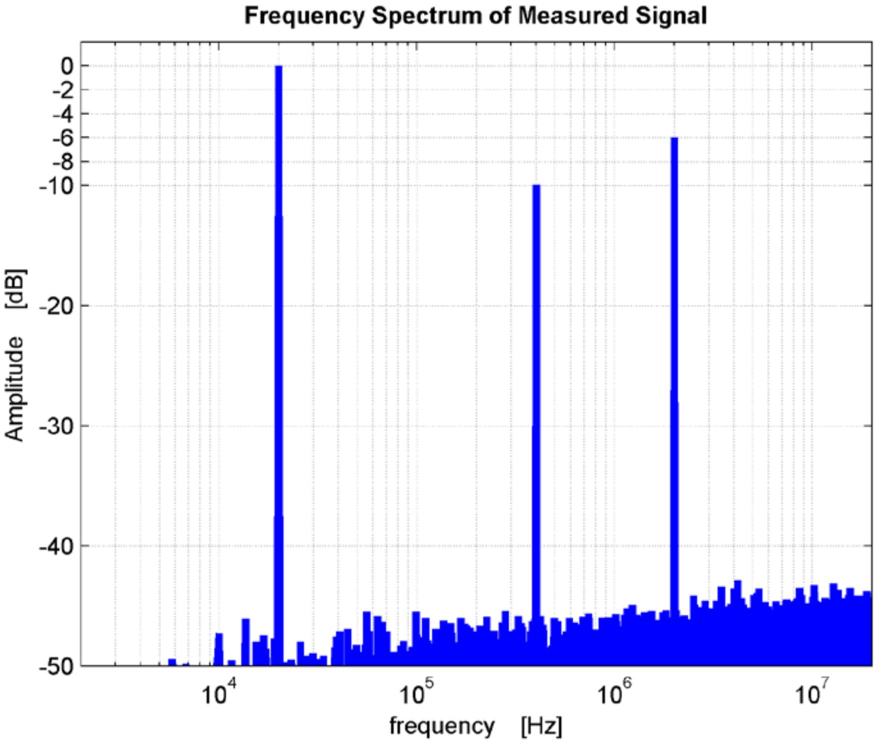


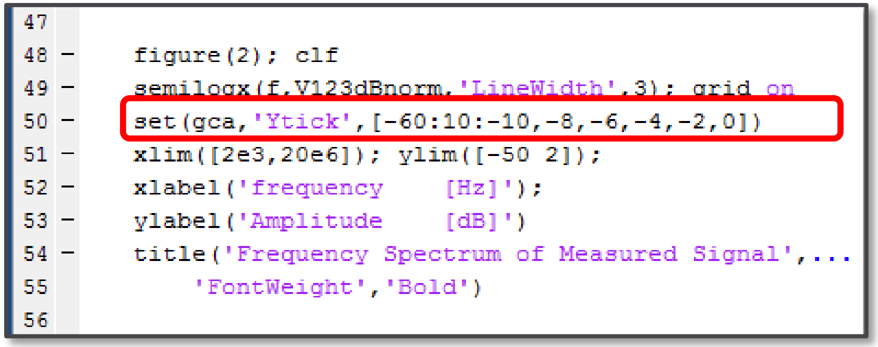
```
   figure(1); clf
   plot(x,y,'-b','LineWidth',2); grid on
   set(gca,'XTick',-2*pi:pi/2:2*pi)
   set(gca,'XTickLabel',...
       {'-2pi','3/2pi','-pi','-pi/2','0',...
        'pi/2','3/2pi','2pi'})
    xlabel('\theta     [radians]');
    ylabel('y(\theta)')
    title('y(\theta) = cos(\theta)',...
        'FontWeight','Bold')
    xlim([-2*pi,2*pi])
```

- `gca` – ***get current axes***
  - Returns a handle to the currently active axes
- ***Cell array*** enclosed in curly brackets, {…}

- `TickLabel` commands ***do not*** interpret TeX characters

Controlling Axis Tick Marks – `YTick`





```
    figure(2); clf
    semilogx(f,V123dBnorm,'LineWidth',3); grid on
    set(gca,'Ytick',[-60:10:-10,-8,-6,-4,-2,0])
    xlim([2e3,20e6]); ylim([-50 2]);
    xlabel('frequency     [Hz]');
    ylabel('Amplitude     [dB]')
    title('Frequency Spectrum of Measured Signal',...
        'FontWeight','Bold')
```

- Non-uniform `Tick` spacing is allowed
- If `TickLabel` is not specified, default (numeric) labels are placed at each tick mark

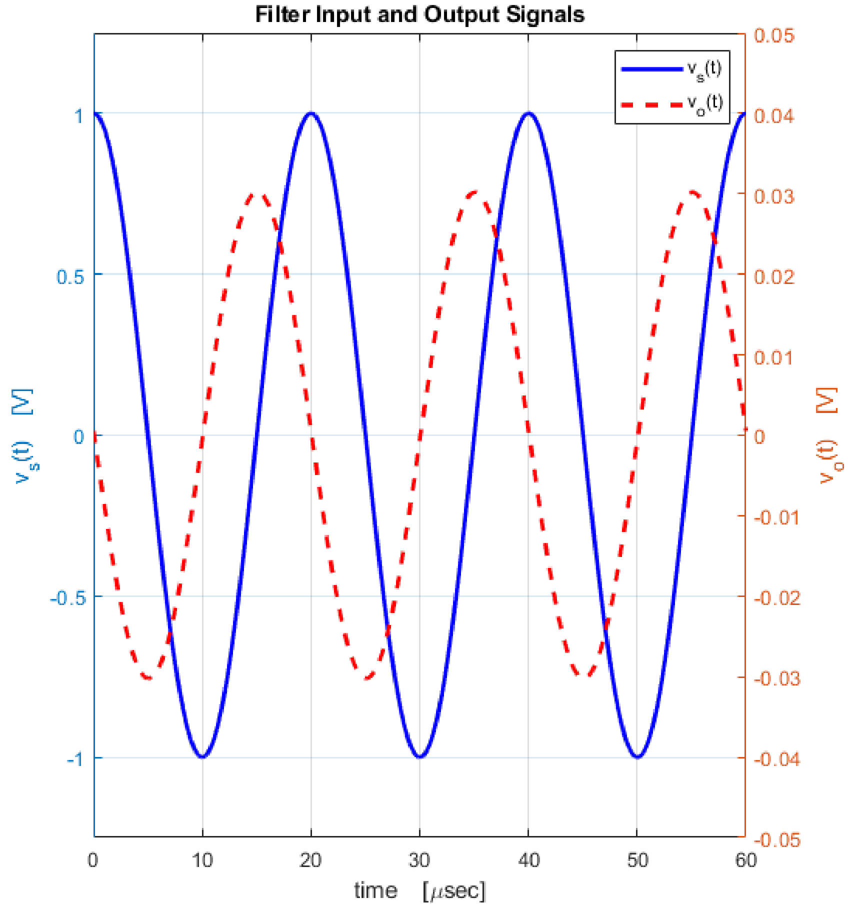
Dual y-Axes – `yyaxis`

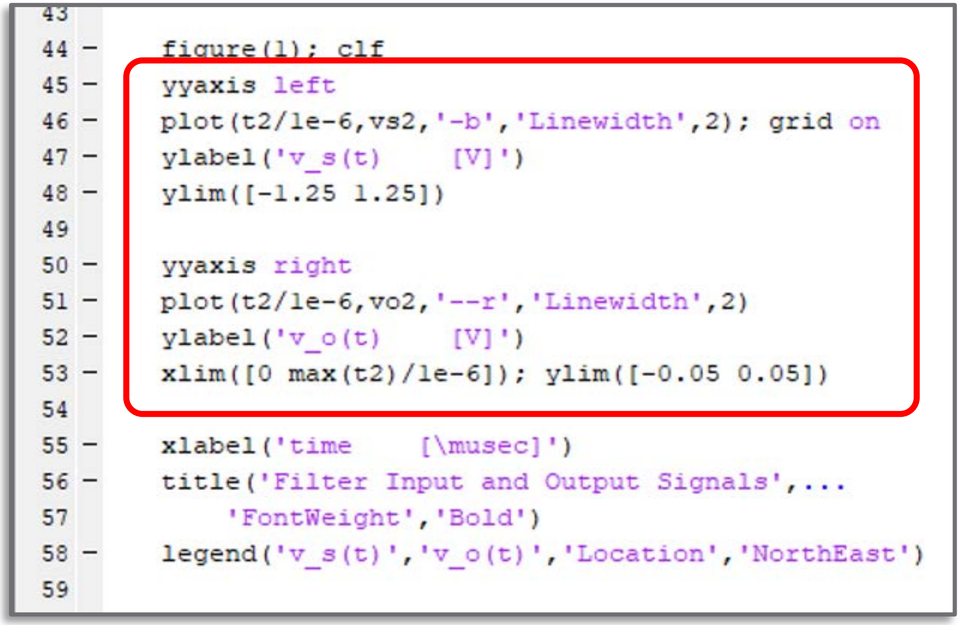
- Generate a plot with ***different y-axes on right and left sides of figure***

```
yyaxis right

yyaxis left
```

  - Useful for superimposing curves with very different vertical ranges and/or different units
- First `yyaxis` command creates a set of axes with a y-axis on both the left and right sides
- `yyaxis` commands activate the left- or right-hand-side axis
  - All subsequent plot and axis control (e.g. `ylabel`, `ylim`, etc.) commands applied to the currently-active y-axis

Dual y-Axes –`yyaxis`





```
figure(1); clf
yyaxis left
plot(t2/1e-6,vs2,'-b','Linewidth',2); grid on
ylabel('v_s(t)    [V]')
ylim([-1.25 1.25])

yyaxis right
plot(t2/1e-6,vo2,'--r','Linewidth',2)
ylabel('v_o(t)    [V]')
xlim([0 max(t2)/1e-6]); ylim([-0.05 0.05])

xlabel('time    [\musec]')
title('Filter Input and Output Signals',...
    'FontWeight','Bold')
legend('v_s(t)','v_o(t)','Location','NorthEast')
```

Logarithmic Axes

- Useful for displaying datasets that span a very large range
- ***Log-log plot*** – both axes are logarithmic

```
loglog(x,y,‘LineSpec’,…)
```

- ***Logarithmic X-axis***

```
semilogx(x,y,‘LineSpec’,…)
```

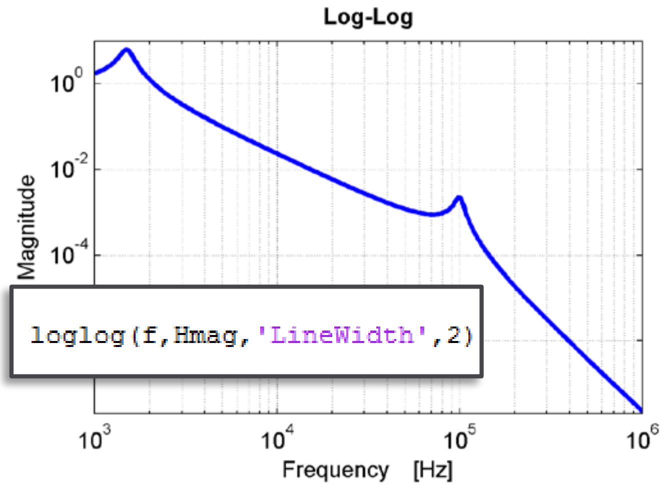
- ***Logarithmic Y-axis***

```
semilogy(x,y,‘LineSpec’,…)
```

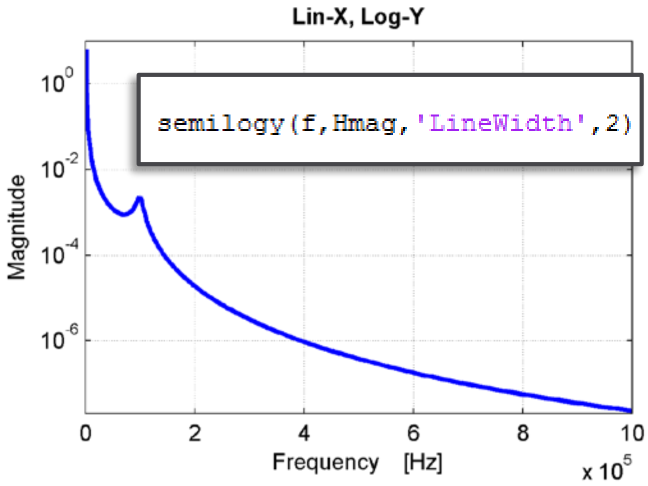
- Generating ind. variable vector for log-x plots:

```
logspace(X1,X2,N)
```

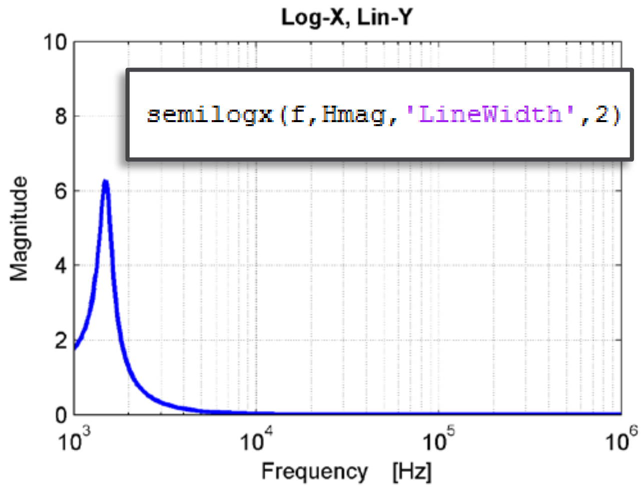
Logarithmic Axes



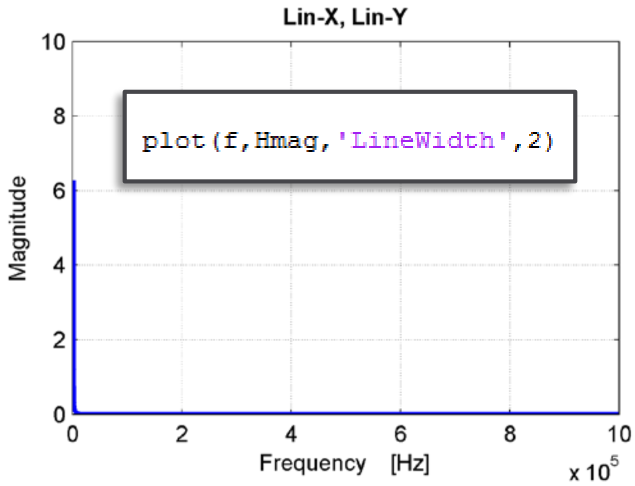
```
loglog(f,Hmag,'LineWidth',2)
```



```
semilogy(f,Hmag,'LineWidth',2)
```



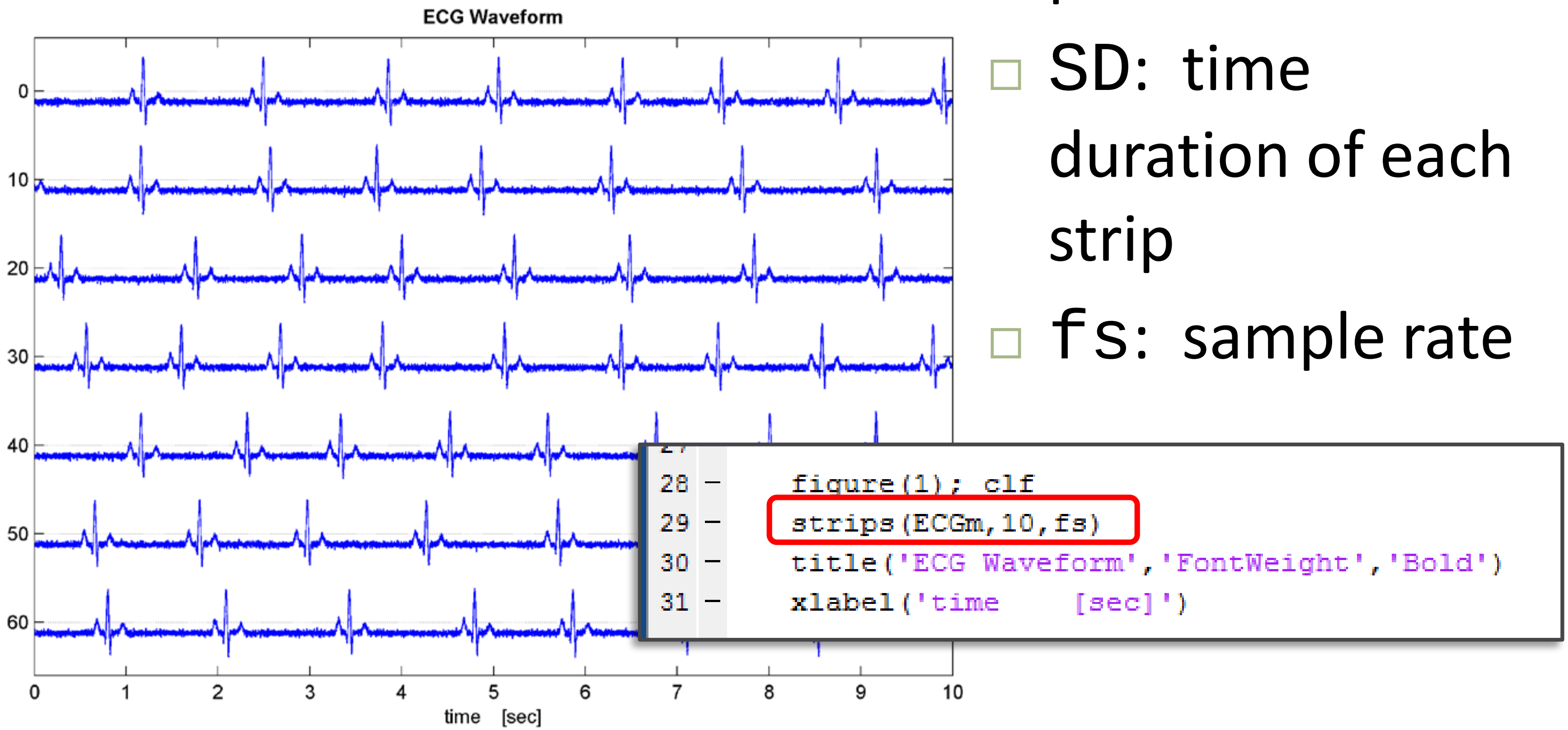
```
semilogx(f,Hmag,'LineWidth',2)
```



```
plot(f,Hmag,'LineWidth',2)
```

Strip Plot – `strips(…)`

```
strips(y,SD,fs,…)
```

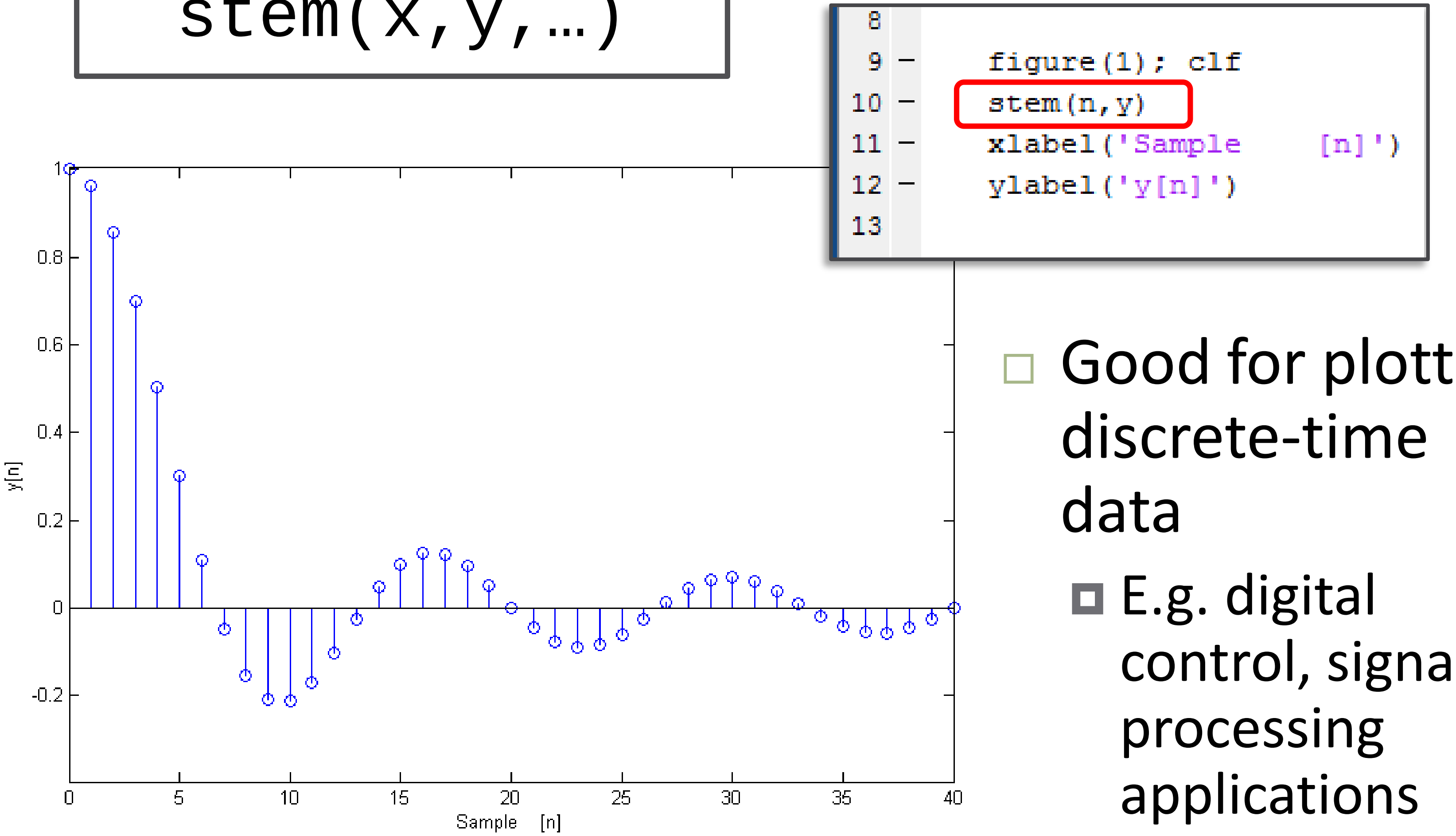


```
figure(1); clf
strips(ECGm,10,fs)
title('ECG Waveform','FontWeight','Bold')
xlabel('time     [sec]')
```

- □ `y`: data vector to plot
- □ `SD`: time duration of each strip
- □ `fs`: sample rate

Stem Plot – stem(…)

```
stem(x,y,…)
```



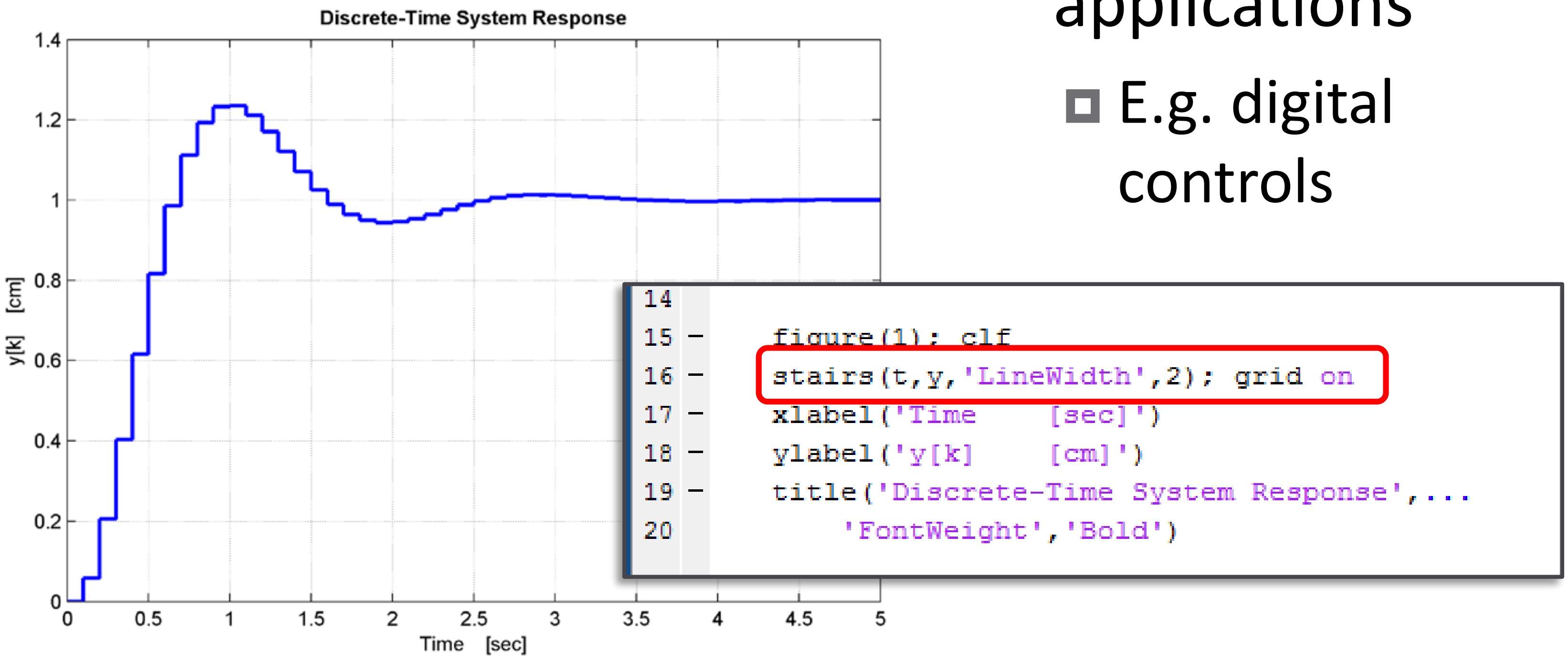
```
figure(1); clf
stem(n,y)
xlabel('Sample     [n]')
ylabel('y[n]')
```

- Good for plotting discrete-time data
  - E.g. digital control, signal processing applications

Plotting Zero-Order-Hold Data – `stairs(…)`

```
stairs(x,y,…)
```

- Again, useful for discrete-time applications
  - E.g. digital controls

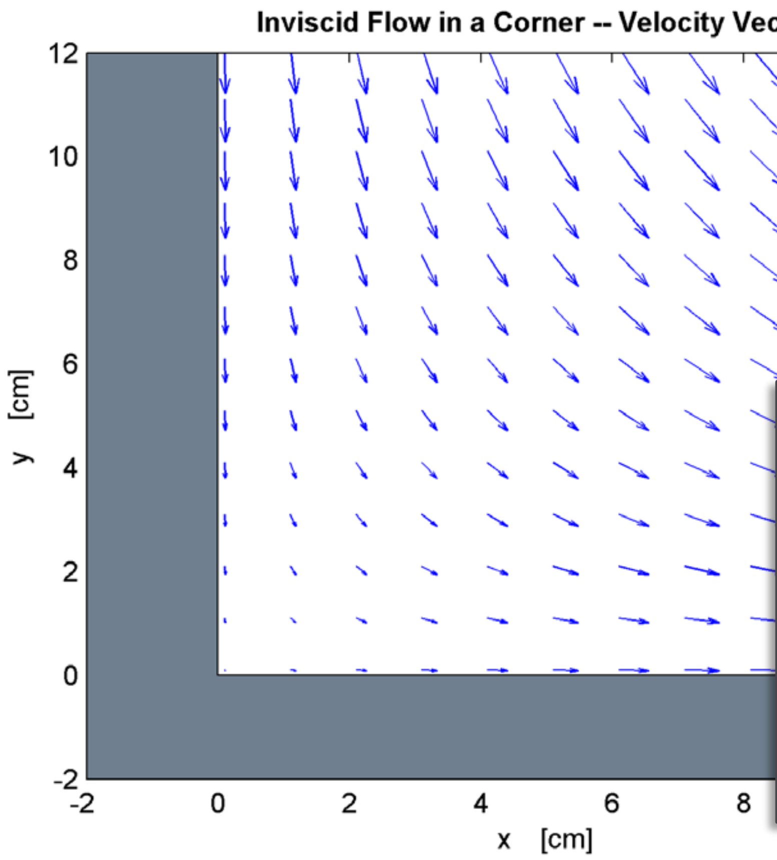


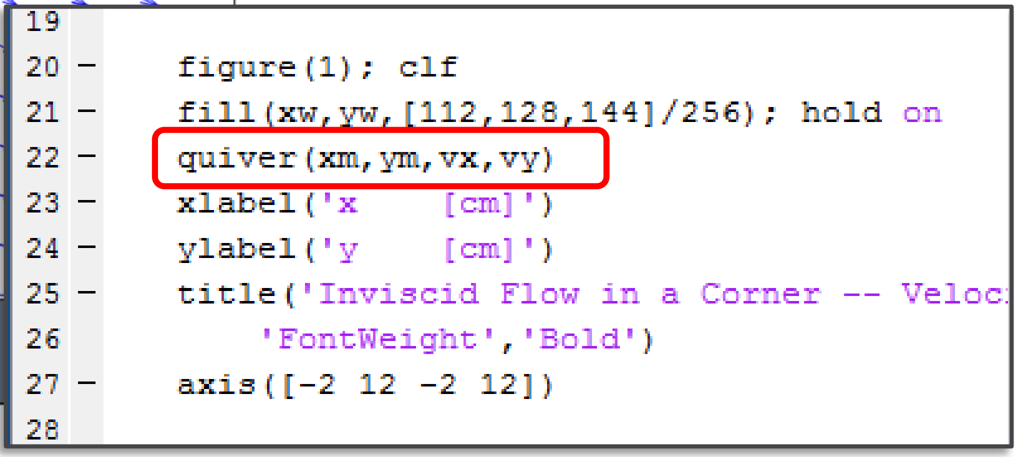
```
figure(1); clf
stairs(t,y,'LineWidth',2); grid on
xlabel('Time      [sec]')
ylabel('y[k]      [cm]')
title('Discrete-Time System Response',...
    'FontWeight','Bold')
```

Plotting Vector Fields – `quiver(…)`

```
quiver(x,y,u,v)
```

- `x,y`: matrices of x,y coordinates – generate with `meshgrid(…)` – more later
- `u,v`: velocity components at `x,y` locations – matrices



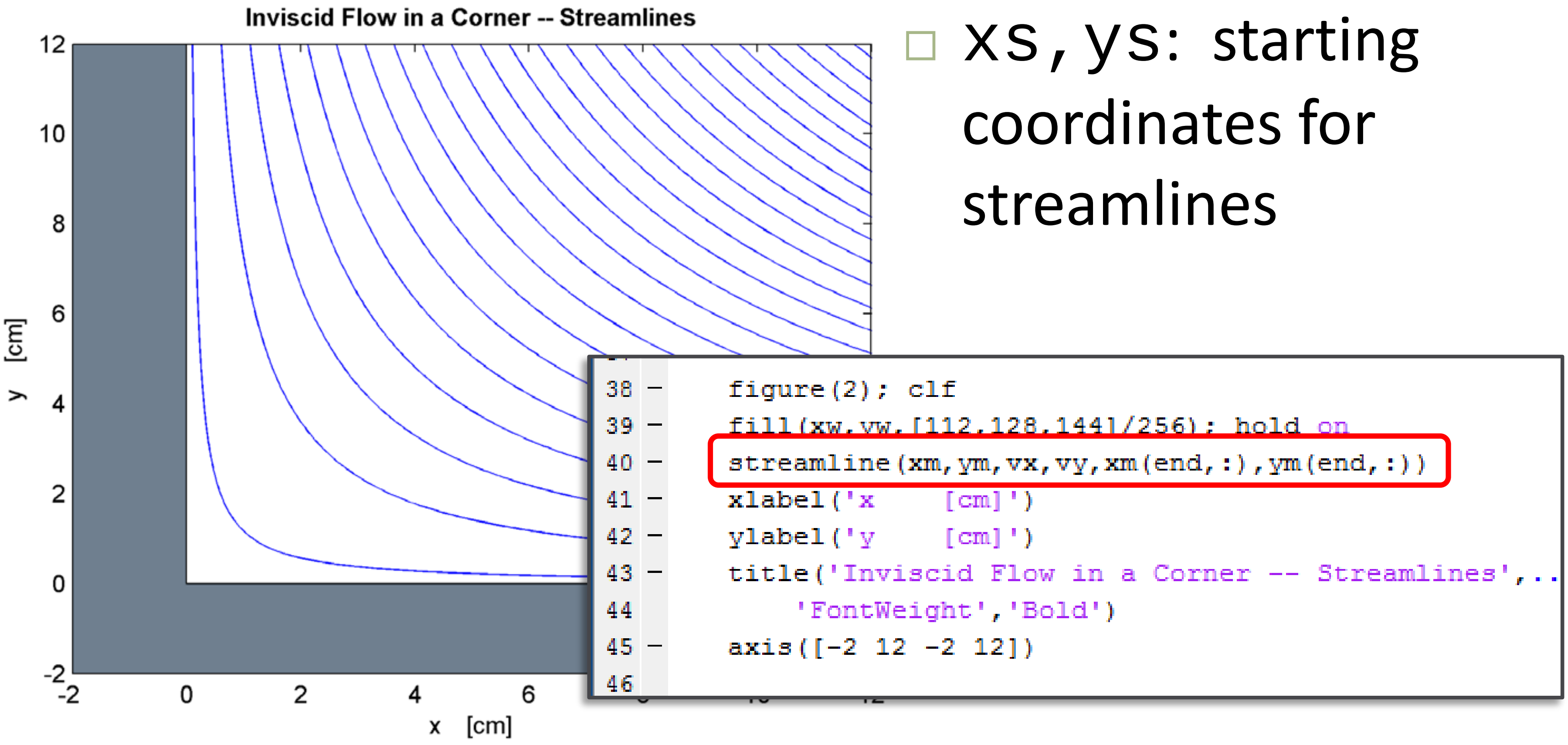


```
    figure(1); clf
    fill(xw,yw,[112,128,144]/256); hold on
    quiver(xm,ym,vx,vy)
    xlabel('x     [cm]')
    ylabel('y     [cm]')
    title('Inviscid Flow in a Corner -- Veloc
        'FontWeight','Bold')
    axis([-2 12 -2 12])
```

Streamline Plots – `streamline(…)`

```
streamline(x,y,u,v,xs,ys)
```

- □ `x,y,u,v`: same as for quiver(…)
- □ `xs,ys`: starting coordinates for streamlines



```
figure(2); clf
fill(xw,yw,[112,128,144]/256); hold on
streamline(xm,ym,vx,vy,xm(end,:),ym(end,:))
xlabel('x     [cm]')
ylabel('y     [cm]')
title('Inviscid Flow in a Corner -- Streamlines',...
    'FontWeight','Bold')
axis([-2 12 -2 12])
```