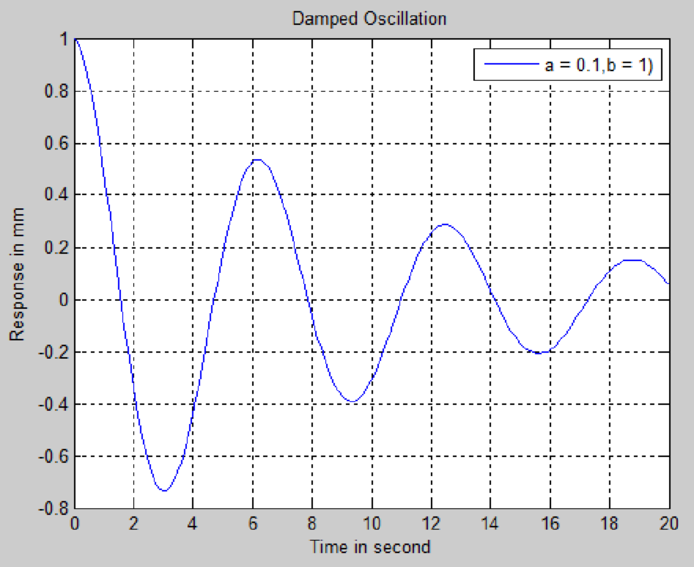
Agenda

- Introducing MATLAB – Language of technical computing
- Minimal MATLAB
  - MATLAB Desktop
  - Computing with MATLAB
  - Array – *creation & working*
  - **Plotting – *creating & visualizing***
  - Automating commands – *creating, saving & executing script file*
  - Conditional control statements
  - Loop control statements
  - Functions – *creating & executing a function file*

Example

Damped Oscillation equation: $y = e^{-at}\cos(bt)$

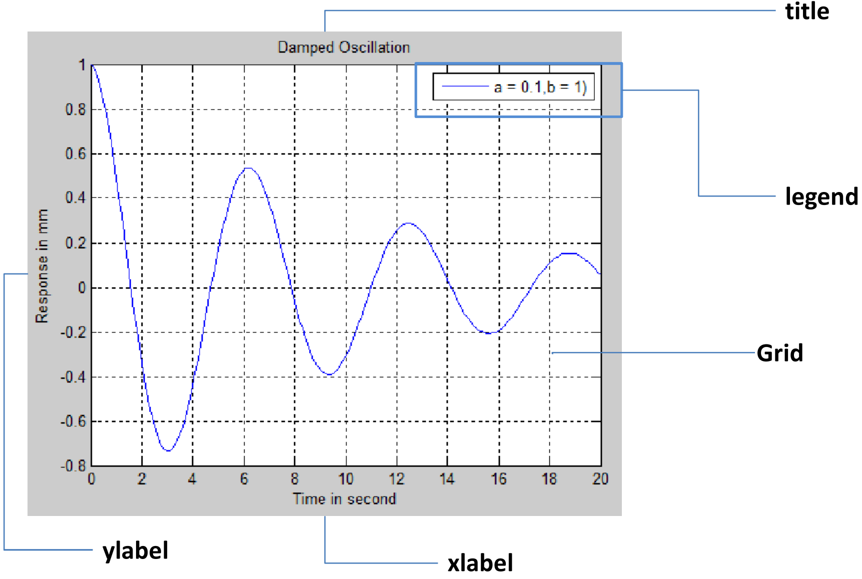


Example

Damped Oscillation equation: $y = e^{-at}\cos(bt)$

```
a = 0.1; b = 1;
t = 0:0.1:20;
y = exp(-a*t).*cos(b*t);
plot(t,y)
grid on;
xlabel('Time in second');
ylabel('Response in mm');
title('Damped Oscillation');
legend('a = 0.1,b = 1)');
```

Overview



title

legend

Grid

ylabel

xlabel

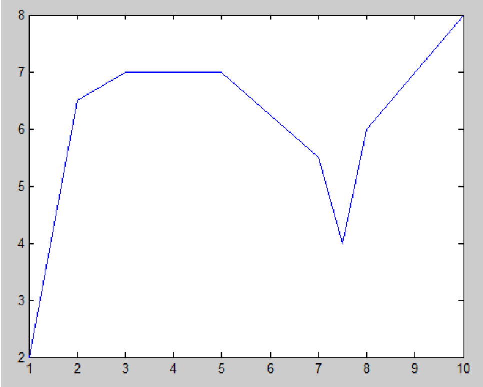
```
For more than one plot
legend('a = 0.1,b = 1)', 'a = 0.2,b = 2)')
```

Graphing

Plot function `plot(x,y)`

where the variables x and y are vector and must be of same size i.e the number of elements should be equal.

```
>>x=[1   2    3   5   7     7.5   8  10];
>>y=[2  6.5   7   7  5.5    4     6    8];
>>plot(x,y)
```



Graphing

Plot function

`plot(x,y,'line specifiers','PropertyName',PropertyValue)`

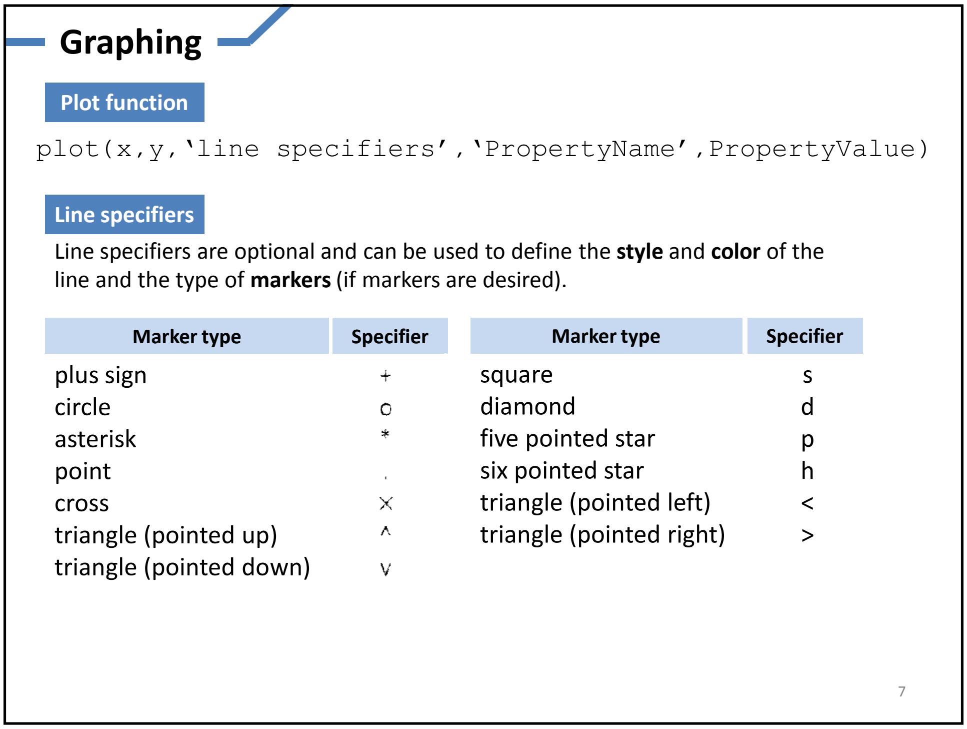
Line specifiers

Line specifiers are optional and can be used to define the **style** and **color** of the line and the type of **markers** (if markers are desired).

Line Style	Specifier
solid (default)	-
dashed	--
dotted	:
dash-dot	-.

Line color	Specifier
red	r
green	g
blue	b
cyan	c

Line color	Specifier
magenta	m
yellow	y
black	k
white	w



Graphing

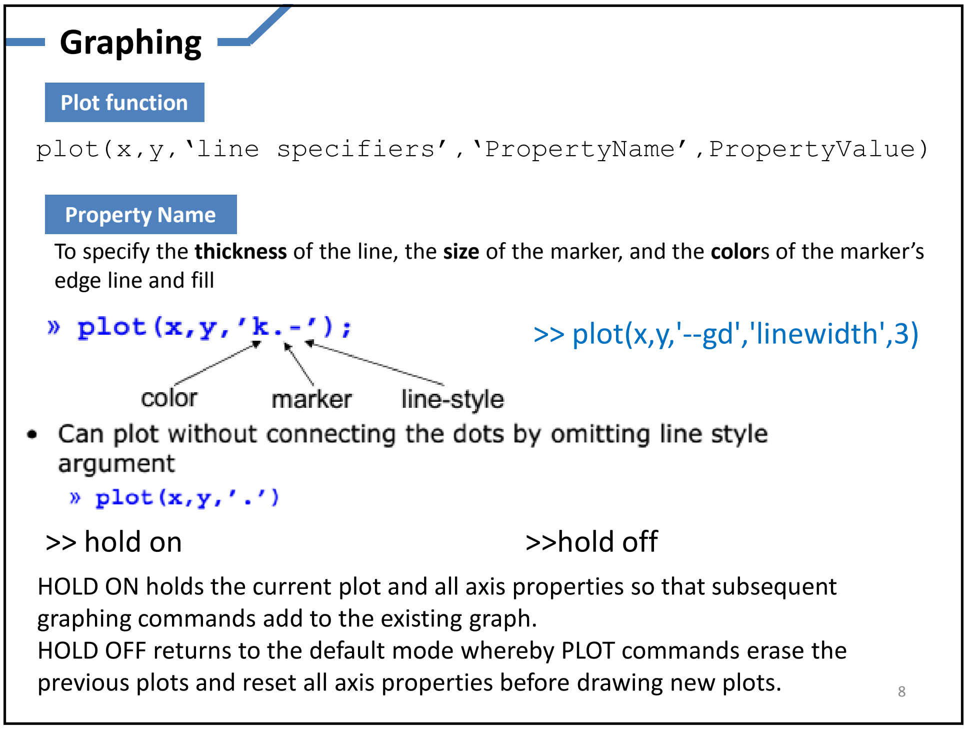
Plot function

```
plot(x,y,'line specifiers','PropertyName',PropertyValue)
```

Line specifiers

Line specifiers are optional and can be used to define the **style** and **color** of the line and the type of **markers** (if markers are desired).

Marker type	Specifier	Marker type	Specifier
plus sign	+	square	s
circle	o	diamond	d
asterisk	*	five pointed star	p
point	.	six pointed star	h
cross	x	triangle (pointed left)	<
triangle (pointed up)	^	triangle (pointed right)	>
triangle (pointed down)	v		



Graphing

Plot function

```
plot(x,y,'line specifiers','PropertyName',PropertyValue)
```

Property Name

To specify the **thickness** of the line, the **size** of the marker, and the **color**s of the marker's edge line and fill

```
» plot(x,y,'k.-');
```

color marker line-style

```
>> plot(x,y,'--gd','linewidth',3)
```

- Can plot without connecting the dots by omitting line style argument

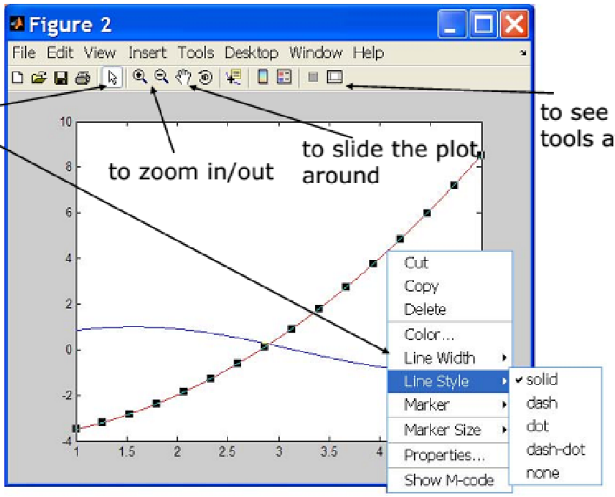
```
» plot(x,y,'.')
```

>> hold on >>hold off

HOLD ON holds the current plot and all axis properties so that subsequent graphing commands add to the existing graph.
HOLD OFF returns to the default mode whereby PLOT commands erase the previous plots and reset all axis properties before drawing new plots.

Graphing

Playing with the Plot



to select lines and delete or change properties

to zoom in/out

to slide the plot around

to see all plot tools at once

Courtesy of The MathWorks, Inc. Used with permission.

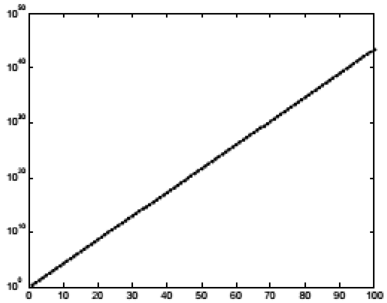
Graphing

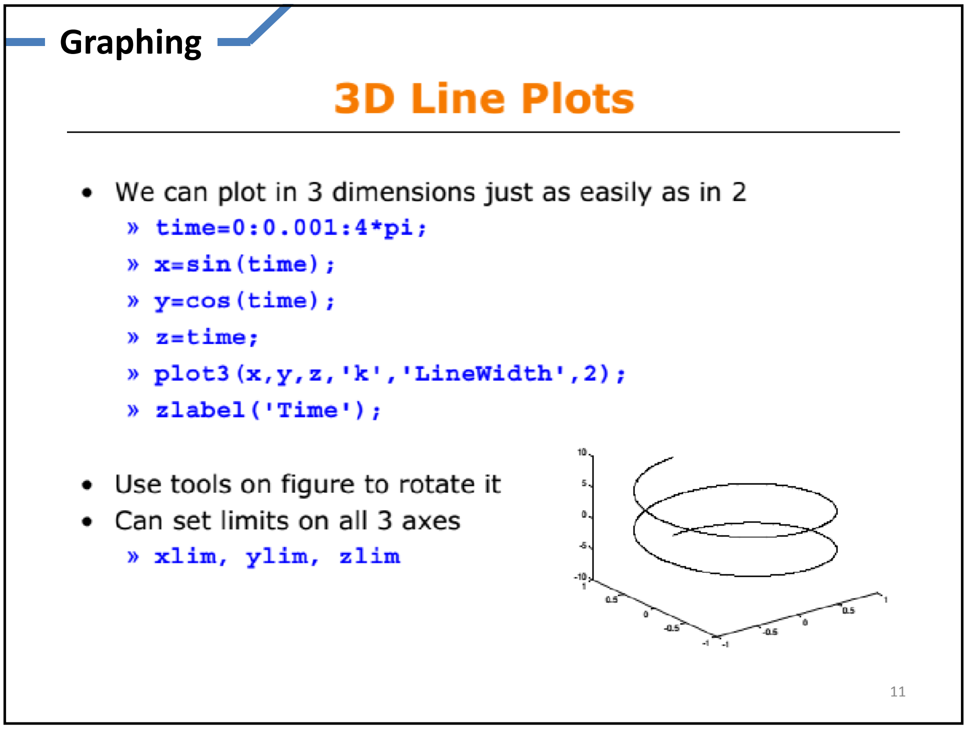
- The same syntax applies for semilog and loglog plots

```
» semilogx(x,y,'k');
» semilogy(y,'r.-');
» loglog(x,y);
```

- For example:

```
» x=0:100;
» semilogy(x,exp(x),'k.-');
```





Graphing

3D Line Plots

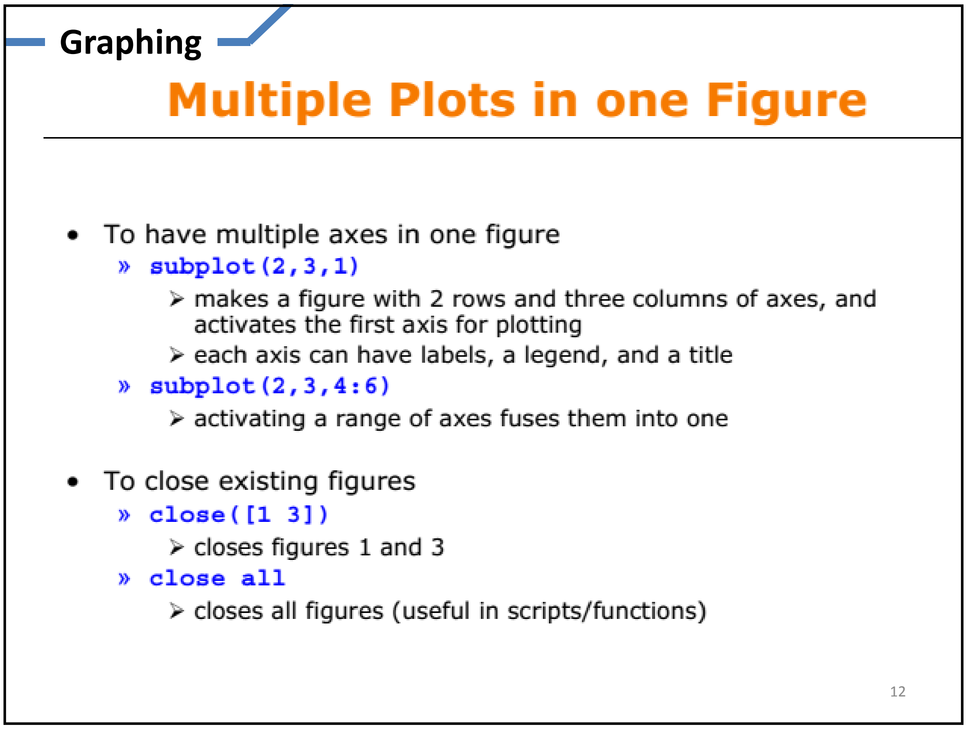
- We can plot in 3 dimensions just as easily as in 2

```
» time=0:0.001:4*pi;
» x=sin(time);
» y=cos(time);
» z=time;
» plot3(x,y,z,'k','LineWidth',2);
» zlabel('Time');
```

- Use tools on figure to rotate it
- Can set limits on all 3 axes

```
» xlim, ylim, zlim
```

11



Graphing

Multiple Plots in one Figure

- To have multiple axes in one figure

```
» subplot(2,3,1)
```

  - ➢ makes a figure with 2 rows and three columns of axes, and activates the first axis for plotting
  - ➢ each axis can have labels, a legend, and a title

```
» subplot(2,3,4:6)
```

  - ➢ activating a range of axes fuses them into one

- To close existing figures

```
» close([1 3])
```

  - ➢ closes figures 1 and 3

```
» close all
```

  - ➢ closes all figures (useful in scripts/functions)

12

Graphing

Copy/Paste Figures

- Figures can be pasted into other apps (word, ppt, etc)
- *Edit→ copy options→ figure copy template*
  - ➢ Change font sizes, line properties; presets for word and ppt
- *Edit→ copy figure* to copy figure
- Paste into document of interest

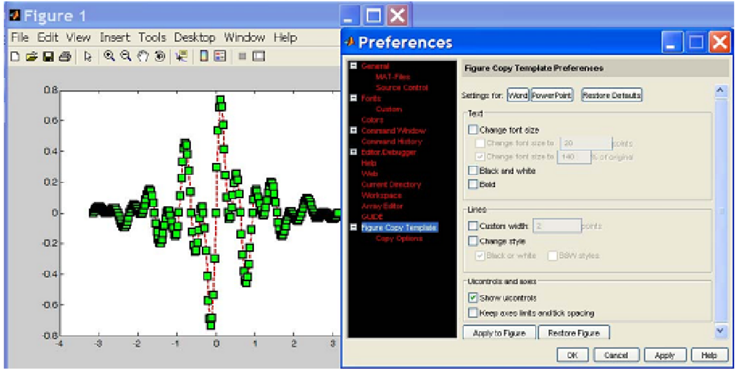


Figure Copy Template Preferences

Settings for:

Text

☐ Change font size

☐ Change font size to 20 points

☑ Change font size to 140 % of original

☐ Black and white

☐ Bold

Lines

☐ Custom width: 2 points

☐ Change style

☑ Black or white ☐ B&W styles

Uicontrols and axes

☑ Show uicontrols

☐ Keep axes limits and tick spacing

Graphing

Saving Figures

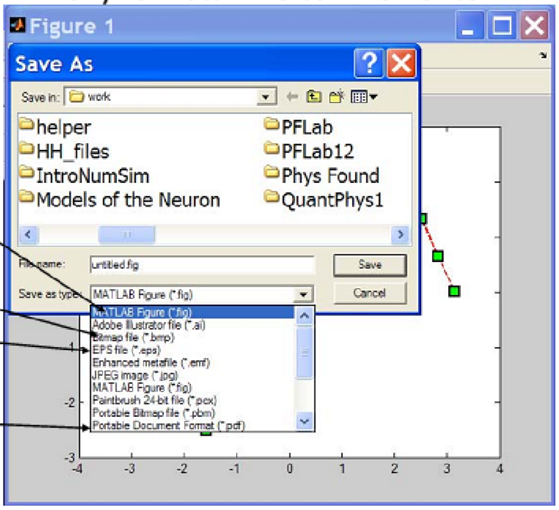
- Figures can be saved in many formats. The common ones are:

.fig preserves all information

.bmp uncompressed image

.eps high-quality scaleable format

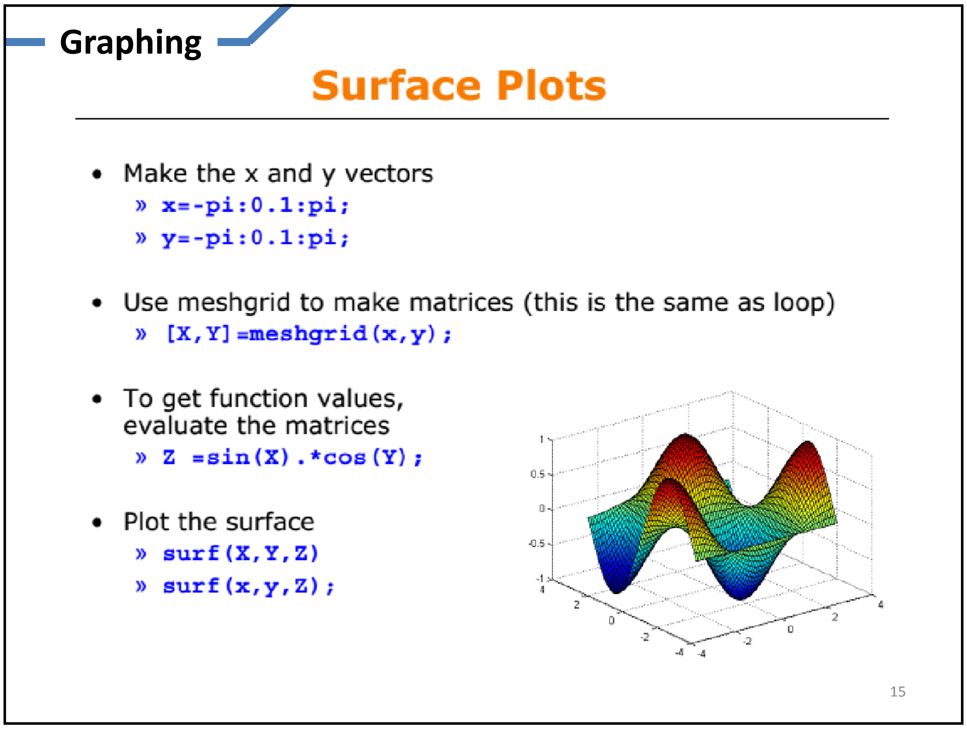
.pdf compressed image



Save in: work

File name: untitled.fig

Save as type: MATLAB Figure (*.fig)

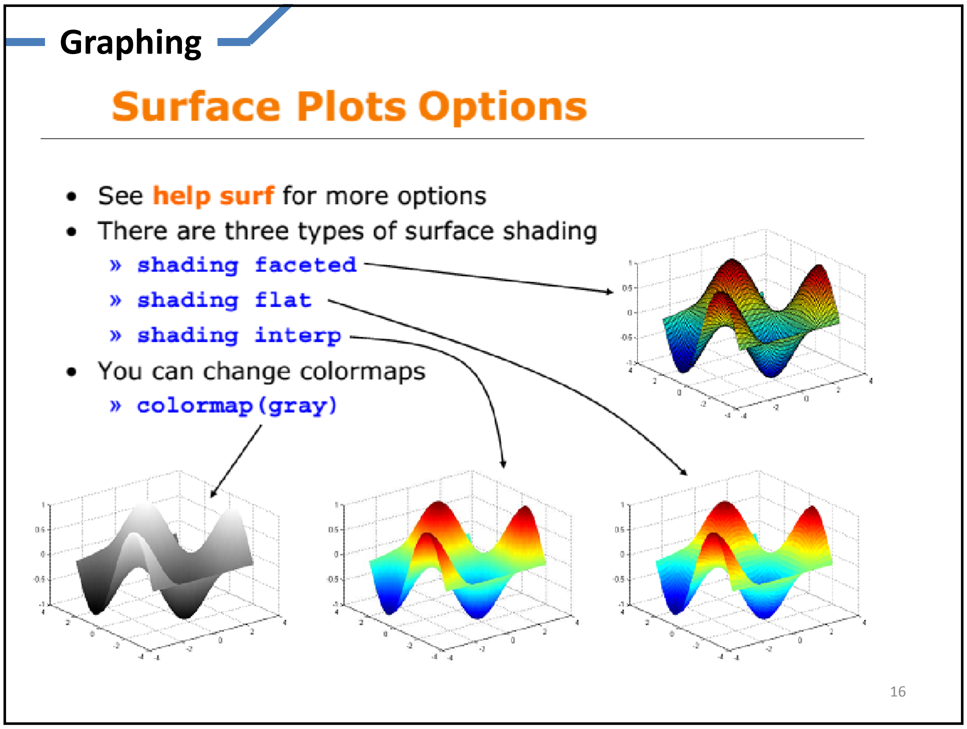


Graphing

Surface Plots

- Make the x and y vectors
  - » `x=-pi:0.1:pi;`
  - » `y=-pi:0.1:pi;`
- Use meshgrid to make matrices (this is the same as loop)
  - » `[X,Y]=meshgrid(x,y);`
- To get function values, evaluate the matrices
  - » `Z =sin(X).*cos(Y);`
- Plot the surface
  - » `surf(X,Y,Z)`
  - » `surf(x,y,Z);`

15

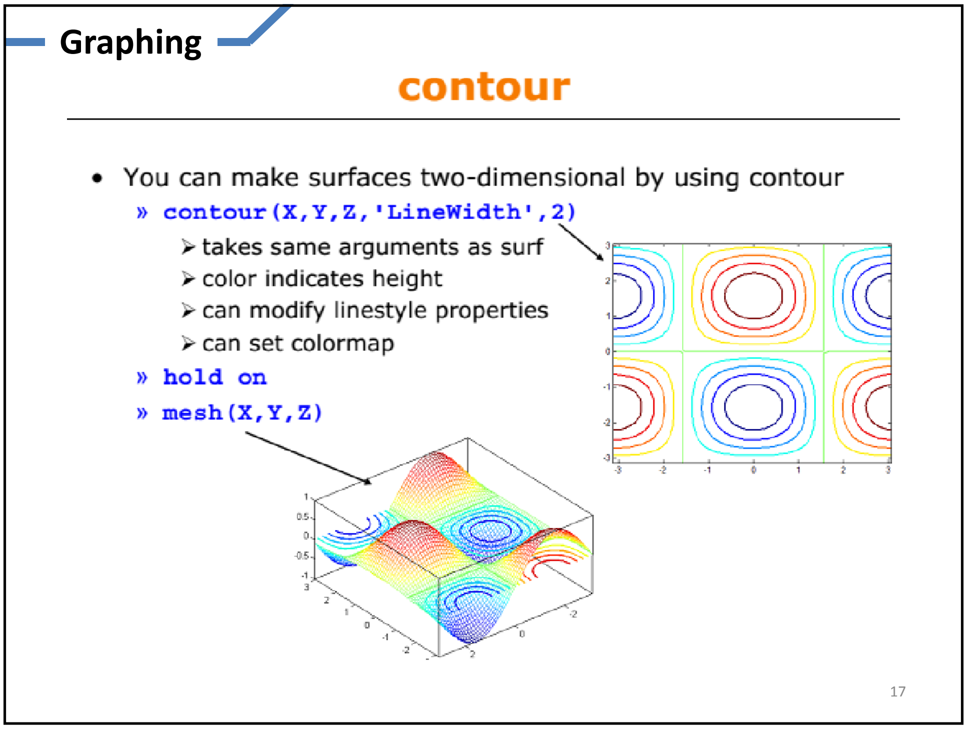


Graphing

Surface Plots Options

- See **help surf** for more options
- There are three types of surface shading
  - » `shading faceted`
  - » `shading flat`
  - » `shading interp`
- You can change colormaps
  - » `colormap(gray)`

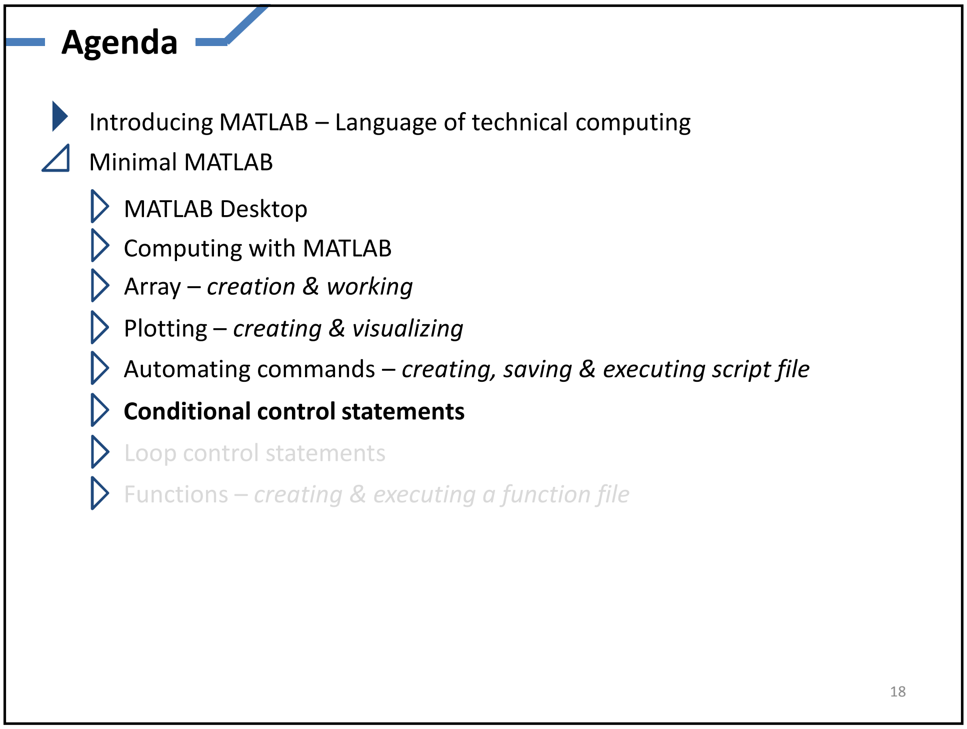
16



Graphing

contour

- You can make surfaces two-dimensional by using contour
  - » `contour(X,Y,Z,'LineWidth',2)`
    - ➢ takes same arguments as surf
    - ➢ color indicates height
    - ➢ can modify linestyle properties
    - ➢ can set colormap
  - » `hold on`
  - » `mesh(X,Y,Z)`



Agenda

- Introducing MATLAB – Language of technical computing
- Minimal MATLAB
  - MATLAB Desktop
  - Computing with MATLAB
  - Array – *creation & working*
  - Plotting – *creating & visualizing*
  - Automating commands – *creating, saving & executing script file*
  - **Conditional control statements**
  - Loop control statements
  - Functions – *creating & executing a function file*

Conditional Statement

Conditional expression consisting of relational and/or logical operators.

Relational Operators		Logical Operators	
Operations	**Operators**	**Operations**	**Operators**
Less than	<	And	&&
Greater than	>	Or	
Less than or equal to	<=	Not	~
Greater than or equal to	>=		
Equal to	==		
Not Equal to	~=	➢ Xor	xor
		➢ All true	all
		➢ Any true	any

- Boolean values: zero is false, nonzero is true
- See **help .** for a detailed list of operators

Example

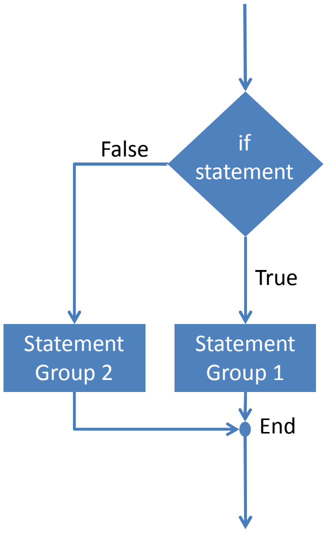
Plot

$$g(t) = \begin{cases} 0, & t < -2 \\ -4 - 2t, & -2 < t < 0 \quad \leftarrow \text{g1} \\ -4 - 3t, & 0 < t < 4 \quad \leftarrow \text{g2} \\ 16 - 2t, & 4 < t < 8 \quad \leftarrow \text{g3} \\ 0, & t > 8 \end{cases}$$

```
t = linspace(-5,10);
g1 = -4 - 2*t;
g2 = -4 + 3*t;
g3 = 16 - 2*t;
g =    g1.*(-2<t & t<=0)...
     + g2.*(0<t & t<=4)...
     + g3.*(4<t & t<=8);
plot(t,g)
```

if/elseif/else Statement

```
if expression
   statements
elseif expression
   statements
else
   statements
end
```



Some Examples

```
if ((a>3) & (b==5))
   Some Matlab Commands;
end
```

```
if (a<3)
   Some Matlab Commands;
elseif (b~=5)
   Some Matlab Commands;
end
```

```
if (a<3)
   Some Matlab Commands;
else
   Some Matlab Commands;
end
```

Switch Statement

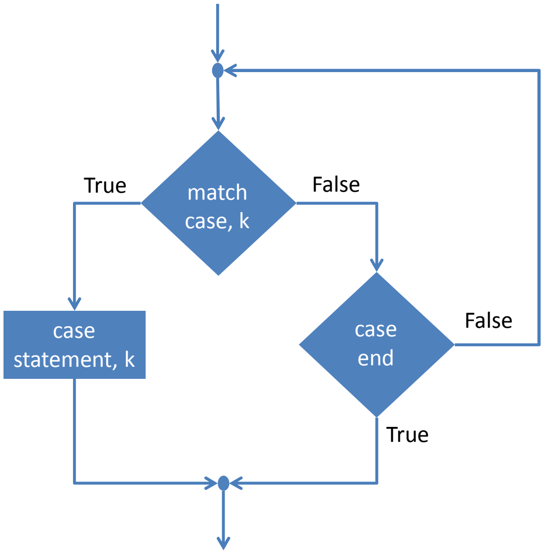
```
switch switch_expression
   case case_expression
      statements
   case case_expression
      statements
     :
   otherwise
      statements
end
```

Example:

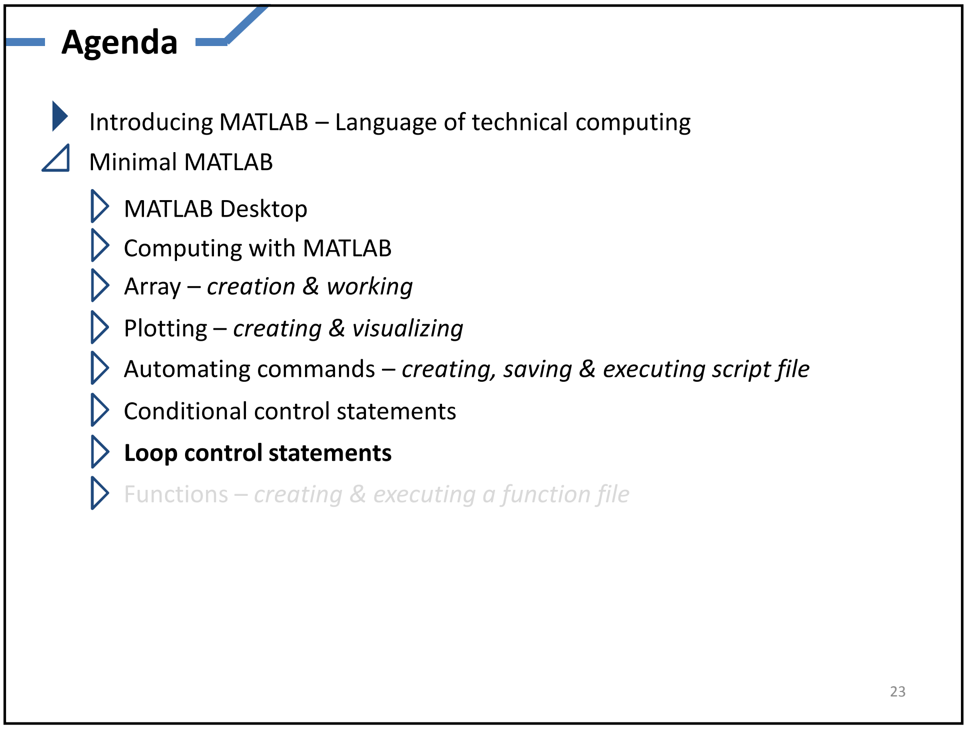
```
method = 'Bilinear';

switch lower(method)
  case {'linear','bilinear'}
    disp('Method is linear')
  case 'cubic'
    disp('Method is cubic')
  case 'nearest'
    disp('Method is nearest')
  otherwise
    disp('Unknown method.')
end
```

Flowchart

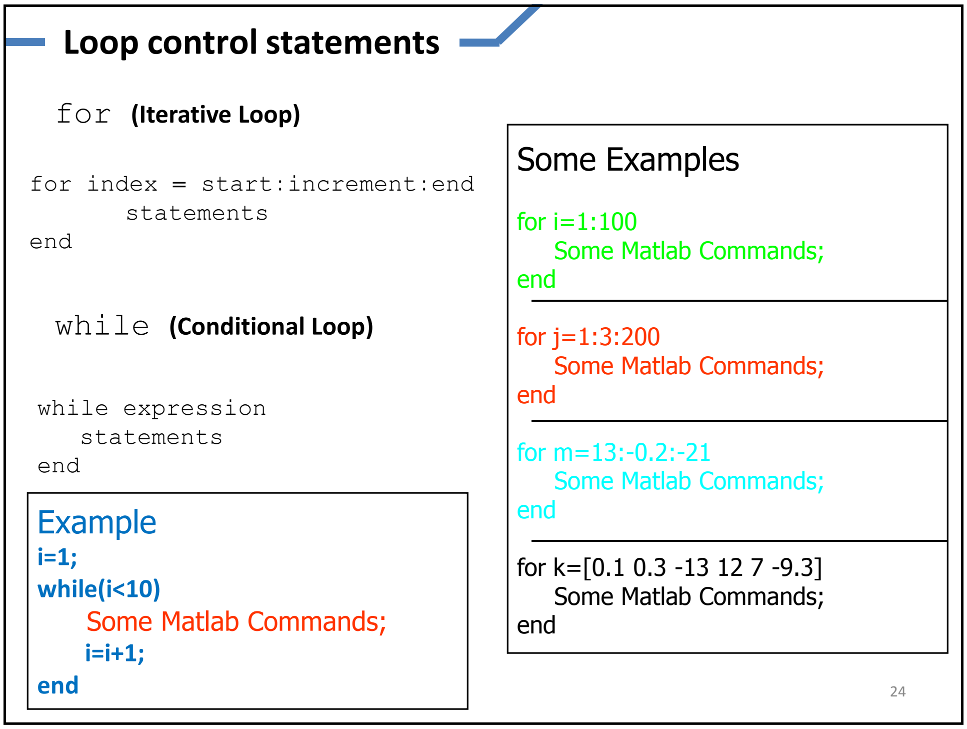


Method is linear



Agenda

- Introducing MATLAB – Language of technical computing
- Minimal MATLAB
  - MATLAB Desktop
  - Computing with MATLAB
  - Array – *creation & working*
  - Plotting – *creating & visualizing*
  - Automating commands – *creating, saving & executing script file*
  - Conditional control statements
  - **Loop control statements**
  - Functions – *creating & executing a function file*



Loop control statements

`for` **(Iterative Loop)**

```
for index = start:increment:end
        statements
end
```

`while` **(Conditional Loop)**

```
while expression
   statements
end
```

Example

```
i=1;
while(i<10)
    Some Matlab Commands;
    i=i+1;
end
```

Some Examples

```
for i=1:100
    Some Matlab Commands;
end
```

```
for j=1:3:200
    Some Matlab Commands;
end
```

```
for m=13:-0.2:-21
    Some Matlab Commands;
end
```

```
for k=[0.1 0.3 -13 12 7 -9.3]
    Some Matlab Commands;
end
```

Continue Statement

```
fid = fopen('magic.m','r');
count = 0;
while ~feof(fid)
    line = fgetl(fid);
    if isempty(line) | strncmp(line,'%',1)
       continue
    end
    count = count + 1;
end
disp(sprintf('%d lines',count));
```

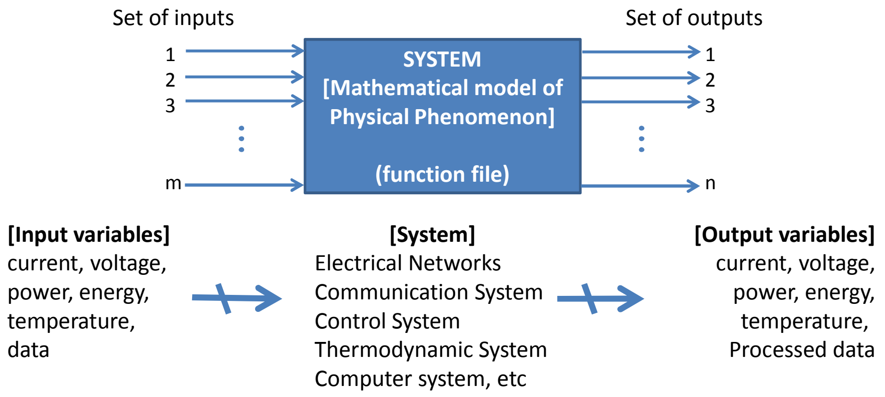
Break Statement

```
a = 0; fa = -Inf;
b = 3; fb = Inf;
while b-a > eps*b
   x = (a+b)/2;
   fx = x^3-2*x-5;
   if fx == 0
      break
   elseif sign(fx) == sign(fa)
      a = x; fa = fx;
   else
      b = x; fb = fx;
   end
end
x
```

Agenda

- Introducing MATLAB – Language of technical computing
- Minimal MATLAB
  - MATLAB Desktop
  - Computing with MATLAB
  - Array – *creation & working*
  - Plotting – *creating & visualizing*
  - Automating commands – *creating, saving & executing script file*
  - Conditional control statements
  - Loop control statements
  - **Functions – *creating & executing a function file***

Overview



Set of inputs

1
2
3
⋮
m

SYSTEM
[Mathematical model of Physical Phenomenon]

(function file)

Set of outputs

1
2
3
⋮
n

[Input variables]
current, voltage,
power, energy,
temperature,
data

[System]
Electrical Networks
Communication System
Control System
Thermodynamic System
Computer system, etc

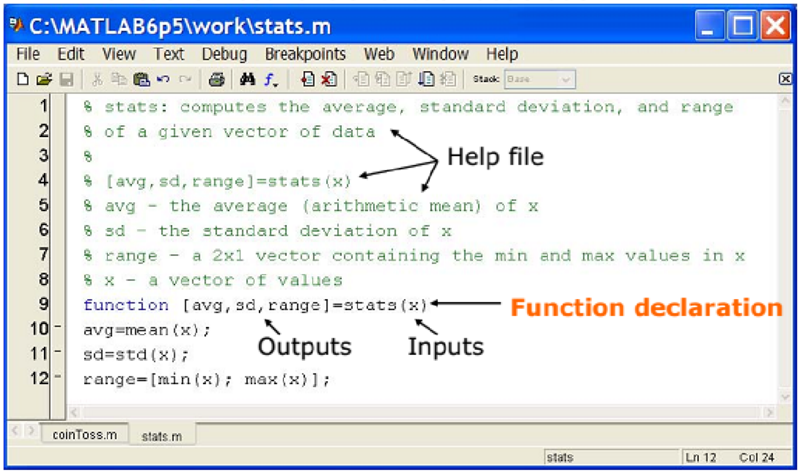
[Output variables]
current, voltage,
power, energy,
temperature,
Processed data

```
function [out1, out2, ...] = func_name(in1, in2, ...)
```

declares the function func_name, and its inputs and outputs

User-Defined Function

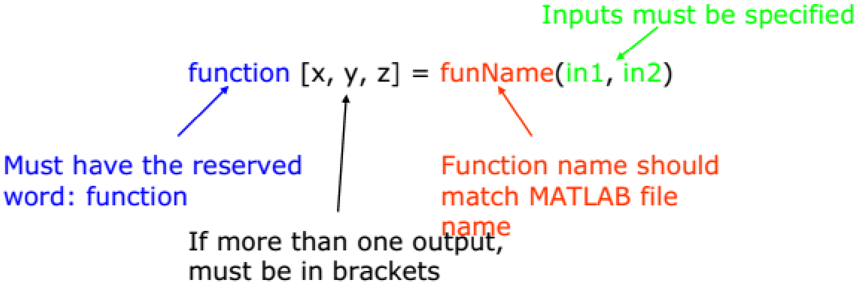
- Functions look exactly like scripts, but for **ONE** difference
  - Functions must have a function declaration



```
% stats: computes the average, standard deviation, and range
% of a given vector of data
%
% [avg,sd,range]=stats(x)
% avg - the average (arithmetic mean) of x
% sd - the standard deviation of x
% range - a 2x1 vector containing the min and max values in x
% x - a vector of values
function [avg,sd,range]=stats(x)
avg=mean(x);
sd=std(x);
range=[min(x); max(x)];
```

User-Defined Function

- Some comments about the function declaration



Inputs must be specified

function [x, y, z] = funName(in1, in2)

Must have the reserved word: function

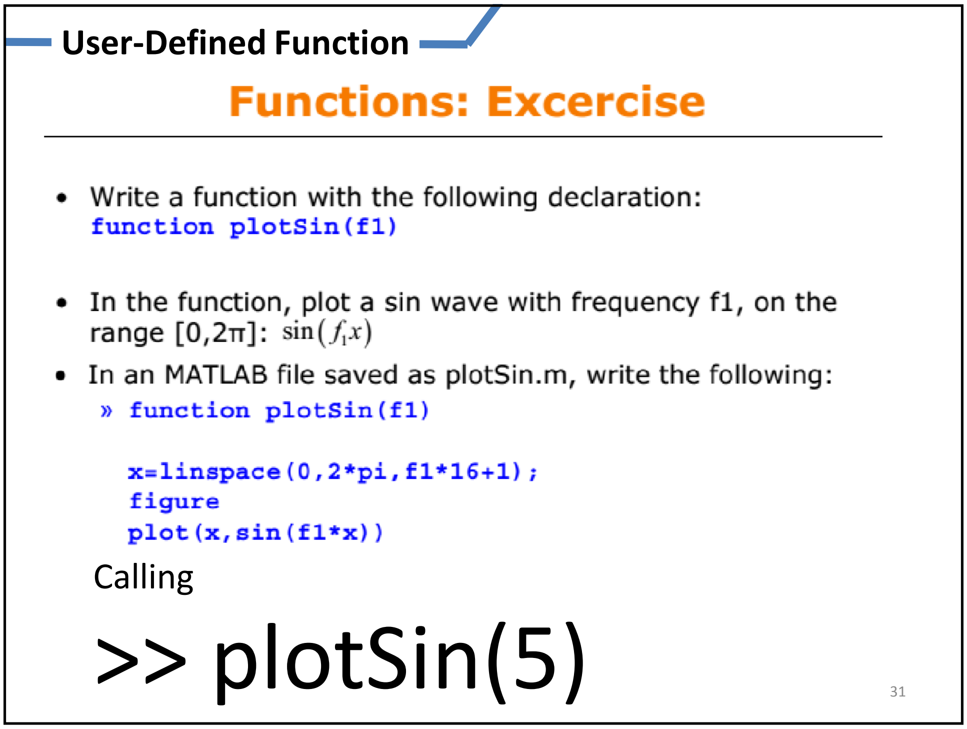
Function name should match MATLAB file name

If more than one output, must be in brackets

- Variable scope: Any variables created within the function but not returned disappear after the function stops running

Local Variables

To make variables appear after the function stops running:
>> help global



User-Defined Function

Functions: Excercise

- Write a function with the following declaration:
  `function plotSin(f1)`
- In the function, plot a sin wave with frequency f1, on the range [0,2π]: $\sin(f_1 x)$
- In an MATLAB file saved as plotSin.m, write the following:

```
» function plotSin(f1)

  x=linspace(0,2*pi,f1*16+1);
  figure
  plot(x,sin(f1*x))
```

Calling

```
>> plotSin(5)
```

31

THANKS

32