Matlab 2: 2D and 3D Graph

Cheng-Hsin Hsu

National Tsing Hua University

Department of Computer Science

Slides are based on the materials from Prof. Roger Jang

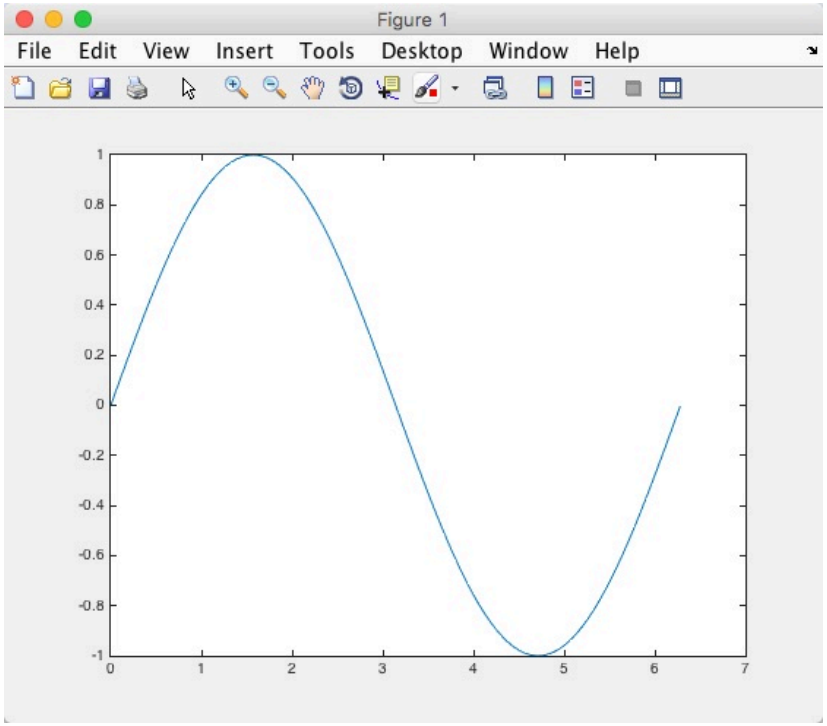
2-D Plot

- Plot takes two vectors as inputs and creates a curve
- We refer to each data point as a sample
- Samples are connected via straight line segments
- Try the following sample code on your computer

```
x = linspace(0, 2*pi);    % create a vector of 100
                          % samples between 0 and 2pi
y = sin(x);     % calculate the sin values of samples
plot(x, y);     % plot a 2D curve in a new window
```

2-D Plot (cont.)

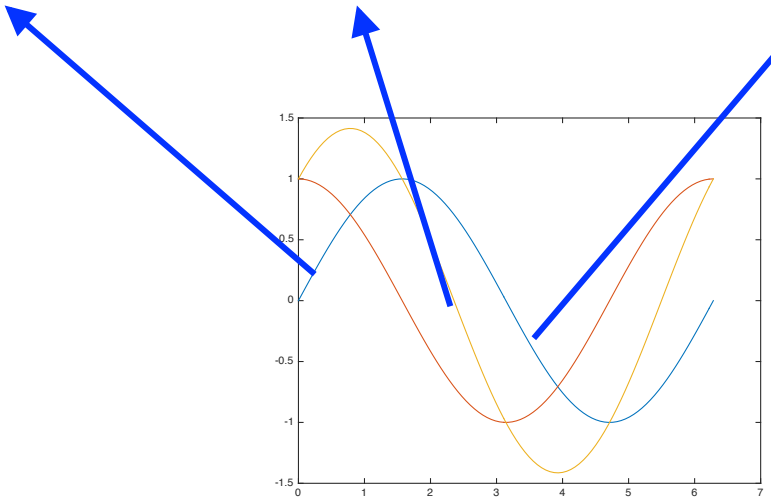
- The two vectors (*x* and *y* in our example) should be in the same length
- With only one vector, it will be used as *y* vector with *x*=*1*:length(*y*)
  - Try plot(y), what do you observe?



Plotting Multiple Curves

- Pass multiple pairs of *x* and *y* into plot(…)
- Each curve has a different color

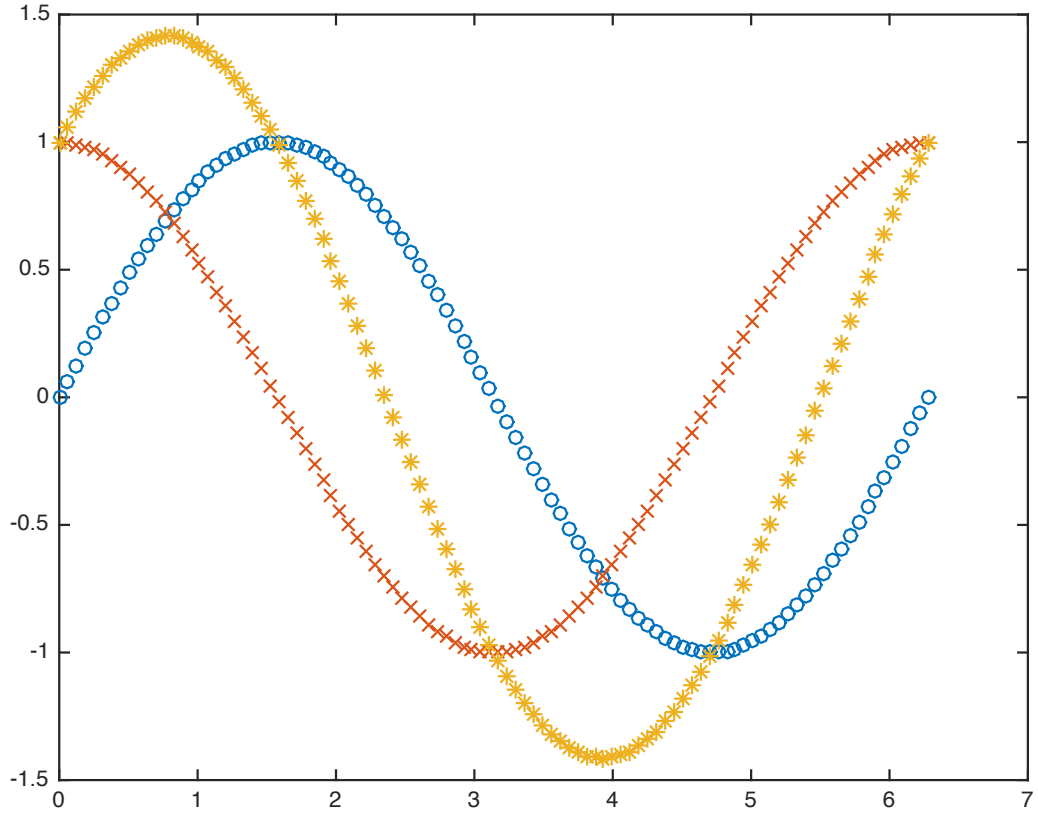
```
x = linspace(0, 2*pi);

plot(x, sin(x), x, cos(x), x, sin(x)+cos(x));
```



Markers

- Add a string to indicate which marker to use

```
x = linspace(0, 2*pi);
plot(x, sin(x), 'o',
x, cos(x), 'x',
x, sin(x)+cos(x), '*');
```

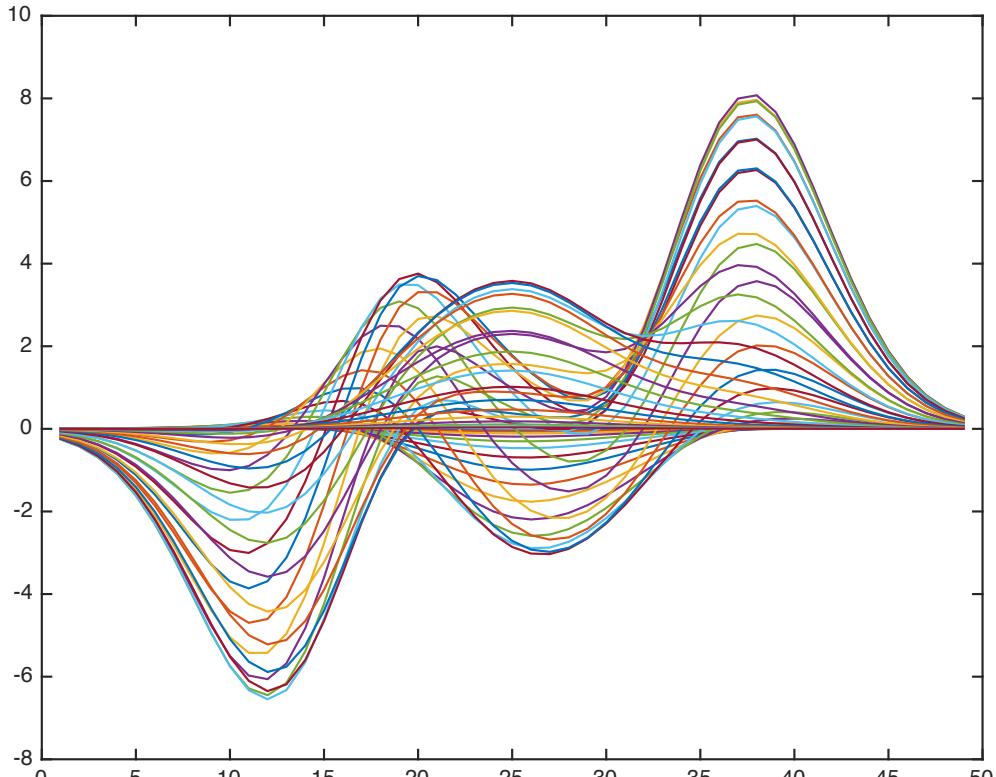


- Use help plot to see all possible markers

Plot a Two-Dimensional Matrix

- Plot commands generate a curve for each column vector of a input matrix

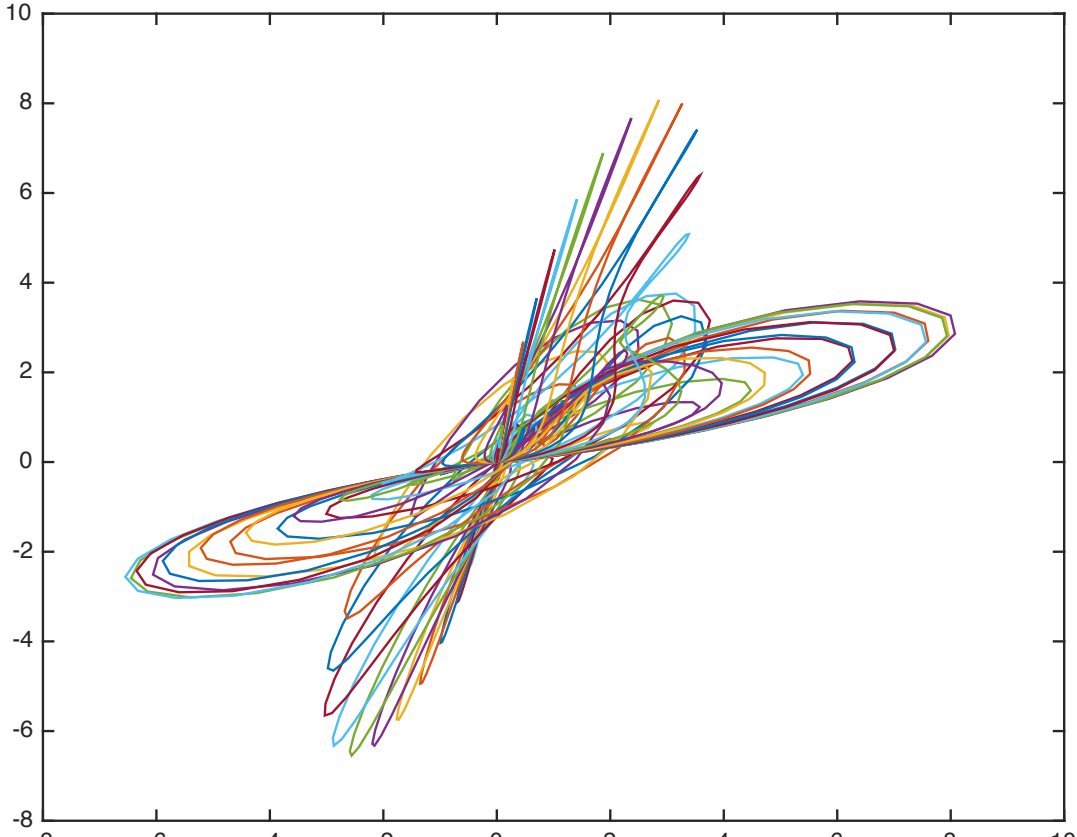
```
y = peaks;  % 49x49 matrix
              of Gaussian
              distributions
plot(y);    % 49 curves
```



Plotting Two 2-D Matrices

- Passing two matrices, plot will create a curve for each pair of column vectors

```
x = peaks;
y = x';
plot(x,y);
```



Conventions in Matlab

- Matlab treats each 2-D matrix as a subset of column vectors
- Passing a 2-D matrix to a function, such as max, min, and mean), the function iteratively processes each column vector

```
>> mean(magic(5))
ans =
    13    13    13    13    13
```

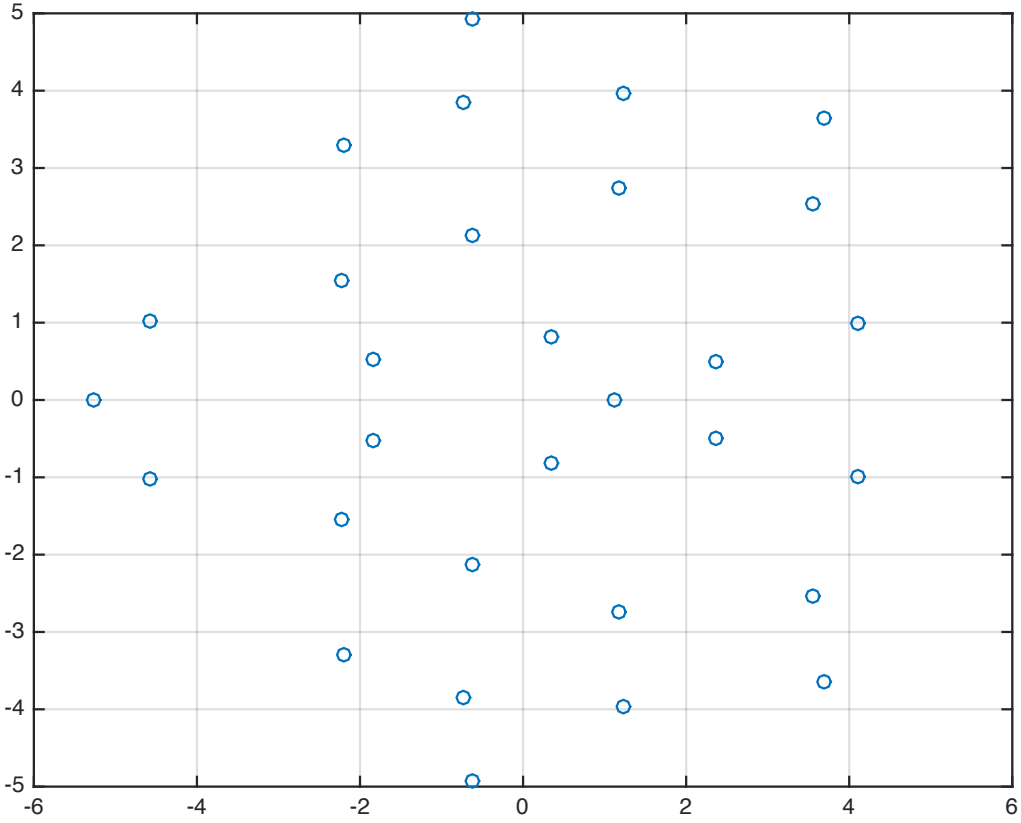
Plotting Complex Numbers

- For a vector of complex numbers, plot (z) is the same as plot(real(z), imag(z))

```
x = randn(30); % 30×30 random numbers (Gaussian)
z = eig(x);    % calculate eigenvalues
plot(z, 'o')
grid on        % add grids
```

Plotting Complex Numbers (cont.)

- For real number eigenvalues, their y values are zero
- For complex number eigenvalues, they always appear in pairs (conjugate), mirrored at x-axis

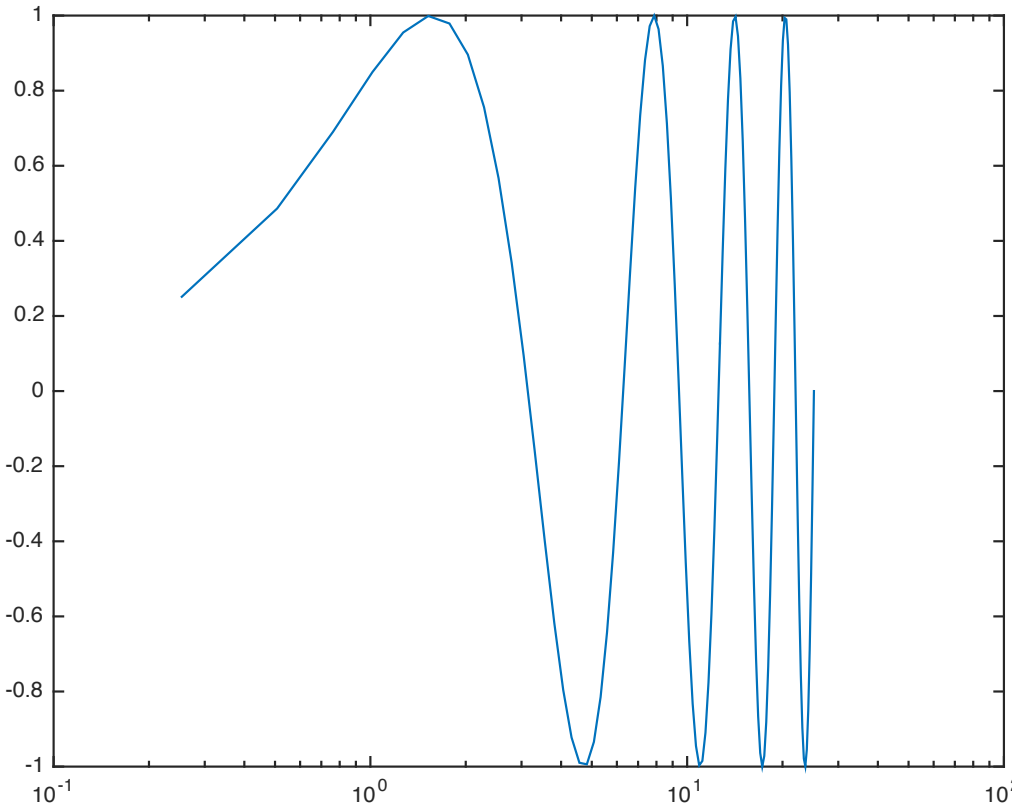


More 2D Plot Commands

- plot: both x and y axes are linear
- loglog: both x and y axes are logarithmic
- semilogx: x axis is logarithmic, while y axis is linear
- semilogy: y axis is logarithmic, while x axis is linear
- plotyy: create two y axes (left and right)

More 2D Plotting Examples

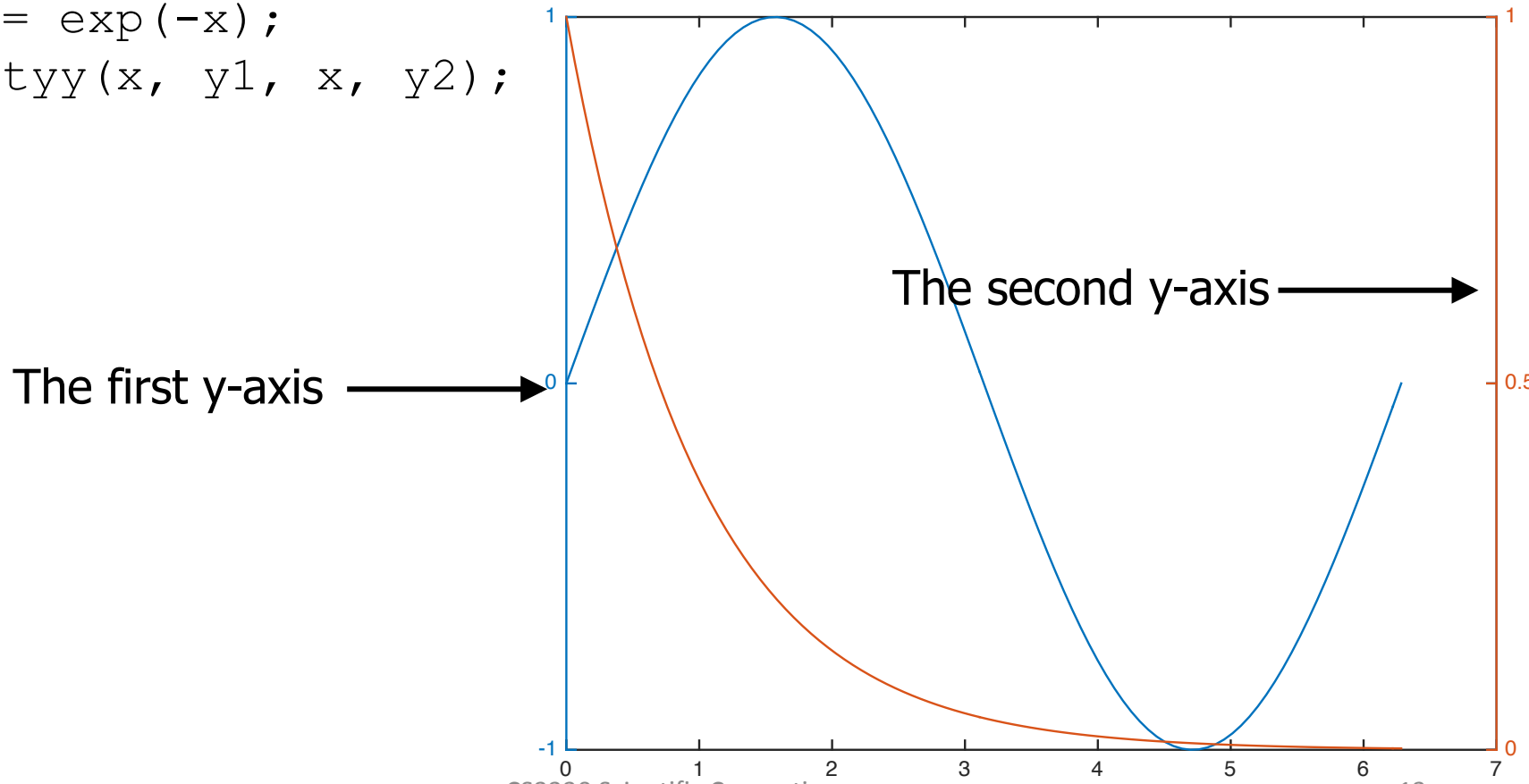
```
x = linspace(0, 8*pi);
semilogx(x, sin(x));
```



X axis is in logarithmic scale

More 2D Plotting Examples (cont.)

```
x = linspace(0, 2*pi);
y1 = sin(x);
y2 = exp(-x);
plotyy(x, y1, x, y2);
```



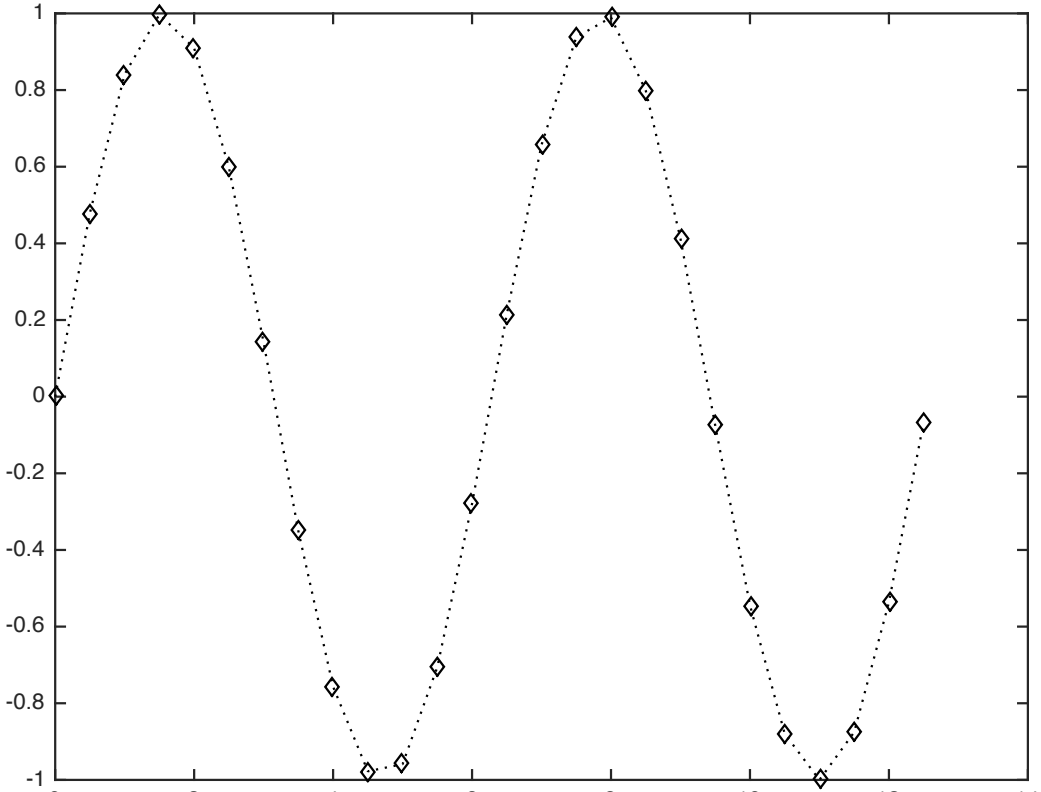
Control the Curves

- Recall that we assign markers to curves using a string
- Syntax: plot(x, y, 'CLM')
  - C: color of the curve ← red, blue, and so on
  - L: line styles ← solid, dash, dots, and etc.
  - M: marker ← circle, asterisk, diamond, and so on

Control the Curves (cont.)

```
x = 0:0.5:4*pi;
y = sin(x);
plot(x, y, 'k:d');
```

- What do they mean?
  - k
  - :
  - d
- Their order is not important
- The complete lists can be found via help

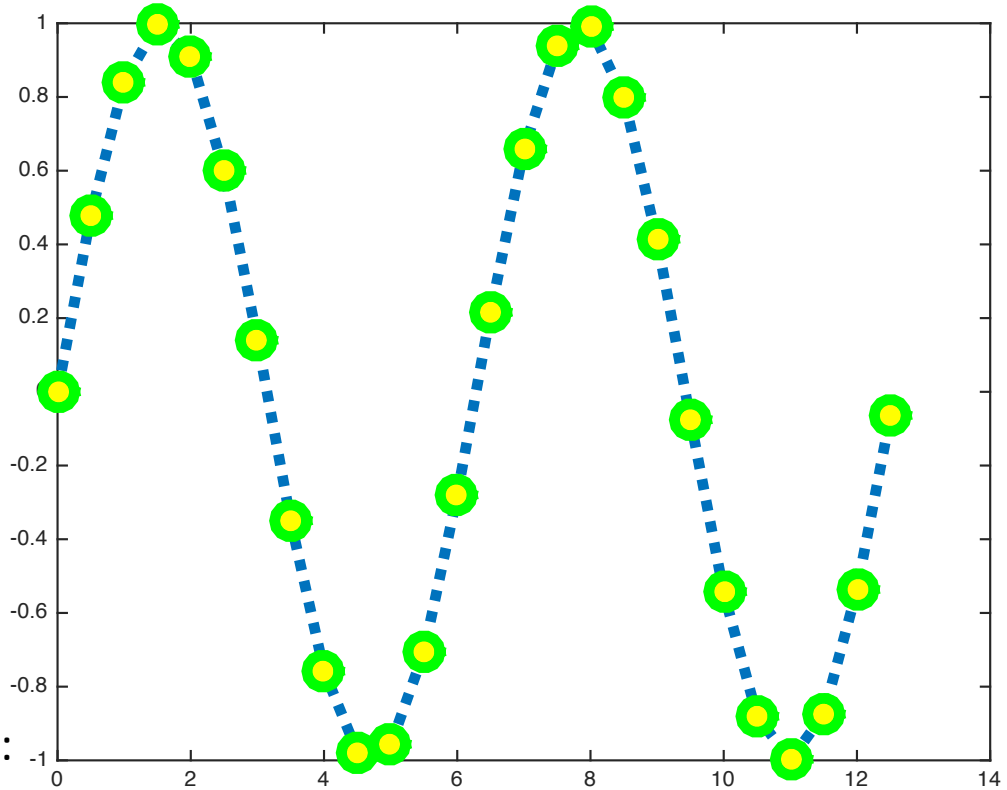


Graphic Handle

- Each curve can be seen as an object
- We use graph handle to change the curve's appearance

```
x=0:0.5:4*pi;
h=plot(x, sin(x));              % Plot a sin curve
set(h, 'marker', 'o');          % Set marker to 'o'
set(h, 'markerSize', 15);       % Set marker size to 15
set(h, 'lineWidth', 5);         % Set line width to 5
set(h, 'lineStyle', ':');       % Set line style to dot
set(h, 'markerEdgeColor', 'g'); % Set marker edge color to green
set(h, 'markerFaceColor', 'y'); % Set marker face color to yellow
```

Graphic Handle (cont.)



Equivalently, you can use:

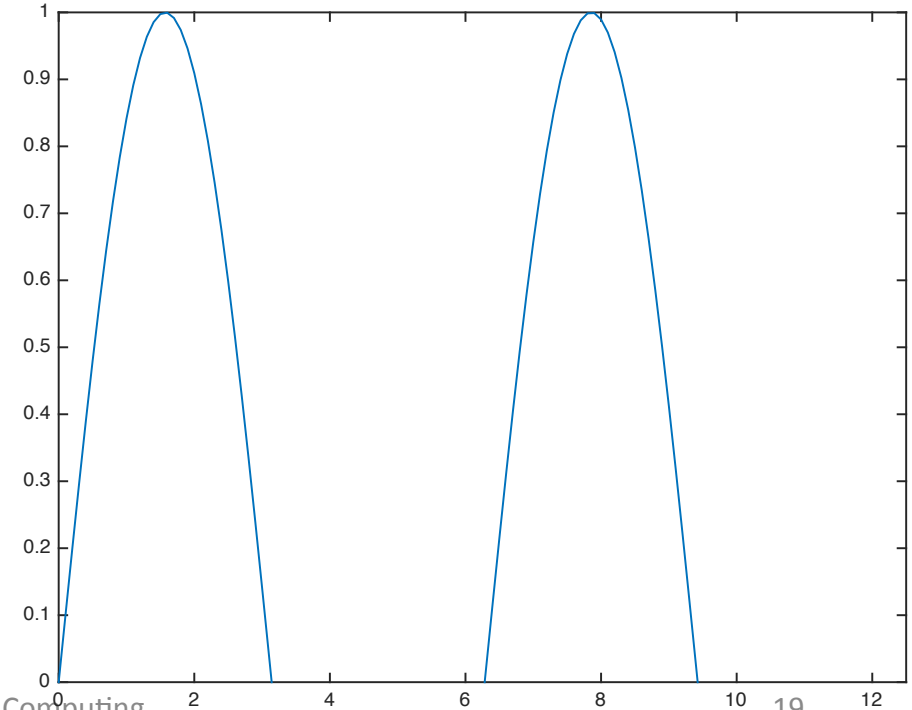
```
x=0:0.5:4*pi;
plot(x, sin(x), 'marker', 'o', 'markerSize', 15,
'lineWidth', 5, 'lineStyle', ':',
'markerEdgeColor', 'g', 'markerFaceColor', 'y');
```

Adjusting the Ranges of Axes

- Matlab assigns default xlim and ylim heuristically
- Or setting them manually
  - axis([xmin, xmax, ymin, ymax])
  - xlim([xmin, xmax])
  - ylim([ymin, ymax])

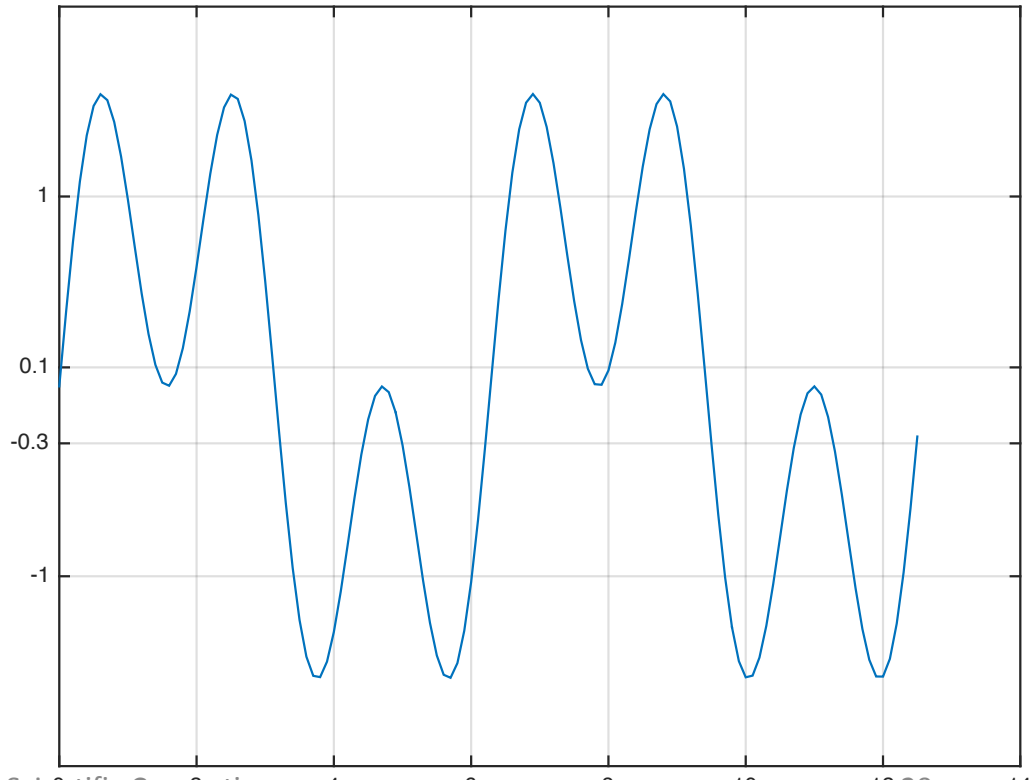
Adjusting the Ranges of Axes (cont.)

```
x = 0:0.1:4*pi;
y = sin(x);
plot(x, y);
axis([-inf, inf, 0, 1]);
% inf means the extreme
  values of samples
```



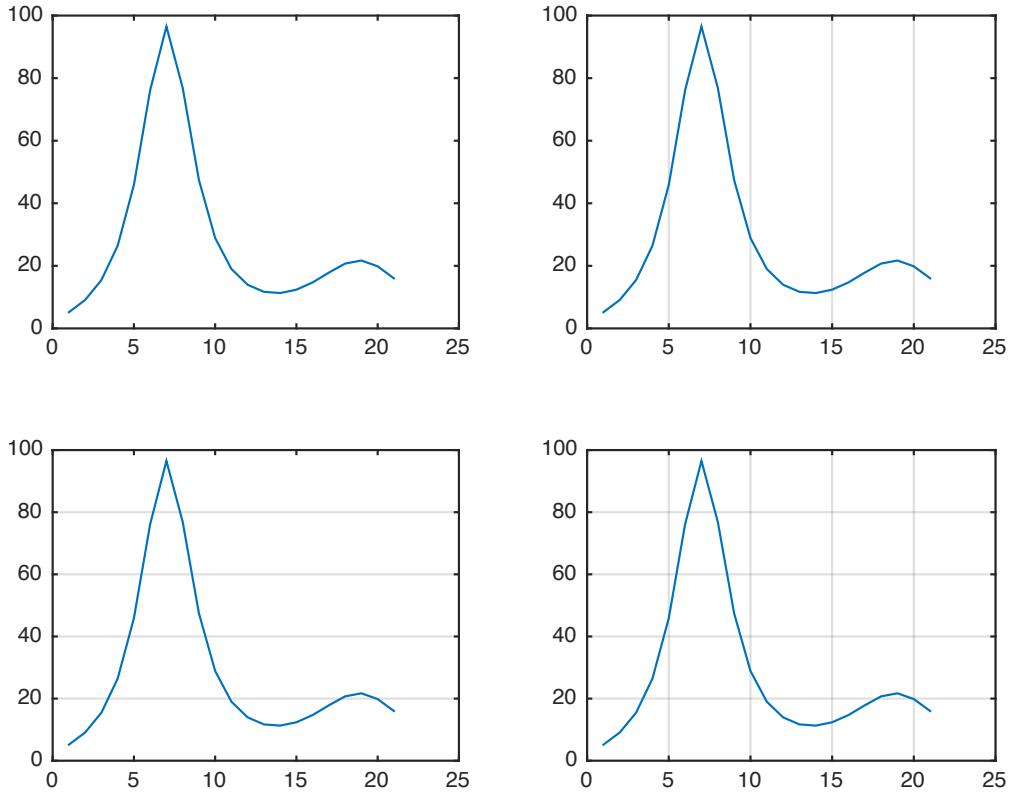
Adjusting Ticks and Grids

```
x = 0:0.1:4*pi;
plot(x, sin(x)+sin(3*x))
set(gca, 'ytick', [-1 -0.3 0.1 1]);
grid on;
% gca returns the
  current axes,
  as an object
```



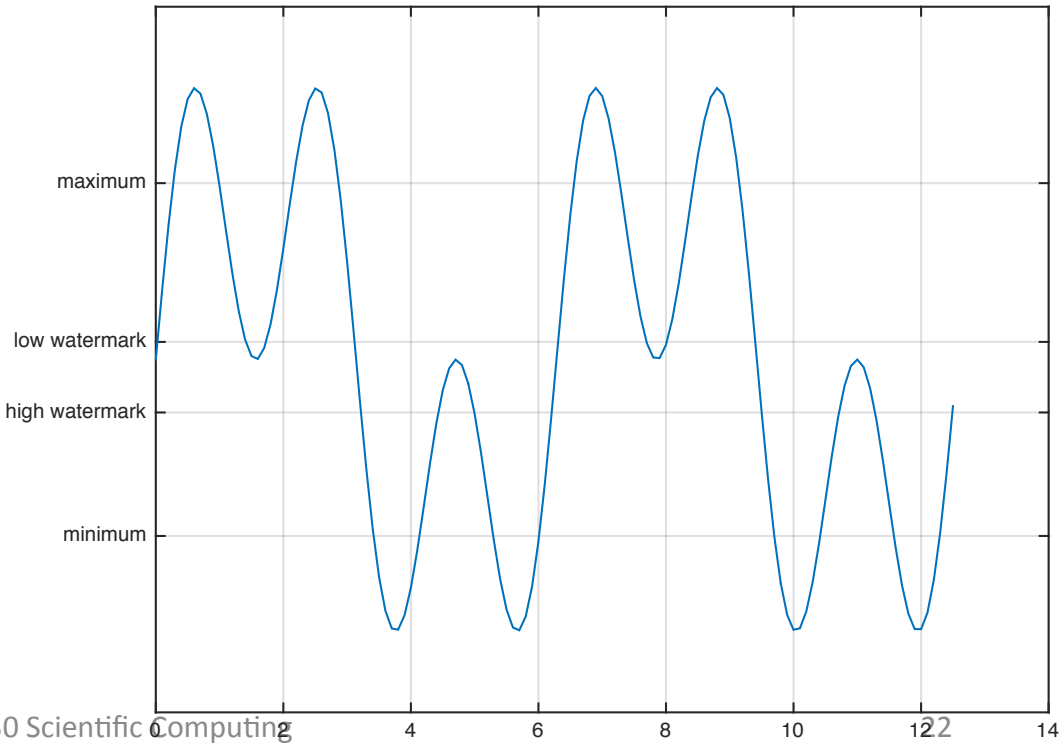
Subplots

```
subplot(221); plot(humps);
subplot(222); plot(humps);
set(gca, 'xgrid', 'on');
subplot(223); plot(humps);
set(gca, 'ygrid', 'on');
subplot(224); plot(humps);
grid on;
```



Adjusting Tick Labels

```
x = 0:0.1:4*pi;
plot(x, sin(x)+sin(3*x))
set(gca, 'ytick', [-1 -0.3 0.1 1]);
set(gca, 'yticklabel', {'minimum', 'high watermark',
'low watermark', 'maximum'});
grid on
```

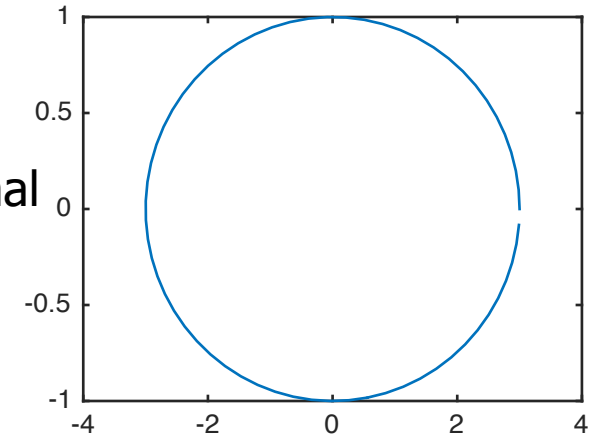


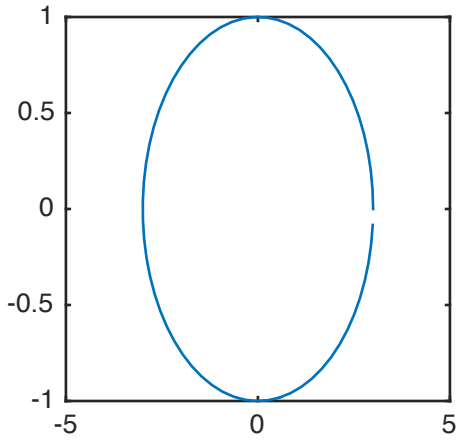
Aspect Ratios of Plots

```
t = 0:0.1:2*pi;
x = 3*cos(t);
y = sin(t);
subplot(2, 2, 1); plot(x, y); axis normal
subplot(2, 2, 2); plot(x, y); axis square
subplot(2, 2, 3); plot(x, y); axis equal
subplot(2, 2, 4); plot(x, y); axis equal tight

% axis normal ← same ratio as figure
% axis square ← ratio = 1
% axis equal  ← same length of unit data on both axes
% axis equal tight ← compact axes
% and others….check help
```

Aspect Ratios of Plots (cont.)

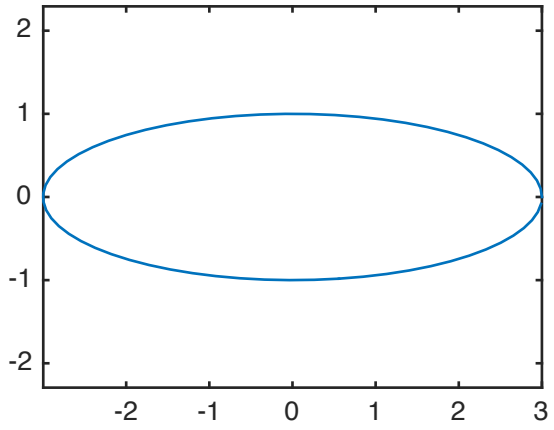
axis normal

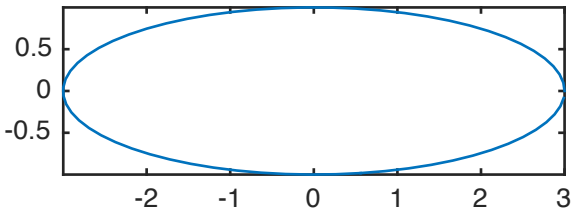




axis square

axis equal





axis equal tight

Additional Controls

- colordef white ← white axes' background
- colordef black
- grid on
- grid off
- box on ← bounding box of the axes
- box off

Commands to Add Descriptive Texts

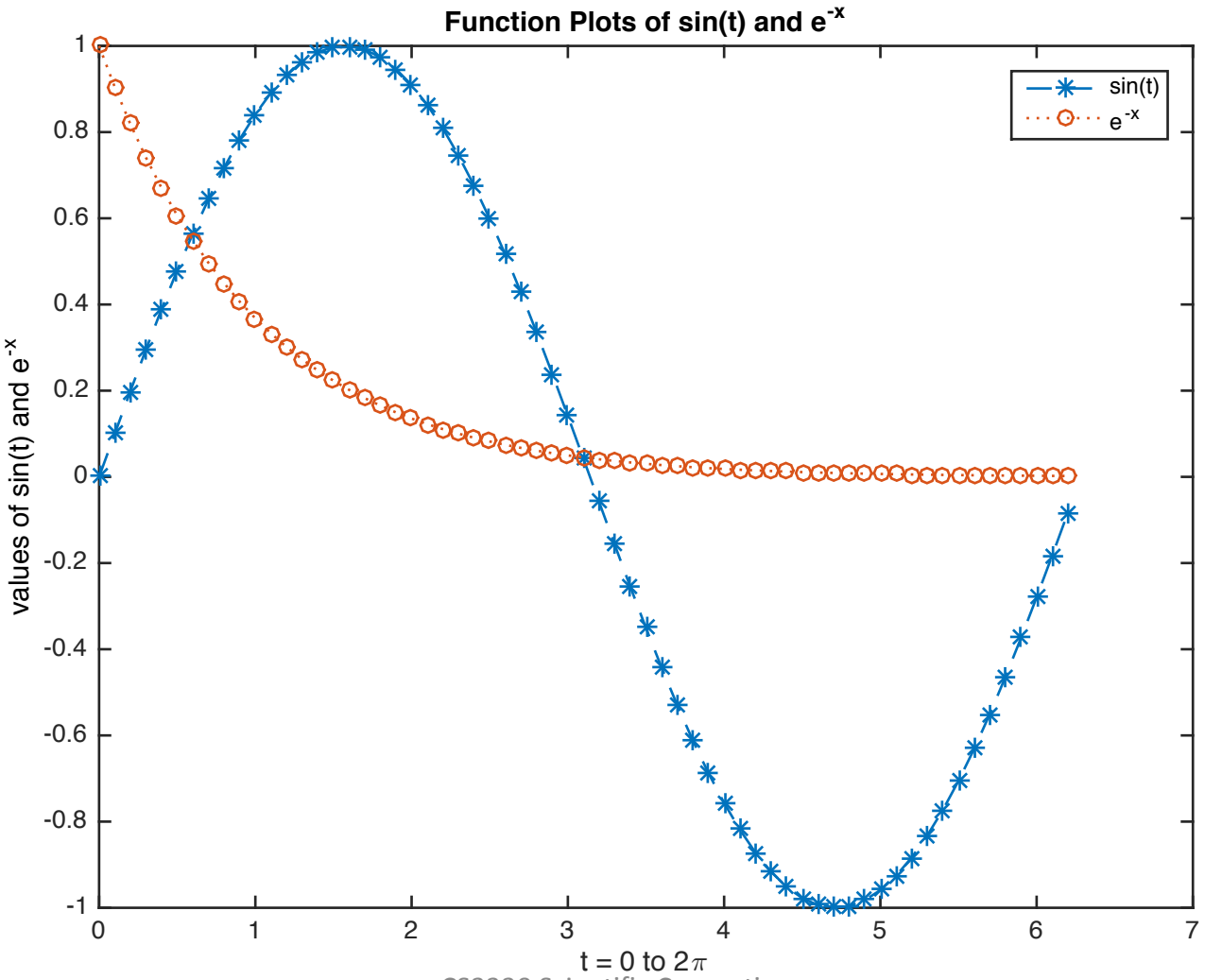
- title: figure title
- xlabel: x-axis text
- ylabel: y-axis text
- zlabel: z-axis text
- legend: descriptions of individual curves
- text: annotations
- gtext: annotations placed by mouse clicks

Example of Texts

```
subplot(1,1,1);
x = 0:0.1:2*pi;
y1 = sin(x);
y2 = exp(-x);
plot(x, y1, '--*', x, y2, ':o');
xlabel('t = 0 to 2\pi');
ylabel('values of sin(t) and e^{-x}')
title('Function Plots of sin(t) and e^{-x}');
legend('sin(t)','e^{-x}');
```

- Legend
- Latex syntax
- Three interpreters

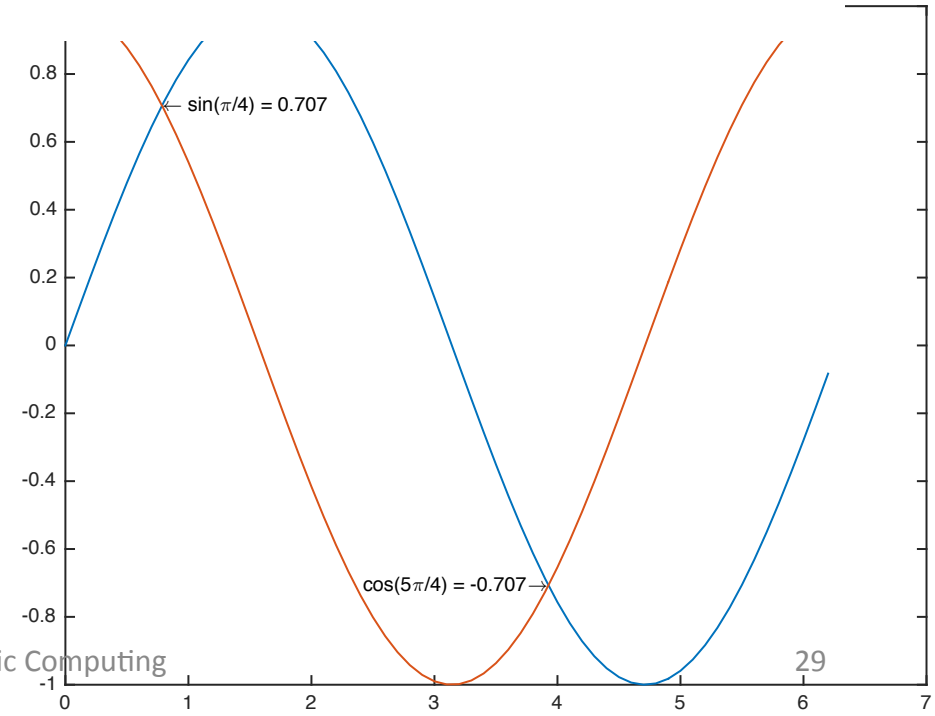
Example of Texts (cont.)



Example of Annotations

- text(x, y, string), option: horizontal alignment

```
x = 0:0.1:2*pi;
plot(x, sin(x), x, cos(x));
text(pi/4, sin(pi/4), '\leftarrow sin(\pi/4) = 0.707');
text(5*pi/4, cos(5*pi/4), 'cos(5\pi/4) =
-0.707\rightarrow', 'HorizontalAlignment', 'right');
```



Example of Annotations (cont.)

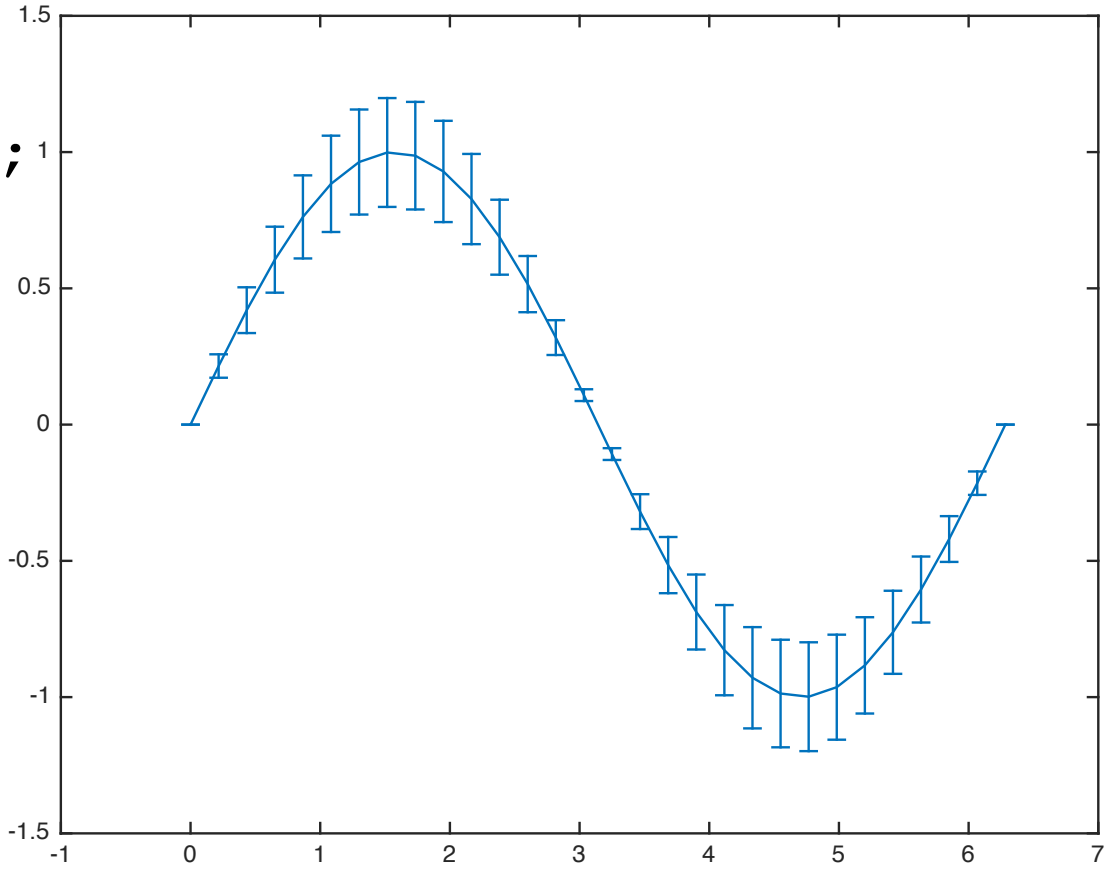
- gtext(string) ← try it on your computer
- After issuing this command, use mouse click to place the string
- Only works in 2-D graphs

Some Other 2-D Graph Commands

- errorbar: curves with error intervals
- fplot, ezplot: plotting with adaptive x sampling
- polar, ezpolar: plotting with polar coordinates
- hist: histogram
- rose: angle histogram

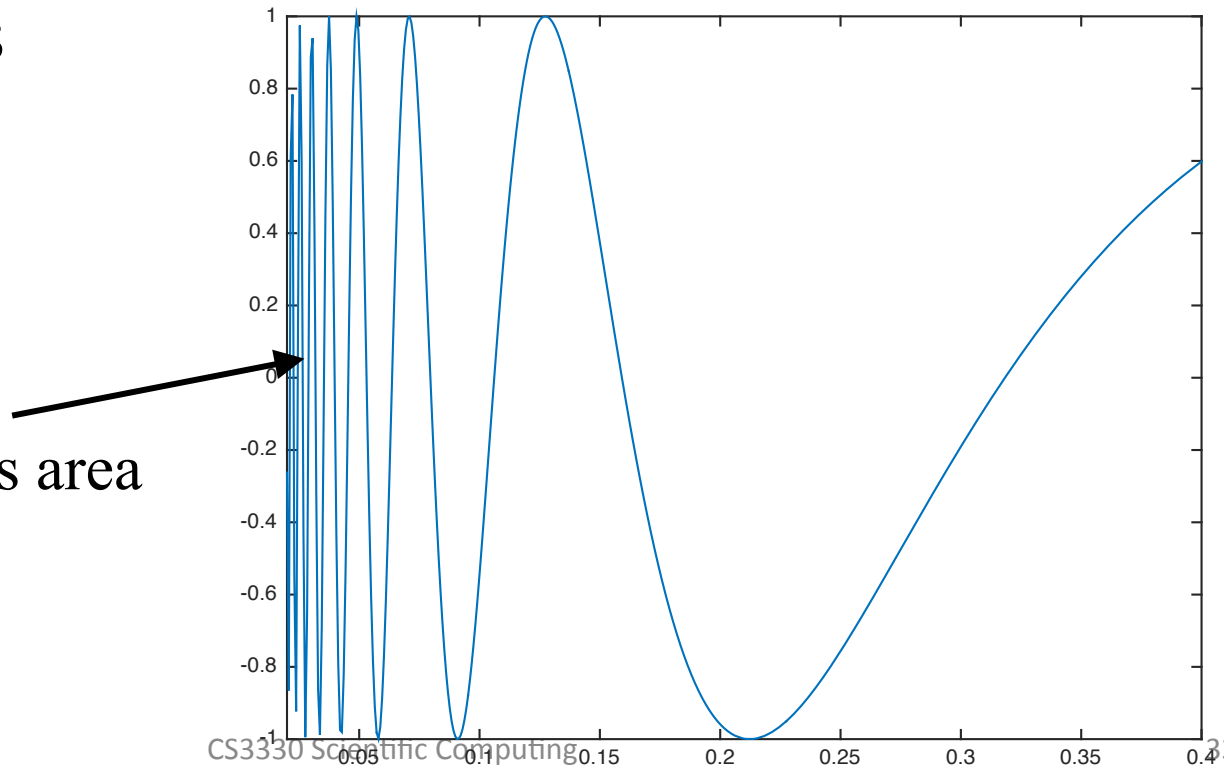
Example of Errorbar

```
x=linspace(0, 2*pi, 30);
y=sin(x);
e=y*0.2;
errorbar(x,y,e);
```



Example of fplot

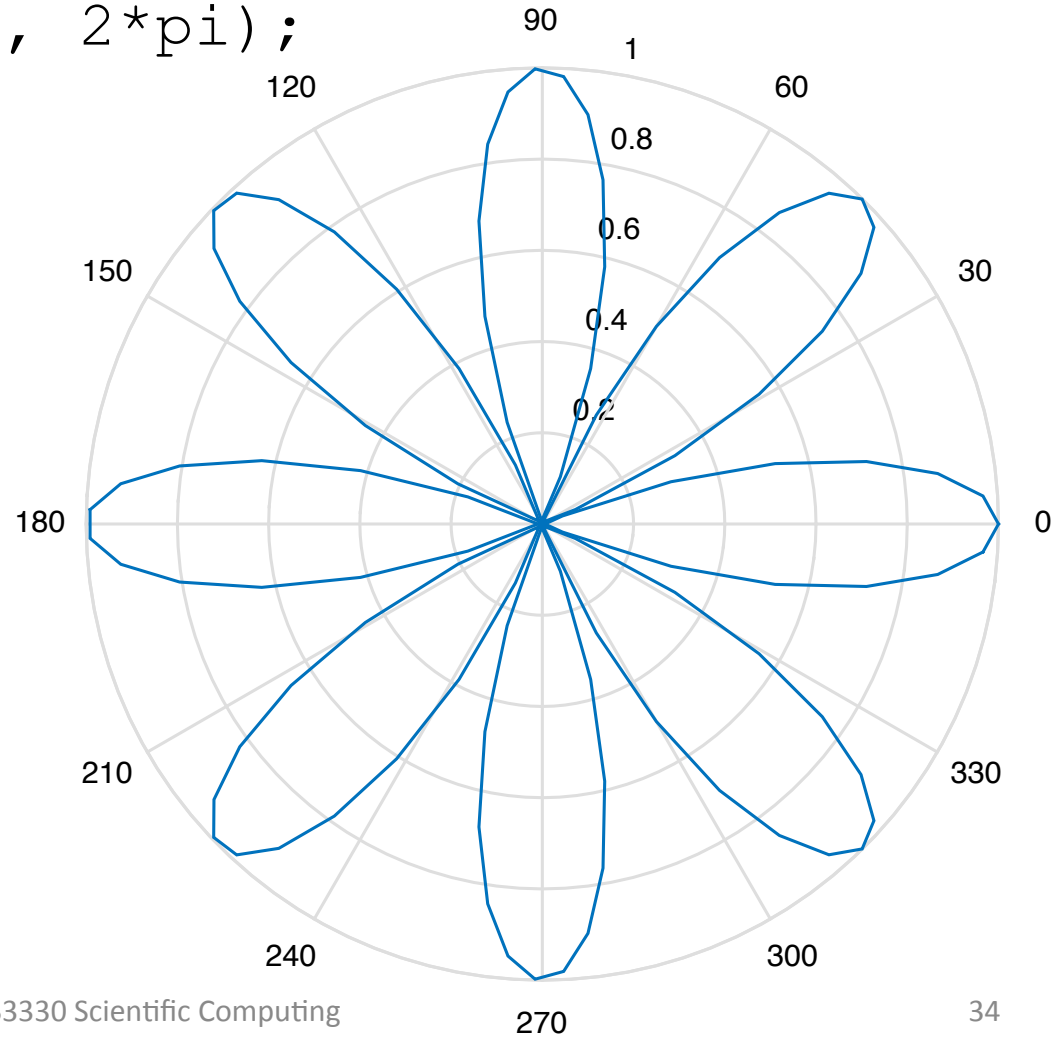
- Problem with `x=0.02:0.01:0.4; y=sin(1./x); plot(x,y)`?
- `fplot('sin(1/x)', [0.02, 0.4])` ← adaptively pick samples



More samples in this area

Example of polar

```
theta = linspace(0, 2*pi);
r = cos(4*theta);
polar(theta, r);
```



Polar versus Cartesian Coordinates

$$(\theta, r) \Rightarrow (x, y) \rightarrow \begin{cases} x = r\cos\theta \\ y = r\sin\theta \end{cases}$$

$$(x, y) \Rightarrow (\theta, r) \rightarrow \begin{cases} r = \sqrt{x^2 + y^2} \\ \theta = \tan^{-1}\dfrac{y}{x} \end{cases}$$

$$z = x + yi$$
$$= r\cos\theta + r\sin\theta i$$
$$= r(\cos\theta + \sin\theta i) = re^{\theta i}$$

$$\Rightarrow plot(x, y) \equiv plot(z) \equiv plot(re^{\theta i})$$

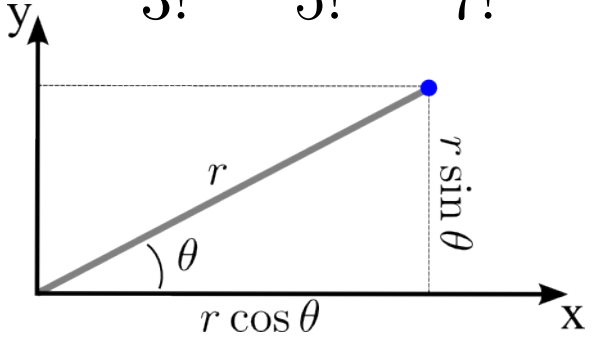
- How to prove $e^{i\theta} = \cos\theta + i\sin\theta$?
- Power series expansion

$$e^z = 1 + \frac{z}{1!} + \frac{z^2}{2!} + \frac{z^3}{3!} + \cdots$$

- Maclaurin series

$$\cos\theta = 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \frac{\theta^6}{6!} + \cdots$$

$$\sin\theta = x - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} + \cdots$$



Animated Polar Plot

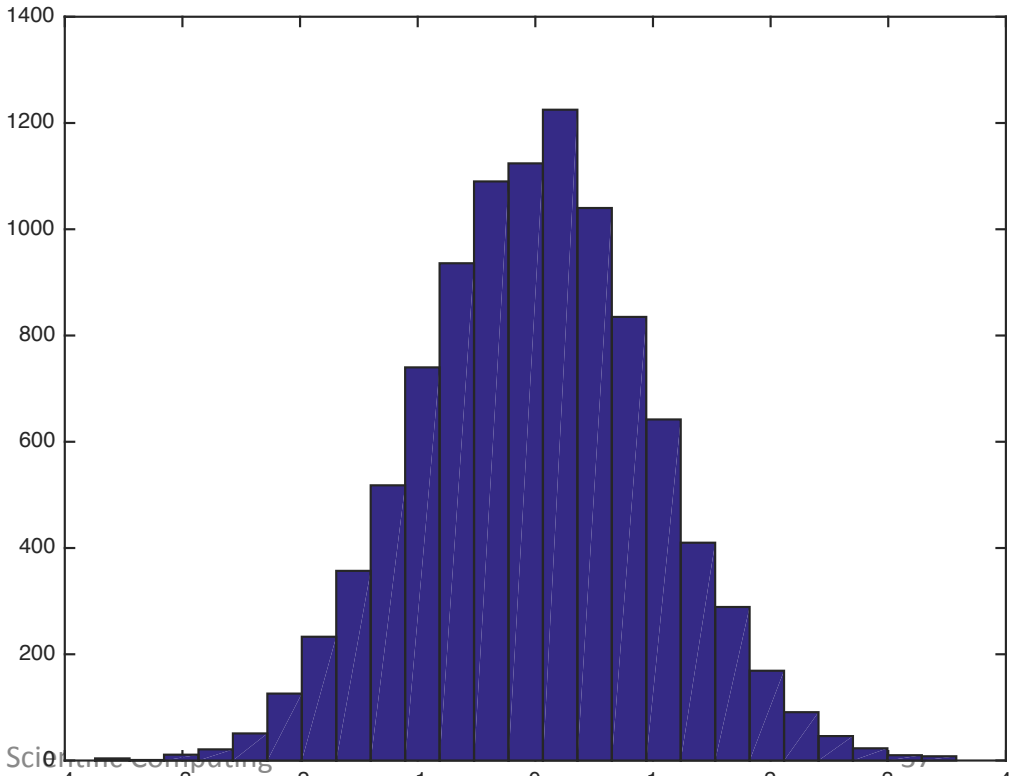
- Try this on your computer for a comet-like trajectory

```
theta = linspace(0, 4*pi, 10001);
r=cos(4*theta);
x=r.*cos(theta);
y=r.*sin(theta);
comet(x,y);
```

Example of Histogram

- hist(vector, number_of_bins)

```
x = randn(10000, 1);
hist(x, 25);
```

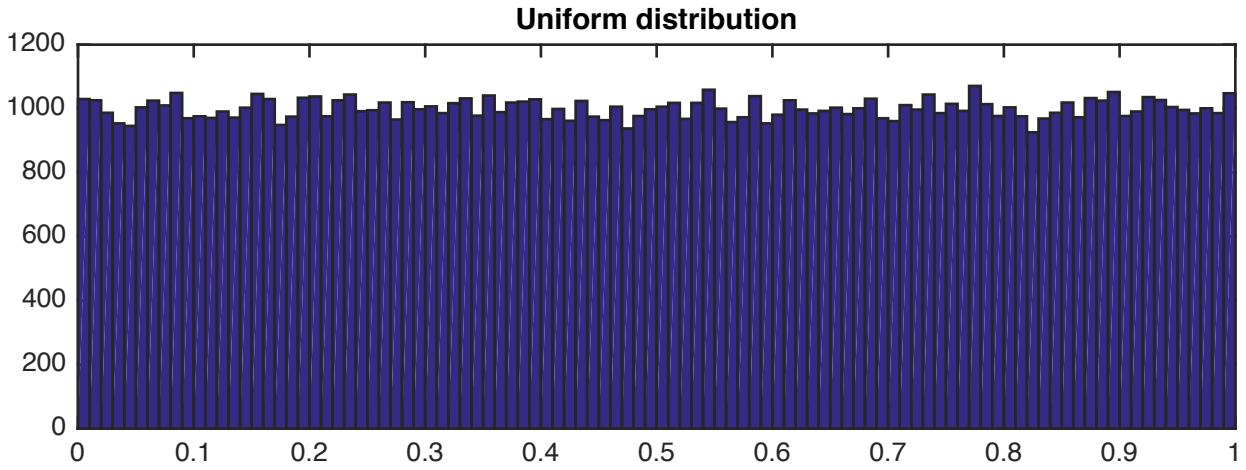


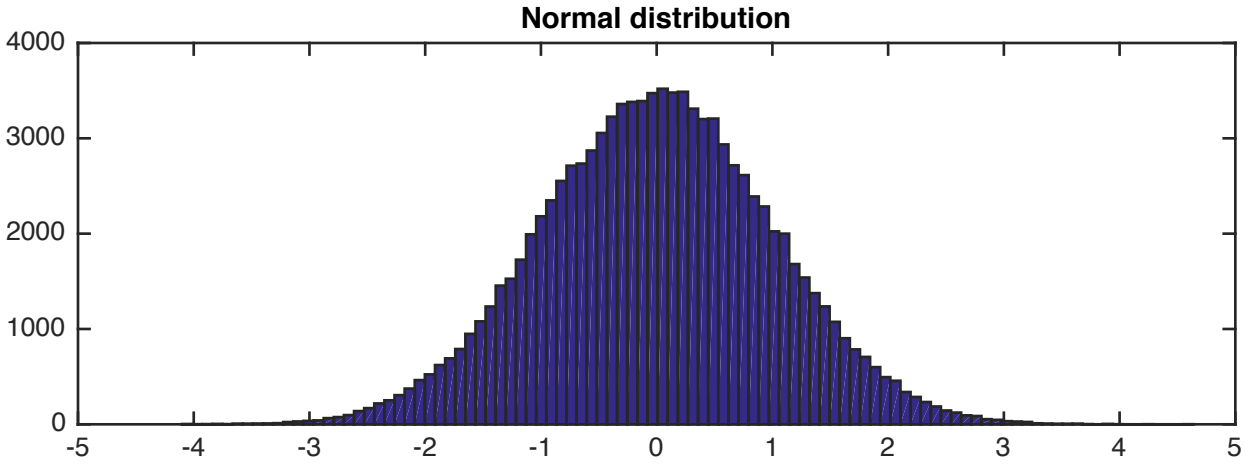
Histogram and Distribution Functions

- More samples, more bins make histogram closers to the original distribution functions

```
n=100000;
bin=1000;
subplot(211); hist( rand(n, 1), bin);
title('Uniform distribution');
subplot(212); hist(randn(n, 1), bin);
title('Normal distribution');
```

Histogram and Distribution Functions (cont.)

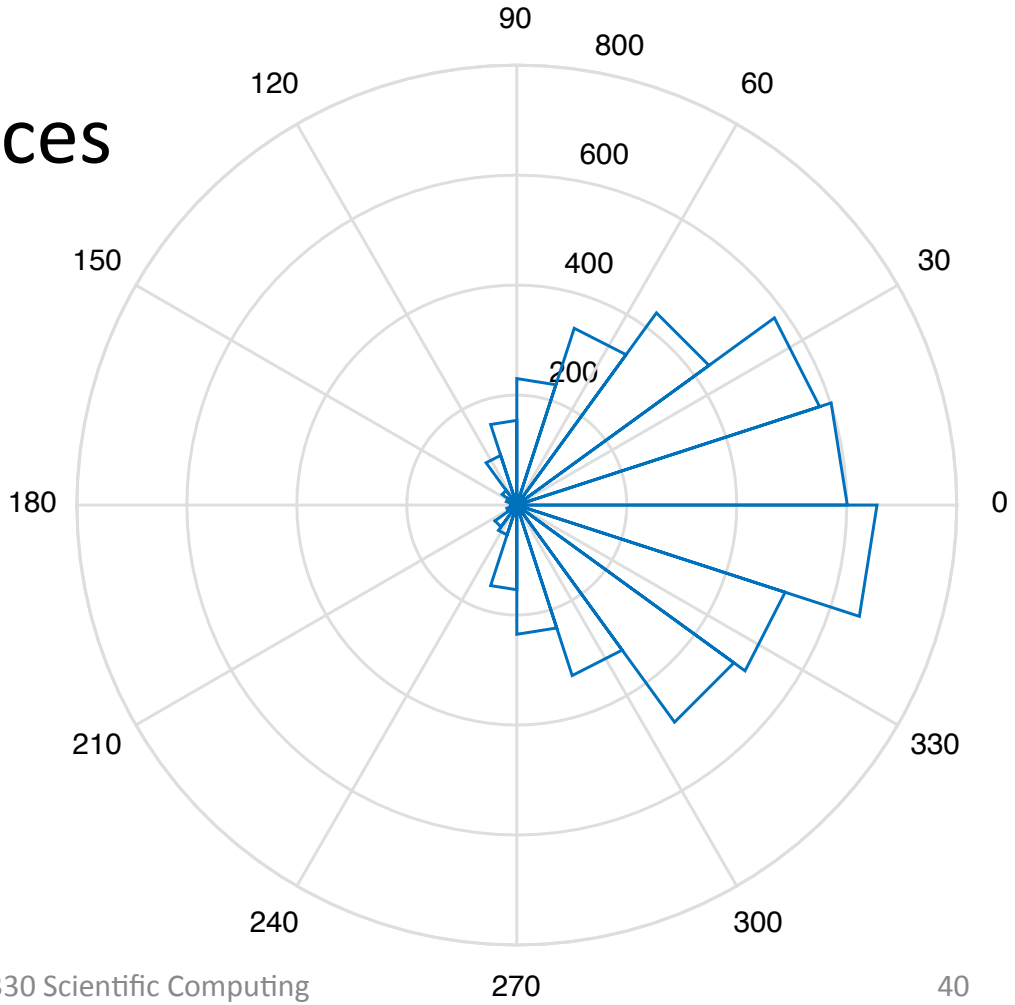




Example of Rose (Angle Histogram)

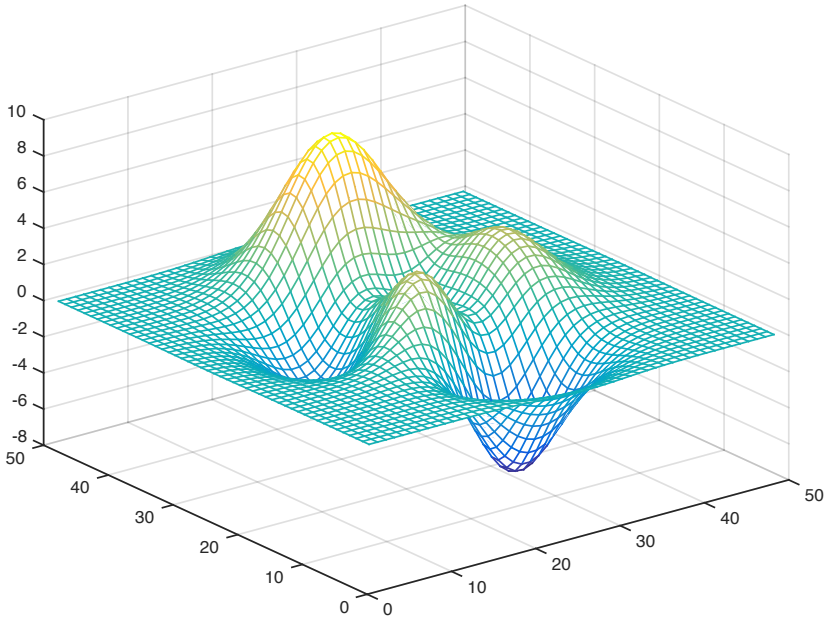
- Angle: value
- Distance: occurrences

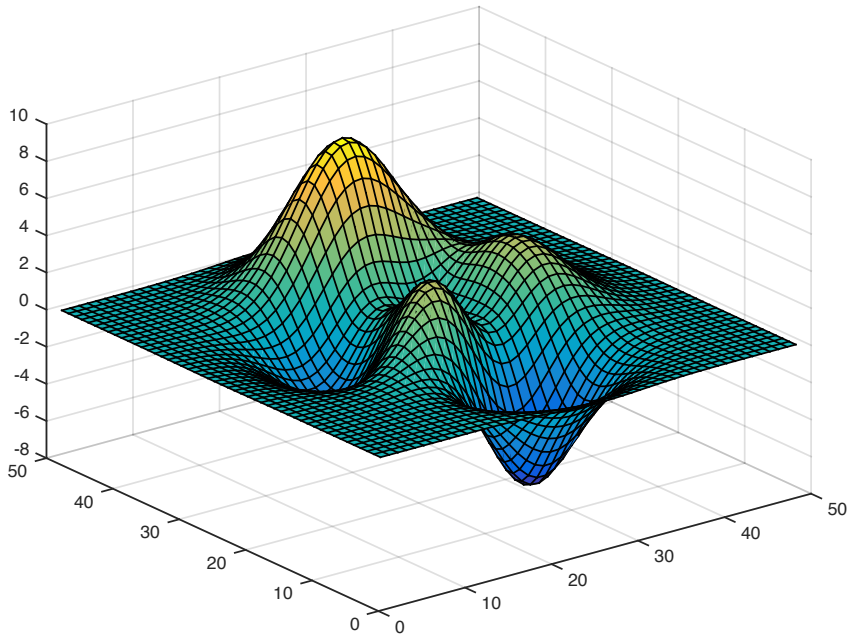
```
x = randn(5000, 1);
rose(x);
```



3-D Graph

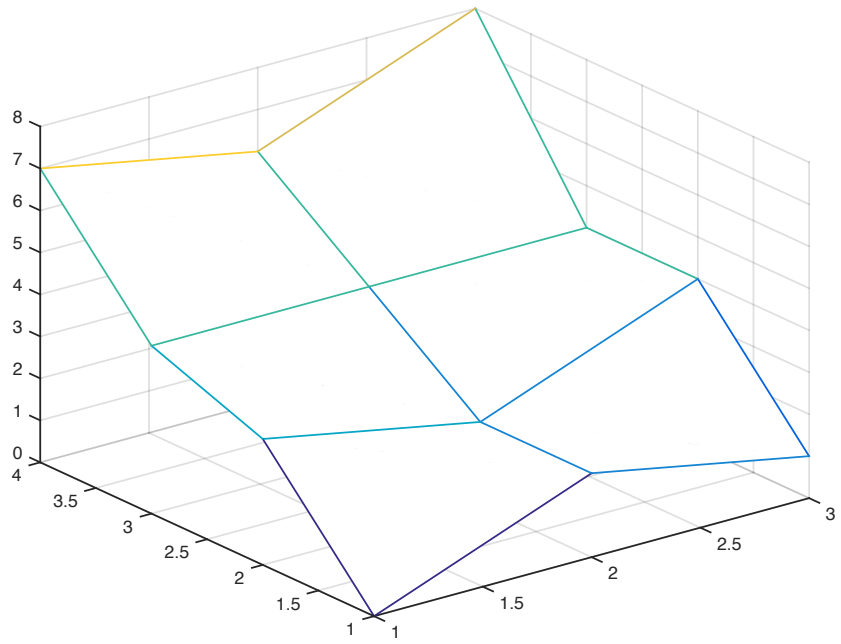
- Mesh: 3-D mesh graph
- Surface: 3-D surface graph

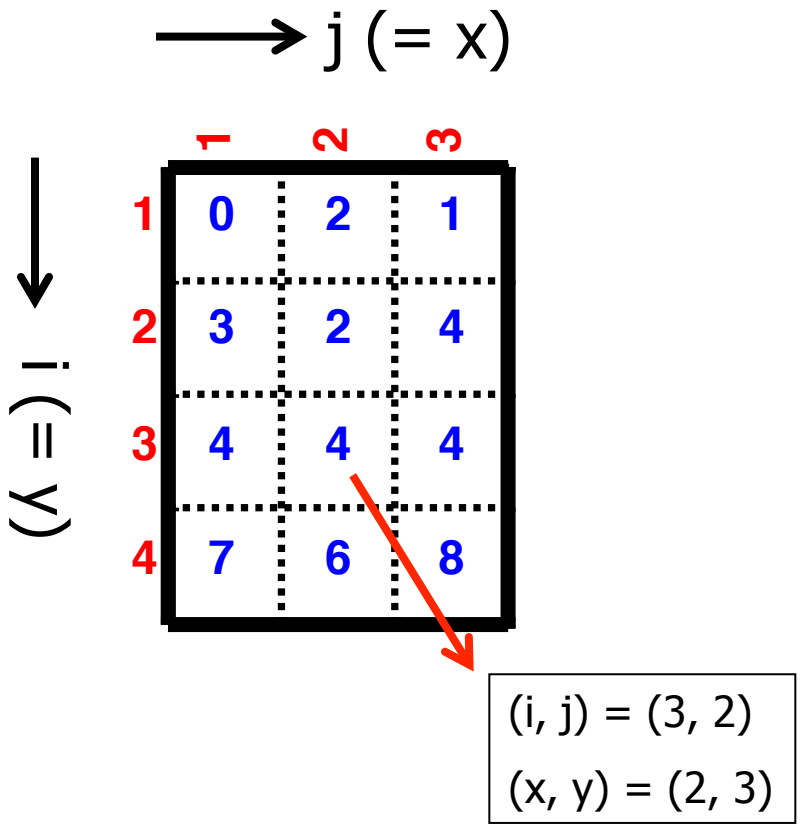




Mesh Example

```
z = [0 2 1; 3 2 4; 4 4 4; 7 6 8];
mesh(z);
```

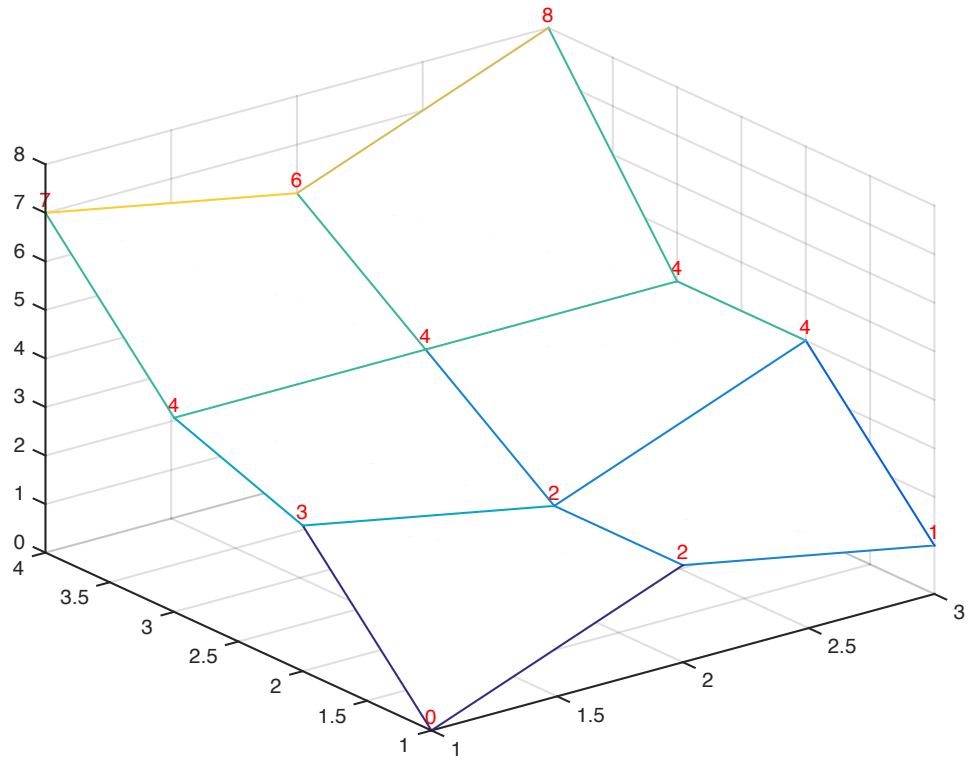




Mesh Example (cont.)

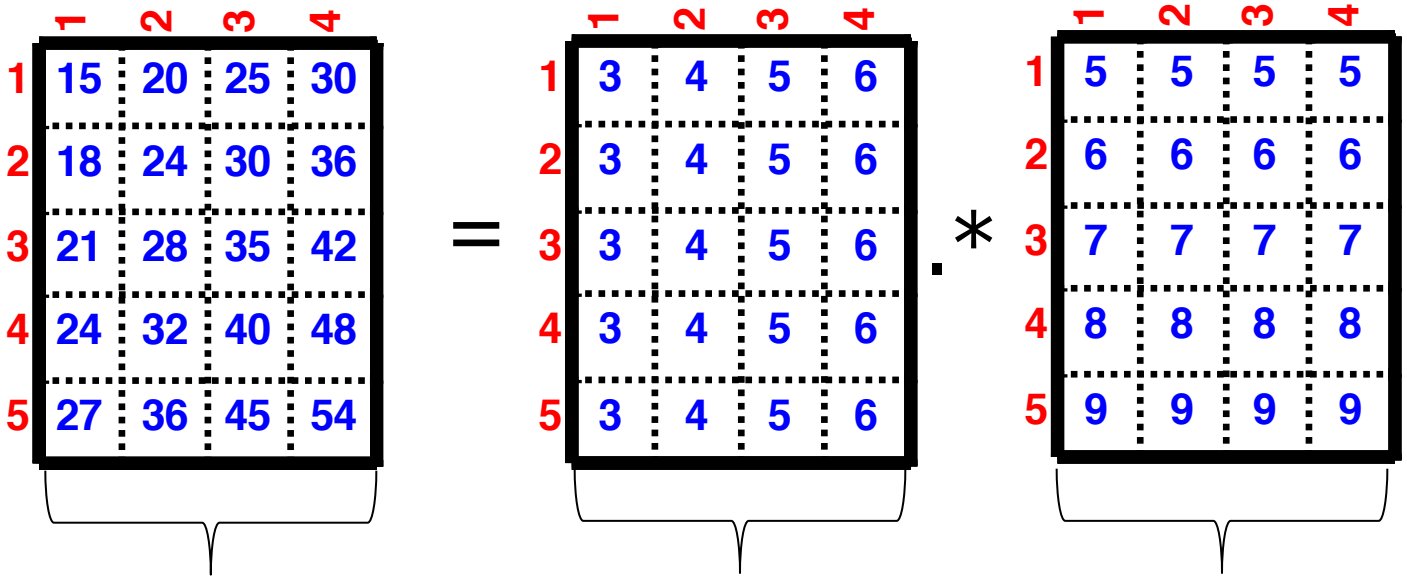
```
for i=1:size(z,1)
    for j=1:size(z,2)
        h=text(j, i, z(i,j), num2str(z(i, j)));
        set(h, 'hori', 'center', 'vertical', 'bottom',
'color', 'r');
    end
end
```

- Try, and observe the resulting y-axes
  - axis ij ← matrix axes
  - axis xy ← Cartesian axes

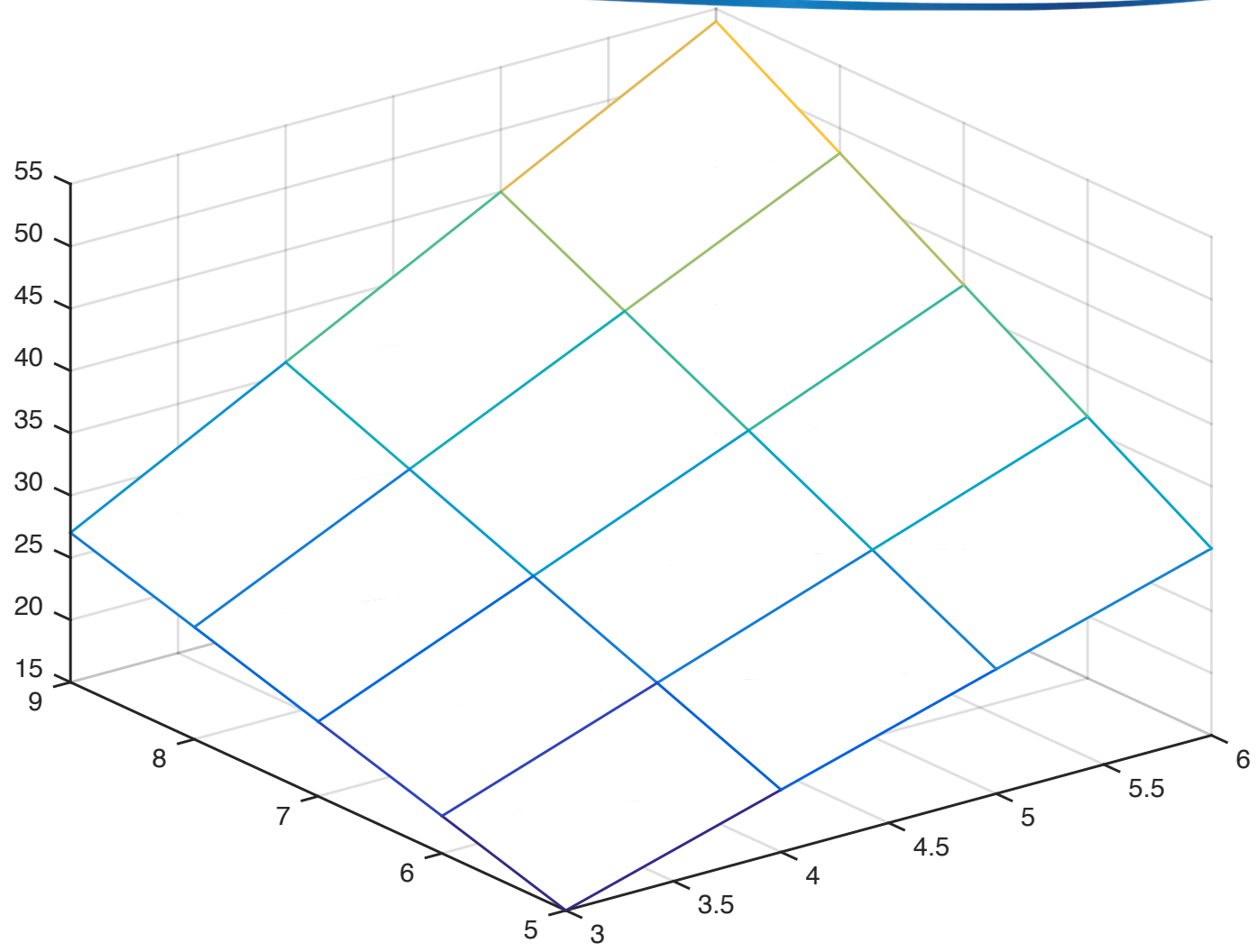


Meshgrid

```
x = 3:6;
y = 5:9;
[xx, yy] = meshgrid(x, y);
zz = xx.*yy;
mesh(xx, yy, zz);
```



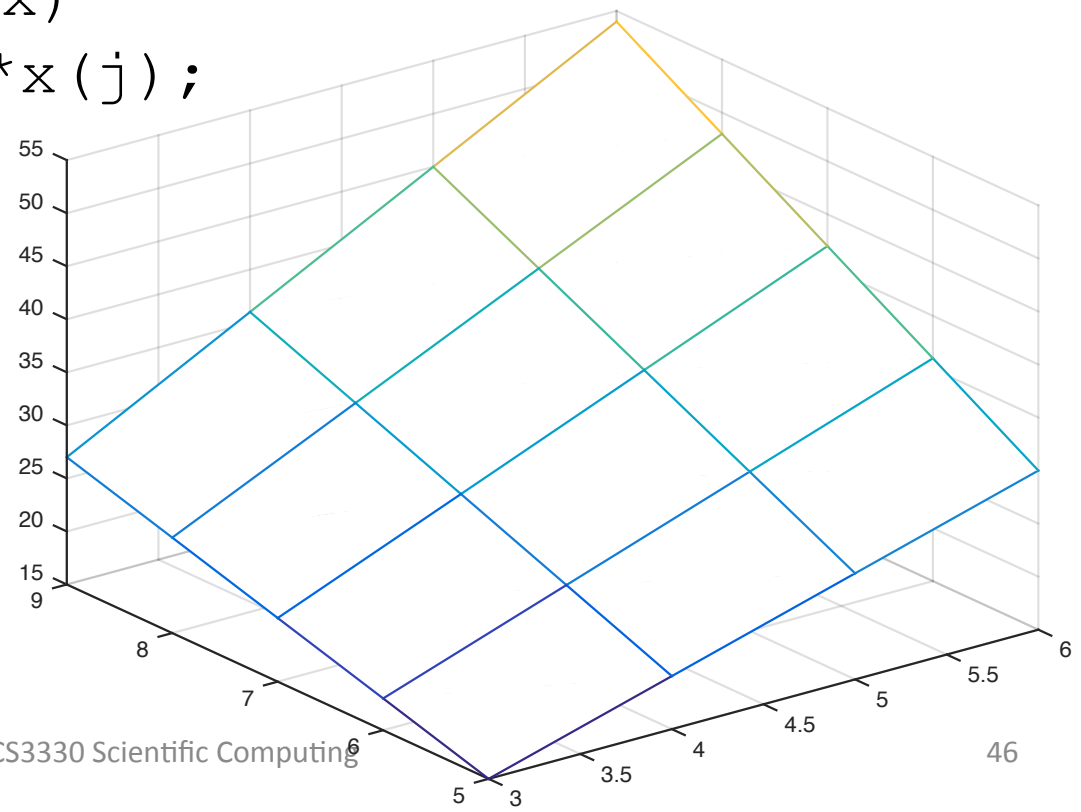
Meshgrid (cont.)



Achieving the Same Using For-Loops

```
x=3:6;
y=5:9;
for i=1:length(y)
   for j=1:length(x)
      zz(i,j)=y(i)*x(j);
   end
end
mesh(x, y, zz);
```

- Vector versus scalar versions



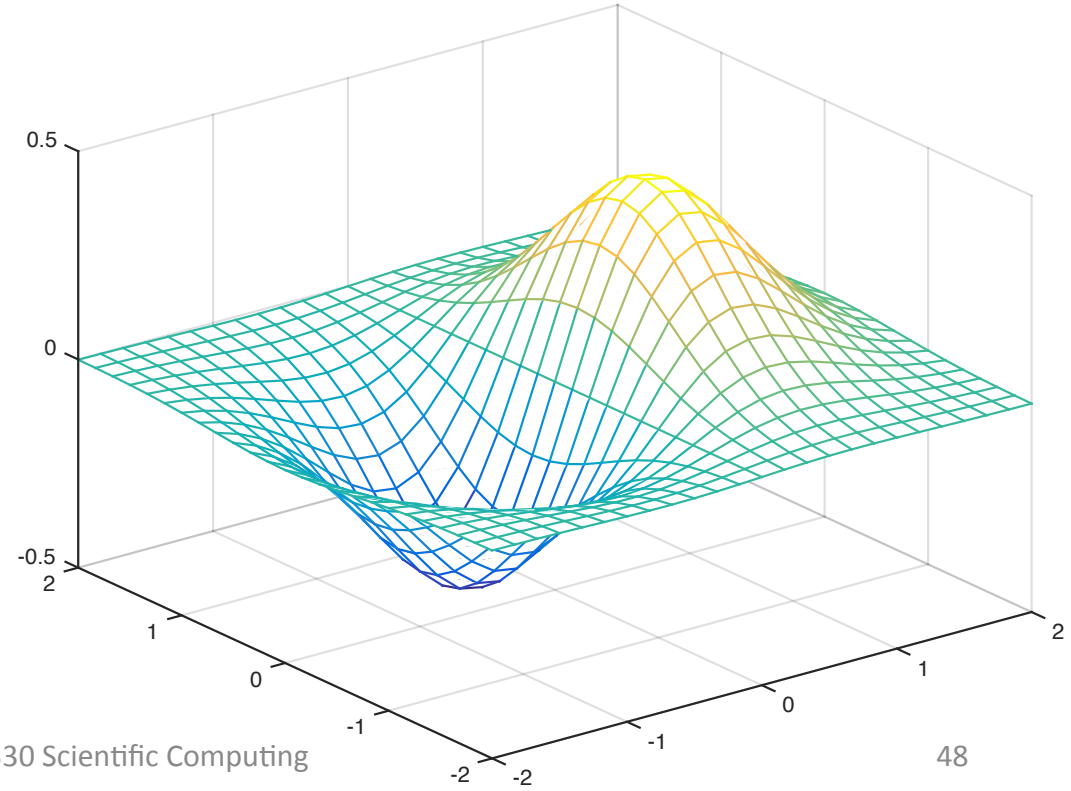
Speed of Vector and Scalar Versions

```
x=1:2000;
y=1:2000;
tic
    [xx, yy] = meshgrid( x, y );
    zz = xx.*yy;
toc
```

```
x=1:2000;
y=1:2000;
tic
for i=1:length(y)
   for j=1:length( x )
      zz( i,j )=y( i )*x( j );
   end
end
toc
```

Finer (More) Samples

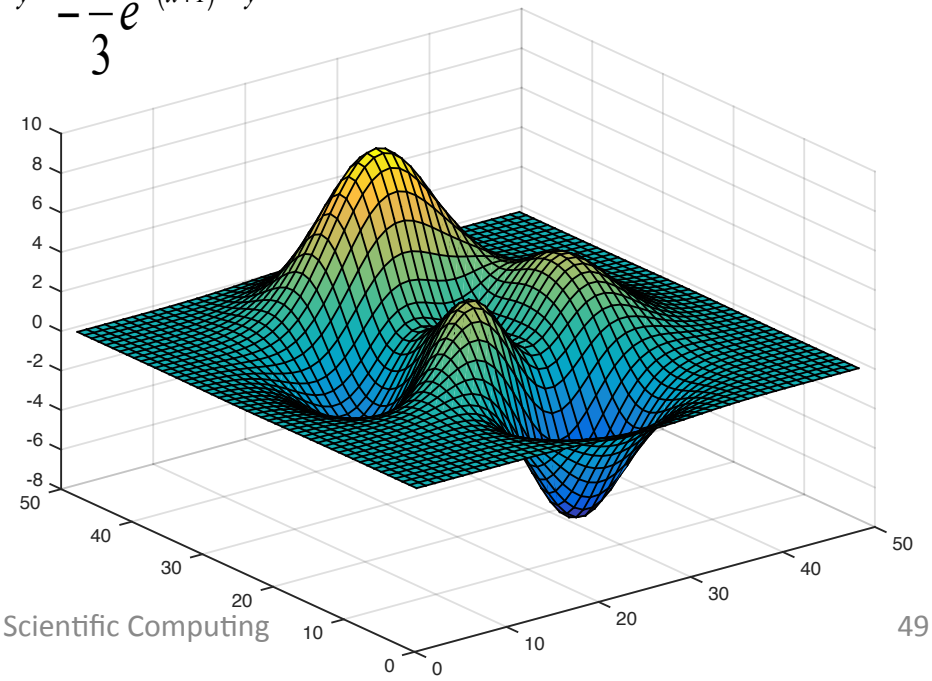
```
x = linspace(-2, 2, 25);
y = linspace(-2, 2, 25);
[xx, yy] = meshgrid(x, y);
zz = xx.*exp(-xx.^2-yy.^2);
mesh(xx, yy, zz);
```



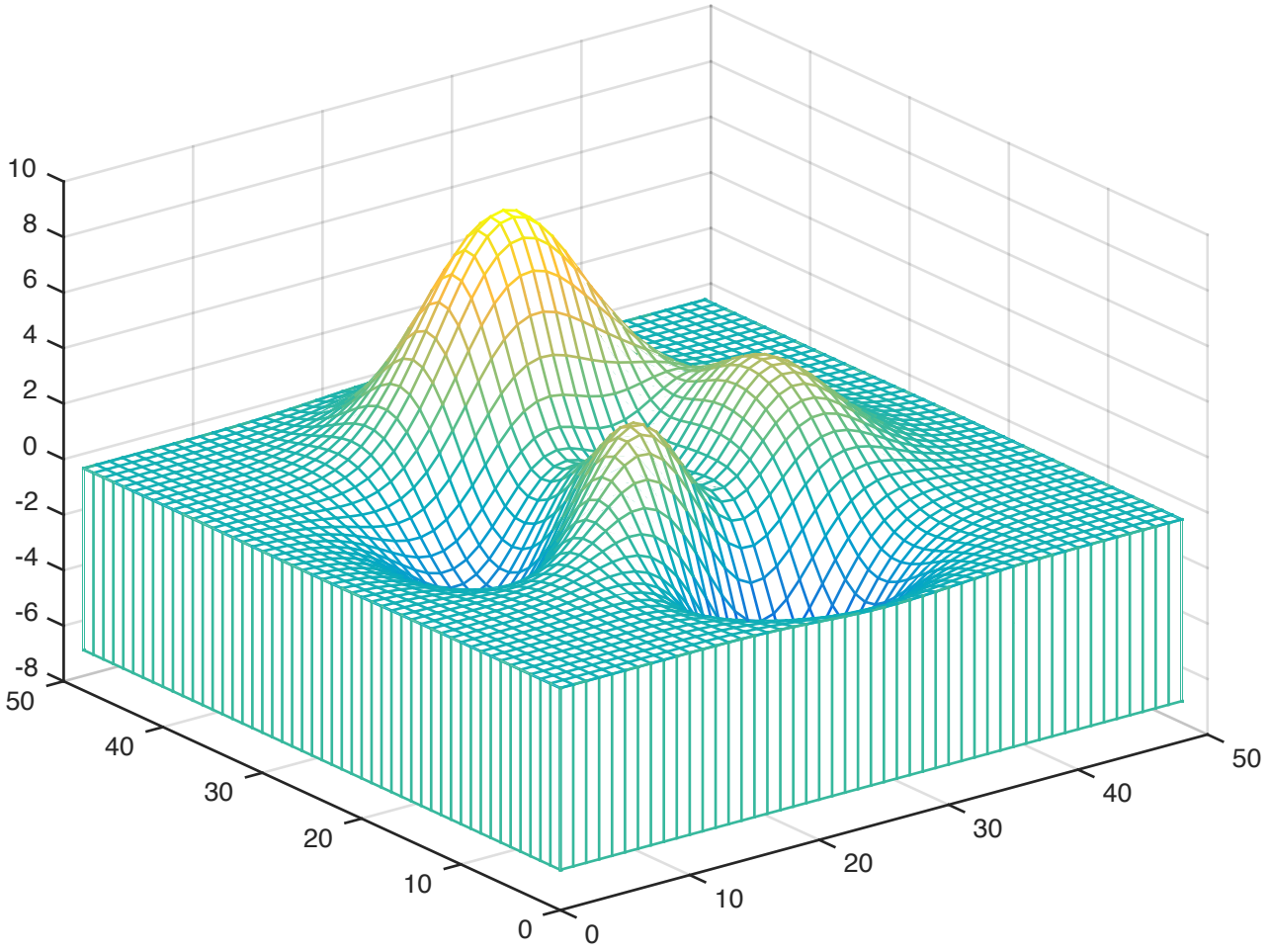
Peak Function

- For demo purpose, has three local minimums and three local maximums
- Exercise: plot this on your own computer

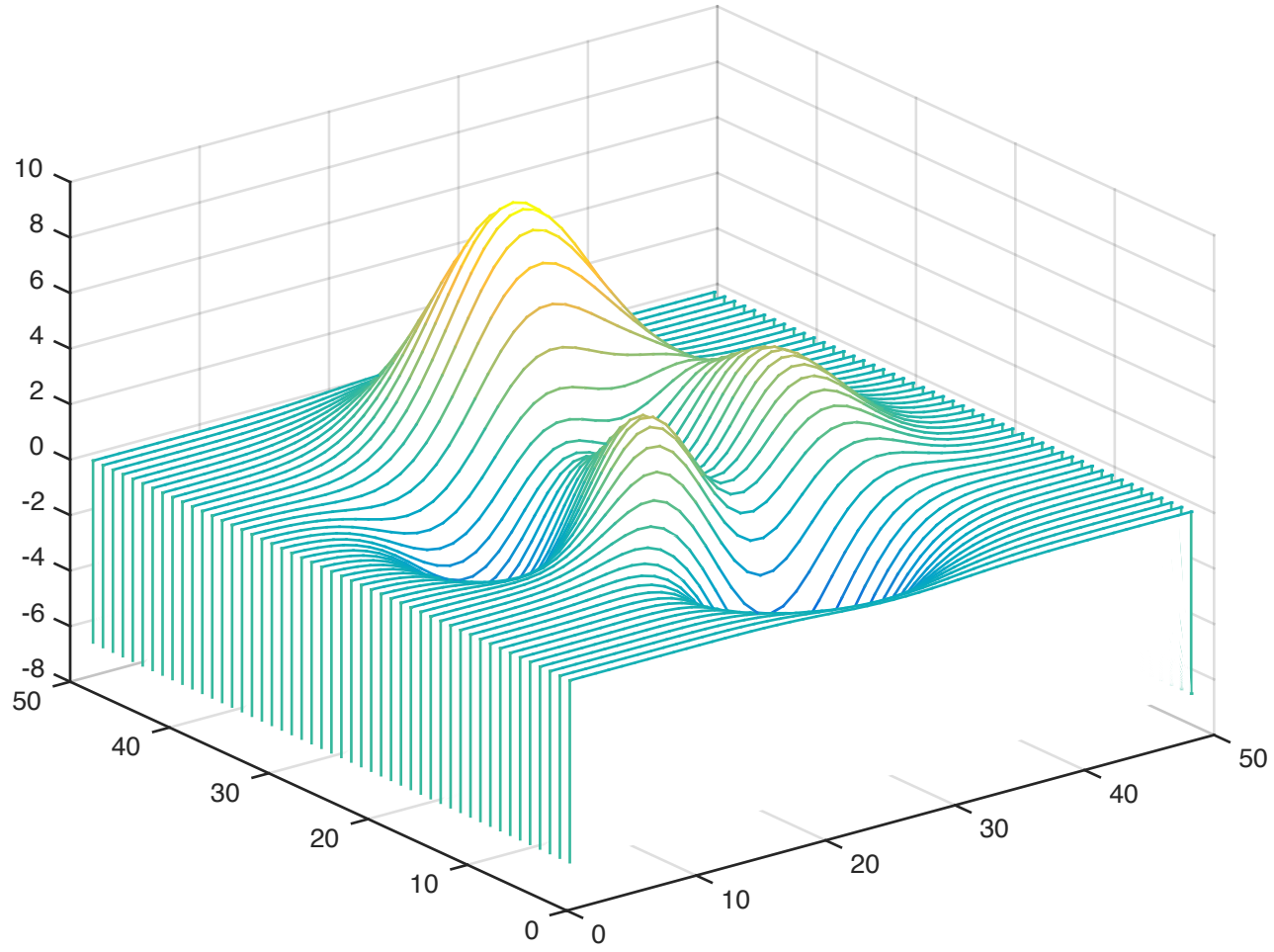
$$z = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10\left(\frac{x}{5} - x^3 - y^5\right)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2}$$



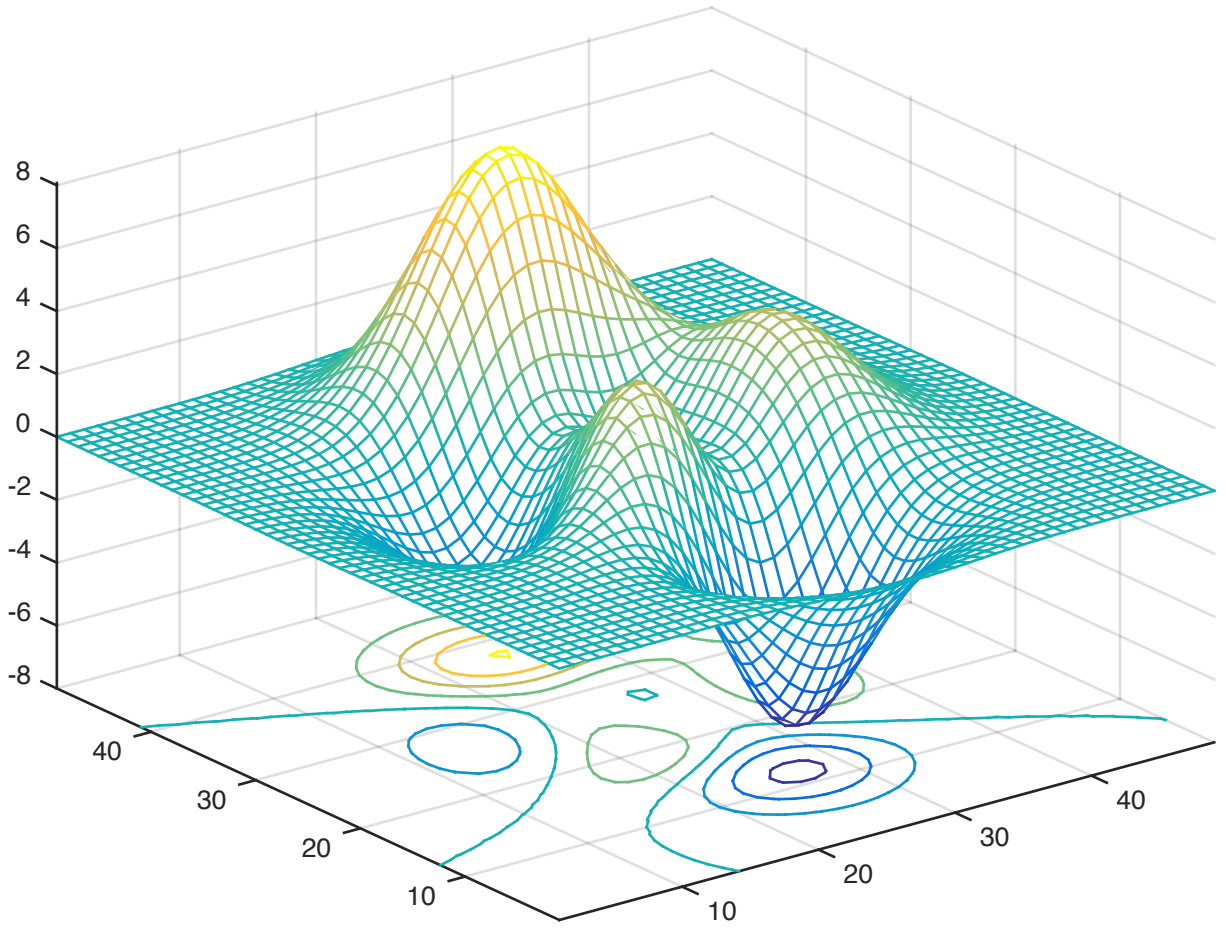
Meshz



Waterfall

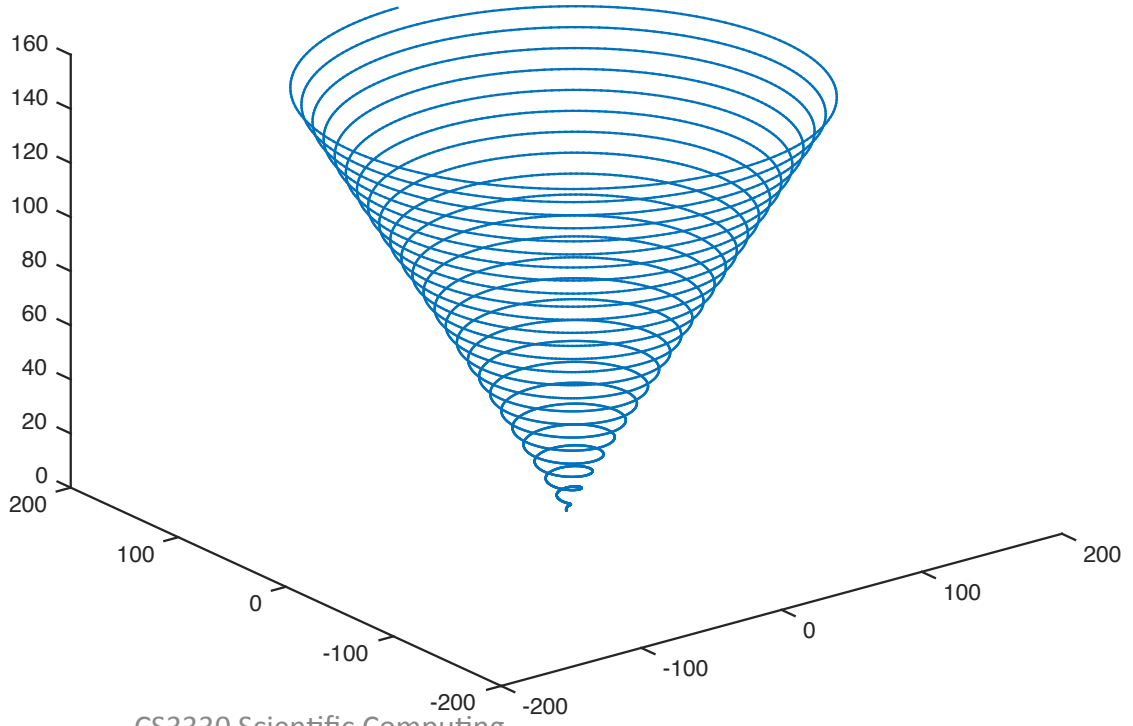


Meshc (Contours)



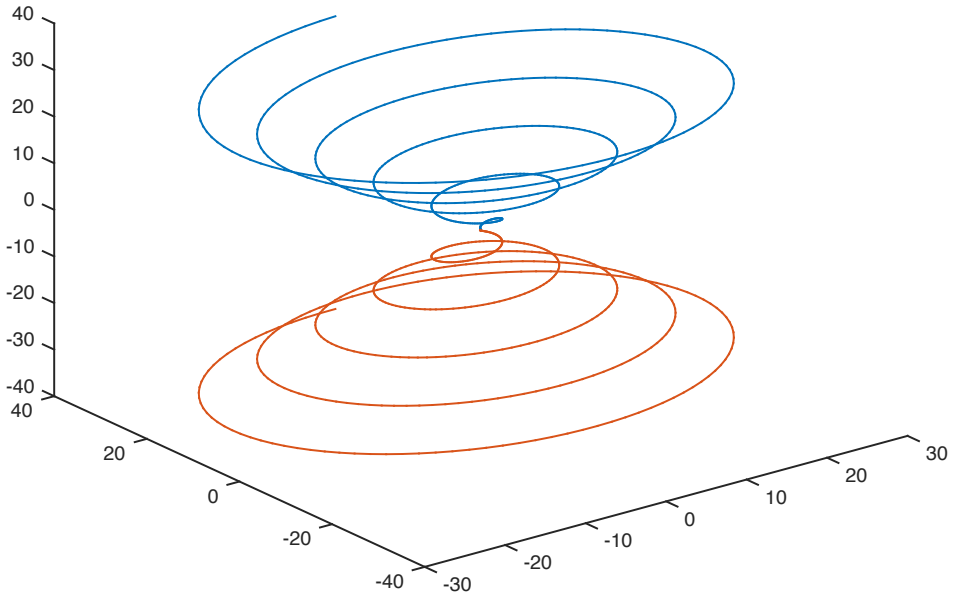
Plots in 3-D

```
t = linspace(0,50*pi, 5000);
plot3(t.*sin(t), t.*cos(t), t);
```



Other Usages of Plot3D

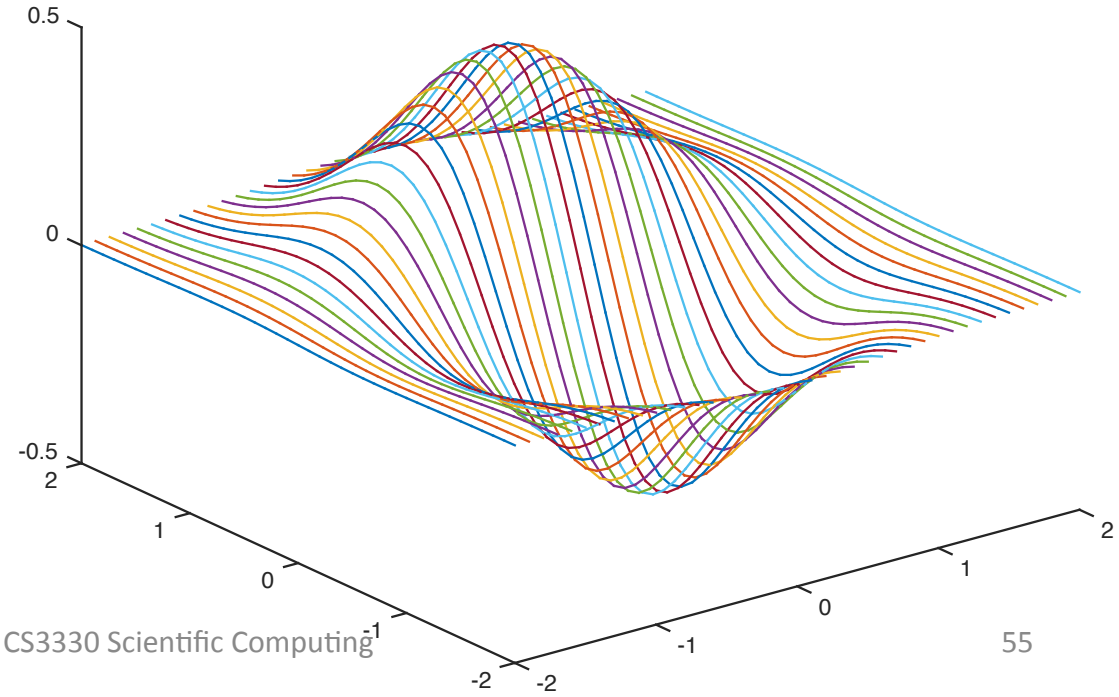
```
t = linspace(0, 10*pi, 501);
plot3(t.*sin(t), t.*cos(t), t,
      t.*sin(t), t.*cos(t), -t);
```



Other Usages of Plot3D (cont.)

- Passing 3 matrices to plot3d, each column will be plotted as a curve

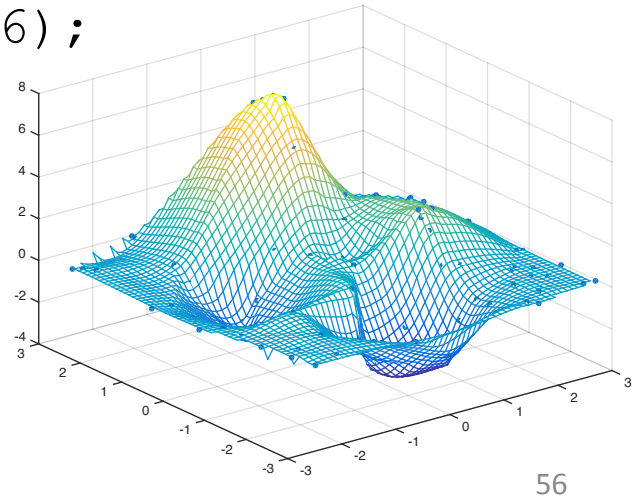
```
[x, y] = meshgrid(-2:0.1:2);
z = y.*exp(-x.^2-y.^2);
plot3(x, y, z);
```



Interpolation Using Griddata

```
x = 6*rand(100,1)-3;
y = 6*rand(100,1)-3;
z = peaks(x, y);
[X, Y] = meshgrid(-3:0.1:3);
Z = griddata(x, y, z, X, Y, 'cubic');
mesh(X, Y, Z);
hold on
plot3(x, y, z, '.', 'MarkerSize', 16);
```

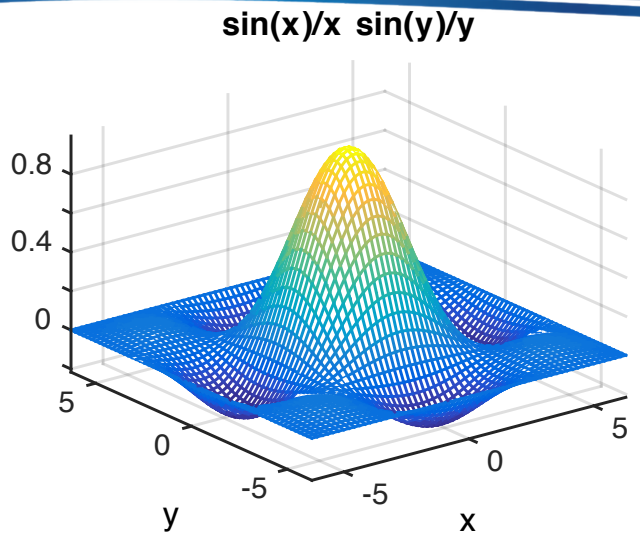
Griddata supports various interpolation algorithm, the default is linear

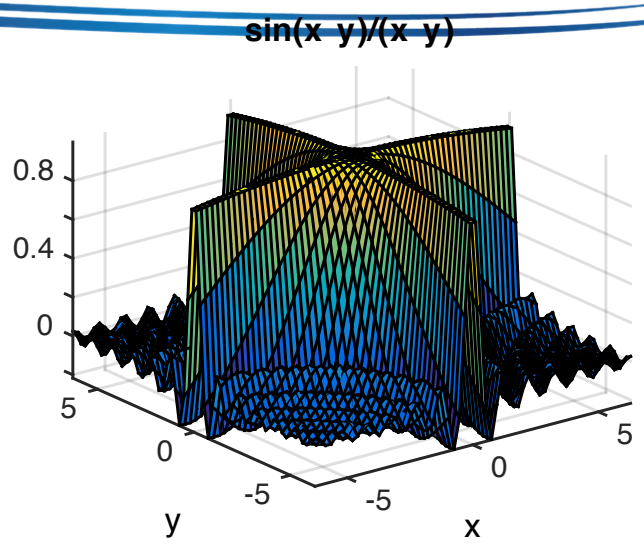


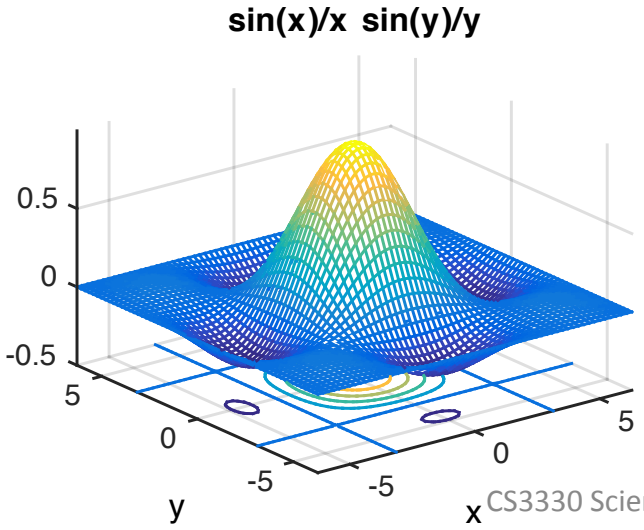
Ezmesh and Ezsurf

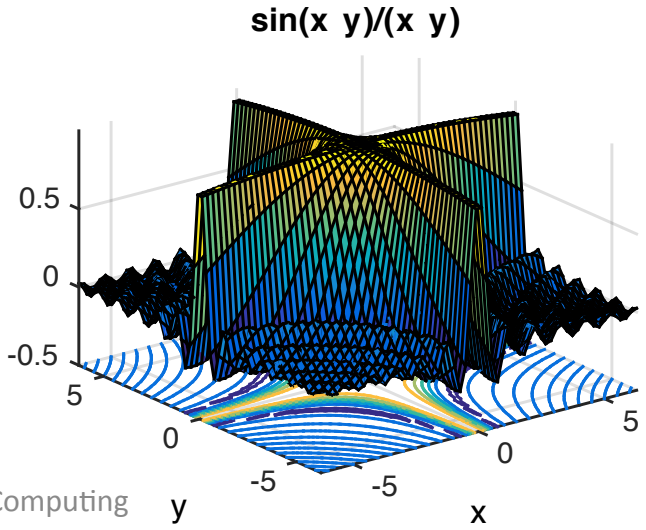
```
subplot(2,2,1);  ezmesh('sin(x)/x*sin(y)/y');
subplot(2,2,2);  ezsurf('sin(x*y)/(x*y)');
subplot(2,2,3);  ezmeshc('sin(x)/x*sin(y)/y');
subplot(2,2,4);  ezsurfc('sin(x*y)/(x*y)');
```

Ezmesh and Ezsurf (cont.)





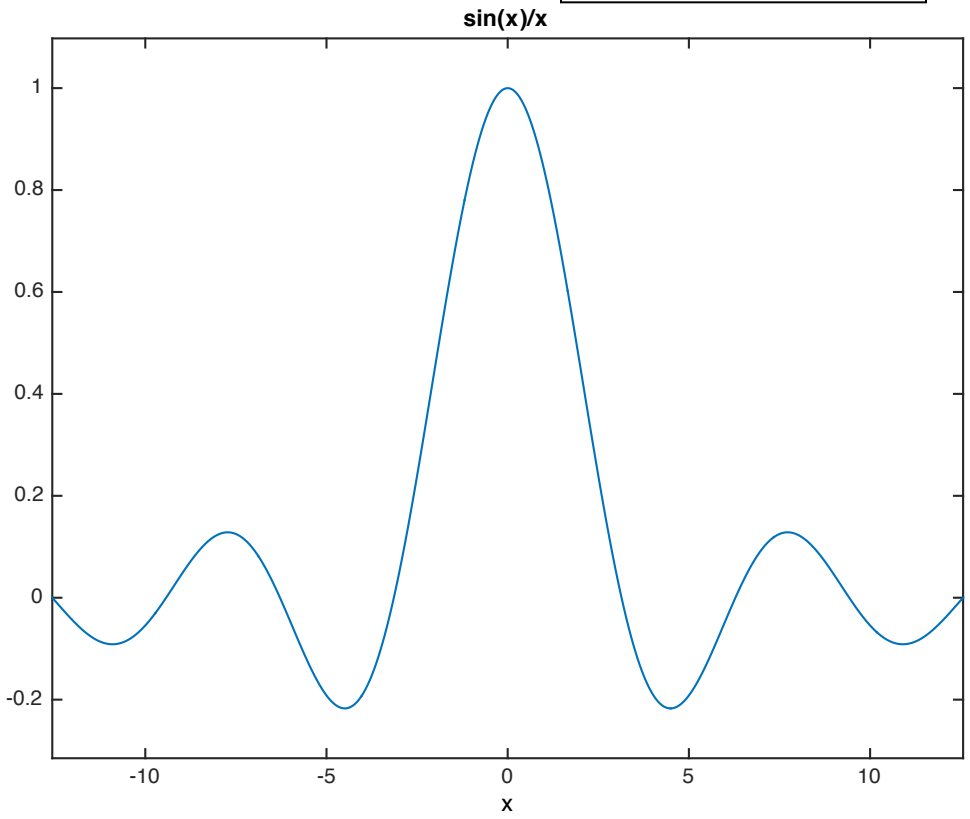




Using Ezplot to Check Math Proof

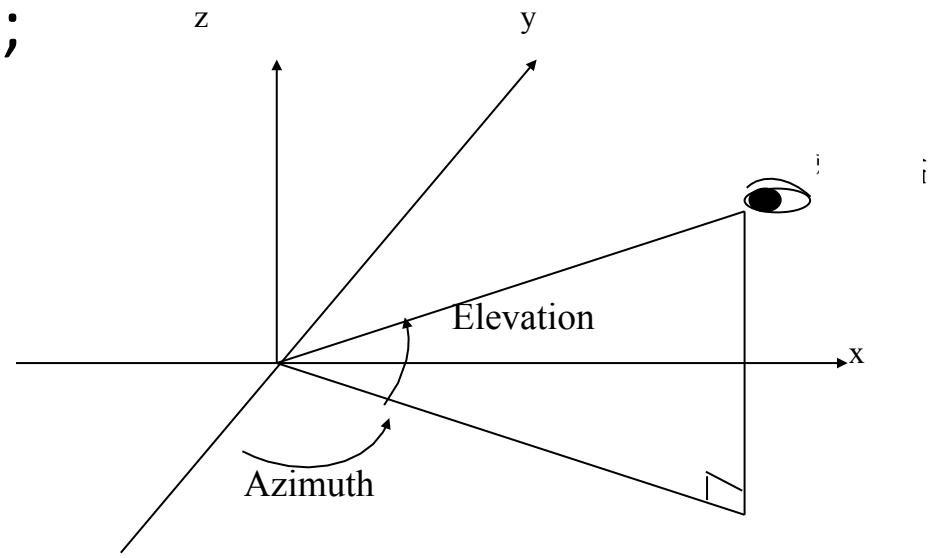
- ezplot('sin(x)/x', 4*pi*[-1,1]);
- Check the proof
- Cannot substitute the proof though

$$\lim_{x \to 0} \frac{\sin x}{x} = 1$$



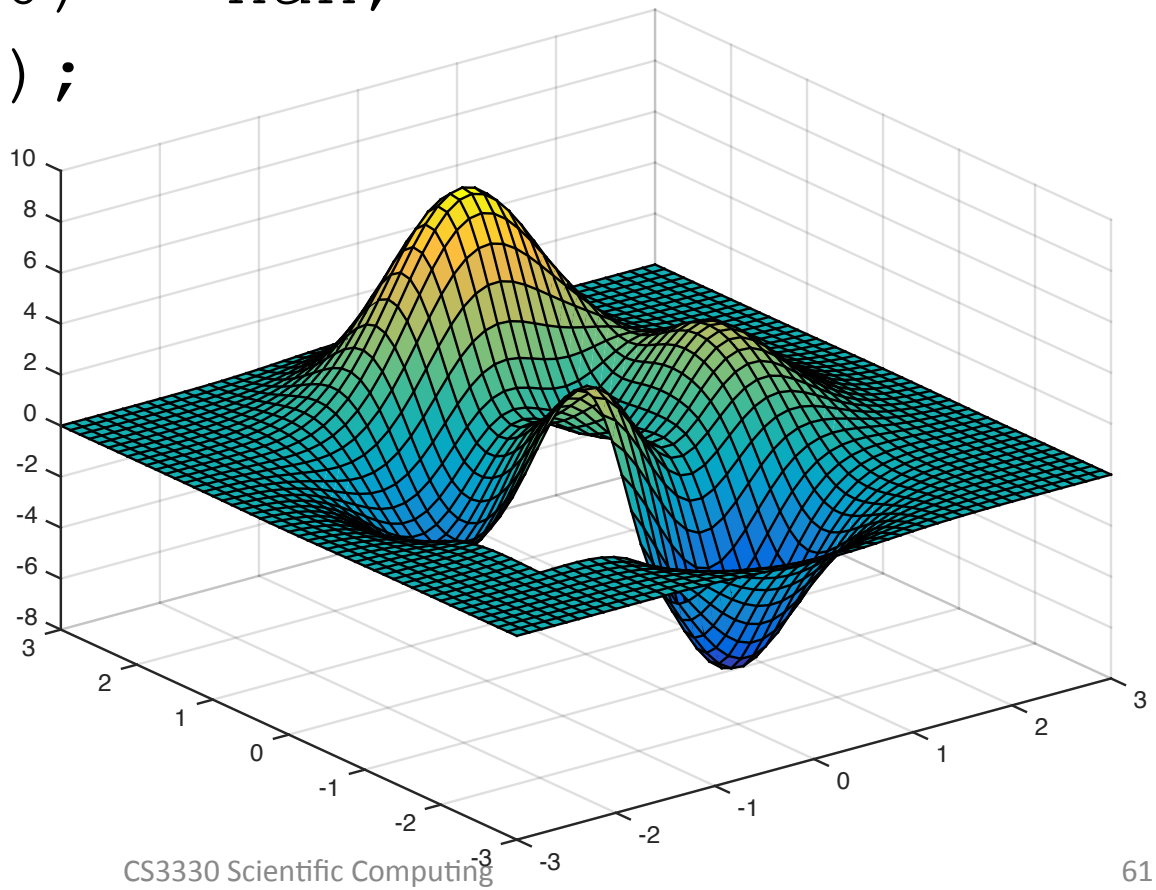
Rotating 3D Graph

- Use GUI
- Use view command
  - peaks;
  - view([az,el]);

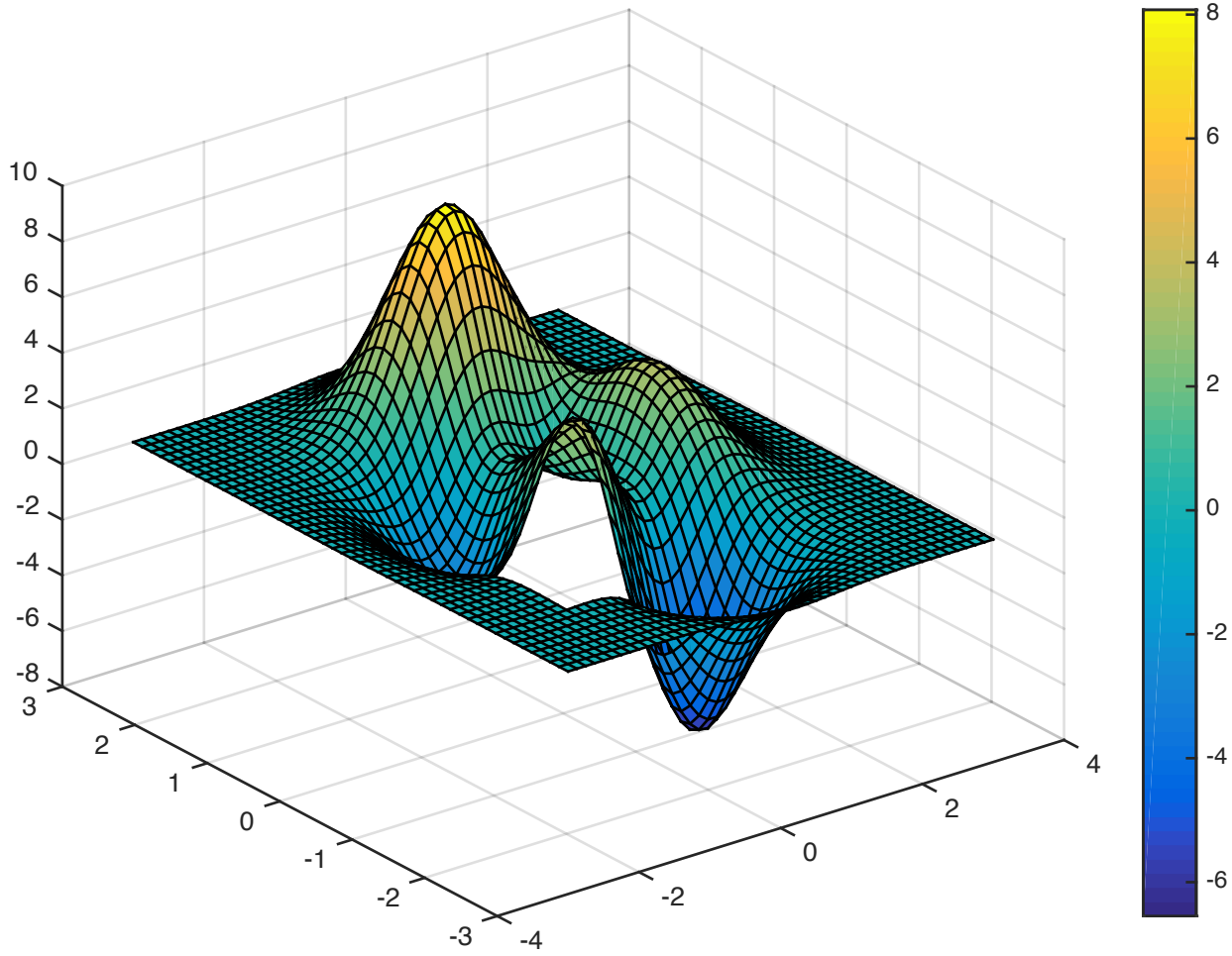


Cut Part of 3D Graph Out Using NaN

```
[X, Y, Z] = peaks;
Z(10:20,10:20) = nan;
surf(X, Y, Z);
```

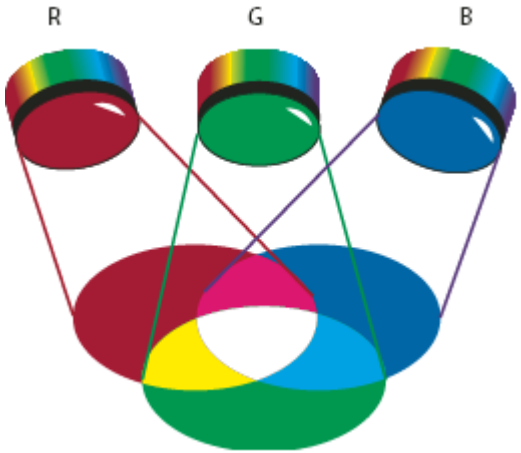


Colorbar



RGB Colorspace

Color	Red	Green	Blue
black	0	0	0
white	1	1	1
red	1	0	0
green	0	1	0
blue	0	0	1
yellow	1	1	0
magenta	1	0	1
cyan	0	1	1
gray	0.5	0.5	0.5
dark red	0.5	0	0
copper	1	0.62	0.4
aquamarine	0.49	1	0.83



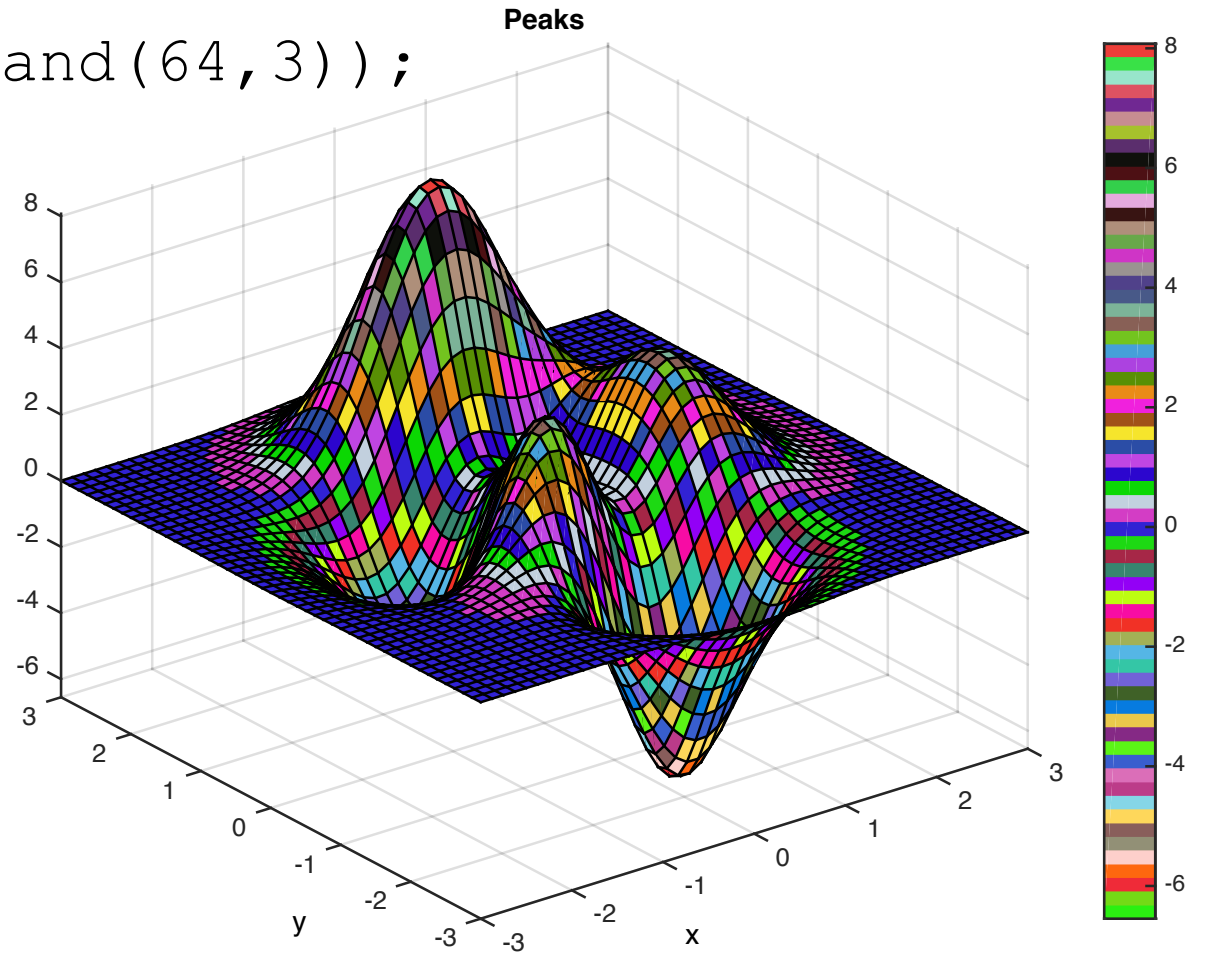
Colormap

- cm = colormap;
- size(cm)
- ans = 64 3

- In 3D figures, the lowest z surface has the color specified in row 1 of cm
- There are 64 z surfaces

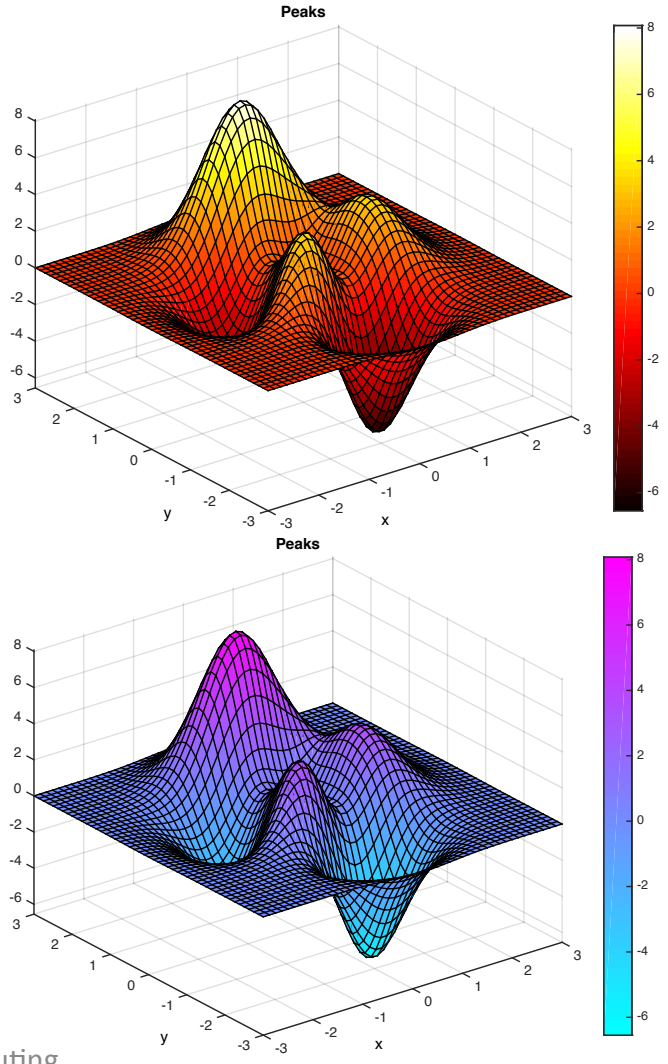
Example of Colormap

- `peaks;`
- `colormap(rand(64,3));`
- `colorbar;`



Predefined Colormaps

- hsv: default
- hot
- cool
- summer
- gray
- …….



Exercise: What Will Happen?

```
peaks;
colormap hsv
colorbar
colormap(colormap.^2)
colormap(colormap.^2)
colormap(colormap.^2)
colormap(colormap.^2)
…
```

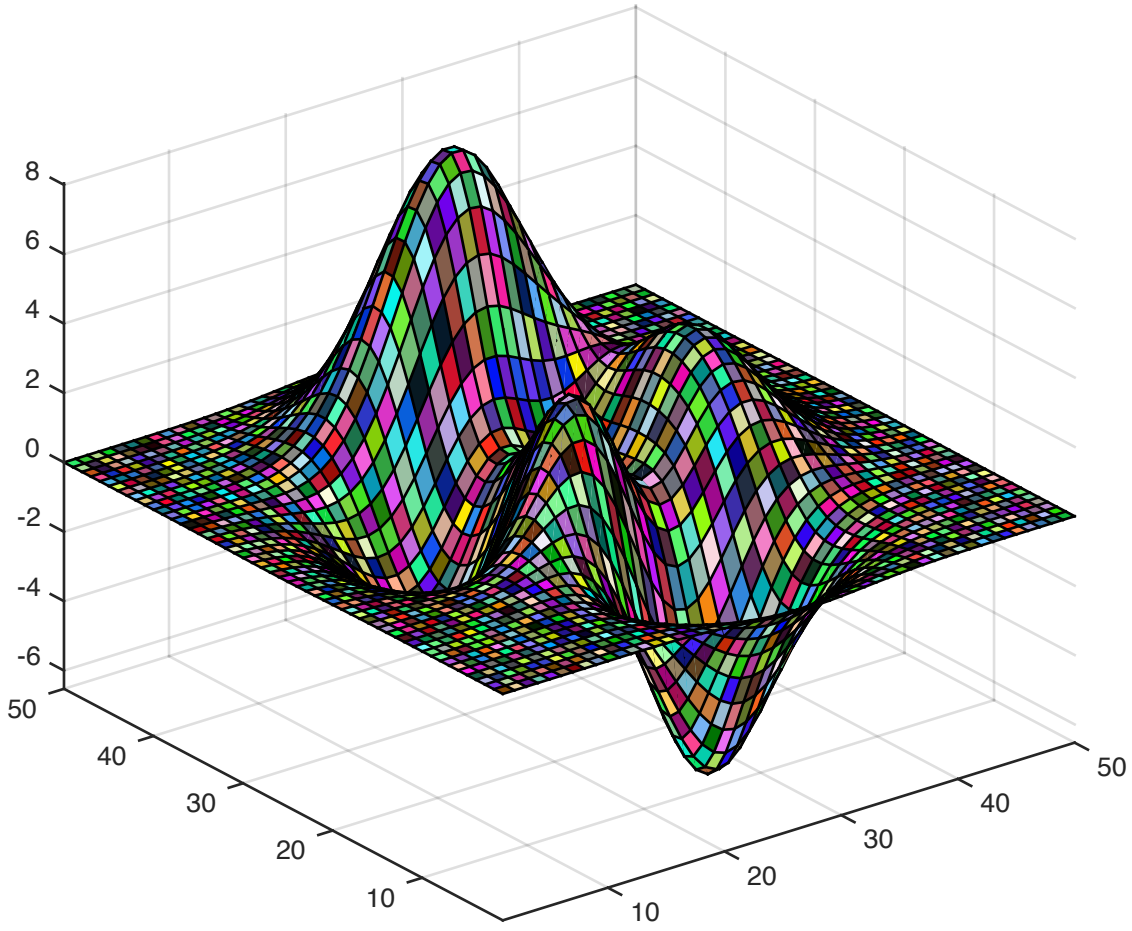
```
peaks;
colormap hsv
colorbar
colormap(colormap.^.5)
colormap(colormap.^.5)
colormap(colormap.^.5)
colormap(colormap.^.5)
…
```

True Colors

- Almost all modern computers can do 24-bit true colors

```
Z = peaks(50);
C(:, :, 1) = rand(50);
C(:, :, 2) = rand(50);
C(:, :, 3) = rand(50);
surf(Z, C);
```

Indexed Colors and True Colors (cont.)

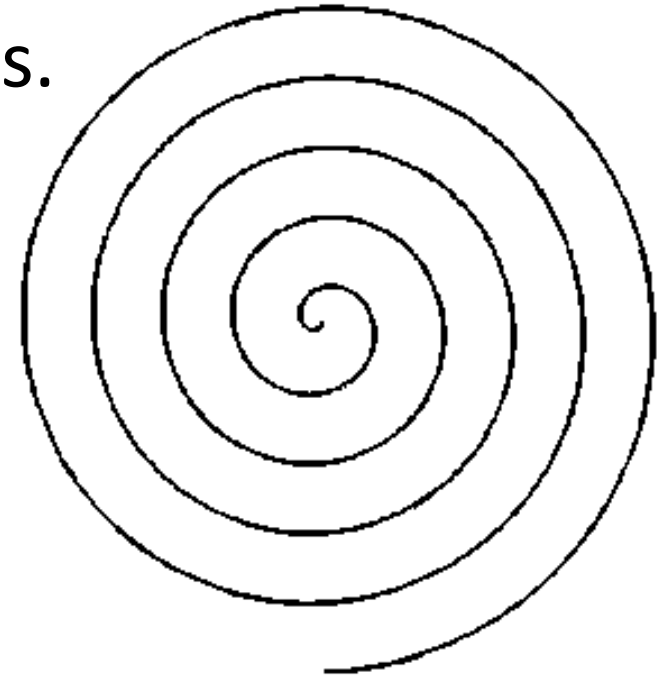


Matlab #2 Homework (M2)

1. (1%) Chebyshev polynomial is defined as $y = \cos(m*\cos^{-1}x)$, where x is between -1 and 1. Let m=1, 2, 4, 6. Plot these 4 curves with different colors and line-styles in a 2-D plot, and create the legend. Source code must be included in your report.

Matlab #2 Homework (M2) cont.

2. (1%) Plot a spiral like this. It is ok to have x, and y axes, and other standard Matlab figure components.



Matlab #2 Homework (M2) cont.

3. (1%) Ellipsoid can be described by this equation: $\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 + \left(\frac{z}{c}\right)^2 = 1$, where a, b, and c are real numbers. Write a function to create smooth ellipsoid without invoking the built-in ellipsoid function. Plot an ellipsoid with a=6, b= 4, c=2.