Matlab 3: More Graph, Animation, and GUI

Cheng-Hsin Hsu

National Tsing Hua University

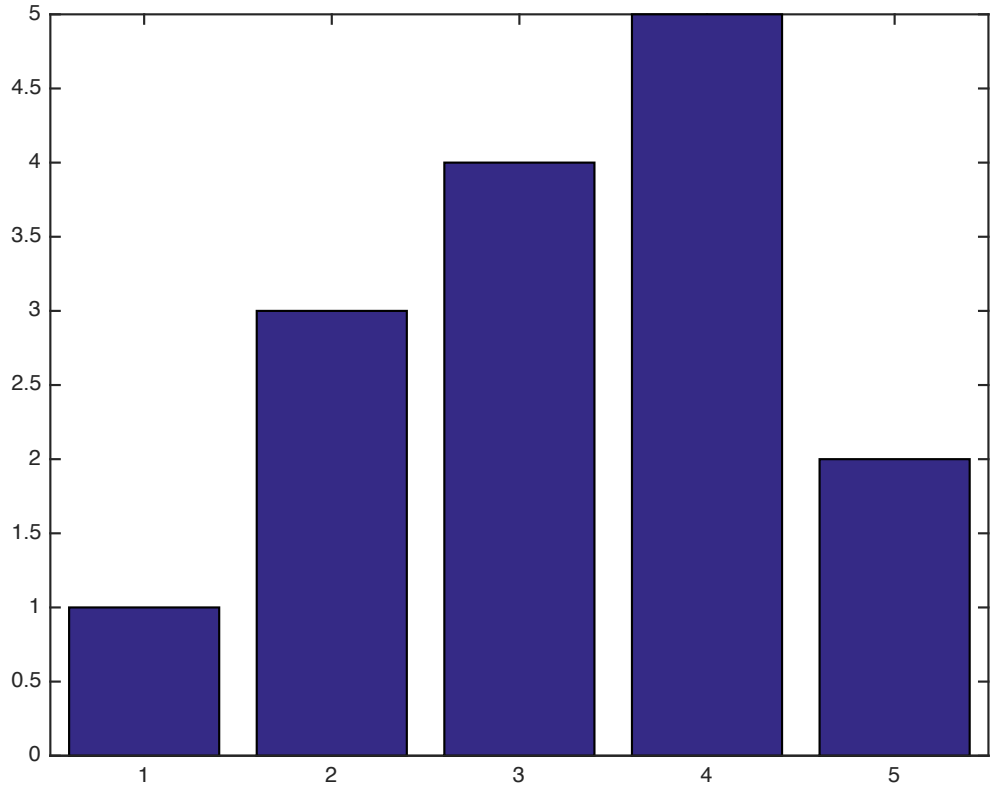
Department of Computer Science

Slides are based on the materials from Prof. Roger Jang

Bar Chart

- Fewer and discrete samples

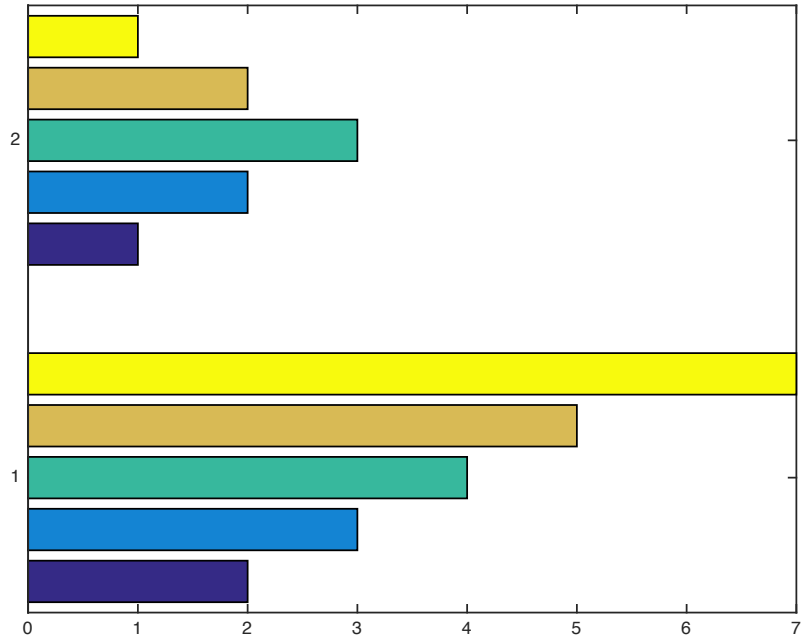
```
x = [1 3 4 5 2];
bar(x);
```



Bar Chart (cont.)

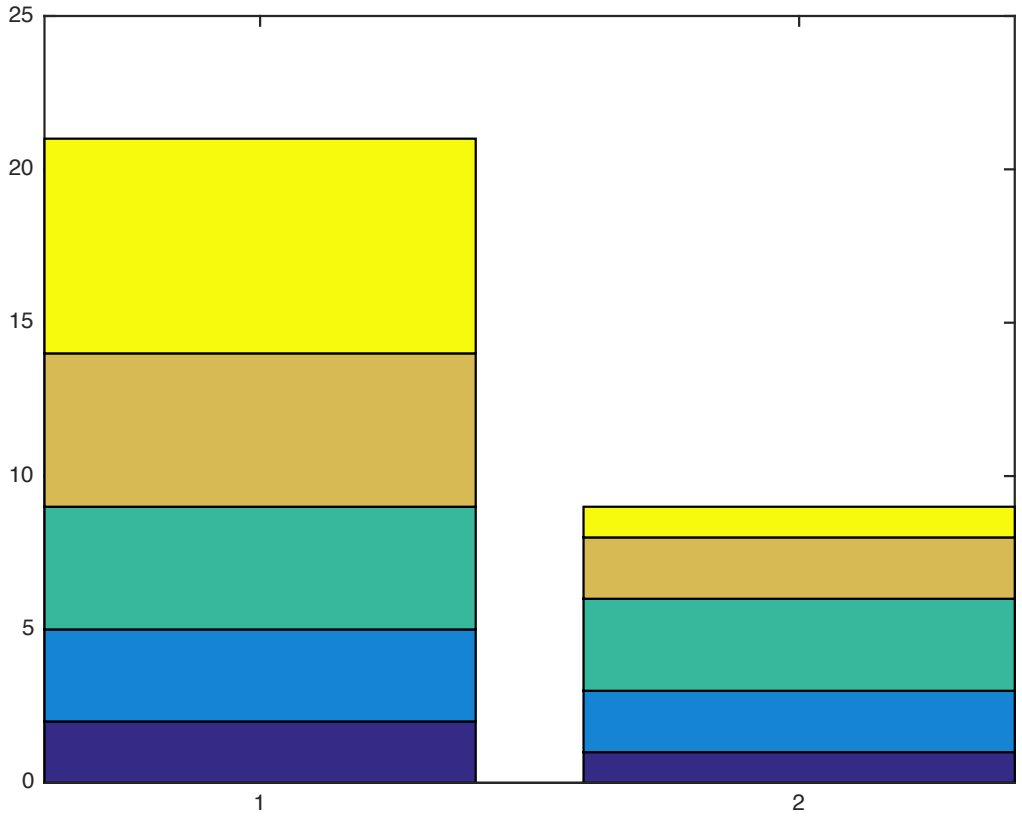
- For 2-D matrices, data in the same row are group together
- barh plots bar charts horizontally

```
x = [2 3 4 5 7;
     1 2 3 2 1];
barh(x);
```



Stacked Bar Chart

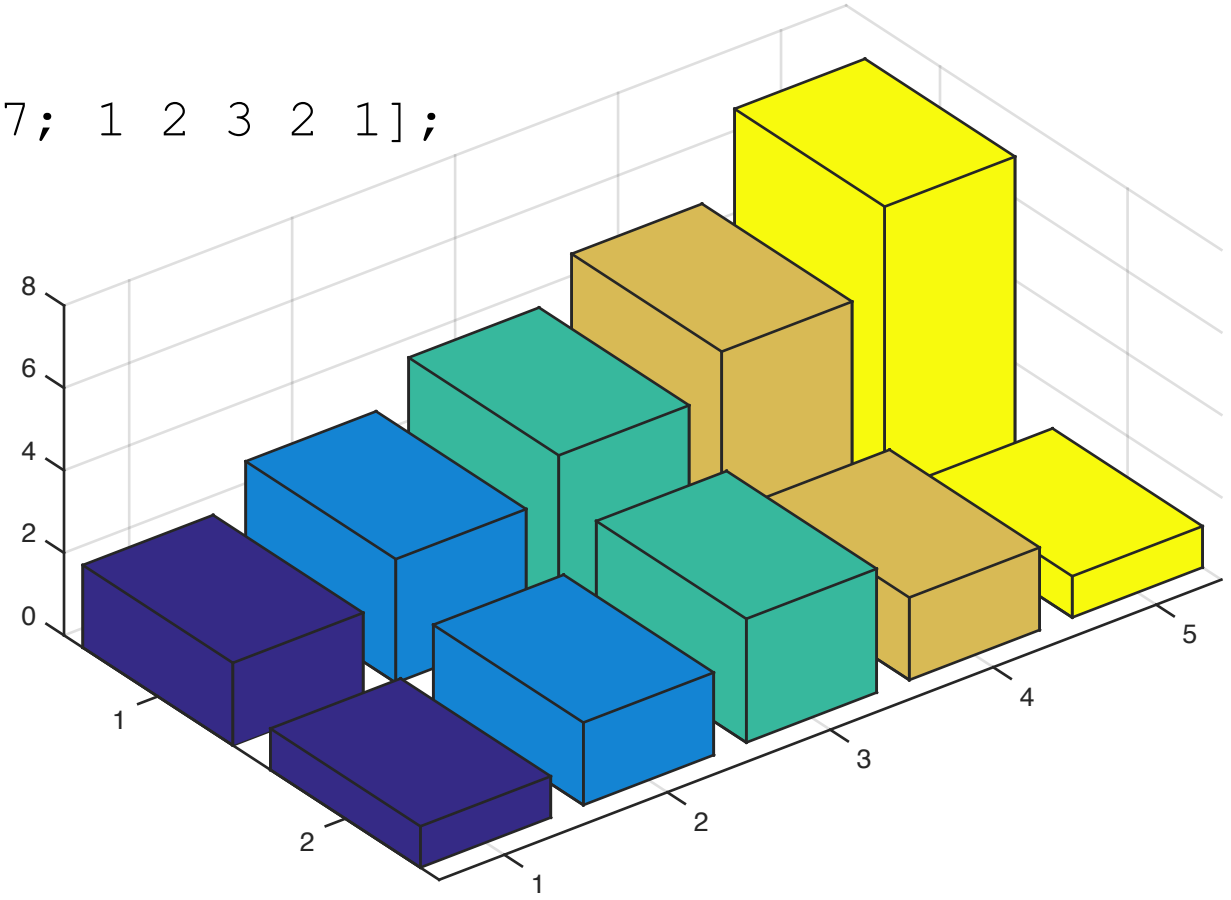
```
x = [2 3 4 5 7; 1 2 3 2 1];
bar(x,'stack')
```



3D Bar Chart

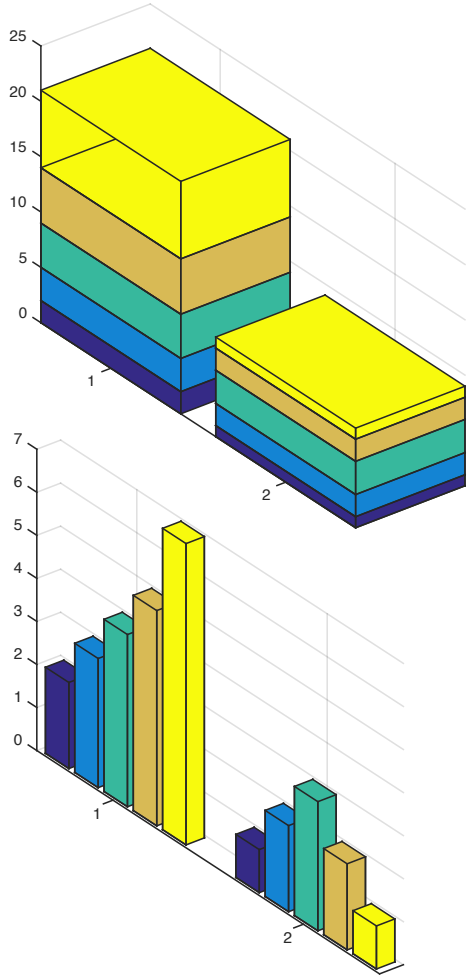
- bar3 allows us to create 3D bar charts

```
x = [2 3 4 5 7; 1 2 3 2 1];
bar3(x)
```



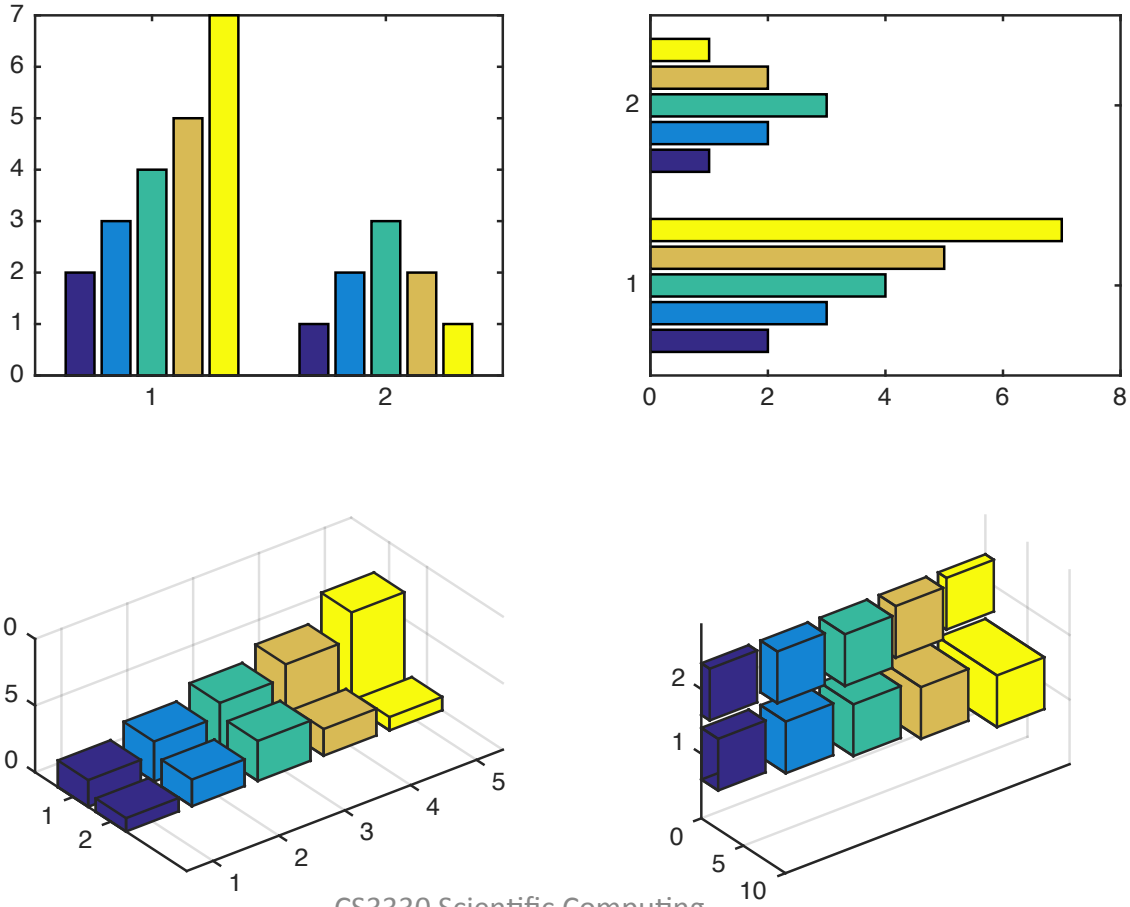
Stack versus Group

```
x = [2 3 4 5 7; 1 2 3 2 1];
bar3(x, 'stack')
figure
bar3(x, 'group')
```



Summary of Bar and Bar3

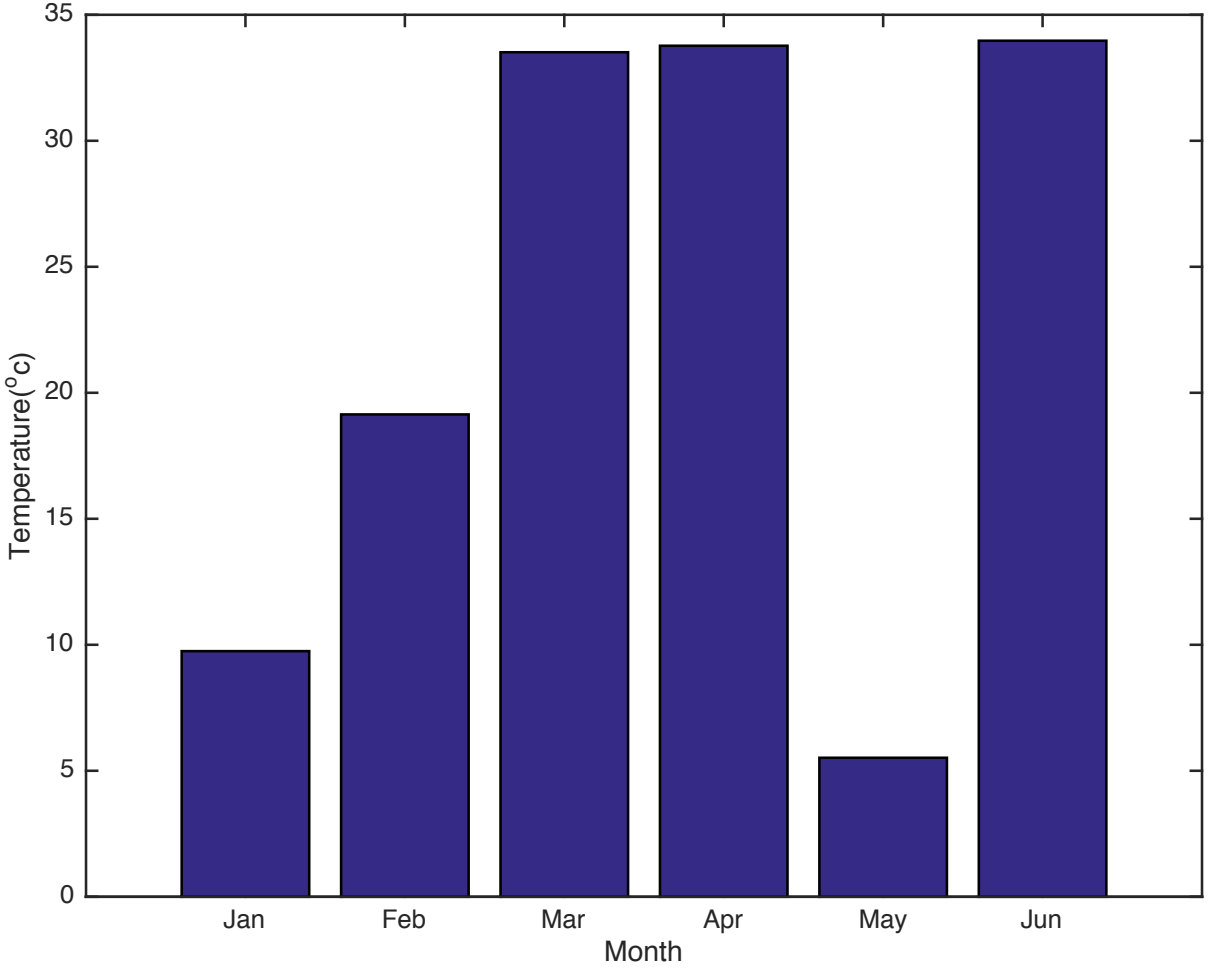
- Exercise: recreate this figure



A Comprehensive Example

```
x = 1:6;
y = 35*rand(1, 6);
bar(x, y);
xlabel('Month');
ylabel('Temperature(^{o}c)');
set(gca, 'xticklabel', {'Jan','Feb','Mar', 'Apr', 'May', 'Jun'});
```

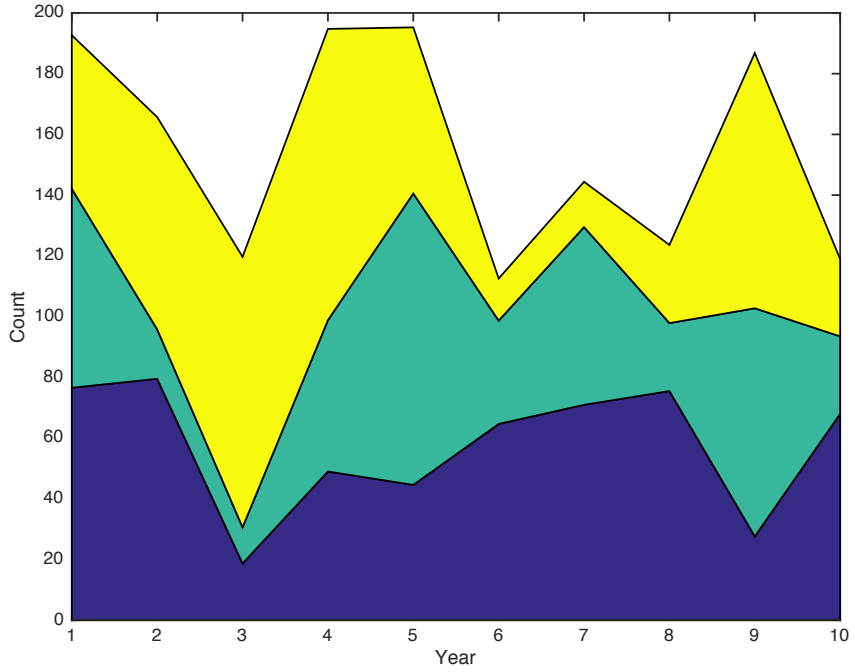
A Comprehensive Example (cont.)



Area Graph

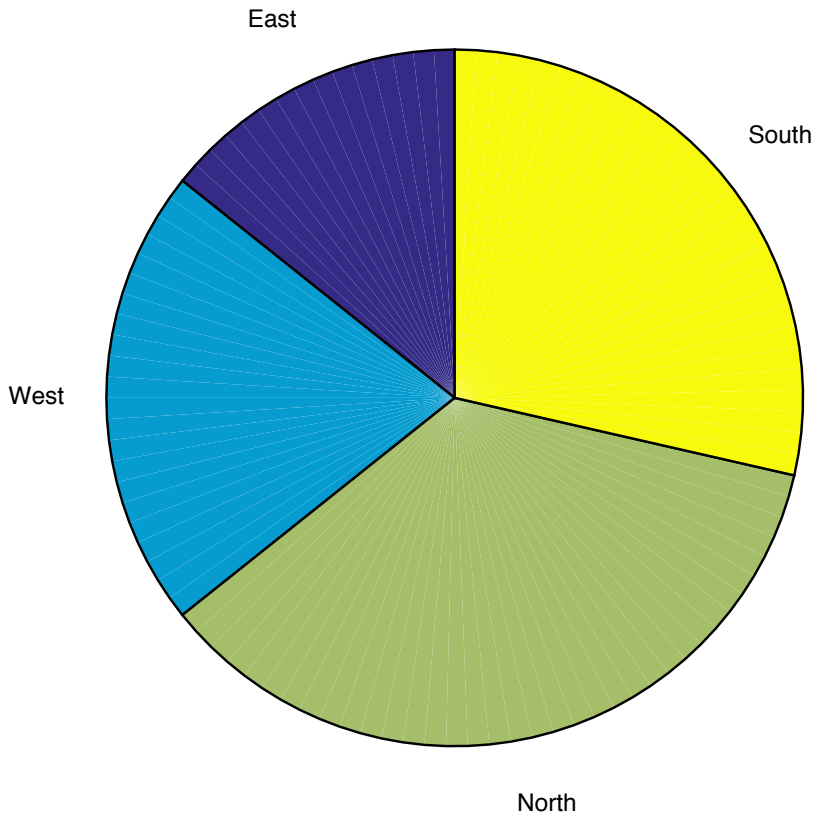
- Similar to stacked bar charts
- E.g., total head counts of undergraduates, graduates, and employees

```
y = rand(10,3)*100;
x = 1:10;
area(x, y);
xlabel('Year');
ylabel('Count')
```



Pie Chart

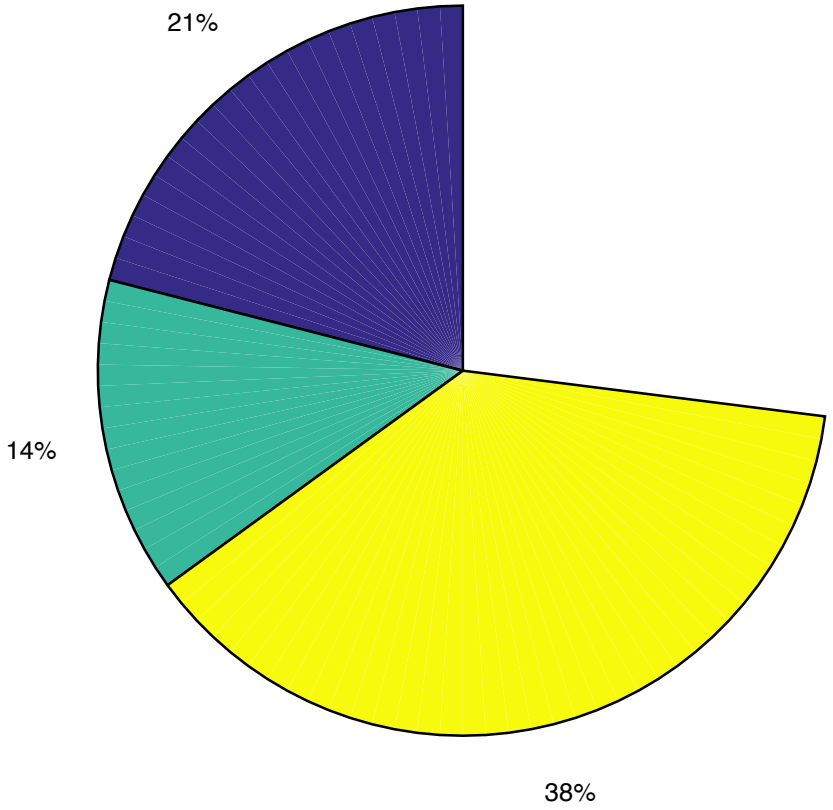
```
x = [2 3 5 4];
label={'East', 'West', 'North', 'South'};
pie(x, label);
```



Partial Pie Chart

- If the sum of the vector is less than 1, we get a partial pie chart

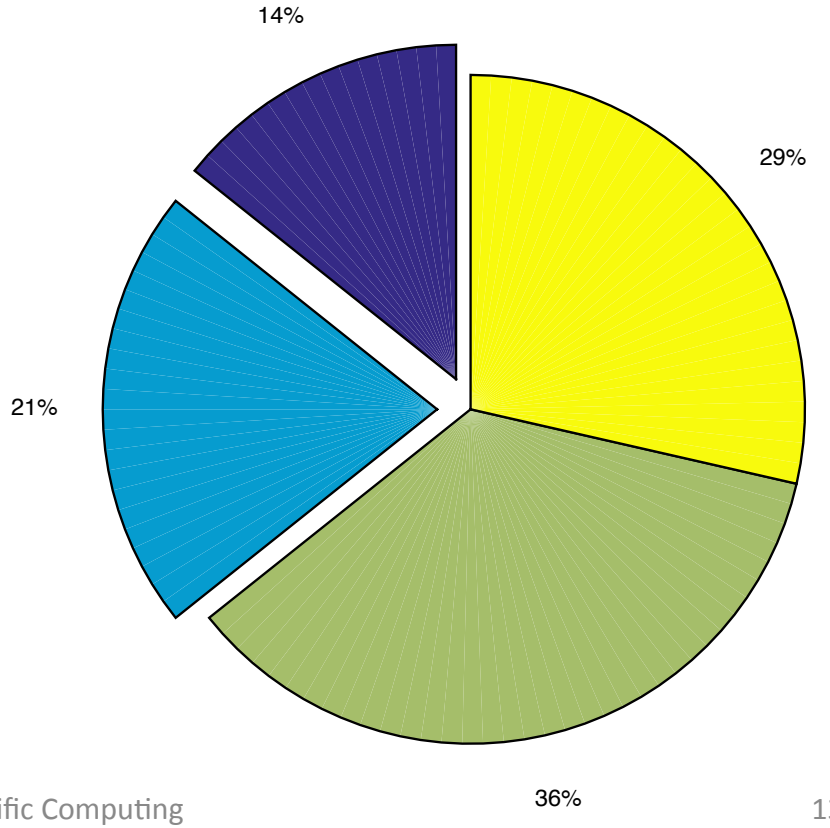
```
x = [0.21, 0.14, 0.38];
pie(x);
```



Explode Pie Chart

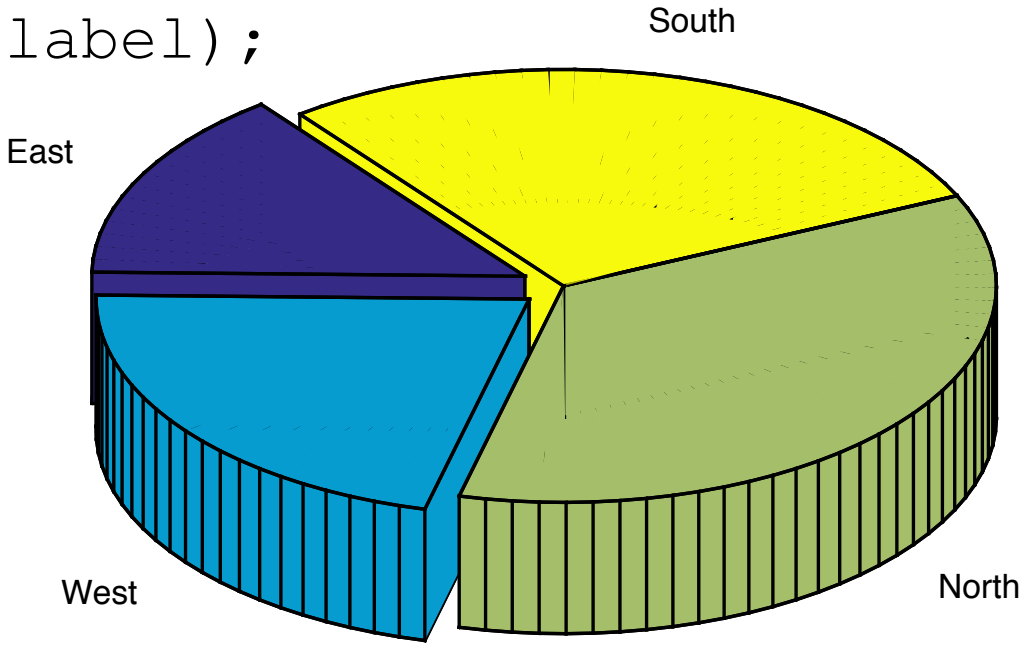
- Use a binary (zero versus nonzero) vector to indicate if a piece should be separated from the rest

```
x = [2 3 5 4];
explode = [1 0.5 0 0];
pie(x, explode);
```



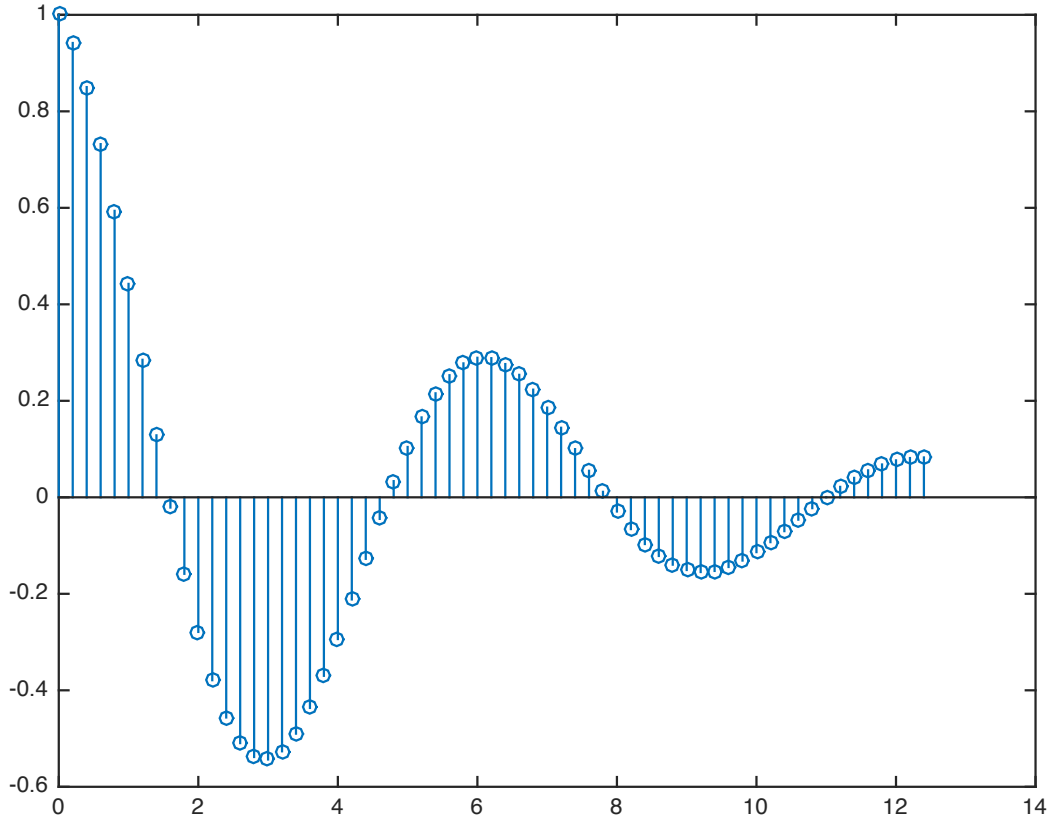
3D Pie Chart

```
x = [2 3 5 4];
explode = [1 0.5 0 0];
label={'East', 'West', 'North', 'South'};
pie3(x, explode, label);
```



Stem Plot

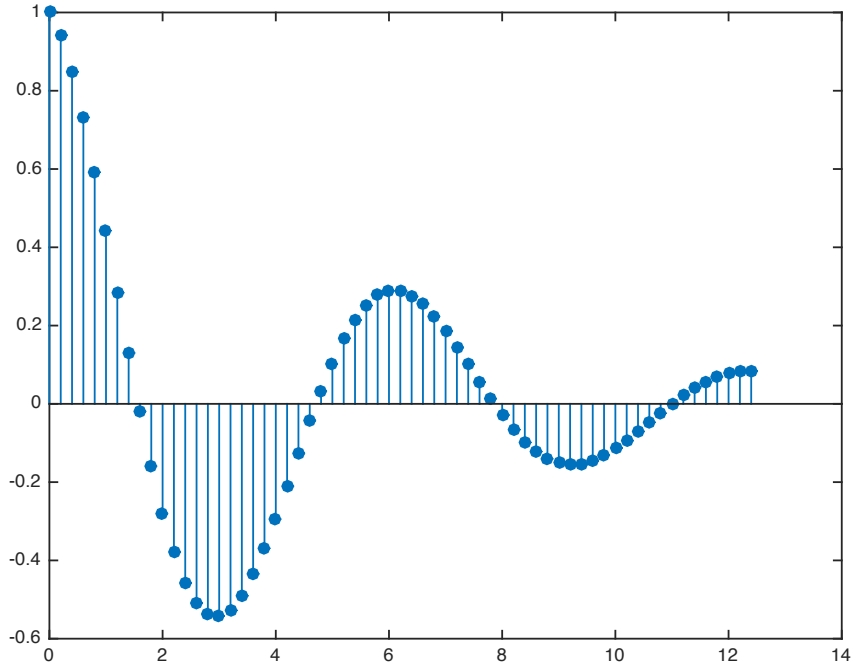
```
t = 0:0.2:4*pi;
y = cos(t).*exp(-t/5);
stem(t, y)
```



Stem Plot (cont.)

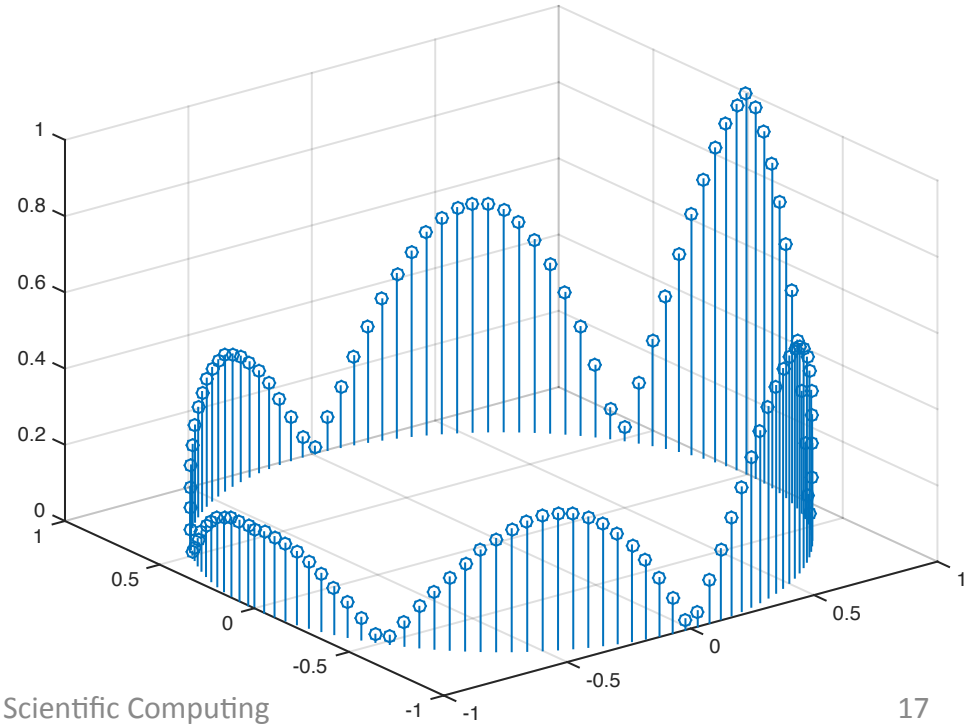
- Suitable for presenting digital signals
- Solid circle can be done by adding ‘fill’

```
t = 0:0.2:4*pi;
y = cos(t).*exp(-t/5);
stem(t, y, 'fill')
```



3D Stem Plot

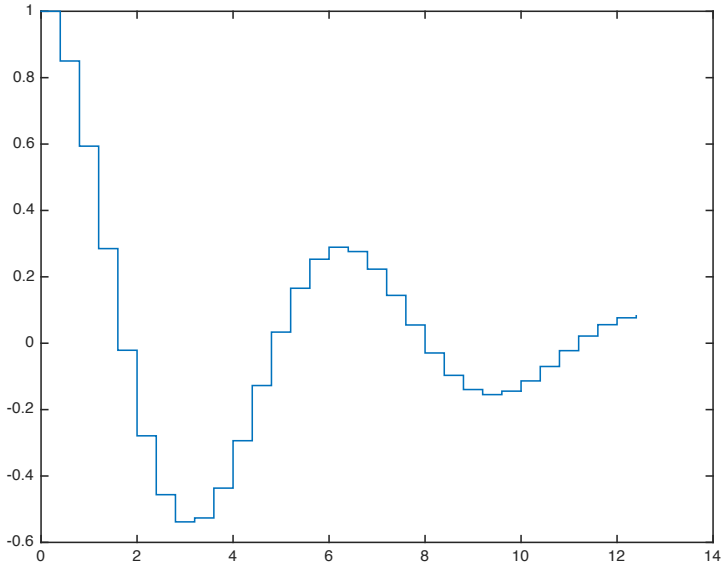
```
theta = -pi:0.05:pi;
x = cos(theta);
y = sin(theta);
z = abs(cos(3*theta)).*exp(-abs(theta/2));
stem3(x, y, z);
```



Stairstep Plots

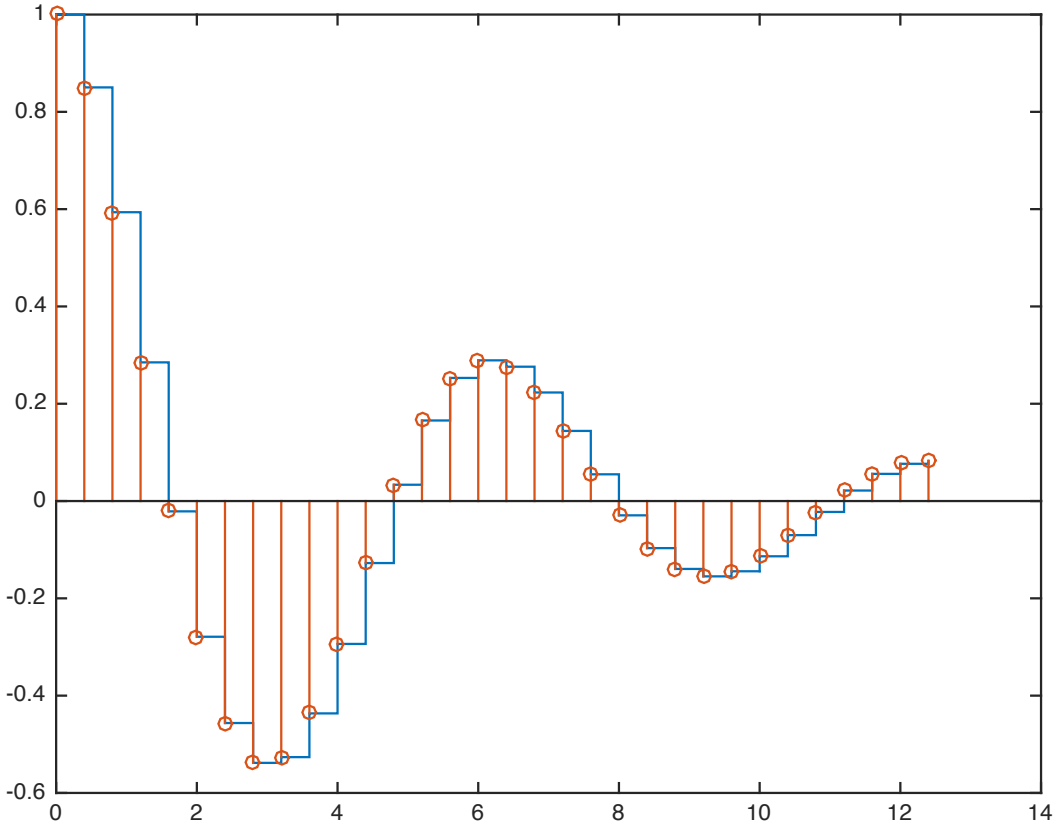
- Similar to stem, but add a horizontal segment from each sample to the sample on its right
  - Known as Zero-Order Hold (ZOH): convert discrete signals into continuous ones

```
t = 0:0.4:4*pi;
y = cos(t).*exp(-t/5);
stairs(t, y);
```



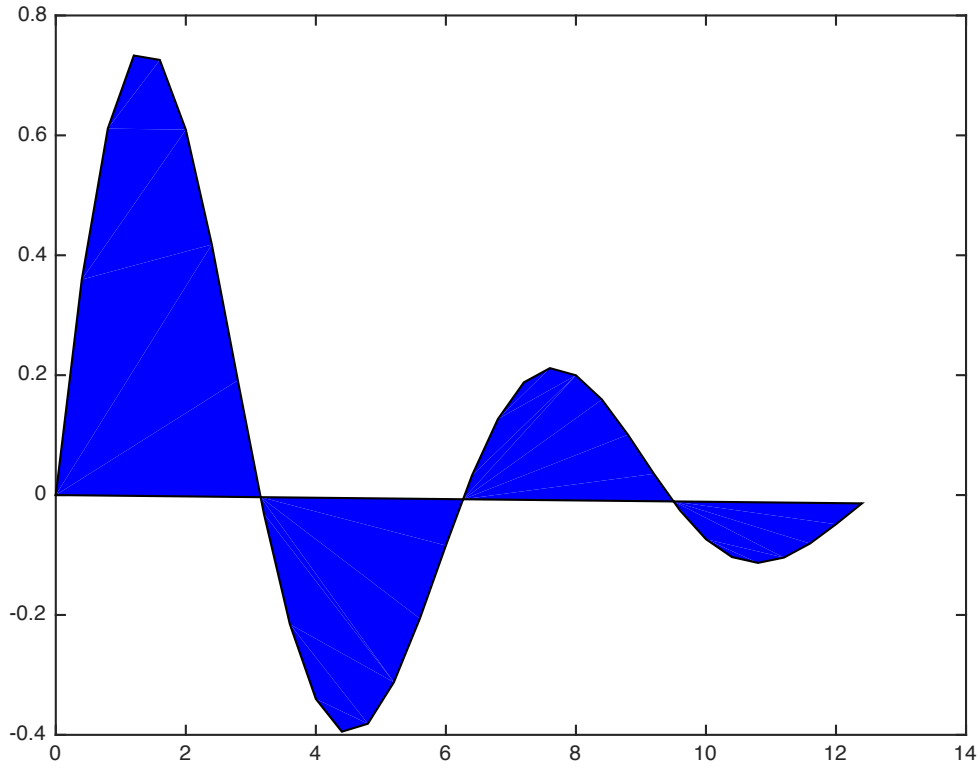
Stem + Stairstep

```
t = 0:0.4:4*pi;
y = cos(t).*exp(-t/5);
stairs(t, y);
hold on
stem(t, y);
hold off
```



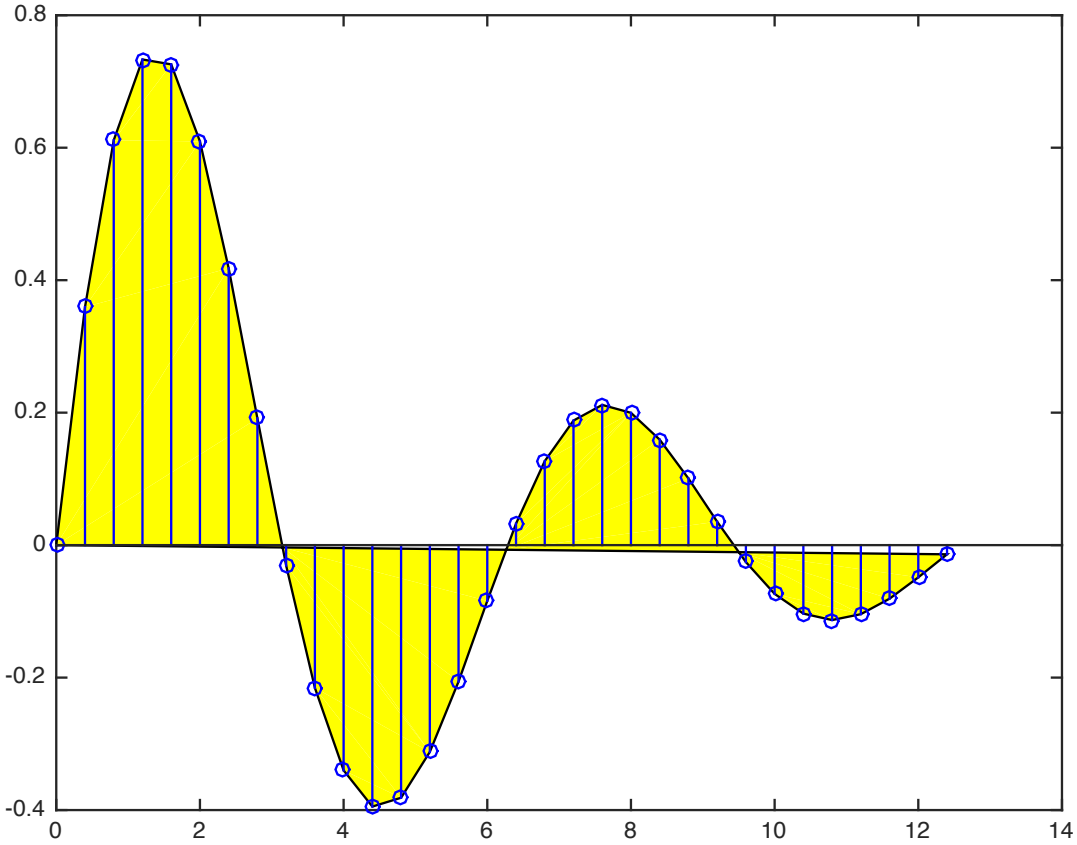
Fill Plot

```
t = 0:0.4:4*pi;
y = sin(t).*exp(-t/5);
fill(t, y, 'b')
```



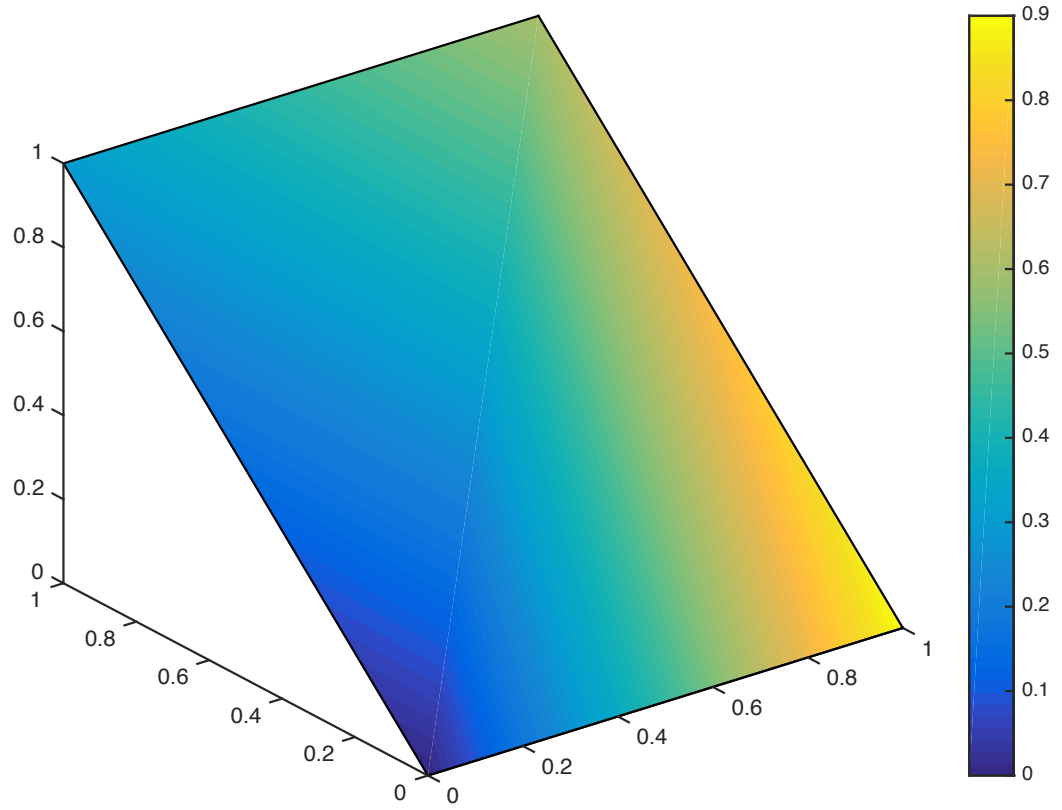
Stem + Fill

```
t = 0:0.4:4*pi;
y = sin(t).*exp(-t/5);
fill(t, y, 'y');
hold on
stem(t, y, 'b');
```



Fill3 Plot

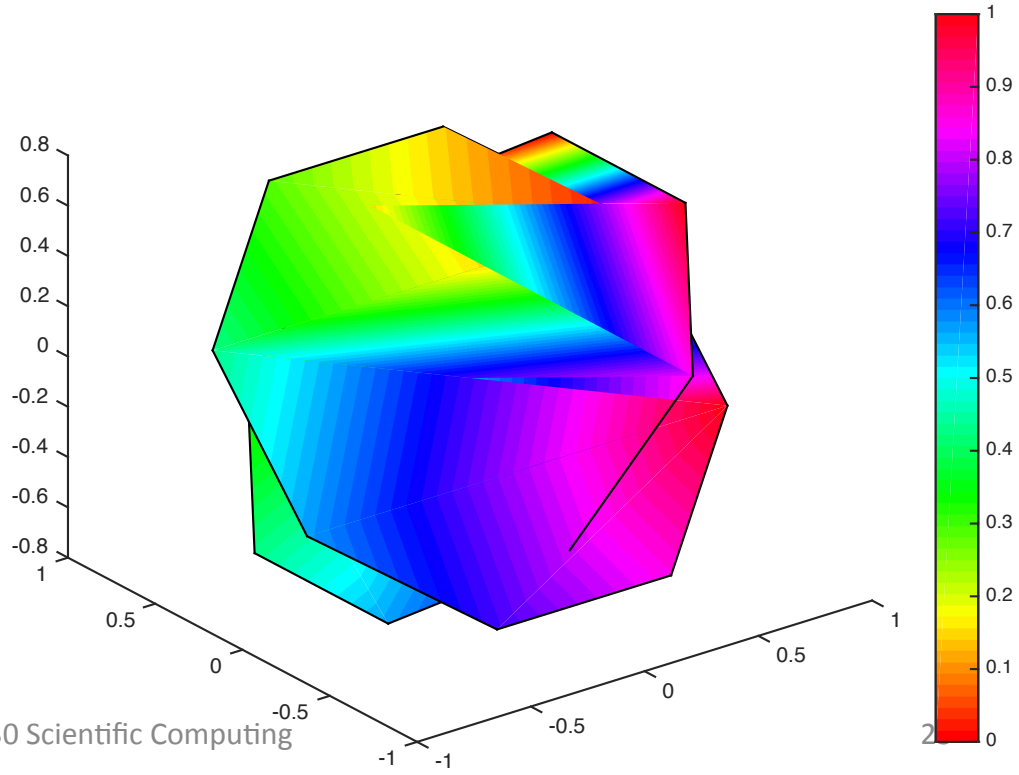
```
X = [0 0 1 1];
Y = [0 1 1 0];
Z = [0 1 1 0];
C = [0 0.3 0.6 0.9];
fill3(X, Y, Z, C);
colorbar;
```



Color interpolation! ← make sure you know how to do this by hand

Fill3 Plot

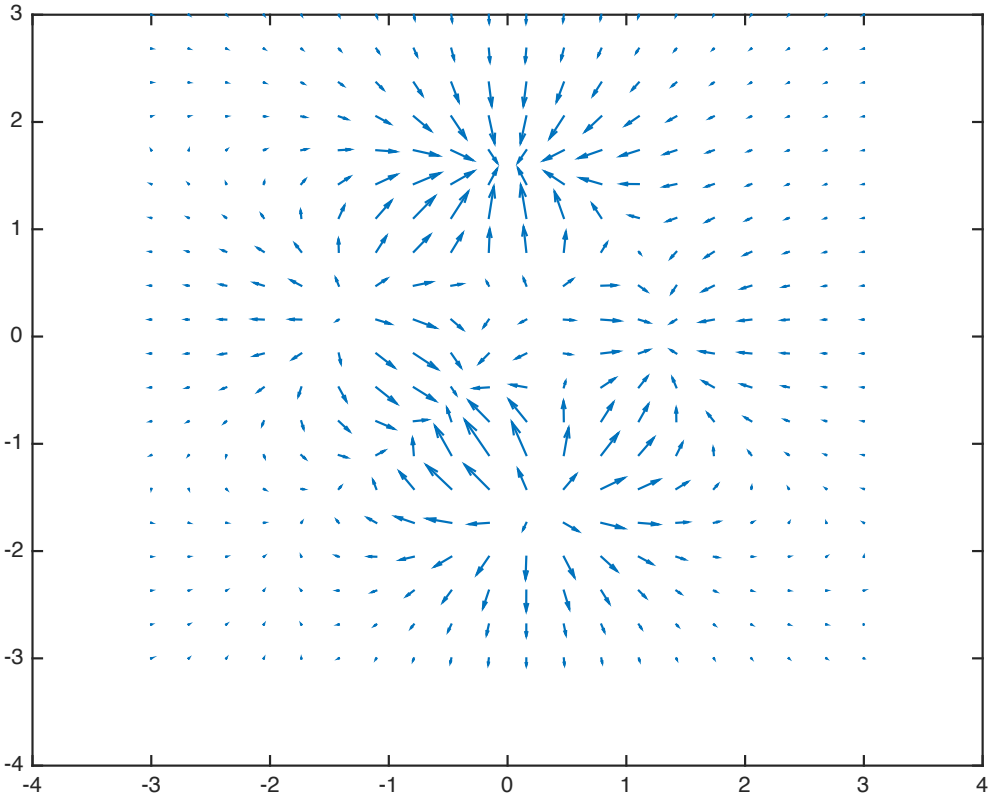
```
t = (1/16:1/8:1)*2*pi;
x = cos(t);
y = sin(t);
c = linspace(0, 1, length(t));
fill3(x, y/sqrt(2), y/sqrt(2), c, x/sqrt(2), y, x/sqrt(2), c);
colorbar;
```



Quiver Plots

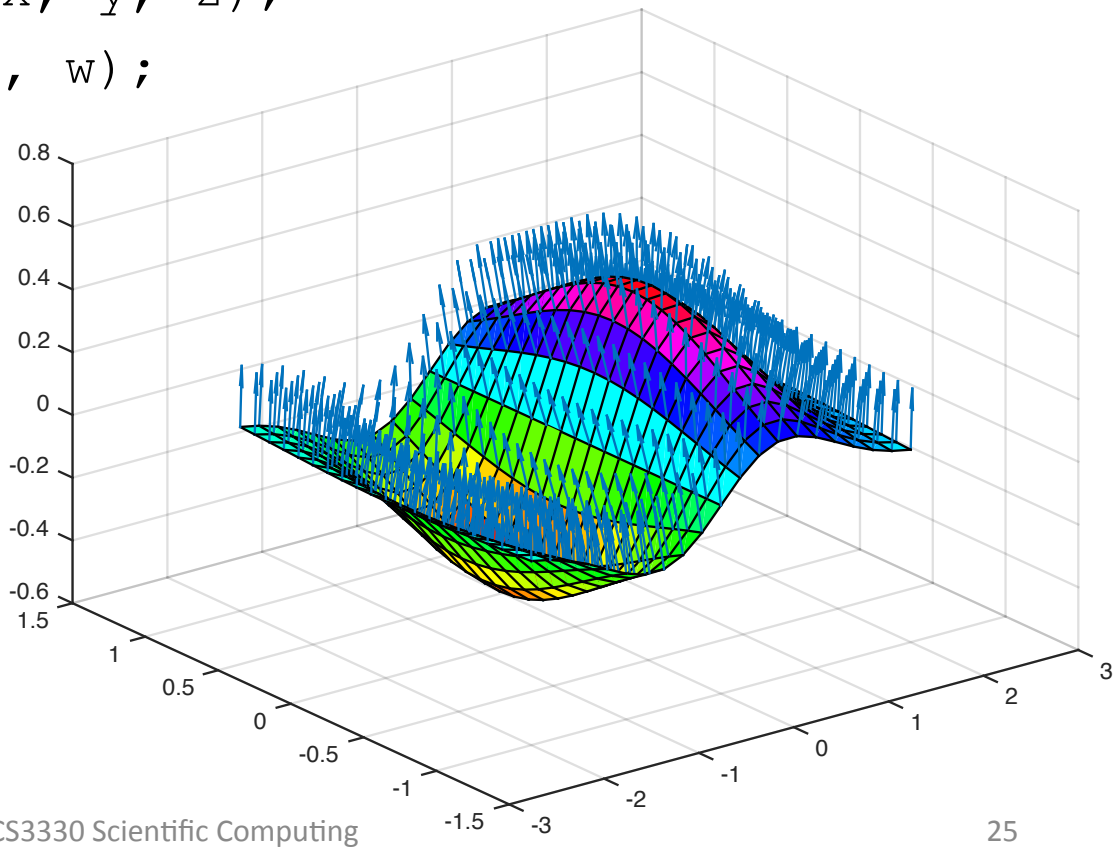
- Suitable for vector fields, such as wind vectors, electric field

```
[x, y, z] = peaks(20);
[u, v] = gradient(z);
quiver(x,y,u,v);
```



Quiver3 Plot

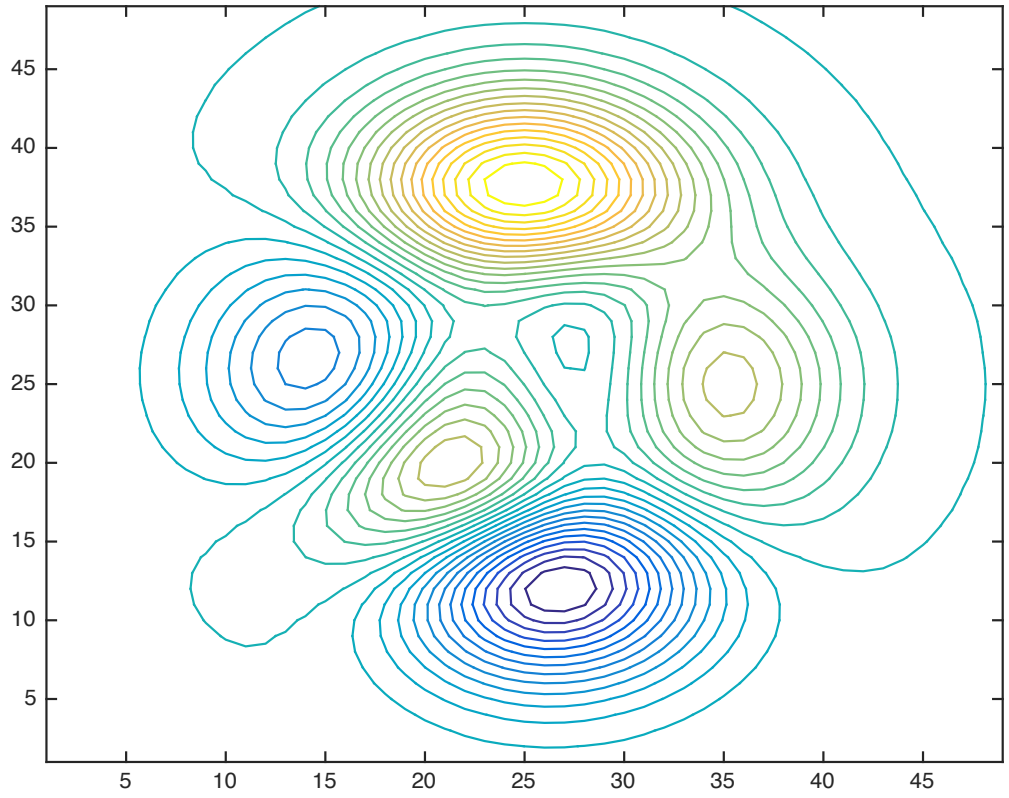
```
[x, y] = meshgrid(-2:0.2:2, -1:0.1:1);
z = x.*exp(-x.^2-y.^2);
[u, v, w] = surfnorm(x, y, z);
quiver3(x, y, z, u, v, w);
hold on;
surf(x, y, z)
```



Contour Plot

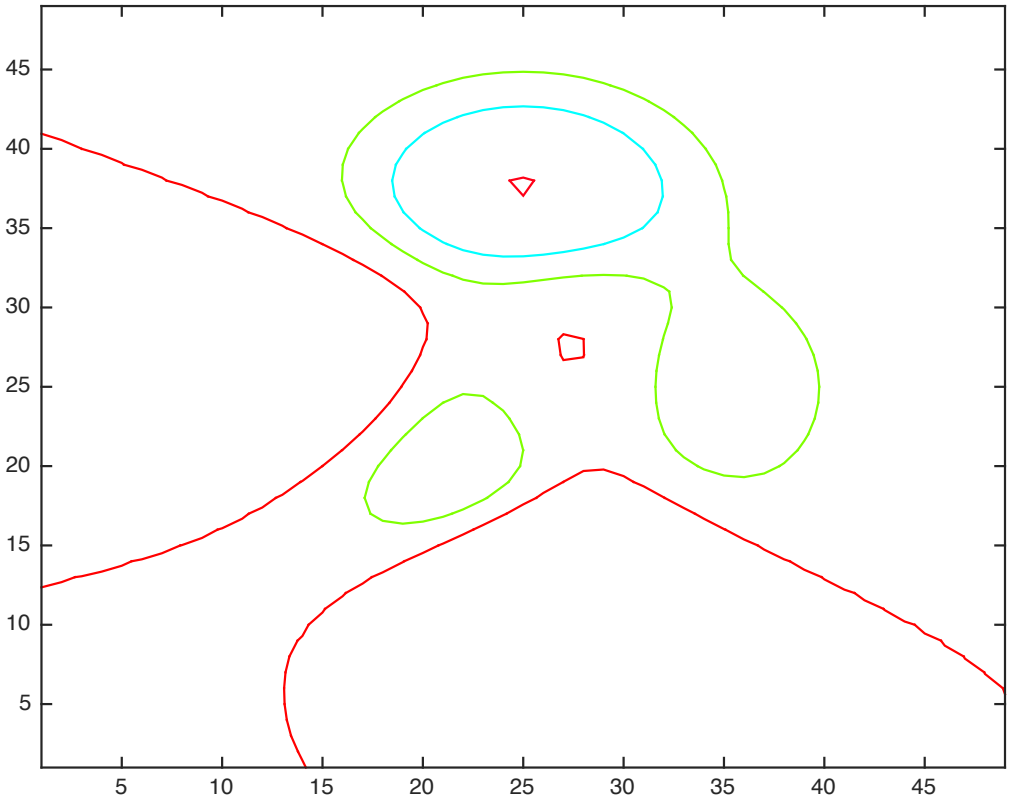
- X and y are column and row indices, and z is the value in the matrix

```
z = peaks;
contour(z, 30);
```



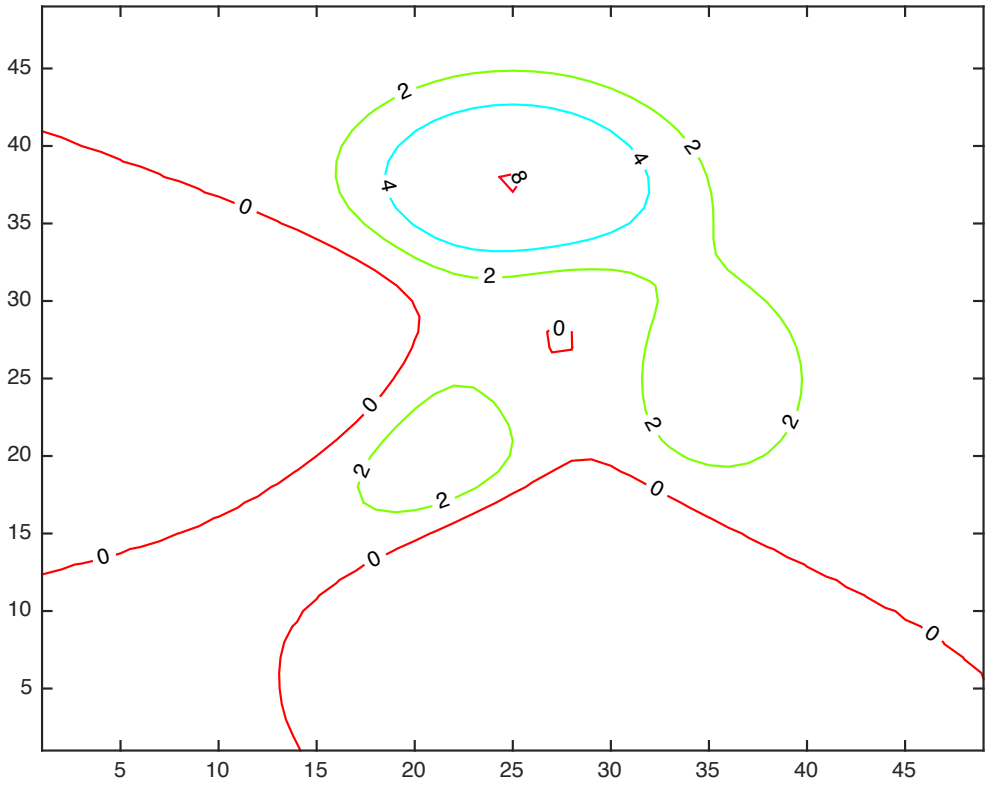
Specify the Levels

```
z = peaks;
contour(z, [0, 2, 4, 8]);
```



Label the Levels

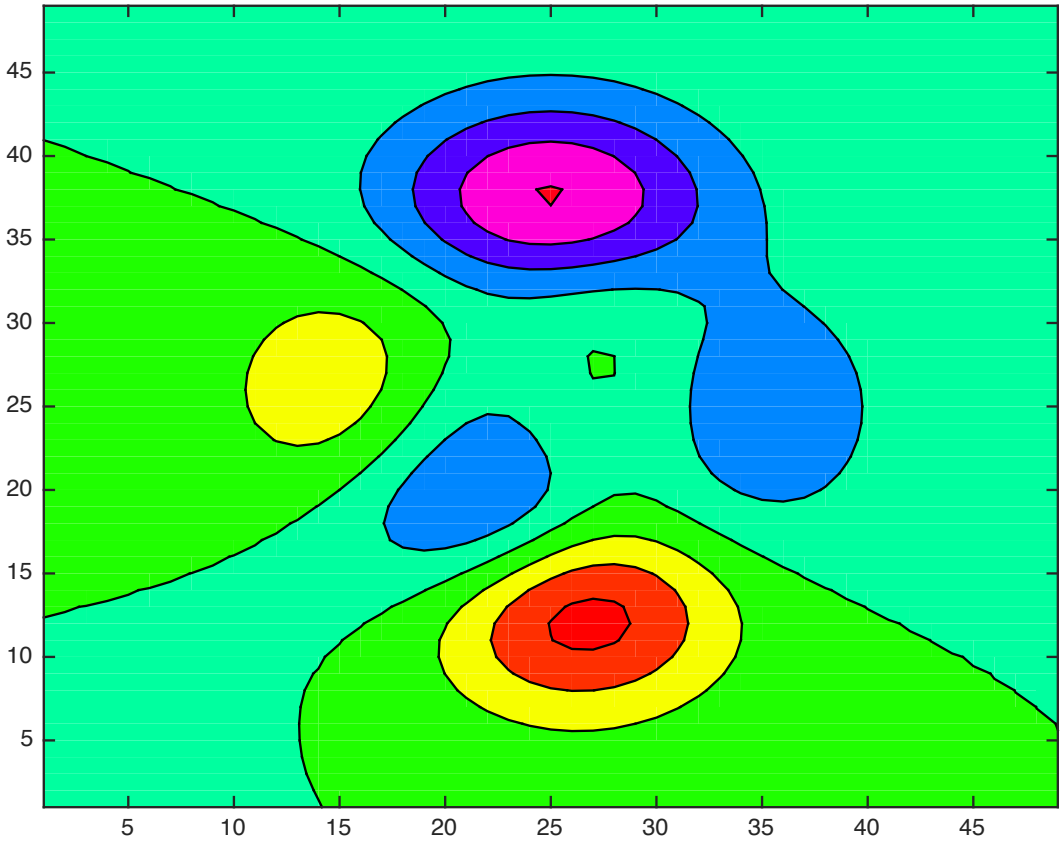
```
z = peaks;
[c,handle] = contour(z, [0, 2, 4, 8]);
clabel(c, handle);
```



Contourf Plot

- Fill in colors among levels

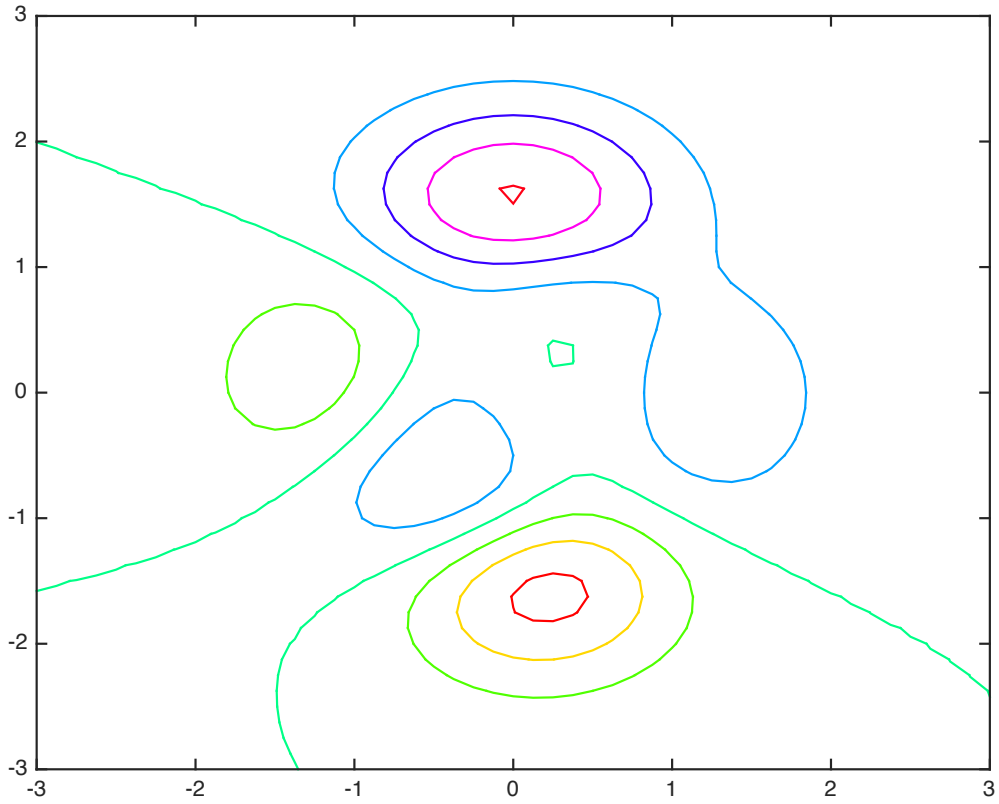
```
z = peaks;
contourf(z);
```



Alternative Way to Plot Contour

- Precisely specify z values

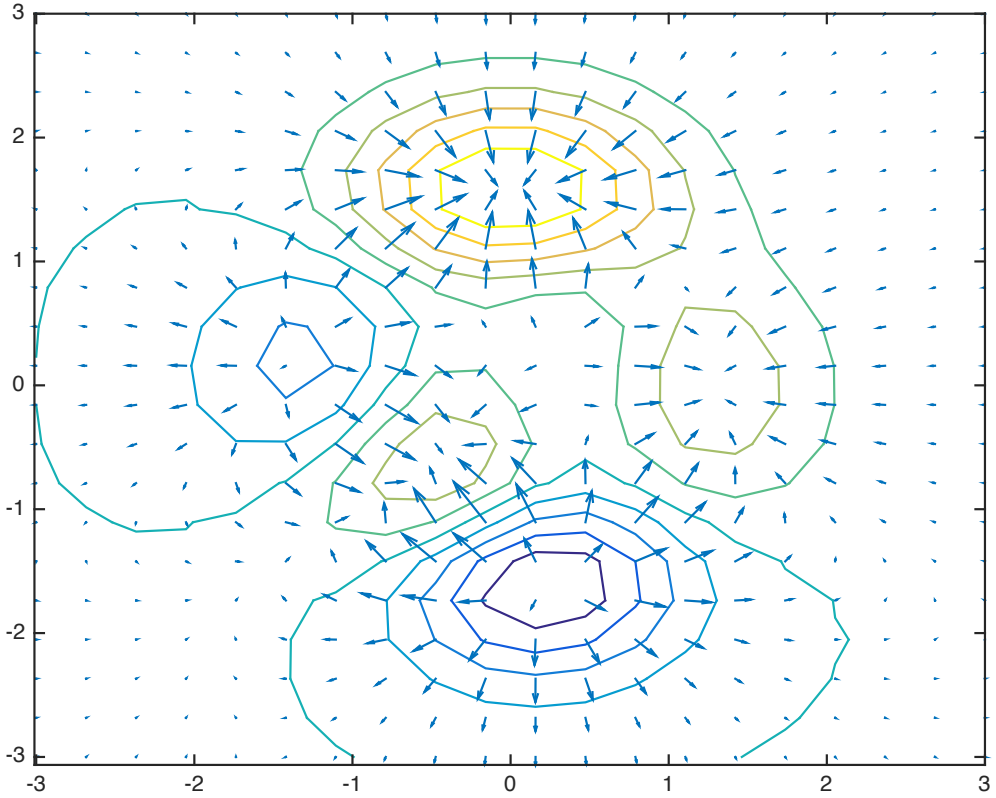
```
[x,y,z] = peaks;
contour(x, y, z);
```



Contour + Quiver

- Contour and gradient vectors are always orthogonal

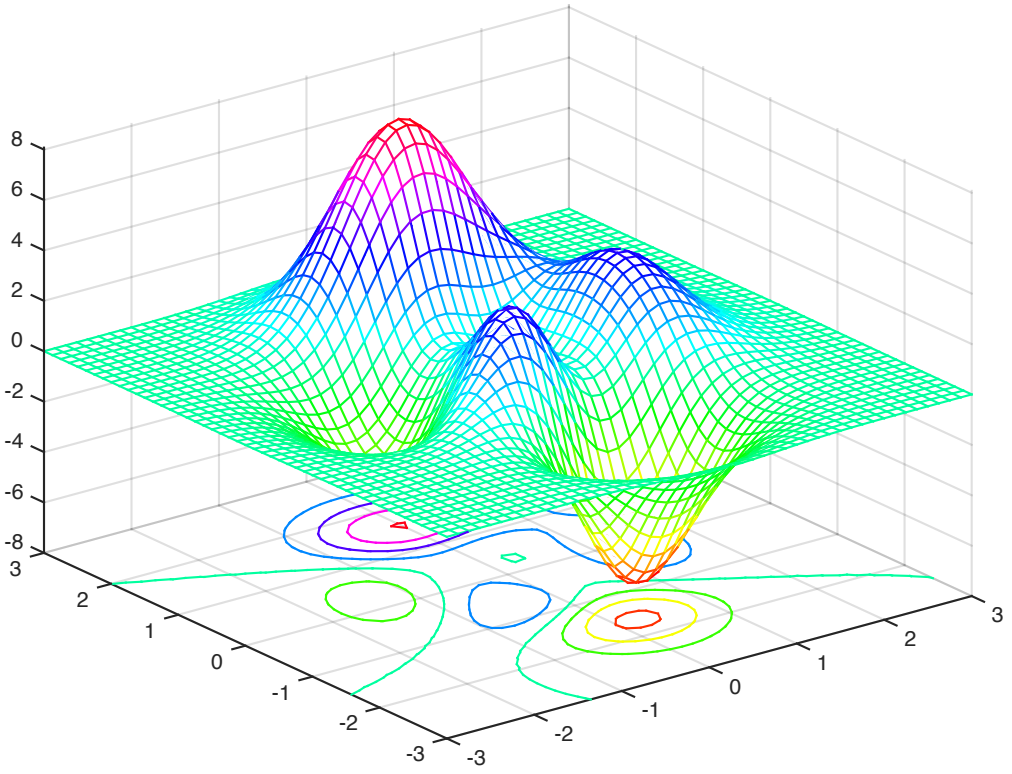
```
[x, y, z] = peaks(20);
[u, v] = gradient(z);
contour(x, y, z, 10);
hold on;
quiver(x,y,u,v);
```



Recap: Surfc and Meshc

- 3D plot with contour

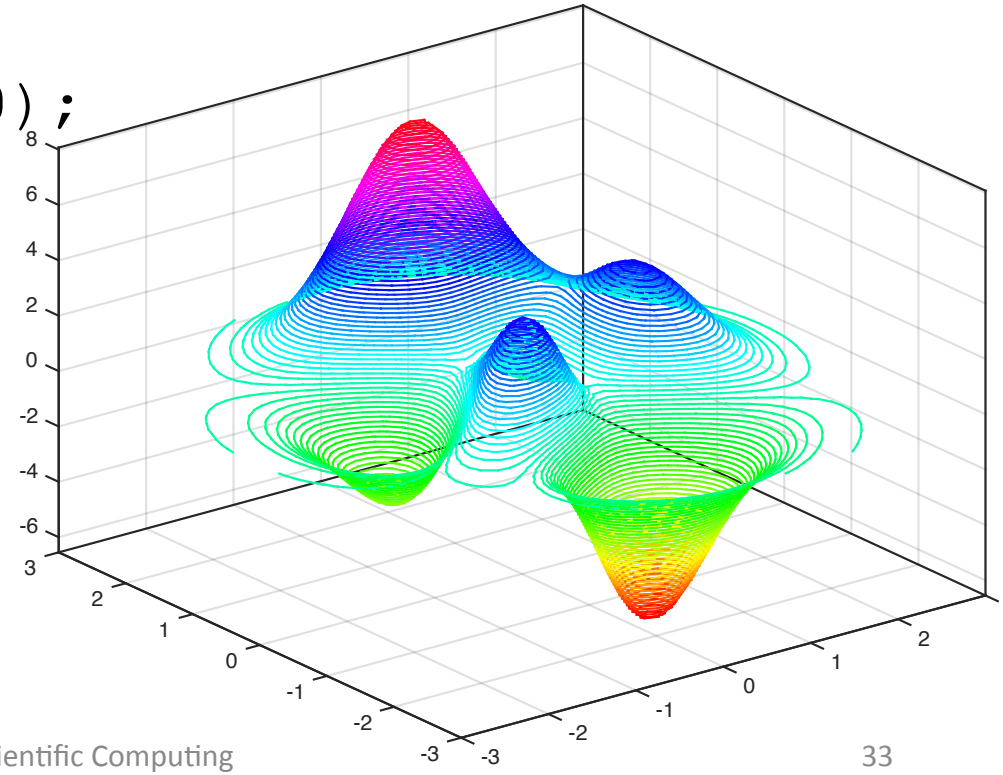
```
[x, y, z] = peaks;
meshc(x, y, z);
```



3D Contour

- What about contour in 3D?

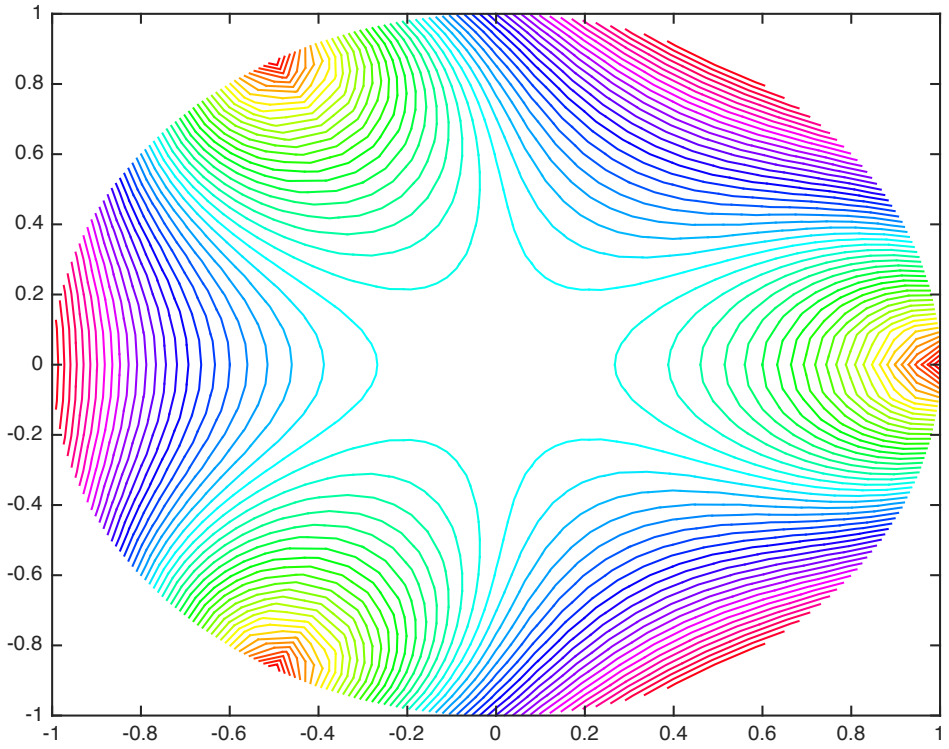
```
[x, y, z] = peaks;
contour3(x, y, z, 100);
```



Contour in Polar Coordinates

- Have to code it! For example: $f(z) = |z^3 - 1|$

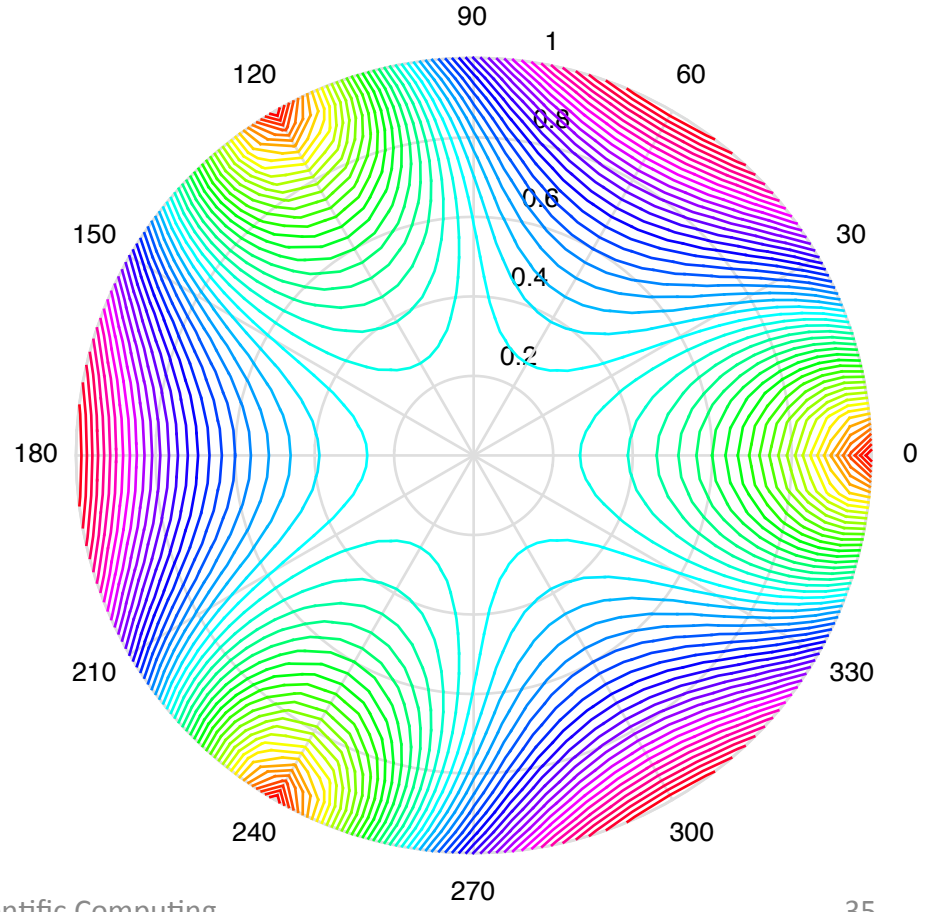
```
t = linspace(0, 2*pi, 61);
r = 0:0.05:1;
[tt, rr] = meshgrid(t, r);
[xx, yy] = pol2cart(tt, rr);
zz = xx + sqrt(-1)*yy;
ff = abs(zz.^3-1);
contour(xx, yy, ff, 50);
```



But Where are the Polar Axes

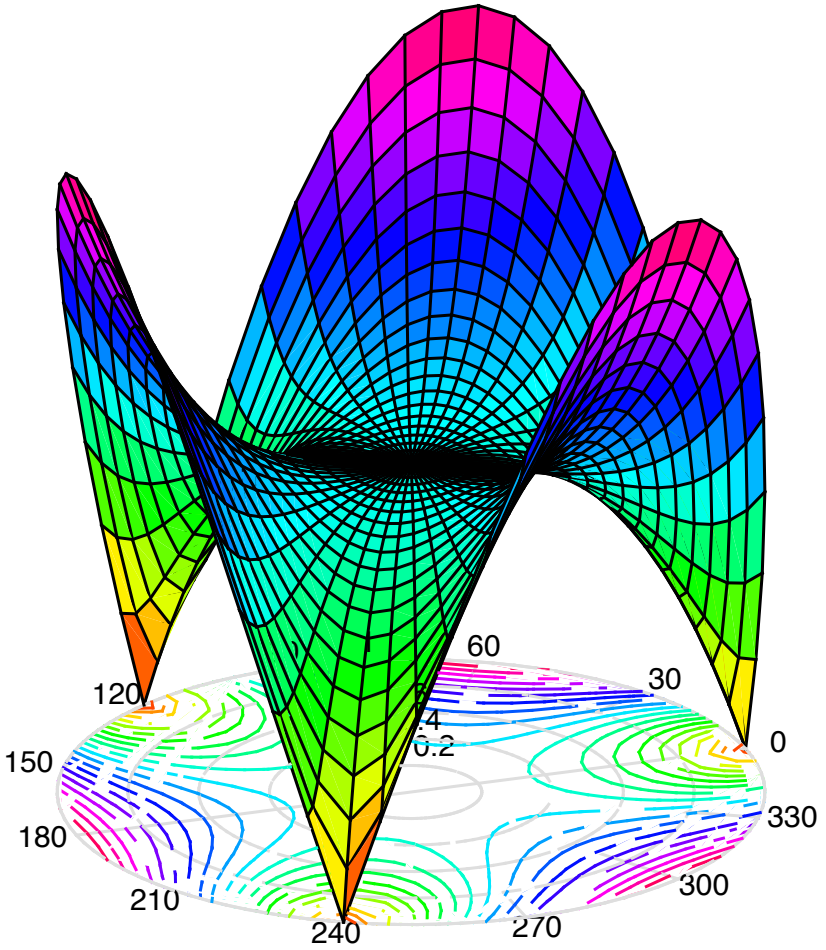
- Need to create them using polar command

```
h = polar([0 0], [0 1]);
delete(h);
hold on
contour(xx, yy, ff, 50);
hold off
```



Putting Everything Together

```
t = linspace(0, 2*pi, 61);
r = 0:0.05:1;
[tt, rr] = meshgrid(t, r);
[xx, yy] = pol2cart(tt, rr);
zz = xx + sqrt(-1)*yy;
ff = abs(zz.^3-1);
h = polar([0 2*pi], [0 1]);
delete(h);
hold on
contour(xx, yy, ff, 20);
surf(xx, yy, ff);
view(-19, 22);
```



Summary: More Plot Commands

- bar, barh, bar3, bar3h
- area
- pie, pie3
- stem, stem3
- stairs
- fill, fill3
- quiver, quiver3
- contour, contourf, contour3

Animations in Matlab

- There are two ways to generate animations (i.e., a sequence of frames essentially)
  - *Movie*: we pre-render multiple frames in memory beforehand, and then display them one after another ← non-realtime, memory hungry
  - *Object*: using graphic handle to manipulate objects ← realtime, but may not work for complicated animation due to computation power

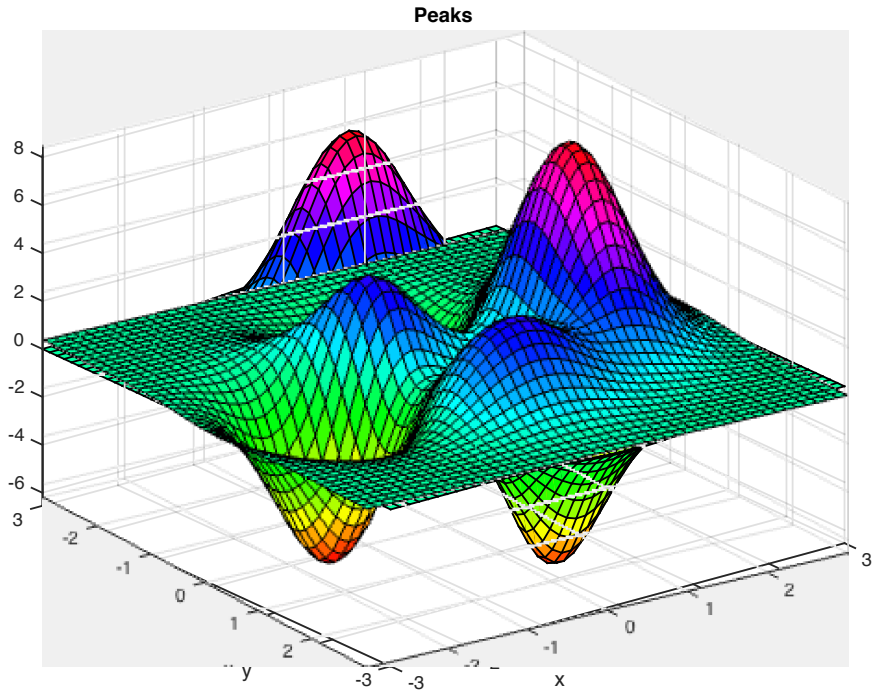
Generate Animation Using Movie

- Pre-rendered animation generation is done in two steps
- First, we plot figures, and use getframe(…) to capture the resulting figure into vector arraies ← the whole movie is a 2-D matrix
- Then, we use movie(...) to play the matrix ← the repeat times and speed can be adjusted

Changing the Viewing Angles

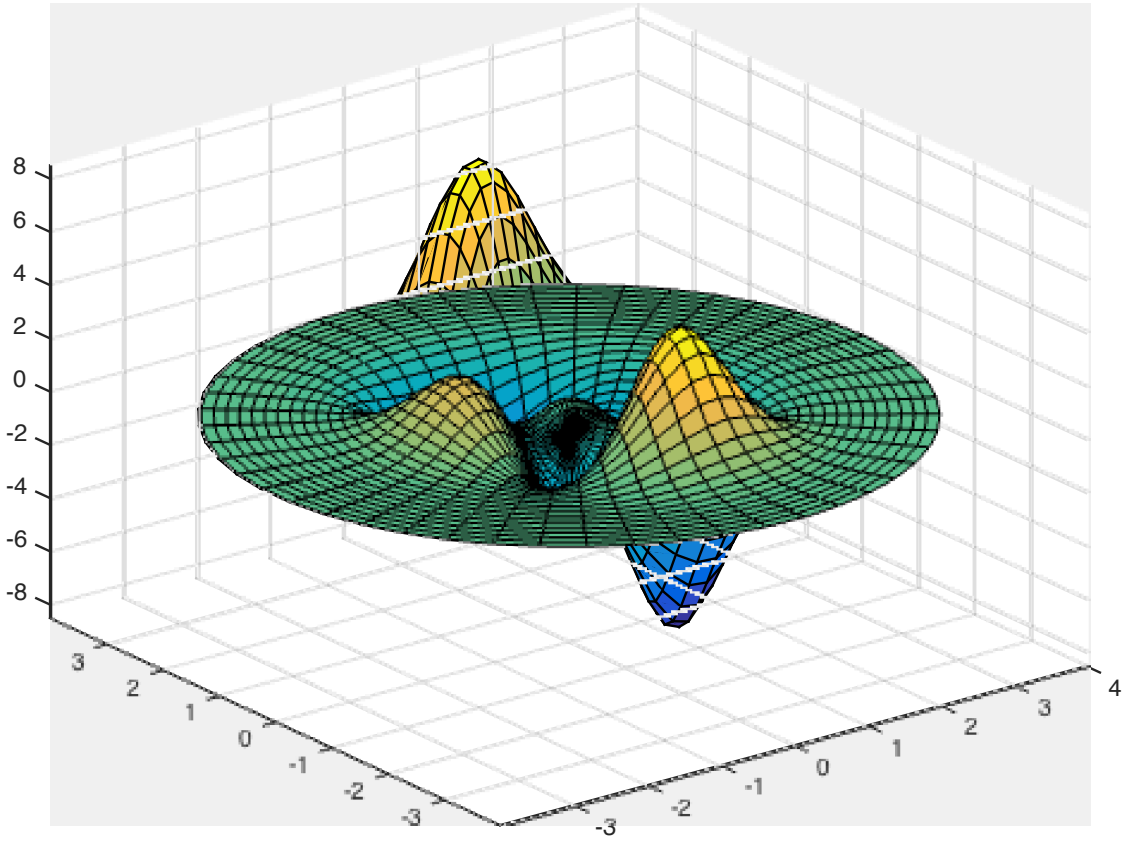
- Try it, and guess what do 3 and 1 do in the movie(…) command

```
n = 50;
colormap hsv;
peaks;
for i = 1:n
    view([-37.5+i*360/n, 30]);
   M(i) = getframe;
end
movie(M, 3, 1);
```



Scaling Z Values

```
clear
r=linspace(0, 4, 30);
t=linspace(0, 2*pi, 50);
[rr, tt]=meshgrid(r, t);
xx=rr.*cos(tt);
yy=rr.*sin(tt);
zz=peaks(xx,yy);
n = 30;
scale = cos(linspace(0, 2*pi, n));
for i = 1:n
        surf(xx, yy, zz*scale(i));
        axis([-inf inf -inf inf -8.5 8.5]);
        M(i) = getframe;
end
movie(M, 5);
```



Load and Display Images

- First save the image: http://s1.dmcdn.net/LVJQN.jpg
- Read the image into matrix X

```
>> X=imread('LVJQN.jpg'); display(size(X));
ans =    1080        1920           3
```

- What type of images it is? RGB…
- Display it using imshow(X) or image(X)

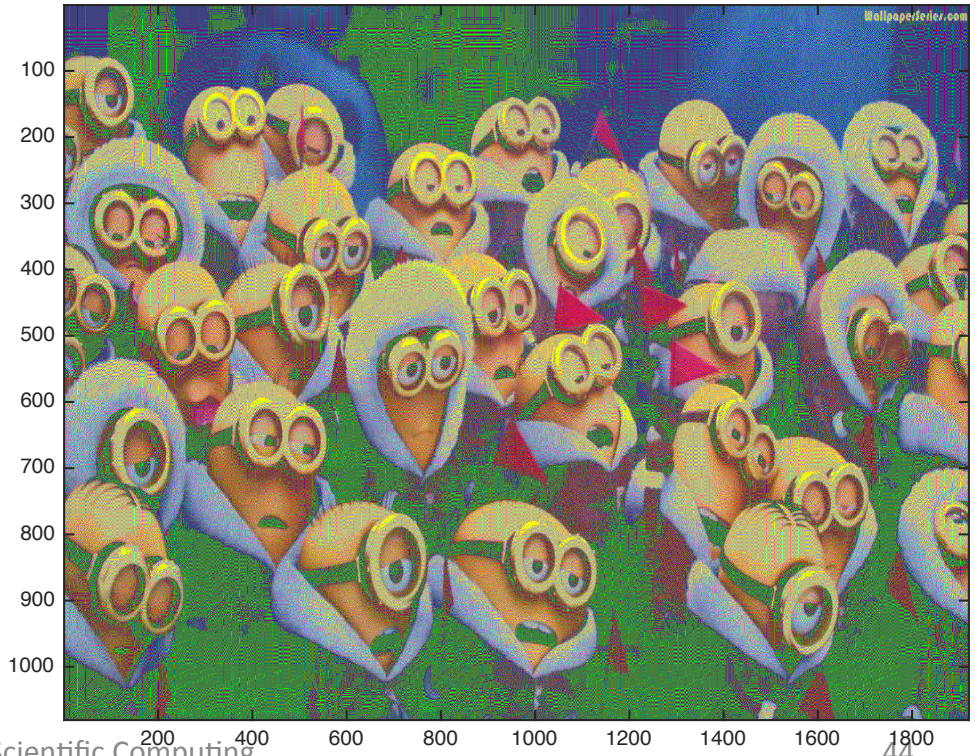
Load and Display Images (cont.)

Convert RGB into Indexed Images

- Try this:

```
colormap hsv;
I = rgb2ind(X, colormap);
image(I);
```

- Doesn't look good, Why?
- Have to find the best colormap!



Quantize the RGB Images

- Quantization is a process to convert continuous values into discrete ones (finite choices)

```
[I,map] = rgb2ind(X,256);
figure;
image(I);
colormap(map);
```

Changing Coloarmap

- Between positive and negative films

```
clear M
image(I);
colormap(map);
n = 30;
for i = 1:n
   colormap(((i-1)*(1-map)+(n-i)*map)/n);
   M(i) = getframe;
end
movie(M, -5);
```

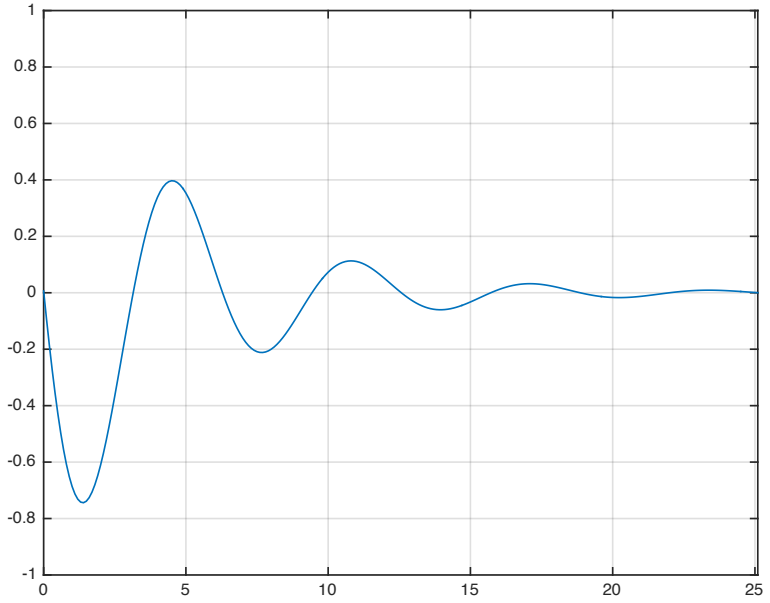
Generate Animation Using Object

- Real-time rendering using object
- Curve is an object, with the following attributes
  - xdata: x values of samples
  - ydata: y values of samples
- Use curve's handle to adjust the value
- Use drawnow command to update the curve

Decaying Sin Wave

- Function: $y = \sin(x-k)e^{-\frac{x}{5}}$

```
x = 0:0.1:8*pi;
h = plot(x, sin(x).*exp(-x/5));
axis([-inf inf -1 1]);
grid on
for i = 1:50000
   y = sin(x-i/50).*exp(-x/5);
   set(h, 'ydata', y);
   drawnow
End
```

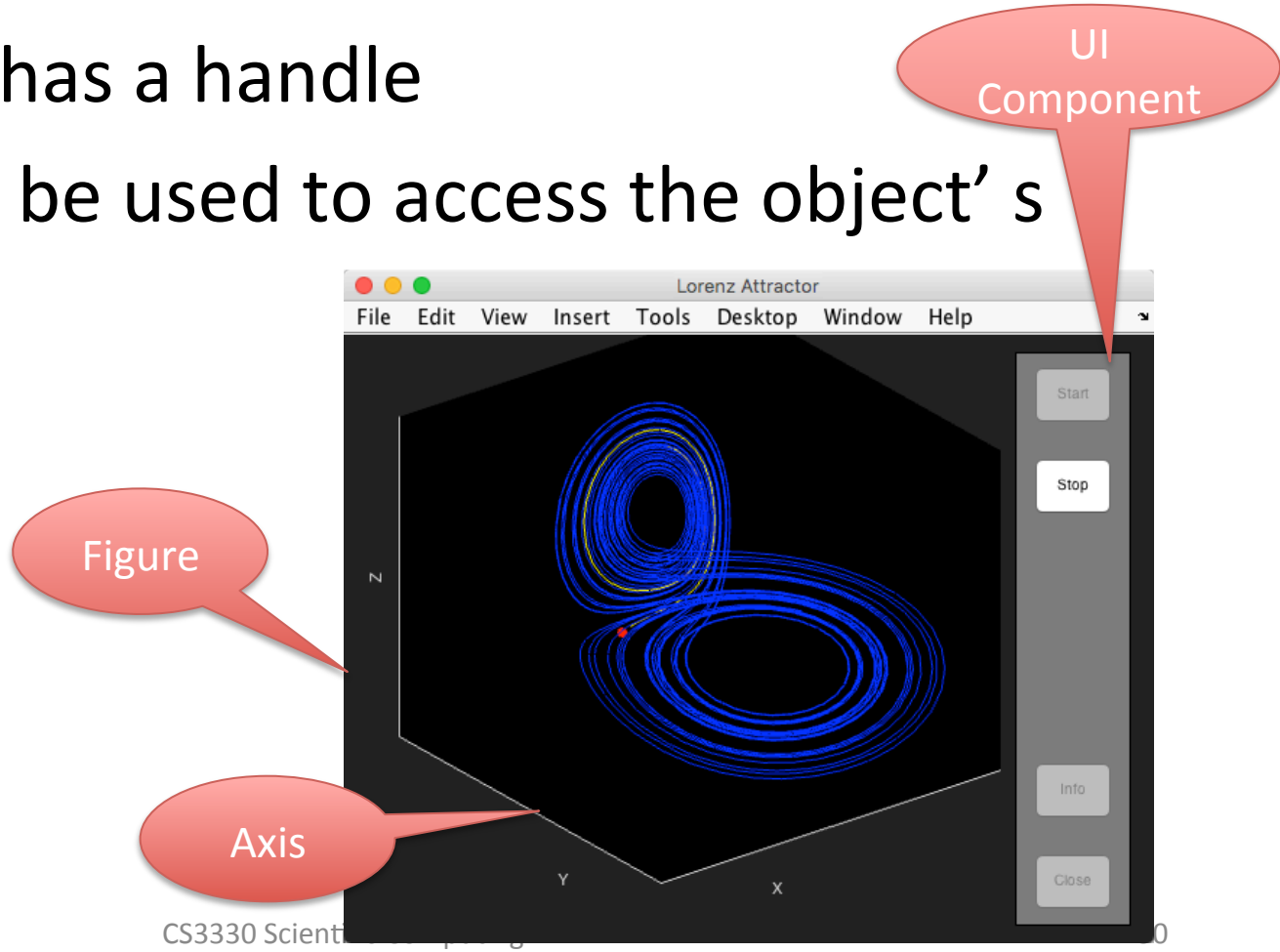


Some Animation Demos in Matlab

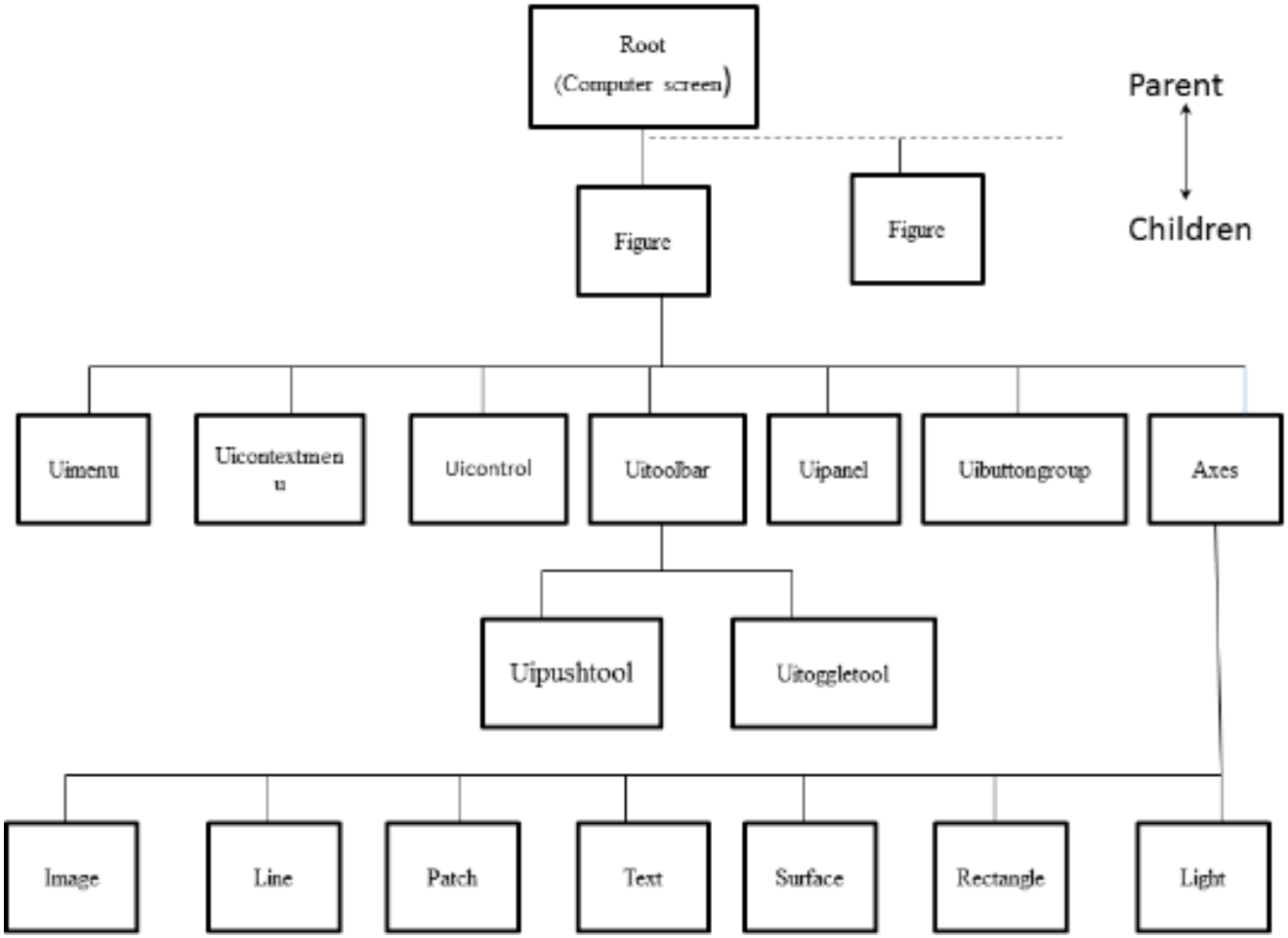
- lorenz: Animated Lorenz chaotic attractor
- truss: bending bridge animation
- travel: solving the traveling salesman problem using a randomized algorithm
- fitdemo: nonlinear curve fitting demo

Handle Graphics

- Each graphic component is an object
- Each object has a handle
- Handles can be used to access the object' s attributes



Hierarchy of Graphics Components



Two Ways of Access Attributes

- Using GUI
  - View → Property Editor → More Properties
  - Command: propedit
- Using commands
  - set: set the value of an attribute
  - get: get the current value of an attribute
  - findobj: finding the right handle

Recap: Set Attribtues

```
t = 0:0.1:4*pi;
y = exp(-t/5).*sin(t);
h = plot(t, y);
set(h, 'Linewidth', 3);
set(h, 'Marker', 'o');
set(h, 'MarkerSize', 20);
```

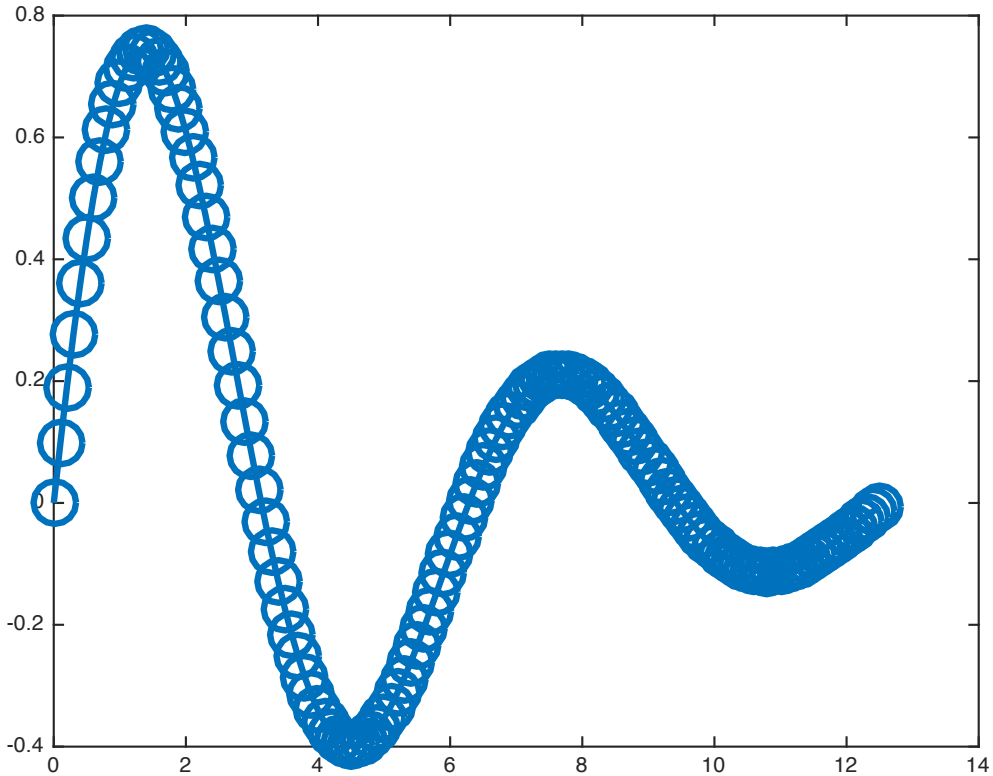


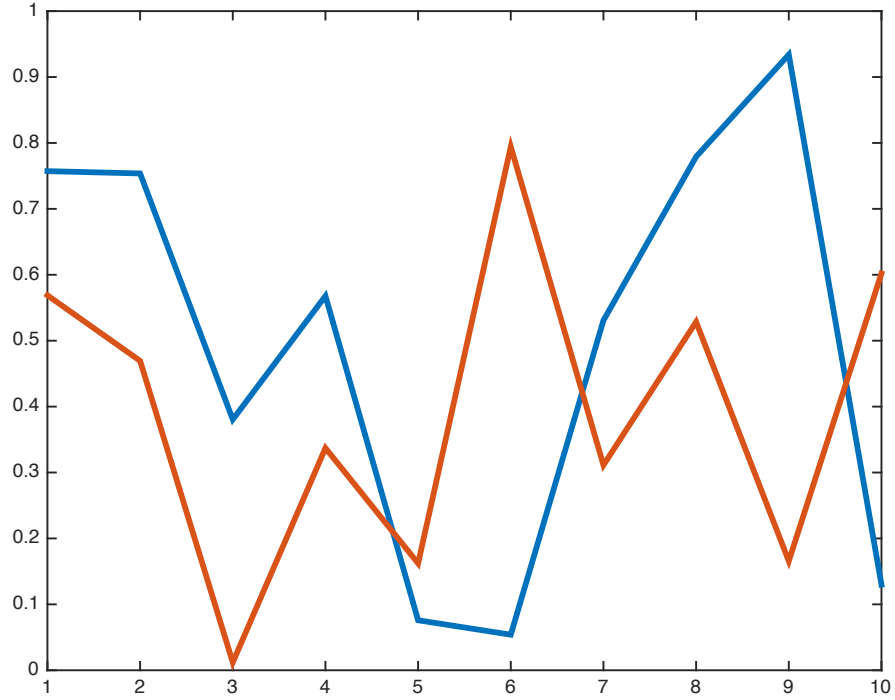
Figure Out All Attributes

```
>> set(h)
   AlignVertexCenters: {'on'  'off'}
          BusyAction: {'queue'  'cancel'}
        ButtonDownFcn: {}
            Children: {}
            Clipping: {'on'  'off'}
               Color: {1x0 cell}
           CreateFcn: {}
           DeleteFcn: {}
         DisplayName: {}
.......
```

How to Use Findobj

- E.g., h=findobj(gcf, 'type', 'line') ← find all the lines in the focused figure

```
plot(rand(10,2));
h=findobj('type', 'line') ;
set(h, 'LineWidth', 3);
```

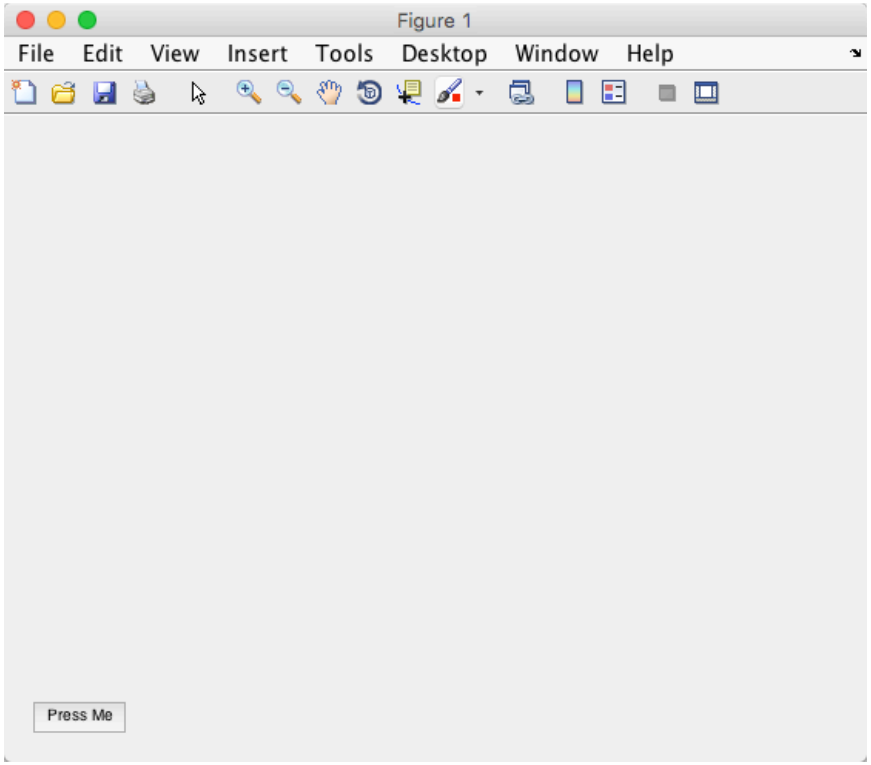


GUI Design

- Similar to what we saw in earlier demos
  - Uicontrol
  - Mouse Events
- Uicontrol creates various UI components
  - Push button, sliding bar, radio button, frame, check box, edit box, list menu, popup menu

Uicontrol Example

```
h = uicontrol;
set(h, 'String', 'Press Me');
cmd = 'fprintf(''Button Pressed!\n''); ';
set(h, 'Callback',cmd);
```

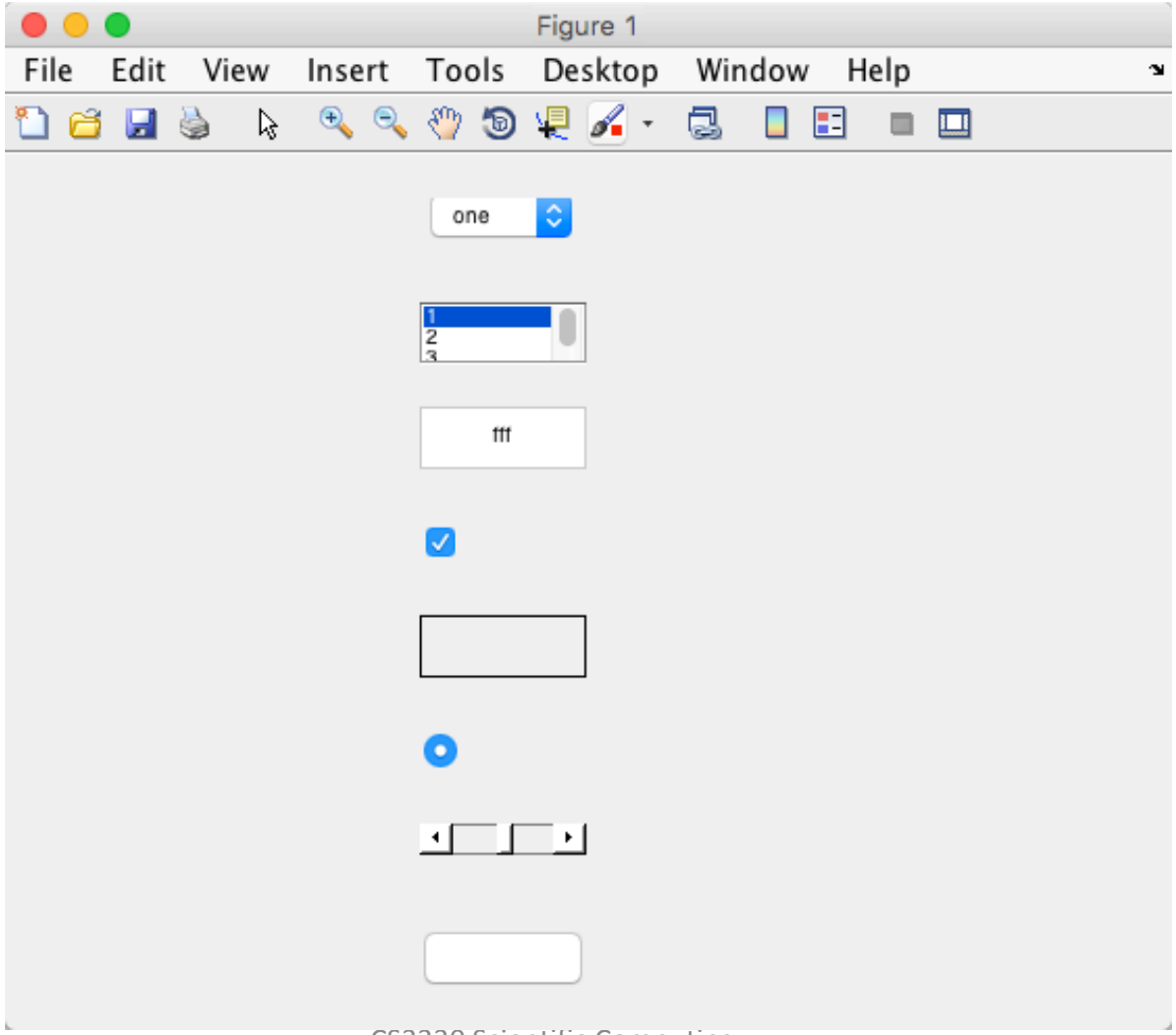


Other Uicontrol Components

- Exercise: Try this in your own Matlab

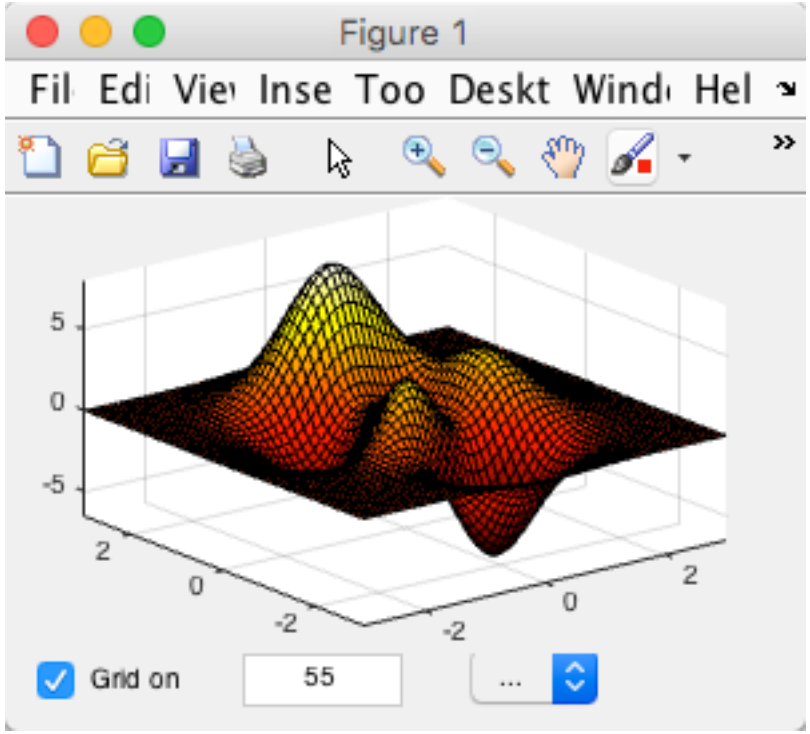
```
uicontrol('style', 'push', 'position', [200  20 80 30]);
uicontrol('style', 'slide', 'position', [200 70 80 30]);
uicontrol('style', 'radio', 'position', [200 120 80 30]);
uicontrol('style', 'frame', 'position', [200 170 80 30]);
uicontrol('style', 'check', 'position', [200 220 80 30]);
uicontrol('style', 'edit', 'position', [200 270 80 30]);
uicontrol('style', 'list', 'position', [200 320 80 30],
'string', '1|2|3|4');
uicontrol('style', 'popup', 'position', [200 370 80 30],
'string', 'one|two|three');
```

Other Uicontrol Components (cont.)



A Complete Example: ui02.m

- Download, execute, and read the code from: http://nmsl.cs.nthu.edu.tw/images/courses/CS3330_2015/ui02.m



Mouse Events

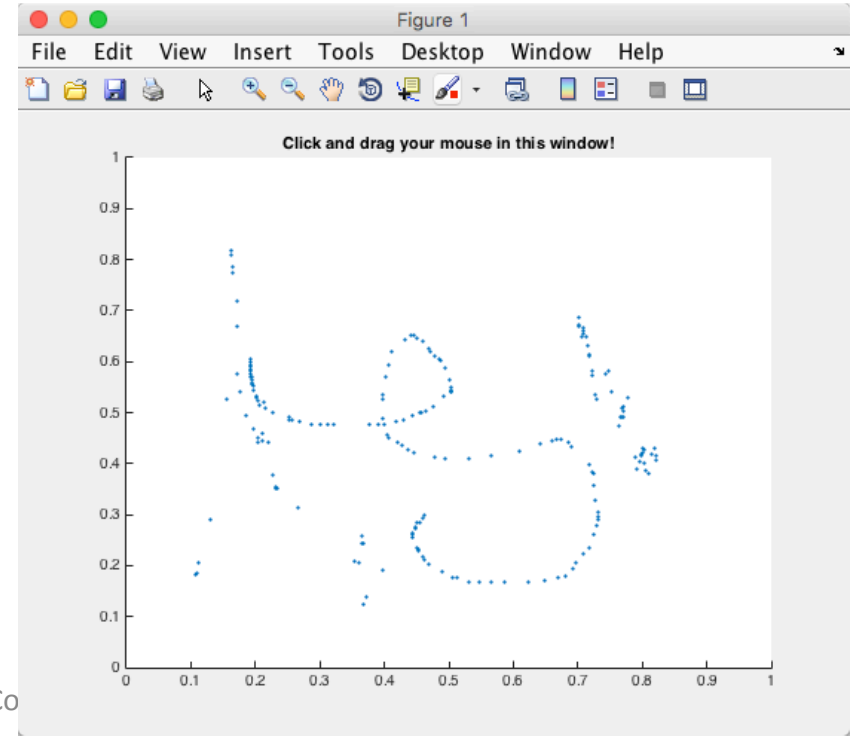
- Main mouse events for callback functions
  - WindowButtonDownFcn: invoked when the button is pressed
  - WindowButtonUpFcn: invoked when the button is released
  - WindowButtonMotionFcn: invoked when the mouse is moving (with and without the button is pressed)

Design of A Drawing Canvas

- High-level idea is
  - When mouse button is pressed: install callback functions for button release and mouse move
  - When mouse button is release: uninstall the two callback functions

A Complete Example: mouse01.m

- Download, execute, and read the code from: http://nmsl.cs.nthu.edu.tw/images/courses/CS3330_2015/mouse01.m



Matlab #3 Homework (M3)

1. (3%) Write a single .m function to realize the following features in one Matlab GUI. Please explain and demo your code to the TAs.
   - A large canvas in which each mouse click add a star on it; and a button to clear the canvas
   - A UI component to allow user to choose color of stars; at least 6 colors should be supported
   - A slider to allow users rotate the canvas in the unit of degrees