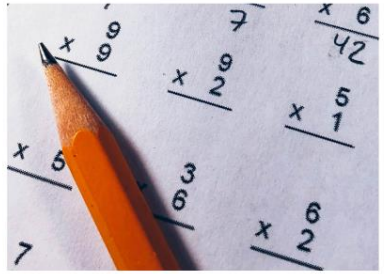
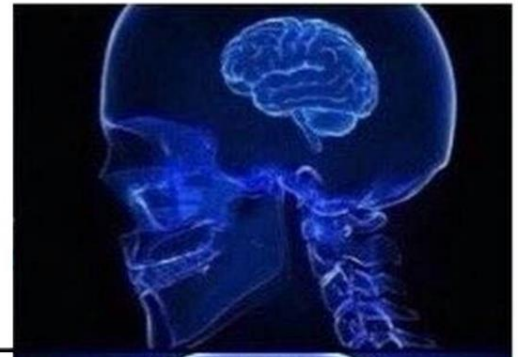
MATH2221
Mathematics Laboratory II

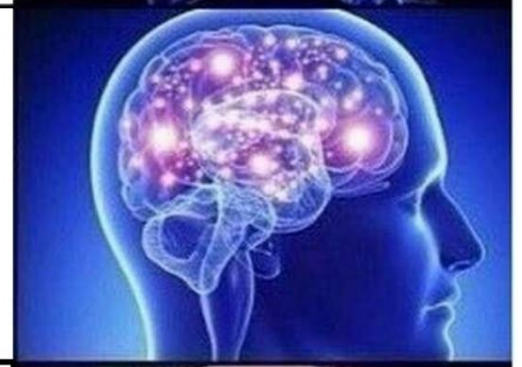
Lecture 6: 2D and 3D Visualization Using MATLAB

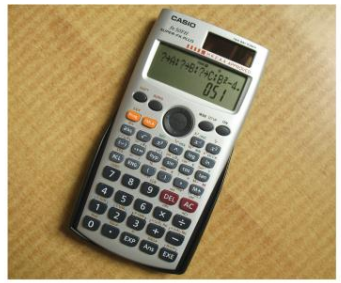
Gary Choi

February 18, 2025









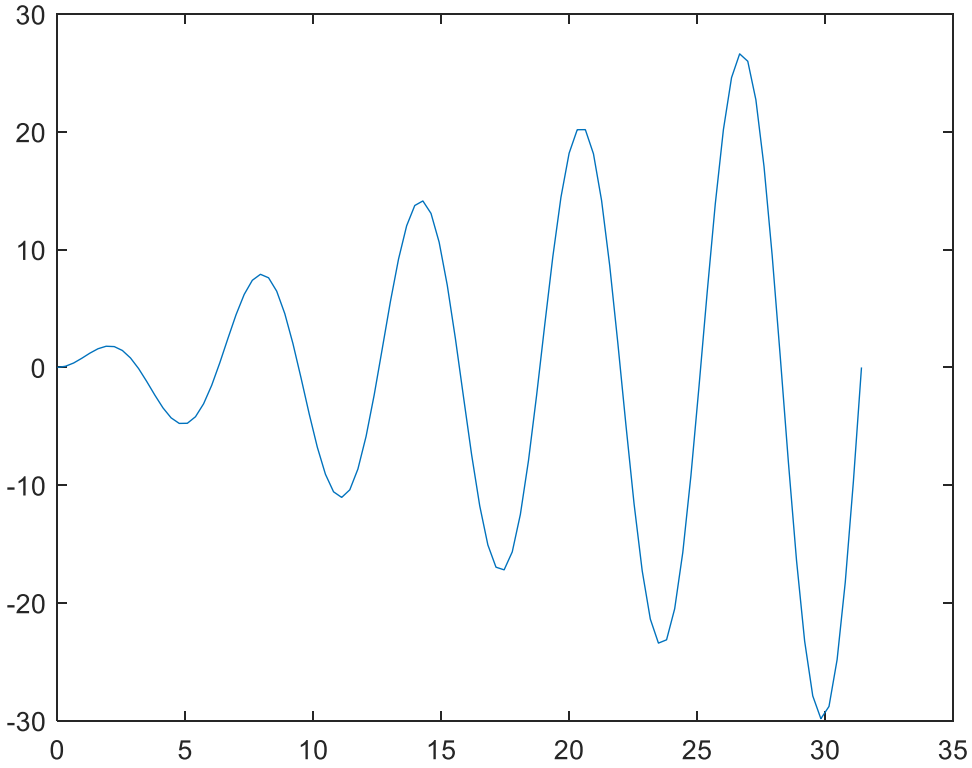
MATLAB

Recall: Introduction to 2D visualization using MATLAB

- Basic commands:
  - figure: create a new figure window
  - plot(x,y): plot the points x,y (can be vectors) in the current figure window
  - Use cursor to zoom in/out, drag etc.

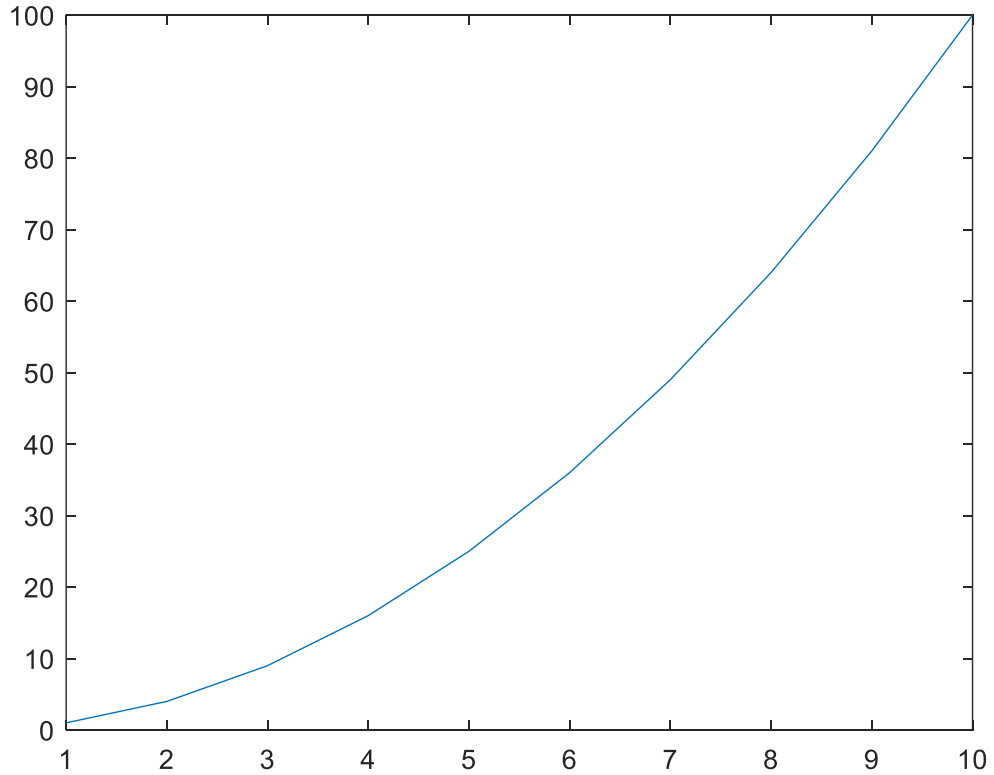
Example:

```
t = linspace(0,10*pi,100);
figure;
plot(t, t.*sin(t));
```



Example:

```
x = 1:10;
y = x.^2;
figure;
plot(x,y);
```

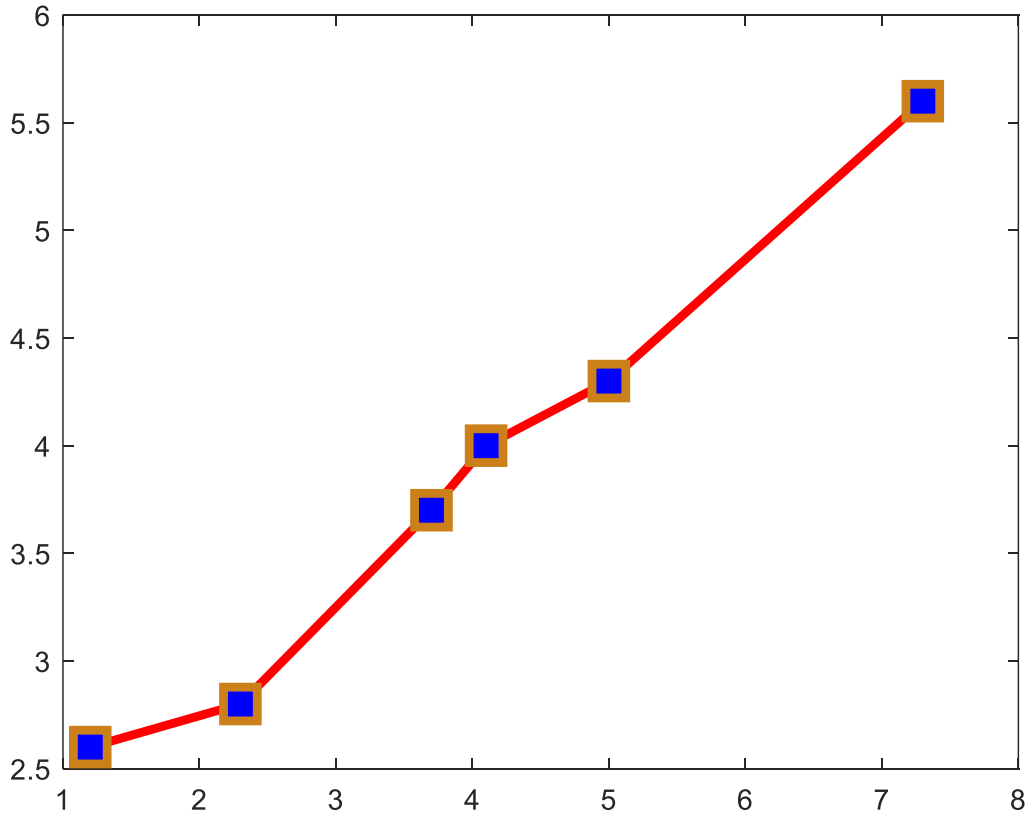


Recall: The *plot* function with different style options

- Adjusting:
  - Line styles ('-', '--', ':' etc.)
  - Marker styles ('o', 's', 'diamond' etc.)
  - Colors ('r', 'g', 'b' etc.)
  - Marker size
  - Line width
  - Marker Edge Color
  - Marker Face Color
  - …

Example:

```
plot(x,y,'rs-','MarkerSize',15, ...
    'MarkerFaceColor','b', ...
    'LineWidth', 3, ...
    'MarkerEdgeColor',[0.8,0.5,0.1]);
```



Recall: Plotting multiple sets of data on the same plot

- Method 1: use plot(X1,Y1,LineSpec1,...,Xn,Yn,LineSpecn)
  - Put everything in the same plot command one by one

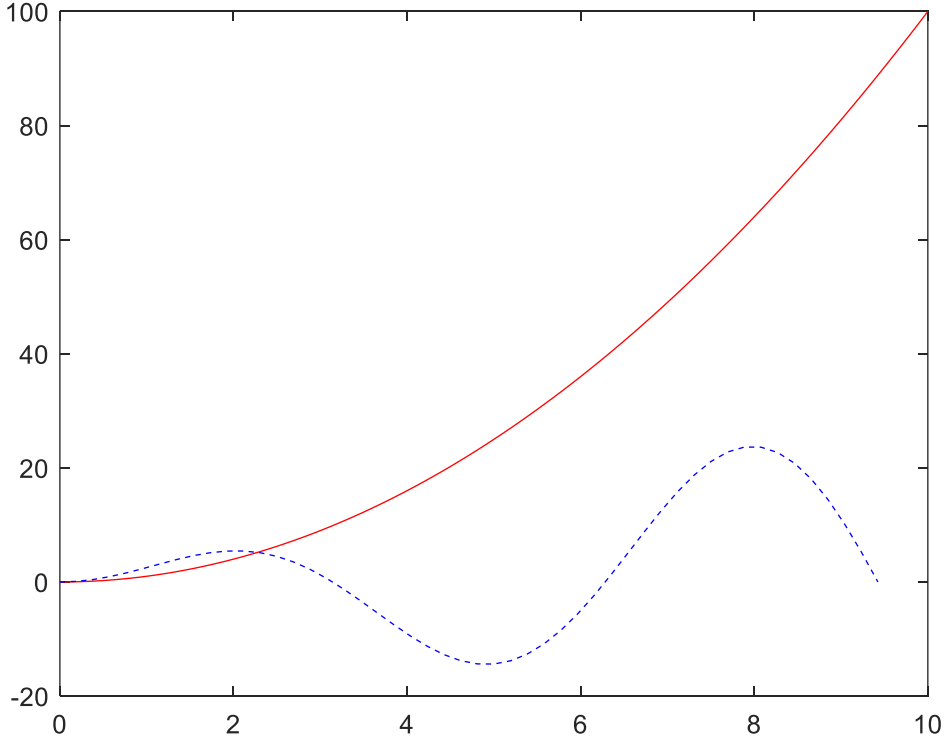
Example:

```
x = 0:0.1:10;
y = x.^2;
t = linspace(0,3*pi,50);
z = 3*t.*sin(t);
figure;
plot(x,y,'r-',t,z,'b--');
```

- Method 2: use hold on
  - To plot two or more graphs in one window
  - To unhold the windows, use hold off

Example:

```
figure;
plot(x,y,'r-');
hold on;
plot(t,z,'b--');
```

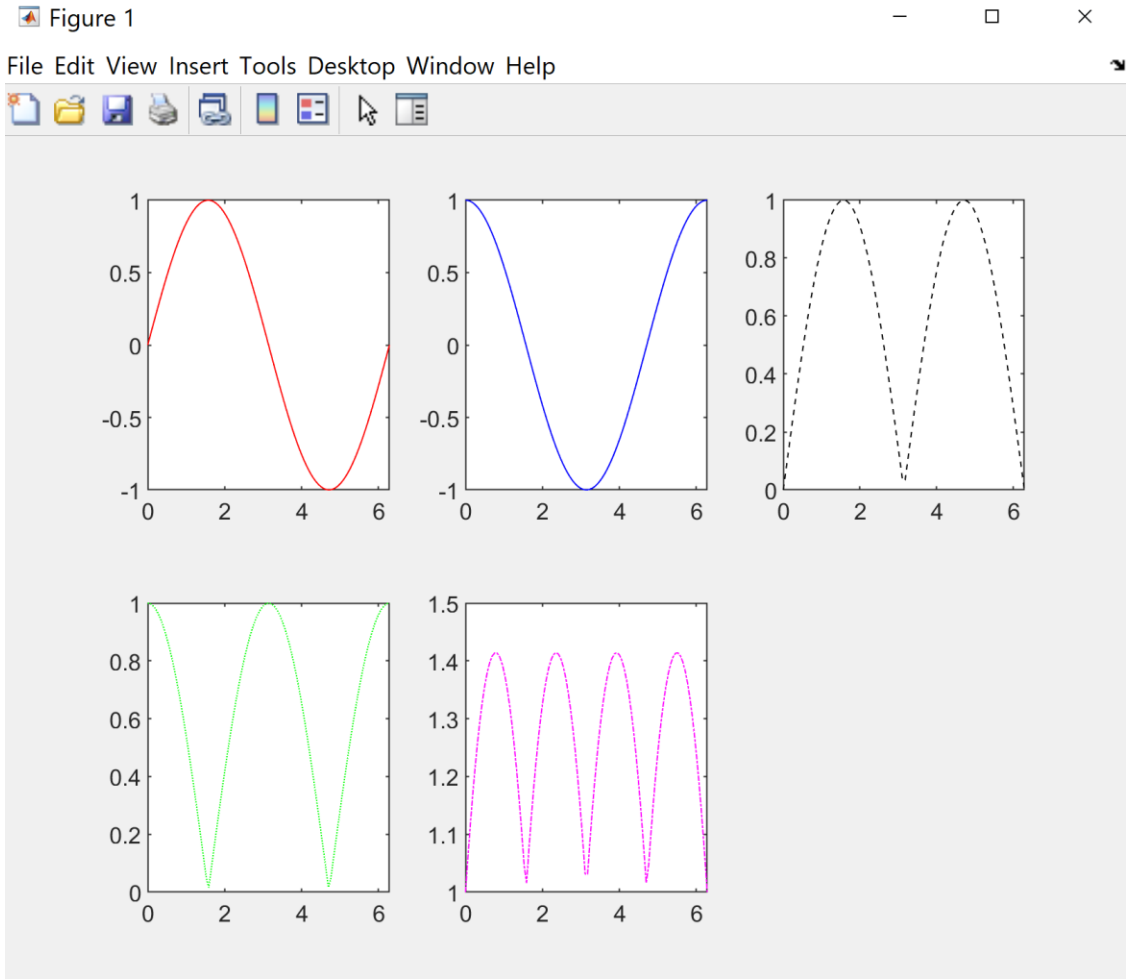


Recall: The *subplot* function

- subplot(m,n,p):
  - Divide the current figure into an m-by-n grid
  - p is the ID of the grid in which you want to put the plot

Example:

```
figure;
x = linspace(0,2*pi,100);
subplot(2,3,1);
plot(x, sin(x), 'r-');
subplot(2,3,2);
plot(x, cos(x), 'b-');
subplot(2,3,3);
plot(x, abs(sin(x)), 'k--');
subplot(2,3,4);
plot(x, abs(cos(x)), 'g:');
subplot(2,3,5);
plot(x, abs(sin(x))+abs(cos(x)), 'm-.');
```

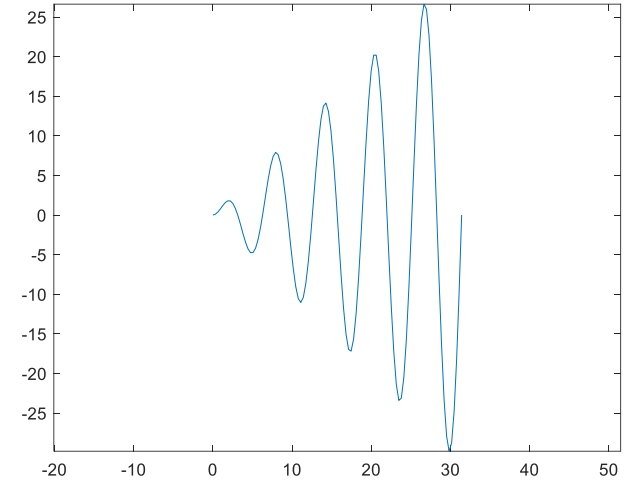


Recall: Adjusting the axes

- Some useful commands:
  - axis on: show the axes (default)
  - axis off: hide the axes
  - axis equal: make the two axes equal in ratio
  - axis tight: set axis limit as the range of the data
  - axis([xmin,xmax,ymin,ymax]): set the axis limits
  - xlim([xmin,xmax]): set the x-axis limit
  - ylim([ymin,ymax]): set the y-axis limit

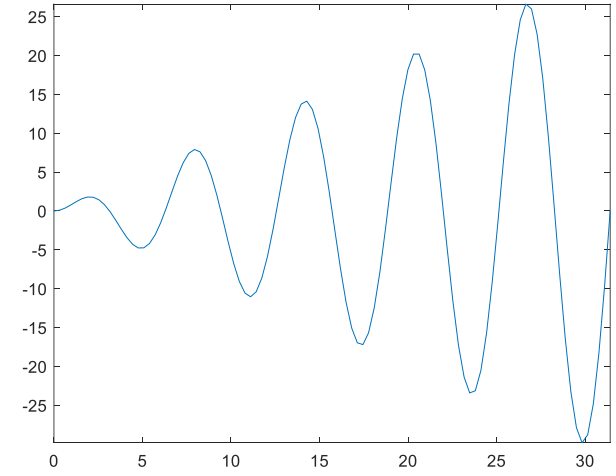
Example:

```
t = linspace(0,10*pi,100);
figure; plot(t,t.*sin(t));
axis equal;
```



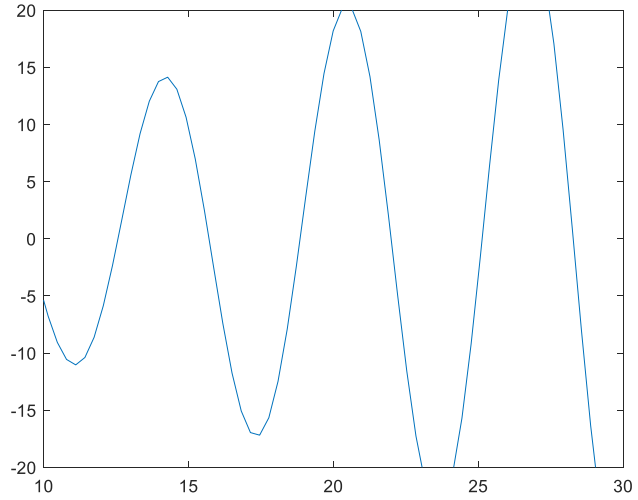
Example:

```
t = linspace(0,10*pi,100);
figure; plot(t,t.*sin(t));
axis tight;
```



Example:

```
t = linspace(0,10*pi,100);
figure; plot(t,t.*sin(t));
axis([10, 30, -20, 20])
```

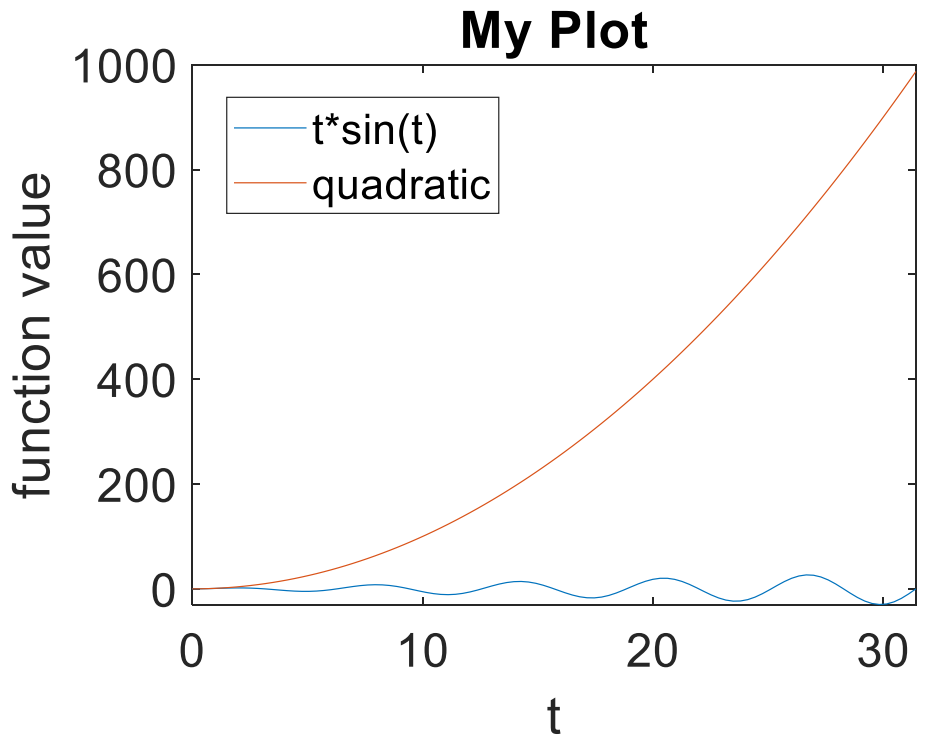


Recall: Adding title, axis labels, and figure legends

- Some useful commands:
  - title('string'): add figure title
  - xlabel('string'): label the x-axis
  - ylabel('string'): label the y-axis
  - legend('string'): label each set of data on the plot
  - set(gca,'FontSize',k): adjust the font size of the title and labels (default k = 10)

Example:

```
t = linspace(0,10*pi,100);
figure;
plot(t,t.*sin(t));                  % the first curve
hold on;
plot(t, t.^2);                      % the second curve
title('My Plot')                    % title
xlabel('t')                         % x-label
ylabel('function value')            % y-label
legend('t*sin(t)','quadratic')      % label both curves
set(gca,'FontSize',20)              % Increase the font size
```



Recall: An overview of useful plotting commands

- Graph generation commands
  - Plot different types of graphs
  - Standard xy plot: plot
  - Semilog plots: semilogx, semilogy
  - loglog plots: loglog
- Management commands
  - Manage the figure windows
  - Create a new window: figure
  - Divide a window into subplots: subplot
  - Plot multiple curves on the same plot: hold (on/off)
- Annotation commands
  - Format the graphs
  - Add title and axis labels: title, xlabel, ylabel
  - Add text and figure legend: text, legend
  - Add box and grid lines: box (on/off), grid (on/off)

Graph generation	Management	Annotation
plot	figure	title
semilogx semilogy	hold on hold off	xlabel ylabel
loglog	subplot	text
polar/ polarplot	close	grid
fill	axis	legend
histogram	view	box
pie	rotate	set

Adding text to graphs

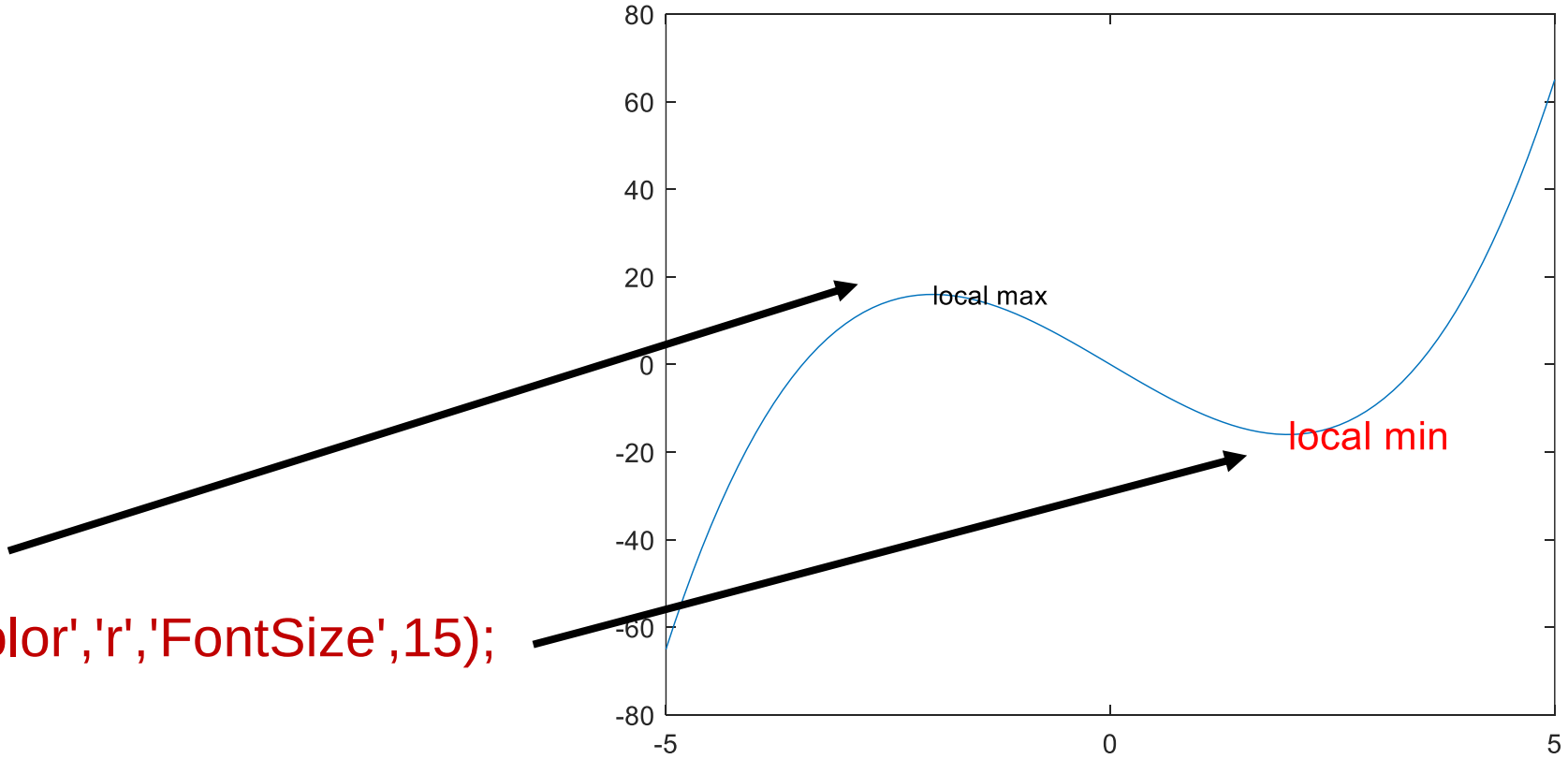
- We can add text to any position of the plot using text
  - text(x,y,txt): add the string txt to the position (x,y)
  - text(x,y,txt, 'Color','r','FontSize',16): more style options

- Example:

```
x = linspace(-5,5,100);
y = x.^3-12*x;

figure;
plot(x,y);
hold on;

text(-2,16,'local max');
text(2,-16,'local min','Color','r','FontSize',15);
```

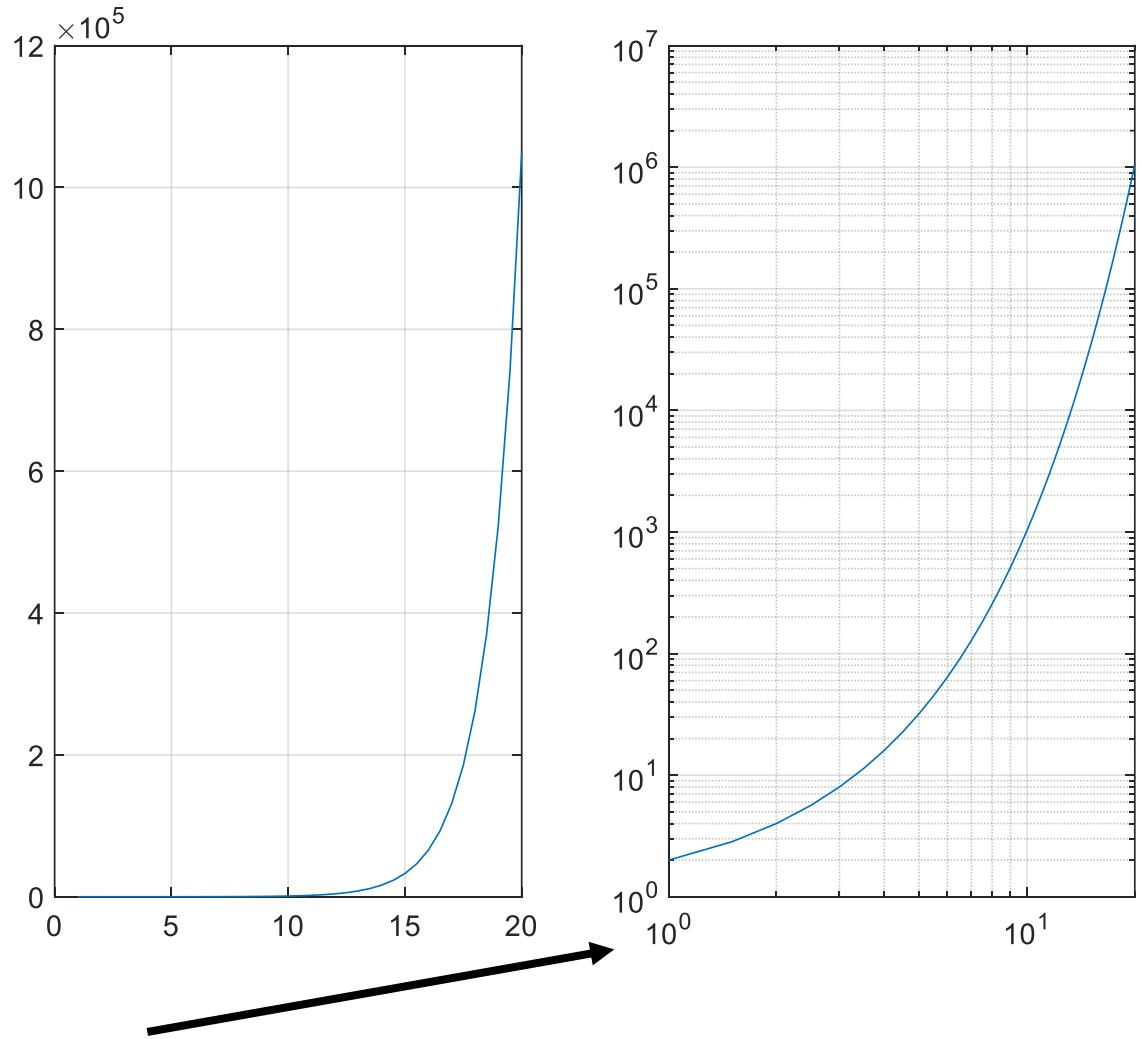


Log-log plots

- loglog(x,y):
  plots x- and y-coordinates using a base-10 logarithmic scale on the x-axis and the y-axis

- Example: plotting $y = 2^x$

```
x = 1:0.5:20;
y = 2.^x;
figure;
subplot(1,2,1);
plot(x,y); % ordinary xy plot
grid on;
subplot(1,2,2);
loglog(x,y); % log-log plot, easier to visualize data with different magnitudes
grid on;
```

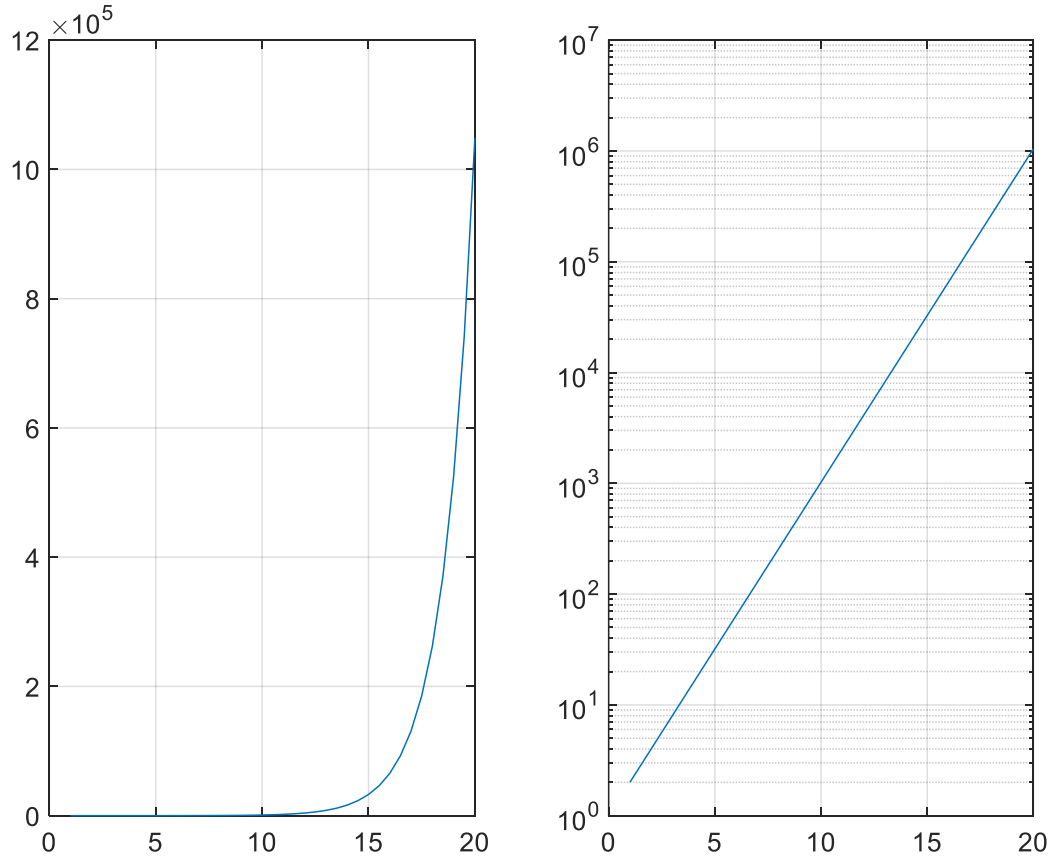


Semi-log plots

- Making semi-log plots
  - semilogx(x,y): only use a base-10 logarithmic scale on the x-axis
  - semilogy(x,y): only use a base-10 logarithmic scale on the y-axis
  - Useful for recognizing the relationships between x and y

- Example: plotting $y = 2^x$

```
x = 1:0.5:20;
y = 2.^x;
figure;
subplot(1,2,1);
plot(x,y); % ordinary xy plot
grid on;
subplot(1,2,2);
semilogy(x,y); % semi-log plot
grid on;
```

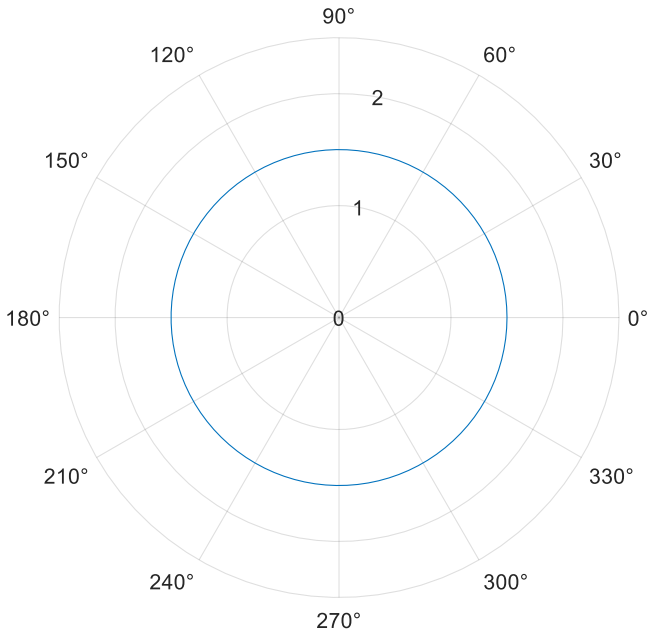


Polar coordinate plots

- Plot a curve $r(\theta)$ in the polar coordinate system, with θ indicating the angle in radians and r indicating the radius value for each point
  - Older MATLAB versions (before R2016a): polar
  - Newer MATLAB versions: polarplot
- Basic syntax: polarplot(theta,rho)

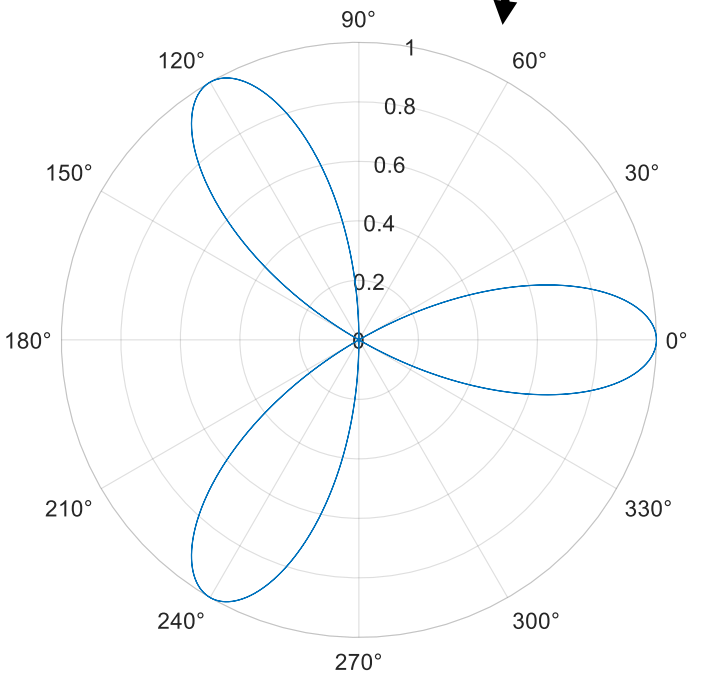
Example:

```
theta = linspace(0,2*pi,100);
rho = 1.5*ones(100,1);
figure;
polarplot(theta,rho)
```



Example:

```
theta = 0:0.01:2*pi;
rho = cos(3*theta);
figure;
polarplot(theta,rho)
```

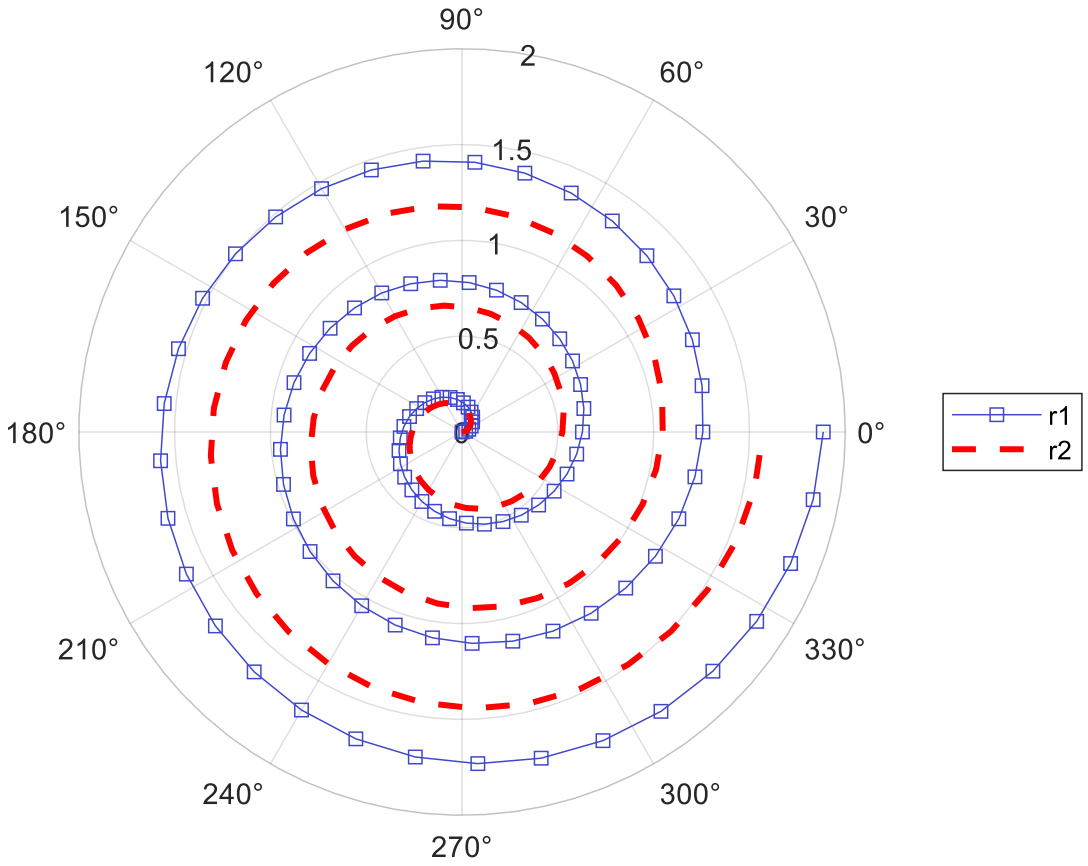


Polar coordinate plots

- More advanced polar plot options: similar to those for the plot function
  - Changing line color, line style, marker style, line width etc. by adding specifications in the polarplot function
  - Plotting multiple curves on the same polar plot: use hold on

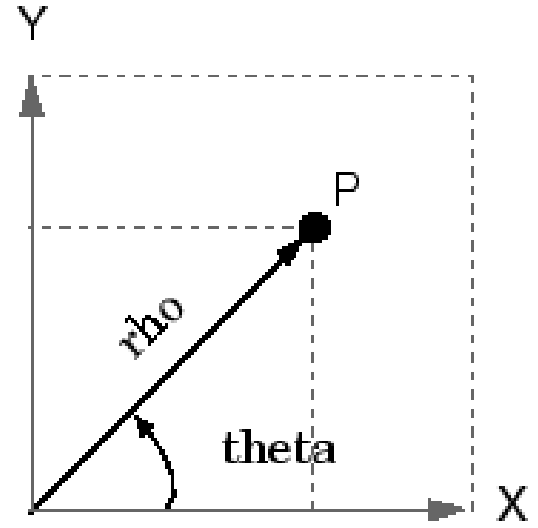
- Example:

```
t = linspace(0,6*pi,100);
r1 = t/10;
figure;
polarplot(t,r1,'s-','Color',[63,72,204]/255)
hold on
r2 = t/12;
polarplot(t,r2,'r--','LineWidth',2)
hold off
legend('r1', 'r2')
```



Polar coordinate plots

- Two relevant useful built-in functions
  - [theta,rho] = cart2pol(x,y): transforms xy coordinates to polar coordinate arrays theta and rho
  - [x,y] = pol2cart(theta,rho): transforms polar coordinate arrays theta and rho to xy coordinates

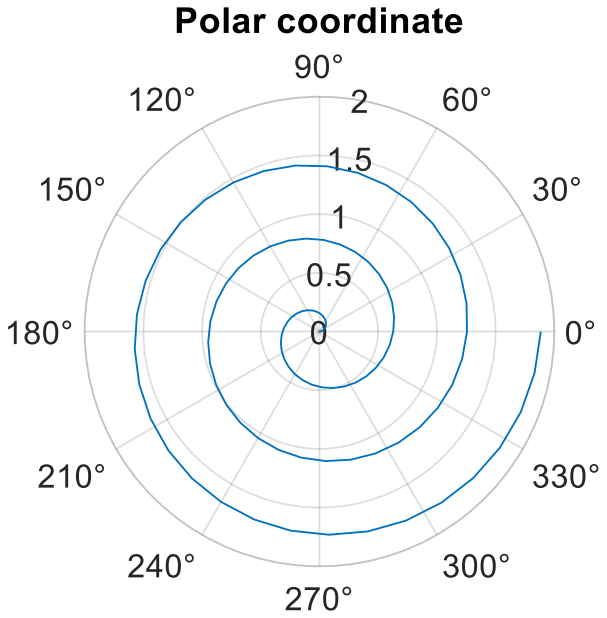


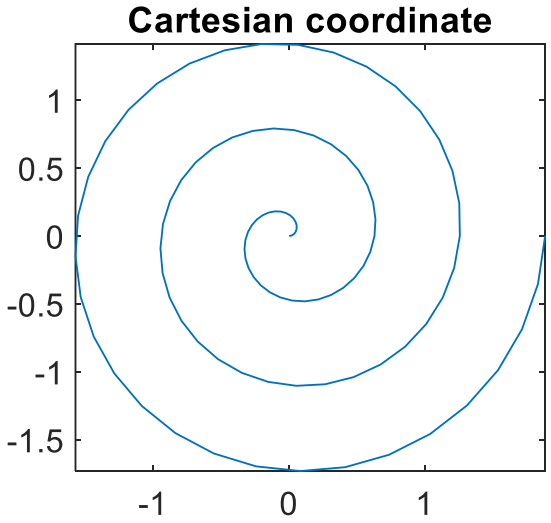
Polar to Cartesian Mapping

```
x = rho*cos(theta)
y = rho*sin(theta)
```

- Example:

```
t = linspace(0,6*pi,100);
r = t/10;
[x,y] = pol2cart(t,r);
figure;
subplot(1,2,1);
polarplot(t,r);
title('Polar coordinate')
subplot(1,2,2);
plot(x,y)
axis equal tight;
title('Cartesian coordinate')
```





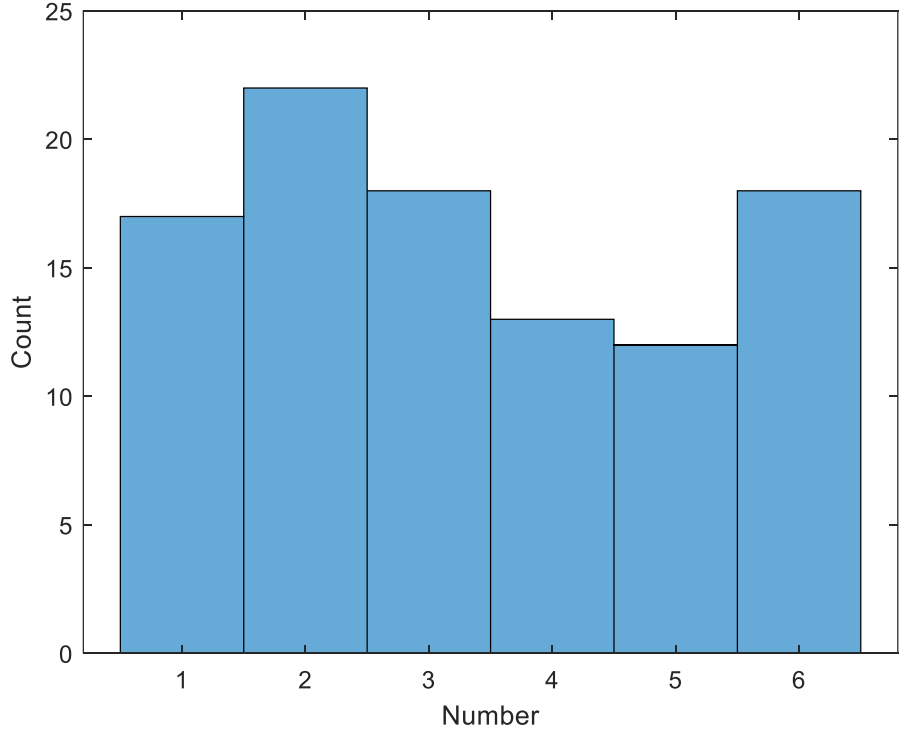
Histogram plots: grouping data into bins

- Basic syntax:
  histogram(X): create a histogram plot of X with automatic binning

- Example:

```
% Randomly generate 100 integers between 1 and 6
X = randi([1,6],100,1);

% Make the histogram plot
figure;
histogram(X);
xlabel('Number')
ylabel('Count')
```

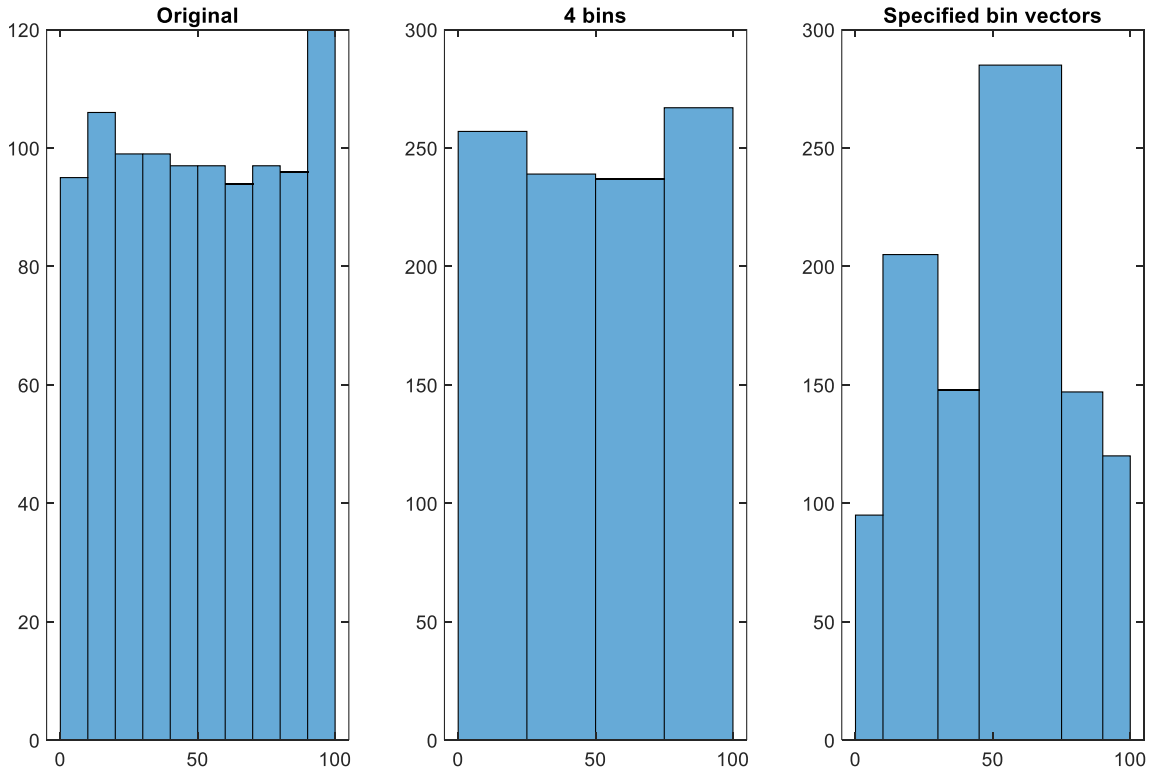


Histogram plots: grouping data into bins

- More options:
  - histogram(X, nbins): specify the number of bins
  - histogram(X, edges): sort X into bins with bin edges specified in a vector

- Example:

```
% 1000 random integers between 1 and 100
Y = randi([1,100],1000,1);
figure;
subplot(1,3,1);
histogram(Y); % automatic binning
title('Original')
subplot(1,3,2);
histogram(Y,4); % specify to be 4 bins only
title('4 bins')
subplot(1,3,3);
histogram(Y,[0,10,30,45,75,90,100]); % specify the bin edges
title('Specified bin vectors')
```

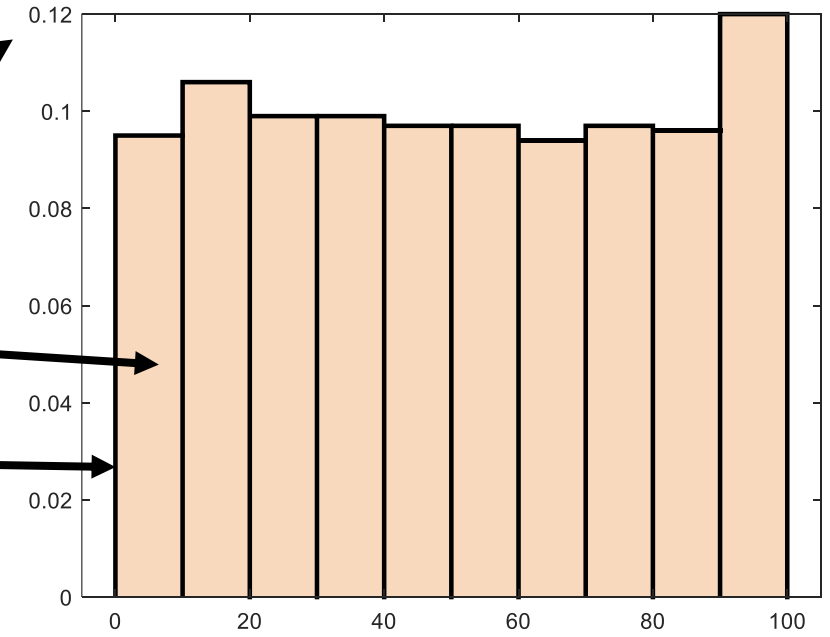


Histogram plots: grouping data into bins

- More options:
  - 'Normalization': 'count' (default), 'probability', 'percentage', etc.
  - 'Orientation': 'vertical' (default) or 'horizontal'
  - 'FaceColor': Change the face color of the bins
  - 'EdgeColor': Change the edge color of the bins
  - 'LineWidth': Change the width of the edges

- Example:

```
figure;
histogram(Y, 'Normalization', 'Probability', ...
    'FaceColor',[244,192,145]/255, ...
    'EdgeColor','k', 'LineWidth',2);
```



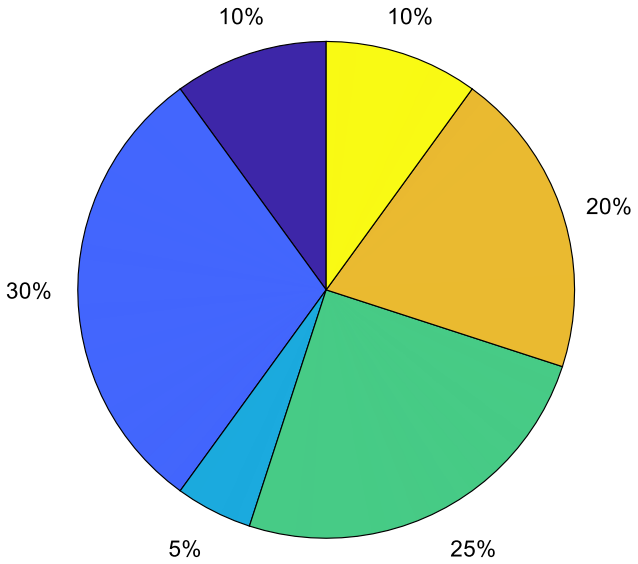
Pie charts

- Making a pie chart to represent data
  - pie(X): draws a pie chart using the data in X
  - pie(X,explode): specifies which slices to offset from the center of the pie chart
  - pie(X,labels): creates a labelled pie chart

Use curly brackets as we are creating a cell array here (more details later)

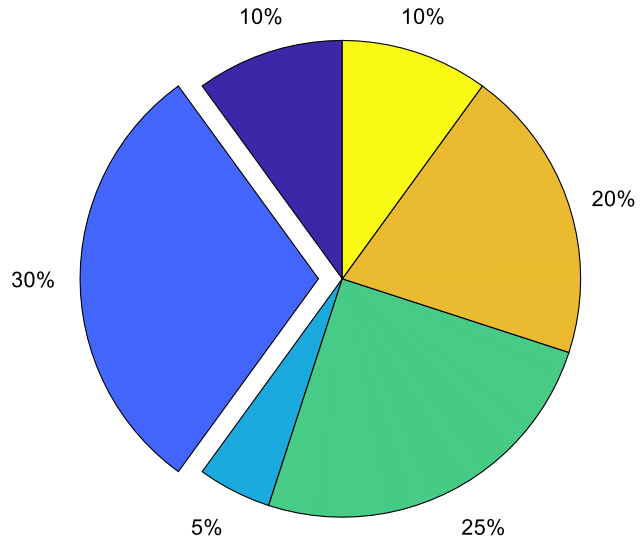
- Example:

```
X = [1, 3, 0.5, 2.5, 2, 1];
figure;
pie(X)
```



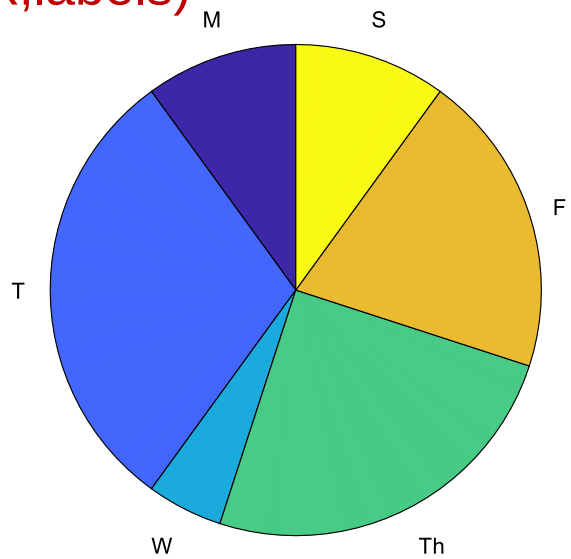
- Example:

```
X = [1, 3, 0.5, 2.5, 2, 1];
explode = [0, 1, 0, 0, 0, 0];
figure;
pie(X,explode)
```



- Example:

```
X = [1, 3, 0.5, 2.5, 2, 1];
labels = {'M','T','W','Th','F','S'};
figure;
pie(X,labels)
```

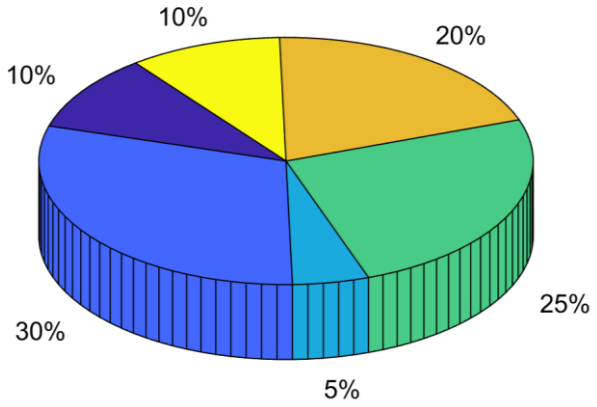


Three-dimensional pie charts

- Making a three-dimensional pie chart to represent data
  - pie3(X): draws a three-dimensional pie chart using the data in X
  - pie3(X,explode): specifies which slices to offset from the center of the pie chart
  - pie3(X,labels): creates a labelled three-dimensional pie chart

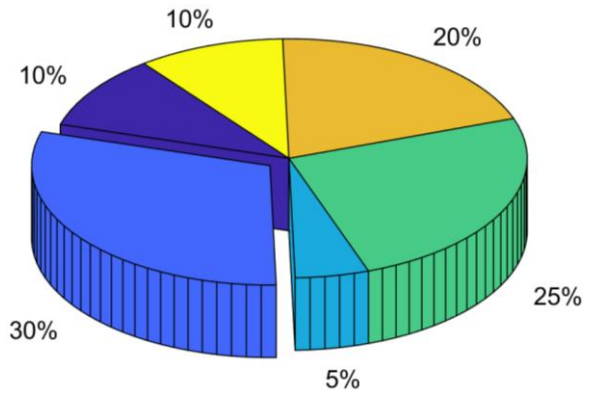
- Example:

```
X = [1, 3, 0.5, 2.5, 2, 1];
figure;
pie3(X)
```



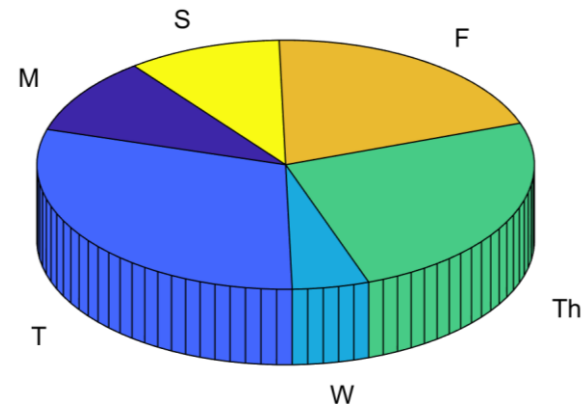
- Example:

```
X = [1, 3, 0.5, 2.5, 2, 1];
explode = [0, 1, 0, 0, 0, 0];
figure;
pie3(X,explode)
```



- Example:

```
X = [1, 3, 0.5, 2.5, 2, 1];
labels = {'M','T','W','Th','F','S'};
figure;
pie3(X,labels)
```

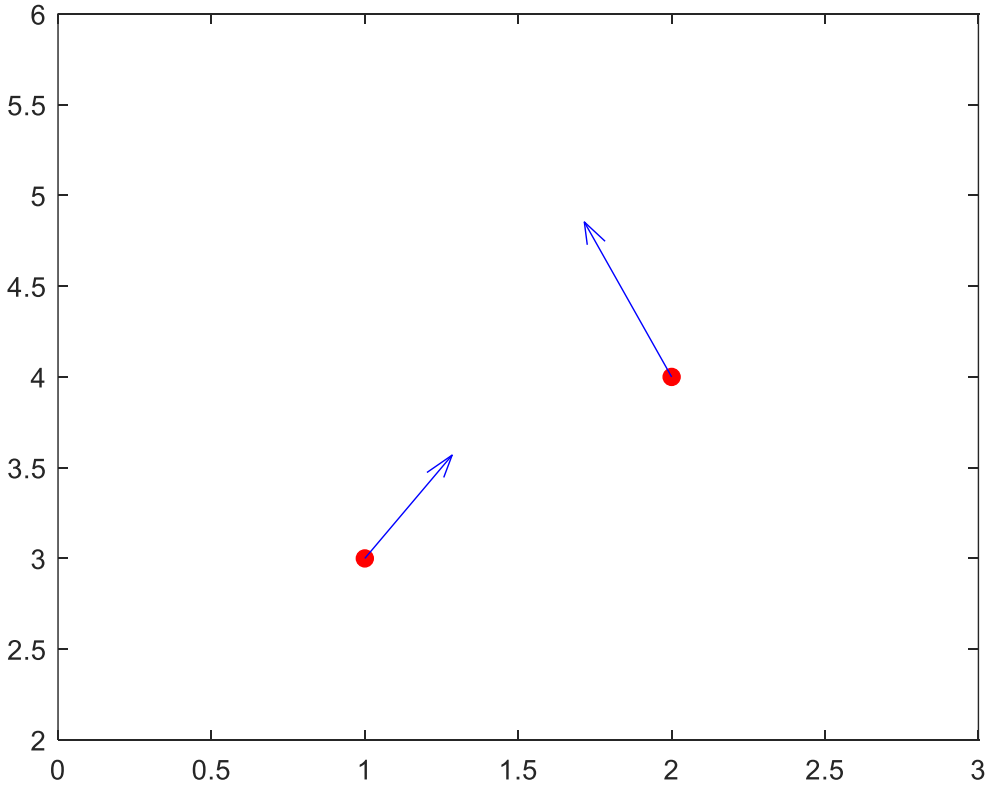


The *quiver* function for making vector plots

- quiver(X,Y,U,V): plots arrows with directional components U and V at the Cartesian coordinates specified by X and Y

- Example:

```
x = [1,2];
y = [3,4];
u = [1,-1];
v = [2,3];
figure;
plot(x,y,'ro','MarkerFaceColor','r'); % the dots
hold on;
quiver(x,y,u,v,'b'); % the arrows
axis([0, 3, 2, 6])
```



The *meshgrid* function

- [X,Y] = meshgrid(x,y):
  - generate 2-D grid coordinates based on the coordinates contained in vectors x and y
  - In other words, generate all possible points (x_i, y_j) using the elements in x and y and store the information as two 2-D arrays X and Y

- Example:

```
x = 1:3; % the desired x coordinates
y = 1:5; % the desired y coordinates
[X,Y] = meshgrid(x,y);
```

X =

1	2	3
1	2	3
1	2	3
1	2	3
1	2	3

Y =

1	1	1
2	2	2
3	3	3
4	4	4
5	5	5

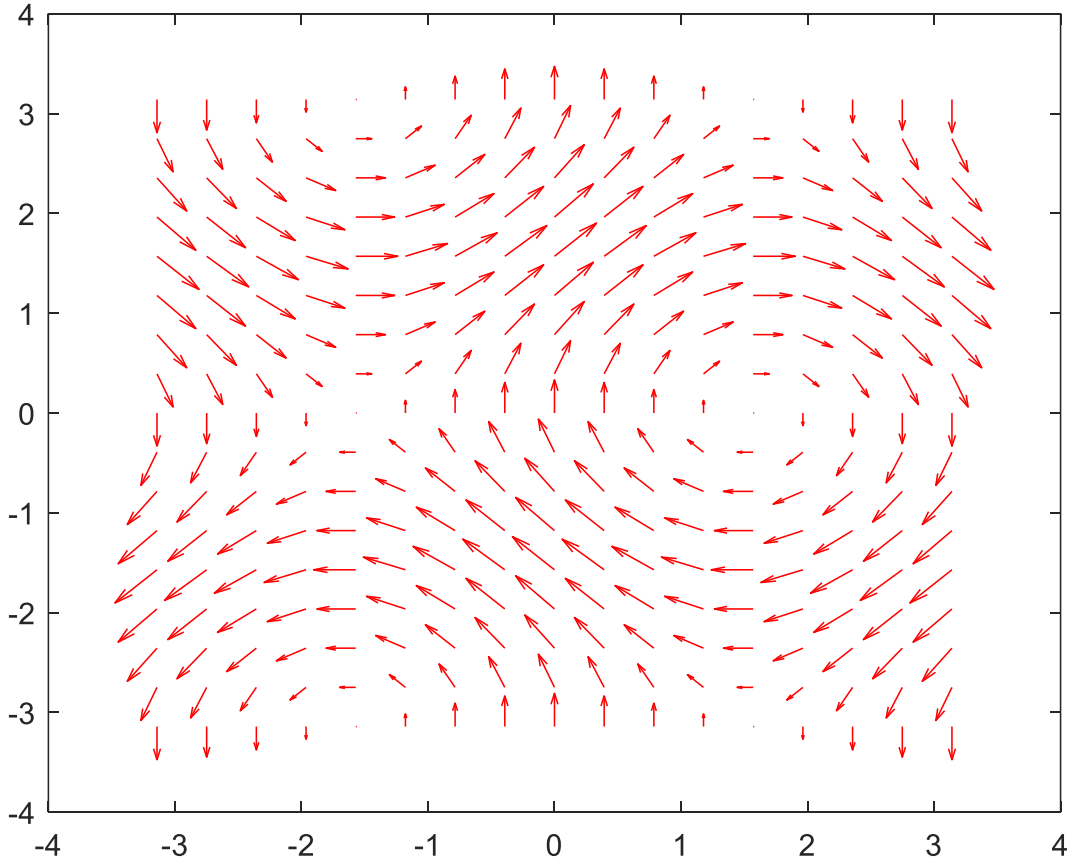
The *quiver* function for making vector plots

- quiver(X,Y,U,V): plots arrows with directional components U and V at the Cartesian coordinates specified by X and Y

- Useful for plotting vector fields when combined with meshgrid

- Example:

```
[X,Y] = meshgrid(-pi:pi/8:pi,-pi:pi/8:pi);
U = sin(Y); % the directions
V = cos(X); % the directions
figure;
quiver(X,Y,U,V,'r')
```



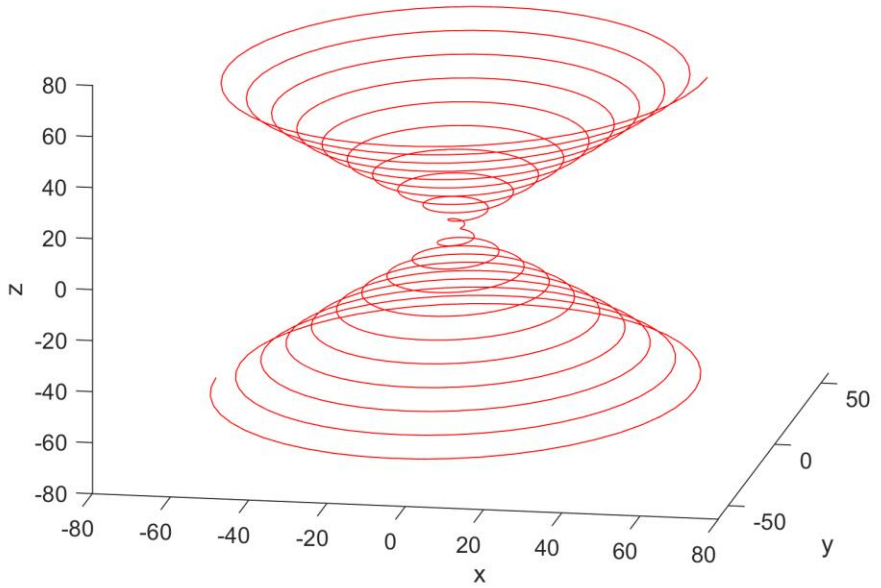
3D plot for points and curves

- Basic syntax: plot3(x,y,z)

- Plotting style options: similar to the ones for plot
  - Changing line color, line style, marker style, line width etc. by adding specifications in the plot3 function
  - Plotting multiple curves on the same plot: use hold on
  - Changing the camera angle: view(a,b)

- Example:

```
t = linspace(-20*pi,20*pi,1000);
figure;
plot3(t.*cos(t),t.*sin(t),t, 'r-');
xlabel('x')
ylabel('y')
zlabel('z')
view(10,20);
```



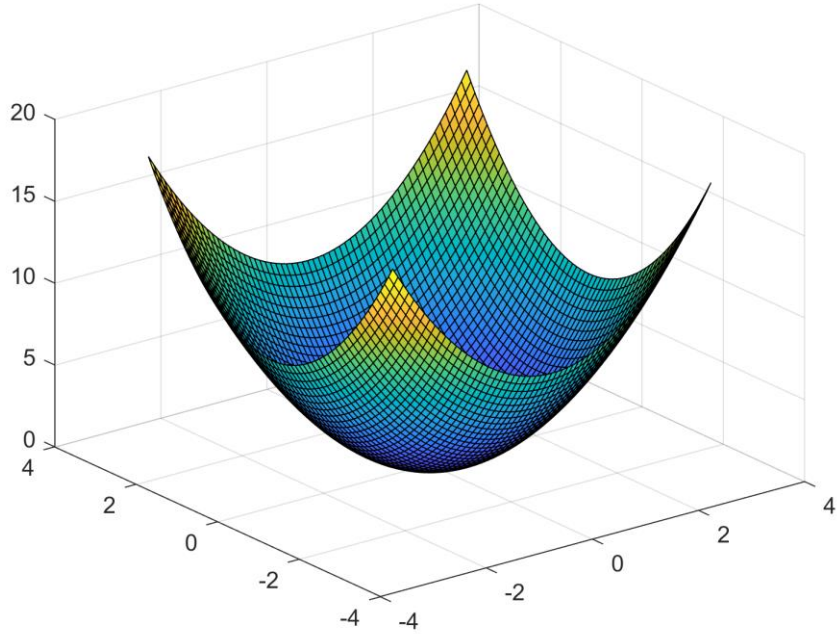
Surface plot

- Basic syntax: surf(X,Y,Z)
  - creates a three-dimensional surface plot based on the 2-D arrays X, Y, Z (usually generated from meshgrid)
  - with solid edge colors and solid face colors by default
  - the face color is based on the Z value
  - (More style options to be covered later)

- Example: paraboloid

$$z = f(x, y) = x^2 + y^2$$

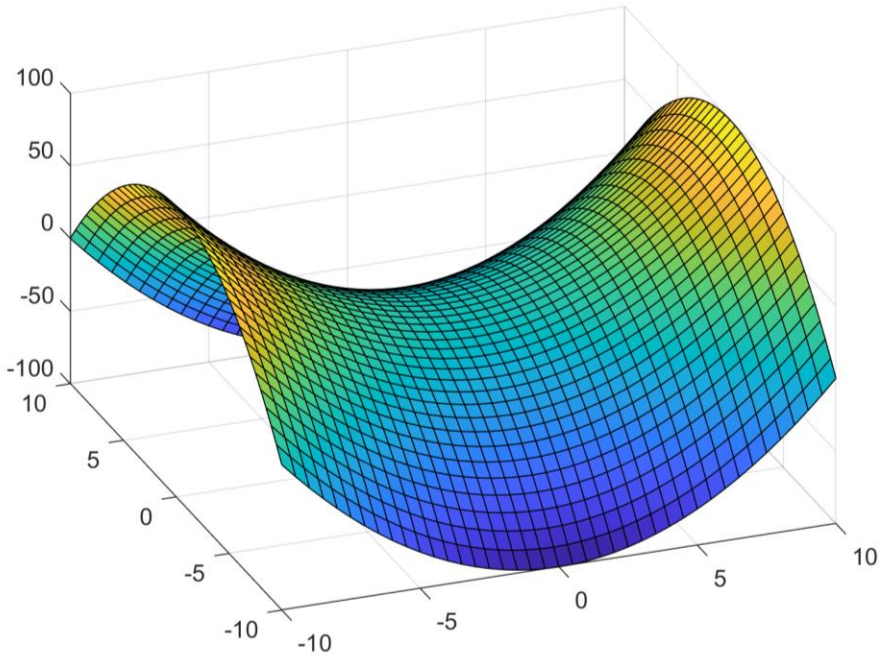
```
[X,Y] = meshgrid(-3:0.1:3,-3:0.1:3);
Z = X.^2+Y.^2;
figure;
surf(X,Y,Z);
```



Surface plot

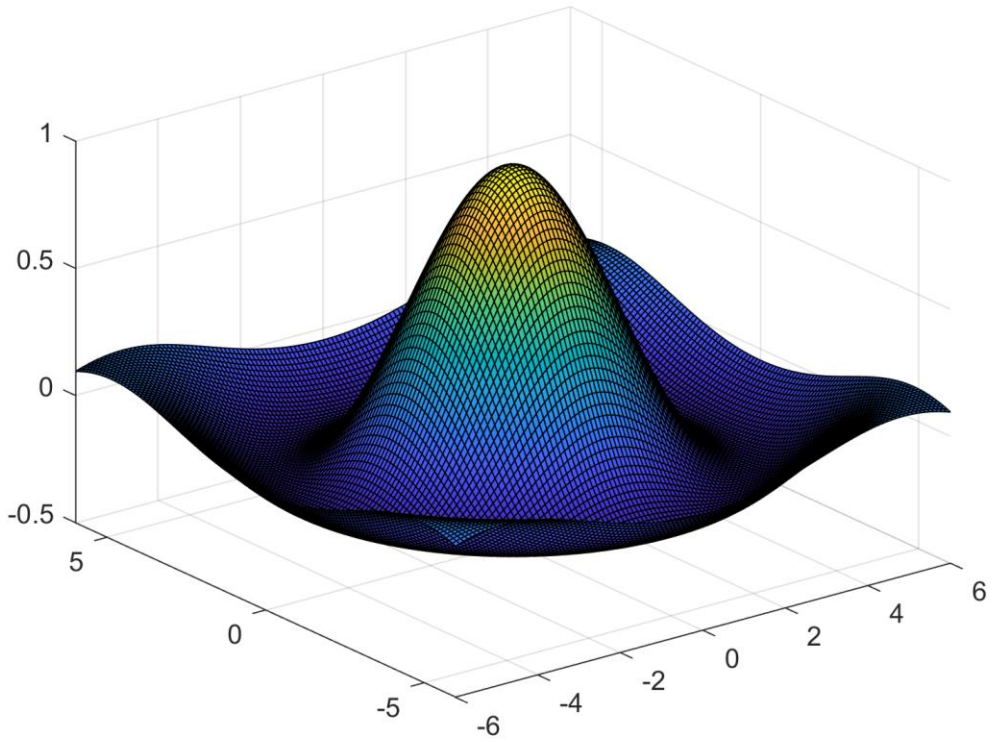
- Example: hyperbolic paraboloid

```
x = -10:0.5:10;
y = -10:0.5:10;
[X,Y] = meshgrid(x,y);
Z = X.^2-Y.^2;
figure;
surf(X,Y,Z);
```



- Example: a hat surface

```
[X,Y]=meshgrid(-6:0.1:6,-6:0.1:6);
R=sqrt(X.^2+Y.^2+.1);
Z=sin(R)./R;
figure;
surf(X,Y,Z)
```



Surface plot

- Example (parametric surface): unit sphere

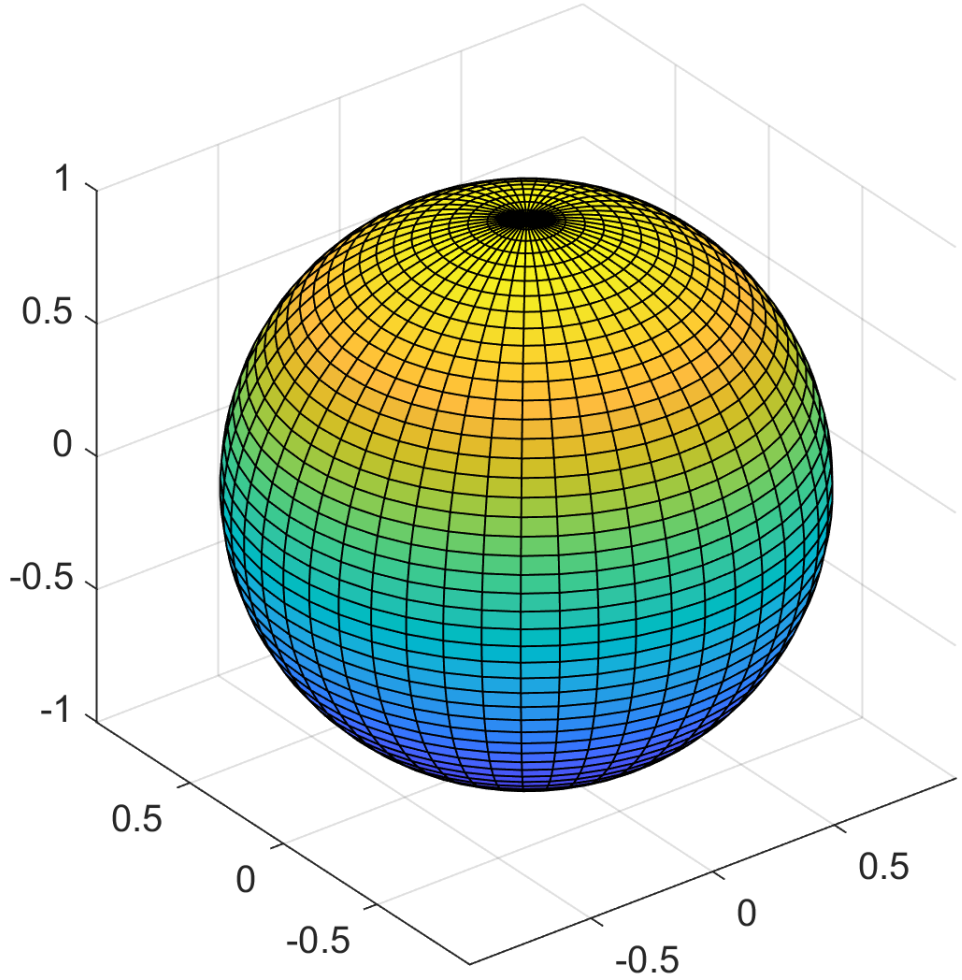
$$\big(X(u,v), Y(u,v), Z(u,v)\big) = (\cos u \cos v\,, \cos u \sin v\,, \sin u)$$

with $-\frac{\pi}{2} \le u \le \frac{\pi}{2}, -\pi \le v \le \pi$

```
u = linspace(-pi/2,pi/2,50);
v = linspace(-pi,pi,50);
[U, V] = meshgrid(u,v);

% sphere equation
X=cos(U).*cos(V);
Y=cos(U).*sin(V);
Z=sin(U);

figure;
surf(X,Y,Z);
axis equal
```



Surface plot

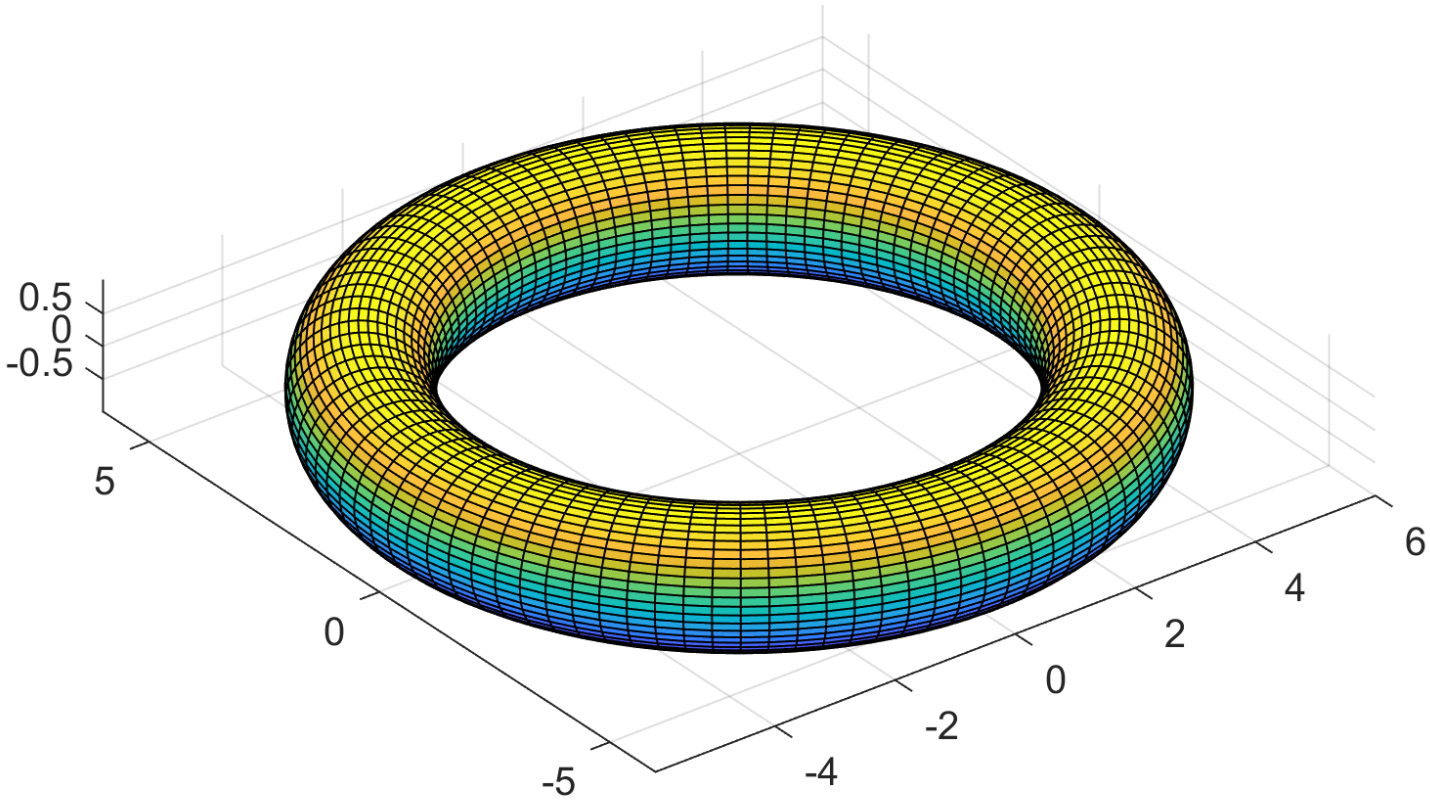
- Example (parametric surface): torus

$$\big(X(u,v), Y(u,v), Z(u,v)\big) = \big((c + a\cos v)\cos u\,, (c + a\cos v)\sin u\,, a\sin v\big)$$

with $0 \le u, v \le 2\pi$

```
u = linspace(0,2*pi,100);
v = linspace(0,2*pi,50);
[U, V] = meshgrid(u,v);
% torus equation
c = 5;
a = 1;
X = (c+a*cos(V)).*cos(U);
Y = (c+a*cos(V)).*sin(U);
Z = a*sin(V);

figure;
surf(X,Y,Z);
axis equal
```

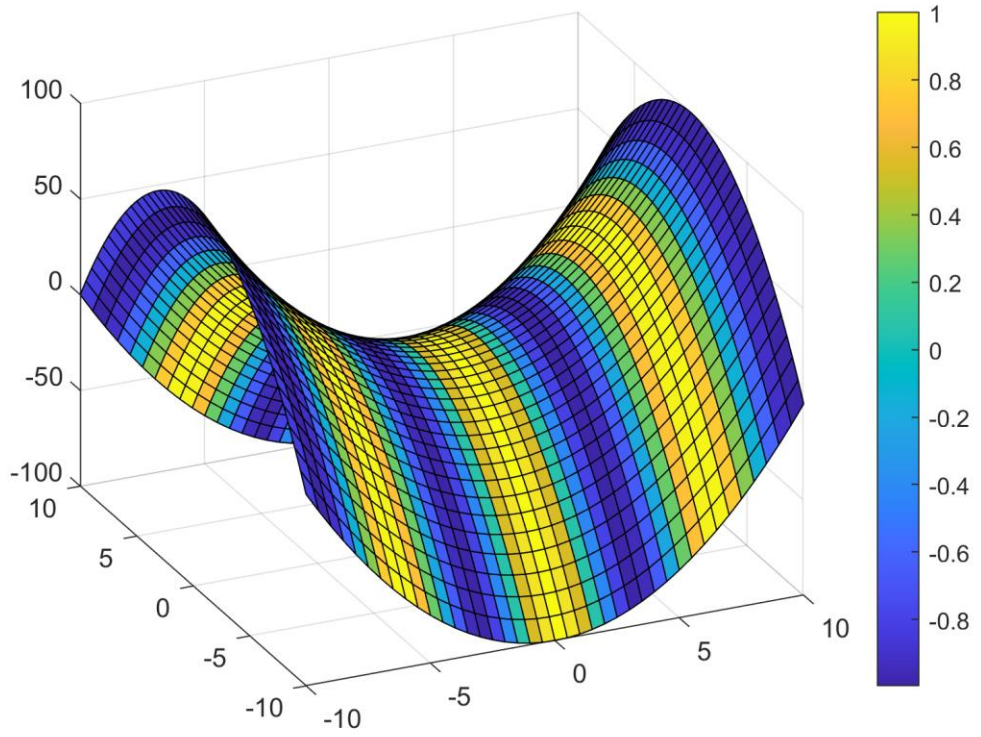


Surface plot style options

- More style options:
  - surf(X,Y,Z,C): creates a three-dimensional surface plot using X, Y, Z, where the face color is based on the C value
  - colorbar: show the color bar to see the corresponding values
  - colorbar off: close the color bar

- Example:

```
x = -10:0.5:10;
y = -10:0.5:10;
[X,Y] = meshgrid(x,y);
Z = X.^2-Y.^2;
C = cos(X);
figure;
surf(X,Y,Z,C);
colorbar
```

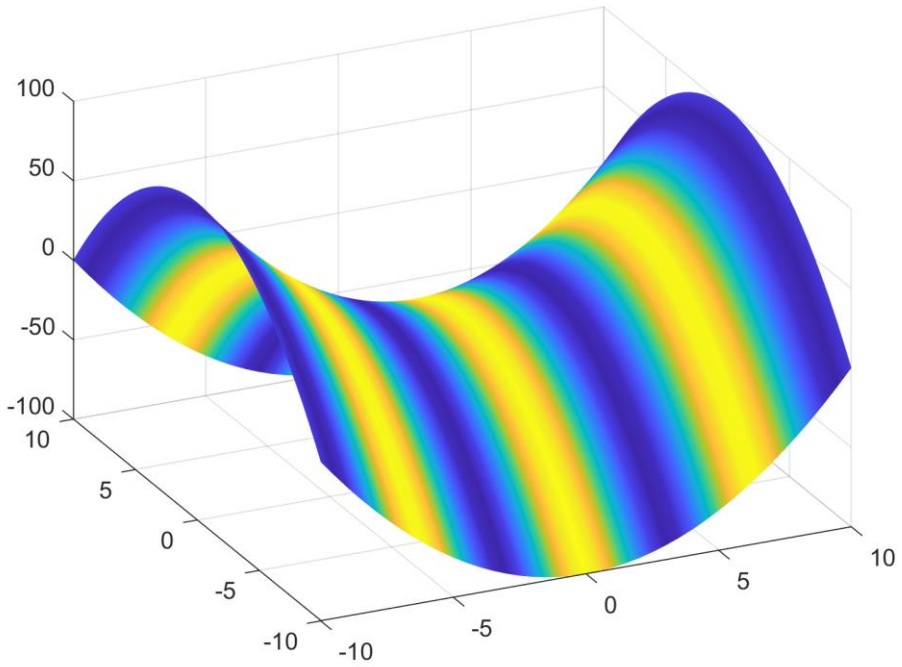


Surface plot style options

- More style options in surf(X,Y,Z,…):
  - EdgeColor: 'k','r', 'none' (no color) etc.
  - LineStyle: '-', '--' etc.
  - FaceColor: 'flat', 'interp' (interpolated coloring) etc.
  - FaceAlpha: a scalar in range [0,1] for face transparency

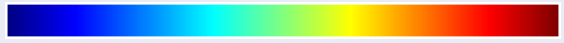
- Example:

```
x = -10:0.5:10;
y = -10:0.5:10;
[X,Y] = meshgrid(x,y);
Z = X.^2-Y.^2;
C = cos(X);
figure;
surf(X,Y,Z,C,'EdgeColor','none','FaceColor','interp');
view(-25,35)
```



Colormap

- There are multiple built-in color schemes in MATLAB:

Name	Color scale
parula	
jet	
hsv	
hot	
cool	
spring	
summer	
autumn	
winter	
copper	
gray	

… and many more!

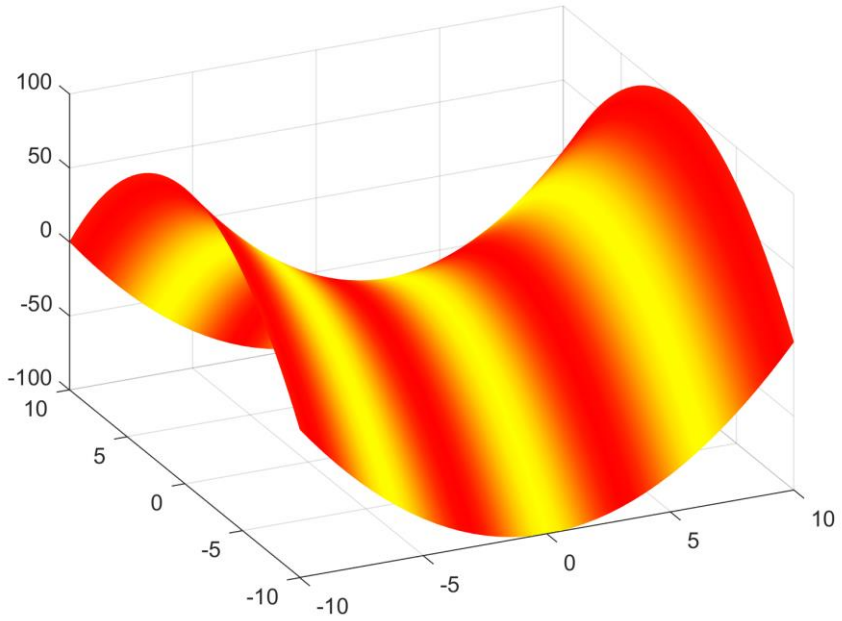
You can also create your own colormaps by specifying a list of RGB values.

See: https://www.mathworks.com/help/matlab/ref/colormap.html

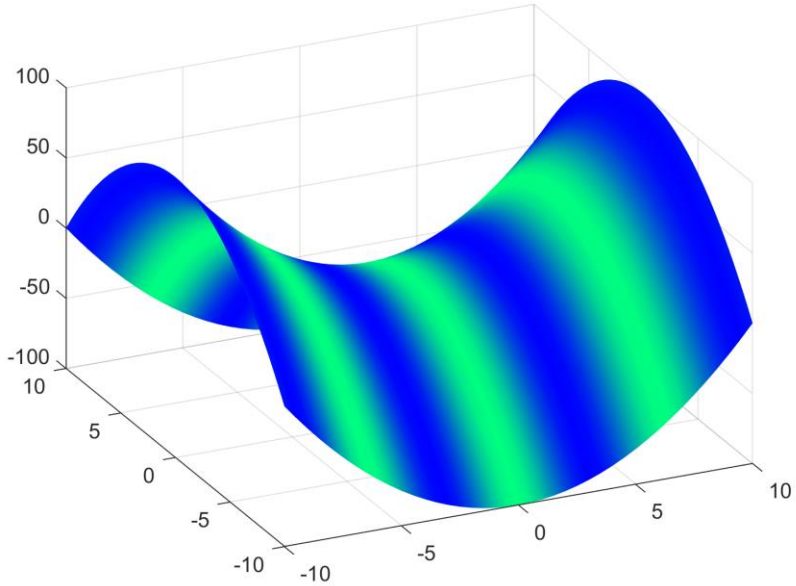
Colormap

- Example:

```
x = -10:0.5:10;
y = -10:0.5:10;
[X,Y] = meshgrid(x,y);
Z = X.^2-Y.^2;
C = cos(X);
figure;
surf(X,Y,Z,C,'EdgeColor','none','FaceColor','interp');
view(-25,35)
colormap autumn
```




```
colormap winter
```



Reminder: Lab 5 this week

January

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	[28]	[29]	[30]	[31]	

February

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						[1]
[2]	[3]	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	1

March

Sun	Mon	Tue	Wed	Thu	Fri	Sat
2	[3]	[4]	[5]	[6]	[7]	[8]
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

April

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17		

Lecture 1- Lecture 13

Lab 1 - Lab 10 (40%)

Test 1 (30%)
Test 2 (30%)

Reminder: Test 1 next week (February 27)

- Take place during the usual lab session (please arrive 10 minutes earlier)
  - **90-minute test**
    - (Class A) 10:30 - 12:00
    - (Class B) 12:30 - 14:00
    - (Class C) 14:30 - 16:00
  - Writing part + Coding part
  - Additional 10 minutes for uploading all files to Blackboard after the test
  - Coverage: Lecture 1-6 and Lab Assignment 1-5

- Format:
  - **Open-note**: May access our lecture notes and lab assignment solutions on the desktop computer
  - Other reference books/materials are NOT allowed
  - Other printed notes/tablets/phones/calculators are NOT allowed
  - NOT open-internet
  - NO discussion

Thank you!

Next time:

- Miscellaneous visualization topics and review for Test 1