While you are waiting

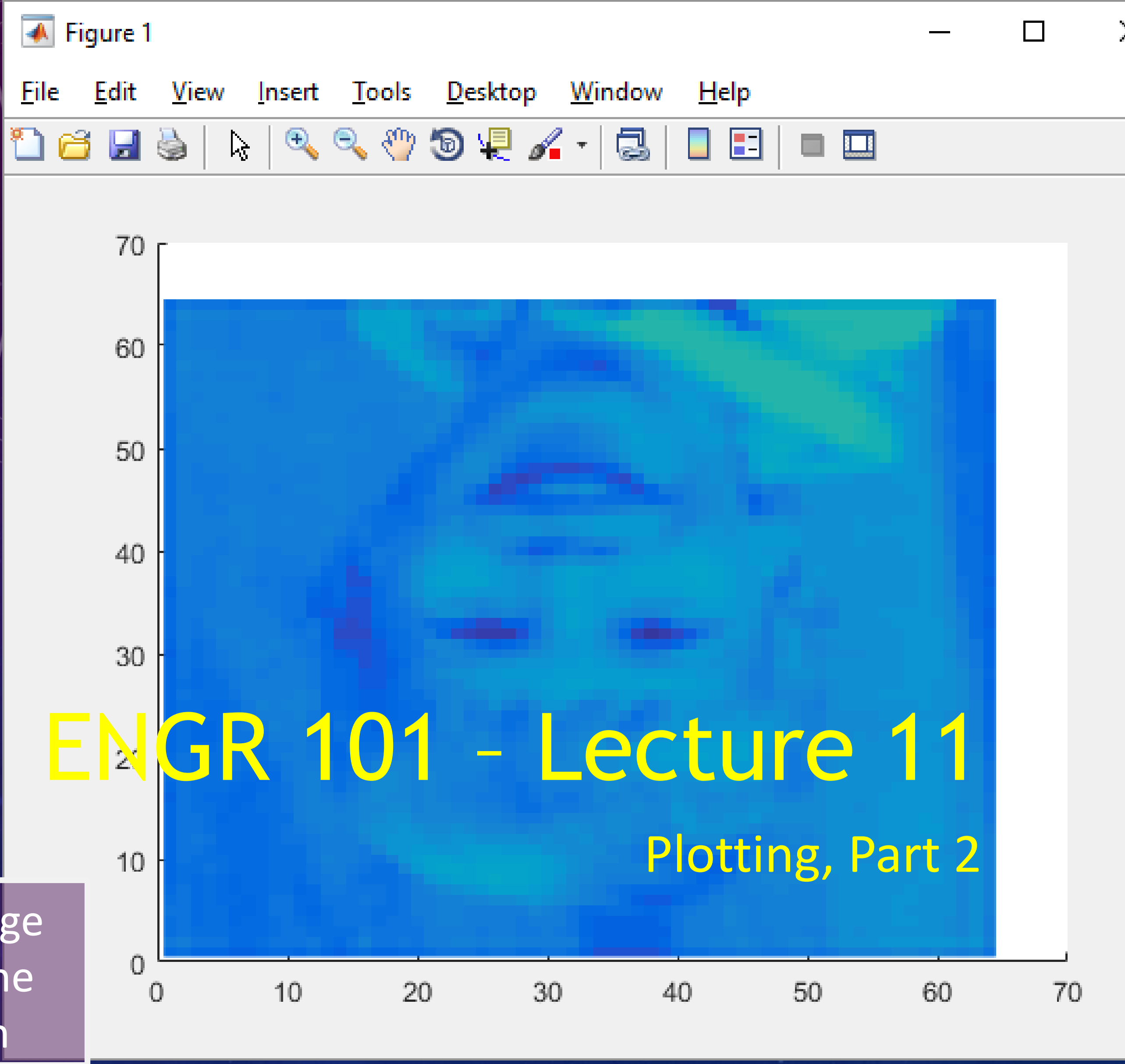
ANAGRAM

Re-arrange the letters to reveal the related words...

an old shoe

ENGR 101 - Lecture 11

Plotting, Part 2



In MATLAB, typing image gives you a figure of the MATLAB author's son

Laura Alford, James Juett, Rick Niciejewski 10/11/17

Announcements

- Project 2 due Thursday, 12-OCT-2017.
  - Everything for today is stored on 00_Todays_Lecture, the two *.mat files (`tensileStrength.mat`, `batteryLife.mat`), the lecture, and the Project_3 overview

- No labs Thursday 12-OCT-2017 to Wednesday 18-OCT-2017
- No lecture Monday 16-OCT-2017

Lecture Goals

- Today's lecture: Plotting, Part II
  - Modifying labels
  - Multiple plots in one figure, subplot
  - Plotting variable range, uncertainties
  - 3D plotting examples
  - Project 3 overview
  - Suggested readings, Attaway, Chap 11

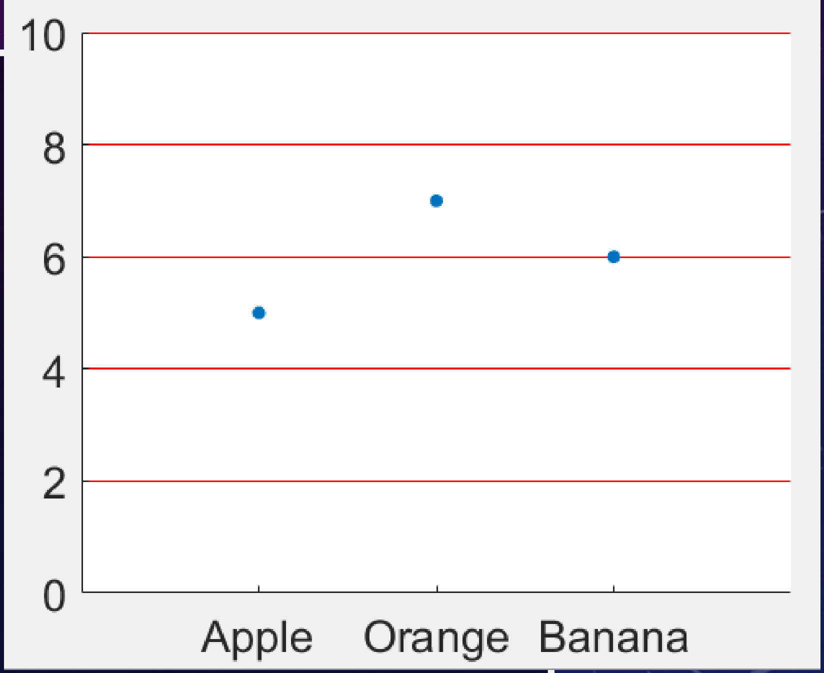
Customizing Plots

- We'll look at a few more methods for customizing the appearance of plots in MATLAB.
- Again, this lecture only covers the basics – see the MATLAB documentation linked in the footnotes for complete details!

gca

- You can use gca ("get current axes") to modify properties of the axes for the current figure.

```
% create a simple bar chart
scatter([1,2,3], [5,7,6], 'filled')

% get current axes in the variable ax
ax = gca;

% modify via ax, CaseSensitive
ax.FontSize = 20;
ax.YLim = [0,10];
ax.XLim = [0,4];
ax.XTick = [1,2,3];
ax.XTickLabel = {'Apple', 'Orange', 'Banana'};
ax.YGrid = 'on';
ax.GridColor = [1,0,0];
ax.GridAlpha = 1;
```



https://www.mathworks.com/help/matlab/ref/axes-properties.html

The set Function

- You can also use the set function instead of the dot notation.

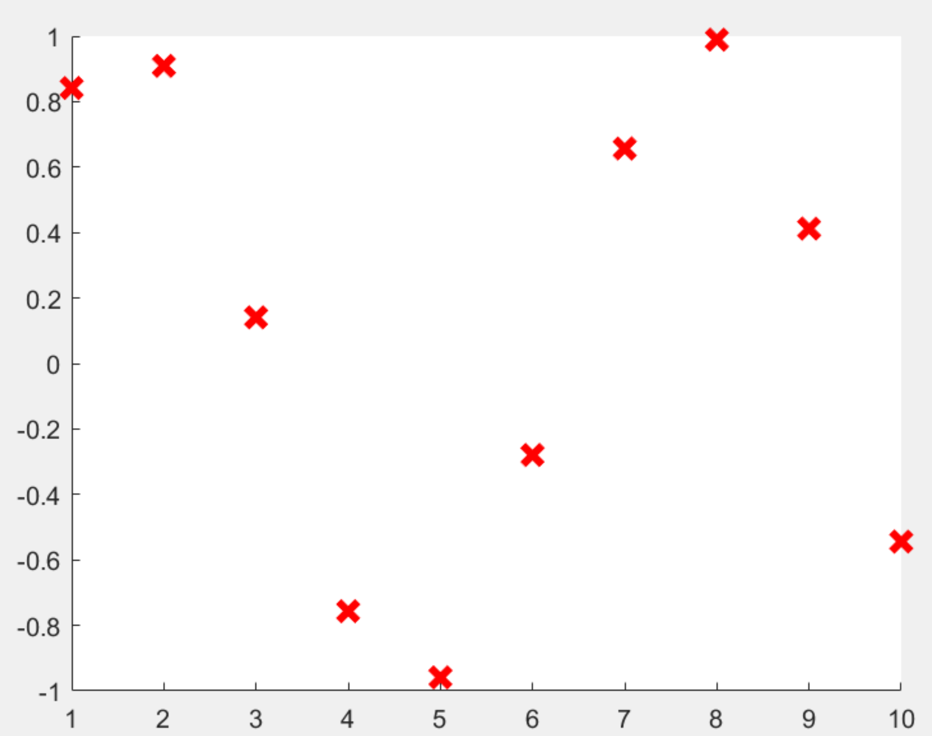
```
% create a simple bar chart
scatter([1,2,3], [5,7,6], 'filled')

% get current axes in the variable ax
ax = gca;

% set the font size using dot notation
ax.FontSize = 20;

% set the font size using the set function on gca
set(gca,'FontSize',20);
```

- **In older versions of MATLAB (pre R2014b), only the set function allows changes to axes properties.**

Graphics Object Properties

- Plotting functions return graphics objects that can be used to customize the appearance of the plot.

```
% create a scatterplot
% store the returned graphics object in s, size=100
x = 1:1:10;
s = scatter(x, sin(x), 100);

% modify properties through s
s.Marker = 'x';
s.LineWidth = 3;
s.MarkerEdgeColor = 'red';
```

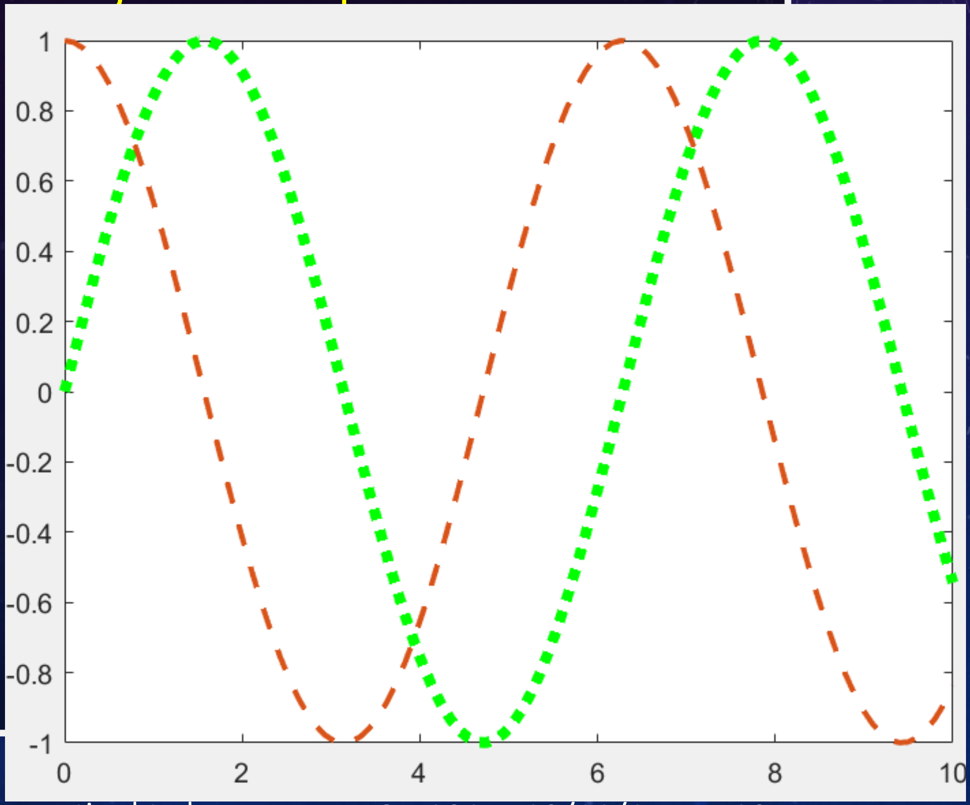


https://www.mathworks.com/help/matlab/graphics-object-prop

Graphics Object Properties

- If you plot multiple functions, you'll get a vector of graphics objects. Index into it to modify properties.

```
% plot multiple functions
% store the returned graphics objects in p
x = linspace(0,10,101);
p = plot(x,sin(x),x,cos(x));

% modify properties via p
% index to select plot
p(1).LineStyle = ':';
p(1).Color = 'green';
p(1).LineWidth = 4;

p(2).LineStyle = '--';
p(2).Color = 'red';
p(2).LineWidth = 2;
```

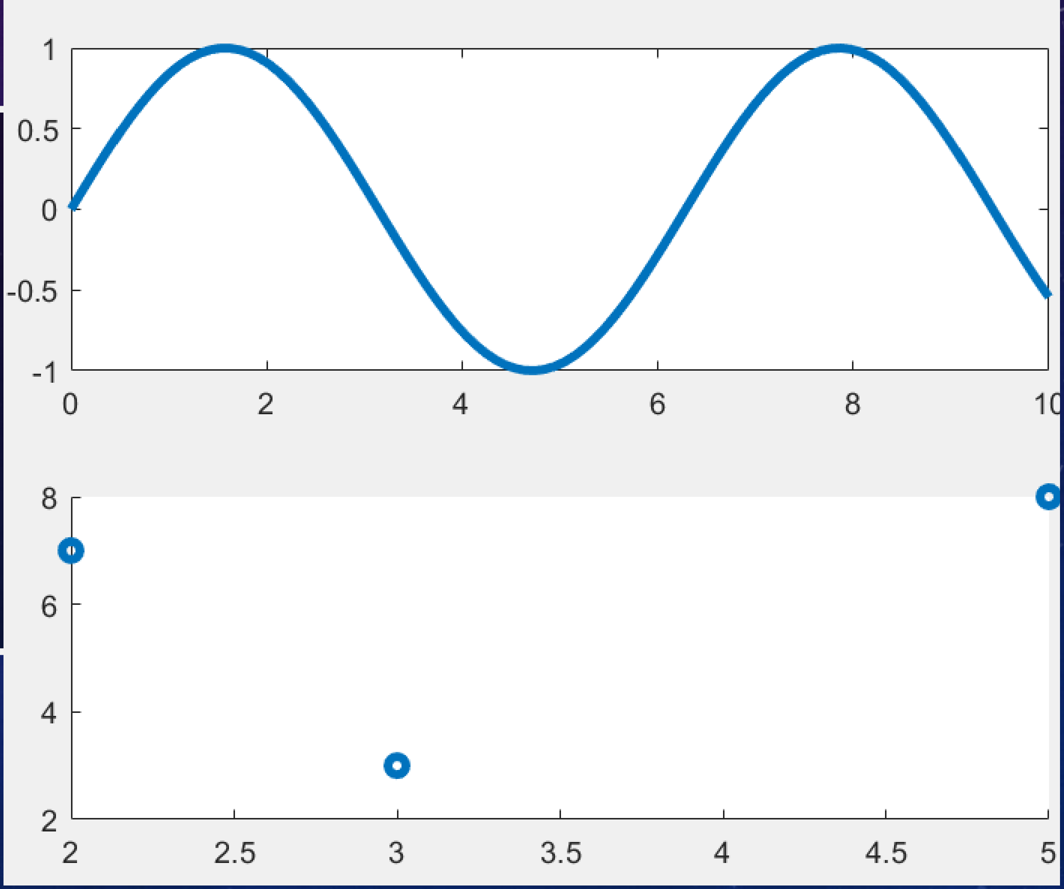


https://www.mathworks.com/help/matlab/graphics-object-properties.html

subplot

- The subplot function creates a matrix of plots in a figure window
- The axes are arranged in a grid-like matrix configuration.
- Example:

```
figure(); % create a figure

% the 1st plot in a 2x1 grid
subplot(2,1,1);
x = linspace(0,10,101);
plot(x, sin(x));

% the 2nd plot in the grid
subplot(2,1,2);
scatter([2,3,5], [7,3,8]);
```

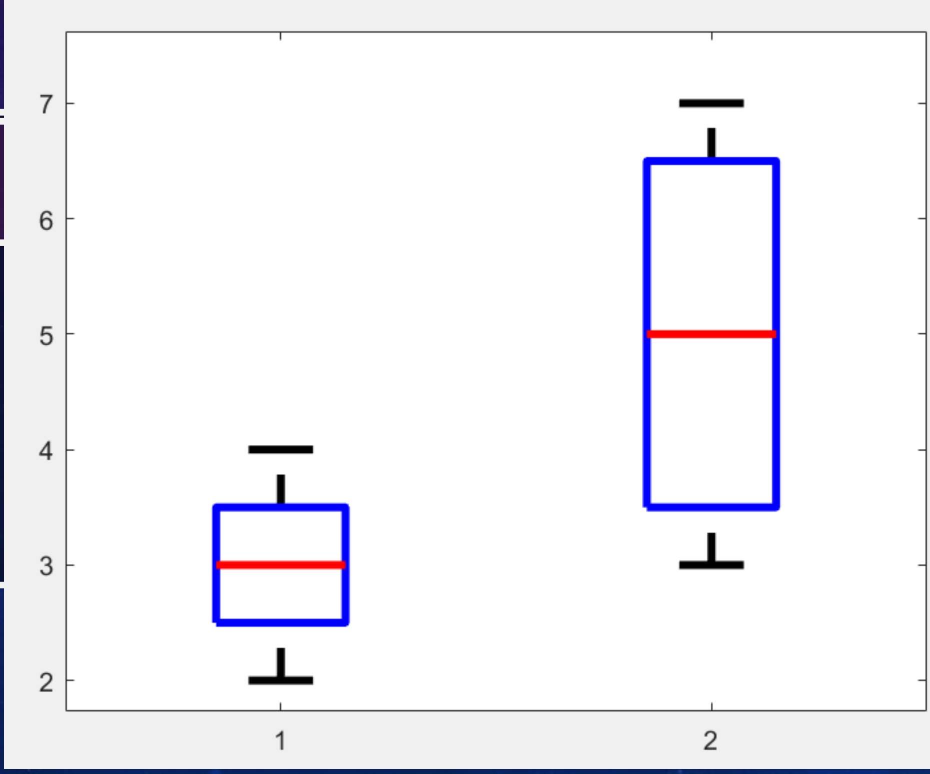


Box and Whisker Plots *(interpretation next page)*

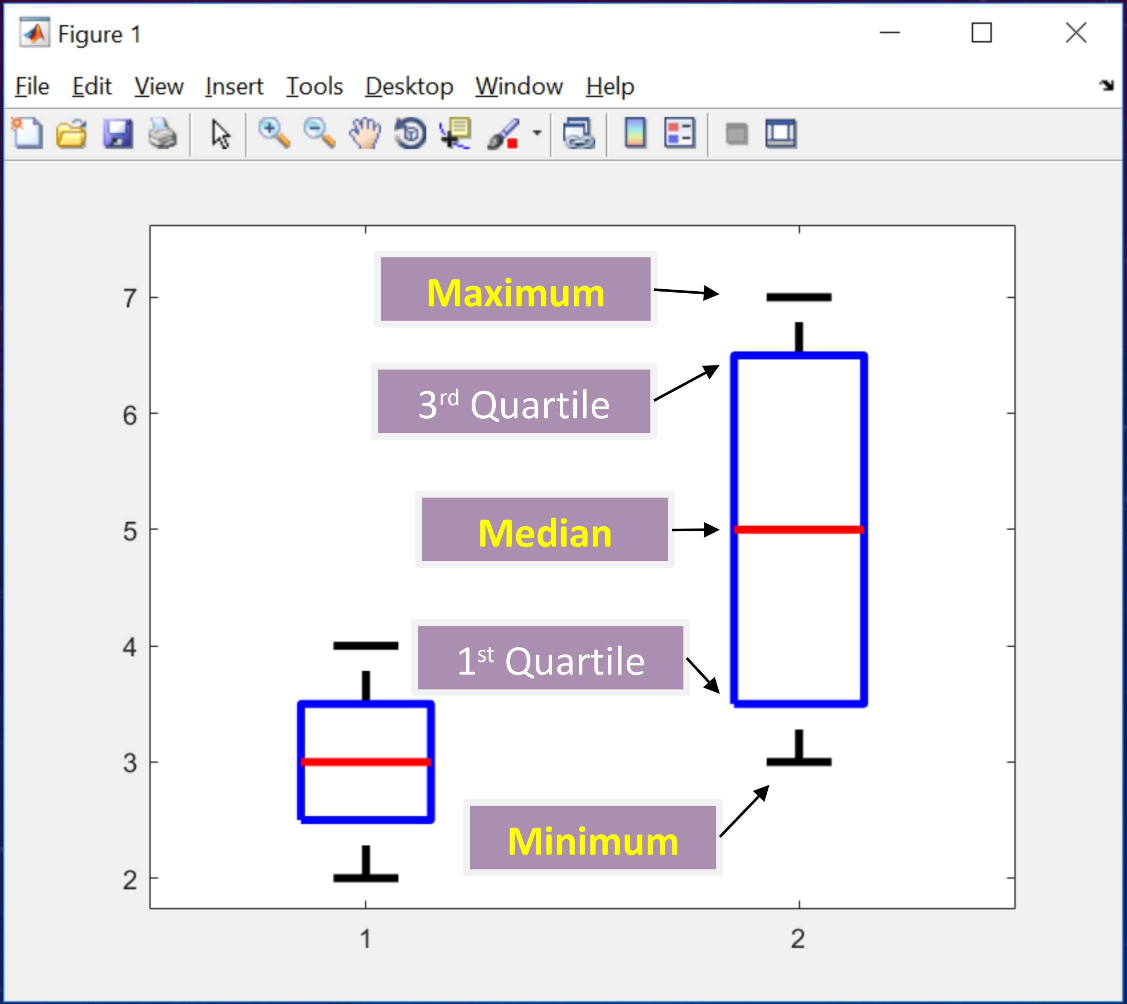
- Use the boxplot function to create a box and whisker plot.
- The input to boxplot should be a matrix where each column corresponds to a different variable.
  - Each entry within the column corresponds to different observations.
- Example:

```
data1 = [3;4;3;2];
data2 = [4;3;6;7];

h = boxplot([data1, data2]);

% adjust for projector :)
set(h, 'LineWidth', 3);
```

Combine columns into one matrix.



Interpreting Box and Whisker Plots



```
data1 = [3;4;3;2];
data2 = [4;3;6;7];
```

Working Example: Comparing Tensile Strengths

- Example: A manufacturing company is considering three different materials for use in structural components.
  - 50 samples of each material have been tested for tensile strength, the higher the value, the better.
  - We would like to visualize the average for each material, but also the spread of the data to get an idea of consistency.

To follow along:

```
load('tensileStrength.mat');
```

BREAK to 1) load file, 2) copy solution on the next slide into your MATLAB Command Window

Solution: Comparing Tensile Strengths

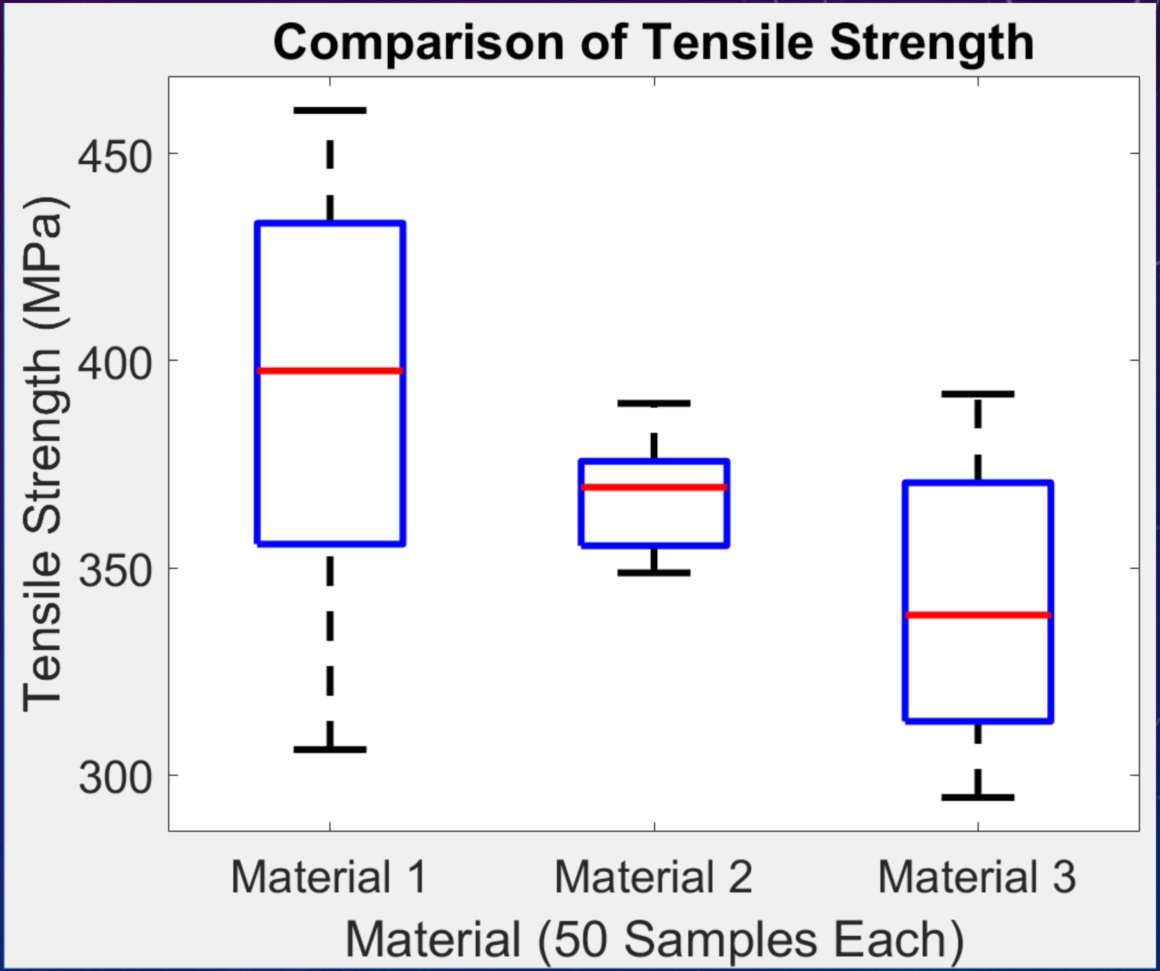
```
% create the box and whiskers plot
h = boxplot([material1, material2, material3]);

% use the graphics object to set the line width
set(h, 'LineWidth', 3); % wider than usual for projector

% Add title and axis labels
title('Comparison of Tensile Strength');
xlabel('Material (50 Samples Each)');
ylabel('Tensile Strength (MPa)');

% use gca to set the font size and x-axis labels
ax = gca;
ax.FontSize = 20;
ax.XTickLabel = {'Material 1', 'Material 2', 'Material 3'};
```

Make one matrix with a column for each variable/box we want to plot.

A cell array containing strings!

Interpretation: Comparing Tensile Strengths

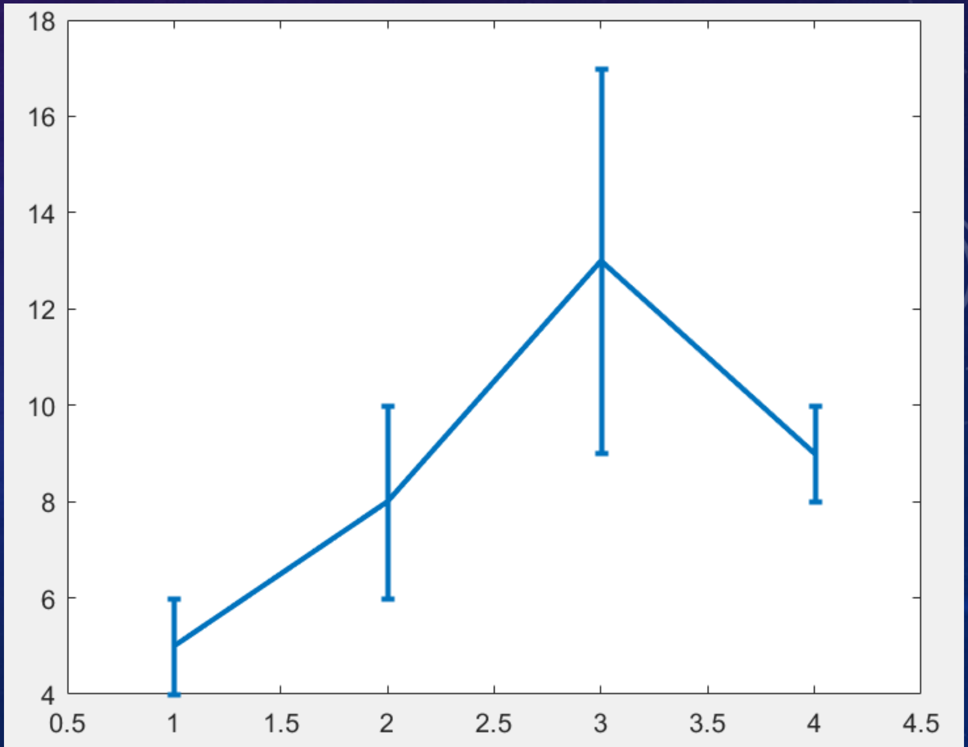
- Material 3 is pretty much inferior to both others.
- Material 1 has the highest "median" tensile strength, but has higher variability between different samples.
- Material 2 is more consistent, but in general not as strong as material 1.



Plots with Error Bars

- Use the errorbar function to show a plot with "error bars" at each data point.
- Error bars can be used to convey a range of values for each point on the plot, or uncertainty about a measured value.
- Example:

```
x = 1:4;
y = [5,8,13,9];
err = [1,2,4,1];
h = errorbar(x, y, err);

% adjust for projector :)
set(h, 'LineWidth', 2);
```



Working Example: Battery Lifecycle Analysis

- Example: A company that produces smartphones needs to analyze battery lifetime throughout several years of use.
  - A set of batteries have been put through several years of simulated use.
  - Twice per "year", the lifetime of each battery was tested and the mean and standard deviation for the set was recorded.
  - We would like to visualize the degradation of battery lifetime throughout the years as well as the amount of variability.
    - e.g. Can we claim that the phone had 3 hours battery life when new?
    - e.g. Can we claim that after 2 years, the phone has 2 hours of battery life?

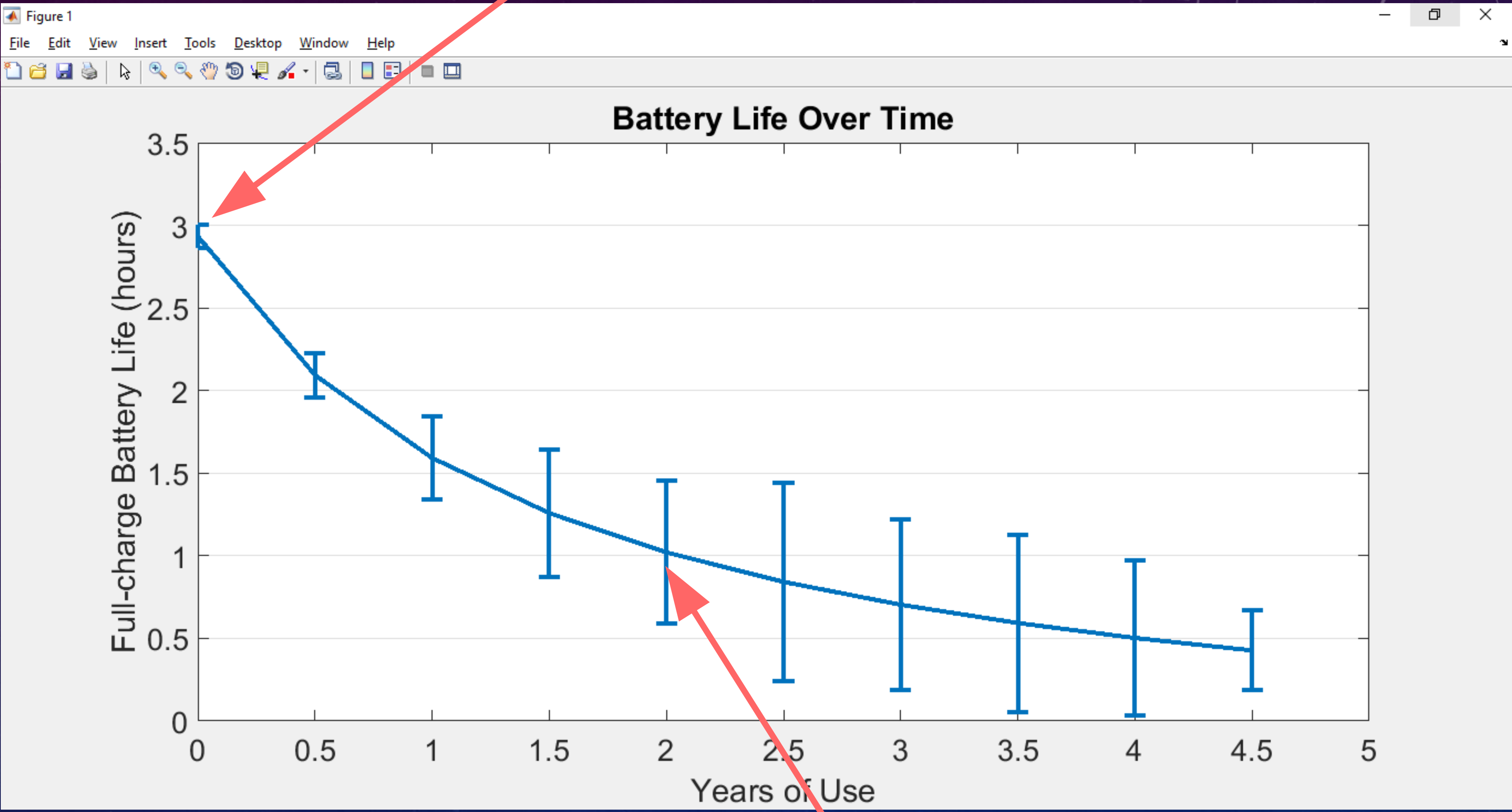
BREAK: to follow along

```
load('batteryLife.mat');
```

Your turn – plot with your laptop

```
% create a plot with error bars
h = errorbar(time, batteryMean, batteryStdDev);

% use the graphics object to set the line width
set(h, 'LineWidth', 3); % wider than usual for projector

% Add title and axis labels
title('Battery Life Over Time');
xlabel('Years of Use');
ylabel('Full-charge Battery Life (hours)');

% use gca to set properties
ax = gca;
ax.FontSize = 20;
ax.XLim = [0,5];
ax.YLim = [0,3.5];
ax.YGrid = 'on';
```

YES, 3hrs when new

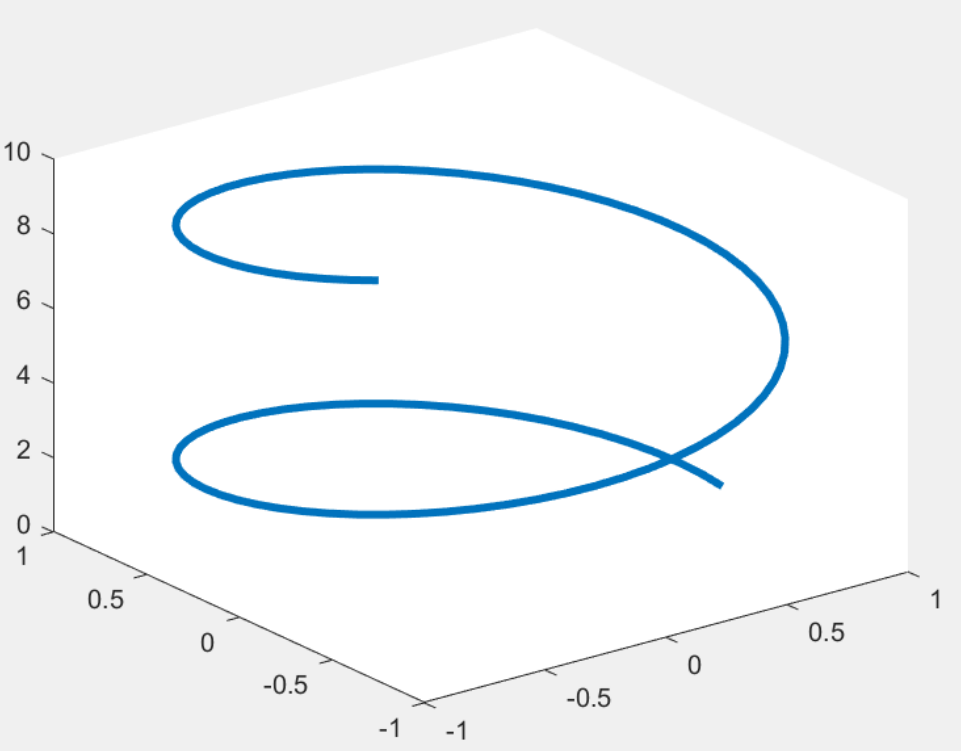


NO, <2hrs when 2yrs old

3D Plotting

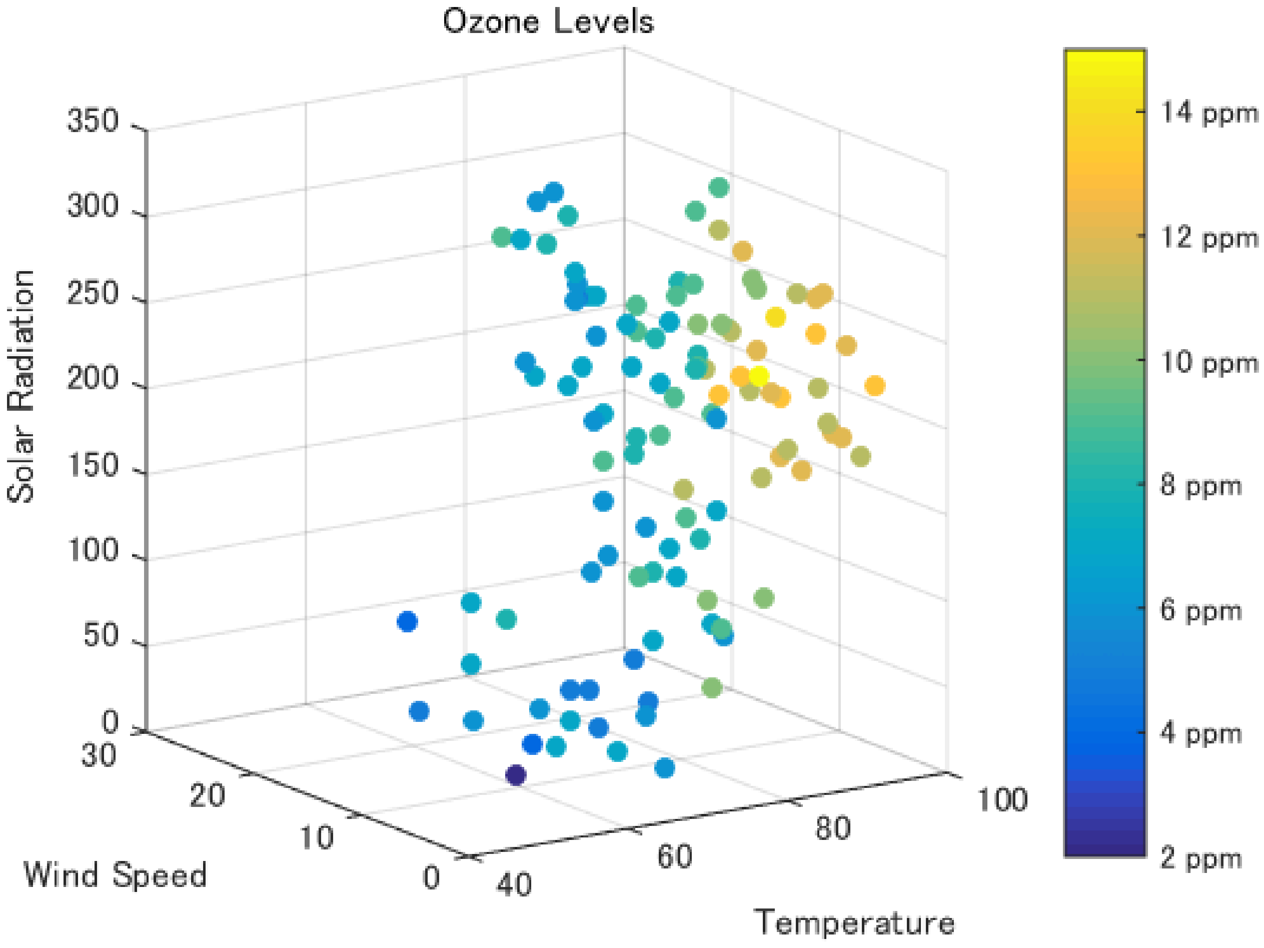
- Use the `plot3` function to create three dimensional plots.
- It works similarly to regular, 2D plotting.
  - Requires (x,y,z) input.

```
% Our third axis is time
% between 0 and 10 seconds
t = linspace(0,10,101);

% x and y vary over time
x = cos(t);
y = sin(t);

h = plot3(x, y, t);

h.LineWidth = 3; % wider than
```



scatter3

- Example: Measurements of ozone levels and other atmospheric conditions. See the MathWorks link below for more details.



Dataset: https://www.mathworks.com/matlabcentral/fileexchange/ 35288-matlab-plot-gallery-scatter-plot-3d/content/html/Scatter_Plot_3D.html

Break Time

We'll start again in 5 minutes.

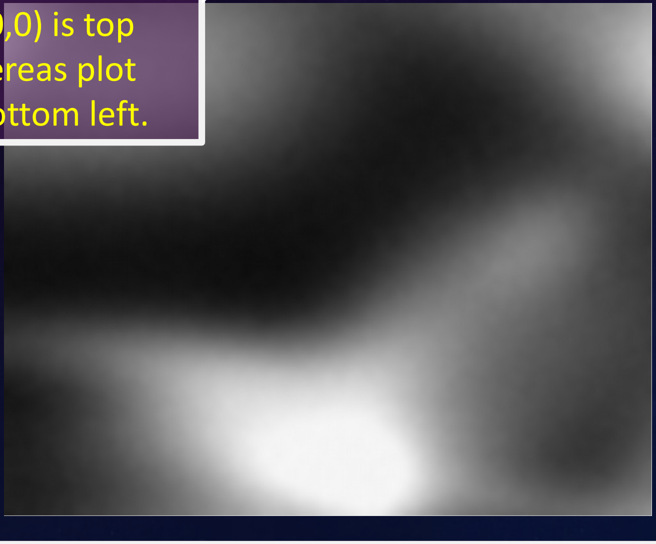
Download the Project 3 overview from 00_Todays_Lecture

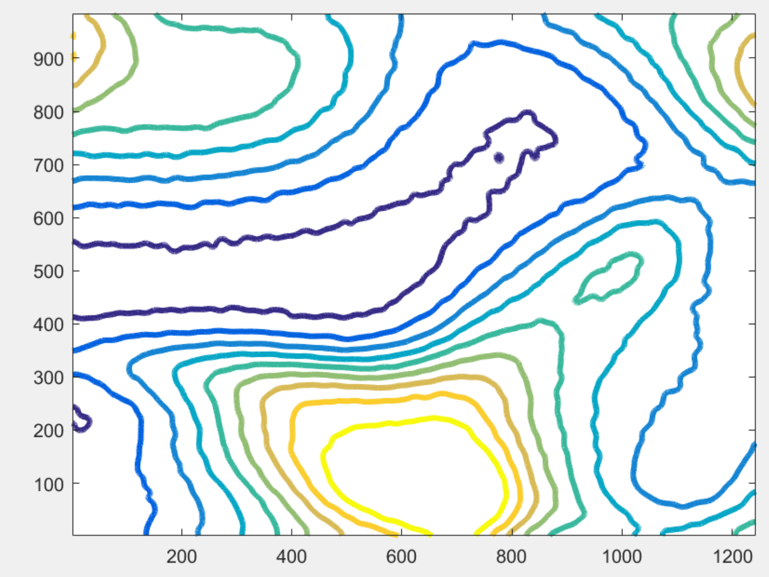
Making Contour Maps

- Use the contour function to create a contour map.
- Each line represents locations with equal Z values.

```
% Example from project 2
Z = removeNoise(scan_radiation(30));
contour(flipud(Z));
```

Image (0,0) is top left, whereas plot (0,0) is bottom left.



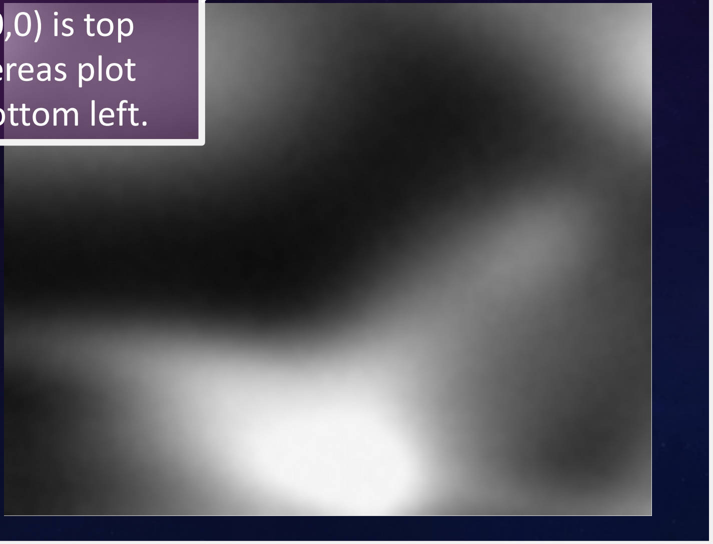


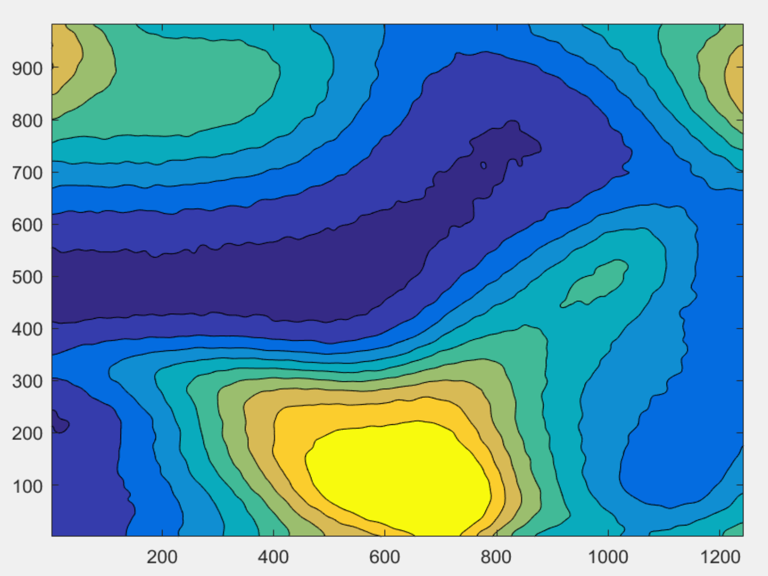
Making Contour Maps

- contourf works like contour, but fills in regions with color.
- Each line represents locations with equal Z values.

```
% Example from project 2
Z = removeNoise(scan_radiation(30));
contourf(flipud(Z));
```

Image (0,0) is top left, whereas plot (0,0) is bottom left.



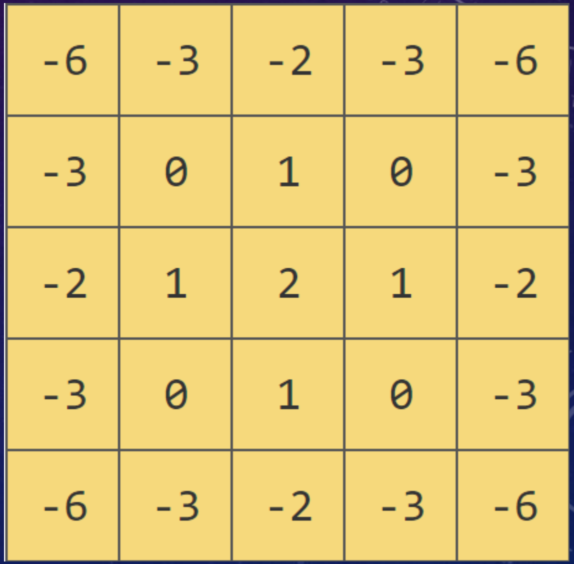


Plotting Functions of Two Variables in 3D

- Consider the mathematical function:

$$z = 2 - (x^2 + y^2)$$

- How could we plot this in MATLAB?
- Remember, MATLAB does NOT plot math functions – it plots x, y, and z data points. We need to calculate the matrix for Z from the x and y coordinates.



-6	-3	-2	-3	-6
-3	0	1	0	-3
-2	1	2	1	-2
-3	0	1	0	-3
-6	-3	-2	-3	-6

Z

meshgrid

x	-2	-1	0	1	2
y	-2	-1	0	1	2

- The meshgrid function is quite useful for calculating functions of two variables. In this example, let Z = X + Y

```
x = -2:1:2;    % x is the x-coordinate of matrix cells
y = -2:1:2;    % y is the y-coordinate of matrix cells
[X,Y] = meshgrid(x,y);    % X is a matrix of x at each (x,y)
Z = X + Y;                % Y is a matrix of y at each (x,y)
```

-2	-1	0	1	2
-2	-1	0	1	2
-2	-1	0	1	2
-2	-1	0	1	2
-2	-1	0	1	2

X

-2	-2	-2	-2	-2
-1	-1	-1	-1	-1
0	0	0	0	0
1	1	1	1	1
2	2	2	2	2

Y

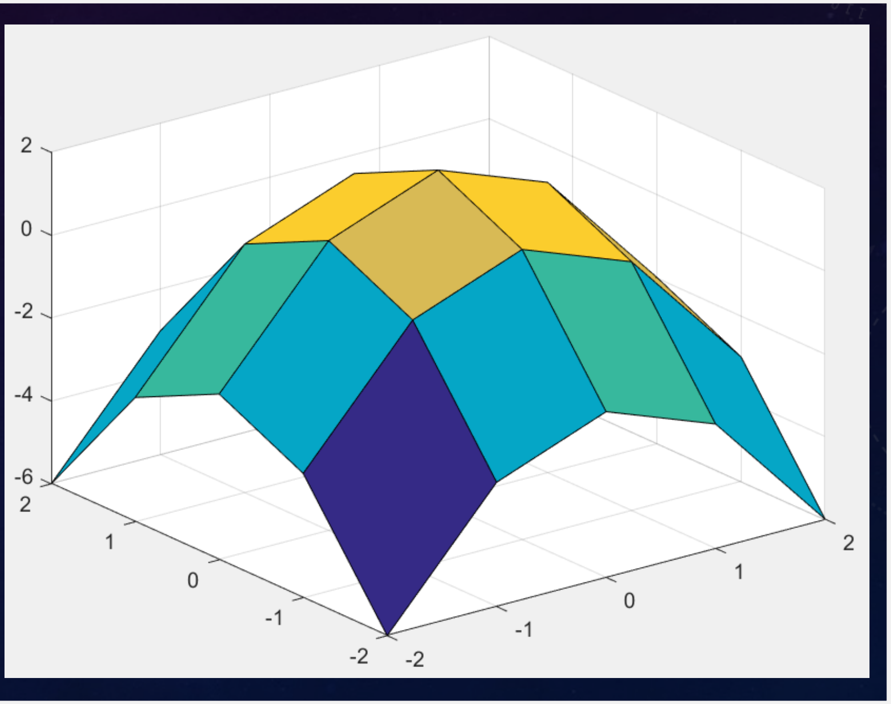
-4	-3	-2	-1	0
-3	-2	-1	0	1
-2	-1	0	1	2
-1	0	1	2	3
0	1	2	3	4

Z

3D Surface Plots

- Use the `surf` function to create 3D surface plots.

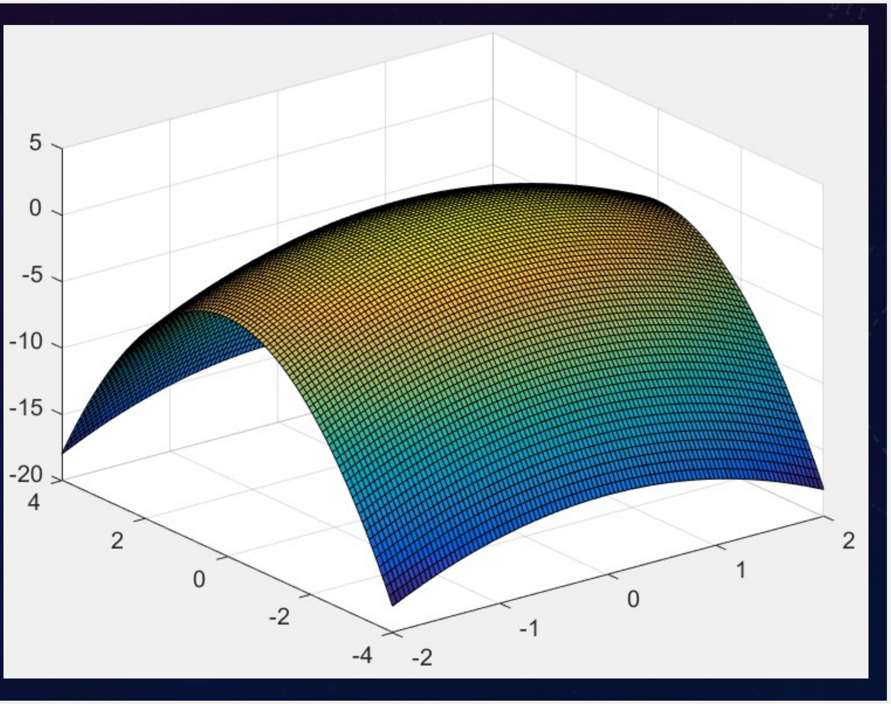
```
x = -2:1:2;
y = -2:1:2;
[X,Y] = meshgrid(x,y);

Z = 2 - (X.^2 + Y.^2);

surf(X,Y,Z);
```



3D Surface Plots

- Use the `linspace` function for higher resolution x and y.
- x and y don't have to be the same.

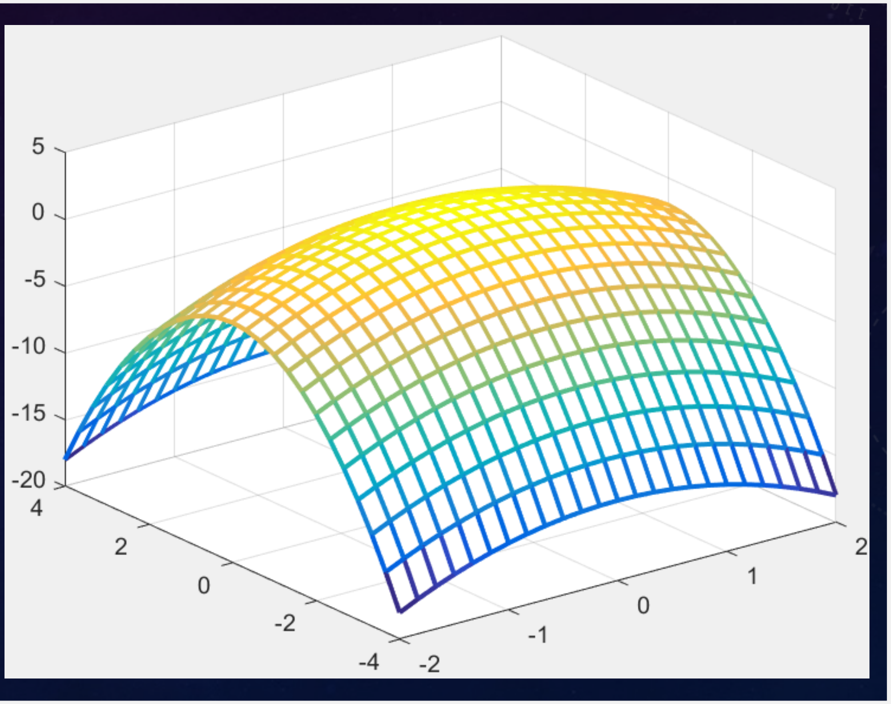
```
x = linspace(-2,2,101);
y = linspace(-4,4,101);
[X,Y] = meshgrid(x,y);

Z = 2 - (X.^2 + Y.^2);

surf(X,Y,Z);
```



3D Mesh Plots

- The mesh function works similarly to surf, but does not color in sections of the surface.

```
x = linspace(-2,2,26);
y = linspace(-4,4,26);
[X,Y] = meshgrid(x,y);

Z = 2 - (X.^2 + Y.^2);

h = mesh(X,Y,Z);
h.LineWidth = 2;
```



Project 3 overview