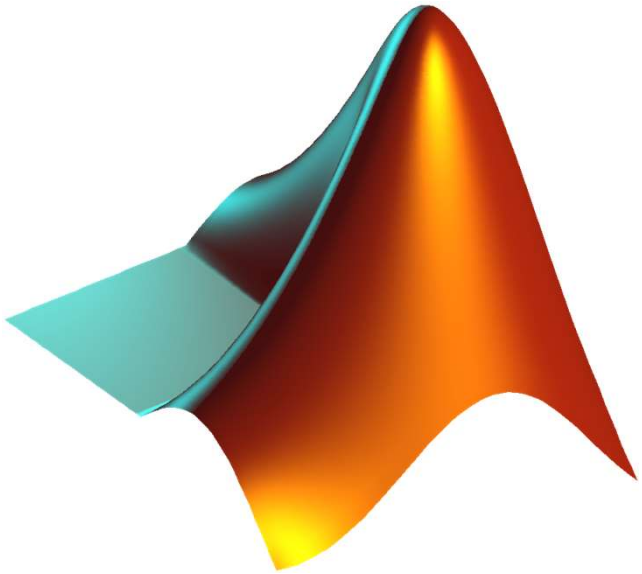
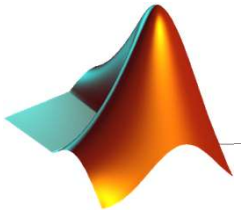
McMaster University



An Introduction to Matlab

By: Eric Dyer, MASc Electrical & Computer Engineering
Dec 1, 2018

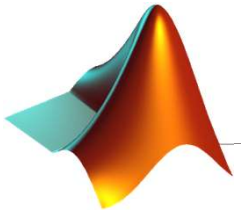


Workshop Objectives

McMaster University

- This workshop is intended to be an introductory course to the MATLAB programming language

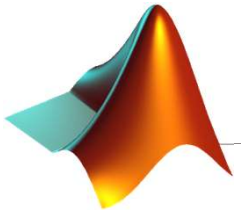
- Objectives
  - Understand where MATLAB may be used as an effective tool
  - Understand basic operations and functionalities
    - File I/O
    - Structures (namely matrices)
    - Symbolic math
    - Plotting
    - Built-in functions
  - Utilize learned skills in a project format & see how far you can get!



Some Disclaimers

McMaster University

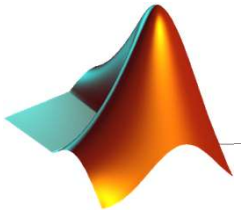
- I was given the somewhat open ended task of leading a beginners MATLAB workshop for a group of graduate students
- This class discusses the **RUDIMENTARY** essentials of MATLAB and is not meant to be an advanced course
- I will still do my best to try and appeal to a range of skill levels with the projects we work on today



Worshop Structure

McMaster University

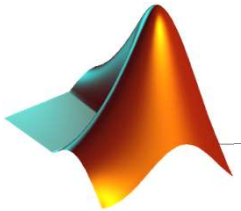
- What is MATLAB and why do I care?
- MATLAB in my own research
- MATLAB Basics (majority of the teaching portion)
  - Environment navigation
  - Variables
  - Vectors
  - Matrices
  - Loops
  - File I/O
  - Plotting



Worshop Structure

McMaster University

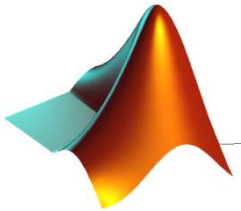
- Practice problem #1 involving reading in data from a file and plotting its data
- Symbolic Toolbox
- Practice problem #2 involving reading in a .wav file and trying different signal processing techniques
- Practice problem #3 fits a model using regression techniques
- I have a few other examples we can look at if people are interested
- Q&A to discuss any content I have covered as well as brainstorm how MATLAB could be useful in your research



Where MATLAB Shines

McMaster University

- MATLAB is an interpreted language, giving it some unique advantages
  - Easy to write/modify/test scripts quickly
  - Intuitive syntax
- It is a high-level language for numerical computation, visualization and application development
- It provides built-in graphics for visualizing data and tools for creating custom plots
- It provides tools for building applications with custom graphical interfaces
- Efficient with linear algebra operations as well as sparse matrix math



Where MATLAB Shines

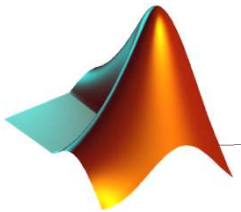
McMaster University

- Rich libraries of math functions and common algorithms
  - Signal/Image processing
  - Data analysis & Deep learning

- Widely adopted by the research community in a wide range of applications
  - Signal Processing and Communications
  - Image and Video Processing
  - Control Systems
  - Test and Measurement
  - Computational Finance
  - Computational Biology

When another tool may be better..

- Although versatile, MATLAB may not be the optimal choice for every problem
- Carefully consider if another language may be more effective if any of the following apply
  - Hard real-time constraints
  - Data acquisition requires expensive, proprietary equipment to work with MATLAB
  - Code must be interfaced with other non-MATLAB systems
  - Code must run on resource sparse or embedded systems



Don’t Forget About Simulink!

McMaster University

- Simulink is a graphical programming environment for modeling, simulating and analyzing multidomain dynamical systems

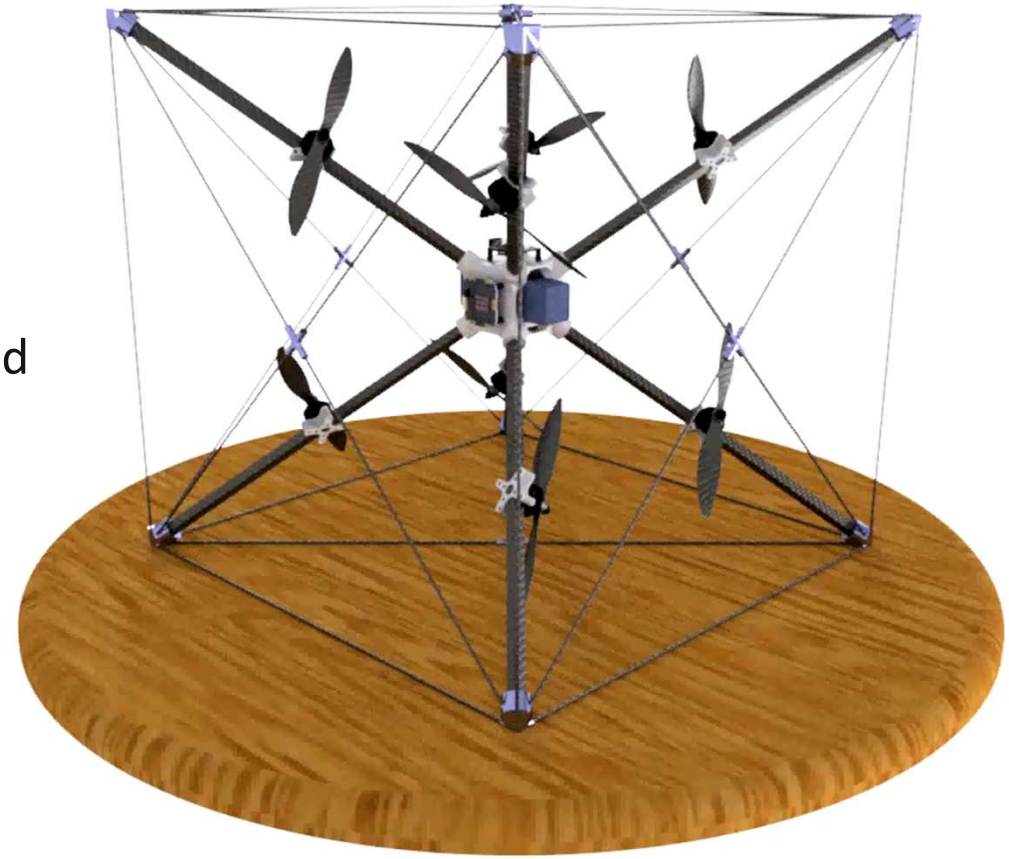
- Just like Matlab, there are a plethora of plugins and libraries available allowing you to connect to in to real-time sensor platforms or simulate mechanical systems

- Applications of Simulink include
  - Simulation of mechanical systems (SimMechanics Toolbox)
  - Real-time control and data collection from sensor systems (QUARC Toolbox)
  - HDL code generation for FPGAs
  - Development of state machines and flowcharts (Stateflow)

MATLAB & Simulink in my own research

McMaster University

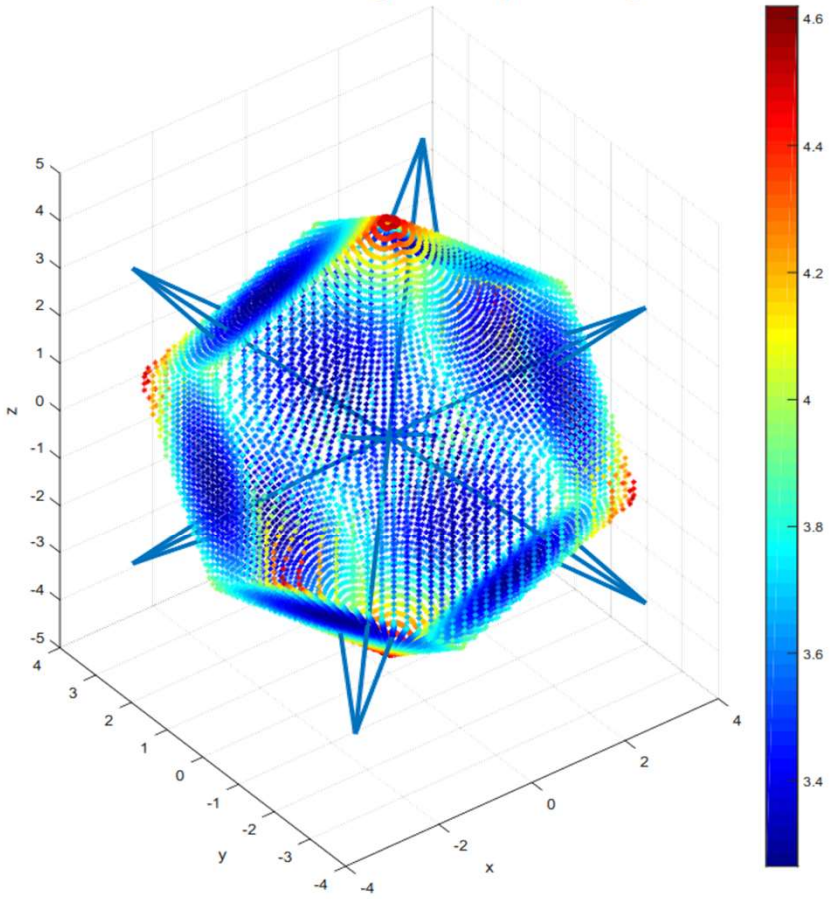
- Design of a fully actuated aerial vehicle with even force-torque distribution
- The design of the vehicle and its control algorithm must be verified using MATLAB & Simulink



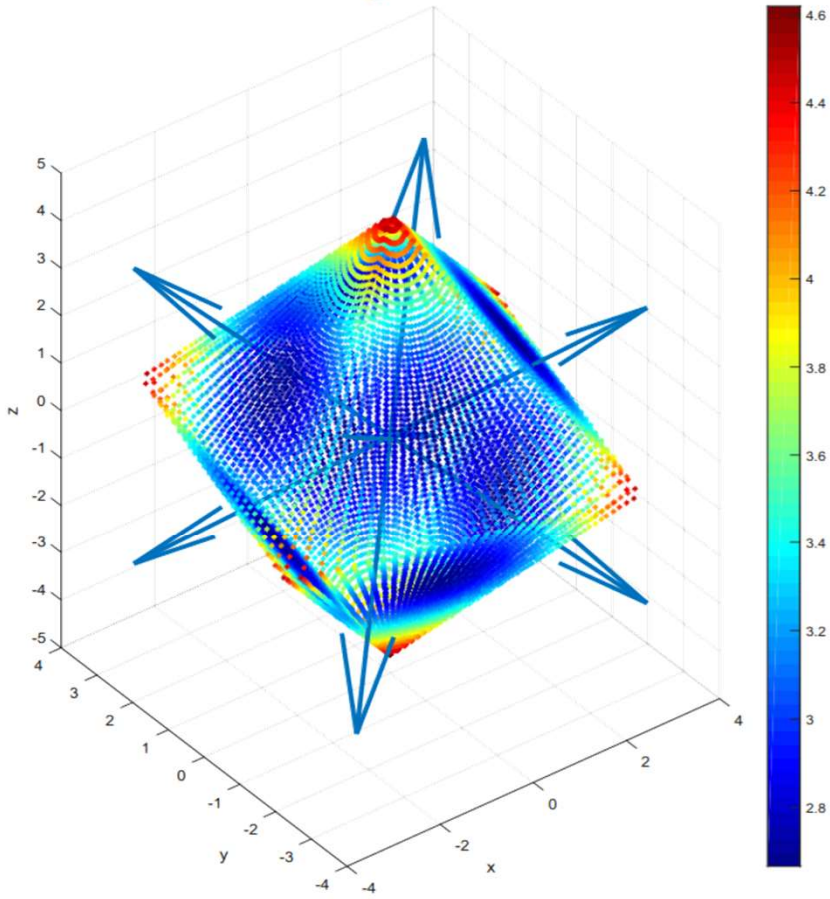
MATLAB & Simulink in my own research

McMaster University

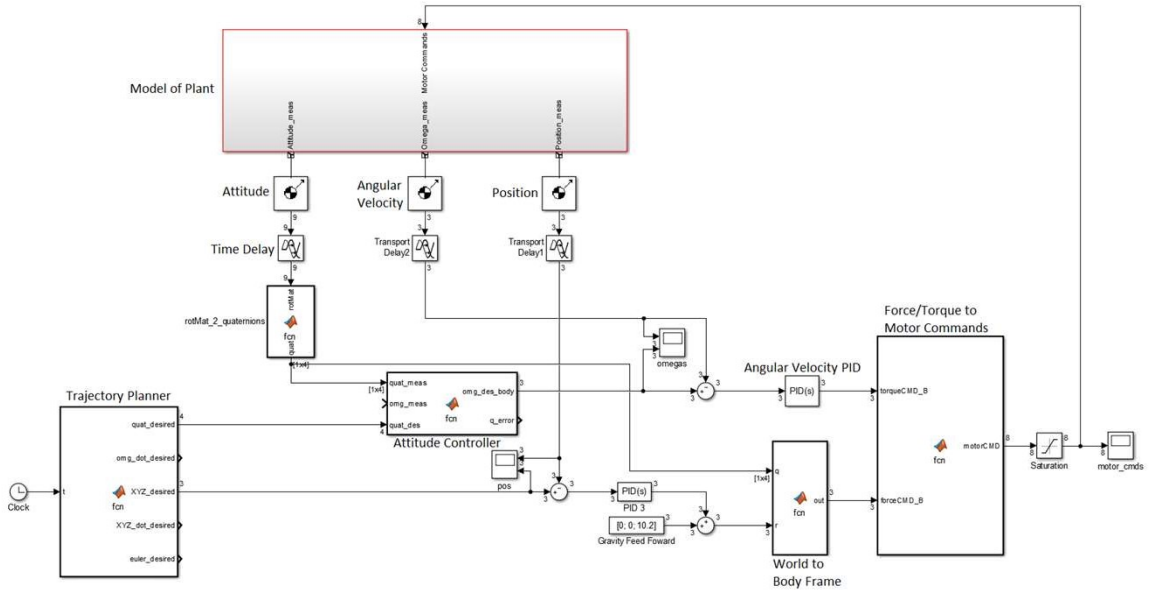
Force Distribution Using Nullspace Optimization

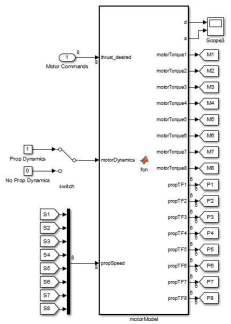


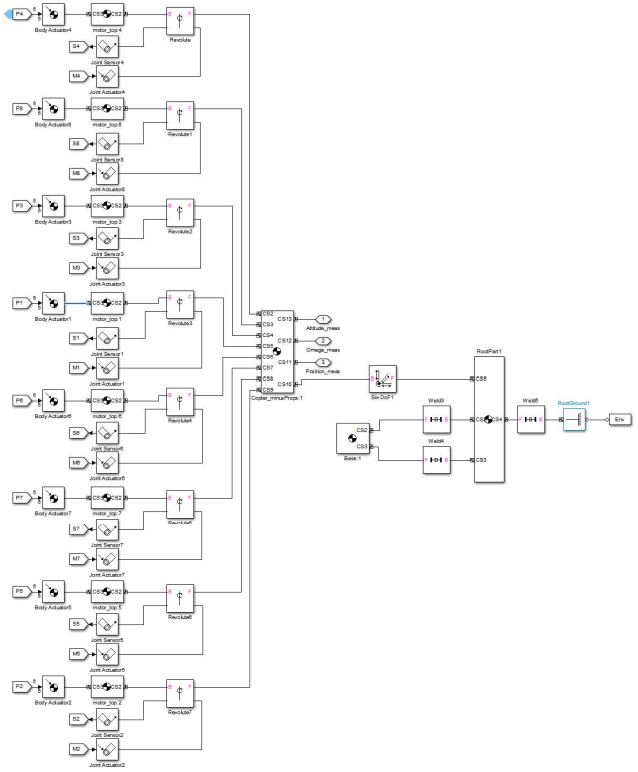
Force Distribution Using Minimum Norm Solution

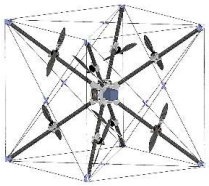


MATLAB & Simulink in my own research





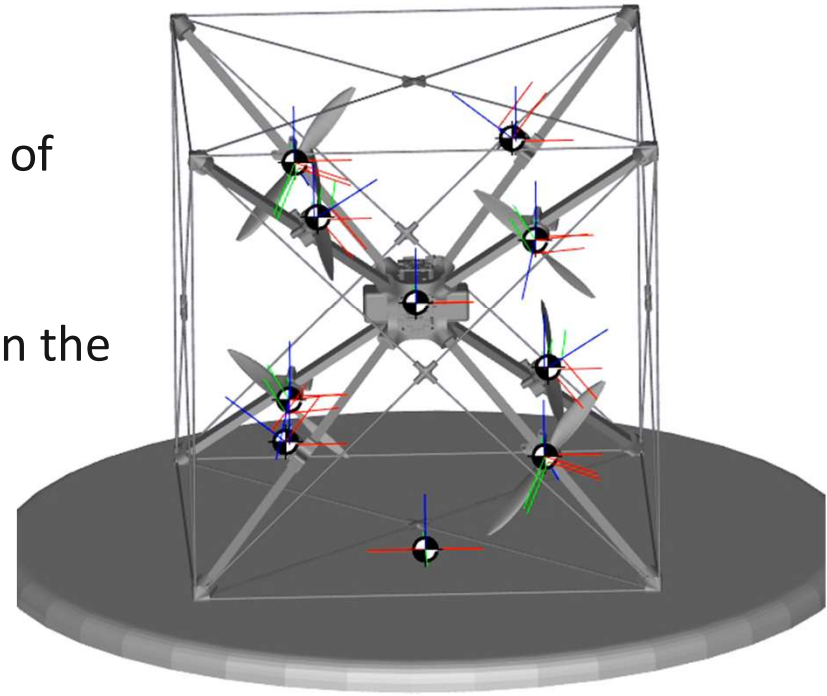




III) Control: Simulation

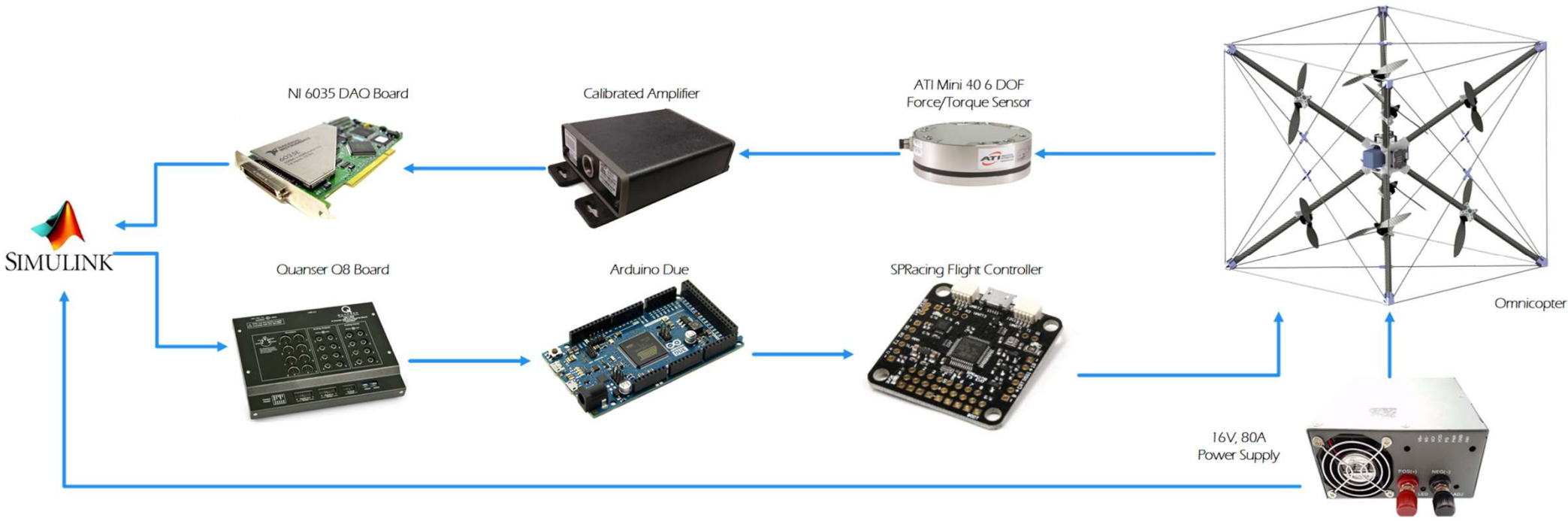
McMaster University

- The control architecture is verified in Simulink
- The SimMechanics tool box allows for simulation of the true omnicopter CAD
- Actuator dynamics and saturations are included in the model
- The effects sensor delay are also studied



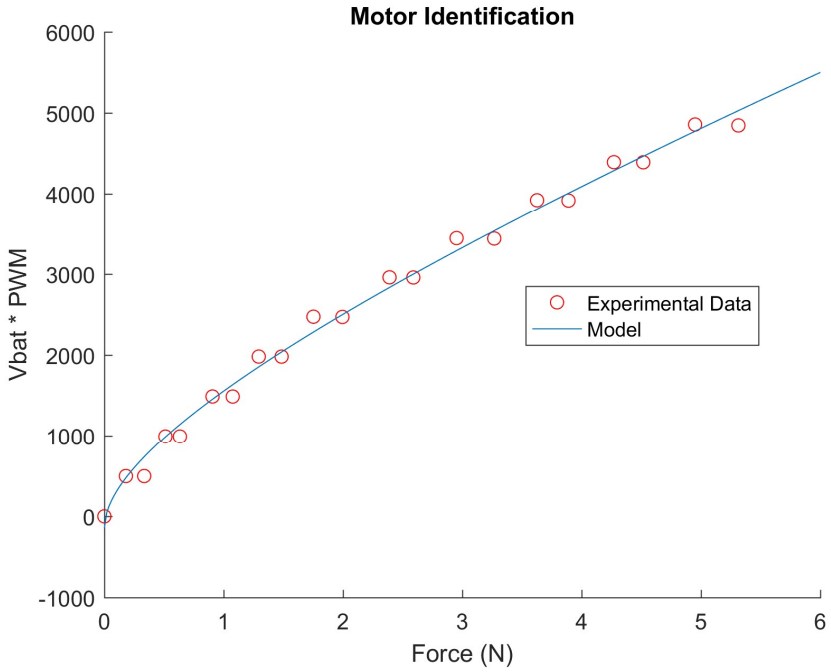
MATLAB & Simulink in my own research

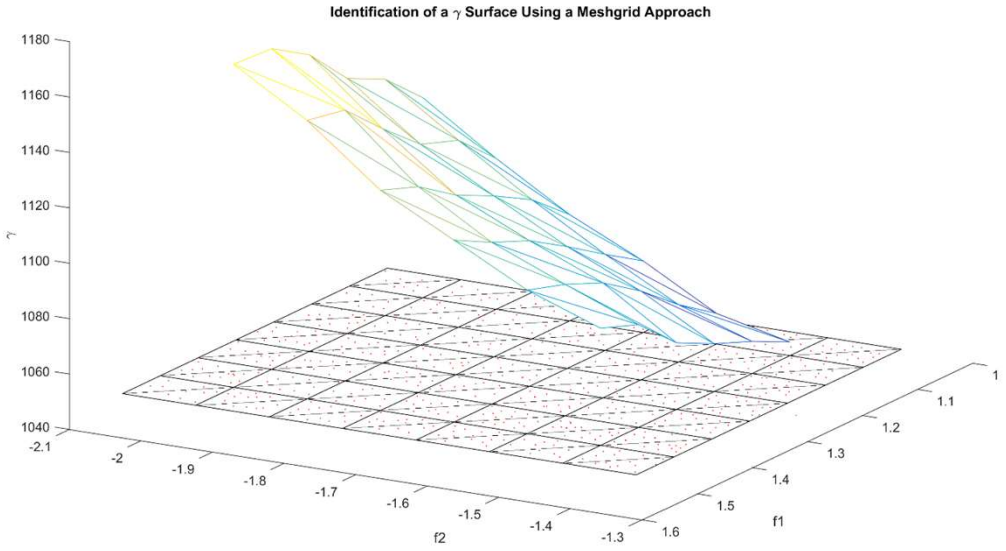
McMaster University

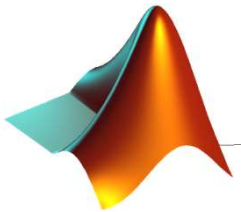


MATLAB & Simulink in my own research

McMaster University







I) Basics

McMaster University

- The Matlab environment
- Matrices & operaters
- Basic file I/O & data management

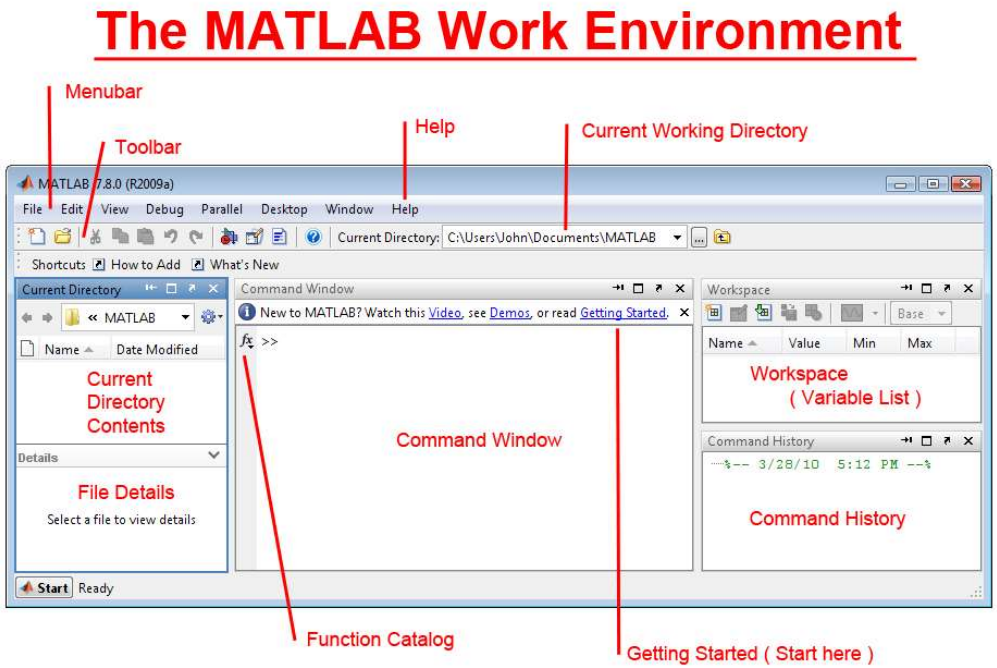
- Built in mathematical operations

- General language semantics
- Plotting

- User Interface & Interaction

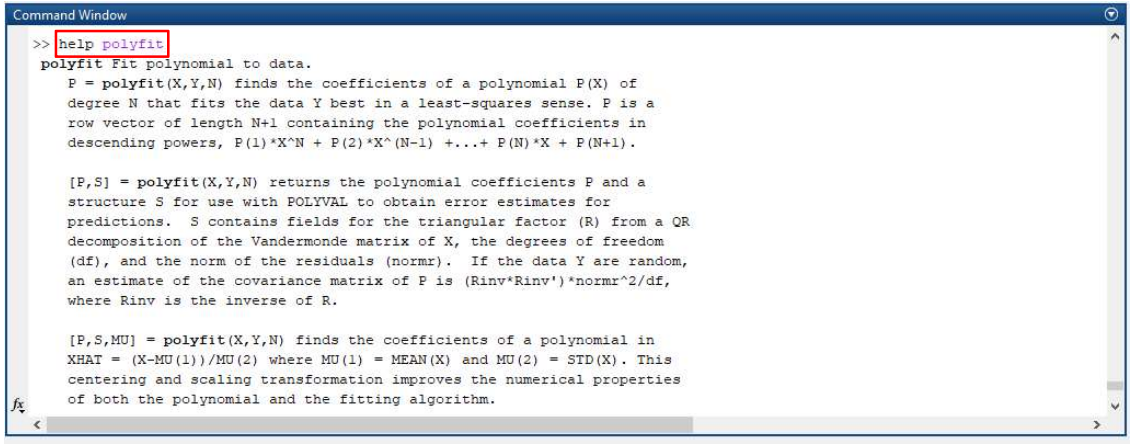
I) Basics

- Take a moment now to start up MATLAB
- Lets familiarize ourselves with the environment



I) Basics: Before we go any further..

- MATLAB has a invaluable help function to provide documentation for any built in functions
- Simply type help <function name> in the console



- Documentation is also available at https://www.mathworks.com/help/matlab/

I) Basics: Command Window Practice

- Test a valid expression in the console
  - 5 + 5 (press enter)
- Test some other mathematical operations
  - Eg. $3 \cdot 2$, 3^2, $\sin\left(\frac{\pi}{2}\right)$, $\sqrt{-1}$, etc
- Try and produce the NaN and Inf results
- The ; operator may be used to suppress program output
- % allows for writing comments
  - %% helps to write block comments and separate code

Do comment your code – you'll thank yourself later

I) Basics: Variables

McMaster University

- Note that MATLAB is CASE SENSITIVE – so be careful to be accurate when naming your variables!

- Valid variable names:
  - Start with a letter
  - Can be followed by letters, digits or underscores
  - Max variable name length = value of namelength max
  - Cannot be a MATLAB keyword
  - Check whether a variable name already exists using

```
exist checkname
```

Valid Names	Invalid Names
x6	6x
lastValue (camelCase)	end, if, pi
n_factorial (snake_case)	n!

I) Basics: Variables

McMaster University

- Try saving some data to a variable and then manipulating the variable

```
x = 3
y = 6
z = x / y
```

- You can use the `ans` keyword to access the answer from the last command

```
ans + 1
```

- Try the who and whos commands to display varaibles you have used

I) Basics: Vectors

McMaster University

- You can assign a row vector to a variable using square brackets

```
x = [1 2 3 4 5]
```

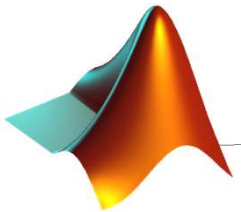
- A column vector is created in a similar way, except you use semi colons to separate the numbers

```
x = [1; 2; 3; 4; 5]
```

- You can access an item in this vector via

```
x(index)
```

- Unlike other languages, MATLAB starts with an index of 1 (this will make many non-programmers very happy, not so much for me)



I) Basics: Matrices

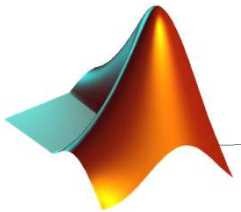
McMaster University

- One of the most useful objects in MATLAB is a matrix
- A matrix is produced as follows

```
A = [1 2 3; 4 5 6; 7 8 9]
    or
A = [1, 2, 3; 4, 5, 6; 7, 8, 9]
```

- There are a few other ways to create matrices in MATLAB

```
B = magic(m)
B = ones(m) or ones(m,n)
B = zeros(m) or zeros(m,n)
B = rand(m), rand(m,n)
```



I) Basics: Matrices - Indexing

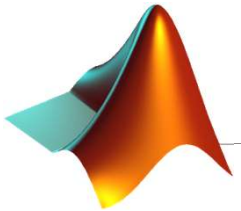
McMaster University

- For a 2D matrix A, an item is referenced as

```
A(row, col)
```

- The ':' operator can be used to specify a range
  - `A(:,:) = A`
  - `A(2:,3)`
  - `A(2, :)`

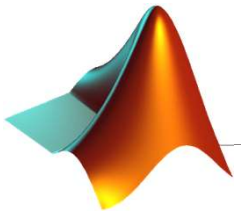
- Experiment with indexing to extract row and column vectors



I) Basics: Matrices - Operations

McMaster University

Operator	Action
+	Element-wise addition
-	Element-wise subtraction
*	Standard Matrix Multiplication
.*	Element-wise multiplication
\	Division (ie. A*x = b -> x = A\b)
‘	Transpose
eig(A)	Find eigenvalues of matrix
svd(A)	Find singular values of a matrix
det(A)	Find determinant of a matrix
[A, B] or [A; B]	Concatenate matrices

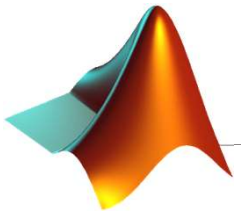


I) Basics: Matrices - Indexing

McMaster University

Practice:

1. Create a matrix A with dimensions 3x4 of random numbers spanning between 1-100
2. Create a second matrix B with dimensions 5x5 using the magic function
3. Assign a 3x4 subset of B to a new matrix C
4. Multiply A by the transpose of C
5. Find the singular values of C



I) Basics: Scripting

McMaster University

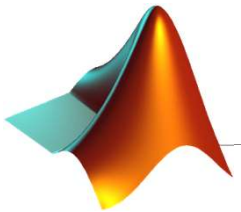
- It would be nice if we could execute all the commands in the practice at once
- This becomes invaluable when executing a large set of consecutive commands

- Since MATLAB is an interpreted language, it is easy to transpose the set of commands you entered in the command window into a script

`disp(S)`

- Create a new script
- Enter all commands you used in the practice on consecutive lines within the script
- If you would like the script to run silently, make sure to add semicolons to the end of each line
- Save and run this script (F5)

- You can explicitly display the value of a variable using `disp(<var name>)`



I) Basics: Session Commands

McMaster University

- MATLAB natively saves the session as you continue to run scripts
- This means that variables in the workspace are not automatically cleared the next time you run a script (this may cause issues)

- There are a few session commands you can execute to clear these artifacts

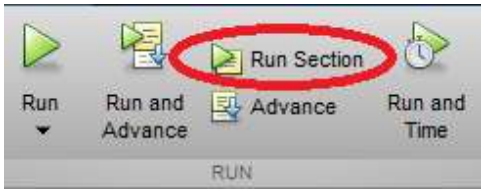
Command	Result
clc	Clears command window
clear	Removes variabls from memory
close	Closes all open figures

- We can add these commands to the top of our script if we would like them to be executed every time (I generally find this is good practice)

I) Basics: Comments

- Comments are critical to code understanding
- You can start a comment using the %

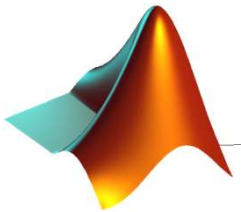
```
% This is a comment
```




```
%% A double Percent adds a comment block allowing different chunks of your script to be run independently
```

- Comment your code now

- Experiment with the block comment and the run section button to run individual blocks of code
  - This could be very useful if you want to replot output data but don't want to run a time-hungry analysis again



I) Basics: Loops

McMaster University

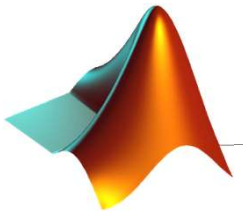
- Syntax

```
for index = values
     statements
end
```

- Example try

```
for i = 1:10
     disp(i)
end
```

- Practice: Use a loop to populate a vector of size 10 that contains the squares of 1 – 10
- Practice: Any ideas on another way to do this?



I) Basics: Loops

McMaster University

- While loop

```
while <expression>
    <loop contents>
    <loop contents>
end
```

- For loop

```
for i = 1:0.1:10
     <loop contents>
     <loop contents>
end
```

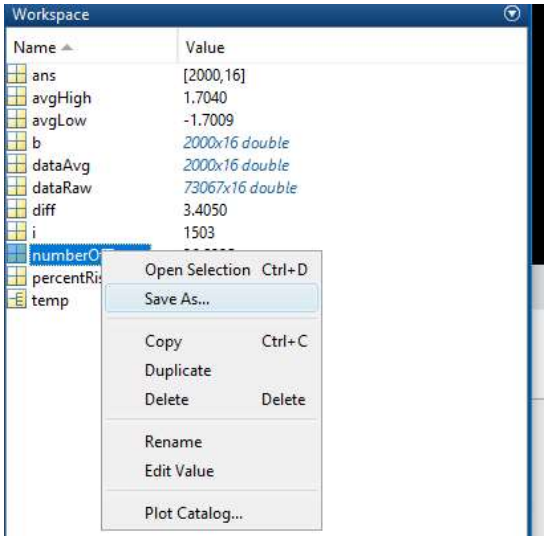
I) Basics: Input/Output

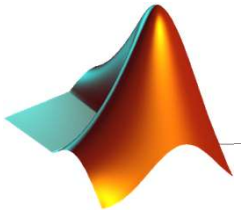
- Now that we understand MATLAB's fundamental building block, it would be very useful to be able to read/store external data between MATLAB sessions

- Matrices may be stored in .mat files and loaded via:

```
A = load('filename.mat')
```

- Variables and matrices used in the current session are displayed in the workspace pane
- Matrices in this pane can be saved





I) Basics: Input/Output

McMaster University

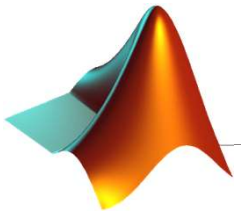
- Another very useful and platform-agnostic data format is the comma separated list
- These are generally stored in .csv files and can be read in via

```
A = csvread('filename.csv')
```

- Each row of the .csv file is read in as a row in the matrix A
- Data may also be written to the current directory via:

```
csvwrite('filename.csv', A)
```

- You can also specify a full file path to the file if in another directory or use the cd command to change the active directory to somewhere else
- Generally it is recommend to keep data close to the .mat file, although splitting it into a separate subdirectory is often helpful



I) Basics: Functions

McMaster University

- Before 2016b, each function needed to be in its own separate file (the name of the file should match the function name)

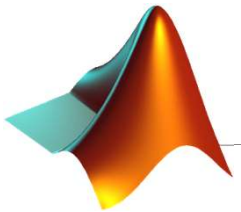
```
function [y1,...,yN] = myfun(x1,...,xM)
```

- For versions 2016b and on, functions can be at the end of script

```
x = 3;
y = 2;
z = perm(x,y)

function p = perm(n,r)
    p = fact(n)*fact(n-r);
end

function f = fact(n)
    f = prod(1:n);
end
```



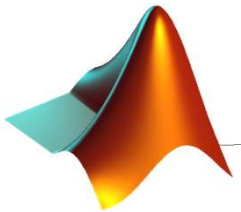
I) Basics: Plotting

McMaster University

- Plotting is an invaluable feature of MATLAB – and its seamless integration into your script is one of the primary reasons MATLAB is the tool of choice for researchers

- At its most basic level you can create a simple plot using
  plot(y) or plot(x, y)

- Lets plot the function $f(x) = 3x^3 - 4x^2 + 5x - 10$
  - Create a number series from -3 to 4 spaced in increments of 0.1
  - Calculate the y vector given x
  - Plot the result



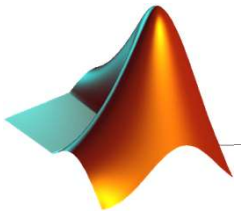
I) Basics: Plotting - Formatting

McMaster University

- To make sure the plot is in a new figure, start the plot block off with a figure statement
- We can also place multiple plots into a single figure using the subplot function

- Example

```
x = -5:0.1:5;
y = sin(x);
y2 = cos(x);
figure;
subplot(1,2,1);
plot(x,y);
subplot(1,2,2);
plot(x,y2);
```



I) Basics: Plotting - Formatting

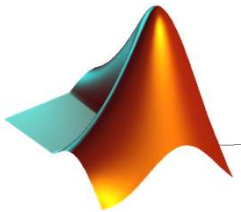
McMaster University

- What if we want to plot multiple sets on the same graph?

```
Plot(x, y1, x, y2)
```

- You can also use the hold function

```
hold on
plot(x, y1)
plot(x, y2)
hold off
```



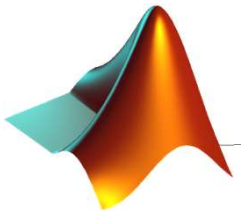
I) Basics: Plotting - Formatting

McMaster University

- Some other formatting methods include

```
legend('label1',...,'labelN')
xlabel('label')
ylabel('label')
title('plot title')
```

- **Note If you have been running this script multiple times without running close all, you may have noticed that your taskbar has become cluttered with tons of figures
- It may be useful to add the command **`close all`** to the beginning of your script to close all open figures each time you execute your code



I) Basics: Plotting - Formatting

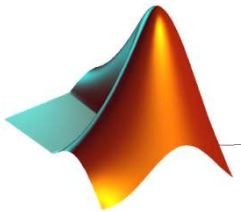
McMaster University

- You may also wish to explicitly format the marker shape and colour of your datapoint

Line Style	Description
-	Solid line (default)
--	Dashed line
:	Dotted line
-.	Dash-dot line

Color	Description
y	yellow
m	magenta
c	cyan
r	red
g	green
b	blue
w	white
k	black

Marker	Description
o	Circle
+	Plus sign
*	Asterisk
.	Point
x	Cross
s	Square
d	Diamond
^	Upward-pointing triangle
v	Downward-pointing triangle
>	Right-pointing triangle
<	Left-pointing triangle
p	Pentagram
h	Hexagram



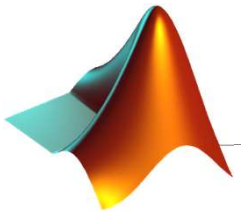
I) Basics: Plotting - Formatting

McMaster University

- You can also set things like marker size and marker edge color, etc

```
plot(x, y, '—gs', 'MarkerSize', 2,…
                  'MarkerEdgeColor', 'b')
```

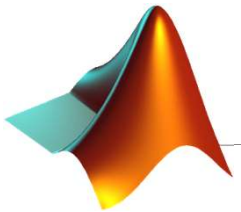
- You can save your plots from within the figure by clicking file -> save and selecting the type of file you'd like to save as
  - Generally, it is good practice to save files as vector images (.pdf, .svg, .tiff) to avoid quality issues in scaling for the future
  - You may also save as a .fig, which is the MATLAB figure file type



I) Basics: Plotting - Practice

McMaster University

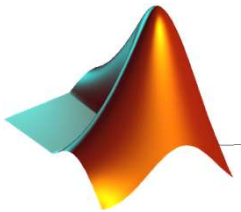
- Now lets practice importing some data and plotting it in a meaningful way
- We will be plotting data from weather-history.csv
- First, open the file with csvread()
  - Note that you may have an issue using the function vanilla – take a look at the docs and see if there's anything that may help you out
- Create a single figure with 3 plots with labelled axes and a legend (if necessary)
  - The 1st plot should show both minimum and maximum temperatures
  - The 2nd plot should show precipitation (try using a bar graph)
  - The 3rd plot will show average wind speed, fastest 2 min & 5 min wind speeds



I) Basics: Plotting - Practice

McMaster University

- **Challenge**
  - See if you can smooth the temperature data
  - You can use a built in moving average filter or a low pass filter OR trying implementing a moving average filter yourself!



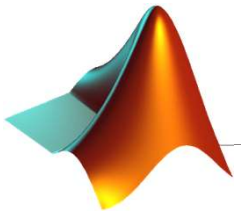
I) Basics: 3D Plotting

McMaster University

- MATLAB is also a fantastic tool for producing 3D plots
- The surf(X, Y, Z) function can be used to produce 3D surfaces
- Start by producing an X/Y meshgrid via

```
[X,Y] = meshgrid(-5:0.5:5)
```

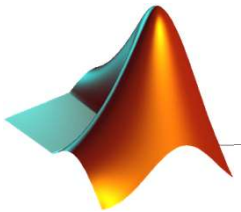
- Try plotting: $\frac{\sin\left(0.3\left(x^2+y^2\right)\right)}{10}$



II) Symbolic Toolbox

McMaster University

- MATLAB also has the ability to do symbolic math
- This can be super useful to avoid grinding through large matrices for general equations
- Only going to briefly touch on the subject at this time, but it is useful to know that the resource is available and can be used for a variety of operations including
  - Integration, differentiation, and other calculus
  - Simplification, substitution, and solving
  - Linear algebra
  - Plotting analytical functions



II) Symbolic Toolbox

McMaster University

- You can declare a variable as symbolic using the syms command

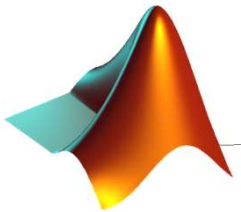
```
pi/6 + pi/4
syms(pi/6) + syms(pi/4)
```

- Create and evaluate functions

```
syms f(x)
f(x) = x^3 - 2*x^2 + 4
f(3)
```

- Find the intersection between two lines using `solve`

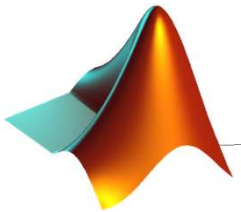
```
syms y1 y2 x
y1 = x+3;
y2 = 3*x;
solve(y1 == y2)
```



II) Symbolic Toolbox – A Practical Example

McMaster University

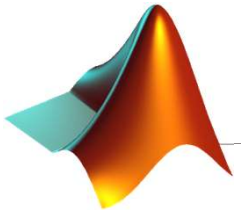
- Open up the Generalized_Jacobian_Symbolic.m file to see an example of how I used the symbolic toolbox in my research
- I needed to write a script that mapped motor thrusts/torques on the omnicopter to the net force/torque output
- If you run the file you'll see that the generated matrix is abosultely massive
- This is where it is very useful to the MATLAB to compute your generalized equations to avoid human error



II) Symbolic Toolbox – A Practical Example

McMaster University

- Open up the Generalized_Jacobian_Symbolic.m file to see an example of how I used the symbolic toolbox in my research
- I needed to write a script that mapped motor thrusts/torques on the omnicopter to the net force/torque output
- If you run the file you'll see that the generated matrix is abosultely massive
- This is where it is very useful to the MATLAB to compute your generalized equations to avoid human error

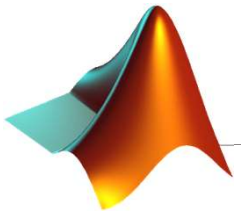


III) Signal Processing

McMaster University

- Lets start a new project
- We are going to read in a .wav file and do some signal processing on it
- Use the audioread function to read in the signal
- Since the .wav is stereo you will need to set signal = just one of the vectors

- Once you get it working, play around with different amounts of delay



IV) Regression Models – A real world problem

McMaster University

- It is often very useful to regress data to a model
- The problem has the form

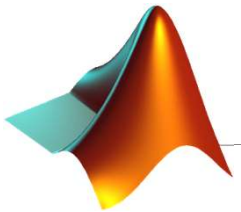
$$\boldsymbol{Ax = b}$$

- Where A is a tall matrix

- This means that you have more data points than model parameters that you are fitting to

- We would like to fit to a model of the form

$$V_{batt}|U_{pwm}| = \gamma_1|f_U| + \gamma_2\sqrt{|f_U|} + \gamma_3$$



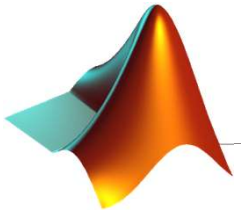
IV) Regression Models – A real world problem

McMaster University

- We would like to fit to a model of the form

$$V_{batt}|U_{pwm}| = \gamma_1|f_U| + \gamma_2\sqrt{|f_U|} + \gamma_3$$

  - V = battery voltage
  - U = motor command
  - $\gamma_1, \gamma_2, \& \gamma_3$ are the parameters to be fitted
  - f_U= the thrust produced by the motor



IV) Regression Models – A real world problem

McMaster University

- You can read the data in from motor_identification.mat
- We are trying to regress and the 3 parameters $\gamma_1, \gamma_2, \& \gamma_3$
- Therefore our Ax = b looks like

$$\begin{bmatrix} |f_1| & \sqrt{|f_1|} & 1 \\ \vdots & \vdots & \vdots \\ |f_n| & \sqrt{|f_n|} & 1 \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{bmatrix} = \begin{bmatrix} V_{batt_1} * |U_{pwm_1}| \\ \vdots \\ V_{batt_n} * |U_{pwm_n}| \end{bmatrix}$$

- We solve the equation by taking the pseudoinverse of A
- Find the gamma parameters using

$$\boldsymbol{x} = A^{\dagger}\boldsymbol{b}$$