CME 102
MATLAB Review

Tim Anderson

What is your background?

- Math background
  - Math 51? Other math class at Stanford?
  - Math class outside of Stanford?
- Programming background
  - Java (CS106A)? Python? C++ (CS106B/X)? R?
- Forgot over break? Didn’t really learn in CME100?
- Just a bad relationship with MATLAB?

What is MATLAB?

- Short for “matrix laboratory”
- Developed in mid-1980s
- Designed as easy environment for doing matrix computations
  - Does calculus-related stuff very well too
- Used **a lot** in education and industry
  - Jonathan Rosenberg (advisor to Larry Page): “If you want to work at Google, make sure you can use MATLAB”

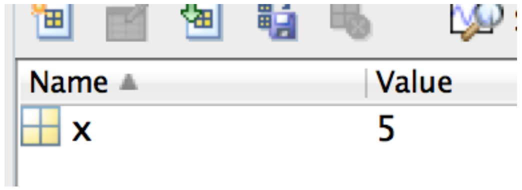
Basics

- Variables/Data types/Scope
- Mathematical operations
- Vectors (not exactly the same as in CME100)
  - Matrices
- Vector operations
- Plotting
- Loops
- Functions/Scripts

Variables

- Way to store and call up a value
  - Declare variable "x" and assign value 5 as:

```
x = 5;
```



Name ▲	Value
x	5

- Omit semicolon if you want to output value:

```
x = 5
```

And this outputs:

```
x =

        5
```

Variables

- Lets you call up the variable:

```
EDU>> x
x =
     5
```

- And you can reassign the value as-needed:

```
EDU>> x = 4
x =
     4
```

Mathematical Operations

- You can perform mathematical operations on variables. Examples:

```
EDU>> exp(x)
ans =
   54.5982

EDU>> log(x)
ans =
    1.3863
```

- Basically, MATLAB can work like a really good calculator

Vectors

- **Not** the same as in CME100
  - CME100: Vector = quantity with direction and magnitude
- Matlab: linear algebra version of vector
  - Basically n-length array of numbers
  - Good for storing series of numbers such as data values

Vectors

- <u>Examples:</u> Let's create a couple vectors
  - Type (if you have MATLAB open on your laptop):

```
x = [1   4   5   8];
y = [2   13   9   4]';
z = [1; 16; 10; 3];
```

- Can index into the vector as:

```
y(4) = 4
```

Vectors

- Output:

```
x =
        1         4         5         8
```

“Row vector”

```
y =                    z =
        2
       13                          1
        9                         16
        4                         10
                                   3
```

“Column vector”

Vectors

- Examples:

```
x = [1   4   5   8];
y = [2   13   9   4]';
z = [1; 16; 10; 3];
```

"Row vector"

Apostrophe stands for "transpose" – flips from row to column vector or vice versa

"Column vector"

Semicolons separate rows when entering data

Matrices

- 2-D arrangement of data
  - Ex:

```
A = [1  4  5  8; 2  13  9  4; 1 16 10 3];
```

  - Gives 3 x 4 matrix A:

```
A =

     1     4     5     8
     2    13     9     4
     1    16    10     3
```

- Matrix dimensions given as rows x columns

Matrices

- Can also be viewed as stacked row vectors:
  - Ex: `B = [x; y'; x];`

```
B =

     1     4     5     8
     2    13     9     4
     1     4     5     8
```

- Or side-by-side column vectors:
  - Ex: `B = [x' y x'];`

```
B =

     1     2     1
     4    13     4
     5     9     5
     8     4     8
```

- Putting vectors together like this is called **concatenation**

Vector Operations

- Operators such as `log`, `exp`, etc. act on vectors **element by element**
  - Ex:

```
exp(x) =
1.0e+03 *
       0.0027     0.0546     0.1484     2.9810
```

- Whatever type of vector (row or column) you put into an operator will come out of it

Vector Operations

- **Operations between vectors must be done on vectors with the same dimension**
- Addition and subtraction are done element-by-element (just like in CME100)
  - Ex: `x + y' =`

```
[3   17   14   12]
```

Vector Operations

- Multiplication/division can be a vector operation (outer/inner product) or element-by-element operation
  - Outer/inner products are from linear algebra and not needed for CME102

- To do multiplication, division, and exponentiation element-by-element, you must write `.*` , `./` , or `.^`
  - The period tells MATLAB that it's an element-by-element operation

Vector Operations

- Examples:

```
y.*z =              y./z =              y.^z =

      2                 2.0000              1.0e+17 *
    208                 0.8125
     90                 0.9000                0.0000
     12                 1.3333                6.6542
                                              0.0000
                                              0.0000
```

```
y(4)*z(4)               y(4)/z(4)
```

Clearing Data

- Can clear a single variable/vector/matrix:

```
clear x
```

- Or you can clear everything:

```
clear all
```

Plotting

- Plot two vectors of equal length as:

```
plot(x,y)
```

- If you need equally spaced data (you usually will for the x-axis variable), make it with either:

```
x = 0:0.1:10;
```

  – Vector from 0 to 10 spaced by 0.1; 101 elements long

  or `x = linspace(0,10,100);`

  – 100 evenly-spaced points from 0 to 10

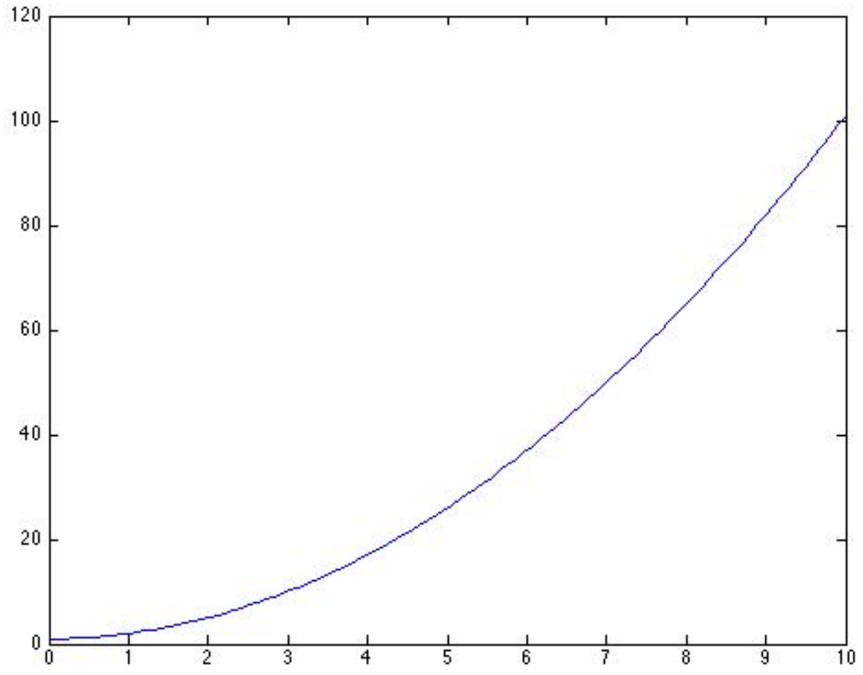
Plotting

- We can generate a y variable by

```
y = x.^2 + 1;
```

- Then plot as:

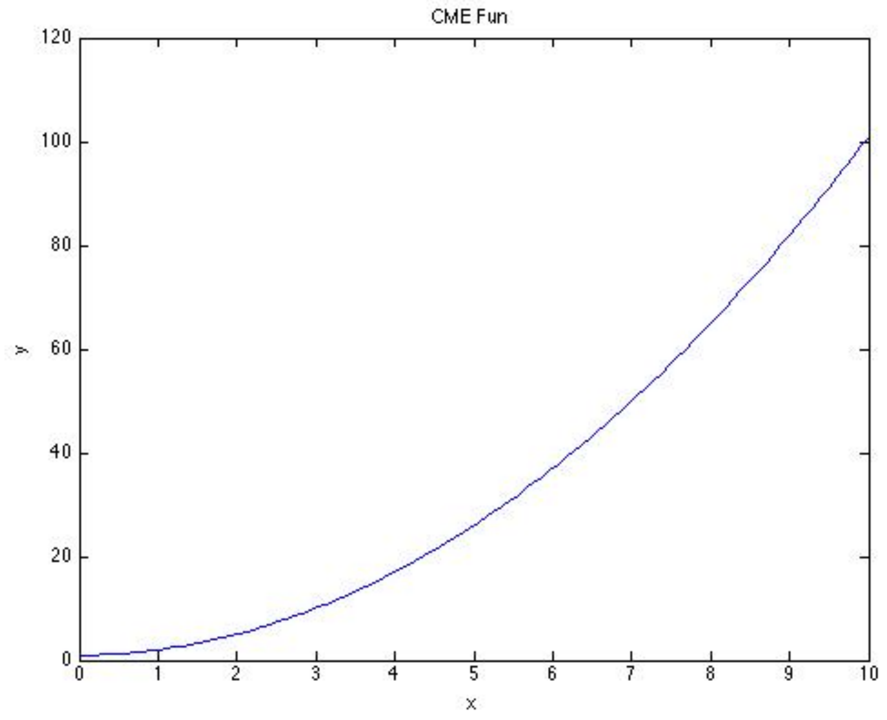
```
plot(x,y)
```



Plotting

- Label plot using:

```
xlabel('x'); ylabel('y'); title('CME Fun')
```



Plotting

- Make new plot using “figure”
- Can plot multiple things with
  `hold on` or `hold all`
  - “on” plots everything in the same color (default = blue), “all” plots each line with a different color
- Can switch between plots with
  `figure(1)`, `figure(2)`, etc.
- Close plots with:
  “`close all`” or “`close figure(1)`”

Loops

- Loops used when you have to do something repeatedly
- E.g. running a function for a changing parameter, getting something to converge below an error, etc.
- MATLAB has `for` loops and `while` loops

Loops

- `for` loop:

```
for i = 1:5
    x = x + 1;
end
```

Loop increments counter variable on each iteration over an interval

In MATLAB, must always terminate loops with "end"

- `while` loop:

```
x = 2;
while x < 6
    x = x + 1;
end
```

Loop continues until condition is met— "while true" condition gives infinite loop

Terminate loop with "end"

Loops with Conditionals

- Can break out of loop prematurely using break statement with if statement:

```
for i = 1:5
    x = x + 1;
    if x > 5
        break
    end
end
```

Code inside if statement executes if and only if condition is met

When "break" executes, it will <u>only</u> break the loop it is in, <u>not</u> all loops

Functions

- MATLAB functions come in two flavors: “library/ built-in functions” and “user-created functions”
  - Built-in/library functions are things like plot(), exp(), log(), sum(), etc.
- Making your own function can sometimes be useful for solving some problems
- **Numerically solving ODEs in MATLAB requires you to make your own functions**

Functions

- You hand a functions some parameters and it will return a **value** or **vector of values**
- Ex:

```
function m = sum_it_up(a,b)
   m = a + b;
end
```

- Use function as:

```
c = 1; d = 2;
x = sum_it_up(c,d);
```

- Output: `x = 3`

Functions

Variable where you store whatever you want the function to return

Function declaration

Function declaration

```
function m = sum_it_up(a,b)
    m = a + b;
  end
```

Parameters being passed to the function

Terminating end statement

Assign something to m, so then m is returned by the function

Functions and Scope

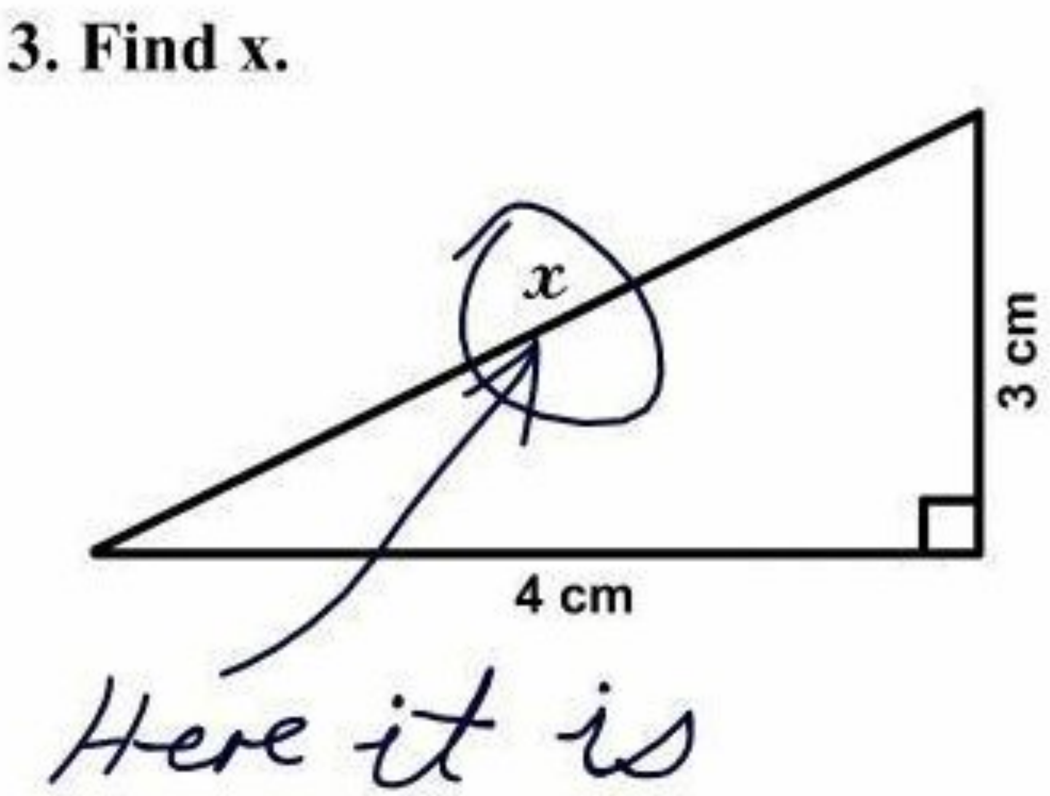
- In MATLAB, variables are **not** global
  - Ex.: If you have variable “a” in your main script, using “a” in a function will not call up the same “a” as in the original script
  - E.g. problematic if your function evaluates a derivative of a function (something you’ll do a lot in this course) and you need to pass a parameter to the function

- There are work-arounds for this issue, but we will cover them later when needed

Scripts

- Calling up other scripts within MATLAB is pretty easy (compared to doing other stuff)
- You save a MATLAB script as a .m file
- Call up a script in the command prompt or within another script by just writing the name of the script you want to call within the script you're writing
  - Can be useful for loading datasets or subroutines
- **<u>Note:</u> calling external scripts/datasets won't be needed much (if at all) in CME102**

Questions?



Wrap-up

- This presentation owes credit to Dr. Eric Darve and Dr. Hung Lê for their development of CME102, and Dr. Vadim Khayms's CME100 MATLAB workbook
- Contact: timmya@stanford.edu
- **If you need to learn how to do something new or troubleshoot your code, www.mathworks.com is a tremendous resource**
- MATLAB is a great skill, learn it early and it will help a ton in your future engineering classes.