MATLAB TUTORIAL WORKSHEET

What is MATLAB?

- Software package used for computation
- High-level programming language with easy to use interactive environment
- Access MATLAB at Tufts here: `https://it.tufts.edu/sw-matlabstudent`

Outline

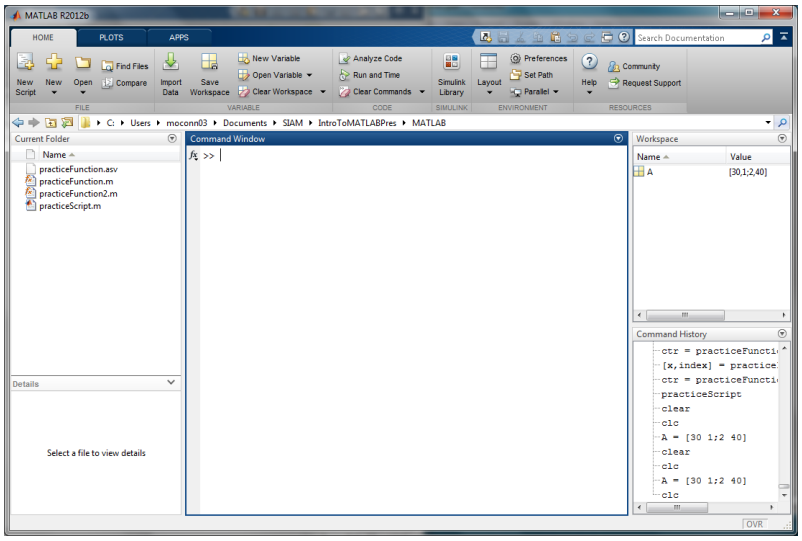
- Windows in MATLAB - discuss the layout
- Getting Started - get acquainted with the command window by using MATLAB as a calculator
- Creating Vectors and Matrices - learn how to create vectors and matrices easily
- For Loops - discuss what a for loop is and the notation in MATLAB
- If Statements - discuss what an if statement is and the notation in MATLAB
- How to use a function with multiple inputs and outputs - discuss functions
- We will do some exercises at the end to practice the material we have covered today!

Notes

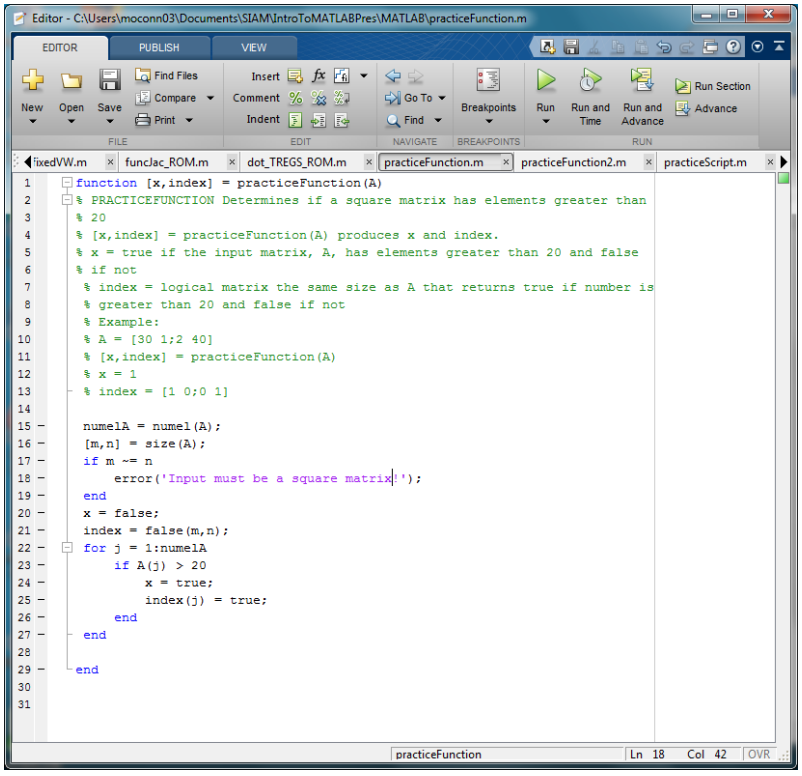
- This is meant to be interactive, so please ask questions!
- There are many things we will not have time to cover today, but we hope we are giving you a foundation of the basics to build upon.
- There are additional resources posted on the course website and I will be holding weekly office hours to assist with the programming component of the course!
    - My Office Hours: Tuesdays from 4:30-5:30 in Bromfield-Pearson Office 216

Windows in MATLAB

- **Command Window** – where you enter data, run MATLAB code, and display results
- **Command History** – displays a log of the statements run in the Command Window
- **Workspace** – displays current variables for use in the Command Window



- **Editor** – where you can write your own functions and scripts



Getting Started
In Command Window type,

```
>> 2+2 <enter>
>> a = 2 <enter>
>> a = 3; <enter>
>> a <enter>
>> b = 4; <enter>
>> a+b <enter>
>> a-b <enter>
>> a*b <enter>
>> sin(0) <enter>
```

We have learned that the semicolon at the end of a statement suppresses the output. Now, let's say you don't know what the sin function does,

```
>> help sin % Prints info about the function in the Command Window
>> doc sin % Opens a new window with info about the function
```

What does the % mean? It is the command in MATLAB that means comment. Let's check in with our workspace and see which variables we have assigned.

```
>> help who
>> who
```

We have a, ans, and b in our workspace. If we are done with these variables, we can get rid of them by typing,

```
>> clear a % This clears just a from your workspace
>> clear % This clears all the remaining variables in our workspace
>> clc % This clears the Command Window screen, not the variables
```

Again, you can always type help (function name) if you want more details about these functions.

Creating Vectors and Matrices

- To enter vectors use square brackets
- To create a row vector separate elements with commas or spaces
- To create a column vector separate elements with semicolons
- The transpose of a vector (or matrix) is expressed using an apostrophe

```
>> x = [1,2,3]
>> x = [1 2 3]
>> y = [4;5;6]
>> x = x'
```

- You can create a row vector by indicating a starting value, an increment, and an ending value, all separated by a colon

```
>> z = 1:1:10
>> w = 1:0.5:5
>> u = [1:1:10]' % If you want a column vector instead
```

- You create matrices in a similar way

```
>> A = [1 2 3;4 5 6;7 8 9]
>> B = [x y]
>> C = [x';y']
```

- Let's explore extracting different elements of a matrix

```
>> A(1,1) % Extracts the element in the (1,1) position
>> A(2,:) % Extracts the second row of the matrix
>> A(:,2) % Extracts the second column of the matrix
```

- A note about multiplication: You should be aware that $*$ does multiplication and matrix multiplication, while $.*$ does element-wise multiplication.

5 min Break

- Use this time to catch up and play around with what we have learned so far
- We will be walking around to assist and answer any questions

for Loops
First, let's clear our workspace:

```
>> clear
>> clc
```

Now, let's create a row vector with 5 elements.

```
>> x = 1:0.5:3;
```

Say we want to add 1 to every element in our vector. How should we do this?

```
>> x(1) = x(1) + 1;
>> x(2) = x(2) + 1;
>> x(3) = x(3) + 1;
>> x(4) = x(4) + 1;
>> x(5) = x(5) + 1;
>> x
```

This is very tedious and imagine if you had a much larger vector, doing this by hand would be very annoying. How can we do this more efficiently?
Option 1:

```
>> x = 1:0.5:3;
>> x = 2:0.5:4
```

But what if you weren't just adding one, this might not be feasible.
Option 2:

```
>> x = 1:0.5:3;
>> x = x+2
```

This is probably the easiest option, but let's use a for loop for learning purposes!
Option 3:

```
>> x = 1:0.5:3;
>>  for i = 1:5
        x(i) = x(i) + 1;
    end
>> x
```

This is called a for loop. Let's read the help documentation!

```
>> help for
```

The documentation tells us that for loops repeat statements a specific number of times.

if Statements

Once again let's clear our workspace.

```
>> clear
>> clc
```

Let's make another vector with 5 elements.

```
>> x = [1 0 1 0 1];
```

How would we update the vector, so that the elements that are 1 become 3 and the elements that are 0 become -1? We could do this by hand or just build a new matrix, but again if we were doing something more complicated this would not be feasible. Solution: Use a for loop and an if statement!

```
>> for i = 1:5
        if x(i) == 1
            x(i) = x(i) + 2;
        elseif x(i) == 0
            x(i) = x(i) - 1;
        end
    end
>> x
```

We used the for loop just like we did above, but we added an if statement. We will now look at the documentation for if statements.

```
>> help if
```

5 min Break

- Use this time to catch up and play around with what we have learned so far
- We will be walking around to assist and answer any questions

How to use a function with inputs and outputs

MATLAB has many built-in functions that will help you do what you want. For example, MATLAB has functions eye, ones, zeros that enable you to create an identity matrix, ones array and zeros array. We will look at eye in more detail.

```
>> clear
>> clc
>> help eye
```

We can see that there are a lot of different ways to use eye, but let's just look at the first two options.

```
>> eye(3) % This will give us a 3-by-3 identity matrix and stores it in ans
>> I = eye(3) % This will do the same thing, but saves it in I instead of ans
>> I = eye(3,3) % This does the same exact thing as above
>> I = eye(3,4) % This gives us a 3-by-4 identity matrix
```

The 3 and 4 are called inputs and I is an output. If a function has more than one output the notation is

```
[output1,output2] = function_name(input1,input2)
```

Here are a few notes about writing your own function:

- Use the editor to write your function
- Save your function as a .m file (m-file)
- See the example of a function above to see the notation
- In order to use your function in the Command Window, your current directory must contain the function m-file (you can type pwd in the Command Window to see what your current directory is)
- If your function isn't working exactly how you want and you can't seem to figure out why, use the debugging feature in MATLAB! This allows you to step through the code one line at a time to see what is really happening.
- If you want to stop any function from running to completion, hit Ctrl+C

Exercise 1

Plot $sin(x)$ on the interval $[-pi, pi]$ using 0.01 spacing.

Exercise 2

Write a function that accepts a vector of any size as input, adds 2 to every element greater than 5 and adds 1 to every element less than or equal to 5 and outputs the updated vector.

Some helpful hints:
– Remember to use the Editor to write your function
– Check out the MATLAB functions length or size

Exercise 1 Solution

```
x = -pi:0.01:pi;
plot(x,sin(x))
```

Exercise 2 Solution

In the Editor write the function and save it as a .m file. Use the function in the Command Window.

```
function x = addTwoAddOne(x)
% addTwoAddOne adds two to each element of a vector that is greater than 5 and
% adds one to every element that is less than or equal to 5.

n = length(x);
for i = 1:n
    if x(i) > 5
        x(i) = x(i) + 2;
    else
        x(i) = x(i) + 1;
    end
end

end
```