06-Feb-2021

# MATLAB Review for MEGR 3122

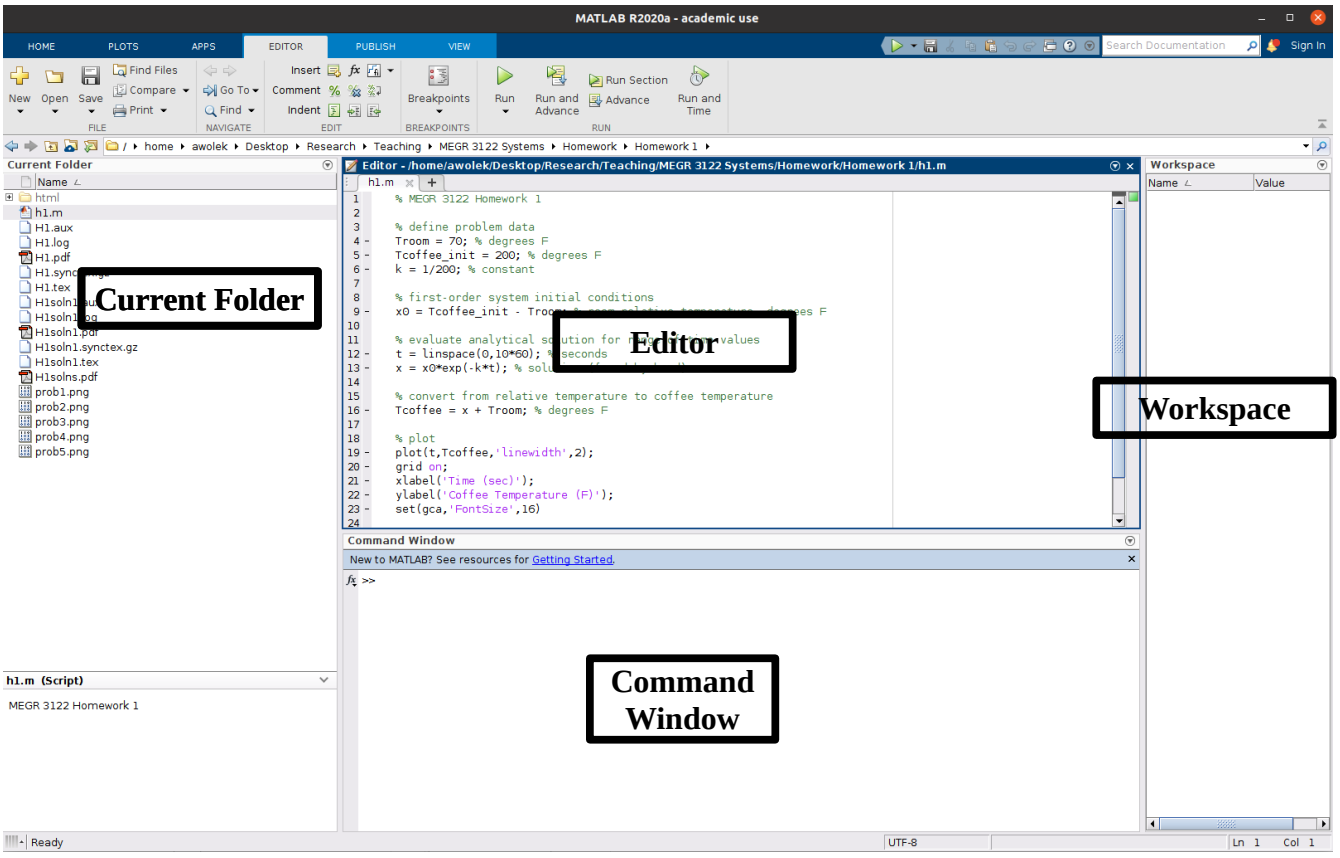
It is highly recommended that students first complete the "MATLAB OnRamp Tutorial" and/or "MATLAB Fundamentals course"

MATLAB On-Ramp
https://www.mathworks.com/learn/tutorials/matlab-onramp.html

MATLB Fundamentals
https://matlabacademy.mathworks.com/?s_tid=gn_trg_cosp

## MATLAB Environment

- **Current Folder** — Access your files.
- **Command Window** — Enter commands at the command line, indicated by the prompt (>>).
- **Workspace** — Explore data that you create or import from files.
  - As you create variables they are listed in the workspace
  - View their size, type, etc.

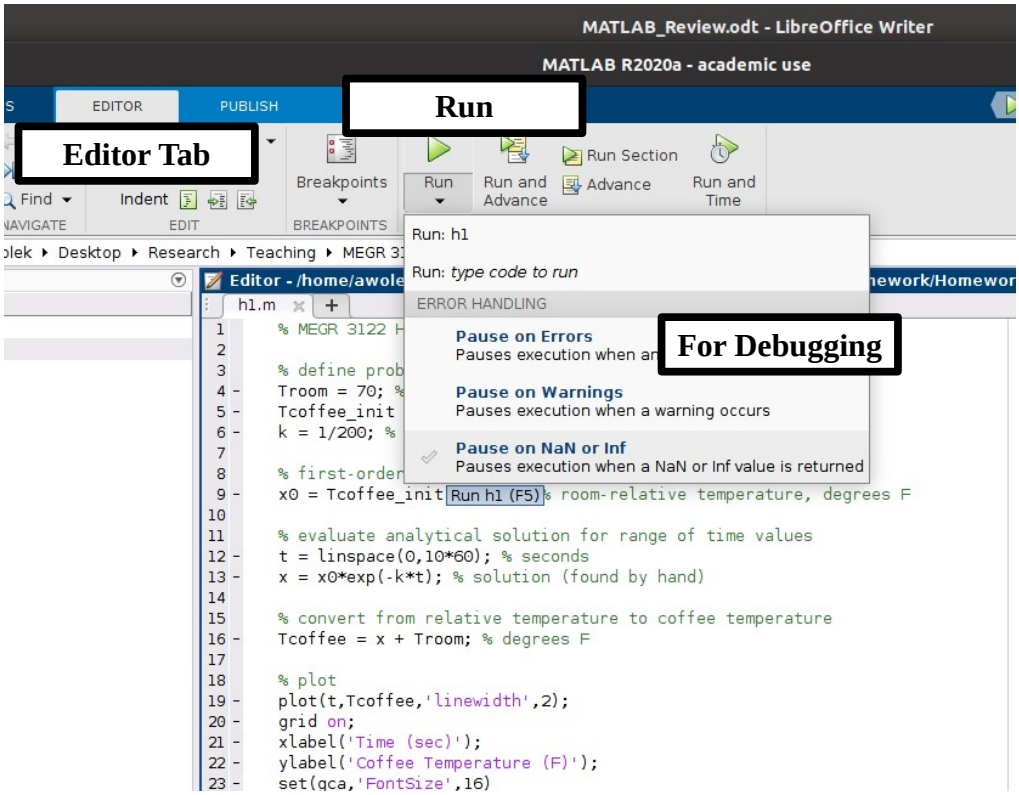


**Three ways to write code in MATLAB:**

1. Directly into the **command window** (such code cannot be used multiple times)
   - Useful for doing a quick calculation, debugging code (checking intermediate variables)
2. As a **script** (recommended for most assignments)
   - The script can have any name (e.g., main.m)
     - Filename must be all one word (no spaces!)
   - Usually the highest level code that defines the inputs, calls sub-functions, and prints outputs
   - Can call other functions *that are in the same folder* or *in MATLAB's path*
3. As a **function**
   - Clearly defined inputs and outputs
   - Begins with a line similar to this:
     `function [output1, output2] = functionName (input1, input2, input3)`
   - Must be saved as a file with the same name as the function (i.e., functionName.m)
     - Again, it must be all one word (no spaces!)
   - Useful to break up complex tasks, make code concise, when a repetitive operation is needed
   - Must assign the outputs and the last line must be `end`

**How to Run Code**

Click the Editor tab and press the green "Run" arrow. If you have errors select "Pause on Errors" option to debug. More information on debugging is here: https://www.mathworks.com/help/matlab/matlab_prog/debugging-process-and-features.html



**Homework Directory**

- Create a new folder for each homework. Place "main" files, functions, and any .m files provided by the instructor in this folder.
- Navigate to this folder from within MATLAB to run your code and complete the assignment

**Accessing the documentation**

- Print a brief help document to the screen: `help funcName`
- Go to a detailed help page: `doc funcName`
- Use the internet, many examples and discussions

**Declaring variables**

- A scalar variable: `>> a = 3;`
- A row vector: `>> a = [1 2 3 4];`
- A column vector: `>> a = [1; 2; 3; 4];`
- A size 2 x 2 matrix: `>> A = [1 2; 3 4];`
- A size 3 x 3 matrix: `>> A = [1 2 3; 3 4 5;6 7 8];`
- A string: `>> myString = 'Hello World!';`
- Note the variable `pi` is a special variable that is pre-defined in MATLAB
- Area: `>> Area = pi*R^2;`

**Common operations**

- Trigonometric functions: e.g., `cos(a), sin(a), tan(a)`
- Powers: `a = 2; b = a^10; c = sqrt(b);`
- Matrix multiplication:
  - `>> A = [1 2 3; 3 4 5;6 7 8];`
  - `>> B = [9 10 11; 12 13 14;15 16 17];`
  - `>> C = A*B`
- Vector-by-matrix multiplication:
  - `>> A = [1 2 3; 3 4 5;6 7 8];`
  - `>> x = [1; 2; 3];`
  - `>> y = A*x`
- Size of a matrix gives (number of rows and cols)
  - `>> size(A)`

**Declaring a vector containing a large range of values**

- To declare a range of `N` values, evenly spaced from `xmin` to `xmax` use `linspace`
  - `xrange = linspace(xmin, xmax, N)`
- To specify the spacing, e.g., from `xmin` to `xma` spacing `dx,` use:
  - `xrange = [xmin:dx:xmax];`

**Element-wise operations**

- To multiply or take the power of each element in a vector include a "." in front of the operator. This is not necessary for trigonometric and some other functions.
  - `xrangeElemsSquared = xrange.^2;`
  - `xrangeElemsSquared = xrange.*xrange;`

**Manipulating matrices**

- Supposed we declare a 3 x 3 matrix as follows:
  - `>> A = [1 2 3; 3 4 5;6 7 8]`
- Access the element in the second row, second column using:
  - `>> A(2,2)`
- Access the first row using
  - `>> A(1,:)`
- Access the first column using
  - `>> A(:,1)`
- Add a fourth row
  - `>> A(4,:) = [9 10 11]`

**Displaying output with fprintf**

- To display an output use fprintf with syntax:
  - `fprintf(formatSpec, A1, A2, … AN);`
  - `formatSpec` is a string containing some indicators (e..g, `%3.3f`) that are replaced with the values stored in the variables `A1, A2, … , AN` when the program is executed. See `doc fprintf` for detailed information.
  - The character`\n` is added to the end of the formatSpec to make the cursor go to a new line
  - `%3.3f` can be used to display decimal numbers
  - `%i` can be used to display integers
- For example, supposed we have x = 2, and y = 3; and we want to print this to the screen in a formatted text using fprintf:
  - `>> x = 2;`
  - `>> y = 3;`
  - `>> fprintf('The variable x = %3.3f, the variables y = %3.3f \ n', x, y);`

**Single for loop**

```
for k3 = [3:3:30];
     fprintf('Iter (k3)=%i \n',k3);
end
```

**Nested for loops**

```
for k1 = [1:1:10]
     for k2 = [2:2:20];
          for k3 = [3:3:30];
               fprintf('Iter (k1,k2,k3)=%i,%i,%i \n',k1,k2,k3);
          end
     end
end
```

**While loops**

```
k = 1;
a = 10;
while (k < a)
     fprintf('k is currently %i and still less than %i \n',k,a)
```

```
     k = k + 1;
end
```

**If/Else Statements**

```
a = rand; % generate a random number from 0 - 1
if (a > 0.75)
     fprintf('the random value %3.3f is > 0.75 \n', a);
elseif (a > 0.5)
     fprintf('the random value %3.3f is > 0.5 \n', a);
else
     fprintf('the random value %3.3f is <= 0.5 \n', a);
end
```

**Clearing Workspace**

- Delete all variables: `>> clear all;`
- Close all figures: `>> close all;`
- Clear the command windows: `>> clc;`

**Stopping runaway code**

- Sometimes a coding error causes a code to run for a very long period of time or indefinitely.
- This process can be terminated by pressing CTRL + C

**Comments**

- Comments are entered by starting with the `%` symbol
- In MATLAB, a semicolon ; is added at the end of each line to suppress the output