MCG 4322A

Parametrization Tutorial

The following files must be downloaded and opened before the tutorial, **ON LAB COMPUTERS ONLY**:

- Go to Brightspace, download file ‘groupABC.zip’ onto your personal uOttawa computer account drive
- Extract the .zip file to your Z:/ drive
- Navigate to the MATLAB subfolder, and open the **MAIN.m** file in MATLAB
- Open a new part window in Solidworks

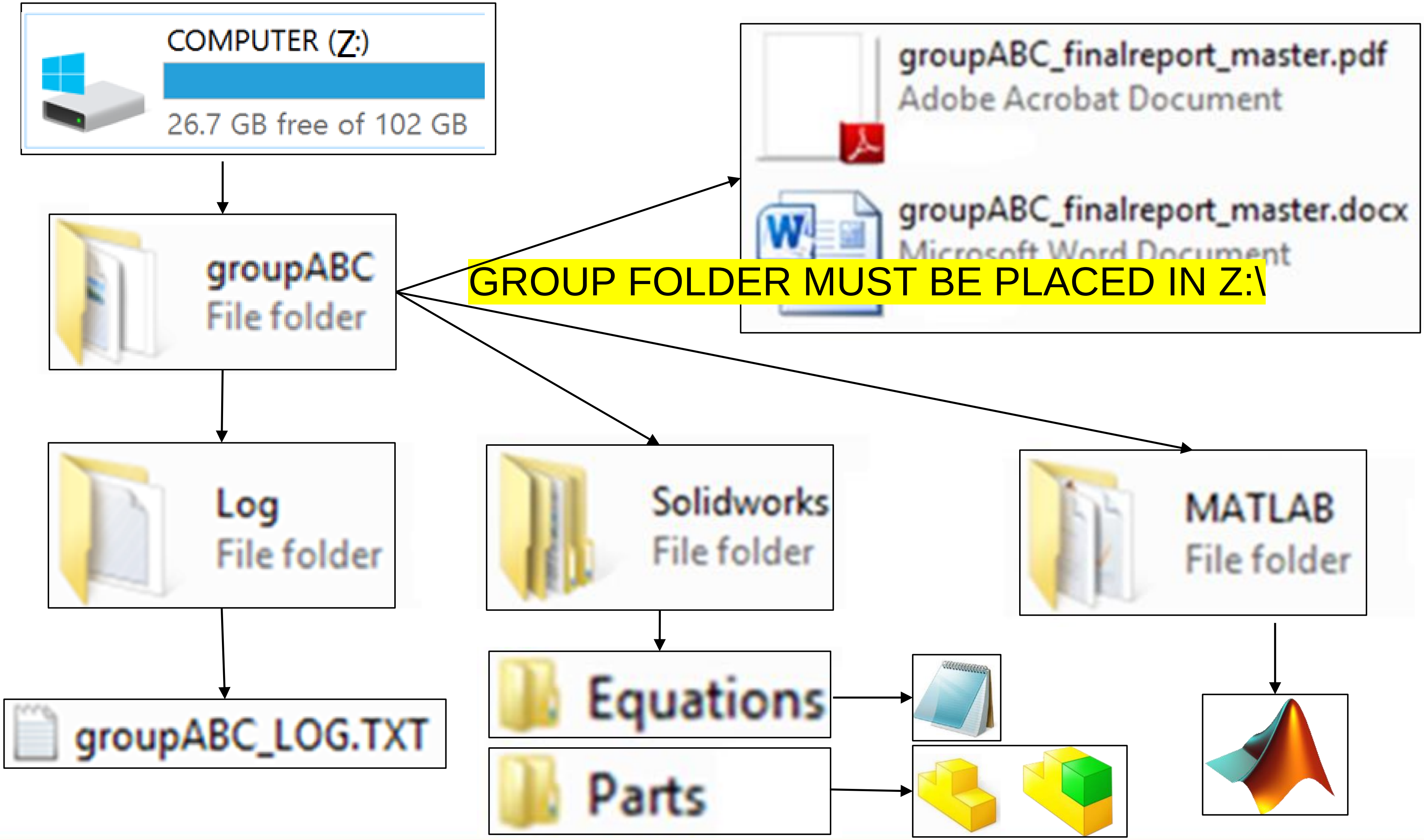
Importance of parametrization

Parametrization will allow you to:

- Verify the entirety of the analysis for your basic criteria
- Check that your design can be adapted to new situations
- Improve the robustness of your CAD model

Critically important for <u>efficient</u> design work

Required File Organization

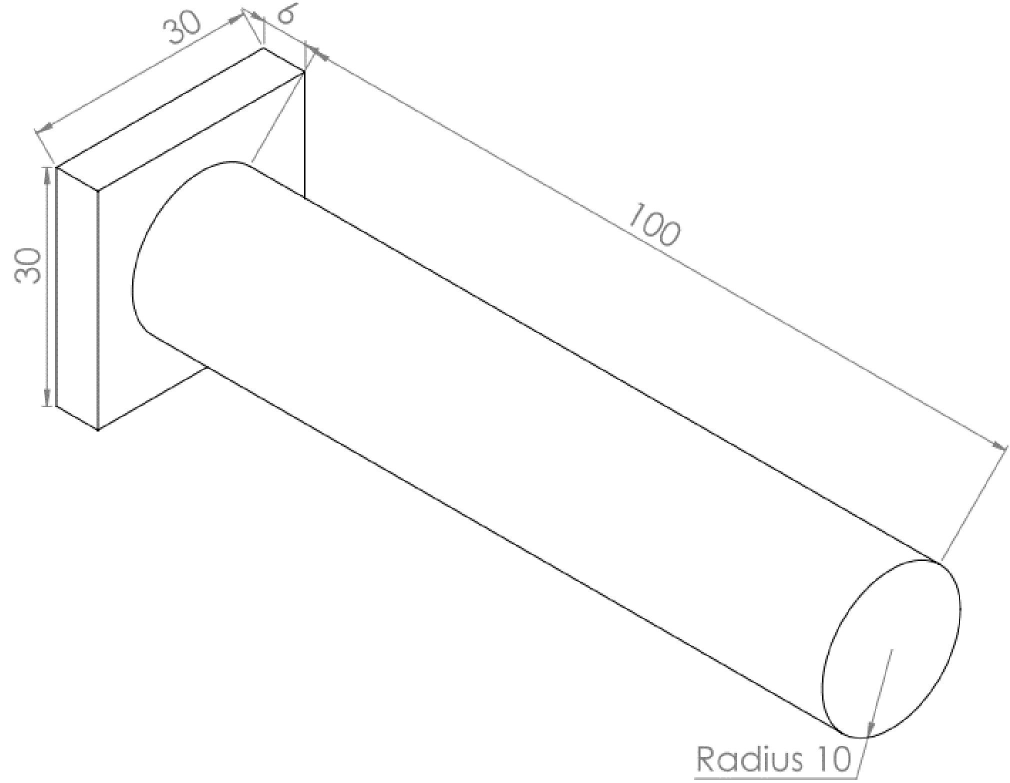


Step 1: Draft a cantilever beam in Solidworks

In Solidworks, model the beam with the following specifications:

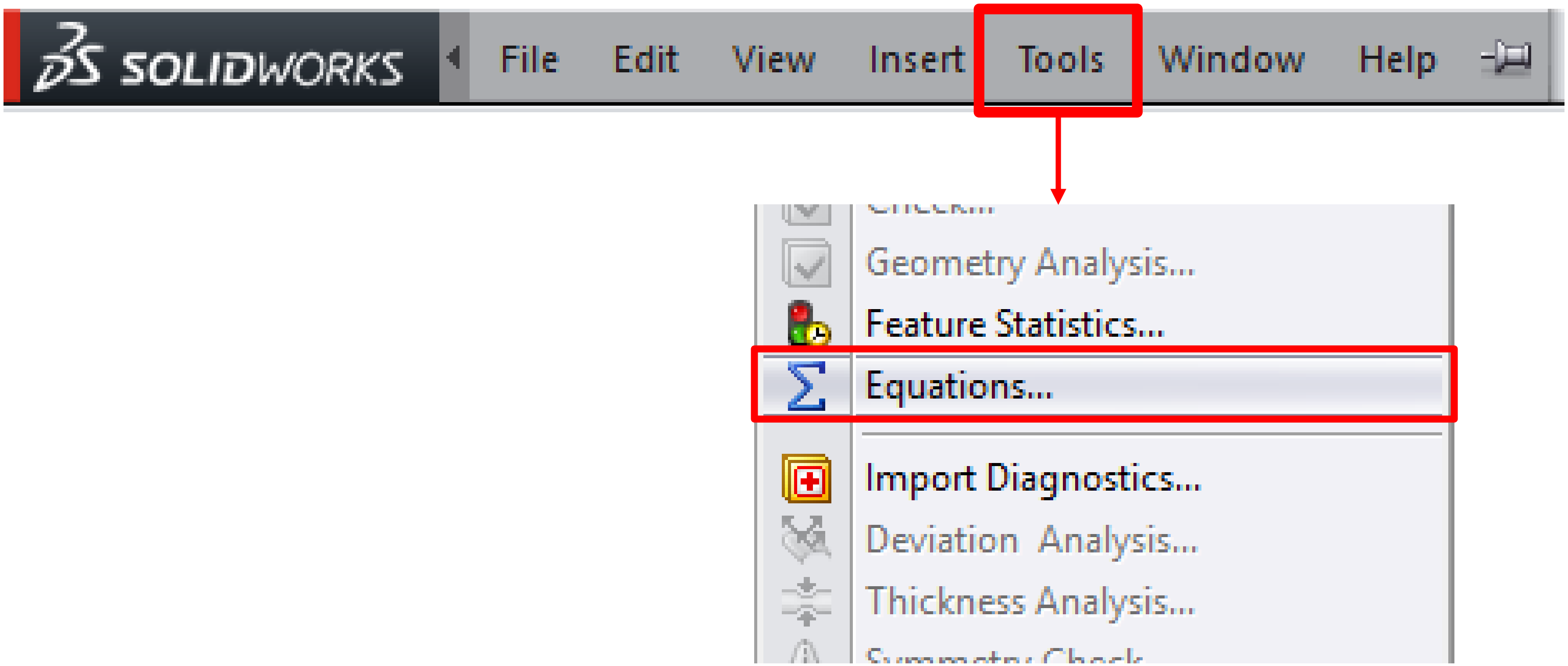
- Round cross section with 20 mm diameter and 100 mm length
- Square base 30 by 30 mm by 6 mm thick

Save as: *shaft.sldprt*



Step 2: Declaring global variables

In top toolbar, navigate to Tools → Equations

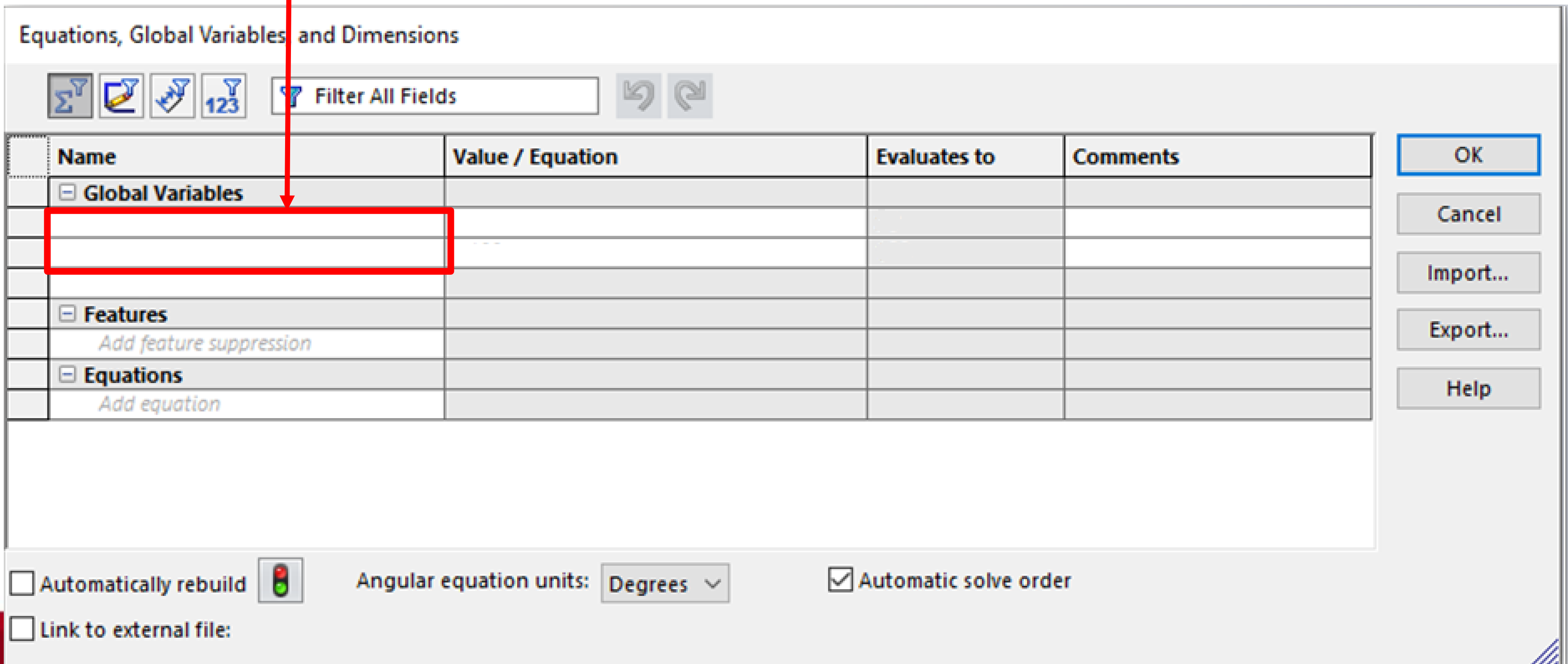


Step 2: Declaring global variables

Add two global variables, and set their values:

-“Diameter”,

- “Length”,



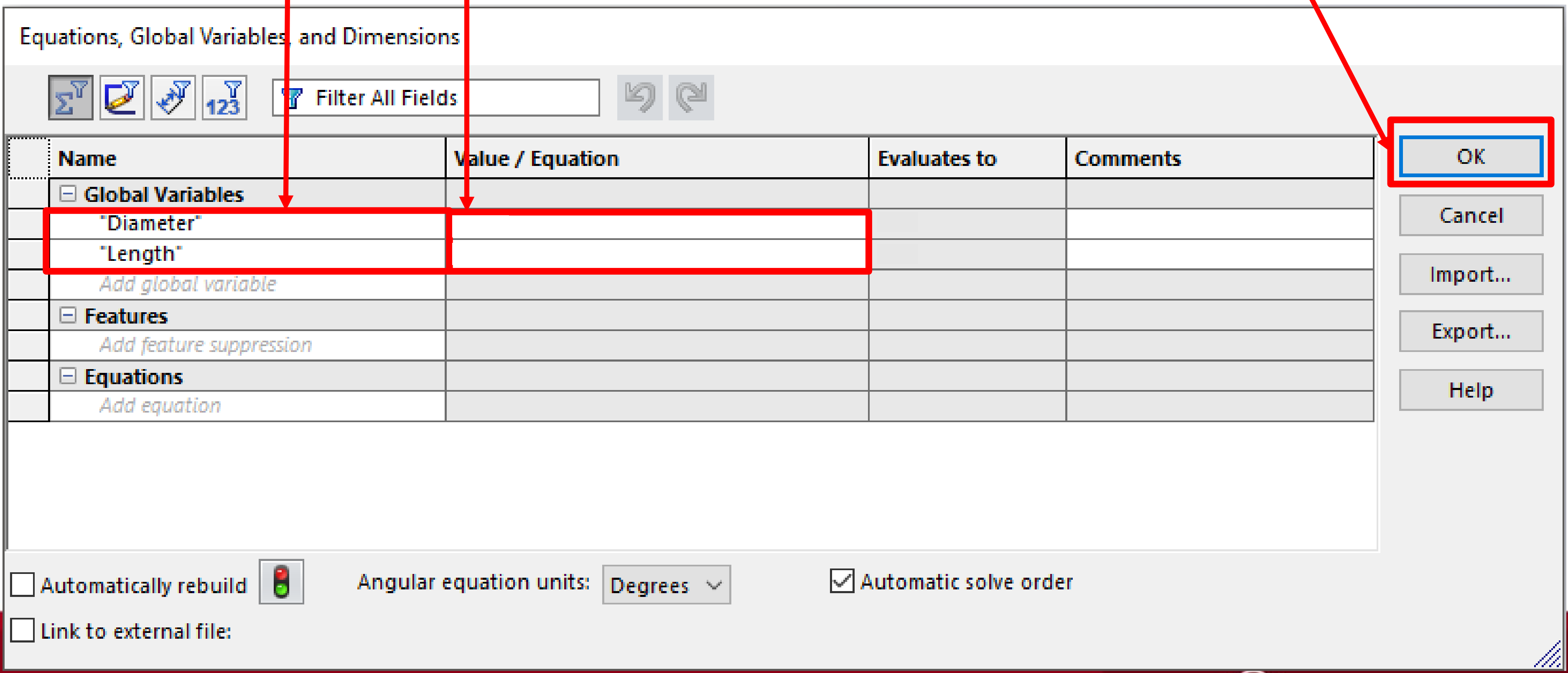
Step 2: Declaring global variables

Add two global variables, and set their values:

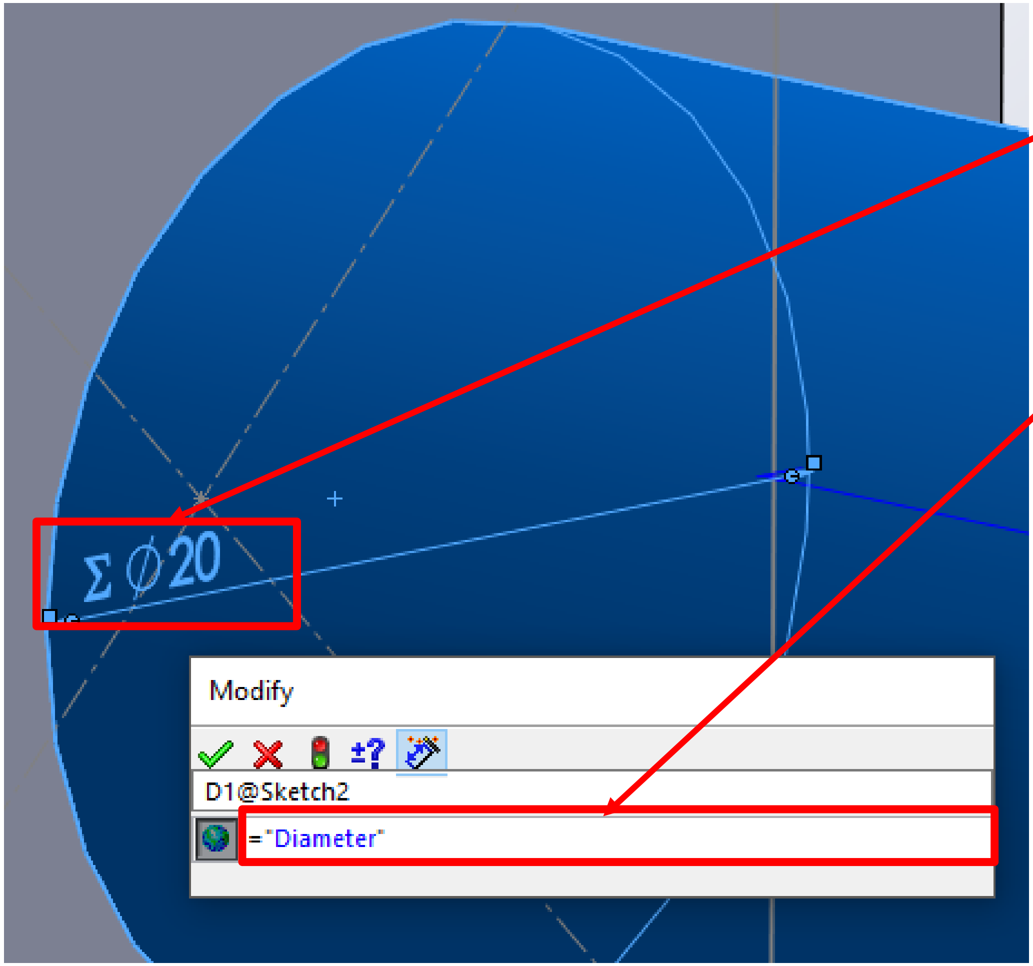
-“Diameter”, Set value to = 20 mm

- “Length”, Set value to = 100 mm

Select ‘OK’



Step 3: Link dimension to global variable



- Double click the diameter dimension of the beam
- Click on Input value box,

  enter ‘ = “Diameter” ‘
- For the beam length dimension,

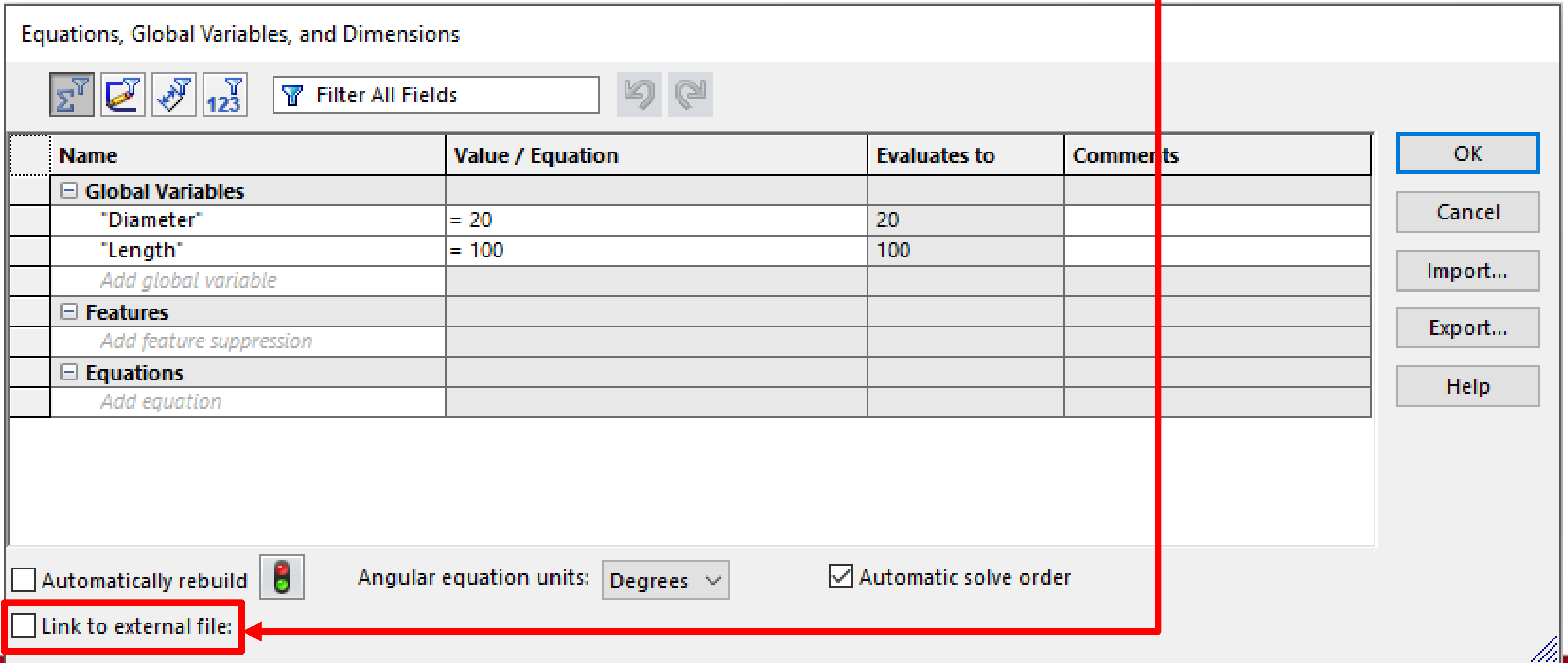
  Input: ‘ = “Length” ‘

Variables are now linked to dimensions!

(Indicated by the epsilon Σ, beside the dimension)

Step 4: Linking variables to text file

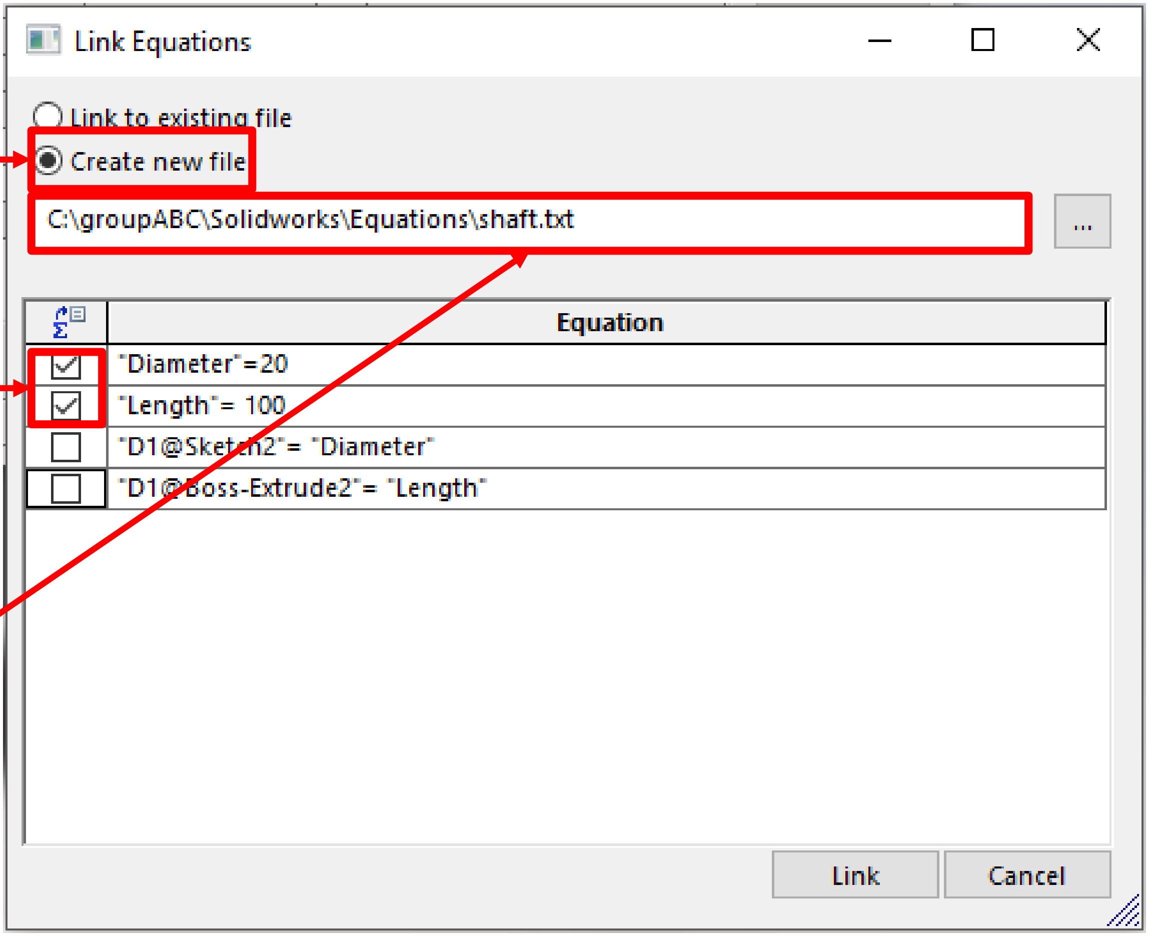
- Navigate back to Equations menu (Tools → Equations)
- Bottom left, check the ‘Link to external file:’ option
  - Pop up menu titled ‘Link Equations’ will open



Step 4: Linking variables to text file

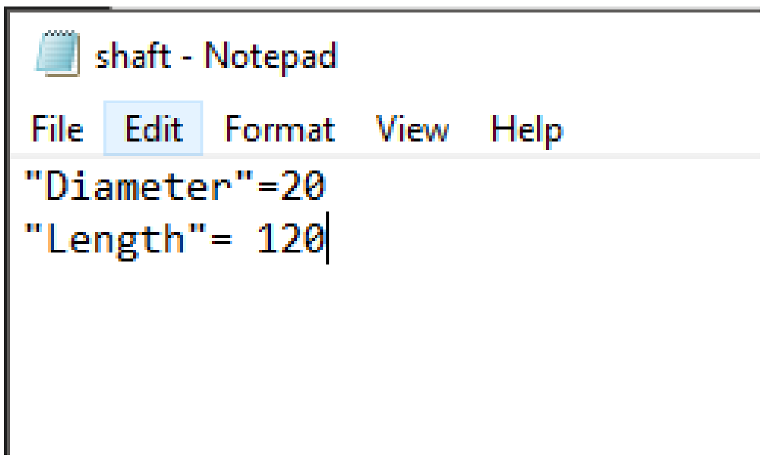
- Within "Link Equations" pop up menu, select radial for 'Create new file'
- Select only checkboxes beside variable entries (e.g. "Diameter"=…)
- To create text file for equations,
  input the filepath:

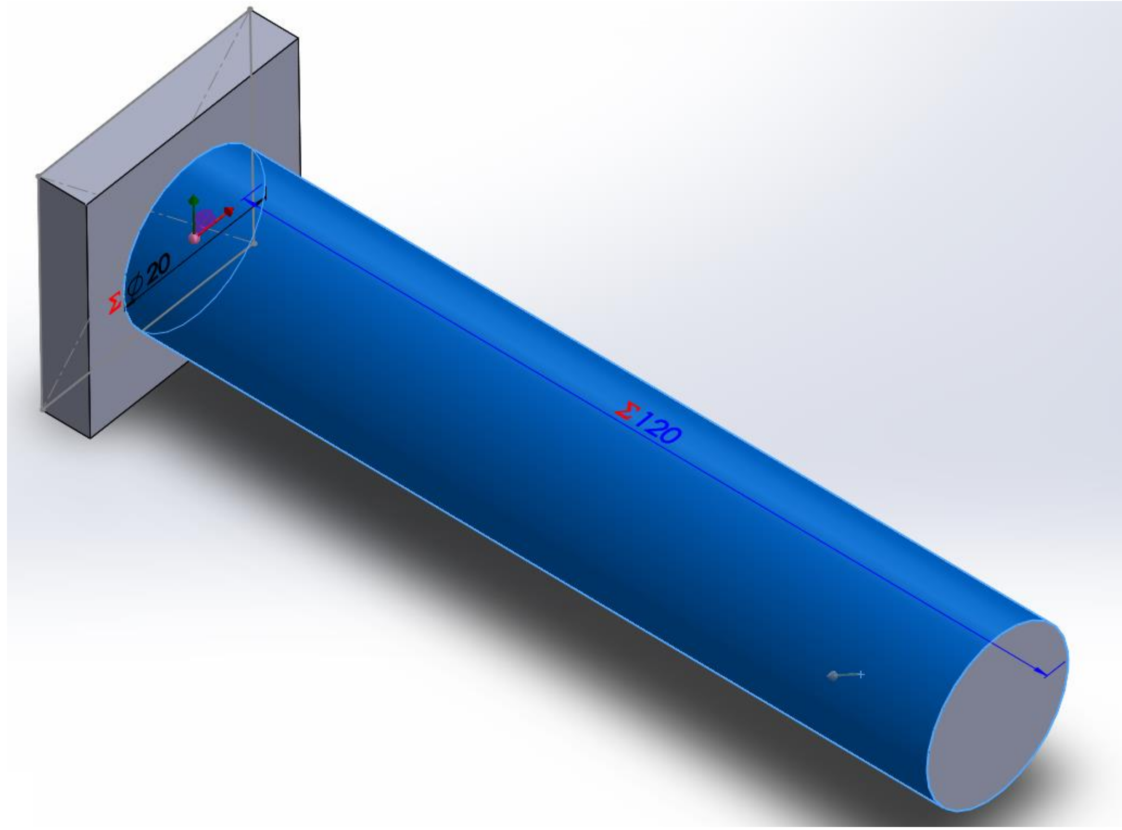
Z:\groupABC\Solidworks\
Equations\shaft.txt



Step 5: Change text file, rebuild

- Open the ‘shaft.txt’ equations file
- Modify such that “Length= 120”, save text file





- Open ‘shaft.sldprt’ file in Solidworks, and rebuild!

Step 6: Coding with MATLAB

The default MATLAB folder consist of three parts

1. MAIN.m file
   - MATLAB Graphic User Interface (GUI) code template, uses built-in ‘GUIDE’ interface
   - Majority of this code is auto generated (do not modify it)

2. MAIN.fig file
   - The figure file that contains your GUI

3. Design_code.m file
   - Contains your parametric modelling and analysis equations
   - Groups primarily modify this code

Step 6: Coding with MATLAB

IMPORTANT REQUIREMENT: Include Code Comments with MATLAB code

Code comments are used to document what the code should do

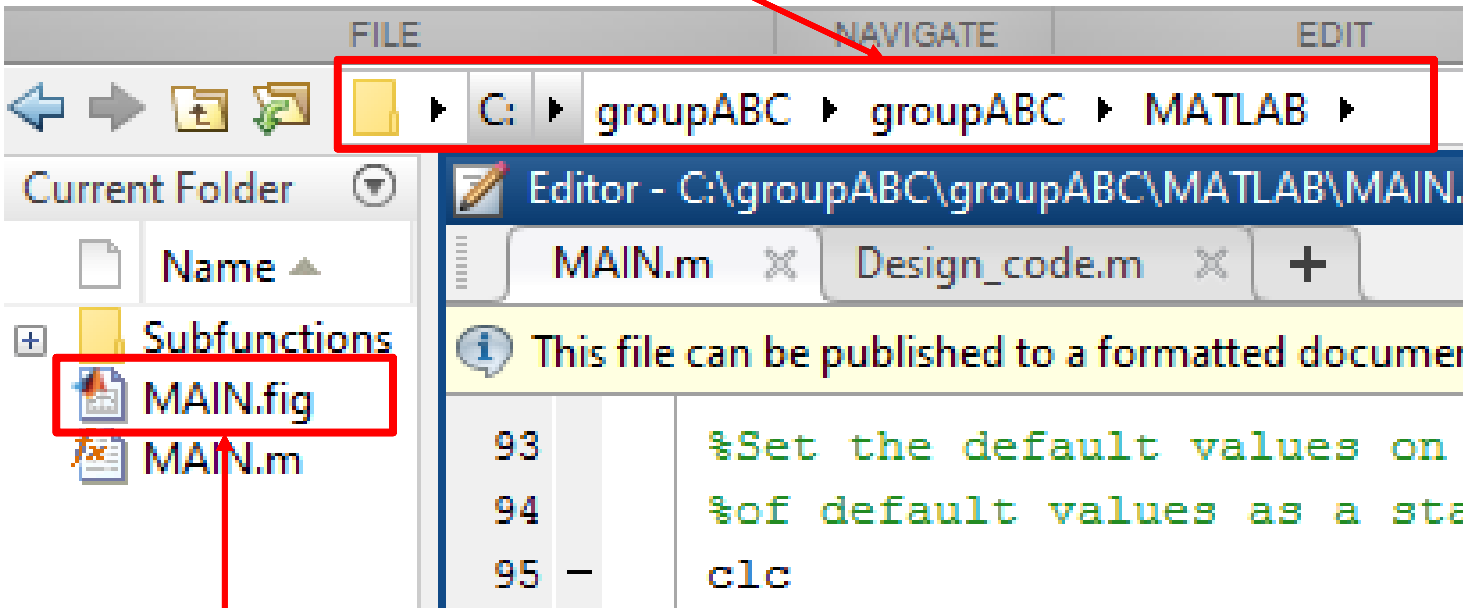
- Helps future developers read the code quickly
- Informs reviewers (e.g. the course instructor, T.A.s) on the purpose of the code
- Assist with the debugging process, by clearly describing the intended purpose of code

For MCG4322, code comments are required

- Final grade will be dependent in part on readability of code, therefore include comments that are clear and concise

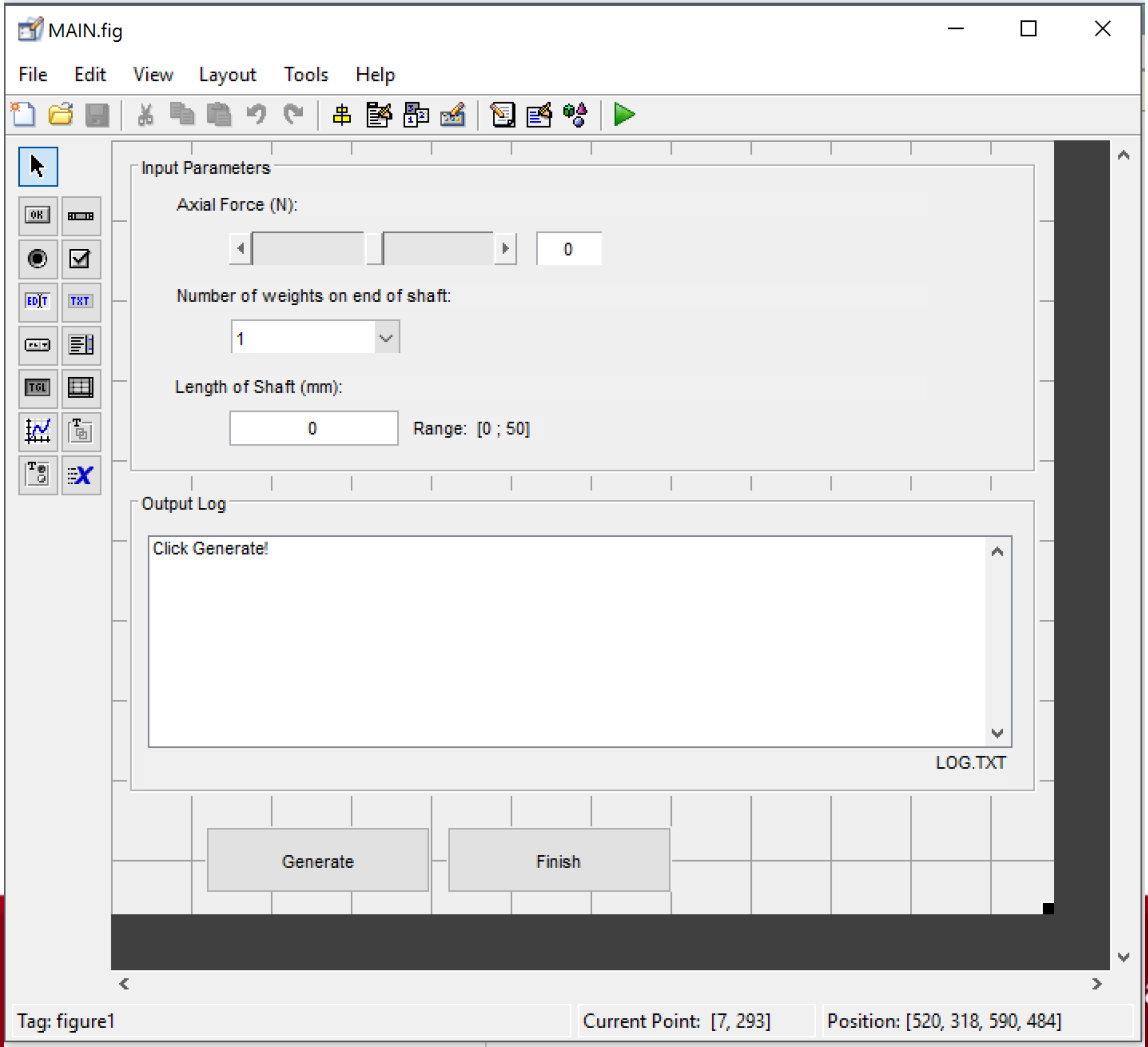
Step 7: MATLAB file navigation

1. Open the MAIN.m file in MATLAB.
2. Within directory selection box, navigate to the directory Z:\groupABC\MATLAB

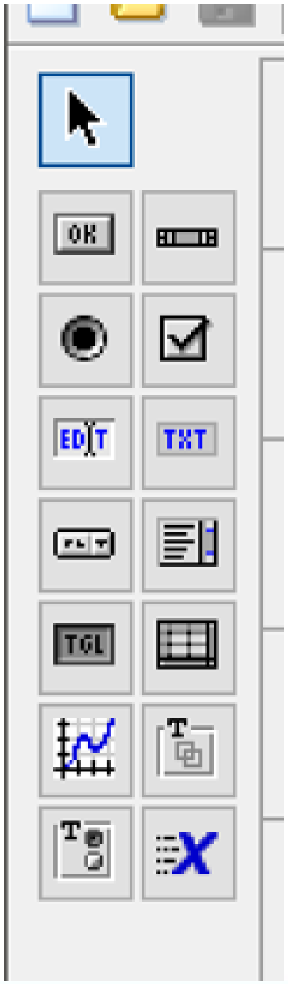


3. To open GUIDE, right click the **MAIN.fig file**, and select 'OPEN IN GUIDE' option

Step 8: GUIDE Interface

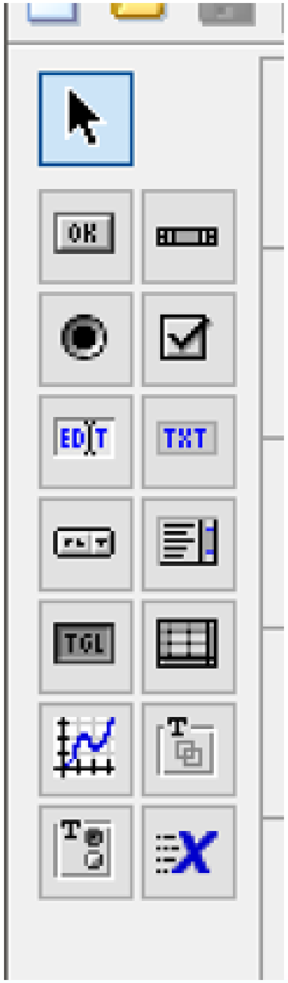


Step 8: GUIDE Interface



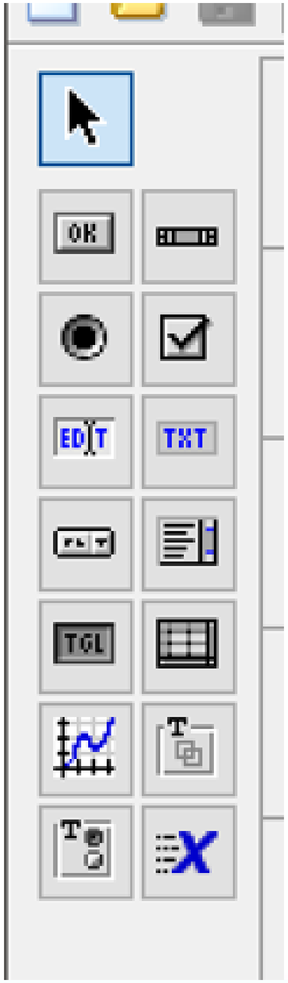
- Toolbar to the left contains choice of GUI elements

Step 8: GUIDE Interface



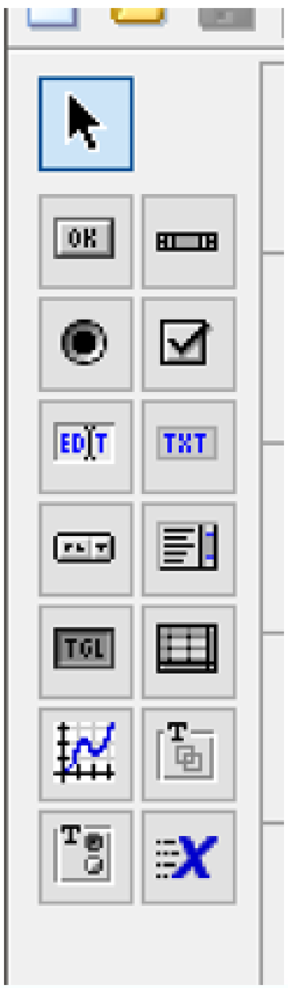
- Toolbar to the left contains choice of GUI elements
- To add an element to code in MAIN.m, drag element into figure, then save the **MAIN.fig** file within GUIDE interface

Step 8: GUIDE Interface

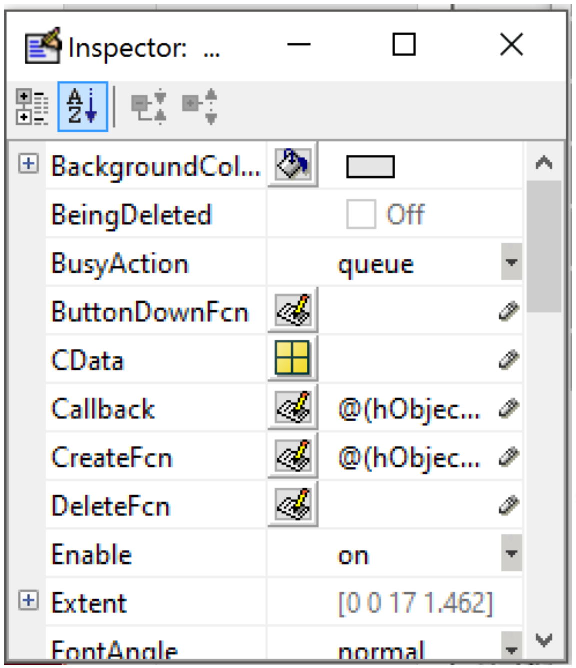


- Toolbar to the left contains choice of GUI elements
- To add an element to code in MAIN.m, drag element into figure, then save the **MAIN.fig** file within GUIDE interface
  - Adding an element will automatically add the necessary code to **MAIN.m**!

Step 8: GUIDE Interface



- Toolbar to the left contains choice of GUI elements
- To add an element to code in MAIN.m, drag element into figure, then save the **MAIN.fig** file within GUIDE interface
  - Adding an element will automatically add the necessary code to **MAIN.m**!
- Double click on an element to access property inspector, (This is where element properties are changed)
- Save figure to save any changes to **MAIN.m**



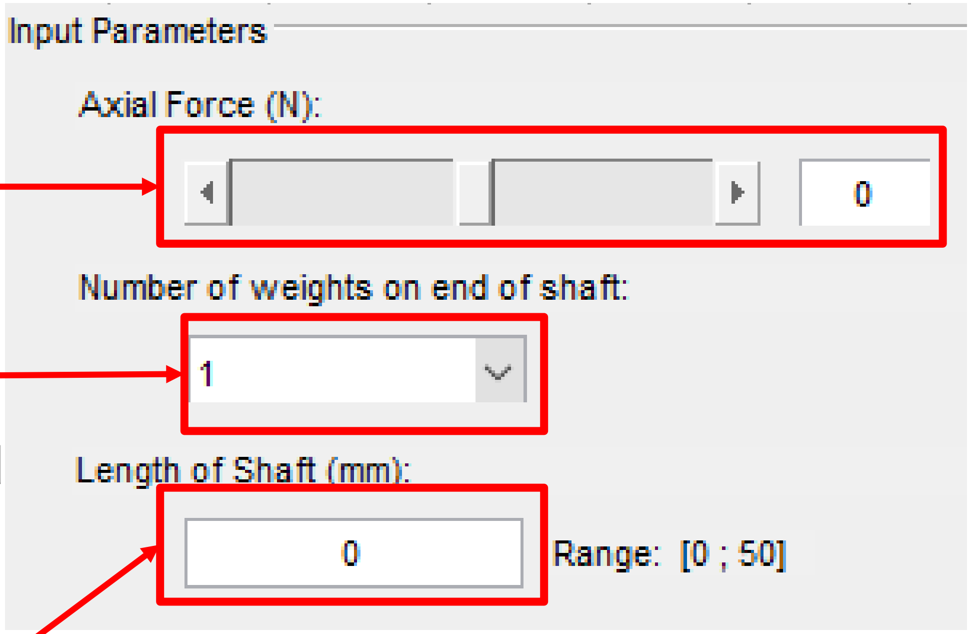
Step 9: Translating GUI inputs to code

Template GUI has 3 input elements corresponding to three parameters

1. Slider
 Takes the axial force acting on cantilever beam

2. Drop down list
Selection from a range of possible number of weights hanging on the end of beam

3. Text field
Reads the value input into the text box for shaft length



Step 9: Translating GUI inputs to code

How to take input data from a slider UI element (For axial force)

1. Open **MAIN.m** code

Step 9: Translating GUI inputs to code

How to take input data from a slider UI element (For axial force)

1. Open **MAIN.m** code

2. Navigate to line 120 of the code

Step 9: Translating GUI inputs to code

How to take input data from a slider UI element (For axial force)

1. Open **MAIN.m** code

2. Navigate to line 120 of the code

3. Add the following code comment to describe what the code does

```
%returns axial force as integer or decimal value
```

Step 9: Translating GUI inputs to code

How to take input data from a slider UI element (For axial force)

1. Open **MAIN.m** code

2. Navigate to line 120 of the code

3. Add the following code comment to describe what the code does

   ```
   %returns axial force as integer or decimal value
   ```

4. On line 121, add the code to read the current slider state selected

   ```
   axial_force = get(handles.Slideraxial_force,'Value’);
   ```

Step 9: Translating GUI inputs to code

How to take input data from drop down list (For number of weights to be supported from the end of the beam)

1. Within **MAIN.m**, navigate to line 122 of the code

Step 9: Translating GUI inputs to code

How to take input data from drop down list (For number of weights to be supported from the end of the beam)

1. Within **MAIN.m**, navigate to line 122 of the code

2. Add the following code to store the option selected by the user from the drop down list as a string

```
%returns NumWeights contents as cell array
COMBOcontents = cellstr(get(handles.NumWeights,'String'));
```

Step 9: Translating GUI inputs to code

How to take input data from drop down list (For number of weights to be supported from the end of the beam)

1. Within **MAIN.m**, navigate to line 122 of the code

2. Add the following code to store the option selected by the user from the drop down list as a string

```
%returns NumWeights contents as cell array
COMBOcontents = cellstr(get(handles.NumWeights,'String'));
```

3. Add the following code to convert the string into a numerical value

```
%returns selected item from NumWeights
Num_weights=str2double(COMBOcontents{get(handles.NumWeights,'Value')});
```

Step 9: Translating GUI inputs to code

How to take input data from a text field (For the shaft length)

1. Within **MAIN.m**, navigate to line 126

Step 9: Translating GUI inputs to code

How to take input data from a text field (For the shaft length)

1. Within **MAIN.m**, navigate to line 126

2. Add the following code to read from a text field, store as a double

```
%returns shaft length as integer or floating point value
shaft_length = str2double(get(handles.TXT_shaftlength,'String’));
```

Step 9: Translating GUI inputs to code

How to take input data from a text field (For the shaft length)

1. Within **MAIN.m**, navigate to line 126

2. Add the following code to read from a text field, store as a double

```
%returns shaft length as integer or floating point value
shaft_length = str2double(get(handles.TXT_shaftlength,'String’));
```

3. At line 135, add the following code to send input data to the design code

```
% The design calculations are done within the function file Design_code.m
Design_code(axial_force, number_of_weights, shaft_length);
```

Step 9: Translating GUI inputs to code

How to take input data from a text field (For the shaft length)

1. Within **MAIN.m**, navigate to line 126

2. Add the following code to read from a text field, store as a double

```
%returns shaft length as integer or floating point value
shaft_length = str2double(get(handles.TXT_shaftlength,'String’));
```

3. At line 135, add the following code to send input data to the design code

```
% The design calculations are done within the function file Design_code.m
Design_code(axial_force, number_of_weights, shaft_length);
```

4. Save your code! (Save often!)

Step 10: Analysis and design code

1. Open the **design_code.m** file

Step 10: Analysis and design code

1. Open the **design_code.m** file

2. At the first line, modify the function input parameters with this code:

```
%This function is the main "design" function.
function Design_code(axial_force, number_of_weights, shaft_length)
```

Step 10: Analysis and design code

1. Open the **design_code.m** file

2. At the first line, modify the function input parameters with this code:

```
%This function is the main "design" function.
function Design_code(axial_force, number_of_weights, shaft_length)
```

3. At line 16, there is a function call to calculate the new shaft diameter. The function is given arguments defined within or sent to current function, and returns a diameter

4. Navigate to line 47-64 to see the optimization algorithm that sizes the new diameter of the shaft based on the inputs supplied

Step 10: Analysis and design code

An example of an shaft diameter optimization algorithm

```
%An example of subfunction.
%Make note that the inputs within the paranthesis have different names than
%the arguments named in the function call. Names themselves do not matter,
%as only the data they contain in passed.
function new_diameter = calc_shaft_diameter(diameter, str, axial_force, number_of_weights, shaft_length)

    %Eq. (##) <- Add a reference to the equation in your report.
    %In this case, we calculate the Von Mises stress from inputs. Ignore
    %Torque input (as it is zero for this example).

    force = number_of_weights*10; %Units (kN)
    n = 0;   %Initial safety factor

    %Optimization loop, change diameter until safety factor 'n' > 1.5
    while n<1.5
        diameter = diameter + 0.001;
        stress = (((4*axial_force)/(pi*(diameter^2)))+((32*force*shaft_length)/(pi*(diameter^3))));
        n = str/stress;
    end

    new_diameter = diameter;
end
```

Step 11: Log text file for results output

The output log file is where you can display all results

The log text file is your responsibility to format

- Should display user friendly information
- Make log file display notable information
- Display calculated outputs data that don't reflect in SOLIDWORKS model
- Show the initial parameters

Be creative and optimize for at-a-glance information!

Step 12: Modifying text files with MATLAB

You will use MATLAB functions to target and write to the Solidworks equation and Log text files

- **IF** the text files change name or changed folder, you must modify your code to reflect those changes

- For all MATLAB commands that edit the linked equation text files, the lines that are added to the text file **must have the same format** as the original text file when it was first linked
  - Possible errors: Changing character spacing, incorrect variable names, changing line order/spacing, missing variables

- After changing an equation file, check to ensure the Solidworks rebuilds the draft properly for all extreme dimension ranges allowed.

Step 12: Modifying text files with MATLAB

First, store the directory for the text file to be modified with a string

```
your_file =
'Z:\\groupABC\\FolderName\\your_text_file.txt';
```

Open the file , and give MATLAB read and write permission

```
fid = fopen(your_file, 'w+t');
```

To write to the text file, use the *fprintf()* command

Within the fprintf statement, combine multiple strings and variables into a single string with the string concentration command: *strcat*

```
fprintf(fid,strcat('Your text goes here', num2str(x),'\n'));
```

- We need to note the data type of the variable before it is accessed
- '\n' is placed at the end of the string, to indicate the start of a new line

When done writing to file, terminate MATLAB's permission to stop further reading/writing

```
fclose(fid)
```

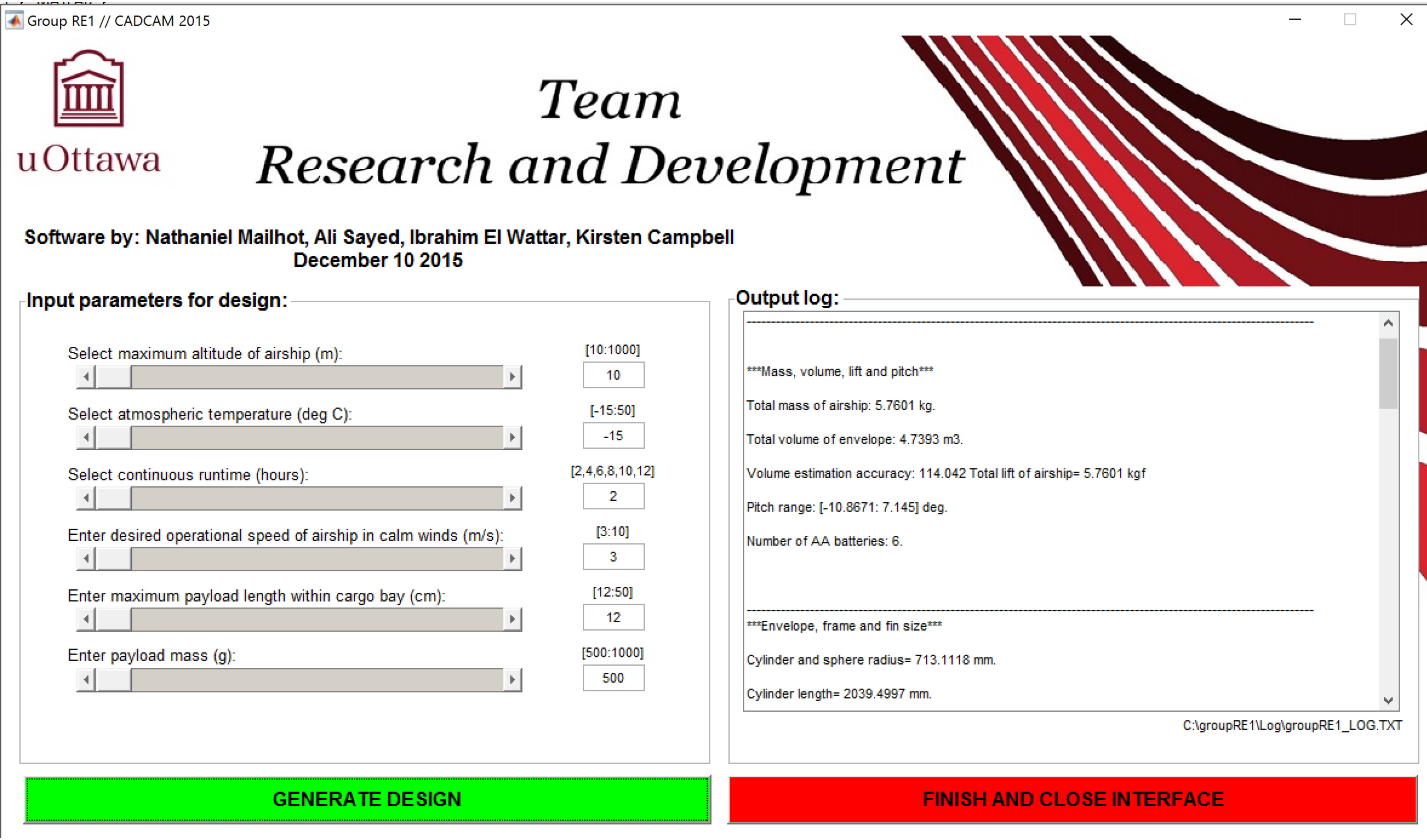
Step 12: Modifying text files

```
18      %Declaring text files to be modified
19      %Files
20 -    log_file = 'C:\\groupABC\\Log\\groupABC_LOG.
21 -    cylinder_file = 'C:\\groupABC\\SolidWorks\\E
22
23      %Write the log file (NOT USED BY SOLIDWORKS,
24      %Please only create one log file for the com
25 -    fid = fopen(log_file,'w+t');
26 -    fprintf(fid,'***Shaft Design***\n');
27 -    fprintf(fid,strcat('Axial force =',32,num2st
28 -    fprintf(fid,strcat('Shaft length =',32,num2s
29 -    fprintf(fid,strcat('There will be',32,num2st
30 -    fprintf(fid,strcat('We assume that the shaft
31 -    fprintf(fid,strcat('Optimized shaft diameter
32 -    fprintf(fid,strcat('Solidworks equations fil
33 -    fclose(fid);
34
35      %Write the equations file(s) (FILE(s) LINKEI
36      %You can make a different file for each sect
37      %or one single large file that includes all
38 -    fid = fopen(shaft_file,'w+t');
39 -    fprintf(fid,strcat('"diameter"=',num2str(new
40 -    fprintf(fid,strcat('"Shaft length"=',num2str
41 -    fprintf(fid,strcat('"Number of weights"=',nu
42 -    fclose(fid);
43 -  end
```

1. Lines 18-21: Defining all text files with their directories

2. Lines 23-33: Writing to the log text file

3. Lines 35-42: Writing to shaft part equation text file to modify SolidWorks dimensions


Group RE1 // CADCAM 2015
uOttawa
Team
Research and Development
Software by: Nathaniel Mailhot, Ali Sayed, Ibrahim El Wattar, Kirsten Campbell
December 10 2015
Input parameters for design:
Select maximum altitude of airship (m):
[10:1000]
10
Select atmospheric temperature (deg C):
[-15:50]
-15
Select continuous runtime (hours):
[2,4,6,8,10,12]
2
Enter desired operational speed of airship in calm winds (m/s):
[3:10]
3
Enter maximum payload length within cargo bay (cm):
[12:50]
12
Enter payload mass (g):
[500:1000]
500
Output log:
Mass, volume, lift and pitch
Total mass of airship: 5.7601 kg.
Total volume of envelope: 4.7393 m3.
Volume estimation accuracy: 114.042 Total lift of airship= 5.7601 kgf
Pitch range: [-10.8671: 7.145] deg.
Number of AA batteries: 6.
Envelope, frame and fin size
Cylinder and sphere radius= 713.1118 mm.
Cylinder length= 2039.4997 mm.
C:\groupRE1\Log\groupRE1_LOG.TXT
GENERATE DESIGN
FINISH AND CLOSE INTERFACE