MCG 4322A
Optimization Tutorial

In advance of the tutorial session, do the following:

1. Set up the client software for uOttawa's RemoteLabs https://it.uottawa.ca/students/remote_labs .
2. Set up local drive sharing for the remote client.
3. Download the GUI template files from Brightspace, save it to local drive shared with the client, extract zip if necessary.
4. With RemoteLabs, start Solidworks app, new part file.
5. Also with RemoteLabs, start Matlab, open the folder containing the GUI template files.

Purpose of Algorithmic Design Optimization

Algorithmic design optimization allows us to automatically:

- Verify the entire analysis for operational criteria.
- Check if the design can be adapted for situations adjacent to nominal operation.
- Improve the robustness of your CAD model for changes from algorithmic process.

A critical tool for *efficient* mechanical design

Relationship between Parametric Design and Optimization

Parametrize

To express in terms of parameters.

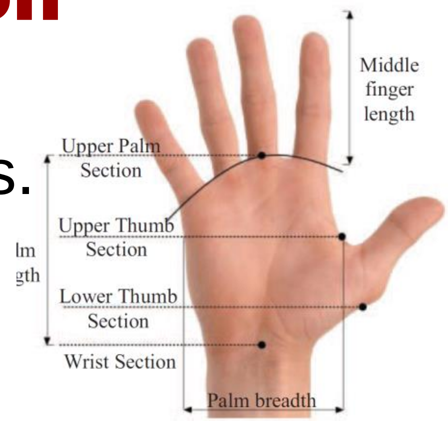
Parametrization

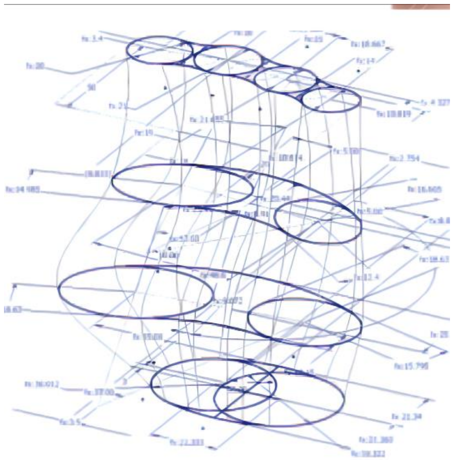
Process to express a model as a function of a set of independent quantities, i.e. the *parameters*.

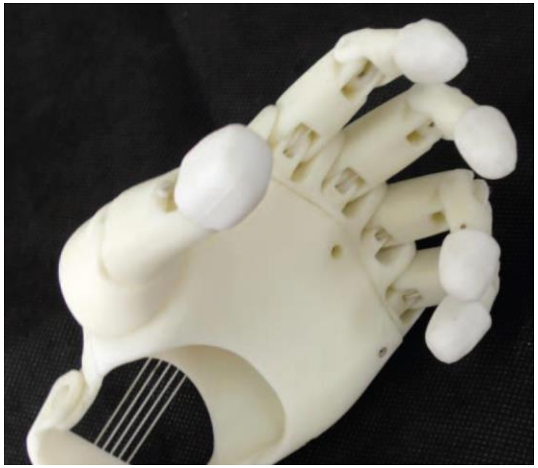
Parametric design

A set of parameters defines an optimal candidate via algorithmic procedure,

e.g. hand prosthesis design

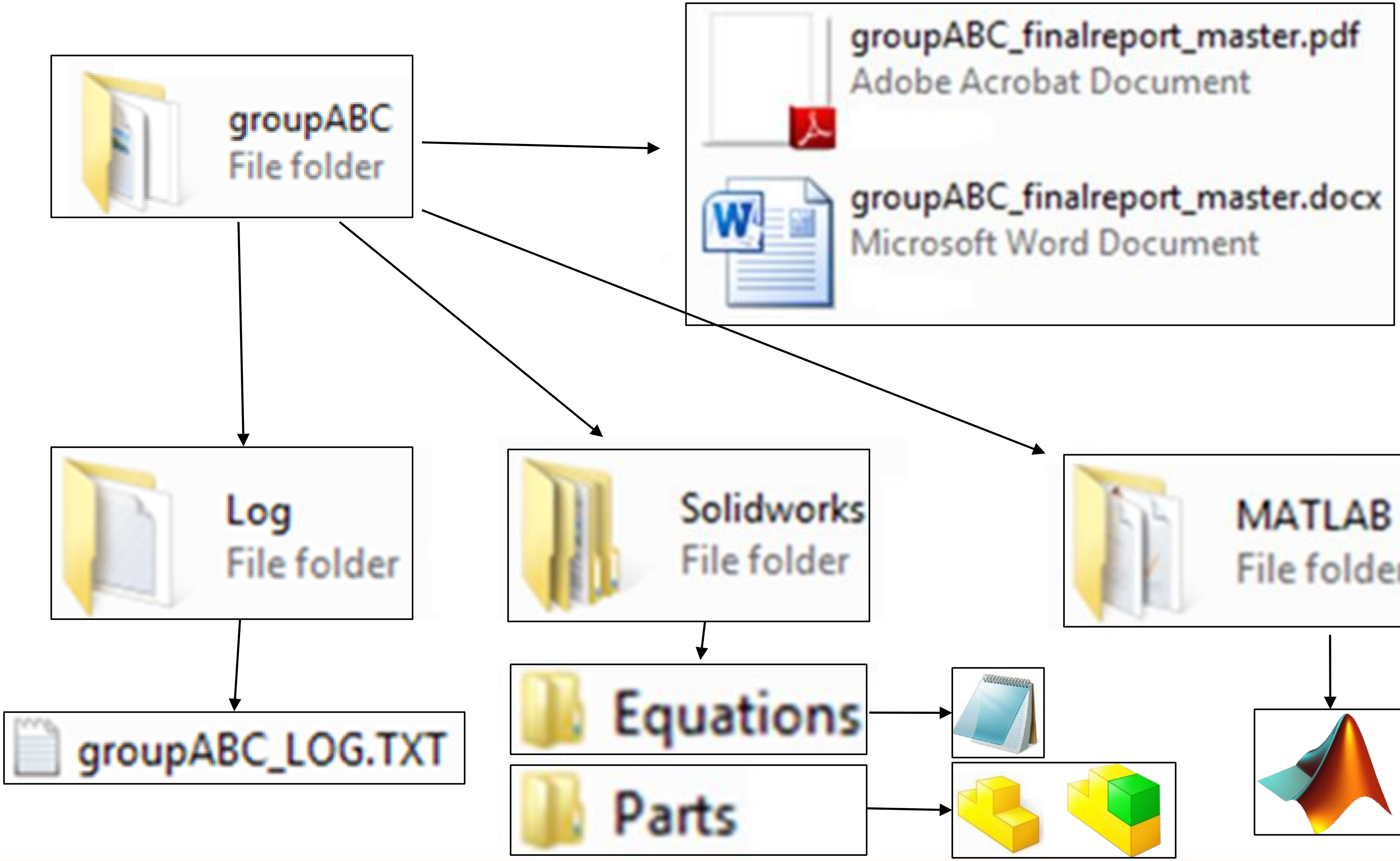






M. Bustamante et al., "A Parametric 3D-Printed Body-Powered Hand Prosthesis Based on the Four-Bar Linkage Mechanism," *2018 IEEE 18th International Conference on Bioinformatics and Bioengineering (BIBE)*, 2018, pp. 79-85.

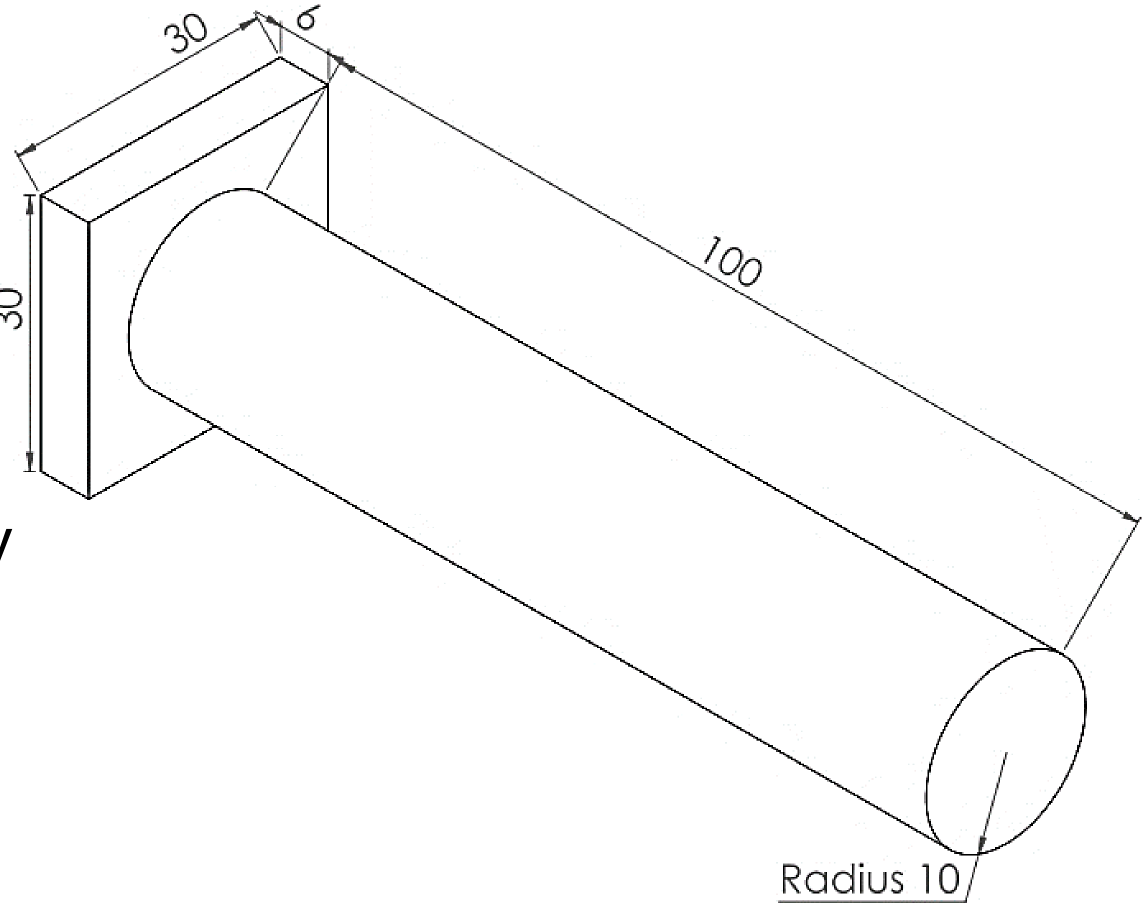
Required Final Submission File Organization



Step 1: Draft a cantilever beam

In Solidworks, model a beam with the following:

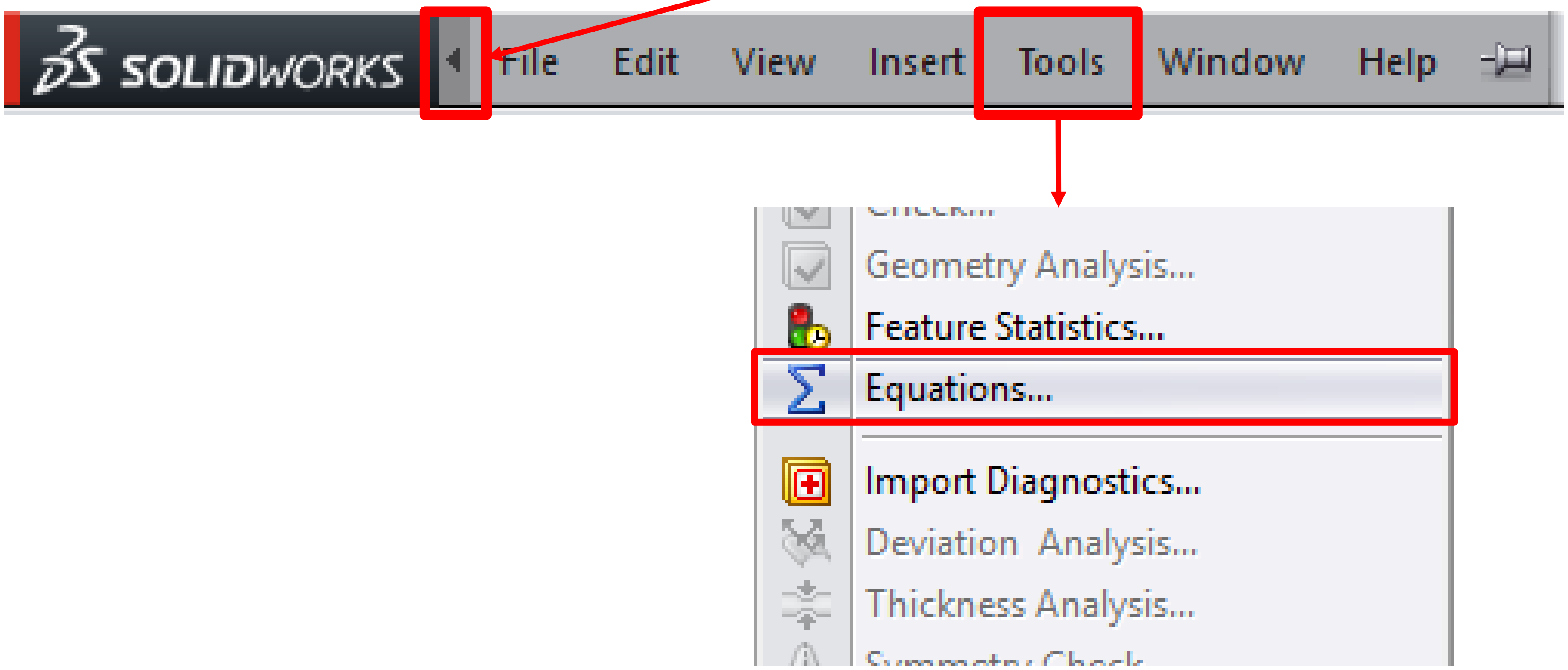
- Round cross section with 20mm diameter and 100mm length,
- Square base 30mm by 30 mm by 6 mm thick.



Save as: *shaft.sldprt*
in H:/groupABC/Solidworks/Parts

Step 2: Declaring global variables

In top toolbar, hover over the arrow and navigate to Tools → Equations

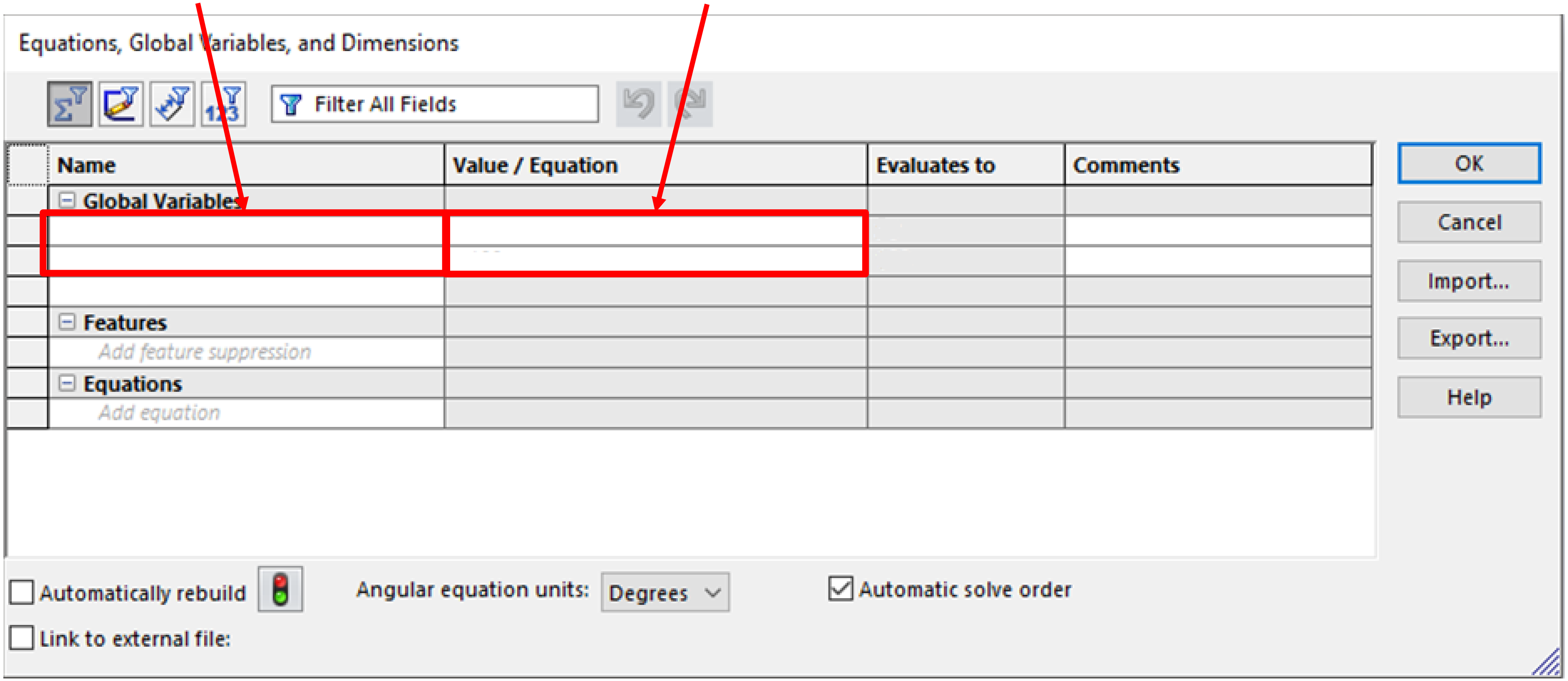


Step 2: Declaring global variables

Add two variables with exactly these names/values:

“Diameter” = 20 mm

“Length” = 100 mm



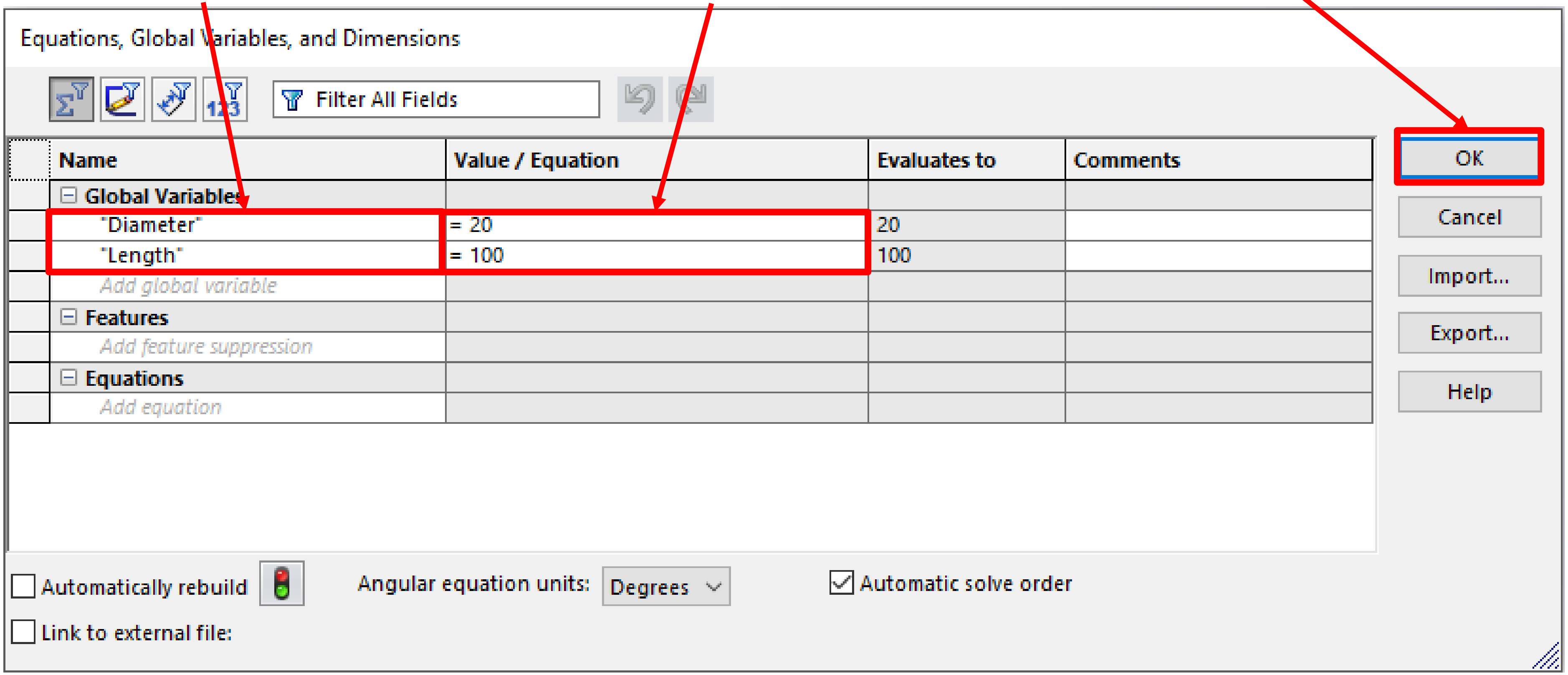
Step 2: Declaring global variables

Add two variables with exactly these names/values:

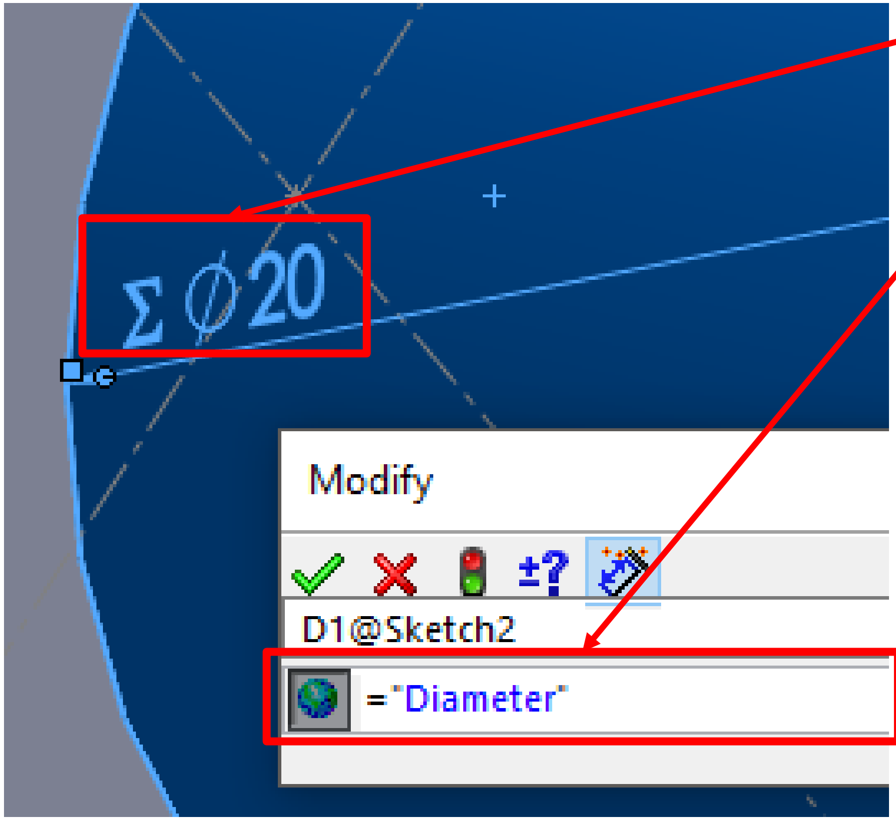
“Diameter” = 20 mm

“Length” = 100 mm

When done, hit ‘OK’



Step 3: Link dimension to a variable



- Double click beam diameter dimension,
- Click on input value field, type exactly:

  = "Diameter"
- Assign beam length dimension, exactly:

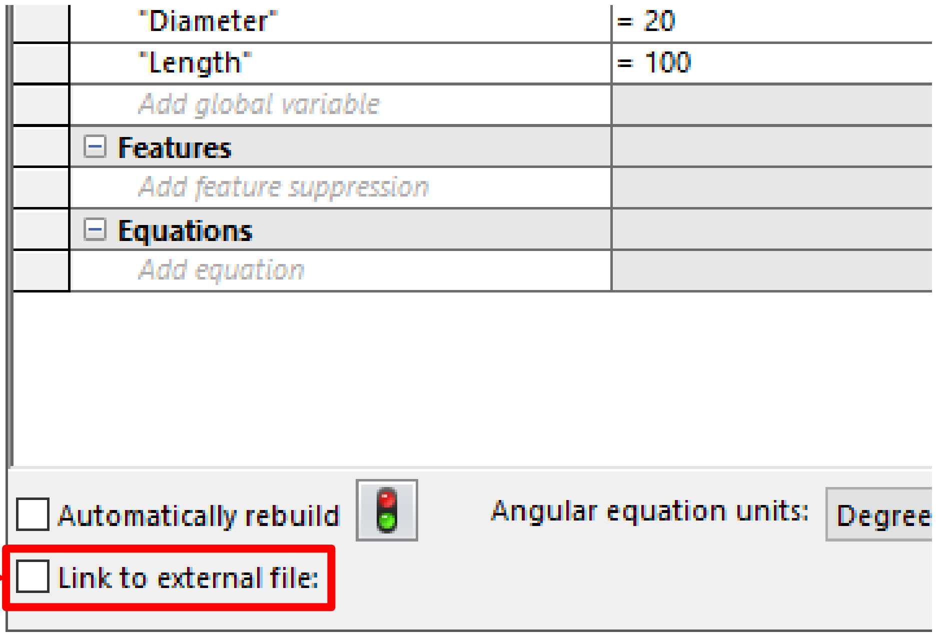
  = "Length"

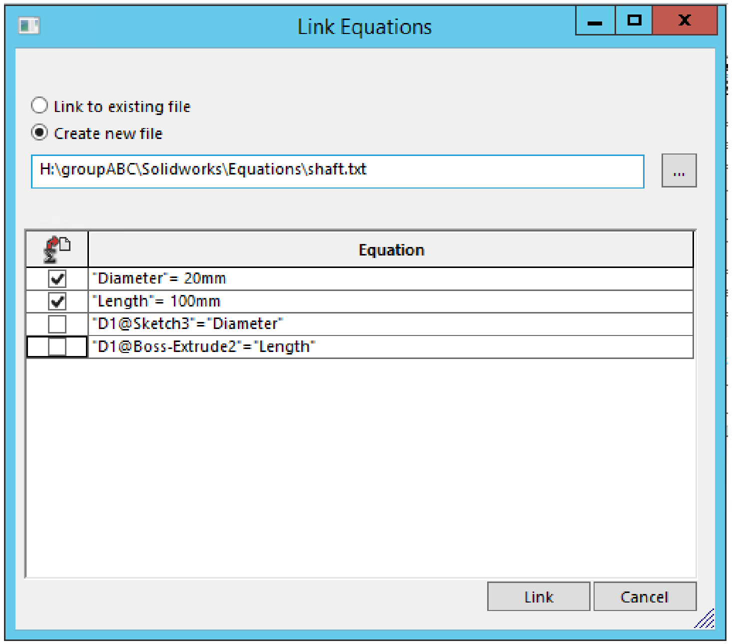
Variables are now linked to dimensions!

Indicated by the sigma Σ, next to the dimension.

Step 4: Linking variables to text file

- Go back to the Equations menu, (Tools → Equations)
- Bottom left, click “Link to external file:”
- The “Link Equations” pop-up menu will open.



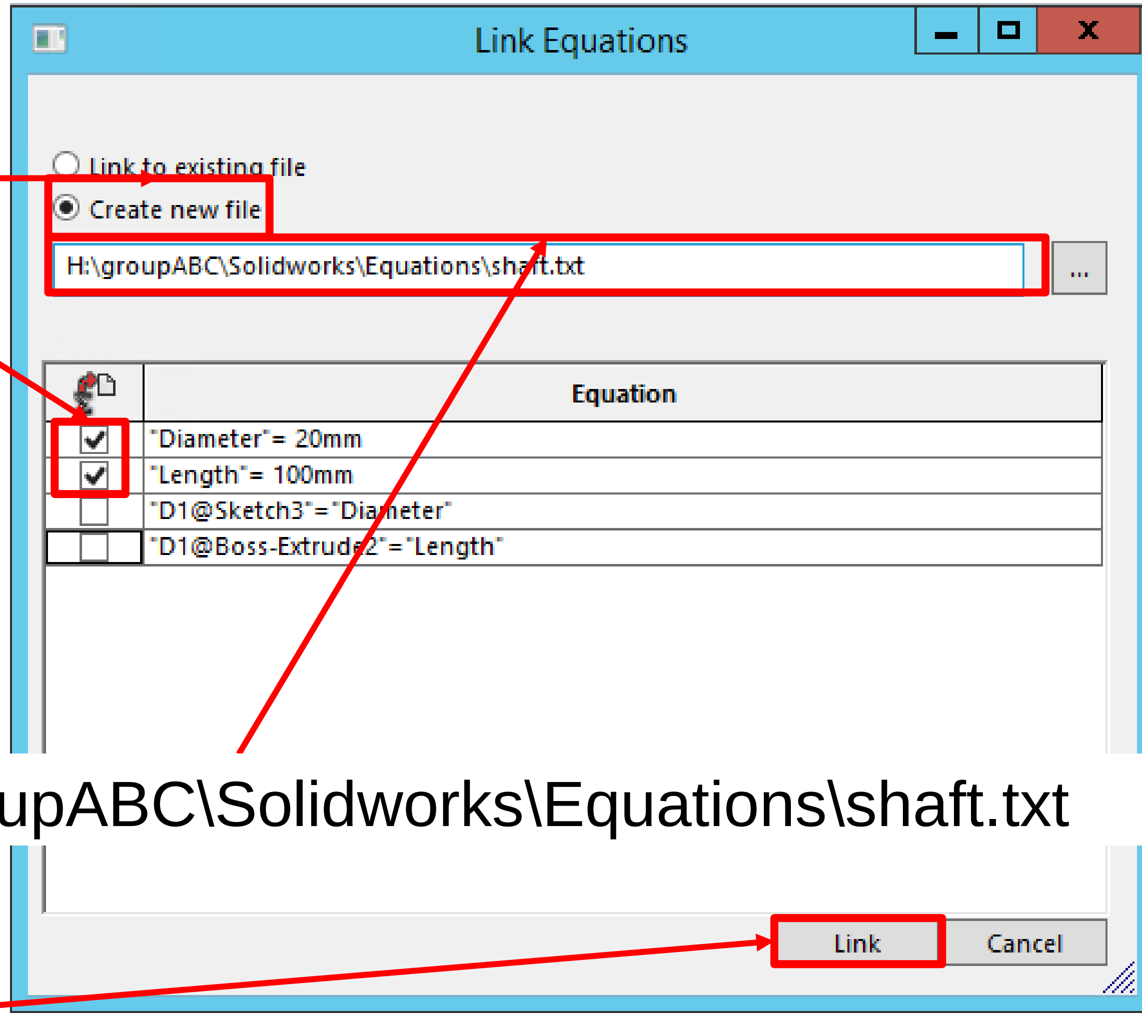


Step 4: Linking variables to text file

- Select 'Create new file',
- **Only** select variable entries,
- Define new text file, with the **correct** filepath:

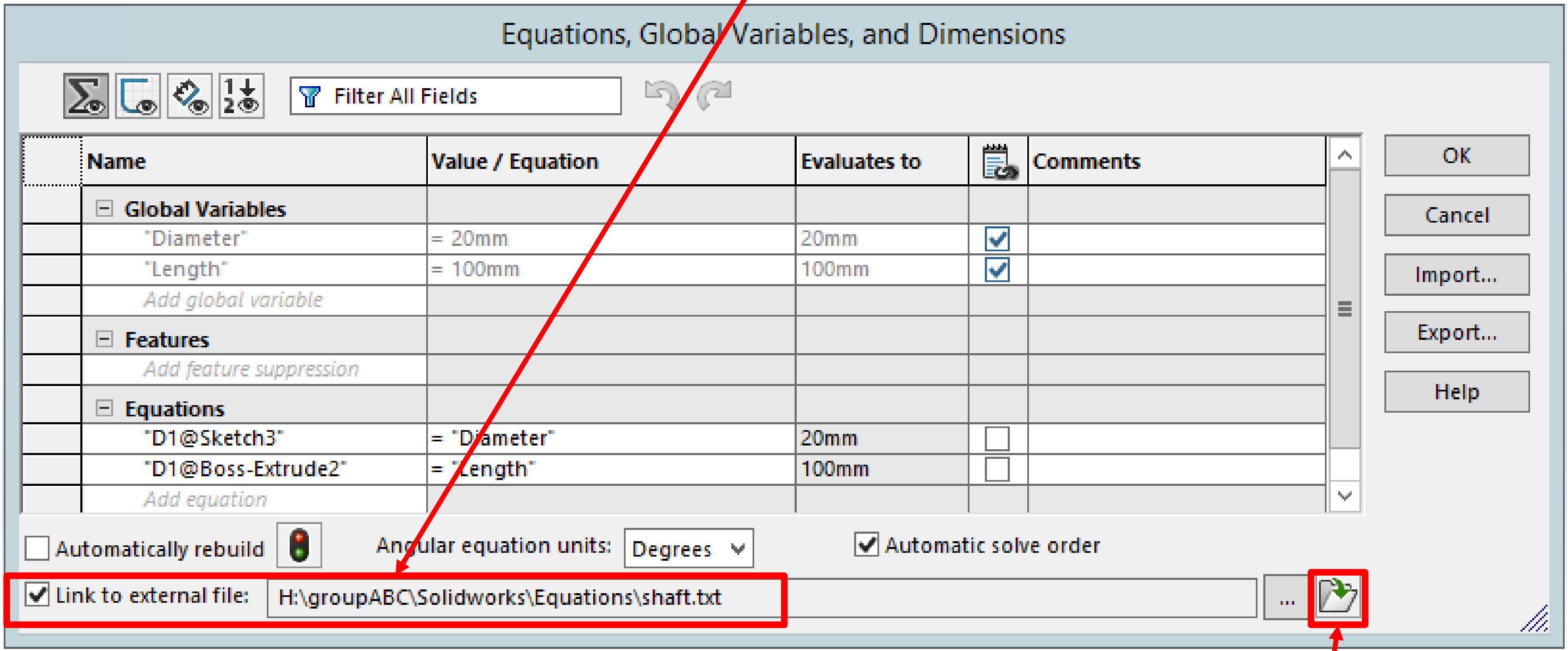
<DIRECT-PREFIX>\groupABC\Solidworks\Equations\shaft.txt

- Hit 'Link' Button. Should now have new file in folder.



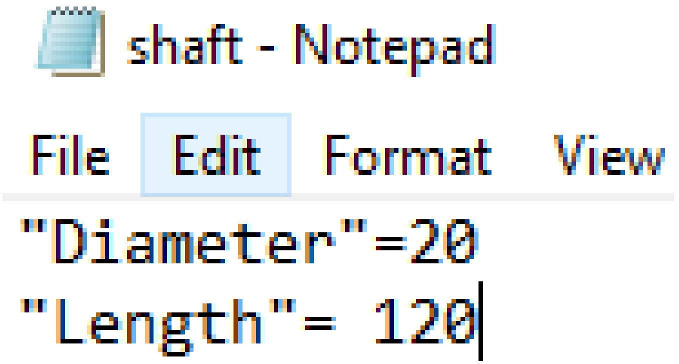
Step 4: Linking variables to text file

Checkbox filled and the file path included if all is good.



Next, click this to open linked text file, or open it directly from shared local drive.

Step 5: Change text file, rebuild part

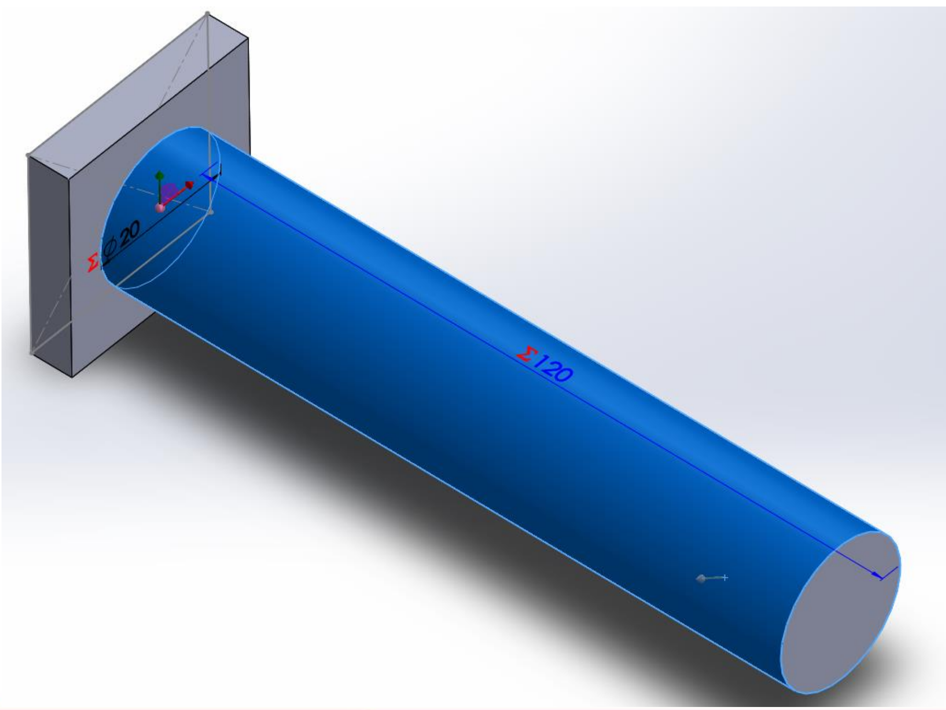


- Change dimension values,
- Keep identical format, change **only** numerical values.
- Save the text file,

- Rebuild Solidworks part...

…hopefully…

…it has changed!



Code expectations

IMPORRTANT: Include Code comments with MATLAB code

In MCG4322A:

- All code has to be done with MATLAB, no Octave, Python, etc.
- Code comments are required, keep clear and concise.
- 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.

MATLAB GUI Template

The template's MATLAB folder consist of two files:

1. MAIN.mlapp

   - Graphic User Interface (GUI) template,

   - uses MATLAB's built-in 'App Designer' tool,

   - passes user selected parameters to Design_code.m,

   - displays log file generated by Design_code.m ,

   https://www.mathworks.com/help/matlab/app-designer.html

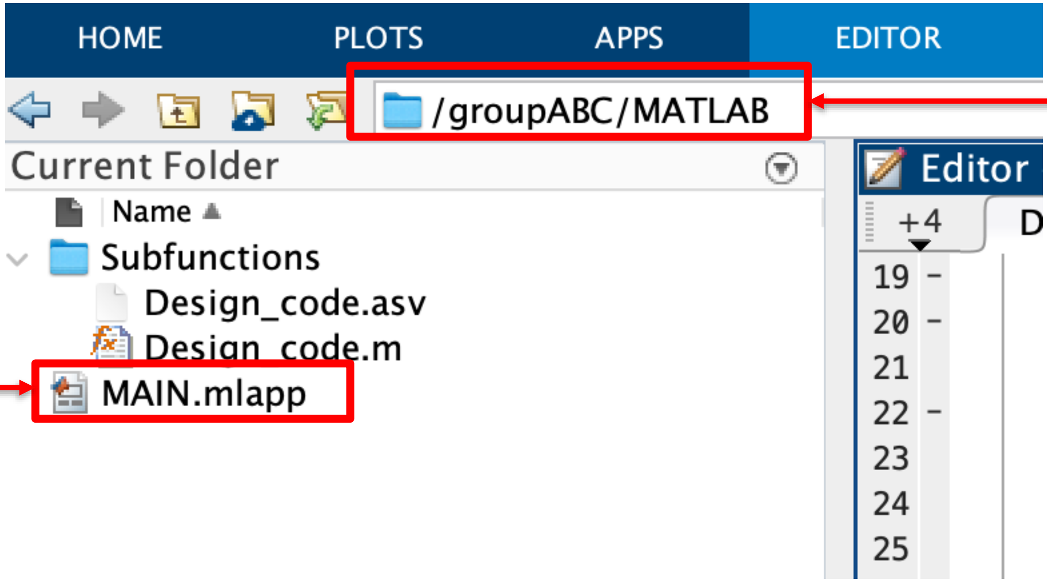
2. Subfunctions folder

   - Contains Design_code.m, (and other files you may add),

   - Design code is where ALL the analysis equations, functions, and optimization algorithms are programmed.

Step 6: MATLAB file navigation

1. In current directory field, input project MATLAB folder directory, …\groupABC\MATLAB,
2. Current folder pane shows file structure.
3. Open MAIN.mlapp

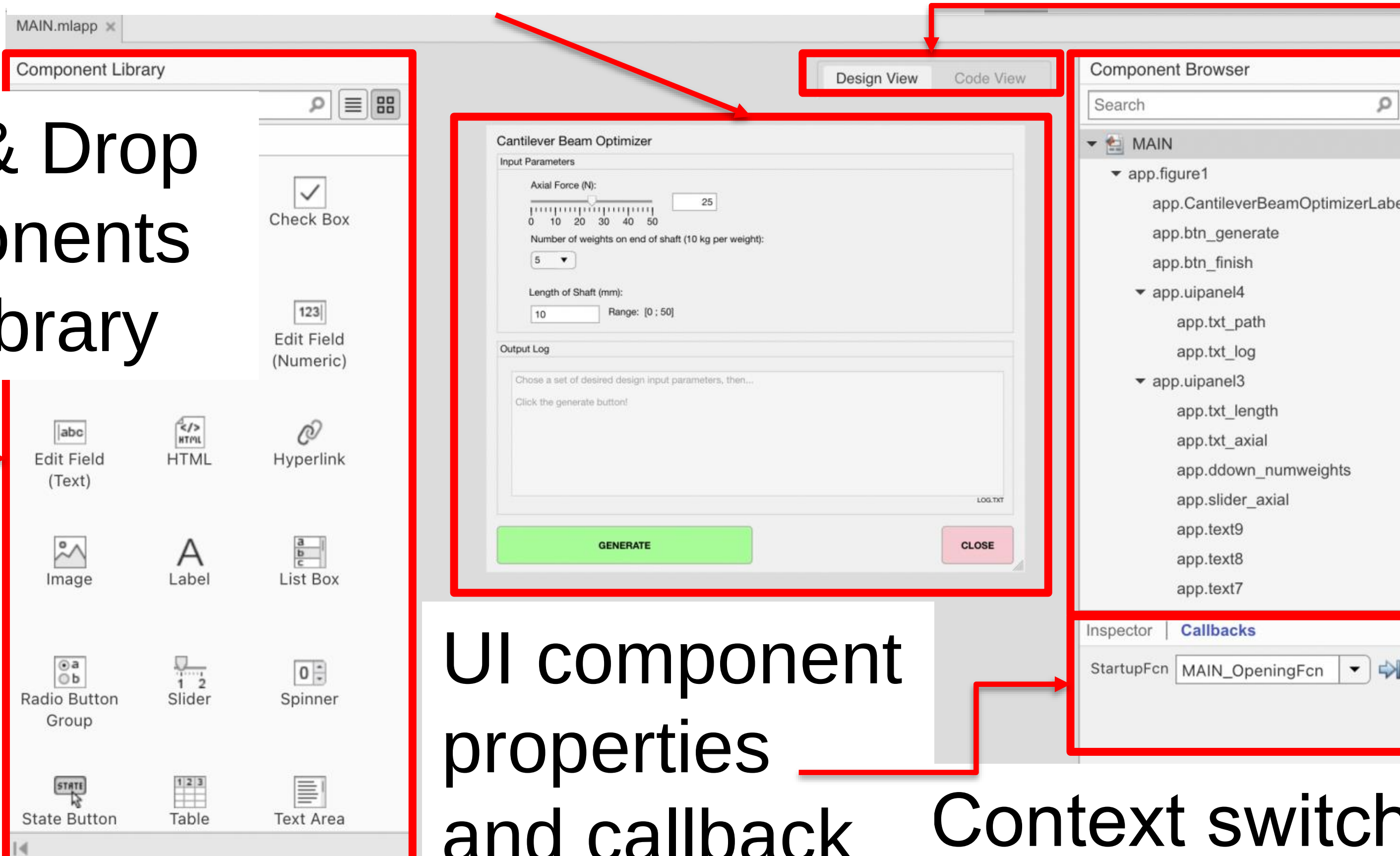


Step 7: App Designer – design interface

Dynamic selection of UI elements to be modified

UI component browser tree

Drag & Drop components from library

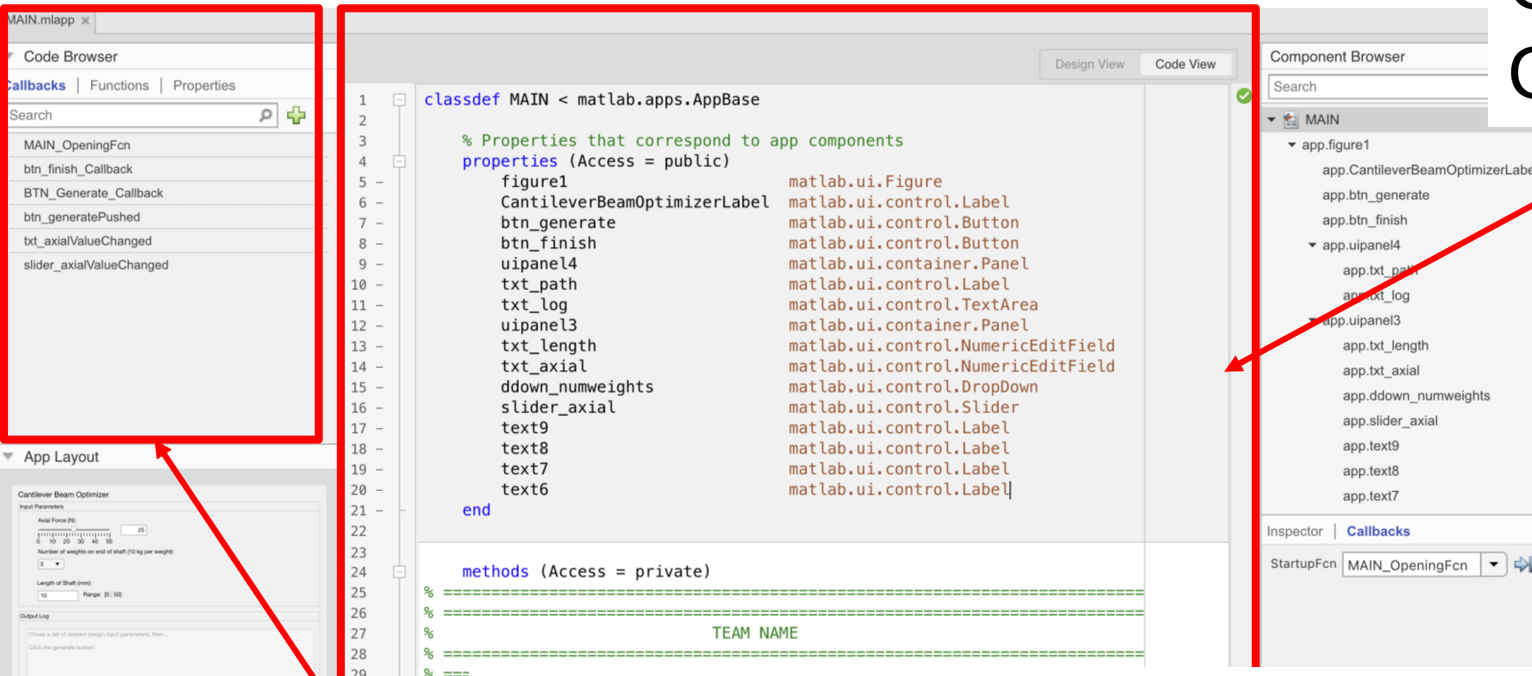


UI component properties and callback functions

Context switcher … select ‘Code View’

Step 8: App Designer – code interface

Code that GUI uses



Left side now shows ‘Code Browser’,

- auto generated by added UI callbacks,
- right click -> delete to remove callback functions

Step 9: Translate GUI inputs to code

Template GUI has 3 input elements corresponding to three parameters.

1. Slider

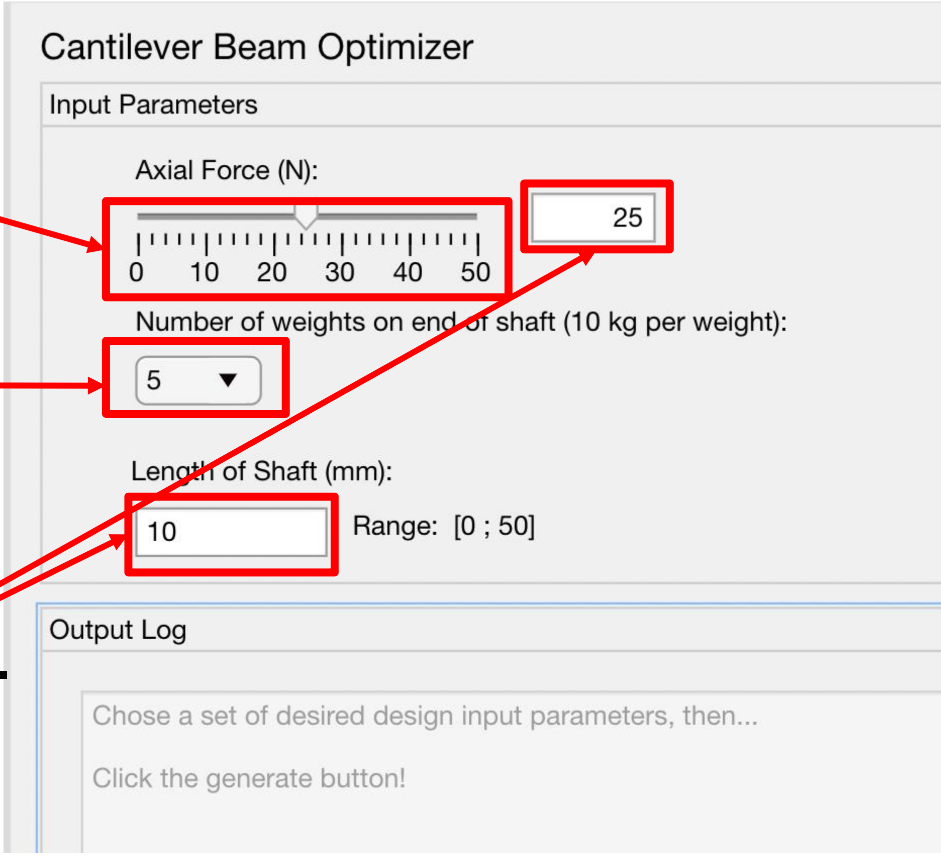
Takes the axial force acting on cantilever beam.

2. Drop down list

Selection from a range of possible number of weights hanging on the end of beam.

3. Edit text field

Reads the value input into the text box for shaft length.



Step 9: Translate GUI inputs to code

Specific examples of code in MAIN.mlapp

- Lines 109-113, we define variables for each input by reading values from GUI

```
axial_force = app.slider_axial.Value;
```

- Line 117, we call the Design_code subfunction, passing our inputs as *arguments*.

```
Design_code(axial_force, num_weights,
shaft_length)
```

- Lines 122-130, we display output of log file.

```
log_contents = char(fread(fid)’);
```

Step 10: Analysis and design code

1. Open the **Design_code.m** file,
2. Line 1, function definition. Arguments: only sequence and data types matter.

```
function Design_code(axial_force,
   number_of_weights, shaft_length)
```

3. Line 16, call the shaft optimization subfunction
4. This function *returns* an output, the optimized shaft diameter,

Study the optimization algorithm example at the end of the file in the subfunction definition.

Step 11: Modify text files with MATLAB

We use MATLAB built-in text editing functions to write to the equation and log text files.

- Directories are hard-coded, so if you change them you need to update code accordingly,
- Text output from MATLAB code **must have the same format** as the original SW's equation file,
- Common errors: Changing character spacing, incorrect variable names, changing line order/spacing, missing variables,

Verify! When MATLAB changes equation file, check that Solidworks rebuilds properly for all scenarios.

Step 11: Modify text files with MATLAB

How to access text files with MATLAB code:

1. Save directory prefix (Required only once per function)

```
directory_prefix_string = extractBefore(pwd,
                                        "groupABC");
```

2. Define the filepath of text file to be modified, save into string variable:

```
your_file =
     strcat(directory_prefix_string,'\groupABC\
FOLDER\YOURFILE.TXT');
```

3. Use string variable to open text file for writing:

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

Step 11: Modify text files with MATLAB

How to write to text files with MATLAB code:

4. Use *fprintf()* *to w*rite to the text file:

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

– *strcat* combine multiple strings and variables into a single string
- https://www.mathworks.com/help/matlab/ref/fprintf.html

5. When done writing to file, terminate the filestream:

```
fclose(fid);
```

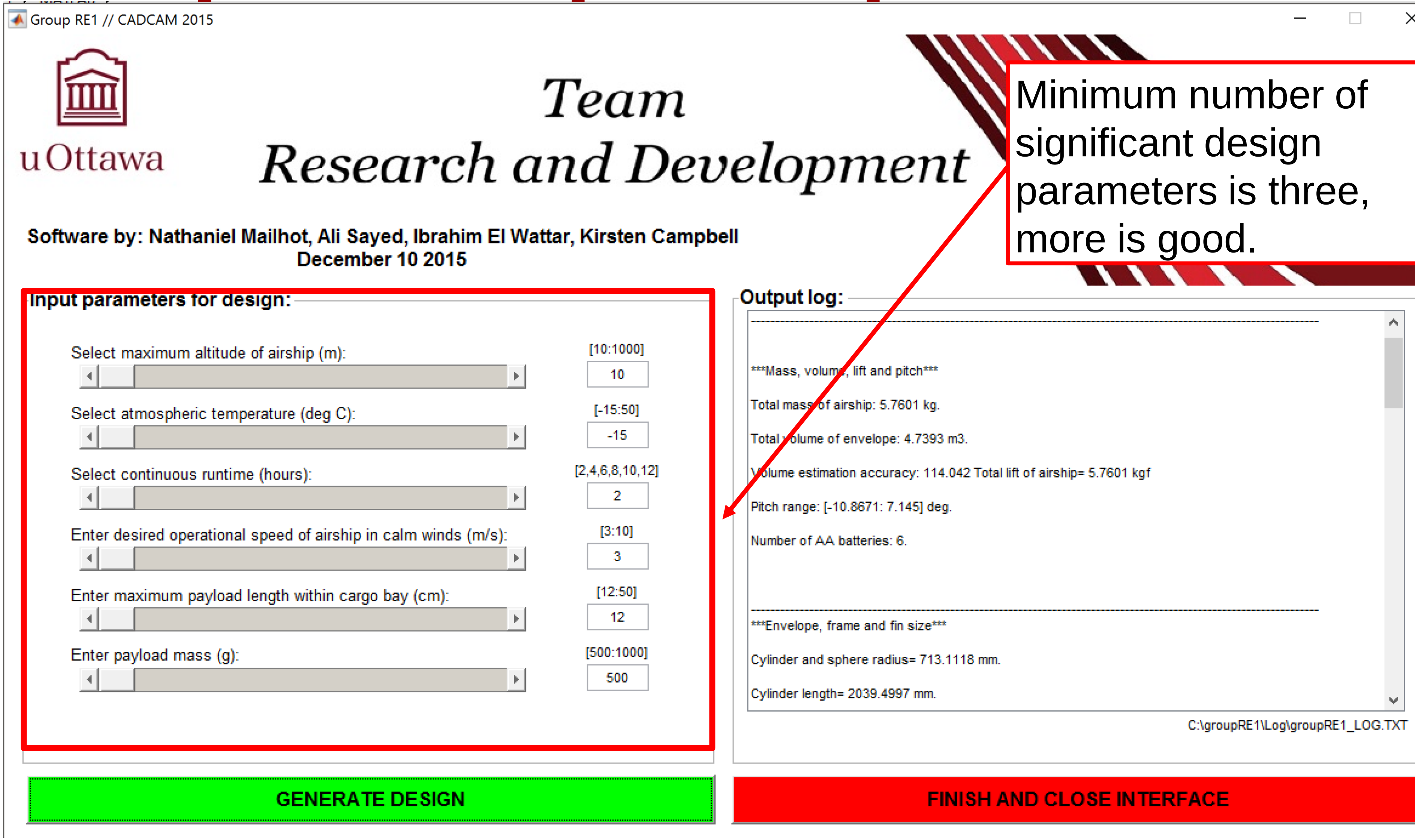
Step 12: Log text file for results output

The output log file is where you display all results. The log text file is your responsibility to format. Make sure that it is:

- clear and human-readable,
- input parameters selected by user are displayed first,
- displays full set of calculated outputs, including values that do not reflect as Solidworks model changes.

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

Example of Completed Optimization GUI



Minimum number of significant design parameters is three, more is good.

Debugging MATLAB

Brief, live example of debugging with MATLAB.

- Play
- Console input, output and workspace context
- Step
- Continue
- End

Documentation (extremely useful to save you time!)

https://www.mathworks.com/help/matlab/matlab_prog/debugging-process-and-features.html

https://www.mathworks.com/help/matlab/matlab_prog/set-breakpoints.html

https://www.mathworks.com/help/matlab/matlab_prog/examine-values.html