

# 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](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.

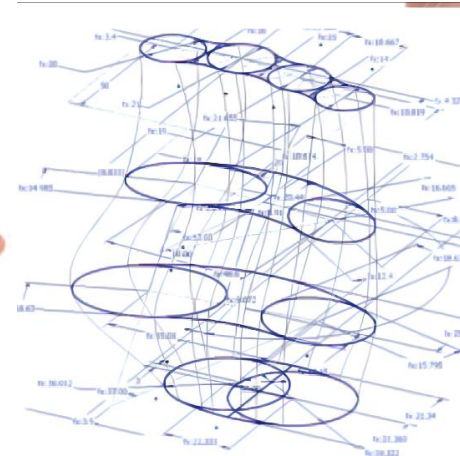
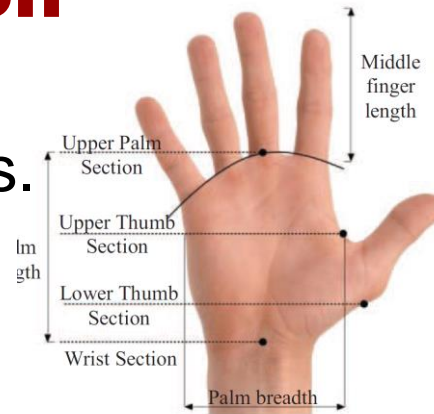
## Parametrization

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

## Parametric design

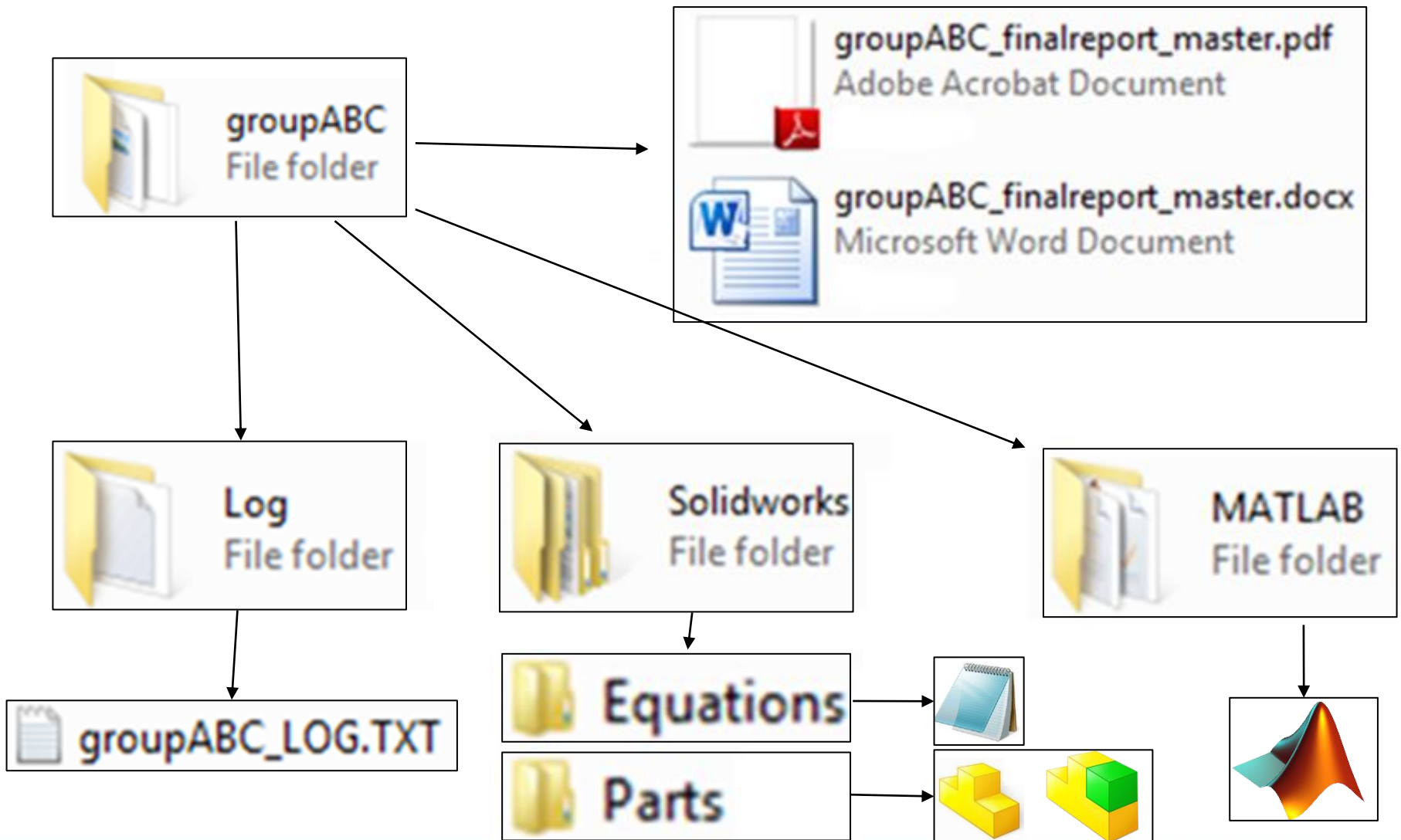
A set of parameters defines an optimal candidate via algorithmic procedure,

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.

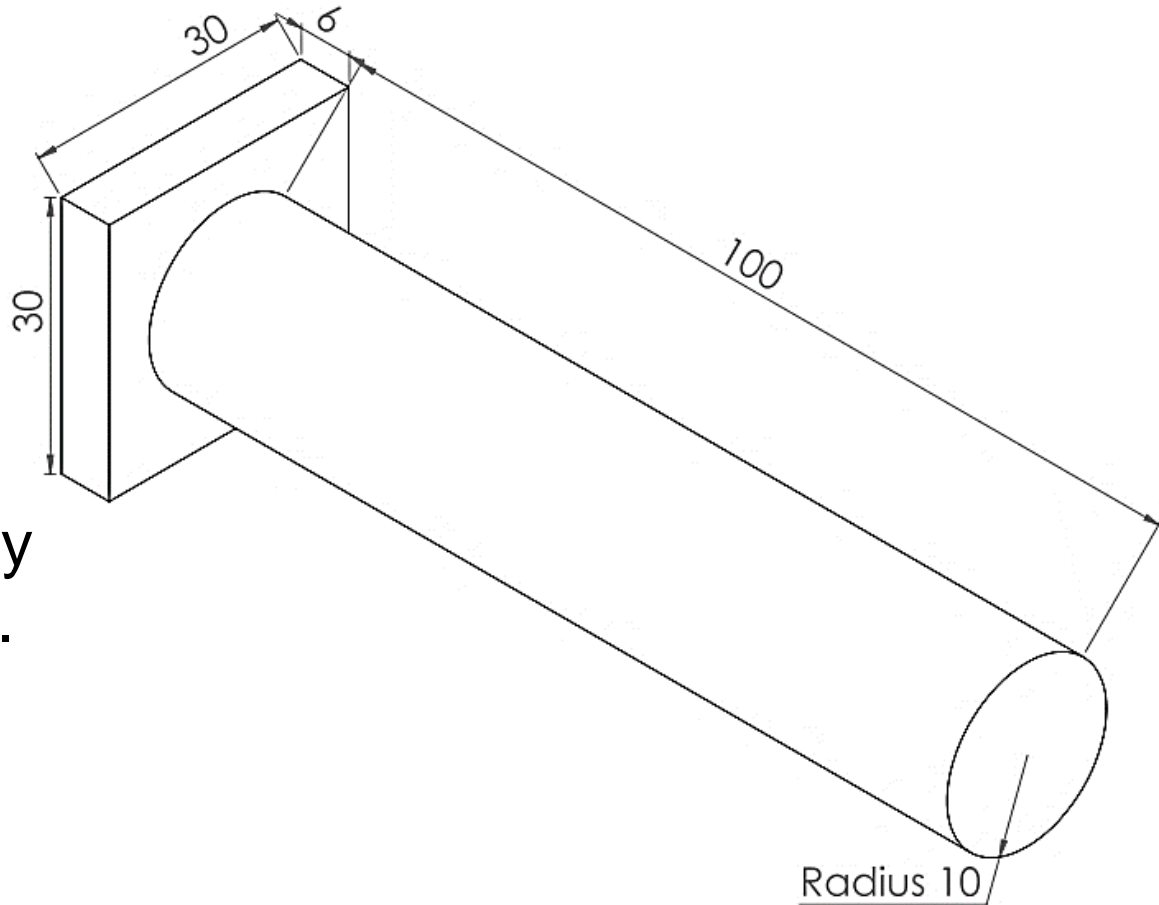
# Required Final Submission File Organization



# Step 1: Draft a cantilever beam

In Solidworks, model a beam with the following:

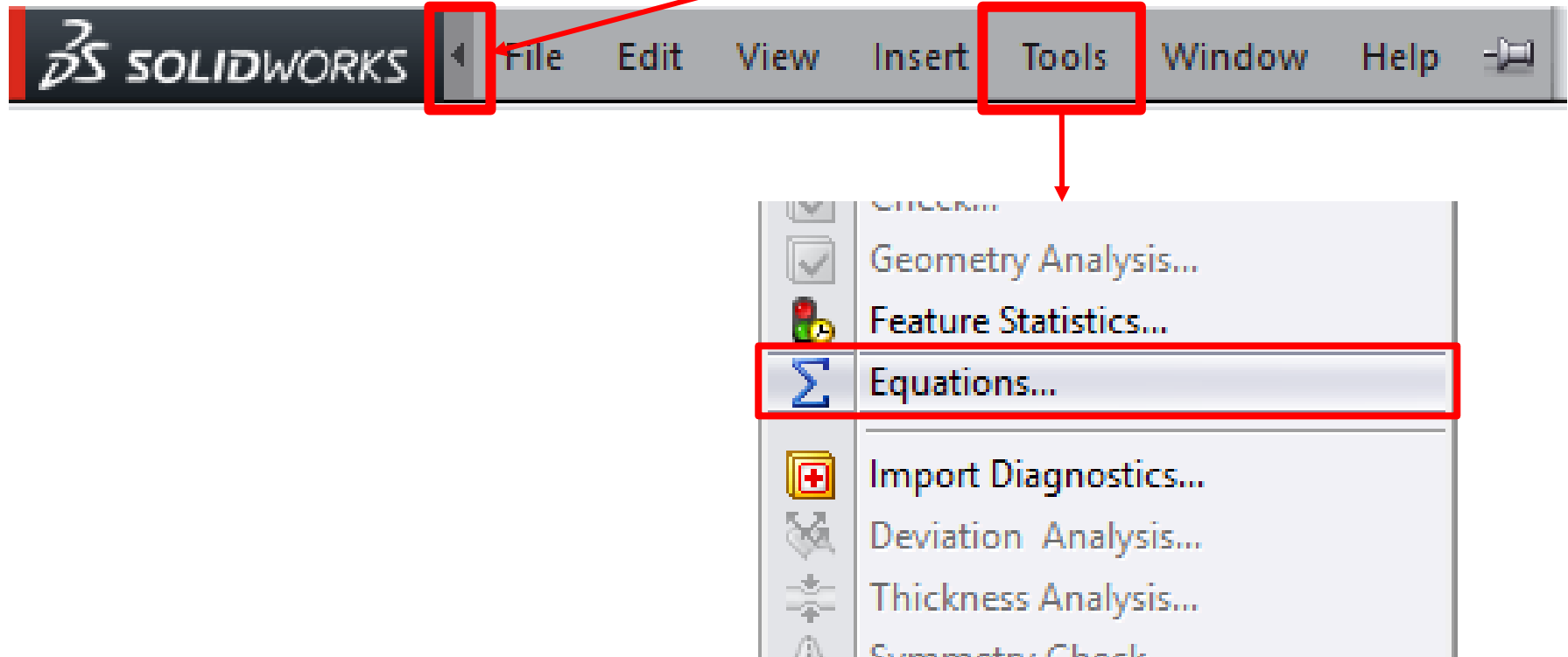
- Round cross section with 20mm diameter and 100mm length,
- Square base 30mm by 30 mm by 6 mm thick.



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

Equations, Global Variables, and Dimensions

Name	Value / Equation	Evaluates to	Comments
 Global Variables			
 Features			
<i>Add feature suppression</i>			
 Equations			
<i>Add equation</i>			

☐ Automatically rebuild  Angular equation units:  ☒ Automatic solve order

☐ Link to external file:

OK  
Cancel  
Import...  
Export...  
Help



# Step 2: Declaring global variables

Add two variables with exactly these names/values:


“Diameter” = 20 mm  
“Length” = 100 mm

When done, hit  
‘OK’

Equations, Global Variables, and Dimensions

 Filter All Fields 

Name	Value / Equation	Evaluates to	Comments
<b>Global Variables</b>			
"Diameter"	= 20	20	
"Length"	= 100	100	
<i>Add global variable</i>			
<b>Features</b>			
<i>Add feature suppression</i>			
<b>Equations</b>			
<i>Add equation</i>			

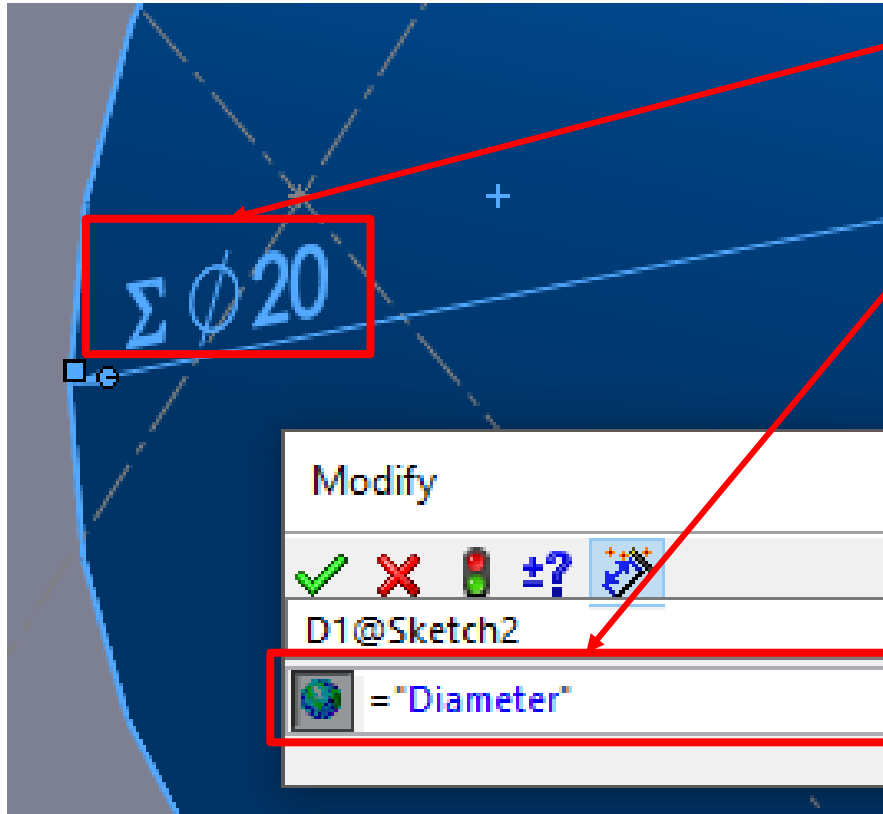
☐ Automatically rebuild  Angular equation units: Degrees ☒ Automatic solve order

☐ Link to external file:

OK  
Cancel  
Import...  
Export...  
Help



# 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  $\Sigma$ , 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.

	"Diameter"	= 20
	"Length"	= 100
	Add global variable	
<input type="checkbox"/>	Features	
	Add feature suppression	
<input type="checkbox"/>	Equations	
	Add equation	

☐ Automatically rebuild  Angular equation units: Degree

☐ Link to external file:

Link Equations

☐ Link to existing file  
☒ Create new file

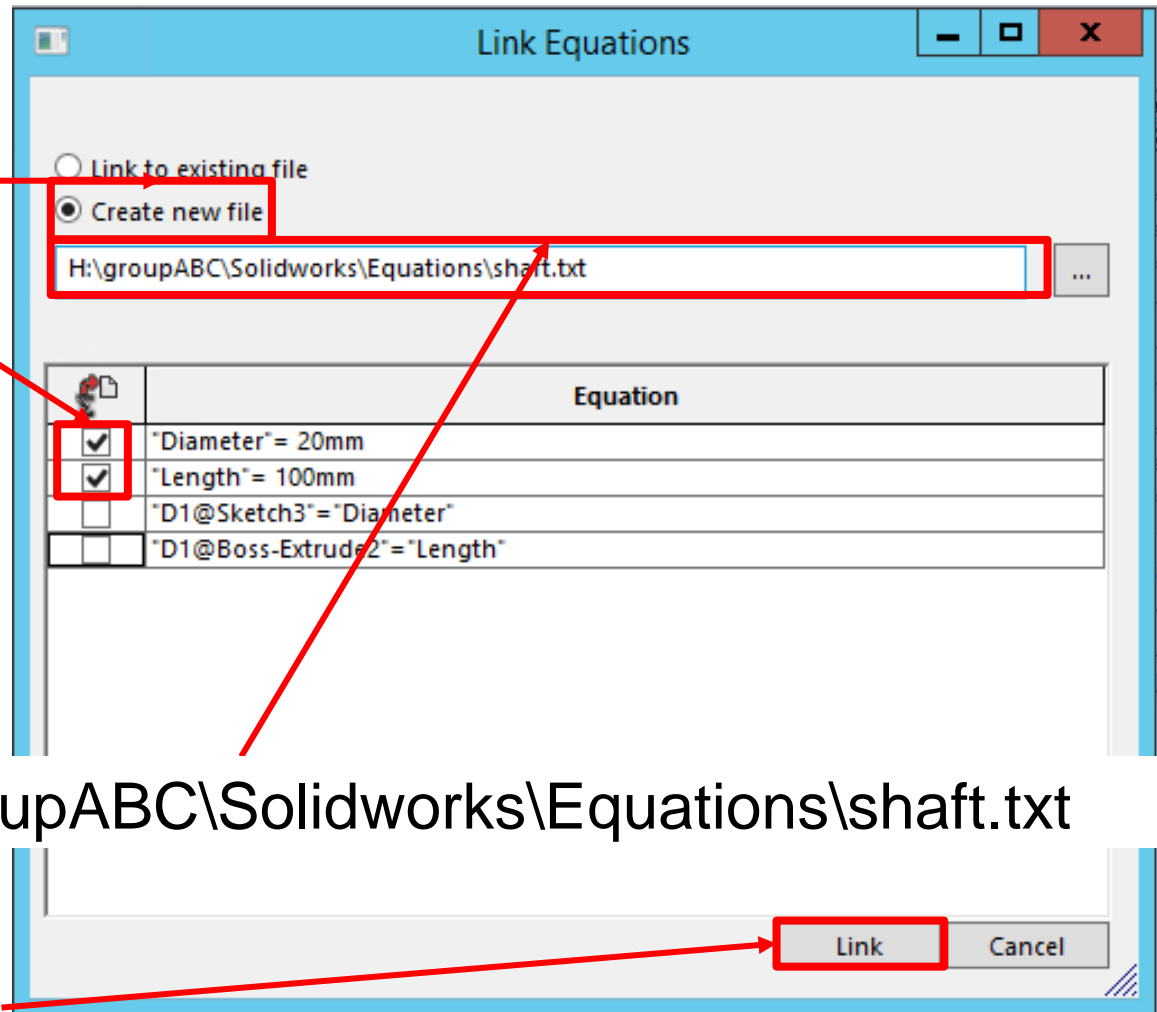
H:\groupABC\Solidworks\Equations\shaft.txt

	Equation
<input checked="" type="checkbox"/>	"Diameter" = 20mm
<input checked="" type="checkbox"/>	"Length" = 100mm
<input type="checkbox"/>	"D1@Sketch3" = "Diameter"
<input type="checkbox"/>	"D1@Boss-Extrude2" = "Length"

Link Cancel

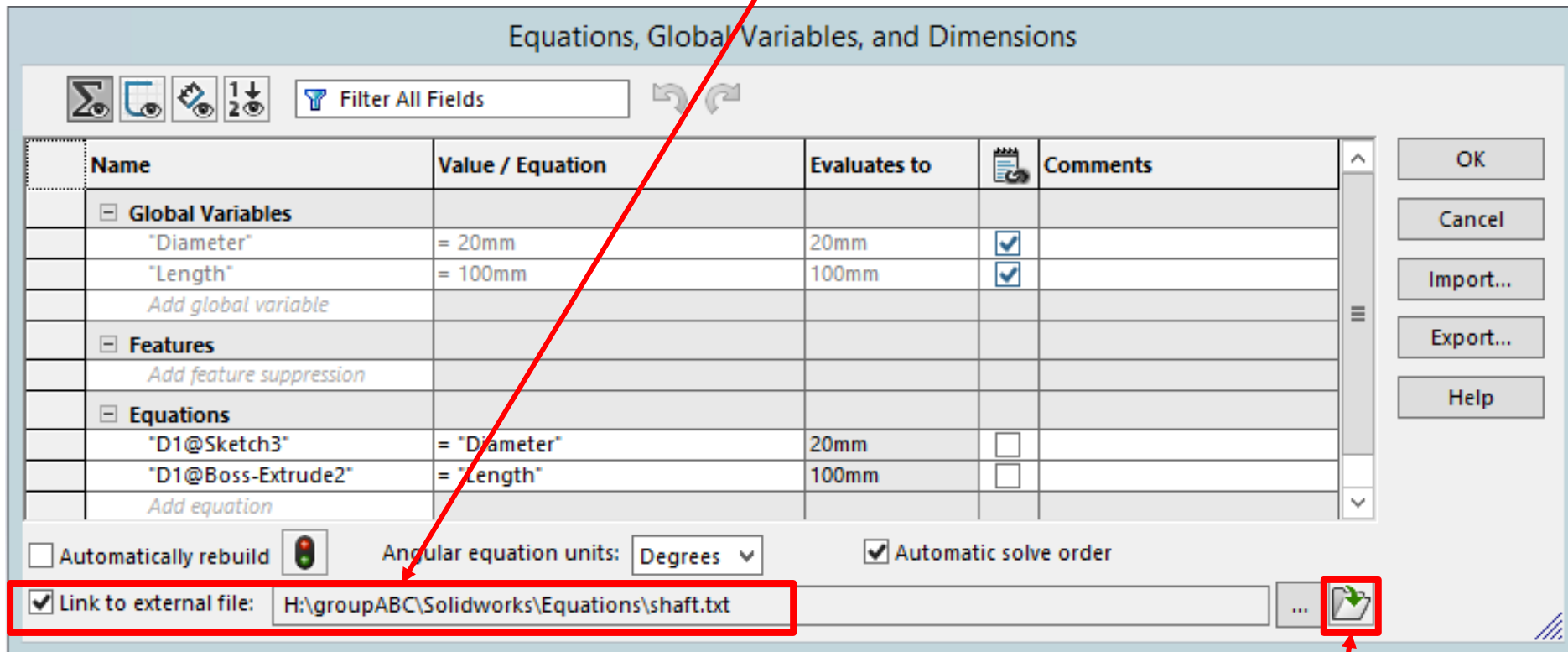
# Step 4: Linking variables to text file

- 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...

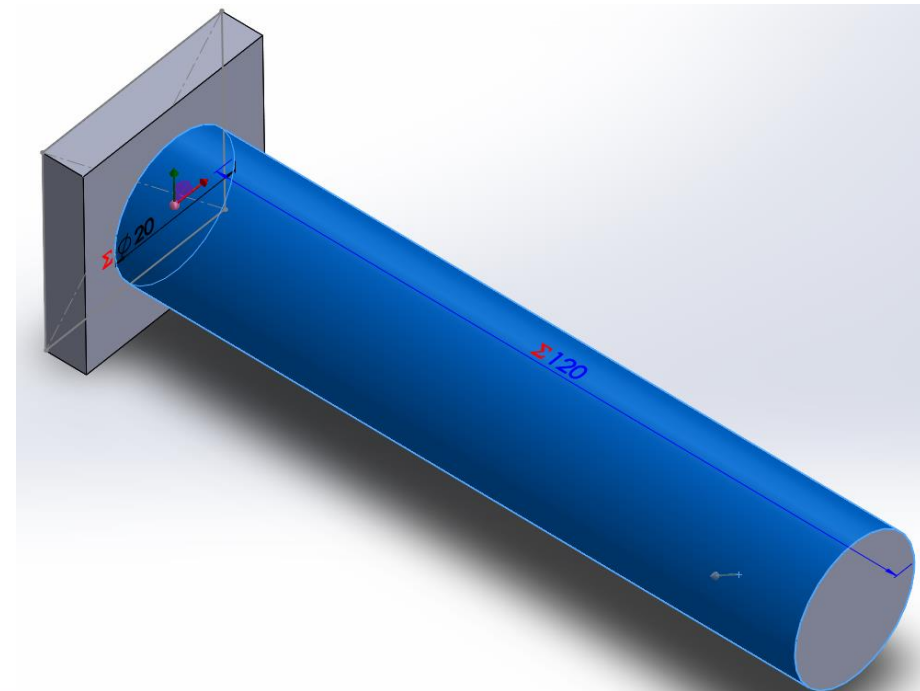
...it has changed!

 shaft - Notepad

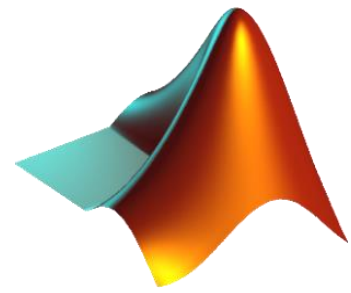
File Edit Format View

"Diameter"=20

"Length"= 120



# Code expectations

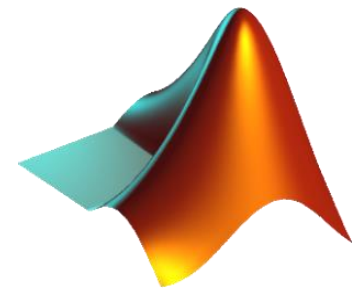


**IMPORTANT:** Include Code comments with MATLAB code

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 ,

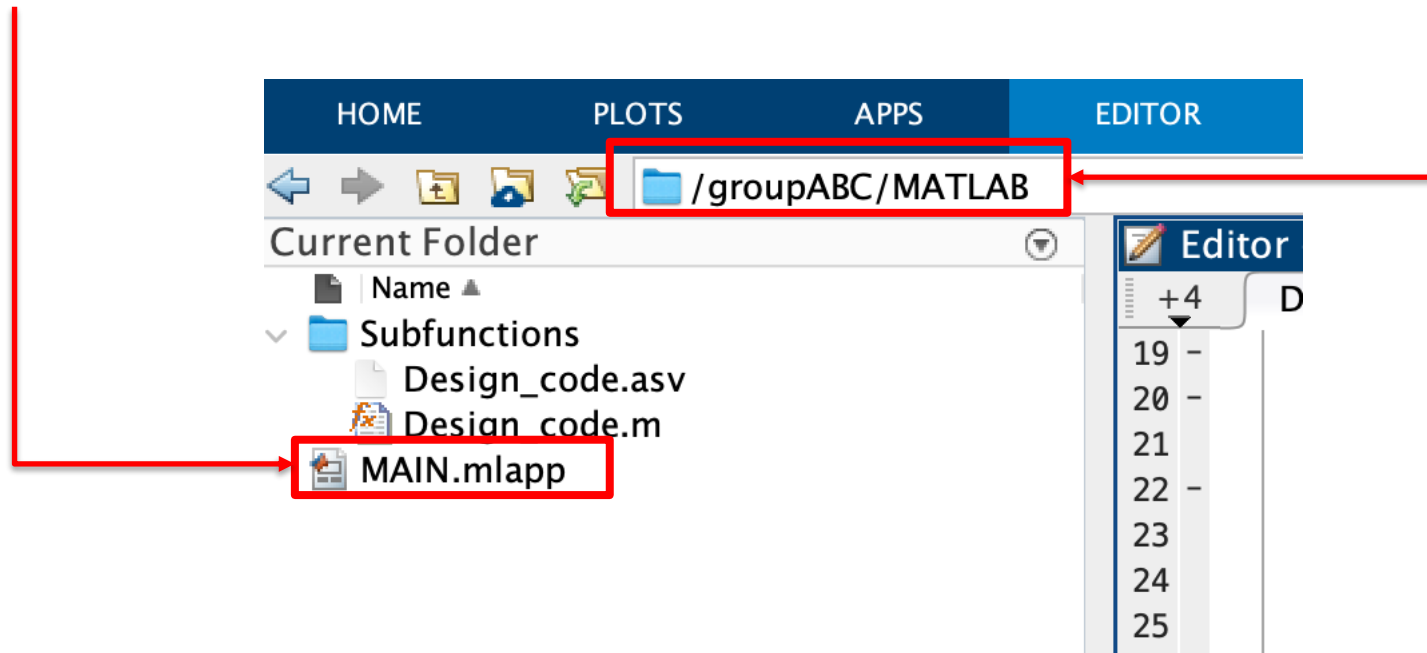
<https://www.mathworks.com/help/matlab/app-designer.html>

## 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



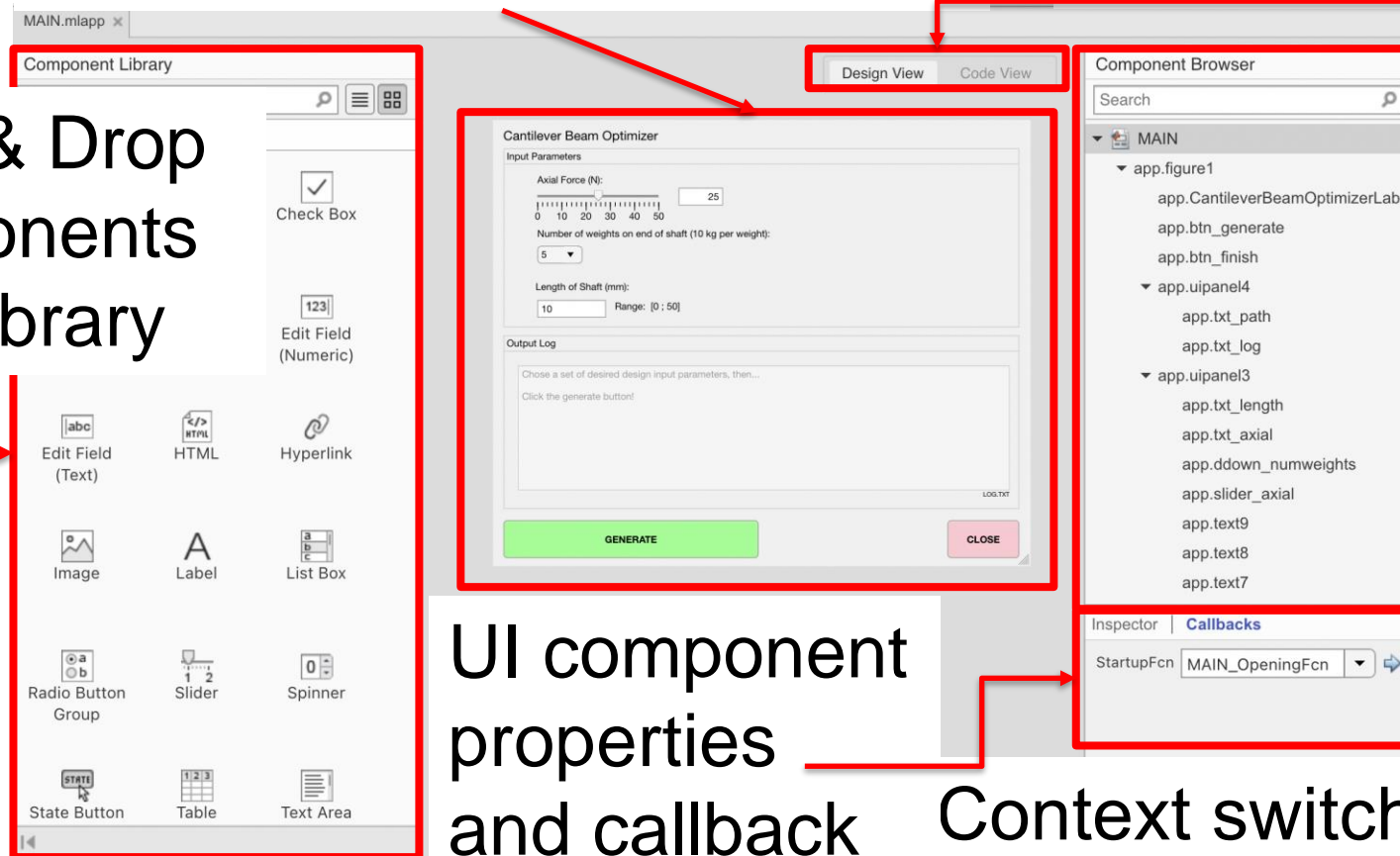


# Step 7: App Designer – design interface

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

The screenshot displays the MATLAB App Designer interface. On the left, the 'Code Browser' is visible, showing a list of callbacks including 'MAIN\_OpeningFcn', 'btn\_finish\_Callback', 'BTN\_Generate\_Callback', 'btn\_generatePushed', 'txt\_axialValueChanged', and 'slider\_axialValueChanged'. Below this is the 'App Layout' showing a 'Cantilever Beam Optimizer' GUI with various input fields and a 'GENERATE' button. The central pane shows the 'Code View' with the following MATLAB code:

```
1 classdef MAIN < matlab.apps.AppBase
2
3     % Properties that correspond to app components
4     properties (Access = public)
5
6         figure1          matlab.ui.Figure
7         CantileverBeamOptimizerLabel matlab.ui.control.Label
8         btn_generate      matlab.ui.control.Button
9         btn_finish        matlab.ui.control.Button
10        uipanel4           matlab.ui.container.Panel
11        txt_path           matlab.ui.control.Label
12        txt_log            matlab.ui.control.TextArea
13        uipanel3           matlab.ui.container.Panel
14        txt_length         matlab.ui.control.NumericEditField
15        txt_axial          matlab.ui.control.NumericEditField
16        ddown_numweights   matlab.ui.control.DropDown
17        slider_axial       matlab.ui.control.Slider
18        text9              matlab.ui.control.Label
19        text8              matlab.ui.control.Label
20        text7              matlab.ui.control.Label
21        text6              matlab.ui.control.Label
22
23    end
24
25    methods (Access = private)
26
27        % =====
28        % TEAM NAME
29        % =====
30
31        % Dev
32        % GRC
33        % Uni
34        % Mec
35        % Lat
36        % MAT
37        % =====
38    end
```

On the right, the 'Component Browser' shows a tree structure of the app's components, including 'app.figure1', 'app.CantileverBeamOptimizerLabel', 'app.btn\_generate', 'app.btn\_finish', 'app.uipanel4', 'app.txt\_path', 'app.txt\_log', 'app.uipanel3', 'app.txt\_length', 'app.txt\_axial', 'app.ddown\_numweights', 'app.slider\_axial', 'app.text9', 'app.text8', and 'app.text7'. Below the Component Browser is the 'Inspector' and 'Callbacks' section, showing the 'StartupFcn' set to 'MAIN\_OpeningFcn'.

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.

The screenshot shows a GUI titled "Cantilever Beam Optimizer" with an "Input Parameters" section. It contains three input elements: a slider for "Axial Force (N)" with a range from 0 to 50 and a value of 25; a drop-down list for "Number of weights on end of shaft (10 kg per weight)" with a value of 5; and a text field for "Length of Shaft (mm)" with a value of 10 and a range of [0 ; 50]. Red boxes highlight each of these three input elements, and red arrows point from the text descriptions on the left to their respective GUI components. Below the input parameters is an "Output Log" section with the text: "Chose a set of desired design input parameters, then..." and "Click the generate button!".

# 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 write 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

Group RE1 // CAD/CAM 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]
<input type="text"/>	<input type="text" value="10"/>
Select atmospheric temperature (deg C):	[-15:50]
<input type="text"/>	<input type="text" value="-15"/>
Select continuous runtime (hours):	[2,4,6,8,10,12]
<input type="text"/>	<input type="text" value="2"/>
Enter desired operational speed of airship in calm winds (m/s):	[3:10]
<input type="text"/>	<input type="text" value="3"/>
Enter maximum payload length within cargo bay (cm):	[12:50]
<input type="text"/>	<input type="text" value="12"/>
Enter payload mass (g):	[500:1000]
<input type="text"/>	<input type="text" value="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.
```

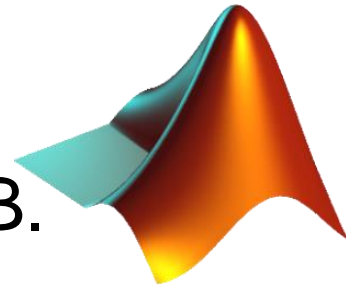
C:\groupRE1\Log\groupRE1\_LOG.TXT

**GENERATE DESIGN**

**FINISH AND CLOSE INTERFACE**

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/debugging-process-and-features.html)

[https://www.mathworks.com/help/matlab/matlab\\_prog/set-breakpoints.html](https://www.mathworks.com/help/matlab/matlab_prog/set-breakpoints.html)

[https://www.mathworks.com/help/matlab/matlab\\_prog/examine-values.html](https://www.mathworks.com/help/matlab/matlab_prog/examine-values.html)