Aircraft Generation Code

REMOTELY DEPLOYED AND RECOVERED UAV

GDP #33

Contents

1. Introduction

The Remotely Deployed and Recovered UAV (RDUAV) Aircraft Generation Code (AGC) serves primarily to analyse different configurations of aircraft generated for the University of Southampton (UoS) entry into the 2018 American Institute of Aeronautics and Astronautics (AIAA) 'Design Build Fly' (DBF) competition. It includes a complete mass model of our aircraft within a heavily parametric Solidworks assembly that allows for reading aircraft mass, centre of mass (CoM) and moments of inertia. The dimensions of this model are informed by user input for driving dimensions and by statistical, structural and aerodynamic sizing in the Matlab code.

The ultimate goal of the code is to generate the best aircraft designs for scoring highly in the competition, taking into consideration the impact of this on structural design, aerodynamic performance and feasibility of manufacture.

Our competition entry is made jointly with the Arizona State University (ASU) Air Devils team under the Air Devils International (ADI) name, and significant aspects of this code have been informed by their analysis, physical testing and data. We have used data from ASU's initial trade study for a previous competition entry data to verify results generated by this code. Their physical propulsion test data and flight testing have also served to heavily influence design choices.

If you are developing the code, please refer to Appendix A for best practice before making any commits.

2. Operation

In figure 1 below a flowchart of code operation is given. The remainder of Section 2 gives a more detailed explanation of its operation. Section 3 contains further information on each of the key functions.

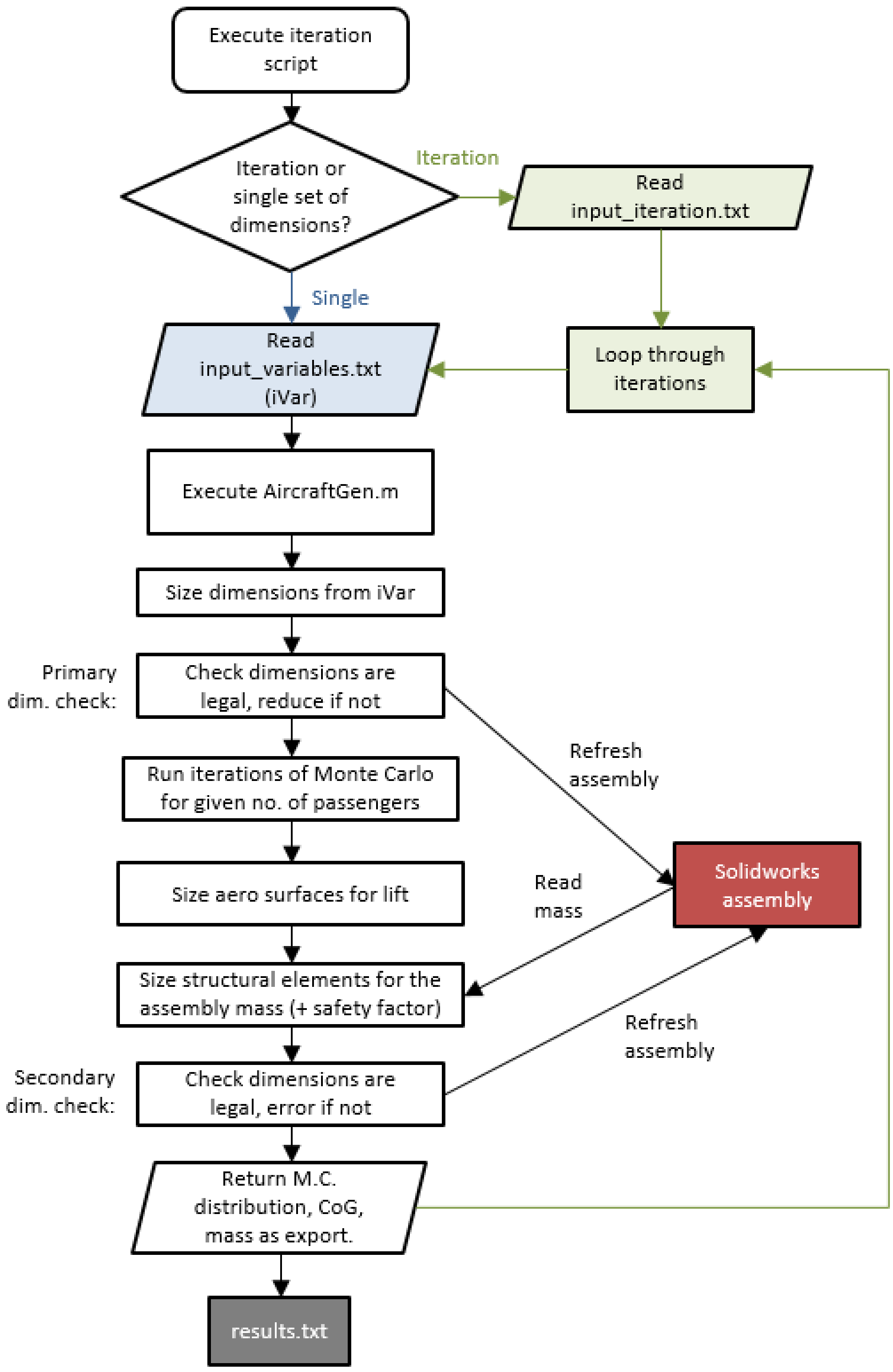
Execute iteration script
Iteration or single set of dimensions?
Iteration
Read input_iteration.txt
Single
Read input_variables.txt (iVar)
Loop through iterations
Execute AircraftGen.m
Size dimensions from iVar
Primary dim. check:
Check dimensions are legal, reduce if not
Refresh assembly
Run iterations of Monte Carlo for given no. of passengers
Size aero surfaces for lift
Read mass
Solidworks assembly
Size structural elements for the assembly mass (+ safety factor)
Refresh assembly
Secondary dim. check:
Check dimensions are legal, error if not
Return M.C. distribution, CoG, mass as export.
results.txt

Figure 1. Operational Flowchart

2.1. Input variables

First, open *input_variables.txt* and input your desired initial dimensions. Values should always be delimited from variable names on the left via tab after the equals sign, and value names should not be altered aside from when developing the code. Dimensions are included in the variable names so should not be included in the value input after the tab as this will disrupt the code's ability to read the input as a numerical value. Comments are made with '%' and will be ignored by the code when reading inputs.

The format given for input variables is taken from the Solidworks equation conventions, as this is the method utilised for updating dimensions within assemblies from Matlab.

Below the subassembly dimensions some dimensions headed by the comment `'% Dimensions defined by sizing equations - no user input!'` These dimensions are updated within the code and remain in the input file for convenience. Do not edit these values as your input will be overwritten and bad initial values could disrupt code operation.

2.2. Input iterations

The iteration file *input_iteration.txt* can be used to generate multiple designs by iterating over specific input variables. These are given in the same format as the *input_variables.txt* file, with the addition of increments and final values in the format of 'for loops' in Matlab. For example:

```
"controlNumPassengerRows"=    10:1:12

"wingRootChord_Length_mm"=    300:100:600
```

These input iterations will generate a total of 12 different aircraft configurations, with three different fuselage lengths (driven by the number of passenger rows input) and four different Aspect Ratios (driven by root chord length input, and wingspan from aerodynamic sizing) for each of those fuselages.

The varying number of passengers will also lead to different mean values and distributions form the Monte Carlo for each of the three fuselages. More information on this can be found in Section 3.3.

2.3. Code operation

To run your inputs (iterations or otherwise) through the code, simply execute the *run.m* script in the Mk1b/matlab directory. You will need to change the userpath string at the top of the code to the location of your Mk1b folder. The code might fail to find some files at first, disrupting operation, in which case please refer to Section 5.

The code will then ask you to input 1, 2 or 3 in the command window for GUI, textfile or iterations respectively. The GUI is currently still in development, so simply select 2 to run your design in *input_variables.txt*, or select 3 to run that design with iterations on specific variables from *input_iteration.txt*.

Assuming no iterations (single), the code will first read the user defined input variables form *input_variables.txt*. These inputs will then be passed from the run script to the *AircraftGen.m* function which will carry out the rest of the code operation. If option 3 is selected then an intermediary function *IterLoop.m* is used to run *AircraftGen.m* multiple times for every design iteration.

The input dimensions are then used to size input-driven dimensions (e.g. fuselage passenger bay length driven by the number of passengers). The Solidworks assembly is refreshed using the ActiveX functionality with the new dimensions to generate an aircraft mass and CoM.

A mean total passenger mass and CoM is generated using the Monte Carlo function, and this is then combined with the aircraft ad payload mass and CoM to give total values. These values are used to size the spars and booms of the aircraft, in addition to the lifting surfaces.

If these dimensions are illegal, they cannot be changed without risking the structural integrity of the design. Therefore an error is returned as the design is invalid. If the final dimensions are legal, the assembly is again refreshed with the new structural component dimensions.

The key results are then exported to a results.txt file, along with the input variables for identification.

If running multiple iterations, the previous steps will be repeated for the next design iteration in the list via *IterLoop.m*.

3. Features

Features.

3.1. Solidworks ActiveX link

The Matlab code links directly with the open

3.2. Dimension Checking

Explain how this works.

3.3. Passenger Allocation through Monte Carlo

Explain how this works.

3.4. Aerodynamic Analysis

Explain how this works. Include LGF stuff.

3.5. Structural Analysis

Explain how this works. Includes secondary dimension checking.

4. Output and Data Plotting

Explain how this works.

5. FAQ and Debugging

Before you start debugging, make sure you have read this Readme in its entirety. There is key information within the text that is not repeated here that will probably answer your problem if you are trying to use the code before reading it.

The list of known bugs below is by no means complete. If you still have issues feel free to ask me.

Code unable to find file

This is likely due to Solidworks failing to initialise the path with all the relevent directories. Check to make sure your userpath is correct (it should be from the top level of your drive, i.e. *'C:/ ... /Mk1b'*). If the problem persists after multiple run attempts simply use the Matlab folder tree on the left to manually add the *Mk1b* files and subfolders to the path by right-clicking *Mk1b* and clicking *'Add to Path > Selected Folders and Subfolders'*.

Issues with importing dimensions into Solidworks assembly (not updating completely or giving errors in Matlab)

Check the Solidworks equation files (the output files) are properly tab delimited, and there are no exotic UNICODE characters in the code that shouldn't be there (e.g. an arrow character at the beginning of the file). Removing these characters and properly delimiting the output should fix this bug.

Equations within Solidworks returning errors

Check the linked equation file is correct. It should be in the same directory as the subassembly.

Strange mass properties being returned

Check all other Solidworks windows aside from the master aircraft assembly are closed. ActiveX links to whatever Solidworks window is currently open, so other open windows could potentially interfere with this.

Appendix

A. Best Practice for Coding

All variables and functions should be named using ‘upper CamelCase’ (or ‘PascalCase’), with all compound words are capitalised, unless specified otherwise. For example:

For functions:	ReadDimensions(filepath, filename, format)
For variables:	TotalLift

Exceptions to this rule are as below.

A.1. Dimension and unit variables, strings

All dimensions and unit variables (including when referred to in strings) are to be written in the following format.

[subassembly][DimensionName]_[DimensionType]_[unit]

For example,

wingSpan_Length_mm

wingMainSpar_InnerDiameter_mm

A.2. Other exceptions

Some variables break the rules set out above due to the frequency of their use in the code, to allow them to stand out, or because their names are so short as to make the standard practice redundant. These variables should be left named as they are unless it is practical to fit them to the standard.

B. Best Practice and Structure for Folders and Files

B.1. Folders

The project folders are organised as follows. All folder names are given in total lowercase without spaces or underscores, regardless of the number of words in the folder title. This is to encourage single-word folder titles.

- **Mk1a:** The aircraft version, given by a number, with its corresponding sub-version, given as a letter.
    - **cad:** All CAD files and folders associated with the given version of the aircraft assembly.
        - **fuselage**
        - **motor**
        - **tail**
        - **wing**
        - **old**
    - **matlab:** All Matlab files and folders, the run script and the ‘input_variables.txt’ and ‘input_iteration.txt’ text files.
        - **fileio:** All functions associated with file input and output.

- **interface:** All functions associated with user and Solidworks interface (except where involving file I/O).
- **ops:** Functions integral to the operation of the software that call all other functions aside from the run script.
- **results:** All results and data generated by the software should be output here.
- **tools:** Tools used to alter the Solidworks assembly, or run any relevant calculations.
- **unused:** Functions and scripts not currently used in the software that may be useful in future.

The CAD folders each contain their respective subassembly and parts, with a Solidworks equation file titled as '[assembly]_equations.txt'. Each folder also contains an 'old' folder, for storing previous versions of parts. The top level CAD folder contains the master assembly, with its own equation file. The only parts included at the top level should be those that connect multiple subassemblies (such as the spars and booms).

B.2. Files

Matlab functions are not versioned as this is handled through Git, however if a working function is removed from the software it should be moved to the 'unused' folder. All functions should be named as they are in the code (e.g. ReadDimensions.m for the function ReadDimensions).

CAD files should be titled in the following 'lower CamelCase' format.

[subassembly][PartName]_v[x.y]

With x corresponding to the aircraft version and y corresponding to the part version. The aircraft version does not include the letter sub-version. For example:

wingAirfoilRib_v1.1

See Appendix A for how to title dimensions corresponding to each part.

Assemblies should be simply titled as follows.

[subassembly]Assem_v[x]

With x defined as before, but with the aircraft sub-version. For example:

masterAssem_v1a