Aircraft Flaps Modeling in OpenMDAO

Kiran J. Marfatia[1] and Jennifer D. Bergeson[2]
NASA Glenn Research Center, Cleveland, Ohio, 44135

The goal of this project was to develop a model for a single subsystem in the aerodynamics discipline, in this case the flaps of an aircraft. Flaps are high-lift devices used by planes to allow for quicker takeoffs, and slower landings. A computer-based aircraft model of the flaps of an aircraft was developed using the Python based open-source framework OpenMDAO. OpenMDAO is used to develop multi-disciplinary aircraft models using gradient-based optimization; design optimization (MDO) is concerned with solving design problems involving numerical models of complex engineering systems. There were 4 components in the model; each has input values, output variables, and equations to calculate said outputs. The variables and equations are sourced from NASA Fortran code from the 1970s, in a project called the General Aviation Synthesis Program (GASP). These variables and equations which create the model are being converted to Python for ease of use. The flaps model developed will be integrated into a larger model of a conventional aircraft's flight phases. All subsystems of the model will first be built using the parameters of a Boeing 737 MAX-8, to validate its functionality and accuracy. Then, the aircraft model will be used for hybrid-electric research; running optimizations to improve efficiency, minimize fuel burn, and advance hybrid-electric technology in the aerospace field.

Nomenclature

AR	=	aspect ratio
avg_chord	=	average aerodynamic chord of wing, ft
BLEOB	=	ratio of leading edge slat span to wingspan
BTEOB	=	trailing edge flap span divided by wingspan
cabin_width	=	width of the fuselage, ft
C_D	=	drag coefficient
center_chord	=	wing chord at fuselage centerline, ft
CFOC	=	ratio of flap chord to wing chord
C_L	=	lift coefficient
CLEOC	=	ratio of leading edge device chord to wing chord
C_{L_max}	=	maximum lift coefficient
CROBL	=	ratio of root chord to fuselage length
D	=	drag, lb
DCD	=	increment on drag coefficient
DCDOTE	=	drag coefficient increment due to trailing edge flap
DCL	=	increment on lift coefficient
DCLMLE	=	input leading edge slat lift increment for leading edge type at optimum leading edge deflection
DCLMTE	=	lift coefficient increment due to trailing edge flap
DELFD	=	flap deflection, deg
DELLED	=	leading edge slat deflection, deg
DELLEO	=	optimum slat deflection angle, deg
fus_len	=	length of entire fuselage, ft
FWOB	=	ratio of body diameter a quarter chord to wingspan
GASP	=	General Aviation Synthesis Program
L	=	lift, lb
l	=	characteristic linear dimension
P	=	static pressure, lb/ft^2

[1]NASA LERCIP Summer Intern, Propulsion Systems Analysis Branch (LTA), Glenn Research Center, Hackley School. Student Member AIAA.
[2]AST, PROPULSION SYSTEMS & TECHNOLOGIES, Propulsion Systems Analysis Branch, 21000 Brookpark Road, Cleveland, OH/5-11, Glenn Research Center

ρ	=	density, lb/ft^3
q	=	dynamic pressure, lb/ft^2
RCLMAX	=	input reference maximum lift coefficient for basic wing
RDELF	=	ratio of flap deflection to optimum flap deflection angle
RDELL	=	ratio of leading edge slat deflection to optimum deflection angle
RNW	=	Reynolds number
root_chord	=	chord length at wing root, ft
S	=	wing area, ft^2
SA	=	speed of sound at sea level, ft/s
SMN	=	Mach number
sweep_c4	=	sweep angle of wing at quarter chord, deg
sweep_LE	=	sweep angle of the leading edge of the wing, rad
T	=	temperature of air cross wing, Rankine
taper_ratio	=	taper ratio of the wing
tc_ratio_avg	=	average wing thickness to chord ratio
tc_ratio_root	=	thickness to chord ratio at the root of the wing
v	=	object velocity
VDEL	=	minimum drag coefficient sensitivities
VLAM	=	maximum lift coefficient sensitivities
wing_loading	=	load force on the wing, lb/ft^2
wingspan	=	distance from left wingtip to right wingtip, ft
XKV	=	kinematic viscosity, ft^2/s

I. Introduction

THE main goal of this project was to develop a model of one subsystem of an aircraft, and to work with Propulsion Systems Analysis engineers to understand the integration of that subsystem into a larger aircraft model. The main project will be integrated into a larger model to calculate aircraft weight, lift, and drag, along with other key aerodynamic aircraft characteristics of a conventional aircraft.

The way the model was developed was using the important method of computer-based aircraft modeling, which has many uses in the aerospace field. Computer developed models are used for research and design purposes with respect to aerospace engineering. Many significant aerospace companies use computer-based aircraft models to run simulations during their research and design phases of developing new technology for air and space crafts. Aerospace engineering is crucial for technology and industry in space flight and aviation. It provides safe and faster, and more fuel-efficient means of flight, space exploration, and more. This project, although small, embodies those goals.

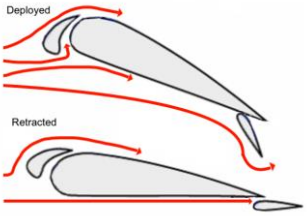

Figure 1: Wing airflow: flaps deployed and retracted.

The subsystem modeled in this project is the flaps of an aircraft, which includes both the flaps and slats on a wing. Looking at Figure 2, the flaps can be seen at the rear of the wing. Outside of the image, the slats are deployed in the front. Flaps and slats are an important part of the wing of an aircraft, as they provide extra lift and drag for the takeoff phase, allowing for a plane to get off the ground quickly, and for the landing phase, to allow for a slower and smoother touchdown. As shown in Figure 1, the airflow of the airfoil changes based on whether the flaps are deployed or not, providing more lift and drag when deployed, and less when retracted.

Figure 2: Flaps on Airbus A321

OpenMDAO was used to develop a model of the flaps of an aircraft, for integration into a bigger model which will be used for research purposes. OpenMDAO is a Python-based open source framework used to develop multi-disciplinary aircraft models, primarily using gradient-based optimization. OpenMDAO was developed in house by NASA and is used for research and design purposes today. Multidisciplinary design optimization (MDO) is concerned with solving design problems involving numerical models of complex engineering systems. OpenMDAO is an open-source MDO framework that uses Newton type algorithms to solve coupled systems and exploits problem structure through new hierarchical strategies to achieve high computational

efficiency [5]. For this project, many different features are used to develop an accurate model, including MetaModel Structured Components, Newton Solvers, and more.

The equations and variables used in the whole model are taken from old NASA Fortran code from the 1970s. These are taken from code developed in a program called the General Aviation Synthesis Program (GASP). The outdated code is being ported into Python and OpenMDAO for ease of use in this new model. This new model is a in a project called GASPy which is the General Aviation Synthesis Program with Python. This is the overall model that the flaps model will be integrated into. GASPy is being developed for the overall purpose of doing hybrid electric research in the future.

II. Flaps Model Development

The project was completed in various Python files using the text editor application Visual Studio Code. In the different files, different elements of the project are written. There is the top-level model, with all the components, equations, and variables from GASP in one large file. There are also many test files used to validate the model to make sure it is running smoothly and accurately. These files were regularly uploaded to GitLab, a version control software, to make debugging and completing the model easier.

A. Project Background

1. GASPy Project Overview

The basis for the model being developed all comes from a 1970s NASA program called GASP. GASP is a vehicle synthesis program used for conceptual design studies of general aircraft configurations. It was developed by the Systems Analysis Branch at NASA in the 1970s, and even further developed in the 1980s and 1990s by Georgia Tech. It uses engineering level analysis across all technical disciplines to predict vehicle performance, estimate subsystem element weights and close the configuration to meet mission requirements and design constraints[2]. The code was written in the then prominent computer programming language Fortran, using geometric parameters (input variables and computed values), along with fixed weights, and technology factors. However, the Fortran is difficult to understand and work with, and it is outdated for the research and engineering done in the aerospace field today. Therefore, the old GASP code is being converted into Python for a project called GASPy (General Aviation Synthesis Program with Python). For this project, geometric parameters and equations from GASP are used to both develop the model and test the model to validate the data and the model's functionality.

The overall flaps model is important, but it is not very helpful on its own. The overall goal for this project was to develop a single functioning subsystem in the aerodynamics discipline. The flaps model was built with the intentions of being integrated into a larger model. The project assisted is creating a multi-disciplinary model of a conventional aircraft's flight phases. The four disciplines are propulsion, trajectory, aerodynamics, and weight. Other engineers in the same project have worked hard to complete the rest of the model. Similar to the flaps model, the entire aircraft uses parameters and variables from GASP. The new model, converted to Python, is called GASPy, and this is the project that is being assisted.

The overall model will be first developed with the parameters of a Boeing 737 MAX-8 aircraft, to test its functionality. Once that is completed, the model will be converted to a hybrid-electric model, which will be used to run optimizations to improve efficiency, minimize fuel burn, and advance hybrid-electric technology in the aerospace field. Advancing hybrid-electric technology is a crucial step in advancing aerospace design, and for the planes of the future. The model will be put to good use helping design technology used by many future generations.

2. Details of OpenMDAO

OpenMDAO utilizes an object-oriented programming standard in addition to an object composition design pattern. Individual functions through narrowly focused classes are connected to achieve the desired functionality while running a model. The four most fundamental types of classes in OpenMDAO are Component, Group, Driver, and Problem. The classes used in the flaps model are the Component, Group and Problem.

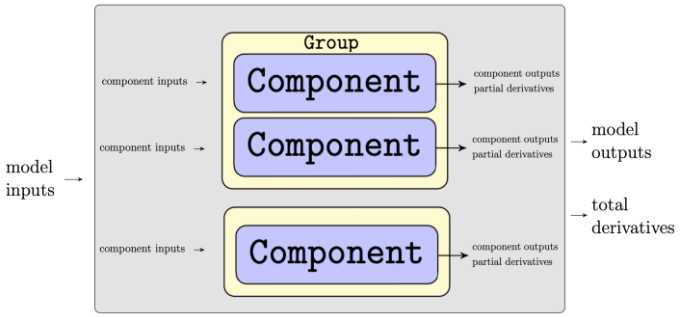

Figure 3: Relationship between Group and Component classes.

The first class, at the top of the hierarchy, in OpenMDAO is the Group, which can consist of other groups, components, or a combination of the two. The relationships between multiple groups and components create a hierarchy, with a top-level group that contains components and other groups, and the bottom end of the hierarchy containing only components. The main purposes of the groups are to group components together for better organization and to enable the use of hierarchical linear and non-linear solvers [5].

Components are generalized and can serve as a whole discipline analysis or perform a smaller set of calculations which embody a smaller part of a whole discipline model. The components also share a common interface, permitting them to be integrated to create a bigger model. A component provides the lowest level of functionality by completing basic algebraic calculations. Each component maps input values to output values by a calculation, initiated by an equation [5]. Figure 3 shows this, taking model inputs, component inputs, putting them through a calculation in the component, and producing component outputs which then lead to model outputs. These are examples of explicit components, simply computing explicit functions. This is one of two types of components used in the flaps model.

Finally, the Problem class is used as a top-level containment tool to hold other objects and to provide a user interface to set up and run the model [5]. Groups, components, and the problem is used in the flaps model and will be discussed later in the paper.

3. Aerodynamics & wing design

This section will explain various aerodynamics terms, which in turn will explain all the variables in the flaps model, as seen in Fig. 5 and defined in the nomenclature section. Some key terms will be defined in this section. First, the leading and trailing edges of an airfoil, or wing, are the front and back edges of said wing, as shown in Fig. 4. Next, the chord of an aerodynamic object such as a flap or a wing is determined by the distance between the leading and trailing edges, in the direction of the airflow. This is shown by the chord line in Fig. 4. When the flaps and slats are deployed, as shown in Fig. 1, pivoting the leading edge of the slat and the trailing edge of the flap downward increases the camber of the airfoil, increasing lift [4]. The chord line cuts an airfoil into an upper and lower surface. The mean camber line is found when the points that lie halfway between the upper and lower surfaces are plotted. The greatest distance between the chord line and mean camber line is called the camber. Camber is a measure of the curvature of the airfoil; the curvature and camber are directly proportional. The biggest distance between the upper and lower surfaces of the wing is the thickness. The span of the wing is the distance from one wingtip to the other. Additionally, the root of the wing is the part of said wing that is closest to the fuselage. The aspect ratio measures the length and slenderness of a wing, and is calculated with Equation 1, with S representing wing area (only the top surface area) [10]:

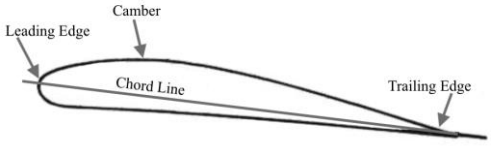

Fig 4: Cross-sectional view of an airfoil.

$$AR = \text{wingspan}^2/S \; . \tag{1}$$

Next, there are the lift and drag, and their coefficients, which are variables that are extremely important in any aerodynamic equation or model. The lift and drag coefficients are dimensionless quantities used to model complex dependencies of shape, inclination, and some flow conditions on lift and drag, and to quantify that force on an object [6]. The maximum lift coefficient is also important, and different parts of the wing, such as the flap, can have its own maximum lift coefficient. Dynamic pressure, defined as ‘q’, is a defined property of a moving flow of gas, and it is defined below, where ρ is density and ν is velocity of the airflow [3]. The equations for dynamic pressure, lift, drag, and their coefficients are:

$$q = \tfrac{1}{2} * \rho * \nu^2 \tag{2}$$
$$L = C_L * q * S \tag{3}$$
$$D = C_D * q * S \tag{4}$$
$$C_L = L / (q * S) \tag{5}$$
$$C_D = D / (q * S) \tag{6}$$

Mach and Reynolds numbers are two variables used heavily in flight. The Mach number of an aircraft represents its speed relative to the speed of sound. Mach number is important because air density changes depending on how high the Mach number gets. At speeds less than Mach 1.0, an aircraft is considered subsonic. All commercial passenger flights (except Concorde), travel at subsonic speeds. Speeds greater than Mach 1.0 means an aircraft is traveling faster than the speed of sound and is at ‘supersonic’ speeds. Any speed greater than Mach 5.0 is considered ‘hypersonic’. In

addition to Mach number, another important measurement is the Reynolds number. The Reynolds number expresses the ratio of inertial (resistant to change or motion) forces to viscous (heavy and gluey) forces [9]. Represented by their OpenMDAO variable names SMN and RNW, the equations for Mach and Reynolds numbers are shown below, where l is the characteristic linear dimension:

$$\text{Mach Number} = \frac{object\ speed}{speed\ of\ sound} \tag{7}$$

$$\text{Reynolds Number} = \frac{\rho * v * l}{XKV} \quad . \tag{8}$$

B. Top-Level Model

The top-level model is the most important part of the project. This program includes all the different groups and components with their aerodynamics equations and variables used in creating the model. There are three explicit components, and one group with MetaModel components, which will be discussed later. Each has its own inputs, outputs, intermediate values, and equations. There was also a section in the code that connects the four and runs the model as a whole, printing any outputs wished.

4. Top-level Group

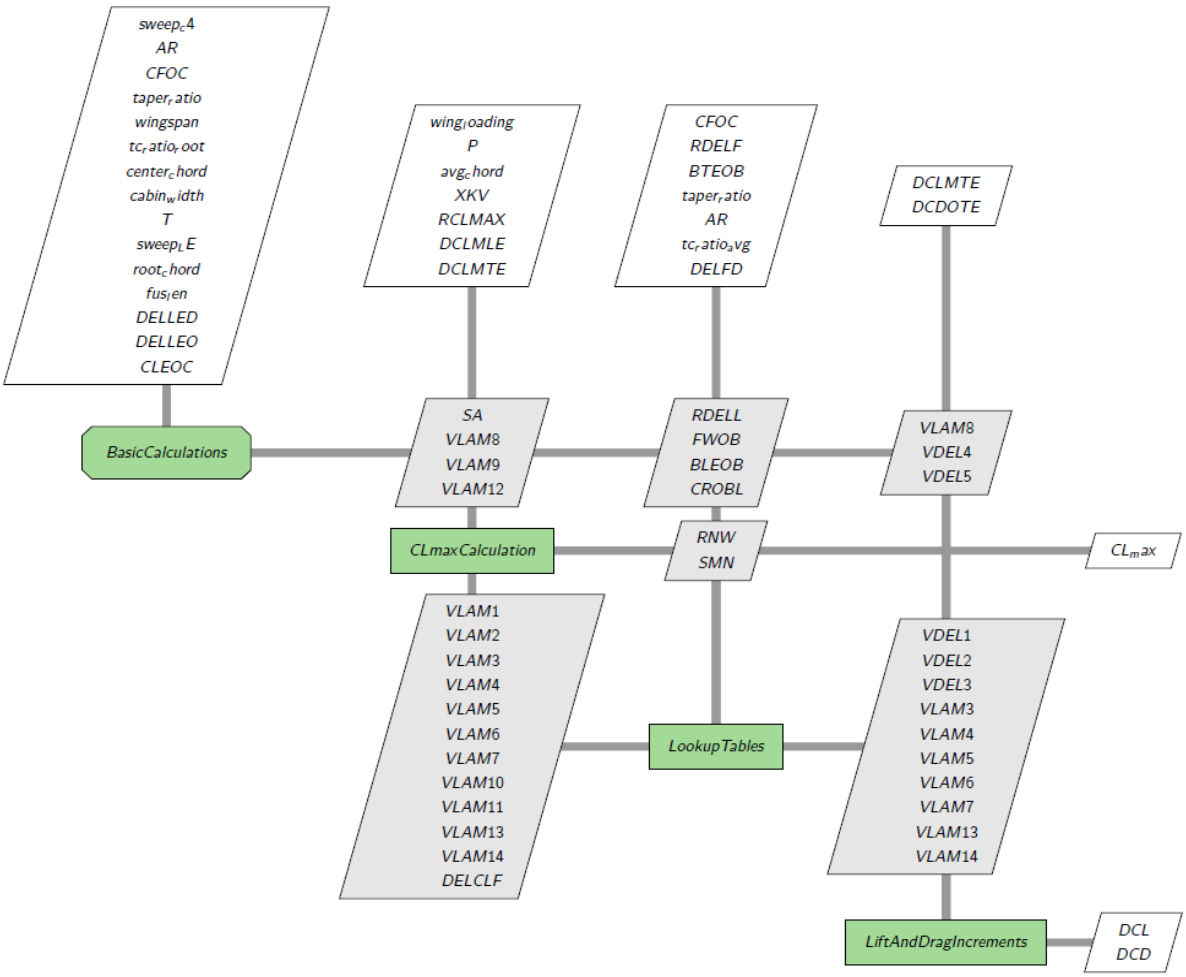

Figure 5. Extended design structure matrix (XDSM) for the flaps model

As shown in Fig. 5, the four classes are called *BasicCalculations*, *CLmaxCalculation*, *LookupTables*, and *LiftAndDragIncrements*. *LookupTables* is a group of MetaModel components, with the other three being explicit

components. They are all connected, and in the model, are run in the order shown in the diagram. In Fig. 5, all values aligned vertically with a component are inputs to that component, and all values aligned horizontally to a component are outputs of it. Each component has many inputs and outputs. If the box is white, then those inputs are only inputs, meaning that they are not connected to anything else. Same goes for outputs, meaning that output is final, and not connected to another component. Each gray box means those variables are connected to multiple components. For example, variables *RNW* and *SMN* are outputs of *CLmaxCalculation* and inputs of *LookupTables.* Other outputs that do not connect to anything (*CL_max, DCL,* and *DCD*) are final outputs.

5. Sensitivity Values

Variable Name	Description
VLAM1	sensitivity of clean wing maximum lift coefficient to aspect ratio
VLAM2	sensitivity of clean wing maximum lift coefficient to wing thickness to chord ratio
VLAM3	sensitivity of flap clean wing maximum lift coefficient to aspect ratio
VLAM4	sensitivity of flap clean wing maximum lift coefficient slope to wing thickness
VLAM5	sensitivity of flap clean wing maximum lift coefficient to wing flap to chord ratio
VLAM6	sensitivity of flap clean wing maximum lift coefficient to wing flap deflection
VLAM7	sensitivity of flap clean wing maximum lift coefficient to wing flap span
VLAM8	Sensitivity of flap clean wing maximum lift coefficient to wing sweep angle
VLAM9	Sensitivity of slat clean wing maximum lift coefficient to slat chord
VLAM10	sensitivity of clean wing maximum lift coefficient to slat deflection angle
VLAM11	sensitivity of slat clean wing maximum lift coefficient to slat span
VLAM12	Sensitivity of slat clean wing maximum lift coefficient to leading edge sweepback
VLAM13	Reynolds number correction factor
VLAM14	Mach number correction factor
VDEL1	sensitivity of flap minimum drag coefficient to flap chord ratio
VDEL2	sensitivity of flap minimum drag coefficient to flap angle
VDEL3	sensitivity of flap minimum drag coefficient to partial flap span
VDEL4	Sensitivity of flap minimum drag coefficient to flap hinge line sweep
VDEL5	Sensitivity of minimum drag coefficient to fuselage width to span ratio

Table 1: Variable descriptions

In the model, there are many sensitivity variables, defined as VLAM or VDEL. Sensitivity in this context is discussing how sensitive one value is to another. For example with VLAM1, how sensitive the clean wing maximum lift coefficient is to the aspect ratio. The value is based on how much the clean wing maximum lift coefficient changes when the aspect ratio changes. These are important values used to calculate many different variables across the model. The definition for all VLAM and VDEL variables are written above in Table 1.

6. BasicCalculations

In the model, Basic Calculations is the first of the four components in the top-level model. In OpenMDAO, this component is an explicit component, a class that simply maps inputs to outputs using calculations and equations [5]. As the name suggests, this component does many simple calculations with the inputs and their values given from the old GASP Fortran files.

As shown in Fig. 5, there are 15 input values for that component. The input values are for an multitude of different geometric measurements for the wing of the aircraft. Some of the variable names for this model in OpenMDAO have been changed from their Fortran names for ease of use. Fortran variable names were limited in the number of characters and therefore were not always representative of the purpose of said variable. Each input was added into the component in the 'setup' section of the component, using the 'self.add_input' command. This command for adding an input includes the name of the variable, its value, units, and a short description of the input.

Similarly, in the same 'setup' section, all the outputs were added using the 'self.add_output' command, including the same information as its input counterpart. There were 10 total outputs, which included different values as shown by the italicized variables in table 1. Other than *SA*, all variables were either sensitivities or ratios of certain measurements of the wing and/or their lift or drag coefficients.

After the 'setup' section of the component comes the 'setup_partials' section. This is where partial derivatives are declared for each component. All outputs which depend on an input of the component (this specifically excludes any intermediate equations in a component) have a partial declared using the 'self.declare_partials' command. In OpenMDAO, partial derivatives are the the derivatives of outputs of each component with reference to the component

inputs. Regarding an explicit component, used when outputs can be computed as an analytic function of the inputs, the partial derivatives are the derivatives of the outputs with respect to the component inputs [5].

The final section in this component is the 'compute' section. This is where the outputs are calculated by giving them equations. In the setup section, outputs were added into the component, but not defined. In addition to the equations calculating the outputs, there are intermediate equations as well to help with this. The equations for the intermediate values are as follows,

```
RLMC4 =  sweep_c4 * 0.017453                                                    (9)
TSWPFH = (tan(RLMC4))-(4.0/AR)*((.75-CFOC)*(1.-taper_ratio)/(1.+taper_ratio))   (10)
SWPFHL = arctan(TSWPFH)                                                         (11)
DBALE = (2.* (tc_ratio_root*center_chord*(cabin_width(tc_ratio_root*center_chord)))**0.5 ) + 0.4   (12)
SWPL12 = sweep_LE-5./57.296 .                                                   (13)
```

RLMC4 is the sweep angle of the wing quarter chord in Radians, TSWPFH is the tangent of the sweep angle of the flap hinge line, DBALE is the fuselage body diameter at leading edge, and SWPL12 is the correction on the change in maximum lift coefficient from slats based on the ratio of the span of the leading edge. Finally, the equations for the output variables of *BasicCalculations* are:

```
VLAM8 = (cos(RLMC4))**3                  (14)
SA =  49.1 * (T**0.5)                    (15)
FWOB = DBALE/wingspan                    (16)
VDEL4 = cos(SWPFHL)                      (17)
VDEL5 = 1.0 – FWOB                       (18)
VLAM9 = 6.65*CLEOC                       (19)
RDELL = DELLED/DELLEO                    (20)
BLEOB = 0.99 - DBALE/wingspan            (21)
CROBL = root_chord/fus_len               (22)
VLAM12 = (cos(SWPL12))**3                (23)
```

7. Component 2: CLmaxCalculation

This is the second component, and it is another explicit one in OpenMDAO. This one is extremely similar in structure to *BasicCalculations*; the slight difference is that some of its input variables are outputs of other components as well. First, *CLmaxCalculation* has 4 inputs from *BasicCalculations*, which are the speed of sound, and 3 VLAM values, which are maximum lift coefficient sensitivities to other measurements. In addition to these, Fig. 5 shows many other inputs, including air-related variables such as static pressure (P), geometric wing measurements, and many sensitivity values which are outputs of *LookupTables*. There are just three outputs from this component: Reynolds number, Mach number, and maximum lift coefficient (hence the name of the component).

All of the inputs from other components are added into the component in the same way as any other; and they were given values from the GASP Fortran code. What is interesting is how this component is connected to the *LookupTables* group, as shown in Fig. 5. There is also a loop of inputs and outputs. Outputs of *CLmaxCalculation* rely on inputs from *LookupTables*, which rely on those same other outputs. Partial derivatives are also declared for all three outputs. The outputs calculated in this component use these equations:

```
CL_max = (RCLMAX*VLAM1*VLAM2+DCLMTE*VLAM3*VLAM4*VLAM5*VLAM6*VLAM7*VLAM8+
          DCLMLE*VLAM9*VLAM10*VLAM11*VLAM12)*VLAM13*VLAM14+DELCLF   (24)
SMN = ((wing_loading/CL_max)/0.7/P)**0.5                            (25)
RNW = (avg_chord*(SMN*SA)/XKV)/100000                               (26)
```

The equations for Mach and Reynolds number are different than the previous equations shown because they use the different OpenMDAO variables. There are only 3 outputs from this component, but three very important ones. CL_max, the maximum lift coefficient, is a final output and not used as an input to any other class, as shown in Fig. 5. As a final output, it will be used for integration into the overall GASPy model. This is an extremely important

measurement to have in aerodynamics because the maximum lift coefficient is the lift coefficient of the place on the lift curve right before the wing stalls, therefore it is used to calculate the stall speed. The other two outputs are SMN, which is the Mach number, and RNW, the Reynolds number. These two values are also crucial to have in this model, because many of the sensitivity factors depend on them. They are outputs of *CLmaxCalculation*, and inputs to *LookupTables*.

8. LookupTables: MetaModel Group

As shown in Fig. 5, *LookupTables* is a group in the OpenMDAO hierarchy, and it contains multiple components. Looking back to Fig. 5 *LookupTables* takes many input values and creates a 'table' of many different sensitivity values, providing inputs to be used in multiple other components. *LookupTables* contains MetaModel Structured Components instead of explicit components used in the rest of the model. Instead of using basic algebra and equations to solve, these components use linear interpolation.

Looking through the group, first, in the 'initialize' section of this class, an option is declared. The reason for this is because for the outputs in this specific component, the training data changes based on flap type. Therefore, an option for various flap types needs to be created so that different training data can be used for various outputs. There are 7 different options for various flap types. The types are plain, split, single slotted, double slotted, triple slotted, fowler, and double slotted fowler. Fig. 6 shows each of the flaps, each with their own pros and cons. For the slotted flap type, a double or triple slotted flap is the same style, but with an extra piece (or two) extending downward. For the double slotted fowler flap, it is again the same style as the standard one, but with an extra piece.

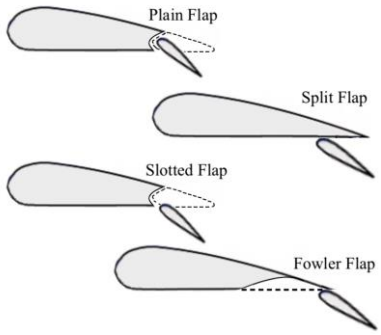

Figure 6: Flap types.

The way the option is executed is through using Python's 'if-else' statements. For example, regarding the output VLAM5, and as shown in Fig. 7. As seen in the figure, there are 3 different sets of training data for the output VLAM5, depending on the flap type. In this case, there are three different categories that the flap types are put into. The first: plain or split, the second: single, double, or triple slotted, and the third: fowler and double slotted fowler. This last group is named as 'else' because the flap types in that group are the only ones left. In the if-else statement, depending on flap type, a different 'add_output' command is run, with different training data. Now, what must be looked at is how the training data and interpolation method works.

```
#VLAM5
VLAM5_interp = self.add_subsystem('VLAM5_intep', om.MetaModelStructuredComp(method='scipy_slinear'),
                                  promotes_inputs=['CFOC', ],
                                  promotes_outputs=['VLAM5'])

VLAM5_interp.add_input('CFOC', 0.3, training_data=[0.,.1,.2,.3,.4,.5],
units=None, desc="ratio of flap chord to wing chord")

if flap_type == 'plain' or flap_type == 'split':

    VLAM5_interp.add_output('VLAM5', 1.0, training_data=[0.0,0.72,0.94,1.00,0.95,0.73],
    units=None, desc="sensitivity of flap clean wing maximum lift coefficient to wing flap to chord ratio")

elif flap_type == 'single_slotted' or flap_type == 'double_slotted' or flap_type == 'triple_slotted':

    VLAM5_interp.add_output('VLAM5', 1.0, training_data=[0.,.575,.83,1.00,1.065,1.09],
    units=None, desc="sensitivity of flap clean wing maximum lift coefficient to wing flap to chord ratio")

else:

    VLAM5_interp.add_output('VLAM5', 1.0, training_data=[0.,.41,.73,1.00,1.22,1.40],
    units=None, desc="sensitivity of flap clean wing maximum lift coefficient to wing flap to chord ratio")
```

Figure 7: VLAM5 MetaModelStructuredComponent

In the group, there are multiple 'Meta Model Structured Components'. These are different to the ExplicitComponents of *BasicCalculations* and *CLmaxCalculation*. They use a set of provided training data values to

calculate for an output of best fit. As seen in Fig. 7, there is training data for both input and output values. This training data is from the GASP Fortran code, just as the input values are in the explicit components. The way the output of best fit is calculated is by using linear interpolation, a method of curve fitting using linear polynomials to construct new data points within the range of a discrete set of known data points. If the two known points are given in coordinate form (eg: (x_1, y_1) and (x_2, y_2)), the linear interpolant is the straight line between these two points [7]. The equation for linear interpolation solves for the value y in an equation of slopes,

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}. \tag{27}$$

where y is the variable being solved for. In the context of the flaps problem, it is quite similar. The example to show this will be the VLAM5 component. The output is VLAM5, and the input to it is CFOC. There are 6 different values in the array of training data for both the inputs and outputs. In this example, the output training data for flap types plain and split will be used. The way that the component knows to use the linear interpolation method is a command for when the component is added. At the end of the first line of code, this method is declared using syntax 'method=scipy_slinear', which is OpenMDAO's definition for linear interpolation. In this context, the x coordinates come from the input array. First, for the value x, a random value within the parameters of the array must be chosen, for example x = 0.25. Next, the values for x_1 and x_2 are the closest values to x in the array. In this case, $x_1 = 0.2$, and $x_2 = 0.3$. Looking at the array in Fig. 7, it can be deduced that 0.2 and 0.3 are the 3rd and 4th values in the input array. This shows us how to gather the y_1 and y_2 values in the equation. y_1 is the 3rd value in the output array, so 0.94, and then y_2 will equal the 4th value, 1.00. Then y is solved for:

$$\frac{y - 0.94}{0.25 - 0.2} = \frac{1.0 - 0.94}{0.3 - 0.2}. \tag{28}$$

Where y = 0.97. One disclaimer, the values given in the 'add_output' command are simply estimates (the 1.0 is just an estimate). This is how linear interpolation works in the MetaModelStructuredComponent. There are 15 of these components in the group.

9. LiftAndDragIncrements

The last and final component in the top-level model is *LiftAndDragIncrements*. This is an explicit component which calculates exactly what it is named for. For input values, this component uses input variables from two locations: the GASP Fortran and from outputs of *BasicCalculations* and *LookupTables*. The output values from other components are sensitivity values for both lift and drag coefficients. The other input variables, DCDOTE and DCLMTE, are lift and drag coefficient increments specifically caused by a trailing edge flap. Derivatives are declared, then outputs are computed.

The two outputs of LiftAndDragIncrements are DCL and DCD. The equations for these variables are:

$$\text{DCD} = \text{DCDOTE*VDEL1*VDEL2*VDEL3*VDEL4*VDEL5} \tag{29}$$

$$\text{DCL} = \text{DCLMTE*VLAM3*VLAM4*VLAM5*VLAM6*VLAM7*VLAM8*VLAM13*VLAM14}. \tag{30}$$

These variables are extremely vital in the model, as they are used to calculate the total lift and drag coefficients of the whole wing. These outputs are final outputs, meaning they are going to be used for integration into the overall model. They are the overall outputs of the program. They will be directly integrated into GASPy along with the output CL_max from component *CLmaxCalculation*.

10. Final model

After all of the components are added, the last part of the code combines and connects all the classes together, creates a model, and runs it. Additionally, a solver is added, and partial derivatives are checked. Last, the desired output values are printed. All of this is contained in an "if __name__ == "__main__":" statement. The purpose of the statement is for when the file is imported in another file. For example, in the test file, which will be discussed below, the code must be imported from the original file. When importing, the only code that is wanted is in the classes. Anything in this statement will be omitted during an import, and all of the print statements, solvers, and code that runs the model is not necessary. Therefore, all of that code goes into this statement.

First, the model is created using the Problem class. As discussed earlier, the Problem class contains and connects all the other objects, both groups and components. It also creates the user interface to execute the model. The problem is defined as 'prob = om.Problem'. To create the model, the command prob.model = model is used.

Next, each class is added to the model as a subsystem and connected to each other. All the variables are also connected using the promotion method. In the model.add_subsystem command, all inputs and outputs are promoted. For example, *BasicCalculations* promotes output 'SA', and *CLmaxCalculation* promotes input 'SA'. This is how they are connected. Four subsystems were added, one for each class in the model.

Next, a solver was added to the model. A solver is a software package that incorporates one or more algorithms for finding solutions to multiple classes in a problem. The reason the solver needed to be added into the loop between *CLmaxCalculation* and LookupTables. In OpenMDAO, there are various options of solvers: the one used in this model is called a Newton Solver. It is the most general solver in OpenMDAO and uses Newton's method; this requires derivatives. The solver uses the derivatives declared in the components.

After the solver is added, two commands are run: prob.setup(), and prob.run_model(). These two commands do exactly what they look like they do. Once OpenMDAO runs the model, it knows the outputs, and they are ready to be printed. Print statements are used to print any of the variables and their value after they have been processed through the components in the model.

11. Validation: Test Files

In addition to the actual flaps model, test programs must be written to prove that the model is functional and giving accurate values. In addition to sourcing variables from the GASP Fortran, numerical values are also given in example output Fortran files. There are both input and final output values for every class. The values are plugged in using test commands in files which import the classes from the model. The test runs the model with these values and checks that the outputs put in match with what is calculated.

There were many different test files written to validate this model. Various tests were written for all the flap types, in the takeoff configuration, and for one flap type in the landing configuration of the flaps. All of the test files were the same in structure, but simply used different values provided in GASP output code. In a file to test the model, one test case is made for each class. In the flaps model test files, there was a *BasicCalculations* test case, and one for *CLmaxCalculation, LookupTables,* and *LiftAndDragIncrements.* The syntax/command for a test in OpenMDAO is 'unittest'.

Going through the structure of each test case, first the problem is defined, and set up. Next, input values are set using the 'prob.set_val' command. Next, the model is run, and a tolerance is defined as 'tol'. The purpose of the tolerance is to give an acceptable margin of error between what is calculated and the output values that are plugged in. The tolerance used in these tests was $2 * 10^{-4}$.

```
reg_data = 1118.21948771
ans = prob['SA']
print('SA:', reg_data, ans)
assert_near_equal(ans, reg_data, tol)
```

Figure 8: Test for variable 'SA'

After this, the output values are added as 'reg_data'. This is regulated data, and is the data that the test will check its own calculations against. The data, like most of the values in this model, also come from the GASP output files. Next, the model's own calculation is defined as the answer, or 'ans'. Lastly, the output is checked using an 'assert_near_equal' command. The command calls (ans, reg_data, tol). It compares the ans vs. the reg_data and sees if they are within the tolerance. If they are, the test passes- if not, it fails. There is also a print statement that will print the two values side by side. The way this is typed out in the file is shown in Fig. 8, with an example of 'SA'. This is done for every variable in every class of the whole model. At the end of the file, a simple command 'unittest.main()' is called to run the test. Then, assuming a successful test, the terminal will read 'OK'. If something is wrong, it will detail which value is wrong, and debugging must be done.

Below, table 2 shows some examples of outputs from GASP, their OpenMDAO calculated value, and the percent difference for each. This percent difference is acceptable due to the lack of precision in the GASP data. The examples shown in Table 2 are for a single slotted flap type in a landing configuration. The tests were successful after review, and this validated the flaps model for use so that it could be integrated into the overall model.

Variable Name	GASP Value	OpenMDAO Value	% Difference
DCL (increment on lift coefficient)	1.0293	1.0292993	1.00e-4%
DCD (increment on drag coefficient)	0.0406	0.04057903	5.17e-2%
CL_max (maximum lift coefficient)	2.8155	2.81546226	1.30e-3%
VLAM8	0.74444	0.74444322	4.00e-4%
VLAM9	0.9975	0.9975	0.0%
RNW (Reynolds Number)	157.1111	157.11115494	3.49e-5 %
SMN (Mach Number)	0.17522	0.17518636	1.92e-2%

Table 2: Model Validation Examples

12. Flaps Model Integration to GASPy

Now that the flaps model has been completed, and validated proving functionality and accuracy of the model, it is time for it to be integrated to the overall multi-disciplinary flight phases model. As discussed earlier, there are four disciplines in the model: propulsion, trajectory, aerodynamics, and weight. The flaps model is going to be integrated into the aerodynamics discipline. In this discipline, there will be two sub-groups for the aircraft wing. One with flaps deployed, and one with flaps retracted. This is because the flaps are only used during a portion of the various flight phases.

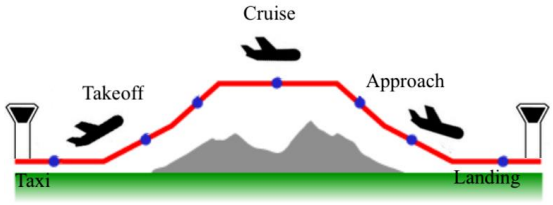

Figure 9: Phases of Flight

Looking at the different flight phases shown in figure 9, flaps are only used during parts of takeoff, ascent, and landing. The flaps are used during ground roll (accelerating to takeoff on the runway), takeoff, and the early part of the ascent. They are then retracted for the rest of ascent, cruise, and for the majority of descent. They are then deployed on the approach to the runway to slow down the plane. All these different flight phases are included in the overall model. The sub-group with flaps included will be used during these flight phases, and the one with the plain wing will be used everywhere else.

Now, it must be discussed how the flaps model will be integrated into the aerodynamics discipline. Looking back to Fig. 5, there are 3 'final outputs.' These are CL_max, DCD, and DCL. These are the overall outputs of the model and are the most important variables being calculated. They come from Explicit Components *CLmaxCalculation* and *LiftAndDragIncrements*, calculating the variables for which the components are named for. These three values are what is going to be directly put into the groups for the aerodynamics discipline. CL_max is used to help calculate for the stall speed. DCL and DCD are used in equations to calculate for the total lift and drag coefficients as well. These three variables outline the effect that the flaps have on the aerodynamics of the wing: the increment the flaps provide on the lift and drag coefficients, and the updated maximum lift coefficient.

III. Conclusion

In this paper, a computer based flaps model was developed in OpenMDAO. The top-level model, the main part of the model, had four classes which were connected by their input and output variables to calculate final outputs. 'ExplicitComponent' and 'MetaModelStructuredComponent' were both used to create structures for different equations to calculate for various output variables. In the components, inputs were added, partial derivatives were declared, and then outputs were calculated using provided equations. All variables and equations were ported from GASP Fortran code into the new GASPy Python programs using OpenMDAO.

The outputs of the model were validated using GASP data for various flap types and configurations. The final outputs were found, and will be integrated into the overall GASPy model, into the aerodynamics discipline. Once this is done, and the entire 737-MAX 8 model is functioning and validated, the model will be used for hybrid electric research. It will be used to run optimizations to improve efficiency, minimize fuel burn, and advance hybrid-electric technology in the aerospace field.

Acknowledgments

I would like to thank many people for making this opportunity happen. Being in high school, my experience working on this project was extremely beneficial, as it taught me so much about aerodynamics, perseverance through obstacles, and about work experience. First and foremost, a big thank you to NASA Glenn Research Center for having the internship program available, even virtually. Next, I would like to thank the Transformational Tools and Technologies (TTT) for sponsoring my internship, and the GASPy project as a whole. Next, I would like to thank Eric Hendricks, Jonathan Burt, Jeff Berton, and Xiao-Yen Wang for valuable knowledge here at NASA. I would also like to thank Jeff Bowles, Justin Gray, and Sydney Schnulo for extra help on the project, assisting me in navigating GASP Fortran, and in learning OpenMDAO. Last but not least, a huge thank you to my mentor, Jennifer Bergeson, for giving up many hours of her summer to both give me this opportunity and help me with all I needed while completing this project. Additionally, this paper was reviewed and edited by Jennifer Bergeson.

References

[1]"Boundary Layer." NASA Glenn Research Center, www.grc.nasa.gov/www/k-12/airplane/boundlay.html.

[2]Bowles, Jeffrey V., "GASP_MDAO_WEIGHTS_Model." NASA Ames Research Center, 2021.

[3]"Dynamic Pressure." NASA Glenn Research Center, www.grc.nasa.gov/www/k-12/airplane/dynpress.html.

[4]"Flaps and Slats." NASA Glenn Research Center, www.grc.nasa.gov/www/k-12/airplane/flap.html.

[5]Gray, J. S., Hwang, J. T., Martins, J. R. R. A., Moore, K. T., and Naylor, B. A., "OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization." OpenMDAO, openmdao.org/pubs/openmdao_overview_2019.pdf.

[6]"The Lift Coefficient." NASA Glenn Research Center, www.grc.nasa.gov/www/k-12/airplane/liftco.html.

[7]"Linear Interpolation." Wikipedia, en.wikipedia.org/wiki/Linear_interpolation.

[8]"Mach Number." NASA Glenn Research Center, www.grc.nasa.gov/www/k-12/airplane/mach.html.

[9]"Reynolds Number." NASA Glenn Research Center, www.grc.nasa.gov/www/k-12/airplane/reynolds.html.

[10]"Wing Geometry Definitions." NASA Glenn Research Center, www.grc.nasa.gov/www/k-12/airplane/geom.html.