Computational Methods and Modelling 3

Group Design Project: Longitudinal Dynamics of a Small Aircraft (Group 17)

Group Member	Contribution
Faisal Bindakhil	Primary coder for A1, A2, A3, B1 and B2 (individually developed final running code). Modular code structure (Wrote all the code in modules with comments). All code comments and explanations. Report Sections 3, 4.3 and 5. ReadMe file.
Ben Ihde	Running Prototype code and development of B1, Development code for A1, A2 and B2, Report and Analysis section 4 and half of section 5, Research for B1, B2
Keir Barbary	Wrote and finalised all code and comments for the user interface (Part C). Wrote UI section of the report, ReadMe file and finalised overall report structure. Development code for A2, B1 and prototype of modular structure.
Jack Rait	Development of code for A3 and B1, graph formatting B2, condensation of code in A1&2. Report – First Draft all sections, Intro, Flow chart & editing.
Christopher Aitken	GitHub creation and organisation. Development code A2 and A3. Code and Report Review.
Daniel Brodie	Development of code for A1 and A2. Code validation section of the report.
Innes Cameron	Prototype code for B2. Report sections 4.3 and 5, Research for B2.

Table of Contents

1: Introduction

Figure 1 is a flowchart that is used to display the functionality of the simulation algorithm visually:

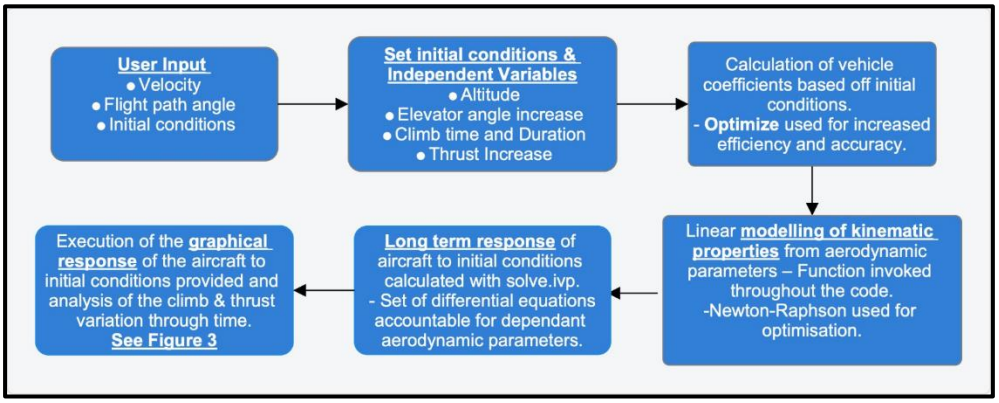

Figure 1: Flowchart of Algorithm Functionality

The simulation models the longitudinal dynamics of a small aircraft by a predetermined set of user-specified conditions via an interactive user interface (*C*). These are synchronised with a standardised set of initial conditions to create a 'guess' using a linear regression function (*A1*), in turn generating an instantaneous snapshot of the trimmed aircraft kinematics for those conditions (*A2*). The regression model variables can be extracted to find a solution for the long-term response of the aircraft by solving a set of differential equations representative of its dependent parameters (*A3*). The above can be utilised to generate trimming conditions covering a range of Independent Thrust (T) and Elevator Angles (δ_E) by limiting drag, lift and pitch moment coefficients (*B1*). Having generated the preceding method whereby trimming conditions are formulated from a set of initial conditions, the model can finally graphically display how this response unfolds over time (*B2*).

2: User Interface (Part C)

The User Interface (UI) is a script which, when executed, displays a user-friendly interface, allowing the user to input chosen variables. These user-defined variables are then used to run the necessary code via imported functions from 'MainCode.py', and the outputs are displayed as shown in the graphical output section of the flowchart (*Figure 1*). 'PySimpleGUI' was chosen instead of 'Tkinter' as the interface module is much easier to use, has advanced customisation options and works across all platforms. Once the UI file is executed, the interface appears in a pop-up window, allowing for a more professional experience.

Input	Description and Limitations	Units	Options (example)
Velocity	Velocity of the aircraft must be positive, with an approximate maximum value of 200.	m/s	100
Flight Path Angle	Angle between the earth axis and trajectory, with an approximate value between -0.25 and 1.	radians	0.05
Initial Altitude	Cruising altitude of a small aircraft should be somewhere between 1000 and 3000.	metres	2000
Value of Elevator Angle Increase	Percentage change in Elevator Angle, can be positive or negative but magnitude should be within a about 50% of initial value.	%	10
Value of Thrust Increase	Thrust step change can be positive or negative, with a magnitude of less than about 1000.	N	-20
Duration of Trim Condition	How long the aircraft will stay at the trim condition calculated prior.	seconds	100
Value of Climb Duration	How long the simulation will last for after the step change is incorporated.	seconds	300

Figure 2: Table of Inputs for UI

The first section of the UI takes user-defined values of 'Velocity' and 'Flight Path Angle' to calculate the parameters at which the aircraft is at dynamic equilibrium, also known as its 'trim condition'. This is based on aerodynamic coefficients calculated with experimental data and equations, with more detail in this report's section on 'Part A'. The second part of the UI takes another set of inputs – as shown in Figure 2 - and evaluates the flight simulation for the given data before outputting graphs of aerodynamic data against time. These visualisations detail fundamental flight dynamics, such as altitude and path angle over time, allowing for in-depth analysis and design optimisation.

For instructions on how to operate the UI, the README.py file has a User Guide in the project's GitHub repository:
ChristopherA0205/CMM-group-project (github.com)

3: Numerical Methods (Part A)

3.1 Linear Regression Model (A1)

The set of aerodynamic experimental data was symmetrically organized using the 'Pandas' library. 'Angles of Attack' and 'Elevator Angles' were converted to radians and paired with corresponding drag, lift and moment coefficients, structured into a data frame, allowing for efficient analysis, concise code and minimal computing power.

When computing the aerodynamic coefficients, the initial conditions were estimated using a linear regression model, 'linregress', from the SciPy library. These values were used as a starting point for the optimisation process. Subsequently, the 'optimise' function from the SciPy library was utilised to solve for the coefficients more accurately.

3.2 Finding Trim Conditions (A2)

Care was taken to organise the code into a few modules, ensuring none were redundant, minimising computational power and enhancing the code's execution speed. Separately defining each coefficient as a function allows for efficient reusability throughout the code. Moreover, the modular structure improves computational efficiency as repeated functions are called when needed. Additionally, this approach aids in debugging and individually testing different code components.

The Newton-Raphson technique was used to calculate the angle of attack from the non-linear equation developed in the code. This method utilises derivatives, allowing it to converge to a solution rapidly. During code testing, bracketing techniques such as the bisection method were used. However, it required significantly more iterations to converge to the same solution, making the technique less computationally efficient. An initial guess of 0.01 radians was chosen for the Newton-Raphson technique as these are typical operating conditions for a small aircraft during trim.

Unlike linear convergence methods like bisection, Newton-Raphson converges quadratically, meaning the error decreases exponentially with each iteration. The Newton-Raphson technique's utilisation of derivatives provides a more accurate depiction of the angle of attack, as it displays quadratic behaviour.

3.3 Simulation of Dynamic Response to Step Changes (A3)

The simulation integrates the functions developed throughout the code into a time-dependent structure to simulate the aircraft's response to different control inputs vital to the user interface functionality. Numerical integration from the SciPy library is used to solve the system of differential equations. SciPy's 'solve_ivp() selects the optimal variant of the Runge-Kutta method for the specific system of differential equations. The system's automated selection of the Runge-Kutta method maximises computing efficiency. The function uses a combination of dynamic and static step sizing. When integrating the differential equations, by nature, the function is dynamic. Thus, the step size is adjusted based on the local error estimate, saving computing power. Additionally, it specifies a set of points in which the solution is to be evaluated and displayed by creating equidistant linearly spaced outputs, utilising static sizing.

During the testing phase, the Implicit Euler technique was initially explored for solving the system of differential equations. However, it was observed that this approach resulted in significant errors and failed to achieve convergence, prompting the shift to the solve_ivp method. This is due to the numerical technique being of the first order, approximating the solution using a linear approach. It was found that the rapid change in behaviour of the system during the elevator angle change is too volatile for a first-order technique to model. In opposition, the Runge-Kutta method considers the rate of change of the system and uses a representative higher-order technique adequate for the data. Finally, the Euler method was also considered. Although relatively stable, it is better suited for stiff equations. Thus, it may not accurately capture the system's oscillatory behaviour, which was ineffective for this data representation. The graphical output for the specified trim conditions can be seen in *(Figure 6)*.

In *(Figure 6)*, the aircraft responds to an abrupt change of elevator angle at 100 seconds, manifesting in oscillatory behaviour across all parameters. This behaviour is indicative of an aircraft's attempt to stabilise after an abrupt change, which is a characteristic of phugoid oscillation. These oscillations are dampened over time until equilibrium due to the stability and control of the aircraft.

3.4 Code Validation of Part A

Below is a comparison of the results obtained from the two trim conditions against the validated results, displayed in figures 4 and 5:

V = 100 m/s % γ = 0.05 radian			
Variable	Group 17	Validated	Error (%)
Angle of Attack (rad):	0.0164	0.0164	0
Elevator Angle (rad):	-0.0519	-0.0519	0
Thrust (N):	3392.3	3392.35	1.50E-05
Theta (rad):	0.0664	0.0664	0
u_B (m/s):	99.987	99.986	3.00E-01
w_B (m/s):	1.641	1.641	0

Figure 3: Comparison of Group and Validated results for Trim Condition 1

V = 100 m/s % γ = 0.00 radian			
Variable	Group 17	Validated	Error (%)
Angle of Attack (rad):	0.0165	0.0164	0.6
Elevator Angle (rad):	-0.052	-0.052	0
Thrust (N):	2755.17	2755.17	0
Theta (rad):	0.01646	0.01646	0
u_B (m/s):	99.986	99.986	0
w_B (m/s):	1.646	1.646	0

Figure 4: Comparison of Group and Validated results for Trim Condition 2

From these results it was decided trim variables could be calculated with a very small margin of error. Moreover, graphical outputs can be compared to verified graphs of certain variables against time (*Figure 6*). For an aircraft operating at trim conditions (V = 100 m/s and γ = 0 rad) for the first 100 seconds, before an Elevator Angle increase of 10% for the remaining climb time duration. We can generate code using the same parameters (*Figure 7*) and compare graphical outcomes:

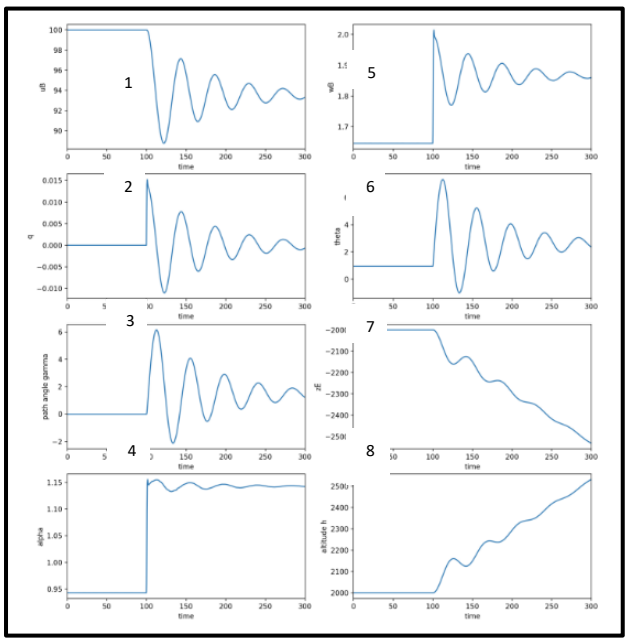

Figure 5: Validated graphical output for specified simulation.

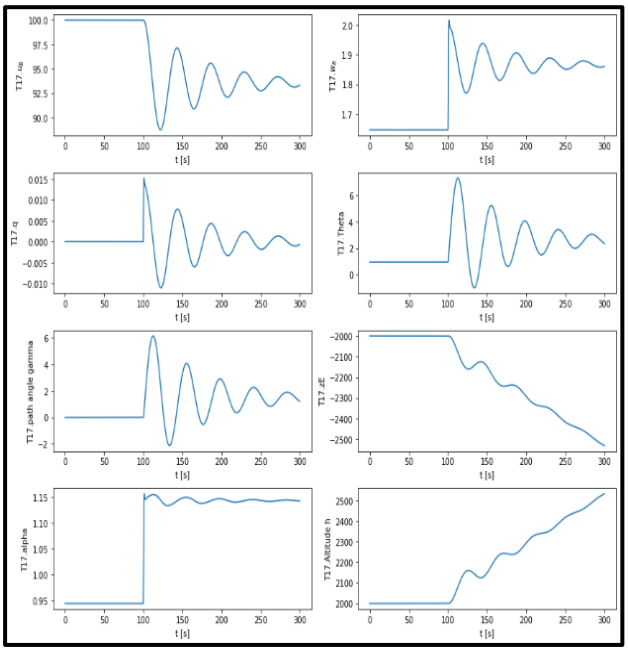

Figure 6: Group 17's graphical output for specified trim conditions and step-change values.

Looking at Figures 5 and 6, the overlapping variables from both outputs are identical, further validating the code against verified sources. The plots made for our group are denoted by T.17 in the vertical axis. Code Validation of Part B is discussed in section 4 of this report.

4: Discussion and Analysis of Design Simulations (Part B)

4.1 Methodology of Simulating a Range of Trim Conditions (B1)

Multiple graphing methods and outputs formats were attempted for part B1, settling on a set of 2-dimensional graphs (*Fig.7*). Other options tested include a 3-dimensional surface plot, which was decided against due to both the computational power required and the difficulty experienced in interpreting the results. Furthermore, multiple data intervals were trialled to find the balance between providing enough data while not harming legibility.

After several rounds of graph plotting, the upper and lower limits for the input values were established. This careful calibration ensured adherence to the system's physical limitations and the constraints imposed by the limited scope of the experimental data. Furthermore, data was extracted and analysed from existing small aircraft designs, such as the Cessna 170 (Blanchard, S. 2021), to form a comparison and validate the constraints placed on the calculations. The input values settled on were 50 to 210 m/s for Velocity and 0 to 1 rad for path angle, as these are reasonable values that do not violate any specified constraints.

4.2 Analysing Input Variables' Impact on Dynamic Parameters (B1)

Elevator Angle vs Velocity:
As both elevator angle and velocity increase, the significance of the flight path angle decreases. Moreover, a higher flight path angle corresponds to a higher elevator angle at a given velocity. This aligns with expectations, as a steeper flight path angle requires a greater elevator lift force to maintain, necessitating a higher elevator angle relative to the aircraft. Additionally, an interesting observation emerges as the velocity tends towards infinity. The relationship between elevator angle and flight parameters asymptotes towards -1 degree of elevator angle. This suggests that at high velocities, the elevator's impact on the flight path angle is amplified, with minor adjustments in the elevator angle inducing substantial changes in the flight path.

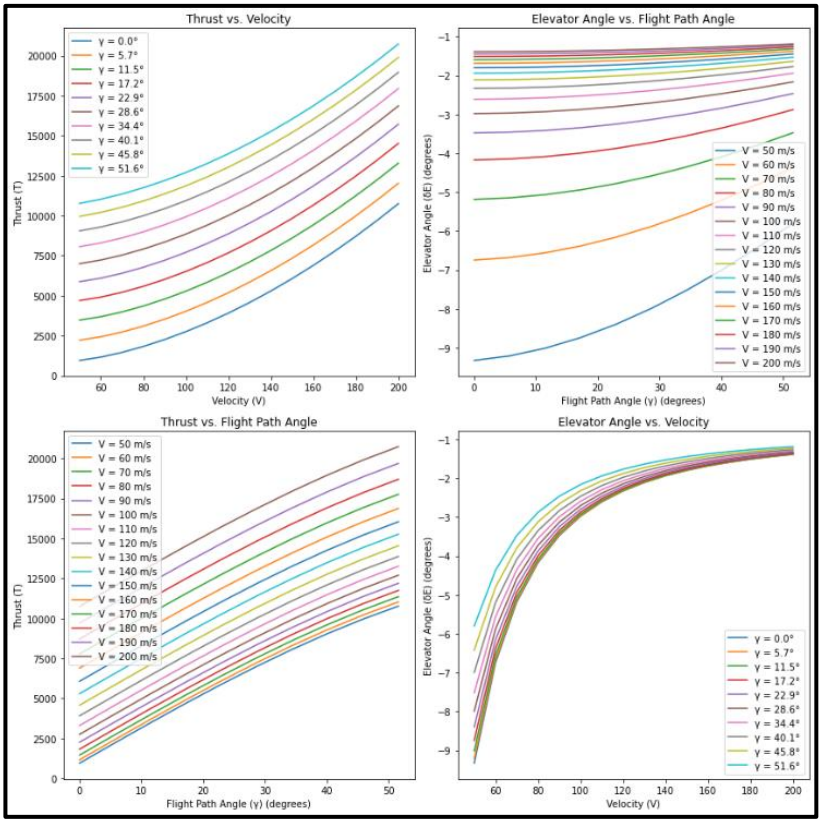

Figure 7: Plots of trimmed Parameters for range of input values

Thrust vs Flight Path Angle:
A notable observation is that the rate of change in thrust with respect to path angle is not dependent on velocity. This implies that, regardless of the aircraft's speed, the adjustment in thrust is roughly similar for a given change in flight path angle. Moreover, the relationship below path angles of 35 degrees (and for small values of thrust and flight path angle) shows a roughly linear correlation. This relationship is significant, considering that most general aviation aircraft stall at angles well below 35 degrees (Turner, T. 2016). The linearity in this range implies a proportional connection of thrust and path angle.

Elevator Angle vs Flight Path Angle:
As flight path angle tends towards infinity, the relationship between elevator angle and flight path angle asymptotes towards -1 degree of elevator angle. A noteworthy observation is that lower velocity corresponds to a higher elevator angle for a given flight path angle. This relationship aligns with aerodynamic principles, as lower velocity implies reduced airflow over the control surfaces, necessitating a higher angle of attack (AOA) to generate the required lift, with this higher AOA requiring an increased elevator angle to maintain. Furthermore, at high velocities, a minimal change in elevator angle is necessary for a significant alteration in flight path angle. This underscores the importance of translation

systems in real aircraft. These systems are designed to convert necessary control inputs into smaller deflections during phases of flight with high velocities.

Thrust vs Velocity:
It is observed that the rate of change of thrust is not dependent on the flight path angle. This implies that a change in path angle would not affect drag, as the frontal area of the aircraft relative to airflow would not change. When considering a fixed flight path angle, increased thrust corresponds to an ever-decreasing change in velocity. This inverse square relationship can be attributed to the drag equation, which is a product of the square of the velocity. The direction and magnitude of the thrust and velocity vectors are related; thus, a linear relationship of direct proportionality is established.

4.3 Analysis of Flight Climb Simulation (B2)

From the simulation (Figure 9), the climb time for the second trim condition is 274 seconds with an initial velocity of 105 m/s. Notably, sudden changes in the flight path angle (gamma) initiate the oscillations in the simulation variables. The first noticeable oscillation occurs when gamma abruptly shifts from 0 to 2 degrees, followed by another set of oscillations as gamma reverts to 0. These oscillations gradually dampen as gamma stabilises. Additionally, changes in the elevator angle contribute to mild oscillations in the system. Comparatively, research on the climb rate of a Cessna 170, which climbs at approximately 3.6 m/s (Blanchard, S. 2021), indicates that a 1000m altitude increase would take around 280 seconds. Thus, our calculated climb time of 274 seconds for a similar altitude gain aligns closely with these real-world figures, reinforcing the accuracy and validity of our simulation.

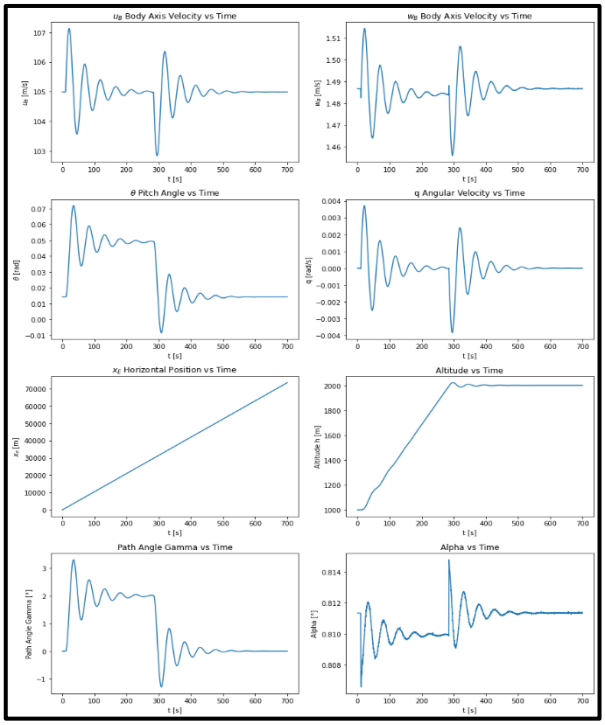

Figure 8: Plots of trimmed Parameters for range of input values

5: Limitations and Conclusion

The model of using external modules for numerical analysis does not allow for calculating any quantitative error values. Since a specific method was not specified for solve_ivp(), there is no information on the Runge-Kutta method variant used to solve the differential equations. These, however, are not of significant importance to the calculated result, as the system can be analysed and verified in the graphical output of the system. The modular structure allows for a much tidier main file. However, this causes a longer run time, thus more computing power, as the system must import files from different modules.

The algorithm can be improved by implementing direct error analysis. Incorporating methods such as Richardson extrapolation to estimate truncation errors into the numerical techniques used throughout the code. Additionally, creating an algorithm for sensitivity analysis would have helped identify which parameters are most critical to the model's accuracy, allowing for more efficient debugging.

6: References

Blanchard, S. (2021). *Cessna 170 - Everything You Need to Know about the Perfect Airplane*. [online] FLY8MA Online Flight Training. Available at: https://fly8ma.com/cessna-170/#:~:text=The%20performance%20of%20the%20170 [Accessed 09 Nov. 2023].

Turner, T. (2016). *Angle & attitude: stall/spin crashes, and how to avoid them | Flight Safety Australia*. [online] Available at: https://www.flightsafetyaustralia.com/2016/03/angle-attitude-stallspin-crashes-and-how-to-avoid-them/#:~:text=This%20image%20illustrates%20that%20in [Accessed 08 Nov. 2023].