Interner Bericht

DLR-IB-FT-BS-2025-155

Simulated System Identification of Morphing Trailing Edge Fixed Wing UAV

Hochschulschrift

Frederik Saldern Nielsen

Deutsches Zentrum für Luft- und Raumfahrt

Institut für Flugsystemtechnik
Braunschweig

Deutsches Zentrum
DLR für Luft- und Raumfahrt

Institutsbericht
DLR-IB-FT-BS-2025-155

Simulated System Identification of Morphing Trailing Edge Fixed Wing UAV

Frederik Saldern Nielsen

Institut für Flugsystemtechnik
Braunschweig

Deutsches Zentrum für Luft- und Raumfahrt e.V.
Institut für Flugsystemtechnik
Abteilung Unbemannte Luftfahrzeuge

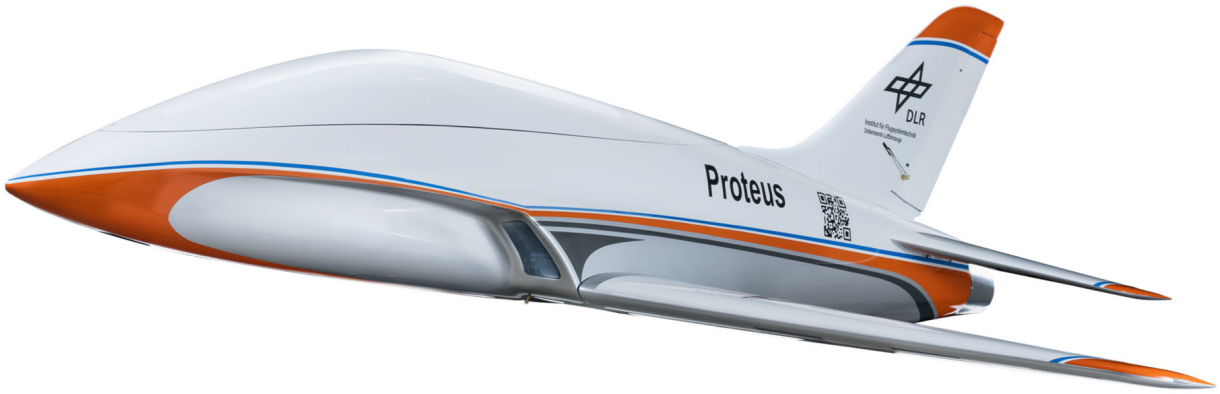
Braunschweig, den 15.09.2025

[redacted]

Institutsleitung:	Prof. Dr.-Ing. S. Levedag
Abteilungsleitung:	Johann Dauer
Betreuer:in:	Lennart Kracke
Verfasser:in:	Frederik Saldern Nielsen

Simulated System Identification of Morphing Trailing Edge Fixed Wing UAV

Masters Thesis



Masters Thesis in Simulated System Identification of Morphing Trailing Edge Fixed Wing UAV

10th Semester Robotics
Frederik Saldern Nielsen

Aalborg University
The Technological Faculty for IT and Design

The Technological Faculty for IT and Design
Niels Jernes Vej 10, 9220 Aalborg Øst
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Simulated System Identification of Morphing Trailing Edge Fixed Wing UAV

Theme:
Masters Thesis in Robotics

Project Period:
Robotics 10. Semester (ROB10)

Project Group:
-

Participant(s):
Frederik Saldern Nielsen

Supervisor(s):
Jan Dimon Bendtsen (AAU)
Lennart Kracke (DLR)
Mark Spiller (DLR)

Copies: 1

Page Numbers: 51

Date of Completion:
September 15, 2025

Abstract:

This report outlines a grey-box approach to controller training and simulation of a Morphing Trailing Edge (TE) fixed-wing Unmanned Aerial Vehicle (UAV). The method combines physical modelling through a Flight Dynamics Model (FDM) with synthetic flight data generation for non-linear system identification. This enables the development of an analytical model that captures the system's dynamic behaviour across a wide flight envelope.
The project is part of the MorphAIR initiative, aiming to enable safe flight and control of a morphing UAV demonstrator. The final goal is to establish a model-based design pipeline, capable of transitioning between conventional and morphing wing configurations while supporting accurate, simulation-based controller development. The thesis developed a working UAV simulation and excitation pipeline but struggles to accurately identify non-linear MIMO dynamics in Closed-Loop simulations. Although the SINDy-based non-linear system identification overfitted in both open- and closed-loop flights, further experiments show that a mixed excitation model can still capture Open-Loop responses under simulation-based excitations.

Preface

This Master's Thesis has been made in close collaboration with Deutsches Zentrum für Luft- und Raumfahrt (DLR)'s ULF (Unbemannte Luftfahrzeuge) department as part of the MorphAIR project. The work is a continuation of an internship conducted at the ULF department in Braunschweig, Germany. The internship work conducted in Braunschweig was focused on setting up a Reinforcement Learning (RL) environment on the developed Flight Dynamics Model (FDM) simulation: PyFDM.

The majority of this thesis work has been conducted at Aalborg University.

Aalborg University, September 15, 2025

Frederik Saldern Nielsen
fsni20@student.aau.dk

Contents

Acronyms

AOA Angle of Attack. 9, 46, 47

APRBS Amplitude Pseudo Random Binary Signal. 25, 26, 30, 34, 38, 39, 41, 43

CAD Computer Aided Design. 30

CFD Computational Fluid Dynamics. 10, 11, 30

DLR Deutsches Zentrum für Luft- und Raumfahrt. i, ii, 5, 6, 18, 28

FDM Flight Dynamics Model. ii, 22

LSODA Livermore Solver for Ordinary Differential Equations. 31

MIMO Multiple-Input Multiple-Output. 23, 25, 26, 43, 44

MorphAIR Morphing Technologies & Artificial Intelligence Research Group. ii, 4, 5, 13, 19, 44

MSE Mean Squared Error. 35

N4SID Numerical Subspace State Space System Identification. 23, 31

NARMAX Nonlinear AutoRegressive Moving Average with eXogenous input. 23

NASA National Aeronautics and Space Administration. 4, 47

NLR Netherlands Aerospace Centre. 4

NN Neural Network. 21

OOMS Orthogonal phase-Optimized MultiSines. 25, 26, 30, 34, 37, 38, 43, 44

PID Proportional – Integral – Derivative. iii, 15, 16, 32, 33

PPO Proximal Policy Optimization. 15, 18, 19

PRBS Pseudo Random Binary Signal. 25

PTERA Prototype-Technology Evaluation and Research Aircraft. 4

PyFCM Python Flight Control Module. 15, 32, 33

PyFDM Python Flight Dynamics Module. ii, 13, 15, 18, 19, 28, 29, 30, 31, 32, 35, 41, 43, 44, 49

RL Reinforcement Learning. ii, iii, 13, 15, 18, 19

SFD Scaled Flight Demonstrator. 4, 5

Sim2Real Simulation to Reality. 14, 19

SINDy Sparse Identification of Non-linear Dynamics. 24, 35, 43, 44

STLSQ Sequential Thresholded Least-Squares. 24

TE Trailing Edge. 3, 5, 6, 7, 8, 9

UAS Unmanned Aircraft System. 5, 30

UAV Unmanned Aerial Vehicle. i, 4, 5, 7, 10, 11, 12, 13, 16, 19, 21, 23, 27, 28, 29, 30, 31, 33, 35, 44, 47

ULF Unbemannte Luftfahrzeuge. ii

VLM Vortex Lattice Method. 9, 10, 11, 22, 27, 30

1 Introduction

In the span from 1990 to 2019 the global commercial passenger aviation and freight has seen a four times increase in demand. The future demand for global commercial aviation is expected to further increase, causing a rise in global CO_2 emissions. [1] The aviation industry is a heavy emitter of CO_2, and greenhouse gases, because of kerosine and contrails, which includes water vapour left in the atmosphere. The relatively young aviation industry is responsible for an estimated 4% of the total rise in human-caused temperature increase. [1, 2]
Although the aviation industry has quadrupled in size from 1990 to 2019, the emitted CO_2 has only doubled. Due to CO_2 emission goals and fluctuating fuel prices, the aviation industry has decreased emissions through advances in biofuels, larger flight hauls, and new technologies. [1]

A group of new flight technologies, promising to save fuel, are morphing wing technologies. These technologies change the shape of the wing, in order to reach optimal flight conditions [3, 4, 5].
Different morphing technologies are currently being developed, such as span-wise morphing altering the wingspan to dynamically change wing-aspect-ratio. High aspect ratios save fuel during cruise conditions, and low aspect-ratios are ideal for slower take-of and landing. Morphing techniques such as variable sweep wing can rotate the wings to gain higher sweeps improving lift to drag ratio, around super and subsonic flights. These technologies have shown promising results in both simulation and wind tunnel experiments. [3].
Another morphing wing technology which has shown promising results in wind tunnel experiments, is the Morphing Trailing-Edge (TE) Wing. Tests of half span morphing TE wing has shown promising results in lowering drag compared to conventional ailerons, and require smaller actuator deflections to produce the equivalent rolling moment. [6].
A major problem with morphing TE wings are their complexity of control, since the camber of the wing changes drastically during simple manoeuvrers, and the aerodynamics also vary continuously. With many actuators on the wings becoming deeply coupled, there is a growing demand for real-time optimisation and control in order to fully benefit from morphing TE wings. [3]
The next step in the deployment and research of Morphing TE wings is real flight tests, where precise control and optimisation of the wing are crucial for demonstrating its performance [3].

1.1 Initial Problem Statement

How can Morphing Trailing Edge Wings safely be controlled and optimised for real flight tests?

---

2 Problem Analysis

The problem analysis begins with an overview of the MorphAIR project, followed by the Proteus demonstrator, which includes both conventional and morphing wings. Finally, a description is provided of how the morphing wing can improve flight performance.

2.1 MorphAIR Project

Morphing technologies show promise in both fuel efficiency, flying with higher loads, and allowing more aggressive manoeuvres. But a fundamental problem in the validation process of emerging morphing technologies, is a lack of real flight tests. In recent years, scaled flight tests have been conducted on UAV demonstrators in order to reduce costs and mitigate risks
Prototype-Technology Evaluation and Research Aircraft (PTERA), developed by NASA (National Aeronautics and Space Administration) and AREA-I, is a scaled UAV of a Boeing 737 transport plane. The Scaled Flight Demonstrator (SFD) is used to test flexible wing tip, and used to counter the influence of turbulence and gusts. [7] Likewise, Airbus has developed a SFD on the A321, called AlbatrossOne, testing flapping wing tips with the aim to reduce the impact of gusts and turbulence, with the goal of optimising fuel usage and wing loads on transport planes. [8] Netherlands Aerospace Centre (NLR) has experimented with SFD's produce at 1:8.5 scale of an Airbus A320, intended for rapid testing of emerging technologies. [9] The three SFD's can be seen in Figure 2.1

(a) In-flight view of NASA PTERA with flexible wingtips

(b) In-flight view of Airbus AlbatrossOne with flapping wing tips

(c) In-flight view of NLR's A320 SFD

(d) Ground test of NASA PTERA

(e) Ground test of Airbus AlbatrossOne

(f) Ground test of NLR's A320 SFD

Figure 2.1: Scaled Flight Demonstrators: (a–c) in-flight, (d–f) corresponding ground tests of NASA PTERA, Airbus' AlbatrossOne, and NLR's A320 SFD.

An iteration of the SFD by NLR used to test electric propulsion, was destroyed when a battery caught fire on the runway, and there has been no news of NLR's SFD since. [10] This is a testament to uncertainties and dangers when testing new technologies, even on scaled airplanes.
The current challenge with SFDs is that producing a model from the ground up to resemble a real aircraft—especially for new and risky technologies—can be expensive. In order to create a less risk-

averse and cost effective SFD test-bed, Deutsches Zentrum für Luft- und Raumfahrt (Deutsches Zentrum für Luft- und Raumfahrt (DLR)) has an interdisciplinary research group MorphAIR (Morphing Technologies & Artificial Intelligence Research Group).
By sourcing commercial, off-the-shelf parts for UAV testbeds, MorphAIR seeks to rapidly test emerging technologies on cheap UAV demonstrators. One of these demonstrators is the fixed wing UAV, Proteus initially used for testing morphing wing TE technologies.

2.2 Proteus

The DLR TE morphing wing demonstrator named Proteus, has a fuselage based of the hobby sport jet called Futura XXL, by Tomahawk Aviation. To facilitate wing modularities, the fuselage has been amended and the landing gear has been moved. A new jet engine has been installed and the avionics has been retrofitted. The commercial UAV and the retrofitted Proteus, can be seen on Figure 2.2.

Figure 2.2: Image of the sport jet Futura XXL (Top) & the retrofitted Proteus (Bottom), picture by DLR.

The first iteration of Proteus features a pair of conventional wings, allowing the first iteration of the UAS (Unmanned Aircraft System) be tested and serves as a baseline for the evaluation of morphing technologies. The second iteration of Proteus is a set of morphing TE wings, these specialised wings can be attached directly on Proteus' fuselage.
Two sets of wings are developed for the same UAV, to establish a baseline of the UAS with conventional and well understood features. This allows potential performance improvements to be compared on the same basis, but now with the morphing wing installed.

Currently DLR has conducted initial piloted flight tests, with the purpose of testing the airframe with the conventional wings. On Figure 2.3 the first flight test of Proteus with conventional wings can be seen, and in Appendix A.3 the full video can be found.

Figure 2.3: The first flight test of Proteus with the conventional wings.

DLR has made progress with the development of the conventual winged demonstrator. However, in order for the Morphing wing to be tested, an ability to control and fly the demonstrator safely is essential. This highlights the need of further research into morphing wings, in order to produce safe flight controllers.

2.2.1 The Morphing Trailing Edge Wing

A simple overview of aerodynamics, covering technical naming of fixed wing aircraft parts, and the basis of lift and drag, can be found in Appendix A.1. This is a general overview explaining lift and drag, moreover it covers the technical terms of wing geometry and naming of actuator surfaces, and lays the foundation for the following equations included in this section.

The implementation of the Morphing wing is still in its early stages. Currently the morphing wing is under construction by DLR. A description of the produced morphing TE wing from a material-scientific point of view, can be found in this preprint by Tikalsky et al. [4] An illustration of the morphing on the TE, can be seen on Figure 2.4.

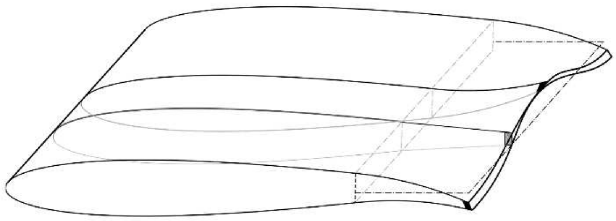


Figure 2.4: Illustration of morphing wing, Tikalsky et al. [4]

The wings consists of 10 servos and each wing segment, with the ability to deform the flexible material on the Trailing Edge. This results in multiple different ways to reach a certain lift distribution for the UAV. The following equations in 2.1 and 2.2 attempt to explain the main area where the morphing TE wing can help to optimise lift.

The drag is expressed as in Equation 2.1:

$$D = \frac{1}{2}\rho V^2 S C_D \tag{2.1}$$

Where:

D is the drag force acting parallel to the airflow.
ρ is the air density.
V is the velocity of the airflow relative to the wing.
S is the wing area.
C_D is the induced drag coefficient, influenced by the wing area and surface shape.

This means in situations with low velocities, induced drag constitutes a large proportion of the total drag. On Figure 2.5, the relation between other drag sources and induced drag can be seen with their impact on total drag.

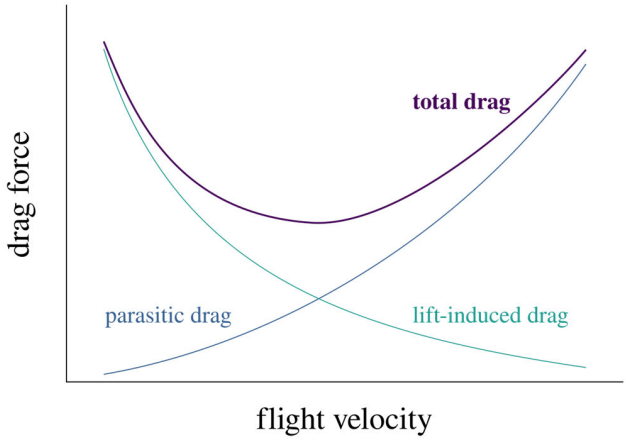


Figure 2.5: The different drag sources' impact on total drag, depending on velocity. Figure collected from [11]

Other drag sources are usually called parasitic drag or form drag, and are more prevalent at higher velocities.

The induced drag coefficient C_D from Equation 2.1 can be expanded to an equation of total drag, seen in Equation 2.2.

$$C_D = C_{D0} + \frac{(C_L)^2}{\pi e_0 AR} \tag{2.2}$$

Where:

C_D is the total induced drag coefficient.
C_{D0} is the zero lift drag, based on the geometry of the wing and potential extremities of the plane, causing a higher zero lift drag coefficient.
C_L is the total lift force, expressed by the equation of lift.
e_0 is the Oswald efficiency number, which will be further explained below.
AR is the aspect ratio, determined by the geometric relation between length and width between the cord and span of the wing.

Since C_D is partially based of the inherent lift of $(C_L)^2$ over wing geometric constants, the drag coefficient rises when the lift coefficient does. The impact of this term will be reduced with a high Oswald efficiency number. This explains why the Oswald efficiency number e_0 of Equation 2.2, is an important factor. It describes how close the wing shape is to the ideal drag-reducing wing shape. This wing shape is the ideal elliptical wing which ensures an Oswald efficiency number of 1. The further away from the elliptical shape, the Oswald efficiency number decreases, and the induced drag coefficient increases. One of the better known planes with wing shapes resulting in high Oswald efficiency numbers are the WW2 Spitfire planes, seen on Figure 2.6

Figure 2.6: Illustration of Spitfire Mk IIa P7350 with near ideal elliptical wing shape. Figure collected from [12]

As seen on Figure 2.5, when planes fly at higher velocities, the induced drag decreases, and thus the specific shape of the wing becomes less important. A morphing TE wing can also save fuel by optimising lift to drag ratio because of its smooth surface deflectors with no gaps. It can improve controllability because of more flexible actuators, the morphing wing might improve the impact of gusts, and alleviate loads during aggressive manoeuvrers. [13]
However, for the Proteus demonstrator equipped with the Morphing TE wing, the most important impact comes from the change of the wing shape and the increase in the Oswald efficiency number when doing deflections. This is because Proteus primarily flies at lower velocities, and is not built for complex manoeuvres.

An example of how the morphing TE wing can improve the Oswald Efficiency number can be seen on Figure 2.7, where both the conventional wing and the morphing wing has been simulated with (VLM).

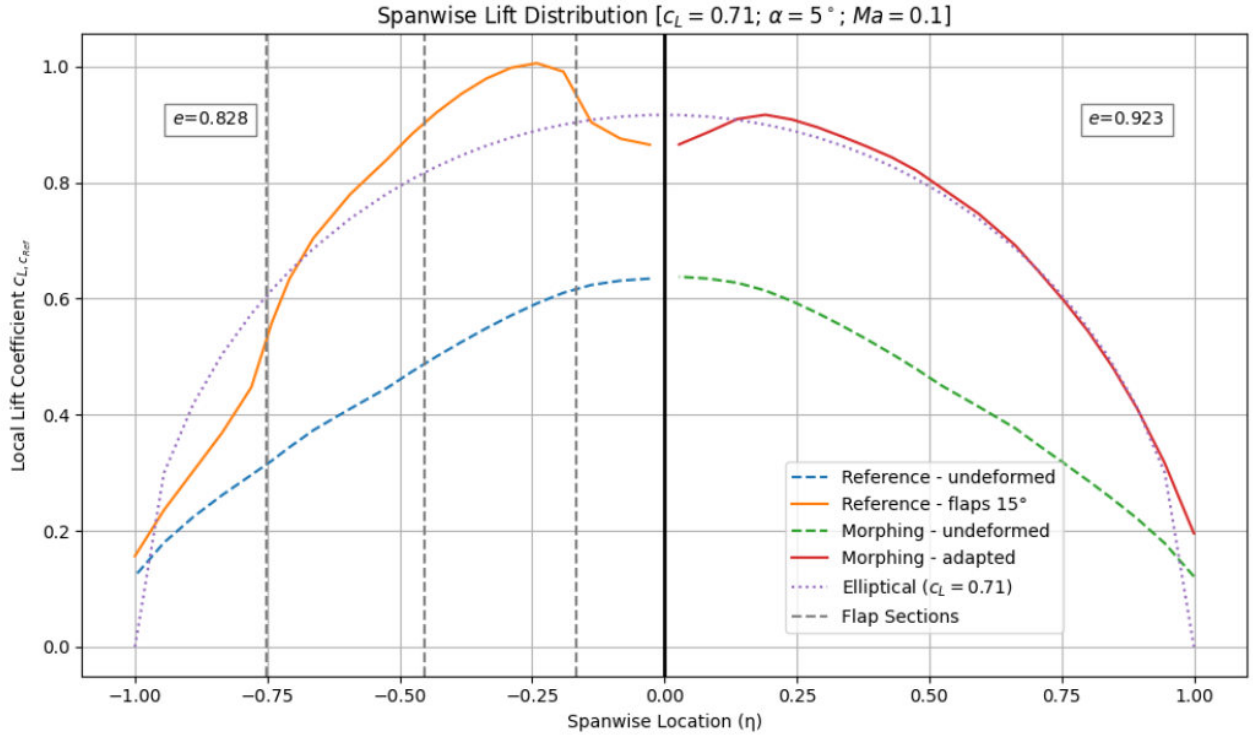


Figure 2.7: Vortex Lattice Method of both the conventional/Reference wing (left) & the Morphing wing (right). The x-axis describes the normalised span of the wing. The y-axis is the local lift coefficients over the span of the wing, all summing up to the lift of $cL = 0.71$ required for reaching an Angle of Attack (AOA) of 5 degrees.

Figure 2.7 is a comparison between the span-wise lift coefficients of the conventional wing to the morphing wing. In the figure, both wings are first compared in their undeformed states, visualised in the blue & green graphs, with a total spanwise lift coefficient of $cL = 0.43$ resulting in an AOA of 0 degrees. Then both wings are tasked to produce the same spanwise lift coefficient of $cL = 0.71$ resulting in an AOA of 5 degrees. The conventional wing has to deform its inner flaps by 15 degrees, where the morphing wing can produce the required lift coefficient in a more distributed manner. The elliptical dotted line is the ideal elliptical wing shape, producing an Oswald efficiency number equal to 1, thus minimising the lift induced drag.
The benefits of the morphing wing can be seen on the Oswald efficiency number of $e_0 = 0.923$ which inherently reduced the induced drag, and the morphing wing need to do a much smaller deflection to reach an AOA of $\alpha = 5^\circ$, compared to the conventional wing.
The values of this can be seen on Table 2.8, where in this specific case the increase in Oswald efficiency factor for the morphing wing results in a lift to drag increase of +4.9%.

Configuration	cL = 0.43 (Alpha = 0 deg)	cL = 0.71 (Alpha = 5 deg)
Oswald:		
Reference Wing	0.886	0.828 (Flaps 10 deg)
Morphing Wing	0.886	0.923
Difference	0	+11.49 %
c_{Di} :		
Reference Wing	0.0100 (100 dcts)	0.0290 (290 dcts)
Morphing Wing	0.0100 (100 dcts)	0.0256 (256 dcts)
Difference	0	-11.76 %
L/D: (D ~ c_{D0} + c_{Di} ; c_{D0} = 0.0300)		
Reference Wing	10.9	12.21
Morphing Wing	10.9	12.81
Difference	0	+4.9 %

Figure 2.8: Table Containing the VLM results of deflection between conventional and morphing wings.

An image of Proteus equipped with the Morphing wing can be seen on Figure 2.9.

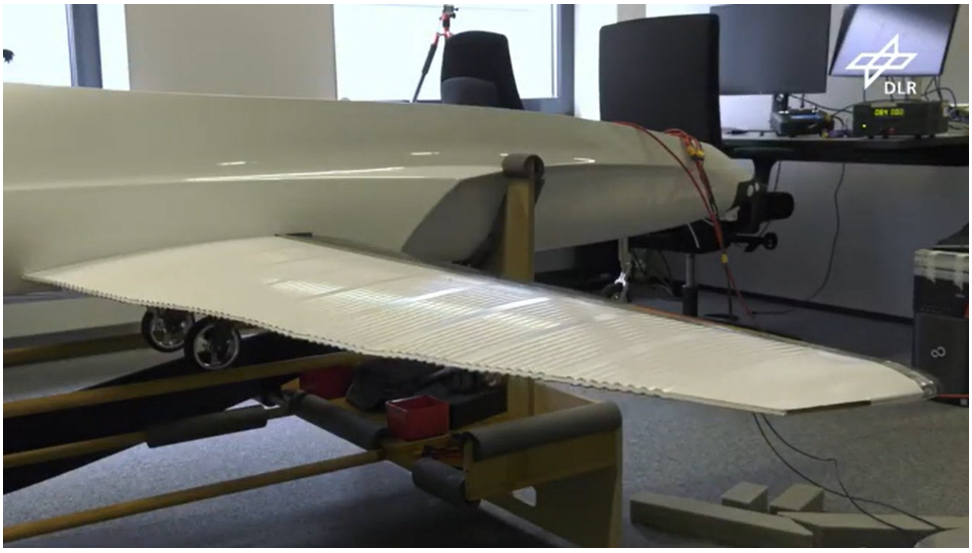


Figure 2.9: Image of Proteus equipped with the Morphing Wing.

A video of the Morphing wing deflecting can be found in Appendix A.3.

2.2.2 Vortex Lattice Method

The Votex Lattice Method (VLM) is an analytical tool that can estimate the spanwise lift distribution of the wings. VLM handles the wing geometry as distributed panels where the lift and drag contribution of all panels are used to estimated lift and drag coefficients. The VLM is used instead of a traditional CFD program, because it can estimate coefficients of the morphing wings in approximately 2-3 minutes on a normal laptop. This is in stark contrast to CFD simulation of the entire UAV, which

takes 1-8 hours on a HPC (High-Performance Computing) depending on the granularity. The VLM makes it feasible to run offline simulations where the lift/drag coefficients are returned for each wing configuration.

2.3 Classical Control of Fixed Winged UAVs

An overview of fixed wing UAV control techniques is needed. A common approach used to describe fixed winged aerodynamics for use in control is calculating the aerodynamic forces and moments directly from lift and drag coefficients. The dynamic equations are often simplified to lateral and longitudinal forces, with no assumed cross coupling. This method is often preferred because it can easily be assumed linear and has closed form solutions.
The forces are described by an expansion of lift and drag coefficients describing the forces of the fixed wing plane. The coefficients used for Proteus can be found in the Appendix A.2. Most of these coefficients can be assumed constant in specific flight conditions, this means in order to model the flight dynamics for an entire flight envelope, a lookup table with the corresponding coefficients is needed. The coefficients are usually generated through different CFD simulations. But VLM can also be used to estimate wing related coefficients.

The total lift force coefficient can be described as:

$$C_L = C_{L0} + C_{L\alpha}\alpha + C_{L\beta}\beta + C_{L_p}\frac{B_{ref}p}{2V_{TAS}} + C_{L_q}\frac{C_{ref}q}{2V_{TAS}} + C_{L_r}\frac{B_{ref}r}{2V_{TAS}} + C_{L_\xi}\delta_a + C_{L_\eta}\delta_e + C_{L_\zeta}\delta_r \quad (2.3)$$

The total drag force, follows the principles of Section 2.2.1, with the parasitic drag being a static coefficient in the look-up table and the induced drag coefficient being calculated based on lift in the Equation 2.2. This quadratic drag term can also be linearized to a coefficient included in a look-up table.
The total lateral force coefficient which explains the side force of the plane, can be calculated by the following equation:

$$C_Y = C_{Y0} + C_{Y\alpha}\alpha + C_{Y\beta}\beta + C_{Y_p}\frac{B_{ref}p}{2V_{TAS}} + C_{Y_q}\frac{C_{ref}q}{2V_{TAS}} + C_{Y_r}\frac{B_{ref}r}{2V_{TAS}} + C_{Y_\xi}\delta_a + C_{Y_\eta}\delta_e + C_{Y_\zeta}\delta_r \quad (2.4)$$

The forces can then be calculated from these total coefficient and their relation the airflow on the wing, as seen on the equation2.5.

$$\begin{aligned} F_x &= -C_D Q S_{ref} \\ F_y &= C_Y Q S_{ref} \\ F_z &= -C_L Q S_{ref} \end{aligned} \quad (2.5)$$

Where:
$C_D(\alpha,\beta,\delta)$ is the drag coefficient (function of angle of attack, sideslip and control deflections).
$C_Y(\alpha,\beta,\delta)$ is the side-force coefficient.
$C_L(\alpha,\beta,\delta)$ is the lift coefficient.
S_{ref} is a constant describing the area of the wing.
Q is the airflow, described by Equation 2.6.

$$Q = \frac{1}{2}\rho {V_{TAS}}^2 \tag{2.6}$$

Where:
ρ is the current atmospheric pressure.
V_{TAS} is the true airspeed of the UAV.

The moments of the plane is similarly explained by total lift and drag coefficients:

$$C_l = C_{l0} + C_{l\alpha}\alpha + C_{l\beta}\beta + C_{l_p}\frac{B_{ref}p}{2V_{TAS}} + C_{l_q}\frac{C_{ref}q}{2V_{TAS}} + C_{l_r}\frac{B_{ref}r}{2V_{TAS}} + C_{l_\xi}\delta_a + C_{l_\eta}\delta_e + C_{l_\zeta}\delta_r$$
$$C_m = C_{m0} + C_{m\alpha}\alpha + C_{m\beta}\beta + C_{m_p}\frac{B_{ref}p}{2V_{TAS}} + C_{m_q}\frac{C_{ref}q}{2V_{TAS}} + C_{m_r}\frac{B_{ref}r}{2V_{TAS}} + C_{m_\xi}\delta_a + C_{m_\eta}\delta_e + C_{m_\zeta}\delta_r$$
$$C_n = C_{n0} + C_{n\alpha}\alpha + C_{n\beta}\beta + C_{n_p}\frac{B_{ref}p}{2V_{TAS}} + C_{n_q}\frac{C_{ref}q}{2V_{TAS}} + C_{n_r}\frac{B_{ref}r}{2V_{TAS}} + C_{n_\xi}\delta_a + C_{n_\eta}\delta_e + C_{n_\zeta}\delta_r$$

The moments can also be derived similarly to the forces, the moment equations can be seen on Equation 2.7.

$$\begin{aligned} M_x &= C_\ell \, Q S_{\text{ref}} \, b_{\text{ref}}, \\ M_y &= C_m \, Q S_{\text{ref}} \, c_{\text{ref}}, \\ M_z &= C_n \, Q S_{\text{ref}} \, b_{\text{ref}}. \end{aligned} \tag{2.7}$$

Where:

b_{ref} is the wing span used for roll and yaw moments.
c_{ref} is the mean aerodynamic chord used for pitch moment.
C_ℓ, C_m, C_n are the non dimensional roll, pitch, and yaw moment coefficients.
These classical fixed wing dynamical equations can easily be linearized over specific flight envelopes [14]

Linearization

For control design and stability analysis, these non-linear aerodynamic coefficients are usually linearized about a steady trim condition. Denote the steady values by a subscript "0" and small deviations by $\Delta(\cdot)$. Then, for example,

$$\Delta C_L \approx C_{L\alpha}\,\Delta\alpha + C_{L_\xi}\,\Delta\delta_a + \cdots + C_{L_q}\,\frac{\bar{c}_{\text{ref}}}{2V_0}\,\Delta q,$$

and similarly for $C_D, C_Y, C_\ell, C_m, C_n$. Usually on UAVs, the dynamic pressure is measured directly by a pitot tube on the aircraft, and does not need to be linearised. Retaining only first-order/linear terms gives a standard state-space form for the rigid-body dynamics. [14].

3 Prior Work

This chapter gives an overview of the work and conclusions previously done on the MorphAIR project. The goal of the internship has been to facilitate training of reinforcement learning controllers in simulation, and prepare a realistic environment for safe implementation of flight controllers. The initial target for the Reinforcement Learning (RL) environment was to create an automatic gain tuner for the inner- and outer-loop controllers intended for the fixed wing UAV. The work is a summation of what was previously done in by the author during the 9th semester. [15]

3.1 Development of PyFDM

To enable training of an RL agent that can be safely deployed on the real UAV, a simulation environment was developed in Python, which is called PyFDM. This simulation suite is a series of modules, ranging from the simple aerodynamic solver to the sensor suite can be individually replaced or changed. The simulation environment is still under continuous development, and modules are being actively added to PyFDM for further granularity of the simulation.

3.2 Disturbance Modelling

Bridging the gap between simulation and reality required the incorporation of realistic atmospheric disturbances. Domain randomization was grounded in established standards, with turbulence and gust models drawn from United States Department of Defence MIL-F-8785C standards and its later revisions. The Dryden turbulence model was selected for its rational transfer functions, facilitating a six-degree-of-freedom wind disturbance profile based on altitude, true airspeed, and specified intensity. Only a low-altitude variant has been implemented, since initial flight envelopes remain near the ground. White-noise trajectories are passed through the Dryden filters and sampled via zero-order hold, yielding time-varying linear and rotational disturbance velocities, that can be used in a discrete simulation.
The Dryden turbulence adds velocity changes to both the rotational and lateral direction, the Dryden turbulence is stochastically generated based on a generalised transfer function. The disturbance varies by height, wing-span, airspeed, and an intensity parameter. An example of generated Dryden turbulence can be seen on Figure 3.1.

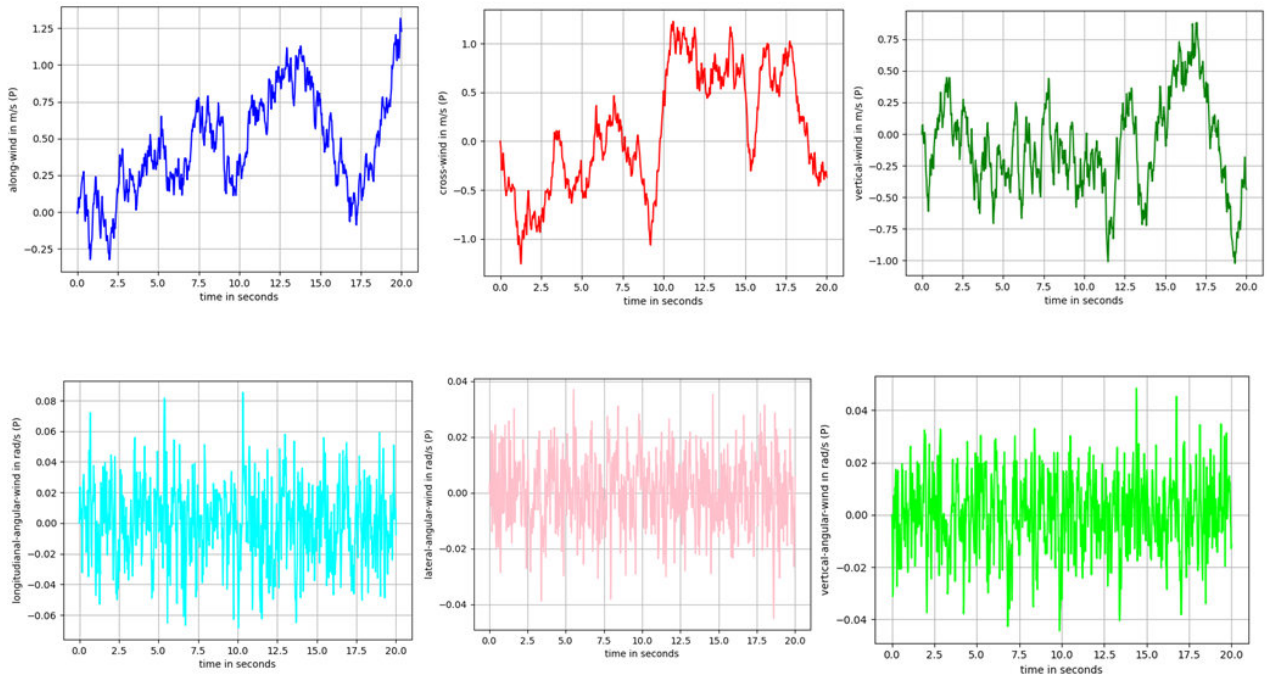


Figure 3.1: Plot of Dryden turbulence velocities, added to the lateral and rotational velocities.

3.3 Servo Modelling

Accurate servo dynamics are critical for Sim2Real transfer. A second-order transfer function was derived from system identification experiments on DroneCAN HBL388 servos. In the test rig, PWM (Pulse Width Modulation) commands sweep from minimum to maximum while the resulting servo rotation is recorded, as seen on Figure 3.2. Only the raw servo angle is logged, since the exact kinematic mapping to surface deflection remains under development on the Morphing wing. The deflection for the conventional wing is assumed a linear scalar from servo rotation to aileron surface deflection.

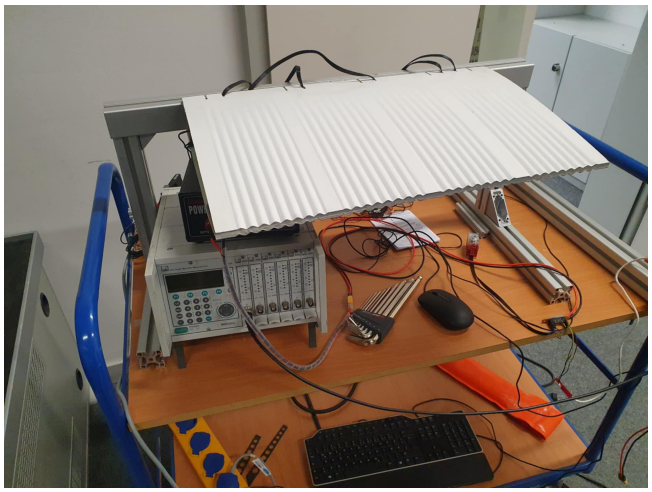


Figure 3.2: Test rig of Morphing wing segment,

This data enables least-squares fitting of time-delay, through subspace methods. Which gives a discrete time model that describes the system lag both from the mechanical inertia and communication latency. Once identified, the servo model is inserted into the simulation loop as a linear system matrix. The RL agent can learn to compensate for the true actuator responses.

3.4 RL PID (Proportional – Integral – Derivative) Autotuner Setup

The automatic gain tuner uses Stable-Baselines3's implementation of Proximal Policy Optimization (PPO). The training environment is wrapped as a Markov decision process in which each episode comprises a single decision, which is the choice of three PID gains for the inner-loop roll controller, part of the Python Flight Control Module (PyFCM). Upon selection, PyFCM applies these gains for the duration of the 500-step simulation, during which PyFDM evolves the aircraft state. Multiprocessing across eight parallel environments accelerates data collection and stabilises PPO gradient estimates for a final model. Figure 3.3, illustrates the environment setup as a Markov decision process, where the agent gets access to the initial dynamic pressure and then outputs suitable PID gains.

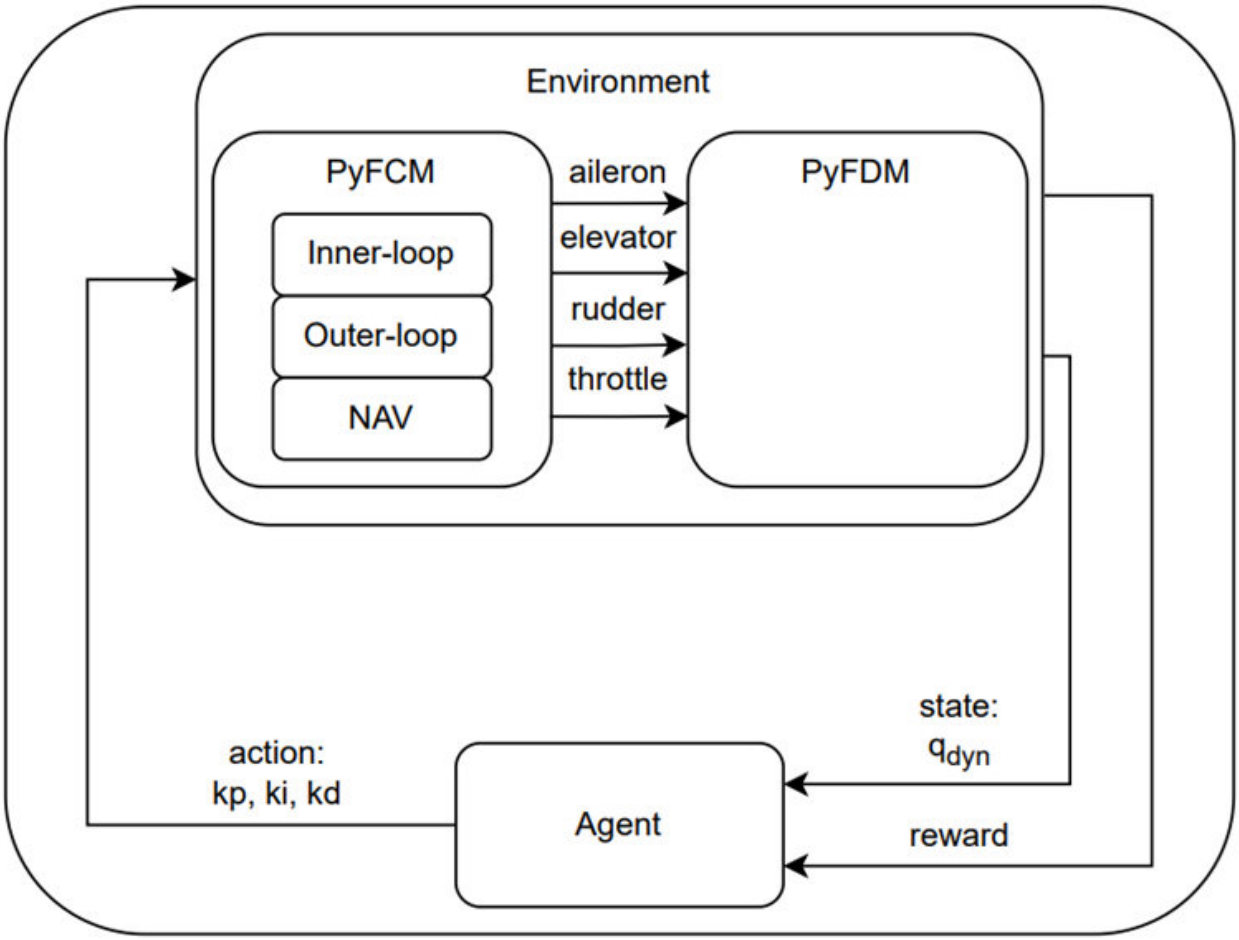


Figure 3.3: Illustration of RL training environment for episodic PID gain tuning.

3.5 Reward Function

A second order reference model for roll dynamics serves as the foundation of the reward function. The penalty is proportional to the squared deviation from the desired roll angle and its distance from the desired response defined by a target damping ratio and natural frequency. The training run is terminated with heavy reward penalties for the UAV doing ground collisions or stalls.
The reason a second order system was used a reward function was to avoid incentivise the agent to choose high PID gains. The second order reward model can be seen on Figure 3.4.

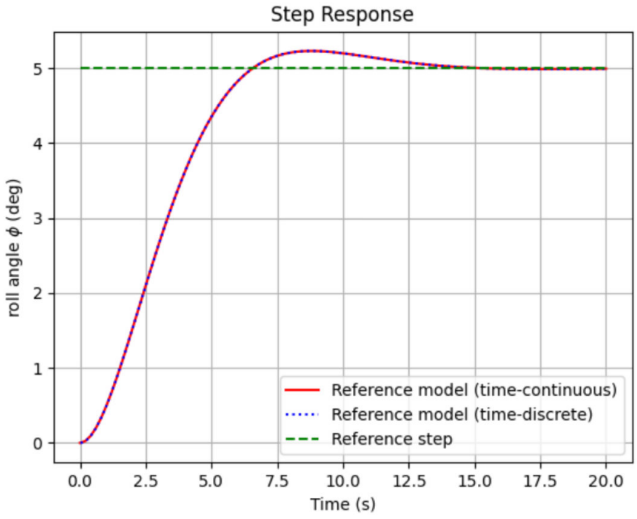


Figure 3.4: Response of second order system with $\zeta = 0.7$ & $\omega = 0.5$.

The specific Natural frequency and damping ratio was hand-tuned so the reference roll response would, mimic common response behaviour for UAVs commanded to roll.
The reason why the agent has been trained to follow a predefined second order system is to ensure that the UAV acts predictably when the PID gains are used in real flight tests, and discourage any aggressive manoeuvres learned in simulation.

3.6 Training and Results

Over 100 000 episodes on multi-CPU setups, the agent converged to relatively low proportional and integral gains that achieved stable roll control. When tested with varying roll set points every 100 steps, the trained policy produced smooth, slightly under-damped responses, which can be seen in Figure 3.5.

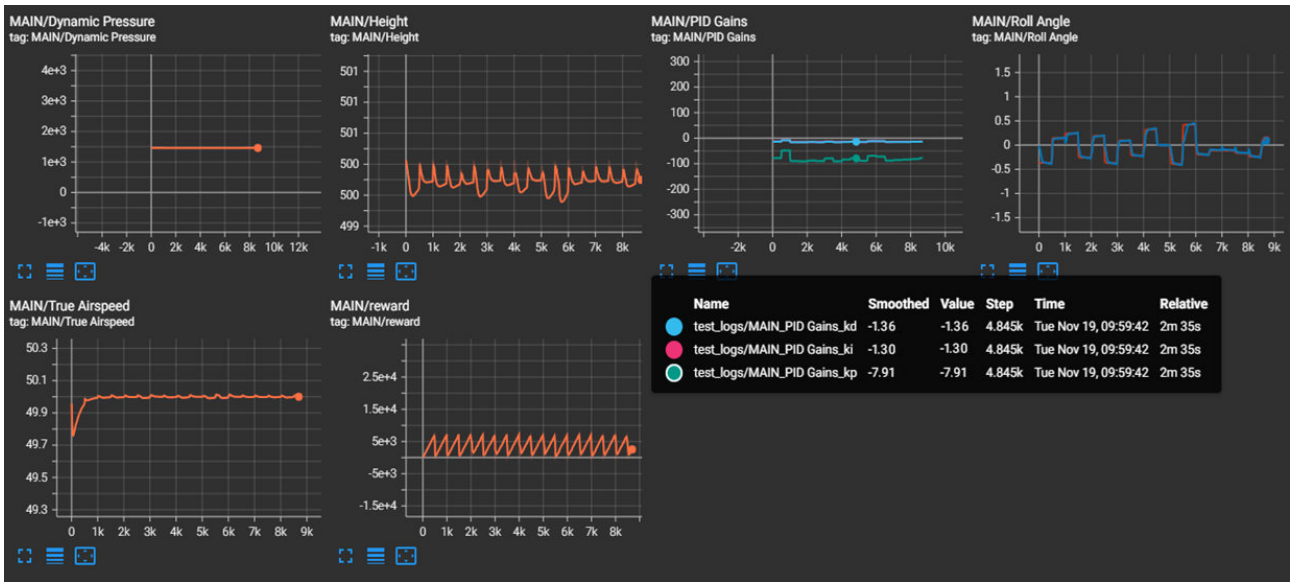


Figure 3.5: Screen-shot of gains converging during training.

These roll gains were still relatively aggressive even with the second order reward function. This means generally over the flight envelope the less aggressive hand tuned gains are safer to fly with.

Module	**Block**	K_p	K_i	K_d	**Limits**
Roll	Controller_{ϕ}	-2.0	0.0	0.0	—
	$\text{Controller}_{\text{RL }\phi}$	-1.36	-1.30	-7.91	—
	Limiter Aileron	—	—	—	$[-25°, 25°]$
Pitch	$\text{Controller}_{\theta}$	-0.5	-0.2	0.0	—
	Limiter Elevator	—	—	—	$[-25°, 25°]$
Throttle	Gain	1.0	—	—	—
Yaw	Controller_{R}	0.0	-1.0	0.0	—
	Limiter Rudder	—	—	—	$[-30°, 30°]$

Table 3.1: Inner-loop controller parameters PID gains and actuator limits, with the RL converged PID roll gains.

Module	**Block**	K_p	K_i	K_d	**Limits**
Course	Controller_{χ}	1.0	0.0	2.0	—
	Limiter Roll Angle	—	—	—	$[-30°, 30°]$
Speed	Controller_{V_a}	1.0	0.2	0.0	—
	Limiter Throttle	—	—	—	$[0, 1]$
Height	Controller_{h}	0.2	0.0	0.0	—
	Limiter Pitch Angle	—	—	—	$[-20°, 20°]$

Table 3.2: Outer-loop controller parameters PID gains and actuator limits.

3.7 Hardware-in-the-Loop Inference

The work also included preparation for launching RL agents on a companion computer for running inference. This NVIDIA Jetson companion computer is installed on Proteus, and enables both training and deploying PPO agents live. The trained PPO policy, originally saved in PyTorch format, was exported to ONNX and accelerated with NVIDIA's TensorRT back-end on a Jetson Orin NX. This conversion ensures even larger and more sophisticated GPU-accelerated models can be deployed to the PX4's companion computer architecture. And RL agents trained in PyFDM can be directly deployed on Proteus,

3.8 Conclusion and Further Work

The internship was concluded with the development of the initial modules for PyFDM. Even though the RL trained gains were slightly too aggressive for automatic gain tuning, the RL training pipeline still works. And DLR are continuing the development of offline RL training in PyFDM.

The current plan is to utilise PPO to learn the optimal dynamics and control of both the conventional wing and the morphing wing of Proteus, and let the RL agent directly control the surfaces.
This should first be done for the conventional wings, and if PyFDM proves realistic enough for training and testing agents on the real Proteus, an RL training environment for the morphing wing could be made.

4 Delimitation

This thesis deviates slightly from previous work, which was focused on setting up the flight simulation, and designing an RL pipeline.The MorphAIR project will focus on training controllers for both conventional and morphing wings, using RL in order to be able to optimise the flight profile of the morphing wing.
This can prove problematic as it relies on a ”one-shot” approach where the Sim2Real (Simulation-to-Real) gap is closed by training an agent in simulation and testing in flight. This is challenging as the PPO RL algorithm used for training agents, is completely black box empirical model.
It would be preferable to have an additional analytical surrogate model, that can be simulated fast enough to predict if any actions taken by the RL agent, can lead to any unstable conditions or constraint breaches in the future.

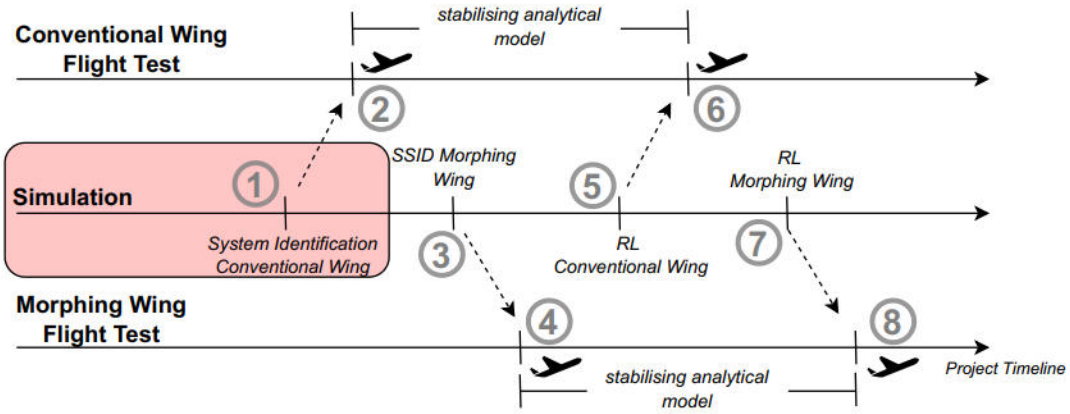


Figure 4.1: Illustration of MorphAIR pipeline, consisting of 3 project timelines. The top timeline represents flight tests of the conventional wing. The middle represents developments in simulation. The bottom represents flight tests with the Morphing wing.

Figure 4.1 represents the different planned steps of MorphAIR. Step 1 is the creation of the simulation environment and the system identification analytical surrogate model for the conventional wing. The aim of step 2 is to verify this model on the real conventional winged UAV and create an analytical safety controller, that can be used on Proteus as a protective safety layer. The same will be done for the morphing winged Proteus in step 3 and 4, where an identified analytical surrogate model should be able to predict future states based on current inputs and states. Where a stabilising analytical model can function as a protective layer for when empirical and untested RL agents has to be deployed, reflected by step 5 & 6, and 7 & 8, for the conventual and morphing wings, respectively.

This thesis focuses on the red part on Figure 4.1, by setting up the simulation pipeline of PyFDM (Python Flight Dynamics Module), and developing system identification strategies, that might eventually capture the dynamics of the morphing wing in simulation.

4.1 Final Problem Formulation

The final problem formulation can be described based on the problem analysis, the prior work and delimitation of the MorphAIR project.

Can a simulation environment for a fixed wing UAV, be non-linearly system-identified over multiple flight suites.

---

5 Methods

The methods chapter delves into the background of different kinds of identified models.Specifically methods focuses on covering the state of the art of system identification within aerospace, including how non linear cases are handled and the trade off's of between open loop and closed loop system identification. Subsequently, a non linear system identification approach will be presented, followed by a suitable excitation scheme.
.

5.1 White, Grey, & Black Box Models

Currently in aerospace, the system identifications of airplanes and UAV's (Unmanned Aerial Vehicle) follow some general approaches. The white box modelling approach utilises the underlying physics of the vehicle, and a set of equations are setup to from physical properties that mimic real world behaviour. [16]

A common approach to creating white box model of UAVs is to base them on underlying physical knowledge, and formulate uncoupled equations of motions. In aerospace, these are usually linearised systems based on lift equations which describes lateral, longitudinal, yawing-moment, and rolling-moment equations of motions are often used for formulating linearized state space models of airplanes. [16] [14]. This decoupled non-linear model, can then be linearized and used as a white box model for a controller, as explained in Section 2.3.

Black box methods are data-based approaches, where data is used to identify responses of an unknown system. Usually the system responses is modelled purely from input/output data, gathered from experiments around operating point of the unknown system. Black box methods do suffer from a complete lack of transparency, since it only attempts to explain the input/output process through mathematical models. [16] [17]

Usually these black box models are trained on flight data, capturing the poorly understood mechanics, or complicated non-linearities. Different black box system identification methods has been used in aerospace, such as subspace identification methods, Neural Network (NN)s, or genetic algorithms. These methods are good at capturing the precise dynamics present in the data, but are generally hard to interpret, as these models do not directly convey physical meaning. [17] [16]

Grey box methods are mix of white and black box approaches, where better known dynamics are derived by underlying physics and poorly understood or highly non-linear parts are covered by flight data. Examples of grey box modelling can be improvements of flight parameters derived from white box approaches, where methods such as parameter estimation or correction for white box approaches, describing specific smaller non-linear parts with data based models in white box descriptions. [16] [18].

Black box methods can also be modified to include grey or white areas, by utilising standard lift equations or more complicated geometric lift descriptions formulated from white box approaches, these can then be simulated generating data that can be identified by a black box approach. With

flight simulators getting better and more sophisticated, this simulated system identification is getting more traction. [16]

Method	Interpretability	Model Fidelity	Examples of Use
White Box	High	As high as the underlying equations allow	Physics-based lift equations
Black Box	Low	Depends on quality of data and is prone to overfitting	Input/output system identification
Grey Box	Medium	Trade-off between physics- and data-based approaches	Mixed parameter estimation or simulation

Table 5.1: Comparison of White, Grey, and Black Box methods

As seen in Table 5.1, the white box methods are generally regarded ideal in aerospace because of the high interoperability, but the model's accuracy is only as good as the underlying physics and assumptions. If a white box approach is used on a system with poorly understood dynamics or weird couplings, a worse model could be derived. On the other hand pure black box modelling suffers from low interpretability, and the data laying the foundation of the black box model must be of high quality and broadly represent the system at the desired operating points in order to reduce over fitting. The middle ground of grey box modelling does seem to contain best of both worlds, with a balance between verifiable models for classical aerodynamics, and precise data-based models for complicated or highly non-linear subsystems.

This work will focus on a grey box pipeline utilising Flight Dynamics Model (FDM) simulation to generate synthetic data for further system identification. The goal is to cover a wide flight envelope and eventually be used for a Vortex Lattice Method VLM simulation.

5.2 Linear System Identification

When performing linear system identification from a purely black-box perspective, a common challenge is that only the system's inputs and outputs are known. Therefore, the internal system dynamics are unknown and must be estimated as illustrated in Figure 5.1.

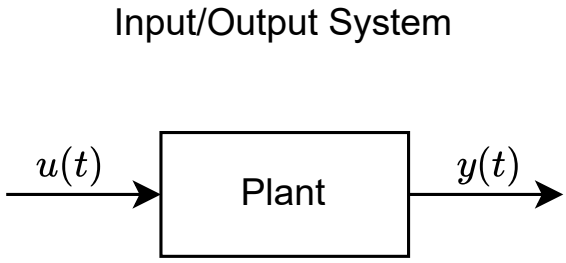


Figure 5.1: Illustration of the system identification problem. The inputs: $u(t)$ and outputs: $y(t)$ are known, however the the plant system is unknown.

When doing system identification in aerospace, the UAVs surface deflections are used as inputs $u(t)$ and the resulting states are used as outputs $y(t)$, and from this the UAV plant dynamics can be found. For estimating linear systems from input/output data, different methods has been developed, and many variations exist.

In aerospace, modern subspace methods are currently preferred for linear system identification. These methods utilise the linear properties of systems to derive system matrices.
Subspace methods such as MOESP (Multivariable Output-Error State Space) and N4SID (Numerical Subspace State Space System Identification) or variations thereof, are popular linear system identification choices in aerospace, since they output linearised state space matrices from Multiple-Input Multiple-Output (MIMO) systems and can handle coupled dynamics.
Both of these methods use the properties of linear system such as Hankel matrices and observability criteria, but because of these linear properties they have trouble handling very non-linear systems.

Since the Morphing wing model, needs to optimise lift and drag, which change dynamics polynomially over different heights and velocities, a system identification method that can handle these non-linearities is needed.

5.3 System Identification of Non-linear Systems

In order to capture the expected non-linearities of the aerodynamics, explained in Section 2.2.1, non-linear system identification methods are needed.
Neural networks are great at fitting complex input/output relationships, however they do not give an explicit analytical description of how inputs are transformed into outputs. Instead they behave purely as black box models with parameters tuned by data, which makes them hard to interpret or analyse in closed form. [19] By contrast, polynomial NARMAX (Nonlinear AutoRegressive Moving Average with eXogenous input) models specify a clear input–output difference equation, but when extended to MIMO systems the number of possible regressors explodes, which especially becomes a problem when identifying highly coupled dynamics such as a an UAV [20]

Another non-linear capable identification algorithm which can both handle non-linearities, return analytical and interpretable models, and do not explode in parameters when identifying MIMO systems,

is Sparse Identification of Non-linear Dynamics (SINDy) (Sparse Identification of Non-linear Dynamics). [19]

5.3.1 SINDy

Sparse Identification of Non-linear Dynamics (SINDy) is a newer system identification method, which seeks to find the most sparse or the least terms to describe input - state pairs. The methods can take input/output trajectories, which SINDy tries to predict using non-linear terms.

$$\mathbf{x}_{k+1} = \Theta\big(\mathbf{x}_k, \mathbf{u}_k\big)\,\Xi. \tag{5.1}$$

Where:
$\mathbf{x}_k$ is the state vector at sample k.
$\mathbf{u}_k$ is the control input at the same sample.
$\Theta(\mathbf{x}_k, \mathbf{u}_k)$ is the library of candidate functions.

Ξ is the coefficient matrix to be identified, each column corresponds to one state equation and will be reduced through a sparsity cut-of parameter.

Equation (5.1) is in discrete time, but SINDy can also handle continuous system. When initialising the SINDy algorithm, a feature library is given, holding potential non-linear terms suspected to be present in the system, as seen in Equation 5.3.1.

$$\Theta(\mathbf{X}) = \begin{bmatrix} | & | & | & | & & | & | & \\ 1 & \mathbf{X} & \mathbf{X}^2 & \mathbf{X}^3 & \cdots & \sin(\mathbf{X}) & \cos(\mathbf{X}) & \cdots \\ | & | & | & | & & | & | & \end{bmatrix} \tag{2.4}$$

This candidate library can be manually constructed with the suspected non-linear terms in the system, but should be kept concise.

For a data set of length N the stacked regression reads

$$X^+ = \Theta\,\Xi, \qquad X^+ = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top,$$

with the matching library matrix Θ. Identification is posed as

$$\Xi^\star = \arg\min_{\Xi} \frac{1}{2}\big\|X^+ - \Theta\,\Xi\big\|_F^2 + \lambda\,\big\|\Xi\big\|_0, \tag{5.2}$$

Where:
$\|\cdot\|_F$ is the Frobenius norm; sums the squared prediction error over *all* states.
$\|\cdot\|_0$ counts the non-zero elements of Ξ, enforcing sparsity.
$\lambda > 0$ is the sparsity threshold. Larger λ gives more coefficients driven to zero.

Problem (5.2) is solved with Sequential Thresholded Least-Squares (STLSQ) (Sequential Thresholded Least-Squares), where ordinary least-squares steps alternate with coefficient pruning until all remaining terms exceed λ in magnitude. The threshold λ and the highest polynomial degree in the library are selected on a basic train/validation grid. Finally, the model is refitted on the combined

train + validation set, returning the coefficient matrix of $\Xi^{\star}$.

Having found the sparse coefficient matrix $\Xi^{\star}$, the identified dynamics can be reconstructed by plugging $\Xi^{\star}$ back into the full feature library.

$$\mathbf{x}_{k+1} = \Theta\big(\mathbf{x}_k, \mathbf{u}_k\big)\,\Xi^{\star}. \tag{5.1}$$

Each column of $\Xi^{\star}$ contains the non zero weights for the corresponding state equation, and "deactivates" the features that do not live up to the threshold of λ.

5.4 Excitations

In order to catch the necessary dynamics for the system identifications algorithms to create representative system models, the input/output data needs to represent the desired operating points. As covered in the Aerodynamics section 2.2.1, the forces and moments of the morphing wing vary a lot, especially based on height and velocity. This makes it crucial to capture output data from as many expected and varied flight conditions as possible. And since the system dynamics are deeply coupled based on inputs, as many combinations of excitations of the input surfaces are needed too.

For linear system identification, common methods include random noise, burst random noise, pulse impact, multisine, and sine dwell. Multisine sweeps are considered state of the art for classic linear models, providing good bandwidth with shorter records and less leakage compared to random noise, this means fewer excitations are needed to capture accurate dynamics. [21].

Non-linear system identification requires inputs that can capture how harmonics change with both input level and base frequency, harmonics being multiples of the input frequency present in the signal. An exponential swept sine or chirp is effective for this, as its continuously varying frequency and amplitude allow for mapping dependencies and separating high order terms. Inputs must span both frequency and amplitude for non-linear systems, in contrast to linear systems. This varying frequency and amplitude is necessary, whether through chirps, phase varying multisines, or amplitude modulated PRBS (Pseudo Random Binary Signals), otherwise higher order terms may be unobservable. [22].

Amplitude modulated PRBS or APRBS, that also varies the amplitude stochastically within maximum and minimum bounds. It is essentially a series of square waves with stochastically sampled amplitude and pseudo random pulse duration, where the sign of the amplitude changes every pulse. APRBS is considered the standard excitation type for non linear system identification. [23].
Another promising excitation method, Orthogonal phase-Optimized MultiSines (OOMS) (Orthogonal Optimised Multisines), works by ensuring that each actuator excites a distinct slice of the spectrum with orthogonal frequency content, and by optimising the phase to minimise the peak factor. Meaning the peaks of the sines on different inputs do not coincide, and create destabilising dynamics. [24]

For MIMO systems, including non-linear cases, multi-sinusoidal signals are preferred over simultaneous uncorrelated PRBS, which can fail to excite low or slow reacting dynamics properly.
This can be fixed by the using orthogonal frequency sets so each actuator excites a distinct part of the spectrum [25], exactly as OOMS does. [24]. It is also advised to increase the scalar amplitude in later runs to capture non linear terms while keeping the frequency content the same. [25].

An illustration of the open-loop excitation scheme, can be seen on Figure 5.2.

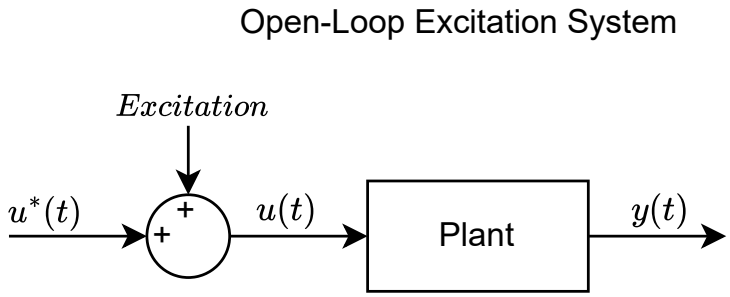


Figure 5.2: Open-Loop excitation scheme, where APRBS and OOMS are intended as excitioans for the system.

In closed-loop control, inputs become correlated through the feedback loop, especially classic identification formulas can misestimate the frequency response because of this.
Augmented identification equations are needed, and they work best when inputs are orthogonal phase-optimised multisines. [26]. OOMS satisfies this by minimising the peak of the different input channels through phase optimisation, keeping the system from overexcited instability while multiple inputs can be isolated. [24].
It is stated as long as the signal to noise ratio is not too high 5% amplitude is enough to capture dynamics, OOMS specifically works poorly when the Signal to Noise ratio is lower than 10. [24].

APRBS and OOMS both seem to be good excitation schemes for non-linear systems, but it is important for MIMO systems that different input-channels are distinguishable through the system.
For closed loop systems it gets a bit more complicated since the controller will bias the inputs. The closed loop system identification would look like the following illustration on Figure5.3.

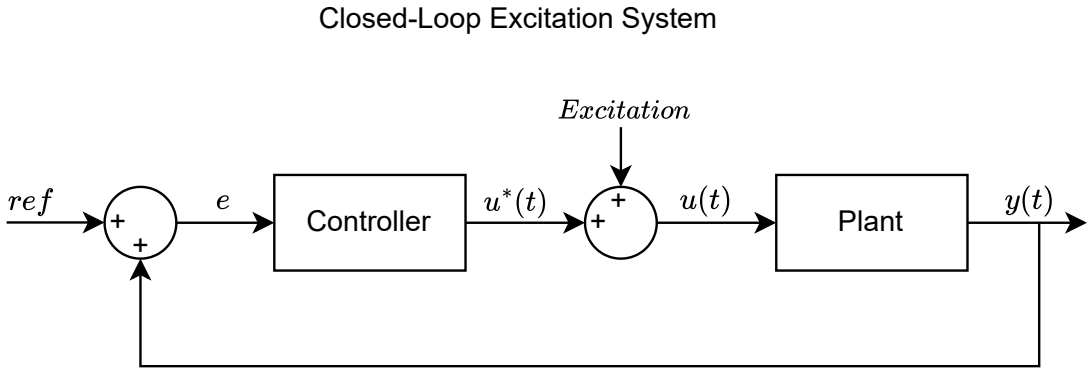


Figure 5.3: Closed loop system excited by adding excitations directly to the controller output: $u^*(t)$

Such a closed-loop system would need to be augmented with in order to unbias the excitations from the controller[26].

When comparing identified models, it is important to cross validate potential hyper parameters and excitation schemes. A common way to score or compare identified models is. [27]
Since the system identification of the conventual winged UAV dynamics and the morphing wing VLM based dynamics is done completely in simulation, it is possible to do the system identification purely in open-loop. Any destabilising or catastrophic excitations will simply end the simulation. But if similar experiments were to be conducted during real flight tests, a supporting controller would be required.

This is why system identification of both open-loop and closed loop will be conducted, well knowing that the closed-loop system identification will be biased.

6 Implementation

This chapter will cover the implementation of PyFDM and the excitations for the system identification. The general pipeline of data generation and system identification can be seen on Figure 6.1

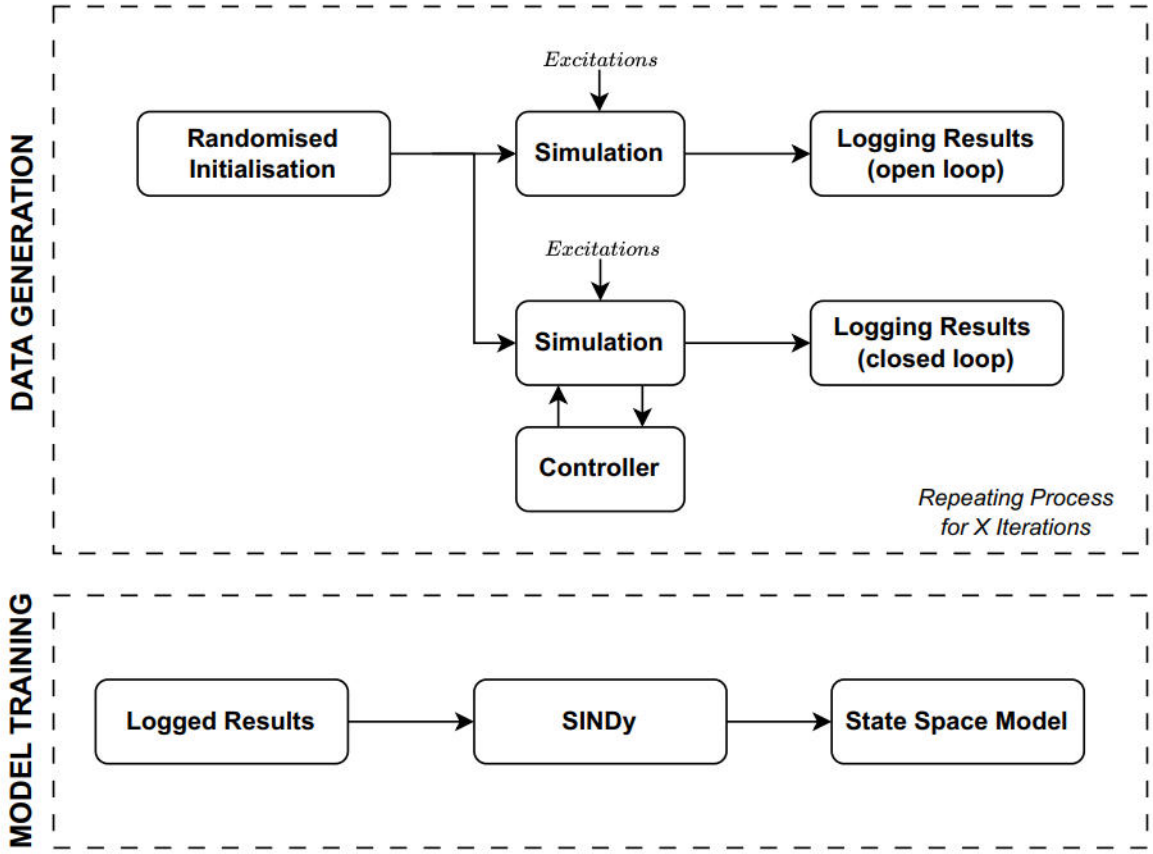


Figure 6.1: Block diagram of data generation and system identification pipelines.

6.1 Simulation

The simulation of Proteus is done in DLR's custom made Flight Dynamic simulator called Python Flight Dynamics Module (PyFDM). Parts of this simulation has been explained in Chapter 3, but the part of interest is explained further.
The simulation follows a modular architecture, where different parts of the simulation pipeline can be retrofitted to support different systems. An example of the modules included in the simulation pipeline and the configuration used for identifying the dynamics of the simplified wing UAV can be seen in Figure 6.2

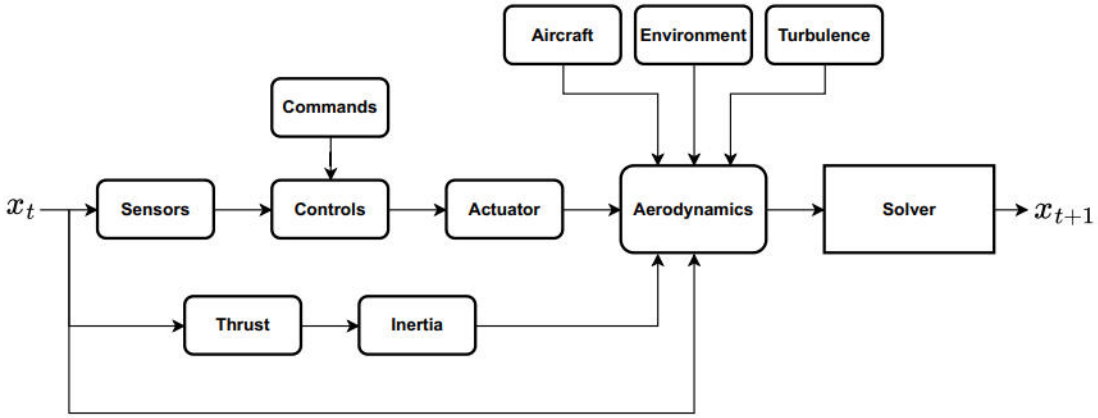


Figure 6.2: Diagram of the PyFDM simulation pipeline, with modules used for simulating fixed-wing dynamics.

All of the modules seen in Figure 6.2, are used to instantiate a simulation module which defines parameters, mathematics, disturbances, and controllers for the specific simulation.
When instantiating the simulation the inherent modularity of PyFDM makes it possible to for example compare two different controllers, or different airplanes to each other in simulation. The following part will explain what components has been used for different modules:

- Simulation.aircraft
  - Aircraft parameters and physical properties used for the specific UAV in simulation.
- Simulation.environment
  - Atmospheric properties of the earth, such as magnetic field and air density.
- Simulation.aerodynamics
  - Aerodynamic model, calculates forces and moments from the aircraft modules coefficients.
- Simulation.inertia
  - Models variable mass form loss of fuel.
- Simulation.thrust
  - Thrust model outputting forces and moments and fuel consumption.
- Simulation.commands
  - Module returning control inputs to the controller.
- Simulation.controls
  - Module serve as an input to the simulation for potential autopilot controllers.
- Simulation.sensors
  - Sensor module allowing for added disturbances on states.
- Simulation.actuator
  - Actuator delay of surface deflections.
- Simulation.turbulence
  - Stochastic disturbance modelling turbulence.
- Simulation.solver
  - Integrates the dynamics and returns the states one step in the future.

6.1.1 Physical Modules

The Aircraft module allows for different physical parameters for the simulated UAVs, for the Proteus demonstrator wing geometry, UAS wing and Inertia tensor, and fixed wing lift coefficients from a CFD simulation set point. The Inertia tensor is estimated from a CAD (Computer Aided Design) model, which has then been simulated in CFD, giving the following properties seen in Appendix A.2. Different aerodynamics models can be used. Initially a simple fixed wing aerodynamic model is used, using the physical parameters from the aircraft module with the current states and surface deflections. Through the fixed wing aerodynamics covered in Section 2.3, the forces and moments acted on the UAV can be found.

Another relevant aerodynamic model that can be used, is a Vortex Lattice Method (VLM), which geometrically solve the lift of each actuator segment of the wing. It should be noted that the VLM currently does not handle the reduction of slat effect by the flexible wing segments. A live VLM module has not been implemented into PyFDM as a module yet, but is the next step, in order to model the morphing wing.

6.1.2 Control Modules

The commands module can represent many different things depending on the configuration of the module. Logically the module is supposed to represent the commanded stick input of the pilot. But for this system identification pipeline the command module is used for generating excitations, through the MultiInput class a preplanned trajectory of excitations can be exerted on the aircraft. An example of the planned excitation event can be seen in Table 6.1.

Input Type	**Surface**	**Start Time [s]**	**Duration [s]**	**Deflection [°]**
Square Pulse	Aileron	5	10	25
Sine	Elevator	7	8	5
Step	Rudder	3	6	15

Table 6.1: Template of excitation commands applied to control surfaces during simulation.

These excitation types can be scheduled at the same time across different surfaces, making it possible to provoke actuator coupled dynamics. The Square Pulse and Sine input types allows for both OOMS and APRBS excitation schemes to be generated and inserted into the simulation.
The controls module can be omitted, by a direct law turning commanded values directly into actuator deflections.
Direct law is the mode that is used for open-loop simulation, so the excitations will be directly applied to the UAV.

But for closed-loop, the controls module facilitates external controllers giving control outputs to the actuator surfaces of PyFDM. The input wrapper class to the controls module called Mixed Controller, ensures that any excitations overrides the control outputs of the controller whenever a surface excitation is activated.

6.1.3 Disturbances & Sensor Modules

It is possible to add sensor noise, to have a discrepancy between the actual simulation states and the reported sensor values. Through this implementation ideal sensors has been assumed, since no proper state of the art filtering strategies has been developed for PyFDM. Noise filters are present in the avionics flight controller for Proteus, and it is assumed that the sensor noise from flight data is negligible.

The servo actuator responses of Proteus are simulated through a linear actuator response module. The N4SID subspace identification method mentioned in Section 5.2, has been used to model the servos of the wing through hardware in the loop data acquisition. Whenever a control output is sent to the servos a linear response model is activated, modelling response delays.
PyFDM does support dynamical disturbances such as atmospheric wind and turbulence. During simulation only Dryden turbulence is used, since the flight regime of Proteus is of too low altitude for the computationally heavy atmospheric model to have large effect on the UAV. The Dryden turbulence described in Section 3, is used to generate the artificial noise of this simulation. The intensity of the turbulence is randomised, so different flight trajectories will experience different levels of noise.

6.1.4 Solver Module

The solver integrates the forces and moment of the vehicle dynamics one step into the future.
Scipy's Livermore Solver for Ordinary Differential Equations (LSODA) integrator is used as a basis for calculating the next states through the dynamics equations.
LSODA ensures that optimisation problems containing changes in stiffness can be solved, this is especially prevalent when simulating at high velocities.
The integrator step of the dynamics, takes in the aircraft lift coefficients from the aircraft model. The inertia model, the trust model, environment model, surface deflections, and the current states, are all passed to the integrator.
In the integrator the aerodynamic forces and moments from the aerodynamic module, are added to the forces and moments from the inertia and thrust modules.
The solver module additionally transforms velocities from the UAV frame, to geodetic states.
The forces and moments are then used in a Newton-Euler function calculates the accelerations. These "dot" states are then projected one time-step in to the future by LSODA and new states and rates are found.
All of these states and rates, are returned as the next states for PyFDM.

6.1.5 Initialise PyFDM

When starting a PyFDM simulation all modules are initialised, and a trimming function is started. The trimming function ensures that the simulation starts in stable conditions. It uses a least squares algorithm to ensure that the initial position of surfaces and the angle of attack, does not create any forces and moments so the UAV starts in complete equilibrium. This is important because a lot of flight mechanisms is only well understood at stable conditions, and the dynamic equations of PyFDM will most likely act non-physical during high instability.

The trim module is passed the initial simulation parameters and fills out the rest of the starting states, an example of initial simulation parameters passed to the trimming module can be seen on Table 6.2.

Name	Symbol	Value
True airspeed	V_T	50.00 m/s
Latitude	ϕ	52.31°
Longitude	λ	10.56°
Altitude	h	500.00 m
Flight-path angle	γ	0.00 rad
Heading	χ	0.00 rad
Yaw angle	ψ	0.00 rad
Sampling time	Δt	0.02 s
Simulation duration	T_{sim}	10.00 s
Trim mode	—	`simple`

Table 6.2: Example of simulation parameters before initial trim.

6.1.6 Run PyFDM Step

When all of the PyFDM modules have been initialised, the simulation parameters are trimmed to initial states, PyFDM can loop through its steps. Excitations, potential controls, disturbances are added to surfaces and states, and the next set of states can be derived through the dynamics solver.

After each step, all commanded control inputs, control output surface deflections, state variables, sensor variables, and wind disturbances are logged in file and saved to disc.

6.2 Control System

For doing closed-loop control, PyFDM support an external controller module input. The flight controller currently used is called Python Flight Control Module (PyFCM).
The control system used for stabilising flight of a inner-loop PID controller and an outer-loop PID controller, is used to follow a reference trajectory of heading, height, velocity.
A diagram of the outer and inner-loop controller used for closed loop simulation can be seen on Figure 6.3.

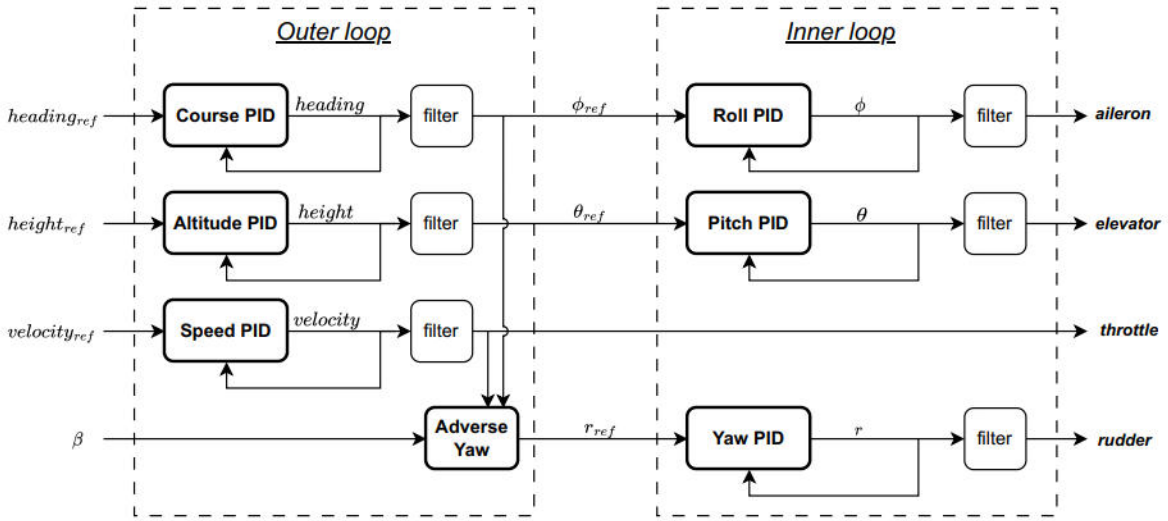


Figure 6.3: Outer and inner-loop PID controllers used for closed-loop simulation.

The PID gains are hand tuned to be relatively low for fixed wing outer/inner-loop controllers, . The outer course and inner-loop roll, and outer height and inner-loop pitch controllers are cascaded, but otherwise they assumed decoupled form each other. The only coupled relation is the adverse yaw calculation from the yaw controller, which is used to correct a slight moment induced on the yaw axis when a fixed wing plane is rolling. The hand tuned PID gains used for PyFCM, can be found in Section 3, Table 3.1. The controller is very simple, and just needs to be able to stabilise the UAV, during excitations for system identification.

6.3 Randomisation of Simulation & References

Since this is an attempt to map a surrogate model of the flight dynamics of an entire flight regime, randomisation is needed to gather representative data.
For the open-loop simulation, the state dynamics are explored through randomising the simulation parameters and then exciting the surfaces. The starting simulation parameters are randomised around the Proteus operating points with a Gaussian distribution, with the logic being that the data should reflect where the UAV actually fly in state-space, more than the extremities where it rarely would fly.

Name	Symbol	μ	Bounds	Unit / note
True airspeed	V_T	50	[40, 10]	$m\,s^{-1}$
Altitude	h	500	[200, 1000]	m
Flight-path angle	γ	0	$[-5, +5]$	deg
Heading	χ	180	[0, 360]	deg
Sampling time	Δt	0.02	fixed	s
Simulation duration	T_{sim}	30	[25, 35]	s

Table 6.3: Randomisation scheme used in `random_simulation_parameters()`. Each variable is drawn from $\mathcal{N}(\mu, \sigma)$ and clipped to its bounds, with $\sigma = (\text{upper} - \text{lower})/6$.

For closed-loop simulation the same randomised simulation parameters are used, but in addition to excitons, stabilising references for the outer-loop controller are added.

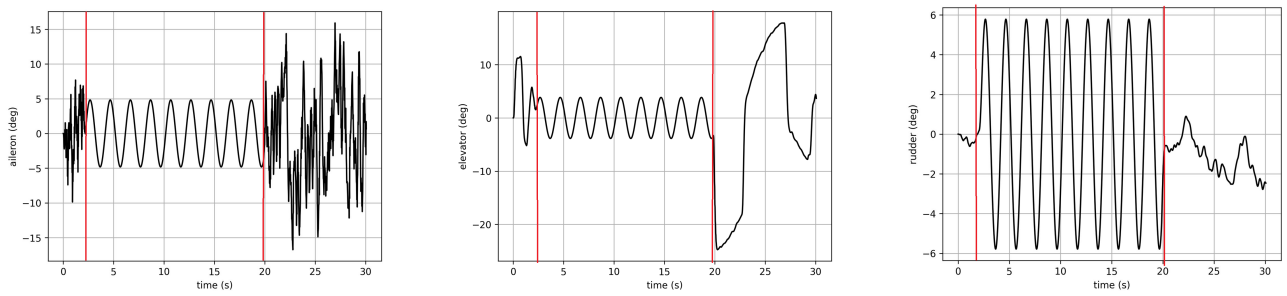
Usually when making aerodynamic models they are focused around a specific close to linear and stable operating point, such as level-flight, rolling with on aileron, yawing with one elevator, or landing. But when the goal is to identify dynamics over multiple flight envelopes, and eventually with very high actuator input morphing wing, randomisation of both flight domains and actuator excitons are needed.

6.4 Excitations

Following the excitations discussed in Section 5.4, three different excitation schemes has been made. Namely APRBS, OOMS and a mixed excitation scheme where there is a 50% chance that either OOMS or APRBS will be chosen for the flight trajectory.

OOMS

For generating OOMS excitations an excitation T length is chosen at random and a global amplitude scale s is drawn from a preset set. The base deflection for each control surface is multiplied by s for changing sine amplitudes, and two integer multipliers k define the sinusoid frequencies $f = k/T$. Each sinusoid is given a random phase and the excitation is the sum of these two sinusoids over the interval of T. The disjoint k sets ensure that the channels remain spectrally isolated and the random phases keep the peak distributed.
Figure 6.4 shows an example of OOMS, excitations used in a closed-loop simulation.

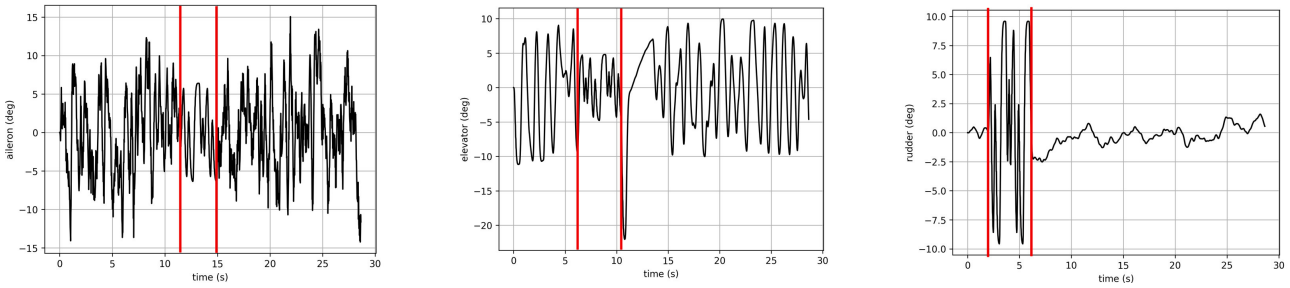


(a) OOMS Aileron excitation. **(b)** OOMS Elevator excitation. **(c)** OOMS Rudder excitation.

Figure 6.4: Side-by-side comparison of three OOMS excitations, inserted in a closed-loop simulation.

APRBS

For generating APRBS excitations, a random sample of a binary sequence is chosen. Then random amplitudes within bounds are assigned for the run, and are multiplied to the binary sequence. The frequency between bit shifts is fixed, but sequence and amplitudes are changed between each run. Figure 6.5, shows the excitations sent to the plant.



(a) APRBS Aileron excitation. **(b)** APRBS Elevator excitation. **(c)** APRBS Rudder excitation.

Figure 6.5: Side-by-side comparison of three APRBS excitations, inserted in a closed-loop simulation. Note that the square pulses are ”rounded” by the actuator delay.

Closed-Loop

As mentioned in Section 5.4, using a closed loop on the system will directly bias the data with the controller. In order to circumvent this bias, the controller is simply ”turned off” on each surface channel when excitations are emitted. If excitations were to destabilise the UAV, the controller will then re-establish after the excitations are finished. This will still mean that the controller will bias the system, but when larger excitations are happening the hope is that any larger amplitudes and different frequencies can be detected by system identification algorithms.

It should be noted that most literature recommends more sophisticated strategies to handle controller bias, as explained in Section 5.4.

6.5 SINDy System Identification

With the SINDy algorithm, a pipeline has been made that takes 20 simulation trajectories and generates the best fitted model, from a training/validation/testing spilt of 16/2/2 trajectories.
The SINDy models are compared and scored with both MSE over the difference in all 11 states, simulated 10 steps into the future. This intuitively makes sense, but it is a bad scoring metric over different domains. For example if the initial rate of change is high, the error will grow larger over the 10 steps and the identified model will score worse because of the initial state.
This is why the R^2 score is used instead. This score compares the total variance explained by the identified model over the unexplained variance, when comparing the real system to the identified model.

When setting up the SINDy system identification both a sparsity hyper parameter and a target library has to be assigned. Based on the fact that the Morphing wing has to optimise the Drag and Lift equations in Section 2.2.1 and that is directly modelled in PyFDM, the non-linearities are assumed to be of polynomial form. In real flight conditions trigonometric non-linearities are also often present, but those are not modelled in PyFDM.

This means that finding the best fitted model, for a given data-set a grid search has been used searching for the best combination of of sparsity parameter and order of library to fit, the best R^2 scoring model is chosen, the grid search goes over the following parameters:

$$\text{Polynomial Degree} \in \{1,2,3\}$$
$$\text{Sparsity Parameter} \in \{0.01,0.05,0.1,0.2,0.5,0.8\}$$

Two separate scoring methods for the identified models have been defined, namely the coefficient of determination noted R^2 and the Mean Squared Error noted MSE. These provide insight into the quality of the identified system.

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}i)^2}{\sum i = 1^N (y_i - \bar{y})^2} \tag{6.1}$$

Where:
y_i are the measured state values at sample i.
$\hat{y}_i$ are the predicted state values from the identified model.
$\bar{y}$ is the mean of all measured state values.

Equation (6.1) measures the fraction of variance explained by the identified system. A value close to 1 indicates that most of the variance in the measured data is captured by the model, while values near 0 or negative, correspond to poor predictive accuracy.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{6.2}$$

Where:
y_i are the measured state values at sample i.
$\hat{y}_i$ are the predicted state values from the identified model.
N is the total number of samples in the trajectory.

Equation (6.2) provides an absolute error metric of an entire trajectory of states. Large deviations are penalised quadratically. The R^2 and MSE scores give both a relative and absolute measure of how well the identified systems replicate the original trained system behaviour.

Since the absolute MSE scoring can deviate significantly depending on starting states and the use of un-nominalised trajectories, the R^2 scoring is used to compare models in the grid-search. Where the relative difference between models works better over different trajectories.

7 Tests

Each trained model has been simulated in 20 different runs with excitations schemes on the surfaces, compounded into a single list of trajectories.

7.1 Open-Loop System Identification

For testing the open loop system identification, 3 different excitation schemes have been used, each having 20 flight trajectories. The simulation starting states are sampled stochastically with normal distributions, described in Section 6.2.

The results of the 3 open-loop models, comparing the different excitation schemes to each other, can be seen on Table 7.1

Technique	Best config	Score	Final MSE	Final R^2	Ensembled MSE	Ensembled R^2
APRBS	degree=1, threshold=0.01	0.7829	3.973×10^{-3}	0.7645	3.554×10^{-3}	0.7644
OOMS	degree=2, threshold=0.01	0.8089	5.435×10^{-3}	0.8154	5.556×10^{-3}	0.8153
Mixed	degree=2, threshold=0.01	0.8418	4.910×10^{-3}	0.7977	4.095×10^{-3}	0.7975

Table 7.1: Open-loop model results for three excitation techniques (20 logs each).

Following the R^2 score, the open-loop OOMS excited model, explains unseen data the best. The identified open loop OOMS model, given the same starting configuration and inputs as the real system, can be seen simulated on Figure 7.1.

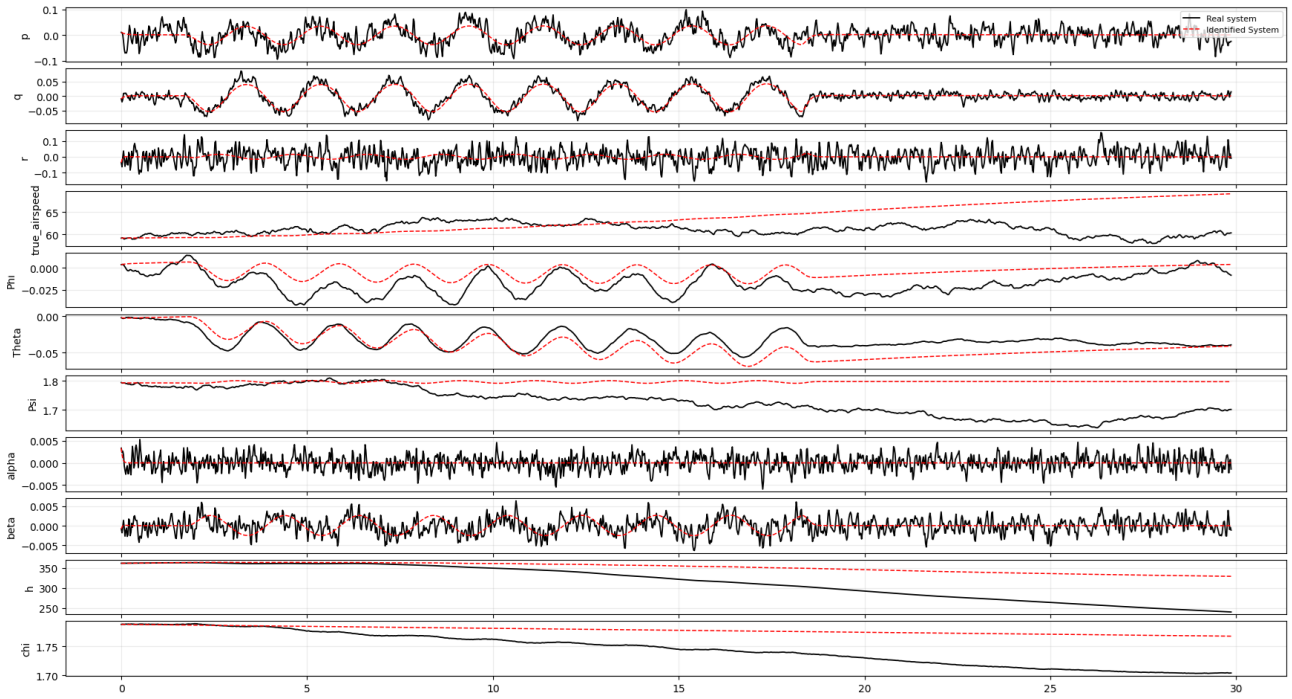


Figure 7.1: Simulation of the identified OOMS excited system.

7.2 Closed-Loop System Identification

The exact same setup is used as the open loop, except for a stabilising inner/outer-loop controller taking in height, course, and velocity. Initially tests with a stable reference has been conducted, in order to whether the system can be identified with controller inputs present.
The results of the 3 closed-loop models with APRBS, OOMS, and Mixed excitations, can be seen in Table 7.2.

Technique	Best config	Score	Final MSE	Final R^2	Ensembled MSE	Ensembled R^2
OOMS	degree=1, threshold=0.01	0.8512	2.107×10^{-3}	0.8475	1.880×10^{-3}	0.8472
Mixed	degree=1, threshold=0.01	0.8373	2.532×10^{-2}	0.8460	2.516×10^{-2}	0.8460
APRBS	degree=1, threshold=0.01	0.8148	6.283×10^{-3}	0.8228	6.611×10^{-3}	0.8226

Table 7.2: Closed-loop model results for three excitation techniques (20 logs each) with static reference height and speed.

Following the R^2 score, the best performing identified closed loop model, is the APRBS exited one. It should be noted that all closed loop models are fitted to 1. order systems. The simulation of the closed loop can be seen in Figure 7.2.

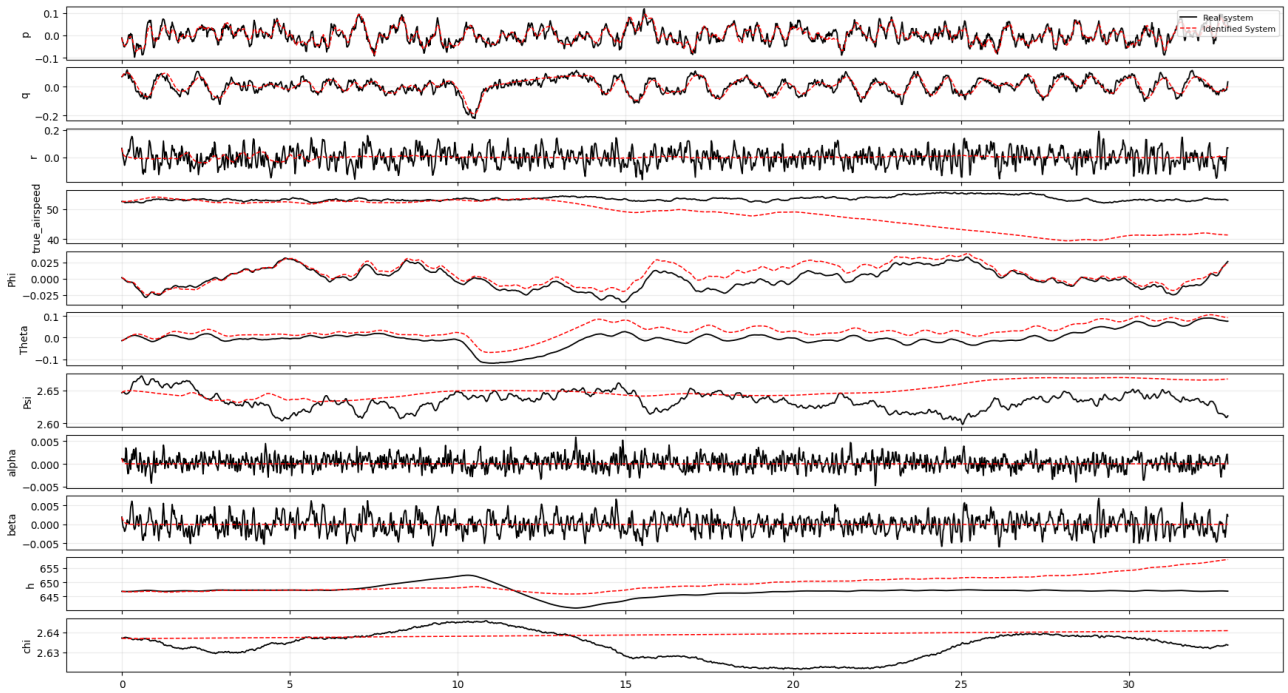


Figure 7.2: Simulation of identified system, from Closed-loop APRBS excited data.

7.3 Further Analysis

It would be of interest to explore how the controller behaves under multiple changes in altitude and velocity in the trajectories. To investigate this, a simulation consisting of 20 closed-loop flights excited by APRBS signals was conducted, where the reference trajectory changed in each run.

With an ensemble and non-ensemble model, using APRBS excitations.
Fitted to second degree, with 0.0 sparsity

$$\text{Final model:} \quad \text{Test MSE} = 5.769 \times 10^{-3}, \quad R^2 = 0.8352$$
$$\text{Ensembled model:} \quad \text{Test MSE} = 5.823 \times 10^{-3}, \quad R^2 = 0.8359$$

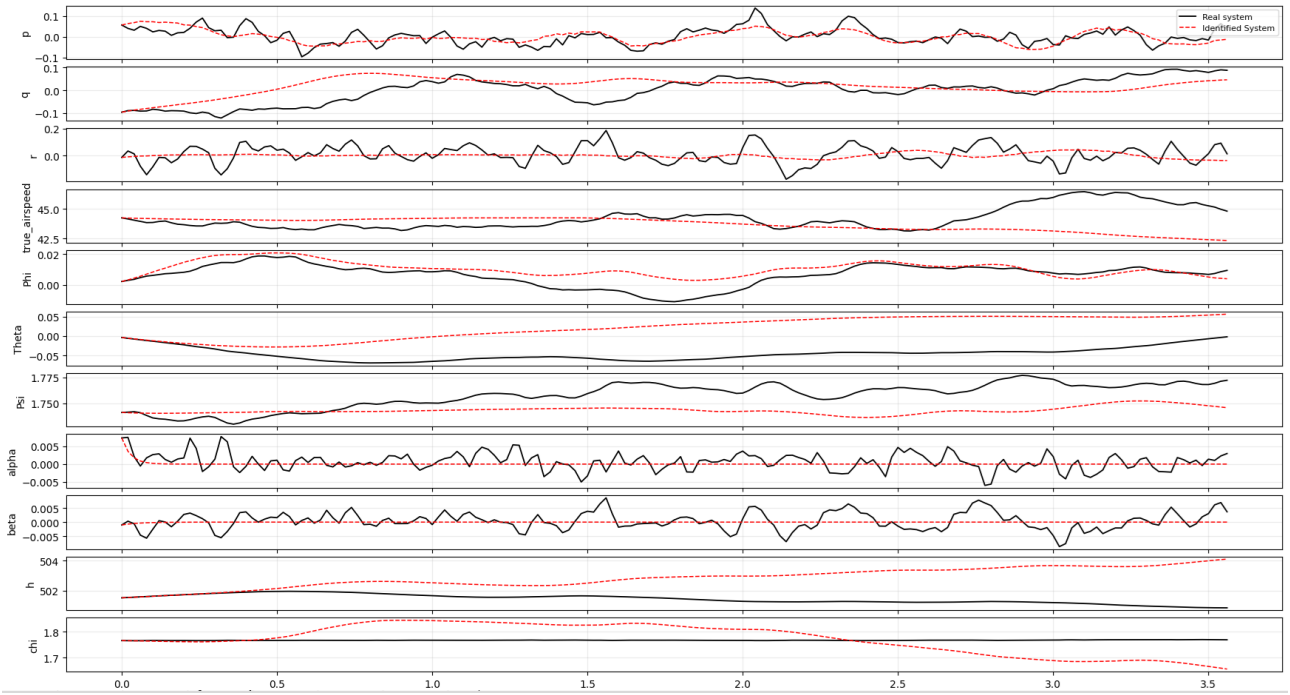


Figure 7.3: Simulation of identified closed-loop simulation where height and velocity references change. Note that only 3.5 second can be simulated before instability ensues.

The system can then be linearised into A and B matrices by taking the Jacobian at the operating point (x_{50}, u_{50}). The corresponding eigenvalues of A are listed in Figure 7.4, which indicate that several modes lie close to the unit circle.

$$\begin{aligned}
\lambda[0] &= 1.0000 & |\lambda[0]| &= 1.0000 \\
\lambda[1] &= 0.1596 & |\lambda[1]| &= 0.1596 \\
\lambda[2] &= 0.4930 & |\lambda[2]| &= 0.4930 \\
\lambda[3] &= 0.4904 & |\lambda[3]| &= 0.4904 \\
\lambda[4] &= 0.9262 & |\lambda[4]| &= 0.9262 \\
\lambda[5] &= 0.7000 & |\lambda[5]| &= 0.7000 \\
\lambda[6] &= 0.9924 & |\lambda[6]| &= 0.9924 \\
\lambda[7] &= 0.9996 & |\lambda[7]| &= 0.9996 \\
\lambda[8] &= 0.9999 & |\lambda[8]| &= 0.9999 \\
\lambda[9] &= 1.0008 & |\lambda[9]| &= 1.0008 \\
\lambda[10] &= 1.0007 & |\lambda[10]| &= 1.0007
\end{aligned}$$

Figure 7.4: Eigenvalues of the linearised system matrix A at the operating point (x_{50}, u_{50}). Values close to $|\lambda| = 1$ indicate near-marginal stability.

The ensemble model is able to simulate in 30 seconds without having terms explode, as seen on Figure 7.5.

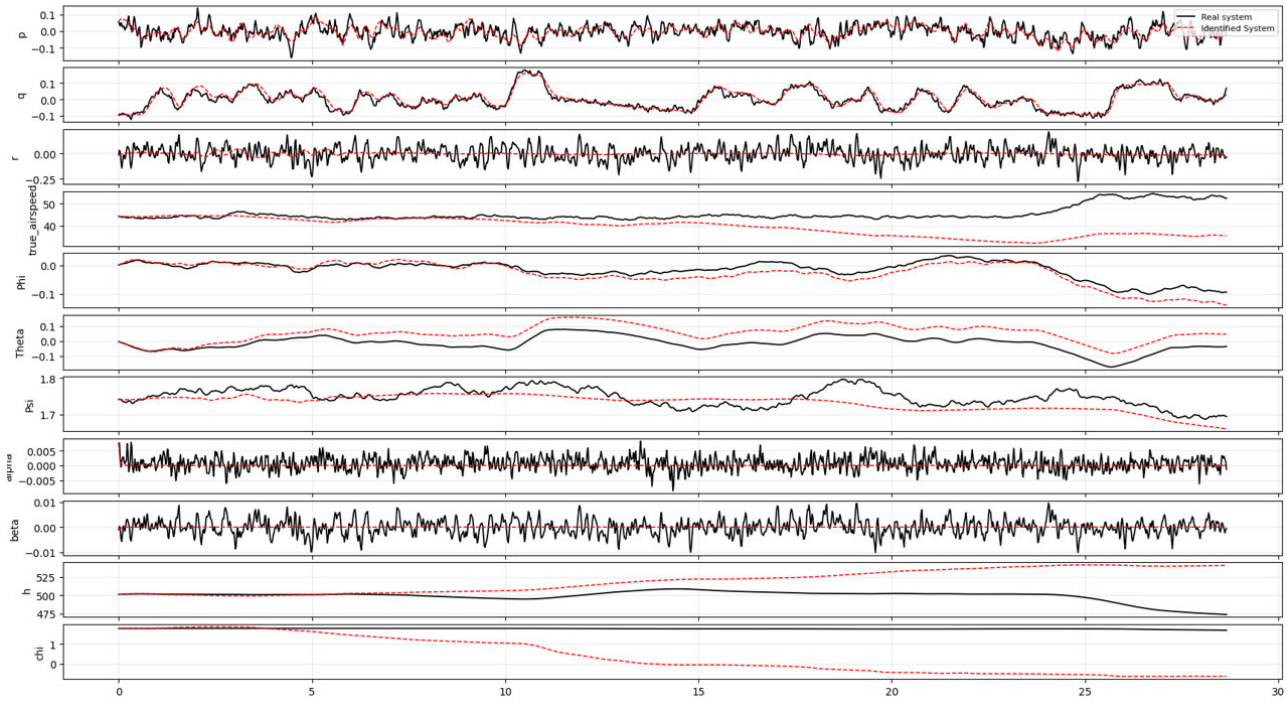
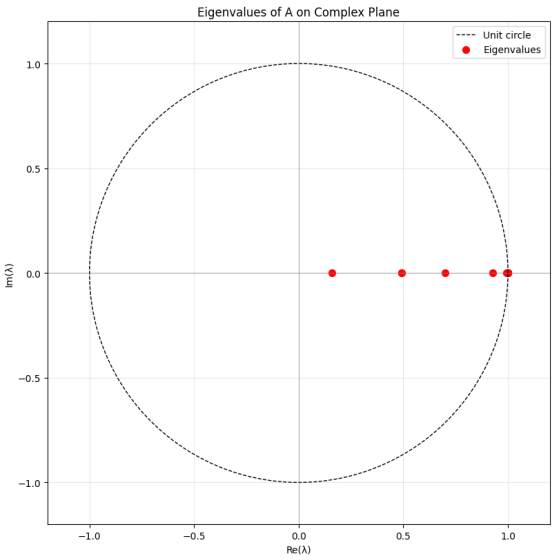


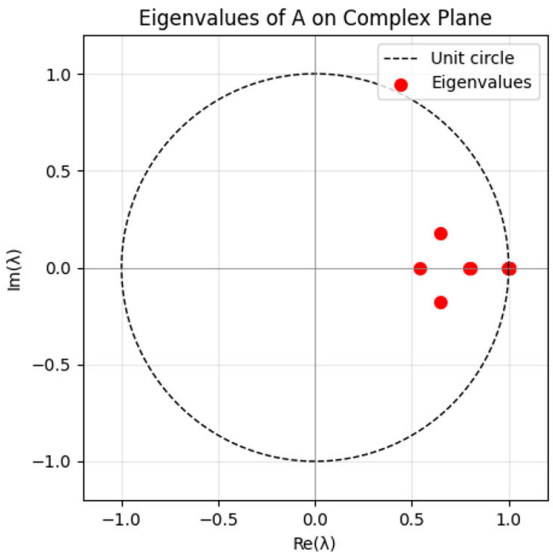
Figure 7.5: Simulation of identified closed-loop simulation where height and velocity references change. Note that the ensemble model allows for simulation.

It should be noted that the Eigen-values are still slightly unstable of the ensemble model, but no oscillation are present indicating that the controller might over-damp the system.

It is also interesting to note that there no imaginary parts in the linearised stability plots of the closed-loop, but imaginary part are present in the open-loop linearised Eigen-values, they can be seen on Figure 7.6.



(a) Linearised Closed-Loop changing controller reference, with APRBS excitation.



(b) Open-Loop Mixed excited stability plot, with slight instability and oscillating parts.

Figure 7.6: Side-by-side comparison of Open and Closed-Loop discrete time Eigen-values, both linearised.

Additional experiments were carried out to test the identified Open-Loop models under square reference changes in different flight regimes. Figures 7.7 and 7.8 show the system responses at low altitude and speed ($h = 100, v = 50$) and at high altitude and speed ($h = 1000, v = 200$).
The following tests show the Open-Loop response of the Mixed excitation model when subjected to square reference inputs. The model is initialised with the same state as the PyFDM simulation and is propagated forward under identical square excitations, applied at different times to each control surface. After the excitation sequence, the model is further propagated without external input. As shown in Figures 7.7 and 7.8, the identified model remains consistent with the reference behaviour, and the errors do not diverge significantly across time. This result supports that the mixed excitation model can generalise across different flight regimes.

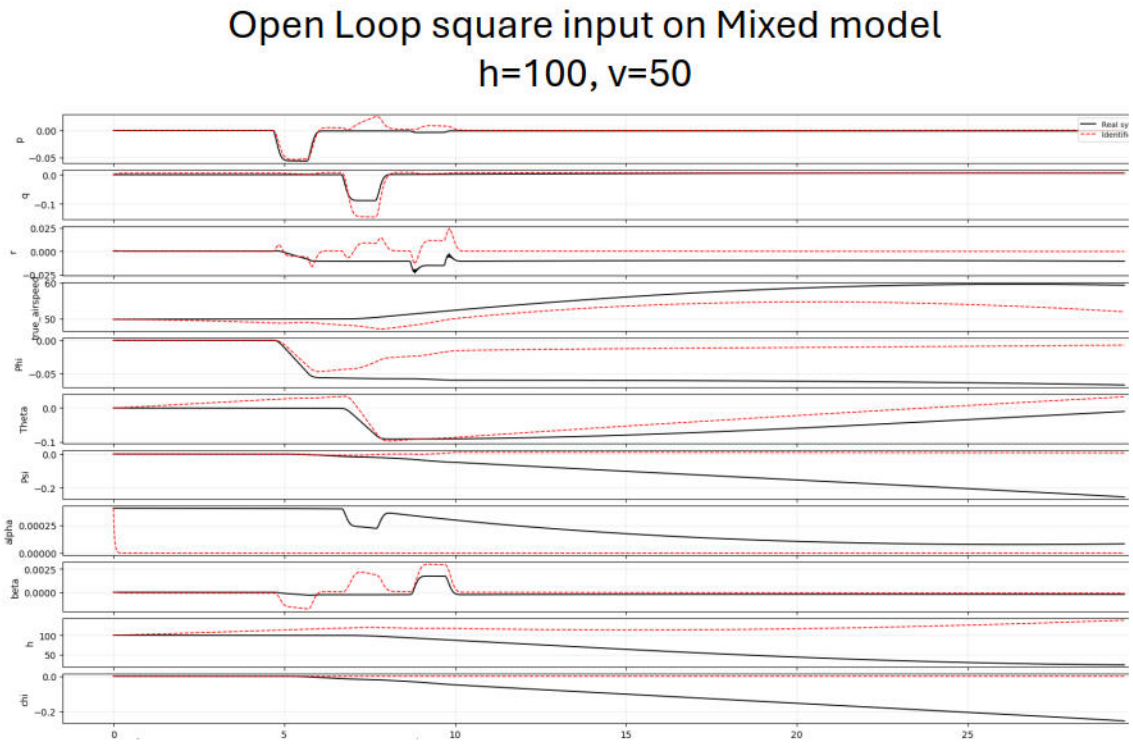


Figure 7.7: Open-Loop mixed model tested with square input at low altitude and velocity ($h = 100, v = 50$).

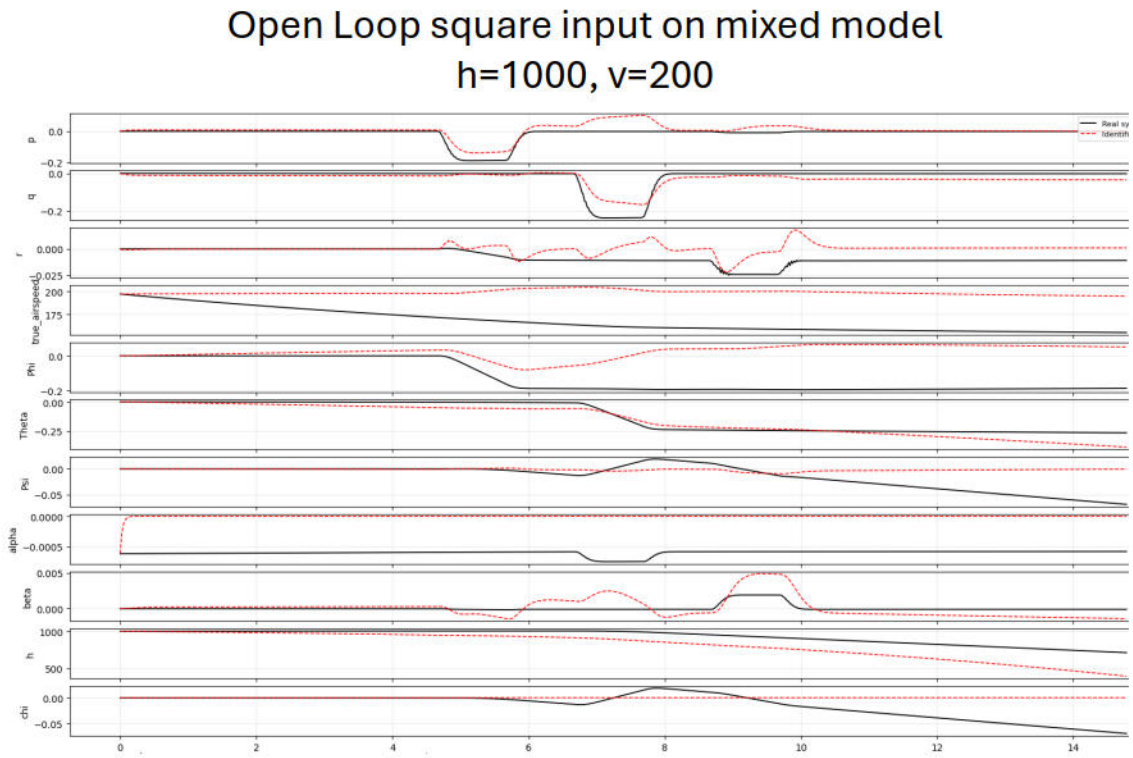


Figure 7.8: Open-Loop mixed model tested with square input at high altitude and velocity ($h = 1000, v = 200$).

8 Discussion

This chapter will discuss the test results and the general implementation of both system identification and excitations.

8.1 Open-Loop

For the Open-Loop system identification the best fitted excitation technique is the OOMS, fitted to a second degree polynomial with the lowest threshold. On Table 7.1 this result might seem great, and 85% variance explained by the identified model could be a good result in itself, especially as models explaining second degree dynamics is the eventual goal. But when looking at the open-loop simulation on Figure 7.1, it becomes clear that the identified model is completely over-fit to the excitation scheme which is also present in the test data.

What should have been done instead, was to use completely uncorrelated test and validation data. What was intended to be a system identification model of MIMO non-linear dynamics, has unfortunately turned out to be a model fitting of excitation signals instead.

8.2 Closed-Loop

For the same reasons as in the Open-Loop excitation test, the Closed-Loop results on Table 7.2 are also inconclusive. This time a stabilising controller minimising the error to a static reference, so it might seem like the APRBS excitation type is the most similar to the first order dominated controller. It can be seen on Table 7.2, that even the OOMS excitation that otherwise over-fitted the open-loop controller to a second order system, now also is of first order in the closed-loop. This might indicate that the controller bias is high enough to dim out any second order fitting to the sinusoids of OOMS.

8.3 Open-Loop Changing Reference

During further analysis when testing the Open-Loop mixed excitation model in different flight regimes, a clearly defined response can be seen on most states when compared to free flight in PyFDM given the same response. This is true in widely differing flight regimes, and suggests that SINDy models can learn flight responses.

In both cases of high-attitude/high-velocity 7.8, and low-attitude/low-velocity 7.7, a clear correspondence can be observed between the model response and the free-flight behaviour in PyFDM. This suggests that the SINDy models are able to capture relevant Open-Loop dynamics across different operating points, even when the excitation and reference profiles differ from those used in training.

8.4 General Remarks

For trimmed open-loop flights the model seems to fit to the excitation used, and closed-loop static reference flights seems to always fit to the controller bias.

Interestingly enough if the controller reference is changed once during simulation the SINDy model did fit a closed-loop model to the second degree, but it caused the simulation to become unstable. This means all that the goal of describing broad dynamics, in a non-linear MIMO system failed, because of over-fitting when identifying the system. But for Open-Loop SINDy models tested with new ”un-seen” excitations, a desirable response can be observed compared to a PyFDM trajectory given the same input excitation. This indicates that given enough training data and doing Open-Loop MIMO non-linear excitation schemes, it could be possible to do an accurate system identification of non-linear dynamics over different flight-regimes.

8.5 Further Work

For further work there should be more focus on the machine learning part of SINDy and training/validation data should be sampled from completely different envelopes, with **no** dominating excitations. If closed-loop system identification is needed, strategies for handling controller bias are strictly needed. The bias is too large even when the controller is momentarily turned off, during excitations. This begs the question whether closed-loop system identification even is needed for the MorphAIR project. Since real flight tests with the morphing wing cannot be safely conducted without an identified model from simulation.

8.6 Conclusion

This thesis investigated non-linear system identification of a morphing fixed-wing Unmanned Aerial Vehicle (UAV) using a simulation-based pipeline. The main objective of robustly identifying broad non-linear Multiple-Input Multiple-Output (MIMO) dynamics could not be fully achieved due to overfitting, especially in Closed-Loop tests where controller bias dominated the response.

Nevertheless, the work demonstrates several important outcomes. Limited non-linear MIMO dynamics were captured in Open-Loop simulations, and randomised Orthogonal phase-Optimized Multi-Sines (OOMS) excitations proved effective for non-linear identification. A full simulation and excitation pipeline was implemented, capable of handling both Open- and Closed-Loop flights. Additional Open-Loop testing against the Python Flight Dynamics Module (PyFDM) system confirmed that the mixed excitation model can reproduce desired dynamics without diverging, suggesting that an Open-Loop model may be feasible through SINDy- based system identification.

Future work should focus on generating more independent validation data to mitigate overfitting, addressing controller bias for closed-loop identification, and extending the approach with real-world flight data.

A Appendix

This is section is based of the Prior internship report of 9th semester, and gives a nesseary understanding of fixed-wing aerodynamics. [15]

A.1 Fundamental Aerodynamics of Fixed-Wing Aircraft

To support an understanding of fixed-wing flight, it is necessary to present a simplified overview of the primary aerodynamic forces. Figure A.1 outlines common aerospace terminology relevant to conventional fixed-wing aircraft:

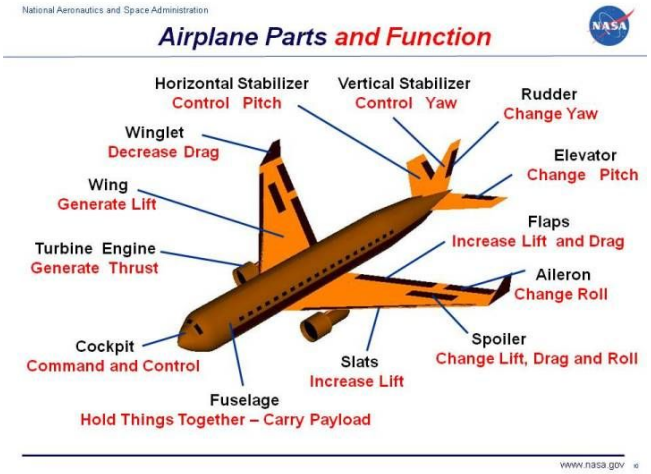


Figure A.1: Terminology commonly used in aerospace engineering related to aircraft structure.

The study of fixed-wing aerodynamics concerns how air flows around an aircraft's wings (or airfoils), generating the key forces required for flight. The four fundamental forces involved are Lift, Drag, Weight, and Thrust. Lift acts upwards and counteracts the pull of gravity, enabling flight. It is generated when air moves more quickly over the upper surface of a curved wing than along its flatter underside, producing a pressure difference. This phenomenon is explained by Bernoulli's principle in combination with Newton's third law. The formula for lift is presented in Equation A.1.

$$L = \frac{1}{2}\rho V^2 S C_L \tag{A.1}$$

Where:

L represents the lift force acting perpendicular to the oncoming airflow,
ρ is the air density,
V is the airspeed relative to the wing,
S is the surface area of the wing,
C_L denotes the lift coefficient, which varies depending on the wing's geometry and angle of attack.

In classical flight mechanics, these aerodynamic coefficients are typically stored in static lookup tables for ease of computation.
In Figure A.2 an illustration describing the laminar flow separation can be seen

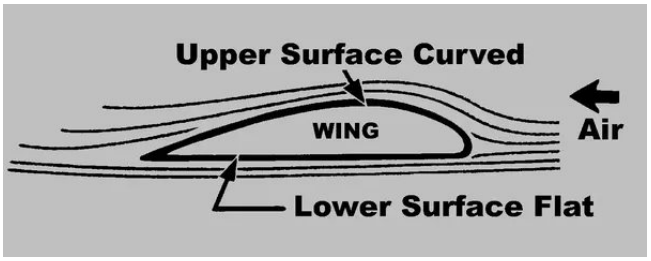


Figure A.2: Illustration of flow separation from the upper surface of a curved wing.

Lift is the counterforce to the airplane's weight which is induced by gravity.
Drag is the resistance experienced as air moves around the aircraft, acting in opposition to the direction of travel. Its calculation, shown in Equation A.1, is structurally similar to the lift equation.

$$D = \frac{1}{2}\rho V^2 S C_D \tag{A.2}$$

Where:

D stands for the drag force, aligned parallel aligned with the relative airflow,
ρ, V, and S retain the same meanings as above,
C_D is the drag coefficient, which is influenced by the aircraft's wing area and surface shape.

Thrust is the forward-driving force generated by the aircraft's propulsion system, overcoming drag and sustaining forward motion. Balanced forces results in steady, level flight, illustrated in Figure A.3.

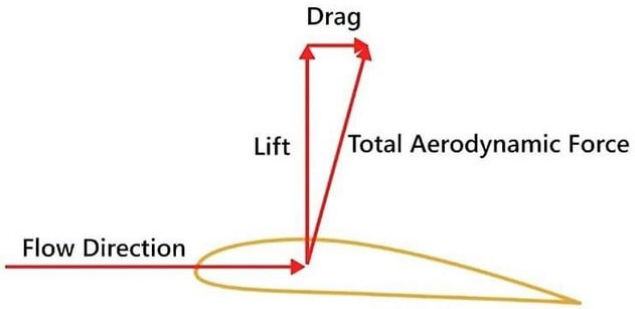


Figure A.3: Depiction of force equilibrium in level flight.

The airflow interacting with the airfoil generates lift, and the angle relative to the oncoming wind is known as the Angle of Attack (AOA). Both lift and drag are produced, and depending on the wing design, small AOAs can generate additional lift; however if the AOA gets too high, the wings laminar flow is reduced and the drag increases significantly,, which can cause the aircraft to lose altitude. This behaviour is depicted in Figure A.4

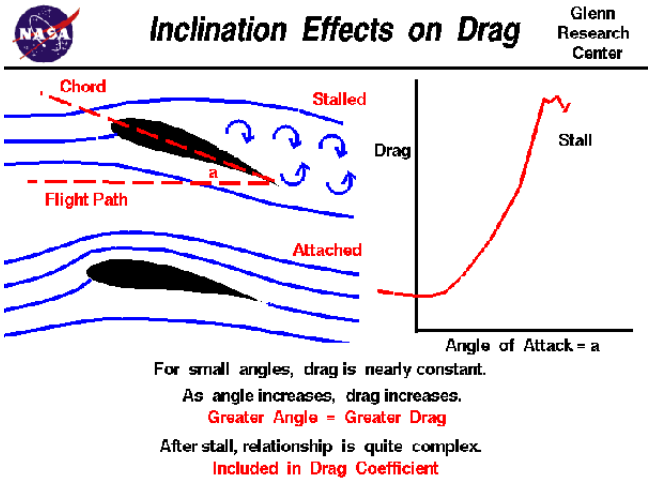


Figure A.4: NASA visualisation showing how increased AOA enhances lift until stall.

A fixed-wing aircraft is controlled by adjusting its moments. Angling a surface against the airflow on one side creates a reaction force, generating a moment. This process is comparable to the concept shown in Figure A.4, except it involves control surfaces.
The unmanned aerial vehicle UAV utilises four control inputs, for in-flight adjustments: the two tail elevators and the tail rudder, the two wing ailerons, and engine generating thrust, as seen in Figure A.5

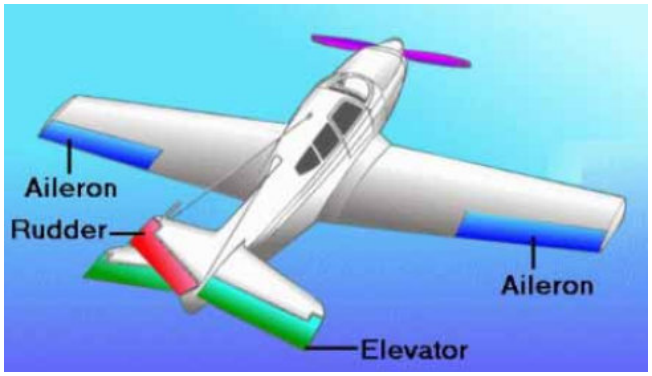


Figure A.5: Illustration of control surfaces on a fixed-wing aircraft

The ailerons adjust the roll movement of the plane. The elevators adjust the pitch movement of the plane. The rudder adjusts the yaw movement of the plane.

From the wing's frame of reference, the aircraft itself appears stationary while the gas flows past it. This assumption is inherent in aerodynamic calculations, with actual aircraft velocity derived afterwards. Another simplification is to treat the Earth as flat for these calculations; only after computing the travelled distance is the aircraft's position projected onto a spherical Earth model, yielding geographic coordinates such as latitude and longitude.
This represents a very rudimentary outline of aerodynamics for a non-morphing wing. In reality, the aerodynamic behaviour of both Proteus with conventual wings and particularly Proteus with morphing wings, is far more complex. Nevertheless, since my work has not involved direct aerodynamic analysis, this level of understanding is considered sufficient for the current context.

A.2 Aircraft Coefficients

Aircraft Geometry and Inertia

Parameter	Value	Unit
S_{ref}	1.41	m^2
C_{ref}	0.50	m
B_{ref}	2.99	m
Mass	60	kg
I_{xx}	3.5177	kg·m^2
I_{yy}	6.74718	kg·m^2
I_{zz}	9.21608	kg·m^2
I_{xy}	0	kg·m^2
I_{xz}	0.6854	kg·m^2
I_{yz}	0	kg·m^2

Table A.1: Geometry and inertia parameters of the aircraft.

Aerodynamic Force Coefficients

Coefficient	Value	Notes
e (Oswald factor)	0.95	Span efficiency
CL_0	0.17473	Zero-angle lift
CL_α	4.64545	AOA lift slope
CL_q	6.80827	Pitch rate effect
CD_{vis}	0.02	Viscous drag
CD_{ind}	0.00177	Induced drag
CY_β	-0.2667	Sideslip force
CY_r	0.1892	Yaw rate effect

Table A.2: Aerodynamic force coefficients.

Aerodynamic Moment Coefficients

Coefficient	Value	Description
Cl_p	-0.4108	Roll damping
Cl_r	0.0671	Yaw-to-roll coupling
Cl_ξ	-0.0039	Aileron effect
Cm_α	-0.5552	Pitch stiffness
Cm_q	-7.2203	Pitch damping
Cm_η	-0.0176	Elevator effect
Cn_β	0.0759	Directional stability
Cn_r	-0.0612	Yaw damping
Cn_ζ	-0.0009	Rudder effect

Table A.3: Aerodynamic moment coefficients for roll, pitch, and yaw.

A.3 External Links

Video of Conventional Wing Flight Test

Google Drive: Video of the First Flight Test of Proteus with the Conventional Wing

Video of Morphing Wing on Proteus

Google Drive: Video of the Morphing Wing Moving on Stationary Proteus

A.4 PyFDM Logged States

This is a representation of the amount of states logged in PyFDM. Most of the values do not directly describe the dynamics, and can be omitted.

All logged Values from PyFDM

Table A.4: Table of all 112 states, including both simulation states, sensor values, control-input & -output, and wind disturbances. This shows the almost comical amount of states present in PyFDM,

Most of these stats are redundant, and system identification is only done on the control outputs, and a subset of the sensor values. A lot of the states such as component velocities or rates, are represented in different frames. This means its possible to reduce the states of interest to following Table A.5.

Reduced States from PyFDM

time	control_outputs				sensor_values																
time	aileron	elevator	throttle	rudder	p	q	r	x	y	z	u_gs	v_gs	w_gs	true_airspeed	Phi	Theta	Psi	alpha	beta	h	chi	gamma
0	0	-5,2E-06	0,000109	0	0,022421	-0,05133	0,120008	0,999513	-0,00034	-0,0041	49,9657	-0,02592	-0,25309	49,52087	0,000259	-0,00084	0,001695	0,006814	0,00635	500	0	0
0,02	0	-0,00016	0,003295	0	0,038437	-0,05191	0,23462	1,998714	-0,00126	-0,00766	49,95409	-0,05353	-0,13604	49,13915	0,00081	-0,00179	0,005956	-0,00131	0,009877	500,0041	-0,00052	0,005065
0,04	0	-0,00055	0,01146	0	0,007838	-0,05884	0,118051	2,997676	-0,00219	-0,00936	49,94186	-0,03849	-0,04845	49,3567	0,001091	-0,0029	0,009505	-0,00056	0,00437	500,0077	-0,00107	0,002723
0,06	0	-0,00171	0,035419	0	-0,00288	-0,07863	-0,08175	3,996263	-0,00251	-0,01222	49,92199	-0,00262	-0,14702	49,704	0,001305	-0,00466	0,009306	0,003871	-0,0035	500,0094	-0,00077	0,00097

Table A.5: Table of the reduced 23 states, time, & inputs. The yellow states represents the values needed for most simpler controllers. The green states might also be needed for more complicated model based flight controllers.

Bibliography

[1] Hannah Ritchie. "What share of global CO_2emissions come from aviation?" In: *Our World in Data* (2024). https://ourworldindata.org/global-aviation-emissions.

[2] Milan Klöwer et al. "Quantifying aviation's contribution to global warming". In: *Environmental Research Letters* 16.10 (2021), p. 104027.

[3] Daochun Li et al. "A review of modelling and analysis of morphing wings". In: *Progress in Aerospace Sciences* 100 (2018), pp. 46–62.

[4] Jan Tikalsky and HP Monner. *Morphing Compliant Trailingedge Skin Concept, **Preprint***. Accessed: 2025-01-01. 2024. DOI: https://doi.org/10.21203/rs.3.rs-3996937/v1.

[5] Zhoujie Lyu and Joaquim RRA Martins. "Aerodynamic shape optimization of an adaptive morphing trailing-edge wing". In: *Journal of Aircraft* 52.6 (2015), pp. 1951–1970.

[6] Alexander M Pankonien and Daniel J Inman. "Aerodynamic performance of a spanwise morphing trailing edge concept". In: *25th international conference on adaptive structures and technologies*. 2014.

[7] NASA Aeronautics, Kamlet, Matt. *NASA Tests New Alloy to Fold Wings in Flight*. Accessed: 2025-04-03. 2025.
URL: https://www.nasa.gov/aeronautics/nasa-tests-new-alloy-to-fold-wings-in-flight/.

[8] Airbus. *How the Albatross is Inspiring Next Generation of Aircraft Wings*. Accessed: 2025-04-03. 2019.
URL: https://www.airbus.com/en/newsroom/press-releases/2019-06-how-the-albatross-is-inspiring-next-generation-of-aircraft-wings.

[9] Netherlands Aerospace Centre (NLR). *R&D case: Scaled Flight Demonstrator - SFD*. Accessed: 2025-04-03.
URL: https://www.nlr.org/newsroom/case/scaled-flight-demonstrator-sfd/.

[10] Netherlands Aerospace Centre. *Setback for research project with electric scale model*. Accessed: 2025-05-20. 2023.
URL: https://www.nlr.org/newsroom/nieuws/setback-for-research-project-with-electric-scale-model/.

[11] Wikipedia contributors. *Parasitic drag – Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/Parasitic_drag. [Accessed: 11-Jun-2025]. 2024.
URL: https://en.wikipedia.org/wiki/Parasitic_drag.

[12] Wikipedia contributors. *Royal Air Force — Wikipedia, den frie encyklopædi*. https://da.wikipedia.org/wiki/Royal_Air_Force. [Accessed: 11-Jun-2025]. 2024.
URL: https://da.wikipedia.org/wiki/Royal_Air_Force.

[13] M. I. Ali et al. "A Systematic Review of Morphing Wing in Aviation Industry". In: *International Journal of Advanced Science and Technology (IJAST)* 31.1 (2022).
URL: https://www.ijast.org/issues/vm01is01/article8.pdf.

[14] "Flight dynamics principles; a linear systems approach to aircraft stability and control, 3d ed". eng. In: *Reference & Research Book News* 28.3 (2013). ISSN: 0887-3763.

[15] Frederik Saldern Nielsen. "RL Control of Morphing Trailing Edge Fixed Wing UAV". Unpublished 9th semester thesis. 2025.

[16] L Nugroho and R Akmeliawati. “Comparison of black-grey-white box approach in system identification of a flight vehicle”. eng. In: *Journal of physics. Conference series* 1130.1 (2018), pp. 12024–. ISSN: 1742-6588.

[17] Aziida Nanyonga et al. “Explainable Supervised Learning Models for Aviation Predictions in Australia”. In: *Aerospace* 12.3 (2025). ISSN: 2226-4310. DOI: 10.3390/aerospace12030223. URL: https://www.mdpi.com/2226-4310/12/3/223.

[18] A. Rahideh and M.H. Shaheed. “Dynamic modelling of a twin rotor MIMO system using grey box approach”. eng. In: *2008 5th International Symposium on Mechatronics and Its Applications*. IEEE, 2008, pp. 1–6. ISBN: 1424420334.

[19] Aoxiang Dong, Andrew Starr, and Yifan Zhao. “Neural network-based parametric system identification: a review”. In: *International Journal of Systems Science* 54.13 (2023), pp. 2676–2688. DOI: 10.1080/00207721.2023.2241957. URL: https://doi.org/10.1080/00207721.2023.2241957.

[20] Stephen A. Billings. *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains*. Chichester, UK: John Wiley & Sons, 2013. ISBN: 9781119943594. DOI: 10.1002/9781118535561.

[21] J. B. Hoagg et al. “Sequential Multisine Excitation Signals for System Identification of Large Space Structures”. In: *Proceedings of the American Control Conference*. Minneapolis, MN, USA, 2006, pp. 418–423.

[22] Antonín Novák et al. “Nonlinear System Identification Using Exponential Swept-Sine Signal”. In: *IEEE Transactions on Instrumentation and Measurement* 59.8 (2010), pp. 2220–2229. DOI: 10.1109/TIM.2010.2047144.

[23] Michael Deflorian and Susanne Zaglauer. “Design of experiments for nonlinear dynamic system identification”. In: *IFAC Proceedings Volumes* 44.1 (2011), pp. 13179–13184.

[24] Eugene A. Morelli. “Optimal Input Design for Aircraft Stability and Control Flight Testing”. In: *Journal of Optimization Theory and Applications* 189 (2021), pp. 1–29. DOI: 10.1007/s10957-021-01942-7.

[25] Ramkrishna Ghosh, Jari Böling, and Kurt Erik Häggblom. “Evaluation of Input Designs for MIMO System Identification Using Subspace Identification”. In: *18th Nordic Process Control Workshop*. Espoo, Finland, 2013, pp. 339–344.

[26] Jared A. Grauer and Matthew J. Boucher. “Aircraft System Identification from Multisine Inputs and Frequency Responses”. In: *AIAA SciTech Forum*. Paper 2020–0287. 2020. DOI: 10.2514/6.2020-0287.

[27] Johan Schoukens and Lennart Ljung. “Nonlinear system identification: A user-oriented road map”. In: *IEEE Control Systems Magazine* 39.6 (2019), pp. 28–99.