UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Trabajo Fin de Máster del
Máster de Investigación en Inteligencia Artificial

Artificial intelligence methods for identification of aerodynamic coefficients from flight data

Fidel Echevarría Corrales

Dirigido por: José Manuel Cuadra Troncoso

Enrique J. Carmona Suárez

Curso: 2020-2021: 3ª Convocatoria

Acknowledgements

Throughout the writing of this dissertation I have received a great deal of support and assistance.

I am extremely thankful to my supervisors, Associate Professor José Manuel Cuadra Troncoso and Associate Professor Enrique J. Carmona Suárez. Their expertise was invaluable in formulating the research questions and methodology, and their insightful feedback pushed me to sharpen my thinking and brought my work to a higher level. I would also like to acknowledge Associate Professor Olga C. Santos for her valuable counsel concerning research and publication methodology. My sincere gratitude to my supervisor at UAV Navigation, Miguel Ángel de Frutos Carro. Thank you for your patient support and for the opportunities I was given to further my research.

I am grateful for my parents and siblings, whose constant love and support keep me motivated and confident. Finally, I owe my deepest gratitude to my dear partner, Rocío, and to my extended family. I am forever thankful for their love and support throughout the entire making of this thesis and every day.

Abstract

In engineering there is an essential necessity to model the systems surrounding us. Conceptual models provide information about the dynamics of the physical systems in study, and allow predictions of their past or future states given an initial condition. Aircraft dynamic modelling is an essential process for many tasks in the aviation industry, including development of effective automatic flight control systems, precise state estimation during flight, or simulator development for pilot training, among many others. Aerodynamic coefficient identification is a key part of aircraft dynamic modelling. Some methods which have traditionally been applied for this task include wind tunnel testing, computational fluid dynamics, or frequency-response analysis in flight test data. In the recent years, arising from the exponential growth in available computing capacity, evolutionary algorithms and deep learning techniques have flourished and taken a very important role in many areas, including system identification. This work focuses on the application of these methods for identification of aircraft aerodynamic coefficients from flight data. Results demonstrate promising identification capabilities, with low associated configuration costs.

Resumen

En el ámbito de la ingeniería existe una necesidad esencial de modelizar los sistemas que nos rodean. Los modelos conceptuales proporcionan información sobre la dinámica de los sistemas físicos en estudio, y permiten realizar predicciones de los estados pasados o futuros dada una condición inicial. El proceso de modelización de la aerodinámica de aeronaves es esencial para muchos sectores de la industria aeroespacial, incluyendo el desarrollo de sistemas automáticos de control efectivos, estimación precisa de estados durante el vuelo, o el desarrollo de simuladores para entrenamiento de pilotos, entre otras. La identificación de coeficientes aerodinámicos es un proceso muy importante en la modelización de la dinámica de aeronaves. Entre los métodos tradicionalmente aplicados a este proceso se podrían mencionar técnicas como ensayos en túnel de viento, dinámica de fluidos computacional, o el análisis de la respuesta en el dominio de la frecuencia a partir de datos de vuelo. En las últimas décadas, fruto del crecimiento exponencial de la capacidad computacional disponible, los algoritmos evolutivos y técnicas de aprendizaje profundo han prosperado y tomado un papel muy importante en muchas áreas, incluyendo la identificación de sistemas. Este trabajo se centra en la aplicación de estos métodos para la identificación de coeficientes aerodinámicos de aeronaves a partir de datos de vuelo. Los resultados demuestran una prometedora capacidad de identificación con bajos costes de configuración asociados.

Contents

Nomenclature

AES Average number of evaluations to solution, page 43

ANN Artificial neural network, page 8

CFD Computational fluid dynamics, page 6

CIFER Comprehensive identification from frequency responses, page 7

CMA-ES Covariance matrix adaptation evolution strategy, page 28

CNN Convolutional neural network, page 9

CSA Cumulative step length adaptation, page 29

CSV Comma-separated values, page 38

DATCOM Stability and control data compendium, page 5

DE Differential evolution, page 66

DNN Deep neural network, page 35

DoF Degree of freedom, page 16

DSR Distance success rate, page 43

EKF Extended Kalman filter, page 7

ES Evolution strategy, page 28

FDM Flight dynamics model, page 37

FFNN Feed forward neural network, page 9

FG FlightGear, page 37

FSR Fitness success rate, page 43

FT Fourier transform, page 7

GD Gradient descent, page 35

GP Genetic programming, page 10

GRU Gated recurrent unit, page 9

GUI Graphical user interface, page 38

LPF Low-pass filter, page 24

LSTM Long short-term memory, page 9

MA Memetic algorithm, page 66

MBF Mean best fitness, page 42

MPC Model predictive control, page 3

MRAC Model reference adaptive control, page 3

MSD Mean solution distance, page 42

NED North, East, Down, page 17

PF Particle filter, page 7

RNN Recurrent neural network, page 9

SINDy Sparse identification of nonlinear dynamics, page 8

TCN Temporal convolutional network, page 9

UDP User datagram protocol, page 37

UKF Unscented Kalman filter, page 7

USAF United States Air Force, page 5

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides an introductory overview of the problem of system identification, particularly applied to aircraft dynamic modelling. Subsequently, the problem to solve is described, and the motivation for performing this work is stated. In addition, the chapter includes a general overview of the state of the art in aircraft dynamic modelling, and the research hypothesis and objectives of this work are postulated. Finally, the scope and limitations of the thesis are provided.

1.1 System identification

A model is a testable representation of a system (Dubois, 2018). Model building is an essential process in engineering, since it allows making predictions about a system's behavior, and study the effects of different components (figure 1.1). Mathematical models are the basis on which the field of numerical simulation is constructed. This field has been around for many centuries, but has undergone a rapid escalation since the 1940s, given the advent of numerical computing. Mathematical models are not only essential for optimizing a system's initial design, but also for iterative improvement of the different components (Stevens et al., 2003).

It is very common in engineering to have knowledge about two of the three components in a mathematical model. The process of inferring a mathematical model to describe a system from measured data is usually referred to as *system identification* (Söderström and Stoica, 1989). The field of iden-

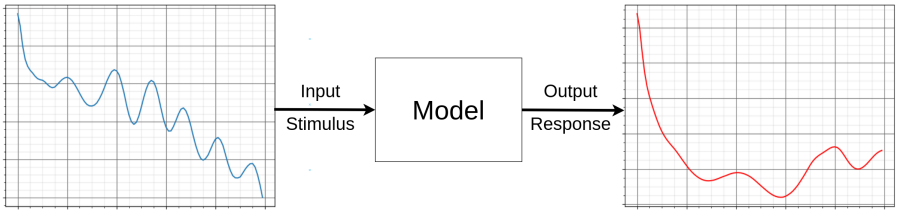

Figure 1.1: Model diagram.

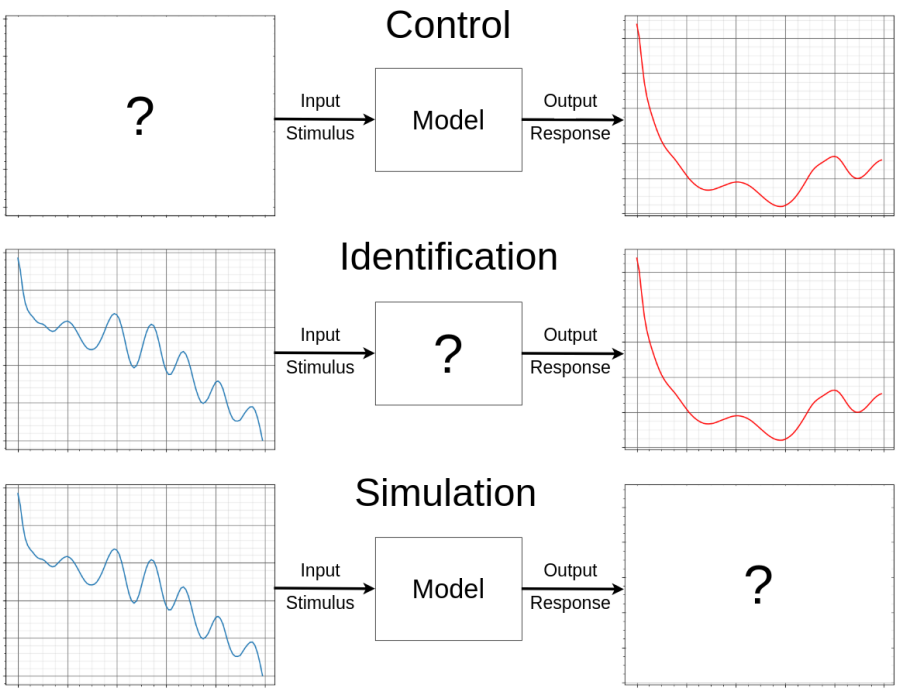

Figure 1.2: Problems of control (top), identification (center) and simulation (bottom).

tification (inferring a model given stimulus and response) is complementary to the fields of simulation (inferring a response given a stimulus and a model), and control (inferring a stimulus given a model and a response) (Eiben and Smith, 2015). This concept is schematized in figure 1.2.

A common way to classify identification problems in literature is according to the level of prior existing knowledge about the system. In this regard, we can distinguish between:

- White box models. The system is perfectly understood and can be fully described with a known mathematical model. There is no need for system identification, since the system is already known.
- Black box models. No prior knowledge exists about the system. The only way to gain insight about the system is to observe the responses caused by input stimuli. This inference process can be performed through system identification.
- Grey box models. Certain level of structure of the system is known beforehand. However, this partial model does not fully describe the system, and some free parameters may need to be inferred in order to complete the model. Similarly to the case in black box models, this inference process can also be performed through system identification.

With some exceptions, it is usually simpler to identify free parameters (grey box system identification) than to infer a complete model (black box system identification), given that the search space is constrained, reducing complexity. Nevertheless, the partial model chosen for grey box system identification should be able to accurately describe the system with a specific set of parameters.

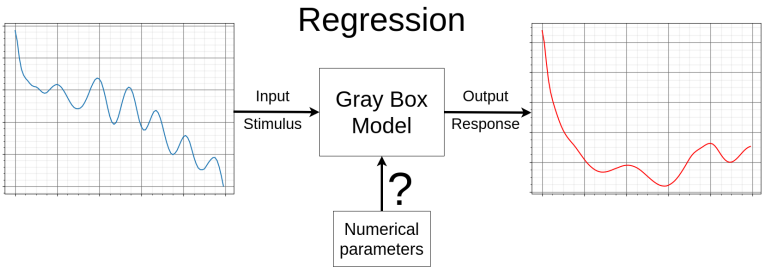

Figure 1.3: Identification problem based on numerical regression.

Mathematical models are essential in the field of aircraft design. Numerical simulation can mitigate the high costs associated with real full-scale aircraft development and testing. High fidelity dynamic models can also improve the performance of automatic flight control systems, using methods such as model predictive control (MPC) or model reference adaptive control (MRAC) (Stevens et al., 2003), and are also useful for the development of simulators for pilot training, among other tasks. Some methods have traditionally been applied for the task of aircraft dynamic modelling, including wind tunnel testing, computational fluid dynamics, or frequency-response analysis in flight test data. In the recent years, arising from the exponential growth in available computing capacity, evolutionary algorithms and deep learning techniques have flourished and taken a very important role in many areas, including system identification. It is in this field that the present work develops.

1.2 Problem description

The problem to solve can be stated as follows.

> "Given a finite history of aircraft states and controls, sampled at discrete time intervals, find a mathematical model that accurately describes this motion, assuming the provided control history values are used for dynamic propagation starting from the initial states provided".

This corresponds to an identification problem. As later explained in section 1.4, the problem is to be solved in a grey box model approach, considering a conceptual model of the type of vehicle needed based on physics. The identification is performed by estimating a set of numerical parameters that describe the aerodynamic properties of the vehicle. These parameters complete the gray box aerodynamic model. Therefore, the problem more specifically corresponds to an identification problem based on *numerical regression* (figure 1.3).

1.3 Motivation

The present work was inspired by the idea of developing a software application with the capability of identifying aircraft aerodynamics purely from recorded flight data, with no additional requirements beyond the need of a flying vehicle, equipment for recording dynamic states during flight and a device with some computational capacity to perform the identification. Even though these type of applications already exist, they usually require a cumbersome configuration process and a certain level of usage experience is needed to obtain accurate results. Ideally, the identification tool should be easy to use, and not require any additional configuration beyond the process of loading the recorded flight data for obtaining results.

1.4 State of the art

Aircraft design is typically performed using a grey box system identification approach, considering a conceptual model of the type of vehicle needed based on physics. In this regard, simple approximate models are often used instead of high fidelity models because they can greatly improve functional visibility of the aircraft dynamics, isolating the dominant characteristics of the system without implying a significant deterioration in the fidelity of the response (Cook, 2013). Aircraft aerodynamic modelling is typically performed by estimating a set of parameters that describe the aerodynamic properties of the vehicle. These correspond to free parameters inherent to the gray box aerodynamic model, and their identification is required to define the full model. The gray box model used in this work is described in the methodology chapter (section 2.2.1). The recent advances in the field of computing have been essential for the application of many algorithms used for this task (Cook, 2013). This process is often iterative, several refinement steps are performed on the physics-based model until achieving accurate inference capability. Resulting models always involve some level of uncertainty, which can be taken into account for robustness in control system design.

This section includes a brief introduction into some of the methods most commonly used for identification of aircraft dynamics: (1) Empirical calculation, (2) wind tunnel testing, (3) computational fluid dynamics and (4) identification from flight data.

1.4.1 Empirical calculation

The most simple method for derivative estimation consists on calculating the derivatives from first principles using approximate mathematical models of the aerodynamic properties of the aircraft. This method is also the least accurate. However, since the models take into account the physical phenomena involved, it can help in providing insight into the dominant characteristics of the system (Cook, 2013). Additionally, these strategies tend to have very low computational requirements, which can be an advantage compared with other methods.

Figure 1.4: NASA wind tunnel with the scale model of an airplane (source: Wikimedia Commons, public domain)

In this regard, it is worth mentioning the United States Air Force (USAF) Stability and Control Digital DATCOM, a software application that implements the methods contained in the USAF Stability and Control Data Compendium, and allows calculation of static stability, control and dynamic derivative characteristics of fixed-wing aircraft based on a geometric description of the vehicle (Stevens et al., 2003). The software outputs the corresponding dimensionless stability derivatives according to the specified flight conditions. This software application has been extensively used in the aviation sector (Rauf et al., 2011; Li et al., 2016).

1.4.2 Wind tunnel testing

Derivative estimation is sometimes achieved by building smaller scale aircraft models and measuring the resulting forces and moments caused by induced wind inside a concealed tunnel (figure 1.4). The measurements are performed for several combinations of wind velocity, aerodynamic angles, and control surface angles, and are typically taken on stationary conditions. For estimating the derivatives which are most coupled between different modes, more complex multi-degree of freedom test rigs are required. These methods are usually more accurate than empirical calculation methods (Cook, 2013). Additionally, flow visualization techniques can be used to provide diagnostic information about the flow field.

According to the theory of aerodynamics, two different aircraft with equal geometry, but not necessarily equal size, have the same aerodynamic coefficients at the same angle of attack, as long as two similarity parameters are maintained identical: Reynolds number and freestream Mach number. Reynolds number is a dimensionless parameter which gives an estimate of the nature of the boundary layer viscous flow. Low Reynolds numbers give way to laminar flow on the boundary layer, while

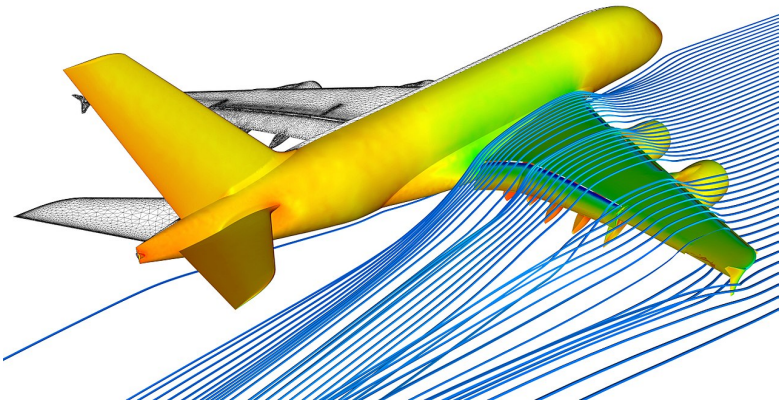

Figure 1.5: Computational fluid dynamics applied to an Airbus A380 (source: Wikimedia Commons, public domain)

high Reynolds numbers yield turbulent flow. Matching of the similarity parameters allows inference of full scale aircraft properties based on the smaller scale wind tunnel results (Stevens et al., 2003). Wind tunnel testing is generally less expensive than CFD techniques, but has inaccuracies due to wall effects, measurement errors and model unsteadiness, among other factors (Chauhan and Singh, 2017b).

1.4.3 Computational fluid dynamics

Computational fluid dynamics (CFD) techniques are extensively used for identification of many systems, including aircraft. These techniques use numerical analysis in order to solve problems that involve fluid flow (Ferziger et al., 2020; Moukallet et al., 2015). The freestream flow of the fluid is simulated in a computer, using simplified models based on the Navier-Stokes equations. Boundary conditions are introduced in order to simulate the presence of aircraft aerodynamic surfaces (figure 1.5).

These techniques can provide very precise models, assuming an accurate description of the aircraft physical characteristics is available. The achievable precision is also directly related to the computational power applied to the problem. These methods are appropriate for theoretical aerodynamic estimation models which include rigid or flexible body modes, or unsteady aerodynamics (Chauhan and Singh, 2017b).

1.4.4 Identification from flight data

Wind tunnel testing and computational fluid dynamics are typically used during the development stages of large commercial or military aircraft. However, these techniques can be expensive, especially if high precision is required. They give way to optimize the initial design before a real prototype is

built.

Here comes in a third method for modelling aircraft dynamics: Inferring the model from real flight data. On the contrary to the previous methods, this technique requires a functional full-scale aircraft. Therefore, the technique is valid for iterative refinement of the design, but not for initial design development. However, given that the method is based on experimental data of the real aircraft, instead of an approximate model, achievable fidelity can be greater than other methods.

Some of these techniques tend to be computationally demanding. Recent advances in computing, together with greater computational resources availability, have enabled their usage for the problem of system identification. Another important consideration when using these methods is the requirement of adequately filtering the noise present in the recorded dynamic state variables, since the accuracy of the estimated derivatives is directly related to it. This translates into the fact that state estimation techniques are frequently used in order to obtain precise estimates of the dynamic states for the identification procedure. Some of the techniques which are used for optimal state estimation of nonlinear systems include the extended Kalman filter (EKF), the unscented Kalman filter (UKF) and the particle filter (PF) (Simon, 2006; Doucet et al., 2001). However, if given adequate resources, the advantages of parameter identification methods from flight data are significant. These techniques admit highly nonlinear models and dynamic conditions, and can provide high identification accuracy, since the real dynamics are considered (Cook, 2013).

Several parameter identification algorithms based on flight data have been used in literature. Most notably, we can distinguish the following classes: (1) Frequency domain analysis, (2) sparse regression techniques, (3) artificial neural networks and (4) evolutionary algorithms. A brief description of these methods is provided in the following sections.

1.4.4.1 Frequency domain analysis

Frequency-response identification methods are particularly well suited to system identification of aircraft dynamic models from flight test data. These methods use mathematical tools like Fourier transforms (FT) to obtain representations of flight data in the frequency domain. Specialized flight-test maneuvers are performed in order to excite the dynamics of concern in different dynamic modes (Tischler and Temple, 2006). Some common approaches for excitation are *frequency sweeps* and *doublets* (figure 1.6), as performed in Morelli et al. (2012). Frequency responses of the system are extracted for a full characterization of the coupled dynamics without a-priori assumptions. Nonlinear search algorithms are subsequently used for extracting state-space models that match the frequency-response data (Center, 2006).

Several tools have been conceived for solving the problem of system identification from flight data using frequency-domain analysis, including the software package *Comprehensive Identification from Frequency Responses* (CIFER®) (Tischler and Temple, 2006). These techniques, however, require very specific aerial maneuvers to be performed in different aircraft dynamic modes. This process can

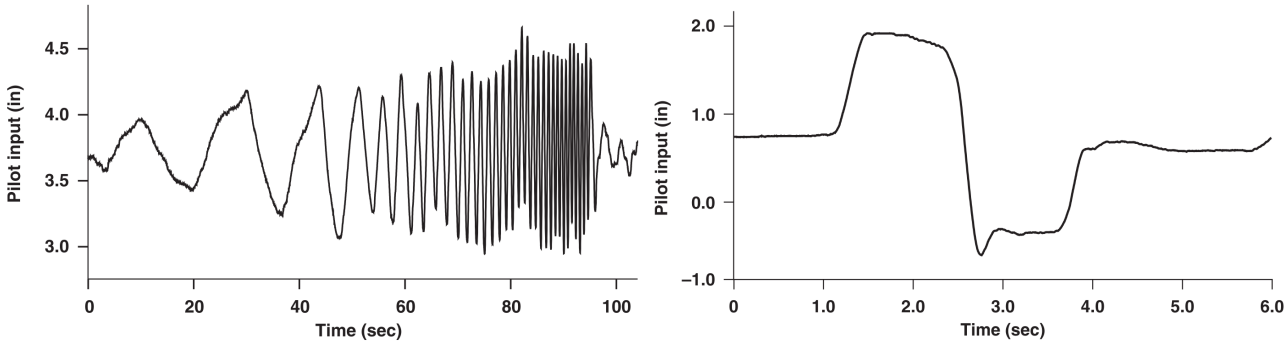

Figure 1.6: Frequency sweep (left) and doublet (right) excitations

be cumbersome compared to other methods. In turn, they tend to be faster than time-domain based methods.

1.4.4.2 Sparse regression techniques

Some algorithms have been used for extracting governing equations from data using sparse regression techniques for nonlinear models. Of particular note is the *Sparse Identification of Nonlinear Dynamics* (SINDy) algorithm (Brunton et al., 2016), which uses sparse regression to determine the fewest terms in the dynamic governing equations required to accurately represent the data. This results in parsimonious models that balance accuracy with model complexity to avoid overfitting. This algorithm shows excellent system identification capabilities, but requires careful configuration in some areas, like the choice of measurement coordinates and function basis for the dynamics, which if not adequately configured may lead to poor results. It is important to note that this method uses a black box approach for identification, although some knowledge of the system may be introduced in the form of mathematical functions or model components known to be related with the dynamics.

1.4.4.3 Artificial neural networks

Artificial neural networks (ANN) can be used to infer a model from observations. The inference capabilities of ANNs are usually the state of the art in many fields, assuming sufficient measured data and computational power is available (Aggarwal, 2018; Skansi, 2018). Neural networks have been previously investigated as an alternative to conventional mathematical models for system identification, proving to be a highly feasible concept decades ago (Raol and Jategaonkar, 1995).

Neural networks can be trained to perform the mapping between aerodynamic derivatives and measured flight data or between the future states given the current states. In order to facilitate the process of data gathering for network training, simulation techniques can be used for data generation, using automatic flight control systems coupled with aircraft mathematical models. This approach for data generation can provide huge amounts of measured data for network training. However, high fidelity

mathematical models for simulation are required and subtleties such as atmospheric turbulence or sensor and actuator models should be taken into account in order to achieve a neural network estimator that can be successfully applied to infer derivatives or states from real flight data. This process can be complex, and high computational power is required.

Wharington et al. (1993) investigate neural networks as an alternative to conventional mathematical models for both model generation and aerodynamic derivative estimation of aircraft dynamics. The approach is demonstrated to be effective through application to simulated data, with capability to withstand measurement noise. Achieved performance exceeds the capabilities of more traditional methods. The concept of applying deep learning for the complete aircraft model generation in a black box approach is also evaluated by Zheng and Xie (2018). Andersson et al. (2019) study the feasibility of the recently proposed temporal convolutional network (TCN) for nonlinear system identification, with very good results even with small datasets. Additionally, long short-term memory (LSTM) networks, despite very rarely applied to system identification problems, demonstrated good overall performance. Bansal et al. (2016); Amiruddin et al. (2020) use recurrent neural network (RNN) architectures such as GRU and LSTM for nonlinear identification of quadcopter dynamics, and apply the resulting models to MPC, with good results even in previously unseen trajectories. The best results were achieved in a hybrid architecture consisting on a convolutional and a recurrent neural network, named CNN-LSTM. Punjani and Abbeel (2015); Kang et al. (2019) address the problem of nonlinear identification of unmanned helicopter models, with good results. The latter uses a CNN architecture and also tests control capabilities with the neural network identifier. Chauhan and Singh (2017a) use feed forward neural networks (FFNN) for aerodynamic derivative identification with promising results.

1.4.4.4 Evolutionary algorithms

Numerical optimization techniques can be applied for solving the problem of aircraft dynamic modelling from flight data. The procedure consists on performing a real flight experiment while recording dynamic states and control inputs during flight. This recorded data is subsequently introduced into a computer program containing a grey box model of the aircraft, which is used to find the optimal set of aerodynamic derivatives that give rise to a set of dynamic responses identical to the recorded dynamic state history when fed to the aircraft mathematical model using recorded control history as input (Klein and Morelli, 2006; Cook, 2013).

Least squares model fitting algorithms such as Gauss-Newton, steepest-descent and Levenberg-Marquardt type methods are extensively used for aircraft aerodynamic identification (Rasheed, 2017; Ahsan et al., 2016; Jamil et al., 2015; Rommel et al., 2017; Padayachee, 2016). These methods are typically used to approximate the solution of overdetermined systems by minimizing the sum of the squares of the result residuals. These methods, however, are only able to guarantee local convergence. Therefore, they are only able to find the global minima of convex functions. Global

Table 1.1: Possible strategies to solve the problem.

	Non-convexity	High-dimensionality
Discrete search	Suitable	Unsuitable (computationally intractable)
Continuous function optimization	Unsuitable (may converge to local minima)	Suitable
Global optimization	Suitable	Suitable

minima of non-convex functions can only be found under certain initial point conditions.

Most aircraft parameter identification problems have a complex non-linear non-convex multidimensional objective function. In many cases, it is possible to choose starting values that will lead to the global minimum of the objective function using a local optimizer (Klein and Morelli, 2006). However, choosing an adequate starting condition for guaranteeing convergence may not be trivial in some problems. The search landscape in this type of problems is both non-convex and high-dimensional. This means that neither discrete search nor continuous function optimization are adequate, since the former makes the problem computationally intractable while the latter may converge to local minima (table 1.1).

Global optimization techniques are required in order to find global minima or maxima of a non-convex function. Common approaches to global optimization problems, where multiple local minima may be present, include evolutionary algorithms, bayesian optimization and simulated annealing (Liberti, 2008; Eiben and Smith, 2015). Evolutionary algorithms draw inspiration from the process of natural evolution, and are suitable for these type of problems (Eiben and Smith, 2015). These methods use metaheuristics based on natural evolution in order to more intelligently explore the search space. Therefore, they can be interpreted as sophisticated methods for trial-and-error searching, or discrete search, which can reduce the computational requirements of more simple discrete approaches for solving high-dimensional problems. These type of algorithms allow to avoid the complicated problem of initial value selection for reaching the global minimum using local optimization methods, and provides higher optimization ability (Hongfeng et al., 2019).

Evolutionary algorithms have been extensively used in the field of system identification. Some examples include Cui et al. (2019), where a biology-based evolutionary algorithm known as particle swarm optimization (PSO) is applied in combination with a neural network for aerodynamic parameter estimation. Braun et al. (2010, 2011) apply a multiobjective evolutionary optimization scheme to the problem of system identification of a hydraulic valve. A class of evolutionary algorithms known as genetic programming (GP) is used in the works of Rodriguez-Vazquez et al. (2004) and Gray et al. (1998) for nonlinear system identification. Grey-box model identification via evolutionary computing

is performed in Tan and Li (2002). Rahimpour et al. (2017) use an evolutionary algorithm algorithm known as CMA-ES in a grey box model approach for mathematical modelling of a power transformer. Dynamic modelling of a wind turbine generation system based on grey-box identification via a genetic algorithm is performed in Jizhen et al. (2017).

Evolutionary algorithms may offer some advantages for aerodynamic parameter identification when compared with other approaches. They do not require extensive amounts of high fidelity data generation through simulation. Although significant computational power is required for the evaluation of individuals through simulation, the method is easily parallelizable. Additionally, the method does not require significant tuning to provide good results out of the box. For these reasons this approach has been selected for solving the problem proposed in this work, along with the deep learning approach (section 1.4.4.3). Details about the specific algorithm chosen and its implementation are discussed in the methodological sections (section 2.3).

1.5 Research hypothesis and objectives

In this study, it is hypothesized that evolutionary algorithms and deep learning methods can be used to solve the problem of aircraft aerodynamic coefficient identification from flight data. In order to prove the research hypothesis, the following specific **objectives** are proposed[1].

- Development of an **aircraft dynamics simulator** based on aircraft aerodynamics and classical kinematics. The simulator contains the aircraft dynamic model and can be used to extrapolate past or future states given initial conditions by means of numeric simulation. The internal propagator accepts multiple parameters as inputs: aerodynamic derivatives, geometrical and inertial properties and external disturbances and forces such as wind. The simulator can also take user control inputs by means of a joystick for model evaluation.
- Development of an **evolutionary optimization framework** for finding optimal sets of aerodynamic coefficients given a history of flight states and controls. This optimizer is based on an evolutionary algorithm and makes use of the dynamic propagator to generate multiple trajectories which are used to calculate the fitness of each individual.
- Integration of a **deep learning framework** for training and inference of an artificial neural network for parameter estimation. The training process requires automatically generated data via the aircraft dynamics simulator.
- Integration of multiple **graphical elements**:
  - A **graphical engine** is proposed for receiving dynamic states from the simulator and rendering real-time graphics of the moving aircraft.

[1] Code implementation is available at the repository `https://github.com/fidelechevarria/DeepAero`

– For improving user interaction with the framework, a simple **graphical user interface** is proposed.

1.6 Scope and limitations

The methodology proposed in this work aims to provide system identification capabilities of aircraft dynamics from flight data. As discussed later in section 2.2.1, the aircraft aerodynamic coefficients are, in practice, specified as functions of the aerodynamic angles, velocity, and altitude. This work considers these conditions approximately constant in the domain of each problem instance to solve, and for this reason a single numeric value is obtained for each aerodynamic parameter. Therefore, the estimated model validity decreases for dynamics that differ from the conditions of the corresponding real flight used for identification. For obtaining a general model valid for every condition in the flight envelope, multiple problem instances would need to be solved, each of them corresponding to different envelope conditions. Subsequently the general aerodynamic coefficient values would have to be estimated by means of interpolating the values between the obtained solutions, which is usually implemented in the form of lookup tables. Alternatively, the framework could be extended for estimation of the complete interpolation points for all coefficients.

Additionally, the nonlinear aircraft model chosen is exclusively valid for fixed-wing aircraft, and not for other classes of aircraft such as rotorcraft. However, the framework could be adapted to any type of vehicle by means of adapting the gray box dynamic model used.

1.7 Thesis outline

The rest of this thesis is structured as follows. Chapter 2 describes the methodology developed for solving the problem. First, a general architecture overview is provided, followed by a description of the different framework modules, namely dynamics simulator, optimization frameworks (evolutionary and deep learning) and graphical elements. Chapter 3 presents and discusses the evaluated identification results corresponding to both frameworks, and compares them with each other and with other approaches. Finally, chapter 4 includes the final conclusions of this work, and provides some guidelines for possible future work in extending the capabilities of the developed framework.

Chapter 2

Methodology

The methodology followed in this work is presented in this chapter. First, the two architectures developed in this work are presented, namely (1) evolutionary optimization and (2) deep learning based. In the remaining sections of the chapter, information about each framework component is provided.

2.1 Framework architecture

The two architectures developed in this work are presented in this section. These high-level architecture descriptions aim to help the reader understand the complete process followed for achieving aircraft identification.

2.1.1 Evolutionary optimization framework architecture

The evolutionary optimization framework architecture is schematized in figure 2.1. The procedure consists on performing a real flight experiment while recording dynamic states and control inputs during flight. This recorded data is subsequently introduced into a computer program containing a grey box model of the aircraft, which is used to find the optimal set of aerodynamic derivatives which give rise to a set of dynamic responses identical to the recorded dynamic state history when fed to the aircraft mathematical model using recorded control history as input (Klein and Morelli, 2006; Cook, 2013).

The framework is divided in two different segments.

- The **flight experiment** has the objective of obtaining measured flight data, which is required for the system identification process. Variables required for recording are control inputs and dynamic states. Flight data generation can be achieved by means of two methods:

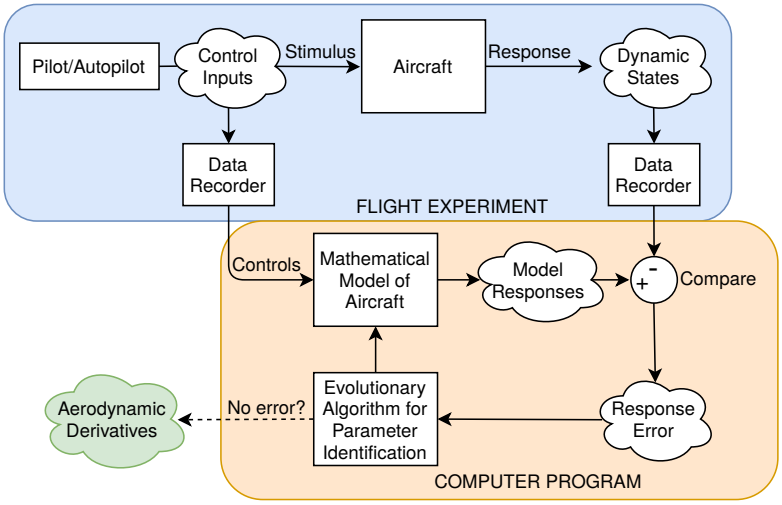

Figure 2.1: Evolutionary optimization framework architecture.

A) **Real flight**, used for real aircraft identification. A full-scale aircraft flight is performed while recording control input commands and dynamic states in real-time.

B) **Simulated flight**, used for evaluation purposes. A mathematical model with some predefined set of aerodynamic derivatives is used to simulate flight for data recording. Since the original aerodynamic derivatives are fully known, the identified derivatives can be evaluated by comparing both instances. Input controls for simulation can be introduced either by an automatic control system or by real-time user commands through a connected joystick device.

- The **computer program** has the objective of identifying the aerodynamic derivatives which better correlate the response of the simulated grey box mathematical model with the recorded dynamic states, if recorded control commands are taken as inputs.

2.1.2 Deep learning framework architecture

The deep learning framework architecture is schematized in figure 2.2. The neural network is trained to perform the mapping between aerodynamic derivatives and measured flight data. In order to facilitate the process of data gathering for network training, simulation techniques can be used for data generation, using automatic flight control systems coupled with aircraft mathematical models. The procedure is performed in two stages.

- During the **training** stage a mathematical model of the aircraft with random aerodynamic parameters is propagated with random control inputs. The controls and resulting states are recorded and fed to a neural network for training, along with the corresponding derivatives. The neural network gradually performs the mapping between controls along with their resulting

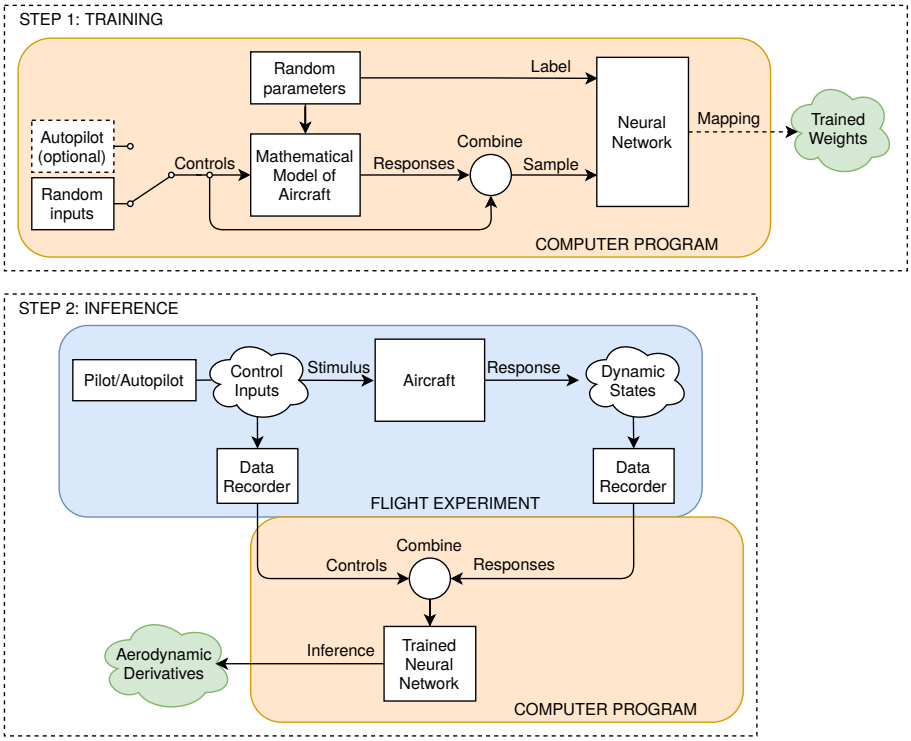

Figure 2.2: Deep learning framework architecture.

states, and the set of aerodynamic derivatives used. This stage is only performed once, since the resulting trained network has the capability of generalizing for estimation of aircraft dynamics.

- During the **inference** stage the trained neural network is fed with real recorded flight data (controls and states) for inference of aerodynamic derivatives. For validation purposes, the recorded flight data can also be simulated.

2.1.3 Architecture design details

Python programming language has been selected as the main language for the development of both frameworks. Its high-level nature, as well as the vast amount of available compatible libraries, has reduced development time of multiple modules which do not particularly require high computing efficiency, and the integration and interfacing of the different modules in the framework. However, some modules like the numerical propagator and simulator require high computational efficiency since they are executed millions of times during each problem instance (each problem requires thousands of candidate solution evaluations, and each candidate solution requires thousands of model propagations). Therefore, these efficiency-critical modules have been implemented in C++ language, which is compiled before execution instead of interpreted at run-time. An API has been designed to be able to use those modules within the Python framework.

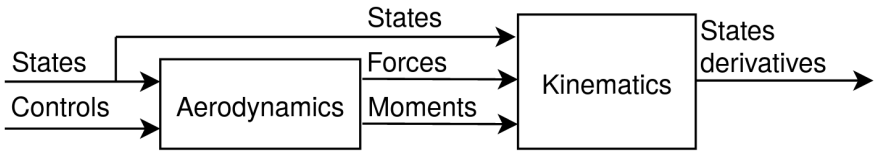

Figure 2.3: Aircraft dynamics

2.2 Aircraft dynamics simulator

The mathematical model of aircraft dynamics serves as a flight data generation tool for obtaining referenced data, allowing evolutionary algorithm evaluation and neural network training. Additionally, the simulator can be used by the user to check the consistency of the final identified models.

As stated in section 1.4, a grey box model approach is proposed for identification of system dynamics from flight data, which combines partial theoretical structure with data to complete the model. The predefined part of the model is based on known physical principles of aerodynamics and kinematics. This known information reduces the vast space of possible solutions for the dynamic model, decreasing the computational complexity required to find one that is feasible. It also increases solution generality, meaning that a solution which satisfies the partial theoretical structure is more likely to generally describe the aircraft dynamics, even in conditions not present in the state and control history input data. The following section presents a generic, nonlinear, six-degrees-of-freedom (6DoF) model for fixed-wing aircraft which is widely used in the industry (Stevens et al., 2003). The complete aircraft model provides information about the changes in states, taking control surface deflections and dynamic states as inputs.

2.2.1 The nonlinear fixed-wing aircraft model

Aircraft dynamics can be separated into two main components (figure 2.3):

- **Aerodynamics** relate the dynamic states and controls of the vehicle with the resulting forces and moments. Aerodynamics are fully dependent on the aircraft physical properties and dynamic states.

- **Kinematic relations** relate forces, moments and dynamic states with the dynamic states derivatives with respect to time. Kinematic relations are not dependent on the aircraft physical properties and thus can be applicable to any type of vehicle. Kinematics are fully known and common to any moving object, and do not contain any free parameters.

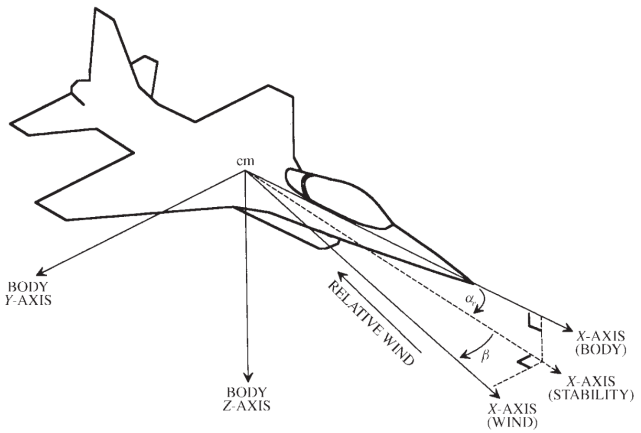

Figure 2.4: Body-fixed coordinate system. Aerodynamic angles α and β are also depicted (obtained from Stevens et al.).

2.2.1.1 Aerodynamic model

Vehicle aerodynamics predict generated forces and moments taking vehicle physical properties and dynamic states as inputs. On textbooks on aerodynamics (Kuethe and Schetzer, 1984) it is shown that, for a body of given shape with a given orientation to the free-stream flow, the forces and moments are proportional to the product of free-stream mass density, ρ, the square of the freestream airspeed, V, and a characteristic area of the body. When modelling aircraft aerodynamics, the characteristic area of the body is typically selected as the wing reference surface, S. It is convenient to define the dynamic pressure, q, which gives information about the kinetic energy in the flow per energy volume, and is defined by

$$q = \frac{1}{2} \cdot \rho \cdot V^2 \tag{2.1}$$

It is useful to define the body-fixed coordinate system (Stevens et al., 2003; Gómez Tierno et al., 2012), which is centered at the aircraft center of mass, has its x-axis parallel to the aircraft fuselage reference line and its z-axis in the conventional aircraft plane of symmetry. The y-axis is defined so that the coordinate system is right-handed.

It is also useful to define the local geographical coordinate system, also called NED (North, East, Down). The NED coordinate system is also centered in the vehicle center of mass. Both the x-axis and y-axis are contained in the plane perpendicular to the local vertical direction. The x-axis points towards the Earth northern axis, while the y-axis points towards the Earth eastern axis. The z-axis points to the local vertical direction.

Two magnitudes known as *aerodynamic angles* are of utmost importance to define the aircraft dynamics (Stevens et al., 2003):

- **Angle of attack**, α, specifies the angle between the fuselage reference line and the projection

of the relative wind on the body z-x plane. It is positive when the relative wind is on the underside of the aircraft. It is the primary parameter in longitudinal stability considerations. A related magnitude exists called incidence angle, which corresponds to the angle between the wing chord line and the body z-x plane. Considering level flight and no wind, incidence angle introduces an effective angle of attack at zero pitch.

- **Sideslip angle**, β, specifies the angle between the relative wind vector and its projection on the body x-z plane. The sideslip angle is essentially the directional angle of attack of the airplane. It is positive when the relative wind is on the right side of the aircraft. It is the primary parameter in lateral-directional stability considerations.

Both angles are depicted in figure 2.4. Aerodynamic angles can be modeled as a function of the velocity components in the body-fixed coordinate system:

$$V = \sqrt{v_x^2 + v_y^2 + v_z^2}$$

$$\alpha = \arctan\left(\frac{v_z}{v_x}\right) \tag{2.2}$$

$$\beta = \arcsin\left(\frac{v_y}{V}\right) \tag{2.3}$$

Where v_x, v_y and v_z represent linear velocity vector components in the body-fixed coordinate system. Angular velocity components in the body-fixed coordinate system are represented as p, q and r.

The wind-axes coordinate system can be obtained from the body-fixed coordinate system by a left-handed rotation of α around the body y-axis followed by a right-handed intrinsic rotation of β around the intermediate z-axis resulted from the first rotation (figure 2.4). The intermediate coordinate system formed after the first rotation is known as the stability-axes coordinate system, and is used for analyzing the effects of perturbations from steady-state flight. The rotation matrix for transforming from wind to body fixed coordinate systems is defined as (Stevens et al., 2003; Gómez Tierno et al., 2012):

$$C_{w/bf} = \begin{bmatrix} \cos\alpha\cos\beta & \sin\beta & \sin\alpha\cos\beta \\ -\cos\alpha\sin\beta & \cos\beta & -\sin\alpha\sin\beta \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \tag{2.4}$$

Drag, D, cross-wind force, C, and lift, L, are defined as the negated values of the aerodynamic force components in wind-axes. They are negated because they were originally derived in an inverted wind-axes coordinate system definition (Pope, 1954). This components can be transformed to the aerodynamic force components in body-fixed axes, namely X_A, Y_A and Z_A through the rotation

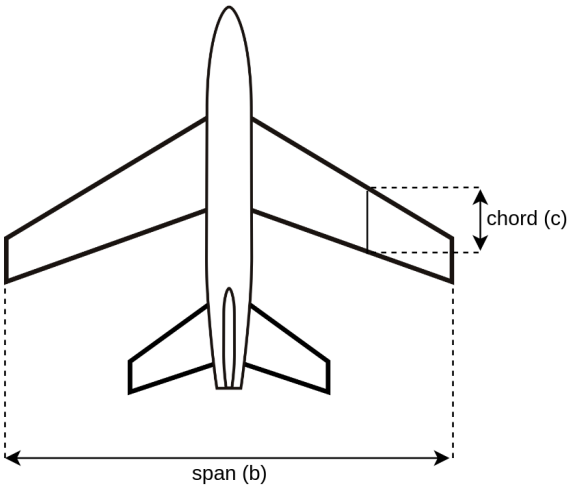

Figure 2.5: Span and chord of a wing. Wing mean aerodynamic chord is defined as the relation between the wing total area and its span.

matrix $C_{w/bf}$. Additionally, l, m and n are defined as the aerodynamic moment components in the body-fixed coordinate system.

Forces and moments acting on the complete aircraft are linearized and defined in terms of dimensionless aerodynamic coefficients, as shown in equations 2.5 and 2.6 respectively. Aerodynamic coefficients represent the ability of the aircraft geometry to produce forces or moments. They depend on the aircraft physical properties, as well as aerodynamic angles, Mach and Reynolds number. If Mach, altitude and air temperature are specified, together with an atmospheric density model, then Reynolds number and dynamic pressure can be determined. Therefore, aerodynamic coefficients are, in practice, specified as functions of the aerodynamic angles, Mach and altitude in the standard atmosphere.

$$\begin{aligned} L &= q \cdot S \cdot C_L \\ D &= q \cdot S \cdot C_D \\ C &= q \cdot S \cdot C_C \end{aligned} \tag{2.5}$$

$$\begin{aligned} l &= q \cdot S \cdot b \cdot C_l \\ m &= q \cdot S \cdot c \cdot C_m \\ n &= q \cdot S \cdot b \cdot C_n \end{aligned} \tag{2.6}$$

C_L, C_D and C_C correspond to the lift, drag and cross-wind force coefficients, whereas C_l, C_m and C_n correspond to the moment coefficient components in the body-fixed coordinate axes. Similarly, C_X, C_Y and C_Z correspond to the force coefficient components in the body-fixed coordinate axes. Moment nondimensionalization is usually performed with additional parameters like wingspan, b, or wing mean aerodynamic chord, c (figure 2.5).

In addition, control surface deflections and propulsion system effects cause changes in the coefficients.

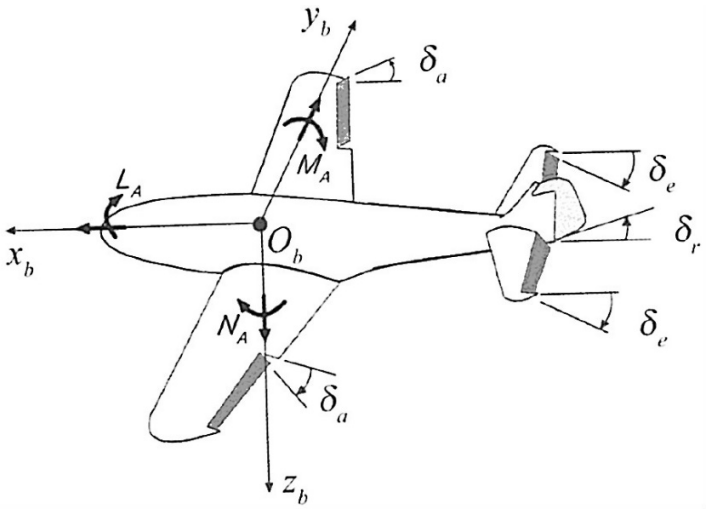

Figure 2.6: Forces, moments and control deflections (obtained from Gómez Tierno et al.).

A control surface deflection, δ_s, effectively changes the camber of a wing, which changes the lift, drag, and moment. For this standard model, three control surface deflections are considered: δ_a measures ailerons deflection, δ_e measures elevator deflection and δ_r measures rudder deflection (figure 2.6).

A common approach to model the force and moment coefficients is to consider the effect of the individual components due to multiple variables. These variables include the aerodynamic angles, control surfaces deflections, or body angular velocities. Generally, the effects of each of these components to the total coefficients are considered to be decoupled. This approximation is advantageous for coefficient estimation and simplifies the mathematical handling. Force and moment coefficients can be approximated by equations 2.7 and 2.8.

Aerodynamic force coefficients

$$\begin{aligned} C_L &= C_{L_0} + C_{L_\alpha} \cdot \alpha \\ C_D &= C_{D_0} + K \cdot C_L^2 + C_{D_\beta} \cdot |\beta| \\ C_Y &= C_{Y_\beta} \cdot \beta + C_{Y_{\delta a}} \cdot \delta a + C_{Y_{\delta r}} \cdot \delta r + \tfrac{b}{2V} \cdot (C_{Y_p} \cdot p + C_{Y_r} \cdot r) \end{aligned} \tag{2.7}$$

Aerodynamic moment coefficients

$$\begin{aligned} C_l &= C_{l_\beta} \cdot \beta + C_{l_{\delta a}} \cdot \delta a + C_{l_{\delta r}} \cdot \delta r + \tfrac{b}{2V} \cdot (C_{l_p} \cdot p + C_{l_r} \cdot r) \\ C_m &= C_{m_0} + C_{m_\alpha} \cdot \alpha + C_{m_{\delta a}} \cdot |\delta a| + C_{m_{\delta e}} \cdot \delta e + C_{m_{\delta r}} \cdot \delta r + \tfrac{c}{2V} \cdot (C_{m_q} \cdot q) \\ C_n &= C_{n_\beta} \cdot \beta + C_{n_{\delta a}} \cdot \delta a + C_{n_{\delta r}} \cdot \delta r + \tfrac{b}{2V} \cdot (C_{n_p} \cdot p + C_{n_r} \cdot r) \end{aligned} \tag{2.8}$$

An aerodynamic derivative "C_{x_y}", also referred to as aerodynamic coefficient or aerodynamic parameter, provides information about the effect on the variable "x" caused by an increment on the

variable "y". Aerodynamic derivatives C_{L_0} and C_{D_0} are called zero angle of attack lift and parasite drag respectively, and they represent the lift and drag contribution which is not due to any other variable and is present even at zero angle of attack and sideslip.

The dimensionless coefficients shown in equations 2.7 and 2.8 are usually named *aerodynamic coefficients* or *aerodynamic derivatives*, and they can be classified into three different groups:

Stability derivatives This group includes parasite derivatives, or the ones that are not related to any magnitude but their effect is always present, as well as the derivatives due to the aerodynamic angles (α and β).

$$C_{L_0}, C_{L_\alpha}, C_{D_0}, K, C_{D_\beta}, C_{Y_\beta}, C_{l_\beta}, C_{m_0}, C_{m_\alpha}, C_{n_\beta}$$

Control derivatives Derivatives due to the effect of control surface deflections are included in this group.

$$C_{Y_{\delta a}}, C_{Y_{\delta r}}, C_{l_{\delta a}}, C_{l_{\delta r}}, C_{m_{\delta a}}, C_{m_{\delta e}}, C_{m_{\delta r}}, C_{n_{\delta a}}, C_{n_{\delta r}}$$

Damping derivatives Damping derivatives include the ones due to the effects of angular velocities on the vehicle. Their name is due to the restoring effect of angular velocity components, which contributes to aerodynamic stability in many aircraft.

$$C_{Y_p}, C_{Y_r}, C_{l_p}, C_{l_r}, C_{m_q}, C_{n_p}, C_{n_r}$$

Note that terms concerning damping derivatives are usually nondimensionalized in the moment coefficients equations. The nondimensionalization factor is dependent on the reference axis, with a factor of $\frac{b}{2V}$ for lateral and directional modes and $\frac{c}{2V}$ for longitudinal mode. This factor is related to the variable $\frac{pb}{2V}$ which corresponds to a dimensionless roll rate. In a continuous roll, with the aircraft center of mass moving in a straight line, the wing tips move along a helical trajectory and this variable represents the helix angle, which turns out to be a useful figure of merit for roll control power (Stinton, 1996).

2.2.1.2 Aircraft kinematics

This section includes the aircraft kinematic equations, which model the derivatives of the aircraft states in terms of the resulting forces and moments from the aerodynamic model, and of the states themselves. Aircraft kinematics are considered known, and are common to any kind of vehicle. A full derivation of these equations is not considered in the scope of this work. More detailed descriptions are covered in Stevens et al. (2003), and partially in Cook (2013).

Force equations Force equations are derived from Newton's second law of motion from classical mechanics, which states that the net force being applied to an object is equal to the product of its mass (m) times its acceleration. Force kinematic equations are defined in equation 2.9.

$$\begin{aligned} \dot{u} &= r \cdot v - q \cdot w - g \cdot \sin(\theta) + (X_A + X_T)/m \\ \dot{v} &= -r \cdot u + p \cdot w + g \cdot \sin(\varphi) \cdot \cos(\theta) + (Y_A + Y_T)/m \\ \dot{w} &= q \cdot u - p \cdot v + g \cdot \cos(\varphi) \cdot \cos(\theta) + (Z_A + Z_T)/m \end{aligned} \tag{2.9}$$

The components u, v and w correspond to the linear accelerations in the body-fixed coordinate system, and $\dot{u}$, $\dot{v}$ and $\dot{w}$ represent their derivatives with respect to time. p, q and r represent the angular velocity components in the body-fixed coordinate system. φ, θ and ψ correspond to the Euler angles (roll, pitch and yaw), which represent the aircraft's orientation with respect to the NED coordinate system in terms of sequential rotations. g represents Earth's gravity. X_A, Y_A and Z_A correspond to the aerodynamic forces components, while X_T, Y_T and Z_T correspond to propulsive force components acting on the vehicle, all of them referred to the body-fixed coordinate system.

Kinematic equations Kinematic equations relate the rate of Euler angles with respect to time with the angular velocities and current Euler angles. They are defined in equation 2.10.

$$\begin{aligned} \dot{\varphi} &= p + \tan(\theta) \cdot [q \cdot \sin(\varphi) + r \cdot \cos(\varphi)] \\ \dot{\theta} &= q \cdot \cos(\varphi) - r \cdot \sin(\varphi) \\ \dot{\psi} &= [q \cdot \sin(\varphi) + r \cdot \cos(\varphi)] / \cos(\theta) \end{aligned} \tag{2.10}$$

Moment equations The moment equations are equivalent to the force equations but describe the rotational dynamics, instead of translational dynamics. They are defined in equation 2.11.

$$\begin{aligned} \Gamma \cdot \dot{p} &= I_{xz} \cdot (I_x - I_y + I_z) \cdot p \cdot q - [I_z (I_z - I_y) + I_{xz}^2] \cdot q \cdot r + I_z \cdot l + I_{xz} \cdot n \\ I_y \cdot \dot{q} &= (I_z - I_x) \cdot p \cdot r - I_{xz} (p^2 - r^2) + m \\ \Gamma \cdot \dot{r} &= [(I_x - I_y) \cdot I_x + I_{xz}^2] \cdot p \cdot q - I_{xz} (I_x - I_y + I_z) \cdot q \cdot r + I_{xz} \cdot l + I_x \cdot n \\ \Gamma &= I_x \cdot I_z - I_{xz}^2 \end{aligned} \tag{2.11}$$

The components I_x, I_y and I_z correspond to the vehicle moments of inertia along the three body-fixed coordinate axes, while I_{xz} is one of the products of inertia, or the moment of inertia around z-axis when the vehicle is rotated around the x-axis. The other products of inertia, I_{xy} and I_{yz} are not taken into account since they are usually zero for most aircraft due to their geometrical symmetry along the body-fixed x-z plane.

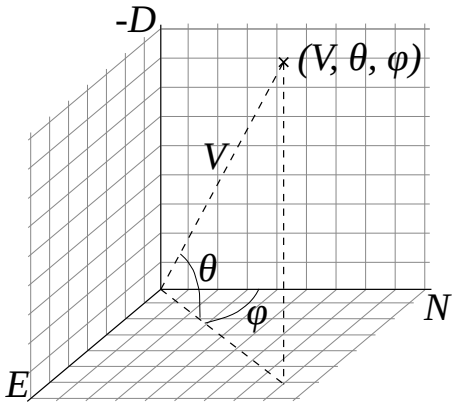

Figure 2.7: Constant wind vector spherical coordinate reference system. Axes N, E, and $-D$ correspond to North, East and the opposite of down directions.

Navigation equations Navigation equations relate the velocity components in NED axes with the body-fixed velocity components through a rotation matrix that considers the Euler angles. They are defined in equation 2.12.

$$\begin{aligned} \dot{p}_N &= u \cdot c\theta \cdot c\psi + v \cdot (-c\varphi \cdot s\psi + s\varphi \cdot s\theta \cdot c\psi) + w \cdot (s\varphi \cdot s\psi + c\varphi \cdot s\theta \cdot c\psi) \\ \dot{p}_E &= u \cdot c\theta \cdot s\psi + v \cdot (c\varphi \cdot c\psi + s\varphi \cdot s\theta \cdot s\psi) + w \cdot (-s\varphi \cdot c\psi + c\varphi \cdot s\theta \cdot s\psi) \\ \dot{h} &= u \cdot s\theta - v \cdot s\varphi \cdot c\theta - w \cdot c\varphi \cdot c\theta \end{aligned} \tag{2.12}$$

Note that h corresponds to altitude, and therefore has opposite sign convention to the NED "Down" axis. In equation 2.12 the notation c has been used to denote the cosine function, and s to denote the sine function.

Kinematic equations, in combination with the aerodynamics equations, model the complete aircraft dynamics (dynamic states) as a function of the control surfaces and the dynamic states themselves.

2.2.1.3 Additional modules

This section presents some auxiliary modules which have been implemented to increase the simulation fidelity in some aspects, including wind and servoactuator modelling.

Wind model A wind model can be really useful for testing the robustness of the vehicle dynamics and control systems. A simplistic approach has been chosen to model wind in the simulator. The model is formed by two main components: (1) A constant wind vector component, with adjustable velocity and direction, and (2) a stochastic wind velocity component in the three dimensions, which tries to replicate the presence of atmospheric turbulence during flight.

The constant wind vector component can be tuned by means of three different parameters: (1) Wind velocity, V_w, (2) wind azimuth, φ_w, and (3) wind elevation, θ_w. These components are referred to an

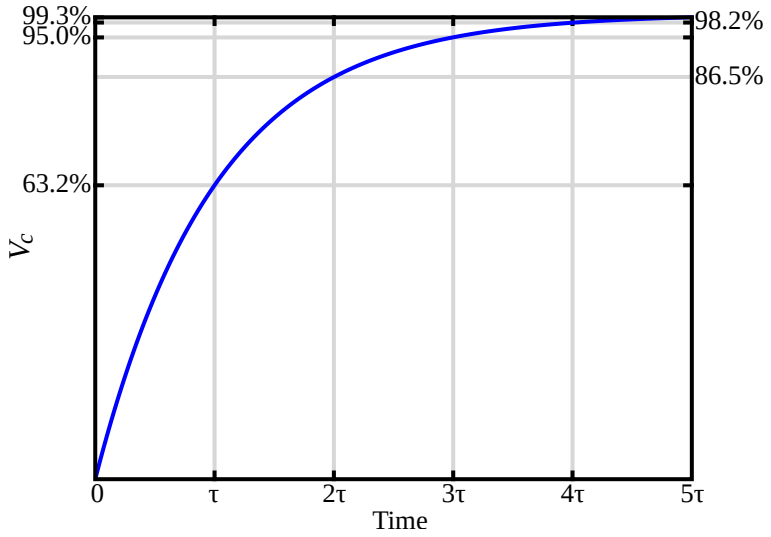

Figure 2.8: Filter response to a step input. Filter response time, τ, and percentage of the original signal in the response (source: Wikimedia Commons, public domain)

Earth-fixed spherical coordinate system (figure 2.7). Constant wind axes components are transformed to standard NED axes by transforming from spherical coordinates to Cartesian coordinates (equation 2.13).

$$\begin{aligned} v_w^N &= V_w \cdot \cos(\theta_w) \cdot \cos(\varphi_w) \\ v_w^E &= V_w \cdot \cos(\theta_w) \cdot \sin(\varphi_w) \\ v_w^D &= -V_w \cdot \sin(\theta_w) \end{aligned} \tag{2.13}$$

In regards to the stochastic wind turbulence contribution, each component in the NED coordinate system is sampled from a continuous uniform distribution $\mathcal{U}$ as indicated in equation 2.14. The turbulence intensity is regulated with the parameter σ_w, which is related with the maximum bounds for sampling the turbulence intensity components.

$$\begin{aligned} z &\sim \mathcal{U}(0,1) \\ v_{turb} = \sigma_w \cdot (10 \cdot z - 5) &\sim \mathcal{U}(-5 \cdot \sigma_w, 5 \cdot \sigma_w) \end{aligned} \tag{2.14}$$

Engine and control actuation model In real aircraft, engine thrust does not vary proportionally to the commanded thrust, δ_t, instantaneously. Engines typically respond to commands after a certain lag. The same is true for the servoactuators that are required to move the control surfaces of the aircraft according to the control commands δ_a, δ_e and δ_r. A simplistic approach has been selected for modelling this actuation lag. Control commands are processed by a low-pass filter (LPF) implemented in a form known as *Brown's simple exponential smoothing* (Brown, 1963), given by equation 2.15.

Table 2.1: Aerodynamic derivatives used as model parameters

C_{D_0}	K	C_{D_β}	C_{Y_β}	$C_{Y_{\delta a}}$	$C_{Y_{\delta r}}$	C_{Y_p}	C_{Y_r}	C_{L_0}	C_{L_α}	C_{l_β}	$C_{l_{\delta a}}$	$C_{l_{\delta r}}$
C_{l_p}	C_{l_r}	C_{m_0}	C_{m_α}	$C_{m_{\delta a}}$	$C_{m_{\delta e}}$	$C_{m_{\delta r}}$	C_{m_q}	C_{n_β}	$C_{n_{\delta a}}$	$C_{n_{\delta r}}$	C_{n_p}	C_{n_r}

$$s_t = \alpha \cdot x_t + (1 - \alpha) \cdot s_{t-1} = s_{t-1} + \alpha \cdot (x_t - s_{t-1}) \tag{2.15}$$

Where s_t is the filtered state at time step t, s_{t-1} is the filtered state at the previous time step $t-1$, α is the *smoothing factor* $(0 < \alpha < 1)$, and x_t is the raw (unfiltered) state at time step t. The response time of an exponential moving average (τ) is the amount of time for the smoothed response of a step function to reach $1 - 1/e \approxeq 63.2\%$ of the original signal (figure 2.8). The relationship between the time constant, τ, and the smoothing factor, α, is given by equation 2.16.

$$\alpha = 1 - e^{\frac{-\Delta T}{\tau}} \tag{2.16}$$

where ΔT is the sampling time interval of the discrete time implementation. If the sampling time is small compared to the time constant $(\Delta T \ll \tau)$, then the smoothing factor can be approximated according to equation 2.17.

$$\alpha \approx \frac{\Delta T}{\tau} \tag{2.17}$$

Therefore, the lag of control surfaces servoactuators is modelled with a parameter τ_s, which represents the response time for the LPF applied to δ_a, δ_e and δ_r, in seconds. Similarly, a parameter τ_e, is used to represent the response time for the engine LPF applied to δ_t, in seconds. The filter is implemented using the approximation in equation 2.17.

2.2.2 Model integration

The non-linear 6DoF aircraft model presented in section 2.2.1 has been used within the grey box model approach. This nonlinear model allows prediction of a discrete future state given current states and controls as inputs. This allows the propagation of dynamics from the initial states using the given state and control histories. Using this partial structure, a complete mathematical model can be defined with a set of 26 aerodynamic parameters, each of which is given a unique real value (table 2.1). The selected partial structure is particularly suited for fixed-wing aircraft, but it could be easily adapted to describe the dynamics of any vehicle class.

Aircraft dynamic state is partially described with position, attitude angles and linear and angular velocities. Control history is described with standard control surface deflections (elevator, ailerons and rudder) and throttle input.

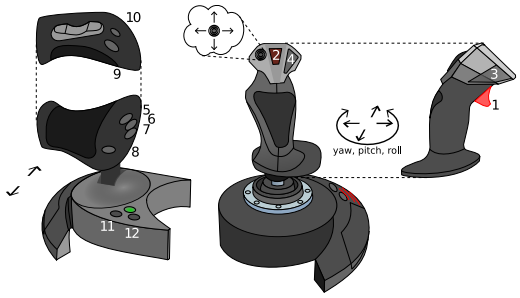

Figure 2.9: Diagram of the joystick device used for manually introducing commands to the simulator (source: Online, public domain)

A Python module named pygame has been used for capturing the commands of a joystick device in real time, allowing the manual control of the aircraft by the user during simulation (figure 2.9). Figure 2.10 represents an example of state and control trajectories recorded with the simulator using joystick control commands.

2.3 Evolutionary algorithm

An evolutionary algorithm has been implemented for aerodynamic derivatives identification from flight data. Evolutionary algorithms draw inspiration from the process of natural evolution, and they act on an environment that can only host a limited number of individuals. These individuals compete for the best adaptability to the environmental conditions, also called *fitness*. Each individual has a unique set of genetic information, or *phenotype*, which is directly related to their *fitness*. Individuals are iteratively recombined, mutated, evaluated and selected. Phenotypic traits which are favorable for the environmental conditions tend to propagate and perpetuate in the population, otherwise they tend to be discarded.

Eiben and Smith (2015) propose the following scheme of the general steps of evolutionary algorithms in pseudocode.

Algorithm 2.1 General scheme of an evolutionary algorithm

```
1: initialize population with random candidate solutions
2: evaluate each candidate
3: while termination condition is not satisfied do
4:     select parents
5:     recombine pairs of parents
6:     mutate the resulting offspring
7:     evaluate new candidates
8:     select individuals for the next generation
9: end while
```

At this stage, the problem reduces to the optimization of a complex real-valued multi-dimensional

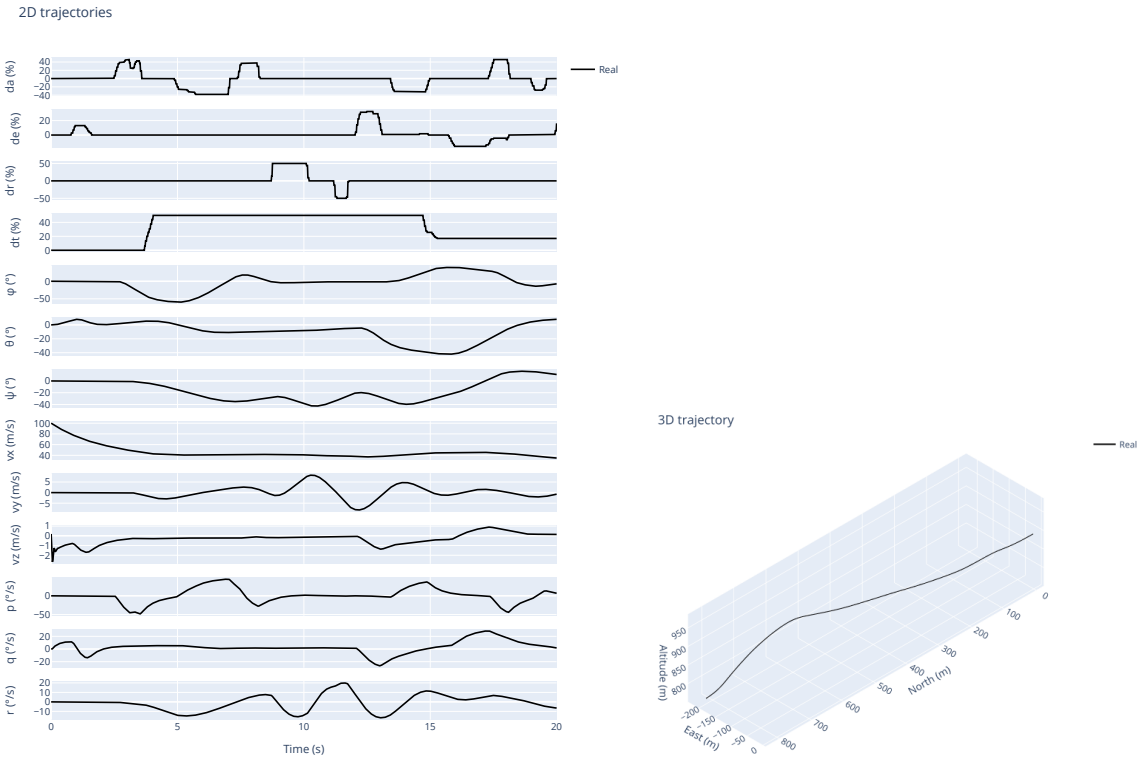

Figure 2.10: Example of aircraft dynamic states and control history. Control and state trajectories can be observed at the left graph. The right graph includes the 3D positional trajectory.

function, which relates the 26 aerodynamic derivatives with the difference between resulting states after simulation and recorded states (the exact fitness function used is described in the next sections). One of the leading solvers for these kinds of problems are a type of evolutionary algorithms known as evolution strategies (ES). These algorithms were invented in the 1960s for their application to shape optimization problems (Eiben and Smith, 2015).

ES belong to the field of evolutionary computing. These type of algorithms are based on some principles of biological evolution, like variation and selection. Individuals, representing candidate solutions, are generated in each new generation by variation, usually in the form of recombination from parents and mutation. Then, individuals are evaluated and selected for survival of the fittest. This process is repeated iteratively, and generally there is a tendency of the best individuals to survive and pass their *chromosomes* to the subsequent generation of individuals, eventually converging towards a function optimum.

2.3.1 CMA-ES: Covariance matrix adaptation evolution strategy

Within the different ES which have been previously applied to similar problems, the CMA-ES (Covariance Matrix Adaptation Evolution Strategy) has been selected to solve the optimization problem presented in this work. This algorithm is usually used to solve complex nonlinear non-convex black-box optimization problems in continuous domain. CMA-ES is a stochastic search method which excels in robust rugged search landscapes, and can also effectively address ill-conditioned, non-separable cost functions. The algorithm is typically applied to search spaces ranging from 3 to 100 dimensions, for unconstrained or bounded constraint optimization problems. It performs well in both local and global optimization problems and is among the leading algorithms for optimization of complex real-valued functions. Most common applications of CMA-ES include curve fitting techniques and shape optimization (Hansen, 2016b,a).

This evolutionary algorithm provides some advantages compared to other faster approaches (e.g. algorithms based on gradient descent). The function to solve is complex, which hinders the analytical calculation of its partial derivatives. CMA-ES algorithm does not require any definition of gradients for the objective function. Additionally, it is known to perform well in global optimization problems, and therefore has the capability to not get *stuck* in local optima. This is important for the problem in hand, since the complex function to solve is non-convex and therefore has multiple minima.

CMA-ES algorithm was presented in Hansen and Ostermeier (1996), and later extended in Hansen and Ostermeier (1997) and Hansen and Ostermeier (2001). The algorithm acts on a vector space, where the chromosome of each individual corresponds to a set of real values or *genes*. In this case, each individual is represented by 26 genes, corresponding to the aerodynamic derivatives (table 2.1). An initial population is randomly generated, and subsequently each individual is evaluated using a predefined fitness function which has been designed to grow proportionally to the difference between the candidate and reference trajectories. A sequential loop is then executed that seeks the

convergence of the individuals to potential function minima until reaching termination condition. Each population of new individuals is generated by sampling a multivariate normal distribution with some specific mean and covariance. The normal distribution is scaled with a standard deviation parameter, also called step-size, at each generation. Mean, covariance and step-size are updated after each generation by taking some information from their parents. The new mean of the search distribution is a weighted average of μ selected parents. The different weights assigned to the parents can also be interpreted as a selection mechanism. The covariance matrix is updated by a mechanism which exploits information from the entire population (known as rank-μ update) and information of correlation between generations (known as rank-one update). The former is important in large population, the latter in small populations. Finally, the step-size, or standard deviation parameter, is adapted by taking into account the direction of the evolution path in different generations, a method known as *cumulative step length adaptation* (CSA). The algorithm is terminated when further function evaluations do not lead to significantly better results. This effect is supervised using multiple criteria, which generally consider standard deviations in the covariance matrix and objective function values. A more detailed description of CMA-ES algorithm can be found in Hansen (2016b).

2.3.2 Fitness function definition

The fitness function definition is a critical part of the framework. This function defines the geometry of the search landscape, and therefore directly affects the capability of the optimizer to find optimum points. Figure 2.11 shows some example reference state trajectories alongside the trajectories of a non-optimal individual. The trajectories associated with the represented individual need to be compared with the reference trajectories so that any difference between them gets penalized in the form of an increment in the fitness value.

The state history comparison is performed by considering the weighted sum of a set of components. Each weighted component corresponds to the mean euclidean distance between three-dimensional vectors alongside the complete trajectory. The three-dimensional vectors considered include the aircraft linear and angular velocities, position and orientation. Each term of the fitness function satisfies equation 2.18:

$$f(\overrightarrow{x}) = \sum_{i=1}^{K} W_i f_i(\overrightarrow{x}) + f_{penalty}(\overrightarrow{x}) \tag{2.18}$$

Where $f(\overrightarrow{x})$ represents the fitness value for candidate $\overrightarrow{x}$ and K denotes the number of weighted components considered. Each of them is assigned a weight, representing W_i the weight assigned to component i. The individual fitness contribution of each component is calculated with equation 2.19.

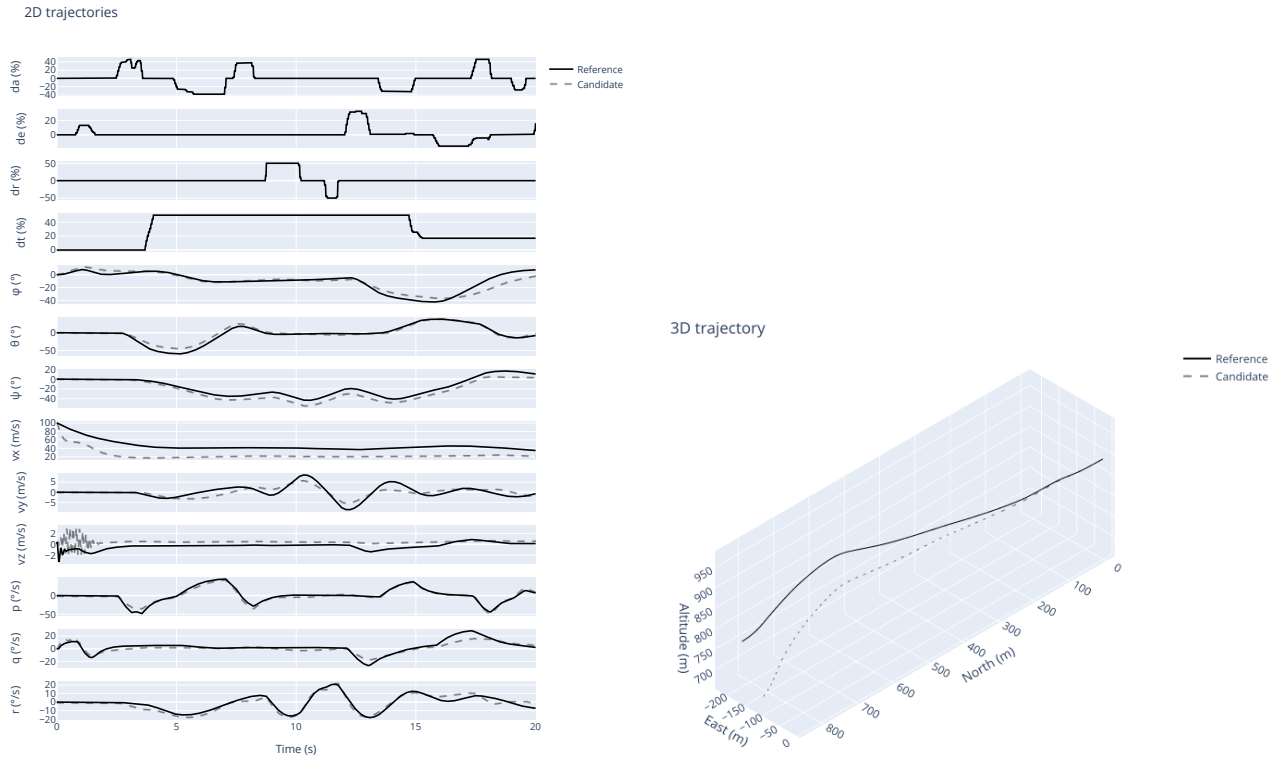

Figure 2.11: Example reference (solid lines) and candidate (dashed lines) state trajectories.

$$f_i(\overrightarrow{x}) = \frac{\sum_{j=1}^{N} \sqrt{\left(x(i)_j^{ref} - x(i)_j^{cand}\right)^2 + \left(y(i)_j^{ref} - y(i)_j^{cand}\right)^2 + \left(z(i)_j^{ref} - z(i)_j^{cand}\right)^2}}{N} \tag{2.19}$$

Where the term N denotes the number of discrete time steps sampled in the candidate trajectory. Variables $x(i)_j$, $y(i)_j$ and $z(i)_j$ represent the three-dimensional components of the dynamic state corresponding to fitness contribution component i at the discrete time step j. Each fitness function contribution outputs zero if the corresponding state histories are identical for candidate and reference, and outputs a positive value otherwise. Several combinations of weighted components are possible. The complete fitness definition ($K = 4$) includes the components due to position ($x \equiv p_N, y \equiv p_E, z \equiv p_D$), orientation expressed as the propagation of angular velocities ($x \equiv \int p\,dt, y \equiv \int q\,dt, z \equiv \int r\,dt$), linear velocity ($x \equiv v_x, y \equiv v_y, z \equiv v_z$) and angular velocity ($x \equiv p, y \equiv q, z \equiv r$), but any combination of these components, using at least one, is possible (the chosen default combination is detailed in section 3.1.1). The weights W_i are supposed to be chosen for achieving a similar mean scaling in each of the fitness contribution terms, since the scaling depends on the unit magnitude chosen for their corresponding dynamic states (meters for position, meters per second for velocity and radians per second for angular velocity).

The penalty term $f_{penalty}$, consists in the sum of the absolute value of all scaled coefficients (equation

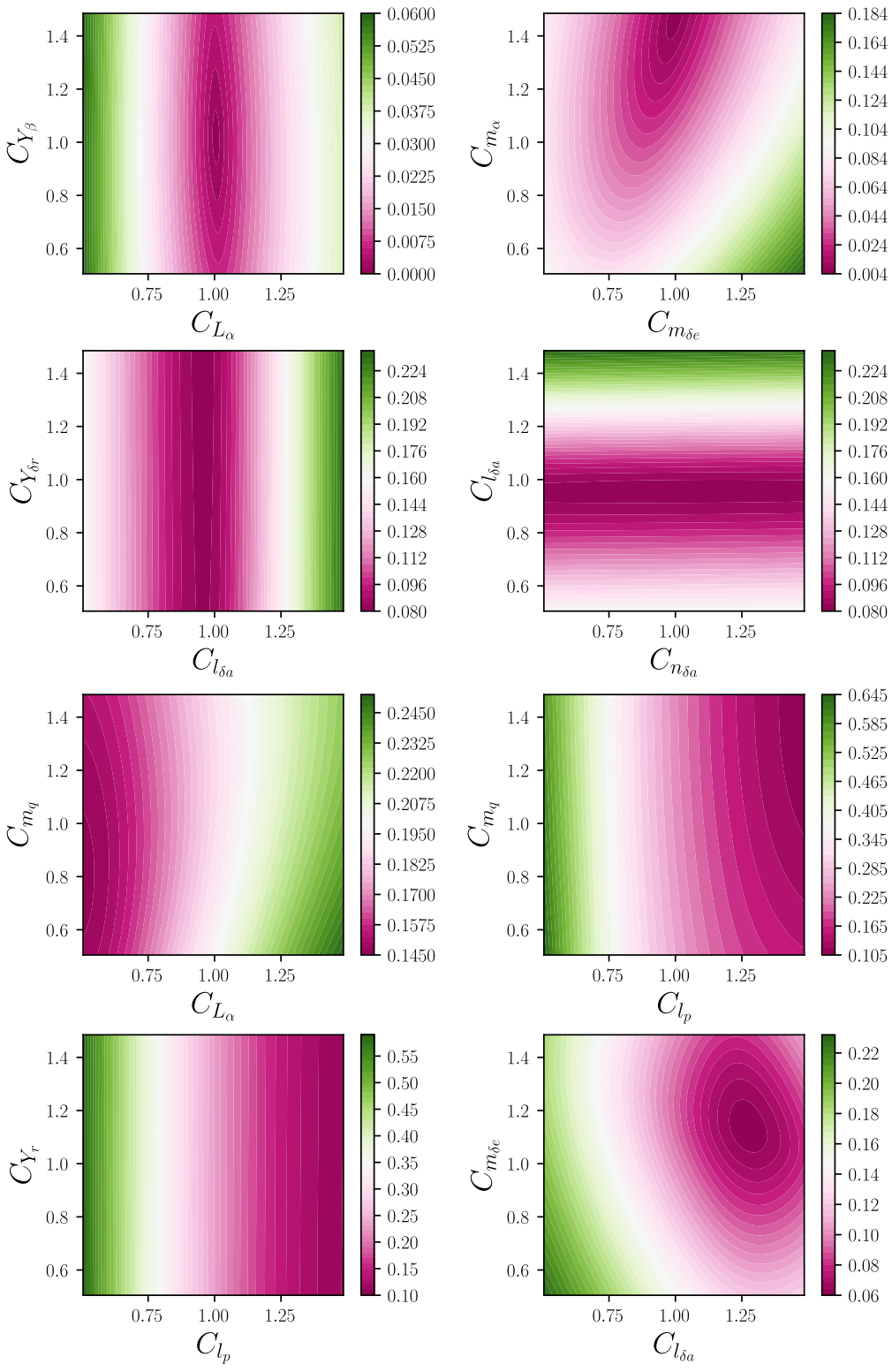

Figure 2.12: Effect of scaling parameter pairs in the resulting fitness. Axes values represent the scale factor applied to each coefficient. The resulting fitness value is represented as a color map.

2.20). This penalty term tries to give more weight to solutions with relatively small normalized aerodynamic coefficients. This is done to maintain physical consistency of the optimized solution, since extraordinarily large values of aerodynamic coefficients are not realistic.

$$f_{penalty}(\overrightarrow{x}) = \sum_{i=0}^{N} |\tilde{x}_i| \tag{2.20}$$

When optimizing using simulated data, considering no stochastic noise such as turbulence is introduced, the minimum achievable fitness is zero at the global minimum, since all dynamic states associated to fitness contribution terms (position, orientation, linear velocity and angular velocity) are identical to the reference trajectories at this point. However, when trying to optimize from real flight data the problem enters the realm of multi-objective optimization, since each fitness contribution term would point the fitness function to a slightly different minimum. This is because the gray box model is not capable of perfectly describing the dynamics of a real aircraft, since it is an approximation. Therefore, a situation would take place where no individual criterion can be better off without making at least another criterion worse off or without any loss in the fitness function. This situation is commonly known as *Pareto efficiency* (Branke et al., 2008), and deals with the impossibility of improving one criterion without harming other criteria. Furthermore, the inclusion of multiple contributing components in the fitness function leads to an increment of the roughness of the search landscape, increasing the number of local minima and hindering convergence. To help salvaging this problem and ensuring a good convergence, the following action sequence is performed.

- (Ω+$f_{penalty}$) Only angular velocities are considered in equation 2.19 during a first optimization run ($K = 1$). The penalty term $f_{penalty}$ is also included. Given that only two individual components affect the fitness function, the search landscape is reasonably smooth, facilitating convergence to the global optimum. Angular velocity is chosen as the first variable to consider in the optimization due to the hypothesis that the control surfaces have a very direct influence in this state. The control has less direct power in the rest of the states (except for linear velocity in the x-body axis with throttle control), and generally more coupled control interactions are required. Therefore, it is hypothesized that the search landscape may be smoother using this dynamic state, which helps convergence.

- (V+Ω+$f_{penalty}$) A second optimization is performed, using the solution resulted from the previous run as initial point. This time, linear and angular velocities are considered in equation 2.19 ($K = 2$). The search landscape is now less smooth since an additional component is contributing to fitness, but the initial point, inherited from the previous optimization result, should be relatively close to the new global optimum, easing convergence.

The contributions of position and orientation have not been considered in the fitness function for two main reasons. On the one hand, in the case that reference linear and angular velocities are achieved

by the optimized solution, this implies that position and orientation would also be very similar to the reference, since they exclusively depend on the propagation of those variables. This means that the variables are not making an appreciable contribution to the fitness function, but do contribute to an increase of the search landscape roughness. On the other hand, a precise response in linear and angular velocities in dynamic models for control techniques are a priority, since control algorithms tend to interact with these variables in a lower level.

The fitness function produces a complex search landscape. In order to gain some insight in this geometry, a simple visualization method has been used. Given that the search landscape has 26 degrees of freedom, pairs of parameters have been selected for performing a sweep around different values and the corresponding fitness results have been represented visually in figure 2.12.

It is important to note that some combinations of aerodynamic coefficients may lead to unstable dynamics, and eventually the divergence of some dynamic states to infinity, causing invalid solutions and increasing the simulation time required for valid data generation.

The integration of the CMA-ES algorithm into the framework has been performed using an open-source library called pycma[1], a Python implementation of CMA-ES algorithm which uses the scientific computing library NumPy at its core, which in turn implements most of its routines in C++, improving efficiency, and allowing the algorithm to execute in parallel using multiple CPU cores.

2.3.3 Implementation details

Some practical hints concerning the implementation of CMA-ES algorithm for solving a particular problem are included in this section (Hansen, 2016a).

2.3.3.1 Encoding of variables

An adequate encoding of variables is critical for the algorithm optimization performance. Essentially, all parameters acting in the vector space should be transformed in order to have the same sensitivity. This justifies the choice of the identity matrix for the covariance matrix initialization. This can be achieved by scaling all parameters so that their typical domain corresponds to a fixed range (e.g. from 0 to 10). Scaling is not the only possible operation, more complex transforms may be required depending on the nature of each parameter and how it affects the objective function.

Typically, this encoding process is performed by implementing a wrapper function which envelops the objective function transforming the input parameters for a correct encoding. When the optimizer converges, the generated solution should also be mutated in order to reverse the transformation.

[1] Available at the repository `https://github.com/CMA-ES/pycma`

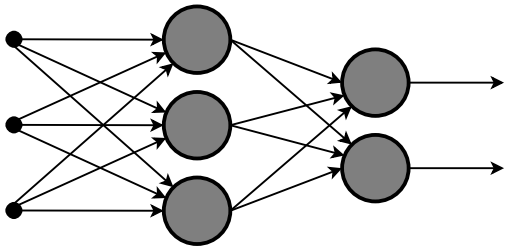

Figure 2.13: An artificial neural network with three layers. Input layer is shown on the left side of the image. The hidden layer follows at the center, and finally the output layer is shown on the right side of the image (source: Wikimedia Commons, public domain)

2.3.3.2 Boundary and constraint handling

Parameter boundaries can be handled by using specific transformations as variable encoding. For example, a lower bound a and an upper bound b can be imposed for the possible values of parameter x by encoding it with the transformation $y = a + (b - a) \cdot (1 - \cos(\pi \cdot x/10)/2)$, which is a strictly increasing bijective mapping into $[a, b]$ in the interval $0 \leq x \leq 10$.

The proposed method is good for handling parameter boundaries (box-constraints), however it cannot handle non-linear constraints. For this, another method can be used which consists of adding penalty terms to the fitness function value for infeasible solutions. This method, however, may work poorly if the optimum is close to the domain boundary.

2.3.3.3 Initial point and standard deviation

Initial point $\vec{x}_0$ and initial standard deviation (step-size) σ_0 must be chosen by the user. Ideally, these parameters should be chosen such that the final desirable solution is inside the range $\overrightarrow{x}_0 \pm 3\sigma_0$ in each coordinate. Therefore, large values of σ_0 are appropriate when searching for a global optimum on rugged or multimodal landscapes (large initial solution position uncertainty), while small values of σ_0 are appropriate when trying to improve a given solution locally (small initial solution position uncertainty).

2.4 Deep neural network

Artificial neural networks are computing systems based on collections of nodes, called neurons, which can transmit information, traditionally in the form of real numbers, to other connected neurons in the network. The output of neurons are typically computed by some non-linear function applied to the sum of its inputs. During the information transmission between neurons, some transformation occurs in each connection, typically in the form of adjustable scale factors, called weights, and offsets, called biases. These weights and biases can be intelligently adjusted for mapping complex functions

in a process called neural network training. Generally, nodes are connected in a layered architecture, composed of an input layer which processes raw input data, an output layer representing the output predictions of the network, and a number of hidden intermediate layers which process information from the input layer to the output layer (figure 2.13). When an ANN contains a deep stack of hidden layers, it is called a deep neural network (DNN) (Géron, 2019).

The process of neural network training for adjusting the weights in order to perform the mapping between inputs and outputs is generally performed using a technique known as backpropagation, introduced in Rumelhart et al. (1986). Backpropagation is based on a technique known as gradient descent (GD), an iterative optimization algorithm for finding a local minimum of a differentiable function by taking repeated steps in the opposite direction of the function's gradient. During training, a loss function is selected which seeks to minimize the difference between the output of the neural network after some input sample, and its reference output label. In backpropagation, the gradients of the complete loss function are automatically computed in two sequential passes of the network, one forward pass and one backward pass, using the chain rule of calculus which defines the derivative of a function composed of other differentiable functions. Once the gradient of the function is known, a gradient descent step can be performed in order to tweak weights and biases to decrease inference error. This process is repeated sequentially until convergence to the solution occurs, and mapping between inputs and outputs is achieved (Géron, 2019).

As stated in section 2.1.2, neural networks can be trained to perform the mapping between aerodynamic derivatives and measured flight data (figure 2.2). Simulation techniques can be used to facilitate the process of data gathering for network training, using automatic flight control systems coupled with aircraft mathematical models.

Depending on the problem, some neural network architectures may require vast amounts of labeled samples during training for achieving an efficient mapping of samples and labels. Obtaining such amounts of data from different real aircraft flights, whose aerodynamic coefficients are fully known, is unfeasible. Similarly, generating simulated trajectories commanding manual control inputs in real time via a joystick device, similarly to the case of the evolutionary framework evaluation, is unpractical for the scope of this work. The only alternative to generate the required amounts of data is to automatically generate trajectories using the dynamics simulator and an automatic logic for commanding controls. Ideally, the automatic control logic could be performed by means of an inversion-based controller, generating random trajectories that are realistic in terms of the flight envelope (only possible for individuals that are controllable, which is not guaranteed in the entire population). Having more realistic trajectories available in the training samples can improve the ability of the network to generalize to real flights. However, the development of an inversion-based controller for the aircraft model has not been considered in the scope of this work. Instead, control inputs have been randomly generated during trajectory generation. For this reason, some samples may correspond to unrealistic trajectories (e.g. inverted flight, high pitch maneuvers, etc..). Nevertheless,

the set of realistic trajectories is contained within the total set of generated samples, and thus real dynamics can, in principle, be identified from realistic trajectories using this approach. Initial variable states are also randomly chosen for the starting point of each simulation. This allows to use a trained network for identification of different flight data sets, regardless of their initial conditions.

2.4.1 Recurrent neural networks

Recurrent neural networks (RNN) are a special class of ANNs where connections between neurons can flow in opposite direction to neurons of previous layers, allowing them to exhibit temporal dynamic behavior (Géron, 2019). RNNs have demonstrated good performance for time-series data prediction and classification (Smirnov and Mephu Nguifo, 2018). They also perform well in parameter estimation problems (Petridis and Kehagias, 2001; Amiruddin et al., 2020). For these reasons, RNNs have been selected for solving the parameter identification problem proposed in this work. Two RNN architectures, namely GRU (Chung et al., 2014) and LSTM (Hochreiter and Schmidhuber, 1997) have been selected for evaluation. These architectures implement storage inside the network in the form of more complex units, called cells, which incorporate time delays or feedback loops, and have demonstrated good performance in parameter estimation problems from time-series data.

The neural network training and inference process has been implemented into the framework using a Python library for machine learning called TensorFlow. More specifically, a higher-level API included in TensorFlow, named Keras, has been used for building, training, evaluating and running the required neural networks for aerodynamic parameter identification from flight data.

2.5 Graphical elements

The graphical elements of the framework have been implemented for improving user interaction and model visibility, as well as facilitating some necessary tasks such as flight data generation via simulation.

2.5.1 Graphical engine

A graphical engine has been implemented into the framework for a visual representation of the aircraft dynamics during simulation. This visualization can be useful for several reasons:

- Allows for a better perception of the dynamic model's response to input commands.
- Helps in simulating realistic trajectories for flight data generation via manual joystick commands.

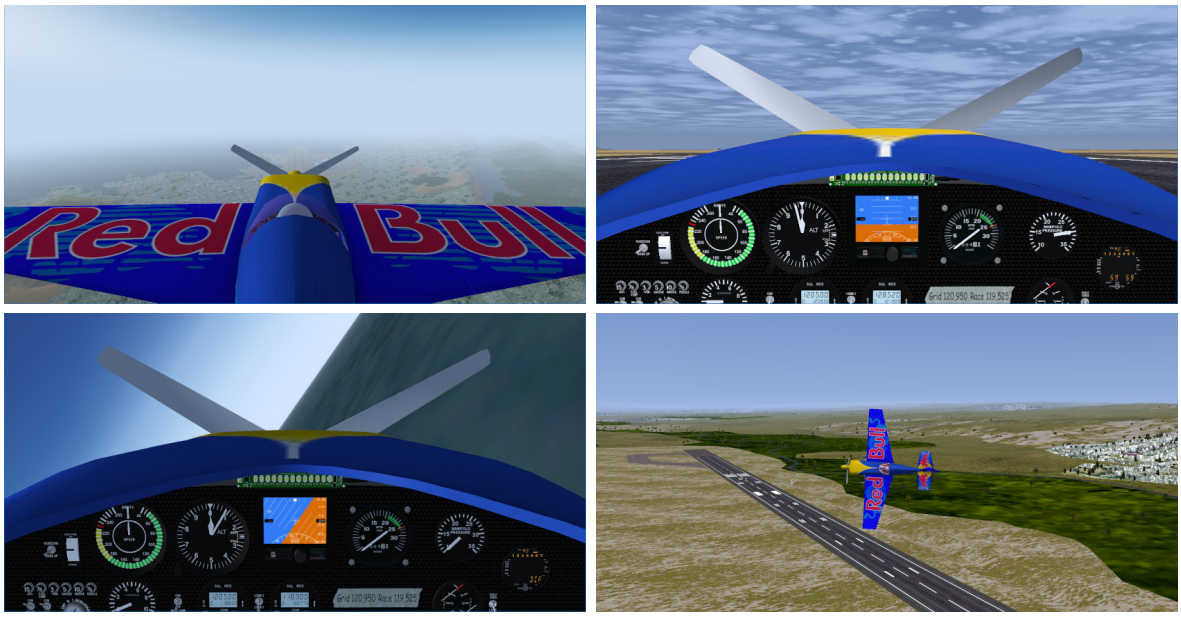

Figure 2.14: Different snapshots of the graphical engine used in the framework.

- Provides better perception of a control system's performance. Control surface deflections and rotor angular velocity, if applicable, can also be rendered on the visualization, as well as other elements which can help in perception of the state of the system, such as lights, landing gear retraction, etc.

The open-source flight simulator *FlightGear*[2] (FG) has been selected for this task. This open-sourced software has been used extensively in the aerospace research industry for vehicle simulation and pilot training. The software includes two main modules: (1) An independent flight dynamics model engine (FDM) named *JSBSim*[3], which defines the movement of the aircraft based on a configurable physics model, and (2) a graphical engine capable of realistic aircraft and atmospheric rendering, including elements such as realistic terrain, clouds, precipitation, atmospheric halos, directional light scattering, among others.

For the purpose of this work, the main interest is the graphical engine, since the flight dynamics model has been independently implemented for the framework (section 2.2.1), allowing more flexibility. For this purpose, FG includes optional bindings which allow sending the aircraft dynamic states in real time. This functionality can be used for feeding the graphical engine with the states generated by a custom dynamic model. The dynamic information is sent via a UDP port at a frequency of 60 Hz for real time rendering. The message is completely customizable, and in the present implementation includes aircraft position and orientation, control surface deflections and rotor angular velocity. Different snapshots of the graphical engine are shown in figure 2.14.

[2]FlightGear simulator official website: `https://www.flightgear.org/`
[3]JSBSim official website: `http://jsbsim.sourceforge.net/`

2.5.2 Graphical user interface

A graphical user interface (GUI) has been included in the framework for facilitating user interaction with the different supported functionalities. This GUI will also improve control and configuration of the tool in case the framework grows and extended capabilities are included.

This module uses the framework *Dash Open Source*, developed by the company *Plotly*[4]. Dash is an open-sourced framework for building web-based analytic applications, and it is commonly used for developing artificial intelligence and data science apps in the industry. The main advantage of web-based applications, when compared to traditional desktop applications, is the ease of accessibility to other users. The framework allows Python as the main language for app implementation, which facilitates integration with the identification framework.

A simple GUI has been designed which allows to select an input data file and shows information about. The input data file selected for optimization must be a locally stored file containing flight data in comma-separated values (CSV) or *Microsoft Excel* formats. An example CSV file snippet containing flight data is shown on listing 2.1.

The required dynamic states are controls, attitude angles, position, and linear and angular velocities. Time step needs to be set as an internal parameter in the optimization framework. File data parsing is performed with the Python library *Pandas* for data analysis.

After parsing the data file, the GUI displays a snippet of the first few lines of the data file in table format, as well as a representation of the dynamic state histories. These are contained within two graphs, a 3-dimensional graph displays position history while a 2-dimensional graph displays the history of the rest of the states. At this point, if the user clicks the "optimize" button, then the optimization takes place, and results are displayed along the existing graphs when the optimization terminates. Figure 2.15 shows the complete user interface after identification displaying real and optimized results.

[4]Plotly Dash official website: `https://plotly.com/dash/`

Listing 2.1: Snippet of an input CSV file example containing flight data.

```
1 ,da,de,dr,dt,roll,pitch,yaw,posNorth,posEast,posDown,vx,vy,vz,p,q,r
2 0,0,0.04,0,0.64,0,0.03,0,327.24,0,-930.53,49.52,0,-1.03,0,-0.19,0
3 1,0,0.02,0,0.63,0,0.02,0,328.06,0,-930.57,49.48,0,-1.02,0,-0.19,0
4 2,0,0.01,0,0.64,0,0.02,0,328.88,0,-930.61,49.44,0,-1.03,0,-0.18,0
5 3,0,0.01,0,0.64,0,0.02,0,329.71,0,-930.65,49.41,0,-1.02,0,-0.18,0
```

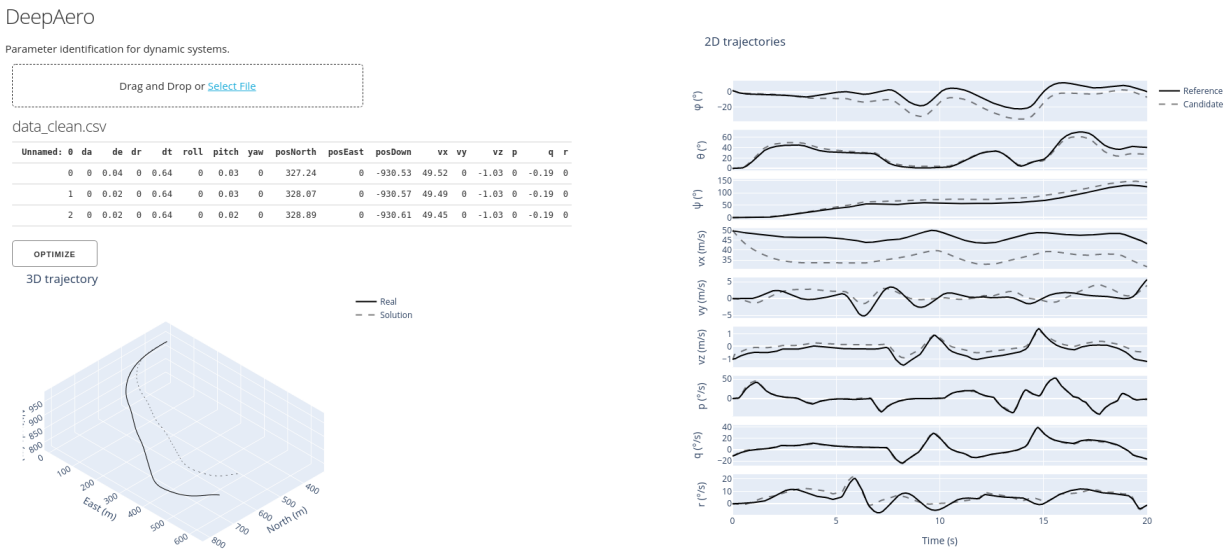

Figure 2.15: User interface displaying results after evolutionary optimization.

Chapter 3

Results and discussion

This section presents the resulting capabilities of the evolutionary and deep learning frameworks for system identification. Both frameworks are compared to similar methods in the field. Subsequently, both methodologies are compared with each other.

3.1 Evolutionary framework

This section presents the resulting capabilities of the evolutionary framework for system identification. The evaluation process has been performed in two parts. First, an objective evaluation is carried out using simulated flight data. Next, the capabilities for identification of real aircraft flight dynamics are tested. Finally, results are compared to similar methods in the field.

3.1.1 Identification from simulated trajectories

An objective analysis of the evolutionary framework identification capabilities has been performed. This has been accomplished by generating a flight dataset using simulation and manually commanding the aircraft via a joystick device. The generated dataset is then fed to the identification framework, allowing the identified model to be directly compared with the actual model used in the simulation. The reference model used for dataset generation via simulation is presented in the following tables. Table 3.1 shows the aerodynamic derivatives used for the reference model, and table 3.2 shows some auxiliary parameters which need to be defined for an actual simulation. This parameters are not considered part of the free phenotype parameters (aerodynamic derivatives in this case), but part of the dynamic model, directly affecting the fitness function. In a real practical identification case, auxiliary parameters should be estimated a priori, and imposed to the identification framework before optimization. Additionally, it is important to note that the penalty term of the fitness function ($f_{penalty}$, defined in equation 2.20) is not necessary for the identification of simulated dynamics, since the algorithm convergence to solution is simpler than the real aircraft case, where the gray box model

Table 3.1: Reference aerodynamic derivatives for simulation

$\overrightarrow{x}_{ref}$												
C_{D_0}	K	C_{D_β}	C_{Y_β}	$C_{Y_{\delta a}}$	$C_{Y_{\delta r}}$	C_{Y_p}	C_{Y_r}	C_{L_0}	C_{L_α}	C_{l_β}	$C_{l_{\delta a}}$	$C_{l_{\delta r}}$
0.05	0.01	0.15	-0.4	0	0.19	0	0.4	0.1205	5.7	-0.0002	-0.33	0.021
C_{l_p}	C_{l_r}	C_{m_0}	C_{m_α}	$C_{m_{\delta a}}$	$C_{m_{\delta e}}$	$C_{m_{\delta r}}$	C_{m_q}	C_{n_β}	$C_{n_{\delta a}}$	$C_{n_{\delta r}}$	C_{n_p}	C_{n_r}
-0.79	0.075	0	-1.23	0	-1.1	0	-7.34	0.21	-0.014	-0.11	-0.024	-0.265

Table 3.2: Auxiliary parameters for simulation

P_{ref}						
$m\,[kg]$	$g\,[m/s^2]$	$\rho\,[kg/m^3]$	$S\,[m^2]$	$T_{max}\,[N]$	$b\,[m]$	$c\,[m]$
750	9.8056	1.225	9.84	7000	7.87	1.25
$I_x\,[kg \cdot m^2]$	$I_y\,[kg \cdot m^2]$	$I_z\,[kg \cdot m^2]$	$I_{xz}\,[kg \cdot m^2]$	$i\,[°]$	$V_w\,[m/s]$	$\theta_w\,[°]$
3531.9	2196.4	4887.7	0	0	0	0
	$\varphi_w\,[°]$	$\sigma_w\,[m/s]$	$\tau_s\,[s]$	$\tau_e\,[s]$	$V_0\,[m/s]$	
	0	0	0	0	100	

only approximates the solution. Therefore the $f_{penalty}$ term has not been taken into account in the simulation included in this section.

The reference model has been inferred using publicly available data of the Zivko Edge 540 (figure 3.1), a small aerobatic single-engine aircraft commonly used in aerobatic competitions (Dreyer and Jensen, 2014).

For achieving an objective framework evaluation, the following metrics have been used.

- Mean best fitness (MBF). Average of the minimum fitness value obtained among all executions.
- Mean solution L_1 norm distance at best fitness (MSD). The distance between the resulting solution and reference points in the search space can provide some insights about the effectiveness of the fitness function to properly encode the problem. That is, it is of no use obtaining a very low fitness value if the resulting aerodynamic derivatives do not really match the real ones. If this happened, it could mean that a good solution was found that could fit the real results in the problem domain, but may not generalize well to other set of control inputs. This could indicate that the model may be unidentifiable with the available observations, and perhaps more observations could help in improving model identifiability. The solution vector to compare in each run is the one corresponding to the individual which generates the minimum fitness. The resulting metric is averaged for all executions.

  For estimating the distance between the solution vectors, L_1 norm is used, also known as

Figure 3.1: Zivko Edge 540 aerobatic aircraft at the Red Bull Air Race competition (author: Ignacio Ferre Pérez, some rights reserved)

Manhattan distance. This distance metric is commonly used for estimating the similarity between high dimensional vectors. For these applications, Manhattan distance (L_1 norm) is shown to be better than euclidean distance (L_2 norm), since it better preserves the meaningfulness of proximity measures (Aggarwal et al., 2001). Manhattan distance between two vectors is the sum of the absolute differences of their coordinates.

- Fitness success rate (FSR). Ratio of successful executions with respect to the total number of executions, assuming that a successful execution requires obtaining a minimum fitness value lower than 0.01.

- Distance success rate (DSR). Ratio of successful executions with respect to the total number of executions, assuming that a successful execution requires obtaining a minimum Manhattan distance lower than 5.

- Average number of evaluations to a solution (AES). Average number of necessary function evaluations for reaching the solution of minimum fitness.

Each hyperparameter is tested by individually modifying its value parting from the default hyperparameter setup (table 3.3). Each problem configuration is tested by averaging the resulting metrics of 10 total optimization runs. All hyperparameters are described in the following sections.

Computation time for a single fitness evaluation is approximately 200 microseconds for the reference model and hyperparameters in the setup used (table 3.4), which allows a rate of approximately 5000 evaluations per second.

Table 3.3: Reference optimization hyperparameters

$\vec{x}_{0_{ref}}$												
C_{D_0}	K	C_{D_β}	C_{Y_β}	$C_{Y_{\delta a}}$	$C_{Y_{\delta r}}$	C_{Y_p}	C_{Y_r}	C_{L_0}	C_{L_α}	C_{l_β}	$C_{l_{\delta a}}$	$C_{l_{\delta r}}$
0.1	0.1	0.1	-0.1	0.1	0.1	0.1	0.1	0.1	1	0.1	-0.1	0.1
C_{l_p}	C_{l_r}	C_{m_0}	C_{m_α}	$C_{m_{\delta a}}$	$C_{m_{\delta e}}$	$C_{m_{\delta r}}$	C_{m_q}	C_{n_β}	$C_{n_{\delta a}}$	$C_{n_{\delta r}}$	C_{n_p}	C_{n_r}
-1	0.1	0.1	-1	0	-1	0	-1	0.1	-0.1	-0.1	-0.1	-0.1

$\sigma_{0_{ref}}$	C_x	$\sigma_{w_{ref}}\,[m/s]$	$T_{ref}\,[s]$	$f_{ref}\,[Hz]$	λ_{ref}
0.2	-	0	20	60	13

Table 3.4: Setup used for framework evaluation

CPU	Intel Core i5-4670K (4 cores at 3.40 GHz)
GPU	NVIDIA GTX 1060 (6 GB VRAM)
RAM	8 GB DDR3 1866 MHz
OS	Debian 10 64 bits

3.1.1.1 Initial point, standard deviation and variable encoding

CMA-ES algorithm requires to set an initial point ($\vec{x}_0$). Individuals are sampled using this point as the normal distribution mean during the first generation. In practice, this means the user is required to set a *best guess* of where the solution lies. For the case of this evaluation problem, an initial point has been chosen by approximating each real aerodynamic coefficient with its order of magnitude (table 3.3). In a real case, this initial point could lead to good results, since aircraft aerodynamic derivatives tend to be within the same order of magnitude for similar aircraft geometries. For evaluation purposes, other initial points have been individually tested, parting from the default hyperparameter settings, including vectors where all terms are assigned the same value (0.05, 0.1 or 0.2). Resulting metrics are shown in table 3.5.

Initial standard deviation (σ_0) also needs to be set by the user. As stated in section 2.3.3, an

Table 3.5: Identification metrics with different initial points

$\vec{x}_0$	MBF	MSD	FSR	DSR	AES
$\vec{x}_{0_{ref}}$	2.18e-4	4.00e-1	1.0	1.0	5.17e+4
$0.\vec{0}5$	1.04e+0	1.50e+1	0.4	0.4	3.39e+4
$0.\vec{1}$	1.27e+0	4.13e+1	0.4	0.4	4.33e+4
$0.\vec{2}$	2.05e+0	4.67e+1	0.2	0.2	2.08e+4

Table 3.6: Identification metrics with different initial standard deviations

σ_0	MBF	MSD	FSR	DSR	AES
0.0001	5.12e-3	7.23e+0	1.0	0.2	2.24e+4
0.001	2.20e-4	3.79e-1	1.0	1.0	5.15e+4
0.01	2.12e-4	3.68e-1	1.0	1.0	5.37e+4
0.2	2.18e-4	4.00e-1	1.0	1.0	5.17e+4
0.6	2.02e-3	2.28e+1	0.9	0.9	6.59e+4
2	5.58e-1	1.91e+2	0.1	0.1	4.58e+4

Table 3.7: Identification metrics with different parameter encodings

C_x	MBF	MSD	FSR	DSR	AES
-	2.18e-4	4.00e-1	1.0	1.0	5.17e+4
$C_{m_q}, C_{L_\alpha} \times 10^{-2}$	2.11e-1	3.68e-1	0.3	0.3	5.53e+4

adequate encoding of variables is critical for the algorithm optimization performance, so these should be transformed in order to have the same sensitivity. Also, σ_0 should be ideally chosen such that the final solution lies within the range $\vec{x}_0 \pm 3\sigma_0$ in each coordinate. Taking into account the initial point chosen ($\overrightarrow{x}_{0_{ref}}$, in table 3.3) and the real solution point ($\vec{x}_{ref}$, in table 3.1), one could estimate that an initial standard deviation of 0.2 and a variable encoding consisting on applying a scale factor of 10^{-2} to C_{m_q} and C_{L_α} parameters seems to be adequate, given that with these conditions the following statement is fulfilled:

$$\max\left(\left|\vec{x}_{ref} - \vec{x}_{0_{ref}}\right|\right) = 0.3 < 3 \cdot \sigma_0 = 0.6 \tag{3.1}$$

This transformation has been chosen since parameters C_{m_q} and C_{L_α} are expected to be at least one order of magnitude bigger than the rest of parameters. As for the rest of parameters, they have not been scaled since their expected values can be significantly bigger than the defaults. Therefore, their original scale is maintained so that the initial population lies within a bigger range of their respective dimensions.

Multiple initial standard deviations and two methods for parameter encoding have been evaluated individually, parting from the default hyperparameter settings. Resulting metrics are included in tables 3.6 and 3.7.

Results show that the reference hyperparameter chosen for initial point is more effective than initialization with the same value in all coefficients. This reinforces the idea that this initial point is closer to the global minimum in the euclidean search space, since aircraft aerodynamic derivatives tend to be within the same order of magnitude for similar aircraft geometries. Additionally, perfect success

rates are found with initial standard deviations in the range of 0.001 to 0.2. Larger values may reduce the probability of the population to converge to the global minimum, since a larger region of the search landscape is covered by the same number of individuals. Lower values may reduce the probabilities of individuals to initialize relatively close to the global minimum, and it seems that local minima are found which meet the fitness success criteria, but not the solution distance success criteria. This may be indicative that the fitness function definition could be improved for increasing the search landscape convexity, reducing the number of local minima close to the global minimum in fitness, but representative of invalid solutions. Finally, it seems an initial standard deviation of 0.01 may lead to slightly better performance compared to the default chosen value of 0.2. Surprisingly, the parameter encoding transformation via a scale factor of 10^{-2} for C_{m_q} and C_{L_α} parameters, although producing smaller fitness values, leads to a very low success rate of 30%, whereas no parameter encoding achieves a perfect success rate. This may be caused because the probabilities of initial individuals to lie close to the solution in these dimensions diminish when scaling these parameters.

3.1.1.2 Robustness against noise

The evaluation methodology is up to this point an ideal case for the optimization framework. This is because, given that the reference model data for identification has been generated with the nonlinear aircraft model proposed in section 2.2.1, it is possible for the framework to obtain very small fitness values, and the search landscape is relatively simple. This is not the case for real aircraft, given that the nonlinear model used is simply an approximation of reality. In order to add some difficulty to the process, stochastic noise has been introduced into the dynamics in the form of wind turbulence (more details in 2.2.1).

For these configurations, the parameters for mean wind velocity (V_w), mean wind direction pitch (θ_w) and mean wind direction azimuth (φ_w) have been set to zero. The only term which has been modified to different values is the standard deviation for the stochastic component representing wind turbulence intensity (σ_w). It is important to point out that each test has been performed with an independent flight log file, although the performed maneuvers were similar. Multiple wind turbulence intensity standard deviation values have been evaluated individually, parting from the default hyperparameter settings. Results are shown in table 3.8.

As expected, achieved fitness value increases with noise. This is due to the fact that the ideal nonlinear model is not able to accurately describe the resulting trajectories when a stochastic noise component is present in data. More important is the fact that the framework is able to find the solution even when noise is present. This means that the framework is relatively robust against noise, which may become useful for identifying real aircraft dynamics.

Table 3.8: Identification metrics with different turbulence intensities

$\sigma_w\,[m/s]$	MBF	MSD	FSR	DSR	AES
0	2.18e-4	4.00e-1	1.0	1.0	5.17e+4
0.1	9.73e-4	1.11e+0	1.0	1.0	5.53e+4
0.5	2.89e-3	1.18e+0	1.0	1.0	5.48e+4
1	6.33e-3	2.01e+0	1.0	1.0	5.19e+4

Table 3.9: Effect of time horizon in optimization performance.

Time horizon	Search landscape complexity	System identifiability
Small	Less complex (desirable)	Less identifiable (not desirable)
Large	More complex (not desirable)	More identifiable (desirable)

3.1.1.3 Effect of time horizon

Total flight time for the recorded data (T) is also a critical hyperparameter. On the one hand, if the time horizon is moderately small, then the search landscape becomes relatively simple, which improves convergence to solution. However, if the time horizon is too small, not enough information is contained within the data in order to allow system identification. This is, system identifiability with the available data is limited, which deteriorates the capability of convergence to the solution, since the fitness function does not lead to it. On the other hand, if the time horizon is moderately large, then the search landscape becomes more complex and difficult to optimize, but in change it improves system identifiability, or the capability of the fitness function to lead to the correct solution. This process is schematized in table 3.9.

Due to the opposing effects on the optimization performance, a careful analysis is required for finding the optimal value for this variable. Therefore, multiple values have been individually tested, parting from the default hyperparameter settings, and results have been included in table 3.10. Each test has been performed with an independent flight log file, although the performed maneuvers were similar.

The best result is achieved with a flight time horizon of 10 seconds. For a real aircraft identification, a stepped approach with increasing time horizon executions may improve the algorithm performance, since the first executions may help in providing good initial points for larger time horizon executions. This can be a good way of salvaging the disadvantages of each configuration and taking advantage of their benefits.

Table 3.10: Identification metrics with different time horizons

$T\,[s]$	MBF	MSD	FSR	DSR	AES
5	5.24e-3	5.19e+0	0.9	0.9	8.41e+4
10	7.25e-5	3.70e-1	1.0	1.0	6.54e+4
20	2.18e-4	4.00e-1	1.0	1.0	5.17e+4
50	4.86e-3	2.46e+1	0.9	0.9	5.39e+4
100	5.18e-2	1.77e+0	0.9	0.9	4.83e+4

Table 3.11: Identification metrics with different time steps

$f\,[Hz]$	MBF	MSD	FSR	DSR	AES
60	2.18e-4	4.00e-1	1.0	1.0	5.17e+4
120	5.24e-4	5.11e-1	1.0	1.0	6.02e+4
400	8.24e-5	5.94e-2	1.0	1.0	4.91e+4

3.1.1.4 Simulation time step

The time step used in numerical simulation, or inverse of simulation frequency (f), takes an important role in the algorithm performance. Numerical integration has cumulative integration errors associated, since it is a discrete process and therefore an approximation of the continuous case, with proportional precision to the integration time step. If time step is too large, then numerical integration errors may become sufficiently large to lead to unstable dynamics, and subsequently the divergence of some dynamic states to infinity. This causes numerous invalid solutions in the search landscape, which can negatively affect search performance. On the other hand, if time step becomes too small the computational load rapidly increases without significant improvement in integration precision. Several time step values have been individually tested, parting from the default hyperparameter settings, and results are included in table 3.11. Similarly to the two previous cases, each test has been performed with an independent flight log file, although the performed maneuvers were similar.

Results show that a simulation frequency of 400 Hz achieves the best algorithm performance. This improvement in fitness performance comes at an increased computational cost.

3.1.1.5 Restarts with increasing population size

Population size (λ) defines the number of individuals present in each algorithm generation. Small population sizes allow for fast convergence, whereas large population sizes improve the global search performance, since the probability of the entire population getting *trapped* near a local minima diminishes.

Table 3.12: Identification metrics with different population sizes

λ	MBF	MSD	FSR	DSR	AES
5	1.00e-3	3.82e0	1.0	0.8	5.29e+4
13	2.18e-4	4.00e-1	1.0	1.0	5.17e+4
50	2.11e-4	3.81e-1	1.0	1.0	8.98e+4
100	2.19e-2	1.71e+3	0.0	0.0	1.68e+5
(13, 50)	1.30e-2	3.16e+2	0.4	0.4	7.19e+4
(13, 25, 50)	2.11e-4	3.85e-1	1.0	1.0	7.34e+4

Table 3.13: Identification metrics with a combination of best hyperparameters

$\overrightarrow{x}_0$	σ_0	C_x	$\sigma_w\,[m/s]$	$T\,[s]$	$f\,[Hz]$	λ	MBF	MSD	FSR	DSR	AES
$\overrightarrow{x}_{0_{ref}}$	0.2	-	0	20	60	13	2.18e-4	4.00e-1	1.0	1.0	5.17e+4
$\overrightarrow{x}_{0_{ref}}$	0.01	-	0	10	400	50	2.60e-4	8.87e-1	1.0	1.0	5.02e+4

The default population size is chosen to be $\lambda = 4 + \lfloor 3 \cdot \log(N) \rfloor$, where N is the number of dimensions of the search landscape (Hansen, 2016b). With the 26 aerodynamic derivatives, this gives a default initial population size of $\lambda = 4 + \lfloor 3 \cdot \log(26) \rfloor = 13$ individuals. The default population size given by the CMA-ES algorithm is relatively small to allow for fast convergence (Hansen, 2016b). Various population sizes have been individually tested, parting from the default hyperparameter settings. Results are shown in table 3.12.

The best algorithm performance has been achieved with a population size of 50. Although recommended by the CMA-ES authors (Hansen, 2016a), restarts with increasing population size have not shown significantly better performance for this specific problem. This may be due to the fact that the initial point and standard deviation allow for a good convergence to the global minimum with a relatively small population size. In a more complex problem, such as a real aircraft identification problem, restarts with increasing population sizes may lead to better performance.

3.1.1.6 Combining best results

Given that each hyperparameter has been tested by individually modifying its value parting from the default hyperparameter setup (table 3.3), the best results for each experiment have been selected and combined looking for an increased algorithm performance. During selection the main priority has been to achieve the best fitness and distance metrics, and the best success rates. Reducing the number of evaluations is a secondary priority.

Surprisingly, the combination of the optimal states of each individual hyperparameter does not achieve better performance than the default hyperparameters. This may be due to the system complexity,

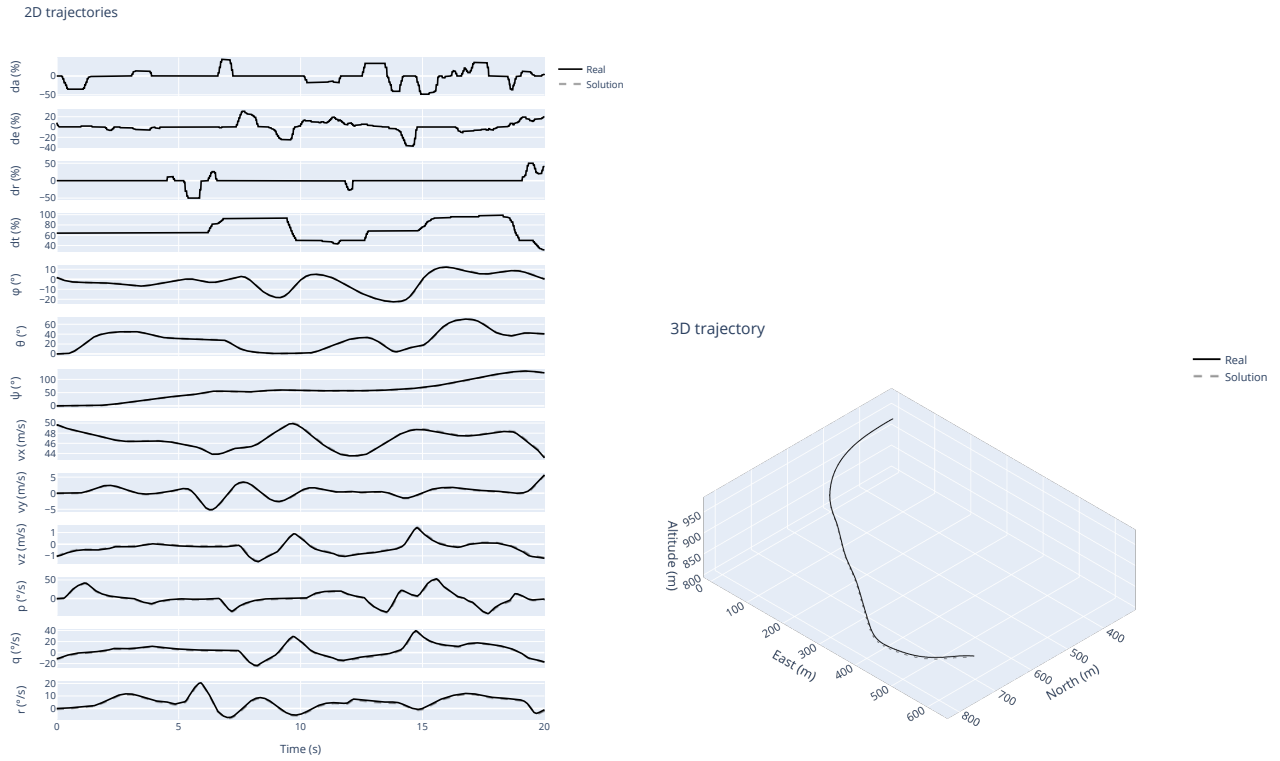

Figure 3.2: Real and solution state trajectories using default configuration hyperparameters. The real trajectory has been generated in the simulator. Real and solution trajectories are overlapping, given that the resulting solution is very close to the global minimum.

hyperparameters may interfere with each other and the overall performance differs from the observed individual effects. Figure 3.2 shows the resulting state trajectories using default hyperparameters.

3.1.2 Identification of a real aircraft

The previous section presented identification results for a simulated flight dataset. That problem may be easier to solve than a real aircraft identification, since the simulated data has been generated using the nonlinear dynamic model used in the optimization framework. This ensures that a global minimum exists within the search landscape which very accurately describes the aircraft motion. However, this may not be the case for parameter identification from real aircraft dynamics, since the nonlinear model used is only an approximation of reality. This means that even the global minimum in the case of a real dataset may not accurately describe the dynamics. In order to solve this problem, the model can be iteratively refined seeking for an accurate representation of reality.

Furthermore, evaluation of real aircraft dynamic identification is more difficult, since the real coefficients of the aircraft in evaluation are not known a priori. This prevents calculation of any kind of metrics which take into account the resulting and real coefficients, such as MSD, FSR or DSR. The only valid metrics in this case are MBF and AES, since resulting fitness and the required number of

Figure 3.3: Small fixed-wing UAV used for framework evaluation.

Table 3.14: Parameters used for framework evaluation with the real aircraft

P_{real}						
$m\ [kg]$	$g\ [m/s^2]$	$\rho\ [kg/m^3]$	$S\ [m^2]$	$T_{max}\ [N]$	$b\ [m]$	$c\ [m]$
20	9.8056	1.225	0.3	100	1	0.3
$I_x\ [kg \cdot m^2]$	$I_y\ [kg \cdot m^2]$	$I_z\ [kg \cdot m^2]$	$I_{xz}\ [kg \cdot m^2]$	$i\ [°]$	$V_w\ [m/s]$	$\theta_w\ [°]$
90	55	110	0	0	0	0
	$\varphi_w\ [°]$	$\sigma_w\ [m/s]$	$\tau_s\ [s]$	$\tau_e\ [s]$	$V_0\ [m/s]$	
	0	0	0	0	23	

evaluations are available. For a more exhaustive evaluation, two approaches can be performed: (1) Comparing the output of the optimized model generated with a different flight dataset of the same real aircraft. (2) Inspecting the optimized model by manually controlling the aircraft via simulation or through meticulous inspection of the model physical consistency. Of these two methods, the former can provide more objective evaluation results.

Dynamic states data has been gathered during a real aircraft flight for evaluation purposes. The platform used consists of a small general purpose fixed-wing UAV with a 1 meter wingspan and a total mass of approximately 20 kg, and is depicted in figure 3.3. The optimization has been performed with the parameters shown on table 3.14. Some of these parameters have been chosen taking into account the aircraft specifications. Recorded real flight dynamic states histories used for optimization are shown in figure 3.4.

Even though the data has been recorded at frequencies smaller than 60 Hz, linear interpolation techniques have been applied for resampling the data to this rate. The developed framework has been used for aerodynamic parameter identification of a model which can reproduce the recorded data. An initial optimization has been performed using the default hyperparameters (table 3.3), and using the default fitness definition of angular and linear velocities fitness contribution terms ($K = 2$). For the case of identification of real aircraft trajectories, the contribution of $f_{penalty}$ to the fitness

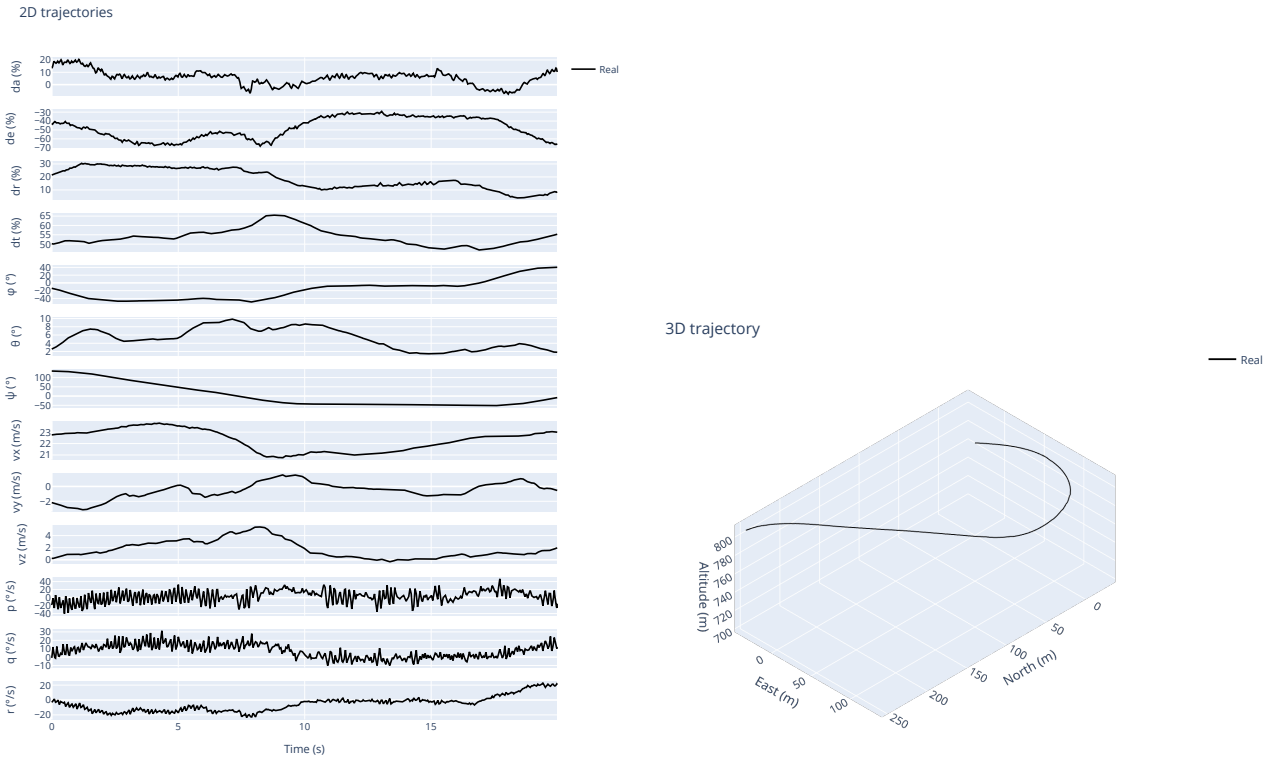

Figure 3.4: Real aircraft flight state trajectories.

function is necessary for maintaining physical consistency of the optimized solution. This prevents solutions with large values in their chromosomes to perpetuate during the optimization, since large aerodynamic coefficients are not realistic. Table 3.15 and figure 3.5 show the resulting identified model, along with optimization metrics MBF and AES.

Results show that even though angular velocities are close to the real trajectories, linear velocities, and consequently position, differ significantly from reference. This translates into a higher resulting minimum fitness value than in parameter identifications based on simulated datasets. In order to try reducing this error in linear velocities, an additional contribution has been added to the fitness function (equation 2.19), the positional distance along trajectories. The resulting identified model with the updated fitness function definition, now using the positional fitness contribution term ($K = 3$), is shown in table 3.16 and figure 3.6, along with resulting optimization metrics MBF and AES.

As expected, not only linear and angular velocities are now similar, but also positional trajectory. Some resulting aerodynamic coefficients are physically coherent, but this is not true for multiple coefficients which result in a value of zero. This outcome could be due to an excessively high contribution of the penalty term in the total fitness, and the effect may be softened by reducing its weight via a scale factor. The model could be iteratively refined from this point in order to obtain a high-fidelity representation of the real dynamics, since achievable identification accuracy and fidelity of the model used are strongly correlated (Liu et al., 2017).

Table 3.15: Resulting coefficients and metrics of default optimization ($K = 2$)

$\overrightarrow{x}_{optim}$								
C_{D_0}	K	C_{D_β}	C_{Y_β}	$C_{Y_{\delta a}}$	$C_{Y_{\delta r}}$	C_{Y_p}	C_{Y_r}	C_{L_0}
0.00	0.19	0.00	0.00	0.00	0.00	0.00	0.00	2.11
C_{L_α}	C_{l_β}	$C_{l_{\delta a}}$	$C_{l_{\delta r}}$	C_{l_p}	C_{l_r}	C_{m_0}	C_{m_α}	$C_{m_{\delta a}}$
5.21	0.00	0.00	0.08	0.00	0.00	0.07	-0.98	0.00
$C_{m_{\delta e}}$	$C_{m_{\delta r}}$	C_{m_q}	C_{n_β}	$C_{n_{\delta a}}$	$C_{n_{\delta r}}$	C_{n_p}	C_{n_r}	
0.00	0.00	0.00	1.22	0.00	0.00	0.00	0.00	

MBF = 8.06e+0, AES = 2.45e+4

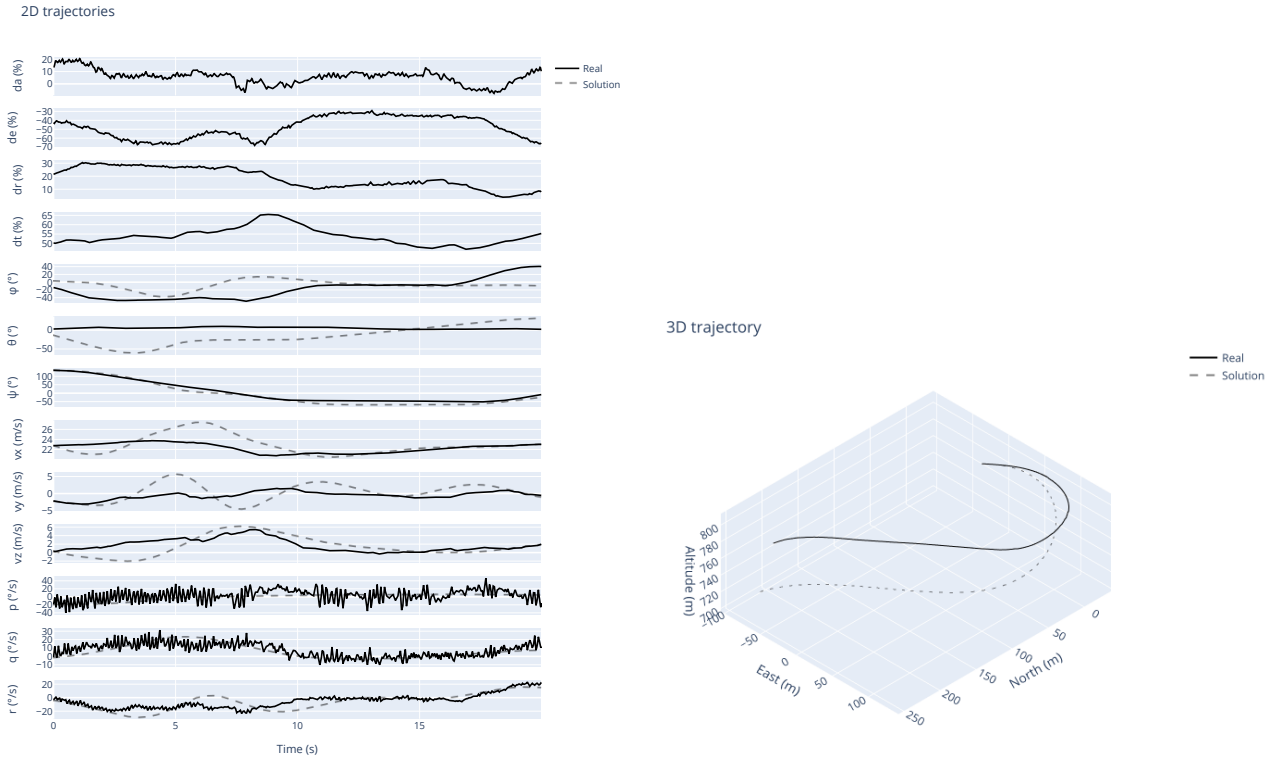

Figure 3.5: Resulting trajectories of default optimization.

Table 3.16: Resulting coefficients and metrics of optimization with updated fitness definition ($K = 3$)

$\overrightarrow{x}_{optim}$								
C_{D_0}	K	C_{D_β}	C_{Y_β}	$C_{Y_{\delta a}}$	$C_{Y_{\delta r}}$	C_{Y_p}	C_{Y_r}	C_{L_0}
0.00	0.10	0.00	0.00	0.00	0.00	0.00	0.00	2.18
C_{L_α}	C_{l_β}	$C_{l_{\delta a}}$	$C_{l_{\delta r}}$	C_{l_p}	C_{l_r}	C_{m_0}	C_{m_α}	$C_{m_{\delta a}}$
9.95	-0.58	0.39	0.00	0.00	0.00	-0.07	-0.92	0.00
$C_{m_{\delta e}}$	$C_{m_{\delta r}}$	C_{m_q}	C_{n_β}	$C_{n_{\delta a}}$	$C_{n_{\delta r}}$	C_{n_p}	C_{n_r}	
0.00	1.59	0.00	1.02	-0.70	0.15	0.00	0.00	

MBF = 1.31e+1, AES = 4.52e+4

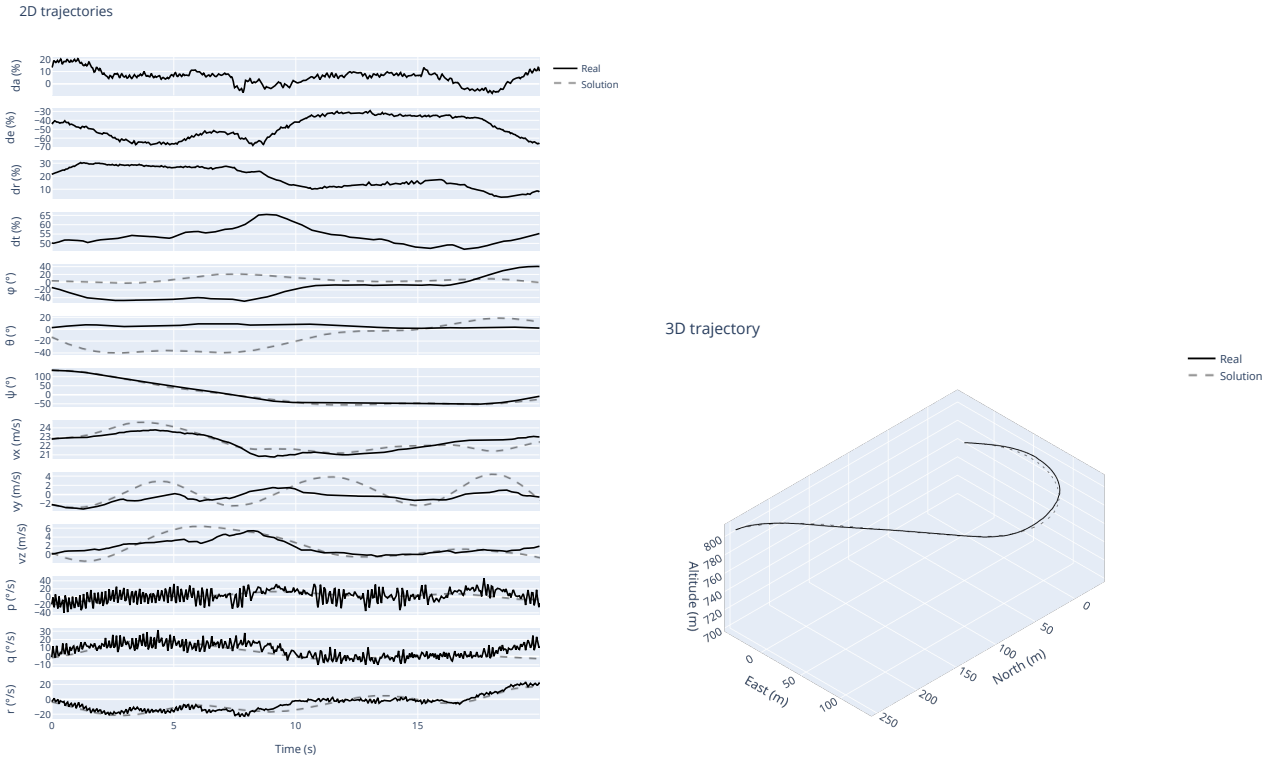

Figure 3.6: Resulting trajectories of optimization with updated fitness definition.

3.1.3 Comparison with similar methods

Although evolutionary algorithms have been extensively applied to the problem of gray-box model identification from time-series data (Cui et al., 2019; Braun et al., 2010, 2011; Rodriguez-Vazquez et al., 2004; Gray et al., 1998; Tan and Li, 2002; Rahimpour et al., 2017; Jizhen et al., 2017), limited research is available on the application of evolutionary algorithms for the specific field of aerodynamic parameter identification from flight data. Local optimization strategies are more common in this field, including least squares model fitting algorithms such as Gauss-Newton, steepest-descent and Levenberg-Marquardt (Rasheed, 2017; Ahsan et al., 2016; Jamil et al., 2015; Rommel et al., 2017; Padayachee, 2016), which have demonstrated satisfactory identification capabilities, provided an approximate initial solution is available.

The evolutionary approach considered in this work has shown promising identification capabilities, as demonstrated with the results of identification from simulated data, where the true solution was found in many algorithm configurations, even when stochastic noise was introduced into the dynamics. However, more research is needed in the area of system identification from real flight data, since results are yet not adequate for practical use in this area, since some local optimization based methods have demonstrated satisfactory results for identification from real data. Some important areas for improvement are the enhancement of model dynamics for a more accurate representation of reality, and the application of other methods for approximating a good initial solution to facilitate algorithm convergence to the global minimum. The framework should also be addressed with a multi-objective optimization approach, which may improve resulting identification capabilities, since multiple criteria are conflicting in the definition of the fitness function. An incremental optimization approach has been performed to try to soften this effect, but results are too dependent on some parameters which need to be manually set, such as the fitness contribution components weights, which makes the process of obtaining an adequate trade-off solution overly tedious.

3.2 Deep learning framework

This section presents the resulting capabilities of the deep learning framework for system identification. The evaluation process has been performed in two parts. First, an objective evaluation is carried out using simulated flight data. Next, the capabilities for identification of real aircraft flight dynamics are tested. Finally, results are compared to similar methods in the field.

3.2.1 Identification from simulated trajectories

This section presents the results and procedure followed for the training and evaluation of a recurrent neural network to solve the problem of aerodynamic parameter identification from simulated flight data. As mentioned in section 2.4.1, flight data generation has been performed by simulation of

Table 3.17: Neural network A architecture.

Neural network A		
Layer	Output shape	# Param
GRU	(None, 16, 256)	274944
Batch norm.	(None, 16, 256)	1024
Flatten	(None, 4096)	0
Dense	(None, 26)	106522
Total params: 382,490 Trainable params: 381,978 Non-trainable params: 512		

Table 3.18: Neural network B architecture.

Neural network B		
Layer	Output shape	# Param
GRU	(None, 16, 1024)	3459072
Batch norm.	(None, 16, 1024)	4096
Flatten	(None, 16384)	0
Dense	(None, 26)	426010
Total params: 3,889,178 Trainable params: 3,887,130 Non-trainable params: 2,048		

Table 3.19: Neural network C architecture.

Neural network C		
Layer	Output shape	# Param
LSTM	(None, 16, 256)	365568
Batch norm.	(None, 16, 256)	1024
Flatten	(None, 4096)	0
Dense	(None, 26)	106522
Total params: 473,114 Trainable params: 472,602 Non-trainable params: 512		

Table 3.20: Neural network D architecture.

Neural network D		
Layer	Output shape	# Param
LSTM	(None, 10, 1024)	4608000
Batch norm.	(None, 10, 1024)	4096
Flatten	(None, 10240)	0
Dense	(None, 26)	266266
Total params: 4,878,362		
Trainable params: 4,876,314		
Non-trainable params: 2,048		

Table 3.21: Neural network E architecture.

Neural network E		
Layer	Output shape	# Param
LSTM	(None, 11, 512)	1255424
Batch norm.	(None, 11, 512)	2048
Flatten	(None, 5632)	0
Dense	(None, 26)	146458
Total params: 1,403,930		
Trainable params: 1,402,906		
Non-trainable params: 1,024		

Table 3.22: Neural network F architecture.

Neural network F		
Layer	Output shape	# Param
LSTM	(None, 10, 512)	1255424
Batch norm.	(None, 10, 512)	2048
Flatten	(None, 5120)	0
Dense	(None, 26)	133146
Total params: 1,390,618		
Trainable params: 1,389,594		
Non-trainable params: 1,024		

Table 3.23: Deep learning settings evaluated and associated results.

NN	Arch.	# Cells	# States	States (*)	# Epochs	Init.	Val. loss
A	GRU	256	16	(C, F, M, V, Ω)	29	Fixed	0.144
B	GRU	1024	16	(C, F, M, V, Ω)	100	Fixed	0.122
C	LSTM	256	16	(C, F, M, V, Ω)	100	Fixed	0.114
D	LSTM	1024	10	(C, V, Ω)	1480	Rand.	0.125
E	LSTM	512	11	(C, Q, P)	335	Rand.	0.234
F	LSTM	512	10	(C, V, Ω)	490	Rand.	0.126

(*) C $\equiv$ Controls, F $\equiv$ Aerodynamic forces, M $\equiv$ Aerodynamic moments,
V $\equiv$ Linear velocities, $\Omega \equiv$ Angular velocities, Q $\equiv$ Quaternions, P $\equiv$ Position

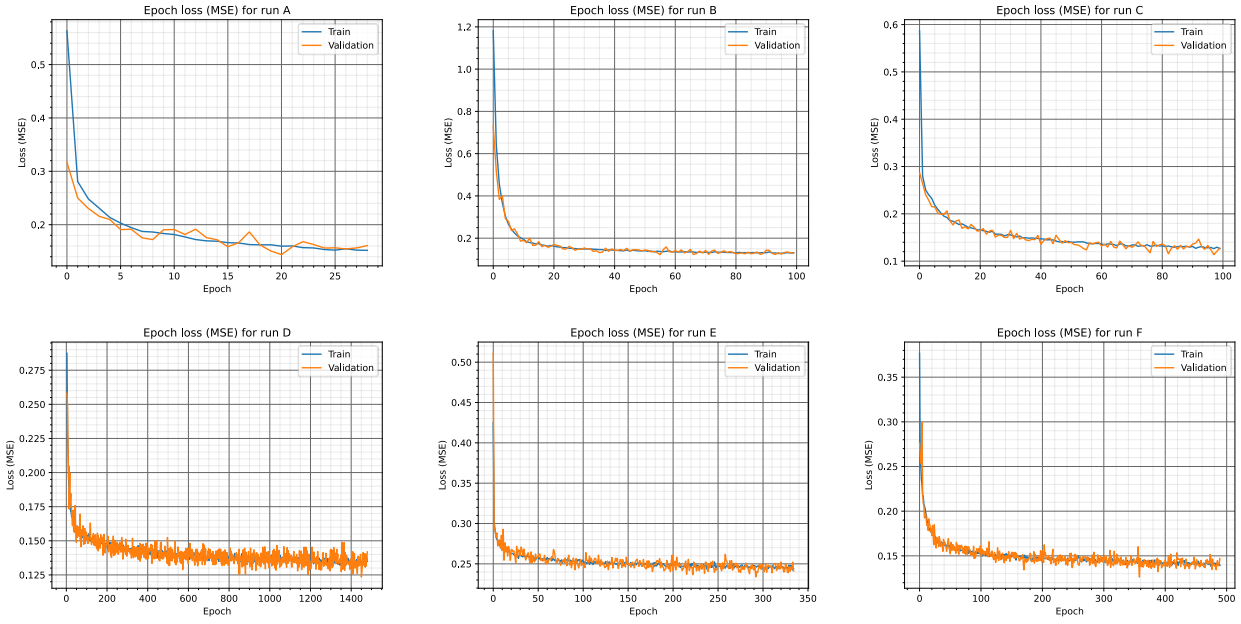

Figure 3.7: Train and validation loss (MSE) in each epoch.

Table 3.24: Prediction of neural network D based on a simulated trajectory of randomly generated aerodynamic coefficients.

MSE = 0.0035									
	C_{D_0}	K	C_{D_β}	C_{Y_β}	$C_{Y_{\delta a}}$	$C_{Y_{\delta r}}$	C_{Y_p}	C_{Y_r}	C_{L_0}
$\overrightarrow{x}_{real}$	0.03	0.01	0.19	-0.51	0.00	0.09	0.00	0.31	0.12
$\overrightarrow{x}_{pred}$	0.06	0.00	0.14	-0.41	0.00	0.19	0.00	0.41	0.12
	C_{L_α}	C_{l_β}	$C_{l_{\delta a}}$	$C_{l_{\delta r}}$	C_{l_p}	C_{l_r}	C_{m_0}	C_{m_α}	$C_{m_{\delta a}}$
$\overrightarrow{x}_{real}$	7.89	0.00	-0.21	0.03	-0.79	0.06	0.00	-1.54	0.00
$\overrightarrow{x}_{pred}$	7.91	0.00	-0.20	0.02	-0.78	0.08	0.00	-1.40	0.00
	$C_{m_{\delta e}}$	$C_{m_{\delta r}}$	C_{m_q}	C_{n_β}	$C_{n_{\delta a}}$	$C_{n_{\delta r}}$	C_{n_p}	C_{n_r}	
$\overrightarrow{x}_{real}$	-1.31	0.00	-8.20	0.19	-0.01	-0.12	-0.02	-0.29	
$\overrightarrow{x}_{pred}$	-1.29	0.00	-8.39	0.23	-0.01	-0.11	-0.03	-0.28	

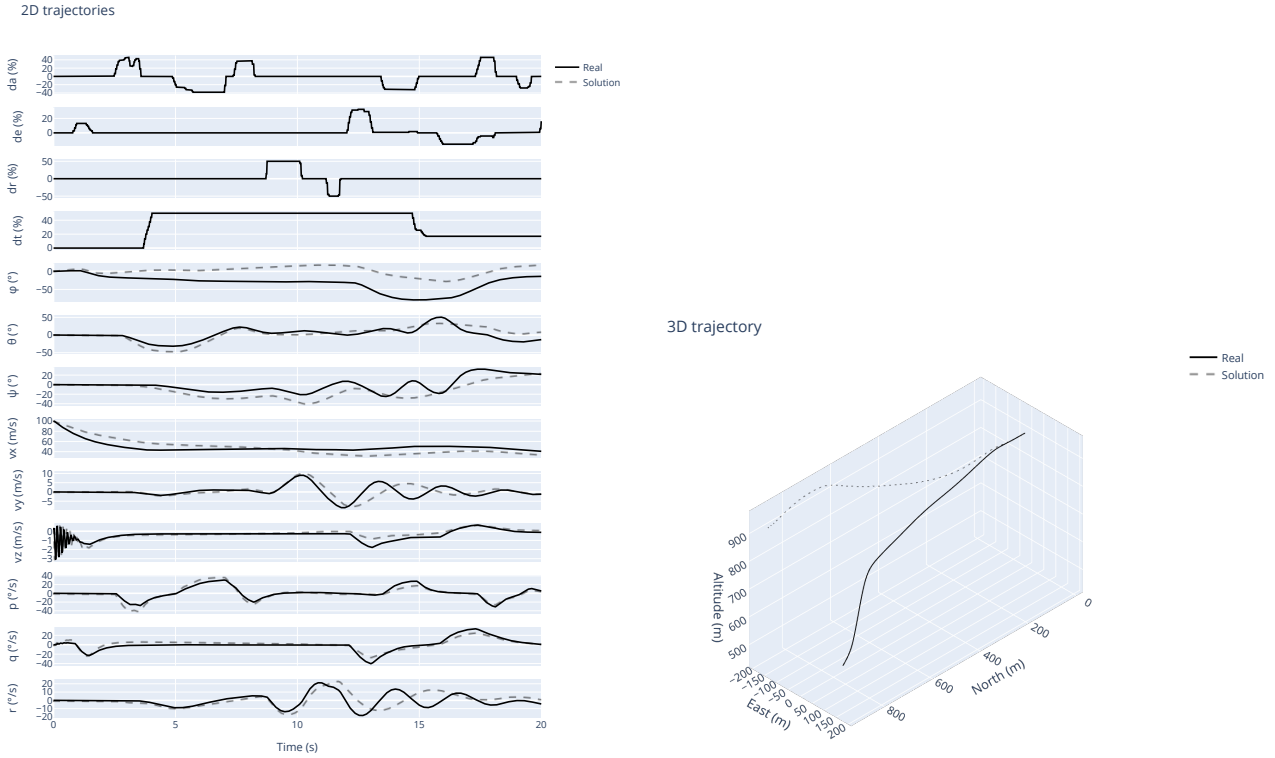

Figure 3.8: Neural network D prediction based on a simulated trajectory of randomly generated aerodynamic coefficients is evaluated them by using a manually generated trajectory. This input control trajectory has never been seen by the neural network. This demonstrates the network capability to generalize.

aerodynamic trajectories. Randomness is introduced in three elements of the framework: (1) Aerodynamic coefficients are randomly chosen for each flight sample, (2) control inputs are randomly updated at each simulation time step during the simulations, and (3) initial dynamic states (and controls) are randomly determined at the start of each simulation. Some combinations of aerodynamic coefficients may lead to unstable dynamics, and eventually the divergence of some dynamic states to infinity, causing invalid solutions and increasing the simulation time required for valid data generation. In order to address this problem, random aerodynamic coefficients are chosen close to the set of the reference dynamics, which is known to be stable and realistic. This assumes aircraft aerodynamic derivatives tend to be similar for similar aircraft geometries. Therefore, new aerodynamic derivatives are sampled according to equation 3.2, where $\mathcal{U}$ represents the continuous uniform distribution.

$$\overrightarrow{x}_{sample} \sim \mathcal{U}(0.5 \cdot \overrightarrow{x}_{ref}, 1.5 \cdot \overrightarrow{x}_{ref}) \tag{3.2}$$

It is important to note that some auxiliary parameters used in simulation, which correspond to inertial, propulsive and geometric properties of the vehicle, as well as simulated wind or actuator model parameters (table 3.2) are always fixed during data generation. This means that the trained neural networks are only valid for those specific conditions, and would need to be retrained if they change (e.g. identification of an aircraft with a different weight or geometry than the one used for simulated data generation during training).

Six neural network architectures have been evaluated using automatically generated data from the dynamics simulator, based on the RNN architectures GRU and LSTM. Training has been performed with batch sizes of 32 labeled samples. Each sample consists on a set of discrete dynamic state and control trajectories, with up to 16 variables with 100 time steps each. Flight data is recorded at 5 Hz in simulation time, therefore the total simulated flight time of each sample is 20 seconds. However, the simulator internally propagates the dynamics at a faster rate of 200 Hz in simulation time[1]. Mean squared error (MSE) has been used as regression loss. *Adam* optimizer has been used for training, a stochastic gradient descent method based on adaptive estimation of first-order and second-order moments, which helps in improving convergence capabilities (Kingma and Ba, 2017). The training and validation data is generated at run-time, so there is no requirement to store it in memory. Each epoch consists on executing the backpropagation algorithm (section 2.4) on 500 input batches. After each epoch, validation is performed on 50 batches, which are created using the same generator function used for training sample generation.

An important consideration for evaluating resulting metrics is to take into account the baseline for considering results a simple linear regression. The networks may learn that the random aerodynamic parameters usually stay within the bounds $\overrightarrow{x}_{ref} \pm 0.5 \cdot \overrightarrow{x}_{ref}$ (equation 3.2). Therefore, a simple

[1]Specified frequencies refer to time steps used within the simulation framework for dynamic propagation, not to real wall-clock time steps.

strategy to minimize loss is to give solutions close to the reference solution. The actual loss associated with always predicting the reference solution can be calculated by measuring its mean MSE loss with respect to multiple randomly generated individuals. The mean MSE loss under these conditions results to be approximately 0.289. This metric can serve as a baseline for evaluation: If a neural network does not provide smaller validation losses, it may be performing similarly to a linear regressor.

Specific architecture details are specified in tables 3.17 to 3.22. Table 3.23 shows a set of different settings of neural network architectures which have been evaluated, along with their resulting validation loss. Note that networks A to C have not been trained with random dynamic states initialization. Interestingly, they seem to achieve lower values of validation loss. This may be due to the fact that the networks trained with random state initialization need to generalize to a greater extent. Additionally, different dynamic states have been provided to different network architectures during training. Aerodynamic forces and moments, although readily available when generating data by means of a simulator, are not directly available during real flight data recording, but they could be estimated from other directly measurable states like linear and angular velocities and accelerations by taking into account the inverse kinematics (section 2.2.1.2). These parameters, along with linear and angular velocities, seem to be good predictors for the aerodynamic coefficients identification, since the networks achieve low validation loss values. Orientation states are not provided in the form of Euler angles, since these do not guarantee to be continuous during flight. Instead, orientation in form of quaternions has been provided, which guarantee continuity along the trajectories. However, position and orientation states seem to not be as good predictors, since the network E has achieved a much higher validation loss, close to the one associated with linear regression on this problem. All architectures achieve lower validation losses than the estimated threshold for linear regression. This indicates that the networks are being able to infer coefficients by having into account the relationship between control and dynamic states trajectories, performing an adequate mapping of the function.

Training and validation loss histories for the different neural network architectures during training are shown in figure 3.7. No signs of overfitting to the training data are observed. This may indicate that the training data is representative of the overall dynamics, and that the networks are generalizing well. Overall, network D is the best performing candidate, as it has achieved the lowest validation value within the set of trained networks using random dynamic states initialization, which allows the usage of this network for inference of any flight data set, regardless of its initial dynamic states. In order to evaluate the prediction capabilities of the trained network, a random set of aerodynamic coefficients has been used to generate a set of trajectories in the simulator, which in turn have been provided to the network as input for inference (table 3.24). Resulting loss for this single sample is small, with an MSE value of 0.0035 (and an MSD value of 0.72). Subsequently, both the initial and predicted sets of aerodynamic coefficients have been used for generating trajectories with a control history that the network has never been trained on before, since it was generated manually via real time commands from a joystick device. This can provide an indication of the network generalization

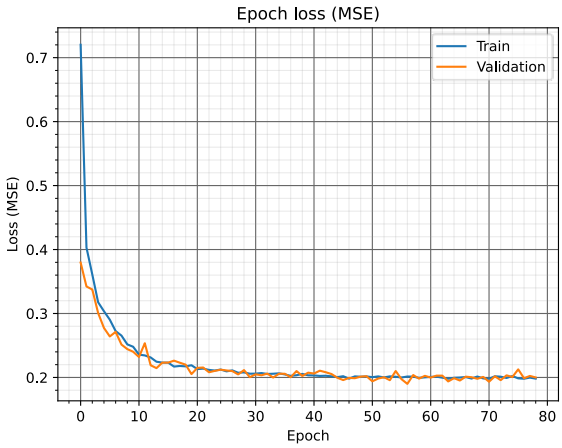

Figure 3.9: Train and validation loss (MSE) in each epoch.

Table 3.25: Neural network predicted coefficients based on real trajectory.

$\overrightarrow{x}_{pred}$								
C_{D_0}	K	C_{D_β}	C_{Y_β}	$C_{Y_{\delta a}}$	$C_{Y_{\delta r}}$	C_{Y_p}	C_{Y_r}	C_{L_0}
0.04	0.02	0.14	-0.39	0.00	0.16	0.00	0.40	0.14
C_{L_α}	C_{l_β}	$C_{l_{\delta a}}$	$C_{l_{\delta r}}$	C_{l_p}	C_{l_r}	C_{m_0}	C_{m_α}	$C_{m_{\delta a}}$
5.85	0.00	-0.39	0.02	-0.77	0.06	0.00	-1.42	0.00
$C_{m_{\delta e}}$	$C_{m_{\delta r}}$	C_{m_q}	C_{n_β}	$C_{n_{\delta a}}$	$C_{n_{\delta r}}$	C_{n_p}	C_{n_r}	
-0.89	0.00	-7.54	0.28	-0.01	-0.10	-0.01	-0.22	

capabilities. Resulting trajectories are shown in figure 3.8. It is important to note that the error in the dynamic states along the trajectories accumulates over time, since any pre-existing error in the dynamic states leads to an increase of the errors for the next sample due to errors during the propagation using the state derivatives.

3.2.2 Identification from real trajectories

The architecture of neural network D has been used for identification of real flight dynamics from real trajectory data. Given that the real aircraft (small fixed-wing UAV, figure 3.3) has different properties than the simulated aircraft used for the original training (medium-sized aerobatic aircraft, figure 3.1), the network needs to be retrained with a different set of fixed auxiliary parameters for dynamics propagation in the simulator, including different inertial, geometric and propulsive characteristics. This is because the real aircraft to be identified has different properties than the reference model used for evaluation.

Resulting train and validation loss histories during training are shown in figure 3.9. The retrained network has been provided with the real aircraft trajectory for inference of the real aerodynamic

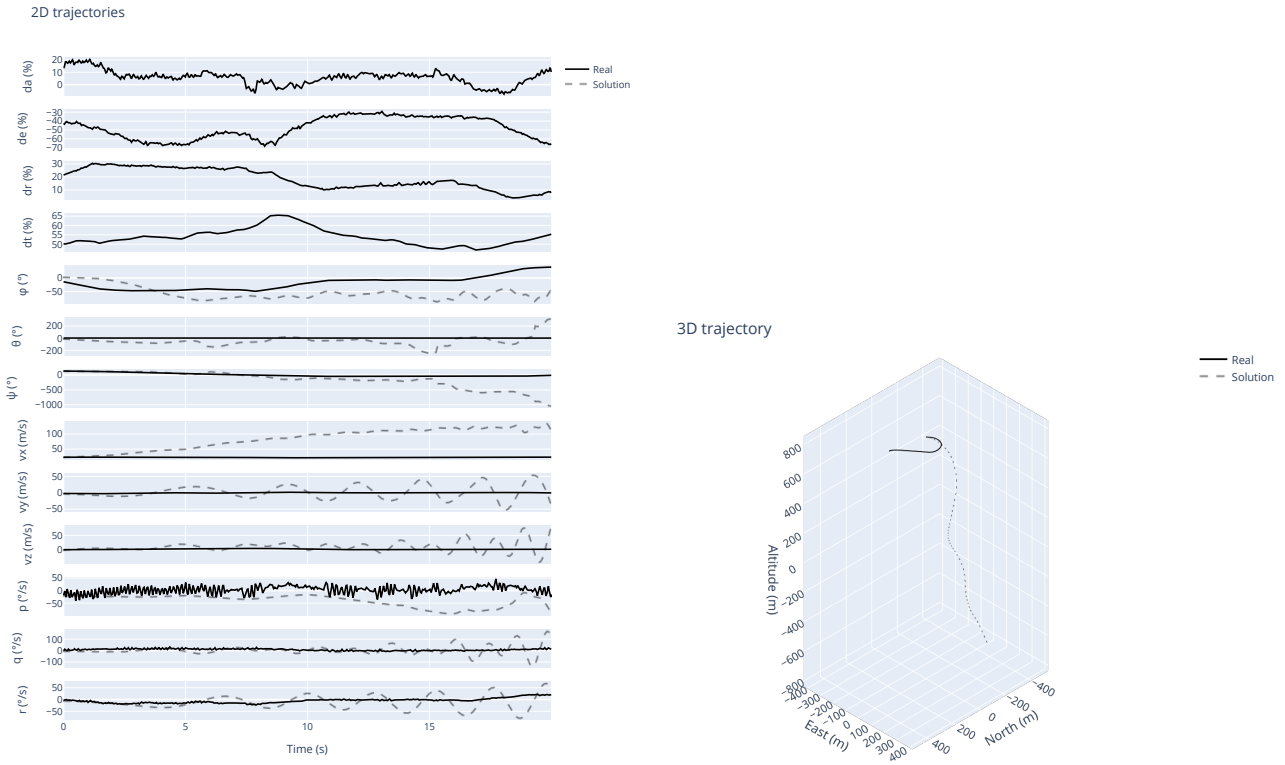

Figure 3.10: Trajectory states predicted by neural network D from identification of real dynamics.

derivatives. Resulting derivatives and associated trajectories are shown in table 3.25 and figure 3.10. Identification results are poor. These results may be indicative of an imprecise choice for fixed auxiliary parameters, causing that the neural network performs well with simulated flight data but not with real data. Results may be improved by iterative refinement of the model and framework settings to most closely resemble the real trajectory during training.

3.2.3 Comparison with similar methods

Application of neural network techniques has experienced a rise in the field of aircraft parameter identification. The results of the framework developed in this work demonstrate the capabilities of RNN to generalize well when using simulated data. This hints at promising identification capabilities, but results are yet not adequate for practical use in this area. Similar implementations demonstrate satisfactory results for identification from real flight data (Wharington et al., 1993; Zheng and Xie, 2018; Bansal et al., 2016; Amiruddin et al., 2020; Punjani and Abbeel, 2015; Kang et al., 2019; Chauhan and Singh, 2017a,b; Majeed and Dongare, 2021; Egorchev M., 2018). Some important areas for improvement are the enhancement of model dynamics for a more accurate representation of reality, and the application of inverse-based automatic control systems for realistic dynamic trajectories generation through simulation.

3.3 Comparison between evolutionary and deep learning frameworks

This work proposes two different approaches for addressing the problem of aerodynamic parameter estimation from flight data. Each of them offers some benefits over the other. Resulting metrics for identification using simulated data are generally better in the evolutionary framework, which achieves an MSD of 0.40 with reference hyperparameters, compared to an MSD of 0.72 when evaluating the best performing trained neural network (architecture D) with simulated data. However, this results are not conclusive, since the performance of both approaches could be increased in future work. In respect to aerodynamic parameter identification from real flight data, the evolutionary framework performs better than the deep learning framework in their current state, although neither of them are able to provide adequate results for practical use in this area, and more research is required for improving performance.

The deep learning framework offers the advantage of requiring a single training procedure, after which the inference of dynamics can be performed very efficiently on each new problem (assuming aircraft properties such as mass or moments of inertia are included as part of the neural network mapping). In turn, the evolutionary framework requires running the entire optimization process for each new identification problem.

In regards to computational requirements, the training process for the deep learning framework requires important amount of computational resources, but inference can be performed in more modest hardware, as is the case with the evolutionary framework.

Chapter 4

Conclusions and future work

This chapter describes the final conclusions of this work, and proposes some areas of improvement which may be addressed in the future.

4.1 Conclusions

This work hypothesized that evolutionary algorithms and deep learning methods can be used to solve the problem of aircraft aerodynamic coefficient identification from flight data. The following statements have been concluded.

- The proposed objectives have been fulfilled. Both system identification frameworks, one based on an evolutionary algorithm and another based on a neural network, have been developed[1], along with an aircraft dynamics simulator based on aerodynamic theory and classical kinematics, which is used internally by both frameworks. The simulator optionally accepts user commands from a joystick device for real-time flight simulation. Additionally, a graphical engine has been integrated for real-time graphics rendering during simulation, and a graphical user interface has been implemented for improving user interaction with the framework.
- Results are not conclusive to prove the initial hypothesis.
- Promising identification capabilities have been demonstrated for identification from simulated flight data from both frameworks.
- More research is needed in the area of system identification from real flight data for both frameworks, since results are yet not adequate for practical use in this area.
- The evolutionary framework achieves slightly better performance than the deep neural network for the case of simulated data.

[1] Code implementation is available at the repository `https://github.com/fidelechevarria/DeepAero`

- The deep learning framework offers the advantage of not requiring to retrain the network for each new problem instance, which increases the method efficiency.
- The nonlinear aircraft model used by both frameworks is exclusively valid for fixed-wing aircraft. However, it could be easily adapted to any other type of vehicle by means of updating the gray box dynamic model used.

4.2 Future work

This section proposes some areas in which the system identification frameworks developed in this work could be further improved.

4.2.1 Evolutionary framework

Some important areas for improvement in the evolutionary framework are the enhancement of model dynamics for a more accurate representation of reality, and the application of other methods for approximating a good initial solution to facilitate algorithm convergence to the global minimum. Additionally, a multi-objective optimization approach may improve resulting identification capabilities, since multiple criteria are conflicting in the definition of the fitness function. This would facilitate the process of selecting an adequate trade-off solution. Of particular importance is the evaluation of decoupling the nonlinear equations into the longitudinal and lateral-directional dynamics to reduce the complete problem complexity.

Some hyperparameters could be further evaluated seeking for performance improvements, such as the parameter encoding, weights for individual fitness contributions or order of integration during simulation. Furthermore, hyperparameters could be optimized using a meta-optimization approach.

CMA-ES algorithm is parallelizable, so leveraging the parallel computation capabilities of commonly available GPUs could decrease the computation time required for model identification. Some GPU compatible implementations of evolutionary algorithms include Strock and Pirnay (2019), which implements the CMA-ES algorithm using the machine learning software library TensorFlow, or the evolutionary optimizer included in TensorFlow Probability library[2], which uses the same library for implementation of another evolutionary algorithm known as differential evolution (DE).

Evolutionary algorithms could also be hybridized with local search methods to improve the effectiveness of the search. Algorithms leveraging this principle are known as memetic algorithms (MA), and they have shown to behave better than standard evolutionary algorithms in extremely high-dimensional problems (Lastra et al., 2015).

[2]Differential evolution optimizer in TensorFlow Probability library: `https://www.tensorflow.org/probability/api_docs/python/tfp/optimizer/differential_evolution_minimize`

4.2.2 Deep learning framework

Areas of improvement in the deep learning framework include the incorporation of an inverse-based automatic control system for realistic dynamic trajectories generation or the increase of the aerodynamic parameters bounds for generalization to more aerodynamic classes. Additionally, some model fixed parameters, such as inertial properties of the vehicle, could be incorporated into the neural network mapping process. Finally, more recent neural network architectures can be evaluated, such as temporal convolutional networks (TCN).

Bibliography

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning*. Springer.

Aggarwal, C. C., Hinneburg, A., and Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. In Van den Bussche, J. and Vianu, V., editors, *Database Theory — ICDT 2001*, pages 420–434, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ahsan, J., Ahsan, M., Jamil, A., and Ali, A. (2016). Grey box modeling of lateral-directional dynamics of a uav through system identification. In *2016 International Conference on Frontiers of Information Technology (FIT)*, pages 324–329.

Amiruddin, B. P., Iskandar, E., Fatoni, A., and Santoso, A. (2020). Deep learning based system identification of quadcopter unmanned aerial vehicle. In *2020 3rd International Conference on Information and Communications Technology (ICOIACT)*, pages 165–169.

Andersson, C., Ribeiro, A. H., Tiels, K., Wahlström, N., and Schön, T. B. (2019). Deep convolutional networks in system identification. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 3670–3676.

Bansal, S., Akametalu, A. K., Jiang, F. J., Laine, F., and Tomlin, C. J. (2016). Learning quadrotor dynamics using neural network for flight control. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 4653–4660.

Branke, J., Deb, K., Miettinen, K., and Slowiński, R. (2008). *Multiobjective Optimization*. Springer-Verlag Berlin Heidelberg.

Braun, J., Krettek, J., Hoffmann, F., and Bertram, T. (2010). Multiobjective evolutionary structural optimization for system identification and controller design. In *2010 4th International Workshop on Genetic and Evolutionary Fuzzy Systems (GEFS)*, pages 9–14.

Braun, J., Krettek, J., Hoffmann, F., and Bertram, T. (2011). Structure and parameter identification of nonlinear systems with an evolution strategy. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 2444–2451.

Brown, R. G. (1963). *Smoothing, Forecasting and Prediction of Discrete Time Series*. Prentice-Hall, Englewood Cliffs, N.J.

Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932-3937.

Center, A. R. (2006). Parameter identification. `https://www.nasa.gov/centers/ames/research/technology-onepagers/paramid.html`. Accessed: 21-08-2021.

Chauhan, R. K. and Singh, S. (2017a). Application of neural networks based method for estimation of aerodynamic derivatives. In *2017 7th International Conference on Cloud Computing, Data Science Engineering - Confluence*, pages 58–64.

Chauhan, R. K. and Singh, S. (2017b). Review of aerodynamic parameter estimation techniques. In *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*, pages 864–869.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling.

Cook, M. V. (2013). *Flight Dynamics Principles*. Butterworth-Heinemann, 3rd edition.

Cui, N., Shao, H., Huang, R., and Han, Y. (2019). Study on aerodynamic parameter estimation method based on wavelet neural network and modified pso algorithm. *IOP Conference Series: Materials Science and Engineering*, 563:052050.

Doucet, A., De Freitas, N., and Gordon, N. (2001). *Sequential Monte Carlo methods in practice*. Springer.

Dreyer, T. and Jensen, R. (2014). Free and open-source flight dynamic model of Zivko Edge 540 for JSBSim. `https://wiki.flightgear.org/Zivko_Edge_540`. Accessed: 28-08-2021.

Dubois, G. (2018). *Modeling and Simulation: Challenges and Best Practices for Industry*. CRC Press, 1st edition.

Egorchev M., T. Y. (2018). Neural network semi-empirical modeling of the longitudinal motion for maneuverable aircraft and identification of its aerodynamic characteristics. *Advances in Neural Computation, Machine Learning, and Cognitive Research.*, 736(15).

Eiben, A. E. and Smith, J. E. (2015). *Introduction to Evolutionary Computing*. Springer Publishing Company, Incorporated, 2nd edition.

Ferziger, J. H., Perić, M., and Street, R. L. (2020). *Computational Methods for Fluid Dynamics*. Springer, Berlin, 4th edition.

Gómez Tierno, M. A., Pérez Cortés, M., and Puentes Márquez, C. (2012). *Mecánica del vuelo [Flight mechanics]*. Garceta, 2nd edition. Original book in Spanish.

Gray, G. J., Murray-Smith, D. J., Li, Y., Sharman, K. C., and Weinbrenner, T. (1998). Nonlinear model structure identification using genetic programming. *Control Engineering Practice*, 6(11):1341–1352.

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2nd edition.

Hansen, N. (2016a). The CMA evolution strategy. `http://cma.gforge.inria.fr/`. Accessed: 21-08-2021.

Hansen, N. (2016b). The CMA evolution strategy: a tutorial.

Hansen, N. and Ostermeier, A. (1996). Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 312–317.

Hansen, N. and Ostermeier, A. (1997). Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The (μ/μI, λ)-CMA-ES.

Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

Hongfeng, T., Yuan, X., and Zixin, Y. (2019). Application of differential evolution algorithm in equivalent system fitting. In *Proceedings of the 2019 8th International Conference on Computing and Pattern Recognition*, ICCPR '19, page 488–493, New York, NY, USA. Association for Computing Machinery.

Jamil, M. A., Ahsan, M., Ahsan, M. J., and Choudhry, M. A. (2015). Time domain system identification of longitudinal dynamics of a uav: A grey box approach. In *2015 International Conference on Emerging Technologies (ICET)*, pages 1–6.

Jizhen, L., Junlin, G., Yang, H., Juan, W., and Hong, L. (2017). Dynamic modeling of wind turbine generation system based on grey-box identification with genetic algorithm. In *2017 36th Chinese Control Conference (CCC)*, pages 2038–2042.

Kang, Y., Chen, S., Wang, X., and Cao, Y. (2019). Deep convolutional identifier for dynamic modeling and adaptive control of unmanned helicopter. *IEEE Transactions on Neural Networks and Learning Systems*, 30(2):524–538.

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

Klein, V. and Morelli, E. A. (2006). *Aircraft System Identification: Theory and Practice*. AIAA education series. American Institute of Aeronautics and Astronautics.

Kuethe, A. M. and Schetzer, J. D. (1984). *Foundations of Aerodynamics*. Wiley.

Lastra, M., Molina, D., and Benítez, J. (2015). A high performance memetic algorithm for extremely highdimensional problems. *Information Sciences*, 293:35-58.

Li, L., Liu, Z., and Wen, N. (2016). Dynamic modeling for a variable-span and variable-sweep unmanned aerial vehicle. In *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, pages 1305–1310.

Liberti, L. (2008). Introduction to global optimization. `http://www.lix.polytechnique.fr/~liberti/teaching/dix/inf572-09/nonconvex_optimization.pdf`. Accessed: 21-08-2021.

Liu, J., Li, S., Song, X., and Wang, C. (2017). Influence of linear and nonlinear aerodynamic models on parameter identification for aircraft. In *2017 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pages 227–232.

Majeed, M. and Dongare, V. (2021). *Aircraft Aerodynamic Parameter Estimation from Flight Data Using Neural Partial Differentiation*. Springer, 1st edition.

Morelli, E., Derry, S., and Smith, M. (2012). *Aerodynamic Parameter Estimation for the X-43A (Hyper-X) from Flight Data*.

Moukallet, F., Mangani, L., and Darwish, M. (2015). *The Finite Volume Method in Computational Fluid Dynamics*. Springer.

Padayachee, K. (2016). Aerodynamic parameter identification for an unmanned aerial vehicle.

Petridis, V. and Kehagias, A. (2001). Recurrent neural nets for parameter estimation.

Pope, A. (1954). *Wind Tunnel Testing*. Springer, New York.

Punjani, A. and Abbeel, P. (2015). Deep learning helicopter dynamics models. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3223–3230.

Rahimpour, E., Rashtchi, V., and Aghmasheh, R. (2017). Parameters estimation of transformers gray box model. In *2017 International Conference on Modern Electrical and Energy Systems (MEES)*, pages 372–375.

Raol, J. and Jategaonkar, R. (1995). *Aircraft parameter estimation using recurrent neural networks - A critical appraisal*.

Rasheed, A. (2017). Grey box identification approach for longitudinal and lateral dynamics of uav. In *2017 International Conference on Open Source Systems Technologies (ICOSST)*, pages 10–14.

Rauf, A., Zafar, M. A., Ashraf, Z., and Akhtar, H. (2011). Aerodynamic modeling and state-space model extraction of a uav using datcom and simulink. In *2011 3rd International Conference on Computer Research and Development*, volume 4, pages 88–92.

Rodriguez-Vazquez, K., Fonseca, C., and Fleming, P. (2004). 'identifying the structure of nonlinear dynamic systems using multiobjective genetic programming. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 34(4):531–545.

Rommel, C., Martinon, P., Bonnans, F., and Gregorutti, B. (2017). Aircraft dynamics identification for optimal control.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). *Learning Internal Representations by Error Propagation*, page 318–362. MIT Press, Cambridge, MA, USA.

Simon, D. (2006). *Optimal State Estimation*. John Wiley & Sons, Ltd.

Skansi, S. (2018). *Introduction to Deep Learning*. Springer.

Smirnov, D. and Mephu Nguifo, E. (2018). Time series classification with recurrent neural networks.

Söderström, T. and Stoica, P. (1989). *System identification*. Prentice-Hall.

Stevens, B. L., Lewis, F. L., and Johnson, E. N. (2003). *Aircraft Control and Simulation*. Wiley.

Stinton, D. (1996). *Flying Qualities and Flight Testing of the Airplane*. AIAA Educational Series, Washington D.C.

Strock, R. and Pirnay, N. (2019). CMA-ES implementation using TensorFlow library. `https://github.com/srom/cma-es`. Accessed: 27-08-2021.

Tan, K. and Li, Y. (2002). Grey-box model identification via evolutionary computing. *Control Engineering Practice*, 10(7):673–684. Developments in High Precision Servo Systems.

Tischler, M. B. and Temple, R. K. (2006). *Aircraft and Rotorcraft System Identification: Engineering Methods with Flight Test Examples*. AIAA.

Wharington, J. M., Blythe, P. W., and Herszberg, I. (1993). The development of neural network techniques for the system identification of aircraft dynamics.

Zheng, Y. and Xie, H. (2018). Review on neural network identification for maneuvering uavs. In *2018 International Conference on Sensing,Diagnostics, Prognostics, and Control (SDPC)*, pages 339–346.