TÉCNICO LISBOA

Nonlinear Control Applied to 4D Transport Aircraft Guidance

Guilherme Goulão de Sousa

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisor(s): Prof. Full Name 1
Dr. Full Name 2

Examination Committee

Chairperson: Prof. Full Name
Supervisor: Prof. Full Name 1 (or 2)
Member of the Committee: Prof. Full Name 3

Month Year

Acknowledgments

A few words about the university, financial support, research advisor, dissertation readers, faculty or other professors, lab mates, other friends and family...

Resumo

Seguindo a mudança de paradigma que acontece atualmente da industria do tráfego aéro, esta tese de mestrado propõe um controlador para aviões comerciais permitindo o seguimento de trajectórias 4D, aumentando assim a segurança e capacidade num dado espaço aéro. A lei de controlo utilizada é baseada no método de linearização por feedback, e uma rede neuronal é também implementada de modo a reduzir os erros causados por incertezas na modelização do sistema, causados por erros de estimação dos parâmetros do avião, perturbações externas ou até falhas de systema. A rede neuronal é treinada online, utilizando o algoritmo de retro-propagação, de modo a otimizar a inversão não-linear, resultando assim num controlador adaptivo. Os resultados de simulações mostram que este controlador é capaz de manter o sistema estável e controlável em condições em que normalmente tal não é possível.

Palavras-chave: controlo não linear, linearização por feedback, rede neural, controlo de voo, back-propagation

Abstract

Following the current ATM industry paradigm shift to allow for commercial aircraft to follow 4D trajectory, in order to increase both safety and air capacity, a controller was designed and implemented in this work to reach such a goal in cruise conditions. The control law used is based on feedback linearisation, and a neural network is also implemented in order to restrain errors caused by modelling uncertainties due to poor estimation of the aircraft parameters, external disturbances or fault systems. The neural network is trained online using the back-propagation algorithm to optimise the nonlinear inversion, resulting in an overall adaptive model-based controller. Simulation results show this controller is able to maintain stability and controllability in conditions that would otherwise render the aircraft unstable.

Keywords: non-linear control, feedback linearisation, neural network, flight control, back-propagation

Contents

List of Tables

List of Figures

Nomenclature

BP	Backpropagation
DI	Dynamic Inversion
FBL	Feedback Linearisation
NN	Neural Network

Greek symbols

α	Angle of attack
β	Angle of side-slip
Ω	Angular velocity vector
ϕ, θ, ψ	Pitch, Roll and Yaw Euler angles
ρ	Density

Roman symbols

p, q, r	Angular Velocity Cartesian components
V_a	Airspeed
C_D	Coefficient of drag
C_L	Coefficient of lift
p	Pressure
T	Thrust
$\mathbf{u}$	Velocity vector
u, v, w	Velocity Cartesian components
W	Weight

Chapter 1

Introduction

Due to their inherent non-linear dynamics, designing control systems for both rotary and fixed wing aircraft is a non-trivial task, where using linear control approaches reveal to be inaccurate. Feedback linearisation is an approach to non-linear control design that algebraically transforms a non-linear system into a linear one. For the linearised system linear control techniques can be used. Note that in this work other equivalent terms to feedback linearisation will be used, namely nonlinear inversion (NLI) and dynamic inversion. However this approach, as well as other state-of-the art control approaches such as model predictive control, backstepping and gain scheduling require the *a priori* exact mathematical model of the system to be controlled [1]. The lack of such an exact model leads to errors in the calculation of the model inversion, necessary to implement feedback linearisation. Indeed, while feedback linearisation control shows good tracking, it is sensitive to parameter uncertainties, that ultimately lead to these inversion errors [2], and in extreme cases to instability. One solution to this problem that has gained momentum over the last years is the use augmentation algorithms, namely neural networks and other intelligent algorithms, to minimize and cancel the resulting guidance error[3], [4], [5].

This paper aims to discuss the existing alternatives to augment the precision, robustness and overall performance of feedback linearisation control. A second goal of this work will be to extend the solutions used to increase the robustness of feedback linearisation control to provide an adaptive control in cases of system failures and strong external disturbances.

1.1 Motivation

As the aeronautic industry grows, so is bound to also grow the air traffic dramatically. To answer this problematic, ATM researchers have proposed over the last few years Trajectory-Based Operations (TBO), a concept allowing the use of 4D trajectories to manage both safety and air capacity. In both the US and Europe, initiatives to put such systems in place are currently being developed and implemented, namely the NextGen by the FAA and SESAR EUROCONTROL. Therefore, in order to adopt this air traffic management paradigm, automation will play a crucial role in 4D guidance control, allowing an aircraft to follow flight plans more accurately.

1.2 Topic Overview

In order to fully automatize a commercial aircraft to follow a 4D trajectory in cruise conditions, this work will focus on designing and implementing an autopilot capable of controlling the aircraft attitude, improving flight quality and stability in hazardous piloting situations, to be integrated in a Fly-by-Wire system. The ultimate aim of this project will be to focus on auto pilot to provide 4D trajectory guidance to a commercial aircraft. To do so a model based controller is used, unlike in the currently implemented framework of robust control composed of several PID layers. This model based controller distinguishes fast and slow dynamics, using a nonlinear inversion of the fast dynamics to determine the necessary deflections of the control surfaces. Unlike PIDs and controllers that use linearisation through the Jacobian using Taylor series expansions of the nonlinear system, that may yield unsatisfactory performance and robustness at a larger range of operating points, a nonlinear inversion where the exact original model is known will also yield a linear model that is the exact representation of the nonlinear aircraft model over a wide range of operating points.

This method, however, also has some limitations, the main one being that the feedback linearisation requires an exact knowledge of the system model, to obtain an exact inversion of the system. This is not usually feasible, and errors in the model of the airplane will inevitably lead to inversion errors, especially in cases of heavy external disturbances. A solution for this limitation will be proposed, studied and implemented in this work. By using an online neural network, these inversion errors will be corrected, leading to an improved performance and robustness. It will also add an adaptive controller to the aircraft, as it will be able to react and adapt to different types of disturbances.

1.3 Objectives

This work can be divided in four progressive goals, that must be sequentially achieved. Firstly, the model of a commercial aircraft will be studied thoroughly, defining both its fast and slow dynamics, as well as methods to obtain its aerodynamic coefficients. Once a model is established, it will be then implemented in a simulation environment, in this case Matlab/Simulink. The first goal will be achieved once the simulation of a commercial aircraft is validated and its results are theoretically coherent. A controller using feedback linearisation will then be designed and tested to control the aircraft plant. The desired objective in this case is to obtain a stable reference following with reduced error.

The third objective of this work will indeed be built on top of the previous two. After studying the different types of errors and external perturbations that the linearised model of the aircraft may be submitted to, its behaviour in such conditions will be discussed. A decrease of the accuracy of the controller and stability are expected of the airplane in these conditions. These tests will serve not only to verify the impact of a reduced knowledge of the exact model of the plane, but also to test the robustness of the aircraft to external perturbations such as wind and icing conditions. For the fourth and last objective an online neural network will then be used to improve the robustness and stability of the aircraft in cases in which the default NLI controller performed poorly or even became unstable. The overall final objective will be to have an adaptive controller for a commercial aircraft based on feedback linearisation, without

the limitations of model based controllers, capable of following 4D trajectories and displaying increased robustness to external disturbances.

1.4 Thesis Outline

In chapter 2, a theoretical overview of the work will be given. This chapter will introduce the reader to the different tools used in this work, the two most relevant ones being the feedback linearisation method and neural networks. A description of the feedback linearisation technique and theory will then be provided, as well as its limitations. Finally the theory behind the online neural network will be studied and how it can reduce the inversion errors of a feedback linearisation controller. The following chapter 3 will focus on the implementation of the concepts described previously. Starting with the implementation of the plane model, this section will describe its fast, translation and actuator dynamics, and precise some details in developing a simulation for a commercial aircraft, stating some of its aerodynamic parameters and methods used to obtain them. The controller architecture will then be described, including the inversion algorithm, linear controller and guidance control law. Finally, from both the plane model and its controller comes the neural network architecture used, along with its training algorithm and how it is included to the baseline controller. In chapter 4 the nonlinear inversion controller is validated and tested in various environments, including disturbances that degrade its performance and robustness. In the same conditions the designed online neural network is included to compare its performance to that of a controller without neural network corrections, finally concluding this study in chapter 5.

Chapter 2

Background on Feedback Linearisation and Neural Networks

In this chapter the tools used to achieve the goals of this project will be described firstly. A theoretical model of the plane dynamics will be discussed, and implementation details will be provided in chapter 3. The control strategy used will then follow, giving an overview of the feedback linearisation approach, as well as its limitations, namely its sensitivity to inversion errors and external interferences. An attempt to solve these limitation will be made, suggesting some solutions and finally describing the chosen methodology for this case.

2.1 Feedback Linearisation

Feedback linearisation, also known as dynamic inversion, is an approach based on the idea of algebraically transforming a non-linear system into a linear one, from which linear control laws can be used to control the resulting system. Unlike Jacobian linearisation, that assumes linearity of the system around an equilibrium value, feedback linearisation implements a feedback loop that cancels non-linearities of the given system [6]. Given a single-input u nonlinear system

$$\dot{x} = f(x, u) \tag{2.1}$$

one must find both a state transformation $z = z(x)$ and an input transformation $\upsilon = \upsilon(x, u)$ in order for the transformed system to be a linear and time-invariant system $\dot{z} = Az + B\upsilon$. This process is called Input-Output linearisation.

To better understand Input-Output linearisation, some mathematical tools must firstly be described. One of these tools is the Lie derivative, that represents the gradient of a given scalar function $h(x) : R^n \rightarrow R$ projected along a given vector function $f(x) : R^n \rightarrow R$. Such a Lie derivative is therefore represented as

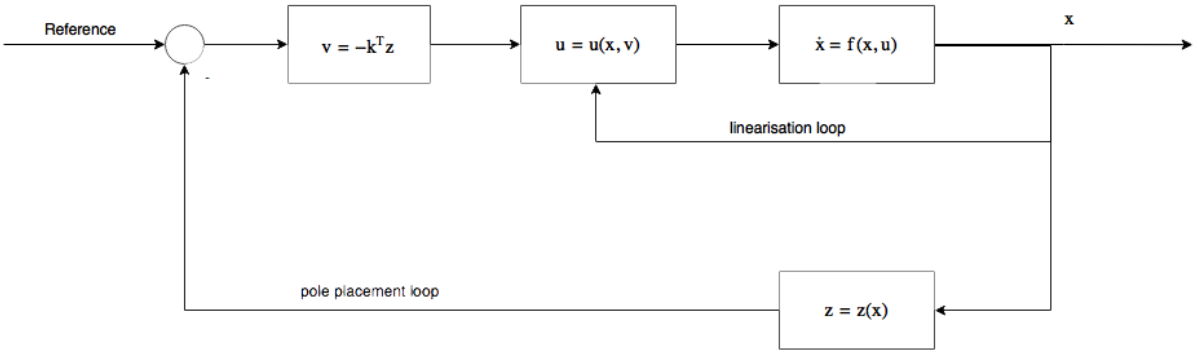

Figure 2.1: Feedback linearisation example [6]

$$L_f h(x) = \nabla h(x) f(x) = \Sigma_{i=1}^{n} \frac{\partial h(x)}{\partial x_i} f_i(x) \tag{2.2}$$

A Lie derivative can also be applied multiple times, resulting in a k^{th} order Lie derivative

$$L_f^k h(x) = L_f \left(L_f^{k-1} h(x) \right) = \nabla \left(L_f^{k-1} h(x) \right) f(x) \quad \text{with} \quad L_f^0 h(x) = h(x) \tag{2.3}$$

The second operator from Lie algebra to be introduced here, quite important in Input-Output linearisation is the Lie bracket defined as, given two vector fields $f(x) : R^n \to R$ and $g(x) : R^n \to R$,

$$[f, g] = ad_f g = \nabla g f - \nabla f g \tag{2.4}$$

Once again Lie brackets can be defined recursively as

$$ad_f^i g = [f, ad_f^{i-1} g] \quad \text{with} \quad ad_f^0 g = g \tag{2.5}$$

Finally, the concept of *diffeomorphism* must be described to fully understand this control technique. According to [6], it is a generalisation of the concept of coordinate transformation, and is formally stated as:

A function $\phi : R^n \to R^n$, defined in a region Ω, is called a diffeomorphism if it is smooth and its inverse ϕ^{-1} exists and is smooth [6]. A diffeomorphism can be used to transform a nonlinear system into another in terms of a new set of states. Let us consider the following dynamic system

$$\dot{x} = f(x) + g(x)u$$
$$y = h(x) \tag{2.6}$$

and a new set of states defined as

$$z = \Phi(x) \tag{2.7}$$

Differentiating z, and using $x = \Phi^{-1}(z)$ comes that

$$\dot{z} = \frac{\partial \Phi}{\partial x} \dot{x} = \frac{\partial \Phi}{\partial x} (f(x) + g(x)u) = f*(z) + g*(z)u \tag{2.8}$$

In Input-Output linearisation, a *diffeomorphism* must be found to transform such that the system from equation 2.8 is a linear one.

2.1.1 SISO Systems

Although an aircraft is always considered as a MIMO system, a description of this control concept will first be provided for a general SISO case, before generalising to a MIMO case. Nonlinear systems that can be represented as affine systems $\dot{x} = f(x) + g(x)u$ will be discussed in this section. From [6], a definition of Input-State linearisation is to find, for a nonlinear system of relative degree n, a *diffeomorphism* $\phi : \Omega \to R^n$ and nonlinear control law

$$u = \alpha(x) + \beta(x)\upsilon \tag{2.9}$$

such that the new state variables $z = \phi(x)$ and *pseudo-input* υ satisfy the linear time invariant system $\dot{z} = Az + B\upsilon$, where the following equations are satisfied

$$\begin{cases} \dot{z_i} = z_{i+1} & if \quad i < n \\ \dot{z_n} = \upsilon & if \quad i = n \end{cases} \tag{2.10}$$

In order to find such a function and control law, one must find a state z_1 such that

$$\nabla z_1 ad_f^i g = 0 \qquad i = 0, ..., n-2 \tag{2.11a}$$

$$\nabla z_1 ad_f^{n-1} g \neq 0 \tag{2.11b}$$

The remaining states are then obtained from $z(x) = [z_1 \quad L_f z_1 \quad ... \quad L_f^{n-1} z_1]^T$ and the input transformation from

$$\alpha(x) = -\frac{L_f^n z_1}{L_g L_f^{n-1} z_1} \tag{2.12a}$$

$$\beta(x) = \frac{1}{L_g L_f^{n-1} z_1} \tag{2.12b}$$

2.1.2 MIMO Systems

These concepts can be extended to MIMO systems in a similar manner, by differentiating the outputs until the inputs explicitly appear. This time however, there are individual relative degrees per output. The sum of these relative degrees is called total degree r and must satisfy $r \leq n$, n being the order of the system. Note however, that in the case of $r < n$, a part of the system dynamics is rendered unobservable in the input-output linearisation, as the closed loop dynamics will have a lower order r than the whole dynamics of the system n. These internal dynamics must therefore be studied to establish the effectiveness of the feedback linearisation. For the case where the internal dynamics is stable, the input-output linearisation method can then be used. In the case of unstable internal dynamics however, these would lead to both undesirable and uncontrollable phenomena, in which case a different control

design is needed.

Given a MIMO system

$$\dot{x} = f(x) + G(x)u \tag{2.13}$$

$$y = h(x) \tag{2.14}$$

the ϕ functions are defined as $\phi_j^i(x) = L_f^{j-1}h_i(x)$ and satisfy a condition similar to 2.10

$$\dot{\phi}_1^i(x) = \phi_2^i, ..., \dot{\phi}_{r_i-1}^i = \phi_{r_i}^i \quad \text{and} \quad \dot{\phi}_{r_i}^i = L_f^{r_i}h_i(x) + \sum_{j=1}^{m} L_{g_j}L_f^{r_i-1}h_i(x)u_j \tag{2.15}$$

As this equation holds for $1 < i < p$, p being the number of outputs of the system, giving an expression for the pseudo control input υ

$$\upsilon = \begin{bmatrix} \dot{\phi}_{r_1}^1 \\ \vdots \\ \dot{\phi}_{r_p}^p \end{bmatrix} \tag{2.16}$$

It is also interesting to note the resulting system is not only linear, but also completely decoupled, as each new pseudo input only affects one output.

2.2 Improving the Feedback Linearisation Method

Although feedback control answers to many of the problems of linear controllers such as PID and LQR, it has a some downsides. Indeed, being a model based controller, it requires the system model to be quite accurately known, and parameter uncertainties can lead to undesired responses. Let υ be the vector of pseudo inputs, for a second order $\ddot{x} = f(x, \dot{x}, u)$, if it is input-output linearisable and the exact system is known, then it comes that, from [7]

$$\ddot{x} = \upsilon \tag{2.17}$$

However, even for simpler models, a real system is difficult to accurately describe, and an approximation $\hat{f}$ of f is usually chosen, resulting in $\upsilon = \hat{f}(x, \dot{x}, u)$. Therefore, defining a dynamic inversion error, $\Delta(x, \dot{x}, u) = f(x, \dot{x}, u) - \hat{f}(x, \dot{x}, u)$, comes that, as stated in [7]

$$\ddot{x} = \upsilon + \Delta(x, \dot{x}, u) \tag{2.18}$$

There can be many causes for a dynamic inversion error to be present, as not only modelling incertitudes can lead to larger errors, but also external interference (wind gusts) and actuator faults. The goal of this section will be to present the reader with some solutions to minimize this error.

2.2.1 Fuzzy Logic Systems

Fuzzy logic systems are based on the paradigm of continuous levels of truth between 0 and 1, rather than the usual discrete true or false levels of truth. Using these continuous truth values, a set of IF-ELSE

rules is chosen. The goal of this type of controller is to duplicate the way a pilot would respond to a given flight situation. These rules can base their control input on variables such as roll, angle of attack and sideslip [8].

This control method is based on three main parts. The first one if *fuzzification*, that consists of converting the plant outputs to fuzzy logic values. *Rule-based inference*, using the previously mentioned set of rules, a fuzzyfied control input is computed, which then goes through the *defuzzyfication* part of the control algorithm.

Fuzzy systems have some advantages: its linguistic interpretation of human knowledge facilitates the interpretation of results, and the knowledge base can be improved through addition of new rules. However, this method comes with some disadvantages. First of all, the set of rules that will improve the control of the aircraft is entirely dependent on expert knowledge, thus resulting in an empiric set of rules. This also means the method has no capability of generalization, as the control is only applied to specific cases, of the form "if the error is greater than a given value, apply this compensation to control input". It is also not robust to topological changes of the system [9]. For these reasons this method will not be further studied and implemented in this work.

2.2.2 Neural Networks

Artificial Neural Networks (NN) are a biologically inspired simulation of the brain nervous system. They are composed of simulated neurons and synapses. The sum of the inputs of a neuron are summed and the result is fed into an activation function, which then return a bounded value, either between -1 and 1 or 0 and 1. The inputs and outputs of a neuron are multiplied by a weight, representing the synapses between neurons. We get the following equation for one neuron

$$y(x_1, x_2, ..., x_n) = f((\sum_{i=1}^{n} w_i x_i) + b) \tag{2.19}$$

where b is a bias term and $f(x) : R \to R$ a given activation function. Neurons are then set in three different types of layers: the input, the output and the hidden layers. These are connected between each other, and the hidden part can be composed of several layers. The weights can then be tuned based on experience, making it suitable for intelligent learning. To tune these weights learning algorithms are used. There are two types of learning algorithms: batch and online training. It is called training to the process of iteratively reaching the ideal set of weights that will minimise the error between the output of the neural network and that of a given function.

Batch training is mostly based on a property of Neural Networks that, according to the Universal Approximation Theorem showed in 1989 by George Cybenko, a feed forward neural network can approximate any continuous function. The network is trained by providing input and output pairs. The weights are found in order to approximate the output of the neural network to that of the original function. Once these weights are computed, the network can then be used to compute outputs from inputs that were not part of the initial knowledge of input-output pairs.

Although this learning method approach is used in many NN applications, it cannot however be easily

used to improve an existing feedback linearisation controller, as it requires an *a priori* knowledge of the modelling error for each given input.

Online training, in contrary to batch training, can deal with dynamically changing environments. This training paradigm is not based on an *a priori* knowledge set, and instead relies on update laws that compute the new weights after each control iteration. It is this method that will be retained to adaptively correct the inversion error from the feedback linearisation. Training a network online, however, is less trivial when compared to batch training. An algorithm named backpropagation will be presented to provide a way of training a network online. A description on the implementation of the neural network and how it will be used to improve the feedback linearisation controller will be given on chapter 3.

The backpropagation algorithm is one of the most common NN training algorithm, commonly used in batch training, to update the weights of each layer of a network, in order to minimize a cost function. However, research has been done in order to use backpropagation techniques to create online adaptive and augmented control such as [10], [11], [12]. Indeed, it is also possible to use such algorithms to continuously update the weights of a network as the data is generated. Backpropagation can only be implemented if the activation function of the NN is differentiable.

The backpropagation algorithm can be described by a two phase cycle, Propagation and Weight update. During this last phase the gradient descent algorithm is used to optimize a cost function. As such, a cost function E must be chosen, and must be a function of the outputs of the neural network. A commonly used error function is

$$E = \frac{1}{2}(y_d - y)^2 \tag{2.20}$$

Where y_d is the desired output of the network and y is the actual output of the network. The $\frac{1}{2}$ factor is added to be cancelled when differentiating. For a given neuron i, the sum of its inputs sum_i is given by

$$sum_i = \sum_{k=1}^{n} w_{ki} x_k \tag{2.21}$$

Where w_{ki} is the weight of the k^{th} input of neuron i, and x_k are the outputs of the neurons form the previous layer. Given an activation function $f(x)$, we define the output of neuron i as

$$x_i = f(sum_i) \tag{2.22}$$

In order to use the gradient descent algorithm, the derivative of the cost function with respect to a given weight w_{ij} must be computed. To do so, the chain rule is used as

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial x_j} \frac{\partial x_j}{\partial sum_j} \frac{\partial sum_j}{\partial w_{ij}} \tag{2.23}$$

These three factors can now be easily computed. The last two factors are independent of weather x_j is a hidden or output layer and will for this reason be calculated firstly. From 2.21 and 2.22 comes that

$$\frac{\partial x_j}{\partial sum_j} = \frac{\partial f(sum_j)}{\partial sum_j} \tag{2.24}$$

For an output layer, where values may be unbounded, linear output functions may be used, for which the equation above has a trivial solution. For output layers of classification neural networks or for hidden layers in general, logistic or tangent sigmoid functions can be used instead. For the first case let $f(x) = \frac{1}{1+e^{-x}}$, then

$$\frac{\partial f(sum_j)}{\partial sum_j} = f(sum_j)(1 - f(sum_j)) \tag{2.25}$$

For the case of a tangent sigmoid activation function $f(x) = tanh(x)$ comes

$$\frac{\partial f(sum_j)}{\partial sum_j} = 1 - f^2(sum_j) \tag{2.26}$$

For the term $\frac{\partial sum_j}{\partial w_{ij}}$, from 2.21 this partial derivative is easily computed

$$\frac{\partial sum_j}{\partial w_{ij}} = \frac{\partial \left(\sum_{k=1}^{n} w_{kj} x_k\right)}{\partial w_{ij}} = x_i \tag{2.27}$$

For the output layer, the first term of equation 2.23 can easily be evaluated, as $x_j = y$ for such a case, giving

$$\frac{\partial E}{\partial x_j} = \frac{\partial(\frac{1}{2}(y_d - y)^2)}{\partial y} = y - y_d \tag{2.28}$$

For input layers however, this derivative is less obvious to estimate. One can consider that the error function E is a function of the inputs of all neurons that take x_j as input. Let S be such a set of inputs, from the total derivative to x_j comes that

$$\frac{\partial E}{\partial x_j} = \sum_{s \in S} \left(\frac{\partial E}{\partial x_s} \frac{\partial x_s}{\partial sum_s} \frac{\partial sum_s}{\partial x_j} \right) = \sum_{s \in S} \left(\frac{\partial E}{\partial x_s} \frac{\partial f(sum_s)}{\partial sum_s} w_{js} \right) \tag{2.29}$$

Note that the term $\frac{\partial E}{\partial x_s}$ represent the all the derivatives of the errors with respect to the outputs of the next layer of neurons. Taking into account that the last layer is easily evaluated without this method, the derivatives of all hidden neurons can be computed. Knowing the value of 2.23, an update law for the weights can now be stated from the gradient descent algorithm

$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}} \tag{2.30}$$

Where α is a learning coefficient, and its choice is crucial to assure a fast convergence of the solution. Indeed, while an exceedingly large learning coefficient can cause this algorithm to miss the minimum error, a small learning coefficient will also increase the convergence rate. A limitation of the backstepping algorithm that must also be taken into account is that it only guarantees local minima convergence. A detailed description on the connection of this algorithm with the control of the aircraft will be given in the next chapter.

Chapter 3

Implementation

This section will go in depth into the implementation of the concepts presented in chapter 2. The plane model to be controlled is first implemented and verified. The first goal after this first step is to design a controller that will allow the aircraft to follow a trajectory from several position waypoints through time. The influence of certain parameters and knowledge of the exact plane model will be studied and discussed in chapter 4. The final goal will be to improve the controller and its robustness by reducing tracking error through the use of an adaptive neural network.

3.1 Plane Model

The work made in this thesis was built on top of the work done by H. Escamilla Nuñez and F. Mora Camino on 4D trajectory tracking [13]. The model used in this work is a six degree of freedom transport aircraft that will be described in this section.

3.1.1 Frames of Reference

The first step before describing the dynamics of a commercial aircraft will be to define the frames of reference used to do so. The first frame of reference, on which 4D trajectories are described, corresponds to the WGS84 frame of reference. A second frame of reference corresponding to the aircraft body frame will be used to provide its fast rotational dynamics. Lastly all aerodynamic forces will be applied in the axial directions of the wind frame. This frame is aligned with the wind speed vector relative to the airplane, given by both the angle of attack α and the sideslip angle β. For these last two frames of reference, a rotation matrix can be defined from wind frame to body frame by

$$R_{BW} = \begin{bmatrix} c_\alpha c_\beta & -c_\alpha s_\beta & -s_\alpha \\ s_\beta & c_\beta & 0 \\ s_\alpha c_\beta & -s_\alpha s_\beta & c_\alpha \end{bmatrix} \tag{3.1}$$

To describe the attitude of the plane Euler, roll, pitch and yaw angles, will also be used, namely $\phi\{-\pi, \pi\}; \theta\{-\frac{\pi}{2}, \frac{\pi}{2}\}; \psi\{-\pi, \pi\}$. From these angles the rotation matrix from the body to the earth frame

is given by

$$R_{EB} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta c_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{3.2}$$

3.1.2 Fast Dynamics

The considered actuators of the aircraft that control its attitude are given by the control surface deflection $\delta = [\delta_{ail}\delta_{ele}\delta_{rud}]^T$, each applying a torque along an axis of the body frame. These torques are given by

$$\begin{bmatrix} L' \\ M \\ N \end{bmatrix} = \frac{1}{2}\rho S V_a^2 \left(\begin{bmatrix} bC_l \\ \bar{c}C_m \\ bC_n \end{bmatrix} + C_\delta \delta \right) \tag{3.3}$$

where $\bar{c}$ and b represent the wing mean chord and its span respectively, C_δ and the moment coefficients $[C_l C_m C_n]^T$ are given by

$$C_\delta = \begin{bmatrix} bC_{l\delta_{ail}} & 0 & bC_{l\delta_{rud}} \\ 0 & \bar{c}C_{m\delta_{ele}} & 0 \\ bC_{n\delta_{ail}} & 0 & bC_{n\delta_{rud}} \end{bmatrix} \tag{3.4}$$

$$\begin{bmatrix} C_l \\ C_m \\ C_n \end{bmatrix} = \begin{bmatrix} C_{l\beta}\beta + C_{l_p} p\frac{b}{2V_a} + C_{l_r} r\frac{b}{2V_a} \\ C_{m_0} + C_{m_\alpha}\alpha + C_{m_q} q\frac{\bar{c}}{2V_a} \\ C_{n\beta}\beta + C_{n_p} p\frac{b}{2V_a} + C_{n_r} r\frac{b}{2V_a} \end{bmatrix} \tag{3.5}$$

Where p, q, r are the body angular rates ($\Omega = [p \quad q \quad r]^T$) and V_a is the airspeed. The method for obtaining the coefficients of equation 3.5 will be provided in the chapter to follow. Having defined the torques applied to the aircraft the rotational dynamics equation can now be stated as per [13], I being the aircraft inertial matrix.

$$\dot{\Omega} = I^{-1} M_{ext} - I^{-1}\Omega \times (I\Omega) \tag{3.6a}$$

$$\dot{\Omega} = \frac{1}{2}\rho S I^{-1} V_a^2 \left(\begin{bmatrix} bC_l \\ \bar{c}C_m \\ bC_n \end{bmatrix} + C_\delta \delta \right) - I^{-1}\Omega \times (I\Omega) \tag{3.6b}$$

These two equations can be rearranged to account for the effect of the wind, allowing further on to simulate the behaviour of the airplane in the presence of wind disturbances. Let $\vec{V_G} = [u \quad v \quad w]^T$ be the speed of the CG relative to the ground, $\vec{V}$ the speed of the CG relative to the air mass and $\vec{W}$ the speed of the wind relative to the ground, then as per Etkin and Reid [14]

$$\vec{V_G} = \vec{V} + \vec{W} = \begin{bmatrix} V_a c_\alpha c_\beta + V_{w_x} \\ V_a s_\beta + V_{w_y} \\ V_a s_\alpha c_\beta + V_{w_z} \end{bmatrix} \tag{3.7}$$

and α and β can be computed by

$$\alpha = arctan\left(\frac{w - V_{w_z}}{uV_{w_x}}\right) \tag{3.8a}$$

$$\beta = arctan\left(\frac{v - V_{w_y}}{V_a}\right) \tag{3.8b}$$

From these three equations, differentiating 3.8a and 3.8b, and from equation 3.7 and the translation dynamics equation 3.13 comes that

$$\begin{bmatrix}\dot{\alpha}\\ \dot{\beta}\\ \dot{V}_a\end{bmatrix} = \begin{bmatrix}H_{11} & H_{12} & H_{13}\\ H_{21} & 0 & H_{23}\\ H_{31} & H_{32} & H_{33}\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} + \begin{bmatrix}Q_1\\ Q_2\\ Q_3\end{bmatrix} \tag{3.9}$$

Where the entries of the matrix are given by

$$H_{11} = \frac{-V_a c_\alpha s_\beta c_\beta - V_{w_y} c_\alpha c_\beta}{V_a(1+\tan^2\alpha)c_\alpha^2 c_\beta^2}$$

$$H_{12} = \frac{V_a(c_\alpha^2 c_\beta^2 - s_\alpha^2 c_\beta^2) + V_{w_x} c_\alpha c_\beta - V_{w_z} s_\alpha s_\beta}{V_a(1+\tan^2\alpha)c_\alpha^2 c_\beta^2}$$

$$H_{13} = \frac{-V_a s_\alpha s_\beta c_\beta - V_{w_y} s_\alpha c_\beta}{V_a(1+\tan^2\alpha)c_\alpha^2 c_\beta^2}$$

$$H_{21} = \frac{V_a s_\alpha c_\beta + V_{w_z}}{V_a c_\beta}$$

$$H_{23} = -\frac{V_a c_\alpha c_\beta + V_{w_x}}{V_a c_\beta}$$

$$H_{31} = 2\left(-V_a s_\beta s_\alpha c_\beta - V_{w_x} s_\alpha c_\beta\right)$$

$$H_{32} = 2\left(-V_{w_z} c_\alpha c_\beta + V_a s_\alpha c_\beta s_\beta + V_{w_z} s_\beta + V_{w_x} s_\alpha c_\beta\right)$$

$$H_{33} = 2\left(V_{w_y} c_\alpha c_\beta - V_{w_x} s_\beta\right)$$

$$Q_1 = \frac{\left(\frac{1}{m}F_{za} + g c_\theta c_\phi - \dot{V}_{w_z}\right)c_\alpha c_\beta - \left(\frac{1}{m}(F_{xa}+T) - g s_\theta - \dot{V}_{w_x}\right)}{V_a(1+\tan^2\alpha)c_\alpha^2 c_\beta^2}$$

$$Q_2 = \frac{\frac{1}{m}F_{ya} + g c_\theta s_\phi - \dot{V}_{w_y}}{c_\beta}$$

$$Q_3 = 2\left(\left(\frac{-1}{m}(F_{xa}+T) + g s_\theta - \dot{V}_{w_x}\right)c_\alpha c_\beta + \left(\frac{-1}{m}F_{ya} - \dot{V}_{w_y}\right)s_\beta + \left(\frac{-1}{m}F_{za} - \dot{V}_{w_z}\right)s_\alpha c_\beta\right)$$

The forces in the air frame F_{xa}, F_{ya}, F_{za} will be further detailed in the next section on translation dynamics.

The angular rates are also related to the Euler angles. The relationship between the Euler angles and the rotation rates is also one that will prove useful when implementing the model in a Matlab simulation, and is given by

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & tg_\theta s_\phi & tg_\theta c_\phi \\ 0 & c_\phi & -s_\phi \\ 0 & \frac{s_\phi}{c_\theta} & \frac{c_\phi}{c_\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.10}$$

3.1.3 Translation Dynamics

This subsection focuses on the forces applied to aircraft, introducing a new actuation variable, the thrust force T. These forces are applied along the three axes of the wind frame, lift, drag and side force, given by

$$\begin{bmatrix} D \\ Y \\ L \end{bmatrix} = \frac{1}{2}\rho S V_a^2 \begin{bmatrix} C_D \\ C_Y \\ C_L \end{bmatrix} \tag{3.11}$$

Once again, the method used to compute these coefficients will be given in the chapter 3 in detail. These coefficients, similarly to the moment coefficients, are functions of the angle of attack, sideslip angle and airspeed, the three most relevant variables when determining aerodynamic forces and moments. Although aerodynamic forces are usually expressed in the wind frame, as the thrust is always applied along the x axis of the body frame, it is necessary to rotate the aerodynamic forces to this frame. This way the sum of the airplane forces can be obtained.

$$\begin{bmatrix} F_{xa} \\ F_{ya} \\ F_{za} \end{bmatrix} = R_{WB} \begin{bmatrix} -D \\ Y \\ -L \end{bmatrix} \tag{3.12}$$

From Newton's Second Law comes the aircraft acceleration

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} \frac{1}{m}(F_{xa} + T) - gs_\theta + rv - qw \\ \frac{1}{m}F_{ya} + gc_\theta s_\phi + pw - ru \\ \frac{1}{m}F_{za} + gc_\theta c_\phi + qu - pv \end{bmatrix} \tag{3.13}$$

An expression in the Earth frame can also be obtained

$$\begin{bmatrix} \ddot{x}_E \\ \ddot{y}_E \\ \ddot{z}_E \end{bmatrix} = \frac{1}{m} R_{BE} \begin{bmatrix} F_{xa} + T \\ F_{ya} \\ F_{za} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{3.14}$$

3.1.4 Actuator Dynamics

Finally, to simulate the delay response in actuation in order to have a realistic simulation, first order systems were introduced to the actuator dynamics as per [13]. For the control surfaces δ_i, given a desired δ_i^d comes

$$\dot{\delta}_i = \frac{1}{\xi_i}(\delta_i^d - \delta_i) \tag{3.15}$$

Similarly for thrust

$$\dot{T} = \frac{1}{\xi_T}(T^d - T) \tag{3.16}$$

ξ_i and ξ_T are time constants. As the responsiveness of the resultant thrust will be much slower than that of the control surfaces, $\xi_T >> \xi_i$. The chosen commercial aircraft that will be simulated is the Boeing 737-200, an aircraft with over 30 years of service for which some information of flight parameters is readily available, such as weight, wing span and mean chord. The simulation was made in a cruise flight environment, at 200 m/s velocity at 10000 m above the ground. The chosen inertial matrix for this aircraft is given by

$$\begin{bmatrix} 1278369.56 & 0 & -135588.17 \\ 0 & 3781267.79 & 0 \\ -135588.17 & 0 & 4877649.98 \end{bmatrix} kg.m^2 \tag{3.17}$$

The ISA atmospheric model was used to compute the air density at any given height.

Table 3.1: Boeing 737-2 parameters

Weight m	52390 kg
Wing Span b	28.35 m
Wing Area S	102.0 m^2
Wing mean chord $\bar{c}$	4.35 m
Length l	30.53 m

The time constants used for the actuators was ξ_{δ_i}50ms and $\xi_T = 4$s for the engines.

3.1.5 Airplane Parameters and Coefficients

In order to simulate the aircraft fast dynamics, its moments must first be computed to use equation 3.6. The torque caused by the actuators is modelled as equation 3.4, using

$$C_{L_{\delta_{ail}}} = 0.02 \quad rad^{-1}$$
$$C_{L_{\delta_{rud}}} = 0.002 \quad rad^{-1}$$
$$C_{M_{\delta_{ele}}} = -0.003 \quad rad^{-1}$$
$$C_{N_{\delta_{ail}}} = -0.002 \quad rad^{-1}$$
$$C_{N_{\delta_{rud}}} = -0.07 \quad rad^{-1}$$

The remaining aerodynamic coefficients from equation 3.5 were obtained from the work of Hector Escamilla Nuñez in [13].

In this work, neural networks were used to interpolate data from the USAF Stability and Control Digital Data Compendium. A two layer feed forward network, with a hidden sigmoid activation function and a linear output activation function, was then trained using the gathered data with the Bayesian Regulation training algorithm [13]. Indeed as stated previously, neural networks can approximate any continuous function according to the Universal Approximation Theorem. This method is therefore opti-

mal to accurately approximate the variation of these coefficients that mainly vary with the angle of attack α and airspeed V_a. The only exception to this method was the drag coefficient, given for this aircraft by

$$C_D = 0.0176 + 0.0515C_L^2 \tag{3.18}$$

The sum of the moments can be computed and used to obtain the rotation rates p, q and r. Equation 3.10 can also be used and integrated to obtain the Euler angles. These will also be necessary for frame of reference changes, namely from body to earth and vice-versa. The aerodynamic forces were computed from equation 3.11 using the outputs of the neural networks. The body forces and acceleration relative to the earth frame were then obtain from equations 3.12 and 3.13. The effects of the wind were also taken into account by adding the wind speed to the integration of the acceleration of the aircraft relative to the earth as per 3.7. At this point the values of α and β were also obtained for their respective equations 3.8a and 3.8b.

A simplified block diagram of the plane simulator is given by figure 3.1.

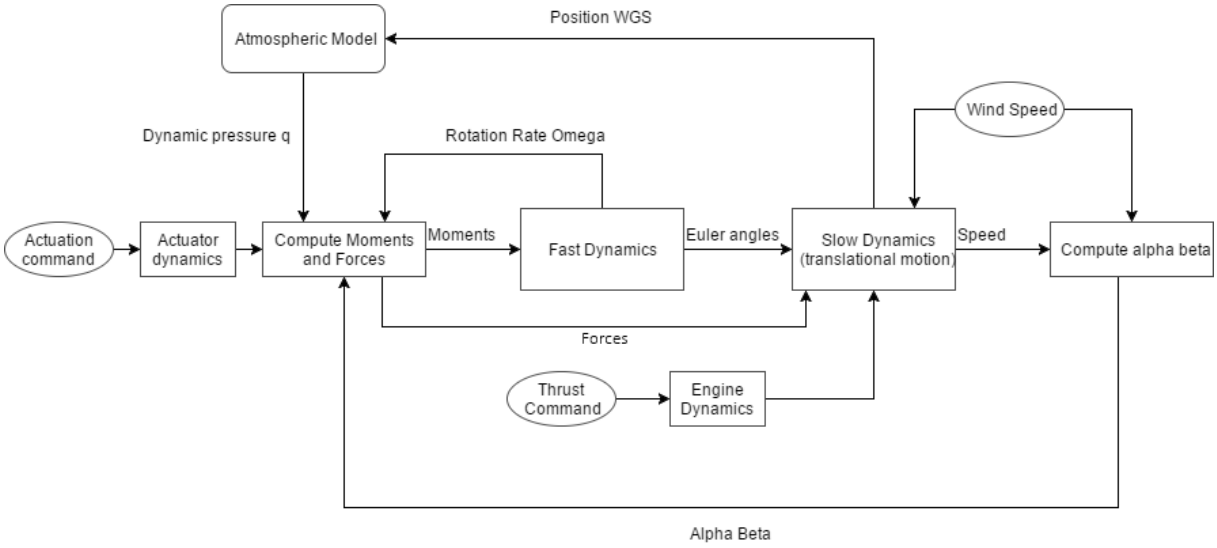

Figure 3.1: Plane dynamics simulator diagram

3.2 Controller Implementation

To invert such a complex system (figure 3.1), two layers of inversion are proposed by H. Escamilla [13], namely for the fast and slow dynamics. Directly controlling these are three of the four actuator control inputs, the control surfaces for the aileron, elevon and rudder. These will therefore be the output of the nonlinear inversion control.

3.2.1 Fast Dynamics

To invert the fast dynamics equation 3.6 must be differentiated once, to account for both the actuator dynamics 3.15 and the wind effects. Doing so yields the jerk vector given by, as per [13]

$$\begin{bmatrix} \dddot{p} \\ \dddot{q} \\ \dddot{r} \end{bmatrix} = \frac{1}{2}\rho S I^{-1} \left\{ V_a^2 C_\delta \xi^{-1} \begin{bmatrix} \delta_{ail}^d - \delta_{ail} \\ \delta_{ele}^d - \delta_{ele} \\ \delta_{rud}^d - \delta_{rud} \end{bmatrix} + V_a^2 C_c \dot{R}_a + 2V_a \dot{V}_a \left(\begin{bmatrix} bC_l \\ \bar{c}C_m \\ bC_n \end{bmatrix} + C_\delta \begin{bmatrix} \delta_{ail} \\ \delta_{ele} \\ \delta_{rud} \end{bmatrix} \right) \right\} + I^{-1} \left(\frac{1}{4}\rho S V_a C_k - I_n \right) \dot{\Omega} \tag{3.19}$$

where

$$I = \begin{bmatrix} I_{xx} & 0 & -I_{xz} \\ 0 & I_{yy} & 0 \\ -I_{zx} & 0 & I_{zz} \end{bmatrix}$$

$$I_n = \begin{bmatrix} -I_{xz}q & (I_{zz} - I_{yy})r - I_{xz}p & (I_{zz} - I_{yy})q \\ (I_{xx} - I_{zz})r + 2I_{xz}p & 0 & (I_{xx} - I_{zz})p - 2I_{xz}r \\ (I_{yy} - I_{xx})q & (I_{yy} - I_{xx})p + I_{xz}r & I_{xz}q \end{bmatrix}$$

$$\xi = \begin{bmatrix} \xi_{ail} & 0 & 0 \\ 0 & \xi_{ele} & 0 \\ 0 & 0 & \xi_{rud} \end{bmatrix}$$

$$C_c = \begin{bmatrix} 0 & bC_{l_\beta} & -\frac{b^2}{2V_a^2}(C_{l_p}p + C_{l_r}r) \\ \bar{c}C_{m_\alpha} & 0 & -\frac{\bar{c}^2}{2V_a^2}C_{m_q}q \\ 0 & bC_{n_\beta} & -\frac{b^2}{2V_a^2}(C_{n_p}p + C_{n_r}r) \end{bmatrix}$$

$$C_k = \begin{bmatrix} b^2 C_{l_p} & 0 & b^2 C_{l_r} \\ 0 & \bar{c}^2 C_{m_q} & 0 \\ b^2 C_{n_p} & 0 & b^2 C_{n_r} \end{bmatrix}$$

$$\dot{R}a = [\dot{\alpha} \quad \dot{\beta} \quad \dot{V}_a]^T$$

To do a feedback linearisation, a pseudo input must be chosen, in this case $\tau = \ddot{\Omega}$. The feedback control law will therefore be, solving equation 3.19 for δ^d,

$$\begin{bmatrix} \delta_{ail}^d \\ \delta_{ele}^d \\ \delta_{rud}^d \end{bmatrix} = \frac{1}{V_a^2}\xi C_\delta^{-1} \left\{ \frac{2I}{\rho S}\tau - \frac{2}{\rho S}(\frac{1}{4}\rho S V_a C_k - I_n)\dot{\Omega} - 2V_a\dot{V}_a \left(\begin{bmatrix} bC_l \\ \bar{c}C_m \\ bC_n \end{bmatrix} + C_\delta \delta \right) - V_a^2 C_c \dot{R}_a \right\} + \delta \tag{3.20}$$

Indeed, by replacing 3.20 in the jerk equation 3.19, the equation $\ddot{\Omega} = \tau$ is obtained, resulting in a

successfully inverted system.

The wind effects will appear in the terms including $\dot{R}_a = [\dot{\alpha} \quad \dot{\beta} \quad \dot{V}_a]^T$, as the equation defining $\dot{V}_a$ can be obtained differentiating the norm of the speed relative to the ground. The value of $\dot{R}_a$ can be computed from equation 3.1.2.

Should all of the parameters mentioned in the equations above be known and no inversion error be made, the resulting system $\ddot{\Omega} = \tau$ will be both linear and decoupled, having three pseudo inputs $\tau = [\tau_p \quad \tau_q \quad \tau_r]^T$. As mentioned in section 2.1, the next step of the nonlinear inversion shall be to propose a linear controller for this resulting system. Taking the desired rotation rates Ω^d as inputs comes a control law for τ, first considering without neural network adaptation

$$\tau = -K_P(\Omega - \Omega^d) - K_D(\dot{\Omega} - \dot{\Omega}^d) + \ddot{\Omega}^d \tag{3.21}$$

Where K_P and K_D are chosen to ensure stability and reference tracking. The previous equation can also be written as

$$\ddot{e} = -K_P e - K_D \dot{e} \tag{3.22}$$

assuming that an ideal inversion is obtained and $\tau = \ddot{\Omega}$, and where $e = \Omega - \Omega^d$.

This way the initial non-linear system becomes a second order linear one, for which natural frequencies and damping values for each dimension must be chosen. Choosing $x_1 = e_i$ and $x_2 = \dot{x_1}$ for a given dimension i of Ω, either p, q, r. Then comes the state equation

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -K_{P_i} & -K_{D_i} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{3.23}$$

The characteristic equation is then obtained as

$$\det(sI - A_i) = \lambda^2 + 2\lambda\omega_{n_i} + \omega_{n_i}^2 = \lambda^2 + \lambda K_{D_i} + K_{P_i} = 0 \tag{3.24}$$

$$\lambda = \frac{K_{D_i}\sqrt{K_{P_i}}}{2K_{P_i}} \tag{3.25}$$

$$\omega_{n_i} = \sqrt{K_{P_i}} \tag{3.26}$$

The chosen values are given in table 3.2.

Table 3.2: Linearised system chosen dynamics

i	K_{P_i}	K_{D_i}	λ	ω_{n_i}(rad s^{-1})
p	100	20	1	10
q	5	1	0.22	2.24
r	100	20	1	10

From this point the next step will be to obtain the desired values of Ω. These are obtained using another PD controller, using Euler angles values to control rotation rates. The yaw rate was set to zero as

the feedback linearisation decouples the plane movement, using roll and pitch motion to turn the aircraft.

$$p = k_{P_p}(\phi^d - \phi) + k_{D_p}(\dot{\phi^d} - \dot{\phi}) \tag{3.27a}$$

$$q = k_{P_q}(\theta^d - \theta) + k_{D_q}(\dot{\theta^d} - \dot{\theta}) \tag{3.27b}$$

$$r = 0 \tag{3.27c}$$

3.2.2 Slow Dynamics

It is now possible to control the attitude of the aircraft, and the fast control loop design is completed. To follow 4D trajectories however, a guidance control loop must be designed to feed attitude and thrust reference values to the system described so far and to achieve the goal of position tracking over time. First the dynamics for speed (V_a), heading (ψ) and flight path angle ($\gamma = \theta - \alpha$) are given by, neglecting wind disturbances, from Newton's Law in the wind frame

$$\dot{V_a} = \frac{1}{m}(T\cos\alpha - D - mg\sin\gamma) \tag{3.28a}$$

$$\dot{\gamma} = \frac{1}{mV_a}(T\sin\alpha + L - mg\cos\gamma) \tag{3.28b}$$

For the yaw rate comes that

$$\dot{\psi} = \frac{g}{V_a}\tan\phi \tag{3.28c}$$

After choosing the desired dynamics to control these variables, the equations above must be solved for both the thrust reference T, the desired AoA and ψ. From these last two, knowing the current roll angle θ, the input for the fast dynamics controller is obtained. The desired dynamics were chosen as first order systems as

$$\dot{V_a} = \frac{1}{\xi_{V_a}}(V_a^d - V_a) \tag{3.29a}$$

$$\dot{\gamma} = \frac{1}{\xi_{\gamma}}(\gamma^d - \gamma) \tag{3.29b}$$

$$\dot{\psi} = \frac{1}{\xi_{\psi}}(\psi^d - \psi) \tag{3.29c}$$

A block diagram of the controller used to compute the pseudo input τ is described in figure 3.2.

3.2.3 Guidance Law

The final step of the controller design will be to propose a guidance law to obtain the values of V_a^d, γ^d and ψ^d from the X,Y and Z reference values along time. For these laws it was assumed that the position reference was given relative to a frame of reference with the origin on the Earth surface, with the Z axis pointing upwards. The guidance controller therefore implements the following equations, assuming the

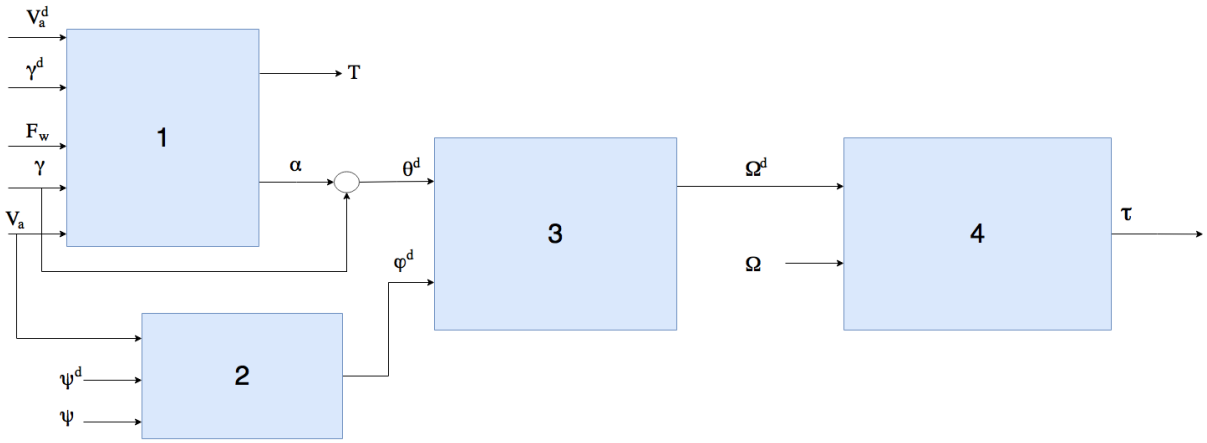

Figure 3.2: This block diagram describes the method used to compute τ, the pseudo input to the fast dynamics inversion. In block **1** equations 3.29a, 3.29b, 3.28b and 3.28b are solved to compute the desired θ^d and thrust. Similarly in block **2** equations 3.29c and 3.28c are also solved to obtain ϕ^d. From these values in block **3** go through the PD controller described in equatio 3.27, resulting in Ω^d, the input of block **4**. This last block is the linear controller described by equation 3.21, which output τ.

errors $\delta x = x_r - x$, $\delta y = y_r - y$ and $\delta z = z - z_r$. The z formula is slightly different to the other two as in the airplane frame of reference the z axis is pointing downwards, allowing this way to use a z reference relative to the Earth surface.

$$\delta V_a = \frac{1}{\tau_{V_a}}(\sqrt{\delta x^2 + \delta y^2 + \delta z^2}) \tag{3.30a}$$

$$V_a^d = \delta V_a + V_{a_0} \tag{3.30b}$$

$$\psi^d = atan2(\delta y, \delta x) \tag{3.30c}$$

$$\gamma^d = \frac{1}{\tau_Z}\delta z \tag{3.30d}$$

This guidance law is intended to be used having at each time interval a x_r, y_r and z_r position of a desired trajectory. From these an airspeed, heading and FPA must be computed and fed to the controller.

Taking first the airspeed equation from 3.30, V_{a_0} is the desired speed of the aircraft in cruise conditions when following the trajectory. This speed is then corrected by the term δV_a, increasing or decreasing aircraft airspeed depending on its position relative to the trajectory desired position at a given time, i.e the distance error. By dividing this error by τ_{V_a}, a desired convergence time, the speed correction to the initial value V_{a_0} necessary to follow a 4D trajectory is obtained. A more accurate method of obtaining an airspeed reference from these equations is specified in 3.31. Also, from the x and y position errors between the aircraft and the current trajectory waypoint the desired heading can also be computed using the $atan2$ function. Finally, the method used to obtain the Flight Path Angle relies on the error between the height of the aircraft and the desired one to compute it.

Using solely equations 3.30 however will not grant optimal results, and may even sometimes cause

loss of control of the aircraft. The goal for this 4D guidance law and of this work is to, at any given time, to be at a 3D coordinate point of a trajectory at that time. To do so, the references of speed, heading and flight path angle must be constantly adjusted to reach a 4D reference. Using the equations described previously for the guidance controller, it may happen that the aircraft would "overtake" the trajectory reference, causing the heading reference to increase to either 180^o or -180^o and therefore making the aircraft diverge significantly from its original trajectory. To avoid this, the following guidance strategy was used:

- **Heading** To prevent the heading reference to excessively grow and make the aircraft do a 180^o turn, a small correction code in Simulink was added (see algorithm 1) in order to detect large errors between the desired and measured heading, caused by the aircraft "overtaking" its trajectory. In case of large errors, this correction would force the desired heading to be equal to its measured value, thus preventing the aircraft from making large turns.

  Data: Ψ and Ψ_d
  Result: Ψ command
  error = Ψ_d - Ψ;
  output = Ψ_d;
  if *error* $> 160^o$ *or error* $< -160^o$ **then**
  | output = Ψ
  end

  Algorithm 1: Ψ command correction algorithm

- **Airspeed** Now that the aircraft no longer does 180^o turns when it overtakes the desired trajectory, the desired airspeed must now be adjusted in order to slow down and therefore reach the desired position at the desired time. To do so the guidance law was slightly modified to

$$V_a^d = \begin{cases} \delta V_a + V_{a_0} & \text{if} \quad |atan2(\delta y, \delta x)| < 90^o \\ \delta V_a - V_{a_0} & \text{if} \quad |atan2(\delta y, \delta x)| > 90^o \end{cases} \tag{3.31}$$

Finally, the block diagram of the controller described thus far is given by figure 3.3, in which the linear controller is described in more detail in figure 3.2, and the plane model in figure 3.1.

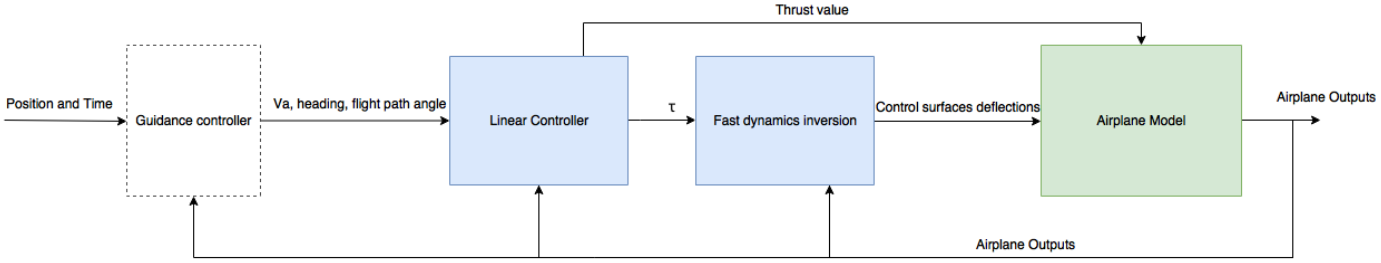

Figure 3.3: Diagram of the controller and model architecture, without NN compensation

3.3 Neural Network

Over the years, intelligent control techniques using neural networks have become a growing research topic, addressing the limitations of state-of-the-art model based controllers such as linear quadratic Gaussian, model predictive control, backstepping and gain scheduling [1]. Indeed, variations in the plane dynamics (e.g., due to payload changes, actuator or sensor degradation) deteriorate the error of the controller. To compensate for such errors, adaptive online neural networks have been studied and implemented in several works [15], [16], [11] and [17] to name a few. These however, have been applied almost exclusively to UAV aircraft, especially multicopters. For this work a feedforward network with one hidden layer (one of the most widely implemented neural network architecture [1]) was used.

3.3.1 Network Architecture

From an error-less nonlinear inversion, the pseuso-input τ would directly control $\ddot{\Omega}$ as it was seen previously. However, in case of inversion errors, that can be caused by several factors such as modelling errors or external disturbances, resulting in an equation similar to 2.18, that $\tau = \ddot{\Omega}^d + \Delta$, where $\Delta = \ddot{\Omega} - \ddot{\Omega}^d$. This method adds an adaptive component τ_{NN} to the pseudo input that will approximate the behaviour of the Δ error. The resulting control law will be given by $\tau + \tau_{NN} = \ddot{\Omega}^d + \Delta$, in which as $t \to \infty$, $\tau_{NN} \to \Delta$, thus adaptively minimizing inversion errors. To do so a simple neural network architecture was chosen: a network with a single hidden layer with therefore two sets of weights V and W that will need to be trained, corresponding to the input weights and the hidden layer ones. As for the activation functions for the hidden layer a sigmoid function was chosen, $sig(x) = \dfrac{1}{(1+e^{-x})}$. As the outputs must not be bound between $[0; 1]$, the chosen output function was a linear one. Indeed sigmoid output functions are usually used in applications where the output must be a boolean value. A description of the chosen architecture can be found in figure 3.4. Different numbers of neurons were used and the results obtained will be discussed in chapter 4.

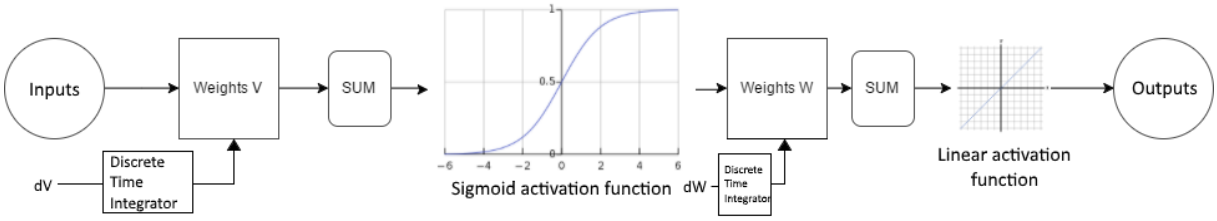

Figure 3.4: Neural Network diagram

There are quite some challenges in designing a feed forward neural network. Besides choosing the number of layers and neurons per layer, the inputs of the network must also be chosen carefully. Some rules to choose the correct inputs are given in [18]:

- **Relevance**: This is the most important consideration to have while choosing inputs, this set must be sufficiently informative of the state of the system for the network to perform correctly. A NN will perform poorly if the output behaviour is not correlated to its input.

- **Redundancy**: A large value of redundant and irrelevant input variables will not only increase the needed computational effort of the NN, but will also increase the difficulty to train the weights of the network and add noise to the system.

Another consideration about the chosen inputs that must also be taken into account is their range. The sigmoid function $sig(x)$ used reaches around 73% when $x = 1$ and 88% when $x = 2$, as can be seen in figure 3.5. For some cases the inputs might therefore need to be normalized. While this is quite a trivial task in the case of batch training, as the minimum and maximum value of each input is easily obtained before even starting training, such is not the case for online training, and a maximum and minimum value must be proposed *a priori* for each input. The mean of these values should also be close to zero. Should the input not be normalized and reach absolute values much greater than the unity, its activation function output would be constant and the system would therefore not react to input change. This, clearly, is not desirable. To map the input x knowing its minimum and maximum values to its normalized equivalent y, the following equation is used

$$y = 2\frac{x - x_{min}}{x_{max} - x_{min}} - 1 \tag{3.32}$$

For this particular case, the state variables of the fast dynamics where used as inputs of the network, namely Ω and $\dot{\Omega}$.

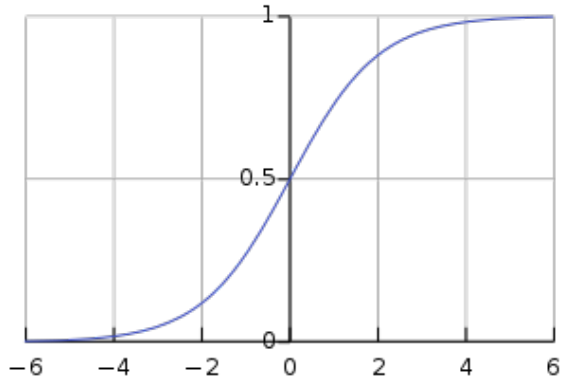

Figure 3.5: Sigmoid Function

3.3.2 Training Algorithm

As mentioned in section 2.2.2, a backpropagation algorithm was used to train both sets of weights online. This algorithm, however, must have an error function that will be minimized by iteratively changing the values of the weights V and W. As seen previously, an inversion error Δ will be present, where $\tau = \ddot{\Omega}^d + \Delta$. This error must be computed for each iteration to train the network to approximate this error. The error can be expressed as $\Delta = \tau - \ddot{\Omega}^d$. From the control law used to obtain τ (3.21) it comes that

$$\Delta = -K_P(\Omega - \Omega^d) - K_D(\dot{\Omega} - \dot{\Omega}^d) \tag{3.33}$$

Finally the cost function used to train the network is given by

$$J = \frac{1}{2}(\Delta - y_{NN})^2 \tag{3.34}$$

The values of $\dot{V}$ and $\dot{W}$ must now be computed, using the gradient descent method described in section 2.2.2. As the name suggests, the back propagation algorithm first computes the weight update values for the last layers, propagating the errors to update the previous weight set. The weights are constantly updated until the error absolute value in lesser than a given value Δ_{max}, in which case these are kept constant as $\dot{V} = \dot{W} = 0$.

One of the parameters that must be tuned that has the largest impact on the performance of the network is the learning rate α. This coefficient can also be thought of as the step size in incrementing the weight sets. Let V^* and W^* be the two ideal set of weights that, for a given input, result in an absolutely minimal cost function. A small value will allow the weights to converge to their ideal value, although taking many iterations to do so. A large learning rate however might converge faster, but might also overshoot and miss its optimal value. The results of the variation of this rate will be studied in chapter 4.

Finally, adding the online neural network to the baseline controller, the final control architecture is obtained. For the following chapter, the controller without the correction will be tested and compared in the same conditions to the same controller including the network. A simplified diagram of the full system, (controller, model and neural network), can be seen in figure 3.6.

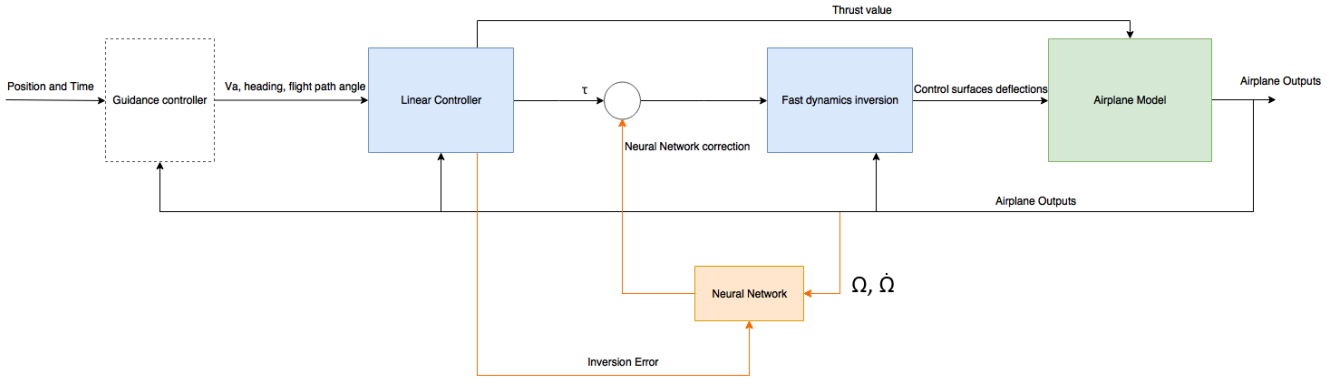

Figure 3.6: Diagram of the controller architecture including NN correction (in orange)

Chapter 4

Results

This chapter presents the simulation results in the behaviour of the aircraft for the different solutions proposed to control and stabilise it. Once the goal for the results of this chapter is properly established, the first step will be to validate the model which has been implemented. This will be done by controlling the aircraft into cruise conditions using the baseline feedback linearisation error. The effects of disturbances and inversion errors will then be studied regarding their effects on the aircraft dynamics. Inversion errors will first be studied, first on the baseline controller then on the NN corrected controller, comparing the results of both simulations. The same methodology will then be used to study responses to system failures.

The next step will be to achieve the goal of this thesis and demonstrate the effects of an on-line neural network in reducing the tracking error in the presence of these disturbances. Finally, from the adaptive controller, including the neural network, the guiding law described in chapter 3 in 3.30 will be added to follow a given trajectory. All the simulations described in this chapter were made in Matlab/Simulink with a fixed step of $f_{sample} = 30$ Hz.

4.1 Model Validation

The goal in this section will be to validate the behaviour of the model in cruise flight. Assuming cruise conditions yelds

$$T = D \tag{4.1}$$

$$W = L \tag{4.2}$$

$$L' = M = N = 0 \tag{4.3}$$

In order to verify the model described so far, the required thrust to have cruise conditions will be computed for a given plausible value of α. From this point the airspeed of the aircraft can also be computed. The graph of C_L versus alpha was also obtained from its respective neural network, as detailed in subsection

3.1.5, given by figure 4.1 From the cruise conditions 4.3, knowing that $L = \frac{1}{2}\rho S V^2 C_L$, solving for the

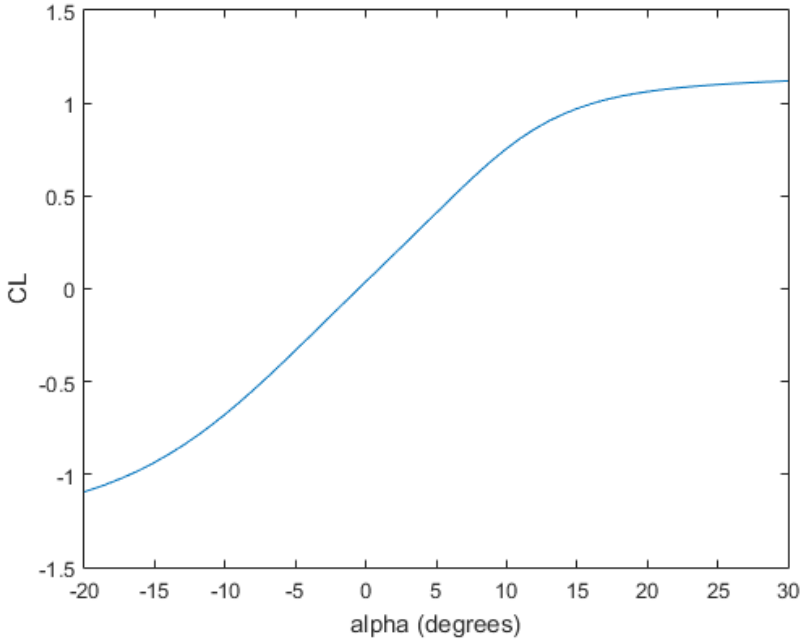

Figure 4.1: C_L versus α graph from Neural Network

airspeed V comes that

$$V = \sqrt{\frac{2mg}{\rho S C_L}} \tag{4.4}$$

The required thrust can also be computed from 4.3, 4.4, 3.11 and 3.18, knowing C_L for a given angle of attack. Proposing some values of α, the following results are obtained

Table 4.1: Required cruise conditions for different values of α

$\alpha(^o)$	C_D	C_L	$V(ms^{-1})$	$T(N)$
0	0.017677131	0.0387	707.4010791	224047.3585
2	0.019379779	0.1859	322.7613368	51133.84325
4	0.023345134	0.334	240.7952785	34283.79709
6	0.029604436	0.4828	200.279994	30076.58604

From these values, to test both the model (as well as the methods used to simulate the dynamics of the aircraft, including the coefficients neural networks) and the feedback linearisation controller, the following references $V_a^d = 200ms^{-1}$, $\gamma^d = 0$ rad and $\psi^d = 0$ rad were used in an attempt to simulate cruise conditions. From there, the values of thrust and angle of attack were computed and compared to the theoretical values of table 4.1.

Following these constant reference values the results obtained can be seen in figure 4.2.

This simulation, made over 800 seconds, shows clearly that the angle oscillates around a 6^o degree angle of attack. As for thrust, it also oscillates around $30000N$. These values correspond to the theoretical values in table 4.1 for $\alpha = 6^o$, indicating that not only the modelled plane is well behaved, but also that

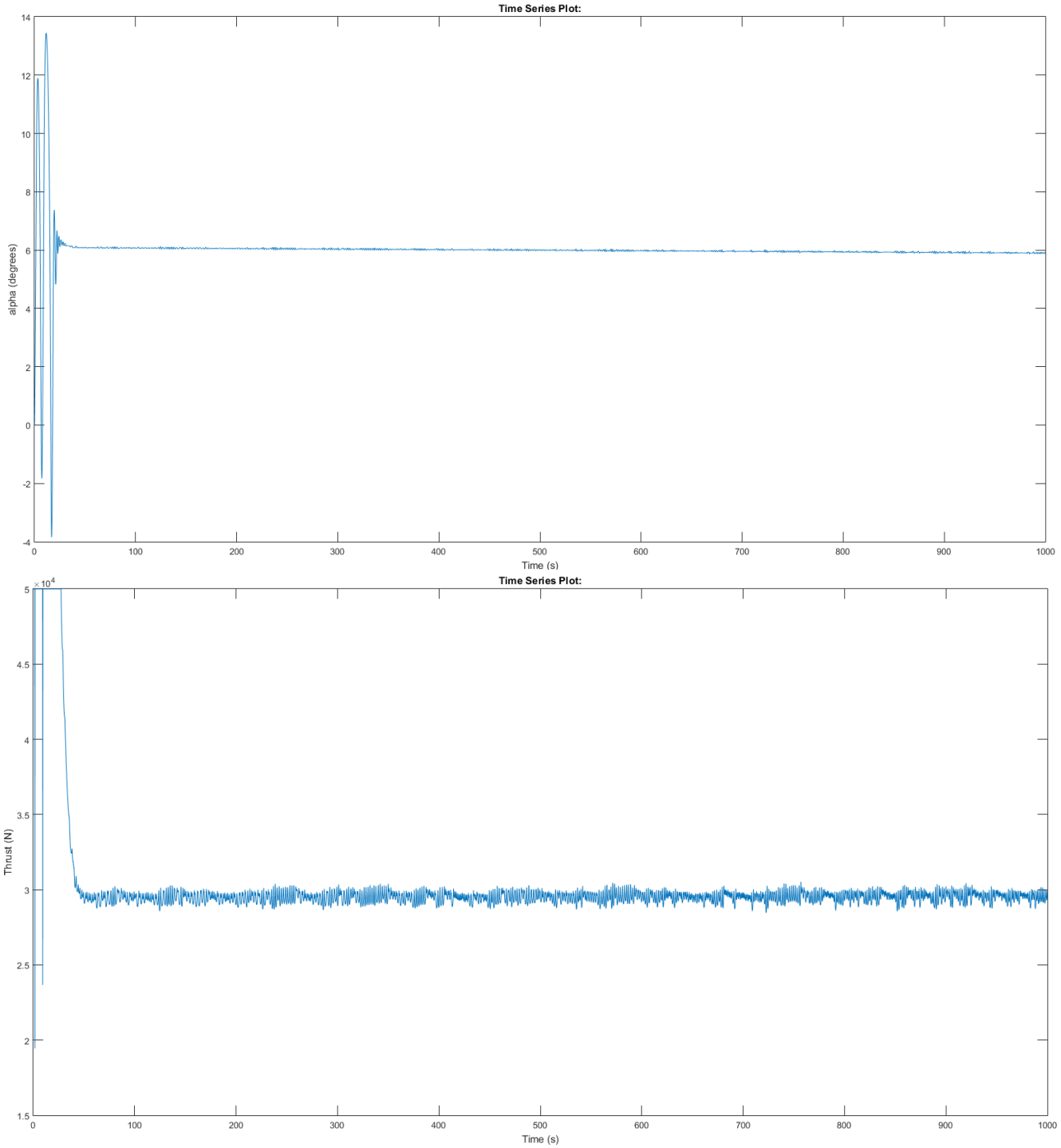

Figure 4.2: Angle of Attack (degrees) and thrust (N) of the controlled aircraft as described in the block diagram 3.3

the NN coefficients are able to simulate an accurate model of the aerodynamic coefficients variation. Notice however that the thrust reaches its saturation value of 50 kN before stabilizing.

4.2 Feedback Linearisation Controller

Once the model has been tested and validated, the described nonlinear inversion controller can be tested. Reference tracking will first be observed with simple constant reference values, without any guidance control law.

Reiterating the example from the previous section, the reference following for $V_a^d = 200ms^{-1}$, $\gamma^d = 0$ rad and $\psi^d = 0$ rad can be seen in figure 4.3.

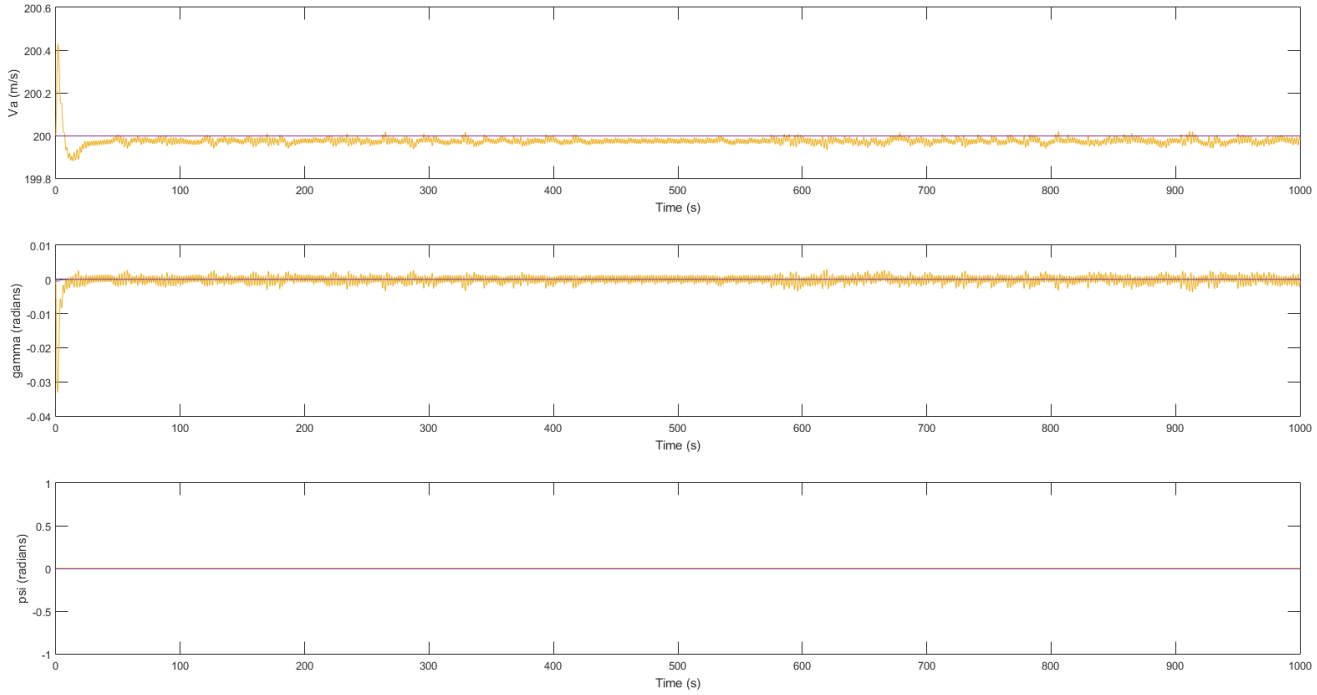

Figure 4.3: Reference (blue) following over $1000s$ of simulation time for $V_a^d = 200ms^{-1}$, $\psi^d = 0$ rad and $\gamma^d = 0$ rad respectively and measured values (yellow)

In figure 4.3 the reference is followed with minimal osculations for the three variables, although with the presence of some steady state error in the case of the airspeed V_a. The small initial convergence of γ^d and V_a^d can be explained by the simulation initial conditions. To maintain $V_a^d = 200ms^{-1}$ at $t = 0s$, as seen in table 4.1, the AoA must be $\alpha = 6^o$. As this simulation starts in cruise condition with the initial condition $\theta = 0$ rad, for this reason γ initially takes negative values before reaching and stabilizing at its desired reference. This also affects the airspeed of the aircraft, although only increasing the following error up to $0.4ms^{-1}$.

To be able to follow trajectories however, a good control of the aircraft heading will be necessary.

Proposing this time the following variation in ψ^d as a reference for the simulated aircraft

$$\psi^d = \begin{cases} 0 \text{ rad} & t < 100 \\ \dfrac{\pi}{2} \text{ rad} & 100 \leq t < 500 \\ 0 \text{ rad} & t > 500 \end{cases} \tag{4.5}$$

The comparison of the measured heading ψ and its desired value comes in figure 4.4.

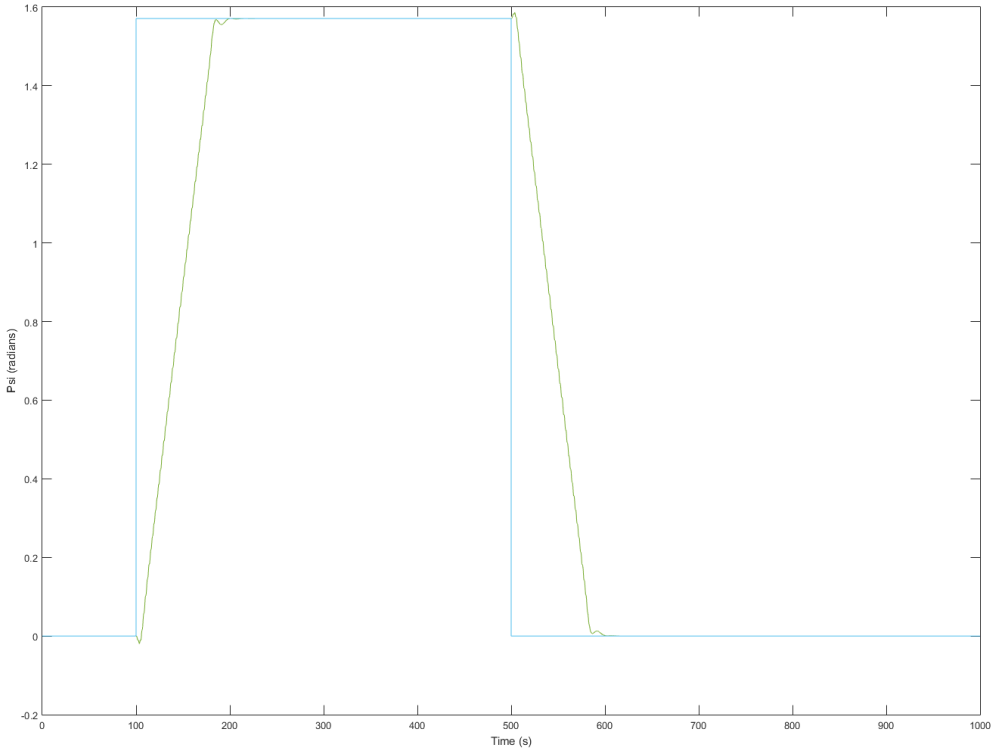

Figure 4.4: Desired (blue) and measured (green) heading

Although the reference value is reached after each step, the convergence takes around $100s$ for a $\dfrac{\pi}{2}$ rad perturbation to converge, without overshoot. These conversion times can be explained by the high inertia and mass of the commercial aircraft. The resulting airplane trajectory can be seen in figure 4.5.

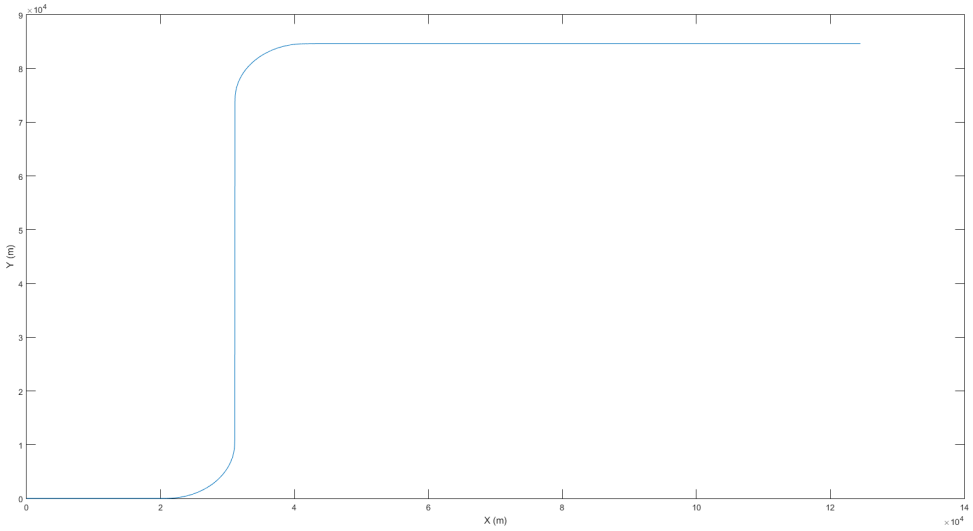

Figure 4.5: Plane trajectory for the nominal control input

From the results obtained so far, it can be concluded that the controller performs correctly following the desired reference. These results however were obtained assuming that the exact model of the aircraft was known, without any kind of external perturbation or simulated system failure. It is these perturbations that will be simulated in this section and studied. To do so the same testing reference as described in equation 4.5 will be used to compare the effects of different types of disturbances with the undisturbed system.

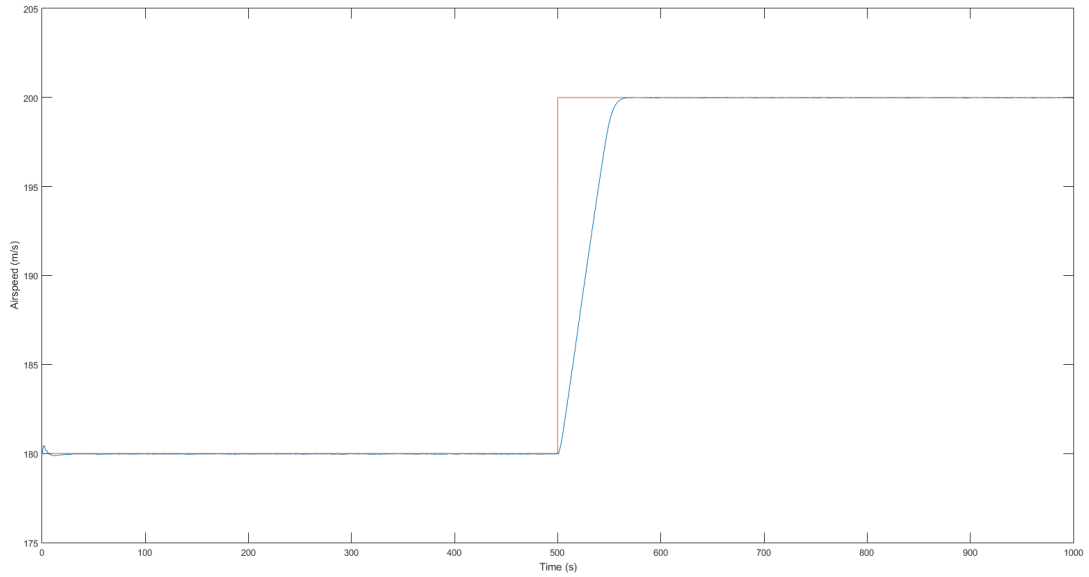

Figure 4.6: Desired (orange) and measured (blue) airspeed

In figure 4.6 a step input was given as an airspeed command, going from $180ms^{-1}$ to $200ms^{-1}$. As can be observed after a delay of approximately $60s$ the aircraft reaches and stabilizes at the new airspeed

reference.

4.3 Inversion Errors

4.3.1 Baseline Feedback Linearisation Controller

This error will be the most frequent when implementing such a controller, as some system parameters estimations may have considerable errors, namely the aircraft inertial matrix. Fortunately, the controller used is, as it will be demonstrated, tolerant to errors in the estimation of this parameter, but can result in an unstable system in extreme cases. To simulate estimation errors, the entries of the inertial matrix I_{xx},I_{yy},I_{zz} and I_{xz} were multiplied by a reducing factor $\zeta \in [0; 1]$ when computing the nonlinear inversion in 3.20. The inertial matrix used in this equation I_{est} is therefore given by $I_{est} = \zeta I$.

Changing the values of ζ to 0.5 and 0.05, increasing the inversion error, results in the following trajectories in figure 4.7

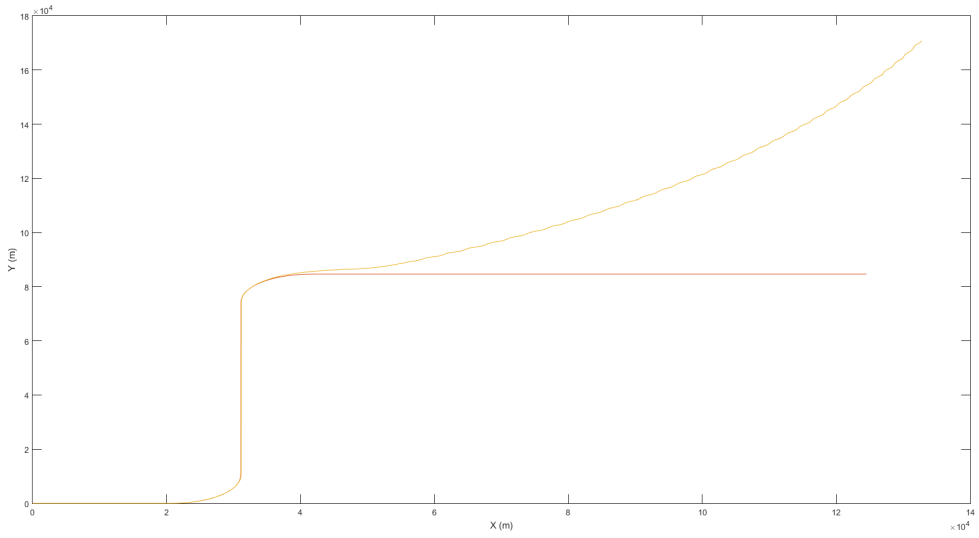

Figure 4.7: Plane trajectory for $\zeta = 0.5$ (red) and $\zeta = 0.05$ (yellow)

Indeed, a 50% error in estimating the inertia of the aircraft when computing the feedback linearisation law results in a negligible reference tracking error, as can be seen in figure 4.8. This however is not the case if ζ is reduced even further to $\zeta = 0.05$ and if the error of the system relative to its references becomes much greater (figure 4.9). As can be seen in figure 4.7, the aircraft is unable to follow yaw demands in this case as it goes unstable. This establishes a first goal to measure the performance of the designed neural network, to try reduce the error for $\zeta = 0.05$, ultimately achieving the performance of a nonlinear inversion with no inertia estimation errors.

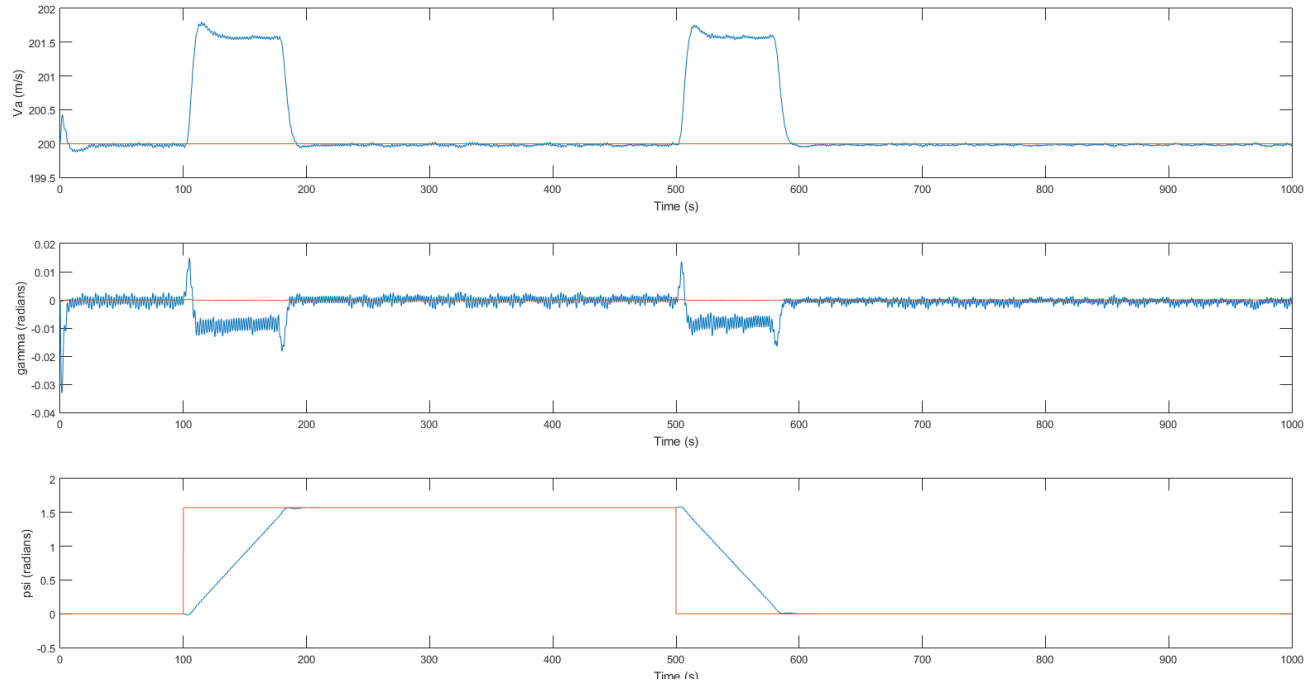

Figure 4.8: V_a, γ and ψ for measured and desired values for $\zeta = 0.5$

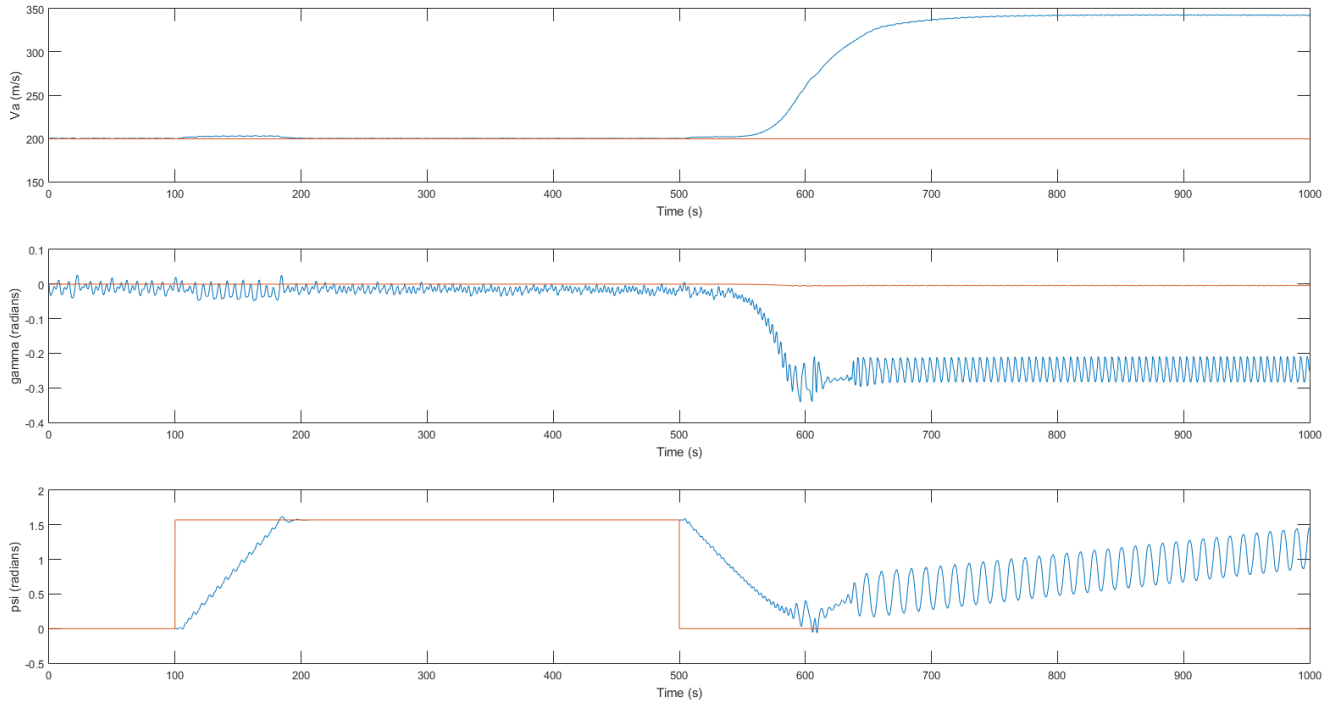

Figure 4.9: V_a, γ and ψ for measured and desired values for $\zeta = 0.05$

4.3.2 Neural Network Correction

The first adaptive capability of this method that will be described is the compensation of inversion error due to an incorrect estimation of the plane's inertia. Taking the previous example of 4.9, it can be observed that the aircraft becomes unstable at $t = 600s$, leading to a significant increase of airspeed from

the desired $200ms^{-1}$ to $350ms^{-1}$. The heading of the aircraft also becomes uncontrollable, oscillating and slowly increasing.

This is due to the Δ component of equation 3.33. At each time step the online Neural Network will converge to this value, compensating this error and therefore decreasing the reference following errors and preventing the aircraft from becoming unstable. Using the same conditions as used to obtain 4.9 comes that:

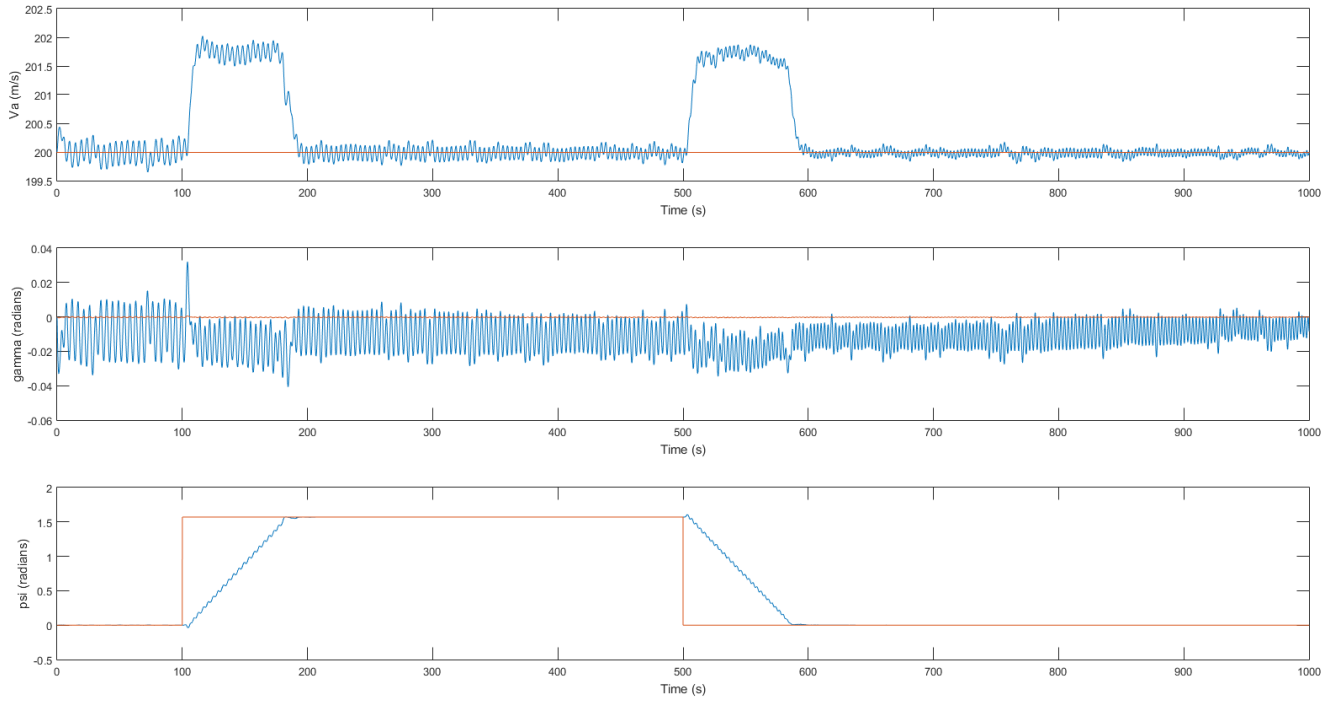

Figure 4.10: V_a, γ and ψ for measured and desired values for $\zeta = 0.05$ with NN correction

This first result shows for the adaptive control some improvements, mainly when comparing after $t = 600s$. Comparing the airspeed, the value V_a remains close to the desired value of $200ms^{-1}$ without reaching greater values, as observed previously. Regarding the yaw, besides an expected delay reaching the desired values, the system follows the reference with little to no error, this time without becoming unstable after $t = 600s$.

In order to more accurately visualise the influence of ζ on the NLI controller for this same case, several simulations were made for values of ζ from 0 to 1. For each simulation the mean error was computed, in order to obtain a plot of these errors for each of the three references. From figure 4.11 can be concluded that the error becomes considerable $\zeta < 0.05$ for a controller without neural network compensation. For a compensated adaptive controller however, it is noticeable that errors only become considerable at $\zeta < 0.02$.

Studying the mean errors for $\zeta > 1$, such an error represents a different situation. In this case, the estimated inertia of the aircraft is smaller than its real value. This can be caused by fuel consumption or, for the case of military aircraft, cargo release during flight. Reproducing the simulations from figure 4.11, with $\zeta = [1, 10]$, the results are obtained as in figure 4.12.

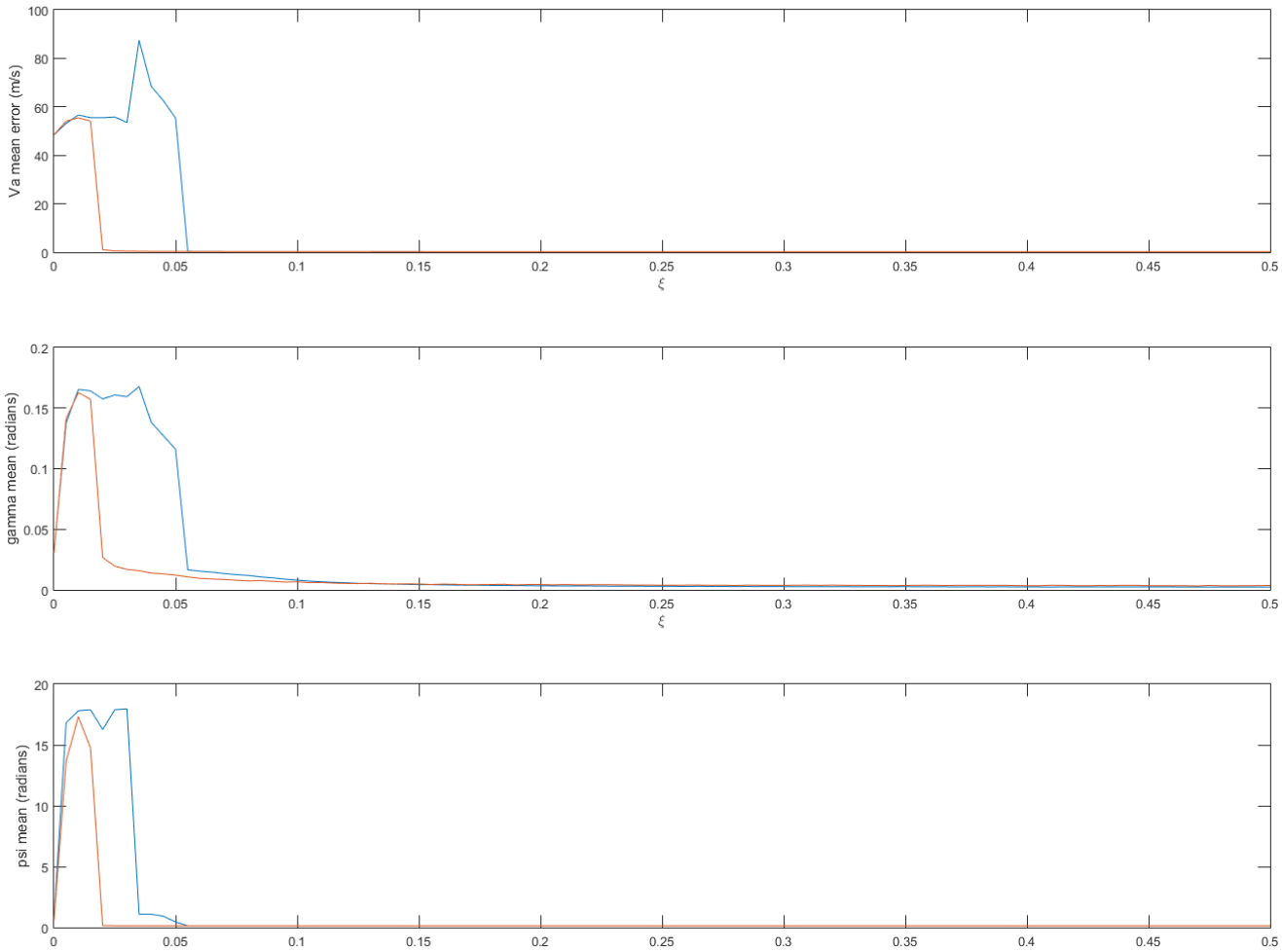

Figure 4.11: Mean errors for V_a, γ and *psi* for $\zeta = [0, 0.5]$ with neural network compensation (orange) and without compensation (blue)

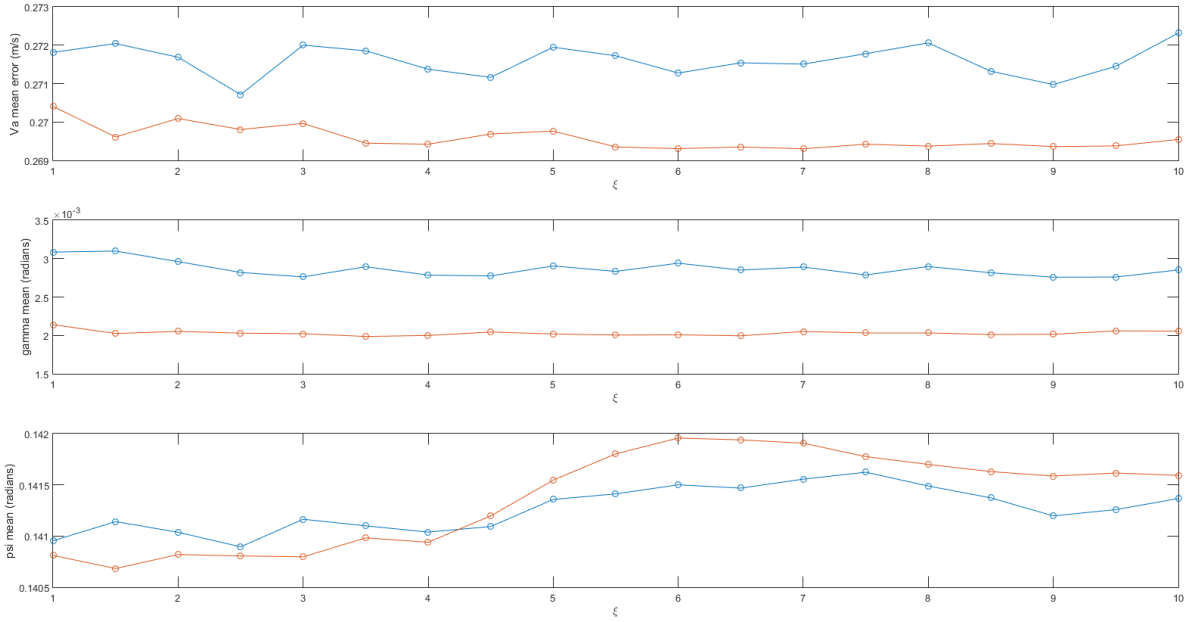

Figure 4.12: Mean errors for V_a, γ and *psi* for $\zeta = [1, 10]$ with neural network compensation (orange) and without compensation (blue)

Although the errors showed in figure 4.12 are small and almost neglectable for these values of ζ, these results still show, for γ and V_a, the NN is capable of reducing the mean error when compared to the non corrected controller. This one however is able to perform correctly with a small tracking error for larger values of ζ.

Although the network does seem to provide an adaptive component to the controller, note that it can only do so if the aircraft is operating within its limitation and the physically possible. Giving, for example, very low V_a commands that wont provide the necessary lift for the aircraft will result, as would be expected, in an uncontrollable aircraft. Indeed, as seen in table 4.1, a V_a value close to $200ms^{-1}$ is needed to maintain the necessary lift. The same also happens if, by simulating actuation failures, the values of $C_{\delta_{ail}}, C_{\delta_{ele}}, C_{\delta_{rud}}$ are too excessively reduced, or if the system is subjected to extreme icing and wind conditions that would render the aircraft unflyable. For these results the conditions were chosen so that while these would increase the command error of the baseline controller, the airplane would still be flyable allowing the network to improve its controllability and reduce this error.

The designed controller, as detailed in section 3.2, includes a linear controller for the system linearised by the nonlinear inversion, more precisely a PD controller for the plane's fast dynamics. A poorly tuned controller however will lead to an undesired behaviour. Using once again the condition used so far (see figure 4.8, with a perfect inertia estimation $\zeta = 1$ from a reduced controller, compared to the same controller corrected by the NN, comes that:

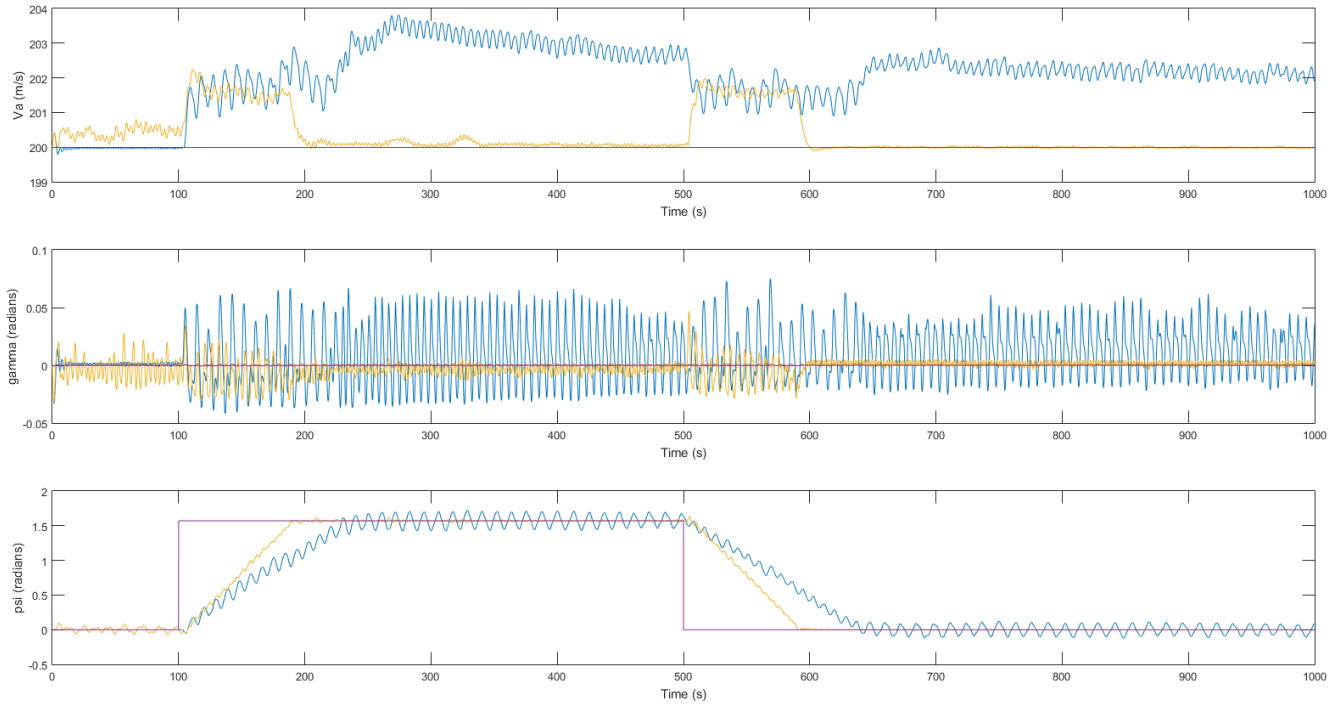

Figure 4.13: V_a, γ and ψ for measured and desired values. In purple can be seen the three references, in blue can be seen the measured values where in equation 3.21 $K_D = 0$ and $K_P = [1, 1, 1]^T$. In yellow the same controller is used, while applying NN adaptive corrections

The first observation that can be taken from figure 4.13 is that the difference the purple and yellow graphs is considerably smaller than that of the blue graph, particularly noticeable in the first V_a plot. In

the two last ones oscillations around the desired value are reduced, as is the yaw convergence time for the NN adaptive controller by $30s$. It is also curious to observe the oscillations of yellow neural network adapted plot decrease with time. This is explained as the weights of the network do not converge instantly to their optimal values.

4.4 System Failures

4.4.1 Baseline Feedback Linearisation Controller

One other cause for inversion errors that can have a great impact on flight trajectory are system failures. This subsection will focus on methods to simulate these failures and observe the behaviour of the airplane trajectory for these cases. The reference trajectory that will be used will be the same as used previously. The first failure to be simulated will be a control surfaces failure, that will lead to reduced controllability of the aircraft. In this work this was replicated in a simulated environment by reducing each moment coefficients for the elevator, aileron and rudder, namely $C_{\delta_{ele}}$, $C_{\delta_{ail}}$ and $C_{\delta_{rud}}$.

By setting each of these coefficients to 20% of their initial values, the effect of the errors in followed trajectory become visible in figure 4.14. This figure shows that the reduced controllability results in a $20km$ difference to that of the undisturbed system. This sets the second goal for the performance of the neural network, to reduce this error as much as possible. This is a preliminary analysis, as actuator failures can a much bigger impact on the aircraft flight and can even render it uncontrollable. For this simulation however, the goal was to simulate failures that would only reduce the controllability of the aircraft, to then use the NN to improve its behaviour.

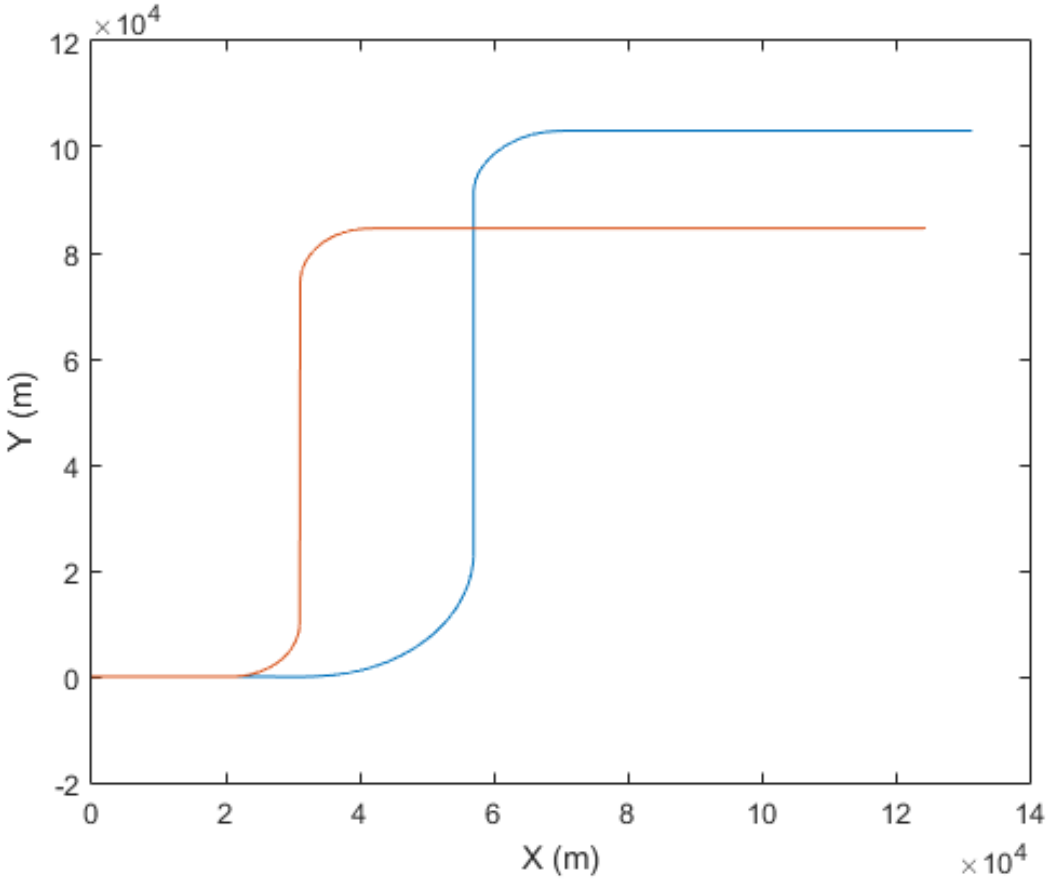

Figure 4.14: Trajectory with 20% reduced actuation (blue) and trajectory without failures (red)

Studying as previously the values of V_a, γ and ψ for the case of actuator failures, the following results are obtained

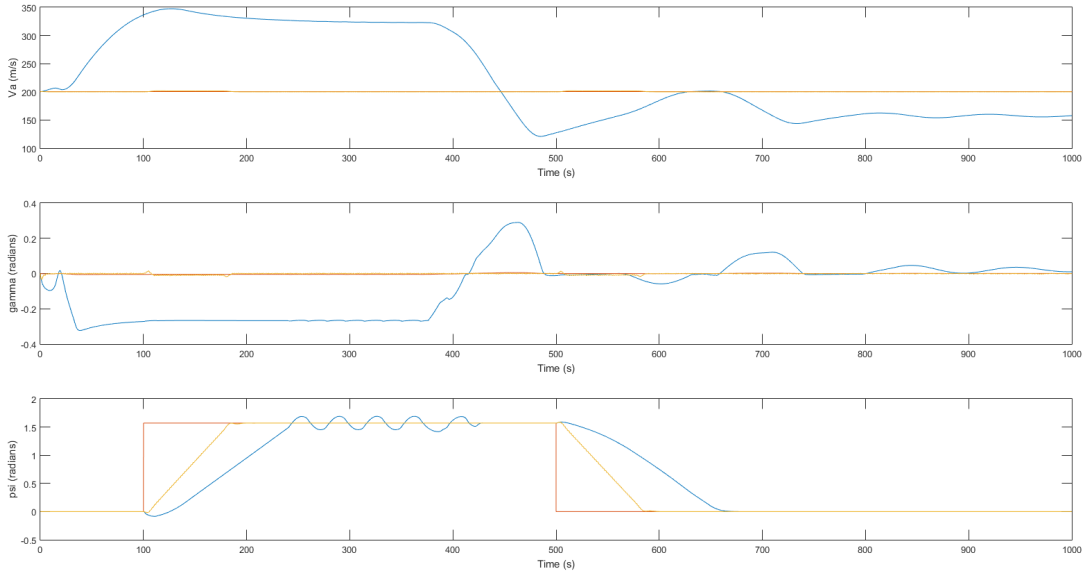

Figure 4.15: V_a, γ and ψ desired values (orange), measured without actuator failures (yellow) and with reduced actuation (blue)

From figure 4.15 the first conclusion that can be drawn is that the tracking error for the simulation with actuator failures is considerably greater than the simulation without failures for V_a and γ. V_a increases up to $150ms{-}1$ above its reference values, whereas γ initially decreases down to -14^o. Both of these values are unrealistic for such a commercial aircraft and are caused by the reduced control surface moments. The effect of these reduced moments is also noticeable for ψ, that in case of actuator failures takes twice as long, up to $160s$ to reach its reference value, up from $90s$ without these failures.

One second type of flight perturbation that will be included in this subsection is the effects of icing conditions in aircraft flight. According to the FAA guide to flight in icing conditions, ice contamination of an airfoil has an important influence on both the Lift and Drag curves [19]. Ways to simulate these perturbations in a modelled environment will be studied, and as for the other described disturbances an online neural network will be used to improve the control of the aircraft in such conditions. The main drawbacks that can be caused by icing are

- **Stall** As seen in figure 4.16, ice contamination of the airfoil leads to a significant reduction in the value of the maximum C_L, causing the aircraft to reach stall conditions at a much lower angle of attack. Reducing speed (e.g for an approach) makes this effect more noticeable for the pilot. The usual reduction on $C_{L_{max}}$ is of 30% [19].

- **Drag** Drag will increase directly with the amount of ice accumulated on the wing, reaching usual values of 100% of the initial drag coefficient.

- **Roll** Icing on the main wings will also affect roll control. As the thickness of the wing decreases towards its tip, so increases its efficiency to collect ice, leading to a partial stall in the tip of the wings, according to the AOPA manual [20].

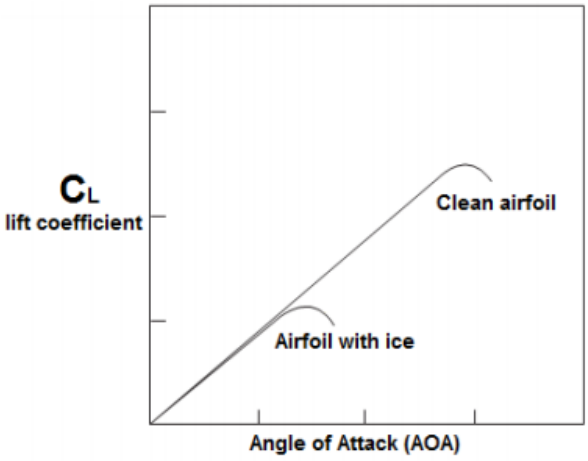

Figure 4.16: Effects of icing on lift coefficient [19]

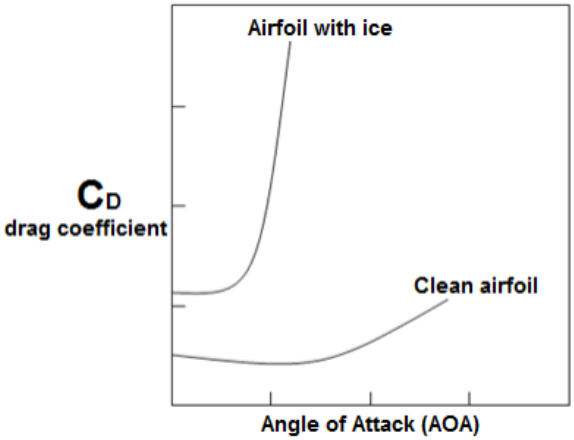

Figure 4.17: Effects of icing on drag coefficient [19]

Following these indications to simulate heavy icing condition, the lift coefficient $C_{L_{max}}$ was reduced by 30%, and the stall angle reduced from 15^o (see figure 4.1) resulting in figure 4.18. The value of the drag coefficient was also augmented by 200% from equation 3.18. Finally, in order to simulate roll control limitations, the value of $C_{\delta_{ail}}$ was reduced by 30%. These values were chosen in order to deteriorate the control of the aircraft, while allowing for the possibility of reference following without reaching actuator saturation (for both control surface deflections and thrust).

For this case a sinusoidal yaw reference was given of $\psi^d = \frac{\pi}{4}\sin(1.1459t)$ rad, t being the simulation time. A sinusoidal reference was used to make the effects of simulated icing more noticeable, and the same references of $V_a^d = 200ms^{-1}$ and $\gamma^d = 0$ rad as previously were used. For this case, the results obtained were as follows

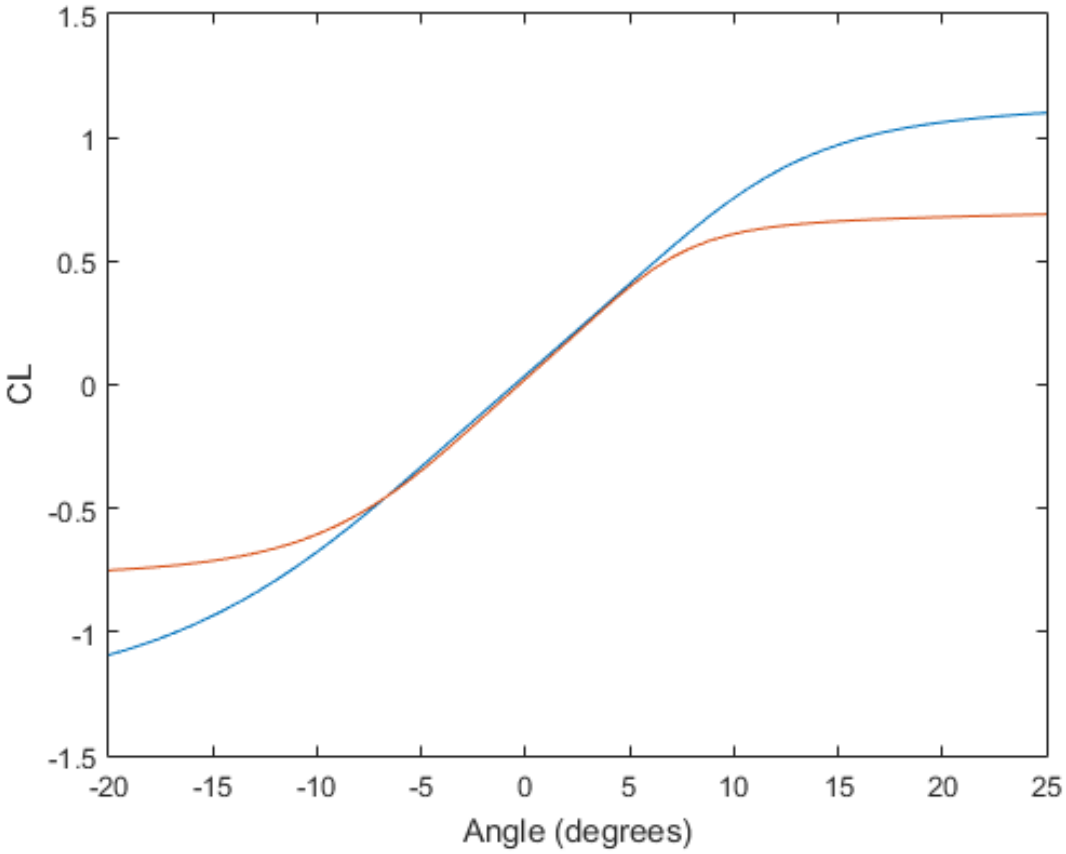

Figure 4.18: Lift coefficient simulation in icing conditions (orange) and in normal conditions (blue)

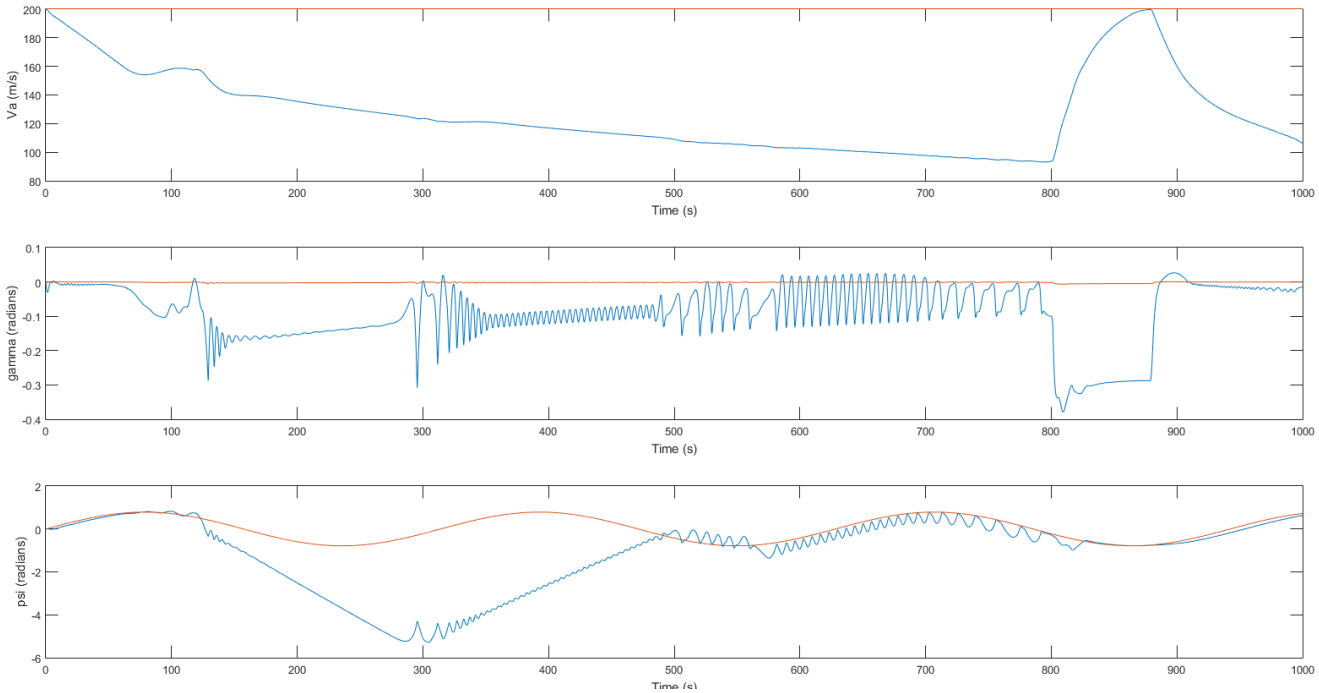

Figure 4.19: V_a, γ and ψ for measured and desired values in icing conditions

From figure 4.19, the system in such high drag conditions can be considered marginally stable, as it

easily increases the error to either of the three commanded inputs to significantly high values. For the case of the flight path angle, the aircraft even reaches under -20^{o} at $t = 600s$, a completely unrealistic value for a normal cruise flight. The impact of the increased drag in noticeable in the reduced speed in the simulation, going under $100ms^{-1}$ for the desired $V_a^d = 200ms^{-1}$. The reduced roll capabilities and lift also cause errors and oscillations in heading and flight path angle following.

These perturbations change the expected behaviour of the aircraft, causing the feedback linearisation control law to perform poorly. In the next section of this work the results of the online neural network will be shown, demonstrating its ability to increase the stability of the aircraft in situations of strong perturbations, or even in cases of a poorly designed linear or feedback linearisation controller. Note however that the conditions used here to induce the aircraft into unstable or marginally stable conditions mimic extreme perturbations, and the described controller can be said to perform well in cases when little to none of these perturbations are present.

4.4.2 Neural Network Correction

The next objective will be to observe if the designed controller is capable of adapting to system failures. Simulation of these failures was described in the previous subsection, where actuator failures and icing conditions were tested.

Actuator failure

In the same conditions as previously, the aerodynamic coefficients of the control surfaces were reduced to 20% of their initial values. Applying the adaptive correction through the online neural network, the following trajectories were obtained

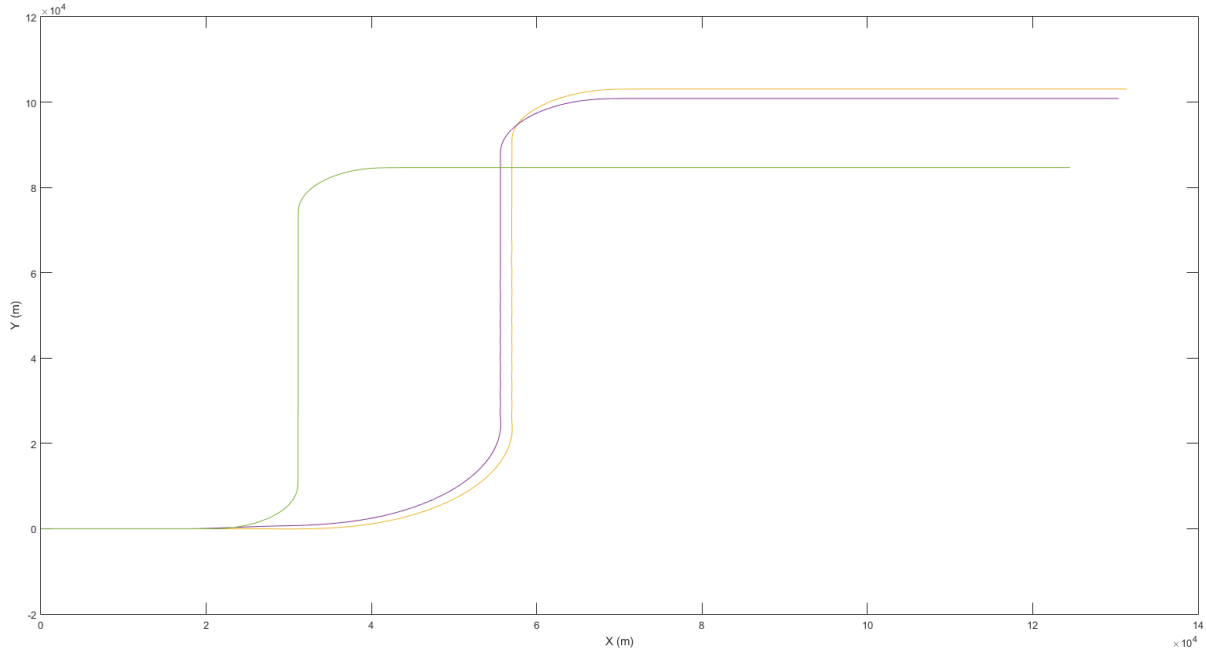

Figure 4.20: Trajectory with 20% reduced actuation (yellow) and trajectory without failures (green). The corrected trajectory by the NN can be seen in purple

The first observation that can be drawn from figure 4.20 is that such an actuator failure severely reduces the controllability of the aircraft.

The main consequence for this case will be, as could be expected, an increase in the convergence time to the desired heading. Note that the desired trajectory is not followed as this correction is only applied to the fast dynamics, and a guidance control law is not yet implemented. Comparing the two cases where the failures were implemented however, it can be observed that system corrected by the neural network converged to the desired heading slightly faster, resulting in a $2000m$ reduction in distance when comparing the non corrected control trajectory.

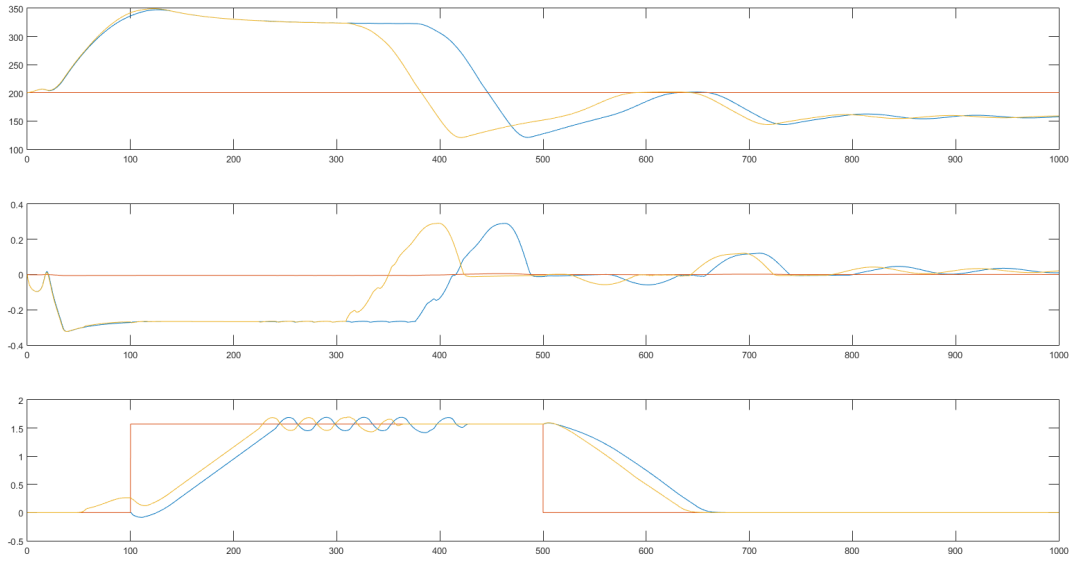

Figure 4.21: V_a, γ and ψ desired values (red), measured with actuator failures and NN correction (yellow) and without NN correction (blue)

Observing this system with the NN corrections, the error with such failures is still just as large as without NN correction, although the network seems to have a lead to the non corrected simulation, causing it to reach stable values faster and, for the case of ψ, to have a shorter delay reaching the desired heading.

Icing conditions

Once again retaking the example from figure 4.19, in the same icing conditions, the online network correction was applied, resulting in the following results

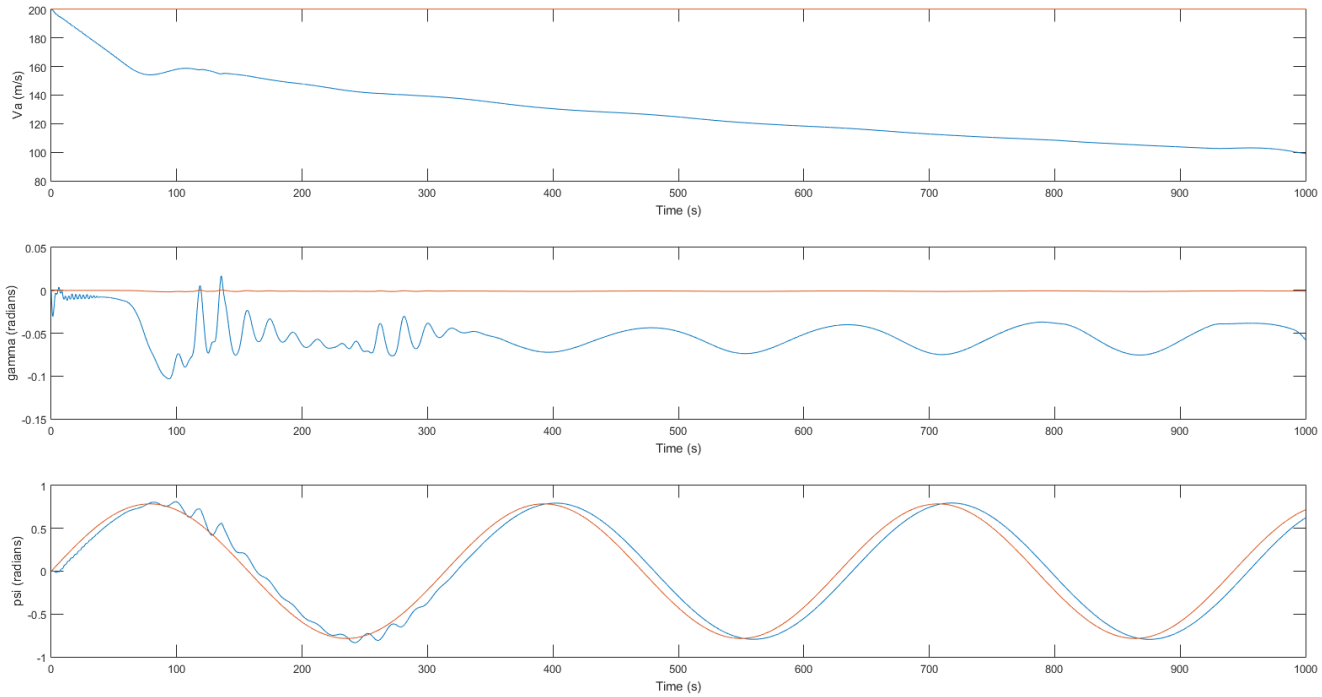

Figure 4.22: V_a, γ and ψ for measured and desired values in icing conditions, with NN correction

Again, these are conditions that have a negative impact on the controllability of the aircraft: the increase of 200% of the value of C_D causes a decrease in the speed of the aircraft, rendering it unable to follow the desired speed of $200ms^{-1}$. Regarding the measured flight path angle γ, although some oscillations are still present, error relative to the reference no longer goes up to 0.3 rad, instead oscillating around 0.05 rad at a much lower frequency and amplitude. The most noticeable improvement however can be seen in the heading following: the aircraft follows the desired ψ^d during the full simulation, with little to no oscillations around that value. It is also interesting to notice that in the first moments of the learning process, up to $t = 300s$, oscillations around the desired heading value are greater than in later moments of the simulation. This is explained by the fact that the weights of the neural network have not yet converged to their optimal values at $t < 300s$.

4.5 Guidance Controller

This final section will focus on the results of the implementation of the guidance controller described in equations 3.30. From the work implemented and results obtained thus far, a NN adaptive controller can now be used to follow references of airspeed, heading and flight path angle. From the equations 3.30 position references can now be used to compute V_a, γ, ψ. Choosing a path to follow using these equations the following results were obtained, showing the aircraft is able to correctly follow a trajectory from a constant feed of way points over time.

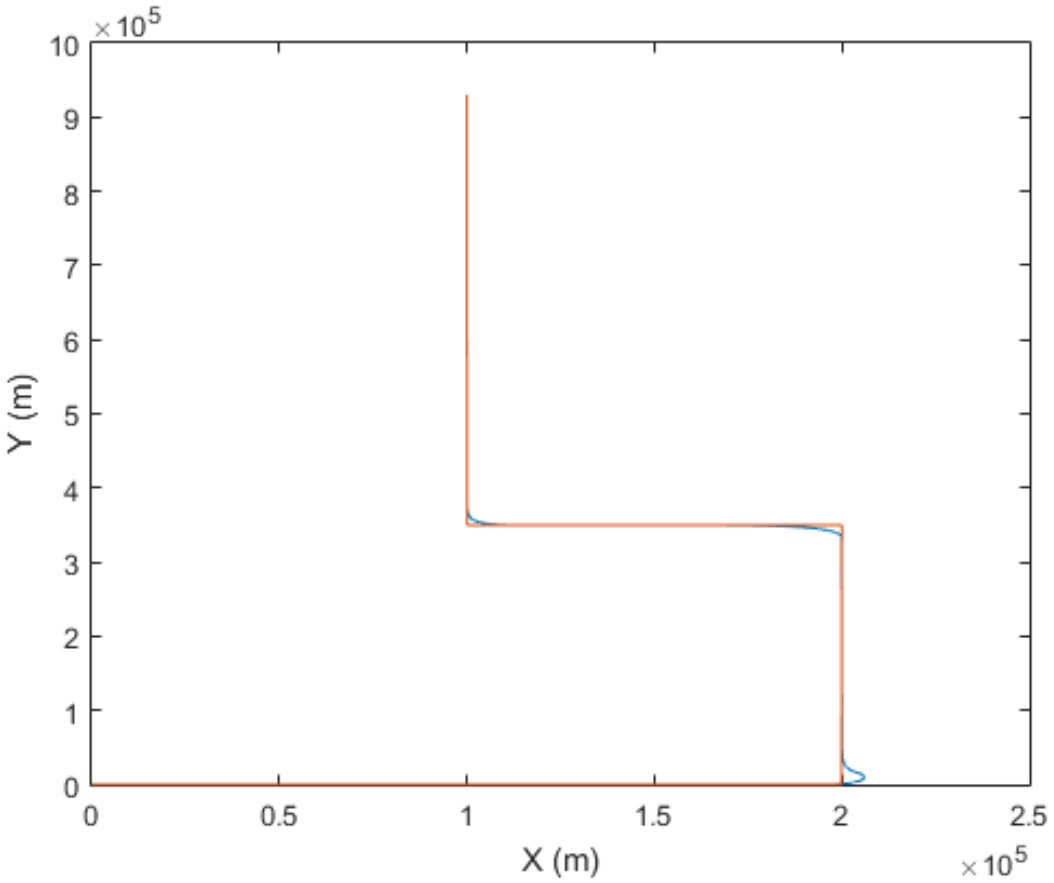

Figure 4.23: Trajectory reference (orange) and obtained trajectory (blue)

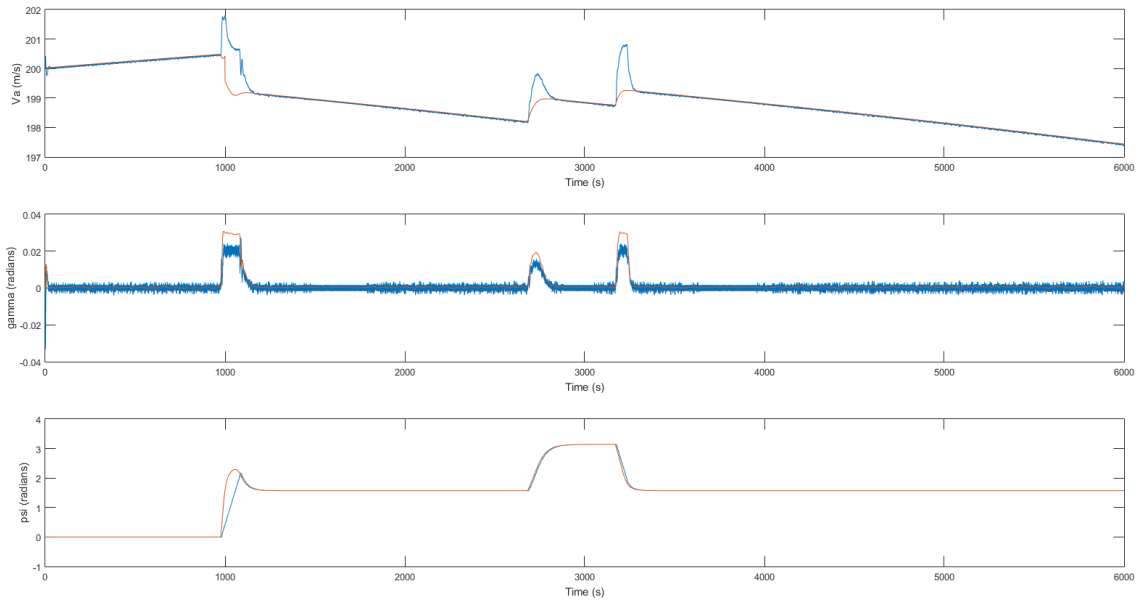

Figure 4.24: V_a, γ and ψ for measured (blue) and desired values (orange) obtained from the guidance controller

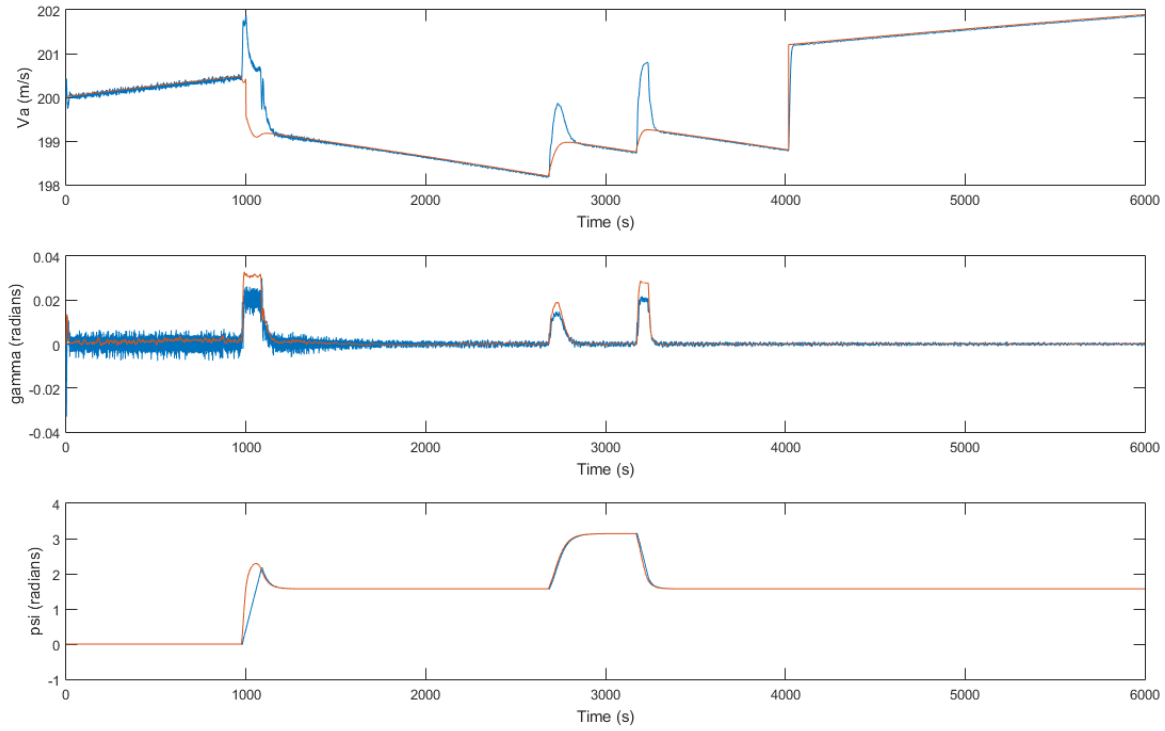

Figure 4.25: V_a, γ and ψ for measured (blue) and desired values (orange) obtained from the guidance controller using NN correction

Figure 4.23 shows the trajectory of the aircraft for a given trajectory, showing slight errors on sudden heading changes. Analysing figure 4.24, for a non corrected system, although a small overshoot in airspeed following, the three variable otherwise follow their references with minimal error. The γ graph however shows that this variable oscillates around 0 rad.

Comparing these results with the NN corrected controller however, although no disturbance was added to the simulation, the flight path angle oscillations are greatly reduced. It is also interesting to notice that while the network is still converging its weights to reach their optimal values for the first $1000s$, these oscillations are greater than in the $4000s$.

Chapter 5

Conclusions

This final chapter will list the conclusions that can be drawn from chapter 4. Reiterating the goals initially established for this work, these were the following

- **Design and validate a plane model**
- **Implement and test a feedback linearisation controller**
- **Adaptively improve control with an Online Neural Network**

These goals and the results obtained will be discussed, to finally conclude on the successfulness of the work made.

5.1 Feedback Linearisation Controller

The first step, before designing and implementing a controller, was to establish the model of a commercial airplane in a numeric simulation environment. The Boeing 737 as used, as information on this aircraft was readily available, simulated in Matlab Simulink. Note that for this work only cruise flight conditions were considered, as the chosen aircraft model was only validated in such conditions. The feedback linearisation controller was also designed having this assumption in mind.

Assuming cruise conditions, in section 4.1 it was demonstrated the airplane model followed the expected theoretical results. This was made using the FBL controller to make the simulation follow cruise conditions at the required speed. Once this first goal was achieved, it allowed to build the rest of the controller structure around it.

The nonlinear control law for fast dynamics was then submitted to different types of perturbations and inversion errors. The controller showed to be robust to not only to errors in inertia estimation as well as small system failures. For more serious control perturbations however, the aircraft could not, as would be expected, follow the desired inputs. Using a 99.5% smaller inertia relative to its true value in the NLI algorithm, increasing by 200% the drag coefficient or by heavily reducing the ability of the control surfaces to influence the plane's dynamics, the aircraft's behaviour showed much higher reference tracking

errors, sometimes even becoming uncontrollable. Concluding this first section the designed controller, using a model based approach, proved to be robust to most external perturbations and internal errors.

5.2 Online Neural Network

Once the limitations of the system were known, an online neural network was added to adaptively reduce these limitations for a wide range of perturbations (e.g. actuator failure, icing conditions, etc.). The network was at no point trained previously to the simulation.

Taking firstly the errors caused by errors in parameter estimations and gain tuning (for the linear law controlling the aircraft's model linearised by the FBL), the network showed improvements in robustness and reduced errors in airspeed and heading following, when compared to the same controller without the network. Similar tests were made with reduced control from actuators and in icing conditions. For the actuator failure case although the network slightly improve heading convergence times, reducing $C_{\delta_{ail}}, C_{\delta_{ele}}, C_{\delta_{rud}}$ by 80% is still a too big perturbation for a commercial aircraft to recover from. For icing condition were simulated reduced lift coefficient, increased drag and reduced roll control ($C_{\delta_{ail}}$ was reduced by 30%). For this case however, while the non corrected was unable to follow a heading and flight path angle references, this was not the case once the online neural network was added to the system. Indeed the network allowed the aircraft to follow a sinusoidal heading reference and to reduce the difference between γ and its desired value.

The same network architecture was used for all cases described above. Taking this into account it can be concluded that the goal of designing a neural network that would make the original NLI law more robust and able to adapt to different perturbations was indeed achieved.

5.3 Future Work

In this work, the neural network used had a relatively simple architecture of one hidden layer, using both sigmoid and linear activation functions. One way to continue this work is to try different architectures that have been used in other works, such as recurrent neural networks, Radial Basis Function neurons or larger numbers of hidden layer, to improve upon its adaptive capabilities.

This method was also only used to correct the control of the fast dynamics of a commercial aircraft, and a similar system could be used to adaptively improve its guidance laws. The results obtained however allow to make a commercial aircraft follow 4D trajectories, even in case of external disturbances or system failures. The next step is now to use these trajectories to optimise air traffic management, allowing even several commercial aircraft to travel along these trajectories, reducing the work load of air traffic controllers while increasing the capacity of a given air control zone.

Bibliography

[1] SANTOSO et al. State-of-the-art intelligent flight control systems in unmanned aerial vehicles. *IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING*, 2017.

[2] A. Zulu and S. John. A review of control algorithms for autonomous quadrotors. *Open Journal of Applied Sciences*, 2014.

[3] A. K. Shastry et al. Neuro-adaptive augmented dynamic inversion controller for quadrotor. *IFAQ Papers Online*, 2016.

[4] M. Perhinschi et al. Augmentation of a non linear dynamic inversion scheme within the nasa ifcs f-15 wvu simulator. *Proceedings of the American Control Conference*, 2013.

[5] T. Xiang et al. Adaptive flight control for quadrotor uavs with dynamic inversion and neural networks. *IEEE lntemational Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2016.

[6] J.-J. E. Slotine and W. Li. *Applied nonlinear control*. 1991.

[7] Y. Ning, W. Liao-ni, and L. Qi. Adaptive flight control law based on neural networks and dynamic inversion for unmanned aerial vehicle. *International Conference on Automatic Control and Artificial Intelligence*, 2012.

[8] M. L. Steinberg. Comparison of intelligent, adaptative and nonlinear flight control laws. *Journal of Guidance, Control and Dynamics*, 2001.

[9] J. Vieira, F. M. Dias, and A. Mota. Neuro-fuzzy systems: A survey. *Proc. 4th WSEAS Int. Conf. Neural Netw*, 2004.

[10] Y. Tang and R. J. Patton. Reconfigurable flight control using feedback linearization with online neural network adaption. *Conference on Control and Fault-Tolerant Systems*, 2013.

[11] T. Xiang, F. Jiang, Q. Hao, and W. Cong. Adaptive flight control for quadrotor uavs with dynamic inversion and neural networks. In *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 174–179, Sept 2016. doi: 10.1109/MFI.2016.7849485.

[12] Q. Lin, Z. Cai, Y. Wang, J. Yang, and L. Chen. Adaptive flight control design for quadrotor uav based on dynamic inversion and neural networks. In *2013 Third International Conference*

on Instrumentation, Measurement, Computer, Communication and Control, pages 1461–1466, Sept 2013. doi: 10.1109/IMCCC.2013.326.

[13] H. E. Nuñez and F. M. Camino. Towards 4d trajectory tracking for transport aircraft. *IFAC World Congress*, 2017.

[14] B. Etkin and L. D. Reid. *Dynamics of Flight Stability and Control*. 1996.

[15] A. K. Shastry, A. Pattanaik, and M. Kothari. Neuro-adaptive augmented dynamic inversion controller for quadrotors. *IFAC-PapersOnLine*, 2016.

[16] X. Liu, Y. Wu, J. Shi, and W. Zhang. Adaptive fault-tolerant flight control system design using neural networks. *IEEE International Conference on Industrial Technology*, 2008.

[17] Lili and Z. hua min. Neural network based adaptive dynamic inversion flight control system design. *IEEE International Conference on Intelligent Control and Information Processing*, 2011.

[18] R. May, G. Dandy, and H. Maier. *Review of Input Variable Selection Methods for Artificial Neural Networks, Artificial Neural Networks - Methodological Advances and Biomedical Applications*. 2011.

[19] *Pilot Guide: Flight in Icing Conditions*. Federal Aviation Administration, 2015.

[20] *Safety Advisory - Aircraft Icing*. AOPA.

[21] F.-A. Buţu and R. Lungu. Adaptive flight control for a launch vehicle based on the concept of dynamic inversion. *20th International Conference on System Theory, Control and Computing (ICSTCC)*, 2016.

[22] J. Campos, F. L. Lewis, and R. Selmic. Backlash compensation in discrete time nonlinear systems using dynamic inversion by neural networks. *Proceedings of the 2000 IEEE lntemational Conference on Robotics and Automation*, 2000.