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

Dedicated to someone special...

Acknowledgments

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

Resumo

Inserir o resumo em Português aqui com o máximo de 250 palavras e acompanhado de 4 a 6 palavras-chave...

Palavras-chave: palavra-chave1, palavra-chave2,...

Abstract

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 often reveal to be inaccurate. Feedback linearization is an approach to non-linear control design that algebraically transforms a non-linear system onto a linear one. From the linearised system linear control techniques can be used. However this approach, as well as other state-of-the art control systems 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 a model leads to errors in the calculation of the model inversion, necessary to implement feedback linearization. Indeed, while feedback linearization control shows good tracking, it has poor disturbance rejection, that ultimately lead to these inversion errors [2]. 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 this error[3], [4], [5]. This thesis therefore aims to discuss and implement the existing alternatives to augment the precision, robustness and overall performance of feedback linearization control.

Keywords: non-linear control, feedback linearization, neural network, flight control

Contents

Acknowledgments . . . v
Resumo . . . vii
Abstract . . . ix
List of Tables . . . xiii
List of Figures . . . xv
Nomenclature . . . 1
Glossary . . . 1

1 Introduction 1
1.1 Motivation . . . 1
1.2 Topic Overview . . . 1
1.3 Objectives . . . 1
1.4 Thesis Outline . . . 1

2 Background 3
2.1 Airplane Model . . . 3
2.1.1 Frames of reference . . . 3
2.1.2 Fast Dynamics . . . 4
2.2 Translation Dynamics . . . 5
2.2.1 Actuator Dynamics . . . 6
2.3 Feedback linearisation . . . 6
2.3.1 SISO systems . . . 7
2.3.2 MIMO systems . . . 7
2.4 Limitations of feedback linearisation . . . 8
2.4.1 Fuzzy Logic Systems . . . 9
2.4.2 Neural Networks . . . 9

3 Implementation 13
3.1 Plane Model . . . 13
3.1.1 Plane Dynamics . . . 14
3.2 Controller Implementation . . . 14
3.3 Neural Network . . . 18

3.3.1 Network Architecture . . . 18
3.3.2 Training Algorithm . . . 19

4 Results **21**
4.1 Model Validation . . . 21
4.2 Feedback linearisation controller . . . 22
4.3 Disturbances and Errors . . . 24
4.3.1 Inversion errors . . . 24
4.3.2 System failures . . . 25
4.3.3 Wind disturbances . . . 25
4.4 Neural Network . . . 25
4.5 Guidance controller . . . 25

5 Conclusions **29**
5.1 Achievements . . . 29
5.2 Future Work . . . 29

Bibliography **31**

A Vector calculus **33**
A.1 Vector identities . . . 33

List of Tables

3.1 Boeing 737-2 parameters 13

4.1 Required cruise conditions for different values of α 22

List of Figures

2.1 Feedback linearisation example 7

3.1 Plane dynamics simulator diagram 15
3.2 Aircraft frames of reference 17
3.3 Neural Network diagram 19
3.4 Sigmoid Function 20

4.1 C_L versus α graph 22
4.2 AoA and thrust validation 23
4.3 Constant reference following of feedback linearisation controller 23
4.4 Desired and measured heading 24
4.5 Plane trajectory 25
4.6 Plane trajectory with inertia estimation errors 26
4.7 Reference tracking for $\zeta = 0.5$ 26
4.8 Reference tracking for $\zeta = 0.05$ 27
4.9 Reference tracking for $\zeta = 0.05$ 27

Chapter 1

Introduction

Insert your chapter material here...

1.1 Motivation

Define: usefulness of NLI vs robust control and problems of NLI

1.2 Topic Overview

Brief description of the solutions to be studied and how they will be used to improve control

1.3 Objectives

Goals: create systems able to adapt to changes of the airplane's dynamic

1.4 Thesis Outline

Background includes in this order: NLI theory, NLI problematic, neural network theory Implementation: ? Results: ???

Chapter 2

Background

In this chapter the aircraft model used for this work 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 Airplane 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 [6]. The model used in this work of a six degree of freedom transport aircraft will be described in this section.

2.1.1 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's 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 to 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 the wind frame to the body frame by

$$R_{BW} = \begin{bmatrix} c_\alpha c_\beta & -c_\alpha s_\beta & -s_\alpha \\ s_\beta & c_\beta & 0 \\ s_\alpha c_\beta & -s_\alpha s_\beta & c_\alpha \end{bmatrix} \tag{2.1}$$

To describe the attitude of the plane Euler angles , roll, pitch and yaw, 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

$$R_{EB} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta c_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{2.2}$$

2.1.2 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

$$\begin{bmatrix} L' \\ M \\ N \end{bmatrix} = \frac{1}{2} \rho S V_a^2 \left(\begin{bmatrix} bC_l \\ \bar{c} C_m \\ bC_n \end{bmatrix} + C_\delta \delta \right) \tag{2.3}$$

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

$$C_\delta = \begin{bmatrix} bC_{l\delta_{ail}} & 0 & bC_{l\delta_{rud}} \\ 0 & \bar{c} C_{m\delta_{ele}} & 0 \\ bC_{n\delta_{ail}} & 0 & bC_{n\delta_{rud}} \end{bmatrix} \tag{2.4}$$

$$\begin{bmatrix} C_l \\ C_m \\ C_n \end{bmatrix} = \begin{bmatrix} C_{l\beta}\beta + C_{l_p} p \frac{b}{2V_a} + C_{l_r} r \frac{b}{2V_a} \\ C_{m_0} + C_{m_\alpha}\alpha + C_{m_q} q \frac{\bar{c}}{2V_a} \\ C_{n\beta}\beta + C_{n_p} p \frac{b}{2V_a} + C_{n_r} r \frac{b}{2V_a} \end{bmatrix} \tag{2.5}$$

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 of the coefficients of equation 2.5 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 [6], I being the aircraft inertial matrix.

$$\dot{\Omega} = I^{-1} M_{ext} - I^{-1} \Omega \times (I\Omega) \tag{2.6a}$$

$$\dot{\Omega} = \frac{1}{2} \rho S I^{-1} V_a^2 \left(\begin{bmatrix} bC_l \\ \bar{c} C_m \\ bC_n \end{bmatrix} + C_\delta \delta \right) - I^{-1} \Omega \times (I\Omega) \tag{2.6b}$$

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 [7]

$$\vec{V_G} = \vec{V} + \vec{W} = \begin{bmatrix} V_a c_\alpha c_\beta + V_{w_x} \\ V_a s_\beta + V_{w_y} \\ V_a s_\alpha c_\beta + V_{w_z} \end{bmatrix} \tag{2.7}$$

and α and β can be computed by

$$\alpha = arctan\left(\frac{w}{u}\right) \tag{2.8a}$$

$$\beta = arctan\left(\frac{v}{V_a}\right) \tag{2.8b}$$

From these three equations, differentiating 2.8a and 2.8b comes that

$$\begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} H_{11} & 1 & H_{13} \\ H_{21} & 0 & H_{23} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \tag{2.9}$$

SEE ANNEX HERE

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 on a Matlab simulation, and is given by

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & tg_\theta s_\phi & tg_\theta c_\phi \\ 0 & c_\phi & -s_\phi \\ 0 & \frac{s_\phi}{c_\theta} & \frac{c_\phi}{c_\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.10}$$

2.2 Translation Dynamics

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

$$\begin{bmatrix} D \\ Y \\ L \end{bmatrix} = \frac{1}{2}\rho S V_a^2 \begin{bmatrix} C_D \\ C_Y \\ C_L \end{bmatrix} \tag{2.11}$$

Once again, the method used to compute these coefficients will be given in the chapter 3 in detail. These coefficients, similarly to the moment coefficient, 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 on 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's forces can be obtained.

$$\begin{bmatrix} F_{xa} \\ F_{ya} \\ F_{za} \end{bmatrix} = R_{WB} \begin{bmatrix} -D \\ Y \\ -L \end{bmatrix} \tag{2.12}$$

From Newton's Second Law comes the aircraft's acceleration

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} \frac{1}{m}(F_{xa} + T) - gs_\theta + rv - qw \\ \frac{1}{m}F_{ya} + gc_\theta s_\phi + pw - ru \\ \frac{1}{m}F_{za} + gc_\theta c_\phi + qu - pv \end{bmatrix} \tag{2.13}$$

An expression in the Earth frame can also be obtained

$$\begin{bmatrix} \ddot{x_E} \\ \ddot{y_E} \\ \ddot{z_E} \end{bmatrix} = \frac{1}{m} R_{BE} \begin{bmatrix} F_{xa} + T \\ F_{ya} \\ F_{za} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{2.14}$$

2.2.1 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 [6]. For the control surfaces δ_i, given a desired δ_i^d comes

$$\dot{\delta_i} = \frac{1}{\xi_i}(\delta_i^d - \delta_i) \tag{2.15}$$

Similarly for thrust

$$\dot{T} = \frac{1}{\xi_T}(T^d - T) \tag{2.16}$$

ξ_i and ξ_T being time constants. As the responsiveness of the resultant thrust will be much slower than that of the control surfaces, $\xi_T >> \xi_i$.

2.3 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 [8]. Given a nonlinear system

$$\dot{x} = f(x, u) \tag{2.17}$$

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

EXPLAIN NECESSARY MATHEMATICAL CONCEPTS HERE

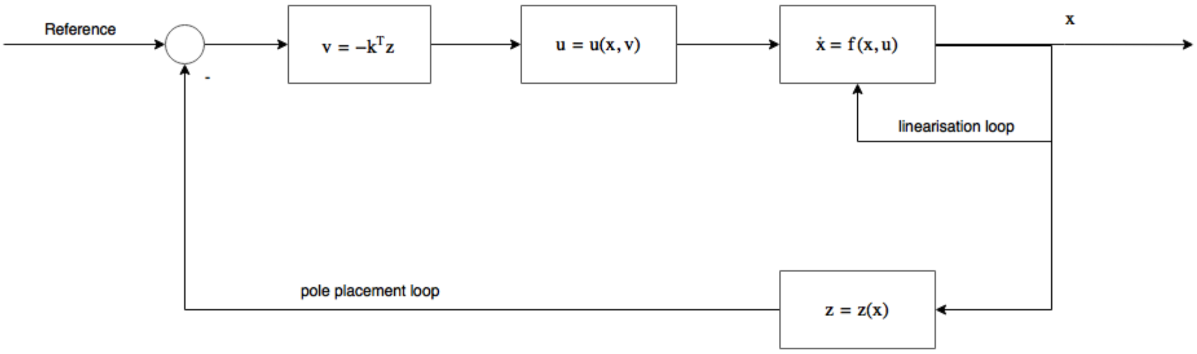

Figure 2.1: Feedback linearisation example [8]

2.3.1 SISO systems

Although an aircraft is always considered as a MIMO system, a description of this control concept will firstly be provided for a general SISO case, before generalising to a MIMO case. Nonlinear systems that can be represented as $\dot{x} = f(x) + g(x)u$ will be discussed in this section. From [8], a definition of Input-State linearisation is to find, for a nonlinear system of relative degree n, a *diffeomorphism* $\phi : \Omega \rightarrow R^n$ and nonlinear control law

$$u = \alpha(x) + \beta(x)v \tag{2.18}$$

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

$$\begin{cases} \dot{z_i} = z_{i+1} & if \quad i < n-1 \\ \dot{z_n} = v & if \quad i = n-1 \end{cases} \tag{2.19}$$

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

$$\nabla z_1 ad_f^i g = 0 \qquad i = 0, ..., n-2 \tag{2.20a}$$

$$\nabla z_1 ad_f^{n-1} g \neq 0 \tag{2.20b}$$

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

$$\alpha(x) = -\frac{L_f^n z_1}{L_g L_f^{n-1} z_1} \tag{2.21a}$$

$$\beta(x) = \frac{1}{L_g L_f^{n-1} z_1} \tag{2.21b}$$

2.3.2 MIMO systems

These concepts can also be extended to to MIMO systems in a similar manner, by similarly 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 $r < n$, n being the order of the system. Given a MIMO system

$$\dot{x} = f(x) + G(x)u \tag{2.22}$$

$$y = h(x) \tag{2.23}$$

the ϕ functions are defined as $\phi_j^i(x) = L_f^{j-1} h_i(x)$ and satisfy a condition similar to 2.19

$$\dot{\phi}_1^i(x) = \phi_2^i, ..., \dot{\phi}_{r_i-1}^i = \phi_{r_i}^i \quad \text{and} \quad \dot{\phi}_{r_i}^i = L_f^{r_i} h_i(x) + \sum_{j=1}^{m} L_{g_j} L_f^{r_i-1} h_i(x) u_j \tag{2.24}$$

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

$$\upsilon = \begin{bmatrix} \dot{\phi}_{r_1}^1 \\ \vdots \\ \dot{\phi}_{r_p}^p \end{bmatrix} \tag{2.25}$$

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.

2.4 Limitations of feedback linearisation

Although feedback control answers to many of the problems of linear controllers such as PID and LQR, it has a big downside. Indeed, being a model based controller, it requires the system model to be quite accurately known. 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 [9]

$$\ddot{x} = \upsilon \tag{2.26}$$

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 [9]

$$\ddot{x} = \upsilon + \Delta(x, \dot{x}, u) \tag{2.27}$$

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.

2.4.1 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 based on variables such as roll, angle of attack and sideslip [10]. 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 facilitate 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 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 [11]. For these reasons this method will not be further studied and implemented in this work.

2.4.2 Neural Networks

Artificial Neural Networks (NN) are a biologically inspired simulation of brain's 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

$$y(x_1, x_2, ..., x_n) = f((\sum_{i=1}^{n} w_i x_i) + b) \tag{2.28}$$

where b is the bias term and f a given activation function. Neuron 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 unknown function.

Batch training is mostly based one a property NN is 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 of 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, 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.

The backpropagation algorithm is one of the most common NN training algorithm, commonly used in batch training, used 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 [12], [13], [14]. 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

$$E = \frac{1}{2}(y_d - y)^2 \tag{2.29}$$

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

$$sum_i = \sum_{k=1}^{n} w_{ki} x_k \tag{2.30}$$

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

$$x_i = f(sum_i) \tag{2.31}$$

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 such

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial x_j} \frac{\partial x_j}{\partial sum_j} \frac{\partial sum_j}{\partial w_{ij}} \tag{2.32}$$

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 2.30 and 2.31 comes that

$$\frac{\partial x_j}{\partial sum_j} = \frac{\partial f(sum_j)}{\partial sum_j} \tag{2.33}$$

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

$$\frac{\partial f(sum_j)}{\partial sum_j} = f(sum_j)(1 - f(sum_j)) \tag{2.34}$$

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

$$\frac{\partial f(sum_j)}{\partial sum_j} = 1 - f^2(sum_j) \tag{2.35}$$

For the term $\frac{\partial sum_j}{\partial w_{ij}}$, from 2.30 this partial derivative is easily computed

$$\frac{\partial sum_j}{\partial w_{ij}} = \frac{\partial \left(\sum_{k=1}^{n} w_{kj} x_k\right)}{\partial w_{ij}} = x_i \tag{2.36}$$

For the output layer, the first term of equation 2.32 can easily be evaluated, as $x_j = y$ for such a case, giving

$$\frac{\partial E}{\partial x_j} = \frac{\partial(\frac{1}{2}(y_d - y)^2)}{\partial y} = y - y_d \tag{2.37}$$

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

$$\frac{\partial E}{\partial x_j} = \sum_{s \in S} \left(\frac{\partial E}{\partial x_s} \frac{\partial x_s}{\partial sum_s} \frac{\partial sum_s}{\partial x_j} \right) = \sum_{s \in S} \left(\frac{\partial E}{\partial x_s} \frac{\partial f(sum_s)}{\partial sum_s} w_{js} \right) \tag{2.38}$$

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 2.32, an update law for the weights can now be stated from the gradient descent algorithm

$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}} \tag{2.39}$$

Where α is a learning coefficient, and its choice is crucial to assure a fast convergence of the solution. Indeed, while a exceedingly large learning coefficient can cause this algorithm to miss the minimum error, a small learning coefficient will also increase the convergence rate. The -1 factor is added in the equation above to converge to a local minima, removing it would make this algorithm maximize the cost function. A limitation of the backstepping algorithm that must also be taken into account is that it only guarantees local minima convergence.

Chapter 3

Implementation

This section will go in depth on the implementation of the concepts presented in the chapter 2. The plane model to be controlled was firstly implemented and verified. The first goal after this first step was to design a controller that would 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 chosen commercial aircraft that will be simulated is the Boeing 737-200, an aircraft with over 30 years of service for which information of flight parameters is readily available. 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

$$\begin{bmatrix} 1278369.56 & 0 & -135588.17 \\ 0 & 3781267.79 & 0 \\ -135588.17 & 0 & 4877649.98 \end{bmatrix} kg.m^2 \tag{3.1}$$

The ISA atmospheric model was used to measure the air density at any given height. The time constants used for the actuators was ξ_{δ_i}50ms and $\xi_T = 4$s for the engines.

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

3.1.1 Plane Dynamics

In order to simulate the aircraft's fast dynamics, its moments must firstly be computed in order to use equation 2.6. The torque caused by the actuators as modelled as equation 2.4, using

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

The remaining aerodynamic coefficients from equation 2.5 were obtained from the work of Hector Escamilla Nuñez in [6]. 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 [6]. Indeed as stated previously, neural networks can approximate any continuous function according to the Universal Approximation Theorem. This method is therefore optimal 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

$$C_D = 0.0176 + 0.0515 C_L^2 \tag{3.2}$$

The sum of the moments can be computed and used to obtain the rotation rates p, q and r. Equation 2.10 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 2.11 using the outputs of the neural networks. The body forces and acceleration relative to the earth frame were then obtain from equations 2.12 and 2.13. 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 2.7. At this point the values of α and β were also obtained for their respective equations 2.8a and 2.8b.

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

3.2 Controller Implementation

To invert such a complex system, two layers of inversion are proposed by H. Escamilla [6], 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. To invert the fast dynamics the equation 2.6 must be differentiated once, to account for both the actuator dynamics 2.15 and the effects of the wind. Doing so yields the jerk vector given by, as per [6]

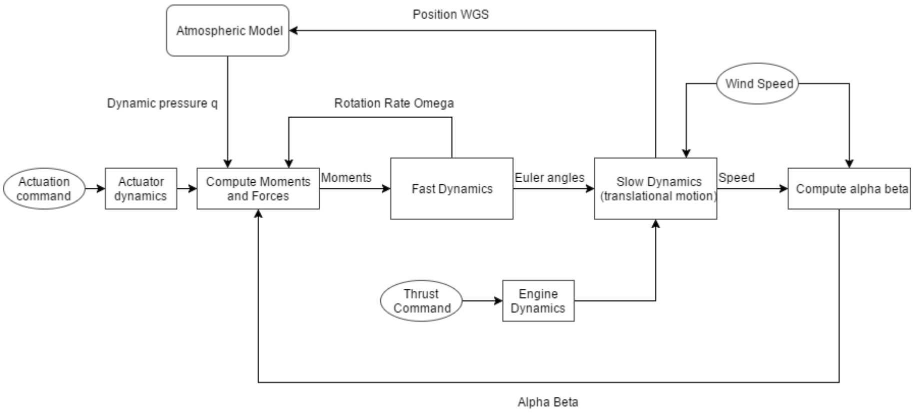
Atmospheric Model
Position WGS
Wind Speed
Dynamic pressure q
Rotation Rate Omega
Actuation command
Actuator dynamics
Compute Moments and Forces
Moments
Fast Dynamics
Euler angles
Slow Dynamics (translational motion)
Speed
Compute alpha beta
Thrust Command
Engine Dynamics
Alpha Beta

Figure 3.1: Plane dynamics simulator diagram

$$\begin{bmatrix} \ddot{p} \\ \ddot{q} \\ \ddot{r} \end{bmatrix} = \frac{1}{2}\rho S I^{-1} \left\{ V_a^2 C_\delta \xi \begin{bmatrix} \delta_{ail}^d - \delta_{ail} \\ \delta_{ele}^d - \delta_{ele} \\ \delta_{rud}^d - \delta_{rud} \end{bmatrix} + V_a^2 C_c \dot{R}_a + 2V_a\dot{V}_a \left(\begin{bmatrix} bC_l \\ \bar{c}C_m \\ bC_n \end{bmatrix} + C_\delta \begin{bmatrix} \delta_{ail} \\ \delta_{ele} \\ \delta_{rud} \end{bmatrix} \right) \right\} + I^{-1} \left(\frac{1}{4}\rho S V_a C_k - In \right) \dot{\Omega} \tag{3.3}$$

Where

$$I = \begin{bmatrix} A & 0 & -E \\ 0 & B & 0 \\ -E & 0 & C \end{bmatrix}$$

$$I_n = \begin{bmatrix} -Eq & (C-B)r - Ep & (C-B)q \\ (A-C)r + 2Ep & 0 & (A-C)p - 2Er \\ (B-A)q & (B-A)p + Er & Eq \end{bmatrix}$$

$$\xi = \begin{bmatrix} \frac{1}{\xi_{ail}} & 0 & 0 \\ 0 & \frac{1}{\xi_{ele}} & 0 \\ 0 & 0 & \frac{1}{\xi_{rud}} \end{bmatrix}$$

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

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

$$\dot{Ra} = [\dot{\alpha} \quad \dot{\beta} \quad \dot{V_a}]^T$$

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 3.3 to δ^d,

$$\begin{bmatrix} \delta^d_{ail} \\ \delta^d_{ele} \\ \delta^d_{rud} \end{bmatrix} = \frac{1}{V_a^2}\xi^-1C_\delta \left\{ \frac{2I}{\rho S}\tau - \frac{2}{\rho S}(\frac{1}{4}\rho S V_a C_k - I_n)\dot{\Omega} - 2V_a\dot{V}_a \left(\begin{bmatrix} bC_l \\ \bar{c}C_m \\ bC_n \end{bmatrix} + C_\delta \delta \right) - V_a^2 C_c \dot{R}_a \right\} + \delta \quad (3.4)$$

Indeed, by replacing 3.4 in the jerk equation 3.3, the equation $\ddot{\Omega} = \tau$ is obtained, resulting in a successfully inverted system.

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

SEE ANNEX HERE

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 2.3, 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, firstly considering without neural network adaptation

$$\tau = -K_P(\Omega - \Omega^d) - K_D(\dot{\Omega} - \dot{\Omega}^d) + \ddot{\Omega}^d \quad (3.5)$$

Where the K_P and K_D coefficients are chosen to ensure stability and reference tracking (HOW TO DO IT NOT EMPIRICALLY?)

From this point the next step shall be to obtain the desired values of Ω. These were obtained using another PD controller, using Euler angle reference values to control rotation rates. Among these reference values to be followed, τ_r was set to zero. This was made to make use of the decoupling through feedback linearisation of the plane's movement, using a roll and pitch motion to turn the aircraft. This results in the following control of rotation rate

$$p = k_P(\phi^d - \phi) + k_d(\dot{\phi}^d - \dot{\phi}) \quad (3.6a)$$

$$q = k_P(\theta^d - \theta) + k_d(\dot{\theta}^d - \dot{\theta}) \quad (3.6b)$$

$$r = 0 \quad (3.6c)$$

It is now possible to control the attitude of the aircraft, and the fast control loop design is finished. 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 to achieve the goal of position tracking over time. Firstly the dynamics for the 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

$$\dot{V_a} = \frac{1}{m}(T\cos\alpha - D - mg\sin\gamma) \tag{3.7a}$$

$$\dot{\gamma} = \frac{1}{mV_a}(T\sin\alpha + L - mg\cos\gamma) \tag{3.7b}$$

These equation can easily be obtained from the sum of forces in figure 3.2. For the yaw rate comes that (SEE ANNEX explanation)

$$\dot{\psi} = \frac{g}{V_a}\tan\phi \tag{3.7c}$$

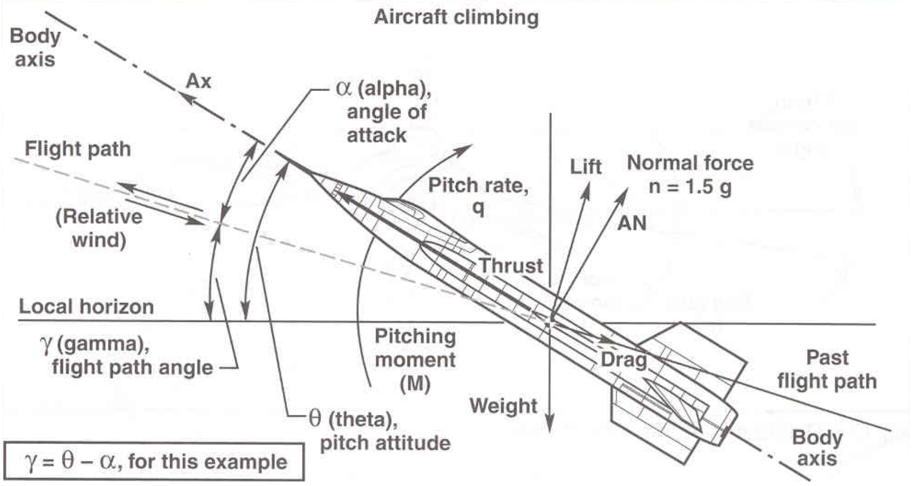
Aircraft climbing
Body axis
Ax
α (alpha), angle of attack
Flight path
(Relative wind)
Pitch rate, q
Lift
Normal force n = 1.5 g
AN
Thrust
Local horizon
γ (gamma), flight path angle
Pitching moment (M)
Drag
Past flight path
Weight
θ (theta), pitch attitude
Body axis
γ = θ – α, for this example

Figure 3.2: γ representation and frames of reference [15]

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

$$\dot{V_a} = \frac{1}{\tau_{V_a}}(V_a^d - V_a) \tag{3.8a}$$

$$\dot{\gamma} = \frac{1}{\tau_\gamma}(\gamma^d - \gamma) \tag{3.8b}$$

$$\dot{\psi} = \frac{1}{\tau_\psi}(\psi^d - \psi) \tag{3.8c}$$

The final step of the controller design will be to propose a guidance law to obtain the values of V_a, γ and ψ from the X,Y and Z reference values along time. For these laws was assumed that the position reference was given relative to a frame of reference with the origin on the surface, with the Z axis pointing upwards. The guidance controller therefore implements the following equations, assuming the errors $\delta x = x_r - x$, $\delta y = y_r - y$ and $\delta z = z - z_r$.

$$\delta V_a = \frac{1}{\tau_{V_a}}(\sqrt{\delta x^2 + \delta y^2 + \delta z^2}) \tag{3.9a}$$

$$V_a^d = \delta V_a + 200ms^{-1} \tag{3.9b}$$

$$\phi^d = atan2(\delta y, \delta x) \tag{3.9c}$$

$$\gamma^d = \frac{1}{\tau_Z}\delta z \tag{3.9d}$$

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, variation in the plat dynamics (e.g., due to payload changes, actuator or sensor degradation) deteriorate the performance and reference tracking error of the controller. To compensate for such errors, adaptive online neural networks have been studied and implemented in several works [16], [17], [13] and [18] to name a few. These however, have been applied almost exclusively to UAV aircrafts, 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 seen previously. However, in case of inversion errors, that can be caused by several factors, the resulting in an equation similar to 2.27, 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, $sig(x) = \frac{1}{(1+e^{-x})}$. 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 3.3. Different numbers of neurons were used and the results obtained will be discussed in chapter 4.

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 [19]:

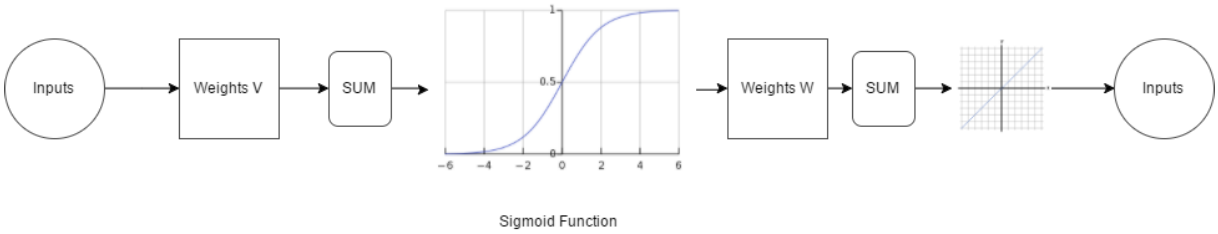

Figure 3.3: Neural Network diagram

- **Relevance**: 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 is 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 the range of values these might take. The sigmoid function $sing(x)$ used reaches around 75% when $x = 1$ and 100% when $x = 2$, as can be seen in figure 3.4. 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 average 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

$$y = 2\frac{x - x_{min}}{x_{max} - x_{min}} - 1 \tag{3.10}$$

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

3.3.2 Training Algorithm

As mentioned in section 2.4.2, a backpropagation algorithm was used to train both set 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, due to inversion errors an error Δ will be present, where $\tau = \ddot{\Omega}^d + \Delta$. This error must be computed on 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 (3.5) comes that

$$\Delta = -K_P(\Omega - \Omega^d) - K_D(\dot{\Omega} - \dot{\Omega}^d) \tag{3.11}$$

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

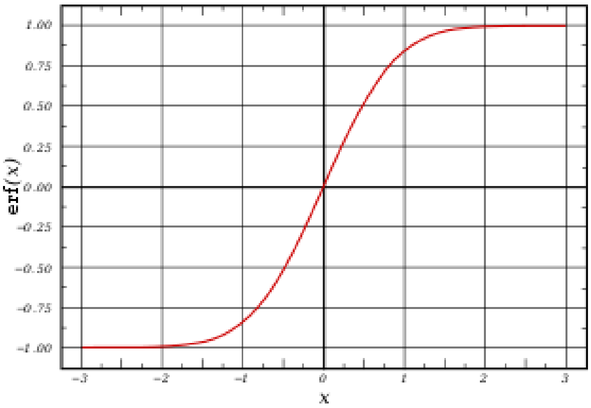

Figure 3.4: Sigmoid Function

$$J = (\Delta - y_{NN})^2 \tag{3.12}$$

The values of $\dot{V}$ and $\dot{W}$ must now be computing, using the gradient descent method described in section 2.4.2. As the name suggest, the back propagation algorithm firstly 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's absolute value in lesser than a given value ξ, 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.

Chapter 4

Results

This chapter will be presenting the 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 that was 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 in aircraft dynamics. 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 3.9 will be added to follow a given trajectory.

4.1 Model Validation

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

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

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

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

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

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

The required thrust can also be computed from both 4.3, 4.4 and 3.2, knowing the C_L for a given angle of attack. Proposing some values of α, the following results are obtained

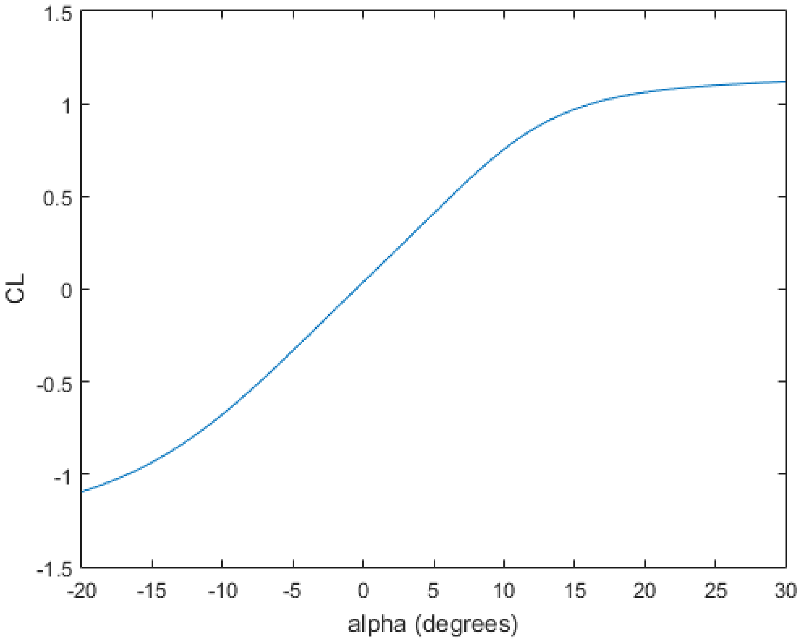

Figure 4.1: C_L versus α graph from Neural Network

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 reference values $V_a^d = 200ms^{-1}$, $\gamma^d = 0rad$ and $\psi^d = 0rad$ 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 were

This simulation, made over 800 seconds, shows clearly that the angle oscillates around a 6^o degree angle of attack, with an amplitude of around 0.4^o degrees. As for the thrust, it also oscillates around $30000N$ with an amplitude of $2000N$. 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 the coefficient neural networks are able to simulate an accurate model of the aerodynamic coefficient's variation.

4.2 Feedback linearisation controller

Once the model has been tested and validated, the described nonlinear inversion controller can now be tested. Reference tracking will firstly be observed with simple constant reference values, without any guidance control law. Reiterating the example from the previous section in cruise conditions for $V_a^d = 200ms^{-1}$, $\gamma^d = 0^o$ and $\psi^d = 0^o$ comes that

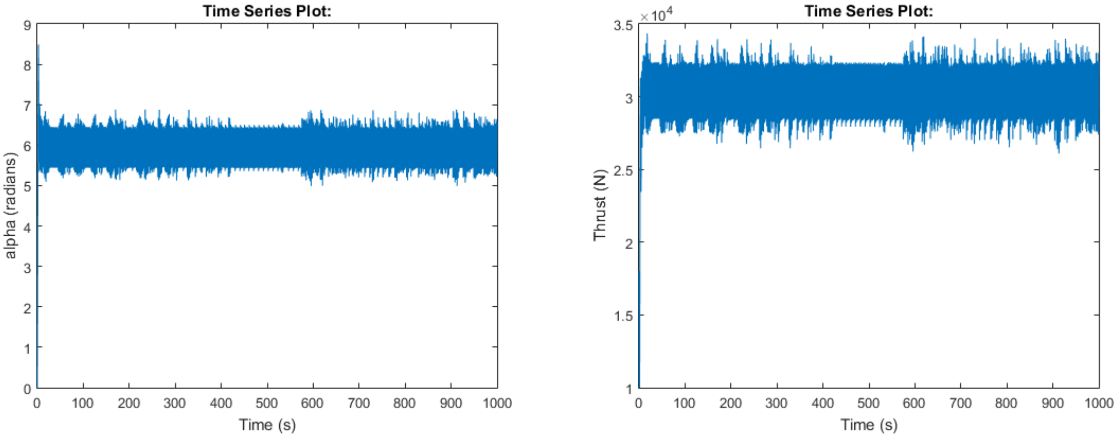

Figure 4.2: Angle of Attack (degrees) and thrust (Newton) of the controlled aircraft

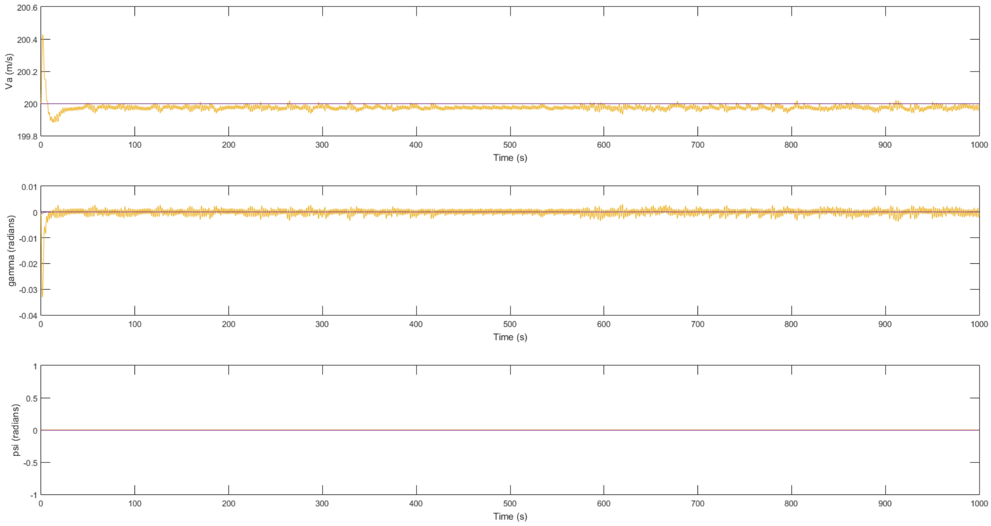

Figure 4.3: Reference (blue) following over 1000s of simulation time for $V_a^d = 200ms^{-1}$, $\psi^d = 0^o$ and $\gamma^d = 0^o$ respectively and measured values (yellow)

As seen in the 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. To be able to follow trajectories however, a good control of the aircraft's heading will be necessary. Proposing this time the following variation in ψ^d

$$\psi^d = \begin{cases} 0^o & t < 100 \\ 90^o & 100 \leq t < 500 \\ 0^o & t > 500 \end{cases} \tag{4.5}$$

The comparing the measured heading ψ and its desired value comes

Although the reference value is reached after each step, the convergence takes around 100s for a 90^o 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.

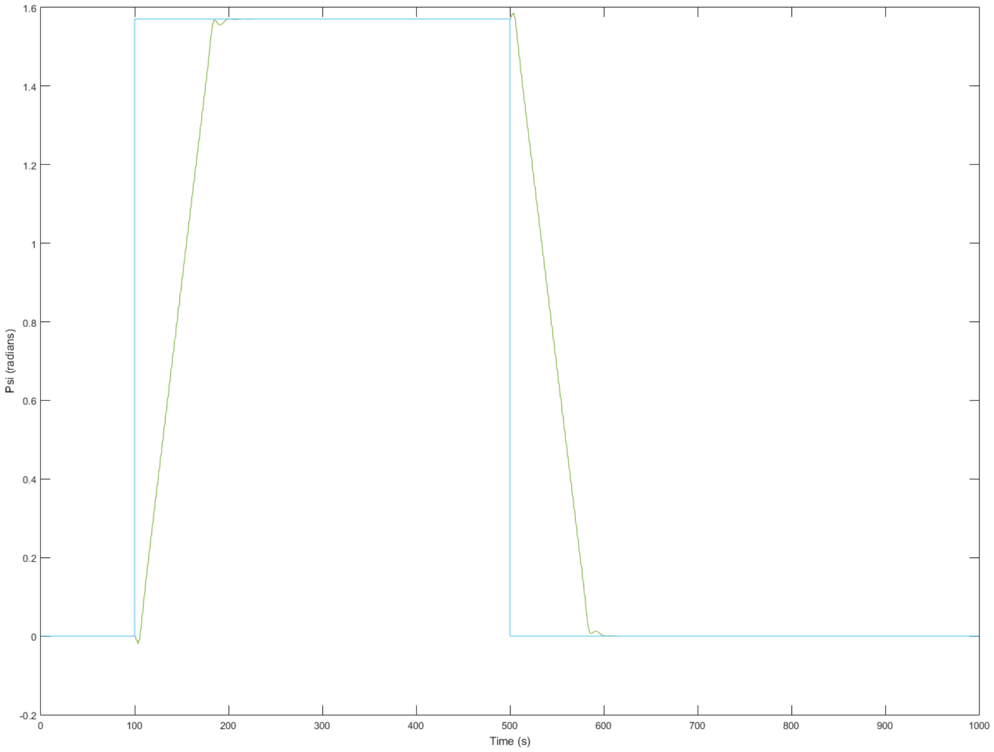

Figure 4.4: Desired (blue) and measured (green) heading

4.3 Disturbances and Errors

From the results obtained in the previous section, it can be concluded the controller performs correctly following the desired reference. These results however were obtain supposing 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.

4.3.1 Inversion errors

This error will be the most frequent when implementing such a controller, as some system parameters estimations may have considerable errors, namely the aircraft's inertial matrix. Fortunately, the controller used is, as 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 A,B,C and E were multiplied by a reducing factor $\zeta \in [0;1]$ when computing the nonlinear inversion in 3.4.

Changing the values of ζ to 0.5 and 0.05, increasing the inversion error, results in the following trajectories in figure 4.6

Indeed, a 50% error in estimating the inertia of the aircraft when computing the feedback linearisation

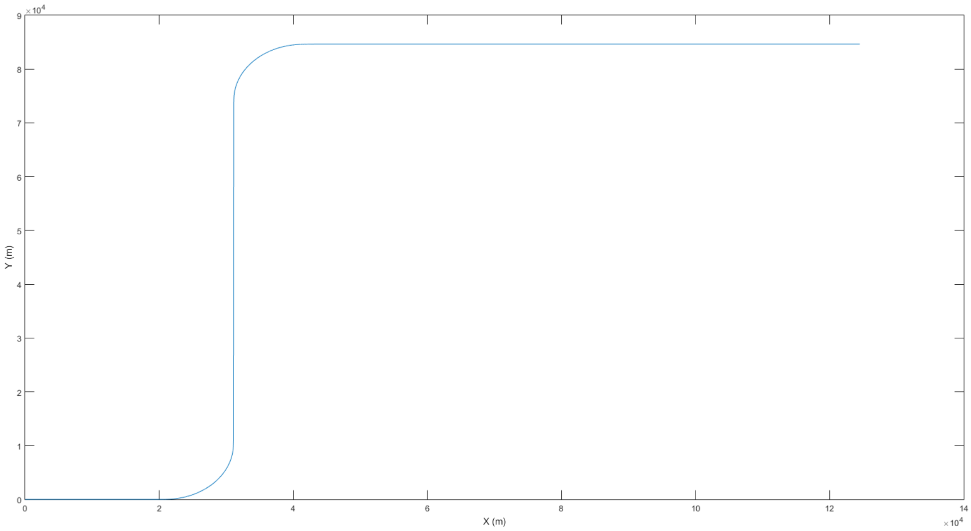

Figure 4.5: Plane trajectory for the mention control input

law results in a negligible reference tracking error, as can be seen in figure 4.7. This however is not the case if ζ is reduced even further to $\zeta = 0.05$ and error of the system relative to its references becomes much greater (figure 4.8. As can be seen in this figure, the aircraft is unable to follow yaw commands in this case as it easily goes unstable. This establishes a first goal to measure the performance of the designed neural network, to try to approach the case of $\zeta = 0.05$ to the reduced errors of a nonlinear inversion with no inertia estimation errors.

4.3.2 System failures

One other cause for inversion errors that can have a much greater 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 shall 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 elevon, 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 4.9. This figure show that the reduced controllability results in a $20km$ difference to that of the undisturbed system.

4.3.3 Wind disturbances

TODO

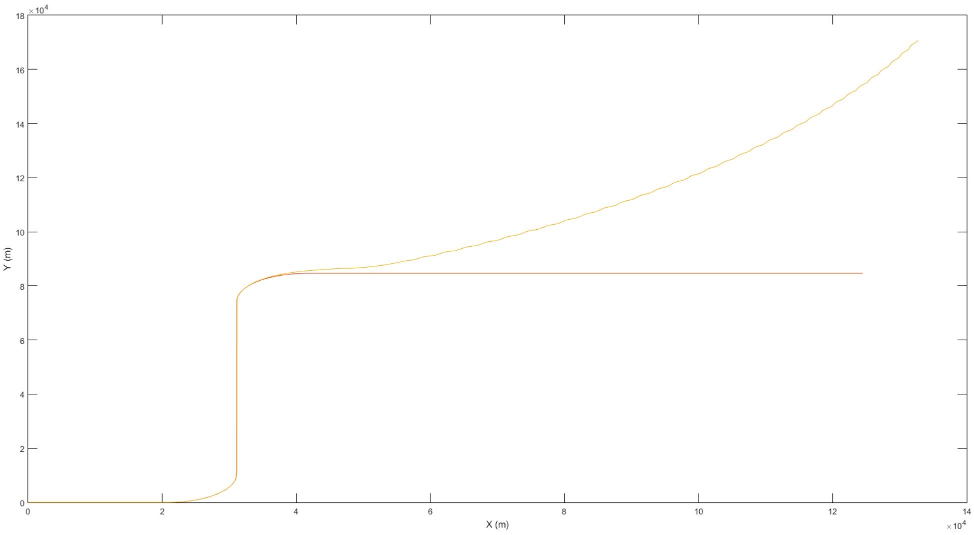

Figure 4.6: Plane trajectory for for $\zeta = 0.5$ (red) and $\zeta = 0.05$ (yellow)

4.4 Neural Network

4.5 Guidance controller

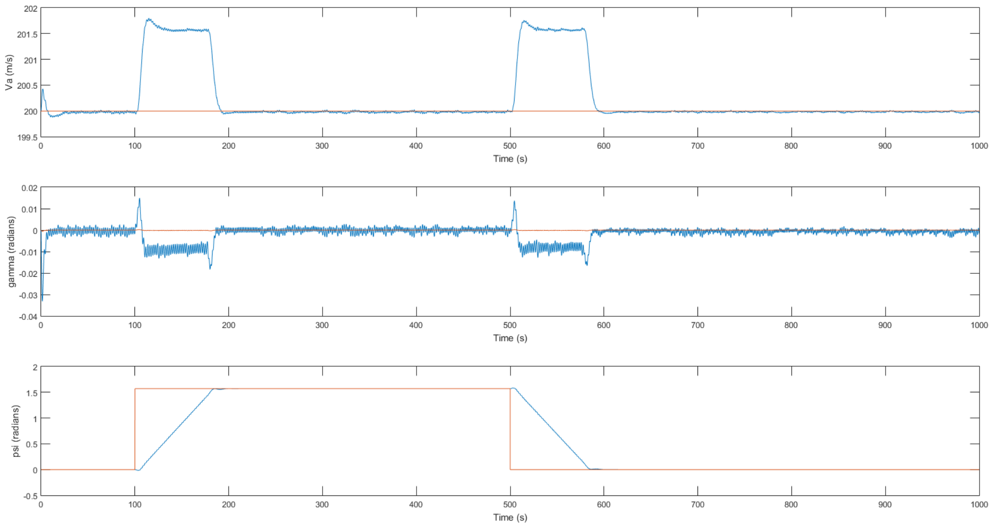

Figure 4.7: V_a, γ and ψ for measured and desired values for $\zeta = 0.5$

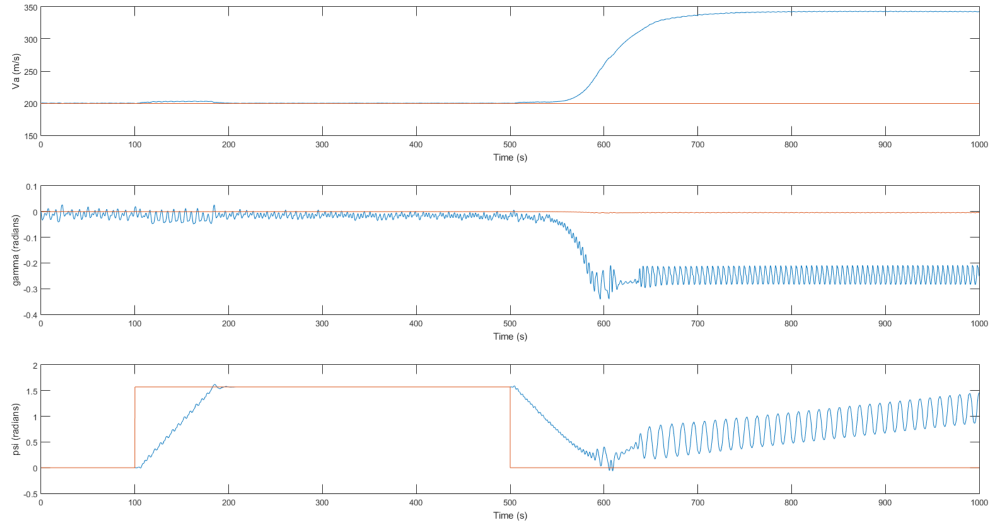

Figure 4.8: V_a, γ and ψ for measured and desired values for $\zeta = 0.05$

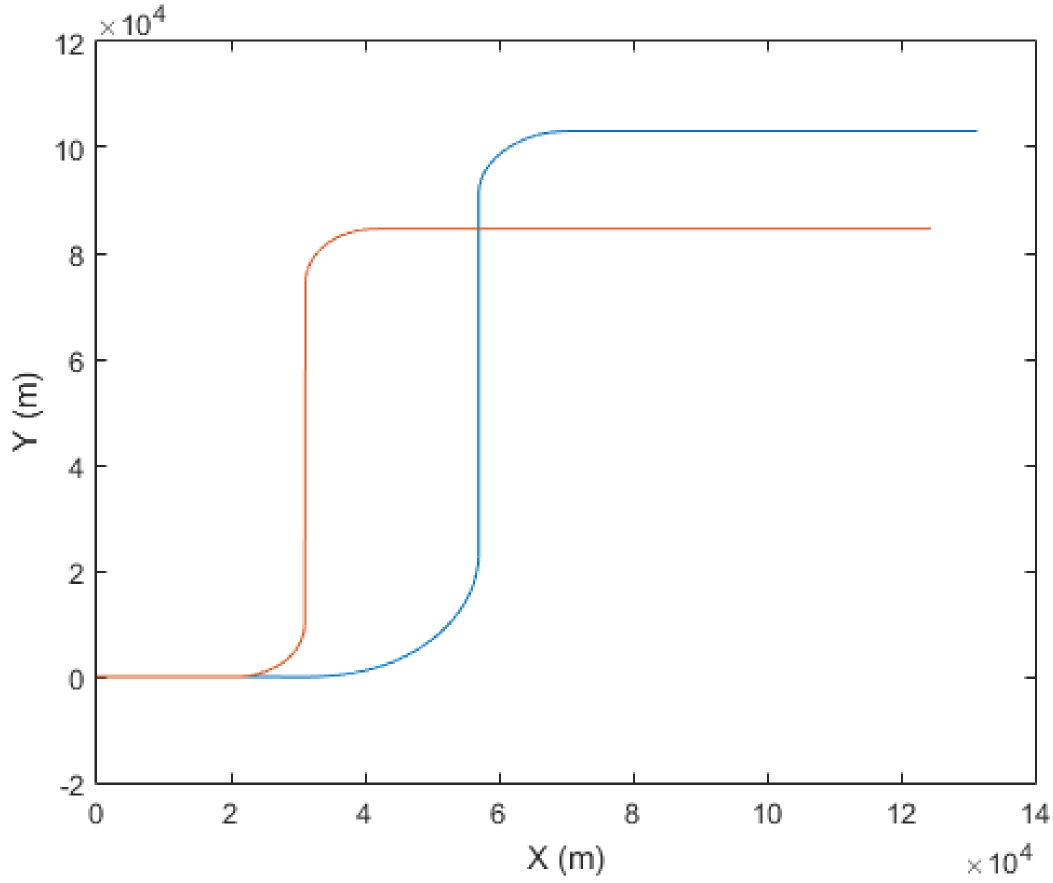

Figure 4.9: Trajectory with 20% reduced actuation (blue) and trajectory without failures (red)

Chapter 5

Conclusions

Insert your chapter material here...

5.1 Achievements

The major achievements of the present work...

5.2 Future Work

A few ideas for future work...

Bibliography

[1] S. 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. S. et al. Neuro-adaptive augmented dynamic inversion controller for quadrotor. *IFAQ Papers Online*, 2016.

[4] M. P. 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. X. 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] F. M. C. H. Escamilla Nuñez. Towards 4d trajectory tracking for transport aircraft. *IFAQ World Congress*, 2017.

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

[8] W. L. Jean-Jacques E. Slotine. *Applied nonlinear control.* 1991.

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

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

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

[12] 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.

[13] 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.

[14] 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.

[15] K. W. Iliff and C. L. Peebles. *From Runway to Orbit: Reflections of a NASA Engineer*. 2004.

[16] A. P. Abhishek Kumar Shastry and M. Kothari. Neuro-adaptive augmented dynamic inversion controller for quadrotors. *IFAC-PapersOnLine*, 2016.

[17] J. S. W. Z. Xiaoxiong Liu, Yan Wu. Adaptive fault-tolerant flight control system design using neural networks. *IEEE International Conference on Industrial Technology*, 2008.

[18] Z. h.-m. Lili. Neural network based adaptive dynamic inversion flight control system design. *IEEE International Conference on Intelligent Control and Information Processing*, 2011.

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

Appendix A

Vector calculus

In case an appendix if deemed necessary, the document cannot exceed a total of 100 pages...

Some definitions and vector identities are listed in the section below.

A.1 Vector identities

$$\nabla \times (\nabla \phi) = 0 \tag{A.1}$$

$$\nabla \cdot (\nabla \times \mathbf{u}) = 0 \tag{A.2}$$