Nonlinear Control Applied to 4D Transport Aircraft Guidance

G. Sousa *

* *Instituto Superior Tcnico, Lisbon, Portugal (guilherme.sousa@ist.utl.pt).*

Abstract:
Following the current ATM industry paradigm shift to allow for commercial aircraft to follow 4D trajectories, in order to increase both safety and air capacity, a controller was design 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 caused by poor estimation of the aircraft parameters, external disturbances or system failures. 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

1. INTRODUCTION

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 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.

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.

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.

Over the recent years, research in intelligent and adaptive flight control systems has seen a consistent increase, in an attempt to solve these limitations, in order to develop flight systems able to adapt to external disturbances SANTOSO et al. (2017). Of the existing intelligent control techniques used to solve the dependency of model-based control systems on an accurate mathematical model and the uncertainties caused by external disturbances or component failures, neural networks have been the most successful in doing so. Applied to UAV control, research works such as Tang and Patton (2013), Xiang et al. (2016), Lin et al. (2013) and **?** have showed neural networks can be used to increase flight control stability and render flight systems adaptable to disturbances. For this work an online neural network was used to improve a model based flight controller of a commercial aircraft.

This paper is organized as follows, Section 2 provides the mathematical model used as well as actuator dynamics. Section 3 focuses on the model-based approach used to control the aircraft, as well as a description on the neural network used to improve said control law. This section will also cover the implementation of the NN described previously and the guidance law used to ensure 4D trajectory following. Lastly Section 4 shows simulation results of the control approach, and conclusions are given in Section 5.

2. FLIGHT DYNAMICS

The work made in this article was built on top of the work done by Nuez and Camino (2017) on 4D trajectory tracking. The model used in this work is a six degree of freedom transport aircraft that will be described in this section.

⋆ Acknowledgements go here...

2.1 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

$$R_{BW} = \begin{bmatrix} c_\alpha c_\beta & -c_\alpha s_\beta & -s_\alpha \\ s_\beta & c_\beta & 0 \\ s_\alpha c_\beta & -s_\alpha s_\beta & c_\alpha \end{bmatrix} \tag{1}$$

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

$$R_{EB} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta c_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{2}$$

2.2 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

$$\begin{bmatrix} L' \\ M \\ N \end{bmatrix} = \frac{1}{2}\rho S V_a^2 \left(\begin{bmatrix} bC_l \\ \bar{c}C_m \\ bC_n \end{bmatrix} + C_\delta \delta \right) \tag{3}$$

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

$$C_\delta = \begin{bmatrix} bC_{l\delta_{ail}} & 0 & bC_{l\delta_{rud}} \\ 0 & \bar{c}C_{m\delta_{ele}} & 0 \\ bC_{n\delta_{ail}} & 0 & bC_{n\delta_{rud}} \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} C_l \\ C_m \\ C_n \end{bmatrix} = \begin{bmatrix} C_{l\beta}\beta + C_{l_p} p\frac{b}{2V_a} + C_{l_r} r\frac{b}{2V_a} \\ C_{m_0} + C_{m_\alpha}\alpha + C_{m_q} q\frac{\bar{c}}{2V_a} \\ C_{n\beta}\beta + C_{n_p} p\frac{b}{2V_a} + C_{n_r} r\frac{b}{2V_a} \end{bmatrix} \tag{5}$$

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 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 Nuez and Camino (2017), I being the aircraft inertial matrix.

$$\dot{\Omega} = I^{-1} M_{ext} - I^{-1}\Omega \times (I\Omega) \tag{6a}$$

$$\dot{\Omega} = \frac{1}{2}\rho S I^{-1} V_a^2 \left(\begin{bmatrix} bC_l \\ \bar{c}C_m \\ bC_n \end{bmatrix} + C_\delta \delta \right) - I^{-1}\Omega \times (I\Omega) \tag{6b}$$

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 $\boldsymbol{V_G} = [u \quad v \quad w]^T$ be the speed of the CG relative to the ground, $\boldsymbol{V}$ the speed of the CG relative to the air mass and $\boldsymbol{W}$ the speed of the wind relative to the ground, then as per Etkin and Reid Etkin and Reid (1996)

$$\boldsymbol{V_G} = \boldsymbol{V} + \boldsymbol{W} = \begin{bmatrix} V_a c_\alpha c_\beta + V_{w_x} \\ V_a s_\beta + V_{w_y} \\ V_a s_\alpha c_\beta + V_{w_z} \end{bmatrix} \tag{7}$$

and α and β can be computed by

$$\alpha = arctan\left(\frac{w - V_{w_z}}{uV_{w_x}}\right) \tag{8a}$$

$$\beta = arctan\left(\frac{v - V_{w_y}}{V_a}\right) \tag{8b}$$

From these three equations, differentiating 8a and 8b, and from equation 7 and the translation dynamics equation 13 comes that

$$\begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{V_a} \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & 0 & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} \tag{9}$$

Where the entries of the matrix are given in annex A.

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 on a Matlab simulation, and is given by

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & tg_\theta s_\phi & tg_\theta c_\phi \\ 0 & c_\phi & -s_\phi \\ 0 & \frac{s_\phi}{c_\theta} & \frac{c_\phi}{c_\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{10}$$

2.3 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

$$\begin{bmatrix} D \\ Y \\ L \end{bmatrix} = \frac{1}{2}\rho S V_a^2 \begin{bmatrix} C_D \\ C_Y \\ C_L \end{bmatrix} \tag{11}$$

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.

$$\begin{bmatrix} F_{xa} \\ F_{ya} \\ F_{za} \end{bmatrix} = R_{WB} \begin{bmatrix} -D \\ Y \\ -L \end{bmatrix} \tag{12}$$

From Newton's Second Law comes the aircraft acceleration

$$\begin{bmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{bmatrix} = \begin{bmatrix}\frac{1}{m}(F_{xa}+T) - gs_\theta + rv - qw\\ \frac{1}{m}F_{ya} + gc_\theta s_\phi + pw - ru\\ \frac{1}{m}F_{za} + gc_\theta c_\phi + qu - pv\end{bmatrix} \tag{13}$$

An expression in the Earth frame can also be obtained

$$\begin{bmatrix}\ddot{x}_E\\ \ddot{y}_E\\ \ddot{z}_E\end{bmatrix} = \frac{1}{m}R_{BE}\begin{bmatrix}F_{xa}+T\\ F_{ya}\\ F_{za}\end{bmatrix} + \begin{bmatrix}0\\0\\g\end{bmatrix} \tag{14}$$

2.4 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 Nuez and Camino (2017). For the control surfaces δ_i, given a desired δ_i^d comes

$$\dot{\delta}_i = \frac{1}{\xi_i}(\delta_i^d - \delta_i) \tag{15}$$

Similarly for thrust

$$\dot{T} = \frac{1}{\xi_T}(T^d - T) \tag{16}$$

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, $\xi_T >> \xi_i$. 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

$$\begin{bmatrix}1278369.56 & 0 & -135588.17\\ 0 & 3781267.79 & 0\\ -135588.17 & 0 & 4877649.98\end{bmatrix} kg.m^2 \tag{17}$$

The ISA atmospheric model was used to measure the air density at any given height. The time constants used for the actuators was $\xi_{\delta_i} = 50$ms and $\xi_T = 4$s for the engines. A simplified block diagram of the plane simulator is given by Figure 1.

Table 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. IMPLEMENTATION

To invert such a complex system (figure 1), two layers of inversion are proposed by H. Escamilla Nuez and Camino (2017), 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.

3.1 Fast Dynamics

To invert the fast dynamics equation 6 must be differentiated once, to account for both the actuator dynamics 15 and the wind effects. Doing so yields the jerk vector given by, as per Nuez and Camino (2017)

$$\begin{bmatrix}\dddot{p}\\ \dddot{q}\\ \dddot{r}\end{bmatrix} = \frac{1}{2}\rho S I^{-1}\left\{V_a^2 C_\delta \xi^{-1}\begin{bmatrix}\delta_{ail}^d - \delta_{ail}\\ \delta_{ele}^d - \delta_{ele}\\ \delta_{rud}^d - \delta_{rud}\end{bmatrix} + V_a^2 C_c \dot{R}_a + 2V_a\dot{V}_a\left(\begin{bmatrix}bC_l\\ \bar{c}C_m\\ bC_n\end{bmatrix} + C_\delta\begin{bmatrix}\delta_{ail}\\ \delta_{ele}\\ \delta_{rud}\end{bmatrix}\right)\right\} + I^{-1}\left(\frac{1}{4}\rho S V_a C_k - I_n\right)\dot{\Omega} \tag{18}$$

where

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

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

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

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

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

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

To do a feedback linearisation, a pseudo input must be chosen, in this case $\tau = \dddot{\Omega}$. The feedback control law will therefore be, solving equation 18 for δ^d,

$$\begin{bmatrix}\delta_{ail}^d\\ \delta_{ele}^d\\ \delta_{rud}^d\end{bmatrix} = \frac{1}{V_a^2}\xi C_\delta^{-1}\left\{\frac{2I}{\rho S}\tau - \frac{2}{\rho S}(\frac{1}{4}\rho S V_a C_k - I_n)\dot{\Omega} - 2V_a\dot{V}_a\left(\begin{bmatrix}bC_l\\ \bar{c}C_m\\ bC_n\end{bmatrix} + C_\delta\delta\right) - V_a^2 C_c\dot{R}_a\right\} + \delta \tag{19}$$

Indeed, by replacing 19 in the jerk equation 18, the equation $\dddot{\Omega} = \tau$ 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

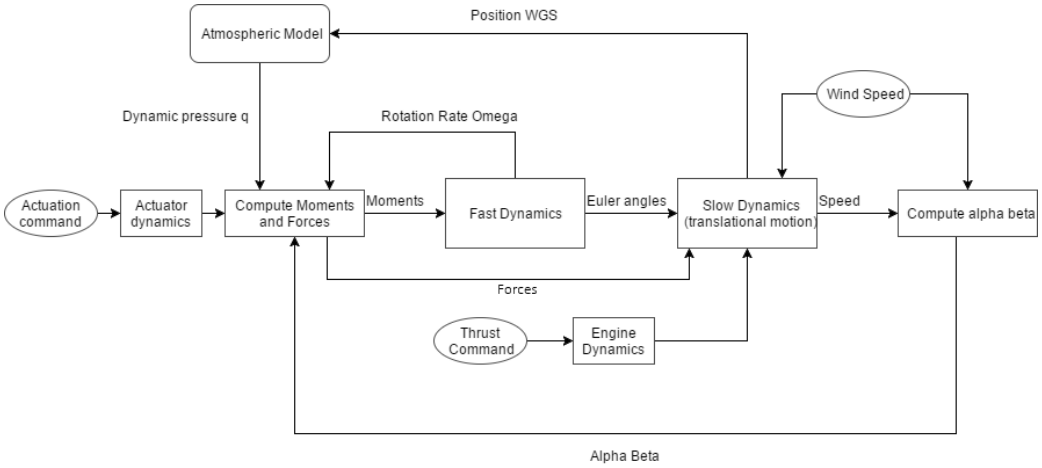

Fig. 1. Plane dynamics simulator diagram

ground. The value of $\dot{R}_a$ can be computed from equation A.

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 the previous section, 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

$$\tau = -K_P(\Omega - \Omega^d) - K_D(\dot{\Omega} - \dot{\Omega}^d) + \ddot{\Omega}^d \tag{20}$$

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

$$\ddot{e} = -K_P e - K_D \dot{e} \tag{21}$$

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

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -K_{P_i} & -K_{D_i} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{22}$$

The characteristic equation is then obtained as

$$\det(\lambda I - A_i) = \lambda^2 + 2\lambda\zeta_i\omega_{n_i} + \omega_{n_i}^2 = \lambda^2 + \lambda K_{D_i} + K_{P_i} = 0 \tag{23}$$

$$\zeta_i = \frac{K_{D_i}\sqrt{K_{P_i}}}{2K_{P_i}} \tag{24}$$

$$\omega_{n_i} = \sqrt{K_{P_i}} \tag{25}$$

The chosen values are given in table 2.

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.

Table 2. Linearised system chosen dynamics

i	K_{P_i}	K_{D_i}	ζ_i	$\omega_{n_i}(\text{rad } s^{-1})$
p	100	20	1	10
q	5	1	0.22	2.24
r	100	20	1	10

$$p = k_{P_p}(\phi^d - \phi) + k_{D_p}(\dot{\phi^d} - \dot{\phi}) \tag{26a}$$

$$q = k_{P_q}(\theta^d - \theta) + k_{D_q}(\dot{\theta^d} - \dot{\theta}) \tag{26b}$$

$$r = 0 \tag{26c}$$

3.2 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

$$\dot{V_a} = \frac{1}{m}(T\cos\alpha - D - mg\sin\gamma) \tag{27a}$$

$$\dot{\gamma} = \frac{1}{mV_a}(T\sin\alpha + L - mg\cos\gamma) \tag{27b}$$

For the yaw rate comes that

$$\dot{\psi} = \frac{g}{V_a}\tan\phi \tag{27c}$$

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

$$\dot{V_a} = \frac{1}{\xi_{V_a}}(V_a^d - V_a) \tag{28a}$$

$$\dot{\gamma} = \frac{1}{\xi_\gamma}(\gamma^d - \gamma) \tag{28b}$$

$$\dot{\psi} = \frac{1}{\xi_\psi}(\psi^d - \psi) \tag{28c}$$

3.3 Online Neural Network

From an error-less nonlinear inversion, the pseuso-input τ would directly control $\dot{\Omega}$ 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, comes 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

$$\tau + \tau_{NN} = \ddot{\Omega}^d + \Delta \tag{29}$$

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.

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 by May et al. (2011):

- **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 75% when $x = 1$ and 100% when $x = 2$. 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

$$y = 2\frac{x - x_{min}}{x_{max} - x_{min}} - 1 \tag{30}$$

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

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 (20) it comes that

$$\Delta = -K_P(\Omega - \Omega^d) - K_D(\dot{\Omega} - \dot{\Omega}^d) \tag{31}$$

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

$$J = \frac{1}{2}(\Delta - y_{NN})^2 \tag{32}$$

From this cost function, the weights can be updated iteratively, where the variation of a given weight from the i^{th} input of the j^{th} neuron from a layer w is given by

$$\Delta w_{ij} = -\alpha \frac{\partial J}{\partial w_{ij}} \tag{33}$$

Where α is the learning coefficient. The chain rule can be used to compute $\frac{\partial J}{\partial w_{ij}}$ as such

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial sig(sum_j)} \frac{\partial sig(sum_j)}{\partial sum_j} \frac{\partial sum_j}{\partial w_{ij}} \tag{34}$$

Where sum_j is given by

$$sum_i = \sum_{k=1}^{n} w_{ki} x_k \tag{35}$$

Where x_k is the output of the k^{th} neuron of the previous layer (or of the k^{th} input in the case of the first weight layer).

These three factors of equation 34 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.

For an output layer, where values may be unbounded, linear output functions may be used, for which $\frac{\partial f(sum_j)}{\partial sum_j}$ has a trivial solution. For the case of a logistic sigmoid activation function comes that

$$\frac{\partial f(sum_j)}{\partial sum_j} = f(sum_j)(1 - f(sum_j)) \tag{36}$$

For the term $\frac{\partial sum_j}{\partial w_{ij}}$, from 35 this partial derivative is easily computed

$$\frac{\partial sum_j}{\partial w_{ij}} = \frac{\partial \left(\sum_{k=1}^{n} w_{kj} x_k\right)}{\partial w_{ij}} = x_i \tag{37}$$

For the output layer, the first term of equation 34 can easily be evaluated, as $x_j = y$ for such a case, giving

$$\frac{\partial J}{\partial x_j} = \frac{\partial(\frac{1}{2}(\Delta - y_{NN})^2)}{\partial y_{NN}} = \Delta - y_{NN} \tag{38}$$

For input layers however, this derivative is less obvious to estimate. One can consider that the error function J 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

$$\frac{\partial J}{\partial x_j} = \sum_{s \in S} \left(\frac{\partial J}{\partial x_s} \frac{\partial x_s}{\partial sum_s} \frac{\partial sum_s}{\partial x_j} \right) = \tag{39}$$

$$\sum_{s \in S} \left(\frac{\partial J}{\partial x_s} \frac{\partial f(sum_s)}{\partial sum_s} w_{js} \right) \tag{40}$$

Note that the term $\frac{\partial J}{\partial x_s}$ represent the all the derivatives of the errors with respect to the outputs of the next layer of neurons. Once this derivative is computed the new set of weights for both layers can be estimated from the update law 33.

3.4 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 errors $\delta x = x_r - x$, $\delta y = y_r - y$ and $\delta z = z - z_r$.

$$\delta V_a = \frac{1}{\tau_{V_a}} (\sqrt{\delta x^2 + \delta y^2 + \delta z^2}) \tag{41a}$$

$$V_a^d = \delta V_a + V_{a_0} \tag{41b}$$

$$\psi^d = atan2(\delta y, \delta x) \tag{41c}$$

$$\gamma^d = \frac{1}{\tau_Z} \delta z \tag{41d}$$

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 flight path angle must be computed an fed to the controller.

Taking firstly the airspeed equation from 41, 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. 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.

Finally, this control architecture is illustrated by the block diagram of Figure 2, specifying fast dynamics inversion, the slow dynamics controller and the guidance control law, as well as the online neural network use to improve the controller and adapt to inversion errors and system failures.

4. RESULTS

This section will focus in testing and comparing the control algorithm developed so far, with and without the online NN corrections. These results will mainly focus in comparing both performances when facing modelling errors and system failures.

4.1 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 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 19. The inertial matrix used in this equation I_{est} is therefore given by $I_{est} = \zeta I$.

In order to more accurately visualise the influence of the ζ coefficient 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 3 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 $\zeta < 0.02$.

Although the errors showed in figure 4 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.

4.2 System Failures and External Disturbances

One other cause for inversion errors that can have a much greater impact on flight trajectory are system failures. The reference trajectory that will be used shall be the same as used previously. The first failure to be simulated will be a

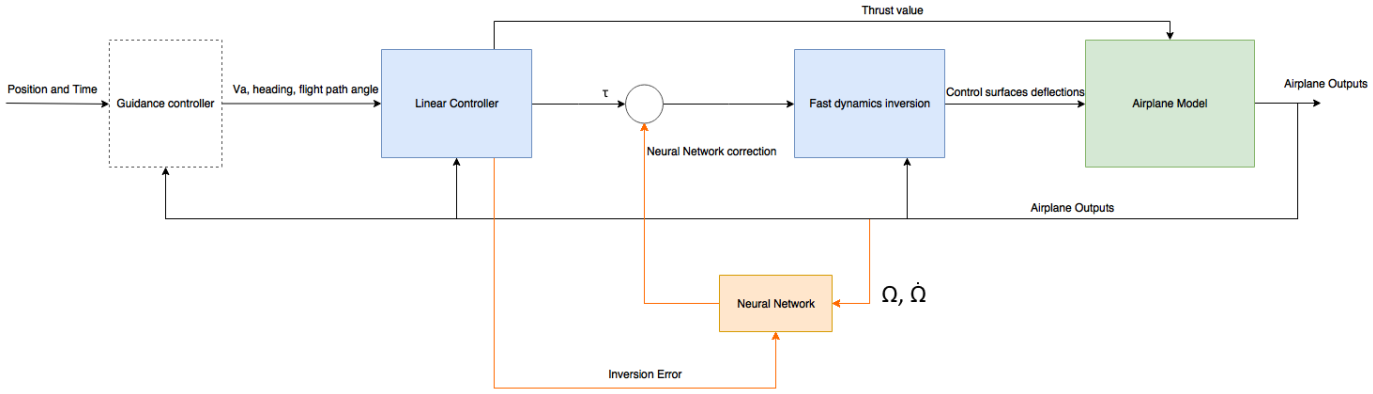

Fig. 2. Diagram of the full controller architecture. In orange is specified the neural network correction system of the initial controller

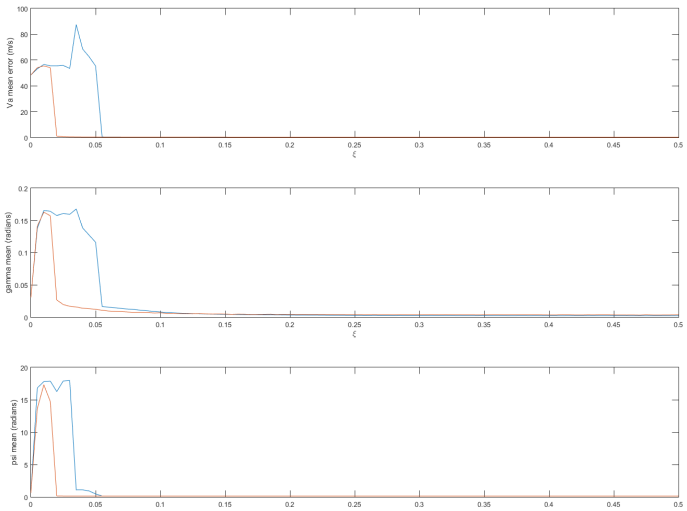

Fig. 3. Mean errors for V_a, γ and psi for $\zeta = [0, 0.5]$ with neural network compensation (orange) and without compensation (blue)

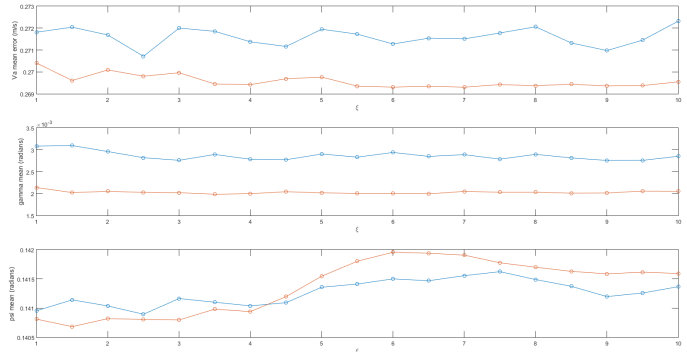

Fig. 4. Mean errors for V_a, γ and psi for $\zeta = [1, 10]$ with neural network compensation (orange) and without compensation (blue)

control surfaces failure, that will lead to reduced controllability of the aircraft. In this work this was replicated in a simulated environment by reducing by 80% each moment coefficients for the elevator, aileron and rudder, namely $C_{\delta_{ele}}$, $C_{\delta_{ail}}$ and $C_{\delta_{rud}}$.

Applying the adaptive correction through the online neural network, the following trajectories were obtained

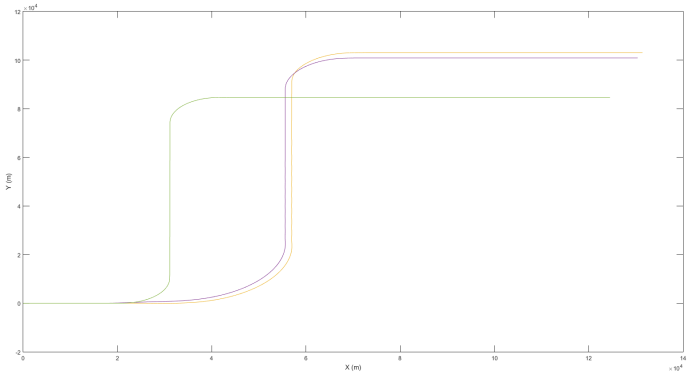

Fig. 5. Trajectory with 20% reduced actuation (yellow) and trajectory without failures (green). The corrected trajectory by the NN can be seen in purple

The first observation that can be drawn from figure 5 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 used. 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.

One second type of flight perturbation that was studied 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, according to the Fed (2015).

- **Stall** 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% Fed (2015).

- **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 AOP.

Following these indications to simulate heavy icing condition, the lift coefficient $C_{L_{max}}$ was reduced by 30%, and the stall angle reduced to 15^o. The value of the drag coefficient was also augmented by 200%. 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 $\psi^d = 45\sin(1.1459t)$ degrees, t being the simulation time. A sinusoidal reference was used to make the effects of simulated icing more noticeable, and the airspeed and FPA references of $V_a^d = 200ms^{-1}$ and $\gamma^d = 0$. For this case, the results obtained were as follows

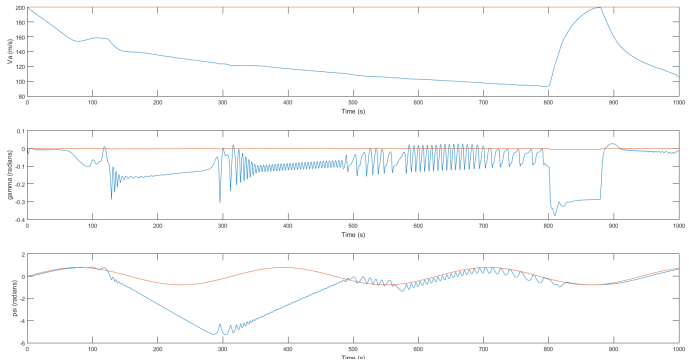

Fig. 6. V_a, γ and ψ for measured and desired values in icing conditions without NN correction

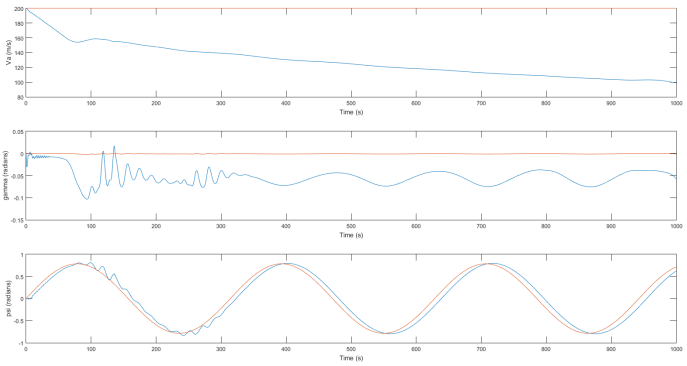

Fig. 7. V_a, γ and ψ for measured and desired values in icing conditions with NN correction

From figure 6, 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 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.

In Figure 7, the aircraft shows a more stable behaviour and smaller tracking errors. 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 $0.3rad$, instead oscillating around $0.05rad$ 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 the desired 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 $t < 300s$.

5. CONCLUSIONS

The nonlinear control law for fast dynamics was 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.

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.

REFERENCES

Safety Advisory - Aircraft Icing. AOPA.

Bernard Etkin and Lloyd Duff Reid. *Dynamics of Flight Stability and Control*, chapter 3. 1996.

Pilot Guide: Flight in Icing Conditions. Federal Aviation Administration, 2015. chapter 3.

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.

Robert May, Graeme Dandy, and Holger Maier. *Review of Input Variable Selection Methods for Artificial Neural Networks, Artificial Neural Networks - Methodological Advances and Biomedical Applications*, chapter 2. 2011.

H. Escamilla Nuez and F. Mora Camino. Towards 4d trajectory tracking for transport aircraft. *IFAC World Congress*, 2017.

SANTOSO et al. State-of-the-art intelligent flight control systems in unmanned aerial vehicles. *IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING*, pages 1,2, 2017.

Y. Tang and R. J. Patton. Reconfigurable flight control using feedback linearization with online neural network adaption. In *2013 Conference on Control and Fault-Tolerant Systems (SysTol)*, pages 324–329, Oct 2013. doi: 10.1109/SysTol.2013.6693903.

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.

Appendix A. $\dot{R_A}$ EQUATION

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

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

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

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

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

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

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

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

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

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

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