JACOBS
UNIVERSITY

Stabilization and Rotation Tracking Controller for Helicopters using Echo State Networks

by

Vlad Zamfir

Bachelor Thesis in Computer Science

Prof. Dr. Herbert Jaeger
Name and title of the supervisor

Date of Submission: May 12, 2017

Jacobs University — School of Engineering and Science

With my signature, I certify that this thesis has been written by me using only the indicates resources and materials. Where I have presented data and results, the data and results are complete, genuine, and have been obtained by me unless otherwise acknowledged; where my results derive from computer programs, these computer programs have been written by me unless otherwise acknowledged. I further confirm that this thesis has not been submitted, either in part or as a whole, for any other academic degree at this or another institution.

Signature
Vlad Zamfir

Place, Date
Bremen, Germany, May 12, 2017

Abstract

Autonomous flight for helicopters is considered to be a very challenging control problem, due to the instability and complex dynamics of the plant. Various successful techniques and approaches have been presented in the past and continue to arise as technology progresses.

Echo State Networks are a type of Recurrent Neural Networks (RNN), which are easy to use because of their reduced complexity and simple, efficient training algorithms. ESNs have proven their applicability in a wide range of engineering fields such as dynamical pattern recognition (e.g. handwriting recognition), communications (e.g. channel equation), pattern generation (e.g. robot modeling), time series prediction (e.g. stock markets).

The objective of this guided research is to design a controller for an unmanned helicopter model using an echo state network. Both stabilization of the helicopter and rotational trajectory tracking are desired. As ESNs are capable of modeling time-varying data in complex non-linear systems, they show good potential in solving the presented task and achieving high precision results.

Contents

1 Introduction and Motivation of Research

As aerial vehicle technology progressed and increased in complexity, new maneuverability difficulties appeared and the development of automatic stabilization and control systems such as auto-pilots became a necessity. Later this led to the field of autonomous flight, an area of high research interest due to its applicability in varied domains including military, surveillance, space exploration and civilian rescue.

As helicopters are non-linear dynamical systems which present instability and uncertainty, their control regarding both stabilization for a given pose and trajectory tracking, represents a challenging problem which was extensively researched during the past few decades. Some classical solutions which work well on simplified models without disturbances involve proportional-integral-derivative (PID) or linearization techniques such as the Linear Quadratic Regulator (LQR) [Belkheiri et al., 2012]. Another common solution uses the concept of backstepping, which divides the system into several controllable subsystems and applies the problem for each simplified case [Azinheira and Moutinho, 2008].

Other methods which recently started to show potential and are researched in the context of the unmanned aerial vehicle (UAV) control make use of artificial neural networks (NNs). Because of their capability to approximate non-linear functions with arbitrary precision, neural networks proved to be very good in identifying and controlling non-linear dynamical systems [Narendra and Parthasarathy, 1990].

Among these approaches, one NN-based control model which achieved relative stability in vertical flight when limited to the roll and pitch axis was presented for a four rotor helicopter [Dunfied et al., 2004]. An altitude controller for helicopters was also designed using neural networks combined, however with other techniques such as PID-control, genetic algorithm and fuzzy logic control [Zein-Sabatto and Zheng, 1997].

A possible improvement could be obtained by considering the echo state networks (ESN), which are easy to use and offer simple training techniques, as it is demonstrated in the following sections. ESNs were confirmed to be good in modeling non-linear dynamical systems and they have been used for simple robot control tasks. Nevertheless they have yet to prove their applicability in controlling more complex, non-holonomic systems such as helicopters, which have more degrees of freedom than was previously demonstrated using ESNs, except for a master thesis whose outcome, however has potential for much further elaboration [Savu, 2014].

The objective of this guided research is to design an ESN-based controller for a simplified helicopter model. The success of the project would demonstrate that echo state networks can have good applicability in modeling controllers for UAVs and for other more general, complex, non-holonomic systems. The primal purpose of the model is to achieve hover mode stabilization for a helicopter in free, vertical position for a given pose, represented as Euler angles. Ultimately, the full picture will include a reference model for trajectory tracking. An extension of these objectives on altitude control is also desired.

The thesis has the following structure: Section 2 introduces the relevant technical knowledge from the three main fields investigated in this research: echo state networks, control theory and unmanned aerial vehicles dynamics. Section 3 describes how the main investigation was conducted, including a more detailed description of the task structure, the construction of the simulator, data generation and the training algorithm. Then Section 4 presents the results analysis together with the main evaluation criteria and testing methods used. The level of success achieved by the project is also stated within this chapter. Finally Section 5 concludes the thesis document, by restating the main points, objectives and achievements while it also contains suggestions for further extensions and improvements.

2 Technical Background

2.1 Echo State Networks

Artificial neural networks (ANNs) are a computational approach inspired by the way a biological brain learns and solves problems. They are represented as a weighted graph, in which the nodes are called neurons and the edges are named synaptic connections or links, having weights as labels [Jaeger, 2017]. The most commonly used ANNs are feed-forward networks, also called multilayer perceptrons (MLPs). These have a wide applicability in very diverse domains as MLPs are proved to be "universal approximators" [Draghici, 2002]. This means that for a certain class F of functions, for any function $f \in F$ there exists a feed-forward neural network that can approximate f with arbitrary precision for all possible inputs. This has been proven for multiple classes F such that together they include all practically relevant functions [Jaeger, 2017].

Recurrent neural networks (RNNs) are a class of neural networks in which the connections between neurons form cycles. This allows them to model non-linear dynamical systems. As opposed to feed-forward networks (which do not contain synaptic cycles [Jaeger, 2011]), they can be used for tasks which require interpretation of temporal, signal inputs, such as speech recognition or handwriting recognition.

Echo state networks are based on the RNN paradigm, with the advantage that they bring simple and computationally efficient training algorithms [Jaeger, 2002]. ESNs and the Liquid State Machines were introduced independently and simultaneously, later forming the field of reservoir computing, which assumes that for RNNs with certain properties, only the training of the output weights is sufficient to achieve performance in many types of problems, while the adaptation of the internal synaptic weights is not necessary [Lukoševičius et al., 2012].

An essential property which is required by the ESNs to achieve stability is the echo state property, which states that after a certain running time, the network should "forget" its initial conditions, i.e. the ESN state is uniquely determined by the teacher input signal. A formal definition is given in [Jaeger et al., 2007].

ESNs are discrete-time neural networks with the structure described below and visually represented in Figure 1 (the notations from [Jaeger, 2002] and [Jaeger, 2010] are followed closely):

The units of an ESN are divided into three categories: K input units, N internal network units which represent the reservoir and L output units. The weighted connections between the neurons are represented by the following real-valued matrices: W^{in} of size $N \times K$ - links the input units with the reservoir; W of size $N \times N$ - interconnects the internal units forming arbitrary cycles; W^{out} of size $L \times N$ - connects internal neurons to output units; and W^{back} of size $N \times L$ - represents the feedback connections from the output units to the reservoir.

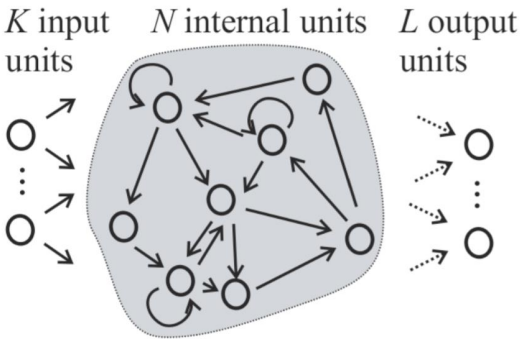

Figure 1: Representation of the echo state network architecture. Figure taken from [Jaeger, 2002].

The activation of the internal and output units, given by the vectors $x(n) = (x_1(n), \ldots, x_N(n))'$, respectively $y(n) = (y_1(n), \ldots, y_K(n))'$, are computed according to the following equations:

$$x(n+1) = f(W^{in}u(n+1) + Wx(n) + W^{back}y(n)), \tag{1}$$

$$y(n+1) = f^{out}(W^{out}(x(n+1))).$$

where $u(n) = (u_1(n), \ldots, u_K(n))'$ represents the input units, while f and f_{out} are the unit transfer functions. Common such functions are the identity $f(x) = x$, the logistic sigmoid or the $tanh$ function [Jaeger, 2016], the last two having the advantage that they induce non-linear behavior to the system and they limit the unit activation in the interval $[0, 1]$ - logistical sigmoid, respectively $[-1, 1]$ - $tanh$ function.

The aim of the training process is to produce the output weights W^{out} with minimal mean squared error. A good solution commonly used in practice is ridge regression (3), which ensures numerical stability and offers a convenient norm regularization scheme for coping with the bias-variance dilemma [Jaeger, 2017]. A summary of the training algorithm, described in detail in [Jaeger, 2010], is given below.

Training of an echo state network

A teacher input-output timed signal $(u(n), y_{target}(n))$, where $n \in \{1, ..., n_{max}\}$ is the discrete time step for n_{max} training data points, is given. Then a trained ESN, whose output $y(n)$ approximates $y_{target}(n)$ for input $u(n)$ and also generalizes well to new data, is desired as result of the algorithm.
Main steps:

1. Fix a large reservoir by choosing the number of input, internal and output units N, K, L and generate matrices W and W^{out} with small random values.
2. Considering the size of the training data n_{max} as the number of time steps, run the network with the teacher input and output, starting with a random initial network state $x(0)$. The activation is computed as in Equation (1). Consider a washout time n_0 (the necessary time for the network to "forget" the initial dynamics generated by the random initialization of the weights and units). Starting with time step n_0, collect the states of the computed reservoir units in a matrix M of size $N \times (n_{max} - n_0)$ and the target output in a matrix T of size $L \times (n_{max} - n_0)$.
3. Compute the output weights W^{out} which minimize the quadratic training error:

$$W^{out} = \underset{W \in R^{L \times N}}{argmin} \sum_{i=1}^{n_{max}-n_0} ||Wx(i) - y_{target}(i)||^2 \tag{2}$$

   One simple solution is to use linear regression:

$$W^{out} = TM^T(MM^T)^{-1},$$

   Another variant which usually gives more stable results is to use ridge regression (Tikhonov regularization):

$$W^{out} = TM^T(MM^T + \beta I)^{-1}. \tag{3}$$

   where I is the identity matrix and β is the regularization coefficient [Lukoševičius, 2012].

Completion of the above steps results in the computation of the output weights matrix W^{out} which can be used afterwards to execute the ESN for any input signal u to generate an approximation of the output signal y. Therefore, a trained echo state network is obtained, as required.

2.2 Control of Dynamical Systems

The problem of designing a controller for a helicopter reduces to the problem of controlling a very complex non-linear dynamical system. The following section will introduce general control theory terms and equations relevant for the researched model, as they are described in [Jordan, 1996].

For many dynamical systems, knowing the input at a given time is usually not sufficient to predict their output, but knowledge about the state variables is also necessary. This way,

given the state of a system and the input at a specific time step, the state at the next time step can be predicted. Then the output is simply represented as a function of the state. Generally, such a dynamical system can be defined by a set of equations, as explained in more detail in [Jordan, 1996], that describe the changes of the current state $x(n)$ and the output $y(n)$ at time step n:

$$x(n+1) = f(x(n), u(n)), \tag{4}$$

$$y(n) = g(x(n)). \tag{5}$$

where $u(n)$ represents the input, f is the next-state function and g the output function. Equations (4) and (5) can also be written together, as:

$$y(n+1) = h(x(n), u(n)).$$

where h is the composition of f and g.

Controlling a dynamical system, called the plant in the control theory context, can be described as computing an input to the system such that a desired output behavior will be obtained. One principal type of control system which is relevant for this research is the error-correcting feedback control, which continuously works on correcting the error between the actual output and the desired output at the current time step.

PID control

The proportional, integral, derivative controller (PID) is the most commonly used type of feedback controller, due to its simplicity, computational efficiency and applicability in a wide range of fields such as flight control, process control, instrumentation, automotive, motor drives etc. [Liu and Daley, 2001].

It controls the plant by continuously computing and using an error term $E(n)$ at each time step n, between a set-point $s(n)$, which is usually a desired state that the plant should achieve, and the current measured state of the plant at time n, $x(n)$. Then the equation of the error is given below:

$$E(n) = s(n) - x(n). \tag{6}$$

The error term is used to compute a control variable of the plant, used to achieve the desired state. The control variable $U(n)$ is usually represented in continuous form as in Equation (7) or in discrete form as in Equation (8), where K_P, K_I and K_D are the tunable parameters of the controller and Δt the time difference between two consecutive time points $n-1$ and n. The PID controller takes its name from the three terms seen in these equations. This way, $K_p E(n)$, the proportional term, reacts directly to the present value of the error to deliver an overall control effect. $K_I \int_0^n E(t)dt$, the integral term, reacts to past values of the error, having as main effect a reduction of the steady-state error for a constant set-point. Finally, $K_D \frac{dE(n)}{dn}$, the derivative term, accounts for probable future error, its effect being dependent on the current rate of change of the error [Li et al., 2006].

$$U(t) = K_P E(t) + K_I \int_0^t E(\tau)d\tau + K_D \frac{dE(t)}{dt}. \tag{7}$$

$$U(n) = U(n-1)+K_P[E(n)-E(n-1)]+K_I\Delta tE(n)+\frac{K_D}{\Delta t}[E(n)-2E(n-1)+E(n-2)] \quad (8)$$

Depending on the task to be accomplished, there are different objectives desired to be achieved by a PID controller, such as: set-point tracking, stability robustness, noise attenuation, robustness in plant and medium uncertainty, performance in rise-time, overshoot and settling time etc. Depending on these objectives, diverse tuning methods exist for the PID parameters, which can be divided in the following main groups, explained in more detail in Ang et al. [2005]: heuristic methods (developed with the help of Artificial Intelligence and manual tuning experience), analytical methods (parameters are computed from known relations between objectives and the plant model), optimization methods (parameters are calculated using numerical optimization techniques), frequency response methods (parameters are obtained using properties of the frequency of the process), adaptive tuning methods (which combine multiple of the previous methods in an automated online tuning).

2.3 Unmanned Aerial Vehicles

For the purpose of demonstrating whether ESNs are capable of modeling controllers for unmanned aerial vehicles (UAVs), only a simplified helicopter model will be used, without considering some external aerodynamic forces, such as those caused by wind and turbulence, and assuming rigid body dynamics. The principal characteristics of interest of the plant are instability and strong non-linearity. The following section will cover the main technical components and equations of the assumed plant, which follows closely the one described in Pathak and Agrawal [2005].

Firstly, as the main objective of this Guided Research is to present a controller for the rotation of the helicopter, the concept of a rotation matrix is introduced and will be used throughout the paper: A rotation matrix is a square, orthogonal matrix, with the property that when it is multiplied with any vector, the latter rotates while its length is preserved. Because of this property, it can be used to numerically represent any rotation of the axes about the origin [Diebel, 2006].

Considering $F = \{F_X, F_Y, F_Z\}$ an inertial referential, the rotation matrix of the helicopter is given by $R = [E_1^a, E_2^a, E_3^a]$, which also represents the helicopter's local referential frame A centered at the center of mass. If F_Z is fixed to point downwards and no pitch or roll are applied, then E_1^a is pointing towards the front of the helicopter, E_2^a towards its right and E_3^a in the same direction as F_Z, as it is illustrated in Figure 2.

The main rotor-blades' tip path plane (TPP) is defined as the longitudinal flapping angle Φ and the lateral flapping angle Θ. The two angles define the unit vector G normal to the TPP in the following way:

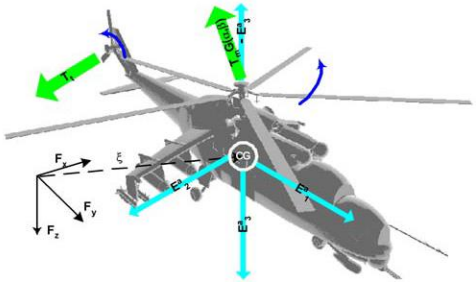

Figure 2: Helicopter coordinate system. Figure taken from Pathak and Agrawal [2005].

$$G(\Phi, \Theta) = \begin{pmatrix} -sin(\Phi) \\ sin(\Theta)cos(\Phi) \\ -cos(\Phi)cos(\Theta) \end{pmatrix} \tag{9}$$

The thrust forces of the two rotors are represented by T_m (10) for the main rotor and T_t (11) for the tail's rotor and their values ($|T_m|$ and $|T_t|$), together with the angle values Φ and Θ, are the four system inputs.

$$T_m = |T_m| G(\Phi, \Theta), \tag{10}$$

$$T_t = |T_t| E_2^a. \tag{11}$$

Consider the coordinates of the hub of the main rotor $l_m = (l_m^1, l_m^3, l_m^2,)$ and the hub of the tail rotor $l_t = (l_t^1, l_t^3, l_t^2,)$ relative to the referential A. Based on these coordinates, define the parameter matrices:

$$K = \begin{pmatrix} 0 & -l_m^3 & -l_t^3 \\ l_m^3 & 0 & 0 \\ -l_m^2 & l_m^1 & l_t^1 \end{pmatrix}, k_0 = \begin{pmatrix} l_m^2 \\ -l_m^1 \\ 0 \end{pmatrix}$$

At the two hubs there will exist reactive anti-torques generated by aerodynamic drag. The reactions of the two rotors are modeled by $Q_m \approx C_m^Q (T_m)^{1.5} + D_m^Q$ and $Q_t \approx C_t^Q (T_t)^{1.5} + D_t^Q$, where $C^Q s$ and $D^Q s$ are experimental parameters.

Finally, considering the symbols: $\xi \in \mathbb{R}^3$ - the Cartesian coordinates of the center of mass of the vehicle, m - the mass of the helicopter, I_m - the inertial matrix in the fixed frame A, Ω - the angular velocity with respect of F, $\breve{\Omega}$ - the cross-product skew-symmetric corresponding to Ω; the following system equations are obtained (more detailed derivations in Pathak and Agrawal [2005]):

$$\dot{\xi} = v, \tag{12}$$

$$m\dot{v} = R \begin{pmatrix} |T_m|G_1 \\ |T_m|G_2 + |T_t| \\ |T_m|G_3 \end{pmatrix} + mge_3, \tag{13}$$

$$\dot{R} = R\breve{\Omega}, \tag{14}$$

$$I_m\dot{\Omega} = -\breve{\Omega}I_m\Omega - |Q_m|G - |Q_t|e_2 + K \begin{pmatrix} |T_m|G_1 \\ |T_m|G_2 \\ |T_t| \end{pmatrix} + |T_m|G_3k_0. \tag{15}$$

The parameters and constants from Table 1 will be used for the helicopter model later in the simulation.

Table 1: Model parameters

Parameter	Value	Unit
m	4.9	Kg
g	9.81	ms^{-2}
$[I_{xx}, Iyy, Izz]$	$[0.142413, 0.271256, 0.271492]$	Kgm^2
$[l_m^1, l_m^2, l_m^3]$	$[0.015, 0, 0.2943]$	m
$[l_t^1, l_t^2, l_t^3]$	$[0.8715, 0, 0.1154]$	m
C_m^Q	0.004452	$m/\sqrt{N}$
C_t^Q	0.005066	$m/\sqrt{N}$
D_m^Q	0.6304	Nm
D_t^Q	0.008488	Nm
$max(T_m)$	$2mg$	N
$min(T_m)$	$0.5mg$	N
$max(T_t)$	$0.5mg$	N
$min(T_t)$	0	N
$max(\Phi)$	$15°$	$deg.$
$max(\Theta)$	$15°$	$deg.$

Constants and parameters of the helicopter plant model, taken from Pathak and Agrawal [2005].

Therefore, the model of the plant was obtained and it can now be used to generate data for the ESN, by operating on the 4 controllable values $(\Phi, \Theta, |T_m|, |T_t|)$ to achieve and maintain a desired rotation, given in angular values.

3 Main investigation

3.1 Task structure and design decisions

The following sections will describe the main steps which were conducted in order to address the presented research questions from Section 1.

At this initial stage, the main focus falls only on achieving rotational stabilization, which was agreed to be the principal and minimal requirement for the success of this Guided Research Thesis. Afterwards, achieving the tracking task and the altitude control extension will be built on top of the stabilization.

More concretely, the stabilization task can be described as follows: Given a fixed set-point for the rotation of the helicopter, the controller must generate values for the controllable input variables $T_m(t)$, $T_t(t)$, $\Phi(t)$ and $\Theta(t)$ such that the rotational state of the helicopter achieves and maintains the given set-point. That is, in terms of the motion equations previously presented, the rotation matrix of the helicopter should ideally reach the value of the rotation matrix obtained from the given set-point. Then, this state should be maintained constant and the angular velocity $\Omega = (\Omega_\alpha, \Omega_\beta, \Omega_\gamma)'$ should stay as close as possible to the value $(0, 0, 0)'$. As a clarification, it is important to note that the values and evolution of the other state variables of the plant, velocity v and position of the center of mass ξ, are not considered within this task and no requirements are addressed on them. For example, even if the helicopter gains a high velocity in a certain direction and changes location rapidly, as long as the rotation remains constant and close enough to the given set-point, the rotational stabilization is achieved.

However, rotation matrices are generally not very intuitive for the human mind regarding the rotational state of an object and it is also not always simple or straightforward to compare such matrices with each other. For this reason, it was decided to use Euler angles for all needed comparisons and to represent the given set-point as a vector $(\alpha, \beta, \gamma)'$, where α, β, γ are the values of the angles to be rotated in sequence about the F_X, F_Y, respectively F_Z axis of the inertial referential frame F, introduced in an earlier section and more clearly indicated in Figure 2. The main motivation behind this decision consists in the fact that Euler angles are one of the simplest and most intuitive ways capable of describing any rotation. As common in the aeronautics field, the Euler angles with sequence (1,2,3) were used, which are explained in Diebel [2006]. Generally, the angle values are represented in radians, but can easily be converted to degrees by the formula: $degrees = radians \cdot \frac{180}{\pi}$. To preserve uniqueness, these values must be kept in the following ranges: $\alpha \in (-\pi, \pi], \beta \in [-\frac{\pi}{2}, \frac{\pi}{2}], \gamma \in (-\pi, \pi]$ in radians i.e. $\alpha \in (-180°, 180°], \beta \in [-90°, 90°], \gamma \in (-180°, 180°]$ in degrees.

Multiple comparisons between the set-point and the states computed using the motion equations of the helicopter will be needed. As these equations use rotation matrices as representation for the rotational state, conversions from Euler angels to such corresponding matrices and the reverse are required. The conversion formulas, described in more detail in Diebel [2006], are given below, using the short notations $c_\alpha = cos(\alpha); s_\alpha = sin(\alpha); c_\beta = cos(\beta); s_\beta = sin(\beta); c_\gamma = cos(\gamma); s_\gamma = sin(\gamma)$:

$$R = \begin{pmatrix} c_\alpha c_\beta & c_\alpha s_\beta & -s_\alpha \\ s_\gamma s_\alpha c_\beta - c_\gamma s_\beta & s_\gamma s_\alpha s_\beta + c_\gamma c_\beta & c_\alpha s_\gamma \\ c_\gamma s_\alpha c_\beta + s_\gamma s_\beta & c_\gamma s_\alpha s_\beta - s_\gamma c_\beta & c_\alpha c_\gamma \end{pmatrix}$$

$$\begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} atan2(R_{23}, R_{33}) \\ -asin(R_{13}) \\ atan2(R_{12}, R_{11}) \end{pmatrix}$$

Now, computing the difference between the rotation matrix R of the helicopter at a given time and the set-point $(\alpha_{desired},\ \beta_{desired},\ \gamma_{desired})'$ becomes much simpler. If the difference in the referential frame F is needed, then R is converted into Euler angles $(\alpha_{current}, \beta_{current}, \gamma_{current})'$ and the vector difference is computed: $diff = (\alpha_{desired} - \alpha_{current}, \beta_{desired} - \beta_{current}, \gamma_{desired} - \gamma_{current})'$. Otherwise, if the difference relative to the helicopter's local referential frame A is needed (i.e. the rotation to be applied to the helicopter to reach the set-point considering the current rotation as $(0, 0, 0)'$), the set-point is firstly converted to a rotation matrix $R_{desired}$. This is then brought in frame A by left multiplying it with the transpose of the helicopter's rotation matrix R: $R_{desiredLocal} = R' R_{desired}$. After converting it back to Euler angles, the required local difference is obtained.

Then the stabilization task can finally be described as building a controller which generates values for the input variables such that this computed difference is minimized and kept to a minimum over a given time period.

The following task to be accomplished is rotation tracking which can be described similarly to the stabilization with the main difference that the given set-point does not necessary remain fixed, but it can change in time. Then the plant must be controlled such that its rotational state closely follows the track created by the evolving set-point. The previously described difference is computed at specific time points in the same way and the aim of the ESN controller is again to minimize it.

An additional task to be attempted in this thesis is altitude control, which consists of stabilization and tracking, but regarding the altitude of the helicopter. Then, while tracking a desired rotational state, the controlled helicopter should also maintain its vertical position $\xi(3)$ constant to a fixed altitude set-point relative to the start position, and later even track a changing given altitude.

3.2 Simulator

The first step of the project was to build a basic simulator for the helicopter model, in order to compute and visualize how the helicopter reacts to data. The main requirements of the simulator were to obtain a flexible framework which could accurately compute and visualize the state of the plant at each time step of a discrete-time, variable interval. All plant and environment related parameters and controllable variables needed to be easily configurable. A basic visual representation of the plant was also needed as a very intuitive way to interpret the data, specially in the early stages of the project.

Firstly, the state of the plant at each time step was computed using the motion equations described in the previous section, starting with the angular velocity, from Equation (15). The Euler method was used for discretization:

$$\Omega(n+1) = \Omega(n) + \Delta t\Bigg(- \breve{\Omega}(n) I_m \Omega(n) - |Q_m| G(n) - |Q_t| e_2 + K \begin{pmatrix} |T_m(n)|G_1(n) \\ |T_m(n)|G_2(n) \\ |T_t(n)| \end{pmatrix} + \\ + |T_m(n)| G_3(n) k_0 \begin{pmatrix} |T_m(n)|G_1(n) \\ |T_m(n)|G_2(n) \\ |T_t(n)| \end{pmatrix} + |T_m(n)| G_3(n) k_0 \Bigg) I_m^{-1},$$

where n is the current time step, Δt is the constant time difference between two consecutive time steps (value of $1/20$ sec was used), $\breve{\Omega}(n) = \begin{pmatrix} 0 & -\Omega_\gamma(n) & \Omega_\beta(n) \\ \Omega\gamma(n) & 0 & -\Omega_\alpha(n) \\ -\Omega_\beta(n) & \Omega_\alpha(n) & 0 \end{pmatrix}$ is the skew matrix corresponding to $\Omega(n)$, $G_1(n)$, $G_2(n)$, $G_1(n)$ are the rows of the matrix $G(n) = G(\Phi(n), \Theta(n))$ defined in Eq. (9) and $T_m(n)$, $T_t(n)$, $\Phi(n)$, $\Theta(n)$ are the values of the controllable input variables at time step n. The rest of the needed constants and parameters can be found in Table 1.

Having the angular velocity, the rotation matrix R can be computed:

$$R(n+1) = R(n) + \Delta t(R(n)\breve{\Omega}(n)),$$

However, due to the limited decimal precision of the digital computations, it is possible that even after a small number of iterations the obtained rotation matrix loses its property of orthogonality and becomes invalid. A simple way to detect such event is by computing the determinant of the matrix. If the module of the determinant is not exactly 1, then the rotation matrix is invalid. Early simulations with random input data showed that the presented situation is encountered in most cases and even if initially the error caused by the limited computational precision is very small (for example $det(R) = 1.00003$ instead of 1), it may increase and reach high values rapidly.

A simple solution to this problem is to apply Polar Decomposition (represented in Equations (16 - 18) on the computed rotation matrix at each iteration step and use the resulted orthogonal component U instead. This approach was chosen due to the following favorable properties of the Polar Decomposition: the factors are unique when R is invertible, the component U is the closest possible orthogonal matrix to R and its computation is simple and efficient. More detailed information about the Polar Decomposition, as well as advantages over other decomposition techniques with similar effects can be found in Shoemake and Duff [1992].

$$R = UP \tag{16}$$

$$P = \sqrt{R^* R} \tag{17}$$

$$U = RP^{-1} \tag{18}$$

After computing and ensuring the validity of the rotation matrix, the velocity and position of the helicopter can be obtained:

$$v(n+1) = v(n) + \Delta t\Bigg(R(n) \begin{pmatrix} |T_m(n)|G_1(n) \\ |T_m(n)|G_2(n) + |T_t(n)| \\ |T_m(n)|G_3(n) \end{pmatrix} + mge_3 \Bigg)/m,$$

$$\xi(n+1) = \xi(n) + \Delta t v(n),$$

Reasonable assumptions regarding the initial states are made: $\xi(0) = (0,0,0)'$, $v(0) = (0,0,0)'$, $\Omega(0) = (0,0,0)'$, $R(0)$ will be computed from the initial given rotation.

After having a framework capable of accurately computing the state of the plant, a non-complex, real-time visual representation of the model was also required in order to gain a very intuitive way of monitoring how the state of the helicopter reacts to the given data. This part was accomplished using built-in Matlab tools. A dummy helicopter model was built from basic geometric figures, as can be seen in Figure 3. Then at each time step the object was redrawn to match the current, computed state of the plant, thus creating an animation.

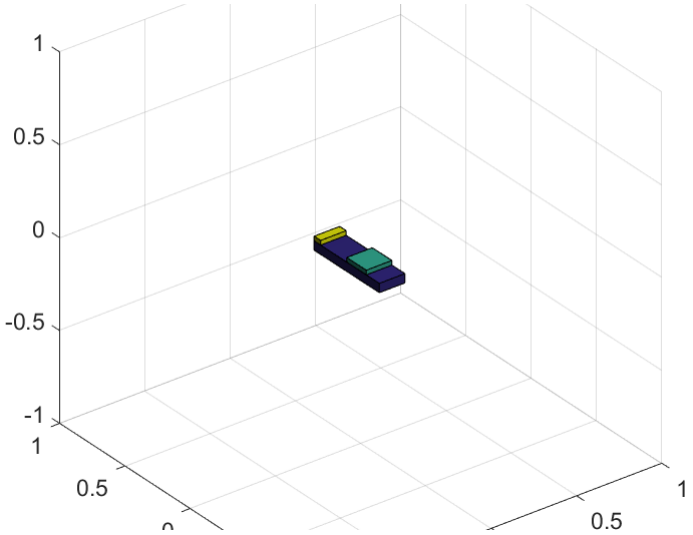

Figure 3: Helicopter simulator with dummy object

3.3 Data generation

The next challenge of the project was the generation of the training data for the helicopter. A big amount of the project's work was allocated into this critical task, as the quality and quantity of the training data have a significant impact on the behavior of the neural network, on the correctness and accuracy of the results produced by it.

Therefore, many requirements and properties need to be considered. Among these, of higher importance is the diversity of the data, which needs to cover highly varied input configurations in order for the controller to learn how to behave in all situations that it may encounter. Another good characteristic is the inclusion of various amounts of noise, which would teach the controller to better generalize on new test data and to also behave properly when facing noise.

The main objective when considering data generation is to sample reactions of the plant to control input in all orientations, such that later the network can be trained to under-

stand the dynamics of the plant as well as how the evolution of its state depends on the controllable variables. For this purpose, the teaching data to be generated is structured into multiple time signals, representing the controllable variables and some components of the plant states at different time steps.

More concretely, the first teaching component consists of specific $(\alpha, \beta, \gamma)'$ Euler angles, representing the rotational evolution of the helicopter in a given time interval. Another component which is relevant in understanding the plant dynamics is the evolution of its angular velocity $(\Omega_\alpha, \Omega_\beta, \Omega_\gamma)'$. Finally, the network needs to learn how these two components depend on the four helicopter controls $(\Phi, \Theta, |T_m|, |T_t|)'$, the last component of the teaching data.

Training data with the presented structure and requirements is not straightforward to acquire. A basic approach would be to submit the plant to random noisy control inputs and record its dynamical reactions. However, due to the complex dynamics of the plant, this method would result in the recording of only a small subset of the possible helicopter poses, which is later insufficient for training the network. Therefore, a method which ensures that the helicopter visits and stabilizes in a large set of different rotations is needed.

By first having such a large and varied set of desired poses $(\alpha, \beta, \gamma)'$, a controller could compute values for the input variables $(\Phi, \Theta, |T_m|, |T_t|)'$ such that the given poses are visited by the helicopter with a certain degree of precision. Then the control input variables computed by the controller and the rotations and angular velocities obtained by feeding this input to the plant, can be used to form the required training data.

For this purpose, a PID controller, with the structure presented in Section 2.2 of this thesis, was chosen, due to its simplicity and computational efficiency. While PID control can achieve very good results for diverse linear systems, it generally achieves sub-optimal outcomes when used to control such complex and non-linear plants as a helicopter. However, after implementing and using a basic PID controller on a small manually generated data set, the initial results were considered acceptable for this type of task.

Then, the first step is to acquire the initial set of desired poses, which is to be used by the PID controller to generate training data. For achieving the first main objective, stabilization in hovering mode, the set-point rotation signal should remain constant for a certain number of time steps, so the network can learn how to keep the current rotation stable in time, before changing to a new pose. For the second objective, trajectory tracking, more complex data configurations may be required, which involve following of diverse, achievable rotational tracks.

Then it can be seen that the generation of the training data can be divided into two parts:

1. getting a signal of set-point angles for the PID controller, which needs to be large and varied such that it contains many different poses to be visited by the helicopter. This signal will not be part of the final teaching data;
2. computing the corresponding controllable variables using the PID controller and the

actual states of the plant using the simulator.

The first choice in generating the set-point poses was to consider all combinations of the three Euler angles (α, β, γ) when increasing each angle by certain step variables: $step_\alpha$, $step_\beta$, $step_\gamma$. For achieving all these combinations, the generation was implemented as a triple nested loop, with the γ component completing its range before any change in β, and then the β component completing its range before any change of α. As previously discussed, each combination of angles would remain constant for a certain number of time steps, $stabilization\_time$, before incrementing any of the variables. However, it soon became clear that this method would result in obtaining a data set too large to be computed and used in training in a reasonable amount of time with the available computation power. For example, with increment step variables of 1° for each angle on the complete ranges and $stabilization\_time$ variable of only 20 time steps (i.e. 1 sec.), a discrete signal of $360 \cdot 180 \cdot 360 \cdot 20 = 466,560,000$ points would be obtained. Therefore, certain simplifications needed to be made, as a trade-off between data coverage and computation time.

Firstly, it was considered that having training data for visiting only limited angle ranges is enough for demonstration purposes, with the mention that these can easily be changed or extended as required by regenerating the signal. Then, the decided final ranges which were used for the angles α, β and γ are $[-30^\circ, 30^\circ]$, $[-30^\circ, 30^\circ]$, respectively the complete range of $(-180^\circ, 180^\circ]$. Secondly, while the step variables $step_\alpha$ and $step_\beta$ were set to 1, $step_\gamma$ was fixed to the value of 3, to compensate for its larger range. A third simplification made was to increment the value of α and β angles without waiting for the complete covering of the range of γ, i.e. the value of β would advance after only 10 increments of γ, instead of $\frac{360}{3} = 120$, but without resetting γ, so the range of 360° would still be covered eventually.

Finally, to ensure the continuity of the generated poses as well as the possibility of the controller to learn to change each angle in both directions, the sign of the step variables was also changed for each angle after the covering of the complete range of the respective component. To this end it was required for the α angle to cover its range twice, firstly from -30° increasing to 30°, then the reverse. Using this algorithm, a data set of $(60 + 60) \cdot 60 \cdot 10 \cdot 20 = 1,440,000$ points was obtained. The generation could be done in a reasonable amount of time.

The advantage of the training data that can be obtained using this set of desired poses is not only that it contains data varied enough for stabilization, as each of the obtained poses remains constant for 20 time steps, but it can also be used for the tracking objective, as there are no abrupt pose changes between consecutive points, so the whole data set can be considered as an achievable track.

When also considering the extended task of altitude control, additional set-point data needs to be added, also with the purpose to be used by a PID controller in computing the final teaching signal: For all obtained angle combinations, an altitude component is considered, which varies for example in the range of $(0, 5)$ meters relative to the initial position. If a step of 1 meter is chosen for both increasing and decreasing altitude in the given range, then the data set increases by a factor of 10. However, even more data may

be needed for achieving acceptable results. In a later step of the project, this task was also attempted, but the altitude component was modified in the same time as the other components, to avoid an increase in data size. Unfortunately, this simplification reduced the quality of the data and it is believed to be the main reason why a more favorable outcome could not be obtained.

Having the desired poses, the PID controller can finally be used to compute values for the control variables such that the helicopter closely follows the given rotations. However, it is important to note that, although the T_m variable has an effect on the rotation of the plant about the F_Z axis, its main purpose is to control the altitude, which is not considered in this part of the project. Then, if this is kept fixed, the control variables Φ, Θ, T_t are sufficient to control the rotations about the F_X, F_Y, respectively F_Z axis. Based on these observations, another simplification was made: The T_m variable will remain fixed to the value of $m \cdot g$ and only the remaining variables will be computed using the PID. $m \cdot g$ is the value for which the helicopter should remain at altitude approximate 0 relative to the initial position when stabilized to rotation $(0°, 0°, 0°)'$. Any other value from the valid range of T_m, found in Table 1 can be used. This simplification together with the ones previously described in this section can be made as they will not ultimately affect the purpose of demonstrating the capability of an ESN to control the rotation of the plant.

Therefore, a PID controller is implemented. At each time step n, the error $E(n)$ from Equation 6 is computed as the difference between the set-point i.e. the input angles $(\alpha(n), \beta(n), \gamma(n))$ and the current rotation of the helicopter, found using the discrete motion equations from Section 3.2. This difference is obtained with the method described in Section 3.1. Then, based on $E(n)$, and by choosing suitable K_P, K_I and K_D parameters, the PID error components can be computed and output values for the three controllable variables are acquired.

As the PID parameters need to be tuned for obtaining more accurate results, a manual tuning was done in the following way: K_D was varied in order to minimize the rise-time necessary for achieving the desired angles, then K_I for eliminating the steady-state error and finally K_D in order to damp the overshoot i.e. the value of the output exceeding the steady-state value. Using this process, the parameters obtained are: $K_P = 10, K_I = 10, K_D = 4$.

Having the PID controller implemented and tuned, the training data can be computed. At each step, a random noise component is added to the controllable variables, with the maximum absolute value equal to 30% of the maximum valid value of the signal (i.e. $30\% \cdot 15 = 4$ for Φ and Θ and $30\% \cdot 0.5 \cdot m \cdot g \approx 7.21$ for T_t). The noise will later be essential for the ESN to probe the dynamics of the plant. Then the obtained values need to be restricted to their valid ranges, found in Table 1. These control variables are fed to the simulator and the obtained rotation and angular velocity are recorded. The rotation and the next desired pose are then used in the following iteration to compute the PID error. Using this process on the entire set of set-point angles, the three necessary signal components are acquired and the training data generation is completed.

When also considering altitude control, another PID controller is used in the exact same

way to generate values for T_m variable such that the helicopter follows the given altitude set-point signal. These values are simply added to the teaching data.

3.4 Training the ESN

A key idea behind the controller architecture to be presented introduces the concept of an input time delay d between different signals fed to the network. The main components of the controller are the input, feedback and output signals. During training, the current pose at time n, given by the Euler angles previously computed with the PID $(\alpha(n), \beta(n), \gamma(n))$, is given as input. However, the past rotation and angular velocity computed d time steps before are given as feedback: $(\alpha(n-d), \beta(n-d), \gamma(n-d))$ and $(\Omega_{alpha}(n-d), \Omega_{\beta}(n-d), \Omega_{\gamma}(n-d))$. Then, the past controllable variables $\Phi(n-d), \Theta(n-d), T_t(n-d)$, also computed by the PID controller are given as output. This way, given a previous state of the plant, the network learns how the current state depends on a previous output. This process is also represented in Figure 4.

In the exploitation phase this translates to tracking: knowing the current pose and angular velocity at time n as feedback from the plant and being given a desired rotation to be achieved at a future time step $n+d$, the network will compute the current output i.e. the controllable variables at time n such that the input state will be achieved. For this purpose, the time delay d needs to be carefully chosen: if it is too large, during training the output and feedback signals may go outside the time interval in which the network can still 'remember' the past state so the controller may not be able to correlate well the present input with the past components. If it is too small, the controller may not have enough time during exploitation to bring the plant from the current state to the desired state and accuracy may be affected.

The next step consists of the initialization and training of an echo state network, using the generated data. The initialization involves deciding on a reservoir size, fixing specific parameters and system constants (such as scalings, regression regularization coefficient, input time delay, fixed gap between consecutive time steps etc.) and generating initial unit activation and weight values. The training part will closely follow the algorithm presented in Section 2.1 while also introducing the time-delay idea described in the beginning of the section. The representation of this process applied on the helicopter model can be seen in Figure 4 and it is described as follows.

The training signal consisting of the present rotation, $(\alpha(n), \beta(n), \gamma(n))$, should be given as input to the reservoir. However, the representation of rotations in Euler angles is not continuous. For example the difference in α component between $180°$ and $-179°$ can be achieved by a rotation of only $1°$, but the numerical difference is of $180-(-179)=359$ units. Because it may be difficult for the ESN to accommodate for this discontinuity, the angles are firstly converted to sine and cosine values, which are continuous and thus solve the presented problem, without lacking any of the Euler angles properties useful for the network. This way a signal of the form $input(n) = (sin(\alpha(n)), cos(\alpha(n)), sin(\beta(n)), cos(\beta(n)), sin(\gamma(n)), cos(\gamma(n)))$ is obtained and given as input to the ESN instead.

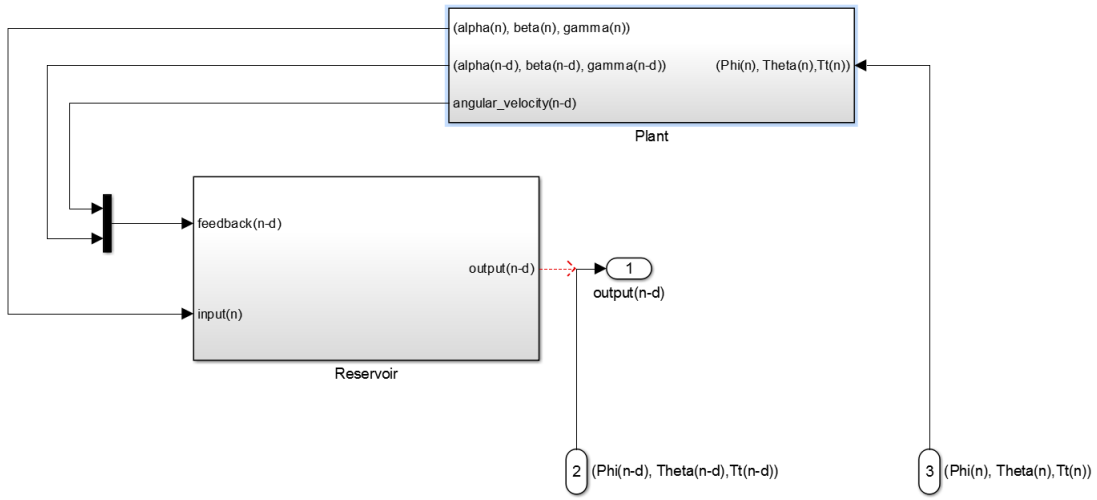

Figure 4: Training of the ESN

The past rotation of the helicopter $(\alpha(n-d), \beta(n-d), \gamma(n-d))$ is also converted into sine and cosine values and given to the reservoir as a $feedback(n-d)$ signal from the plant together with the past angular velocity of the helicopter $(\Omega_\alpha(n-d), \Omega_\beta(n-d), \Omega_\gamma(n-d))$. This is also relevant information about the rotational evolution of the helicopter and it is given to the network to facilitate its "understanding" regarding the dynamics of the plant. A washout size of 100 time steps is used in order to achieve the echo state and "forget" the initial state of the network. The state of the neurons is stored in a matrix M of size: reservoir size $\times$ (training size - washout size - input delay) i.e. $100 \times 1,439,895$ and the output controllable variables are stored in a vector T of size: $3\times$ (training size - washout size - input delay) i.e. $3 \times 1,439,895$.

Then, the activation of the neurons at each step is obtained by the following equation:

$$x(n+1) = tanh(scale_{Win} \cdot W^{in} \cdot input(n) + scale_{Wback} \cdot W^{back} \cdot feedback(n-d) \\ + scale_W \cdot W \cdot x(n) + scale_{Bias} \cdot bias).$$

In Figure 5 the activation of 5 random neurons is shown, on different time intervals with increasing size. The proximity of the signal to the values 1 and -1 is a proof of the non-linear behavior of the network, induced by the $tanh$ activation function.

To finalize the training, the output weights W^{out} are obtained using ridge regression (Equation 3). This way, the network observes the plant state at time points $t-d$ and t and the output at $t-d$ and learns how the current state depends on the past state and output.

After training, the neural network is ready to be used on new test data. The algorithm is described as follows: The neurons activation is computed using the same formula as in training, where the input signal is given by the desired set-point angles $input(n) = (\alpha_{desired}(n+d), \beta_{desired}(n+d), \gamma_{desired}(n+d))'$, to be achieved in d time steps, represented

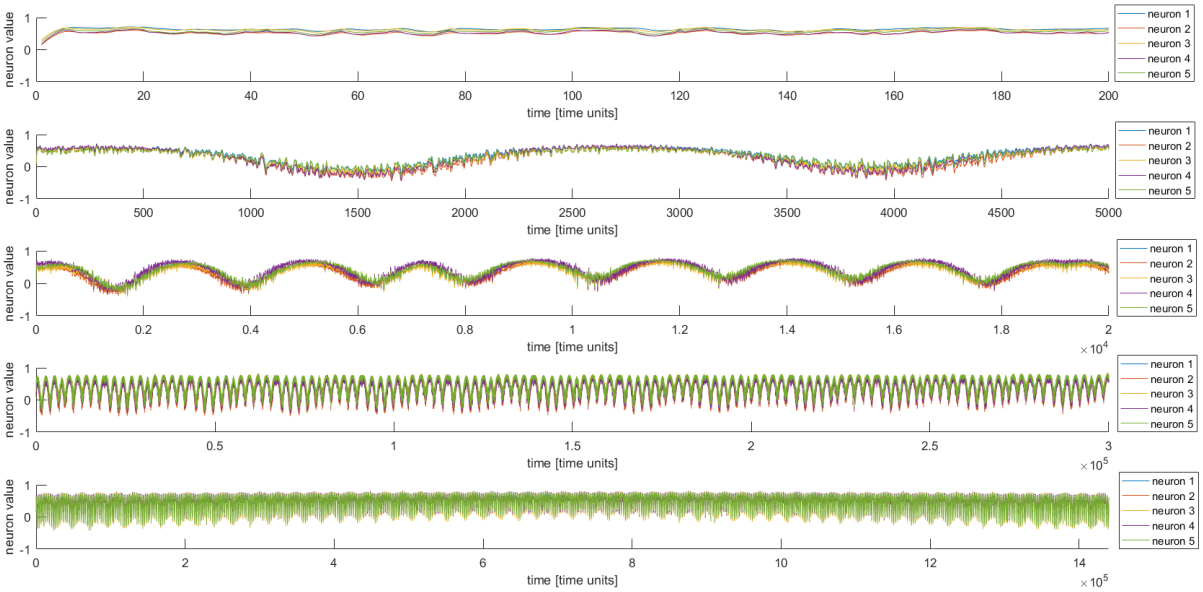

Figure 5: Activation of 5 random neurons on intervals with different sizes

in sine and cosine values. The feedback signal is now given by the current rotation of the helicopter (in sine and cosine values) together with the current angular velocity. Then the values of the controllable variables are computed at each step n by the formula: $(\Phi(n), \Theta(n), T_t(n)) = W^{out\prime} \cdot neurons(n)$

Some initial basic test cases show that the controller stabilizes the rotation, but with a certain steady state error between the stabilized angles and the given set-point. This problem is easily overcome by computing the integral error between the previous desired input state which was aimed to be achieved at the current time step i.e. $input(n-d)$ and the current actual pose computed by the plant using the controller output. This integral error is then added to the desired set-point fed to the network, as shown in Equations 19 - 21, where $integral(0) = 0$ and Δt is the time gap between consecutive time steps, in this case with the value of $0.05s$. This method effectively diminishes the observed steady state error.

$$E(n) = input(n-d) - (\alpha(n), \beta(n), \gamma(n))', \tag{19}$$

$$integral(n) = integral(n-1) + E(n) * \Delta t, \tag{20}$$

$$input(n) = input(n) + integral(n). \tag{21}$$

After multiple runs on various testing data sets with different network configurations regarding scalings and parameters, the following configuration seen in Table 2 is observed to give good results:

Table 2: Network parameters and scalings

Parameter	Value
$reservoir\_size$	100
d	5
$scale_W$	0.015
$scale_{Win}$	0.1
$scale_{Wback}$	0.1
$scale_{Bias}$	0.001
$regression\_regularizer$	0.01

Constants, parameters and scalings used in configuring the ESN.

Although the focus of the project was on the stabilization task, the training and execution processes bring no differences when used for the tracking objective. Because of the flexibility of the constructed training data which can also be considered as trajectory data, the same obtained ESN, with eventual different parameter configurations, can be used within the tracking task. When considering altitude control, the algorithms are similar, with the exception that a desired altitude is added to the input signal, while a current altitude is added to the feedback signal. The controller output would also contain an additional component for T_m variable, which should not remain fixed anymore.

4 Results analysis

The main evaluation criteria used to measure the success of the controller is the mean squared error (MSE) of each of the three angle components between a desired pose and the computed rotation over a certain time interval. The structure of the test sets to be presented is briefly introduce here, to avoid repetition and to keep the reporting of the results in a simple, concise format. Then, for each testing session, the following are reported consistently: the values of the average, maximum and minimum MSE obtained over the respective set in the form $(MSE_\alpha, MSE_\beta, MSE_\gamma)'$ and their corresponding given setpoints $(\alpha, \beta, \gamma)'$ are included. When relevant, a visual representation of the case with maximum MSE, i.e. the worst case scenario of the respective testing session, is also presented. Additional interesting plots are also shown, depending on the case.

Some tests will refer to a random noise or a constant perturbation component which could be caused in reality by unforeseen environment conditions, such as strong wind, air turbulence, inaccurate or malfunctioning sensors etc. This is represented by a vector of values $(noise_\Phi, noise_\Theta, noise_{T_t})'$ added to the ESN output. For stabilization, the random noise has positive or negative values, randomly chosen from the ranges $[-20\%, 20\%]$ of the maximum possible value of the respective variable. Concretely, the noise ranges are: $[-3, 3]$ for Φ and Θ and $[-4.8, 4.8]$ for T_t. When referring to stabilization constant perturbation, the following is used: $(3, -3, 4.8)$. In the context of trajectory tracking, the random noise is chosen in the same way, but the range $[-20\%, 20\%]$ is reduced to $[-15\%, 15\%]$ i.e. $[-2.25, 2.25]$ for Φ and Θ and $[-3.6, 3.6]$ for T_t.

4.1 Stabilization

Testing session 1: All combinations of angles $(\alpha, \beta, \gamma)'$ within the ranges $[-30°, 30°]$, $[-30°, 30°]$, $[0°, 180°]$ with a step of $2°$ on α and β and $5°$ on γ, are given as desired input over a period of 200 time steps (i.e. 10 seconds) to the controller in separate tests.

a) No noise:

- Average MSE = $(0.0029°, 0.0008°, 0.0030°)'$;
- Minimum MSE = $1.0e{-}04(0.8812°, 0.8027°, 0.9906°)'$ for desired angles $(-22°, -30°, 25°)'$;
- Maximum MSE = $(0.0011°, 0.0073°, 0.0309°)'$ for desired angles $(-30°, -30°, 165°)'$, represented in Figure 6.

A certain initial number of time steps is needed until convergence is reached, afterwards the signal is stabilized very accurately.

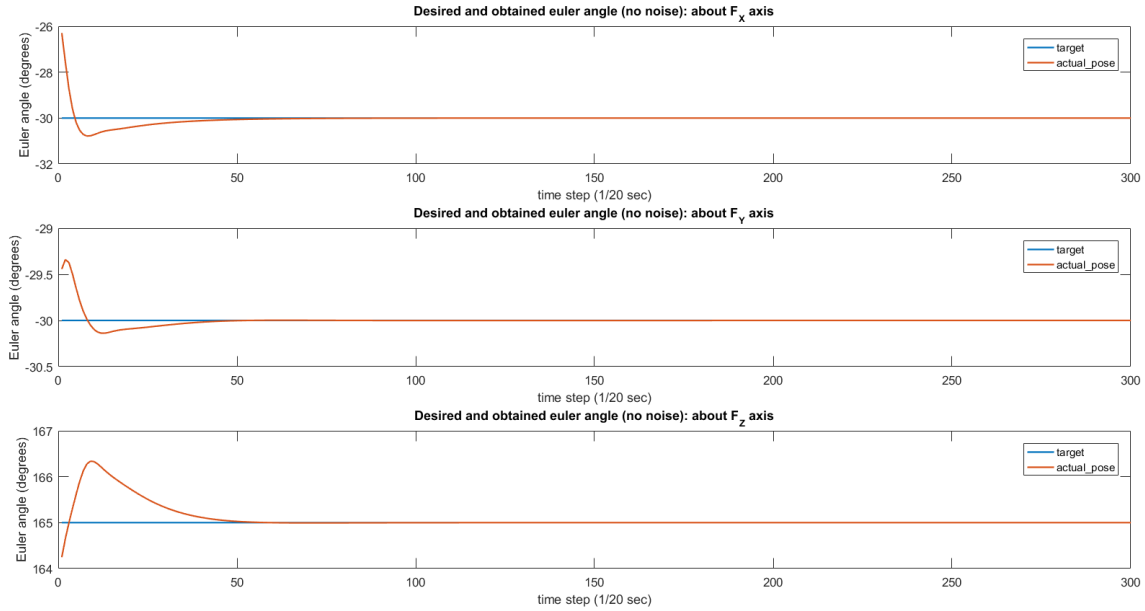

Figure 6: Test 1 a) Desired and obtained rotations, with no noise

b) Random noise:

- Average MSE = $(0.5412°, 0.0971°, 0.1099°)'$;
- Minimum MSE = $(0.1558°, 0.0309°, 0.0672°)'$ for desired angles $(4°, -10°, 180°)'$;
- Maximum MSE = $(6.6140°, 1.6851°, 0.9011°)'$ for desired angles $(30°, 28°, 45°)'$, represented in Figure 7. The controllable variables including the noise components are also represented in Figure 8.

The errors are increased, however the obtained signal still follows the set-point signal reliably.

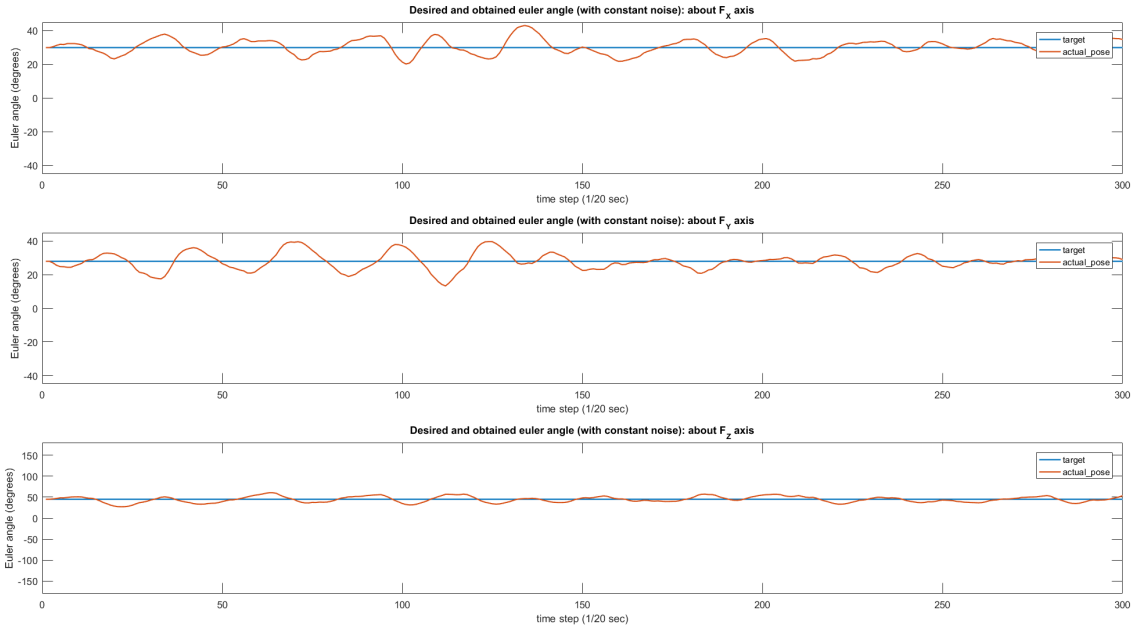

Figure 7: Test 1 b) Desired and obtained rotations, with random noise

c) Constant perturbation:

- Average MSE = $(0.3662°, 0.0840°, 0.1007°)'$;
- Minimum MSE = $(0.1663°, 0.0361°, 0.0953°)'$ for desired angles $(0°, -30°, 155°)'$;
- Maximum MSE = $(0.8432°, 0.1164°, 0.1824°)'$ for desired angles $(-30°, -30°, 30°)'$, represented in Figure 9.

By observing multiple plots, it can be concluded that in the case of constant noise, the initial time of arriving at steady state is longer, but then the network learns to adapt very well to the noise component and achieves very accurate stabilization.

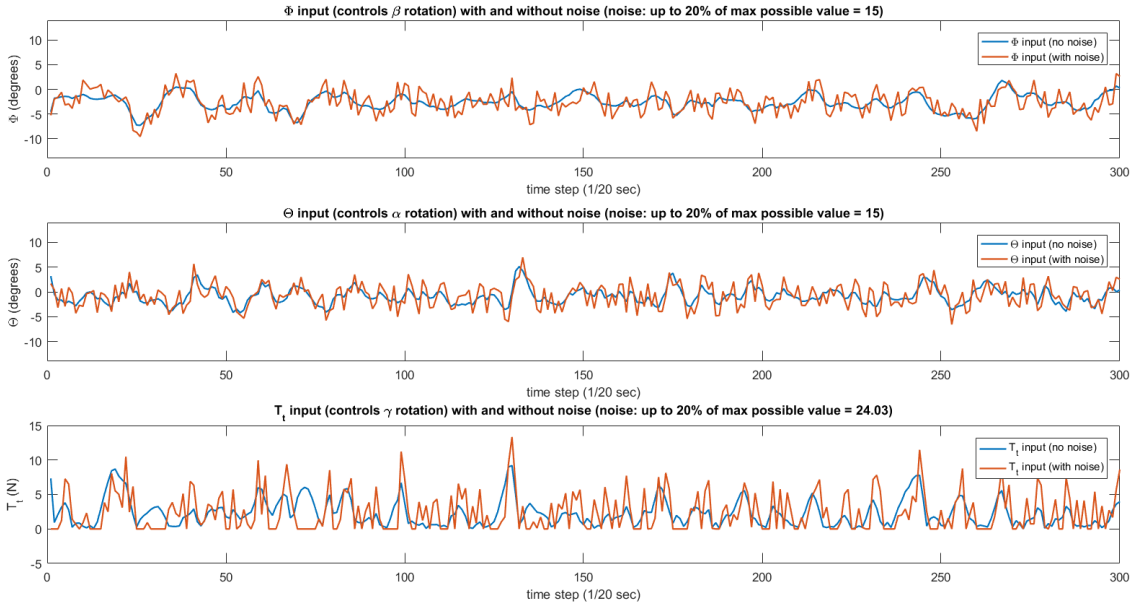

Figure 8: Test 1 b) Controllable variables with and without noise

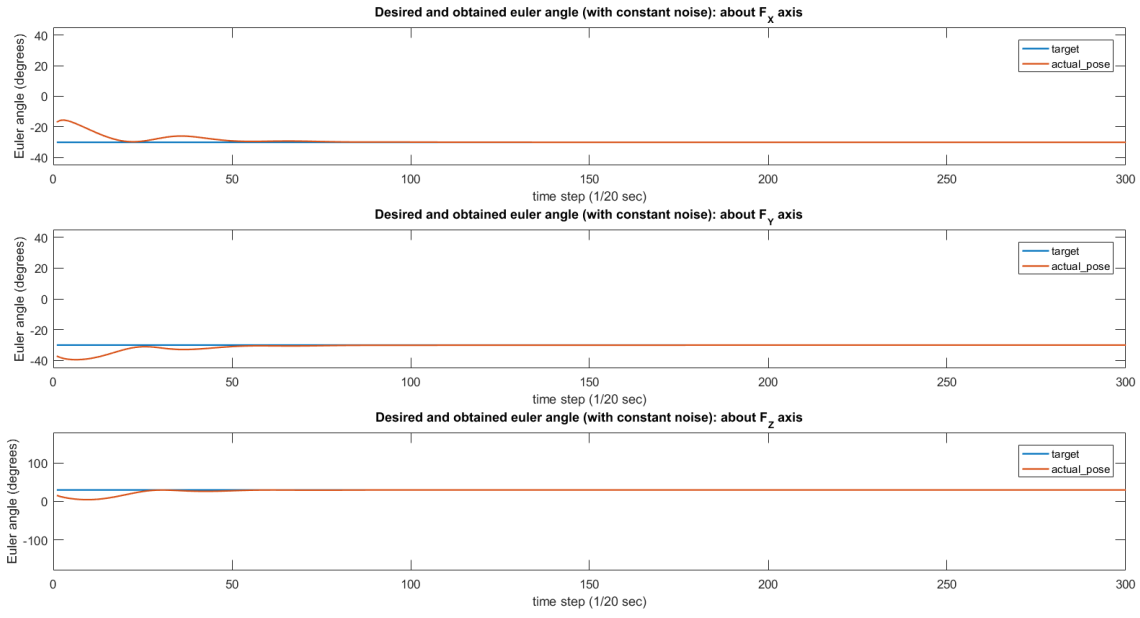

Figure 9: Test 1 c) Desired and obtained rotations, with constant noise

The very low errors obtained prove that the ESN-controller can successfully stabilize the rotation of a helicopter model in any desired given pose, chosen from a wide range, even when faced with data containing strong noise. A few additional tests were conducted having the desired components α and β outside the trained range of $[-40°, 40°]$ and accurate results were still obtained. For example, the tests from session 1 a) obtained MSE of $(0.0212°, 0.0271°, 0.0241°)'$ for the desired pose $(60°, -55°, 0°)'$ which is relatively

distant from the trained range of poses. Although larger than the computed average, this error still suggests a very precise result. This shows that the ESN controller has the capability of generalizing well on new data even on complex, non-linear systems such as a helicopter model. Therefore, the stabilization objective of the thesis is achieved with success.

4.2 Tracking

Regarding the rotation tracking task, MSE is again used as the main evaluation criteria, with the mention that this method can be accurate only when the given testing tracks are achievable for the helicopter, i.e. without very abrupt changes in too short periods of time. For this purpose, in any tests containing multiple angular coordinates to be reached by the helicopter in a given time interval, an interpolation will be computed between any two consecutive desired states on the available time points such that a smooth track will be created and given as input to the ESN instead, on the specific time interval.

Initially, the controller is given only basic tests, such that the given track requires change in only one of the angular components:

Testing session 2.1: This set of tests only requires a complete rotation of 360° about F_Z from starting angle 0°, in a time interval of 400 time steps (i.e. 20 seconds), followed by stabilization for 100 steps (i.e. 5 seconds). The desired angles for the α and β components remain fixed along the track, and all combinations of (α, β) angles are given in different tests, from the ranges $[0^\circ, 30^\circ]$, respectively $[0^\circ, 30^\circ]$, with a step of 2°.

a) No noise:

- Average MSE = $(0.0140^\circ, 0.0011^\circ, 0.0019^\circ)'$;
- Minimum MSE = $(0.0080^\circ, 0.0009^\circ, 0.0026^\circ)'$ for desired angles $\alpha = -20^\circ; \beta = 20^\circ$;
- Maximum MSE = $(0.0285^\circ, 0.0049^\circ, 0.0061^\circ)'$ for desired angles $\alpha = -30^\circ; \beta = -30^\circ$, represented in Figure 10.

The input trajectory signal is accurately followed.

b) Random noise:

- Average MSE = $(0.3367^\circ, 0.0648^\circ, 0.0726^\circ)'$;
- Minimum MSE = $(0.1879^\circ, 0.0221^\circ, 0.0476^\circ)'$ for desired angles $\alpha = -4^\circ; \beta = -2^\circ$;
- Maximum MSE = $(0.7788^\circ, 0.2249^\circ, 0.2692^\circ)'$ for desired angles $\alpha = 28^\circ; \beta = -28^\circ$, represented in Figure 11.

Tracking is achieved reliable, however with increased error.

Testing session 2.2: This testing session is done in a similar manner, with the difference that this time the β and γ angles remain fixed and taken from the ranges $[0^\circ, 30^\circ]$, respectively $[0^\circ, 180^\circ]$ with steps of 2° and 5°. The given track requires a rotation about F_X axis

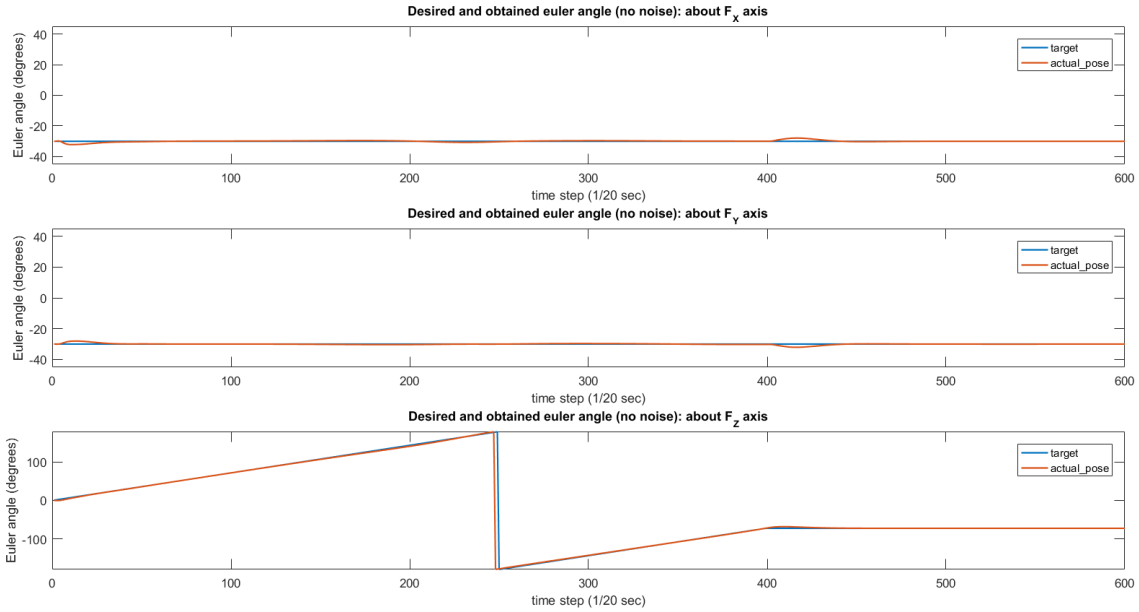

Figure 10: Test 2.1 a) Desired and obtained rotations, with no noise

from initial angle $30°$ to final angle $-15°$, in an interval of 100 time steps (i.e. 5 seconds), followed by stabilization for another 100 steps.

a) No noise:

- Average MSE = $(0.0047°, 0.0028°, 0.0154°)'$;
- Minimum MSE = $(0.0014°, 0.0007,° 0.0092°)'$ for desired angles $\beta = 16°; \gamma = 130°$;
- Maximum MSE = $(0.0213°, 0.0121°, 0.0320°)'$ for desired angles $\beta = -30°; \gamma = 180°$.

b) Random noise:

- Average MSE = $(0.3408°, 0.0555°, 0.0843°,'$;
- Minimum MSE = $(0.1486°, 0.0271°, 0.0528°)'$ for desired angles $\beta = -6°; \gamma = 155°$;
- Maximum MSE = $(0.8697°, 0.0930°, 0.2002°)'$ for desired angles $\beta = 30°; \gamma = 30°$.

Testing session 2.3: Another testing session is done by keeping the α and γ angles fixed and taken from the ranges $[0°, 30°]$, respectively $[0°, 180°]$ with steps of $2°$ and $5°$. The given track requires a rotation about F_Y axis from initial angle $-30°$ to final angle $30°$, in an interval of 100 time steps (i.e. 5 seconds), followed by stabilization for another 100 steps.

a) No noise:

- Average MSE = $(0.0085°, 0.0279°, 0.0029°)'$;

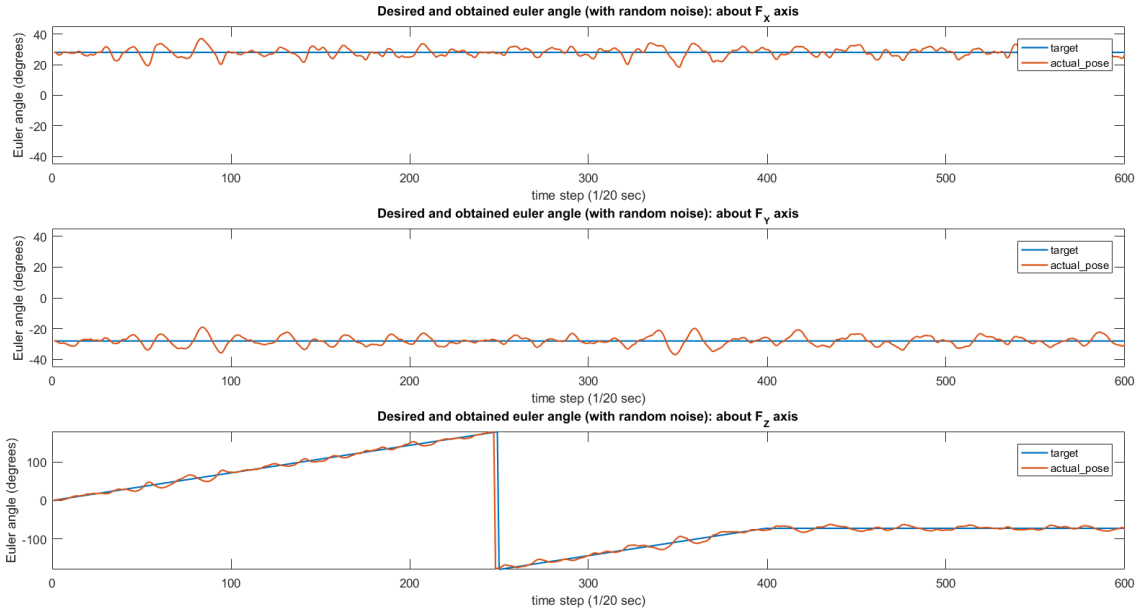

Figure 11: Test 2.1 b) Desired and obtained rotations, with random noise

- Minimum MSE = $(0.0031°, 0.0235°, 0.0019°)'$ for desired angles $\alpha = 30°; \gamma = 115°$;
- Maximum MSE = $(0.0423°, 0.0245°, 0.0106°)'$ for desired angles $\alpha = -30°; \gamma = 170°$.

b) Random noise:

- Average MSE = $(0.3851°, 0.1121°, 0.0877°)'$;
- Minimum MSE = $(0.1610°, 0.0355°, 0.0523°)'$ for desired angles $\alpha = -4°; \gamma = 165°$;
- Maximum MSE = $(2.1152°, 0.4411°, 0.1791°,)'$ for desired angles $\alpha = -30°; \gamma = 20°$,

The results obtained are comparable with those from sessions 2.1 and 2.2.

Testing session 2.4: A more interesting set of tests is when the given tracks require changes in multiple angular components instead of only one. For this purpose two random poses are chosen, with the components α, β, γ from the ranges $[-180°, 180°]$, $[-30°, 30°]$, respectively $[-30°, 30°]$, as the initial state and the final, desired state. Stabilization of 50 time steps is firstly desired in the initial state. Then a track between the two is constructed by linear interpolation, within an interval of 100 time steps, followed by stabilization of another 100 steps. The test is repeated 500 times, with new random initial and final poses.

a) No noise:

- Average MSE = $(0.1102°, 0.0133°, 0.0138°,)'$;
- Minimum MSE = $1.0e{-}03(0.1234°, 0.5201°, 0.2733°)'$ for initial pose $(-16°, -14°, -17°)'$

and final pose $(-16°, -6°, -24°)$;

- Maximum MSE = $(1.0876°, 0.0491°, 0.0253°)'$ for initial pose $(13°, -9°, -45°)'$ and final pose $(7°, 21°, 160°)$, represented in Figure 12.

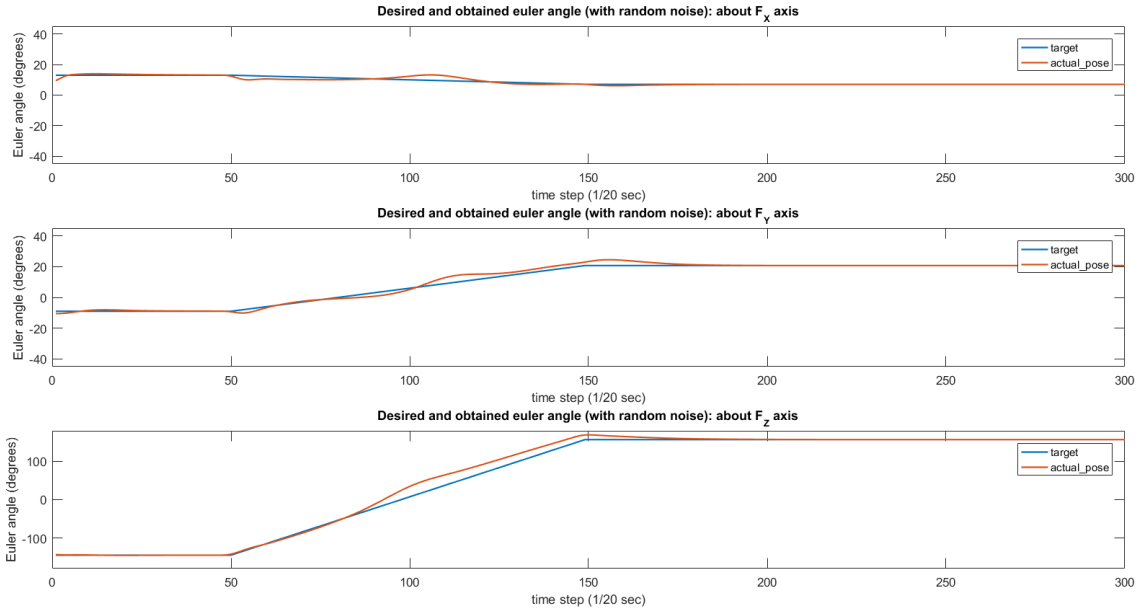

Figure 12: Test 2.4 a) Desired and obtained rotations, with no noise

b) Random noise:

- Average MSE = $(0.4387°, 0.0702°, 0.0767°)'$;
- Minimum MSE = $(0.1410°, 0.0338°, 0.0520°)'$ for initial pose $(-26°, 8°, 20°)'$ and final pose $(-22°, -2°, 11°)$;
- Maximum MSE = $(2.3599°, 0.4895°, 0.1845°)'$ for initial pose $(16°, -15°, -167°)'$ and final pose $(29°, 26°, 81°)$, represented in Figure 13.

More complex tests which follow trajectories described by multiple given desired states were not conducted in this project. This is because it is considered that adding more such desired poses at specific time points is just an extension of the previously presented test session which would not bring more insight in the capability of the controller. That is, knowing that a trajectory described by two states can be tracked, adding a third state would only repeat the experiment with the last two states as new initial and final. Inductively, adding any number of new desired states is equivalent with a conducting multiple repetitions of the original experiment.

Although some tests achieved decreased accuracy, the overall results have shown that the ESN controller has the capability of reliably tracking given angular trajectories, even when faced with noisy data. More improvements, not conducted in this thesis because of time constraints, can be made to the controller to improve performance and accuracy, as it will be explained in a following section. However, based on the previous observations,

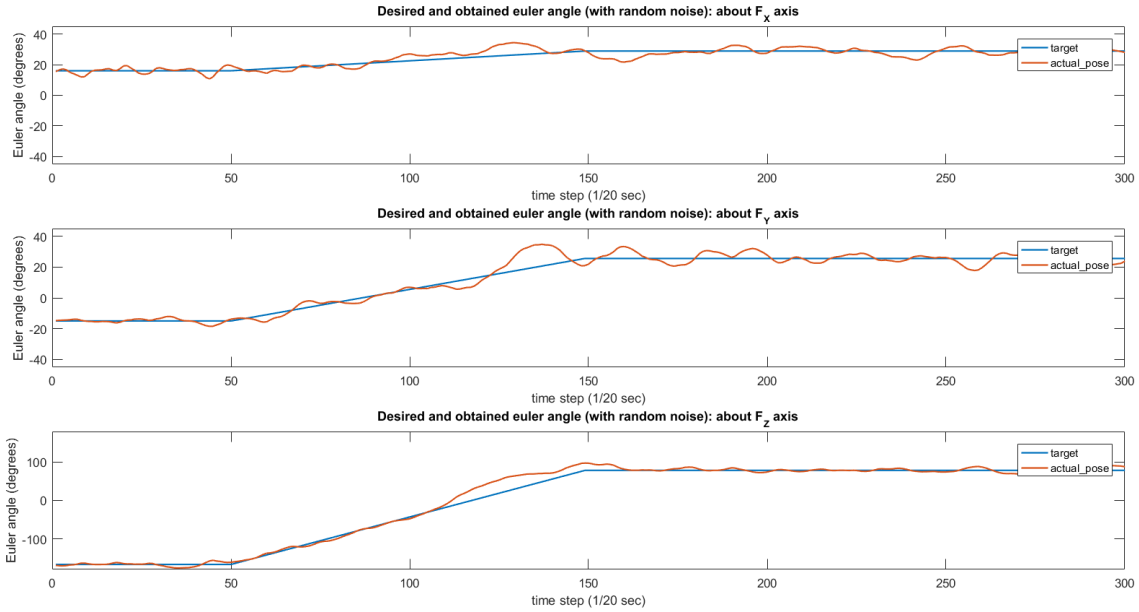

Figure 13: Test 2.4 b) Desired and obtained rotations, with random noise

it is clear that the ESN controller has shown its potential in both stabilization and tracking control on complex, non-linear, dynamical systems. Thus it can be considered that the principal objectives of the Guided Research were achieved with success.

4.3 Altitude control

On top of stabilization and trajectory tracking, altitude control was also attempted to be achieved with the ESN controller. The additional necessary work conducted in this sense in order to teach this ability to the network was already described in the previous sections.

However, when tested on a few simple cases, it was observed not only that the altitude control achieved only poor outcome, but the accuracy of the angular stabilization and tracking was drastically reduced, and eventually the pose of the helicopter even diverged from the desired pose. An example is represented in Figure 14, for stabilization in the pose $(0°, 0°, 0°)$ and altitude 3.

Some of the tests achieved good results nevertheless, for example stabilization in pose $(0°, 0°, 0°)$ and altitude 0, as seen in Figure 15. This suggest that certain improvements, described in the following section, can be made such that this objective can also be achieved. Further investigation has not been conducted in this project due to time constraints.

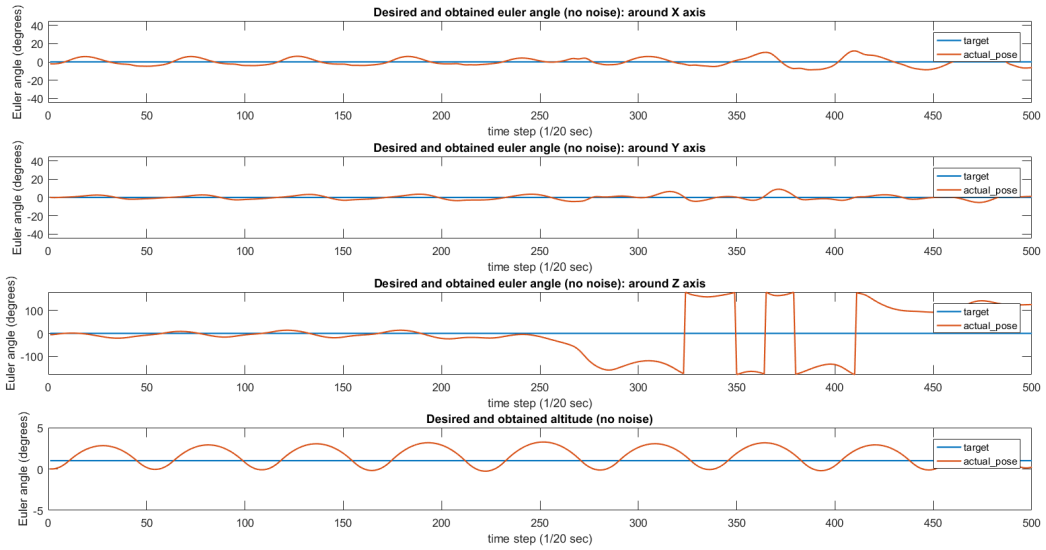

Figure 14: Pose stabilization and altitude control for pose $(0°, 0°, 0°)$ and altitude 3

5 Conclusions, further work and possible improvements

Unmanned aerial vehicle control and autonomous flight have been fields of significant research interest during the past decades and new approaches continue to be developed as the technology advances.

The purpose of this guided research, successfully achieved, was to investigate whether such an approach for UAV control can be established by using echo state networks, which have achieved very good results in modeling non-linear, dynamical systems. The obtained results have demonstrated the potential of echo state networks in a wide range of applications for controlling complex, non-holonomic systems.

More specifically, the design of an ESN-based feedback controller for the rotational stabilization and tracking of a helicopter was achieved.

The techniques employed in the development of this Guided Research project allow for further possible improvements and even for further elaborations to new objectives built on top of the presented achievements. Some of them will be introduced in this section.

A minimal principal task which could be researched next is achieving accurate and reliable altitude control in both stabilization and tracking, in addition to the pose control. This topic has been attempted in this project, however only limited success was reached. It is believed that the main limitation comes from the reduced variation in the training data. At this stage, the problem could not be overcome without involving a considerable increase in the data volume, which was not considered feasible with the available time and resources. Another possible reason for the sub-optimal outcome may be a difference in the dynamics of the plant regarding the change speed in different components. More con-

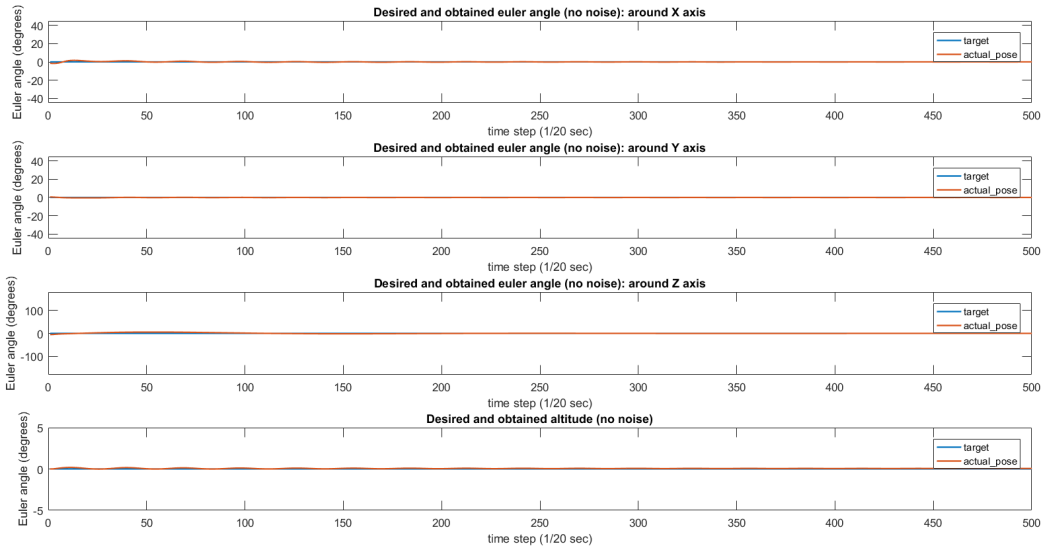

Figure 15: Pose stabilization and altitude control for pose $(0°, 0°, 0°)$ and altitude 0

cretely, the rotations controlled by the input variables Φ, Θ and T_t present faster dynamics than the altitude change caused by the thrust T_m, for which the time gap between consecutive changes should be larger. Although this difference was not considered within this thesis, a possible way to modify the presented ESC-controller such that the problem may be overcome is to use an echo state network with leaky-integrator neurons, described in Jaeger et al. [2007].

Another possible improvement which could also solve the problems faced when considering altitude control is the investigation of new, better ways to design and construct the training data. The choice which proved to bring good results in rotational control was one of the initial design choices with only a few later simplifications made. Therefore important research questions to be answered in the future arise: Are there other data generation methods which would produce more qualitative data in less quantity? How much can the obtained training data be reduced without affecting the accuracy of the results?

The quality of the teaching data could also be improved by making use of more complex controllers, tuning methods and noise structures in the output generation step. The PID controller presented and used in this sense is very simple and only manual tuning was done to find good parameters. It could be investigated if better results can be obtained by using other types of controllers or at least better PID tuning techniques, for example the methods introduced in Section 2.2. More complex noise generation could also bring certain improvements.

If these improvements are achieved, a much more challenging and complex task to be researched is the development of a complete rotation and location ESN-based controller. This would be the main step in creating a helicopter autonomous flight system using Echo State Networks.

References

Kiam Heong Ang, Gregory Chong, and Yun Li. PID control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, 13(4):559–576, July 2005. ISSN 1063-6536. doi: 10.1109/TCST.2005.847331.

José R. Azinheira and Alexandra Moutinho. Hover control of an UAV with backstepping design including input saturations. *IEEE Transactions on Control Systems Technology*, 16(3):517–526, May 2008. ISSN 1063-6536. doi: 10.1109/TCST.2007.908209.

Mohammed Belkheiri, Abdelhamid Rabhi, Ahmed El Hajjaji, and Claude Pegard. Different linearization control techniques for a quadrotor system. In *CCCA12*, pages 1–6, Dec 2012. doi: 10.1109/CCCA.2012.6417914.

James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58(15-16):1–35, 2006.

Sorin Draghici. On the capabilities of neural networks using limited precision weights. *Neural Networks*, 15(3):395 – 414, 2002. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(02)00032-1. URL http://www.sciencedirect.com/science/article/pii/S0893608002000321.

J. Dunfied, M. Tarbouchi, and G. Labonte. Neural network based control of a four rotor helicopter. In *Industrial Technology, 2004. IEEE ICIT '04. 2004 IEEE International Conference on Industrial Technology (ICIT)*, volume 3, pages 1543–1548, Dec 2004. doi: 10.1109/ICIT.2004.1490796.

Herbert Jaeger. Short term memory in echo state networks. GMD Report 152, Fraunhofer Institute for Autonomous Intelligent Systems, May 2002. URL http://minds.jacobs-university.de/sites/default/files/uploads/papers/STMEchoStatesTechRep.pdf.

Herbert Jaeger. The echo state approach to analysing and training recurrent neural networks with an Erratum note 1. GMD Report 148, Fraunhofer Institute for Autonomous Intelligent Systems, 2010. URL http://minds.jacobs-university.de/sites/default/files/uploads/papers/EchoStatesTechRep.pdf.

Herbert Jaeger. *Advanced machine learning lecture notes*. Jacobs University, 2011. URL http://minds.jacobs-university.de/sites/default/files/uploads/teaching/lectureNotes/LN_ML_Fall11.pdf.

Herbert Jaeger. *Machine learning in a tiny nutshell lecture notes for PSM, Part 4*. Jacobs University, 2016. URL http://minds.jacobs-university.de/sites/default/files/uploads/teaching/lectureNotes/LN_PSM_Part4.pdf.

Herbert Jaeger. *Machine learning lecture notes*. Jacobs University, 2017. URL http://minds.jacobs-university.de/sites/default/files/uploads/teaching/lectureNotes/LN_ML4IMS.pdf.

Herbert Jaeger, Mantas Lukoševičius, and Dan Popovici. Optimization and applications of echo state networks with leaky integrator neurons. In *Neural Networks 20(3)*, pages 335–352. 2007. URL http://minds.jacobs-university.de/sites/default/files/uploads/papers/leakyESN.pdf.

Michael I. Jordan. Chapter 2 Computational aspects of motor control and motor learning.

In Herbert Heuer and Steven W. Keele, editors, *Motor skills*, volume 2 of *Handbook of Perception and Action*, pages 71–120. Academic Press, 1996. doi: http://dx.doi.org/10.1016/S1874-5822(06)80005-8. URL http://www.sciencedirect.com/science/article/pii/S1874582206800058.

Yun Li, Kiam Heong Ang, and Gregory Chong. PID control system analysis and design. *IEEE Control Systems*, 26(1):32–41, Feb 2006. ISSN 1066-033X. doi: 10.1109/MCS.2006.1580152.

Guo-Ping Liu and Steve Daley. Optimal-tuning PID control for industrial systems. *Control Engineering Practice*, 9(11):1185 – 1194, 2001. ISSN 0967-0661. doi: http://doi.org/10.1016/S0967-0661(01)00064-8. URL http://www.sciencedirect.com/science/article/pii/S0967066101000648.

Mantas Lukoševičius. A practical guide to applying echo state networks. In *Neural Networks: Tricks of the Trade, G. Montavon, G. B. Orr, and K.-R. Müller (eds.), 2nd ed. Springer LNCS 7700*, pages 659–686. 2012. URL http://minds.jacobs-university.de/sites/default/files/uploads/papers/PracticalESN.pdf.

Mantas Lukoševičius, Herbert Jaeger, and Benjamin Schrauwen. Reservoir computing trends. In *KI - Künstliche Intelligenz*, pages 1–7. 2012. URL http://minds.jacobs-university.de/sites/default/files/uploads/papers/2508_Lukoseviciusetal12.pdf.

Kumpati S. Narendra and Kumar Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, Mar 1990. ISSN 1045-9227. doi: 10.1109/72.80202.

Kaustubh Pathak and Sunil K. Agrawal. An Integrated Spatial Path-planning and Controller Design Approach for a Hover-mode Helicopter Model. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1890–1895, April 2005. doi: 10.1109/ROBOT.2005.1570389.

Robert Savu. Helicopter stabilization with an ESN-based controller. Master's thesis, Jacobs University, 2014.

Ken Shoemake and Tom Duff. Matrix animation and polar decomposition. In *Proceedings of the conference on Graphics interface*, volume 92, pages 258–264, 1992.

S. Zein-Sabatto and Yixiong Zheng. Intelligent flight controllers for helicopter control. In *Neural Networks,1997., International Conference on*, volume 2, pages 617–621 vol.2, Jun 1997. doi: 10.1109/ICNN.1997.616092.