university of groningen

faculty of science and engineering

Echo State Network based Helicopter Control

Master Project Applied Mathematics

January 2021

Student: G.K. van Tilburg

First supervisor: Prof.dr. H. Jaeger

Second supervisor: Dr.ir. B. Besselink

ABSTRACT

In an Echo State Network (ESN), a large recurrent neural network is used as a "reservoir" of dynamics and for training, only the connection weights from the reservoir to the output units are calculated. ESNs can be trained easily and efficiently, and have proven to be applicable on multiple tasks, including control problems. This is done by presenting the ESN dynamics of the to be controlled system, from which the ESN can learn how the system behaves and exploit this to generate system inputs for control purposes. Autonomous helicopter flight represents a challenging nonholonomic control problem with complex, noisy, non-linear dynamics. In this work, the application of ESNs for various stabilization and tracking tasks on a model helicopter is explored.

Contents

1 Introduction

Reservoir computing refers to a family of recurrent neural networks (RNNs) that can be trained easily and efficiently. Echo state networks (ESNs) [1] is such an architecture and exploits the fact that under some conditions the activation state of a RNN is a function of the input history presented to the network. In an ESN, a large RNN is used as a "reservoir" of dynamics which can be excited by suitably presented input and/or fed-back output. The connection weights of the reservoir network are not changed in training, but in order to compute desired output dynamics only the connection weights from the reservoir to the output units are calculated. This is a linear regression and therefore ESNs can be trained easily and efficiently.

Today, the original unique selling point of ESNs, stable and simple training algorithms, has dissolved with the progress in the field of gradient descent based training for deep learning. However, methods of reservoir computing are still worth considering when the modeled system is not too complex, and when cheap, fast and adaptive training is desired. This has led to applications of reservoir computing in biosignal processing [2], remote sensing [3] and robot motor control [4].

Autonomous helicopter flight represents a challenging nonholonomic control problem with complex, noisy, non-linear dynamics. Where fixed-wing unmanned aerial vehicles are ideal for long flight and high payload applications, rotorcraft has advantages in maneuverability through the use of rotary blades. This enables it to produce aerodynamic thrust forces without the need of relative velocity and therefore vertical take-off and landing, hovering, and flight at very low altitude are possible. A helicopter has two inputs regarding the rotor thrust magnitudes which can be varied by changing the collective pitch of the main and tail rotor. Furthermore, a helicopter has two inputs regarding the orientation of the main rotor-blades' tip path plane, which can be changed by using the cyclic pitch of the main rotor. Therefore the considered system is underactuated by two.

To illustrate the complexity of helicopter flight, consider a helicopter hovering in place. A horizontally-oriented main rotor is attached to the helicopter and suppose that the main rotor rotates clockwise, viewed from above. The main rotors' thrust force causes the blades to blow air downwards and hence generating upward thrust. By applying clockwise torque to the main rotor to make it rotate, the helicopter experiences an anti-torque that tends the helicopter's body to spin counter-clockwise. Therefore it is necessary to use a tail rotor to blow air towards the right to generate a force to counteract the spin. But, this sideways force now causes the helicopter to drift leftwards. So, for a helicopter to hover in place it must actually be rotated over the axis parallel with the helicopter's tail. This enables the main rotors' thrust force to be directed downwards and slightly to the left, in order to counteract the tendency to drift sideways.

Helicopter controllers typically fit in one of three main categories: linear, non-linear, or model-free [5]. The categories are depending on the model representation used to describe the rotorcraft dynamics, which are of nonlinear nature. Linear models occur around a particular condition and therefore these controllers are only valid in some region of flight dynamics, while being relatively easy to design and implement. Non-linear models are more difficult to derive and implement due to the coupled, non-linear differential equations considered to fully describe the system dynamics, but do include phenomena not captured by linear systems such as the presence of multiple equilibria. Model-free controllers do not require a model of the helicopter and utilize learning or human based algorithms. These models rely on data presented to the algorithms and exploit the useful information extracted from the data.

When using non-linear control methods for autonomous helicopter flight, the main difficulties arise from 'small body' forces and moments and significant parameter and model uncertainty due to the complicated aerodynamic nature of the thrust generation. The small body forces and moments appear due to coupling of rotational and translational dynamics. A common approach (e.g. [6]) is to ignore small body forces by using an approximate dynamical model for the helicopter. A control function can be obtained for the approximate model and an a priori bound on the performance can be given when including small body forces [7]. Another approach is to include small body forces in the dynamical model which considers only the hovering low-speed mode of operation in low speed winds. From this, one can construct a two-level controller operating on different timescales for rotational and translational controller dynamics [8].

The introduction of reinforcement learning and apprenticeship learning to autonomous helicopter flight has led to impressive accomplishments. Early results enabled sustained autonomous hovering [9] and autonomous hover in regimes where the orientation is fairly close to upright [10]. Later results led to successful inverted hover [11] and even acrobatic manoeuvres such as a frontal flip [12].

In this work, the application of ESNs for autonomous model helicopter flight is explored. The objectives of the research are based on stabilization and trajectory tracking tasks for a small scaled helicopter. The objectives are attempted to be achieved first in a simplified model in which the helicopter has a reduced number of degrees of freedom. In the stabilization objective, the task is to hover in some position for as long as is desired. In tracking, the goal is to follow some reference trajectory, which can be continuous or discontinuous. Disturbances can be added to the equations used in the simulation, and by monitoring whether the controller is still able to complete its task successfully one can test the robustness of a controller.

This work is organized as follows: In Section 2.1, the framework of ESNs will be given. In Section 2.2, a dynamical model for a helicopter is presented and the discretization of the model for simulation is discussed. In the dynamical model, one can intuitively couple the 4 control inputs of the model with 4 degrees of freedom of the helicopter regarding rotation and altitude of the helicopter. This allows for the generation of a dataset with helicopter dynamics by using PID-control, which consists of helicopter dynamics from many short simulation episodes. In Section 3 it is discussed how this dataset can be generated and how an ESN-based controller can be constructed from this. In Section 4, two experiments are discussed. In the first experiment, an ESN-based controller is used to control helicopter altitude in a simplified model. In the second experiment, an ESN-based controller is used to control the helicopter's position and heading. Finally, conclusions, experimental analysis and suggestions for future work are provided in Section 5.

2 Theoretical Background

2.1 Echo State Networks

The ESN architecture considers discrete-time neural networks which consists of a reservoir of N internal units, to which K input and L output units are connected. Real-valued activations of input units at time step n are $\mathbf{u}(n) = (u_1(n), \ldots, u_K(n))^\top$, of internal units are $\mathbf{x}(n) = (x_1(n), \ldots, x_N(n))^\top$, and of output units are $\mathbf{y}(n) = (y_1(n), \ldots, y_L(n))^\top$. There are links in between the units and the real-valued connection weight w_{ij} is the weight on the link from unit j to unit i. Weights are collected in a $N \times K$ weight matrix $\mathbf{W}^{\text{in}} = \left(w_{ij}^{\text{in}}\right)$ for the input weights, in an $N \times N$ matrix $\mathbf{W} = (w_{ij})$ for internal weights, in an $L \times N$ matrix $\mathbf{W}^{\text{out}} = \left(w_{ij}^{\text{out}}\right)$ for the connections to the output units, and in a $N \times L$ matrix $\mathbf{W}^{\text{back}} = \left(w_{ij}^{\text{back}}\right)$ for the connections that project back from the output to the internal units.

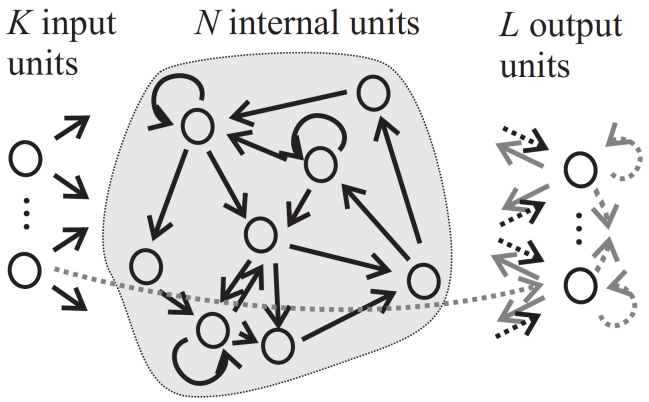


Figure 1: The arrows indicate connections and the grey dashed arrows indicate connections that are not considered in this work. The black dashed arrows indicate the (trainable) output weights. Image taken from [1].

To determine the state of the network, the state update equation is used:

$$\mathbf{x}(n+1) = \mathbf{f}\left(\mathbf{W}^{\text{in}}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}(n) + \mathbf{W}^{\text{back}}\mathbf{y}(n)\right) \tag{1}$$

where $\mathbf{f} = (f_1, \ldots, f_N)^\top$ are the internal unit's output functions, of which the most common choice is to use the hyperbolic tangent for all units. The output of the network is computed using the output equation:

$$\mathbf{y}(n+1) = \mathbf{f}^{\text{out}}\left(\mathbf{W}^{\text{out}}\mathbf{x}(n+1)\right), \tag{2}$$

where $\mathbf{f}^{\text{out}} = (f_1^{\text{out}}, \ldots, f_L^{\text{out}})^\top$ are the output unit's output functions, of which a common choice is to use the identity function.

When the network is updated according to Equation (1), then under certain conditions the network state becomes independent of initial conditions $\mathbf{x}(0)$. More precisely, if the network is started from two arbitrary initial states $\mathbf{x}(0)$, $\tilde{\mathbf{x}}(0)$ and is presented the same input sequence then the resulting state sequences $\mathbf{x}(n)$, $\tilde{\mathbf{x}}(n)$ will converge to each other. If this condition holds then the network state will asymptotically depend only on the input history and the network is said to possess the echo state property. It has been empirically shown that ESNs possess the echo state property whenever the spectral radius $|\lambda_{\max}|$ of $\mathbf{W}$ is smaller than 1. More details can be found in [1].

2.1.1 Training Echo State Networks

We will now discuss how to construct an ESN whose output $\mathbf{y}(n)$ approximates the teacher output $\mathbf{d}(n)$ generated by an (unknown) system. In order to do this, we assume that we are given a training input/output sequence: $(\mathbf{u}(n), \mathbf{d}(n)), n = 1, \ldots, T$. This is the training/teaching data from which we can construct the ESN, and we want the network to give a good approximation of it. More importantly, we want our network to give a good approximation of testing output data from independent test data sets generated by the same system which also generated the teacher data. From the previous section we can conclude that, when given $\mathbf{f}, \mathbf{f}^{\text{out}}$ and input signal $\mathbf{u}(n), \mathbf{u}(n-1), \ldots$, the output of an ESN is defined by the tuple $\left(\mathbf{W}^{\text{in}}, \mathbf{W}, \mathbf{W}^{\text{back}}, \mathbf{W}^{\text{out}}\right)$. Therefore the task at hand is to find an ESN defined by $\left(\mathbf{W}^{\text{in}}, \mathbf{W}, \mathbf{W}^{\text{back}}, \mathbf{W}^{\text{out}}\right)$ whose output approximates $\mathbf{d}(n)$ when the ESN is driven by $\mathbf{u}(n)$.

In the process of training an ESN, the values of $\mathbf{W}^{\text{in}}, \mathbf{W}, \mathbf{W}^{\text{back}}$ are fixed before training. In order to achieve desired results, these values should be chosen such that the network possesses the echo state property and the set of dynamics of the reservoir defined by (1) are "rich", i.e. internal units exhibit mutually interestingly different dynamics when excited. Now the network is fixed and the output weights $\mathbf{W}^{\text{out}}$ can be calculated, which metaphorically speaking "tap" from the input dynamics in order to approximate the desired output.

Calculating the output weights is a regression task and the most universal and stable way of computing the output weights is to use ridge regression. The training dynamics can be sampled after washout time T_0, the time to forget the initial state. The states $\mathbf{x}(n)$ for $n = T_0 + 1, \ldots, T$ are column-wise collected in a matrix $\mathbf{M}$ of size $N \times (T - T_0)$. The inverted teacher outputs $(\mathbf{f}^{out})^{-1}\mathbf{d}(n)$ are column-wise collected for $n = T_0 + 1, \ldots, T$ in a matrix $\mathbf{T}$ of size $L \times (T - T_0)$. Then, ridge regression takes

$$\mathbf{W}^{\text{out}} = \mathbf{T}\mathbf{M}^{\top}\left(\mathbf{M}\mathbf{M}^{\top} + bI\right)^{-1}, \tag{3}$$

where I is the $N \times N$ identity matrix and b is the regularization coefficient.

2.1.2 Leaky Integrator Neurons

When using Equation (1) to update the state, the value $\mathbf{x}(n+1)$ depends only fractionally and indirectly on the previous state and therefore the units have no memory. This motivates the use of leaky integrator units which, instead of (1), use the following state update equation:

$$\mathbf{x}(n+1) = (1-a)\mathbf{x}(n) + a\mathbf{f}\left(\mathbf{W}^{\text{in}}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}(n) + \mathbf{W}^{\text{back}}\mathbf{y}(n)\right), \tag{4}$$

where $a \in (0, 1]$ is the leaking rate. Note that Equation (4) becomes Equation (1) when taking $a = 1$, and full leaking occurs. Using leaky integrator units can be beneficial for learning slowly and continuously changing systems. The leaking rate a can be regarded as the speed of the reservoir update dynamics discretized in time. The leaking rate a should be set to match the speed of the dynamics of $\mathbf{u}(n)$ and/or $\mathbf{d}(n)$ [13]. When the task requires modeling the time series producing a dynamical system on multiple time scales, it might be useful to set different leaking rates to different units and collect them in a vector $\mathbf{a} \in \mathbb{R}^N$.

2.2 Helicopter Dynamics and Control

2.2.1 Helicopter Dynamics

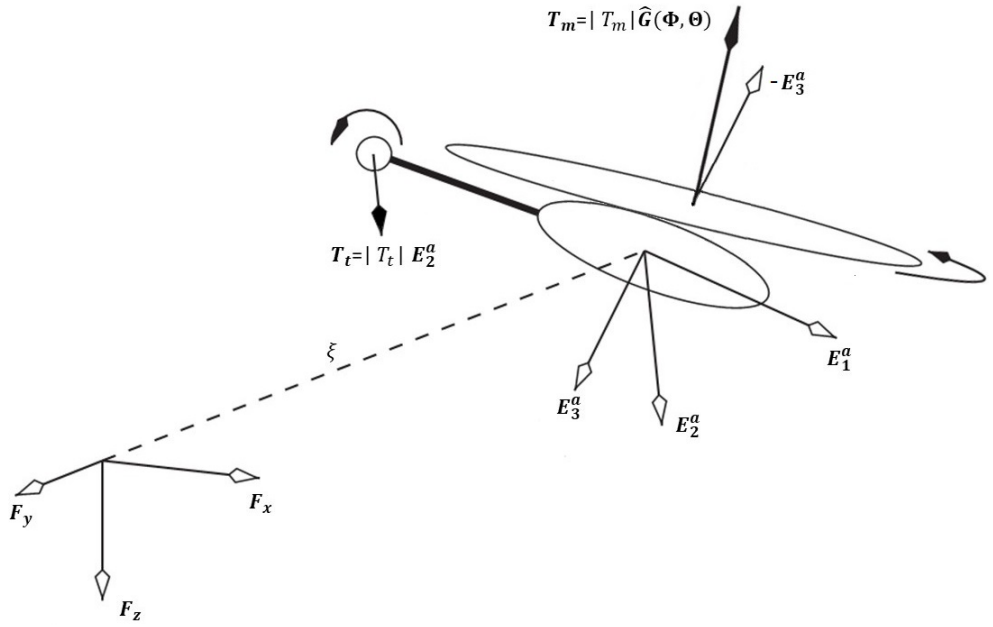


Figure 2: Coordinate system for the helicopter. A position vector ξ is used to describe the position of the helicopter and a rotation matrix R is used to describe the orientation of the helicopter. The four helicopter inputs determine the forces exerted by the tail and main rotor of the helicopter. The normal $\hat{\mathbf{G}}(\Phi,\Theta)$ to the main rotor-blades' tip path plane is obtained by rotating by Φ about $\mathbf{E}_2^a$ and by Θ about the original $\mathbf{E}_1^a$, where $\mathbf{E}_1^a$ and $\mathbf{E}_2^a$ are axes are fixed to the helicopter, originating in its center of gravity.

.

In this section, a dynamical model of a typical model helicopter is introduced. The approach taken is the same as in [8] and is illustrated in Figure 2. Let $\mathcal{F} = \{\mathbf{F_x}, \mathbf{F_y}, \mathbf{F_z}\}$ be an inertial reference frame and let $\mathcal{A}$ denote the helicopter-fixed referential centered at the center of mass. The Cartesian coordinates in $\mathcal{F}$ to the center of mass of the of the helicopter are given by $\xi = (\xi_1, \xi_2, \xi_3)^\top \in \mathbb{R}^3$. The rotation of $\mathcal{A}$ with respect to $\mathcal{F}$ is given by a rotation matrix $R = [\mathbf{E}_1^a, \mathbf{E}_2^a, \mathbf{E}_3^a]$. If no rotations about $\mathbf{E}_1^a, \mathbf{E}_2^a$ are applied then $\mathbf{E}_3^a$ and $\mathbf{F}_z$ are aligned and point down parallel to the gravity vector $\mathbf{g}$. Furthermore, $\mathbf{E}_1^a$ points to the helicopter's front and $\mathbf{E}_2^a$ points to the helicopter's right.

Euler angles α, β, γ in sequence Z-Y-X are used to read off the helicopters orientation from the rotation matrix, which means that we first rotate with α over $\mathbf{F}_z$, then with β over $\mathbf{F}_y$ and then with γ over $\mathbf{F}_x$. The ranges (in radians) of the Euler angles are $\alpha \in (-\pi, \pi], \beta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right], \gamma \in (-\pi, \pi]$. We are able to go from Euler angles to R and back by using the following equations:

$$R = \begin{bmatrix} c_\beta c_\alpha & c_\beta s_\alpha & -s_\beta \\ s_\gamma s_\beta c_\alpha - c_\gamma s_\alpha & s_\gamma s_\beta s_\alpha + c_\gamma c_\alpha & c_\beta s_\gamma \\ c_\gamma s_\beta c_\alpha + s_\gamma s_\alpha & c_\gamma s_\beta s_\alpha - s_\gamma c_\alpha & c_\beta c_\gamma \end{bmatrix}, \quad \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} \operatorname{atan2}(R_{12}, R_{11}) \\ -\operatorname{asin}(R_{13}) \\ \operatorname{atan2}(R_{23}, R_{33}) \end{pmatrix}, \tag{5}$$

where $c_\alpha = \cos\alpha, s_\alpha = \sin\alpha$,$c_\beta = \cos\beta, s_\beta = \sin\beta$, $c_\gamma = \cos\gamma, s_\gamma = \sin\gamma$ and R_{ij}, $i, j = 1, 2, 3$, is

the element in R at row i, column j.

The helicopter is considered to have 4 input variables, 3 inputs related to the main rotor and 1 input related to the tail rotor. The input related to the tail rotor is the magnitude of the tail rotors' thrust force $|T_t|$ and determines the tail thrust force $\mathbf{T}_t$ acting in the direction of body y-axis: $\mathbf{T}_t = |T_t|\,\mathbf{E}_2^a$. The 3 other input variables determine $\mathbf{T_m}$, the thrust force perpendicular to the main rotor blades' tip path plane (TPP), which is given by $\mathbf{T}_m = |T_m|\,\hat{\mathbf{G}}(\Phi,\Theta)$, where $|T_m|$ is the magnitude of the main rotors' thrust force and $\hat{\mathbf{G}}(\Phi,\Theta) = [\hat{G}_1, \hat{G}_2, \hat{G}_3]^T$ is the unit normal vector to the TPP, given by

$$\hat{\mathbf{G}}(\Phi,\Theta) = \begin{bmatrix} -\sin\Phi \\ \sin\Theta\cos\Phi \\ -\cos\Phi\cos\Theta \end{bmatrix}. \tag{6}$$

The normal $\hat{\mathbf{G}}(\Phi,\Theta)$ is arrived at by the following sequence of rotations: (i) A rotation of Φ about $\mathbf{E}_2^a$ and (ii) A rotation of Θ about the original $\mathbf{E}_1^a$. It is assumed that the angles Φ and Θ are directly controllable. Furthermore, it is assumed that the rotor thrust magnitudes $|T_m|$ and $|T_t|$ have positive ranges and can be varied directly by changing the collective pitch of the main rotor and tail rotor respectively. The dynamical model consists of 4 differential equations regarding position and orientation of the helicopter:

$$\dot{\xi} = \mathbf{v} \tag{7}$$

$$m\dot{\mathbf{v}} = R\begin{pmatrix} |T_m|\hat{G}_1 \\ |T_m|\,\hat{G}_2 + |T_t| \\ |T_m|\,\hat{G}_3 \end{pmatrix} + mg\mathbf{e}_3, \tag{8}$$

$$\dot{R} = R\breve{\Omega} \tag{9}$$

$$I_m\dot{\Omega} = -\breve{\Omega}I_m\Omega - |Q_m|\,\hat{\mathbf{G}} - |Q_t|\,\mathbf{e}_2 + K\begin{pmatrix} |T_m|\,\hat{G}_1 \\ |T_m|\,\hat{G}_2 \\ |T_t| \end{pmatrix} + |T_m|\,\hat{G}_3\mathbf{k_0}. \tag{10}$$

More details and derivation of the dynamical model can be found in [6]. In the dynamical model, $\mathbf{v}$ is the velocity of the helicopter, m is the mass of the helicopter, $\mathbf{e}_j$ denotes a unit column vector with a 1 in the j-th row, Ω is the angular velocity of $\mathcal{A}$ with respect to $\mathcal{F}$, $\breve{\Omega}$ is the cross-product skew-symmetric matrix corresponding to Ω, the position of the hub of the main and tail rotor in the helicopter referential $\mathcal{A}$ are respectively given by $\mathrm{l_m}$ and $\mathrm{l_t}$, where

$$\mathrm{l_m} = \begin{pmatrix} l_m^1 \\ l_m^2 \\ l_m^3 \end{pmatrix}, \mathrm{l_t} = \begin{pmatrix} l_t^1 \\ l_t^2 \\ l_t^3 \end{pmatrix}, K \equiv \begin{pmatrix} 0 & -l_m^3 & -l_t^3 \\ l_m^3 & 0 & 0 \\ -l_m^2 & l_m^1 & l_t^1 \end{pmatrix}, \mathbf{k}_0 \equiv \begin{pmatrix} l_m^2 \\ -l_m^1 \\ 0 \end{pmatrix}. \tag{11}$$

Reactive anti-torques generated at the two hubs due to aerodynamic drag and the reaction of the rotormotors are denoted by $|Q_m|$ and $|Q_t|$ at the main rotor and the tail rotor respectively and these are modeled as in [14]. Finally, I_m is the inertial matrix in $\mathcal{A}$. For the physical constants, the values from Table 1 are used.

Stimulation parameters		
Parameter	Value	Unit
Δt	0.01	s
m	4.9	Kg
g	9.81	m s^{-2}
$[I_{xx}, I_{yy}, I_{zz}]$	$[0.142413, 0.271256, 0.271492]$	Kg m^2
$[l_m^1, l_m^2, l_m^3]$	$[0.015, 0, -0.2943]$	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
$\min(T_m)$	0	N
$\max(\lvert\Phi\rvert)$	15	$^\circ$
$\max(\lvert\Theta\rvert)$	15	$^\circ$

Table 1: Simulation parameters of a scaled model. Values are taken from [8].

Equations (7),(8),(9),(10) completely specify the behavior of the helicopter under initial conditions and inputs. We use approximate difference equations with step size Δt to discretize these equations for simulation purposes. We assume that we are given the state variables of the helicopter at time n, specified by $\Omega(n), R(n), v(n), \xi(n)$, and the helicopter inputs at time $n+1$. For convenience, we drop the argument for the input variables $\Phi, \Theta, |T_m|, |T_t|$. The discretized equations are:

$$\Omega(n+1) = \Omega(n) + \Delta t I_M^{-1} \left[-\breve{\Omega}(n) I_m \Omega(n) - |Q_m| \hat{\mathbf{G}} - |Q_t| \mathbf{e}_2 + K \begin{pmatrix} |T_m| \hat{G}_1 \\ |T_m| \hat{G}_2 \\ |T_t|, \end{pmatrix} + |T_m| \hat{G}_3 \mathbf{k_0} \right],$$

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

$$v(n+1) = v(n) + \frac{\Delta t}{m} \left[R(n+1) \begin{pmatrix} |T_m| \hat{G}_1 \\ |T_m| \hat{G}_2 + |T_t| \\ |T_m| \hat{G}_3 \end{pmatrix} + mg\mathbf{e}_3 \right],$$

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

2.2.2 Helicopter PID-Control

Training an ESN-controller requires helicopter data, and in the training phase a helicopter input signal is required, as will be explained in Section 3.3. In this work, a proportional, integral, derivative (PID) controller is used in order to generate a dataset with helicopter dynamics. The PID-controller is a commonly used type of feedback controller due to its simplicity, computational efficiency and applicability in a wide range of fields. Using PID-control for a model helicopter is a naive approach since the model described above is an under-actuated nonlinear multiple-input multiple-output (MIMO) system which cannot be decoupled into single-input single-output (SISO) subsystems. The approach taken allows the helicopter to hover in a certain pose around a certain position without much precision. The found dynamics of the helicopter are used to generate a dataset from which an ESN-controller can learn to control the helicopter in a desired fashion.

In discrete time systems, the PID-control input of the system is determined by using $E(n)$, an error term of the system at time n given by $E(n) = s(n) - x(n)$, where $s(n)$ is the desired set-point of the system and $x(n)$ is the state of the system. The PID control variable $u(n)$ for a system with step size Δt is determined by

$$u(n) = K_P E(n) + K_I z(n) + \frac{K_D}{\Delta t}[E(n) - E(n-1)], \tag{12}$$

where K_P, K_I, K_D are the parameters of the control law corresponding to respectively proportional, integral and derivative error terms, and $z(n)$ is the integrated error satisfying

$$z(n) = z(n-1) + \Delta t E(n). \tag{13}$$

By using Equation (12) and (13), we can obtain

$$u(n) = u(n-1) + K_P[E(n) - E(n-1)] + K_I \Delta t E(n) + \frac{K_D}{\Delta t}[E(n) - 2E(n-1) + E(n-2)], \tag{14}$$

which allows us to specify the control input $u(n)$ without explicitly using the integrated error term $z(n)$.

We will now illustrate how PID-control can be attempted to be used for stabilization of the dynamical model described in Section 2.2. Suppose that the helicopter is in a position where it is only rotated about $\mathbf{F}_y$, such as in Figure 2, and we want to bring the helicopter in a position such that $(\alpha, \beta, \gamma) = (0, 0, 0)$. Recall that β is the Euler angle corresponding to rotation about $\mathbf{F}_y$, and let $E_\beta(n)$ be the error in β at time step n: $E_\beta(n) = 0 - \beta(n) = -\beta(n)$. We can assign a control input that has most influence on $\beta(n)$ and use a PID-controller based on $E_\beta(n)$ to determine this control input. In this case, the control input that has most influence on β is Φ, which is the control input that causes $\hat{G}(\Phi, \Theta)$ to be rotated about the body $\mathbf{E}_2^a$. For example, in Figure 2, β is negative and $\alpha = \gamma = 0$. Therefore the error E_β is positive and $E_\alpha = E_\gamma = 0$. Increasing Φ results in $\hat{G}(\Phi, \Theta)$ to rotate about the $\mathbf{E}_2^a$-axis and this should result in increasing values of β, which is as desired. So for Φ, the following control law can be used:

$$\Phi(n) = \Phi(n-1) - K_{P,\Phi}[E_\beta(n) - E_\beta(n-1)] - K_{I,\Phi}\Delta t E_\beta(n) - \frac{K_{D,\Phi}}{\Delta t}[E_\beta(n) - 2E_\beta(n-1) + E_\beta(n-2)], \tag{15}$$

where $K_{P,\Phi}, K_{I,\Phi}, K_{D,\Phi}$ are the positive tunable parameters for PID-control for Φ. Note the sign change for the PID terms in Equation (15) compared to Equation (12), which is in order to increase or decrease Φ when E_β is respectively positive or negative. Similar reasoning allows us to couple variables α, γ, ξ_3 with control inputs $|T_t|, \Theta, |T_m|$ respectively. However, the helicopter dynamics are coupled and therefore changing one input variable also has influence on the components of the helicopter controlled by a different input variable. This means that four separate PID-controllers should be tuned together in order to stabilize the system in a desired fashion.

3 Methodology

The ESN-controller is learned from a dataset, which is constructed from short simulation episodes of helicopter dynamics. For each episode, the horizontal coordinates ξ_1, ξ_2 as well as the angular and linear velocity Ω and $\mathbf{v}$ are initially set to zero. A *pose* specifies the rotation and altitude of the helicopter, and can be specified by values of $\alpha, \beta, \gamma, \xi_3$. The initial pose will be different for each episode. For each episode, control inputs based on a PID controller that aims at stabilizing the helicopter in the initial pose will be used. It is not necessary (nor desirable) that the PID controller stabilizes the helicopter with great precision. Instead, the control inputs should leave the helicopter to remain uncrashed for the duration of the episode, such that the dataset contains useful information concerning helicopter dynamics. The helicopter dynamics and control inputs for all episodes are saved in the dataset, and from this the ESN-controller is constructed.

The used dataset was chosen such that a wide range of helicopter poses is visited and that computation time is acceptable. The intervals from which initial values are taken for α, β, γ and ξ_3 are $[-154°, 154°]$,$[-12°, 12°]$,$[-12°, 12°]$ and $[-1\text{m}, 1\text{m}]$ respectively. The stepsize of the heading α will be set to $22°$, the stepsize of the Euler angles β, γ will be set to $4°$ and the stepsize of ξ_3 will be set to 0.5m. By considering all possible combinations we obtain a set of $15 \cdot 7 \cdot 7 \cdot 5 = 3,675$ starting values for simulation episodes. With simulation step size set to $\Delta t = 1/100$s and episode duration of 0.9 seconds, this gives a dataset of $3,675 \cdot 91 = 334,425$ timesteps.

Throughout the following sections, we will use the values for episode 1838 and 3675 to visualize the results. Episode 1838 is the episode in the middle of the dataset, with initial values $\alpha = \beta = \gamma = 0°$ and $\xi_3 = 0$m. Episode 3675 is the last episode of the dataset, with starting values $\alpha = 154°, \beta = 12°, \gamma = 12°$ and $\xi_3 = 1$m.

3.1 PID-Inputs

The first step towards training an ESN-controller is to generate PID-control inputs which leave the helicopter uncrashed in that episode. At every time step n, we are given the state of the system specified by $\Omega(n), R(n), v(n), \xi(n)$ and from this we can calculate the current values of $\alpha, \beta, \gamma, \xi_3$. We can calculate the errors $E_\alpha, E_\beta, E_\gamma, E_{\xi_3}$ from the current values and the desired values of $\alpha, \beta, \gamma, \xi_3$, where the desired values are equal to the values at the beginning of an episode. From the errors, we can calculate inputs $\Phi, \Theta, |T_m|, |T_t|$ using the PID controller described in Section 2.2. In the PID-controller, only the proportional parameters K_P are set to nonzero values and the control is determined via the following equations:

$$\Phi(n) = \Phi(n-1) - 4[E_\beta(n) - E_\beta(n-1)] \tag{16}$$

$$\Theta(n) = \theta(n-1) - 0.4[E_\gamma(n) - E_\gamma(n-1)] \tag{17}$$

$$|T_m|(n) = |T_m|(n-1) + 1300[E_{\xi_3}(n) - E_{\xi_3}(n-1)] \tag{18}$$

$$|T_t|(n) = |T_m|(n-1) + 22.92[E_\alpha(n) - E_\alpha(n-1)] \tag{19}$$

where all angles are in degrees, forces in Newtons and distances in meters. The values of K_P are the same for every episode and are chosen such that together they do a decent job in stabilizing the helicopter's Euler angles and altitude. In Figure 3, the results of episode 1838 are plotted. In the middle column of plots the error is shown, and it can be seen that the helicopter inputs depend proportionally on the error.

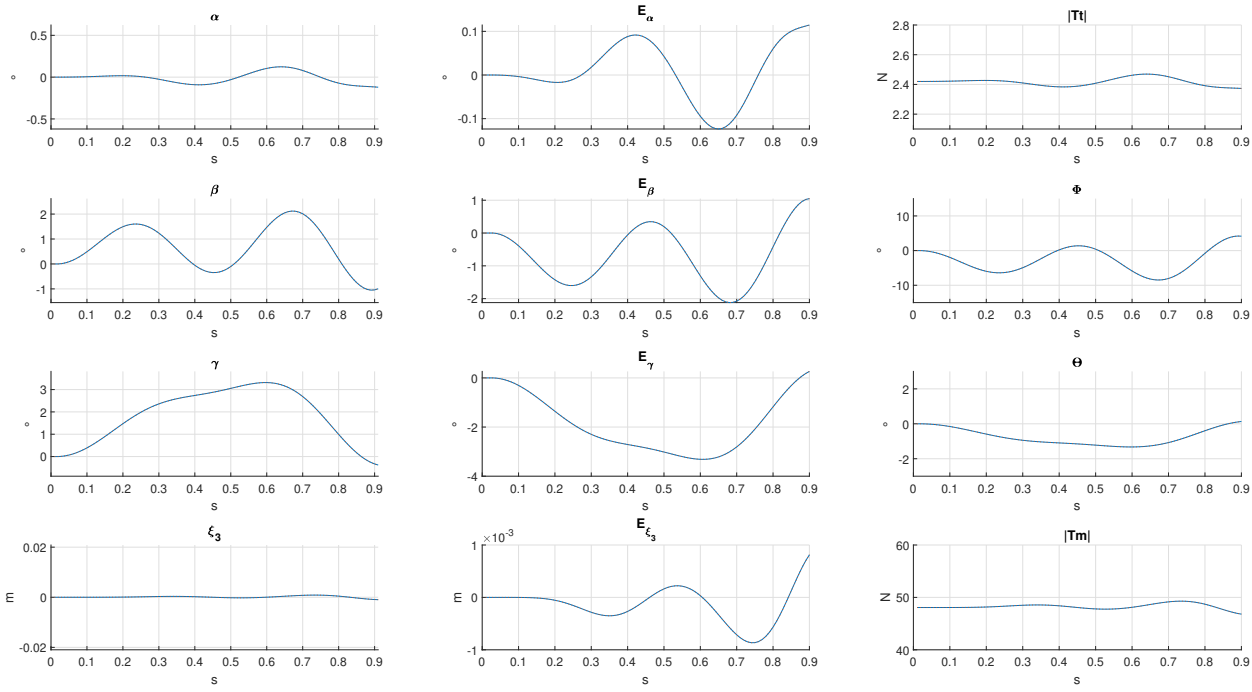


Figure 3: PID-control in episode 1838, based on stabilizing the helicopter in $\alpha = \beta = \gamma = 0°$ and $\xi_3 = 0$m. The helicopter inputs plotted in the column of plots on the right are obtained by using the error plotted in the middle column of plots.

In Figure 4, the same plots are given for episode 3675, where the PID-controller is trying to stabilize the helicopter in $\alpha = 154°, \beta = 12°, \gamma = 120°$ and $\xi_3 = 1$m. As expected, the PID-controller has more difficulties stabilizing the helicopter in this position. Therefore the errors and helicopter inputs show more wild behavior when comparing with Figure 3. The procedure of finding helicopter PID-inputs is exactly the same for all other episodes, and the results are similar as those of episodes 1838 and 3675.

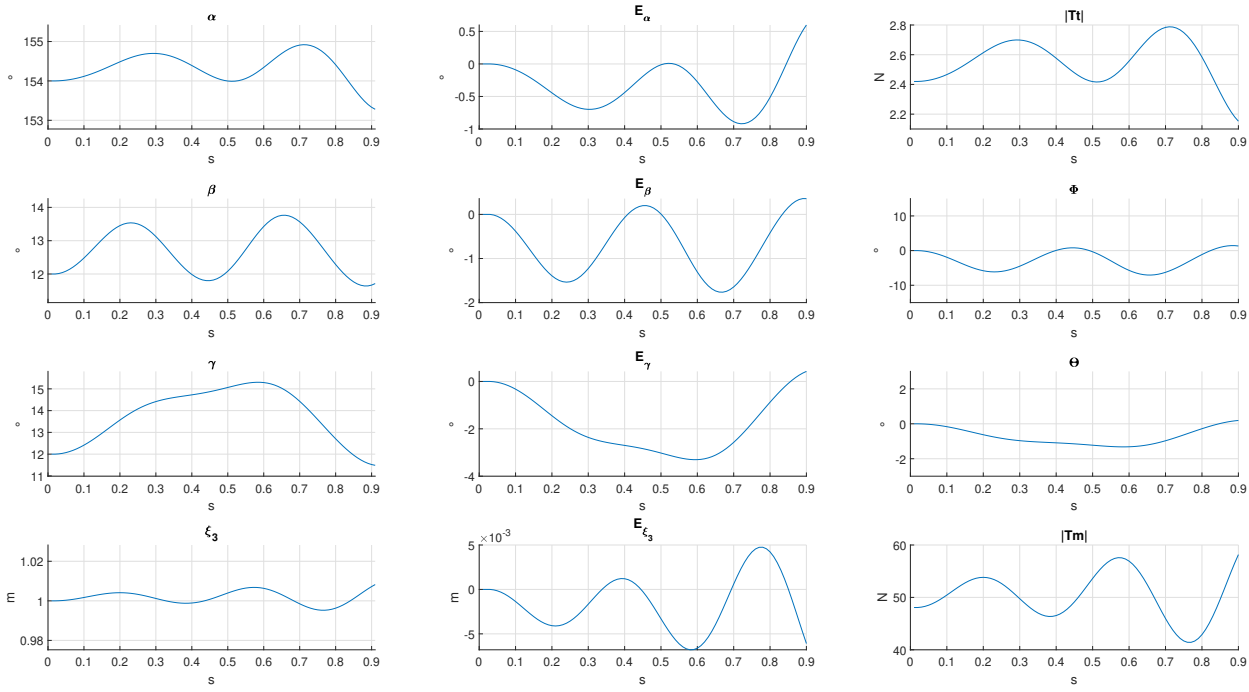


Figure 4: PID-control for episode 3675, where the PID-controller tries to stabilize the helicopter in $\alpha = 154°, \beta = 12°, \gamma = 12°$ and $\xi_3 = 1$m. It is harder for the helicopter to stabilize in this episode than in episode 1838 and therefore the PID-controls show more wild behavior.

To illustrate the instability of the found PID-controller, episode 1838 is extended and the result of stabilizing the helicopter for 2.5s is plotted in Figure 5. The control input Φ reaches its physical boundary of 15° after about 1.5s. This saturation value is not taken into consideration in the simulation, and the helicopter is expected to turn into turbulent rotations even faster when it is modelled in the simulation. When this experiment was extended for another 1.5 seconds, the heading of the helicopter reached a value of $-150°$ and the simulation program gave errors shortly thereafter. In more challenging positions such as episode 3675, this happened within an even shorter testing period.

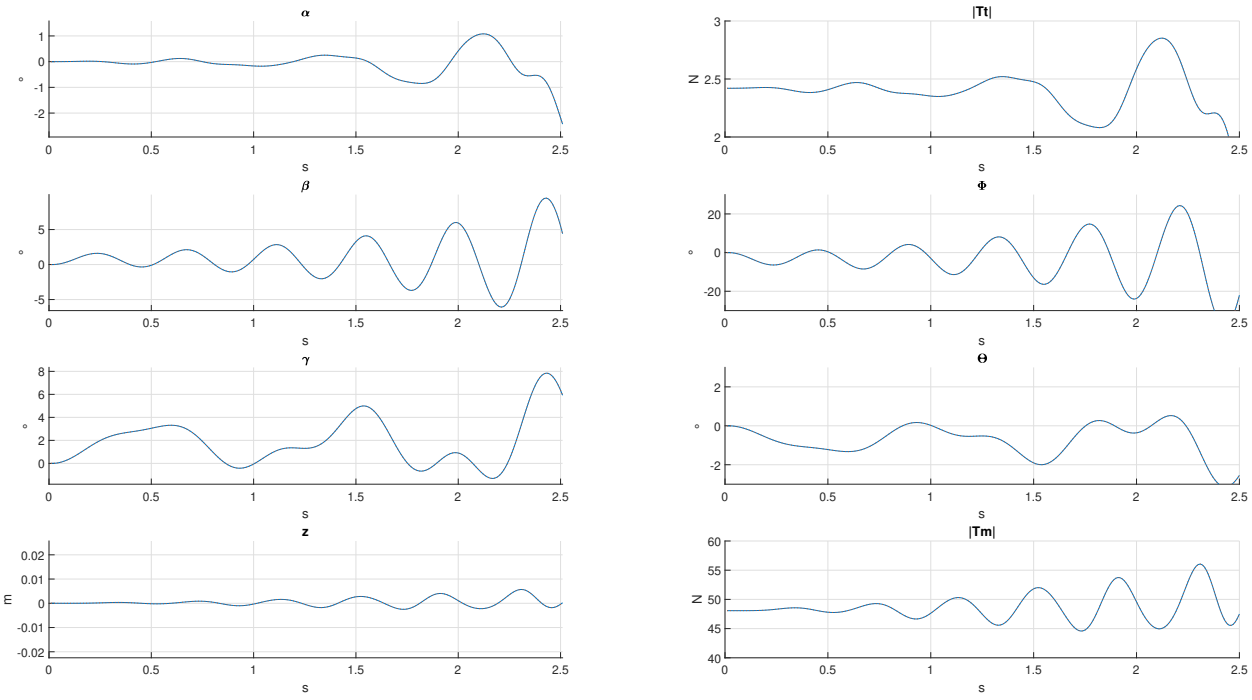


Figure 5: Stabilizing the helicopter using the PID controller for 2.5s. The controller has difficulties to prevent the helicopter from crashing after a short time period and in a relatively easy position.

3.2 Introducing Noise

In order for the ESN-controller to behave properly in many situations, a dataset with diverse helicopter dynamics is needed. To introduce more variation in the dataset, a noise signal is added to the PID-control inputs discussed above. The noise signals are obtained as follows. First, for each of the 4 control inputs, a normal distribution based on the PID-control inputs is obtained. This normal distribution has a mean equal to the mean of the PID-control signal for that respective input and a deviation equal to the deviation of the PID-control signal of that respective input.

After obtaining noise signals for all inputs, the signals are passed through a low-pass filter(LPF). A LPF removes frequencies above the cutoff frequency f_c, and in this work a LPF based on the discrete Fourier transform is used. This filter calculates the discrete Fourier transform of the signal and uses the sampling rate and cutoff frequency to determine which part of the signal can be discarded. The discrete Fourier transform of the signals after filtering is plotted in Figure 6.
The reason for using a LPF is that the helicopter model does not have time to react to high frequency noise, and therefore these noise signals appear as unwanted values in the training data. Therefore, one should consider the natural timescales of the helicopter to determine the cutoff frequencies. The cutoff frequency of the noise signals should be carefully chosen and should be in line with the ESN-controller delay. For example, when one uses a cutoff frequency of 2Hz, then the corresponding period is 0.5s and it assumed that the delay should be chosen to be below 0.5s. After the signals have gone through the LPF, the mean of the resulting signal is subtracted to

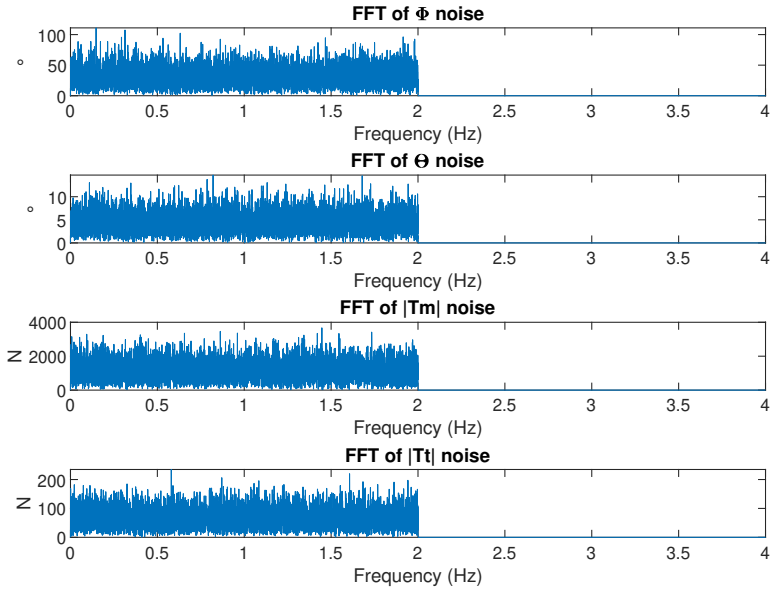


Figure 6: Plotting the Fourier transform of noise on signals which are added on the PID-inputs, with cutoff frequency $f_c = 2$Hz.

obtain a signal with zero mean. Finally, this signal is added on the original PID-control input signal. The result of this can be seen in Figures 7, 8.

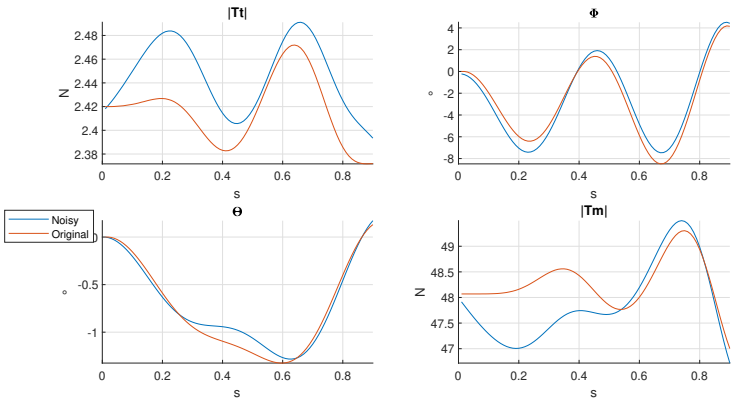


Figure 7: Comparing the new, "noisy" input signal with the original PID-inputs for episode 1838.

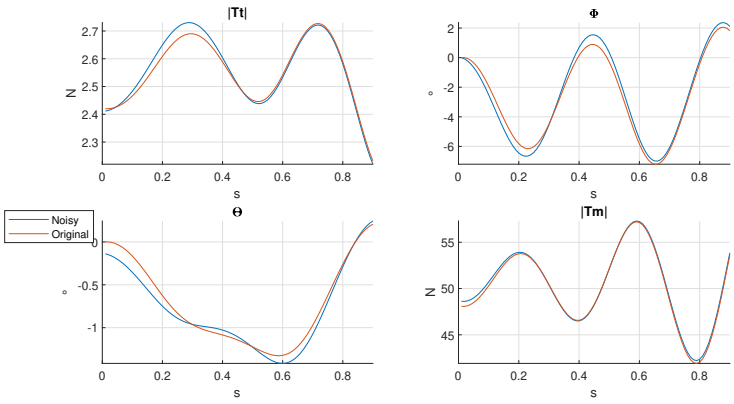


Figure 8: Comparing the new, "noisy" input signal with the original PID-inputs for episode 3675.

When having obtained new input signals, one can see what the effects are of adding noise to the the PID-inputs. The flown trajectories are visualized in Figures 9, 10 for episodes 1838 and 3675 respectively.

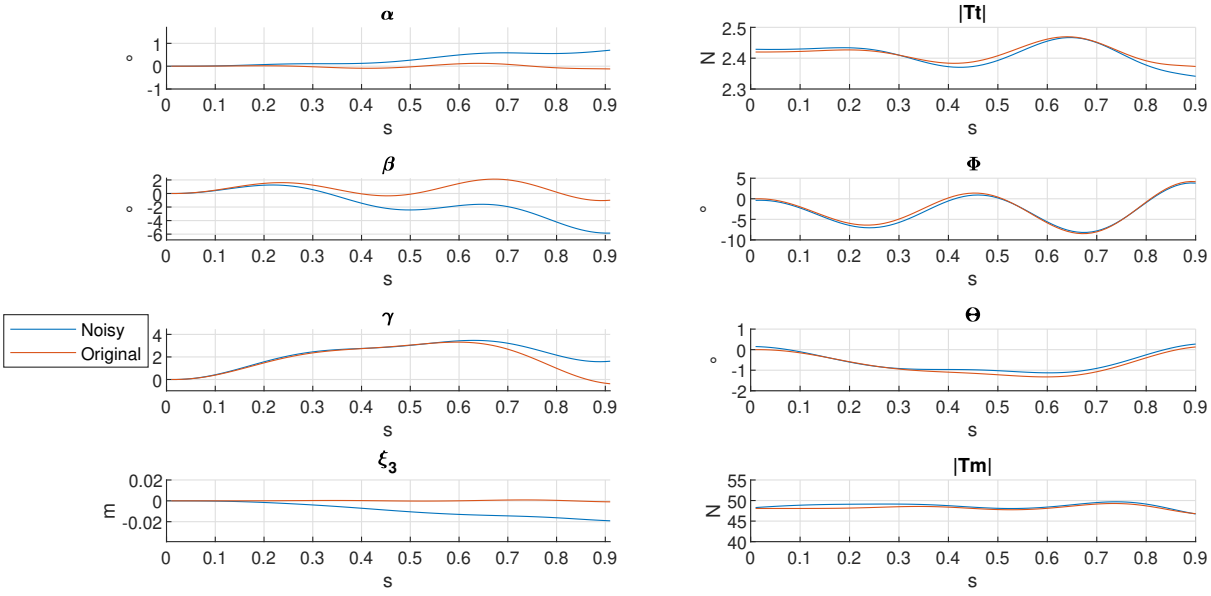


Figure 9: Impact of adding noise on the PID-inputs of episode 1838.

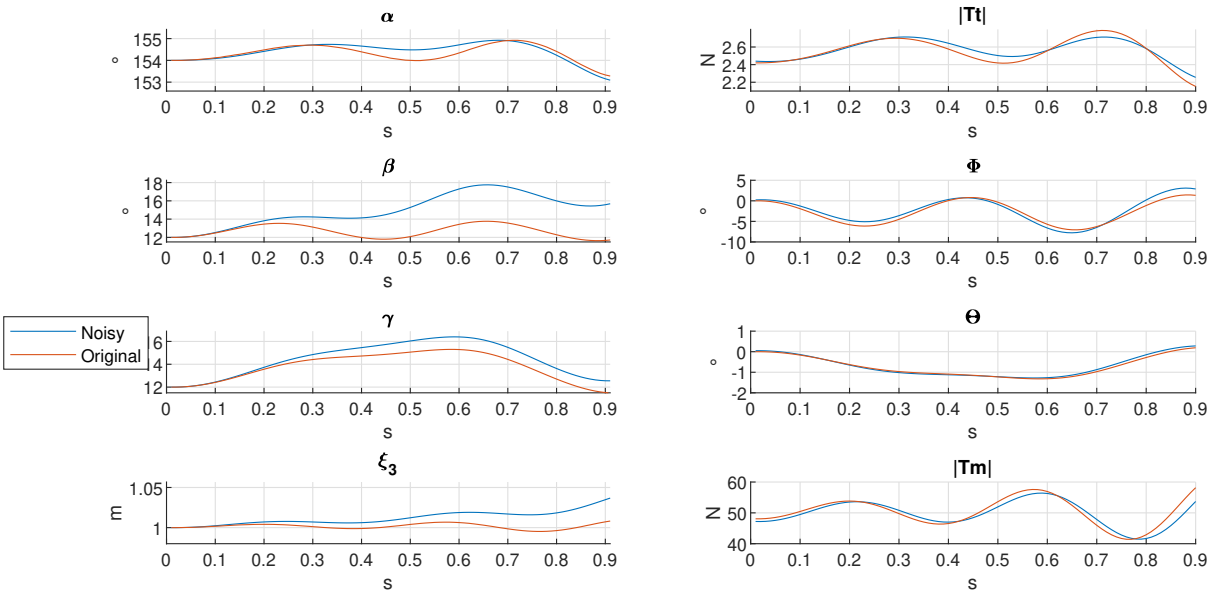


Figure 10: Impact of adding noise on the PID-inputs of episode 3675.

3.3 ESN-based Helicopter Control

Several helicopter control tasks will be considered in this work, and we need a way to make this precise. Consider, for example, the task to stabilize the position ξ and the heading α of the helicopter. The helicopter position and heading at timestep n is specified by

$$\mathbf{u}(n) = (\alpha(n), \xi_1(n), \xi_2(n), \xi_3(n)) \,. \tag{20}$$

The control objective can then be specified by requiring that the sequence $(\mathbf{u}(n))_{n=1,\ldots,T}$ should follow some given reference sequence $(\mathbf{u}_{\text{ref}}(n))_{n=1,\ldots,T}$. Since the task at hand is stabilization, the desired values remain constant for all n. This task can be extended to tracking, where the desired values are specified by some trajectory of the helicopter, i.e. the helicopter changing position and heading over time.

In Figure 11(a), it is shown how echo state networks can be trained in order to be used for control purposes. In this case, some time signal $\mathbf{y}(t)$ is given which is taken to be the input of the helicopter. The helicopter reacts to these inputs and is brought into a certain state, which is indicated by $\mathbf{s}(t)$ and $\mathbf{u}(t)$. The network is trained such that its outputs approximate the delayed signal $\mathbf{y}(t-d)$ when given $\mathbf{u}(t)$ and $\mathbf{s}(t-d)$ as inputs. The reason for this becomes apparent when looking at Figure 11(b). The network observes helicopter state $\mathbf{s}(t)$ and is given some reference state $\mathbf{u}_{\text{ref}}(t+d)$, which is determined by the control objective. The network output $\mathbf{y}(t)$ is fed to the helicopter and, when trained properly, this is the control input that brings the helicopter into state $\mathbf{u}_{\text{ref}}(t+d)$, the helicopter state d timesteps in the future. Therefore, the echo state network can be given a desired future helicopter state and will calculate helicopter inputs such that the helicopter will be brought into this state.

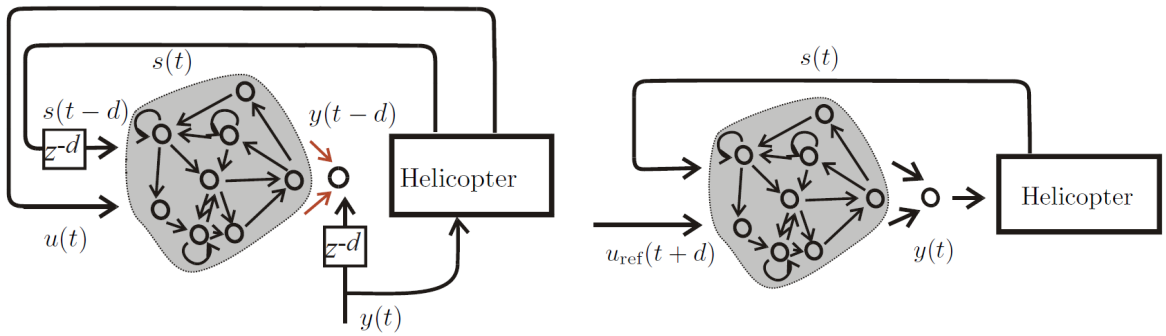


Figure 11: **(a)**: Training phase. The network learns how helicopter inputs $\mathbf{y}(t)$ depend on helicopter states $\mathbf{s}(t), \mathbf{u}(t+d)$. **(b)**: Exploitation. The network calculates helicopter inputs $\mathbf{y}(t)$ to bring the helicopter from state $\mathbf{s}(t)$ to state $\mathbf{u}_{\text{ref}}(t+d)$. Images taken and modified from [15].

To realize the flow architecture in Figure 11, a modified version of the state update equation (Equation (1)) will be used: instead of including a feedback term $\mathbf{W}^{\text{back}}\mathbf{y}(n)$ from output to hidden units, we include a feedback term from some of the helicopter state variables to the hidden units. For this, we assume that the helicopter variables are directly measurable and we collect these in a helicopter state vector $\mathbf{s}(n) \in \mathbb{R}^P$. The weights from the helicopter state vector to the reservoir are collected in $\mathbf{W}^{\text{state}} \in \mathbb{R}^{N \times P}$. The selection of state components $\mathbf{s}(n)$ should be done in such a way that the ESN-controller knows the helicopter variables that are most relevant for

the task at hand.

The training phase flow architecture is realized by using the following state update and output equation:

$$\mathbf{x}(n+1) = \mathbf{f}\left(\mathbf{W}^{\text{in}}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}(n) + \mathbf{W}^{\text{state}}\mathbf{s}(n+1-d)\right), \tag{21}$$

$$\mathbf{y}(n+1-d) = \mathbf{f}^{\text{out}}\left(\mathbf{W}^{\text{out}}\mathbf{x}(n+1)\right), \tag{22}$$

where d is the ESN-controller delay, the number of timesteps between the time-shifted signals. The time delay d must be chosen carefully since the network should be able to correlate $\mathbf{y}(n)$ with $\mathbf{s}(n)$ and $\mathbf{u}(n+d)$. If d is too small then the helicopter did not have enough time to react to inputs, and if d is too large then there is no useful relation between state and input values.

The dataset from which the ESN-controller is learned should cover a wide range of poses and helicopter dynamics, since the ESN-controller learns from the dataset how the helicopter dynamics depend on control inputs. The noise should lead to a rich set of simulation dynamics but should not lead to unstable dynamics, i.e. the helicopter crashing before the end of an episode. By assuming that the network possesses the echo state property and shifting the equations forward in time with d timesteps, we obtain the following versions of these equations during the exploiting phase:

$$\mathbf{x}(n+1) = \mathbf{f}\left(\mathbf{W}^{\text{in}}\mathbf{u}(n+d+1) + \mathbf{W}\mathbf{x}(n) + \mathbf{W}^{\text{state}}\mathbf{s}(n+1)\right), \tag{23}$$

$$\mathbf{y}(n+1) = \mathbf{f}^{\text{out}}\left(\mathbf{W}^{\text{out}}\mathbf{x}(n+1)\right). \tag{24}$$

Now, the network can be trained by using the dynamic dataset. Since ESNs have a state forgetting character and start from an arbitrary state, all states $\mathbf{x}(n)$ corresponding to the beginning of an episode are discarded. Furthermore, we can only start sampling data in an episode after d timesteps. Therefore the size of the dataset for the regression task will be reduced by a factor $T_0 + d$ per episode, where T_0 is the washout time of the network.

4 Results

The training and testing results of two experiments will be discussed in the following sections:

1. Controlling altitude in a simplified helicopter model (Section 4.1)
2. Controlling position and heading of the helicopter (Section 4.2)

Initially another experiment was also considered, in which the Euler angles and altitude of the helicopter were controlled. This control objective is the same as the control objective that was used for the PID-controller used for data generation. However, this control objective turned out to be under-specified since movement of the helicopter in the horizontal plane was not monitored. Therefore the control objective could be met while accelerating in the horizontal plane, which eventually led to the helicopter becoming unstable and the ESN-controller failing its task. Therefore this experiment and control objective was discarded.

In the following sections the results of training an echo state network with the following settings are discussed:

- A reservoir of $N = 100$ internal units.
- For the internal unit's output function $\mathbf{f} = (f_1, \ldots, f_N)^\top$, the hyperbolic tangent is used for all units.
- For the output unit's output function $\mathbf{f}^{\text{out}} = (f_1^{\text{out}}, \ldots, f_L^{\text{out}})^\top$, the identity function is used for all units.
- For $\mathbf{W}$ a sparse matrix with 5% of its entries nonzero and equal to 1 or -1 with equal probability is used.
- For the input weights $\mathbf{W}^{\text{in}}$ and the state weights $\mathbf{W}^{\text{state}}$ a matrix with entries in between $[-1, 1]$ sampled from a uniform distribution is used.
- In the state update equation, a bias vector $\mathbf{b} \in \mathbb{R}^N$ is added. For $\mathbf{b}$ a vector with entries in between $[-1, 1]$ sampled from a uniform distribution is used.
- A washout time of $T_0 = 30$ timesteps is used.

The network parameters with their respective considered ranges are:

- A scaling factor W_{scale} for $\mathbf{W}$, which determines the spectral radius $|\lambda_{\max}|$ of $\mathbf{W}$. Only values considered such that $|\lambda_{\max}| \in [0.1, 1.5]$.
- A scaling factor $W_{\text{scale}}^{\text{in}} \in [0, 1.5]$ for $\mathbf{W}^{\text{in}}$.
- A scaling factor $W_{\text{scale}}^{\text{state}} \in [0, 1.5]$ for $\mathbf{W}^{\text{state}}$.
- A scaling factor $b_{\text{scale}} \in [0, 2]$ for $\mathbf{b}$.
- The leaking rate $a \in (0, 1]$.
- The regularization coefficient b such that the average absolute size of the learnt $\mathbf{W}^{\text{out}}$ is smaller than 2.

Implementing the above leads to the following state update equation:

$$\begin{aligned}\mathbf{x}(n+1) &= (1-a)\mathbf{x}(n) \\ &\quad + a\tanh\left(W^{\text{in}}_{\text{scale}}\mathbf{W}^{\text{in}}\mathbf{u}(n+d+1) + W_{\text{scale}}\mathbf{W}\mathbf{x}(n) + W^{\text{state}}_{\text{scale}}\mathbf{W}^{\text{state}}\mathbf{s}(n+1) + b_{\text{scale}}\mathbf{b}\right),\end{aligned}$$

and the output weights are calculated by using Equation (3). The network parameters are determined by iterative manual optimization, where first finding a global minimum in training normalized root mean square error(NRMSE) is used as criterion. After this, the mean square error(MSE) in the last couple of testing datapoints is monitored together with testing plots corresponding to different network parameters in the vicinity of the found global minimum with stepsize 0.01 for all parameters. The network parameters achieving the best results follow from this procedure.

In order to use the ESN-controller, it has to be brought into a relevant state. Therefore the helicopter is first "frozen" into a desired position, which allows the ESN-controller to stabilize in a relevant state. By plotting the states(see Figure 12) it becomes apparent when we can start using the ESN-controller for testing purposes. There are more sophisticated ways to solve this problem, e.g. by using a hybrid controller where initially the ESN-controller is updating its state but where its outputs are ignored, but this approach was chosen because it was determined to be sufficient for the task at hand.

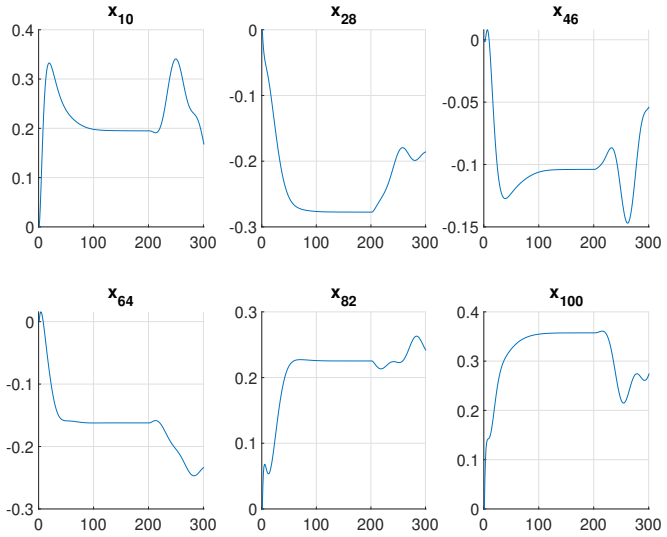


Figure 12: It can be seen that the states reach a "stable" value after about 100 timesteps. After 200 timesteps, the ESN is used for control purposes.

Cutoff frequency f_c	1.0Hz
ESN-controller delay d	0.16s
Spectral radius $\lvert\lambda_{\max}\rvert$	0.88
$W^{\text{in}}_{\text{scale}}$	0.70
$W^{\text{back}}_{\text{scale}}$	0.70
b_{scale}	0.08
Leaking rate a	0.18
Regularization coefficient b	1

Table 2: Network parameters found for experiment 1, in which only altitude of the helicopter is controlled in a simplified model.

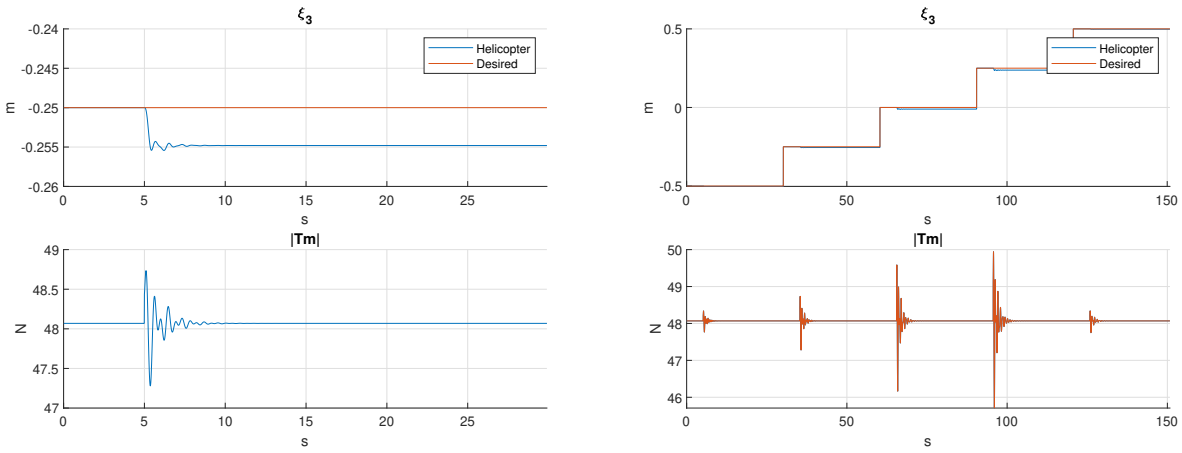
4.1 Controlling Altitude in a Simplified Model

The first experiment was done in a simplified helicopter model, in which only the helicopter's altitude and the velocity in the corresponding coordinate where allowed to change. This was implemented by modifying the forward Euler method such that only ξ_3 and v_3 were updated. Therefore, the inputs Φ, Θ and $|T_t|$ can be discarded and only $|T_m|$ is considered. To introduce more variation in the dataset, the deviation of the normal distribution from which the $|T_m|$ noise is sampled is multiplied by 10. For this task, the following helicopter variables are considered by the ESN:

$$u = \xi_3, \quad \mathbf{s} = \begin{bmatrix} \xi_3 \\ v_3 \end{bmatrix}, \quad y = |T_m|. \tag{25}$$

The network parameters which gave the best results are summarized in Table 2 and with these settings a training NRMSE of 0.121 was obtained.

4.1.1 Stabilizing



(a) (b)

Figure 13: **(a)** Stabilizing the helicopter for 25 seconds in $\xi_3 = -0.25$m after freezing for 5 seconds. **(b)** Repeating this task 5 times in different positions, where the controller and helicopter position are resetted after 30s.

The first task in which the ESN-controller was tested was stabilizing the helicopter in some position for 25 seconds, after freezing its position for 5 seconds to obtain a relevant ESN-state. To verify whether the controller can stabilize in multiple positions, this task was repeated 5 times. The procedure of freezing the helicopter's position was repeated in each testing position. The results of this are plotted in Figure 13. It can be seen that there are some initial control oscillations, but after a while the controller has settled in a value for $|T_m|$, which is equal to mg, the gravity acting on the helicopter. The position in which the helicopter settles is slightly different than the value which is desired, and there is more deviation when the initial control oscillations are larger.

4.1.2 Force Disturbance

In order to investigate whether an ESN-controller would be applicable in real-world scenarios, an experiment is conducted to test the robustness of the ESN-controller. The dynamical model used does not consider external aerodynamic forces, such as those caused by wind and turbulence, and the model assumes rigid body dynamics. In order to account for these disturbances, noise can be added to the force balance of the dynamical model (Equation (8)) to investigate the influence of unaccounted forces to the helicopter. In Figure 14, the result of this is plotted. First, the helicopter is stabilized by the controller, after freezing for 5 seconds. Then, after 10 seconds a disturbance to the force equation is added for 0.07 seconds. It can be seen that this leads to some oscillation in $\xi_3, |T_m|$ but that the controller is able to recover from the disturbance and stabilize the helicopter in the same position as before.

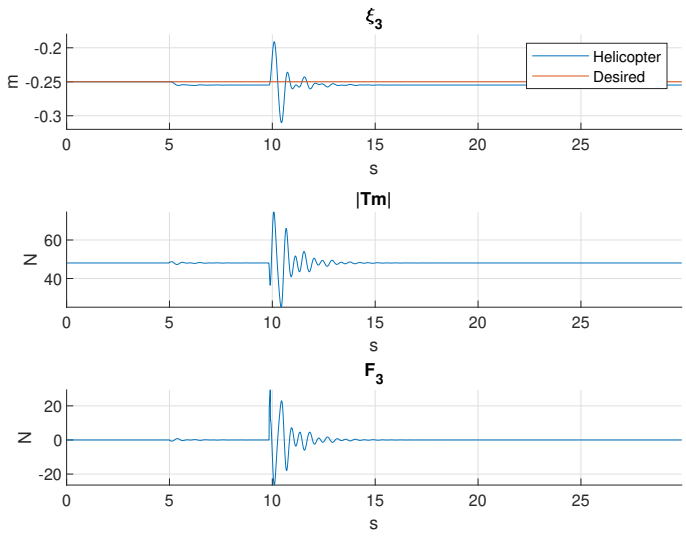


Figure 14: Effect of a force disturbance after 10 seconds. The controller is able to recover from the disturbance and continues to stabilize the helicopter.

4.1.3 Trajectory Tracking

To test the controller on a more challenging task, a sinewave input was given to the controller as reference signal. Before testing, the helicopter is not frozen into one position, but in the position specified by the reference signal. In Figure 15 the result of this is shown and it is clear that after

some initial oscillations, the controller is able to let the helicopter follow this signal. Also, some deviation from the desired signal is visible at all points. Experiments show that this deviation is larger when the initial oscillations are larger, similar to the stabilizing task.

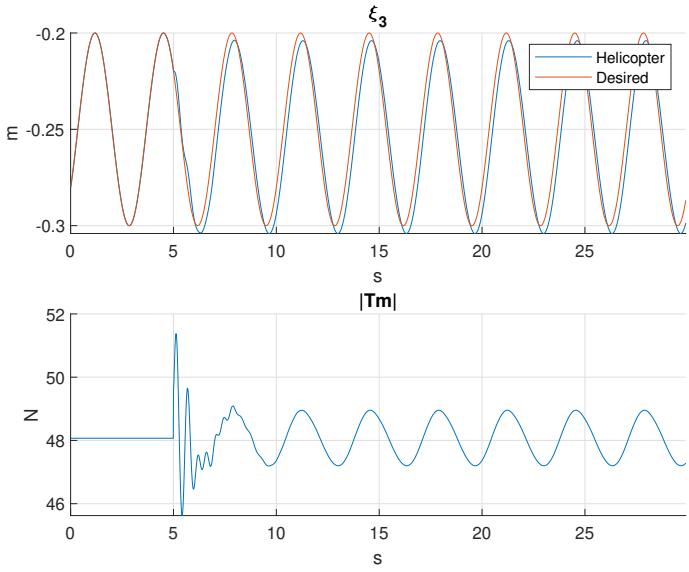
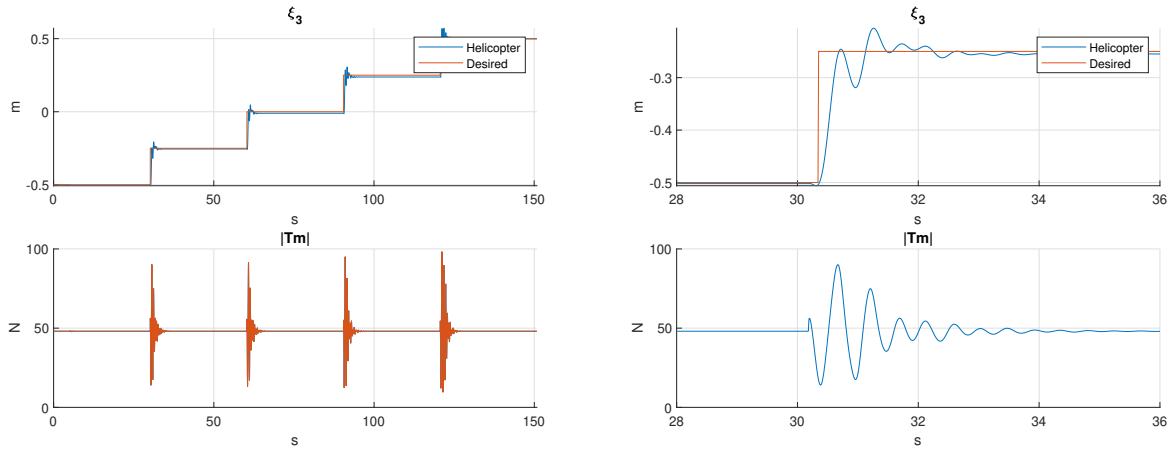


Figure 15: The helicopter follows a reference sinewave signal.

An additional experiment is conducted in which the ESN-controller had to stabilize the helicopter in 5 positions, but now without resetting the controller state and helicopter position after the previous one. So this control objective can also be described as trajectory tracking, where the trajectory is a step function. The result of this is plotted in Figure 16. It can be seen that the initial control oscillations after freezing are of much smaller amplitude than the control input oscillations at the discontinuities.



(a) (b)

Figure 16: **(a)** Stabilizing the helicopter in 5 different positions *without* resetting position and controller state after each position. Initially, the helicopter position is frozen only in the first position. **(b)** Plotting a time interval in which one of the discontinuities in the reference signal occurs.

4.2 Controlling Position and Heading

Cutoff frequency f_c	2.0Hz
ESN-controller delay d	0.04s
Spectral radius $\lvert\lambda_{\max}\rvert$	0.90
$W^{\text{in}}_{\text{scale}}$	0.25
$W^{\text{back}}_{\text{scale}}$	0.18
b_{scale}	0.20
Leaking rate a	0.20
Regularization coefficient b	1

Table 3: Network parameters found for experiment 2, in which the heading and position of the helicopter are controlled.

In this experiment the to be controlled values are the position and heading of the helicopter, which is is specified in $\mathbf{u}$. All helicopter's inputs are available for this task and the best results were obtained when including everything we know about the helicopter state in $\mathbf{s}$. Therefore the following values were used in the signals for the ESN-controller:

$$\mathbf{u}_{\text{ref}} = \begin{bmatrix} \alpha \\ \xi \end{bmatrix}, \quad \mathbf{s} = \begin{bmatrix} \xi \\ \mathbf{v} \\ \alpha \\ \beta \\ \gamma \\ \Omega \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \Phi \\ \Theta \\ |T_m| \\ |T_t| \end{bmatrix}. \tag{26}$$

Training an ESN with low training errors turned out to be a challenging task with many attempts needed before some results came in. In Table 3, the network settings are summarized which achieved the best results.

The obtained training NRMSE's are

$$0.033, \quad 0.026, \quad 0.033, \quad 0.033, \tag{27}$$

for $\Phi, \Theta, |T_m|, |T_t|$ respectively. The network output and teacher signal for episode 1838 are plotted in Figure 17, and it can be seen that these signals are almost indistinguishable.

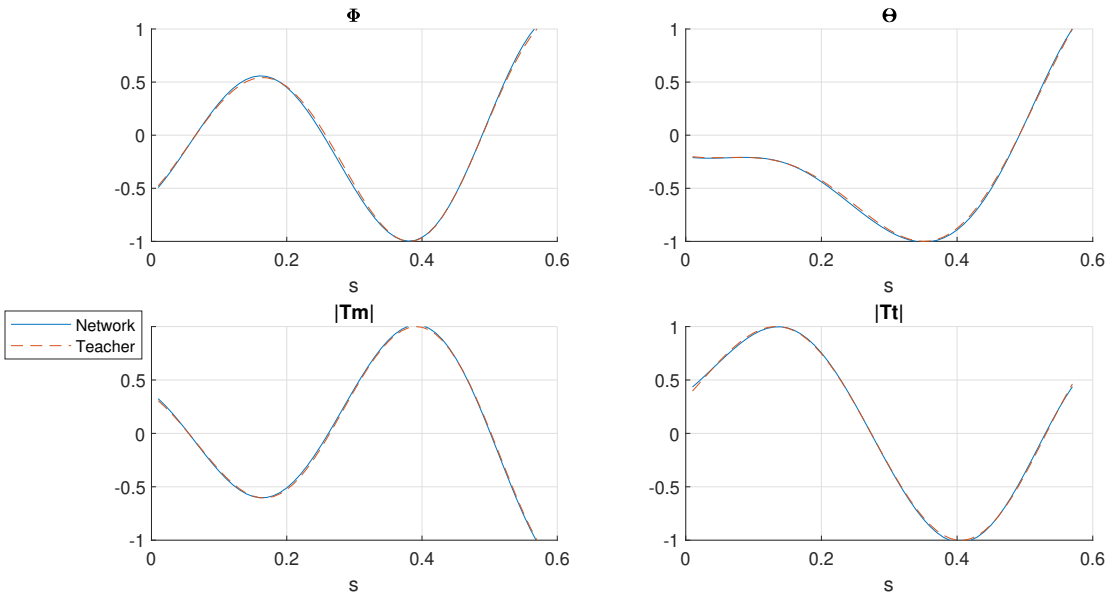


Figure 17: Plotting network output and teacher signal of episode 1838. The values are scaled in between the interval $[-1, 1]$ and are converted back to helicopter scales when used for control purposes.

4.2.1 Stabilizing

After training an ESN-controller, it is tested on a stabilizing task. This was done by stabilizing the helicopter in 5 different positions, after freezing the helicopter's position for 2 seconds. The result of this is plotted in Figure 18. It can be seen that the heading α and position in the horizontal plane specified by ξ_1, ξ_2 is not stabilized in a good manner and that the altitude ξ_3 behaves very undesirably. In all testing scenarios, the control input for the main rotors' thrust force $|T_m|$ is oscillating around a value which is higher than the value of gravity acting on the helicopter. Therefore the helicopter rises quickly, as shown in the bottom left plot(a negative ξ_3 corresponds to the helicopter moving upwards). Since this causes the helicopter to accelerate, this is believed to be the main reason why the helicopter dynamics becomes unstable when attempting to stabilize the helicopter for a longer duration.

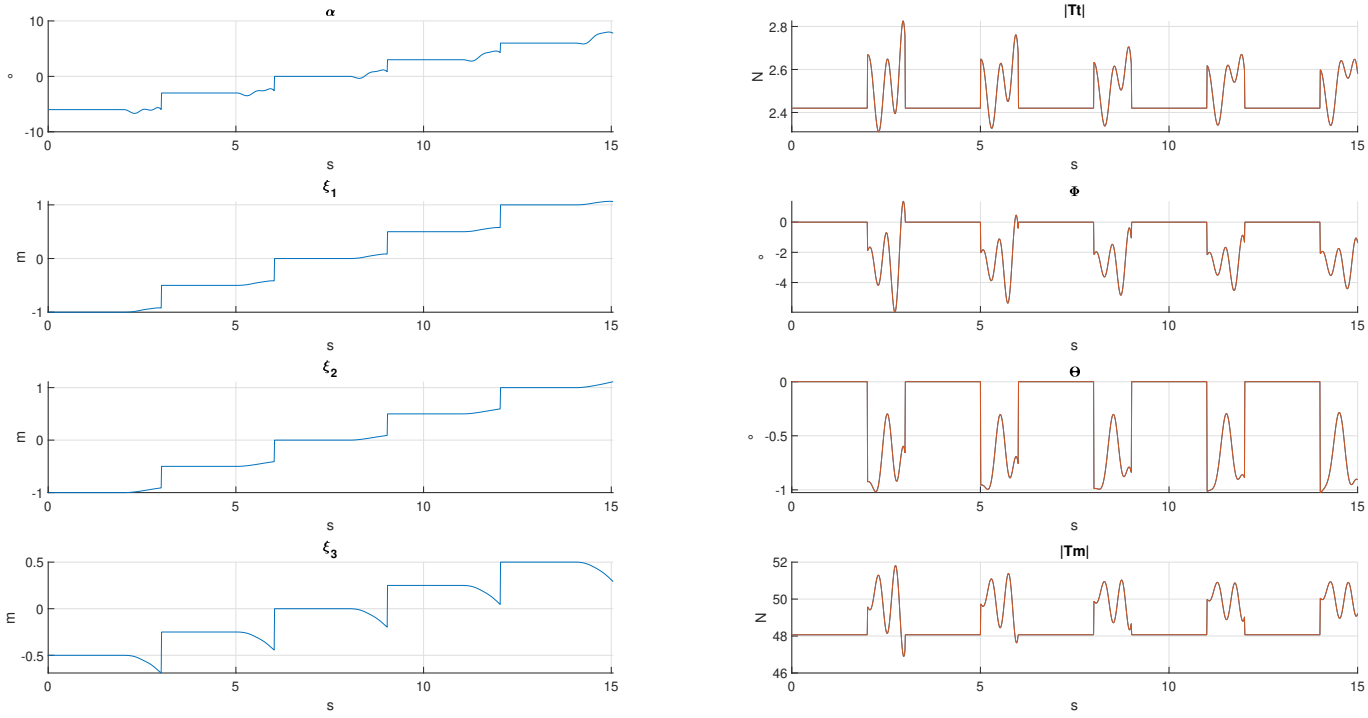


Figure 18: Stabilizing position and heading 5 times for one second

5 Conclusion

For the simplified helicopter model the ESN-controller completed all tasks successfully after some initial control oscillations. The controller is also able to stabilize, recover from disturbances added to the force equations of the helicopter model and to do trajectory tracking. The position in which the helicopter settles is slightly different than the value that is desired, and there is more deviation when the initial control oscillations are larger. The control oscillations are expected to be removed by fine tuning the network settings, with most improvement expected when experimenting with the noise cutoff frequency and the ESN-controller delay. Since the deviation from the desired position is larger when larger initial control oscillations are present, it is expected that the improved controller will also be able to stabilize the helicopter more accurately and follow the desired trajectory more closely.

When giving the ESN-controller the task to control β, γ, undesirable side effects are believed to be the main reason for the ESN-controller not being able to complete its task. This happened as the velocity in the horizontal plane was not monitored by the ESN, and controlling β, γ allowed the helicopter to accelerate in the horizontal plane causing the helicopter to become unstable. Therefore it can be concluded that this is not a good control objective.

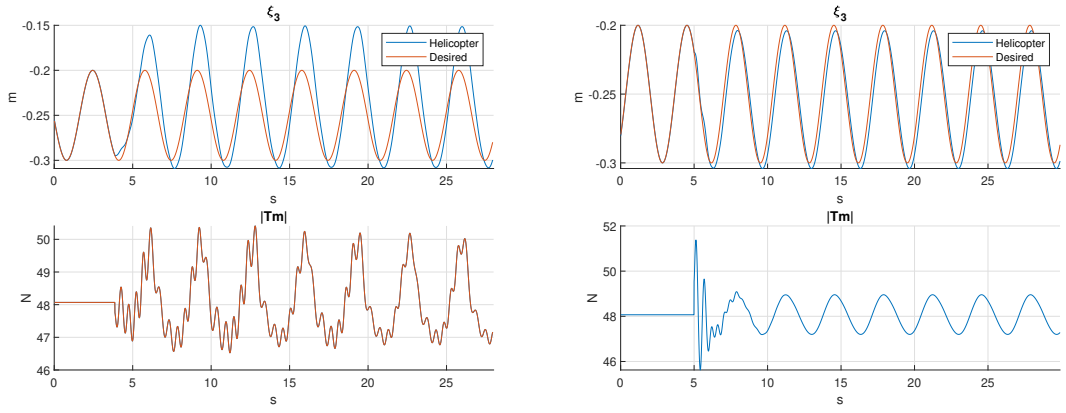
When controlling position and heading of the helicopter, low training error was achieved but undesired behavior during testing was visible. In general, the ESN-controller performed better within regions where the weights associated with input signals were lower, therefore it can be concluded that the values chosen for the cutoff frequency and ESN-controller delay can be improved. The controller was tested on a stabilizing task on 5 different positions and in all positions similar results were achieved. The heading of the helicopter α and the position of the helicopter in the horizontal plane specified by ξ_1, ξ_2 were not stabilized in a good manner, but the main reason for concern was the altitude of the helicopter ξ_3. The main rotors' thrust force $|T_m|$ is the control input responsible for this since it oscillates around a value which causes the helicopter to accelerate upwards.

To conclude, an ESN-based controller shows good potential for helicopter control. Results on a simplified model indicate that the controller is able to complete tasks with great precision and that an ESN-controller is robust enough to recover from disturbances appearing in the helicopter model. In order to use an ESN-controller on the complete model, many parameters need to be set to values such that together they form a controller which is able to stabilize and to track trajectories while being robust enough to recover from disturbances.

6 Discussion

6.1 Experiment Analysis

Both testing and training results turned out to be very sensitive to parameter changes. For example, in the first experiment with the simplified helicopter model the ESN-controller delay was first set to $d = 0.08$s after which is was doubled to $d = 0.16$s. The training NRMSE decreased from 0.129 to 0.121, but the testing results improved more significantly. The influence of this in testing can be seen in Figure 19, where the controller has as objective to follow a sinewave reference signal. The controller using $d = 0.08$s has a gain which is too large and there are more oscillations present in the signal. This shows that careful selection of parameters should be done in order to obtain good testing results.



(a) (b)

Figure 19: **(a)** Following a reference sinewave with $d = 0.08s$. **(b)** Following a reference sinewave with $d = 0.16s$.

During the simulations with the PID-controller, it became apparent that different parts of the helicopter react to inputs on different timescales. For example the helicopter's altitude reacted much slower to the main rotors' thrust force than the helicopter heading reacted to the tail rotors' thrust force. Therefore it is assumed that it would be useful to use multiple ESN-controller delays and cutoff frequencies for the different helicopter inputs and corresponding state variables.

When changing parameters, one should also keep in mind that many parameters are depending on each other. For example, when using a cutoff frequency of $f_c = 2$Hz, the corresponding time period is 0.5s and therefore it is assumed that the ESN-controller delay should be set to values below $d = 0.5$s. This relation seems reasonable, but the results on the full helicopter could not yet confirm this. Therefore it would be useful to test this relation in a simplified model after which it can be used in the full model. This can be extended and other relations can be tested similarly.

In some cases, better results were obtained with smaller gains for the PID-controller. There could be two reasons for this. The first reason is that the helicopter behaves less wild and therefore it is easier for the ESN to understand the dynamics. The second reason is that the PID-inputs change too fast for the helicopter to respond, and therefore the amount of unwanted noise in the training data increases when using higher PID gains.

In experiment 2, the main rotors' thrust force is the control input that seems to be responsible for the bad testing results, since this input oscillates around a value which causes the helicopter to accelerate upwards. An explanation for this might be the relative small ESN-controller delay of $d = 0.04$s, causing the gain of the controller to be too large. This might have the most impact on $|T_m|$ since the timescales at which the helicopter responds to this input were seen to be slower than the other inputs.

In experiment 2, the training errors are low but the testing results are undesirable. One reason for this could be that the standard deviation of the normal distribution from which the noise comes from is set to a too small value. Therefore the set of helicopter dynamics is not rich enough and the ESN controller cannot generalize to new tasks. The same can also be said for the episode length. Since the helicopter starts each episode at rest and β, γ are among the controlled variables, the helicopter behaves more wildly as time progresses. Increasing the episode length might lead to the controller learning more about the dynamics of the helicopter and therefore improving its performance.

6.2 Future Work

The experiment analysis can be summarized by that there are many parameters which should be tuned together in order to obtain a good ESN-based controller, and that a proper balance between the size of the exploration space and the amount of training data has to be achieved. Therefore, suggestions for future work are:

- Similar to experiment 1, one can simplify the helicopter model for other inputs and helicopter variables. For example the model can be simplified such that only the heading is able to change by using the tail rotors' thrust force. By using such models one can optimize some parameters, for example the ESN-controller delay and cutoff frequency, and also find useful relations which can later be used for the full model.

- It would be very useful to quantify the testing error accurately to investigate the influence of the size of the exploration space and training data to controller performance. In this work, the mean square error in the data points at the end of a testing session were used to quantify the testing error and by using this and plotting the results it was determined whether the controller is stable. One improvement could be to look at the exponential divergence rate, which gives a measure for determining whether the obtained controller is stable or not. Once this has been successfully done, it can be used to make qualitative statements about the training data. Using this, one can for example see whether the data has enough variation in it and do experiments to see whether the dataset is large enough.

References

[1] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001.

[2] Dhireesha Kudithipudi, Qutaiba Saleh, Cory Merkel, James Thesing, and Bryant Wysocki. Design and analysis of a neuromemristive reservoir computing architecture for biosignal processing. *Frontiers in Neuroscience*, 9:502, 2016.

[3] Eric A Antonelo, Eduardo Camponogara, and Bjarne Foss. Echo state networks for data-driven downhole pressure estimation in gas-lift oil wells. *Neural Networks*, 85:106–117, 2017.

[4] Athanasios Polydoros, Lazaros Nalpantidis, and Volker Krüger. Advantages and limitations of reservoir computing on model learning for robot control. In *IROS Workshop on Machine Learning in Planning and Control of Robot Motion, Hamburg, Germany*, 2015.

[5] Jessica Alvarenga, Nikolaos I Vitzilaios, Kimon P Valavanis, and Matthew J Rutherford. Survey of unmanned helicopter model-based navigation and control techniques. *Journal of Intelligent & Robotic Systems*, 80(1):87–138, 2015.

[6] Robert Mahony, Tarek Hamel, and A Dzul. Hover control via lyapunov control for an autonomous model helicopter. In *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No. 99CH36304)*, volume 4, pages 3490–3495. IEEE, 1999.

[7] Robert Mahony and Tarek Hamel. Robust trajectory tracking for a scale model autonomous helicopter. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 14(12):1035–1059, 2004.

[8] Kaustubh Pathak and Sunil Kumar 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. IEEE, 2005.

[9] J Andrew Bagnell and Jeff G Schneider. Autonomous helicopter control using reinforcement learning policy search methods. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 2, pages 1615–1620. IEEE, 2001.

[10] H Kim, Michael Jordan, Shankar Sastry, and Andrew Ng. Autonomous helicopter flight via reinforcement learning. *Advances in neural information processing systems*, 16:799–806, 2003.

[11] Andrew Y Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental robotics IX*, pages 363–372. Springer, 2006.

[12] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Advances in neural information processing systems*, pages 1–8, 2007.

[13] Mantas Lukoševičius. A practical guide to applying echo state networks. In *Neural networks: Tricks of the trade*, pages 659–686. Springer, 2012.

[14] T John Koo and Shankar Sastry. Output tracking control design of a helicopter model based on approximate linearization. In *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No. 98CH36171)*, volume 4, pages 3635–3640. IEEE, 1998.

[15] Herbert Jaeger. Echo state networks: introduction and current trends. Jacobs University Bremen, Personal communication, 2009.

7 Appendix

While executing the research, some technical issues occurred. In this section we explain these issues and briefly explain the methods used to address these issues.

7.1 Rotation Matrix

When solving the differential equations regarding helicopter dynamics, the used equation for updating the rotation matrix is given by

$$R(n+1) = R(n) + \Delta t R(n)\breve{\Omega}(n+1). \tag{28}$$

This might lead to the resulting rotation matrix $R(n+1)$ violating the orthogonality property of rotation matrices. To solve this issue, the singular value decomposition was used. That is, instead of Equation (28), we use

$$\begin{aligned} &R'(n+1) = R(n) + \Delta t R(n)\breve{\Omega}(n+1). \\ &\text{Find } U, \Sigma, V \text{ such that } R'(n+1) = U\Sigma V^\top. \\ &\text{Take } R(n+1) = UV^\top. \end{aligned}$$

7.2 Euler Angles

Numerical problems might arise when considering differences in Euler angles. For example, the difference between a heading $\alpha = 180°$ and a heading $\alpha = -179°$ can be overcome with a rotation of $1°$, whereas the numerical difference is $180 - (-179) = 359°$. This problem also occurs when including Euler angles in signals presented to the ESN. In this thesis, this problem is omitted by taking the sine and cosine values of angles for which this may occur. So, for example in Equation (26), where we wrote

$$u_{\text{ref}} = \begin{bmatrix} \alpha \\ \xi \end{bmatrix}, \tag{29}$$

the signal presented to the network is actually given by

$$u_{\text{ref}} = \begin{bmatrix} \sin\alpha \\ \cos\alpha \\ \xi \end{bmatrix}. \tag{30}$$

7.3 Feature Scaling

Before the signals are presented to the network, feature scaling is used such that all values are in the interval $[-1, 1]$. For a value x, this is done by first finding a relevant minimum $\min(x)$ and maximum $\max(x)$ such that it always holds that $x \in [\min(x), \max(x)]$, after which we take

$$x' = -1 + 2\frac{(x - \min(x))}{\max(x) - \min(x)}, \tag{31}$$

which ensures that $x' \in [-1, 1]$. To convert network values back to the original scaling, we use

$$x = \min(x) + \frac{(x'+1)(\max(x) - \min(x))}{2}. \tag{32}$$