Echo state networks

TTK4550 - Engineering Cybernetics, Specialization Project

Desember 18 2018

Written by

Sivert Vonen

NTNU – Trondheim
Norwegian University of
Science and Technology

Summary

The purpose of this project is to provide a better understanding of Echo State Network (ESN) in the contexts of system identification. This is an artificial neural network and it is used as a ”black-box” model approach. ESN is a subgroup of Recurrent Neural Network which means it has a memory. This gives an elegant solution to model history dependent systems.

ESN was implemented and trained to model a nonlinear system. The example chosen to model in this project, was the velocity of the height of a tank based on volume inflow. The training data was also contaminated by measurement noise. The results from this test was that ESN in a satisfying way was able to model the tank height velocity.

For ESN there exist a stability property called echo state property, which is true if it is able to ”forget” the initial states over time. This report elaborate on the tuning of the range of initialized weights to achieve the echo state property. For this example the best performance was found in the boarder area of the echo state property. This result supports known theory on how to select parameters [9].

Another positive aspect of ESN is that it is easy to train, it is done with regression. This report explores the differences in using Ridge, OLS and LASSO regression. Ridge and LASSO showed good results while OLS showed variable results. LASSO provides a simpler model but is more vulnerable to overfitting than ridge.

NTNU

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Engineering Cybernetics

PROJECT DESCRIPTION SHEET

Name of the candidate: Sivert Vonen

Thesis title (English): Echo-State Networks

Background

Echo State Networks (ESN) are a type of Recurrent Neural Networks (RNN) that can be used for modelling certain classes of nonlinear dynamic systems, while at the same time being easy to train, which makes them suitable for system identification and control applications. This task should investigate ESNs for modelling, and potentially optimization.

Work description

1. Give an introduction to ESN, in the contexts of system identification and (recurrent) neural networks.
2. Discuss what classes of nonlinear systems can be suitably approximated by ESN.
3. Implement training of ESN.
4. As far as time permits, investigate to what extent the structure of ESNs make them suitable for dynamic optimization.
5. Test system identification based on ESN on a case study selected together with the supervisor.

Start date: August 22, 2018 **Due date:** December XX, 2018

Supervisor: Lars Imsland
Co-advisor(s):

Trondheim, August 22, 2018

Lars Imsland
Supervisor

Address
Sem Sælandsvei 5
NO-7491 Trondheim

Org.no. 974 767 880
E-mail:
postmottak@itk.ntnu.no
http://www.itk.ntnu.no

Location
O.S. Bragstads plass 2D
NO-7034 Trondheim

Phone
+47 73 59 43 76
Fax
+47 73 59 45 99

Phone: +47 47 23 19 49

Table of Contents

List of Figures

Abbreviations

ESN	=	Echo State Network
RNN	=	Recurrent Neural Network
OLS	=	Ordinary Linear Squares
LASSO	=	Least Absolute Shrinkage and Selection Operator
RSS	=	Residual Sum of Squares

Chapter 1

Introduction

This project aim is to provide a better understanding of Echo State Network (ESN) . Echo states networks can be used for ”black box” modeling of certain nonlinear problem. ESN’s main advantage is that they are easy to train.

1.1 Motivation

System identification is the process of deriving a mathematical model of a system using observed data. It is an important subject in optimization and control where a good model is essential. Most real-world processes is nonlinear and cannot be completely modeled or a considerable modeling effort is needed.

The use of artificial neural networks to system identification is today a hot topic in the industry. It can make a model only based on data instead of the laws of physics. Neural networks consist of many neurons which again consist of inputs, weights, input function and an activation function, see fig.1.1. The input to a neuron can either be an output from other neurons or the input to the network. A combination of neurons with a

special combination of weights can be a model for a nonlinear system. The combination of weights are found through a learning process [11].

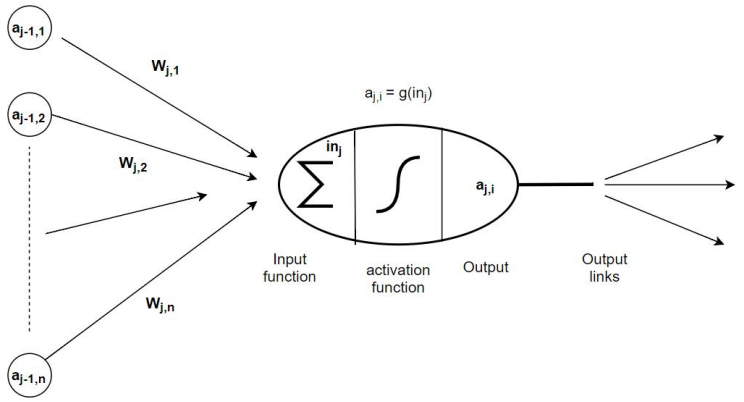
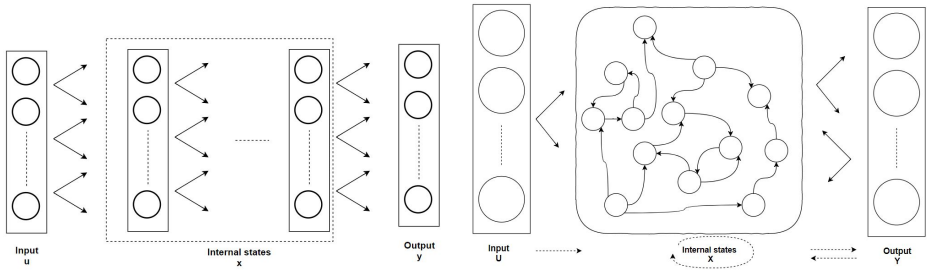


Figure 1.1: Neuron, Drawn freely after [11]

Recurrent neural networks are a subgroup of Artificial Neural Networks. What separates Recurrent neural networks from other types is that it has a memory of previous inputs and/or outputs. This is in contrast to feedforward neural networks where one set of input only can give one output, thus independent of the history of inputs. Notice in fig 1.2 that in RNN the direction of the internal states x goes back to itself instead of just forward like in feedforward. However a feedforward network can have memory outside the network and thus have the history as input. Memory makes Recurrent neural networks applicable for system prediction of systems that are dependent of the history. One of the most popular types of RNNs are deep learning using gradient descent on all hidden weights. The drawbacks of this method are the long time to train and the vanishing gradient problem. This can give a slow convergence and no guarantee for reaching a global convergence [5].



(a) Feedforward network, drawn freely with inspiration from[11]

(b) ESN subgroup of RNN, Drawn freely with inspiration from [7]

Figure 1.2: Structure of different neural networks

ESNs are a type of Recurrent Neural Network that can be used for modelling certain classes of nonlinear systems, while at the same time being easy to train using one-shot regression instead of gradient descent. This makes them suitable for system identification and control application [8].

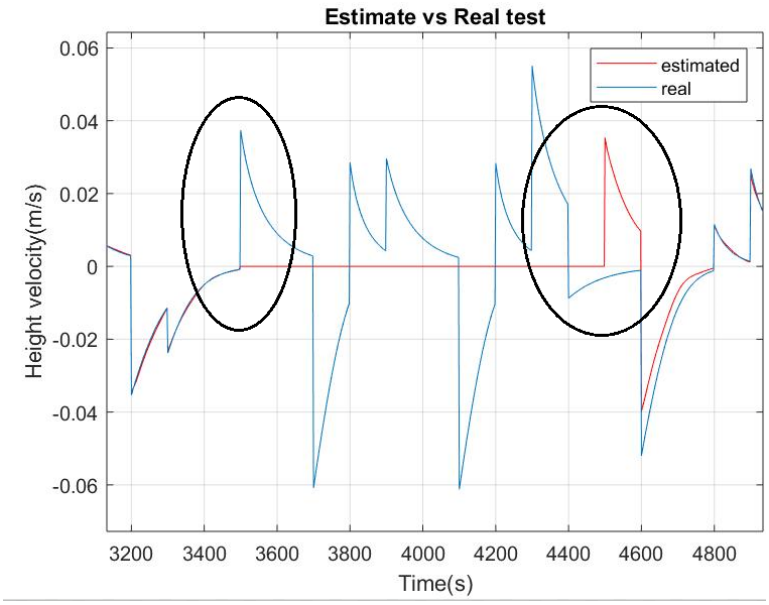


Figure 1.3: ESN test with hold from time 3500 to 4500

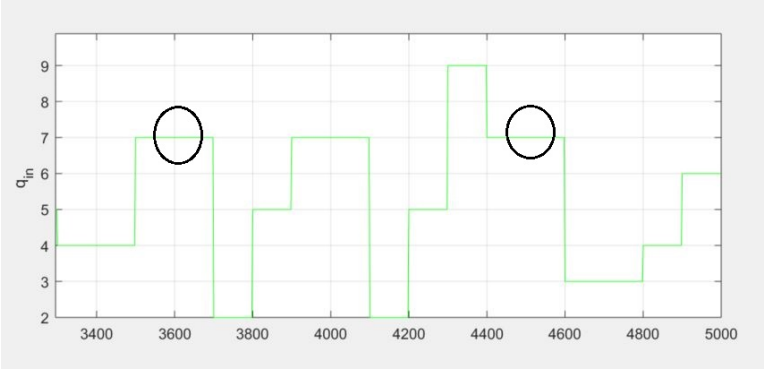


Figure 1.4: Input related to fig. 1.3.

Fig 1.3 is a test where an ESN estimates a function but is temporarily put on hold. When the network starts to estimate again, it contains the memory from when it was put on hold. In addition, the input for the system is the same when it went on hold and when it began estimating again. So it estimates the path for where it was put on hold. It i also worth to notice that it forgets the memory with time so it follows good in the end. **This was meant to illustrate the same behaviour we also find in an echo. It keeps repeating but slowly fades away.** Other positive sides to ESN is that it allows to add output from same internal states, by just training the output weights separately. Echo state network was proposed by Herbert Jaege in 2001 [6]. It has successfully been used in system identification several times, one example from the oil industry is: downhole pressure estimation in gas-lift oil wells [1].

1.2 Structure of Work

The main part of this task is based on an example with a simulated tank. This was implemented to test system identification based on ESN. The system to be identified is the derivative of the height of a fluid inside a tank which is a nonlinear problem.

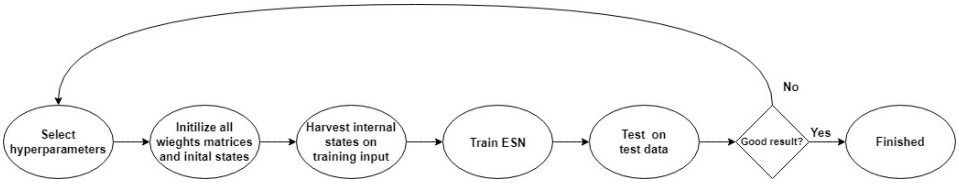


Figure 1.5: Work flow when working with ESN

There are several choices and hyperparameters that has to be considered before one can obtain an ESN. This report will focus on the size of the reservoir, within what range the weights are in, leaking rate, and method of learning. Figure 1.5 gives an overview of the main part of implementing ESN. Harvesting states based on training input is needed since training of ESN is based on internal states. Good result is mainly based on the error between estimated and real value, but there will also be a discussion about stability, efficiency, and simplicity.

Chapter 2

Background

2.1 Mathematical properties of Echo state network

The main body of ESN that separates ESN from other network is the way internal states (neurons) X and output Y is updated. They have the following equations [9]:

$$x[n] = (1-\alpha)x[n-1] + \alpha f(W_{in} * u[n] + W * x[n-1] + W_{fb} * y[n-1]) \quad (2.1)$$

$$y[n] = W_{out} * x[n] \quad (2.2)$$

The memory features of an ESN can here be seen by the fact that new states are dependent on prior states. The structure represented in fig.1.2 b) and the neuron 1.1 is in more detailed described by equation 2.1 and 2.2.

Consider an Echo state network with N_u inputs, N_x internal states and N_y outputs. W_{in}, W, W_{out} and W_{fb} are the weight matrices of the Echo state network. Their given dimensions are: input weights $W_{in} \in \mathbb{R}^{N_x \times N_u}$, state reservoir $W \in \mathbb{R}^{N_x \times N_x}$, output weights $W_{out} \in \mathbb{R}^{N_y \times N_x}$, and feedback weights $W_{fb} \in \mathbb{R}^{N_x \times N_y}$

$\alpha \in (0, 1]$ is a hyper-parameter called the **leakingrate** which decides how fast the states forgets previous states, input and output. This weights the influence previous states and input has on the new states. For the special case of $\alpha = 1$ we say the model is without leaky integration. Leaky integration is mathematical term that is used to describe a component or system that takes the integral of an input, but gradually leaks a small amount of input over time. To include a leaking rate means we have a leaky integrator ESN.

f is a sigmoid function. A sigmoid function is a function being shaped as like an "S". There exist several examples that can be used here and the most common are: the logistic function, tanh, and arctan. For a neuron this is called the activation function [4].

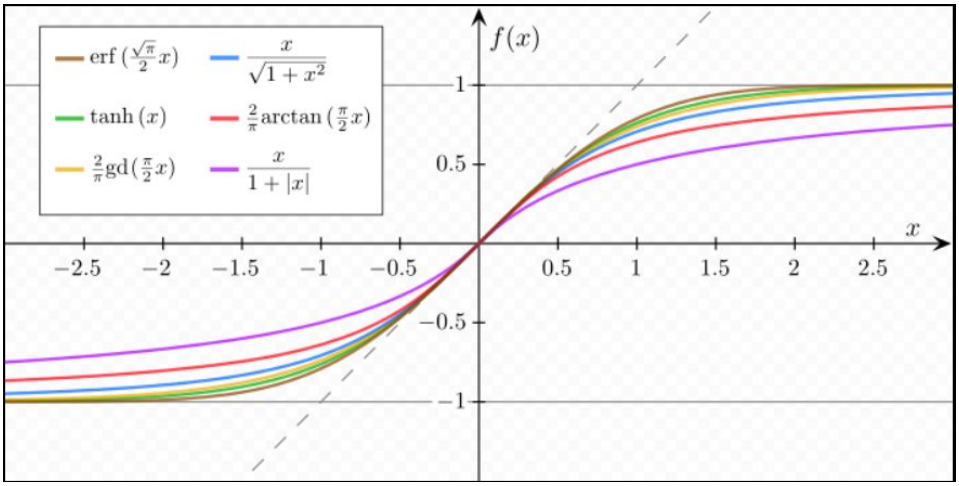


Figure 2.1: Sigmoid functions from [2]

2.2 Training of Echo states network

In general the weight matrices W_{in}, W and W_{fb} are chosen arbitrarily with a uniformly distribution and W_{out} is trained using regression [9]. The idea behind this approach is that for a sufficient given random reservoir one linear combination of the states will be a good approximation for the system. W_{out} Will then represent this linear combination. To train an Echo state network one need to have initialized W_{out} usually uniformly random

and obtained training data for a training period $n = 1, .., T$ which consists of the input $\mathbf{U} \in \mathbb{R}^{N_u \times T}$ and the output $\mathbf{Y_{target}} \in \mathbb{R}^{N_x \times T}$ training data for the system. Then run through the equations 2.1 and 2.2 with training inputs $\mathbf{U}$ for all timesteps in order to collect states $x[n]$ and output $y[n]$ over the timeseries combining them to a new matrices $\mathbf{X} \in \mathbb{R}^{N_x \times T}$ and the same for output $y[n]$ to give $\mathbf{Y} \in \mathbb{R}^{N_y \times T}$. This relates to harvesting states in fig. 1.5 [9].

The training of W_{out} has the goal of minimizing the error between the output $\mathbf{Y}$ and $\mathbf{Y_{target}}$. The training is then done with one shot learning using regression. The different methods will be discussed next in this chapter.

2.2.1 Ordinary Least Squares (OLS) Regression

Ordinary Least Square is a standard method for estimating parameters in linear regression. It takes a set of known data points (x_i, y_i) and produces a hyperplane based on this. To make the hyperplane it can be represented as a minimization problem with the objective function:

$$\mathbf{RSS}_{OLS} = \sum_{i=1}^{n}(y_i - (\beta_0 + \sum_{j=1}^{p}\beta_j x_{ij}))^2 = \|\mathbf{Y_{target}} - \mathbf{XW_{out}}\|_2^2 \tag{2.3}$$

$\beta \in \mathbb{R}^{p+1 \times 1}$ is the variables that will be solved in the optimization problem. The vector consisting of all β is $\mathbf{W_{out}}$ in an ESN. Recall from equation 2.2 how $\mathbf{Y}$ is estimated. The hyperplane is made up of these parameters and states $\mathbf{X}$. $\mathbf{RSS}$ stands for Residual Sum of Squares. A residual is the distance between our data point (x_i, y_i) and the hyperplane.

The analytic solution will be:

$$\mathbf{W_{out}} = (X'X)^{-1} X'\mathbf{Y_{target}} \tag{2.4}$$

2.2.2 Ridge regression

Ridge regression is also known as Tikhonov regularization and $L_2 - Regularisation$. It consist of the same part as $\mathbf{RSS}_{OLS}$ in 2.3 and a parameter λ times the weights of W_{out}^2. This new terms adds a penalty on high weights [3].

$$RSS_{\text{Ridge}} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{i=1}^{p} \beta_j^2 = RSS_{OLS} + \lambda \sum_{i=1}^{p} \beta_j^2 \quad (2.5)$$

The analytic solution will now be [3]:

$$\mathbf{W_{out}} = (X'X + \lambda I)^{-1} X' \mathbf{Y_{target}} \quad (2.6)$$

2.2.3 LASSO

Least Absolute Shrinkage and Selection Operator LASSO is a regularization algorithm much similar to Ridge. The difference is that it contains the absolute value of the weights instead of squared. The result is that now weights belonging to irrelevant states will be set to zero [12].

$$RSS_{LASSO} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{i=1}^{p} |\beta_j| = RSS_{oLS} + \lambda \sum_{i=1}^{p} |\beta_j| \quad (2.7)$$

2.3 Echo State Property

The echo state property is a basic stability property for the network. It says that an ESN needs to be independent of it's initial states. The reason it is so important to be able to forget the initial states is that for the first iteration there is no history. The inital state is therefore set to zero and be wrong. Therefore it will have a hard time to estimate until it do have a sufficient history. On the next page follows a definition, sufficient condition and necessary condition for echo state property as proposed in [8].

Definition[8] An ESN with reservoir states $\mathbf{x(n)}$ has the echo state property if for any compact $\mathbf{C} \subset \mathbb{R}^{\mathbf{k}}$, there exist a null sequence $(\delta_h)_{h=1,2,...}$ such that for any input sequence $(\mathbf{u(n)})_{\mathbf{n=0,1,2},...} \subseteq \mathbf{C}$ it holds that $\|\mathbf{x(h)} - \mathbf{x'(h)}\|_2^2 \leq \delta_h$ for any starting states $\mathbf{x(0)}, \mathbf{x'(h)}$ and $h \geq 0$

Sufficient condition[8] Assume leaky a integrator ESN according to equation 2.1 where the sigmoid f the tanh function and there are no output feedbacks, that is, $\mathbf{W_{fb}} = \mathbf{0}$. Let σ_{max} be the maximal singular value of $\mathbf{W}$. Then if:

$$|1 - \alpha(1 - \sigma_{max})| \leq 1 \tag{2.8}$$

is satisfied. The ESN has the echo state property. $|1 - \alpha(1 - \sigma_{max})|$ is a global Lipschitz rate by wich any two states approach each other in a network update.

Necessary condition[8] Assume a leaky integrator ESN according to equation 2.1, where the sigmoid f is the tanh function. Then we look at the matrix:

$$\tilde{\mathbf{W}} = \alpha\mathbf{W} + (\mathbf{1} - \alpha)\mathbf{I} \tag{2.9}$$

Where $\mathbf{I}$ is the identity matrix of same size as $\mathbf{W}$. If $\tilde{\mathbf{W}}$ has a spectral radius greater than 1 the ESN does not have the echo state property. This gives the Necessary condition for an ESN to have the echo state property:

$$\rho(\tilde{\mathbf{W}}) = \max(|eig(\tilde{\mathbf{W}})|) \leq 1 \tag{2.10}$$

A suggestion for practical use of ESN that is based on empirical data suggests that the if spectral radius of W is smaller than 1. ESN have the Echo state property. This is however not a proof nor does a larger spectral radius means that it does not have echo state property. It was also suggested that values close to 1 is a good choice [9].

$$\rho(\mathbf{W}) = \max(|eig(\mathbf{W})|) \leq 1 \tag{2.11}$$

Another practical suggestion to check if it not have the echo state property is to implement zero input to the ESN and then check if the system is unstable. If it is unstable for zero input it does not have the echo state property [8]. Related to the definition it can be seen that different initial states will then provide different states even if the weights and inputs are the same.

Chapter 3

Experiment

This section look at an example of system identification on a nonlinear problem, the velocity of the height of a fluid inside a tank. This experiment was implemented in Matlab and simulink and can be found in the appendix.

3.1 Tank Dynamics

In this section we will go through the dynamics of the velocity of a fluid inside a tank. The tank is described by fig. 3.1.

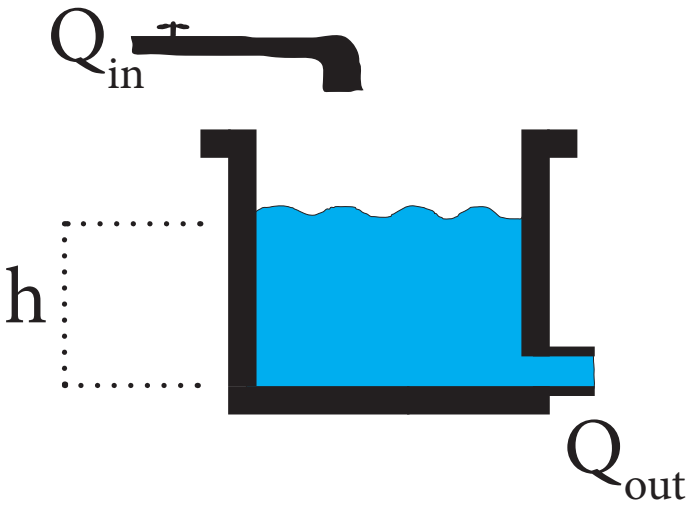


Figure 3.1: An illustration of tank

Several simplification is done in this model. We assume that the flow is steady and incompressible, that the friction by viscous forces is zero, that neither work nor heat is exchange between the surface and outflow. With these assumptions the Bernoulli equation is valid.

The change of volume inside a tank is equal to the change of height times the area of the bottom. Since we assumed incompressible fluid the change of volume is equal to the difference between inflow and outflow.

$$\dot{V}_{tank} = \dot{h}A_1 = Q_{in} - Q_{out} = Q_{in} - A_2 * v_2 \tag{3.1}$$

Next step is to use the Bernoulli equation:

$$\frac{v_1^2}{2} + \frac{\rho}{P_1} + gz_1 = \frac{v_2^2}{2} + \frac{\rho}{P_2} + gz_2 \tag{3.2}$$

Both sides ends in the atmosphere meaning $P_1 = P_2 = P_{atm}$. We also make the assumption $v_1 << v_2$ meaning $v_2^2 - v_1^2 \approx v_2^2$. The reference height is set with $z_2 = 0$ and $z_1 = h$. From Bernoulli we now get the following expression for v_2:

$$v_2 = \sqrt{2gh} \tag{3.3}$$

Combining both equations

$$\dot{h} = \frac{Q_{in}}{A_1} - \frac{A_2}{A_1} \cdot \sqrt{2gh} \tag{3.4}$$

Radius of tank is $r_1 = 5m$, Radius of outflow pipe is $r_2 = 0.5m$. The inflow is discrete integer values in the range from $2 - 6[m^3/s]$. The inflow changes every 100 seconds to a random valid number. The output is measured every 2 seconds. Equation 3.4 implemented results in fig. 3.2.

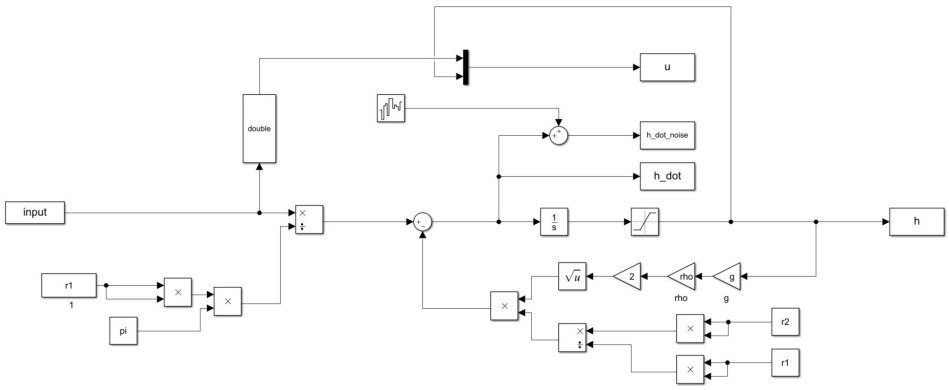


Figure 3.2: Simulink model of tank dynamics.

3.2 Initialization of Weights

There are several factors to be chosen for this part of producing an ESN. Most significantly there is the size of the reservoir N_x and the distribution of weights in W.

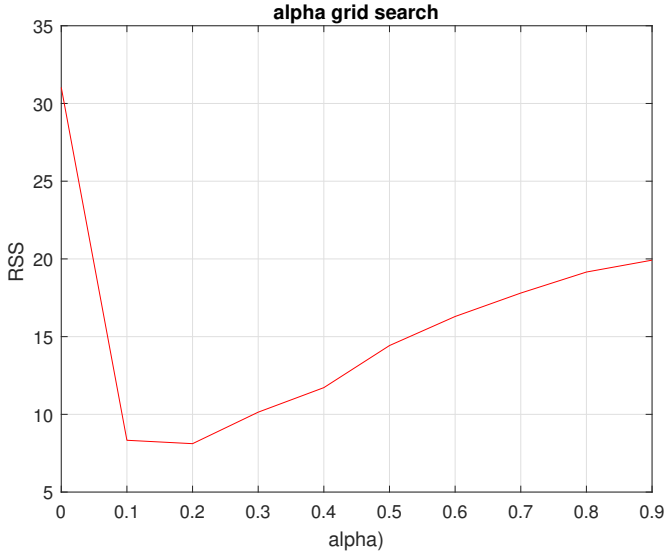
A good size for the reservoir is related to the limit of computation time, complex-

ity of the system, acceptable error, available training data, ”memory” needed. The larger the reservoir size N_x the better is the performance. The drawback is that the time to train increases exponentially. Therefor $N_x = 1000$ was chosen. As ESN still has reasonable training time.

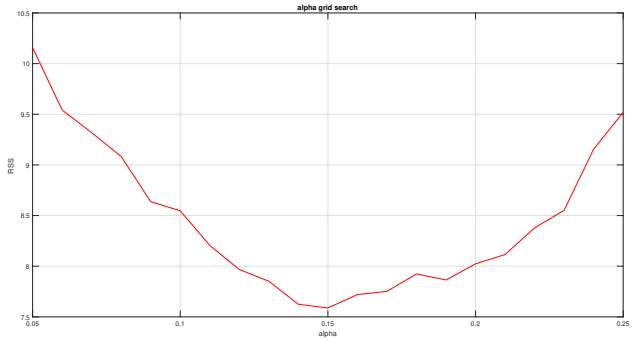
There are several different suggestions for distribution of W, W_{in}, and W_{fb} including uniformly, normal, discrete bi-valued, and a sparse distribution. Normal and uniformly distribution gives much of the same performance. A sparse matrix will set most of the values to zero and thus make the computation more effective. Discrete bi-valued distribution is more interpretive but typically gives poorer performance due to its space of outcomes decreases. The range of the distribution C will affect the spectral radius of W, therefor it's important to choose a value that does not break the necessary condition of Echo state property. Therefore it will be chosen after further testing. To include feedback W_{fb} or not is depending on the task. The general approach is to only include it if the task requires it due to the complexity as it might decrease the stability and simplicity. For this task a uniform distribution around zero is chosen, the reason behind this is its continuity of values and boundedness.

3.3 Leaking Rate

Selecting leaking rate α was done by a line search with α against the sum of errors ($\sum_{i=200}^{T} |y_i - W_{out}x_i|$) in test data. The error does not include the time to converge 200s. The search was done two times. First with large intervals fig.3.3 a) and then a smaller between 0.05 and 0.25 fig.3.3 b). The smaller area was chosen because the sum of error where smallest at $\alpha = 0.1 \quad and \quad \alpha = 0.2$. The line search was done using ridge with $lambda = 0.8$, $N_x = 1000$, and $C = 0.05$. The training and test data was the same for all iterations. The value chosen was in the end $\alpha = 0.15$ as it gave the best result fig. 3.3b).



(a) line search with intervals of 0.1



(b) line search with intervals of 0.05

Figure 3.3: line search to find best leaking rate α. Note y-axis should say error not RSS

3.3.1 Echo State Property

The approach for selecting range of distribution of weights C was based on echo state analysis and performance. What is meant by performance in this context is the sum of errors between estimated values with ESN and real value, excluding the time to converge (100s). Same as in section 3.3. The echo state analysis consist of sufficient condition,

necessary condition and $spectralradius(W)$ (as described in section 2.3) plotted against values of C. The analysis was done using ridge with $lambda = 0.8$, $N_x = 1000$, and $\alpha = 0.15$. The training and test data was the same for all iterations.

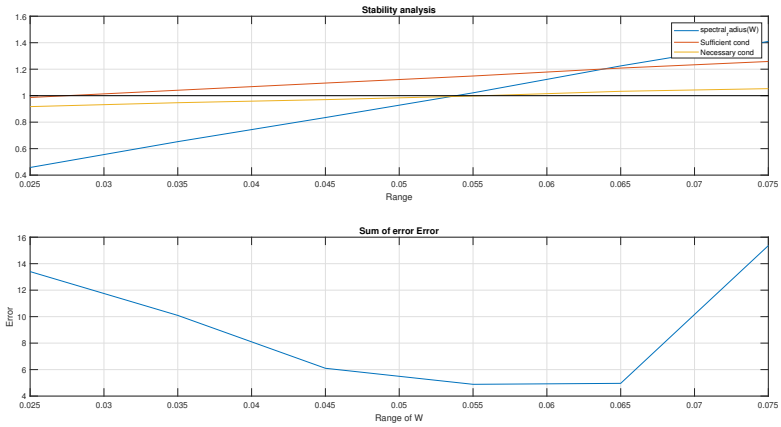


Figure 3.4: Echo state analysis (above) and error (under) as function of C with interval (0.025:0.05:0.075)

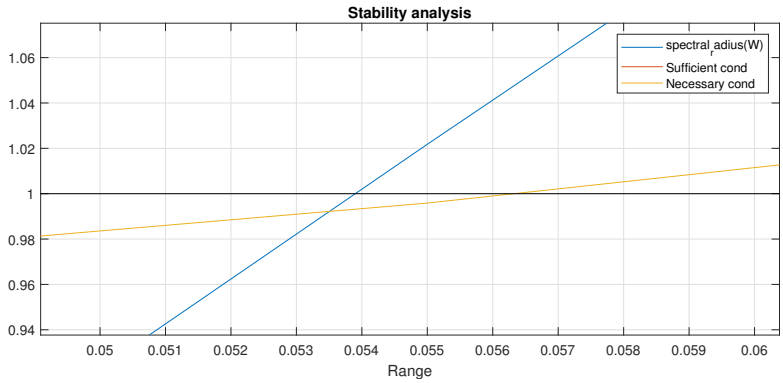
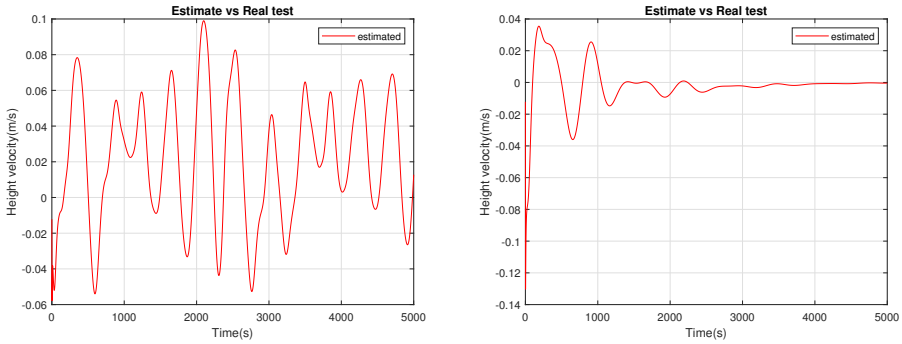


Figure 3.5: Echo state analysis of fig.3.4 zoomed

From fig.3.4 and it can be seen that the sum of errors is at its lowest with C in the area of 0.055 to 0.065. However neither of those values satisfies the sufficient condition and have a spectral radius larger than 1. For $C = 0.055$ it is barely under the necessary condition fig. 3.5 while for $C = 0.065$ is just above. To further investigate this a zero

input signal was sent. The result from fig. 3.6 shows that C=0.065 is unstable and does not have the echo state property.



(a) C=0.065 **(b)** C=0.055

Figure 3.6: zero input stability test

The final test to check if the ESN for $C = 0.055$ has the echo state property is to use the definition described in section 2.3. The initial condition of x was selected with random numbers between 0 and 1 and with all zeros. From fig. 3.7 and fig. 3.8 wee can see that the sum errors between all internal states from two different initial states $\sum_{i=1}^{N_x} |x_i - x_i'|$ decreases to zero meaning it becomes independent of the initial state. It is therefore likely to have the echo state property and testing for all inital states would prove the echo state property.

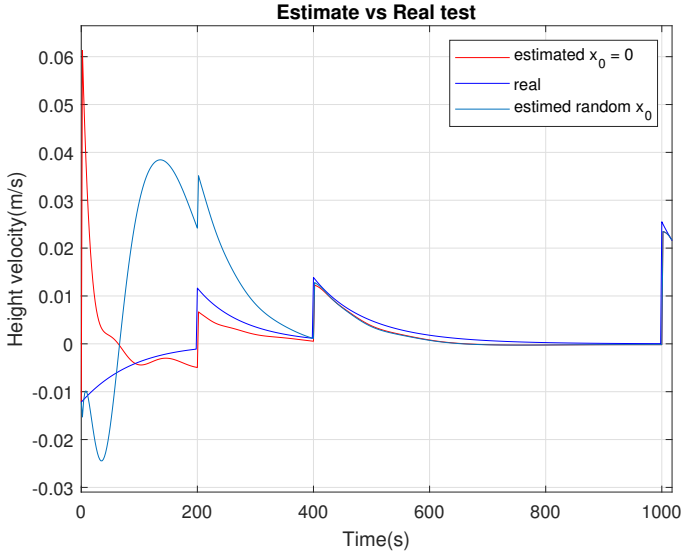


Figure 3.7: Estimations with different initial internal states

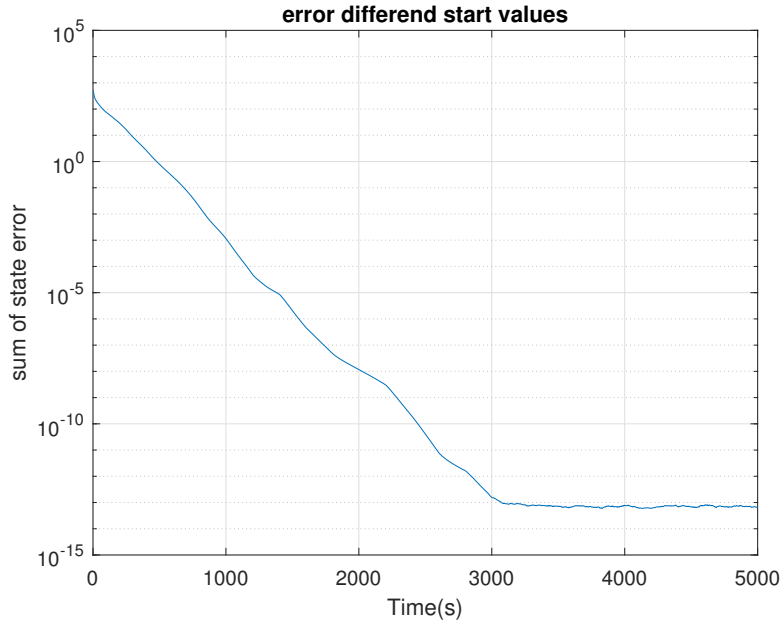


Figure 3.8: Error between states (Note logarithmic scale)

3.4 Simulation

All simulation was done with a training set of 5000 sampling points while changing the input 50 times and a test set with 2500 sampling points while the input changes 25 times. All simulations in this section used $\alpha = 0.15$, $c = 0.55$, $N_x = 1000$, tanh as sigmoid function, and Ridge regression is used with $\lambda = 0.8$.

Figure 3.9: Training set: height of tank and input

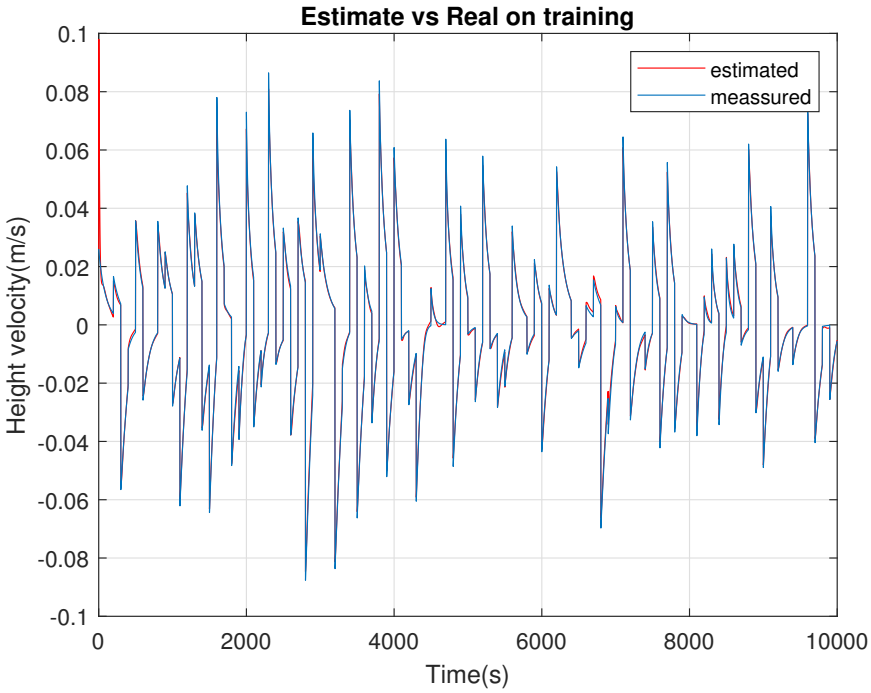


Figure 3.10: Training set: results of estimation

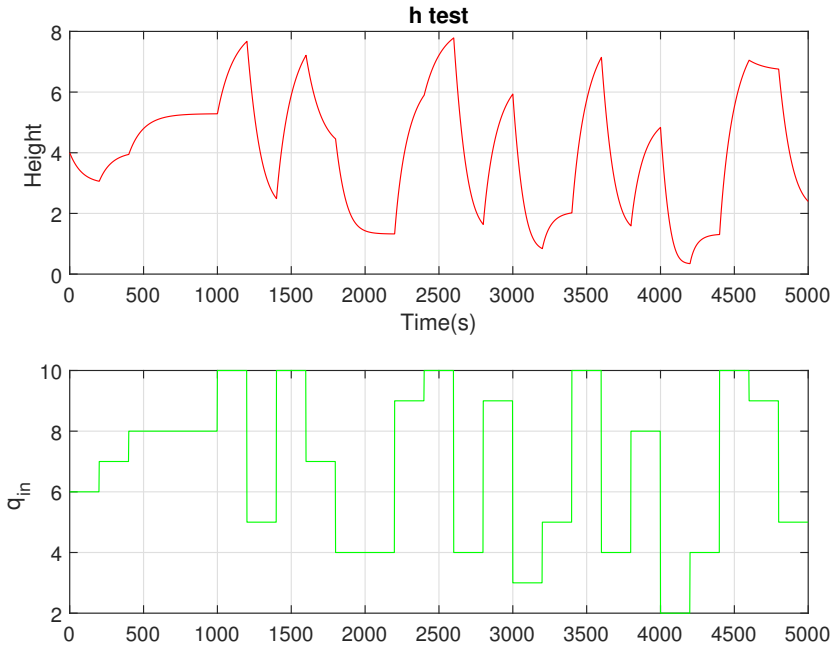


Figure 3.11: Test set:height of tank and input

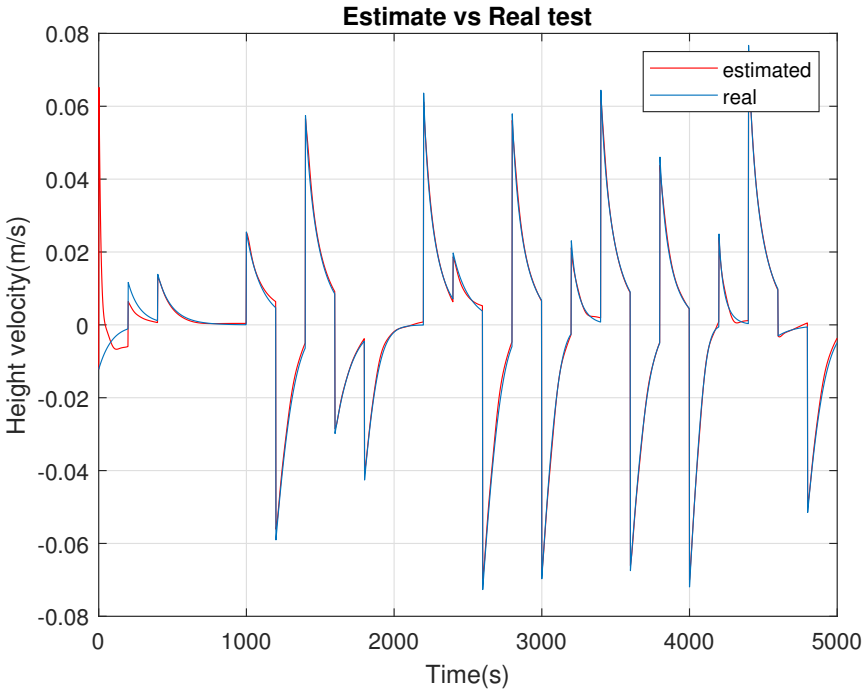


Figure 3.12: Test set: results of estimation

3.4.1 System Identification with Noise

White noise was added to Y_{target} from the training set as can be seen in 3.2. This represents measurement noise from the real world.

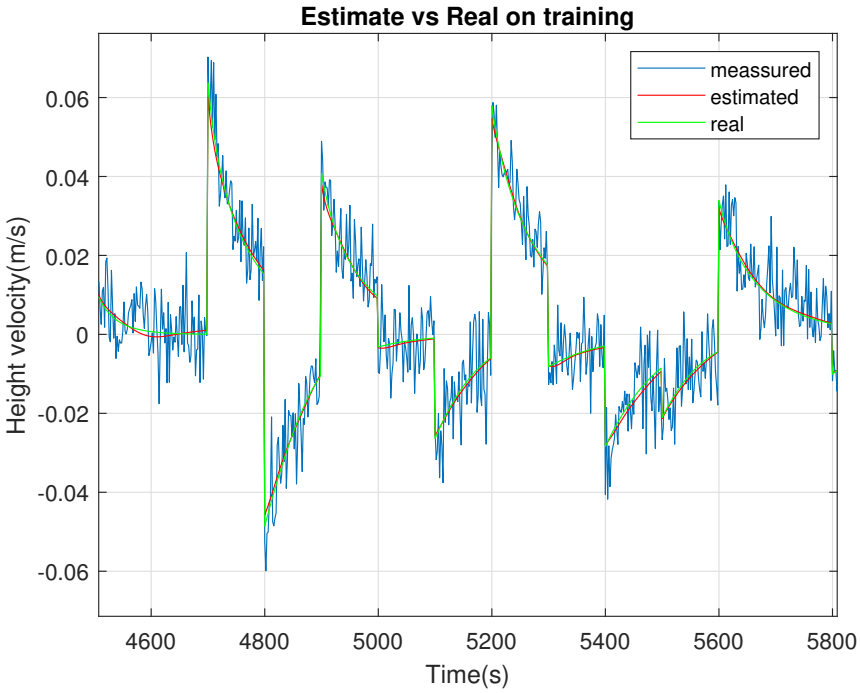


Figure 3.13: Training set: results of estimation with noise

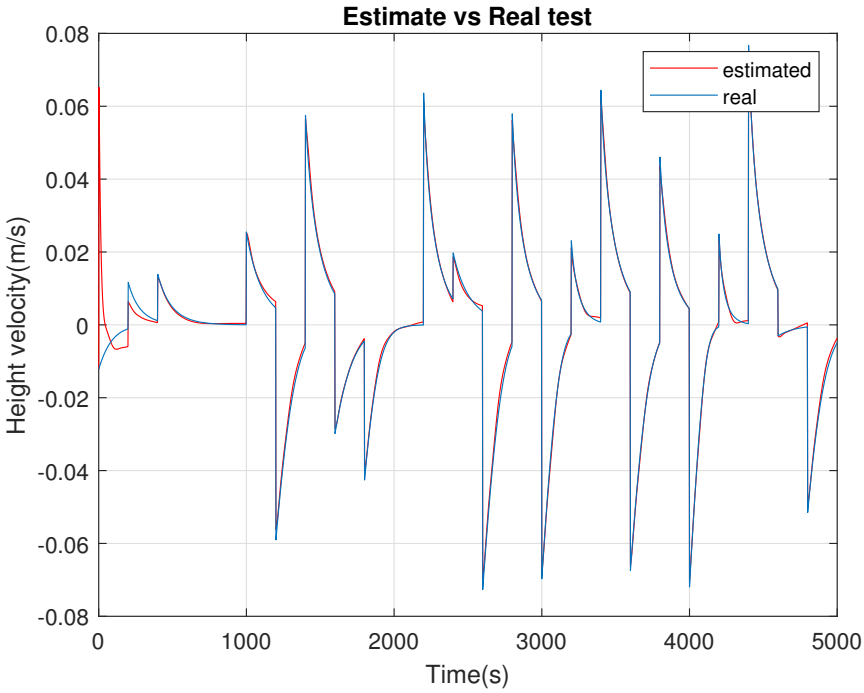


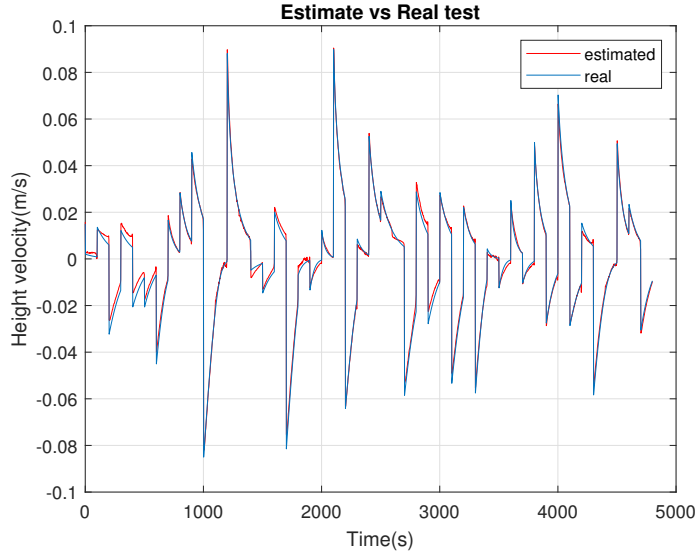
Figure 3.14: set: results of estimation, trained with noise

3.4.2 System Identification using Feedforward

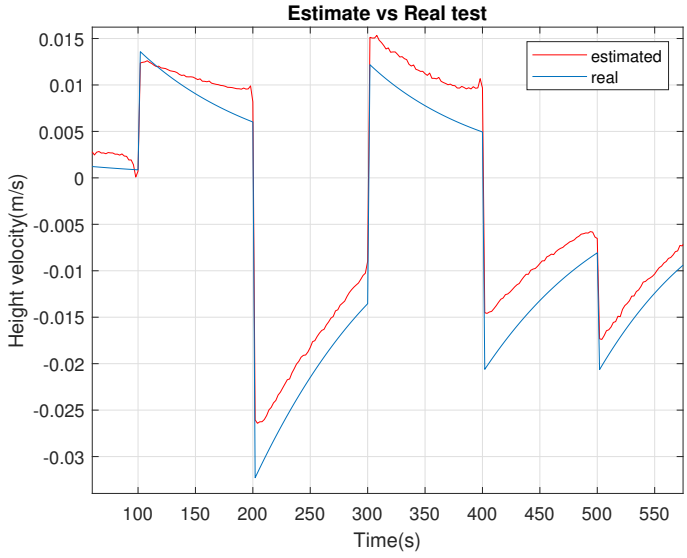
For comparison a feedforward network was implemented in a similar way as ESN. It included a random reservoir W and ridge regression was used to train W_{out}. All the parameters used except for C=0.025 and the training and test data was the same as for ESN. The input was a vector with all the inputs from 200 seconds back in time and current input to the system. $U(n) = [u(n), u(n-1), ..., u(n-99)]$ note u is discrete variable despite normal parentheses. The feed forward network was updated by following equations:

$$x[n] = tanh(U[n] * W_{in}^{T} W) \tag{3.5}$$

$$y[n] = W_{out} x[n] \tag{3.6}$$



(a) complete timeline



(b) zoomed

Figure 3.15: Test set: results of feedforward

3.4.3 Improving Time and Simplicity

This section will investigate how to improve training and running time as well as the simplicity. To improve this, different learning algorithms were used as well a sparse input matrix and decreasing reservoir size. Running time is the time ESN uses to predict 2500 points. Training time is the time to train the network. To measure the time the matlab functions tic and toc were used [10]. This was done using a standard laptop computer. LASSO have a high training time as it uses matlab's built-in LASSO function tests it for different λ parameter while for Ridge it was solved using the analytic solution from section 2.6 with λ set manually. It is therfore not fair to compare them.

	Training time(s)	Running time(s)
Ridge,sparsity(W_i)=0%, $N_x = 1000$	0,0999	1,3478
Ridge, sparsity(W_i)=95%, $N_x = 1000$	0,1044	1,3915
OLS, sparsity(W_i)=95%, $N_x = 1000$	0.0982	1.3924
LASSO,sparsity(W_i)=95%, $N_x = 1000$	107,3330	1.2847
LASSO,sparsity(W_i)=90%, $N_x = 500$	56.4692	0.2792

The differences between LASSO and Ridge is shown with histograms of the W_out matrices. The histograms are based on $N_x = 1000$ and no sparsity.

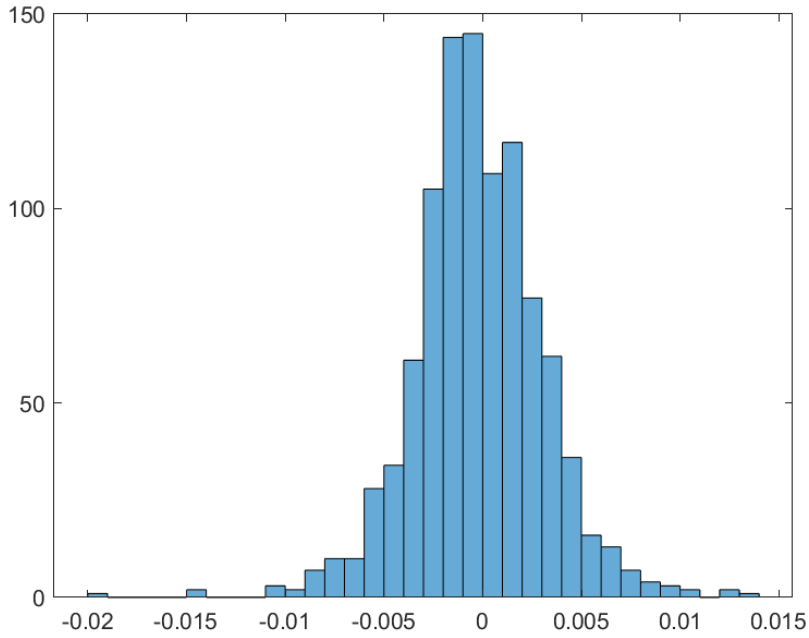


Figure 3.16: Histogram RIDGE W_{out}

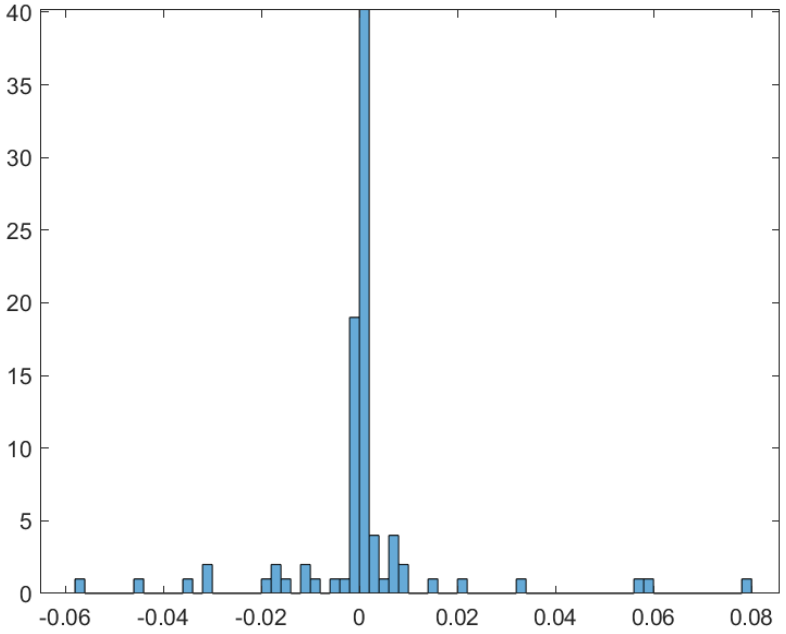


Figure 3.17: Histogram LASSO W_{out}. 62 of 1000 weights are non-zero

Lastly the performance of the different learning algorithms are tested. This time the test was run with a size 500 and a sparsity of 90% on W_{out}. The results are shown below in fig.3.18 and 3.19.

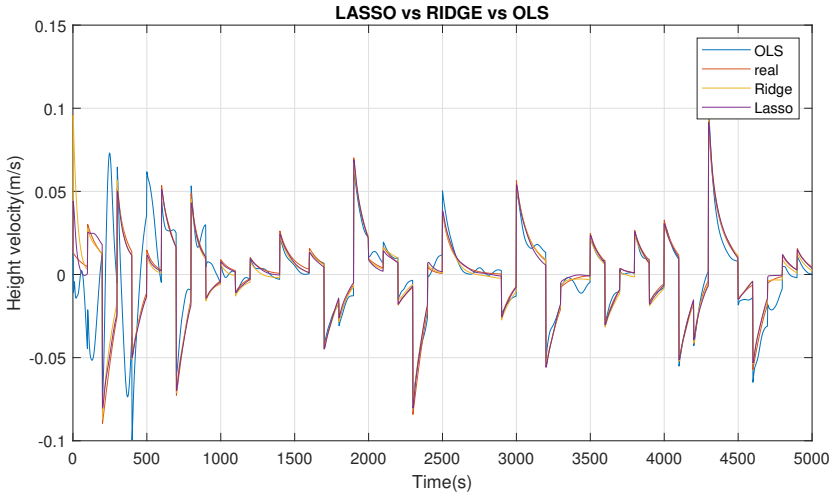


Figure 3.18: Test set: results of estimation with different regression methods.

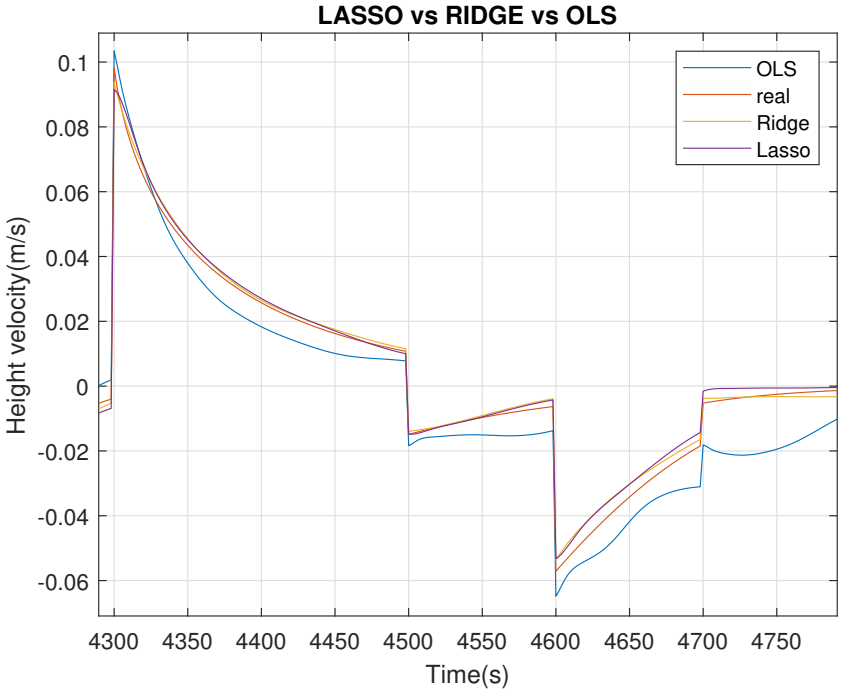


Figure 3.19: fig. 3.18 zoomed

Chapter 4

Discussion

4.1 Results

The result from fig. 3.12 shows that ESN to a high level of success can be used to model the nonlinear tank model. The ESN handles the introduction of noise fig. 3.14 very well. It can therefore be said that it filters the measurement noise away. The time to converge seconds for ESN to converge is dependent of the time. The typical time to converge was about 400s like in fig. 3.7. The fact that there only is marginally differences in performance between restult from test and training data suggest that it is robust against overfitting. One of the reason for this is that the training set was selected sufficiently large.

The trained network is in the margin area of having the echo state property. It barely satisfied necessary condition in section 3.5 such that small changes in the random W weights could be enough to not have the echo state property. A thing to notice is that the sum of error seem to be optimal in the boundary area.

When comparing ESN to feed forward the feature of ESN to have memory included in the states felt like a much more elegant approach than having 200 inputs. When

comparing **??** and 3.14 the results was better for ESN since feed forward was fast oscillating and in general further away from target. It is however worth to mention that significantly more time was spent on improving ESN than feed forward. But then again ESN has more parameters.

4.2 Errors and Model limitations

When describing implementing the tank dynamics 3.2 several assumptions were made. These assumptions are simplifications of the real world, but are likely not significant. In 3.2 there is no delay between a change in input and change in the inflow. In real world this can be a valve opening which is not happening instantly. This could actually make the modelling easier since the dynamics becomes slower. The measurement noise in 3.2 was implemented as white noise. White noise is not found in the real world and is therefore an error. The impact real world noise would have is dependent on the sensors. If there also was an constant error in the measurements which happens in the real world, ESN would fit the model to these bad data. This is one drawback with neural network, it does not have any knowledge to determine if the data make sense. Therefore the user needs to know this.

4.3 Different Learning Algorithms and Sparsity

The general discussion of choosing between Ridge and LASSO is whether or not all states are useful since LASSO can set the weights to those who are not to zero. Ridge is considered a good method if when countering overfitting since the weights are small. Notice the difference in range between 3.16 and 3.17. Ridge is therefore the most common approach to train ESN. LASSO can be useful if the reservoir size N_x is large related to the complexity and sufficient training data available. An interesting observation is that the whith a decrease in N_x from 1000 to 500 it gave an increase in non-zero elements in W_{out}. While increasing zero elements in W_{in} increases non-zero elements in W_{out}. To use LASSO

provides a simpler model that slightly decreases running time with same performance as Ridge 3.18 and 3.19. OLS did not perform well and was not much investigated as it is a special case of ridge. A sparse input matrix had only a negligible impact on performance, training and running time. Reducing the size of N_x had a large impact on the running time, but it also impacts the performance.

Chapter 5

Conclusion

5.1 ESN for system identification

The problem discussed in this project gave the expected results that ESN can successfully be used in modeling nonlinear problems. This support the previous work done in this field [1] [7].

One of the difficult parts of this project was to find good values for the hyperparameters. There are several values to consider as well as satisfying the echo state property. For this example the best performance was found in the boarder area of the echo state property. This result supports known theory on how to select parameters [9]. From section 3.4.3 Improving Time and Simplicity, the result shows that for task with strong time requirements, ESN is suitable for modeling. Ridge and LASSO is to prefer of OLS method of regression as they provided better results. To choose between LASSO and ridge is dependent on the problem. LASSO provides a simpler model but is more vulnerable to overfitting than ridge.

5.2 Future Work

Recommended future work is to investigate to what extent the structure of ESNs make them suitable for dynamic optimization. A useful application could be to use ESN as a predictor in Model Predictive Control (MPC). One advantage would then be that a model could be the target of online learning. Meaning the model is re-trained as new measurements are acquired. ESN as could also open up to new problems suitable for MPC since it is data driven modeling.

To test ESN on more complex systems to explore the limitations of where ESN is useful. For some higher dimension problems like language and speech processing, probably the best method is RNN with deep learning on all the hidden weights and not ESN. This is due to the size of the reservoir needed in ESN to solve higher dimension problem would be much larger[7].

There are also several parts of ESN that could be further explored in this example. The most interesting aspect not discussed in this report is feedback from the output back to the reservoir. It was tested once but the performance decreased and was not explored any further. Including feedback provided a better result for a more complex system: downhole pressure estimation in gas-lift oil wells [1]. Other thing that could be further explored is limitations of available training data. It could therefor be interesting to further investigate what type of problems it is useful for.

Bibliography

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

[2] W. Commons. Sigmoid functions, 2010.

[3] A. Hadgu. An application of ridge regression analysis in the study of syphilis data. *Statistics in Medicine*, 3(3):293–299, 1984.

[4] J. Han and C. Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In J. Mira and F. Sandoval, editors, *From Natural to Artificial Neural Computation*, pages 195–201, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[5] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 06(02):107–116, 1998.

[6] H. 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, 01 2001.

[7] H. Jaeger. Echo state network. *Scholarpedia*, 2(9):2330, 2007. revision #186395.

[8] H. Jaeger, M. Lukoeviius, D. Popovici, and U. Siewert. Optimization and applications of echo state networks with leaky- integrator neurons. *Neural Networks*, 20(3):335 – 352, 2007. Echo State Networks and Liquid State Machines.

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

[10] Mathwaorks. Start stopwatch timer.

[11] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[12] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58:267288, 01 1996.

Appendix

Matlab Code

```
close all;
clc;
rng('default'); % for repeatability of random numbers

%% Build ESN
n_r = 1000;  % reservoir size
n_i = 1;     % input size 1 for input flow and 2 for input flow and ...
    hight
n_o = 1;     %output size
% range of uniformly distrubution of weights
good_ranges = [0.055, 0.085, 0.025];
range = good_ranges(1); % n_r -> index: 1000 -> 1, 500 -> 2, ...
    1000(FF) -> 3
fb = 0; % 1 for including feedback, 0 for exclude
zero_elements = 0; % percentage og zero elements in W_in
[W_r,W_i,W_o,W_f] = ...
    Initilize_weights(n_r,n_i,n_o,range,fb,zero_elements);

alpha = .15; % leaking rate
lambda = .8; % regularization coefficient for analythic solution
reg_choise = 1; % choose regularization method 1 for LASSO and ...
    10^-5 for Ridge

```

```
20
21 % analasys of Echo state property
22 spectral_radius = max(abs(eig(W_r)))
23 max_singular_value = max(svds(W_r));
24 ESP_test = 1-alpha*(1-max_singular_value) % Sufficient condidtion ...
       for < 1
25 W_tilde = alpha*W_r+(1-alpha)*eye(n_r);
26 effective_spectral_radius = max(abs(eig(W_tilde))) % necessary ...
       condotion  for < 1
27
28 %% Generate training and test data
29 noise=0.000; % 0.00005 to add noise to meassurements h
30 time_length = 10000; %simulation time
31 n_inputs = 100; % how many different inputs
32 sample_time_u = time_length/n_inputs; %hold time for input
33 sample_time_sys = 2; %sample stime for system
34 training_samples = time_length/sample_time_sys;
35 h_0 = 4; %initial tank hight
36 rho = 1; % density fluid
37 g = 9.81; %gravity constant
38 r1=5; %radius of tank
39 r2=0.5;   %radius of output pipe
40 q_low = 2; %lowest inflow
41 q_max =10; %highest inflow
42 input.time=sample_time_u*(0:n_inputs-1)';
43 input.signals.values =randi([q_low, q_max],n_inputs,1);
44
45 %simulate
46 options = simset('SrcWorkspace','current');
47 sim('generate_trainingset',[],options);
48 Y_target=h_dot_noise;
49 h_dot_real = h_dot;
50 h_training=h;
51 U_training=u;
52
53 %generate test data
54 noise=0;
55 time_length=5000; %simulation time
```

```
n_inputs = 50; % how many different inputs
sample_time_u = time_length/n_inputs;
test_samples = time_length/sample_time_sys;
input.time=sample_time_u*(0:n_inputs-1)';
input.signals.values =randi([q_low, q_max],n_inputs,1);

%simulate
options = simset('SrcWorkspace','current');
sim('generate_trainingset',[],options);
h_dot_test=h;
U_test = u;

%% harveset states X
x = zeros(n_r,1); %initial contidion for states
y = Y_target(1); %initial contidion for estimate
X=zeros(n_r+n_i,training_samples+1);
for i = 2:training_samples+1
    x = update_x_ESN(x,y,U_training(i,1:n_i),W_r,W_i,W_f,alpha);
    %x = update_x_FF(u(i,1:n_i),W_r,W_i)';
    x_o=[x;U_training(i,1:n_i)']; % connect input directly to output
    %y = update_y(x_o,W_o);
    X(:,i)=x_o;
end

%% Train ESN
tstart = tic; %start time of training
W_o=((Y_target' * X')/( X * X' + lambda * eye(n_r+n_i)))'; % RIDGE ...
    analytic solution
output_constant = 0; % no constant part
% [BlassoAll,FitInfo] = lasso(X',Y_target', 'Alpha', reg_choise, ...
    'CV', 10);% training using built in function
% Blasso=[FitInfo.Intercept(FitInfo.Index1SE); ...
    BlassoAll(:,FitInfo.Index1SE)];
% output_constant = Blasso(1); % constant part
% W_o=Blasso(2:n_r+n_i+1); %
t_training_time = toc(tstart); % time to train ESN

%% test on training data
```

```
x = zeros(n_r,1); %initial contidion for states
h_dot_est_training = zeros(training_samples+1,1);
y = h_dot_real(1);
h_dot_est_training(1)=y;

for i = 2:training_samples+1 % start on two
    x = update_x_ESN(x,y,U_training(i,1:n_i),W_r,W_i,W_f,alpha);
    %x = update_x_FF(u(i,1:n_i),W_r,W_i)';
    x_o=[x;U_training(i,1:n_i)']; % connect input directly to output
    y = update_y(x_o,W_o)+output_constant;
    h_dot_est_training(i)=y;
end

%% test ESN

x_0_zero_states = zeros(n_r,test_samples+1);
x = zeros(n_r,1); %initial contidion for states
x_0_zero_states(:,1)= x;
h_dot_est_test = zeros(test_samples+1,1);
y = h_dot(1); %initial contidion for y
h_dot_est_test(1)=y;
tstart5= tic; % timw to train ESN %start time of training
for i = 2:test_samples+1
    x = update_x_ESN(x,y,U_test(i,1:n_i),W_r,W_i,W_f,alpha);
    %x = update_x_FF(u(i,1:n_i),W_r,W_i)';
    x_o=[x;U_test(i,1:n_i)']; % connect input directly to output
    y = update_y(x_o,W_o)+output_constant;
    h_dot_est_test(i)=y;
    x_0_zero_states(:,i) = x;
end
t_running_time = toc(tstart5); % timw to run ESN
%% print figueres
t=sample_time_sys*(0:test_samples)';
figure(1)
plot(t,h_dot_est_test(:,1))
hold on
plot(t,h_dot)
title('Test h_dot')
```

```
xlabel('Time(s)')
ylabel('Height velocity(m/s)')
legend('Estimated', 'real')
grid on;

figure(4)
title('h test')
xlabel('Time(s)')
subplot(2,1,1)
plot(t,h_dot_test(:,1),'r')
title('h test')
xlabel('Time(s)')
ylabel('Height ')
grid on;
subplot(2,1,2)
plot(t,u(:,1),'g')
ylabel('q_{in} ')
grid on;

t=sample_time_sys*(0:training_samples)';
figure(2)
plot(t,h_dot_est_training(:,1),'r')
hold on
plot(t,Y_target)
hold on
%plot(t,h_dot_real,'g')
xlabel('Time(s)')
ylabel('Height velocity(m/s)');
title('Estimate vs Real on training')
legend('estimated', 'meassured', 'real');
grid on;

figure(3)
subplot(2,1,1)
plot(t,h_training(:,1),'r')
ylabel('Height ');
xlabel('Time(s)')
title('h training')
```

```
grid on;
subplot(2,1,2)
plot(t,U_training(:,1),'g')
xlabel('Time(s)')
ylabel('q_{in} ')
grid on;
```

```
function x = update_x_ESN(x_prev,y_prev,input,W_r,W_i,W_f,alpha)
x = (1-alpha) * x_prev + alpha * tanh((W_r * x_prev + W_i * input' ...
    + W_f * y_prev));
end
```

```
function x = update_x_FF(input,W_r,W_i)
x = tanh((input*W_i'*W_r));
end
```

```
function y = update_y(x_o,W_o)
y =  W_o' * x_o;
end
```

```
function [W_r,W_i,W_o,W_f] = ...
    Initilize_weights(n_r,n_i,n_o,a,fb,zero_elements)
W_r = -a+(2*a).*rand(n_r); %reservoir wieghts
W_i = -a+(2*a).*rand(n_r,n_i); %input weights
W_o = -a+(2*a).*rand(n_r+n_i,n_o); % output weights
W_f = 0.*rand(n_r,n_o); % feedback weights
if fb == 1
    W_f = -a+(2*a).*rand(n_r,n_o); % feedback weights if fb included
end
W_i(rand(n_r,1)<zero_elements) = 0; % set  percentage to zero
end
```