Summary

Virtual Flow Metering (VFM) is a method for estimating flowrates for different phases of a multi-phase flow without measuring them directly. Instead it makes use of data related to the flow. VFM is a topic of research in the oil and gas industry, where it is difficult to measure and model the three-phase flow from a well consisting of oil, water and gas. A common approach in the industry is to obtain multi-phase measurements is to allocate the wellstream into a test tank. Information about the composition of the well stream could potentially be used to better plan the production, improve redundancy and safety and reduce interruptions in the production. This work uses ESN to estimate a multi-phase wellstream entering a three-phase gravity separation tank. The input to the ESN includes measurements of water level, liquid level and pressure, as well as the data on the control variables outflow from the water, oil and gas phase of the tank. For the experiments, simulations were used to represent the tank. In this work, ESN was shown able to recreate wellstream with both a stationery and varying water level in the tank and also rejecting noise on states. In this work ESN showed better results than an observer based on Extended Kalman Filter.

Echo State Networks (ESN) are a type of Recurrent Neural Networks (RNN) and can be used to model certain classes of nonlinear dynamical systems. The ESN contains a large recurrent neural network with fixed weights that are defined at random, which are called the reservoir. The intuition behind ESN is that under the influence of input signals the reservoir is a high-dimensional collection of nonlinearly state signals from which a desired output signal can be combined. RNNs tends to be costly to optimize while ESNs have a low computational cost. This is because only the output layer is trained, and popular methods, like least square, are computationally effective.

Preface

This work was done in a bilateral agreement between the two universities NTNU and UFSC (Federal University of Santa Catarina). I therefore had the pleasure of doing my work at the UFSC. The software made available to me from NTNU included Matlab and Simulink.

First, I would like to thank Prof. Eduardo Camponogara for all the advising and support, enabling the development of this work, also for all the hints in the writing and for believing in me.

I would also like to thank Dr. Christoph Backi for sharing his work with three-phase gravity separator model and extended Kalman filter observer.

Further, I would also like to thank Eng. Marcos Vinicius de Carvalho Gomes, for inviting me to CENPES/Petrobras to present my work. The staff of the logistic and optimization department at CENPES Petrobras gave useful feedback on my research.

I would like to thank all the people at my research group, for all the support, fun times, and help regarding the development of my work. Their company was valuable and they gave me the inspiration and insight necessary for the development of this work.

Lastly I would like to thank all the friends I have made in Brazil. Brazilians are the most welcoming people I have ever met. It has truly been a wonderful time here and provided me with unforgettable moments.

Table of Contents

List of Tables

List of Figures

Abbreviations

ESN	=	Echo State Network
VFM	=	Virtual Flow Metering
OLS	=	Ordinery Least Square
RSS	=	Residual Sum of Squares
ANN	=	Artificial Neural Network
RNN	=	Recurrent Neural Network

Chapter 1

Introduction

1.1 Motivation

Today there are many real-world problems that are difficult to obtain a physics based model. In oil and gas production, the multiphase wellstream is often considered an unknown variable, because it is difficult to both model and measure directly. Multiphase flowrates play an important role in production optimization, rate allocation and reservoir management. Virtual Flow Metering (VFM) is a term used to describe the estimation of multiphase flowrates. In oil production, a wellstream may enter a three-phase gravity separation tank for a first rough separation. In the three-phase separator there are other measurements available like pressure, water level and liquid level, as well as outflow from the different phases. Could ESN then be used to solve the inverse problem, going from pressure and level measurements from a separation tank to find the multiphase inflow rate?

In the broadest picture Echo State Network (ESN) is a Recurrent Neural Network (RNN). All neural networks are built up with artificial neurons with connecting weights, inspired by the way our biological brain works. RNNs are recognized by the fact that they do not only propagate forwards but also backwards. ESNs have a large re-

current neural network with fixed weights that are defined at random, which are called the reservoir. The intuition behind ESN is that under the influence of input signals the reservoir is a high-dimensional collection of nonlinearly state signals from which a desired output signal can be combined [12]. A large random reservoir is a powerful tool as long as it is ”big enough”, because then it is likely that there exist a linear combination within the reservoir that can recreate the signal. Linear regression is used to train the output layer from reservoir states to the output signal.

Linear regression is under the category supervised learning, as it uses available input and output data to create a model for the input-output relation. Linear regression is a learning method which has a low computational cost compared to other methods used in RNNs. Previous work with ESN includes data-driven downhole pressure estimation [2] and stock price prediction [17].

1.2 Objective

The objective of this work is to demonstrate the effectiveness of ESN in the identification of complex water-oil-gas separation units. The goal is then to use ESN to estimate the load entering a three-phase separation tank based on measurements and outflow rates from the tank. The separation tank was simulated using a model as described in [4]. To demonstrate the use of ESN for different ”levels” of complexity, the simulation considered three distinct cases that define three test sets. For the first test the controller had a constant desired value for all the states. The second had a random time-varying signal for the desired water level to the controller, which increased the complexity of the process. The third test introduced noise on the measurements.

1.3 Document structure

The document contains in total six chapters.

- Chapter 2 reviews the basic knowledge about dynamical systems, system identification, and virtual flow metering.
- Chapter 3 gives an introduction to Echo State Networks. It describes its mathematical properties and the tuning process.
- Chapter 4 will introduce a three-phase gravity separation process, the model which was selected for the simulated experiments in this work. The chapter also lay out a description of the use of Echo State Network as Virtual Flow Metering.
- Chapter 5 provides the implementation details, the experiments and simulation results included in this work.
- Chapter 6 is a conclusion of this work.

Chapter 2

Fundamentals on Systems and Controls

This chapter presents a brief overview of dynamical systems, system identification, and virtual flow metering. This chapter aims to give a short description of what to consider when deciding what method to use for system identification. The choice of a model should consider linear/nonlinear model, interpretable model, exists prior knowledge, fast computation and complexity.

2.1 Dynamical Systems

A system is thought of as a way to explain the internal behavior from an input to an output. A dynamical system is then further characterized by the output being a result of all previous inputs and the initial condition. In contrast nondynamical system are only dependent on current input.

State-space representation is the common representation of dynamical system

in control theory. It is often used for modeling and control, this representation is easy to expand for higher dimension systems with multiple states, inputs and outputs. It makes dynamical systems easier to analyse and to compute. This is because the state-space representation introduces states denoted by $\mathbf{x}(t)$ which are dependent on itself and the inputs. The dependence of the states on itself provides some form of memory, since the current state is a result of all previous inputs. However it is more efficient to compute using the states rather than all previous inputs. States are expressed with a Ordinary Differential Equation (ODE) as in Eq. (2.1) with $\mathbf{x}(t)$ being the state, $\mathbf{y}(t)$ being the output which normally is referred to as measurable states such as temperature or pressure and $u(t)$ being the input to the system. The input $u(t)$ is the control variable, examples of this can be valves. The equations Eq.(2.1) then gives the input to output relation.

$$\dot{\mathbf{x}}(t) = \mathbf{f}(x(t), u(t)), \quad x \in \mathbb{R}^n, \quad u \in \mathbb{R}^k \tag{2.1}$$

$$\mathbf{y}(t) = \mathbf{g}(x(t), u(t)) \quad y \in \mathbb{R}^m \tag{2.2}$$

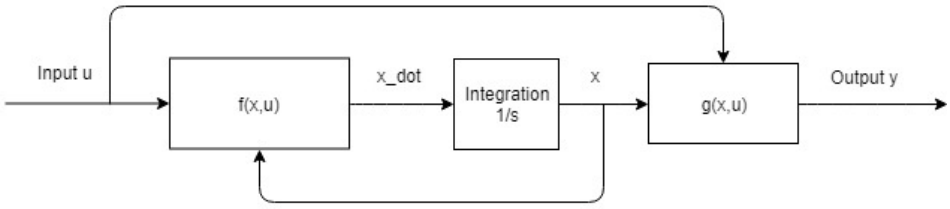


Figure 2.1: Graphical representation of Eq. (2.1)

Causality is the property that the output is not dependent of future inputs only the present and the past. Noncausal systems then are systems dependent on future inputs. Noncausal system is rare as all physical dynamical systems are considered causal.

A system is linear if it satisfies the superposition principle: $\alpha f(x) + \beta f(y) =$

$f(\alpha x + \beta y)$. This allows linear systems to be expressed in terms of Eq. (2.1)

$$\dot{x}(t) = A(t)x(t) + B(t)u(t), \quad x \in \mathbb{R}^n, \quad u \in \mathbb{R}^k, \quad A(t) \in \mathbb{R}^{n\times n}, \quad B(t) \in \mathbb{R}^{n\times k} \tag{2.3}$$

$$y(t) = C(t)x(t) + D(t)u(t), \quad y \in \mathbb{R}^m, \quad C(t) \in \mathbb{R}^{m\times n}, \quad D(t) \in \mathbb{R}^{m\times k} \tag{2.4}$$

With $\mathbf{A}(\mathbf{t})$, $\mathbf{B}(\mathbf{t})$, $\mathbf{C}(\mathbf{t})$ and $\mathbf{D}(\mathbf{t})$ as parameters. The real world is almost always nonlinear but sometimes the nonlinearities are weak and a linear system can provide a good local approximation. When one recognizes nonlinear phenomena or otherwise have hard nonlinearities. It is not enough with a linear model. For a linear model one can find the analytic solution in time domain by looking at the impulse response. This is often not possible for a nonlinear system. A linear system is also easier to investigate properties as stability, observability, and controlability. To check if a linear timeinvariant system is stable one only needs to check if the eigenvalues of matrix $\mathbb{A}$ from Eq. (2.3) have negatice real parts.

A system is time-invariant if the equations will not change with time. A time-variant system does change with time. For time-variant systems there is a small modification of equation (2.1) such that $\dot{\mathbf{x}}(t) = \mathbf{f}(x(t), u(t), t)$. The function f is dependent on t. Almost all real world systems are varying with time but can in many situations be neglected.

A discrete system will introduce time step δ, which dictates how often to update the states and output. In contrast to a continuous time system which updates instantaneously. The output will then be computed as an Euler discretization on the time interval $[t, t + \delta]$. The real world is continuous and computers are discrete. A small δ gives good approximation. A linear time invariant discrete system is characterized by the equations:

$$x[t + 1] = Ax[t] + Bu[t] \tag{2.5}$$

2.2 System Identification

System identification is the process of obtaining a mathematical model for a system. Figure 2.2 is meant to illustrate that it is about finding the input to output relation. For a dynamical system identification is about finding the function $f(x, u)$ and $g(x, u)$ from Eq. (2.1). This knowledge can then be used to predict future states which is useful for control. There exists many algorithms and frameworks for system identification today. Things to consider when choosing method can depend on, the complexity of the system, if it linear/nonlinear, and if there are some prior knowledge of the system available. One common way to classify models are white, black and grey box.

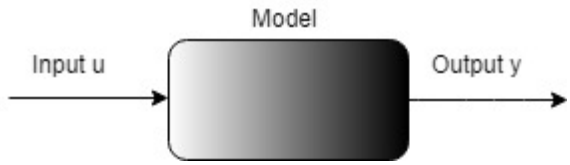


Figure 2.2: A system as seen from the outside

- White Box modeling. It is a physics based method with a equations of motion derived from a first principle of physics like for example Newton's law. Models classified as white box models are fully interpretable models. It can be a mechanistic model that only requires tweaking of parameters.
- Black Box models requires no prior knowledge of the system since it is a model solely driven by data. It can be an Artificial Neural Network (ANN). Such models are today inherently difficult to interpret.
- Grey Box models uses some prior knowledge to partly model the system. It is considered a hybrid method between white and black box modeling. It is both theory and data driven. It can be a combination of a mechanistic model and principal component analysis regression.

The concept of interpretability used here was defined by [15] as: "Interpretability is the degree to which a human can consistently predict the models result". This def-

inition does not give a clear way to measure interpretability but still makes it possible to classify model's results. Some benefits of interpretability are given by [22]:

- Reliability or Robustness: Ensures that small changes in the input do not lead to large changes in the prediction.
- Causality: Check that only causal relationships are picked up.
- Trust: It is easier for humans to trust a system that explains its decisions compared to a black box.

The knowledge of how the system works is useful for many problems. Theory driven model normally does not cover all behaviour of a real world system. However it is often possible to measure the input and output behavior which in turn can be used to create a model.

After creating a model it should be verified. Ideally this is done on a completely different set of data. This makes it possible to detect a problem like overfitting which occurs when the model fits the first data too well instead of generalizing.

2.2.1 Linear Regression

$$y = \beta_1 u_1 + \beta_2 u_2 + \cdots + \beta_n u_n \tag{2.6}$$

Linear regression modeling creates the input-output relation from Figure 2.2 as a sum of its features u and according weights β_i, Eq. (2.6). Features u are the inputs used for prediction, can consist of control variables as well as measurements. What is common for all regression methods is that an input-ouput relation is found by solving an optimization problem minimizing the error between estimated and the data value. The set of weights β_i are the parameters to be found.

Now follows an example of deriving a linear dynamic model using linear re-

gression for a linear time-invariant discrete full state observable system as Eq. (2.5). With the output for the system now being $y = x$ (fully state observable) and u being a control variable affecting the system. There exist meassurments for $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^k$ over a timeseries for $[0...T]$. The matrices $\mathbb{A}$ and $\mathbb{B}$ are then the only unknown parts. However a solution with the data directly $\mathbf{X} = \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} \mathbf{X_{-1}} \\ U \end{bmatrix}$ is not feasible unless the data is perfectly the solution using the data directly which is never the case with real world data. This means that it becomes an optimization problem to minimize the error between estimated $\tilde{\mathbf{X}}$ and data $\mathbf{X}$. $\beta = \begin{bmatrix} A & B \end{bmatrix}$ is the variable.

$$
\begin{aligned}
\mathbf{X} &= \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1T} \\ x_{21} & x_{22} & \cdots & x_{2T} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nT} \end{bmatrix}, \mathbf{X_{-1}} = \begin{bmatrix} x_{10} & x_{11} & \cdots & x_{1T-1} \\ x_{20} & x_{21} & \cdots & x_{2T-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n0} & x_{n1} & \cdots & x_{nT-1} \end{bmatrix}, \\
\mathbf{U} &= \begin{bmatrix} u_{10} & u_{11} & \cdots & u_{1T-1} \\ u_{20} & u_{21} & \cdots & u_{2T-1} \\ \vdots & \vdots & \ddots & \vdots \\ u_{k0} & u_{k1} & \cdots & u_{kT-1} \end{bmatrix} \\
\tilde{\mathbf{X}} &= \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} \mathbf{X_{-1}} \\ U \end{bmatrix}
\end{aligned}
\tag{2.7}
$$

Linear models are naturally much simpler to obtain than nonlinear models. A linear model as in Eq. (2.6) is assumed to be interperable as well, as it is easy to investigate the impact of the different features. Some methods for linear regression have been around for a very long time, such as Ordinary Least Squares Linear Regression (OLS) which was developed by Carl Friedrich Gauss in 1795, [7]. Linear regression is then guaranteed to find optimal weights, given all assumptions of the linear regression model are met by the data. These assumptions are

- Linearity, as described by Eq. (2.3). This is for many systems not possible as they have hard nonlinearities.
- Normality, it is assumed that the target outcome given the features follows a normal distribution.
- Homoscedasticity, the variance of the error terms is assumed to be constant over the entire feature space. This means that an error term as $\pm 10 unit$ is valid while $\pm 10\%$ of the target value would violate this assumption.
- Independence, it is assumed that each instance is independent of any other instance.
- Fixed features, input features are considered fixed. Fixed means that they are treated as given constants and not as statistical variables.
- Absence of multicollinearity. You do not want strongly correlated features, because this messes up the estimation of the weights.

Some treatment of data before training might be advantageous in some cases. This includes outliers detection and removal. Principal Component Analysis (PCA) which is a form of feature selection can handle irrelevant and correlated features better.

A linear model will in many cases be underfitted, restricted and oversimplified. However some tricks such as Kalman filter combines measurement as well as a model. Meaning that estimation does not fully trust the model. This can make an underfitted and oversimplified model acceptable with good measurements.

Another advantage of using a linear model is that it is much faster. Solving an nonlinear optimization problem is greatly more difficult than a linear one. This is because nonlinear problems are nonconvex. The difficulties lies in that there exist many suboptimal solutions. A solver is needed to solve a Nonlinear Programming Problem (NLP). This greatly increases runtime and is less robust as there is no guarantee that a solution is found. [5].

Linear regression can be used to find weights for a neural network. This means

that nonlinearities and high dimension workspace is introduced which makes it possible to model nonlinear behavior. This the method will be used to train ESN and is described in section 3.4.

2.2.2 Artificial Neural Networks

For systems with hard nonlinearities, high dimension, and exposed to high noise, one increasingly popular tool is Artificial Neural Networks (ANN). Artificial neural networks are inspired by the way our biological neural network in our brain works. Neural network is a framework containing many different algorithms in machine learning. For this work the focus is on the use of supervised learning for system identification. Supervised learning means that it is driven by input and output data, as illustrades in Figure 2.2 and given by Eq. (2.1). A common basic architecture for neural networks is seen in Figure. 2.3. It consists of many artificial neurons or computational units that receives input and then to apply an activation function to produce the output. The activation function adds nonlinearity to the model.

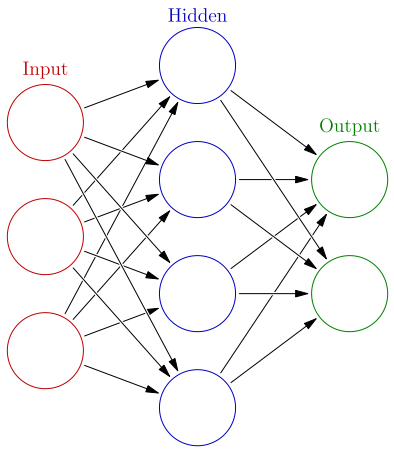


Figure 2.3: A basic common structure for ANN.

In a simple model, the first layer is the input layer, followed by one hidden layer, and lastly by an output layer. Each layer can contain one or more neurons. By adding

more neurons or more hidden layers of neurons the model becomes increasingly complex which allows to identify more systems. This may classify as Deep Learning which are used describe certain types of neural networks and related algorithms. They often consume raw input data and process this data through many layers of nonlinear transformations of the input data. This is done in order to calculate a target output [6]. Alongside more powerful computer to handle deep neural network, more data is also being gathered nowadays. This increased the number of problems suitable for system identification using ANN.

There exist many subcategories under ANN. Two notably is Feedforward and Recurrent Neural Network. The main difference between Feedforward and RNN is that RNN contains memory. A traditional RNN is recursively dependent on itself. Feedforward has the architecture shown in Figure. 2.3. For having memory RNNs are suitable for modeling dynamical systems which also contain a memory.

For ANN as well as linear regression an optimization problem should be solved to compute the network weights. The generic approach to minimizing the error between estimated output and data output is by gradient descent, called back-propagation. This is an iterative algorithm that looks at the derivative of error and moves the weight in the direction that minimizes the error. This step will then typically be repeated until a optimum is achieved. There exist some disadvantages with backpropagation such as the problem of a vanish gradient as well as being computationally expensive. The problem of a vanish gradient is a result from the gradient being small making it difficult to update the weights. This might result in a suboptimal solution.

As the model is created on data overfitting can become an issue. Since ANNs are un-interperetable, un-excpected behaviour may occur when subjected to inputs from outside of the validated area. This is a drawback with black box modeling.

2.3 Virtual Flow Meetering

Virtual Flow Metering (VFM) is a method for estimating flowrates for different phases of a flow without measuring them directly. Instead take use of data related to the flow. One such example can be the three phase flow from a well consisting of oil, water and gas. The data input can be from sensors installed at various measurement points (nodes) in the wellbore, on the seabed and in surface facilities. It is a system identification problem. There are two main different types of VFM: hydrodynamical and data driven.

Hydrodynamical is normally considered a white box modeling. It takes advantage of existing knowledge and uses existing mathematical models and data to estimate the parameters. It is also possible to use different models for different stages of an oilfield. One such model is implemented by [4]. It is derived from first principles such as Stokes law. Parameters that might requires tuning are the droplet distribution of water and oil, as well as how the three phases are initially mixed. It solves an inverse problem for a three phase gravity separator by using data about the tank and outflows from different phases of the tank.

Data-driven is a black box modeling approach. A Long Short-Term Memory (LSTM) as VFM showed promising results in [1]. LSTM is a subgroup of RNN networks. It was able to to estimate flow rates for oil and water at current time, as well as predicting for a sequence of future time instants. The data used to create the model here is a synthetic data set of pressure, temperature, and oil, gas and water rates from a well test.

The data used can either be from a test well or simulations with varying rates of oil, water and gas or from historic data. The advantage of a test well is that it is easy to change the rates. To have a variable set is important since it gives a larger space for model validation. The data used can also include information about the well, and/or information about the separation tank. The benefit of obtaining a model using data from the well is that it can predict at an earlier stage.

Chapter 3

Fundamentals on Echo State Networks

This section will describe the different parts of Echo State Networks (ESNs). Starting with an intuitive explanation before introducing the mathematics for ESN. To give a better understanding how the process of building up and tuning an ESN for system identification, this Section will present a case example.

3.1 Intuition

ESN is an artificial neural network that is used here as a black-box model approach. A set of previous input and output data is used to train an ESN to model a dynamic system. As of today it is inherently difficult to interpret a neural network as opposed to modeling using physics and therefore called a black-box.

ESN is a subgroup of Recurrent Neural Networks which means it has a memory. This gives an elegant solution to model history dependent systems. In practice history

dependency can be implemented with previous states being the input to the next state. This is described in more details with equations in Section 3.2. For an ESN it is not only the ability to keep memory which is important, but also the ability to forget with time. This is because of the behaviour of ESN is wrong in the beginning like in Figure 3.11. This ability to be independent of the initial state is called Echo State Property and is discussed in Section 3.3.8.

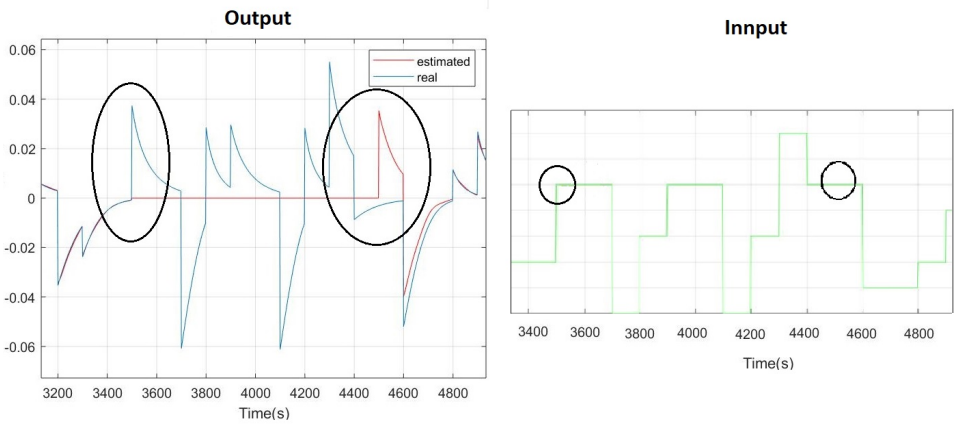


Figure 3.1: The Left figure is the simulation of the system from Section 3.3.2 (blue) and a estimation using ESN (red) with hold from time 3500 to 4500. The black circles marks interesting response for the beginning of the pause and restarting of the estimation. Right is the input to the system above.

The left part of Figure 3.1 is a result from a simulation where an ESN estimates (red line) a function with the right part of Figure 3.1 being the input to the ESN. The details of the system is described in Section 3.3.2. What to is special in Figure 3.1 is that the estimation is temporarily put on hold between time 3500 and 4500, meaning it does not update internal states for this timespan. The real value (blue line) continuous to update with the input. Notice that at time 4500 when the ESN starts to estimate again, it still contains the same memory as when it was put on hold at time 3500. In addition, the input for the system is the same as when it was put on hold and when it began estimating again. So the ESN then estimates the same path as the real value (blue line) did for the time when it was put on hold at time 3500. It is also worth to notice that it forgets the memory with time so the test in Figure 3.1 follows good in the end. This was meant to illustrate the

same behaviour we also find in an echo. It keeps repeating (”remembering” inputs) but slowly fades away. Other positive sides to ESN is that it allows to add multiple outputs from same internal states, by just adding one more dimension to output weights as well as training them separately.

This main structure of ESNs can be seen in Figure 3.2. The body of the ESN consists of a large reservoir of neurons connected with random weights. This reservoir of neurons is refereed to as internal states with matrix $\mathbf{X}$. The reservoir are connected internally with a given distribution.

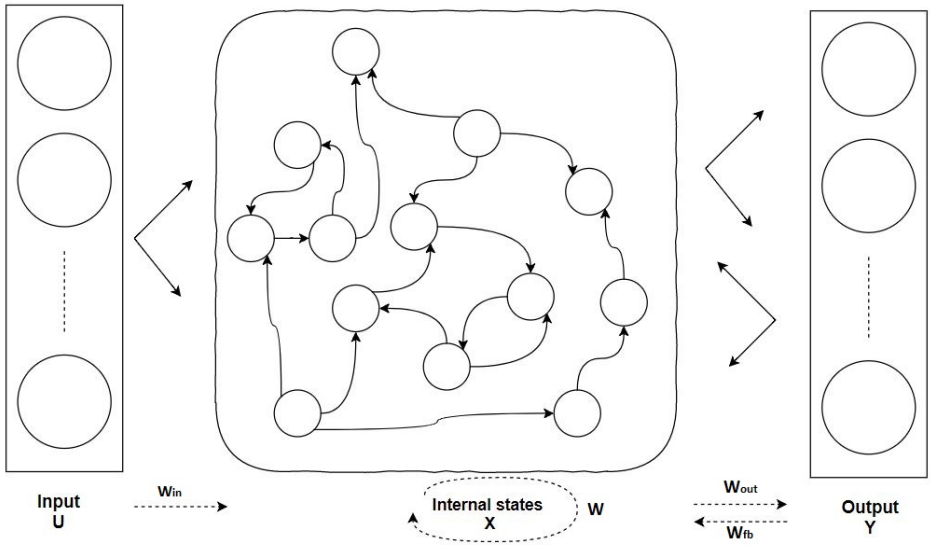


Figure 3.2: Structure of neurons and weights for ESN, Drawn freely with inspiration from [13].

The other main part is the output matrix. Which is the link from a reservoir of random weights to the desired output. These weights are represented by W_{out}, seen in Figure 3.2. This connection is achieved by training the weights using linear regression. Which is the learning method described in Section 3.4. The intuition behind this is that a linear combination of a large number of neurons can be used to represent any function, even if the sytem is nonlinear. Training is refereed to as solving an optimization problem with the goal of minimizing the error between the data and estimated output values by

varying W_{out}, the link between the reservoir and and output.

The training method is the most important difference between ESN and other RNNs. This is because linear regression is a very fast method and and only the output weights of an ESN are trained. This is in contrast to RNN's using backpropagation which is a slow training method.

3.2 Updating states in ESN

The main structure seen in Figure 3.2 is represented by Eq. (3.1) and Eq. (3.2). These equations describe the way internal states (neurons) X and output Y are updated. It is also part of what separates ESN from other neural network. They typically have the following equations as described by [18]:

$$x[k] = (1-\alpha)x[k-1] + \alpha f(W_{in}[k] + W\ x[k-1] + W_{fb}\ y[k-1]) \tag{3.1}$$

$$y[k] = W_{out}\ x[k] \tag{3.2}$$

The leak rate $\alpha \in (0, 1]$ is a hyper-parameter for ESNs. The fact that the new states of ESN are dependent on the prior states is provides some sort of memory of previous inputs. So with leak rate directly impacting the influence between $x[n-1]$ and $f(W_{in}\ u[n] + W\ x[n-1] + W_{fb}\ y[n-1])$ for the new state in Eq.(3.1). The leak rate will directly affecting the memory ability. 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 the leak rate means we have a leaky integrator ESN.

The function f is a sigmoid function. A sigmoid function is a function shaped like an "S" and typically has the boundary of $(-1, 1)$ as the input to f can be $(-\inf, \inf)$. There exist several examples that can be used here such as the ones shown here in Figure 3.3. The most common are: the logistic function, tanh, and arctan. For a neuron this is

called the activation function as it resembles the way different real neurons in the brain can be activated.

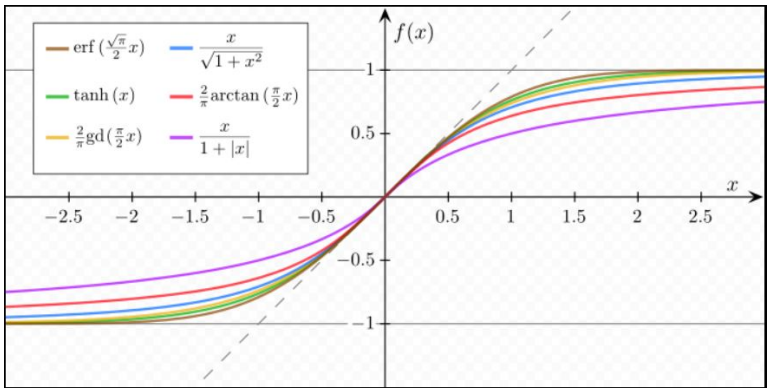


Figure 3.3: Examples of different sigmoid functions from [8].

W_{in}, W, W_{out} and W_{fb} are called the weight matrices of the Echo state network. W_{in} is the link from the input $\mathbf{u}$ to the internal states $\mathbf{x}$. W is the link from internal states on previous timestep $\mathbf{x_{k-1}}$ to internal states at the current timestep $\mathbf{x_k}$. W_{out} is the link from internal states $\mathbf{x}$ to the output $\mathbf{y}$. W_{fb} is the link from output on previous timestep $\mathbf{y_{k-1}}$ to internal states at the current timestep $\mathbf{x_k}$. The weights can be described as parameters that dictate the influence the different input values has on a function with multiple inputs. In practice this means the input value can be scaled up or down before entering the function.

There exist some different notations for Eq. (3.1) and Eq. (3.2). Like in [14] they describe a continuous version. Eq. (3.1) and Eq. (3.2) could then be derived by using Euler discretization on the continuous-time dynamics of a leaky-integrator. A discretization with a too big stepsize could then lead issues with stability. The discrete version for Eq. (3.1) and Eq. (3.2) in [14] then in addition to a leak rate α also included a global time constant and a stepsize as parameters. The tuning is simpler the less parameter one have, and as they infect the same variables as the leak rate they are not included for this work. The update for one internal state $x_{k,i}$ with k being timestep and i being state number, is graphically represented in Figure (3.4). The difference is that in Figure (3.4) is an illustration for a

single neuron while Eq. (3.1) are compact using vectors and matrices.

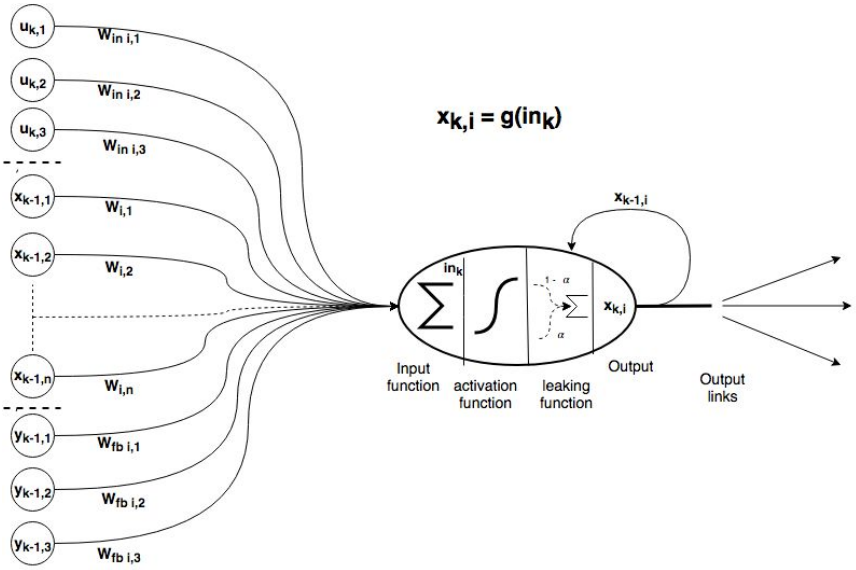


Figure 3.4: Updating internal state $x_{k,i}$ expressed as a Neuron for ESN. Drawn freely.

Include a linear part of input u directly to output. The benefit of this is that the linear relation between input u and output y is modeled directly and becomes more interperetable. This will in practice be the same as saying that the input is a state. This means a small change to the output matrix from Eq. (3.2) to:

$$y[i] = W_{out1}\, x[i] + W_{out2}\, u[i] = W_{out} \begin{bmatrix} x[i] \\ u[i] \end{bmatrix} \tag{3.3}$$

3.3 Tank Case Study

Figure 3.5 gives an overview of the main parts of implementing ESN to generate a model for a system. To illustrate the use of ESN for system identification a case study for a single input single output nonlinear tank system is provided. This system is described in Section 3.3.2.

3.3.1 Example of workflow with ESN

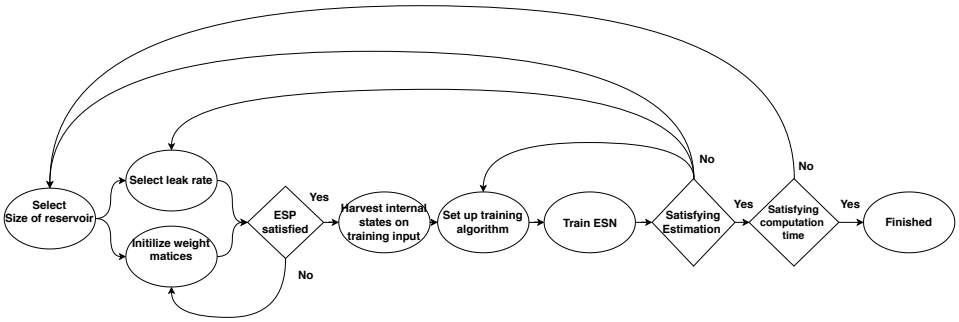


Figure 3.5: Work flow for implementing ESN

The first three bubbles in Figure 3.5 corresponds to the selection of hyperparameters and choices in general for the network. For hyperparameters such as selecting the size of the reservoir, refer to Section 3.3.3, for leak rate Section 3.3.6, and for distribution range of weight matrices refer to Section 3.3.7. Further choices to make when implementing an ESN are whether or not to include feedback, refer to Section 3.3.5, and sparse weight matrices, refer to Section 3.3.4, which sigmoid function to use, and which distribution the weight matrices of the ESN should be initialized with. Echo State Property (ESP) is a stability property of ESN's to be satisfied, and it is described in Section 3.3.8. Harvesting internal states is the part the input is sent into ESN with to store all internal states over time. The bubble "Set up training algorithm" is meant to reflect the choice of algorithm as well as coherent hyperparameters. The tests "satisfying estimation" and "satisfying computational time" are very dependent on task and goal of the problem. 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.

3.3.2 Introduction to tank example

This case study provides a simple nonlinear system, ESN aims to model the rate of the liquid height dh/dt inside a tank. It is illustrated with a simple sketch in Figure 3.6. The

inflow Q_{in} is varying with time, being a result of factors outside this system. The outflow Q_{out} is for this case only dependent on the liquid height in the tank and not on any control factors such as valve opening. The dynamic equation of liquid height is derived from Bernoulli defined as follows:

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

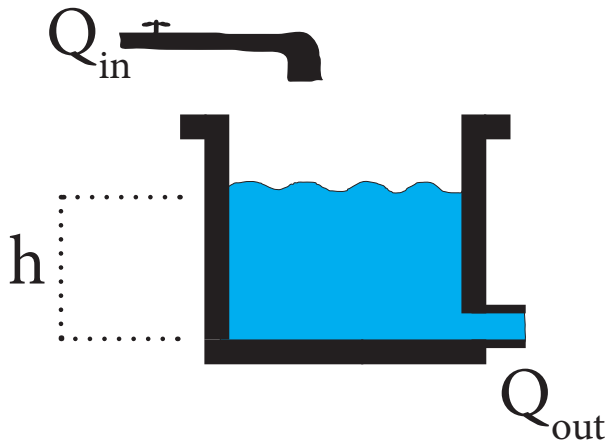


Figure 3.6: An illustration of tank

Table 3.1: States and parameters

h	Height of the liquid level [m]
$\dot{h}$	velocity of the liquid height $[m/s]$
g	gravity constant: $9.81[m/s^2]$
A_1	surface area of tank $[m^2]$
A_2	the outflow pipe dimension $[m^2]$
Q_{in}	volume inflow to the tank $[m^3/s]$

The tank model was generated in Matlab/Simulink. The training data used represents 10 000s with a sampling rate of 2s which gives 5000 data points. The test set is half the size, with 2500 data points. From Table 3.1 the input $\mathbf{u}$ for the ESN is Q_{in} and the output y is velocity of the height of the water level $\dot{h}$.

3.3.3 Build network and select reservoir size

In practice building a network means to set the necessary parameters and generate the weight matrices. For the tank example described in Section 3.3.2 the ESN reservoir is built as a fully connected weight matrix connected with a random uniform distribution. The parameters most significant parameters related to building the network are the reservoir size N_x and the range of the distribution σ. The system only has one input, the inflow to the tank. This gives $N_u = 1$. The system has one output, the derivative of the liquid level inside the tank. This gives $N_y = 1$. The weights matrices of the ESN W_{in}, W, W_{out} and W_{fb} are are given by the following dimensions: 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 $W_{fb} \in \mathbb{R}^{N_x \times N_y}$

A good size for the reservoir N_x is related to the limit of computation time, complexity of the system, acceptable error, and available training data. Normally the larger the reservoir size N_x the better is the performance. The drawback is that the time to train increases quadratically. Also the risk of overfitting increases with the size N_x. Figure 3.7 shows the response of estimations with ESNs for different reservoir sizes. One can here note that there is big gap in performance between $N_x = 100$ and $N_x = 500$, and that there was not much improvements going from $N_x = 500$ to $N_x = 1000$. Figure 3.8 shows the correlation between the computational time to train the network and the reservoir size. The same training set and parameters were used except for N_x and σ. For this task N_x= 500 can be considered a good value as it shows good performance and reasonable training time.

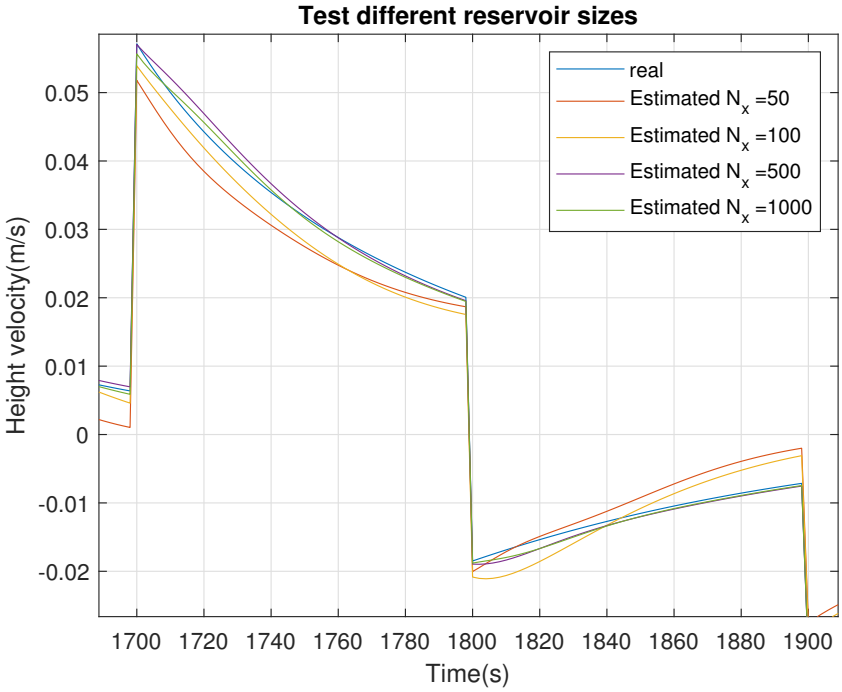


Figure 3.7: Performance for different reservoir sizes

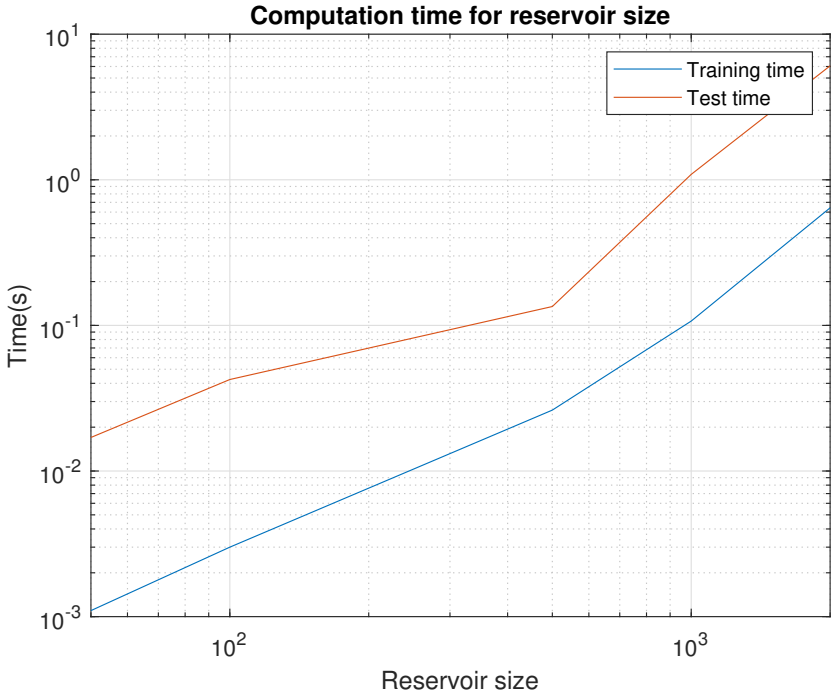


Figure 3.8: Computation time related to reservoir size in logarithmic scale

3.3.4 Sparsity

Here sparsity means the distribution of non-zero elements in the reservoir, making the connections in W_{in}, W and W_{fb} sparse. The goal of introducing sparsity to reduce the computation cost. This is generally considered when it comes to large matrices as the need to reduce computational cost increases. Sparsity was not included for the tank case with since N_x =500 meant the reduction in training time was negligible. Sparse weight matrices also reduces the randomness of the network performance, as the reservoir consist of fixed random weights.

3.3.5 Feedback

The decision to include feedback W_{fb} or not is task depending dependent. The general approach is to only include it if the task requires feedback due to the complexity, as it might decrease the stability and simplicity. When included for the tank case it proved to be more difficult to achieve the desired performance than without feedback. So for simple problems it is better to leave the feedback out.

3.3.6 Leak rate, α

Adjusting the leaking rate in practice means adjusting the speed of the dynamics of the internal states. So to find a good value for α knowledge related to the system dynamics can be used. If the estimation is too slow it might be a good idea to increase α. Another suggestion is to use a line search to select parameters.

Selecting leak rate α with a line search in 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 in Figure 3.9 and then with a smaller interval between 0.05 and 0.25 in Figure 3.10. The smaller area in Figure 3.9 was chosen because the sum of errors was the smallest at $\alpha = 0.1$ and $\alpha = 0.2$. 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 as depicted in Figures 3.10.

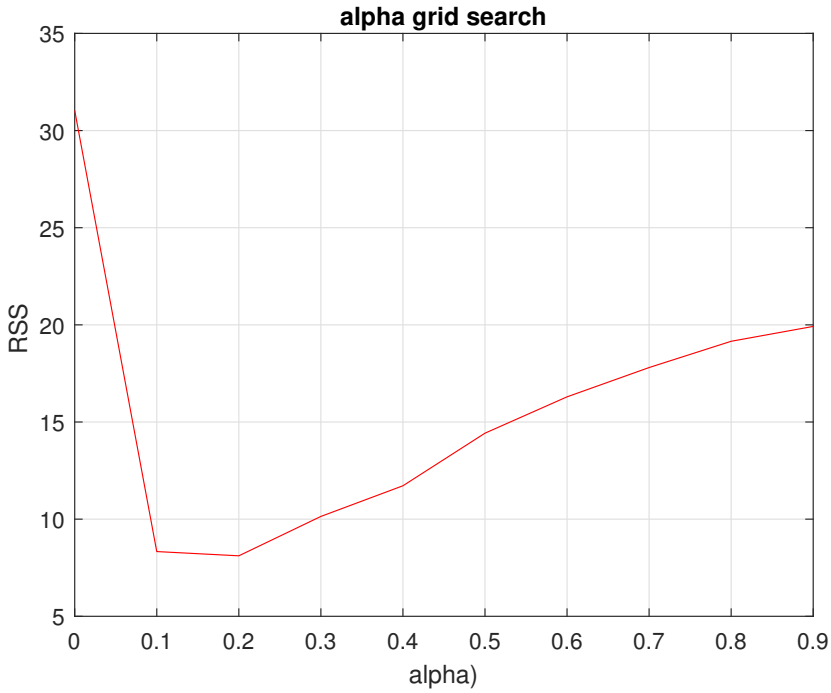


Figure 3.9: line search to the leak rate α, x-axis that minimizes the sum of error, y-axis. intervals of 0.1

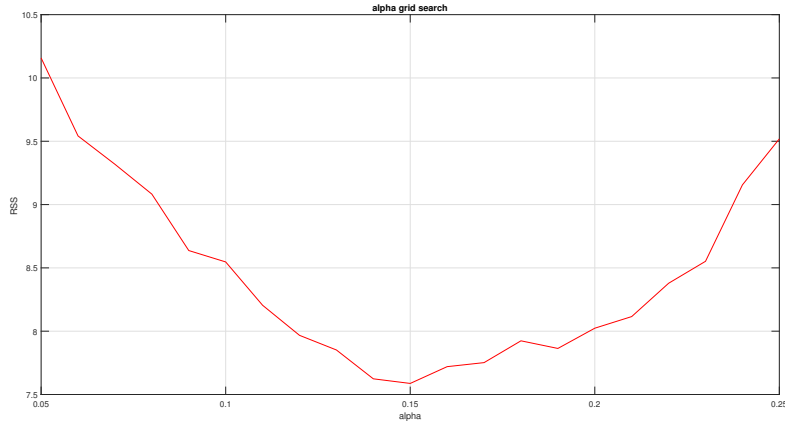


Figure 3.10: line search with intervals of 0.05

3.3.7 Range of distribution in the reservoir, σ

The distribution is a tuning parameter which is directly linked with the memory capacity. For a higher σ the input and internal states will contribute more to what the next state will be in Eq. 3.1. There is some guidelines for selecting σ towards satisfying the Echo State Property, which will be discussed in details in Section 3.3.8.

3.3.8 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 its initial states. The property is relevant to be able to "warm up" as the initial states of ESN often are wrong. This means the ESN able must be able to "wash out" the impact from initial states. For the first iteration there is no history so the initial states are therefore set to zero and then be inevitably wrong. Therefore it will have a hard time to estimate until the network has accumulated sufficient history. On the next page follows a definition of sufficient condition, necessary condition, and a suggestion for practical use of echo state property as proposed in [14] which is exemplified in the tank from Section 3.3.2. The definition uses $\mathbf{x}(\mathbf{h})$ and $\mathbf{x}'(\mathbf{h})$ as notation for the states from two identical ESNs with different initial conditions.

Definition for ESP from [14]: An ESN with reservoir states $\mathbf{x}(\mathbf{n})$ has the echo state property if for any compact $\mathbf{C} \subset \mathbb{R}^{\mathbf{k}}$, there exist a null sequence $(\delta_h)_{h=1,2,\ldots}$ such that for any input sequence $(\mathbf{u}(\mathbf{n}))_{\mathbf{n}=\mathbf{0},\mathbf{1},\mathbf{2},\ldots} \subseteq \mathbf{C}$ it holds that $\|\mathbf{x}(\mathbf{h}) - \mathbf{x}'(\mathbf{h})\|_2^2 \leq \delta_h$ for any starting states $\mathbf{x}(\mathbf{0}), \mathbf{x}'(\mathbf{0})$ and $h \geq 0$.

To summary the definition, for ESP to be satisfied an ESN with two different initial states $\mathbf{x}(\mathbf{0})$ and $\mathbf{x}'(\mathbf{0})$, the difference between the updated states with different initial condition will go towards zero. To illustrate the definition on the tank case from Section 3.3.2. Two different initial conditions for $\mathbf{x}(\mathbf{0})$ was created, one with random numbers between 0 and 1 and the second with all zeros initial state. From Figure 3.11 one observes that the output from ESN with different inital states are different in the beging before

moving towards each other. From Figure 3.12 it can be observed that the sum of errors $\sum_{i=1}^{N_x} |x_i - x_i'|$ (y-axis) for all internal states from two different initial states decreases to zero. Meaning that it becomes independent of the initial state. It is therefore likely to have the echo state property. However as the definition states the network property must hold for any input sequence and any starting states Figure 3.11 and Figure 3.12 does not yield as a proof of ESP.

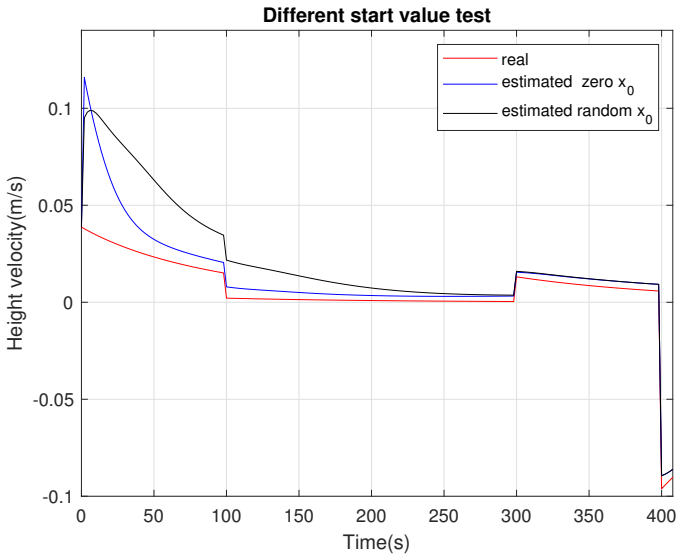


Figure 3.11: Estimations with different initial internal states.

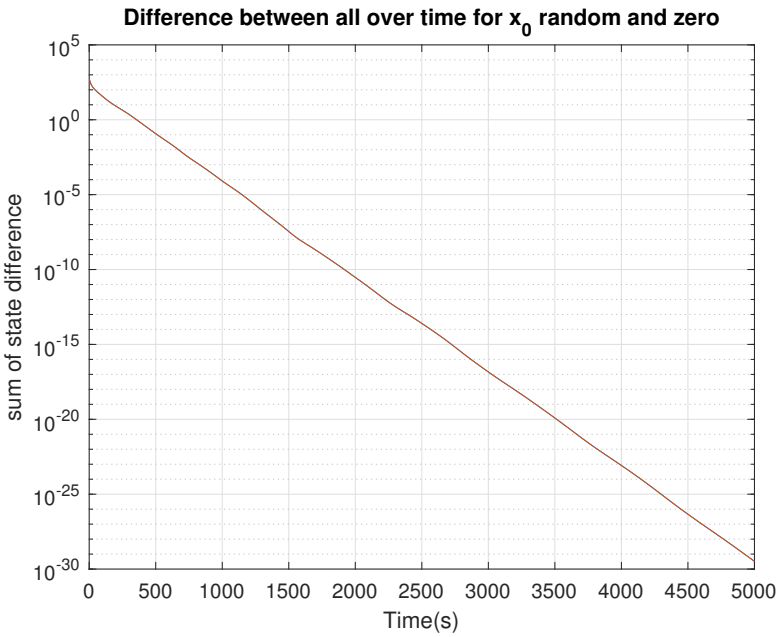


Figure 3.12: Sum of error between internal states (Note logarithmic scale).

Sufficient condition for ESP is proposed in [14]. Assume leaky a integrator ESN according to equation Eq. (3.1) where the sigmoid f is the tangent hyperbolic function $tanh$ and there are no output feedbacks, that is, $\mathbf{W_{fb}} = \mathbf{0}$. Let σ_{max} be the maximal singular value of $\mathbf{W}$. If:

$$|1 - \alpha(1 - \sigma_{max})| \leq 1 \tag{3.5}$$

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

Necessary condition for ESP as proposed in [14] assume a leaky integrator ESN according to equation Eq.(3.1), where the sigmoid f is the hyperbolic tangent $tanh$. Then we look at the matrix:

$$\tilde{\mathbf{W}} = \alpha\mathbf{W} + (\mathbf{1} - \alpha)\mathbf{I} \tag{3.6}$$

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:

$$\rho(\tilde{\mathbf{W}}) = \max(|eig(\tilde{\mathbf{W}})|) \leq 1 \tag{3.7}$$

A suggestion for practical use of ESN that is based on empirical data suggests that the spectral radius of W should be smaller than 1. For an ESN to have the Echo state property. This is however not a proof, nor does a larger spectral radius means that the ESN does not have the echo state property. It was also suggested that a value close to 1 for the spectral radius is a good choice [18].

$$\rho(\mathbf{W}) = \max(|eig(\mathbf{W})|) \leq 1 \tag{3.8}$$

The approach for selecting range of distribution of weights σ 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 values, excluding the time to converge (100s). The echo state analysis consist of a sufficient condition, a necessary condition and $spectralradius(W)$ (as described in section above) plotted against values of σ. The analysis was done using Ridge regression with $\lambda = 0.8$, $N_x = 1000$, and $\alpha = 0.15$. The training and test data was the same for all iterations.

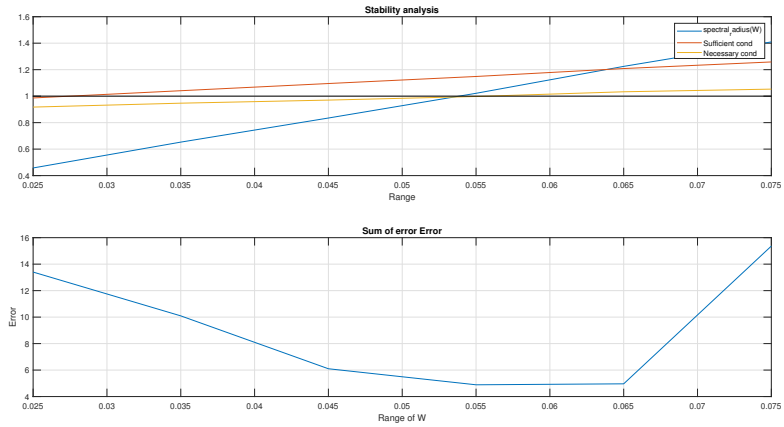


Figure 3.13: Echo state analysis (above) and error (under) as a function of σ with interval (0.025:0.05:0.075)

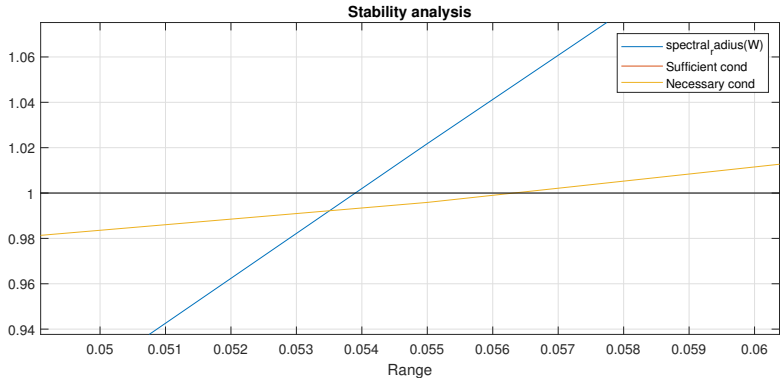
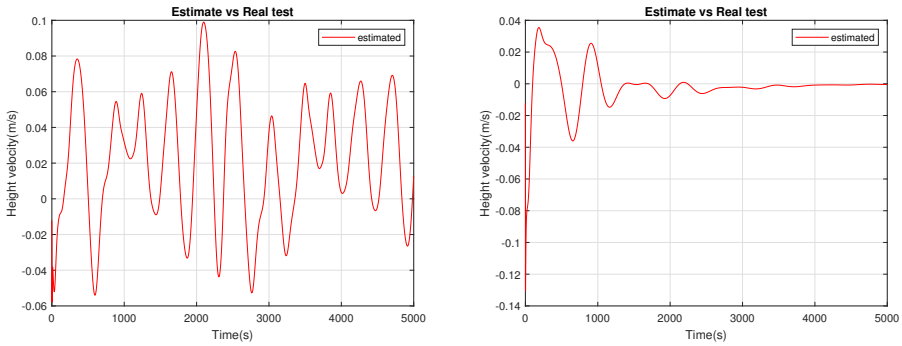


Figure 3.14: Echo state analysis of Figure3.13 zoomed

From Figure3.13 (below) it can be seen that the sum of errors is at its lowest with σ in the range of 0.055 to 0.065. However neither of those values satisfies the sufficient condition and have a spectral radius larger than 1. For $\sigma = 0.055$ it is barely under the necessary condition, Figure 3.14 while for $\sigma = 0.065$ is just above.

Another practical suggestion to check if the network do not have the echo state property is to implement zero input to the ESN and then investigate the stability. The

network does not have the echo state property if it is unstable for the zero input [14]. 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. To investigate this issue a zero input signal was injected into the network. The result from Figure 3.15 shows that $\sigma = 0.065$ is unstable and does not have the echo state property.



(a) σ=0.065 **(b)** σ=0.055

Figure 3.15: zero input stability test

3.4 Training the network

What is meant by training the network is to solve an optimization problem with the goal of minimizing the error between estimated output and known output data. 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 [18]. To train an echo state network one need to have initialized W_{out} usually in the same way as the other weight matrices, uniformly random. Training data is also necessary, is being given for a training period $n = 1 \dots T$ 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 iterate through the equations (3.1) and (3.2) with training inputs $\mathbf{U}$ for all timesteps n in order to collect estimated 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 Figure 3.5.

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N_x} \\ x_{21} & x_{22} & \cdots & x_{2N_x} \\ \vdots & \vdots & \ddots & \vdots \\ x_{T1} & x_{T2} & \cdots & x_{TN_x} \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1N_y} \\ y_{21} & y_{22} & \cdots & y_{2N_y} \\ \vdots & \vdots & \ddots & \vdots \\ y_{T1} & y_{T2} & \cdots & y_{TN_y} \end{bmatrix} \tag{3.9}$$

$$\mathbf{Y} = \mathbf{X}\mathbf{W}_{\mathbf{out}} \tag{3.10}$$

The training of W_{out} has the goal of minimizing the error between the estimated output $\mathbf{Y}$ from eq. (3.10) and $\mathbf{Y}_{\mathbf{target}}$. Eq. (3.10) is just eq. (3.2) in matrix version over time vertically. The training is then done with one shot learning using regression. The different training methods, ordinary least square, Ridge and LASSO will be detailed in the next sections.

3.4.1 Ordinary Least Squares (OLS) Regression

Ordinary Least Squares 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:

$$\mathbf{RSS}_{OLS} = \sum_{i=1}^{n}(y_i - (\beta_0 + \sum_{j=1}^{p}\beta_j x_{ij}))^2 = \|\mathbf{Y}_{\mathbf{target}} - \mathbf{X}\mathbf{W}_{\mathbf{out}}\|_2^2 \tag{3.11}$$

$\beta \in \mathbb{R}^{(p+1)\times 1}$ is the variables that will be solved in the optimization problem. The vector consisting of all β is $\mathbf{W}_{\mathbf{out}}$ in an ESN. Recall from equation (3.2) how $\mathbf{Y}$ is estimated. The hyperplane is made up of these parameters and internal states over time $\mathbf{X}$. $\mathbf{RSS}$ 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 can be obtained by multiplying with $\mathbf{X}^{\mathbf{T}}$ in eq. (3.10) and then isolate $\mathbf{W}_{\mathbf{out}}$ which gives:

$$\mathbf{W_{out}} = (X'X)^{-1} X'\mathbf{Y_{target}}, \quad \boldsymbol{W_{out}} = \begin{bmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1N_y} \\ \beta_{21} & \beta_{22} & \cdots & \beta_{2N_y} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{T1} & \beta_{T2} & \cdots & \beta_{TN_y} \end{bmatrix} \tag{3.12}$$

3.4.2 Ridge regression

Ridge regression is also known as Tikhonov regularization and $L_2 - Regularisation$. It consists of the same part as $\mathbf{RSS}_{OLS}$ in eq. (3.11) and a parameter λ times the weights of W_{out}^2. By adding this new term to OLS a penalty on high weights which will reduce the chance of overfitting, [10].

$$RSS_{\text{Ridge}} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{i=1}^{p} \beta_j^2 = RSS_{OLS} + \lambda \sum_{i=1}^{p} \beta_j^2 \tag{3.13}$$

The analytic solution will now be [10]:

$$\mathbf{W_{out}} = (X'X + \lambda I)^{-1} X'\mathbf{Y_{target}} \tag{3.14}$$

3.4.3 LASSO

The Least Absolute Shrinkage and Selection Operator LASSO is a regularization algorithm much similar to Ridge regression. 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, [25]. This provides a simpler model but no analytic solution

exist.

$$RSS_{LASSO} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{i=1}^{p} |\beta_j| = RSS_{oLS} + \lambda \sum_{i=1}^{p} |\beta_j| \tag{3.15}$$

3.4.4 Comparison of LASSO, RIDGE and OLS

The differences in practice between LASSO and Ridge is shown with histograms of the W_{out} matrices. The histograms are based on $N_x = 1000$ and no sparsity. The difference between Ridge regression are visualized with Figure 3.16 and Figure 3.17, that LASSO drives the output weights to zero. The output produced by LASSO is

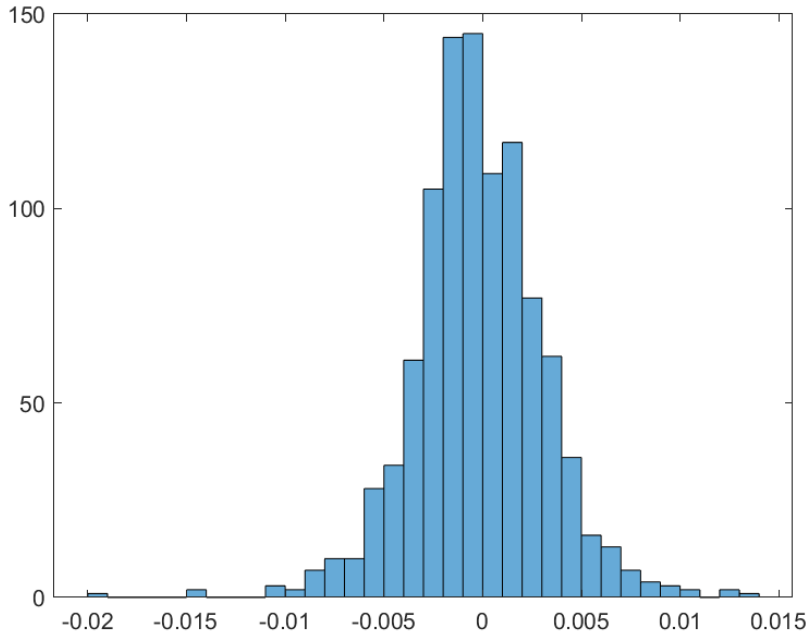


Figure 3.16: Histogram Ridge regression W_{out}, x-axis is the value of the weights in W_{out}, y-axis is number of weights

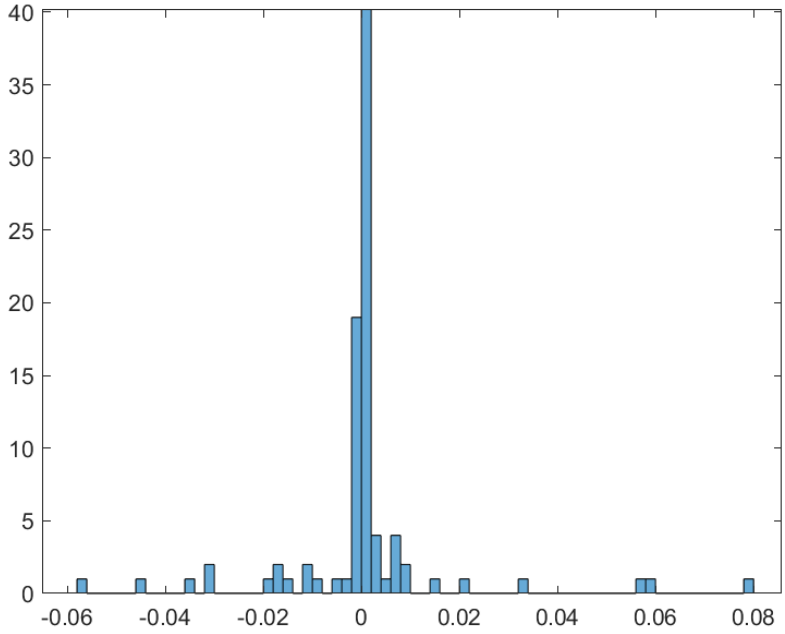


Figure 3.17: Histogram LASSO W_{out}. x-axis is the value of the weights in W_{out}, y-axis is number of weights. 62 of 1000 weights are non-zero

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 [19]. This was done using a standard laptop computer. LASSO has a high training time as it uses Matlab's built-in LASSO function tests it for different λ, parameter while for Ridge and OLS were solved using the analytic solution from Section 3.14 with λ set manually. The difference in computation time is represented in the table 3.2 and the difference in performance is represented in Figure 3.18 and Figure 3.19.

	Training time(s)	Running time(s)
Ridge, $N_x = 1000$	0,1044	1,3915
OLS, $N_x = 1000$	0.0982	1.3924
LASSO, $N_x = 1000$	107,3330	1.2847
LASSO, $N_x = 500$	56.4692	0.2792

Table 3.2: Table, with results of training and running time for different training methods

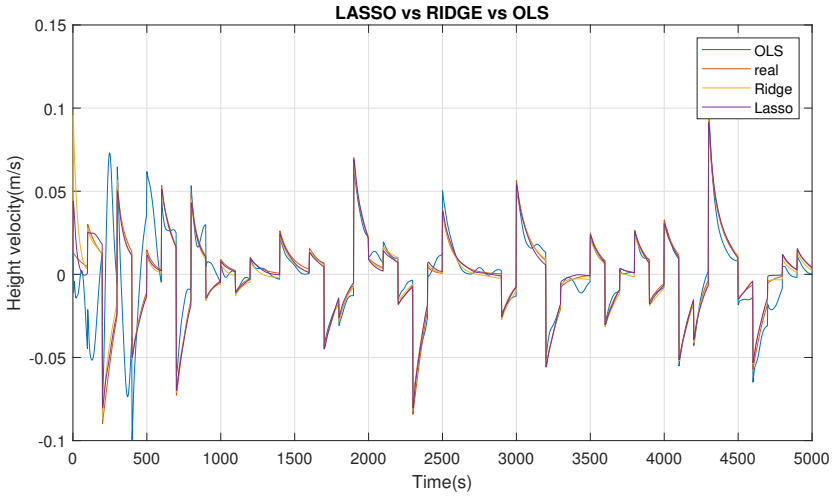


Figure 3.18: Test set: results of estimation with different regression methods.

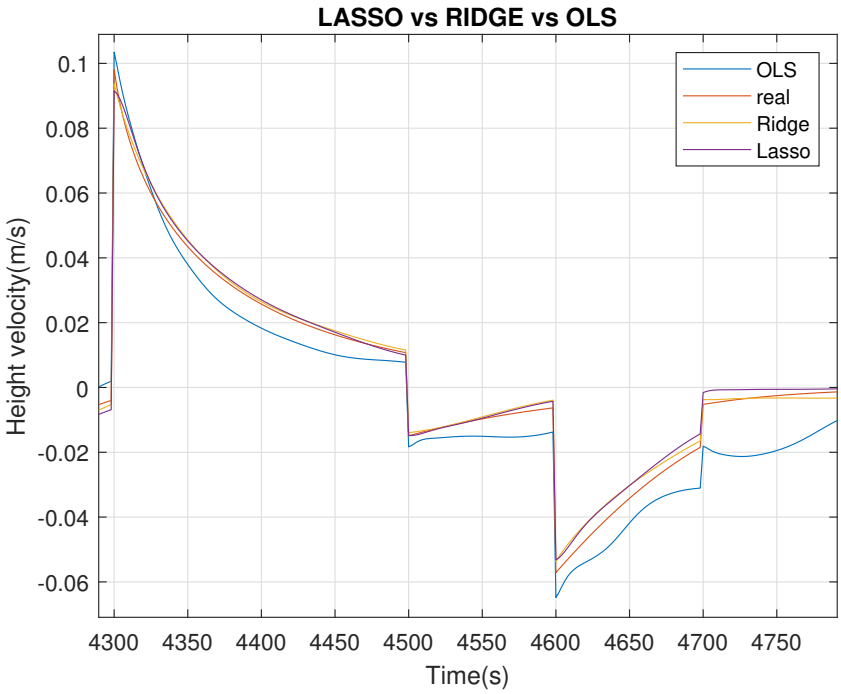


Figure 3.19: Figure 3.18 zoomed

Ridge and LASSO regression produce equal good results while OLS has a weaker performance. Ridge regression is faster to train as the analytic solution can be used. LASSO has one advantage that it has a simpler model due to most of the output weights being zero. Ridge regression would be the standard method to use when training ESN and LASSO and OLS could be beneficial in special cases.

Chapter 4

VFM on a Three-phase Gravity Separation Tank

This chapter aims to give an overview over the problem. The chapter start with an introduction to the system identification of a three-phase gravity separation tank. Section 4.1 works as a summary of the most relevant part of a model for a three phase gravity separation tank previously implemented by Backi [4]. His model will be used as a plant model in substitute of a real tank as this a for the studies herein. Next follows an introduction to the virtual metering problem for the gravity separator tank. A discussion of the benefits and how the situation is today, including a discussion of realizability and possible problems. Then comes a part of a suggested solution with data-driven modeling using an ESN for system identification. The chapter ends with a basic introduction to Extended Kalman Filtering for virtual flow metering on the three phase gravity separation tank. Section 4.4 also works as a summary of the most relevant parts of an observer for a separation tank previously implemented Backi [1].

4.1 Tank model

A three-phase gravitation separation tank is a rough separation stage from a well stream. Typically it is the first step in the separation process. The well stream will is composited of gas, oil and water and various effects and controls influence the composition and inflow rate into the tank. The goal of the tank separator is to separate gas, oil and water. It is driven by gravitational forces. This is possible because of the different densities of the elements.

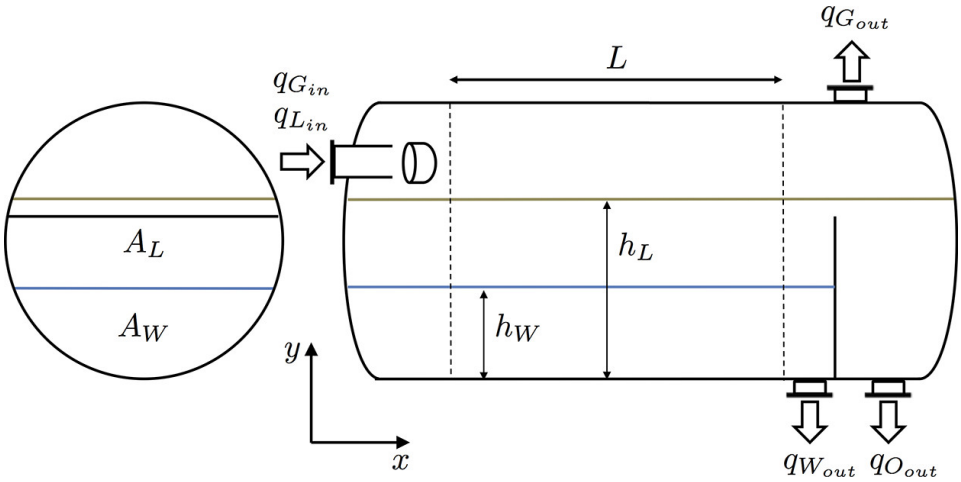


Figure 4.1: Sketch of a dish-head separation tank from [4].

A gravity separation tank, as seen in Figure 4.1, consists of three zones: the inlet zone, the active separation zone and the outlet zone. The active separation zone is indicated between the two dashed line with length L with the inlet zone to left and outlet zone to the right. The tank also consists of three different phases, water phase beneath the blue line given h_w, liquid phase (oil and water) above the blue line and under the brown line h_L, and a gas phase above the brown line. There is a physical barrier between the oil and gas outflow in order to separate the outflows so that only the oil phase reaches the oil outlet marked by $q_{O,out}$. Depending on the effectiveness of the separation there will be an amount of water in the oil phase and oil in the water phase in the outlet. A three phase separation tank is considered the first step in the on-shore production.

A Control- and Estimation-Oriented Gravity Separator Model was created by [4]. The plant model is implemented in Matlab and Simulink. It consist of a three main modules: a controller, the dynamic equation part, and a static equations part for droplet calculations. The main variables in this system are given in Tables 4.1, 4.2, 4.3.

Table 4.1: The load - Inflow to the tank

$q_{L,in}$	Liquid inflow (water and oil) to the tank $[m^3/s]$
$q_{G,in}$	Gas inflow to the tank $[m^3/s]$
γ	Split factor, fraction of liquid entering into the water phase

Table 4.2: Control variables - Outflow from the tank

qW_{out}	Water phase outflow from the tank $[m^3/s]$
qO_{out}	Oil phase outflow from the tank $[m^3/s]$
qG_{out}	Gas phase outflow from the tank $[m^3/s]$

Table 4.3: Internal states - Measurements in the tank

h_w	Height of the water level [m]
h_l	Height of the total liquid level, water and oil [m]
p	Pressure inside the tank $[N/m^2]$

The split factor γ is a variable dependent of several factors. This is in contrast to the seemingly similar watercut α which denotes the water fraction of the total liquid inflow q_{in}. The water cut is only a result of the composition of the inflow. This is because of the mixture in the phases. In the model it is assumed that all the inflow gas goes into the gas phase. However not all oil goes into the oil phase, and not all water goes to the water phase. Therefore the ϕ_o fraction of inflow oil going into the oil phase and ϕ_w is the fraction of inflow water going into the water phase. These variables α, ϕ_o, ϕ_w also have their counterparts $(1-\alpha)$, $(1-\phi_o)$ and $(1-\phi_w)$ respectively the fraction of oil in the total liquid entering the tank, the fraction of oil going into the water phase, fraction of water going into the oil phase. This gives the relation to the split factor. The fraction of liquid

entering into the water phase is:

$$\gamma = \alpha\phi_w + (1-\alpha)(1-\phi_o) \tag{4.1}$$

4.1.1 Controller

In the separation tank system one module is the controller, it is a PI controller and it implementation is detailed in [4] . The controller is controlling the states in Table 4.3. All of these states are crucial as a gravity separator has working conditions necessary to work properly. Such conditions are the water and liquid levels. If the water level h_w is high lo you will get water in the oil phase, and if it is too low you will get oil in the water phase. As can be seen in Figure 4.1, the liquid level h_l has a minimum height. If the liquid level is too low the oil phase will not reach the output and not be able to produce oil. The pressure can also have minimum and maximum values to ensure safe and stable operation. A controller has not only the task to ensure bounds but also optimizing production, stability, and separation. Since it gives the possibility of actively changing h_w, h_l and p. This is useful in several situations as it can change the effectiveness of the separation and the production of oil and gas or a desire to keep these states constant. The control variables are flow rates given by Table 4.2. These values are available with good measurements of flow meters. Physically controlling the outflow is here assumed controlling a valve. For simplicity, as there is direct relation between the output flow rate, and the valve opening flow rates are used as the unit. A saturation is included to represent the maximum valve opening. Here it is not assumed a minimum outflow other than zero as the flows only leave the tank. The controllers input are the states of Table 4.3 with respective reference values for h_w, h_l and p representing the desired value. These references are inputs to the system as a whole.

4.1.2 Dynamic Part Model

The dynamic part is a system like in Figure 2.1 describing an input-output relation. For the dynamic part it is a system where the input is the inflow (Table 4.1) and outflow (Table 4.2) and output is the measurements (Table 4.3). This input-output relation is described by three Ordinary Differential Equations (ODE) (4.2), (4.3), (4.4) for the states in Table 4.3. For a visual representation of the dynamic system, refer to Figure 2.1. The inputs enter the ODEs and then follows an integrator block. The output after the intigrator is then the modeled measurements for the internal states. The parameters for these ODEs (Equations (4.2), (4.3), (4.4)) are: r is the radius of the tank, L is the length of the active separation zone, R is the universal gas constant, T is the temperature, ρ_G is the density of gas, M_G is the molar mass of gas, V_{Sep} is the volume of active separation zone, A_L is the cross segment of the liquid area. The variables: water leaving the oil phase V_{Oil}^{Water} and oil leaving the water phase V_{Water}^{Oil} are given by adding the droplet calculations of Figure 4.2.

$$\begin{aligned} \frac{dh_L}{dt} &= \frac{dV_{\mathrm{L}}}{dt} \frac{1}{2\mathrm{L}\sqrt{h_{\mathrm{L}}\left(2r - h_{\mathrm{L}}\right)}} \\ \frac{dV_L}{dt} &= q_{L,in} - q_{L,out} \end{aligned} \tag{4.2}$$

$$\begin{aligned} \frac{dh_W}{dt} &= \frac{\mathrm{d}V_{\mathrm{W}}}{\mathrm{d}t} \frac{1}{2\mathrm{L}\sqrt{h_{\mathrm{W}}\left(2r - h_{\mathrm{W}}\right)}} \\ \frac{dV_W}{dt} &= q_{W,in} - q_{W,out} + V_{Oil}^{Water} - V_{Water}^{Oil} \end{aligned} \tag{4.3}$$

$$\frac{dp}{dt} = \frac{RT\frac{\rho_G}{M_G}\left(q_{G,\mathrm{in}} - q_{G,\mathrm{out}}\right) + p\left(q_{L,in} - q_{L,out}\right)}{V_{Sep} - A_L L} \tag{4.4}$$

4.1.3 Static Model Equations

The static model equations represents the droplet distribution calculations. These equations are divided into three subsystems: first subsystem for the calculation of oil particles

leaving the water phase, second subsystem for the calculation of water droplets leaving the oil phase and third subsystem consist of a function for the efficiency calculation of oil and water removal. A model based on computational fluid dynamics [11], concluded that the size of droplets affect the velocities of the separation. In the liquid phase of the separation tank there happens phenomena like sedimentation (droplets settling), and creaming (droplets rising).

The calculation of oil leaving the water phase and oil leaving the oil phase is carried out with the same principle. Calculating the horizontal and vertical speed of a particle to make an estimate of the number of particles leaving their respective bulk phase. Firstly the length of the tank in the x-direction is divided into N_s segments. This is done such that the calculations are done for the segment and then added together for so to be averaged. This deviding of the active separation zone into segments is giving a more accurate model. The model does not includes any timedelay for the segments. For a real tank the residence time for droplets, the time one droplet uses to leave one segment in horizontal direction would impact the process. This leads to variation in the inflow and outflow to the tank that will instantly affect the states [4].

The velocity in the vertical direction is here assumed to be given by Stokes law. Not including a decelerating correction factor as introduced in [24]. It gives the velocity of a droplet rising or settling Eq. (4.5). It is used both for oil droplets in the water phase and water droplets in the oil phase. The initial distribution of droplet particles is therefore affecting the separation of this model.

$$v_v = \frac{gD^2\left(\rho_{\mathrm{d}} - \rho_{\mathrm{c}}\right)}{18\mu_{\mathrm{c}}} \tag{4.5}$$

Where v_v is the vertical velocity of a droplet, g is the gravitational acceleration, D is the diameter of the droplet, ρ_{d} is density of the dispersed droplet fluid, ρ_{c} is the density of the continuous fluid the droplet is dispersed in, and μ_{c} is the viscosity of the continuous fluid the droplet is dispersed in. The residence time for a droplet is then given in Eq. (4.6). The oil phase height is given as $h_O = h_L - h_W$. Residence time of a segment is important as it determines wether or not a given droplet size will leave the continuous phase or not for

a segment.

$$t_{v_i} = \begin{cases} \frac{h_{\mathrm{W}}}{v_{v_i}} = \frac{18\mu_{\mathrm{O}}}{gc_i^2(\rho_{\mathrm{W}}-\rho_{\mathrm{O}})}h_{\mathrm{W}} \\ \frac{h_{\mathrm{O}}}{v_{\mathrm{i}}} = \frac{18\mu_{\mathrm{W}}}{gc_i^2(\rho_{\mathrm{O}}-\rho_{\mathrm{W}})}h_{\mathrm{O}} \end{cases} \tag{4.6}$$

The horizontal speed for the water and oil phase is assumed to be equal to the the volumetric inflows to the tank q_W, q_O of a given phase. This means that the time a droplet would use to leave its segment called residence time, and is given by Eq. 4.7. A segment has the length $\frac{\mathrm{L}}{N_{\mathrm{S}}}$, with cross segment area of oil and water A_O, A_W.

$$t_{h_{\mathrm{W}}} = \frac{\mathrm{L}}{N_{\mathrm{S}}}\frac{A_{\mathrm{W}}}{q_{\mathrm{W}}} \tag{4.7}$$

$$t_{h_{\mathrm{O}}} = \frac{\mathrm{L}}{N_{\mathrm{S}}}\frac{A_{\mathrm{O}}}{q_{\mathrm{O}}} \tag{4.8}$$

By intuition the calculation for droplets leaving its bulk phase is simple. An initial distribution of position and class are given. Then for the first segment an if-else statement is done. If the time for a droplet to leave the segment horizontally t_hw and t_ho is smaller than the time to leave the bulk phase t_vi. A new position for the droplet is based on time traveled vertically. If the statement is false and the time to leave the bulk phase is smaller than the time to leave the segment horizontally, then the droplet class is added to its new phase and removed from the bulk phase. This algorithm is depicted in Figure 4.2 for one segment $m \in N_S$ with i representing one of the 500 different classes.

for $i = 1{:}\ 500$

if $t_{h_W} < t_{v_i}$

$$pos_i^{\text{segment } m} = t_{h_W} v_{v_i} + pos_i^{\text{segment } m-1}$$

$$n_i^{\text{into segment } m+1} = n_i^{\text{from segment } m-1}\left(1 - \frac{t_{h_W} v_{v_i}}{h_W - pos_i^{\text{segment } m-1}}\right)$$

$$V_i^{\text{into oil phase}} = n_i^{\text{from segment } m-1} V_i^{\text{droplet}} \frac{t_{h_W} v_{v_i}}{h_W - pos_i^{\text{segment } m-1}}$$

else

entire volume of droplet class i goes into its bulk phase

end

end

Figure 4.2: The algorithm for oil droplet calculation for one segment from [4].

Where $pos_i^{segment\ m-1}$ is the vertical position of the class i at the end of the segment m and $n_i^{into\ segment\ m+1}$ gives the number of droplets that leaves segment m into $m+1$ still being dispersed without leaving its bulk phase. $V_i^{droplet}$ defines the volume for one droplet in the class i. $V_i^{into\ oil\ phase}$ is the volume of oil leaving the waterphase for one class. The algorithm is performed in a series for every segment. The algorithm in Figure 4.2 can easily be changed to calculate water droplets by simply change the parameters to fit water calculation. Recalling equation (4.3) ODE for water height included the variables water leaving the oil phase V_{Water}^{Oil} and oil leaving the water phase V_{Oil}^{Water}. V_{Water}^{Oil} is found by simply adding $V_i^{into\ oil\ phase}$ for all classes first and then all segments N_S. And similarly for V_{Oil}^{Water} adding $V_i^{into\ water\ phase}$ for all classes first and then all segments N_S.

The calculation of efficiency of oil removal is dividing the amount of oil that left the water phase V_{Water}^{Oil} with the amount of oil existing in the begining. And for effeciency of water removal it follows the same way by deviding the volume of water that left the oil phase with the volum inital water in oil phase. These tells how effective the separation is. This is useful as the measurements of oil in water and water in oil are often not reliable or even not available.

When the inflow changes value, so does the residence time do to. Meaning the inlet zone will move. This introduces a scaling factor for the initial distribution of water and oil droplets. These factors are given by Eq. (4.9)

$$F_{\mathrm{W}} = t_{h_{\mathrm{o}}}^{\mathrm{inlet}} \left[\alpha \left(1 - \phi_{\mathrm{w}}\right)\right] q_{\mathrm{L,in}} \frac{1}{V_{\mathrm{W}}} \tag{4.9}$$

$$F_{\mathrm{O}} = t_{h_{\mathrm{w}}}^{\mathrm{inlet}} \left[\left(1 - \alpha\right) \left(1 - \phi_{\mathrm{o}}\right)\right] q_{\mathrm{L,in}} \frac{1}{V_{\mathrm{O}}} \tag{4.10}$$

4.2 Virtual Flow Meetering on a Three-phase Gravity Separator

Virtual flow metering is as mentioned in Section 2.3 the estimation of the wellstream. The goal of this research is to achieve a virtual flow metering. This is done by creating a data-driven model approach using ESN. The data is the information from the three-phase gravity separator with the controller described in Section 4.1.

Knowledge about the composition of a wellstream is something major oil companies need. The normal procedure for today in the oil industry is to once a month to allocate the wellstream into a test tank. This test separation tank will then be used to investigate the composition of the wellstream. This information is used in the plan for production, to detect and to prevent slugging incidents to increase stability and safety.

Measurements for $q_{L,in}$ and $q_{G,in}$ can be obtained using multiphase metering. However multiphase metering has a poor accuracy as well as being expensive. To obtain measurements for γ several methods are proposed but they have some drawbacks. Low-field Nuclear magnetic resonance (NMR) is sample-based and not available for online measurement [26]. Electrical resistance tomography (ERT) only works for certain flow regimes, and additionally calibration might be difficult [9].

To achieve the composition of the wellstream online an alternative to instruments is the use of VFM. Online data can be used to better plan the production. This could be in

meeting production demands and optimal use of equipment to reduce maintenance. Also to prevent or reduce the interruptions in the production by allocating the wellstream in order to measure the composition. Lastly also in the case with systems including a multiphase metering a virtual model could be used as a redundancy if the instrument would not be available for a period of time.

The inflow to the gravity separation tank can have variations. This can be caused by a change of one wellstream in composition and volume rate as the multiple wellstreams that enters the same tank may change. A gravity separator will give different results in effectiveness of the separation and production depending on the inflow. To get online data on the inflow will help to plan the control of a separation tank.

A data-driven black-box modeling approach to achieve VFM for the tank system described in Section 3.3.2 means finding a model for estimating the inflow to the tank q_{Lin}, q_{Gin} and γ. The input to the model can then be the measurements of the outflows. These are available data. It can be seen from the ODEs Equations (4.3),(4.2) and (4.4) that there is a relation from the inflow and outflow to measurements. For the plant model itself the actual inflow into the separator can be considered a disturbance since it is not controllable and is assumed to have no available measurements.

The difficulty of measuring the inflow is the biggest difficulty of a data-driven method for VFM. Good data is required to create a good data-driven model. It is not normal with multiphase flow measurment instuments in the industry today. If the inflow measurements were available, online learning could also be used such that the model is updated as parameters and conditions change with time. This could give one benefit over a theory-driven model as it can be difficult to find the correct parameters as they might be changing over time.

4.3 Echo State Network as virtual flow meetering

To create the data-driven model for VFM, an ESN is used. The structure of ESN is described in Section 3.3.8. The Input to ESN will then be the outflow rates of the gas phase $q_{G,out}$, oil phase $q_{O,out}$ and water phase $q_{W,out}$, and the internal states are the water level h_w, liquid level h_l and pressure p. The output of the ESN will be the estimated incoming load: gas inflow rate $q_{G,In}$, liquid inflow ratio $q_{L,In}$ and the split ratio fraction of liquid entering into the water phase γ. Training data from the simulation of the model in Section 4.1 will be used to learn this input-output relation. To train the ESN one needs measurements that are typically unavailable. The overall structure of the system is shown in Figure 4.3.

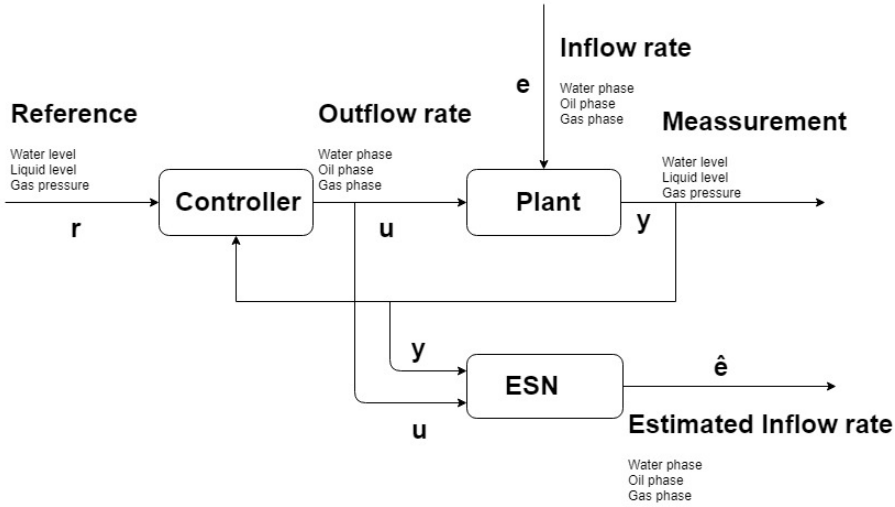


Figure 4.3: Structure of ESN and plant model

For the internal states h_w, h_l and p there exists several methods to obtain these measurements in the real world. To obtain the pressure p there exists instruments like Bourdon tubes, and piezoresistive pressure sensor [23]. h_w and h_l are a more complex to obtain. One method is single-electrode capacitance probe which assumes clear interfaces and has disadvantages in spatial resolution [16]. Another method to measure h_w, h_l is ultrasonic-based devices for accurate measurement of oil, emulsion and water levels

are presented in [20] and nucleonic level measurements are used which provide density measurements and hence enable determination of phase interface locations [21]. With measurements for h_w, h_l and p they might be subject to bad calibration, sensor drift and measurement noise.

For the controller variables $q_{G,out}$, $q_{O,out}$, $q_{W,out}$ there exist several ways to obtain real life data. An approach with a valve controller that takes in reference for desired outflow and the measurement from a flow meter. Flow meters come in a wide range of different complexity, precision robustness and cost.

4.4 Extended Kalman Filter for Virtual Flow Metering

Another approach for VFM is to estimate the load (q_{Gin},q_{Lin},γ) using the Extended Kalman Filter (EKF) observer. This is done in [3]. The extended Kalman filter method is results from the combination of one estimation using a model and measurements. The model equations are shown in Eq.(4.11) and are based on models from Section 4.1. The main difference between the plant and observer model lies in assuming that the inflow is stable. For a varying load EKF will have a deviation in the model estimation of internal states and measurements of internal states which leads to a change in the final estimation of the load.

$$\begin{aligned}
\frac{d\hat{h}_L}{dt} &= \hat{f}_1 = \frac{\hat{q}_{L,in} - q_{W,out} - q_{O,out}}{2L\sqrt{\hat{h}_L\left(2r - \hat{h}_L\right)}}, \qquad \frac{d\hat{q}_{L,in}}{dt} = \hat{f}_4 = 0 \\
\frac{d\hat{h}_W}{dt} &= \hat{f}_2 = \frac{\hat{q}_{L,in}\hat{\gamma} - q_{W,out}}{2L\sqrt{\hat{h}_W\left(2r - \hat{h}_W\right)}}, \qquad \frac{d\hat{q}_{G,in}}{dt} = \hat{f}_5 = 0 \\
\frac{d\hat{p}}{dt} &= \hat{f}_3 = \frac{RT\frac{\rho_G}{M_G}\left(\hat{q}_{G,in} - q_{G,out}\right) + \hat{p}\left(\hat{q}_{L,in} - q_{L,out}\right)}{V_{Sep} - \hat{V}_L}, \quad \frac{d\hat{\gamma}}{dt} = \hat{f}_6 = 0
\end{aligned} \tag{4.11}$$

As the model estimation for $q_{L,in}$ and h_L are assumed independent of the other estimations Eq.(4.11). For a simpler tuning the observer is then decoupled into cascaded observers as

shown in Figure 4.4. A more detailed overview of the observers can be seen in Figures 4.5a and 4.5b.The Riccati block is a solver for calculating the kalman gains K_{O1} and K_{21}. The Kalman gains are defining the ratio of how much you trust the model and how much you trust the measurements. The tuning parameters of EKF are two matrices Q and R inside the Riccati block, which relates to which variables should be prioritized to estimate, putting a penalty on the error for the different variables. The "Jacobian Observer" block is the Jacobian matrix of the observer model Equations (4.11).

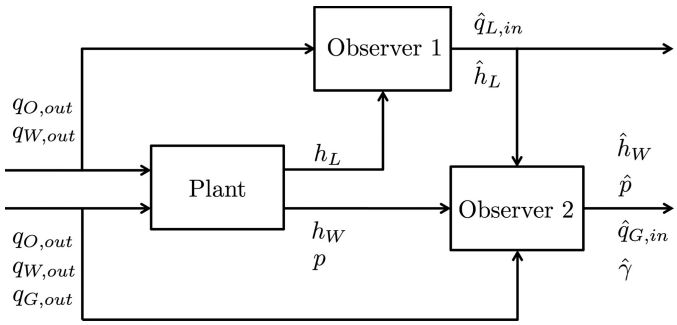
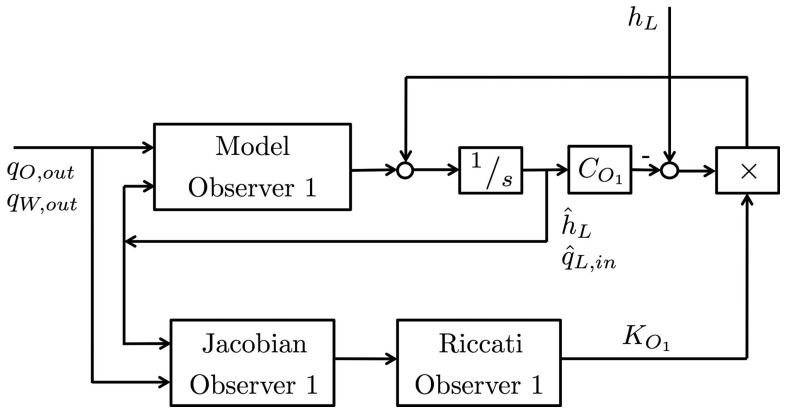
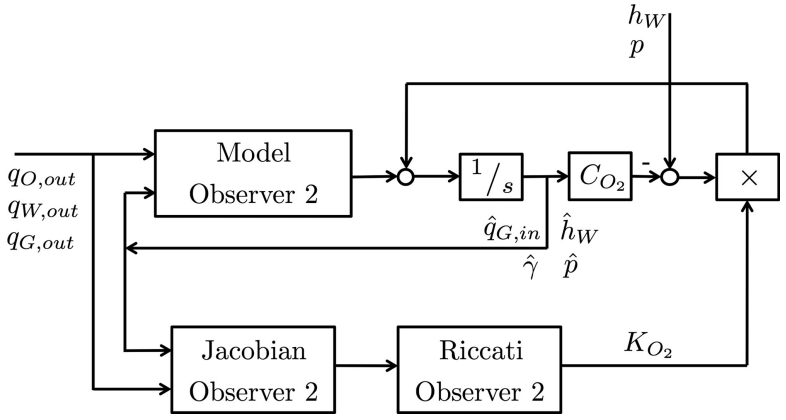


Figure 4.4: The cascaded structure of the EKF illustration from [1].



(a) Observer 1 illustration from [1]



(b) observer 2 illustration from [1]

4.5 Summary of Problem Description

There is an interest from the industry towards virtual flow metering and there exists previous research on the topic like [1]. A comparison between the two methods EKF and ESN is interesting, because they have different strengths and weaknesses as EKF is theory-driven and ESN is data-driven. The assumptions towards obtaining all the ESN input data from a real world system are reasonable, as they are all measurable. There is however difficult to obtain training data for the ESN output. The inflow being difficult to obtain the data for is also part of what motivates this work. The system described in Section 3.3.2 is a

complex nonlinear system. To be able to successfully recreate a model using ESN would be a demonstration of systems suitable for system identification using ESN.

Chapter 5

Experiments and Results

This chapter will first present the creation of the training set which is based on Section 4.1. A section on the building of the ESN follows with selecting hyperparameters based Figure 3.5. The first experiment, concerns a varying load and constant controller references. For that the ESN produced good estimations. The second experiment allowed a varying controller reference for water level. The results were not optimal which led to suggestions for improvements. The improvements provided the ESN with more information by adding data from a previous timestep and controller references as input to ESN. A change of training parameter λ for Ridge regression was also analysed. The third experiment was to include noisy measurements which had a minimal impact on the estimations. The fourth experiment was a comparison between ESN and EKF estimation.

5.1 Training Set

The creation of the training and the test set followed the model discussed in Section 4.1. The signals for the inflow to the tank, $q_{G,in}$, $q_{L,in}$ and γ where allowed to vary over time. The variable γ is not varied directly. Instead the system has a varying watercut α which is

γ with a bias, because ϕ_o and ϕ_w for these simulations are constant [see Eq. (4.1)].

These signals for $q_{G,in}$, $q_{L,in}$ and γ were created by a random number generator with uniform distribution and a given range. A filter was added to give the variation a more smooth and realistic behaviour. The variation consists of two signals with different frequencies and amplitudes added together. The high frequency holds the values $q_{G,in}$, and $q_{L,in}$ for 20 seconds whereas for γ the high frequency holds a value for 10 seconds. The low frequency holds the values $q_{G,in}$ and $q_{L,in}$ for 200 seconds whereas for γ the low frequency holds the values for 100 seconds. The high frequency was made to occurs in different timespans for γ, $q_{G,in}$ and $q_{L,in}$. This was done in order to create a variate training set. The amplitude of the high frequency is the low frequency amplitude divided by 1.3. The point of having two frequencies is to simulate for different scenarios such that ESN will not learn specific frequency. These variations on frequencies and ranges are depicted in Figure 5.1.

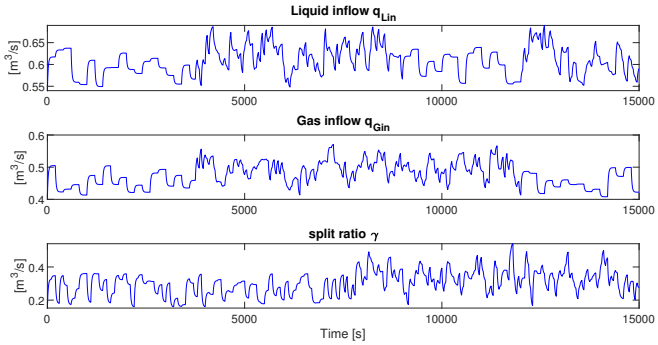


Figure 5.1: Training set data for inflow.

The training set is from a simulation over 15 000 seconds with a sampling rate of 0.5 seconds. A standard laptop performs the simulations in about one minute. The controller described in Section 4.1.1 helps keeping the data inside the valid envelope. Its reference values are for the first experiment to be kept constant. When a controller is included it is likely that the ESN to some extent also learns this correlation from state measurements to controller variables. An alternative would be to directly feed random

outflows to the controller. This could induce a good variation in the training set. The drawback is that it does not represent a realistic behavior and proved difficult to upheld the conditions. From Figure 5.2 one can notice that the controller try to keep the states stable as the inflow changes. Other remarks can be made by looking at the areas of which high frequencies occurs in the water level, which does not seem to be highly correlated by $q_{G,in}$ and $q_{L,in}$ but more with γ. It vary with a low frequency from time 0 to 7500 and with a higher frequency from time 7500 to 15000 which is the behavior as γ has in Figure 5.1. By investigating Figure 5.3 it seems that gas outflow $qGout$ correlates with with the gas inflow $qGin$. They both vary with a low frequency from time 0 to time 4000, and from time 12000 to 15000 and high frequency from time 4000 to 12000.

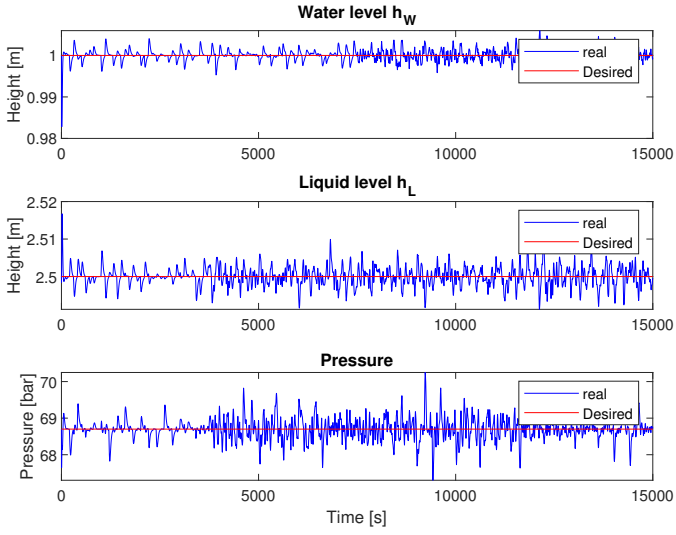


Figure 5.2: Training set data for internal states

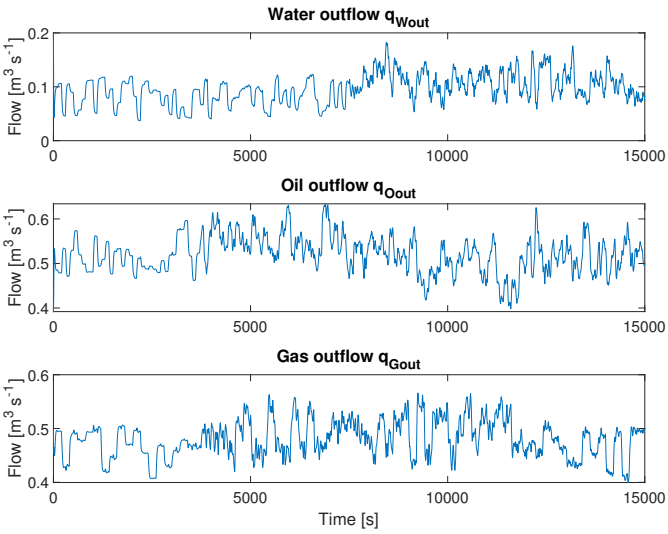


Figure 5.3: Training set data for outflow

5.2 Building the Network

The first step in building the ESN is as shown in Figure 3.5, to find a size for the ESN reservoir taking into consideration training time and error. A range search will be used to find the reservoir size. The training method used for these simulation was Ridge regression with $\lambda = 0.8$, for being faster and robust. The training set used is described in Section 5.1. The weight distribution for the reservoir was chosen such that the spectral radius ranged between 0.9 and 1 for all the simulation with different reservoir sizes. This is the range which is likely to yield good performance and satisfy the ESP.

To measure the performance a form of measurement of the error is needed. Error is calculated by taking the absolute value of the difference between estimated and real values. This is calculated for on a test set generated exactly like the training set with new random numbers. The first 200 seconds are used to ”warm up” and are therfore not included. Since the system has multiple outputs with different ranges the built-in Matlab

function resize(v) is used. It creates a new matrix or vector and scales it such that all the values are transformed to be between 0 and 1. One concern using this method is that it can look like there is an equal amount of error for different variables, when in fact one is superior. A visual inspection of the plots is therefore also performed to verify the results. To get a fair comparison between multiple simulations using the resize function, it is used in a way that for each of the values the maximum error among all simulation are 1 and the minimum among all is zero. If the resize function was used individually for each simulation it would not be a true comparison as the maximum error for one simulation and another would both be equal to one, when in fact they could be different. All simulations must therefore be done before using the resize function. After the resizing one can for each simulation add together the resized errors for all values over time. The result can be seen in Figure 5.4. For the different simulations the same data was used. For the simulations for the different reservoirs sizes the training time was measured. The result can be seen in Figure 5.5. The time to train is considered short for all the simulations. The formula for the total error, which is used to analyse performance then is:

$$Total\ error = \sum_{i=M/2}^{M} resize_{q_{G,in}}(|q_{Gin,i} - \widehat{q}_{Gin,i}|) + resize_{q_{L,in}}(|q_{Lin,i} - \widehat{q}_{Lin}, i|) + resize_{\gamma}(|\gamma_i - \widehat{\gamma}_i|)$$

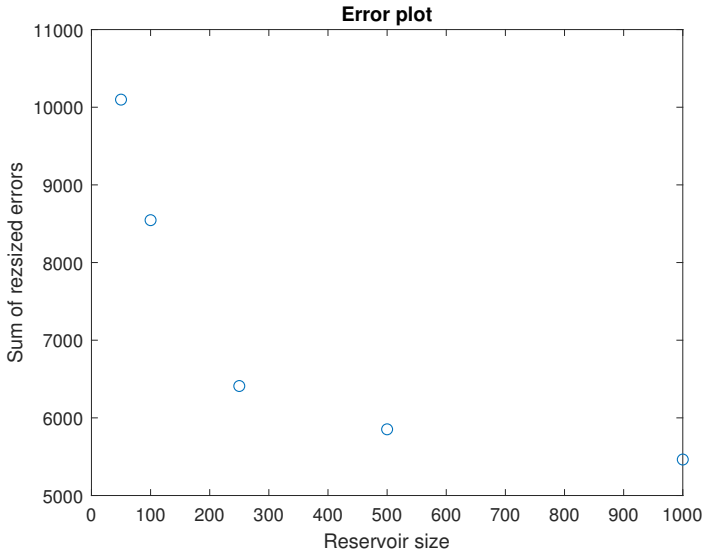


Figure 5.4: Sum of error for different simulations with varying reservoir size.

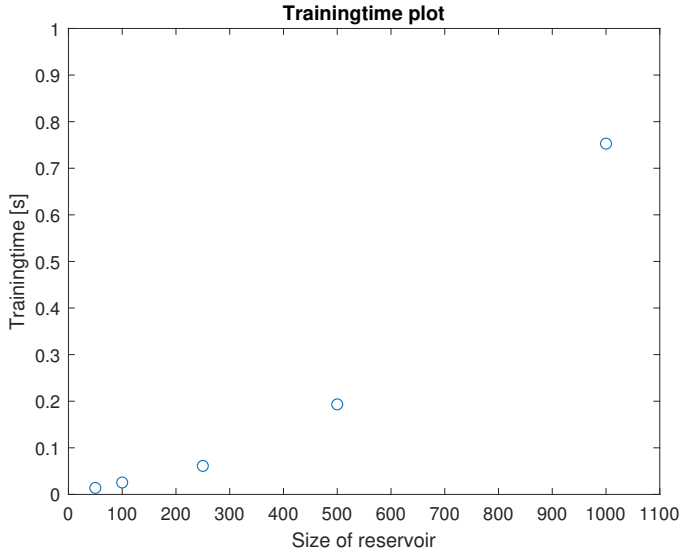


Figure 5.5: Training time for different simulations with varying reservoir size.

The size of the reservoir was chosen to be 500, because there was a very small

improvement from 500 to 1000, see Fig. 5.4. The training time was also small for all the reservoir sizes. It can be seen from Figure 5.5 that it increases exponentially with reservoir size. Training time for reservoir size 500 was 0.22 seconds. The distribution range of the reservoir was 0.075, which gave a spectral radius $\rho = 1.0535$, and it also satisfied the ESP necessary condition Eq. (3.7) with $effective - spectral - radius = 0.9818 < 1$, but failed the sufficient condition test, Eq. (3.5) with the result 1.8643 being greater than one. The analyse of ESP was therfore inconclusive, it could not be proven if the ESN has ESP. ESP is more detailed in Section 3.3.8.

A range search was followed to decide the leak rate α. In the same way the total error sum was used for selecting reservoir size but instead now varying the leak rate. As the sum of errors are scaled to fit between 0 and the results in Figures 5.4 and 5.6 can not be compared directly.

The error was low for all simulations with leak rates from 0.1 to 1, but some small differences, as depicted in Figure 5.6. The best performance seems to be in range from from 0.5 to 1. A more detailed search was then performed by testing leak rates from 0.5 to 1 with a step of 0.05. The search was done ten times with different inputs to the ESN. As depicted in Figure 5.7 the best leak rate is not the same for all inputs to the ESN. The error from the different leak rates were then averaged and the result is shown Figure 5.8. Here 0.6 showed the best averaged performance and was therefore chosen for next tasks. However there are some uncertainty to this value. One possibility to why this was difficult is that the error was low for all the leak rates. This combined with the use of resize function which can make low errors significant.

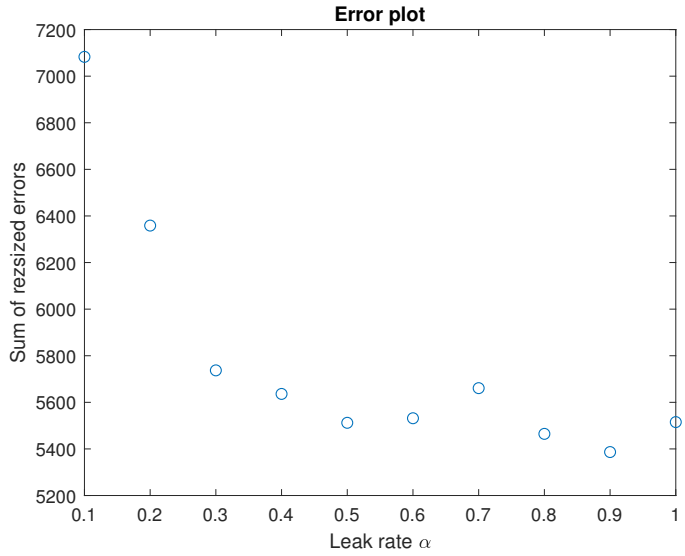


Figure 5.6: Sum of error for different simulations with varying leak rate

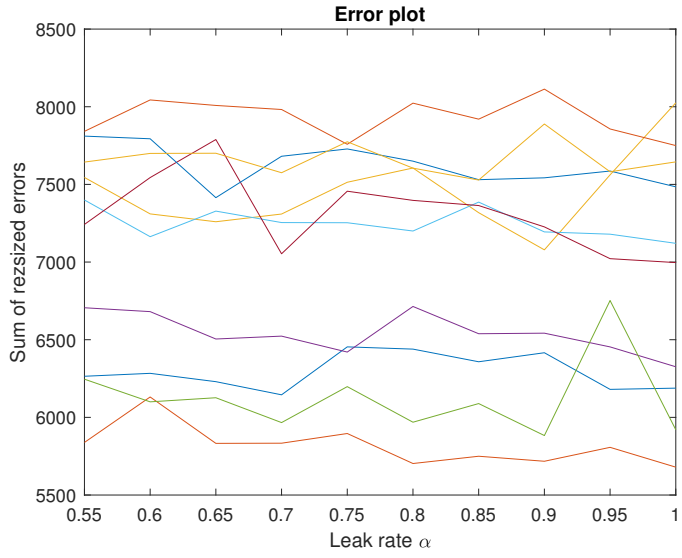


Figure 5.7: Sum of error for different simulations with varying leak rate. Each line is a test of leak rate one data set. The inflow was selected randomly for all data sets.

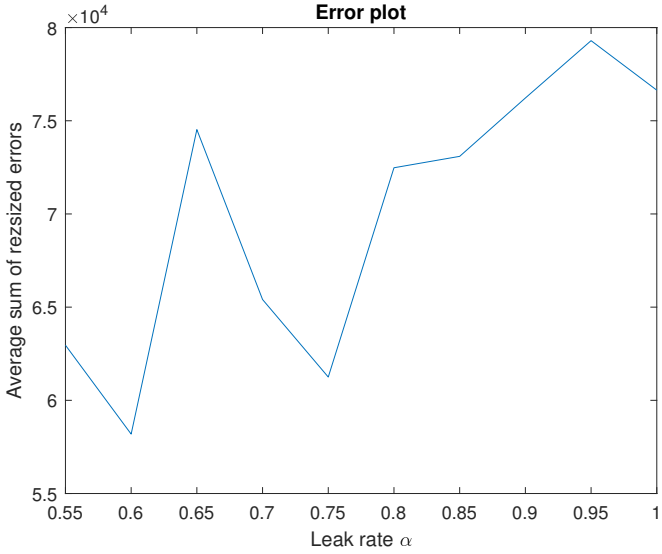


Figure 5.8: Averaged sum of resized error over ten simulations with varying leak rate.

From Figure 5.9 one can note a direct relation between the input signal $q_{G,in}$ into the tank and the output signal q_{Gout}. This observation tells us that there likely is a linear relation from q_{Gout} to $q_{G,in}$. It is also likely for q_{Gout} and q_{Wout} to $q_{L,in}$ and γ but it is not as clear to see. This means that it is beneficial to include the linear-term for the output matrix as in Equation (3.3).

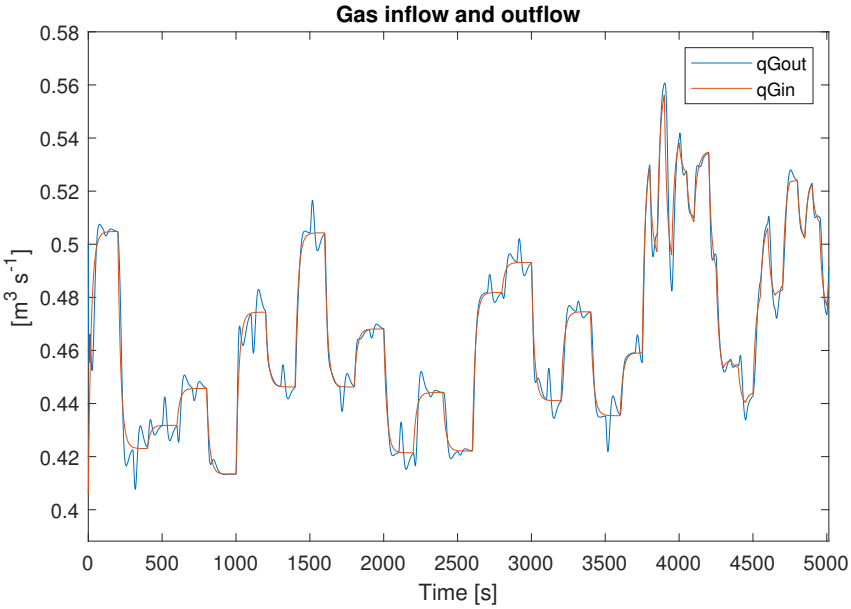


Figure 5.9: Similarities between the Gas outflow (input to ESN) and gas inflow (output from ESN).

5.3 Test with Varying Load

A new simulation with with same model but newly generated random input was used to form the test set. The experiment consisted of simulation of 1600 seconds. The states and results can be seen in Figures 5.10 and 5.11. One can observe the ”warm up” phase for the first 100 seconds. The initial states of ESN were set to zero and therefore a period of time was needed to ”warm up” the network. Thereafter the ESN estimation is good. There are no continuous biases and only small fluctuations. This means that for the scenario of keeping all the states stable the ESN provides a good model.

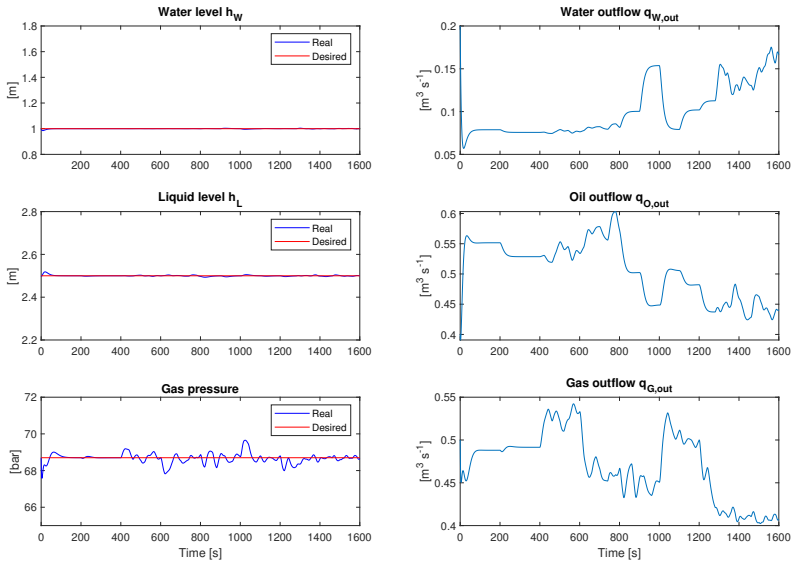


Figure 5.10: Test data set.

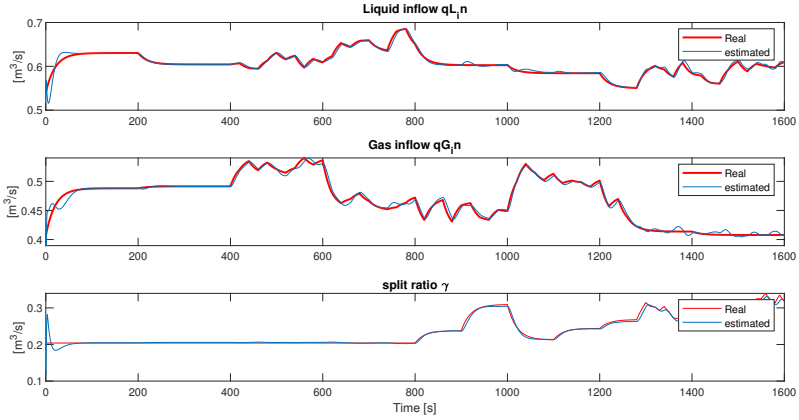


Figure 5.11: Results on test set.

5.4 Results with Varying Water Level

The introduction of variation in the water level brings additional challenges. There is a correlation between the water level adjusting to a new desired level and outflows. This adds more nonlinearities. The controller only knows the internal states and changes in the outflows. This means that it is hard for the network to know whether the change in outflow is due to a change in reference value or the inflow. The variation for the inflow is the same as before in Section 5.1.

In Figure 5.12 one can see that the water level is varying. The desired liquid level and pressure are kept constant but have greater variation than in Figure 5.2. The variation of the reference value for the water level was created with a random number generator as well as filter. The random numbers where generated with a uniform distribution between 1 and 1.8. The filter was added because without it the control variables could be predictable by always make a jump and then hold a value, an example of this is $q_{W,out}$ in Figure 5.22. Introducing the filter, made the control variables vary more slowly and therefore more variations in the training set.

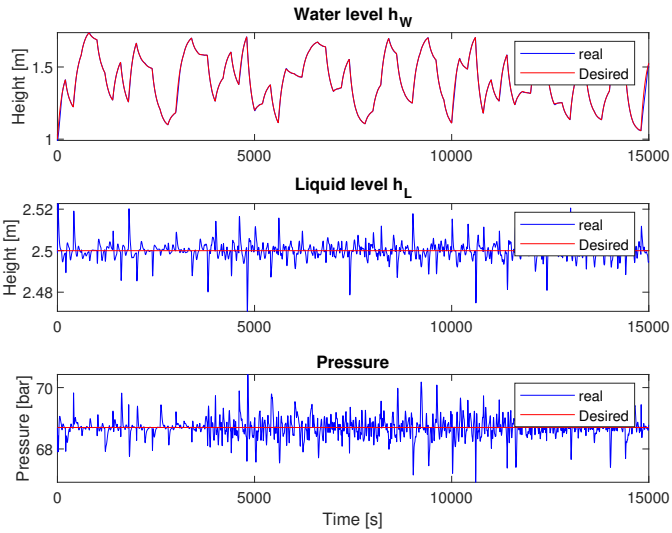


Figure 5.12: Training set data for internal states.

The test set shown in Figure 5.13 was created running a new simulation with new inputs. The results in Figure 5.14 show that the ESN is able to estimate $q_{G,in}$ and $q_{L,in}$ while giving poor results for γ. It seems to be more difficult to estimate γ when the water level h_w is changing more rapidly, like from time 1400 to 1600 seconds. ESN is struggling to "see" the difference in variation imposed on the outflows imposed by varying water level and inflow.

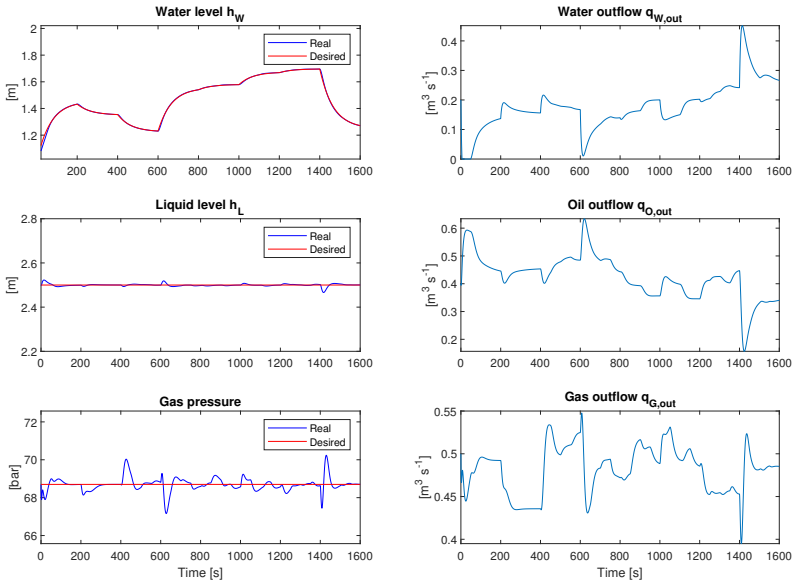


Figure 5.13: Test data set

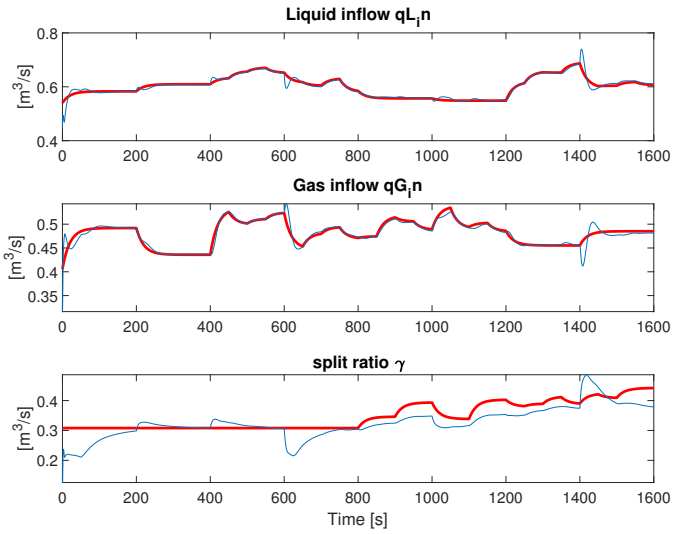


Figure 5.14: Results on the test set obtained with the ESN. Red is real value and blue is estimated.

In one effort to improve the performance a timedelay was included to the ESN input. The timedelay is included by adding six extra inputs to the ESN. These new six inputs are the inputs from a previous timestep $[t - \tau_d]$. To decide how far back to go, the test was run with multiple values for the timedelay τ_d which represents how far back in time the input should go. The result can be seen in Figure 5.15. It is difficult to visually see how the different timedelays compare so a new error plot was introduced in Figure 5.16a. The error plot in Figure 5.16a only includes the error estimating γ. This is because γ had the largest error. The result, Fig. 5.16b shows that the best performance was with $\tau_d = 8$ seconds. From now this value will be used. The timedelay for the inflow to affect the internal states of the system is not included in the model described in Section 4.1. Yet the test to include the previous inputs into the ESN was showing better results as it gave the ESN more data to work with. For a real world system it might be advantageous to include multiple inputs with different timedelays and maybe even also future inputs. This because the effect from the load enters until it affects pressure, water level and liquid level do not happen instantaneous [4].

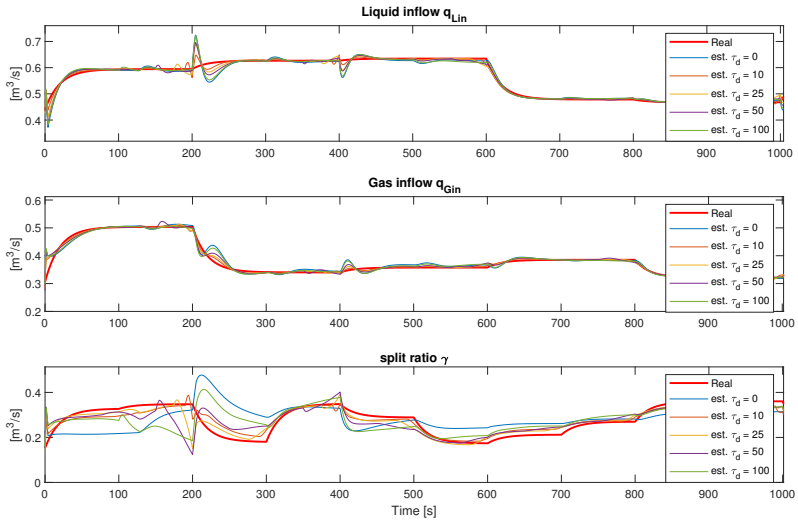
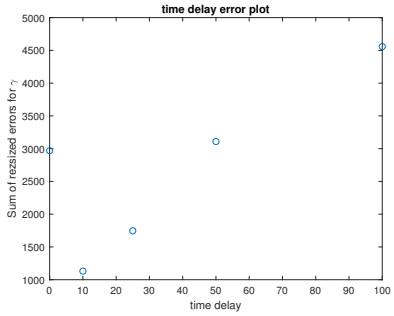
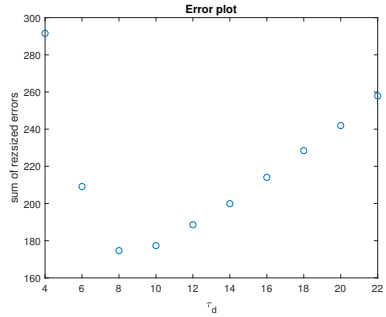


Figure 5.15: Simulations with different timedelays



(a) Time delay wide line search



(b) Time delay narrow line search

Further steps to try to improve the performance included references values to the controller as input to the ESN. This was done because the reference is input to the system as a whole. From Figure 5.18 one can see visually a small improvement in the sum of errors from the blue to brown line. The only difference is that for the brown line the controller reference was included as an input to the ESN.

Lastly there was a effort to improve the performance by tuning the parameter λ.

This parameter is a penalty on high weights during training using Ridge. A range search was done to fin the best value for λ. The results are shown in Figure 5.17. To switch the parameter λ from 0.8 to 0.3 gave a better performance. This can be seen in Figure 5.18 by following the black line. For more detail on the parameter λ and training with Ridge regression please refer to Section 3.4.2.

There was also some failed attempts for improving the performance of the ESN. To include feedback as discussed in Section 3.3.5 led to a worse deterioration in performance. Also using a sparse matrix as described in Section 3.3.4 was tried. For low sparsity there was no significant change of the response, but for strong sparsity it yielded a worse performance.

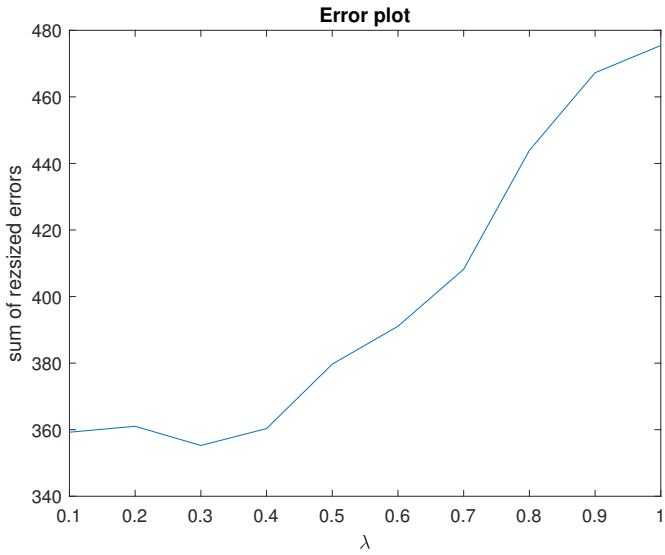


Figure 5.17: Line search for λ

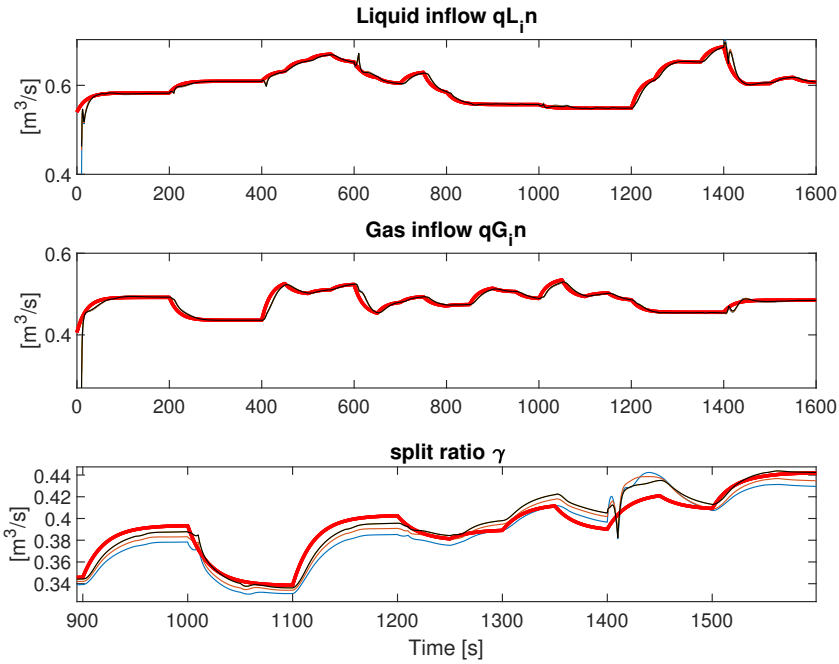


Figure 5.18: Red line is the real value. Blue is without reference as input and with $\lambda = 0.8$. Brown is with reference as input and with $\lambda = 0.8$. Black is with reference as input and with $\lambda = 0.3$.

The result after including the improvements described in this section the results can be seen in as the black line in Figure 5.19. This shows that the network can be able to estimate the load with a varying water level. The improvement that had the most impact was to include the time delay.

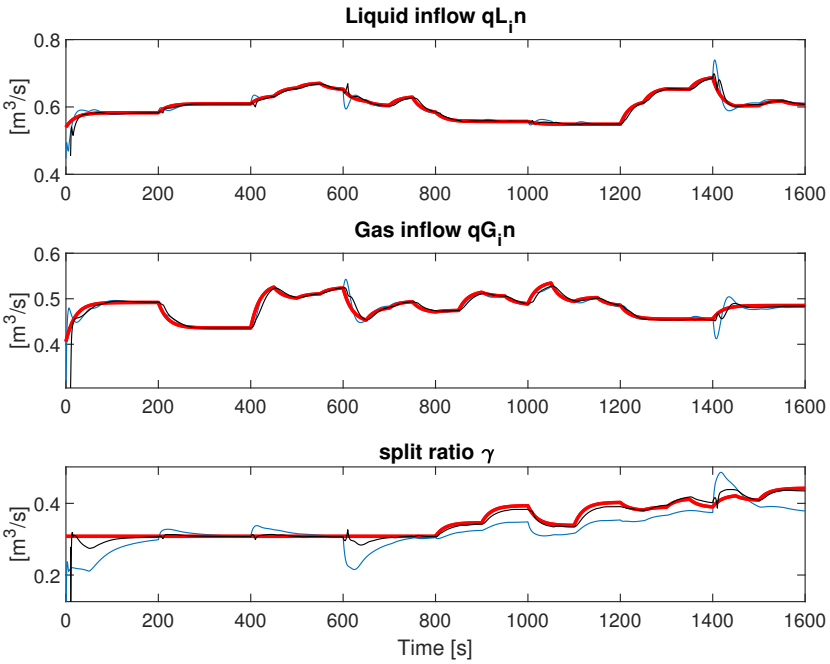


Figure 5.19: Red line is the real value. Blue is without reference as input and with $\lambda = 0.8$ and $\tau_d = 0$. Black is with reference as input and with $\lambda = 0.3$ and $\tau_d = 10$.

5.5 Results with Noise

Noise was added to the state variables on the liquid level, water level, and pressure. This gives a more challenging task as noisy signals adds more nonlinearities and variation to the signals. White noise generators from the Simulink library were used. It can be a representation of noisy measurements from the real world. The noisy signals were not sent to the controller. From Figure 5.20 one can see that only the internal states and not the outflows are noisy. The training set also included these noisy measurements. A desired behavior would be for the ESN to filter out this noise and not provide noisy estimates. The ESN achieved this successfully, as depicted in Figure 5.21.

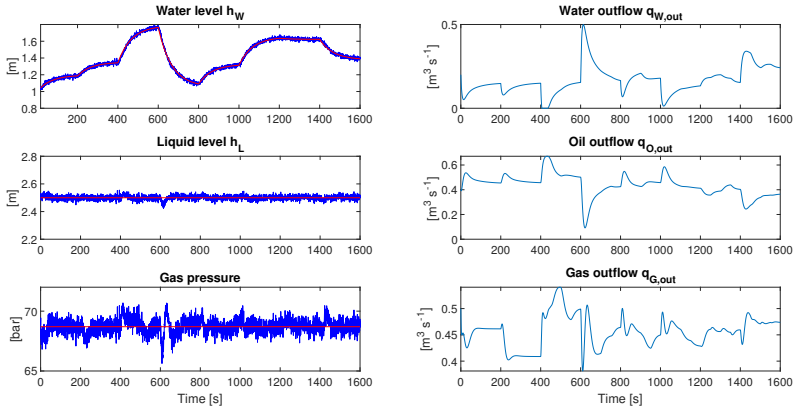


Figure 5.20: Test data with noise added to the measurments of state variables

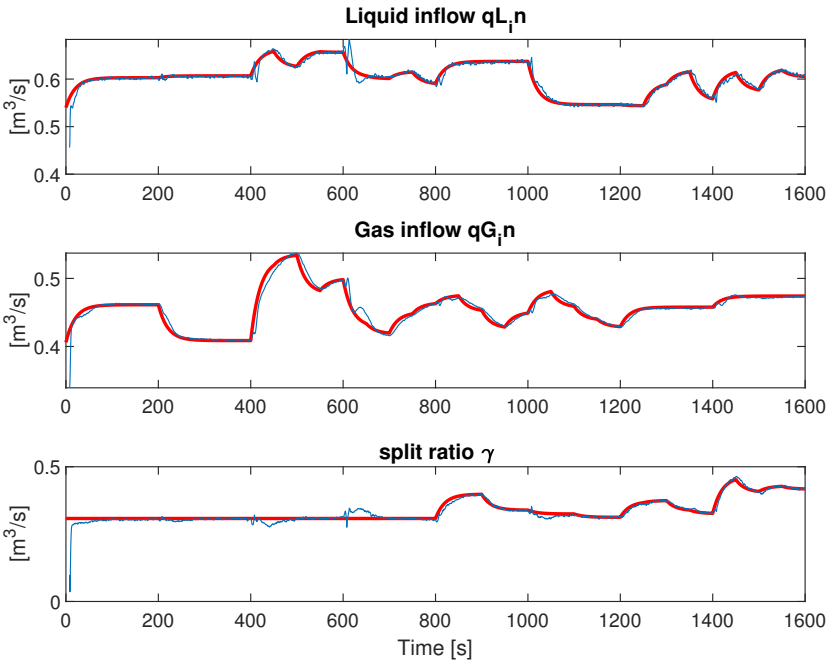


Figure 5.21: Estimation with noisy states. Red is the real value. Blue is the ESN estimation

5.6 Comparison of EKF and ESN

Another interesting approach to estimating load is the use of Extended Kalman Filter (EKF) as discussed in Section 4.4. This chapter will test an ESN and EKF observer on the same test set. The test data used here have some smaller changes from previously. There is no filter on controller reference or inflow signals. This makes the change instantaneous. Noise was included for gas and liquid inflow. The training set for the ESN was without these changes and the test data can be seen in Figure 5.22. The results can be seen in Figure 5.23. All estimations from the ESN (red) has some error at the time when the water level is changing. For the split ratio γ it holds a constant error for about the same time the water level is changing, like in the timespan from 200 to 280 seconds. The error from change of water level also creates small error for the ESN estimation of liquid and gas inflow. These are shorter and clearly visible in Figure 5.23.

EKF estimation of gas inflow also has an error after 200 seconds. This is likely a result of the step in liquid inflow at 400 seconds as the step in water level also occurs at 200 seconds. And there is no error at 200 seconds. EKF has a constant bias for the split ratio γ. EKF is somewhat slower than ESN in reaching the steps of liquid and gas inflow. This behavior is due to the structure of the EKF estimation wich adjusts gains gradually by solving the Ricatti equation.

EKF (blue) is not able to successfully estimate the split ratio γ. From the paper that described the details of the EKF observer [3] it is noted that: "The split factor cannot reach its nominal value since the static droplet calculations are included in the plant model, but not in the observer model, hence a plant-model mismatch is the consequence". Today it is difficult to obtain measurements for droplet distributions and it was therefore assumed unknown for the EKF estimation. The ESN estimation does not have this bias. All parameters were the same for its training and test set. A mismatch here can lead to biases as well. This is shown in fig. 5.24. There a change is made from the trainingset to the testset. The inital distribution of oil in water $1 - \phi_o$ and water in oil $1 - \phi_w$ are both changed from 0.3 to 0.1. The result from this is an almost identical response except for

ESN which now has a bias when estimating γ. ESN showed some robustness in this test by handling the small variation between test and training set, no filter on reference, and inflow and noise on the inflow.

By comparison from this simulation the ESN providing a better overall result than EKF. It has some larger fluctuations at steps but no bias if it is trained with correct data. ESN also lacks the interperetability EKF has. This is because ESN is data driven while EKF is based on models.

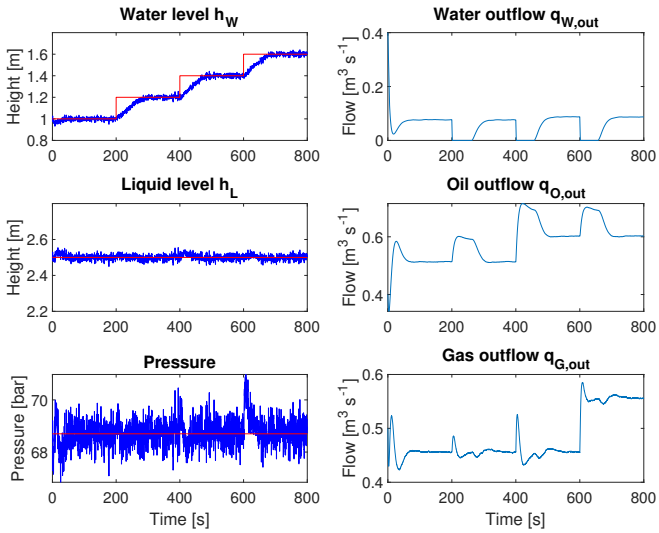


Figure 5.22: Test data, Red is controller reference. Blue is measurements

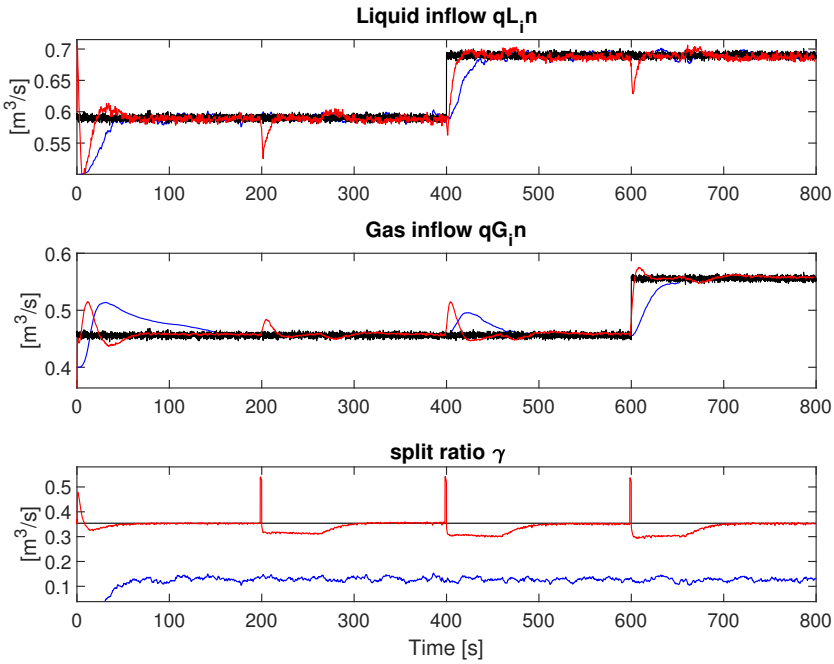


Figure 5.23: Results. Black is real value. Blue is EKF. Red is ESN

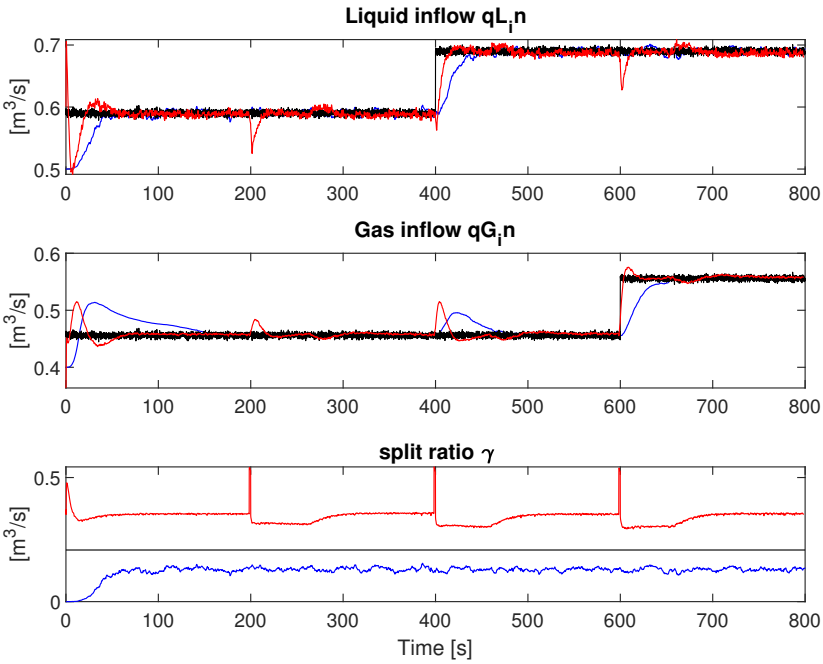


Figure 5.24: Results with different parameters for ESN training and testset. Black is real value. Blue is EKF. Red is ESN

Chapter 6

Conclusion

In this work, Echo State Networks (ESNs) where used for Virtual Flow Metering (VFM), estimating the flowrate for the different phases of water, gas and oil for a wellstream entering a separation tank. This is a black-box modeling approach, creating a model for the system using no prior knowledge. The main goal was to test the effectiveness of such models induced by ESNs.

The data used to train and test the ESN comes from on a three-phase gravity separator model obtained from [4]. The tank model included a controller for the pressure, water and liquid level. The control variables were the outflows from the water, oil and gas phase. The first test had a constant reference signal into the controller, in which the ESN yielded a good performance and was successfully able to reproduce the signals for water, oil and gas flow entering the tank. Such experimental set-up renders the model difficult to tune. Varying some hyperparameters such as the leak rate did little to change the response as the response was almost perfect either way. This became clear in the second test when the reference value for the controller was varied. This is a more complex system as multiple factors now affect the states. Obtaining a good response with the ESN was somewhat difficult initially. This led to several improvements of the ESN. These

improvements provided the ESN with more data by adding the reference values and data from a previous timestep as input signals. Then the ESN was able to successfully estimate the signals for water, oil and gas flow entering the tank. The third test introduced noise to the states. ESN successfully rejected the noise. When comparing the estimation from ESN and EKF, ESN provided better results. EKF used longer time than ESN to reach the correct value for q_{Gin}, q_{Lin}. When estimating the split rate γ, EKF was not able to reach the correct value.

The biggest challenge for using ESN as VFM is the availability of a suitable training set. This is a problem that affects neural networks and other data-driven methods in general. Most significantly is the lack of good data for the three-phase well stream because it is difficult to measure. Without such measurement it is not possible to train an ESN to learn the model. It can be a challenge to have the same droplet distribution for the training data as the real system.

A future improvement could be to apply online learning to the ESN. This could potentially counter challenges like drift in instruments or other slow variation in parameters and which would render the ESN more robust.

One disadvantage using Figure 3.5 to implement ESN, can be that too large reservoirs are chosen. This is a result of the reservoir size being the first parameter being tuned. This means that the ESN gave a worse performance than what it would do with better parameters. This happened for the case example from Section 3.3.2, because it was a simple system and a reservoir size of 500 is quite large.

As ESNs have been shown to successfully learn the inverse model for the separator tank. Another task could be to use the ESN to learn the non-inverse tank model. Here the input to the ESN would be the multiphase inflow to the tank q_{Gin}, q_{Lin} and γ and the outflow from the different phases of the tank $q_{G,out}$, $q_{O,out}$ and $q_{W,out}$. The output from the ESN would then be the pressure p, water level h_w and liquid level h_L. This model could be used as a redundant soft estimator for the states or as a model to predict future states in Model Predictive Control (MPC).

Bibliography

[1] N. Andrianov. A machine learning approach for virtual flow metering and forecasting. *Proc. of 3rd IFAC Workshop on Automatic Control in Offshore Oil and Gas Production,*, 2, 2018.

[2] 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.

[3] C. Backi and S. Skogestad. Virtual inflow estimation with simplified tuning using cascaded and kalman-like least squares observers. *Computer Aided Chemical Engineering*, 43, 06 2018.

[4] C. J. Backi, B. A. Grimes, and S. Skogestad. A control- and estimation-oriented gravity separator model for oil and gas applications based upon first-principles. *Industrial & Engineering Chemistry Research*, 57(21):7201–7217, May 2018.

[5] T. A. N. H. Bjarne Foss. *Merging Optimization and Control*. 2016.

[6] A. CASTROUNIS. Artificial intelligence, deep learning, and neural networks explained, 2018.

[7] Clockbackward. Ordinary least squares linear regression: Flaws, problems and pitfalls, 2009.

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

[9] P. Durdevic, L. Hansen, C. Mai, S. Pedersen, and Z. Yang. Cost-effective ert technique for oil-in-water measurement for offshore hydrocyclone installations. *IFAC-PapersOnLine*, 48(6):147 – 153, 2015. 2nd IFAC Workshop on Automatic Control in Offshore Oil and Gas Production OOGP 2015.

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

[11] A. Hallanger, F. Soenstaboe, and T. Knutsen. *SPE-36644-MS*, chapter A Simulation Model for Three-Phase Gravity Separators, page 12. Society of Petroleum Engineers, Denver, Colorado, 1996.

[12] 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.

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

[14] 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.

[15] B. Kim, R. Khanna, and O. O. Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2280–2288. Curran Associates, Inc., 2016.

[16] A. P. . S. Laleh, W. Y. . Monnery, and W. D. Computational fluid dynamics-based study of an oilfield separator - part i: A realistic simulation. *Oil Gas Facil. 2012*, pages 57–68, 2012.

[17] X. Lin, Z. Yang, and Y. Song. Short-term stock price prediction based on echo state networks. *Expert Systems with Applications*, 36(3, Part 2):7313 – 7317, 2009.

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

[19] Mathwaorks. Start stopwatch timer, 2018.

[20] R. McClimans, O. T.; Fantoft. Status and new developments in subsea processing. *Offshore Technology Conference. Houston, TX, USA*, page 731 740, 2006.

[21] M. Meribout, M. Habli, A. Al-Naamany, and K. Al-Busaidi. A new ultrasonic-based device for accurate measurement of oil, emulsion, and water levels in oil tanks. volume 3, pages 1942 – 1947 Vol.3, 06 2004.

[22] C. Molnar. *Interpretable Machine Learning*. 2019.

[23] A. S. Morris and R. Langari. Chapter 15 - pressure measurement. In A. S. Morris and R. Langari, editors, *Measurement and Instrumentation (Second Edition)*, pages 463 – 491. Academic Press, Boston, second edition edition, 2016.

[24] J. Richardson and W. Zaki. The sedimentation of a suspension of uniform spheres under conditions of viscous flow. *Chemical Engineering Science*, 3(2):65 – 73, 1954.

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

[26] N. van der Tuuk, G. SRLAND, and J. Sjblom. Methods for droplet size distribution determination of water-in- oil emulsions using low-field nmr. *Diffusion Fundamentals*, 9, 01 2009.