Use of Neural Network model for State Estimation

A Dissertation
Submitted in partial fulfillment of
the requirements for the degree of
Master of Technology
by

Vinayak Agrawal
(Roll No. 193020056)

Supervisor:
Prof. Mani Bhushan

Department of Chemical Engineering
Indian Institute of Technology Bombay
Mumbai 400076 (India)

4 July 2021

Acceptance Certificate

Department of Chemical Engineering

Indian Institute of Technology, Bombay

The dissertation entitled “Use of Neural Network model for State Estimation” submitted by Vinayak Agrawal (Roll No. 193020056) may be accepted for being evaluated.

Mani B

Date: 4 July 2021

Prof. Mani Bhushan

Approval Sheet

This dissertation entitled "Use of Neural Network model for State Estimation" by Vinayak Agrawal is approved for the degree of Master of Technology.

Digital Signature
Sharad Bhartiya (i02150)
08-Jul-21 06:21:14 PM

Digital Signature
Sachin C Patwardhan (i01116)
08-Jul-21 06:16:31 PM

Examiners

Digital Signature
Mani Bhushan (i05135)
08-Jul-21 04:49:41 PM

Supervisor

Digital Signature
Sachin C Patwardhan (i01116)
08-Jul-21 06:17:30 PM

Chairman

Date: 4th July 2021

Place: Mumbai

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I declare that I have properly and accurately acknowledged all sources used in the production of this report. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Vinayak Agrawal

Date: 4 July 2021

(Roll No. 193020056)

Abstract

Many advanced process operations task in process industry, such as state estimation and advanced control, need to be performed on a real-time basis. Often these strategies employ a model of the process in their decision making. For many systems, a detailed first principles model can be too cumbersome to be used by these approaches as it can make these approaches infeasible for online implementation.

With increasing availability of either historical plant data, or data from elaborate plant simulators, it is possible to train a data driven model which is more suitable for advanced control or estimation applications. In the current thesis, we investigate such an approach. In particular, we propose to use Recurrent Neural Networks (RNN) to model complex dynamical systems which are governed by nonlinear differential equations. The RNNs represent the relationship of past states and inputs of the process to the current states in an algebraic form. To facilitate the use of RNN model for state estimation, the RNN representation is formulated in a nonlinear, discrete time state-space form. The RNN model is then used by an Extended Kalman Filter (EKF) to estimate the system states.

This strategy was implemented and compared with existing conventional methods in the literature on several case studies to check the performance of the proposed work. Experimental validation of the proposed work has also been performed. Based on the results of the proposed method, it can be concluded that the neural networks can capture the system dynamics well and, when combined with EKF, result in satisfactory estimation performance with reduced computation cost and better accuracy.

Keywords: Recurrent neural network, Extended Kalman Filter, mechanistic model, reduce computation time.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The chemical plants consist of a highly complex process. Thus, to control the whole process, these plants are usually highly instrumented and have many sensors, collecting measured output variables for process control and monitoring. In the past few decades, researchers have come up with a predictive model which uses a large amount of data, and these models are called soft sensors. Using these soft sensors, we try to predict the process variable, which is essential for the overall plant operations and is measured either by low sampling rate or offline methods.

The soft sensor is based on a mathematical model of the process. As the industrial processes are quite complex, the problem of finding an exact or approximate model for the dynamical systems frequently occurs in engineering applications Ambaw (2005). The mechanistic model generated for these complex systems often ends up with substantial computational cost due to complex coupled differential equations and thus not feasible to be used online. Therefore data-driven techniques can be used, which uses the measured process variables to model the plant dynamics. Furthermore, this model generated can be used instead of the mechanistic model generally used in the soft sensor. With the upcoming advancement in AI and computational power, we can use a deep neural network to generate such a model. Once this model is determined, it can be clubbed with various algorithms for estimating the states of the plant efficiently. Although linear models can provide acceptable results for many process systems, these are unable to give good results as per the expectation if significant non-linearities are present in the system. A model that reflects the nonlinear relationship between the output and the input variables must be taken for such systems. Various studies have shown that artificial neural networks (ANNs) may provide a generic, nonlinear solution to such process systems. With the rapid advancement in the computational power available now, ANNs are the most efficient and economical technique to model plant dynamics. Applying highly efficient estimators like

Moving Horizon Estimation (MHE), Extended Kalman Filter (EKF) Jazwinski (2007) using the ANN model will solve the major problem faced by the chemical industries. Thus solving such a problem is the strong motivation to study more about the application of ANNs accompanied with various state estimation techniques.

Most of the chemical process industry consists of a large number of states. A mechanistic model for these may be difficult, time-consuming, and expensive to derive with added computation cost. So, a universal technique for modeling such complex systems is required to replace such mechanistic models and be used for state estimation. A recurrent neural network can achieve this as it has capabilities of dynamic mapping of the system. This model of the system can now be used in the prediction step of state estimators like EKF. Further, all the parameters derived from the mechanistic model can directly be derived from this RNN model. This RNN model can now be clubbed with any state estimator to estimate the states.

1.1 Report Organization

A study of a different way to generate a model of the system instead of a physics-based approach and combining it with a state estimator has been done from existing literature in Chapter 2. Background work that is EKF algorithm and modeling of the system using artificial neural network specifically RNN has been discussed in Chapter 3. The proposed method that is combining RNN with EKF and its various permutations has been described in Chapter 4. A comparative analysis between existing methods used for state estimation with the proposed methods with experimental validation with the help of few case studies has been done in Chapter 5. Conclusion of the results from the proposed method has been reported in Chapter 6 of this report.

Chapter 2

Literature Survey

There is some existing method that can be used to model the system to reduce the calculation complexity, which has been discussed in chapter 3. In the existing literature, two possible ways to approximate system dynamics have been found which can be used by the soft sensor to solve the state estimation problem Slišković *et al.* (2011).

- First principle modelling (theoretical modelling)
- Data-based modelling (data-driven or empirical modelling)

2.1 First principle modelling

First principle modeling is also referred to by many other names like white-box models, phenomenological, or mechanistic modeling. Developing the first principle model involves a rigorous approach and is based on:

- Material and energy balance of the whole process plant
- Physical laws and constitutive relationships within the process variables
- Kinetic and thermodynamic models supporting the process plant
- Heat and mass transfer models

2.2 Data-based modelling

Data-based modeling is done with the help of the measured variable database. There are two ways to collect the data for the data-driven models:

1. **Experimental methods :**
   In which specific experiments are carried out to generate data. The data generated from these are considered very optimal for the modeling as it would have a minimal amount of error and noise, which will help achieve better accuracy. However, the conduction of the experiment is sometimes not allowed, or it is costly.

2. **Plant data or historical data:**
   Chemical plants are equipped with many sensors that measure the various process variables at each time instant. This accumulation of the data over the years in the process database can be a valuable source of information for model building. Plant modeling using the process data is generally relatively cheap, but the data have various impurities which are to be taken care of before using it for the modeling. Hence, proper data preprocessing is to be done on the data set before using it for modeling.

   Mechanistic models are often inefficient, expensive, and time-consuming. They can also be computationally heavy, requiring iterative calculations that are unsuitable for online use. Hence, switching to efficient data-based modeling methods will help in rectifying this problem. Multilayer networks have proved highly successful in pattern recognition problem Kumpati *et al.* (1990). Thus, these can solve the problem of modeling complex chemical plants, which are challenging to model using conventional ways. There have been various papers supporting this idea. Srinivasan and Prakash (2006) discussed a systematic approach to design a non-linear observer to estimate the state vector of a non-linear dynamic system. In the paper, the author took two neural networks for solving the state estimation problem. The first neural network has been used for model prediction, and the second neural network generates filtered values of the states. Hence in the above paper, the state estimation problem has been solved purely using neural networks.Moreover, a case study has also been reported in the article to check the efficacy of the proposed work. In Yadaiah and Sowmya (2006), a neural network-based state estimator for a general class of non-linear dynamic systems is presented. The proposed state estimator uses cascading of a recurrent neural network structure (RNN) that learns the dynamical system's internal behavior and a feedforward neural network (FFNN) which learns the measuring relations of the system from the input-output data through prediction error minimization. A dynamic learning algorithm for training the recurrent neural network has been developed. A similar analogy like above is also shown in Ku and Lee (1995) and many more. Hence, using neural networks for system identification

has been explored and recognized in many research papers like above and thus can be used for modeling the system.

Among all the neural networks, recurrent neural networks are capable of doing dynamic mapping of the system due to their unique property of using a window of past values for the prediction. Now, this RNN model can be used with an estimator to solve the state estimation problem. Various combinations of such work are explored in the literature. As shown in Srinivasan and Prakash (2006) and Yadaiah and Sowmya (2006), one way for solving the state estimation problem is by purely driving the model and estimator using neural networks. Second, a combination of the neural network model with the traditional state estimator like Kalman filter, Extended Kalman Filter, etc., were also reported in the existing literature. For model predictions both, a standard feed-forward and RNN with an estimator can be used, and both of these methods have been extensively discussed in the literature. A few of the papers discussing the idea of RNN as the model with conventional estimators have been mentioned below in the report. Habtom and Litz (1997) paper addressed the problem of estimating the states of the system using a recurrent neural network as the model and EKF as the state estimator. As the EKF requires state-space representation of the model, hence methodology for converting the RNN into a state-space model has been reported in this paper. Using this state-space model, pertinent parameters like Φ and Γ of the system were obtained, which were used by the estimator to find filtered values of the states. Moreover, case studies were also shown in the paper to test the efficacy of the proposed work. Further, many articles have mentioned solving the problem of converting the RNN model into state-space representation like Stubberud *et al.* (1995) and Parlos *et al.* (2001).

Hence motivated by the literature presented above, various strategies have been proposed in Chapter 4 to solve the state estimation problem. The foundation of all the strategies revolves around the idea of modeling the system by recurrent neural network and estimating states by EKF, which is purely driven by RNN (RNNEKF). Using such strategies helps in massive cost reduction, which is incurred in conventional mechanistic modeling. Moreover, there is a reduction in the computational time too which have been reported using application of the proposed work on few case studies reported in chapter 5.

Chapter 3

Background Work

In this chapter implementation method of Artificial Neural Network for the modelling of the chemical process system and EKF for the estimation of states will be discussed as available in numerical analysis literature.

3.1 Artificial Neural Network

An artificial neural network(ANN) is one of the most popular techniques for approximating complex nonlinear systems. However, in ANN, the ratio of data required to approximate the system is more than any other modeling technique to achieve successful parameter estimation. But, because of the properties like universal approximation theorem, ease in handling multi-input output systems, and high computational power available today, neural networks are one of the most efficient ways to model these complex systems.

Universal approximation theorem claims that the standard multilayer feedforward neural network with an activation function are universal approximators. It is not the type of activation function used but rather the multilayer feed-forward architecture which helps in approximating any function. Output units in the architecture are always assumed to be linear. The theorem in mathematical terms:

Theorem 3.1.1 (universal approximation theorem):

Let $\varphi(.)$ *be an arbitrary activation function. Let* I_m *denote the the m-dimensional unit hypercube* $[0\ 1]^m$. *Let* $C(I_m)$ *be the space of real valued functions on* I_m . *Then for any function* $f \in C(I^m)$, *given any* $\varepsilon > 0$, *there exists an integer* $n \in N$, w_{ij} , b_i , $k_i \in R$, $i \in 1...n$, $j \in 1...m$:

As an approximation of function f(.) That is,

$$\| f - F(x_1, .., x_m) \| < \varepsilon \tag{3.1}$$

The working of most basic neural network, feed forward neural network(FNN) is as follows:

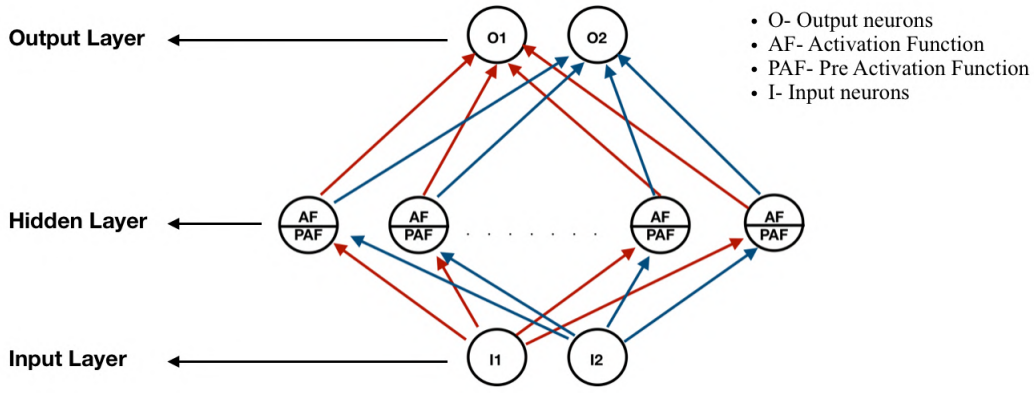


Figure 3.1: Neural network architecture used for plant modelling and state estimation

FNN starts with the input layer consisting of all the features associated with the system through which output can be calculated. In the chemical process, these usually represent the measured variables. The last layer of the network is known as the output layer. It gives the predictions calculated by the neural network according to the problem being solved (regression or classification). All the layers between the input and output layer are nomenclature as the hidden layers. All the arrows in Fig. **??** describe the weights associated with each neuron in the layer.
Each of these layers consist of 2 functions:
Pre Activation Function (PAF) - This comprises of simple linear transformation Eq (3.2) of the previous layer output to give output which is then fed to activation function.

$$a = x.W + b \tag{3.2}$$

Activation Function (AF) - The value obtained from (PAF) is fed to to this function. Which Finally gives an output for the layer. This function is usually nonlinear, which helps in approximation complex nonlinearities associated with the system. Though many types of neural networks can be used for system identification, recurrent neural networks (RNN) have shown excellent results for capturing plant dynamics.

3.1.1 Recurrent Neural Network

As the data generated in the chemical process industry falls under the category of time series data, thus most widely accepted techniques in Artificial Intelligence to capture the dynamics of such systems are RNN and its variation like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU).

The RNNs include cyclic connections of the neurons such that they can incorporate the past values for the prediction of the current time instant. Due to this, RNNs can do dynamic mapping of the process. The fig. 3.2 explains the working of the RNN:

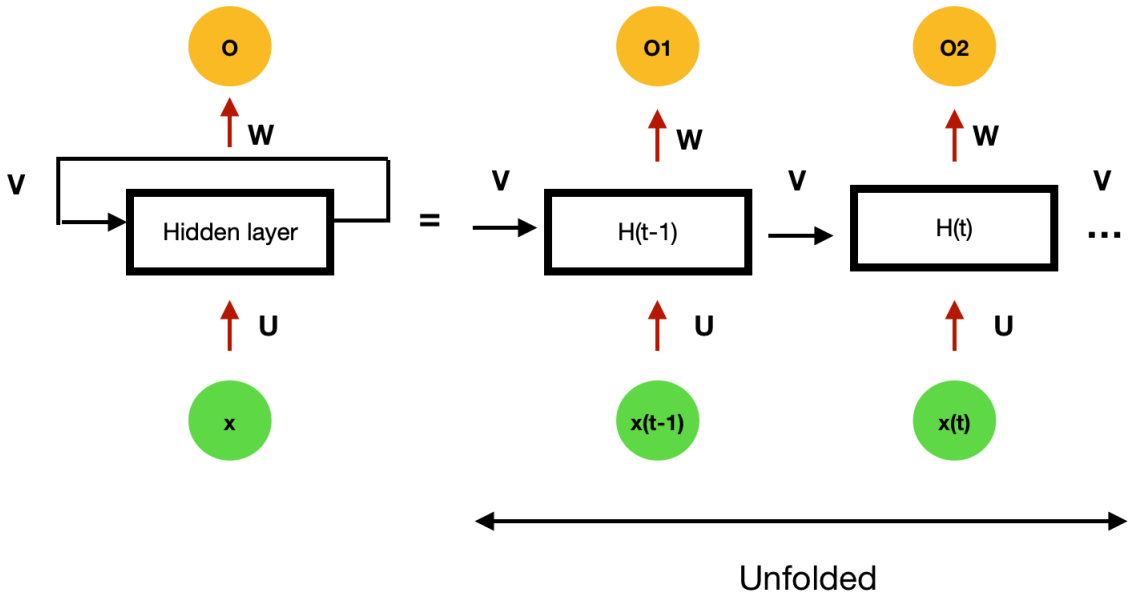


Figure 3.2: Recurrent Neural Network

In Fig. 3.2, RNN has one hidden layer where the x represents the neurons of the input layer. O represents the neurons of the output layer. And the middle layer is known as the hidden layer. Here, U, W, and V are the respective weights between the current input layer and the hidden layer, hidden layer, and output layer between past hidden layers and the current hidden layer. As we can see in the unfolded RNN, the prediction of each time instant is being calculated using the past hidden layer values. Hence it can do the dynamic mapping of the system.

Mathematically RNNs can be represented as:

$$h_t = \Phi(U^T x_t + V^t h_{t-1} + b_h) \tag{3.3}$$

$$\hat{y}_t = \Phi(W^T h_t + b_o) \tag{3.4}$$

Table 3.1: RNN parameters

Parameters	Definition
Φ	Activation function used in the network
b_h	Bias added in the hidden layer
b_o	Bias added in the output layer
U	Weight matrix between input and hidden layer
V	Weight matrix between past hidden layer value and current hidden layer
W	Weight matrix between hidden layer and output layer
$\hat{y}$	Output of the RNN
x_t	Input to RNN at current instant

Due to the properties like universal approximation theorem and dynamic mapping of the system, RNNs are best to model process which involves time series data. As chemical equipment falls into the category of time series data thus, RNNs can be used to model complex chemical processes. Many research papers have validated the use of RNNs for the same, few of them being Ku and Lee (1995), Shahriari-Kahkeshi and Askari (2011), etc.

3.2 Extended Kalman Filter

Consider a nonlinear process dynamics and nonlinear measurement equation can be represented as:

$$X(k+1) = F(X(k), u(k), d(k)) \tag{3.5}$$

$$With\ d(k) \sim \mathcal{N}(0, Q)$$

$$Y(k) = h(X(k)) + v(k) \tag{3.6}$$

$$With\ v(k) \sim \mathcal{N}(0, R)$$

In Eq. 3.5, X(k) $\in \mathfrak{R}^n$ represents the states of the system , u(k) $\in \mathfrak{R}^n$ represents the manipulated inputs at the time instant k and d(k) $\in \mathfrak{R}^n$ represents a zero-mean Gaussian white noise with covariance matrix Q. In Eq. 3.6, Y(k) $\in \mathfrak{R}^n$ represents the measurement of nonlinear functions of the states, and v(k) $\in \mathfrak{R}^n$ represents zero-mean white Gaussian

noise with covariance R. Further, d(k) and v(k) are assumed to be independent and uncorrelated of each other. Noises at different time instant are uncorrelated. Steps of EKF are discussed as in Bhusan and Patwardhan (2018). The continuous system is converted into a discrete system using ODE-IVP at every time instant. The first-order approximation of Taylor's series of the nonlinear system equation around states estimated at the current instant is used by EKF.

$$\mathbf{X}(k+1) = \mathbf{F}(\hat{\mathbf{X}}(k|k), \mathbf{u(k)}, \mathbf{d(k)}) + \Phi(k)(\mathbf{X}(k) - \hat{\mathbf{X}}(k|k)) \tag{3.7a}$$

$$Where \;\; A(k) = \left.\frac{\partial f}{\partial x}\right|_{(\hat{x}(k|k),u(k),d(k))} \tag{3.7b}$$

$$\phi(k) = exp(A(k)t_s) \tag{3.7c}$$

Prediction step in EKF :

$$\hat{\mathbf{X}}(k+1) = \mathbf{F}(\hat{\mathbf{X}}(k|k), \mathbf{u(k)}, \mathbf{d(k)}) \tag{3.8a}$$

$$P(k+1|k) = \phi(k)P(k|k)\phi(k)^T + \gamma_d Q \gamma_d^T \tag{3.8b}$$

$$Y(k) = h(X(k)) + v(k) \tag{3.8c}$$

The correction step of EKF is given by:

$$K(k+1) = P(k+1|k)C(k)^T(C(k)P(k+1|k)C(k)^T + R)^{-1} \tag{3.9a}$$

$$\hat{X}(k+1|k+1) = \hat{X}(k+1|k) + K(k+1)(y(k+1) - h(\hat{x}(k+1|k))) \tag{3.9b}$$

$$P(k+1|k+1) = P(k+1|k) - K(k+1)C(k)P(k+1|k) \tag{3.9c}$$

$$Where \;\; C(k) = \left.\frac{\partial h}{\partial x}\right|_{(\hat{x}(k+1|k))} \tag{3.9d}$$

In Eq 3.7c ts represents sample time of the process. In Eq. 3.7c and Eq 3.9d, Φ(k) and C(k) are the Jacobian matrices, which are required to calculate predicted covariance, P(k + 1|k) and Kalman gain matrix, K(k + 1) in the Eq. 3.8b and Eq. 3.9a respectively.

Chapter 4

Proposed Work

State estimation problem for nonlinear systems is solved using estimators like EKF, UKF, MHE, and many others. For the prediction of the states, the estimator requires a model of the system. Capturing the dynamics of a nonlinear system using a mechanistic approach is cumbersome. Thus, artificial neural networks can be used for the same as discussed in section 3.1. Thus following strategies have been used to solve the state estimation problem, and the same has been reported in this chapter:

- RNN with EKF (**RNNEKF**)
    1. Tanh - RNN with EKF
    2. Relu - RNN with EKF

4.1 System identification using RNN

As every estimator requires a model, a recurrent neural network has been used in the proposed strategy to capture the dynamics of the system. A detailed explanation of the training of the RNN has been discussed in this section. There are different modeling strategies provided in the literature for capturing system dynamics using RNN. Most commonly used are Wang (2017):

1. Parallel mode
2. Series parallel mode

In the parallel mode, the current instant output of the neural network is predicted using system inputs and past outputs predicted by the neural network itself. Whereas, in the series-parallel mode, the current prediction of the network is made using manipulated inputs of the system and the past plant outputs. The series-parallel model is preferable

to parallel mode as it generates stable adaptive laws Chang (2014). Hence series-parallel method has been used for training the RNN model.

4.1.1 Data generation and Preprocessing

For the modeling of the system, firstly, open-loop simulation in MATLAB was carried out to generate data that is to be used to train the neural network. Thus a system with a differential equation representing the plant is required. Moreover, the sampling time of the system is chosen by looking at the dynamics of the system. Two simulations are done in MATLAB for generating data. The data generated for the training of the RNN is without adding any noise, i.e., neither process nor measurement noise was used to create the data.
Further, one more simulation, which includes the process and measurement noise, was finally used to get data that corresponds to the actual plant data. The EKF uses this data to generate filtered values of the states. The inputs of the system (manipulated inputs) and the output variables (measured and unmeasured) are stored as separate CSV files from both the simulation performed. As we are applying a series-parallel model, our model's input will be the manipulated variables of the systems and past output values from the plant. Therefore, in the input CSV file at each current time instant, previous time instant true state values are augmented to finally generate the input CSV, which consists of all the variables which are to be used. At the same time, the output CSV consists of the process states of the system, which are used as the target variable. The input fed to the RNN should be of the following form:

$$B \times T \times F \tag{4.1}$$

where,

- B: Number of batches,
- T: Time sequence,
- F: Number of features

Thus each CSV file input and output will be split into the form in Eq.4.1. A custom data loader was programmed, which breaks the generated 2d data into a multidimensional form which the RNN requires. Now, this data can be used to train the RNN.
This data is then split into following:

- Training dataset
- Validating dataset

- Test dataset

The data was split into above 3 section in the ratio of 80:10:10. In which Training data set was used for training of the network. Whereas, rest 2 sections are used to check the performance of the model.

4.1.2 Hyperparameter Tuning of the RNN

Hyperparameter tuning of the network refers to the fine-tuning of the hyperparameters of the neural network such that the network yield better results. Though there is no exact analytical way to get to the best architecture of the network, a common practice is first to consider a network with one hidden layer with few neurons and a user-specified activation function. If the results are not good, the number of neurons in the layer is increased. If the results are still not good, then the extra hidden layer is added, and the procedure is repeated. This is iterated until model accuracy becomes constant or starts decreasing.
Apart from tuning the network parameters, suitable optimizer and loss function are also chosen depending upon the problem. Generally, an Adam optimizer with a small learning rate is used, and the loss function depends upon the problem that RNN is solving. Mainly for regression problems, mean squared error is used as a loss function, and for classification problems, the cross-entropy function is applied. The optimum values of the parameters are obtained using the grid search technique. A different range of values is assigned to each parameter, and a grid consisting of different combinations of these values is used to analyze the model's accuracy. Then this field is narrowed down to a small range at which the model yields the best accuracies. This procedure gives us the optimum value of each parameter and thus makes the hyperparameter tuning of the network completed. After hyperparameter tuning, the model's efficacy is analyzed from the validated and test dataset.

4.2 RNN with EKF (RNNEKF)

To solve the state estimation problem, EKF has been used as the estimator. As every estimator requires a model, the RNN trained in section 4.1 has been used as a model and has been used in the prediction step of the EKF. Moreover, the pertinent parameters required in the correction step of the EKF have also been derived from the RNN and have been discussed below.

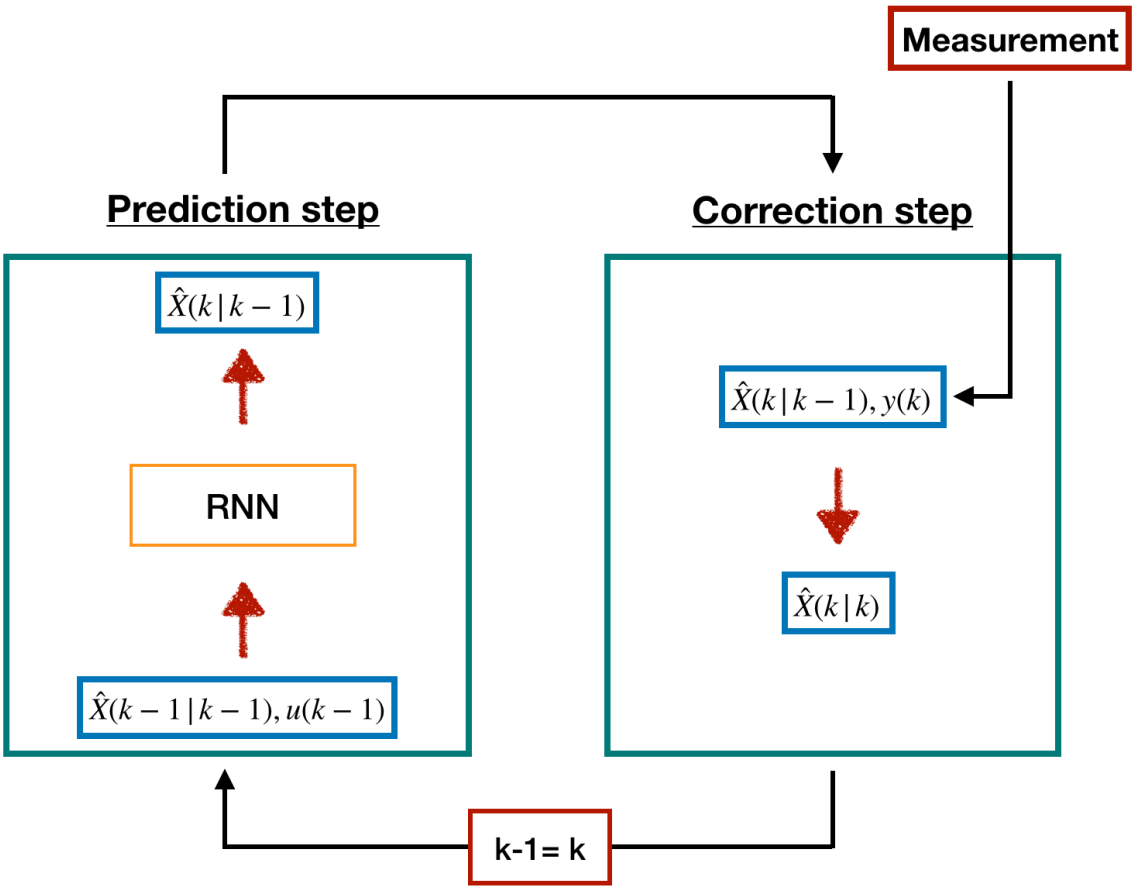


Figure 4.1: Flowchart for RNNEKF

4.2.1 EKF combined with RNN

Using the system's model generated in the above section 4.1, the state estimation problem has been solved using EKF. Filtered values of the states of the system are predicted using EKF as described in chapter 3. The prediction step in Extended Kalman Filter Eq. 3.8a can be replaced by the algebraic equation obtained from the recurrent neural network model of the system as shown in Eq. 3.3 and Eq. 3.4. Further, the parameters required from the model like Φ used in Eq. 3.8b can be derived from the algebraic RNN equations. The working of the RNNEKF strategy has been shown in Fig. 4.1 The way to derive the parameters has been taken from Habtom and Litz (1997) and is explained in detail below:

State space model derived from RNN Habtom and Litz (1997)

The algebraic equation that we get from the RNN as stated in Eq. 3.3 and Eq.3.4 can be re written as below in terms of control relevant terminology:

$$x(k+1) = f(W^u u(k) + W^y x^y(k) + W^d x(k) + b^u) \tag{4.2a}$$

$$x^y(k+1) = g(W^x f(W^u u(k) + W^y x^y(k) + W^d x(k) + b^u) + b^x) \tag{4.2b}$$

$$y(k) = x^y(k) \tag{4.2c}$$

Table 4.1: Parameters for recurrent neural network

Parameters	Definition
x(k)	Output vector of hidden units
$x^y(k)$	Output vector (states of the system) of the network at time k
u(k)	Vector of manipulated variables of the system
W^d	Weight matrix connecting the vector x(k) back to the inputs of the hidden units
W^u	Weight matrix connecting the hidden units with the input vector u(k)
W^y	Weight matrix for the connection between the hidden units and the vector x
b^u	Input biases
b^x	Output biases
$f(.)$	Activation function for hidden units
$g(.)$	Activation function for output units

Considering the RNN with external recurrence stated in Eq. 4.2, to derive Φ from these equations, we formulate an augmented state model in the sequel. To capture the dynamics of systems properly, we take hidden nodes of the neural network as additional states in the state model. Representing the above state model in the matrix form,

$$\begin{bmatrix} x(k+1) \\ x^y(k+1) \end{bmatrix} = \begin{bmatrix} f(.) \\ g(.) \end{bmatrix} + \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} w_1(k) \\ w_2(k) \end{bmatrix} \tag{4.3}$$

$$y(k) = \begin{bmatrix} 0 & | & I \end{bmatrix} \begin{bmatrix} x(k) \\ x^y(k) \end{bmatrix} + v(k) \tag{4.4}$$

Where f(.) and g(.) in Eq. 4.3 are as explained in 4.1, while I and 0 are identity and null matrices, respectively. The linearization of the state transition matrix in Eq. 4.4, which need to be computed at each time step k, results in the following form,

$$F = \begin{bmatrix} F1 & F2 \\ F4 & F5 \end{bmatrix} \tag{4.5}$$

where **F1** and **F2** are the linearized functions of **f(.)** with respect to the states x and x^y respectively, while **F3** and **F4** are that of **g(.)**. This **F** derived (Eq. 4.5) can now be used as $\mathbf{\Phi}$ inside EKF for getting filtered values of all the augmented states. Hence using the above analogy, the final output of the EKF is filtered values of states of the system and values of the augmented states, which were the hidden nodes of the RNN. Out of these values, only filtered values of the system states are fed back to the RNN, which gives prediction for the system states, which are again fed to EKF with values of hidden nodes, and this cycle is repeated.

Note: Unlike conventional EKF, here filtered values of hidden nodes are not used in prediction of $\hat{\boldsymbol{x}}(k+1|k)$

4.2.2 Two types of RNNEKF proposed

Various activation functions can be used in RNN, two activation functions has been explored for RNNEKF strategy which were as follows:

1. ReLU - RNNEKF:
   In this strategy, RNNEKF used the ReLU activation function inside RNN to capture the physics of the system. ReLU stands for Rectified Linear Unit, which is mathematically represented as Eq. 4.6 and pictorially as in figure 4.2. It has become the default activation function for many neural networks as it makes the network's training easier and often yields better performance.

$$\boldsymbol{g(x) = max(0, x)} \tag{4.6}$$

2. Tanh - RNNEKF:
   Tanh activation function stands for the tan hyperbolic function. Mathematically tanh is represented as Eq. 4.7 and pictorially as in figure 4.3. As in EKF, the $\boldsymbol{\phi}$ is generated from differentiating the algebraic equation given by RNN. Hence tanh is more preferred than ReLU as an activation function. But as we can see in figure 4.3 tanh saturates fast hence there is a need for standardizing the input before inputting it into RNN. Therefore, while using tanh - RNNEKF, the input data is first standardized and then fed to the RNN.

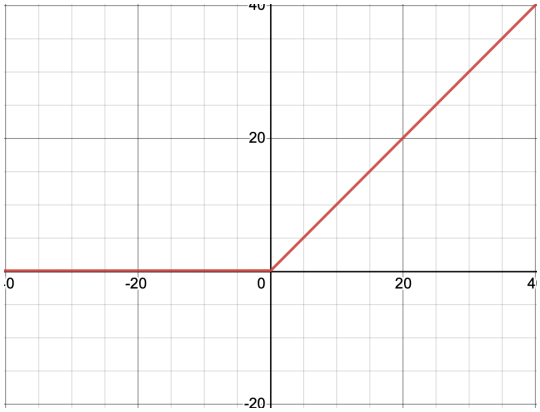


Figure 4.2: Rectified Linear Unit

$$g(x) = tanh(x) \tag{4.7}$$

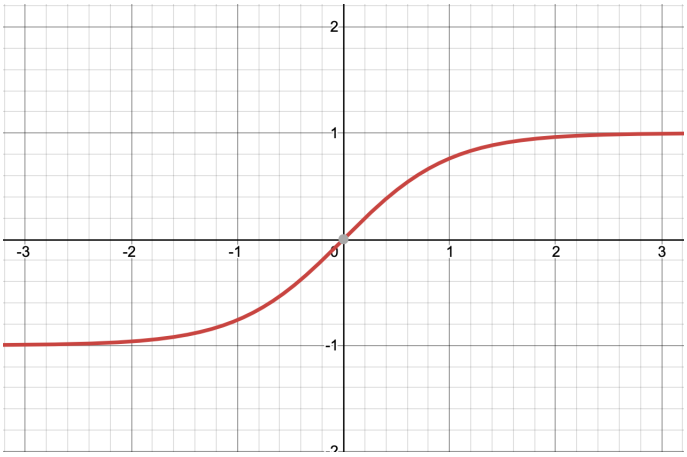


Figure 4.3: Tan hyperbolic function

Similar to RNNEKF, a hybrid technique has also been explored (HRNNEKF). Unlike RNNEKF, where for the correction step, the state-space model was derived from the RNN, in HRNNEKF, the state-space model used in EKF has been derived from the first principle model of the system. Hence in this strategy, the prediction step in EKF is being done using RNN, and a physics-based model is used for all other steps in EKF.
The steps for modeling the system using RNN remain the same as explained in section 4.1.1. After training the RNN, this trained RNN is used in the prediction step of the EKF, and finally, from EKF, we get filtered values of the states. Hence this acts like a grey box strategy which utilizes RNN for solving modeling problem and uses a physics-based model to run EKF. This strategy has been implemented in the CSTR case study, and the results for the same have been reported in Appendix A.

Chapter 5

Case Studies

The efficacy and performance of the proposed strategies on three case studies have been discussed in this chapter. The first section of the chapter consists of results obtained by applying proposed methods on **CSTR**, followed by **Fourtank**, and **Temperature Control Lab** system.

5.1 Continuous Stirred Tank Reactor

This case study consist of nonlinear process dynamics and measurement dynamics is linear in nature. Considering the irreversible first order reaction as given in Eq. (5.1)

$$A \longrightarrow B \tag{5.1}$$

From the first principles Marlin (1995), the model of the reactor can be obtained as:

$$\begin{aligned} \frac{dX_1}{dt} &= f_1\left(X_1, X_2, U_1, U_2, D\right) \\ &= \frac{U_2}{V}\left(D - X_1\right) - k_0 X_1 \exp\left(-\frac{E}{RX_2}\right) \end{aligned} \tag{5.2a}$$

$$\begin{aligned} \frac{dX_1}{dt} &= f_2\left(X_1, X_2, U_1, U_2, D\right) \\ &= \frac{U_2}{V}\left(T_0 - X_2\right) + \frac{\left(-\Delta H_r\right) k_0}{\rho C_p} X_1 \exp\left(-\frac{E}{RX_2}\right) - \frac{Q}{V\rho C_p} \end{aligned} \tag{5.2b}$$

$$Q = \frac{a\left(U_1\right)^{b+1}}{U_1 + \left(\frac{a(U_1)^b}{2\rho_c C_{pc}}\right)}\left(X_2 - T_{cin}\right) \tag{5.2c}$$

Table 5.1: Parameters for recurrent neural network

Parameters	Definition
$\boldsymbol{X_1}$	Concentration of A in the reactor
$\boldsymbol{X_2}$	Reactor Temperature
D	Disturbance, inlet concentration of A
$\boldsymbol{U_1}$	Manipulated variable 1, Feed flow rate of reactant A
$\boldsymbol{U_2}$	Manipulated variable 2, Flow rate of coolant to jacket

Following are the variables which were used in ordinary differential equation:

The state and measurement vector are defined as follows:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{X_1} \\ \boldsymbol{X_2} \end{bmatrix}, \qquad \boldsymbol{y} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} \boldsymbol{x} \tag{5.3}$$

Non isothermal CSTR reaction is a continuous system, which is discretized with sampling interval (Ts) of 0.1 min. Following are the model parameter values used for the system:

Parameter (↓) Operating Point (→)		Stable
Reaction rate constant (k_0)	min^{-1}	10^{10}
Density of the reagent A (ρ)	g/m^3	10^6
Specific heat capacity of A(C_p)	cal/g^0C	1.0
Heat of reaction (ΔH_r)	$cal/kmol$	$-130 * 10^6$
Density of the coolant (ρ_c)	g/m^3	10^6
Specific heat capacity of coolant (C_{pc})	cal/g^0C	1.0
Volume of the CSTR (V)	m^3	1.0
Inlet temperature of the coolant (T_{cin})	^0K	365
Inlet temperature of A (T_0)	^0K	323
a		$1.678X10^6$
Reaction Rate Parameter (E/R)	$(^0K)^{-1}$	8330
b		0.5

(a) Model parameters

Parameter (↓) Operating Point (→)		Stable
Inlet flow rate of A ($U_1 \equiv F$)	m^3/min	1.0
Coolant flow rate ($U_2 \equiv F_c$)	m^3/min	15
Reactor concentration of A ($X_1 \equiv C_A$)	$kmol/m^3$	0.265
Reactor temperature ($X_2 \equiv T$)	K	393.954
Inlet concentration of A ($D \equiv C_{A0}$)	$kmol/m^3$	2.0

(b) Equilibrium operating conditions

Figure 5.1: Parameters for CSTR system

For the data generation, simulation for non-isothermal CSTR reaction has been done with MATLAB version R2020b online. Where 10% PRBS step changes were introduced in the manipulated inputs U1 and U2 for capturing effective dynamics of the system. Root Mean Square Error (RMSE) is used for measuring performance which is mathematically represented as Eq. 5.4

$$\mathbf{RMSE} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(\mathbf{x}(k) - \widehat{\mathbf{x}}(k \mid k))^T(\mathbf{x}(k) - \widehat{\mathbf{x}}(k \mid k))} \tag{5.4}$$

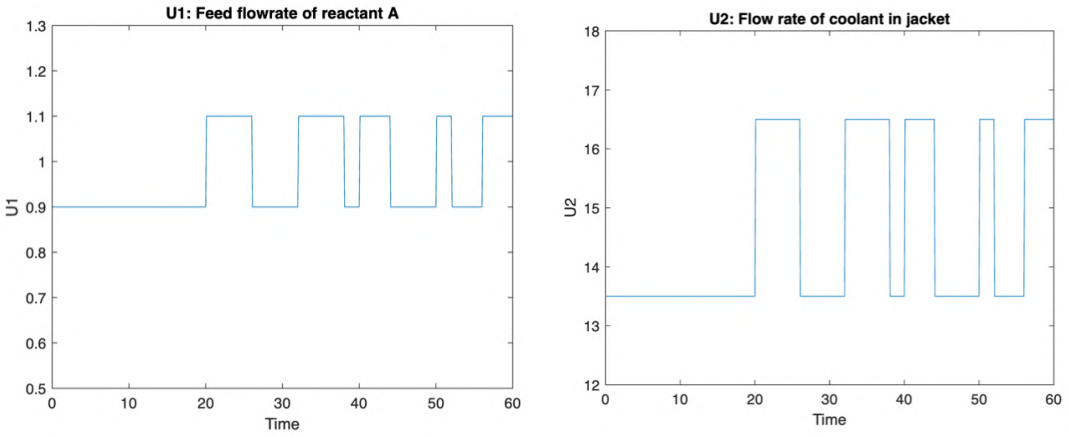
where, $\boldsymbol{x(k)}$ and $\hat{\boldsymbol{x}}\boldsymbol{(k|k)}$ true state vector and estimated state for time instant for $\boldsymbol{k^{th}}$ time instant and N represent the number of time instant.

5.1.1 ReLU-RNNEKF on CSTR

As explained in section 4.1 ReLU RNNEKF was implemented on the CSTR system and the results of the same has been reported below:

Modelling of CSTR using RNN

As discussed in section 4.1.1, datasets for the CSTR system are generated for the training of the RNN. Thus for the training and testing of the neural network, 16666 points were developed in MATLAB using a mechanistic model of CSTR (Eq. 5.2). Below are the graphs of the manipulated input and states obtained from the simulation:



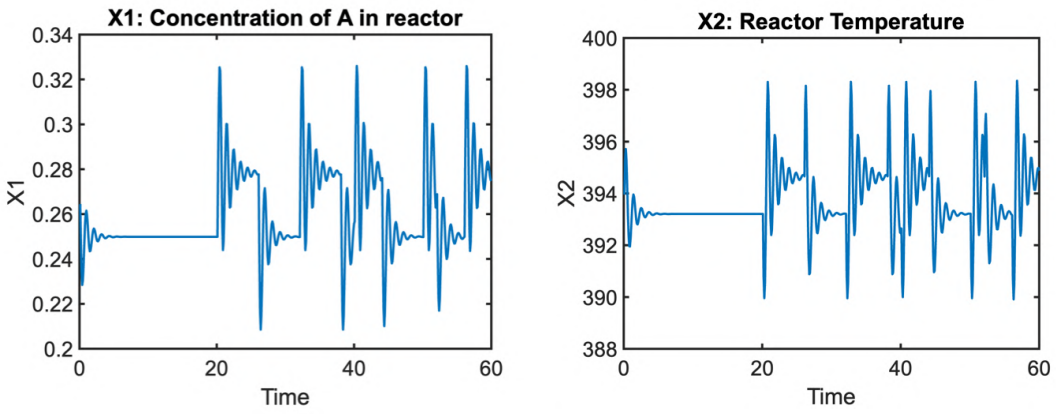
(a) Feed flow rate vs Time

(b) Coolant flow rate vs Time

Figure 5.2: Manipulated inputs vs TIme

As we can see from Fig. 5.3, the range of both the states was different. Hence, incorporating the loss associated with concentration effectively, concentration was scaled by a factor of 100 for training. Then this dataset was divided into three sets:

- Training dataset. (13332 points)
- Validating dataset. (1666 points)
- Test dataset. (1668 points)



(a) Concentration of A vs Time

(b) Temperature vs Time

Figure 5.3: States of CSTR vs Time

With MSE as loss function, and Adam Kingma and Ba (2014) as the optimizer, using the training dataset, the RNN was trained. After hyperparameter tuning of the RNN following is the values of parameters that were used for the prediction of the states:

Table 5.2: Tuned hyperparameters for CSTR RNN

T	Learning rate	Neurons in each hidden layer	Number of hidden layers	Epochs
10	0.00015	8	2	210000

T represents the length of the time sequence or the window size used to predict the current instant values. Using the above hyperparameter values, RNN was trained using a training dataset consisting of 13333 data points for 2.1 lakh epochs.

Loss plots for CSTR-RNN

The values of the loss function to epochs are shown in Fig. 5.4. Further, due to the high range of loss values, figure 5.4 and 5.6 have been plotted, which represents the loss values for <900 epochs and >900 epochs, respectively.

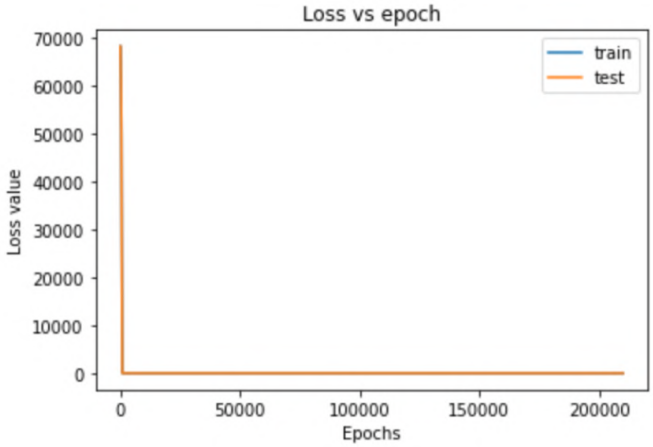


Figure 5.4: ReLU RNNEKF - CSTR: Loss vs Epochs

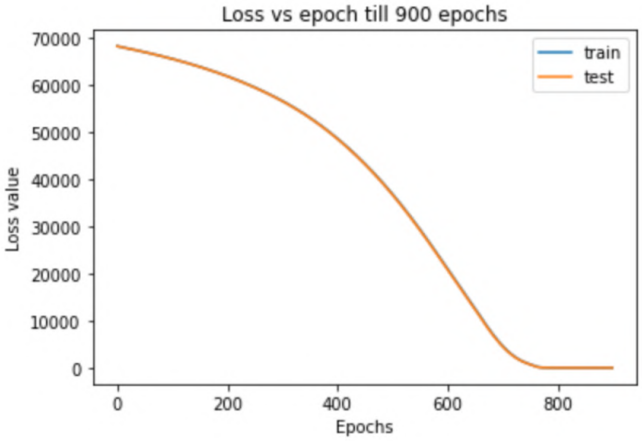


Figure 5.5: ReLU RNNEKF - CSTR: Loss vs Epochs

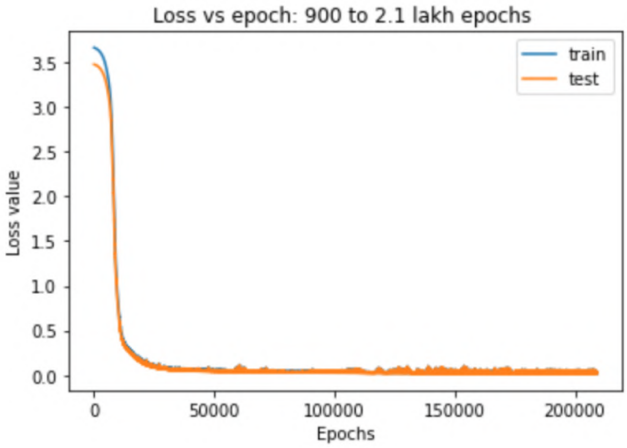
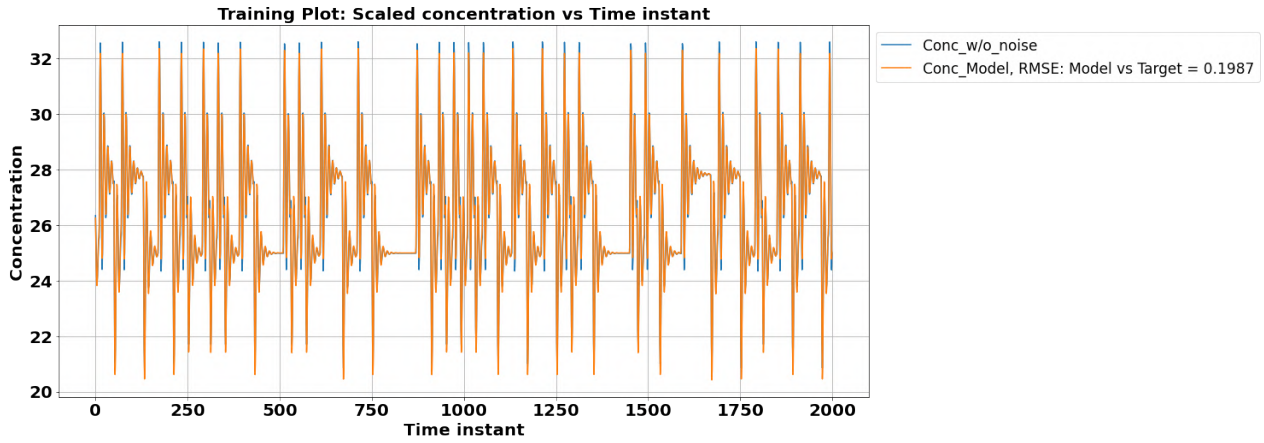


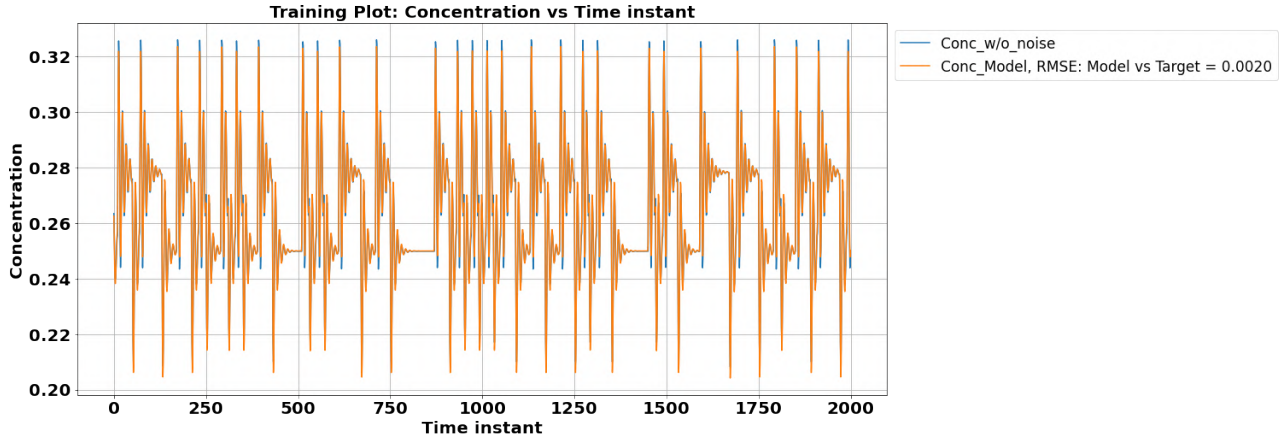
Figure 5.6: ReLU RNNEKF - CSTR: Loss vs Epochs

One step predictions for CSTR-RNN

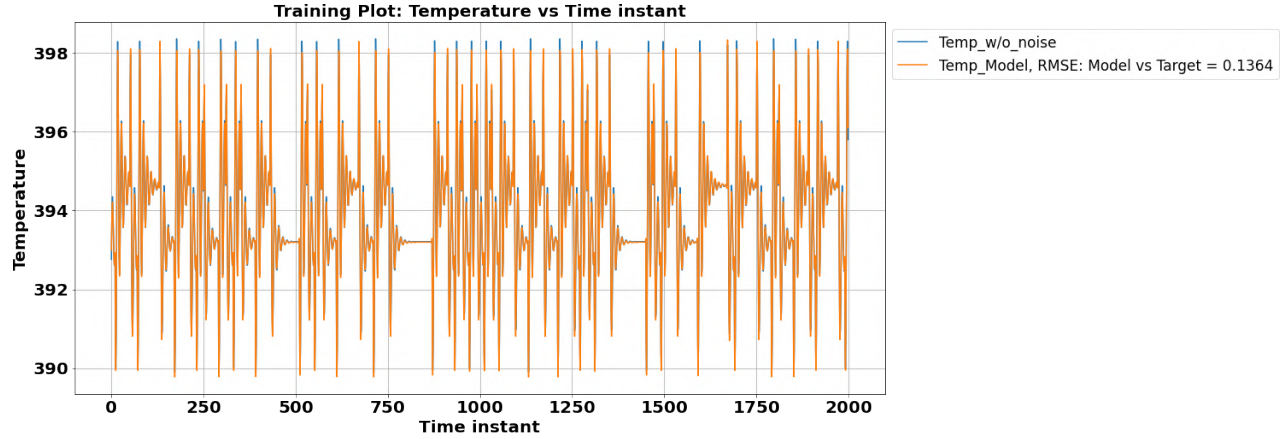
Following were the mapping done by the RNN for one step prediction:



(a) Concentration scaled vs Time instant



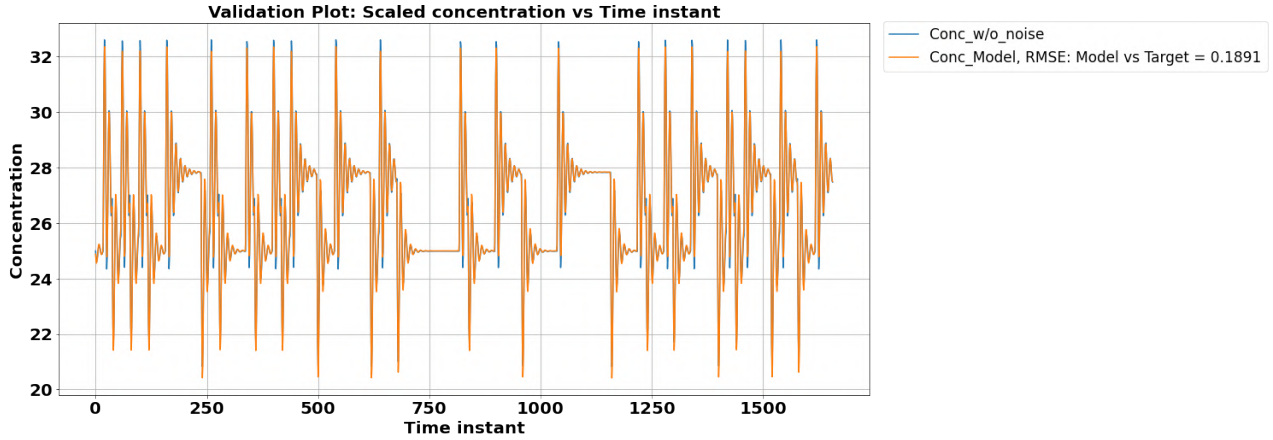
(b) Concentration vs Time instant



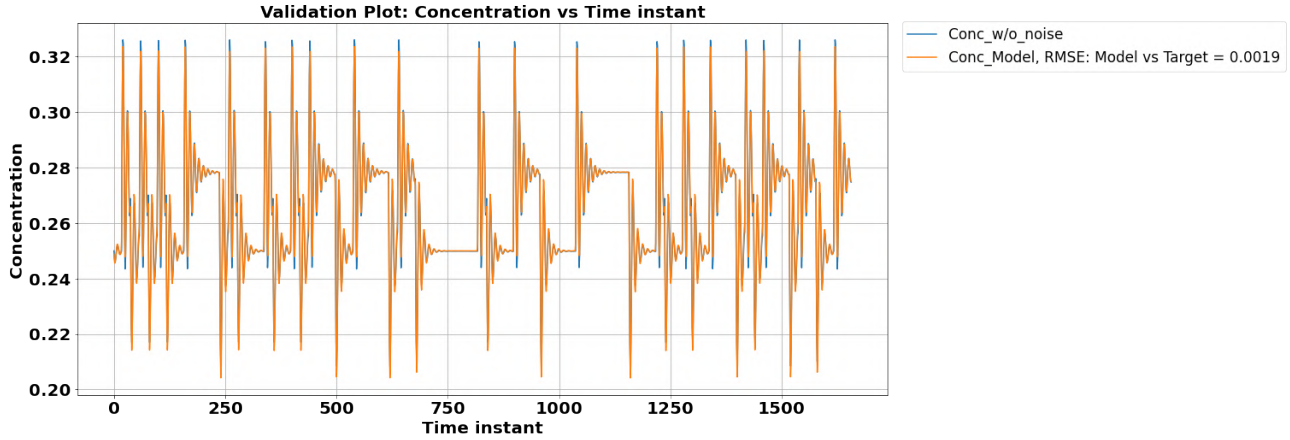
(c) Temperature vs Time instant

Figure 5.7: ReLU RNNEKF Train: States of CSTR vs Time instant

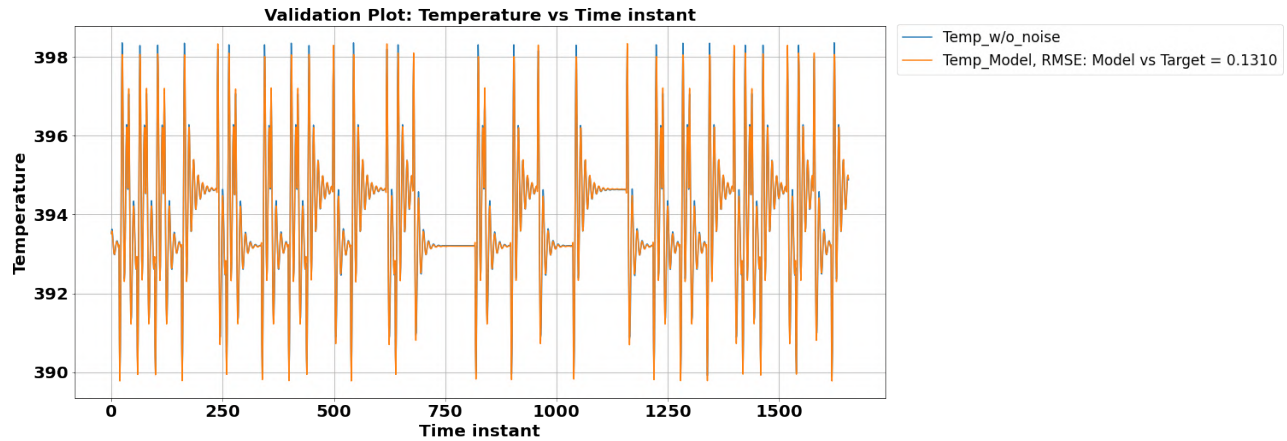
Results on the validation set



(a) Concentration scaled vs Time instant



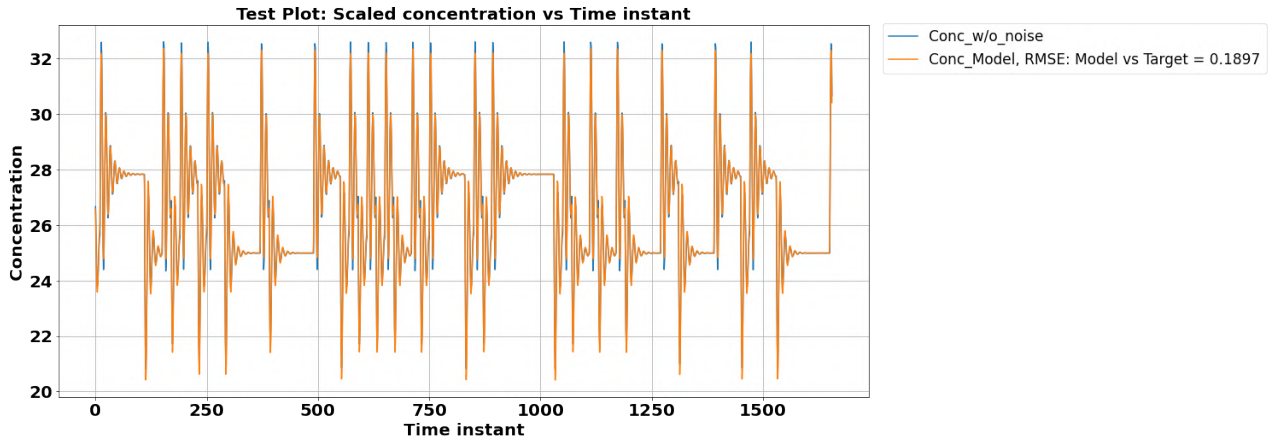
(b) Concentration vs Time instant



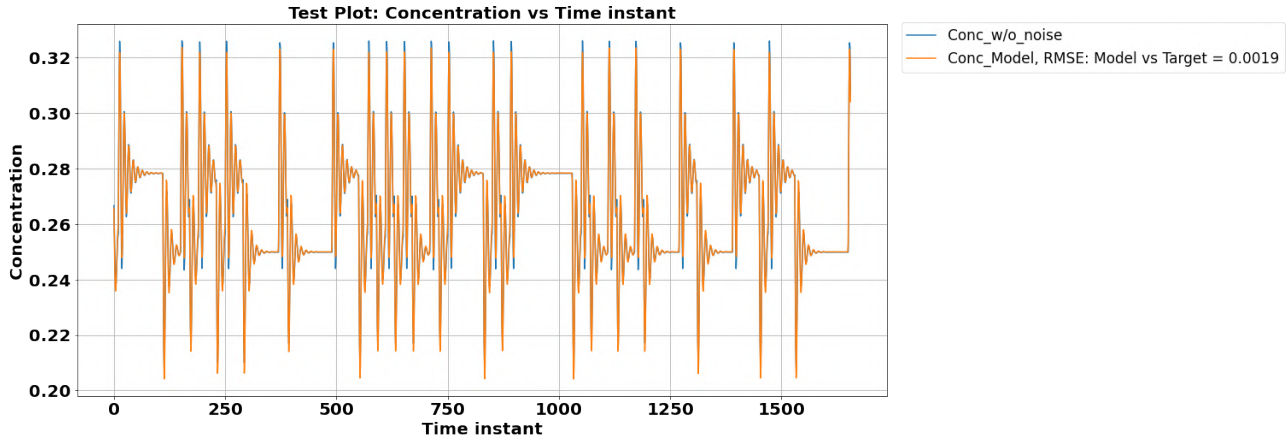
(c) Temperature vs Time instant

Figure 5.8: ReLU RNNEKF Validation: States of CSTR vs Time instant

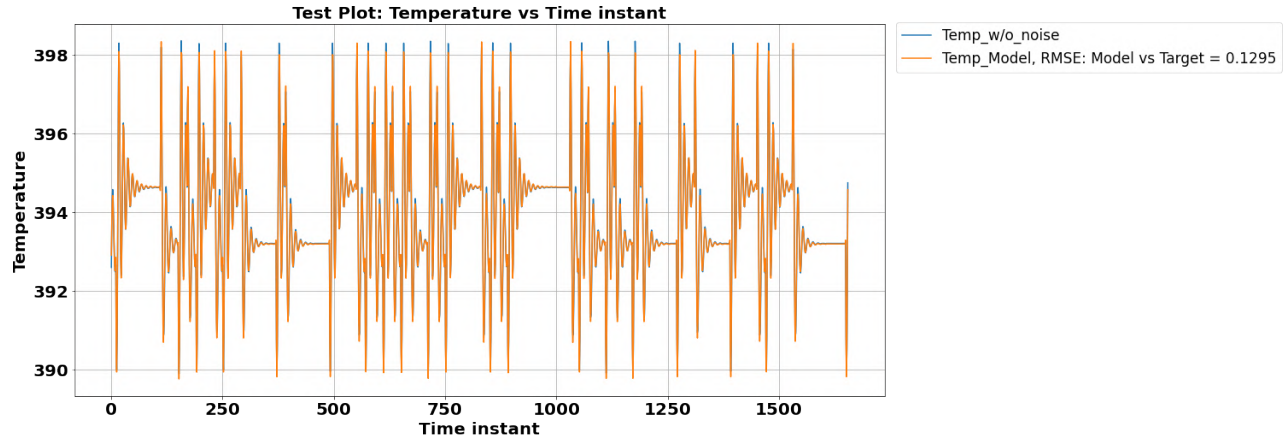
Results on the Test set



(a) Concentration scaled vs Time instant



(b) Concentration vs Time instant



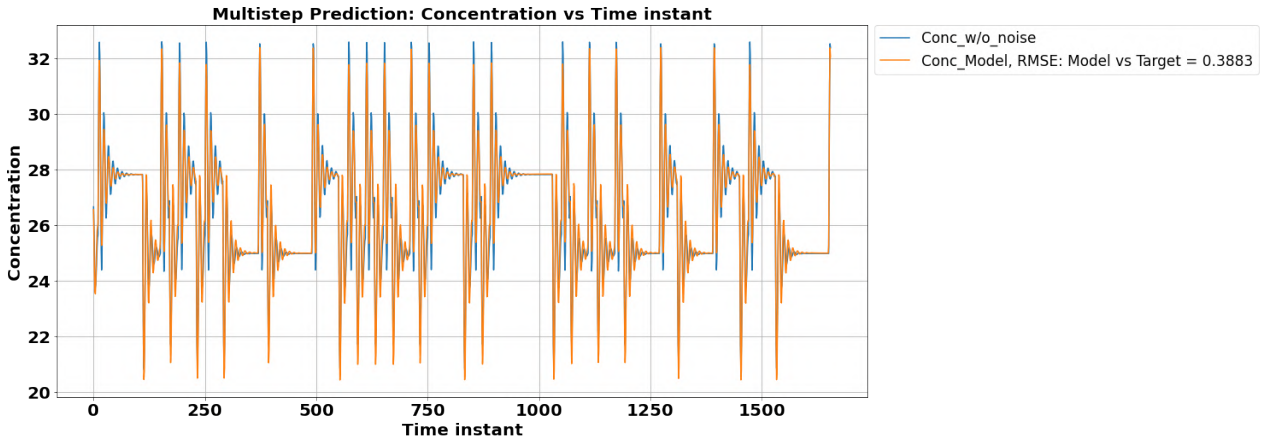
(c) Temperature vs Time instant

Figure 5.9: ReLU RNNEKF Test: States of CSTR vs Time instant

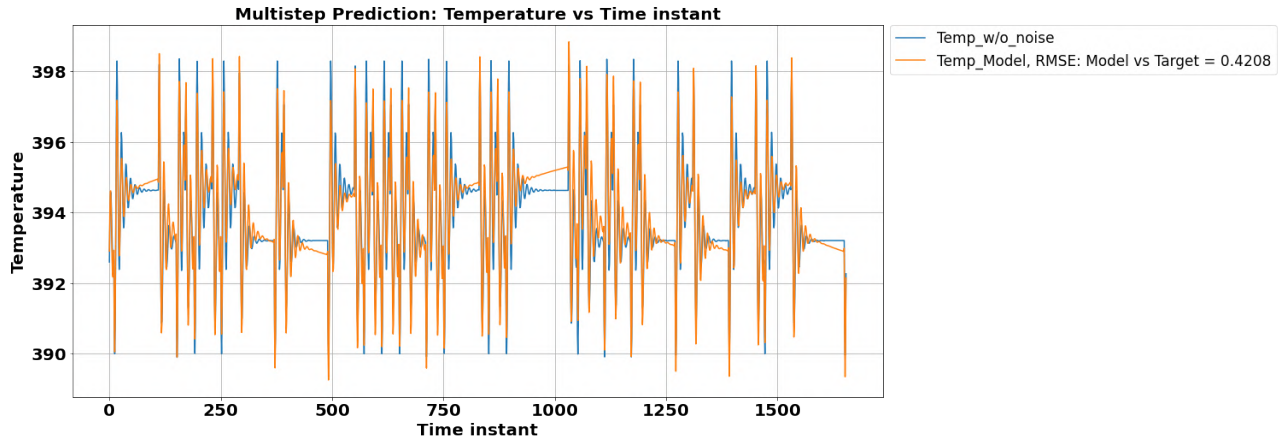
Fig 5.7 , Fig. 5.8 and Fig. 5.9 shows the plots for the states of the system vs time for the training, validation and test data set respectively.

Multi step predictions for CSTR-RNN

After training the network using series-parallel mode now, the network was tested for multi-step prediction. The output of the previous instant is directly fed as input for the prediction of the following instance. Thus, this will help in giving predictions purely on estimated values done by the network. Test data set was used to evaluate the performance of the model for the multi-step prediction paradigm. Results were as follows:



(a) Concentration vs Time instant



(b) Temperature vs Time instant

Figure 5.10: ReLU RNNEKF Multistep Test: States of CSTR vs Time

As we can see from Fig. 5.10, the RNN model of the CSTR has captured the physics of the system well. Hence, the model having parameters as shown in Table 5.3 can now be combined with EKF to give filtered values of the states to solve the state estimation problem for the CSTR.

Table 5.3: Summary of RNN model of CSTR

T	Learning rate	Neurons in each hidden layer	Number of hidden layers	Epochs	Loss Train	Loss Test
10	0.00015	8	2	210000	0.0290	0.0265

States estimated using ReLU RNNEKF strategy for CSTR

Using the model summarized in Table 5.3, filtered values of the states of the system are estimated using EKF as an estimator. The values of Q, R, C, P(0|0) have been taken from Patwardhan (2018) and has been transformed due to the addition of augmented states as discussed in section 4.1.2. Following initial values of the states and covariance matrix were taken for the estimation:

$$x(0|0) = \hat{x}(0|0) = \begin{bmatrix} 0.257 \\ 391.57 \end{bmatrix} \tag{5.5a}$$

$$P(0|0) = \begin{bmatrix} 0.0001 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0.0001 & \ddots & \ddots & & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & & \ddots & \ddots & 0.0001 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0.0001 \end{bmatrix}_{18x18} \tag{5.5b}$$

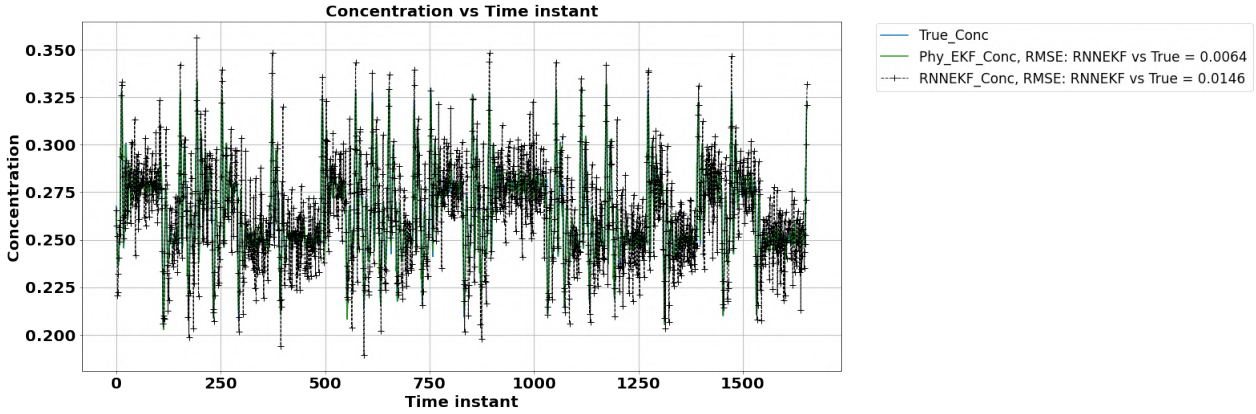
$$C = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix}_{1x18} \tag{5.5c}$$

Plant system model and measurement are incorporated with white Gaussian noise with zero mean and covariance matrix Q and R as:

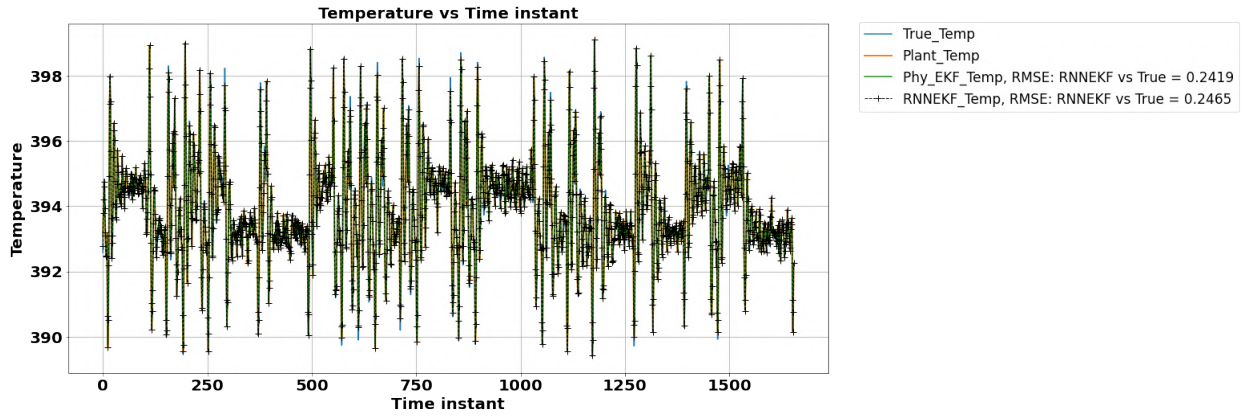
$$Q = \begin{bmatrix} 0.012^2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0.012^2 & \ddots & \ddots & & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & & \ddots & \ddots & 0.012^2 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0.012^2 \end{bmatrix}_{18x18} \tag{5.6a}$$

$$R = 0.25^2 \tag{5.6b}$$

As discussed in section 4.1.2, the state-space model of CSTR has been generated from RNN. More detailed derivation for generating $\mathbf{\Phi}$ from RNN for the CSTR system has been explained in Appendix B. Thus EKF was applied with the RNN model, and the results of combined RNN with EKF has been reported below:



(a) Concentration vs Time instant



(b) Temperature vs Time instant

Figure 5.11: Estimated states using ReLU RNNEKF on CSTR

Fig. 5.11 gives the values of filtered values of states from the EKF, which was derived from the RNN itself. From these results, it can be concluded that the proposed idea of using RNN as the model and combining it with EKF gives good results for the CSTR state estimation problem with low RMSE values. As we can see from Table 5.4, both RNNEKF and conventional physics-based EKF give similar RMSE for both states. Moreover, the computation time taken by RNNEKF was found to be **3.782*s***, which is **69.62%** less than the computation taken by conventional EKF. Hence, with less time consumption of modeling and expense, RNNEKF may be a better alternative to traditional strategies used until now.

Table 5.4: Summary of HRNNEKF and conventional EKF on CSTR

States	Proposed Work	RMSE	Computation Time(s)
Temperature	RNNEKF	0.2465	3.782
Concentration	RNNEKF	0.0146	3.782
Temperature	Physics EKF	0.2419	12.45
Concentration	Physics EKF	0.0064	12.45

5.2 Quadruple Tank

This benchmark problem consists of four tanks, as shown in Fig. 5.12. It comprised four states: the height of liquid in every tank, in which the height of liquid in Tank 1 and Tank 2 was used as the controlled variable. The height of liquid in Tank 1 and Tank 2 has been controlled using two pump flow rates, F1 and F2. Fig. 5.12 shows the process diagram of quadruple tank problem which has been reproduced from Mukherjee (2020).

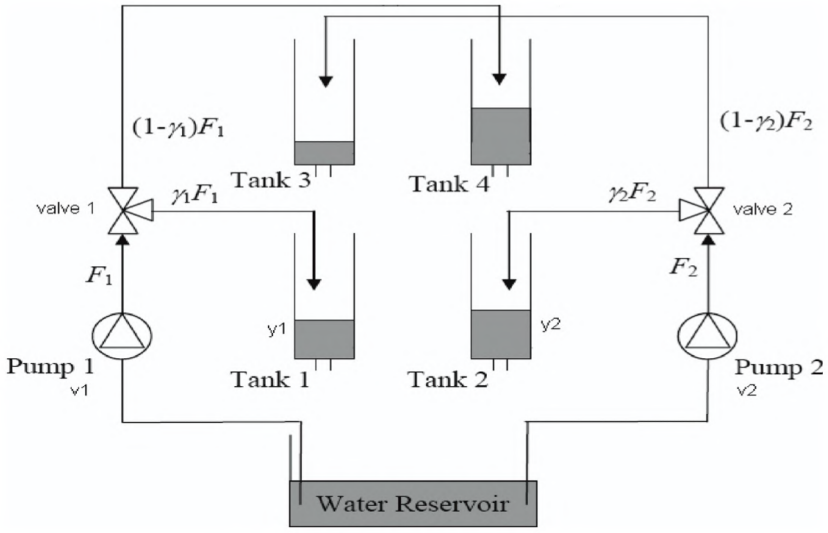


Figure 5.12: Quadruple Tank system

The physics based differential equation for quadruple tank is as follows:

$$\frac{dh_1}{dt} = -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 F_1}{A_1} \tag{5.7a}$$

$$\frac{dh_2}{dt} = -\frac{a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2 F_2}{A_2} \tag{5.7b}$$

$$\frac{dh_3}{dt} = -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{1-\gamma_2 F_2}{A_3} \tag{5.7c}$$

$$\frac{dh_4}{dt} = -\frac{a_4}{A_4}\sqrt{2gh_4} + \frac{1-\gamma_1 F_1}{A_4} \tag{5.7d}$$

In Eq. (5.7a) - (5.7d) h_i represents the i^{th} state, which is the height of liquid in i^{th} tank. F_1 and F_2 represent the total flow rate generated by two respective pump, which has been used as manipulated inputs. a_i represents the cross section area of bottom outlet nozzle of the i^{th} tank and A_i represents cross sectional area of the i^{th} tank. γ_1 and γ_2 are predetermined parameters. Height of liquid in Tank 1 and Tank 2, that is h_1 and h_2 has been used as controlled outputs. States has been estimated by EKF by the strategies proposed. Sampling time (ts) has been taken as 5 sec. Parameters of the quadruple tank model given in Table 5.5 Thosar *et al.* (2020). The implementation of the proposed strategies **ReLU RNNEKF** and **Tanh RNNEKF** onto quadruple tank has been discussed in the next section.

Table 5.5: Model parameters for quadruple tank

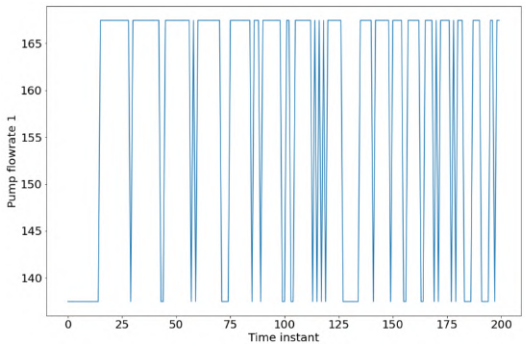
$A_1 = A_2 = A_3 = A_4 = 192cm^2$
$\gamma_1 = 0.55$
$\gamma_2 = 0.47$
$a_1 = 0.852cm^2$
$a_2 = 0.755cm^2$
$a_3 = 0.661cm^2$
$a_4 = 0.612cm^2$
$g = 981cm^2$

5.2.1 ReLU-RNNEKF on quadruple tank

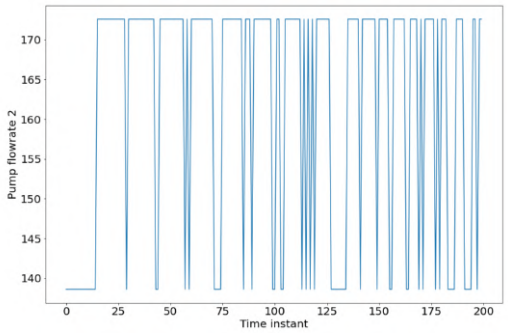
As discussed in section 4.1, ReLU RNNEKF was implemented on quadruple tank system and below results for the same has been discussed.

Modelling of quadruple tank using RNN

As explained in section 4.1.1, open-loop simulation was carried out to generate datasets of the quadruple tank for the training and validation of the RNN. 10% PRBS fluctuations were given to the steady-state values of the manipulated variable to activate the dynamics of the system. Total 16666 points were generated in MATLAB using a mechanistic model of the quadruple tank (Eq. 5.7). Below are the graphs of the manipulated input and states obtained from the simulation:

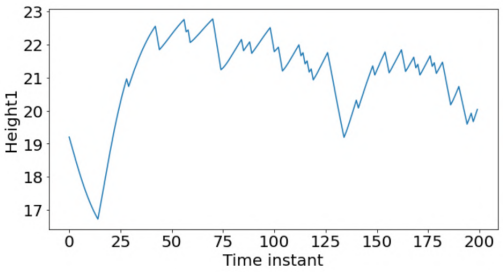


(a) Pump flow rate 1 vs Time instant

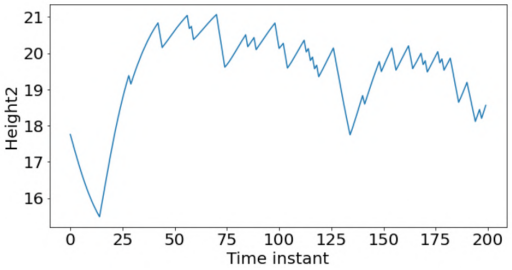


(b) Pump flow rate 2 vs Time instant

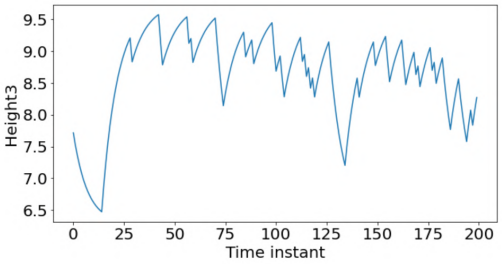
Figure 5.13: Manipulated inputs vs Time instant



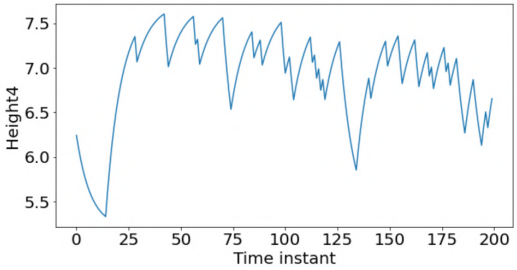
(a) Height 1 vs Time instant



(b) Height 2 vs Time instant



(c) Height 3 vs Time instant



(d) Height 4 vs Time instant

Figure 5.14: States of quadruple tank vs Time instant

Fig. 5.13 shows the PRBS fluctuations given to manipulated variables to activate the dynamics of the system as shown in Fig. 5.14. Then this dataset was divided into 3 sets:

- Training dataset. (13332 points)
- Validating dataset. (1666 points)
- Test dataset. (1668 points)

With MSE as loss function, training data set, and Adam Kingma and Ba (2014) as the optimizer, the RNN was trained. After hyperparameter tuning of the RNN following is the parameter which was used for the prediction of the states:

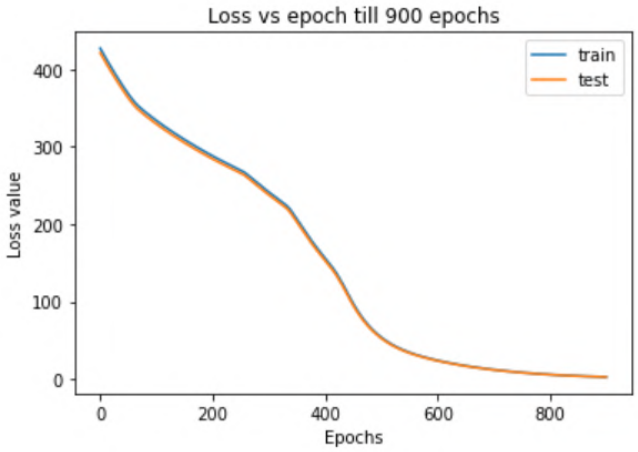
Table 5.6: Tuned hyperparameters for quadruple tank RNN

T	Learning rate	Neurons in each hidden layer	Number of hidden layers	Epochs
10	0.0001	10	2	21000

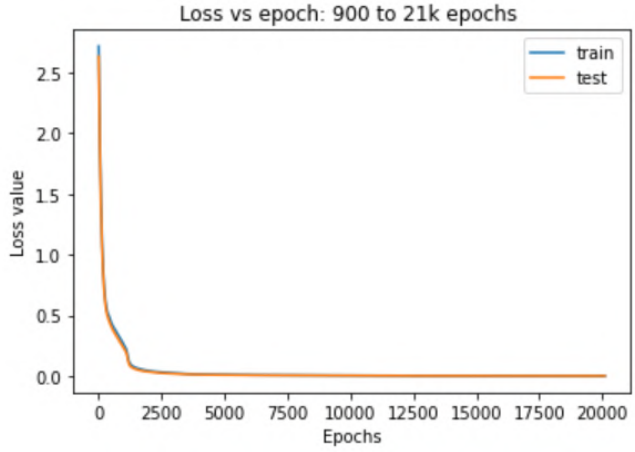
T represents the length of the time sequence or the window size used to predict the current instant values. RNN was trained using a training dataset consisting of 13333 data points for 21k epochs using the above hyperparameter values.

Loss plots for quadruple tank-RNN

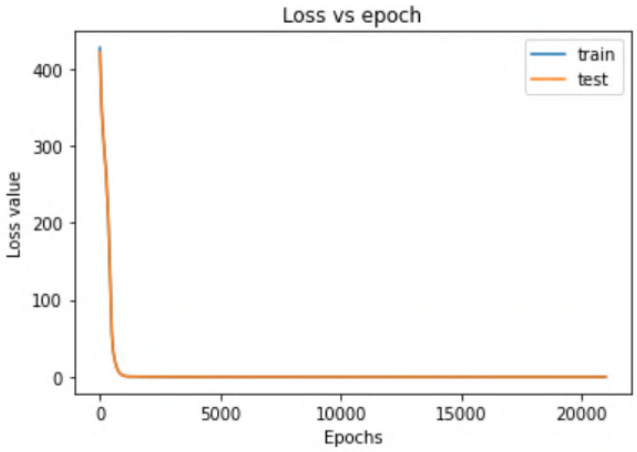
The values of the loss function to epochs are shown in figure 5.15c. Further due to high range of loss values, figure 5.15 and 5.15b has been plotted which represents the loss values for < 900 epochs and > 900 epochs respectively.



(a) Loss vs Epochs



(b) Loss vs Epochs

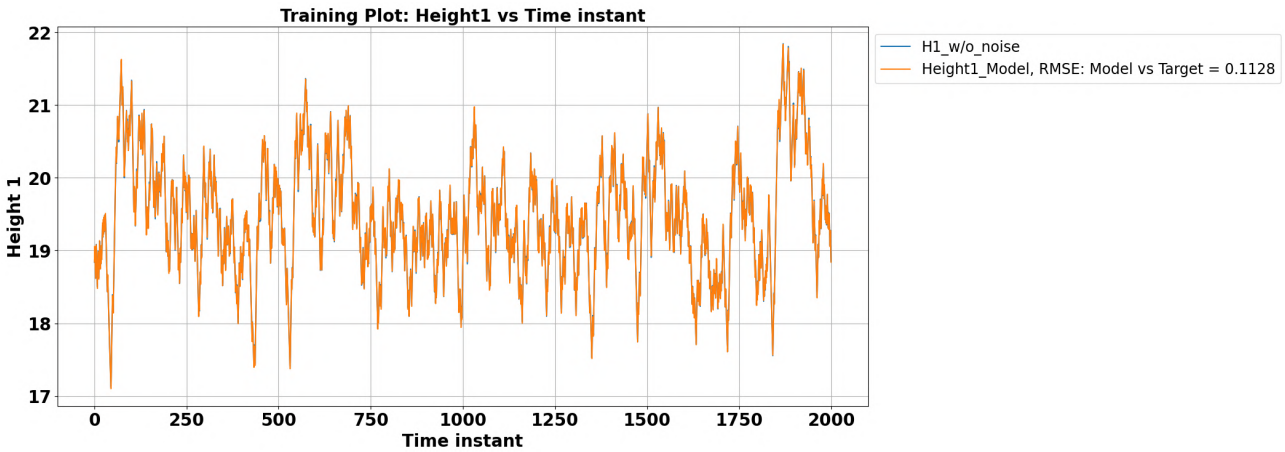


(c) Loss vs Epochs

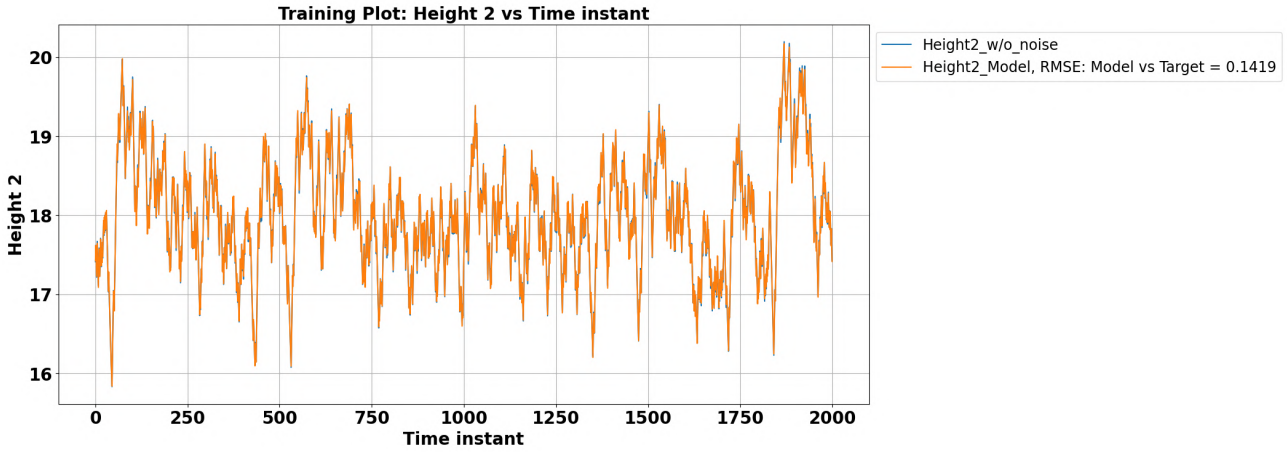
Figure 5.15: ReLU RNNEKF Quadruple Tank: Loss vs Epochs

One step predictions for quadruple tank-RNN

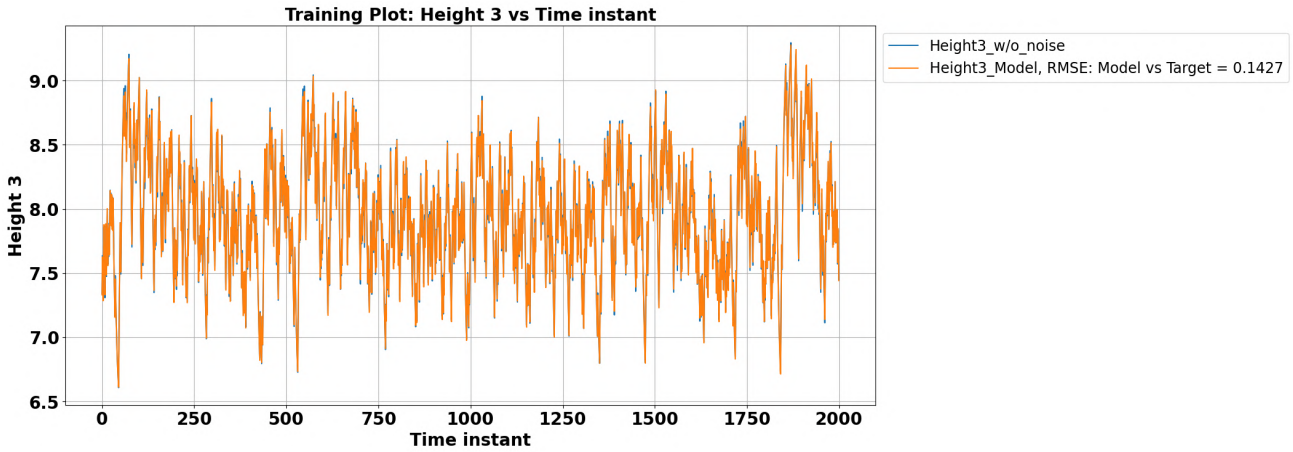
Following were the mapping done by the RNN for one step prediction of the states:



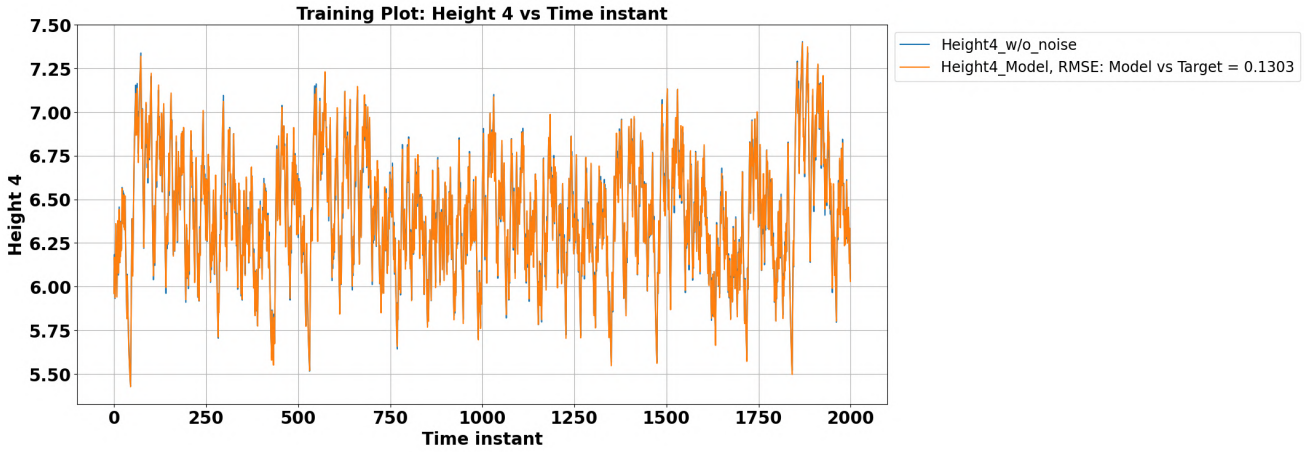
(a) Train: Height 1 vs Time instant



(b) Train: Height 2 vs Time instant



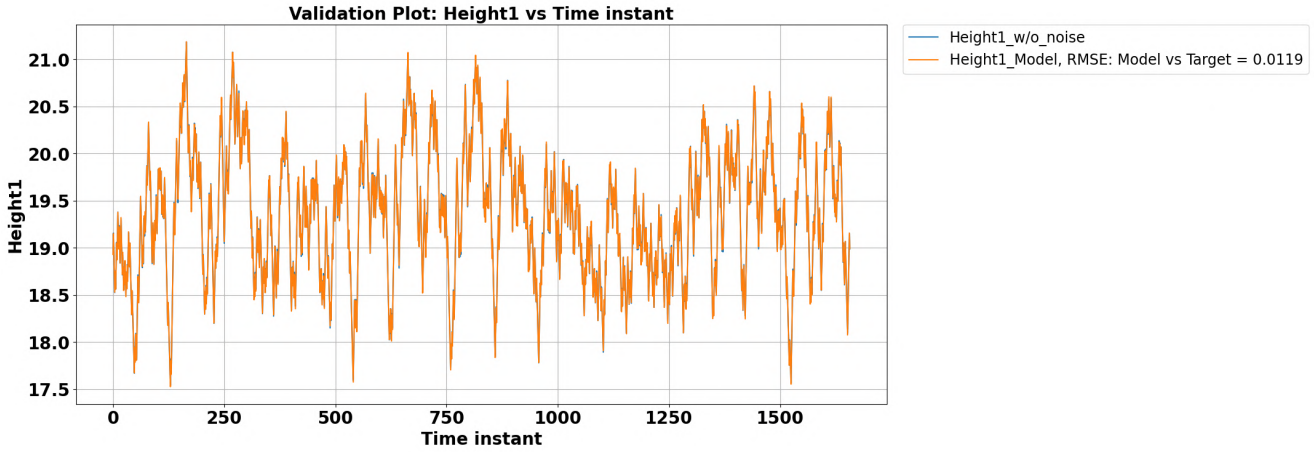
(c) Train: Height 3 vs Time instant



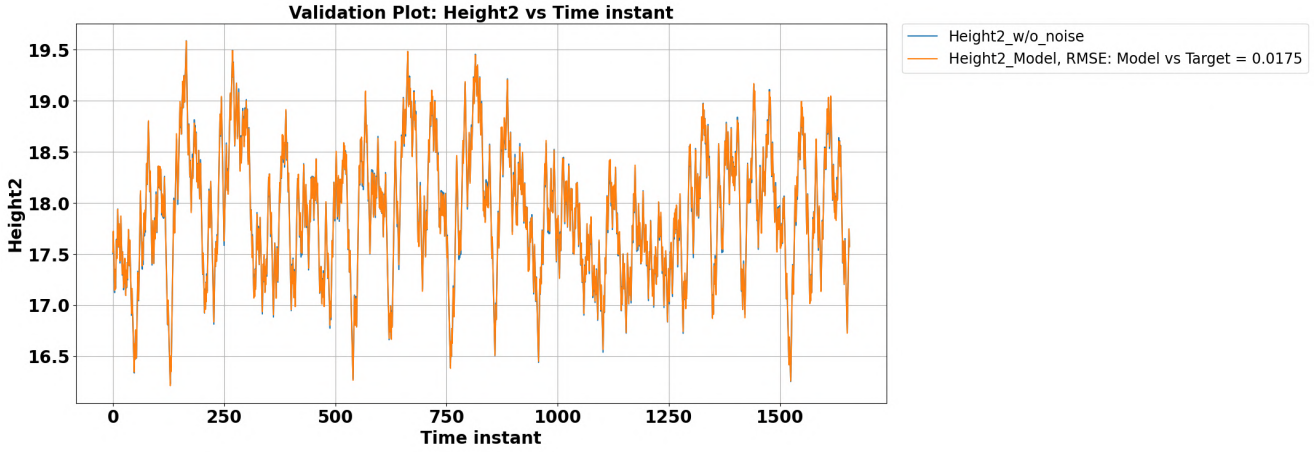
(d) Train: Height 4 vs Time instant

Figure 5.16: ReLU RNNEKF Train: States of quadruple tank vs Time instant

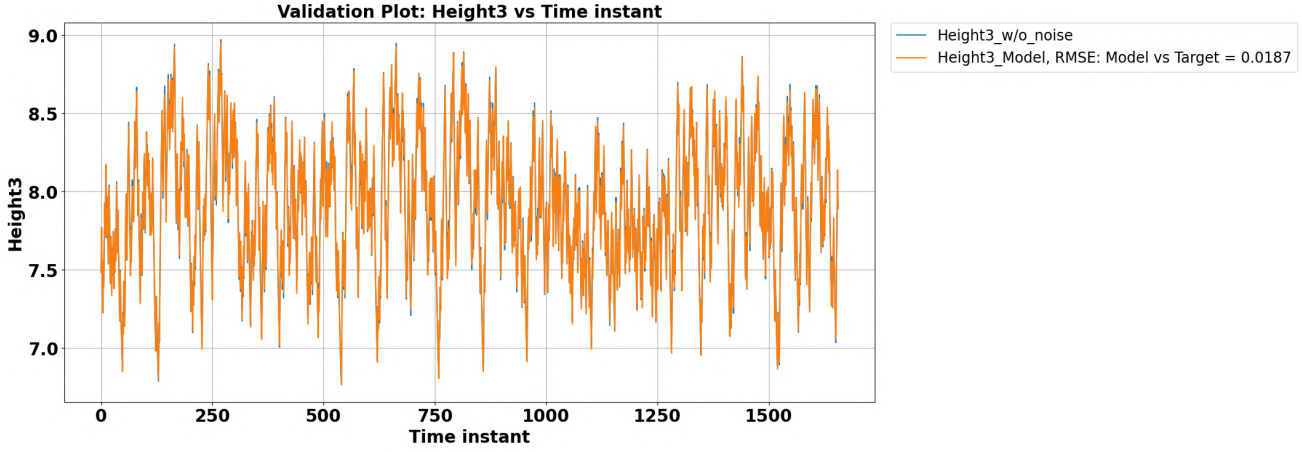
Results on the validation set



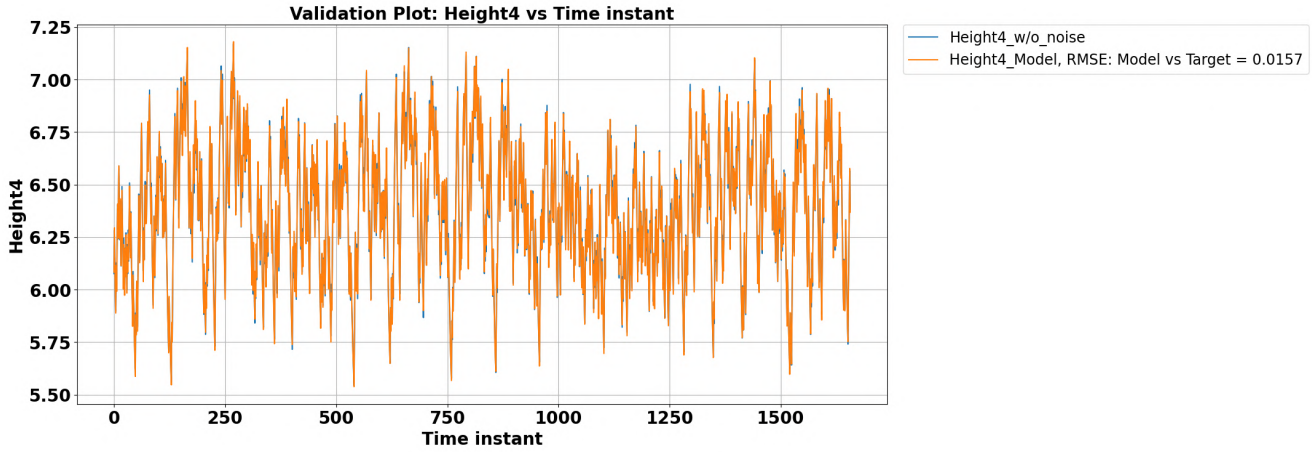
(a) Validation: Height 1 vs Time instant



(b) Validation: Height 2 vs Time instant



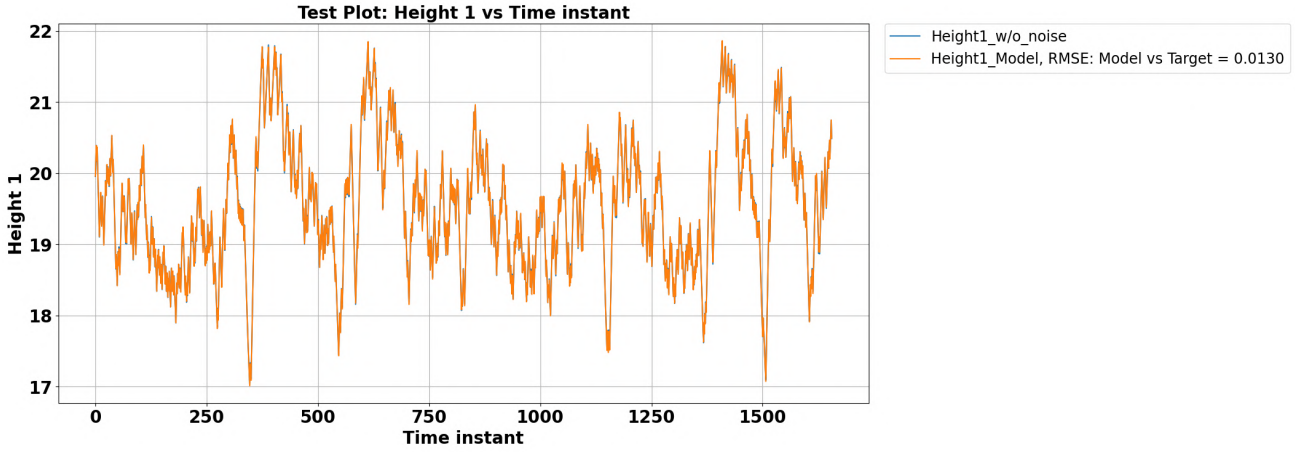
(c) Validation: Height 3 vs Time instant



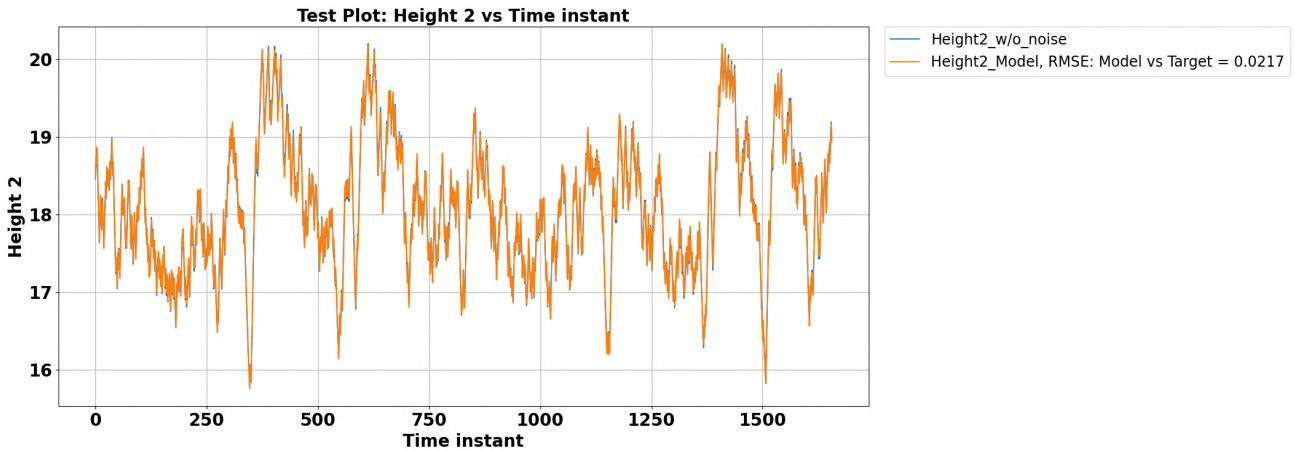
(d) Validation: Height 4 vs Time instant

Figure 5.17: ReLU RNNEKF Validation: States of quadruple tank vs Time instant

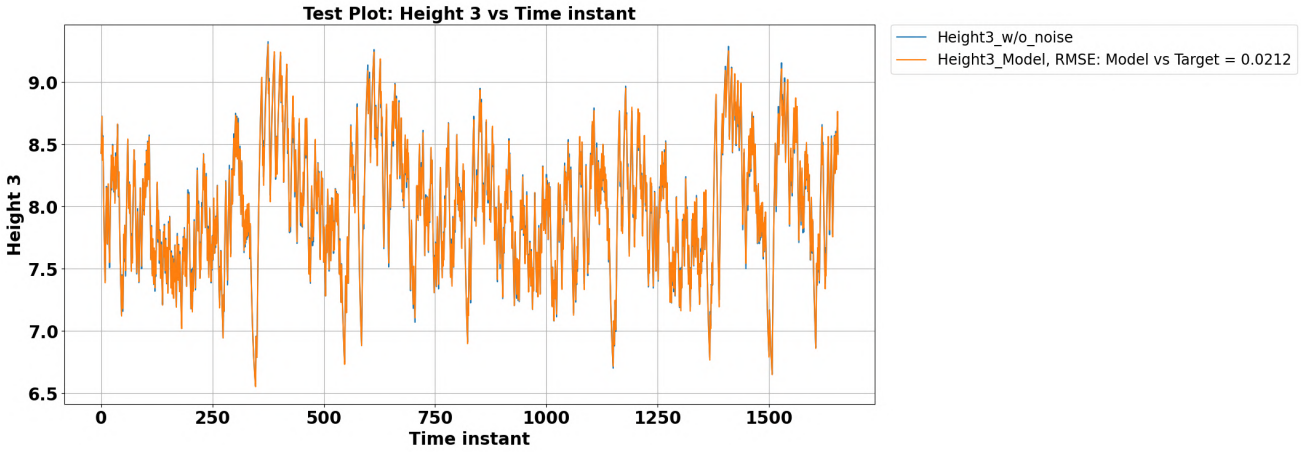
Results on the Test set



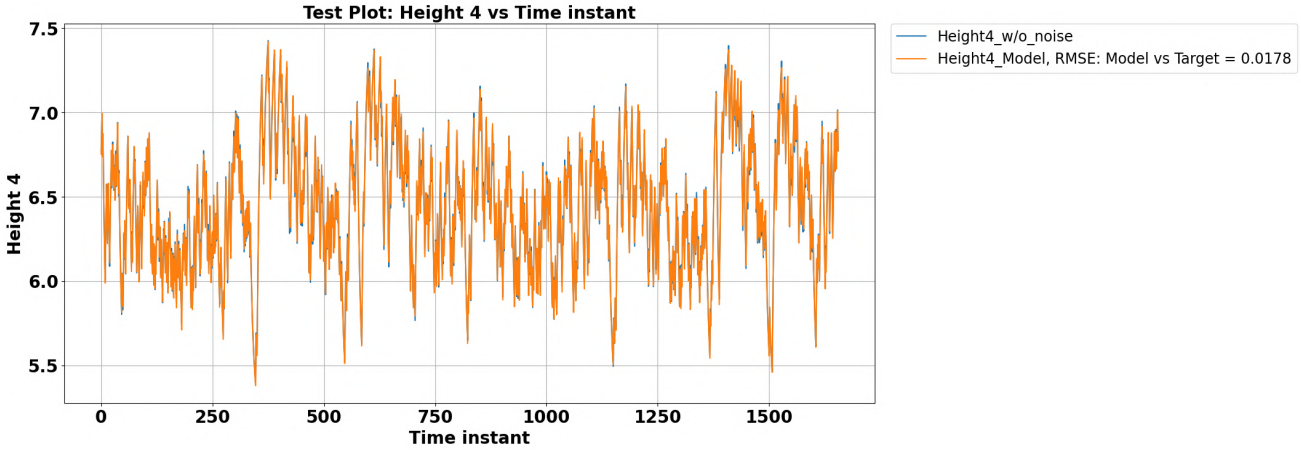
(a) Test: Height 1 vs Time instant



(b) Test: Height 2 vs Time instant



(c) Test: Height 3 vs Time instant



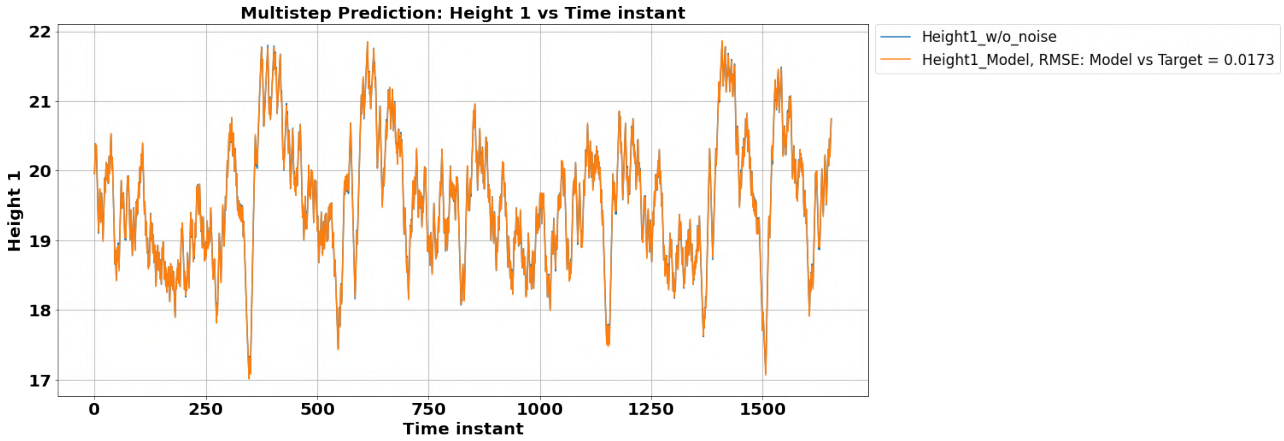
(d) Test: Height 4 vs Time instant

Figure 5.18: ReLU RNNEKF Test: States of quadruple tank vs Time instant

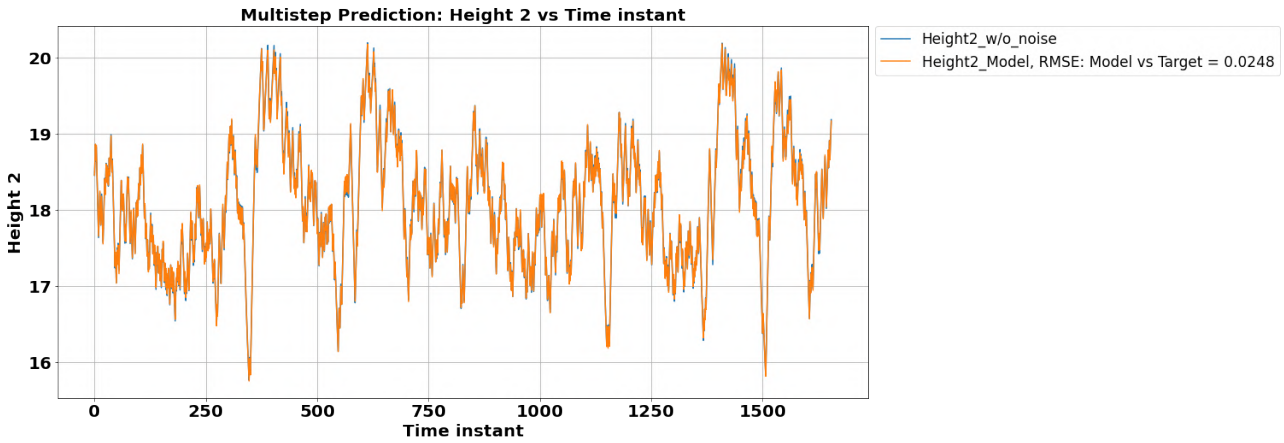
Fig (5.16) - (5.18) shows the plots for the states of the system vs time instant for the training, validation and test data set respectively.

Multi step predictions for quadruple tank-RNN

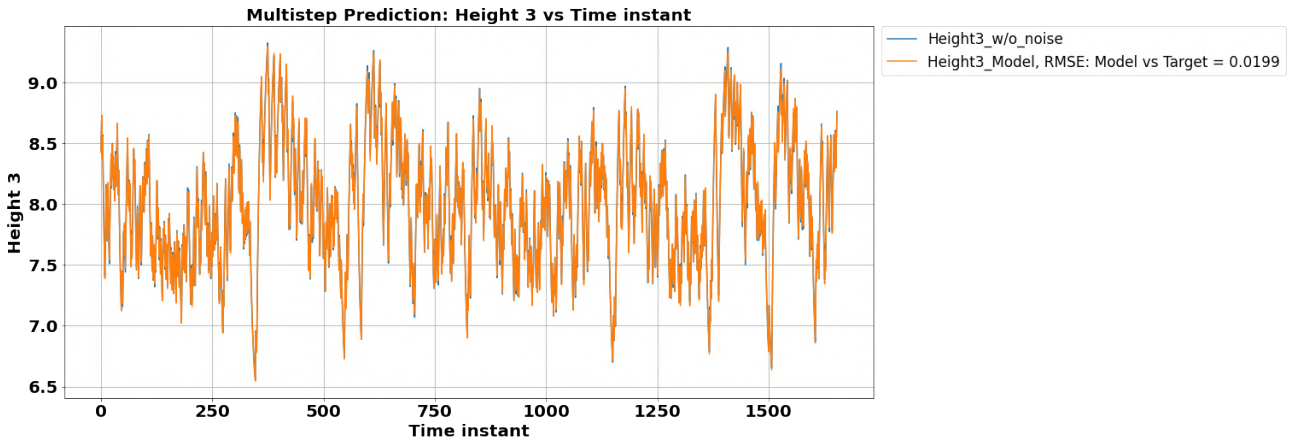
After training the network using series-parallel mode now, the network was tested for multi-step prediction. The output of the previous instant is directly fed as input for the prediction of the next instance. Thus, this will help in giving predictions purely on estimated values done by the network. Therefore, tests data set was used to evaluate the model's performance for the multi-step prediction paradigm. Results were as follows:



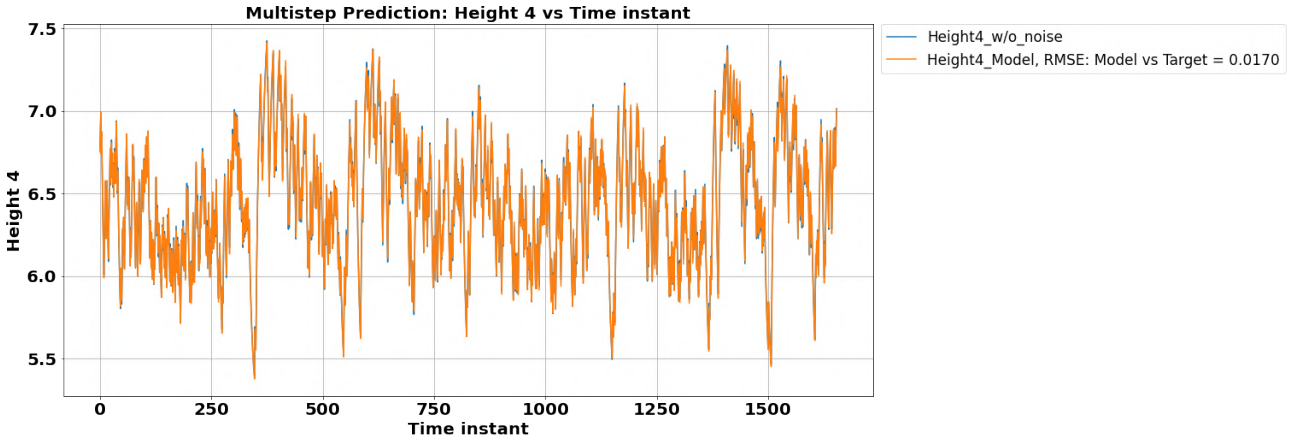
(a) Multistep: Height 1 vs Time instant



(b) Multistep: Height 2 vs Time instant



(c) Multistep: Height 3 vs Time instant



(d) Multistep: Height 4 vs Time instant

Figure 5.19: ReLU RNNEKF Multistep Test: States of quadruple tank vs Time instant

As we can see from Fig.5.19, the RNN model of the quadruple tank has captured the physics of the system well. Hence, the model having parameters as shown in Table 5.7 can now be combined with EKF to give filtered values of the states.

Table 5.7: Summary of RNN model of Quadruple Tank

T	**Learning rate**	**Neurons in each hidden layer**	**Number of hidden layers**	**Epochs**	**Loss Train**	**Loss Test**
10	0.0001	10	2	21000	0.0049	0.0043

States estimated using ReLU RNNEKF strategy for Quadruple Tank

Using the model summarized in Table 5.7, filtered values of the states of the system are estimated using EKF as an estimator. The values of Q, R, C, P(0|0) has been taken from Mukherjee (2020) and has been transformed due to the addition of augmented states as discussed in section 4.1.2 Following initial values of the states, and covariance matrix were taken for the estimation:

$$\boldsymbol{x(0) = \hat{x}(0) = \begin{bmatrix} 19.4255 \\ 17.9628 \\ 7.9311 \\ 6.4053 \end{bmatrix}_{4x1}} \tag{5.8a}$$

$$P(0|0) = \begin{bmatrix} 0.1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0.1 & \ddots & \ddots & & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & & \ddots & \ddots & 0.1 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0.1 \end{bmatrix}_{24x24} \tag{5.8b}$$

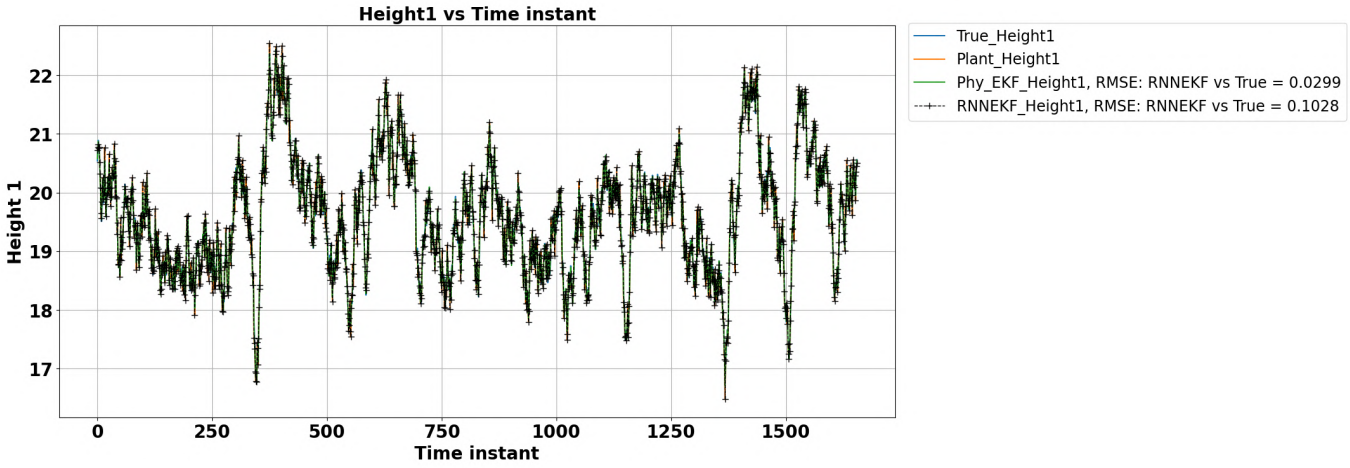
$$C = \begin{bmatrix} 0 & 0 & \cdots & 1 & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 & 0 \end{bmatrix}_{2x24} \tag{5.8c}$$

Plant system model and measurement are incorporated with white Gaussian noise with zero mean and covariance matrix Q and R are:

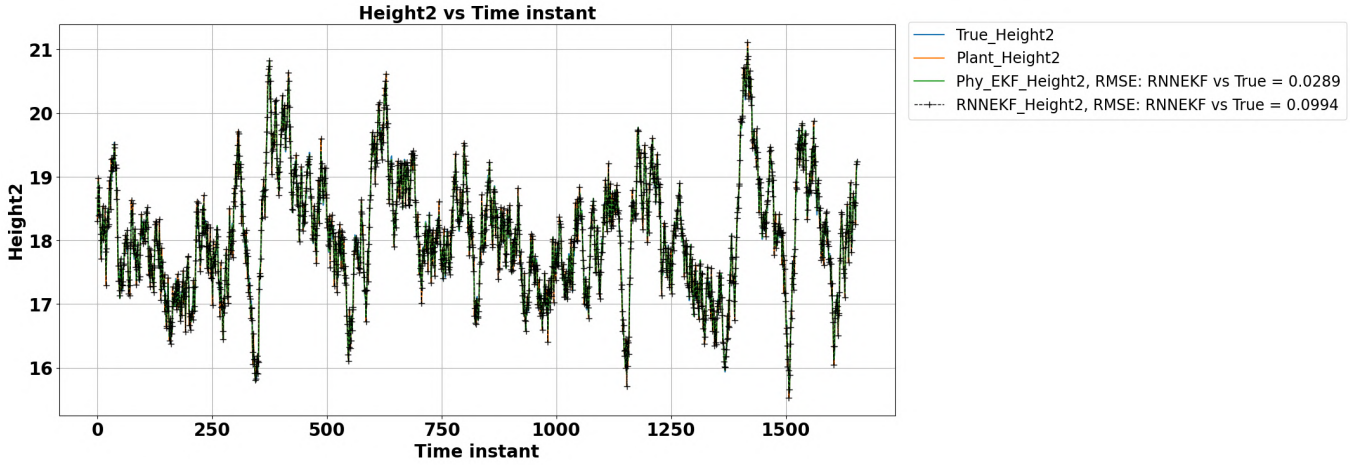
$$Q = \begin{bmatrix} 0.01 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0.01 & \ddots & \ddots & & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & & \ddots & \ddots & 0.01 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0.01 \end{bmatrix}_{24x24} \tag{5.9a}$$

$$R = \begin{bmatrix} 0.0001 & 0 \\ 0 & 0.0001 \end{bmatrix}_{2x2} \tag{5.9b}$$

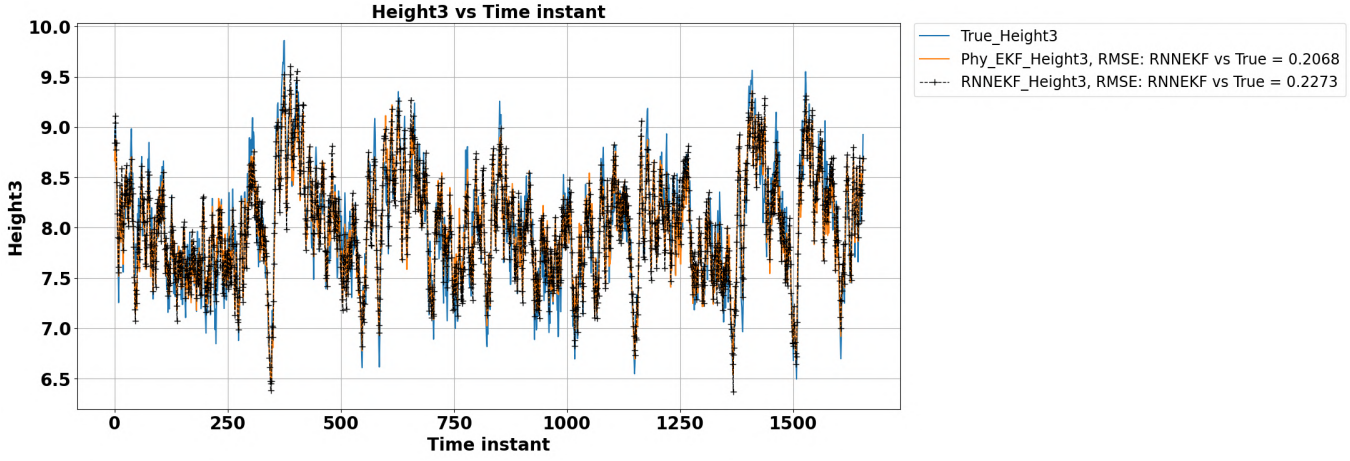
As discussed in section 4.1.2, the state-space model of CSTR has been generated from RNN. As shown in Appendix B, we can similarly derive $\mathbf{\Phi}$ for the quadruple tank system. Thus EKF was applied with the RNN model, and the results of combined RNN with EKF has been reported below:



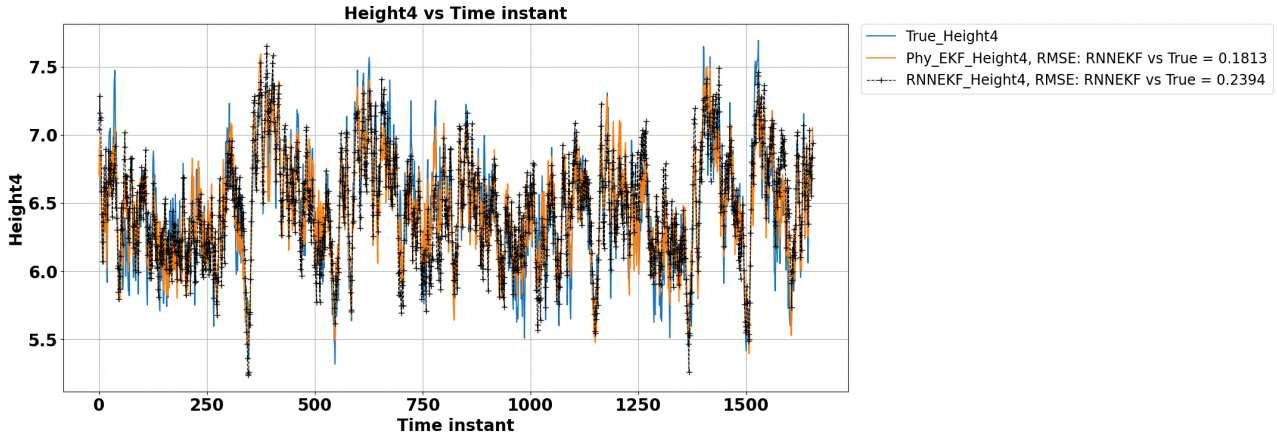
(a) ReLU RNNEKF: Height 1 vs Time



(b) ReLU RNNEKF: Height 2 vs Time



(c) ReLU RNNEKF: Height 3 vs Time



(d) ReLU RNNEKF: : Height 4 vs Time

Figure 5.20: Estimated states using ReLU RNNEKF for quadruple tank

Fig.5.20 gives the filtered values of the states from the EKF, which was derived from the RNN itself. Similar results like ReLU-RNNEKF of CSTR were seen for the quadruple tank system. ReLU RNNEKF was able to estimate the trajectory of the states with low RMSE values. As the ReLU activation function has a point of non-differentiability at x=0, this urges us to explore more activation functions. Hence, in the next section, Tanh-RNNEKF results have been reported, and the RMSE of both the strategies has been compared.

5.2.2 Tanh-RNNEKF on quadruple tank

Tanh-RNNEKF's work is similar to that of ReLU-RNNEKF and is explained in detail in section 4.1. This section uses this strategy to solve the state estimation problem of the quadruple tank system, and the results are reported below.

Modelling of quadruple tank using RNN with tanh as activation function

The training and other procedure of hyperparameter tuning of the network are the same as discussed in section 5.2.1. The results of open-loop simulation done in MATLAB for generating data have been shown in Fig. 5.13 and 5.14. The data is first standardized and then used to train the RNN. Following were the parameters which were used after hyperparameter tuning of the network for the prediction of the states:

Table 5.8: Tuned hyperparameters for quadruple tank tanh-RNN

T	Learning rate	Neurons in each hidden layer	Number of hidden layers	Epochs
10	0.0001	10	2	35000

T represents the length of the time sequence or the window size used to predict the current instant values. Using the above hyper-parameter values, RNN was trained using a training dataset consisting of 13333 data points for 35k epochs.

Loss plots for Quadruple Tank Tanh-RNN

The values of loss function with respect to epochs is shown in Fig. 5.23. Further due to high range of loss values, Fig. 5.21 and 5.22 has been plotted which represents the loss values for < 900 epochs and > 900 epochs respectively.

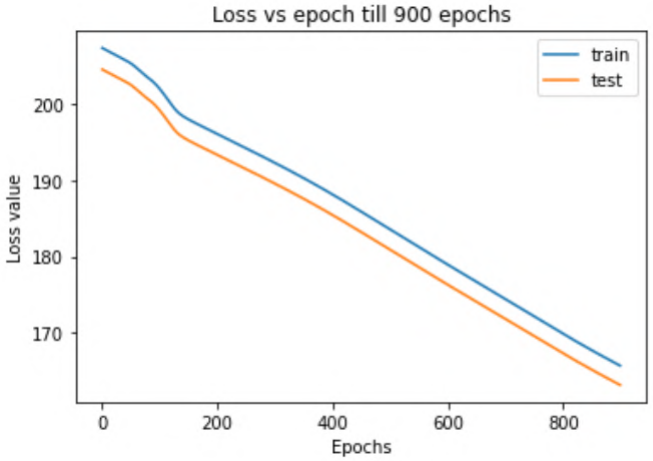


Figure 5.21: Tanh RNNEKF: Loss vs Epochs

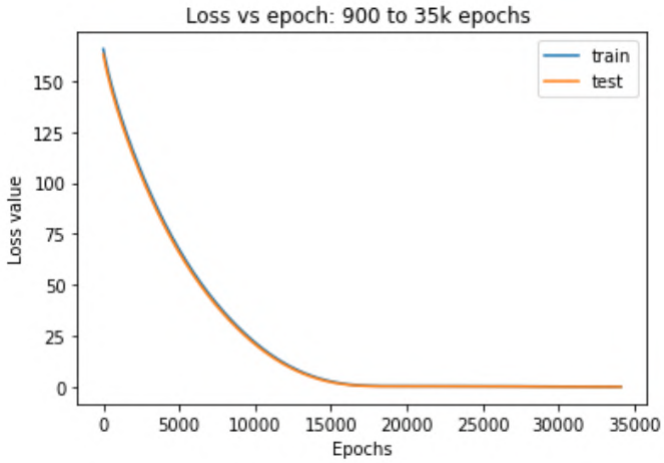


Figure 5.22: Tanh RNNEKF: Loss vs Epochs

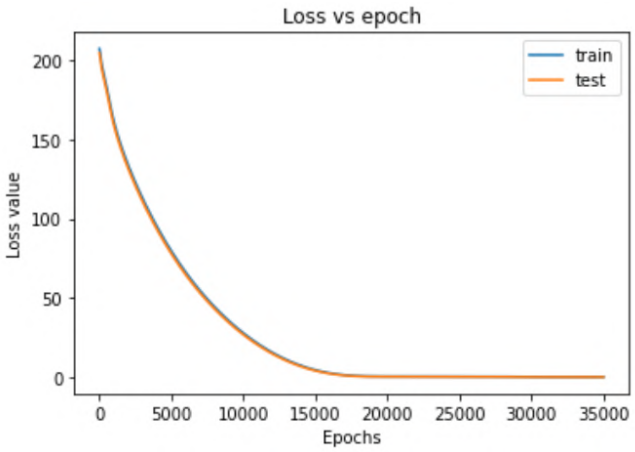
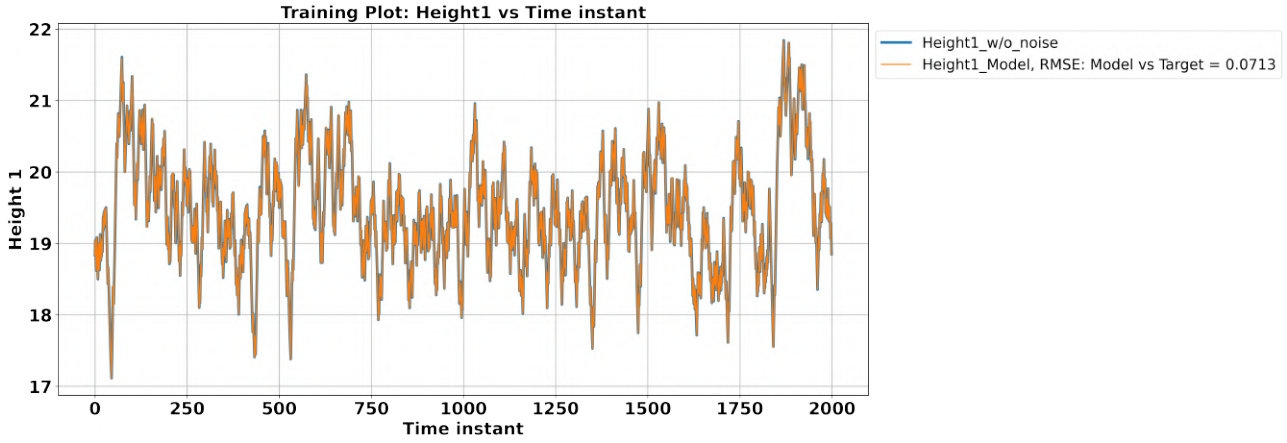


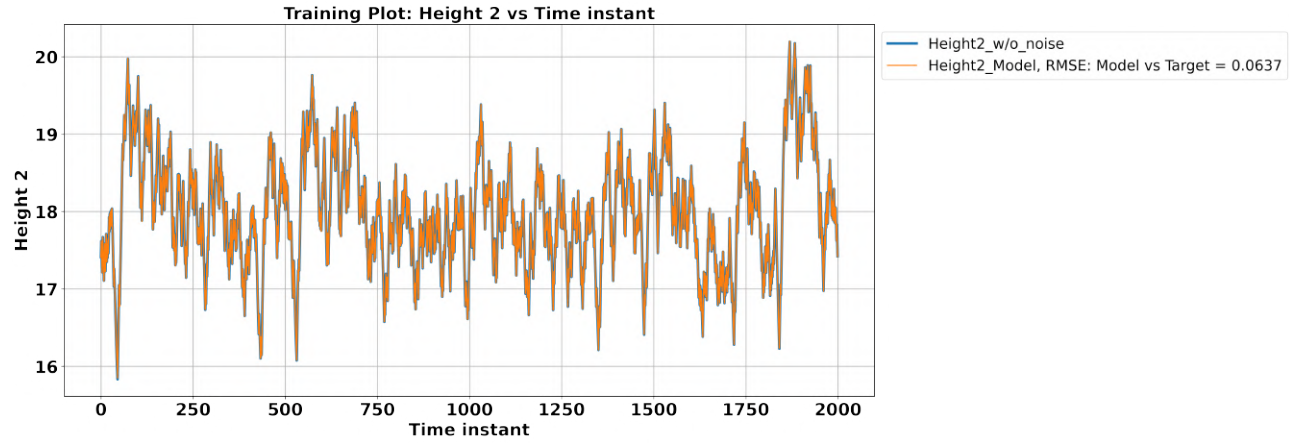
Figure 5.23: Tanh RNNEKF: Loss vs Epochs

One step predictions for quadruple tank-RNN

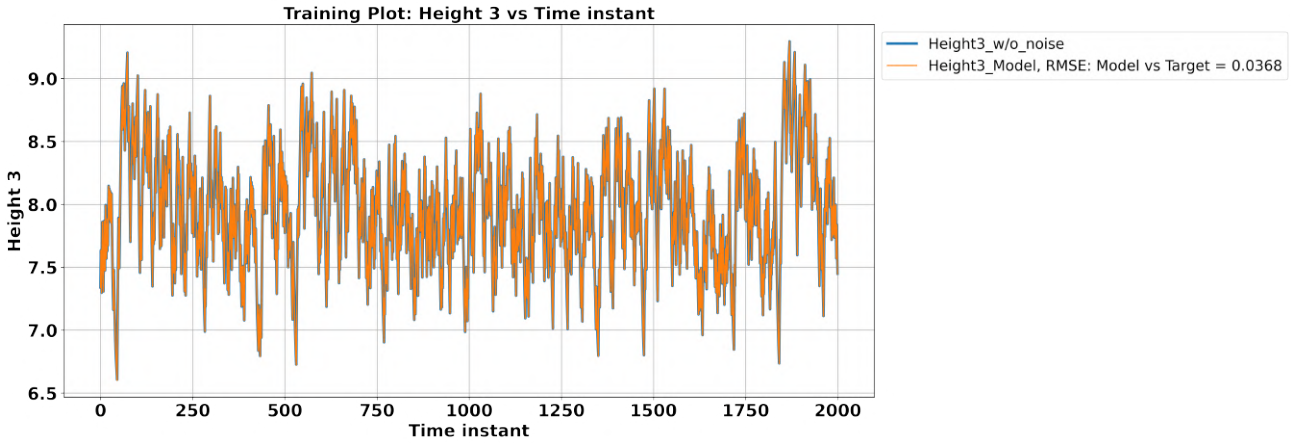
Following were the mapping done by the RNN for one step prediction of the training dataset of the states:



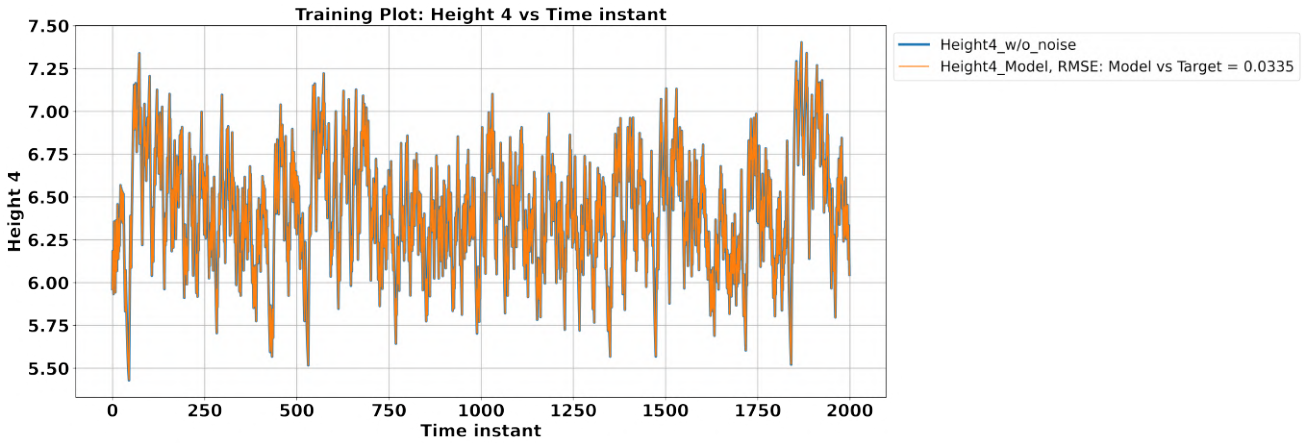
(a) Train: Height 1 vs Time instant



(b) Train: Height 2 vs Time instant



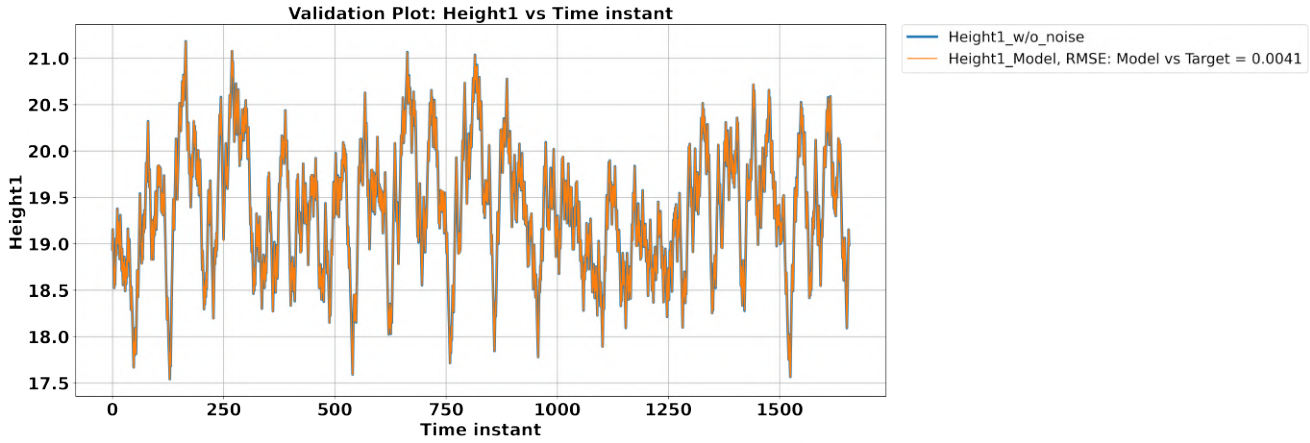
(c) Train: Height 3 vs Time instant



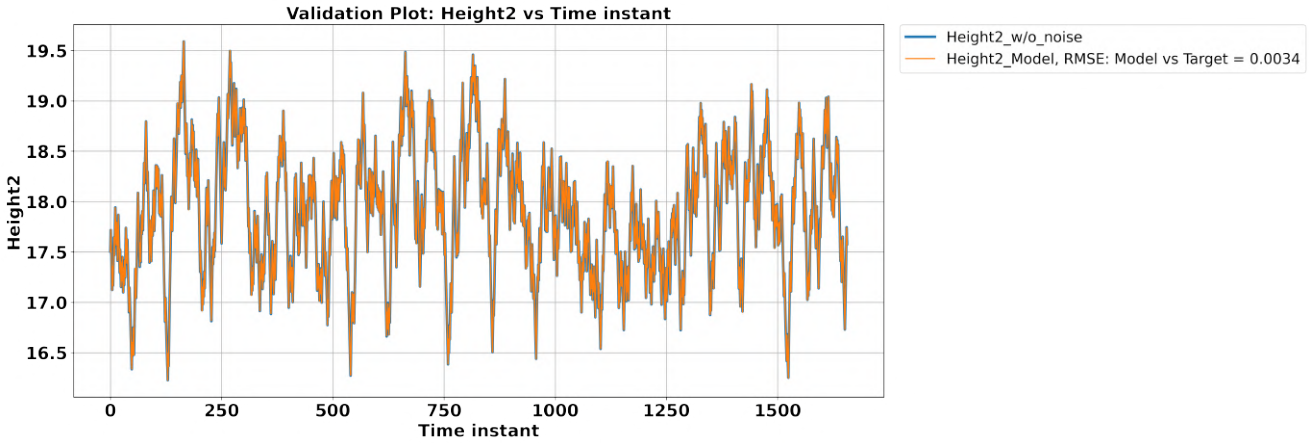
(d) Train: Height 4 vs Time instant

Figure 5.24: Tanh RNNEKF Train: States of quadruple tank vs Time

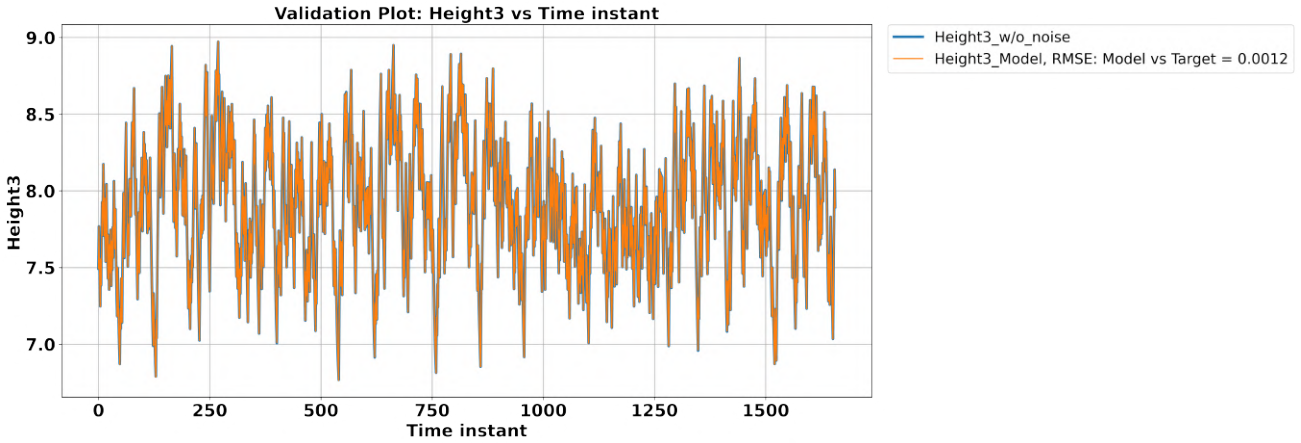
Results on the validation set



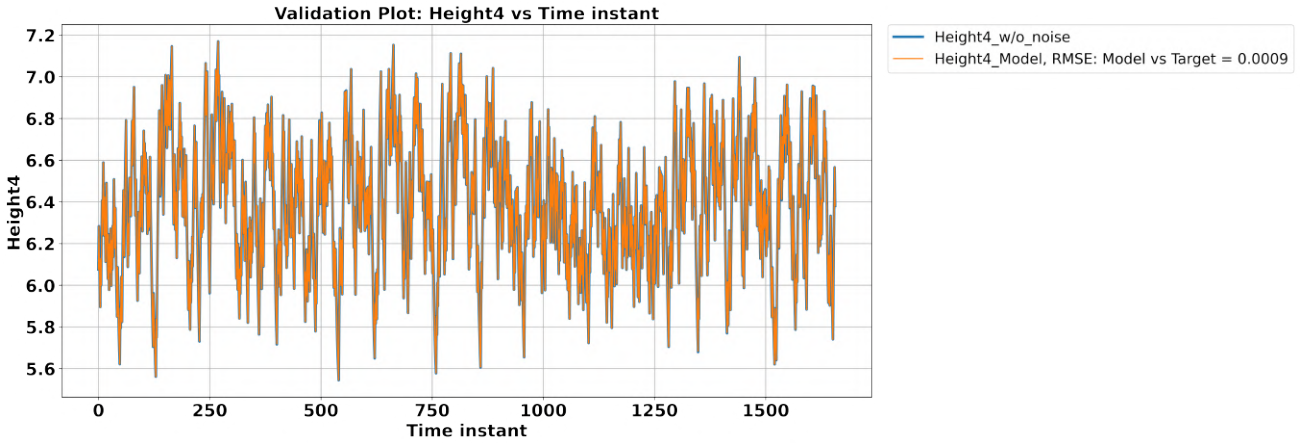
(a) Validation: Height 1 vs Time instant



(b) Validation: Height 2 vs Time instant



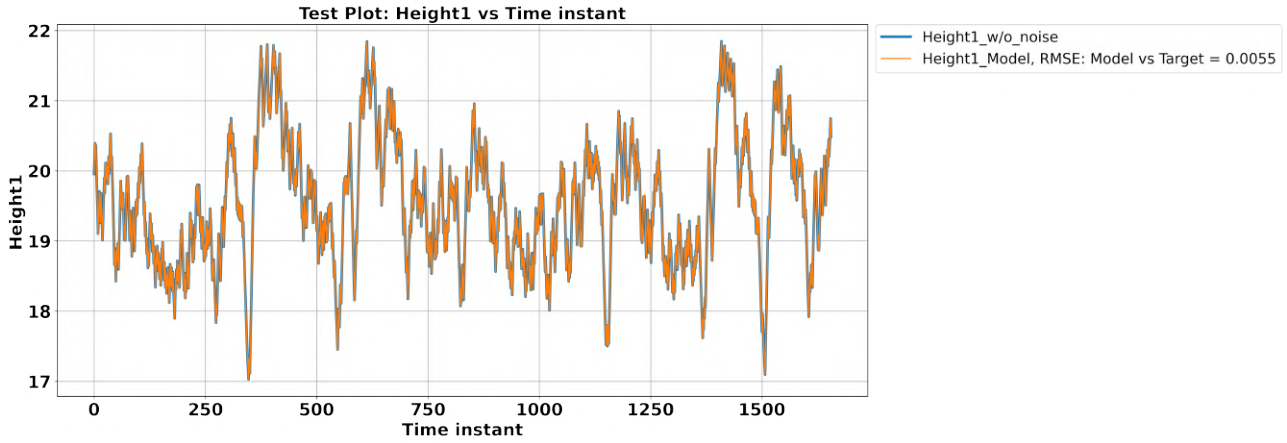
(c) Validation: Height 3 vs Time instant



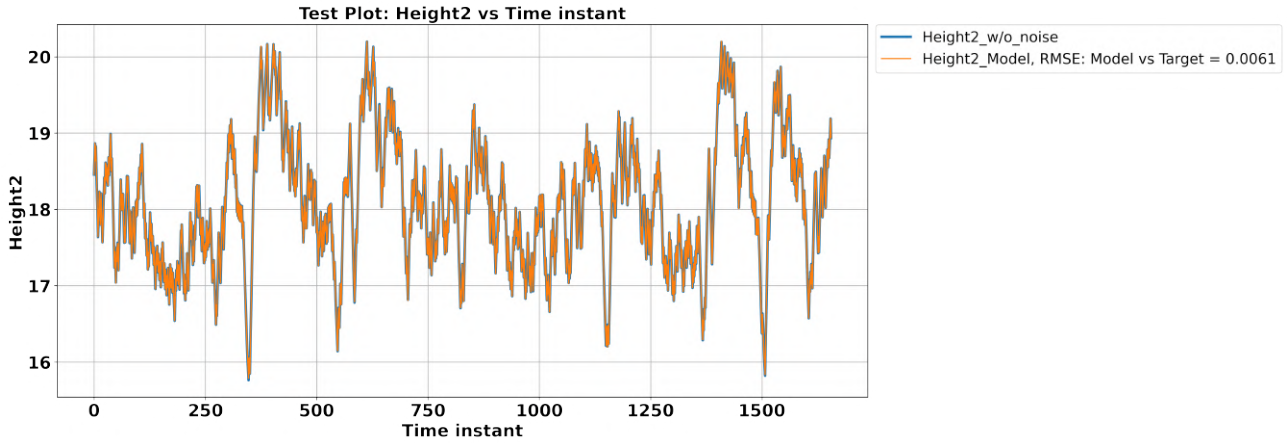
(d) Validation: Height 4 vs Time instant

Figure 5.25: Tanh RNNEKF Validation: States of quadruple tank vs Time instant

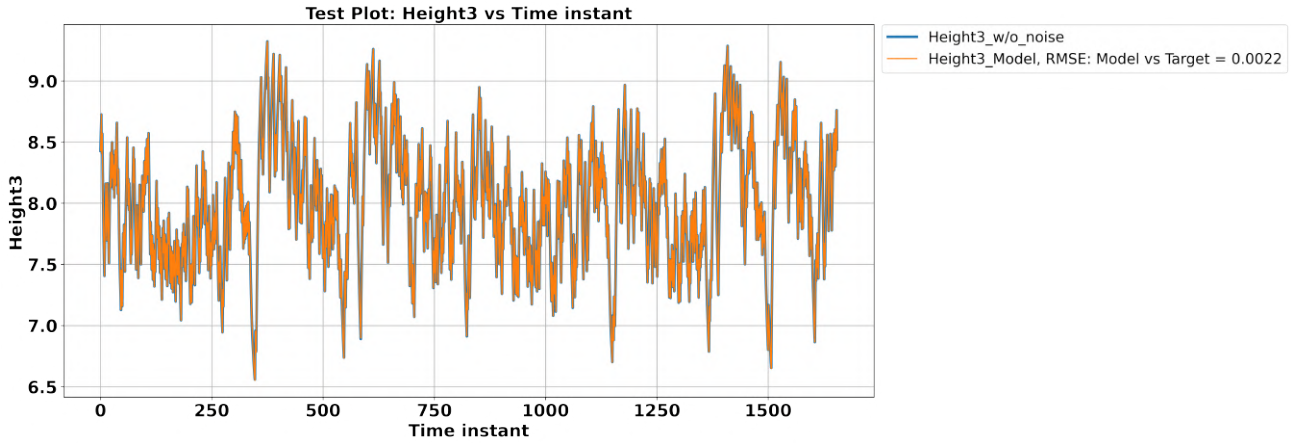
Results on the Test set



(a) Test: Height 1 vs Time instant

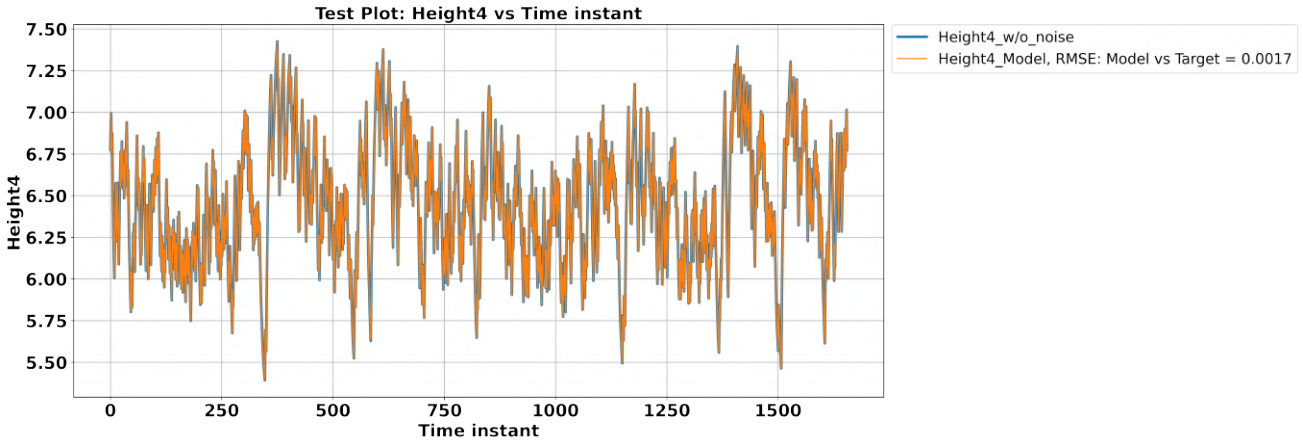


(b) Test: Height 2 vs Time instant



(c) Test: Height 3 vs Time instant

Fig (5.24) - (5.26) shows the plots for the states of the system vs Time instant for the training, validation and test data set respectively.

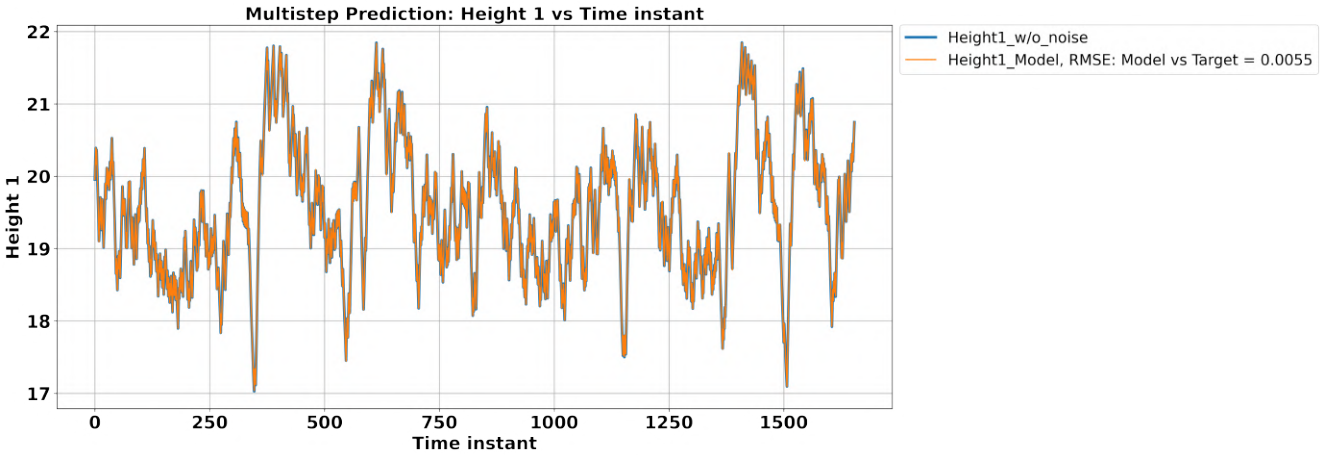


(d) Test: Height 4 vs Time instant

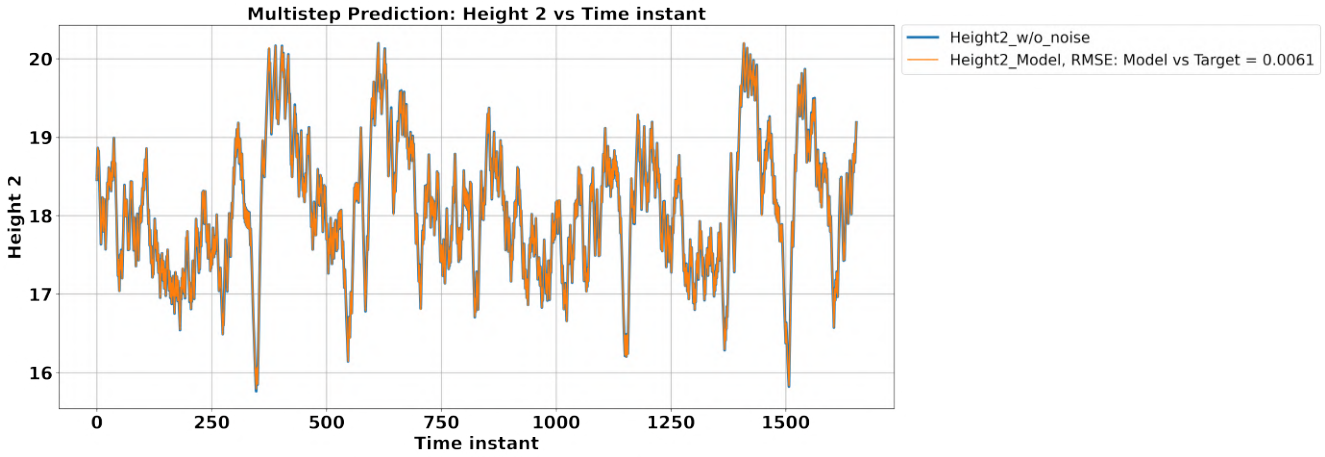
Figure 5.26: Tanh RNNEKF Test: States of quadruple tank vs Time instant

Multi step predictions for Quadruple Tank-RNN

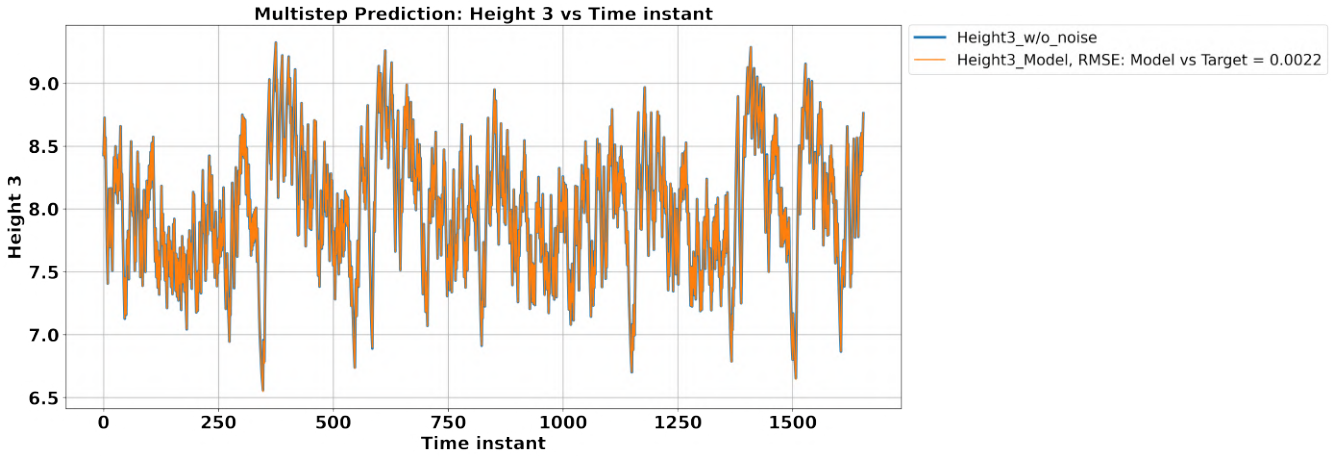
After training the network using series-parallel mode now, the network was tested for multi-step prediction. The output of the previous instant is directly fed as input for the prediction of the next instance. Thus, this will help in giving predictions purely on estimated values done by the network. Therefore, tests data set was used to evaluate the model's performance for the multi-step prediction paradigm. Results were as follows:



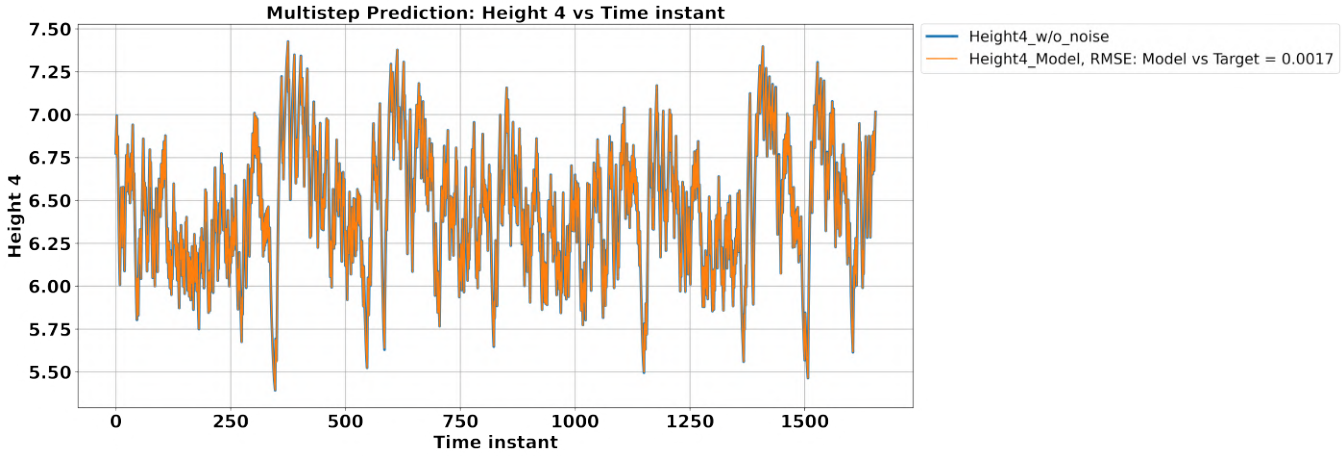
(a) Multistep Test: Height 1 vs Time instant



(b) Multistep Test: Height 2 vs Time instant



(c) Multistep Test: Height 3 vs Time instant



(d) Multistep Test: Height 4 vs Time instant

Figure 5.27: Tanh RNNEKF Multistep Test: States of quadruple tank vs Time instant

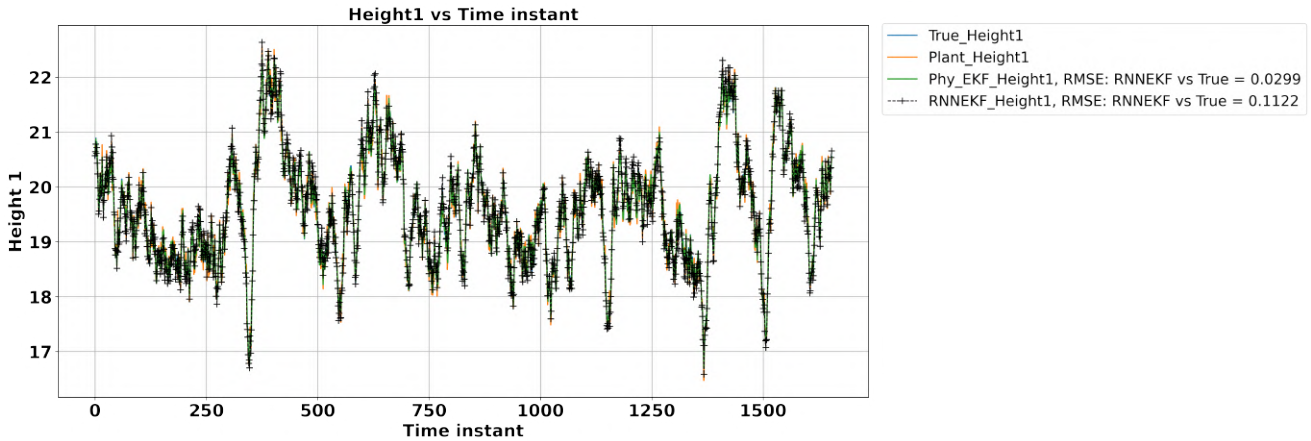
As we can see from the Fig.5.27, the RNN model of the quadruple tank has captured the physics of the system well. Hence, the model having parameters as shown in Table 5.9 can now be combined with EKF to give filtered values of the states.

Table 5.9: Summary of Tanh RNN model of Quadruple Tank

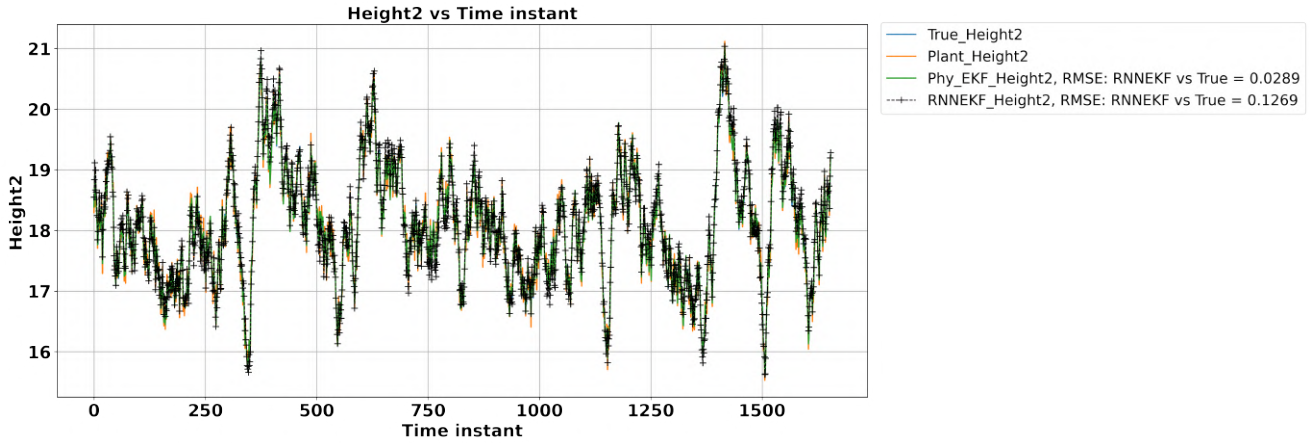
T	Learning rate	Neurons in each hidden layer	Number of hidden layers	Epochs	Loss Train	Loss Test
10	0.0001	10	2	21000	0.00189	0.00114

States estimated using Tanh RNNEKF strategy for Quadruple Tank

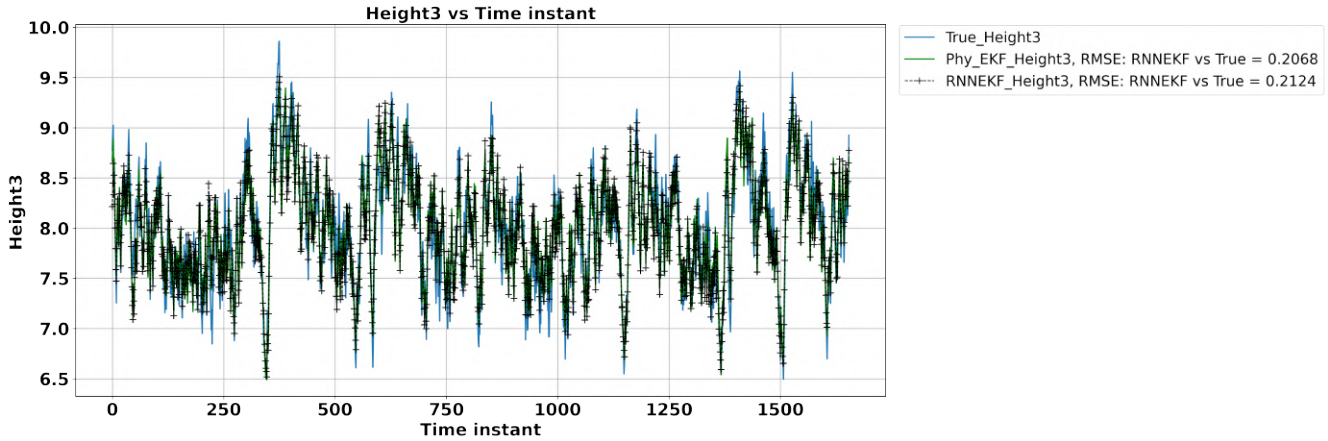
Using the model summarized in Table 5.9, filtered values of the states of the system are estimated using EKF as an estimator. As the model is in the standardized subspace, the EKF has also been converted to normalized subspace. The detailed derivation of converting the model parameters like Q, R, C has been derived in Appendix C and was used by EKF to give an estimate of the states. Moreover, as discussed in section 4.1.2, the state-space model of the quadruple tank has been generated from RNN. As shown in Appendix B, we can similarly derive $\mathbf{\Phi}$ for the quadruple tank system. Thus EKF was applied with the RNN model, and the results of combined RNN with EKF is reported below:



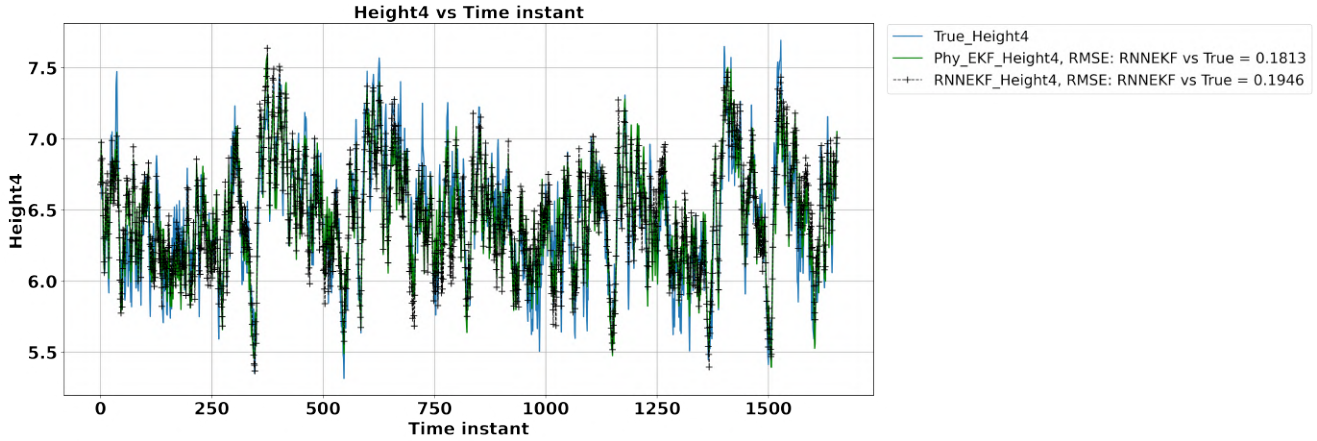
(a) Tanh RNNEKF quadruple tank: Height 1 vs Time instant



(b) Tanh RNNEKF quadruple tank: Height 2 vs Time instant



(c) Tanh RNNEKF quadruple tank: Height 3 vs Time instant



(d) Tanh RNNEKF quadruple tank: Height 4 vs Time instant

Figure 5.28: Estimated states using Tanh RNNEKF for Quadruple Tank

Fig.5.28 gives the values of filtered values of states from the EKF, which was derived from the Tanh RNN. From the figure, it can be concluded that the Tanh RNNEKF technique can capture the dynamics and able to estimate the states of the quadruple tank system with low RMSE. RMSE obtained from both of the proposed works ReLU RNNEKF and Tanh RNNEKF with conventional physics-based EKF for the quadruple tank has been reported in Table 5.10. It was found that the RMSE obtained from the proposed strategy was similar to that of physics-based EKF. Moreover, the computation time taken by ReLU-RNNEKF was found to be **4*s***, which is **70.6%** less than the computation taken by conventional EKF. At the same time, the computation time taken by the Tanh-RNNEKF was similar to that of the traditional method. Hence with the provided advantage of less time consumption of modeling and expense, data-driven techniques like RNNEKF can solve the state estimation problem. A case study of the Temperature control system has been discussed in the next section.

Table 5.10: Summary of RNNEKF on quadruple tank

States	Proposed Work	RMSE	Computation Time(s)
Height 1	ReLU RNNEKF	0.1028	4.003
Height 2	ReLU RNNEKF	0.0994	4.003
Height 3	ReLU RNNEKF	0.2273	4.003
Height 4	ReLU RNNEKF	0.2394	4.003
Height 1	Tanh RNNEKF	0.1122	19.87
Height 2	Tanh RNNEKF	0.1269	19.87
Height 3	Tanh RNNEKF	0.2124	19.87
Height 4	Tanh RNNEKF	0.1946	19.87
Height 1	Physics EKF	0.0299	13.607
Height 2	Physics EKF	0.0289	13.607
Height 3	Physics EKF	0.2068	13.607
Height 4	Physics EKF	0.1813	13.607

5.3 Temperature Control Lab

The temperature control lab (TCL) is an application of feedback control with an Arduino, an LED, two temperature sensors, and two heaters. Desired temperature set point is achieved by adjusting the heater power output. Thermal energy from the heater is transferred by conduction, convection, and radiation to the temperature sensor. Heat is also exchanged with the surroundings.

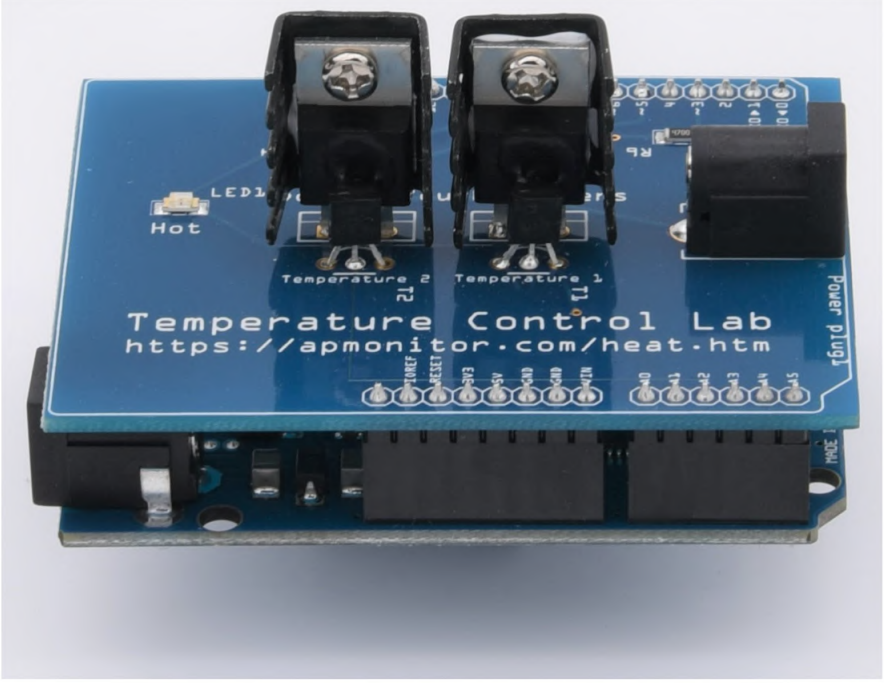


Figure 5.29: Temperature control lab equipment

The physics based differential equation for TCL apmonitor (2020) can be written as follows:

$$Q_{C12} = UA_s\,(T_2 - T_1) \tag{5.10a}$$

$$Q_{R12} = \epsilon\sigma A\left(T_2^4 - T_1^4\right) \tag{5.10b}$$

$$mc_p\frac{dT_1}{dt} = UA\,(T_\infty - T_1) + \epsilon\sigma A\left(T_\infty^4 - T_1^4\right) + Q_{C12} + Q_{R12} + \alpha_1 Q_1 \tag{5.10c}$$

$$mc_p\frac{dT_2}{dt} = UA\,(T_\infty - T_2) + \epsilon\sigma A\left(T_\infty^4 - T_2^4\right) - Q_{C12} - Q_{R12} + \alpha_2 Q_2 \tag{5.10d}$$

In Eq. (5.10a) - (5.10d) Q_{C12} convection heat exchange between heater 1 and heater 2, Q_{R12} radiation heat exchange between heater 1 and heater 2, U is the overall heat transfer coefficient, A_s is the surface area between heaters, T_1 and T_2 are the respective temperatures of heater 1 and heater 2, ϵ is the emissivity, σ is Stefan Boltzmann constant, m and c_p is the mass and heat capacity of the heater respectively, T_∞ is the ambient temperature, A is the surface area not between heaters, α_1 and α_2 are the heater factors, Q_1 and Q_2 represents output of heater 1 and heater 2 respectively and are the manipulated variable while T_1 and T_2 are the controlled outputs. Sampling time (ts) has been taken as 4 sec. Parameters of the TCL model is given in Table 5.10 Jha (2021). The implementation of the proposed strategy **Tanh-RNNEKF** has been reported in the next section

Table 5.11: TCL parameters

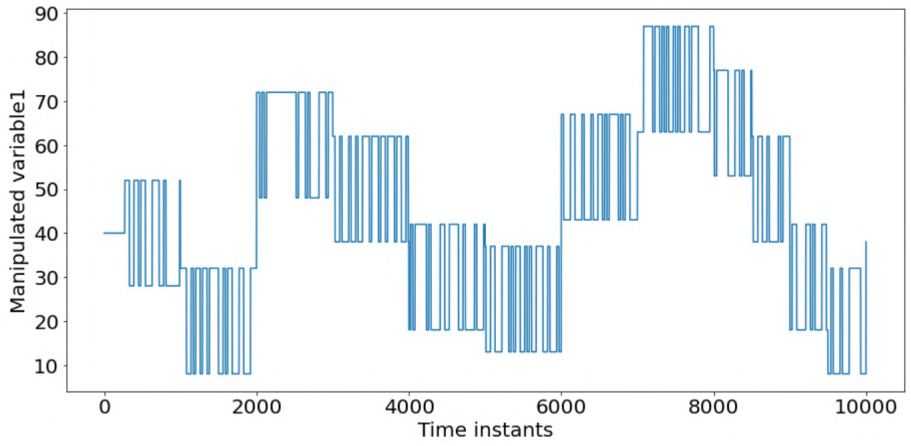
Quantity	*unit*	*Value*
Q_1	W	0 to 1
Q_2	W	0 to 0.75
c_p	J/KgK	500
A	m^2	$1 * 10^{-3}$
A_s	m^2	$2 * 10^{-4}$
m	Kg	0.004
ϵ	-	0.9
σ	W/m^2K^4	$5.67 * 10^{-8}$
U	W/m^2**deg** C	6.7853
α_1	W/(%heater)	0.0050
α_2	W/(%heater)	0.0036

5.3.1 Tanh-RNNEKF on TCL

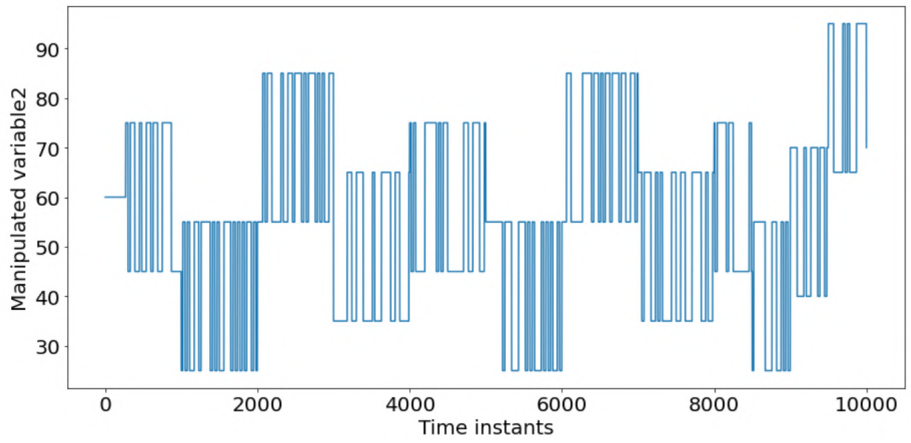
Modelling of the RNN will be done for the TCL and then will be combined with the EKF derived with RNN model. Further a comparitive study of **Tanh-RNNEKF** with first principle based model and EKF results was done and has been discussed below.

Modelling of TCL using RNN

For the training of the RNN, open-loop simulation was carried out to generate datasets of the TCL system. TCL simulator was used for the same Jha (2021). Unlike other systems described in the above sections, in TCL, multiple PRBS signals were introduced into manipulated variables to cover the wide dynamic operating conditions of the system. Total 10k points were generated, out of which 8000 were used for training of the RNN, 1k each was used as validation and test dataset. Below are the graphs of the manipulated input and states obtained from the simulation:



(a) Output heater 1 vs Time instant



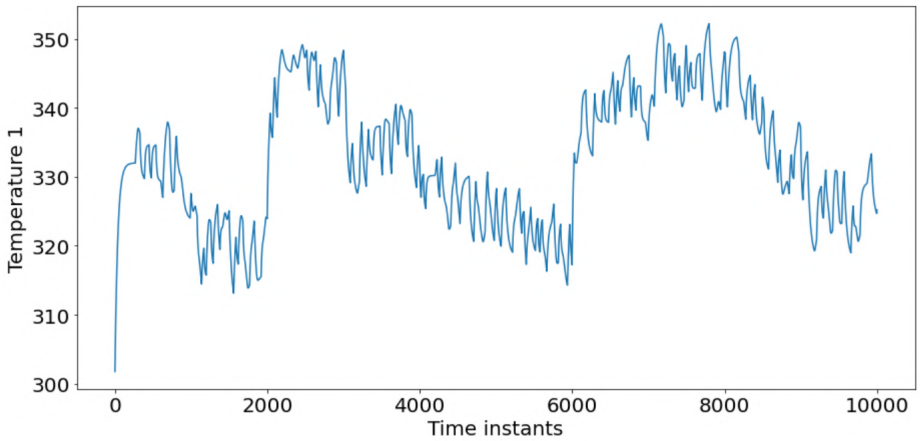
(b) Output heater 2 vs Time instant

Figure 5.30: Manipulated inputs vs Time instant

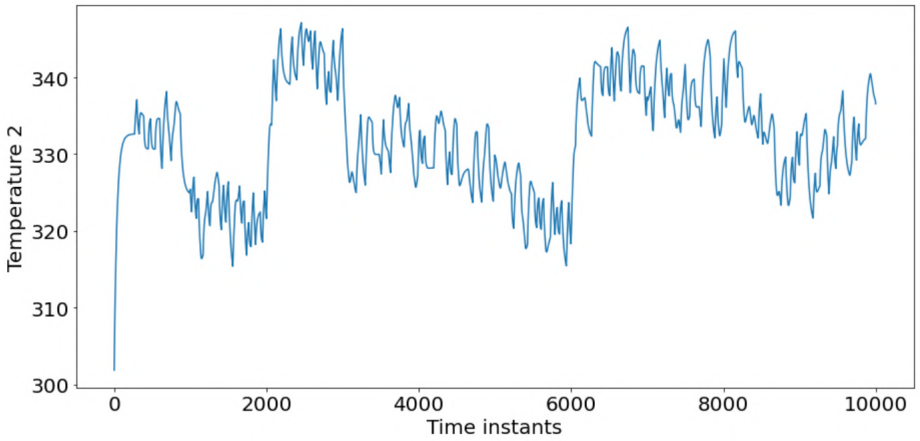
Fig. 5.30 shows the PRBS fluctuations given to manipulated variables to activate the dynamics of the system as shown in Fig. 5.31. Then this dataset was divided into 3 sets:

- Training dataset. (8000 points)
- Validating dataset. (1000 points)
- Test dataset. (1000 points)

With MSE as loss function, training data set, and Adam Kingma and Ba (2014) as the optimizer, the RNN was trained. After hyperparameter tuning of the RNN, the parameter which was used for the prediction of the states are stated in Table 5.12. T represents the length of the time sequence or the window size used to predict the current instant values. Using the above hyperparameter values, RNN was trained using a training dataset consisting of 8000 data points for 25k epochs.



(a) Temperature of heater 1 vs Time



(b) Temperature of heater 2 vs Time

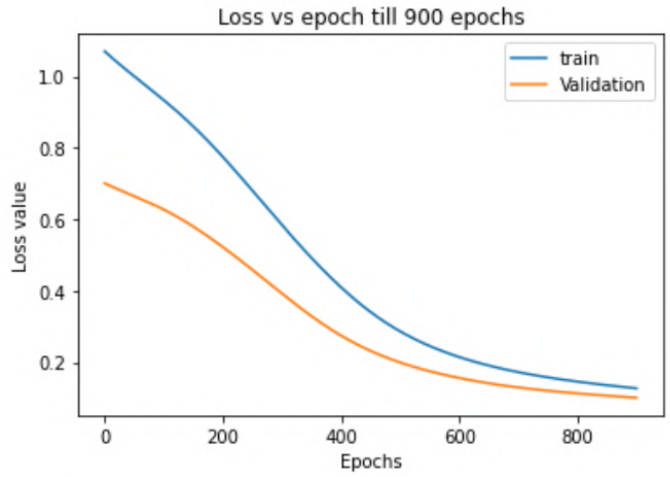
Figure 5.31: States vs Time instants

Table 5.12: Tuned hyperparameters for TCL RNN

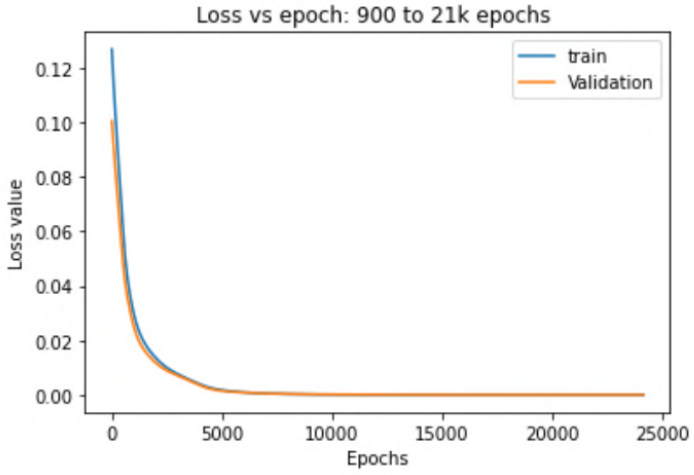
T	Learning rate	Neurons in each hidden layer	Number of hidden layers	Epochs
10	0.0001	10	2	25000

Loss plots for TCL tanh-RNN

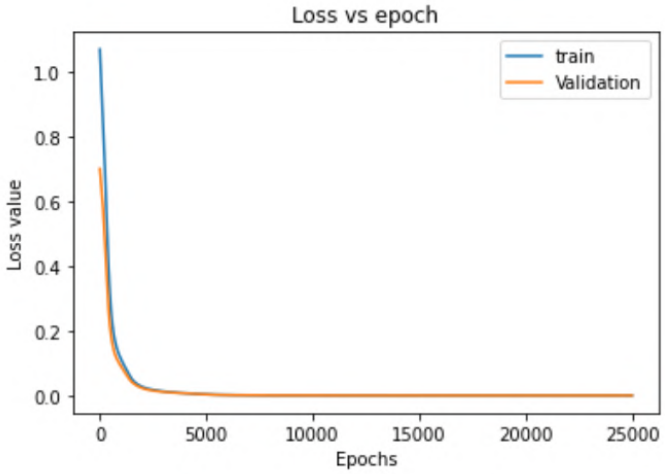
The values of loss function with respect to epochs is shown in Fig. 5.32c. Further due to high range of loss values, Fig. 5.32a and 5.32b has been plotted which represents the loss values for < 900 epochs and > 900 epochs respectively.



(a) Loss vs Epochs



(b) Loss vs Epochs

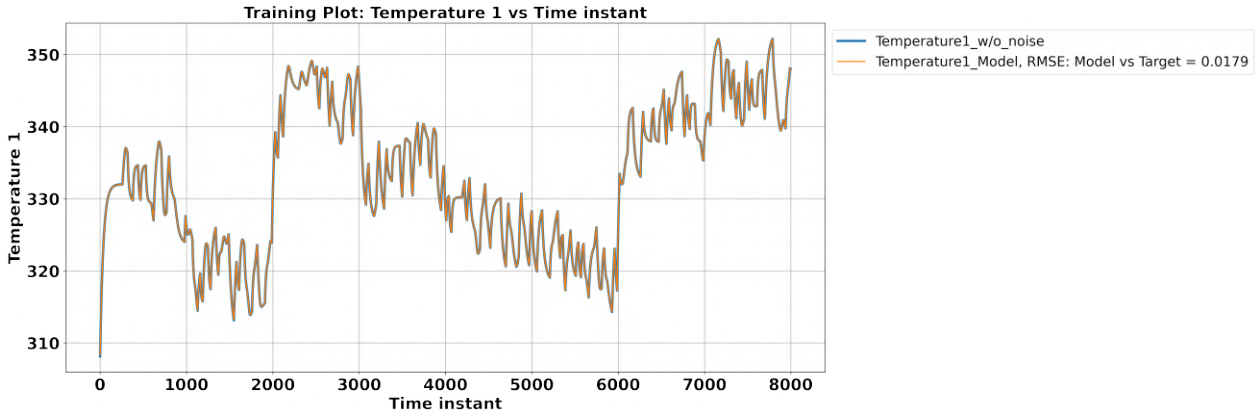


(c) Loss vs Epochs

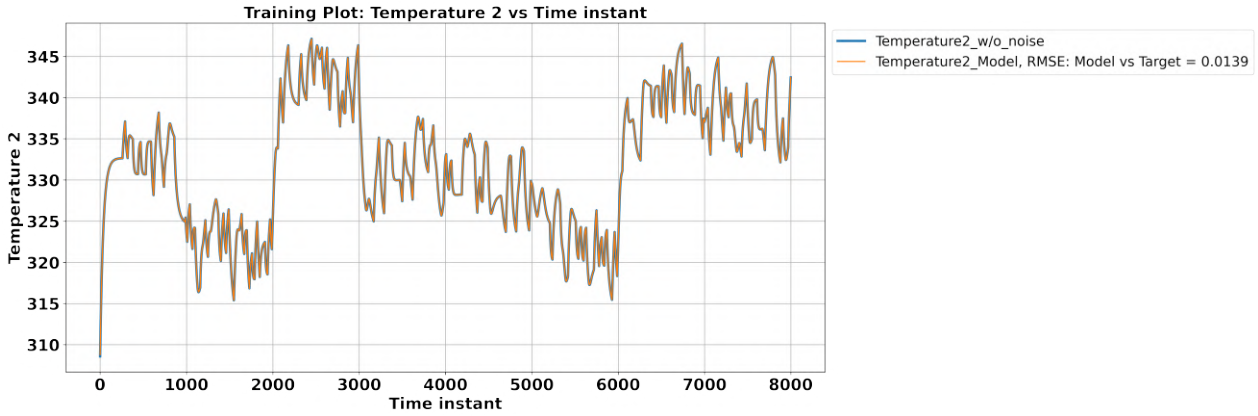
Figure 5.32: TCL Tanh-RNNEKF: Loss vs Epochs

One step predictions for TCL-RNN

Following were the mapping done by the RNN for the training dataset:



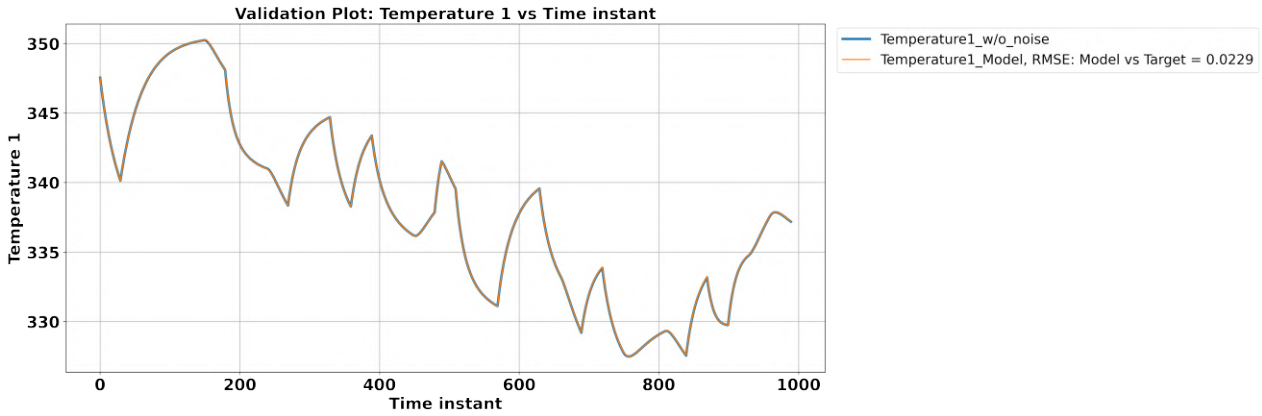
(a) Temperature 1 vs Time Instants



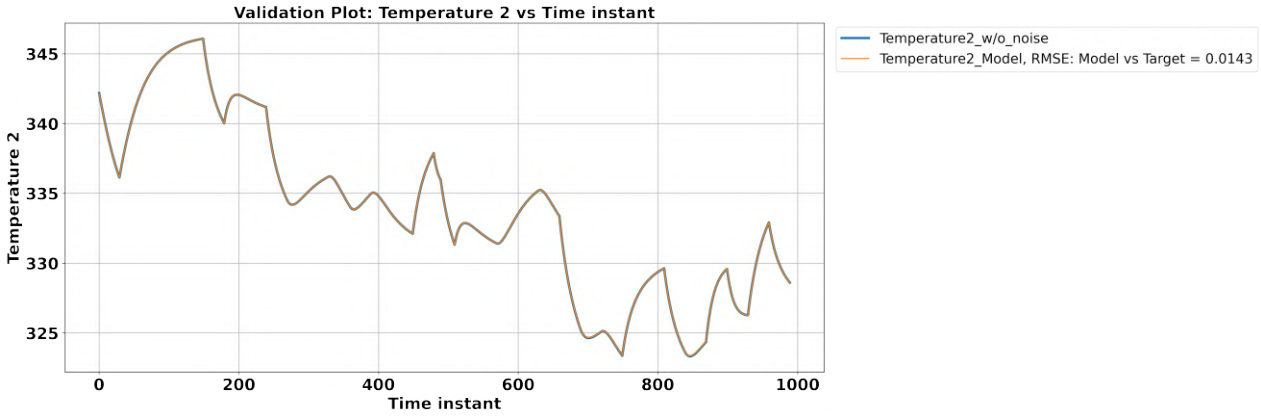
(b) Temperature 2 vs Time Instants

Figure 5.33: One step predictions Training: TCL States vs Time Instants

Results on the validation set



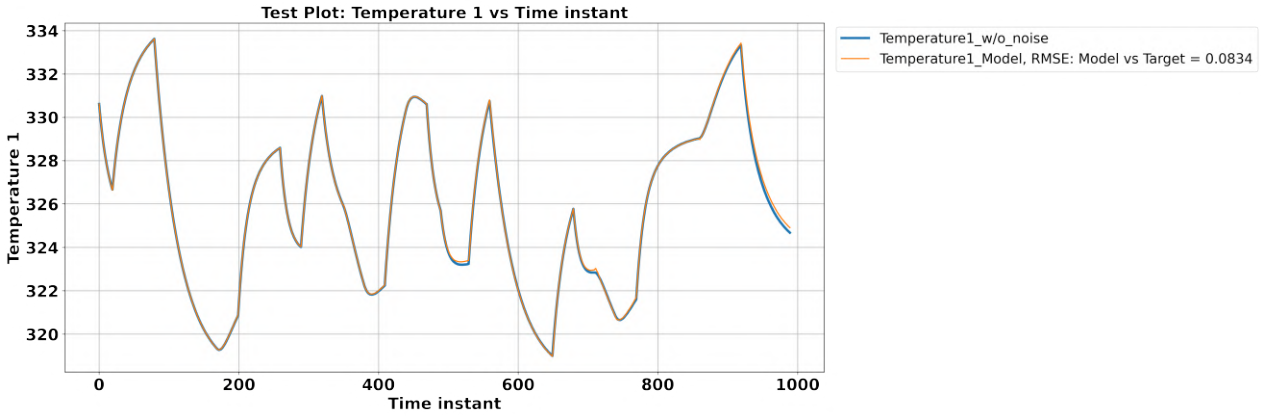
(a) Temperature 1 vs Time Instants



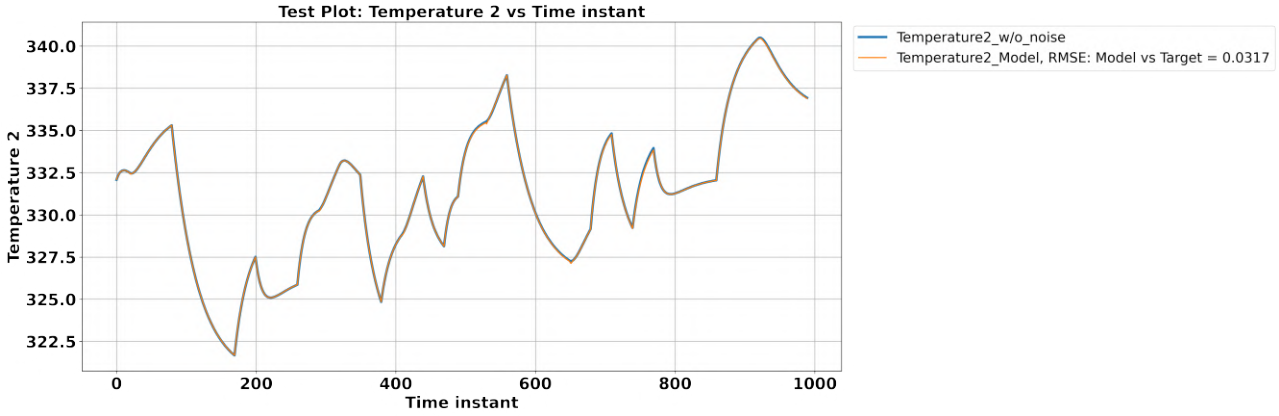
(b) Temperature 2 vs Time Instants

Figure 5.34: One step predictions Validation: TCL States vs Time Instants

Results on the Test set



(a) Temperature 1 vs Time Instants



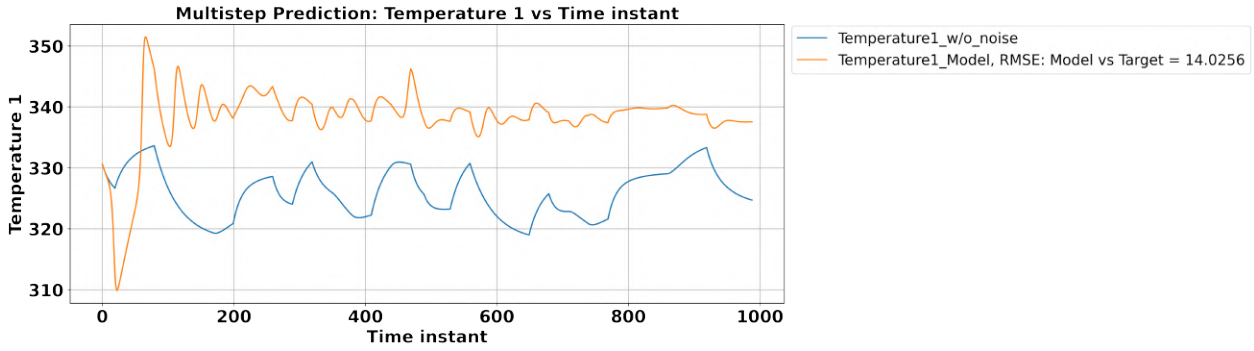
(b) Temperature 2 vs Time Instants

Figure 5.35: One step predictions test: TCL States vs Time Instants

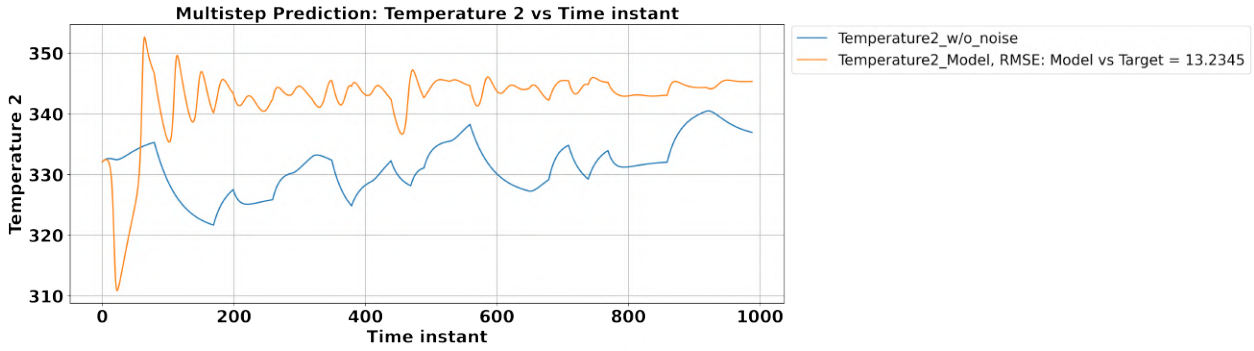
Fig (5.33) - (5.35) shows the plots for the states of the system vs Time for the training, validation and test data set respectively.

Multi step predictions for TC lab-RNN

After training the network using series-parallel mode now, the network was tested for multi-step prediction. The output of the previous instant is directly fed as input for the prediction of the next instance. Thus, this will help in giving predictions purely on estimated values done by the network. Therefore, tests data set was used to evaluate the model's performance for the multi-step prediction paradigm. Results were as follows:



(a) Temperature 1 vs Time Instant



(b) Temperature 2 vs Time Instant

Figure 5.36: Multi step predictions test: TCL States vs Time Instant

As we can see from Fig. 5.36, the RNN model of the TCL was not able to make multi-step prediction accurately. Hence to explain this deviation in multi-step results than one-step results, sequential step predictions were performed, i.e., 1 step, 2step, and so on, to see how the model behaves for different step predictions. Moreover, a step test was also performed with the RNN to gather more insights about the model behavior. The results of both various step predictions and step tests have been reported in Appendix D.
The model having parameters as shown in Table 5.13 was now combined with EKF to give filtered values of the states.

Table 5.13: Summary of Tanh RNN model of TCL system

T	Learning rate	Neurons in each hidden layer	Number of hidden layers	Epochs	Loss Train	Loss Test
10	0.0001	10	2	25000	$3.3 * 10^{-6}$	$4.5 * 10^{-6}$

Using the model summarized in Table 5.13, filtered values of the states of the system are estimated using EKF as estimator. State estimation has been done for 2 cases:

- When both states taken as measurement for estimation
- When single measurement taken for estimation

Results of RNN derived EKF using tanh-RNN as model for 2 measurements

Firstly the case of both the states as measurement has been discussed. Following initial values of the states and covariance matrix were taken for the estimation:

$$\boldsymbol{x(0|0) = \hat{x}(0|0) = \begin{bmatrix} 301 \\ 301 \end{bmatrix}_{1x1}} \tag{5.11a}$$

$$\boldsymbol{P(0|0) = \begin{bmatrix} 0.1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0.1 & \ddots & \ddots & & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & & \ddots & \ddots & 0.0631 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0.1346 \end{bmatrix}_{22x22}} \tag{5.11b}$$

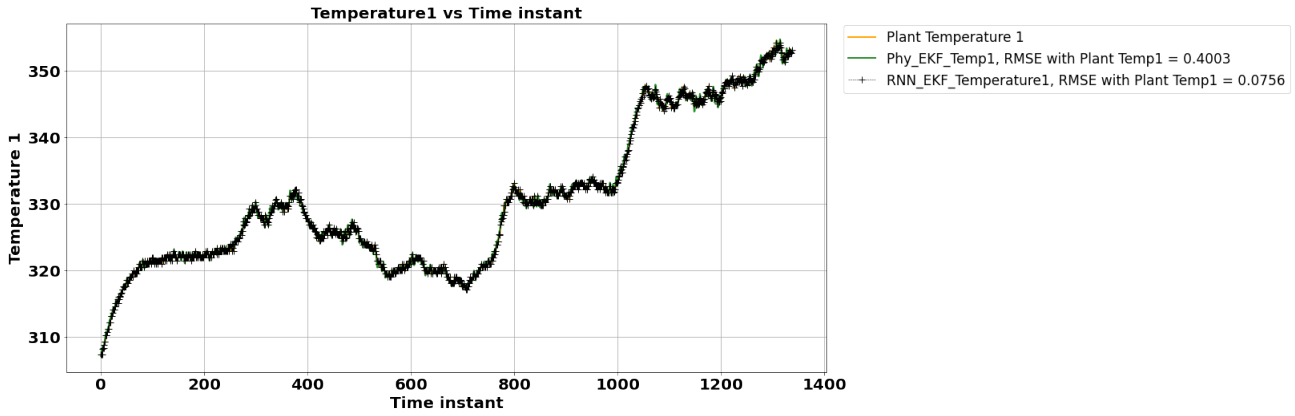
$$\boldsymbol{C = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0.9971 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0.9983 \end{bmatrix}_{2x22}} \tag{5.11c}$$

Plant system model and measurement are incorporated with white Gaussian noise with zero mean and covariance matrix Q and R (Jha (2021))

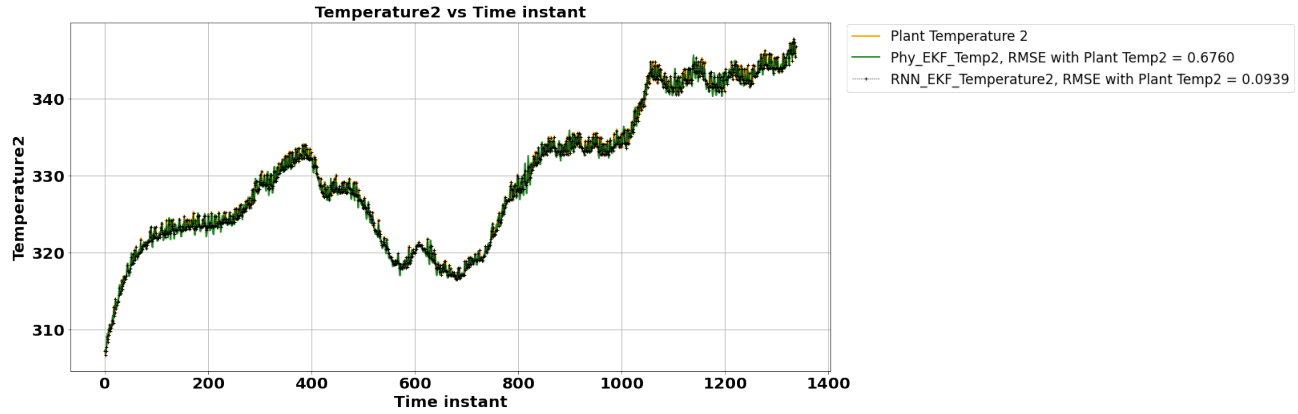
$$Q = \begin{bmatrix} 0.01 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0.01 & \ddots & \ddots & & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & & \ddots & \ddots & 1.02 * 10^{-3} & 0 \\ 0 & 0 & \cdots & 0 & 0 & 3.08 * 10^{-3} \end{bmatrix}_{22x22} \tag{5.12a}$$

$$R = \begin{bmatrix} 5.114 * 10^{-4} & 0 \\ 0 & 1.53 * 10^{-3} \end{bmatrix} \tag{5.12b}$$

As discussed in section 4.1.2, the state space model of TCL has been generated from RNN. Thus EKF was applied with the RNN model and the results of combined RNN with EKF is reported below and is compared with the physics model driven model and EKF:



(a) Temperature 1 vs Time Instant



(b) Temperature 2 vs Time Instant

Figure 5.37: Tanh-RNNEKF & Physics based EKF with both states measured: TCL States vs Time Instant

Fig. 5.37 shows results of both the proposed **Tanh RNNEKF** strategy and the conventional **physics based driven EKF and model**. Looking at the plots, we can conclude the Tanh RNNEKF gives similar results to the conventional method. Final RMSE obtained from both the methods for TCL state estimation has been discussed in Table 5.14. From Table 5.14 it can be concluded, the RMSE values of pure RNN driven estimator were found to be even less than the conventional physics model-based estimator. Hence with the provided advantage of less time consumption of modeling and expense, RNNEKF may give a better alternative to conventional strategies used till now.

Table 5.14: Summary of RNNEKF and conventional EKF of TCL when both states measured

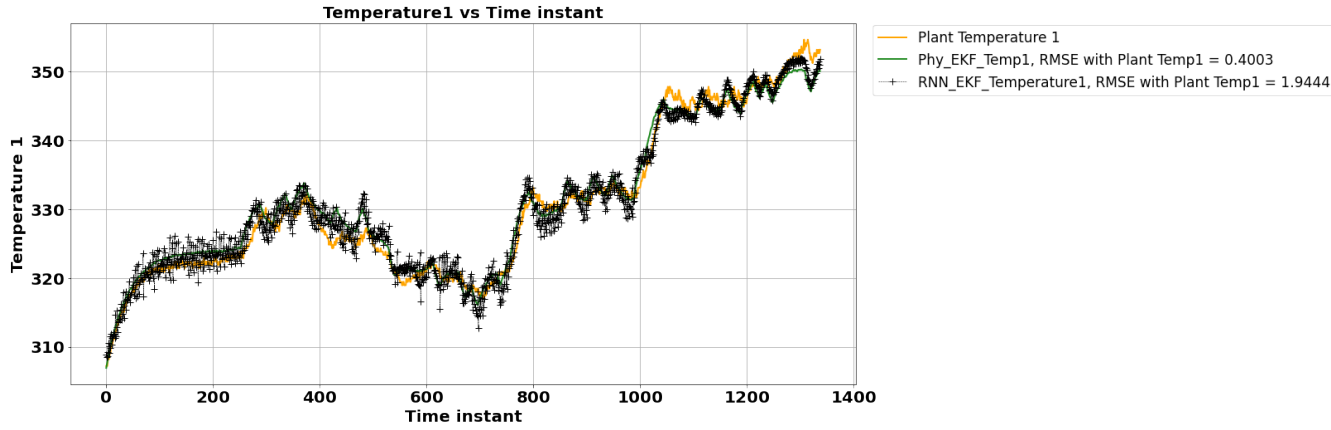
States	Proposed Work	RMSE	Computation Time(s)
Temperature 1	Tanh RNNEKF	0.0756	13.251
Temperature 2	Tanh RNNEKF	0.0939	13.251
Temperature 1	Physics EKF	0.4003	12.24
Temperature 2	Physics EKF	0.6760	12.24

Results of RNN derived EKF using tanh-RNN as model for 1 measurement The values for $\boldsymbol{x(0|0)}$, $\boldsymbol{P(0|0)}$ and $\boldsymbol{Q}$ remains same as stated in Eq. 5.11a, Eq. 5.11b and Eq. 5.13a. The values of R and C are as follows:

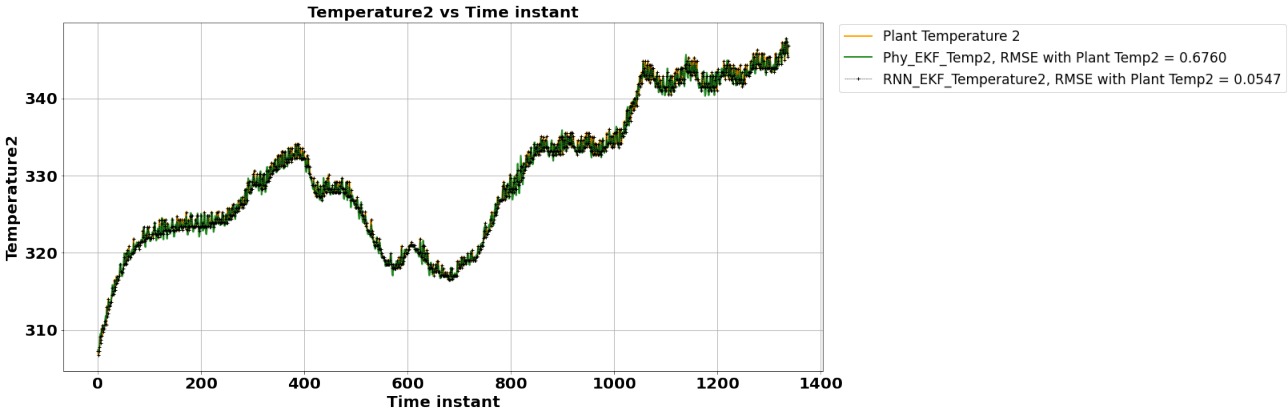
$$\boldsymbol{R = \left[1.53 * 10^{-3}\right]} \tag{5.13a}$$

$$\boldsymbol{C = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 0.9983 \end{bmatrix}_{1x22}} \tag{5.13b}$$

Using the above parameters EKF was applied to the Tan RNNEKF and the results of the Tanh RNNEKF and conventional physics based EKF are reported below:



(a) Temperature 1 vs Time Instant



(b) Temperature 2 vs Time Instant

Figure 5.38: Tanh-RNNEKF and Physics based EKF with 1 measurement: TCL States vs Time Instant

Fig. 5.38 shows results of both the proposed **Tanh RNNEKF** strategy and the conventional **physics based driven EKF and model** for single measurement. From the plots, we can deduce Tanh RNNEKF is similar to that of the mechanistic model. Few qualitative measures between both methods have been discussed in Table 5.15. From Table 5.15 it can be concluded, the RMSE values of pure RNN driven estimator were found to be even less than the conventional physics model-based estimator. Hence this shows the accuracy of the proposed work. Hence with the provided advantage of less time consumption of modeling and expense, RNNEKF may give a better alternative to conventional strategies used till now.

Table 5.15: Summary of RNNEKF and Physics EKF of TCL when 1 measured state

States	**Proposed Work**	**RMSE**	**Computation Time(s)**
Temperature 1	Tanh RNNEKF	1.9440	12.95
Temperature 2	Tanh RNNEKF	0.0547	12.95
Temperature 1	Physics EKF	1.7936	12.07
Temperature 2	Physics EKF	0.6713	12.07

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this report, different combinations of the model and estimator were discussed, which can efficiently solve the state estimation problem. Mechanistic models are often considered expensive, time-consuming, and require iterative calculations. Hence they are unsuitable for online applications. Therefore, switching to efficient data-based modeling methods can rectify this problem. Specifically, the recurrent neural network with dynamic mapping capabilities and being a universal approximator can be used instead of conventional physics-based models. The same idea has been implemented in the proposed strategies. Different case studies have been discussed in the report to analyze the performance of the proposed methods. For each case study, root mean squared error was used as a quantitative measure. It was found that the proposed methods resulted in low RMSE values and was able to estimate the states accurately as stated in Table 5.4, Table 5.10, Table 5.14 and Table 5.15. Moreover, looking at the computation time taken by all the estimators, **ReLU-RNNEKF** has been found with a $\mathbf{60-80\%}$ reduction in the computation time than other estimators. So, it can be concluded that a significant amount of calculation complexity has been reduced using the proposed work while not affecting the performance of the state estimator.

Experimental validation of the RNNEKF strategy was also done using the TCL system stated in section 5.3. This case study consists of a comparative study between the proposed work and the physics-based EKF through an experiment performed on a real system. It was found that when both states were measured, the RMSE between estimated states and the experimental data for the RNNEKF was **81.114%** and **86.109%** less than that of the physics-based EKF for state one and state two, respectively (Table5.14). It was also observed that the computational time taken by Tanh-RNNEKF was

similar to that of physics-based EKF, **13.251*s*** and **12.24*s*** respectively for TCL. Similar results were observed when a single measurement case was taken. There was a significant drop in the RMSE between RNNEKF and physics-based EKF, being **(1.9940, 1.7936)** and **(0.0547, 0.6713)** for state one and state two respectively (Table5.14) with computation time alike. Various other plots associated with the performance of the proposed methods have also been attached in the report. Looking at the results of the proposed strategy, it can be concluded that these methods can be used to solve the state estimation problem with less computation time and good accuracy than the conventional physics-based estimation, which is time-consuming and expensive. This gives strong motivation to study more about these methods, and thus the future work inclines in the extension of the proposed work.

6.2 Future work

Following are the things which can be explored taking the proposed work as basis:

1. Implementation of an advanced estimator like MHE with RNN as a model. It is expected that replacing the physics-based model with an RNN will save massive computation time to solve the optimization problem inside the MHE framework.

2. In order to reduce the computation time of the proposed work, parallel computing can be explored where a high GPU system can be accessed, and each core inside the system can be utilized to reduce the computation time to many folds.

3. More benchmark problems like tennessee eastman can be solved using the proposed algorithm to check the performance of the strategy on a real world type problems.

4. The algebraic representation of the model due to RNN can further be leveraged to reduce the computation time for the frameworks which solve optimization problem like MPC.

Appendix A

HRNNEKF on CSTR

As discussed in section 4.1 now, a RNN model of the CSTR system is generated, which is used in the prediction step of the EKF. Other pertinent parameters required from the model to run EKF are derived from the physics-based model stated in Eq. 5.2. As the method for modeling of the system is the same for HRNNEKF and RNNEKF, hence the results of RNN model performance for one step and multi-step prediction is the same as shown in section 5.1.1 from Fig. (5.6)-(5.10).

States estimated using HRNNEKF strategy for CSTR

Following initial values of the states and co-variance matrix were taken for the estimation:

$$\boldsymbol{x(0) = \hat{x}(0) = \begin{bmatrix} 0.257 \\ 391.57 \end{bmatrix}_{1x2}} \tag{A.1a}$$

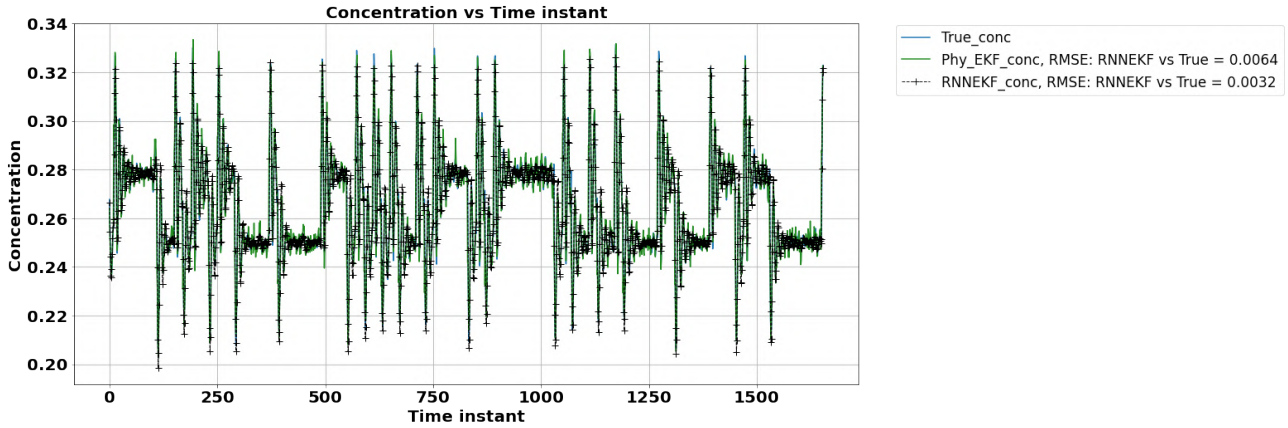
$$\boldsymbol{P(0|0) = \begin{bmatrix} 0.0001 & 0 \\ 0 & 0.0001 \end{bmatrix}_{2x2}} \tag{A.1b}$$

$$\boldsymbol{C = \begin{bmatrix} 0 & 1 \end{bmatrix}_{1x2}} \tag{A.1c}$$

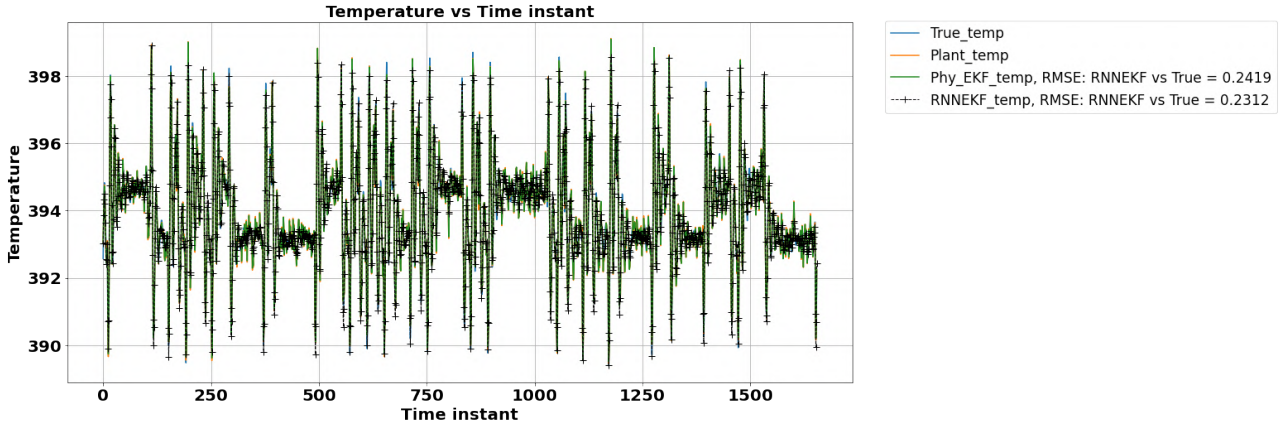
Plant system model and measurement are incorporated with white Gaussian noise with zero mean and covariance matrix Q and R (Patwardhan (2018)) which are given below:

$$\boldsymbol{Q = 0.012^2} \tag{A.2a}$$

$$\boldsymbol{R = 0.25^2} \tag{A.2b}$$



(a) Concentration vs Time instant



(b) Temperature vs Time instant

Figure A.1: Estimated states using HRNNEKF on CSTR

Using the model summarized in Table 5.3, filtered values of the states of the system are estimated using EKF, which uses both RNN and the physics-based model. Fig. **??** shows the filtered values of the states vs. time instant obtained from the HRNNEKF strategy. From the above plots, it can be concluded that using RNN as the model in EKF gives good results for the CSTR state estimation problem with very low RMSE values. As we can see from Table A.1 there has been a **50%** reduction in the RMSE value for concentration and a **4.42%** reduction in temperature. Moreover, the computation time taken by HRNNEKF was found to be **5.125*s***, which is **58.83%** less than the computation taken by conventional physics model-based EKF. Hence with the provided advantage of less time consumption of modeling and expense, RNNEKF and HRNNEKF may give a better alternative to traditional strategies used till now.

Table A.1: Summary of HRNNEKF and conventional EKF on CSTR

States	**Proposed Work**	**RMSE**	**Computation Time(s)**
Temperature	HRNNEKF	0.2312	5.125
Concentration	HRNNEKF	0.0032	5.125
Temperature	Physics EKF	0.2419	12.45
Concentration	Physics EKF	0.0064	12.45

Appendix B

Generating Phi(Φ) for RNNEKF-CSTR

The equations of the RNN trained for system identification of the CSTR were as follows:

$$x_{h1}(k+1) = f[w^u u(x) + w^y x_y(k) + w^d x_{h1}(k) + b_1] \tag{B.1a}$$

$$x_{h2}(k+1) = g[w^{d2} f[w^u u(x) + w^y x_y(k) + w^d x_{h1}(k) + b_1] + w^x x_{h2}(k) + b_2] \tag{B.1b}$$

$$x_y(k+1) = Wg[w^{d2} f[w^u u(x) + w^y x^y(k) + w^d x_{h1}(k) + b_1] + w^x x_{h2}(k) + b_2] + b_3 \tag{B.1c}$$

Where f and g are the ReLU activation function. As discussed in section 4.2.1, we will now generate $\mathbf{\Phi}$ from the RNN for the CSTR system. Below are the equations obtained after differentiating each Eq. B.1 with the augmented states x_{h1}, x_{h2} and x_y

$$f \xrightarrow{\max(x,0)} 0 \tag{B.2a}$$

$$f \xrightarrow{\max(x,0)} \begin{cases} \partial f/\partial x_y = w^y \\ \partial f/\partial x_{h1} = w^d \\ \partial f/\partial x_{h2} = 0 \end{cases} \tag{B.2b}$$

$$g \xrightarrow{\max(x,0)} 0 \tag{B.3a}$$

$$g \xrightarrow{\max(x,0)} \begin{cases} \partial g/\partial x_y = w^{d2} w^y \\ \partial g/\partial x_{h1} = w^{d2} w^d \\ \partial g/\partial x_{h2} = w^x \end{cases} \tag{B.3b}$$

$$z \xrightarrow{\text{Linear}} \begin{cases} \partial z/\partial x_y = W w^{d2} w^y \\ \partial z/\partial x_{h1} = W w^{d2} w^d \\ \partial z/\partial x_{h2} = W w^x \end{cases} \tag{B.4a}$$

Hence using differentiating values from Eq. B.2,B.3 and B.4 $\mathbf{\Phi}$ matrix is generated below:

$$\mathbf{\Phi} = \begin{bmatrix} \partial x_{h1}(1,k+1)/\partial x_{h1}(1,k) & \cdots & \cdots \\ \cdots & \ddots & \cdots \\ \cdots & \cdots & \partial x_y(2,k+1)/\partial x_y(2,k) \end{bmatrix}_{18x18} \tag{B.5}$$

This $\mathbf{\Phi}$ generated in Eq. B.5 has been used in EKF to generate filtered values of the states. As due to augmented states the size of the $\mathbf{\Phi}$ matrix has been increased, therefore proportionaly the size of Q and C matrix has to be adjusted for working of the EKF.

Appendix C

Converting EKF into standardized subspace for Tanh-RNNEKF

The physics based model of the system can be represented by following equations

$$\boldsymbol{x(k+1) = f(x(k), u(k), d(k)) + w(k)} \tag{C.1a}$$

$$\boldsymbol{y(k) = Cx(k) + v(k)} \tag{C.1b}$$

As the variables of the equations are considered as independent variables, hence can use laws of linearity of expectation and variance to derive parameters Q, R and C in standardized subspace, which are further used in EKF to yield filtered values. Using Eq. C.1b we apply expectation and variance on both sides, we get

$$\boldsymbol{\mu_y = \mu_x + 0} \tag{C.2a}$$

$$\boldsymbol{var(y) = var(x) + var(v)} \tag{C.2b}$$

We can rearrange Eq. C.2a - C.2b as follows:

$$\boldsymbol{\frac{y - \mu_y}{\mathbf{var}(y)} = \frac{x - \mu_x}{\mathbf{var}(y)} + \frac{v(x)}{\mathbf{var}(y)}} \tag{C.3a}$$

$$\boldsymbol{y^N = \frac{\sigma_x}{\sigma_y} x^N + \frac{v}{\sigma_y}} \tag{C.3b}$$

From Eq. C.3b we can derive the standardized C matrix

$$\boldsymbol{C' = \left[\frac{\sigma_x}{\sigma_y} C\right]} \tag{C.4}$$

From Eq. C.3b we can derive the standardized R matrix

$$v'_k = \frac{v_k}{\sigma_y} \tag{C.5a}$$

$$var(v'_k) = \frac{var(v_k)}{\sigma_y^2} \tag{C.5b}$$

$$R' = \frac{R}{\sigma_y^2} \tag{C.5c}$$

Similarly we can derive Q matrix for the standardized subspace using Eq. C.1a

$$\mu_{x_{k+1}} = \mu_{f_{x(k)}} + 0 \tag{C.6a}$$

$$\frac{x_{k+1} - \mu_{x_{k+1}}}{\mathbf{var}(x_{k+1})} = \frac{f(x_{k..}) - \mu_{f(x_{k..})}}{\mathbf{var}(x_{k+1})} + \frac{w_k}{\mathbf{var}(x_{k+1})} \tag{C.6b}$$

$$x^N_{k+1} = \frac{\sigma_{f(x_{k..})}}{\sigma_{x_{k+1}}} f(x_{k..})^N + \frac{w_k}{\sigma_{x_{k+1}}} \tag{C.6c}$$

Hence the standardized Q matrix derived is,

$$w'_k = \frac{w_k}{\sigma_{x_{k+1}}} \tag{C.7a}$$

$$var(w'_k) = \frac{var(w_k)}{\sigma^2_{x_{k+1}}} \tag{C.7b}$$

$$Q' = \frac{Q}{\sigma_x^2} \tag{C.7c}$$

using Eq. C.4, C.5c and C.7c and Eq. 5.8c and 5.9 the final normalized Q, R and C matrix are as follows:

$$C = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 0.9171 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 & 0.8502 \end{bmatrix}_{2x24} \tag{C.8a}$$

$$R = \begin{bmatrix} 0.01378 & 0 \\ 0 & 0.01502 \end{bmatrix}_{2x2} \tag{C.8b}$$

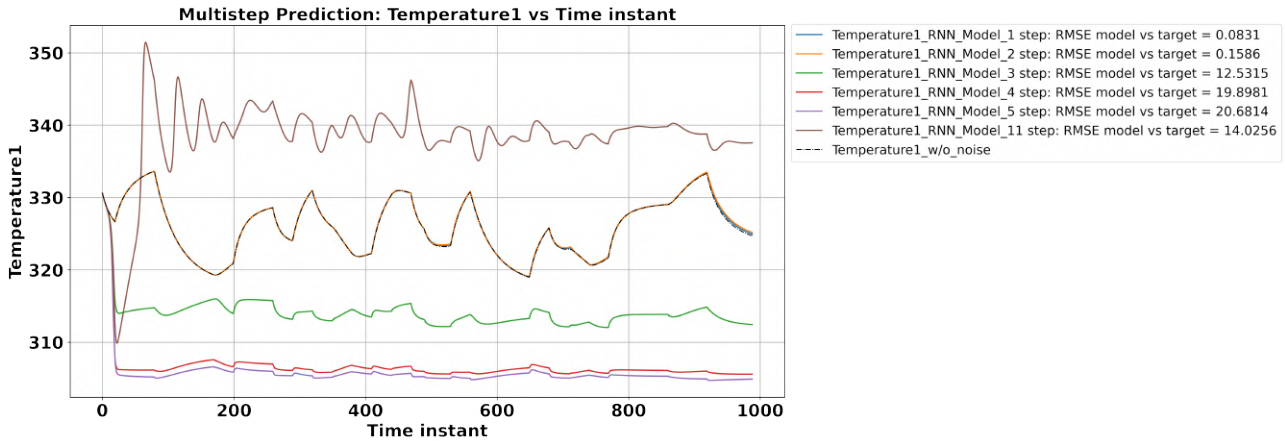
$$Q = \begin{bmatrix} \mathbf{0.01} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0.01} & \ddots & \ddots & & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \ddots & \ddots & \ddots & \vdots & \mathbf{0} \\ \vdots & \ddots & \ddots & \mathbf{0.01503} & \ddots & \mathbf{0} & \vdots \\ \vdots & \ddots & \ddots & \ddots & \mathbf{0.0176} & \mathbf{0} & \vdots \\ \mathbf{0} & & \ddots & \ddots & \mathbf{0} & \mathbf{0.0454} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0.08375} \end{bmatrix}_{\mathbf{24x24}} \tag{C.8c}$$

These are finally used by Tanh RNNEKF for estimating the states of the quadruple tank system.

Appendix D

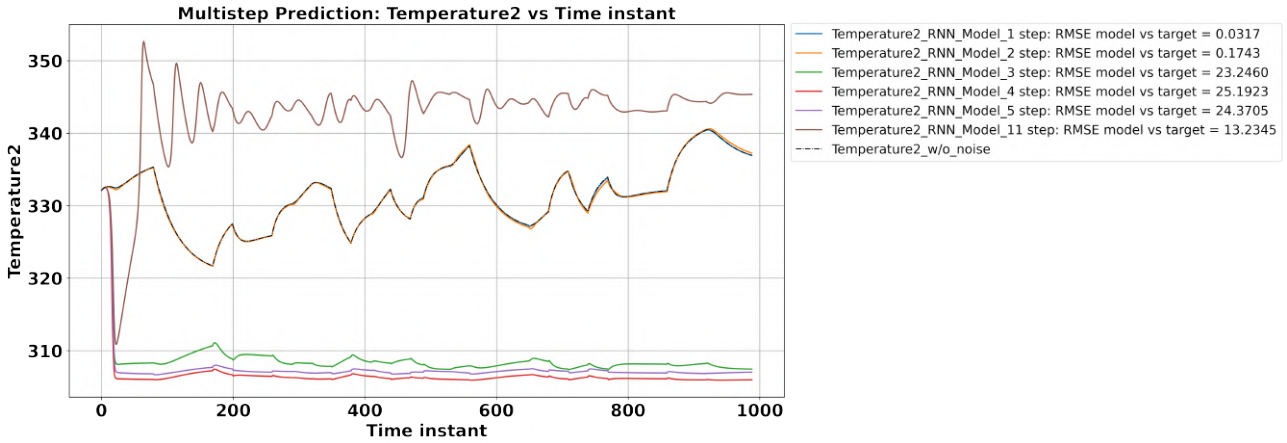
Various step prediction and step test for Quadruple Tank

From Fig. 5.36 we can deduce that the RNN was not able to make the multistep prediction accurately for the quadruple tank system. Hence, to see where the multistep prediction is deviating, various step predictions have been performed to analyze the RNN for different step predictions. Below are the plots for the same:



(a) Temperature 1 vs Time Instant

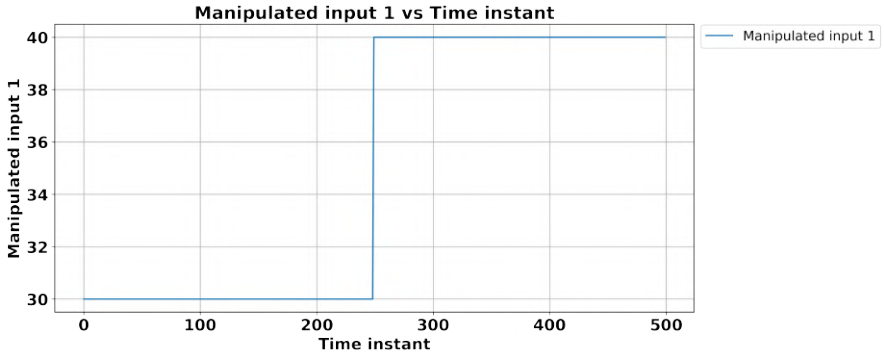
From Fig. D.2, it can be deduced that as we are increasing the step of the prediction, the prediction done by the model is degrading. Hence, from this, we can say that the deviation is due to the vanishing gradient problem of the RNN. Thus, to resolve this vanishing gradient problem, long short-term memory RNN (LSTM's) can be used as the model to capture the system's dynamics, eliminating this problem due to its complex architecture. The results for the same has been discussed in the Appendix E.



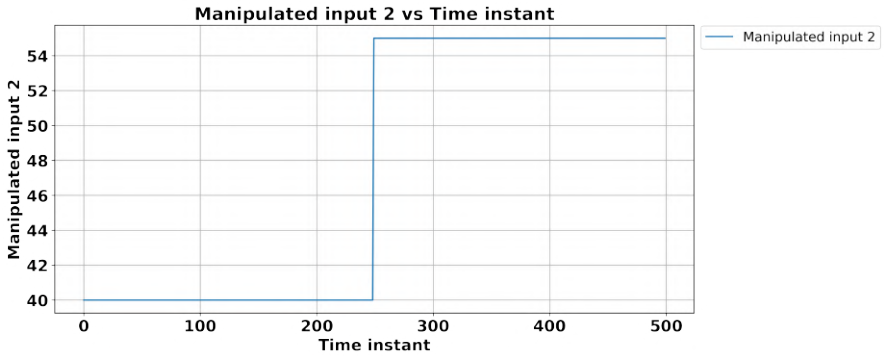
(a) Temperature 2 vs Time Instant

Figure D.2: Various step predictions test: TCL States vs Time Instant

To understand more about the working of RNN on the TCL system, a step test has also been discussed below. Fig. D.3 shows the manipulated input used for the step test.



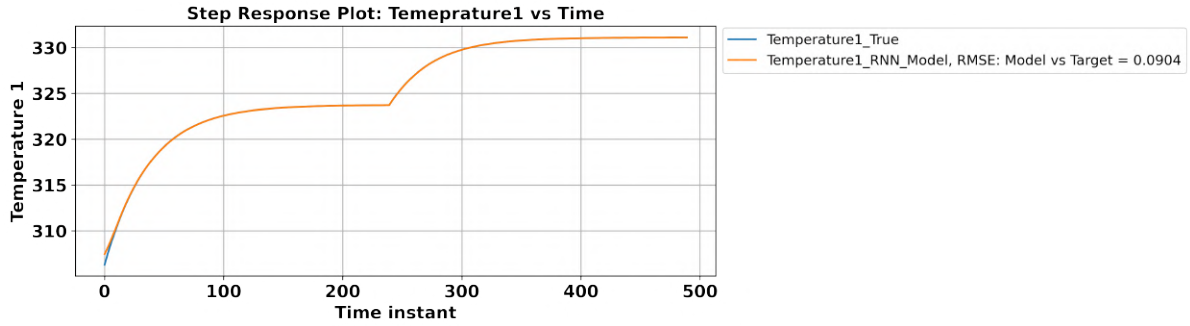
(a) Manipulated input 1 vs Time Instant



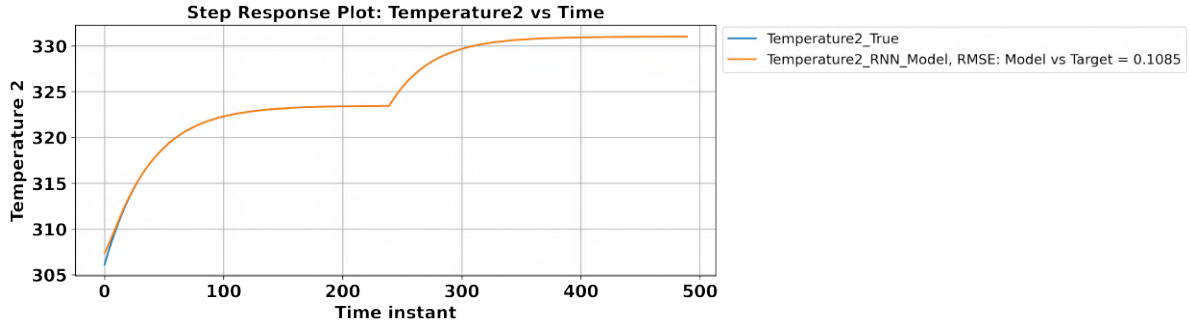
(b) Manipulated input 2 vs Time Instant

Figure D.3: Step test: Manipulated inputs vs Time Instant

Thus Fig. D.3 has been sent as input to the RNN model and below were the predictions done by the model.



(a) Temperature 1 vs Time Instant



(b) Temperature 2 vs Time Instant

Figure D.4: Step test: States vs Time Instant

from Fig. D.4 we can conclude that the RNN model for the TCL system was able to capture the step test trajectory of the true states.

Appendix E

Modelling of TCL system using LSTM

LSTM's are type of RNN which have complex architecture than the simple RNN. Pictorially and mathematically LSTM's are depicted as shown in Fig. 4.2 Varsamopoulos *et al.* (2018)

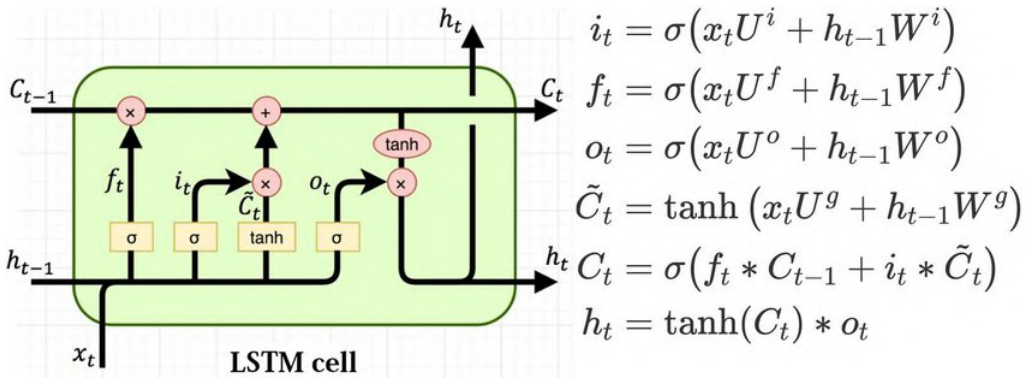
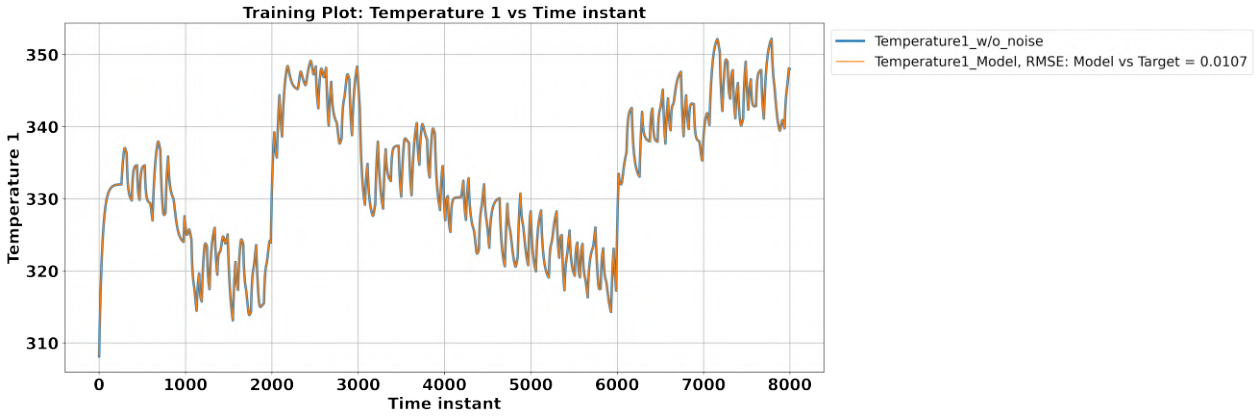


Figure E.1: Long short term memory network

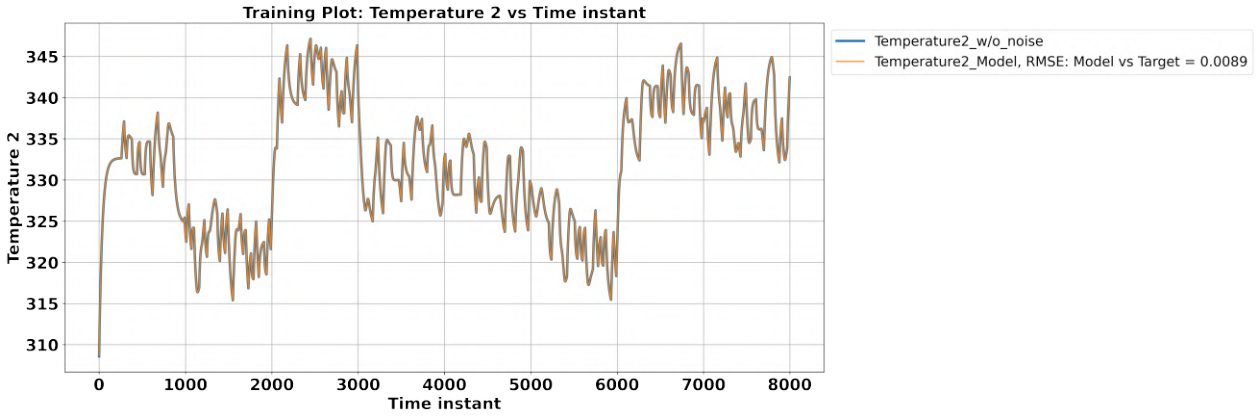
For training of the LSTM's, the data generation and preprocessing remain the same as discussed for RNN in section 4.1.1. Thus for the training of LSTM for the TCL system same data has been used as shown in Fig. 5.30 and 5.31. The same model parameter has been used as RNN as shown in table 5.13 for the comparison. Below are the results generated using LSTM as a model:

One step predictions for TCL-RNN

Following were the mapping done by the RNN for the training dataset:



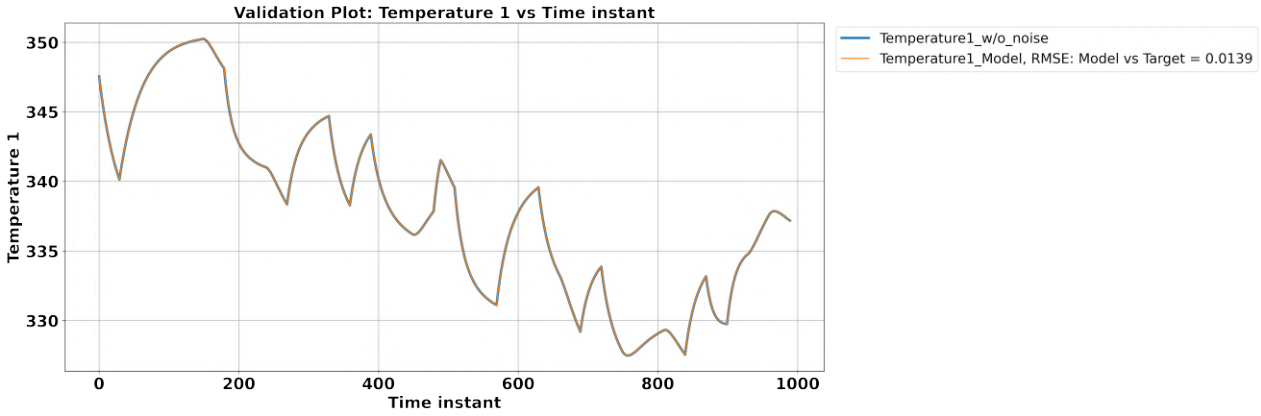
(a) Temperature 1 vs Time Instant



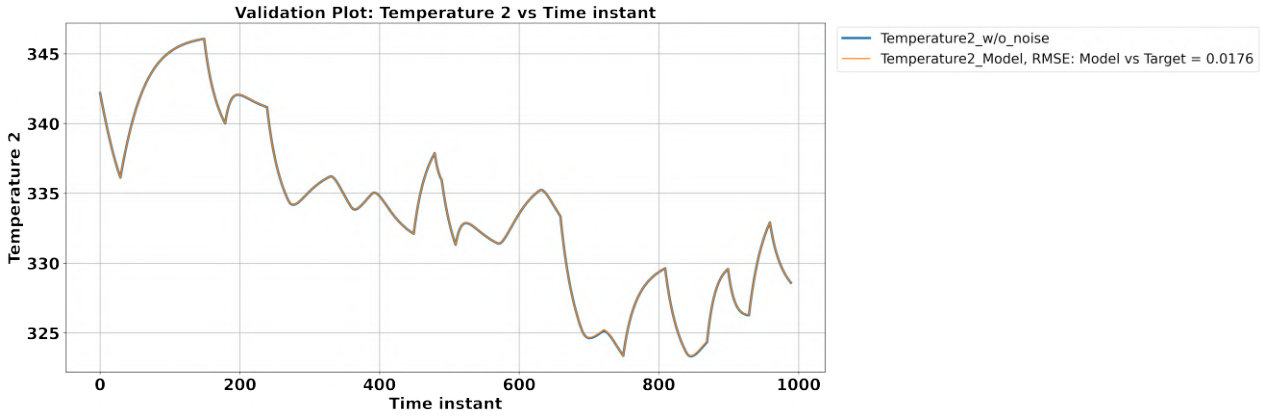
(b) Temperature 2 vs Time Instant

Figure E.2: LSTM one step predictions training: TCL States vs Time Instant

Results on the validation set



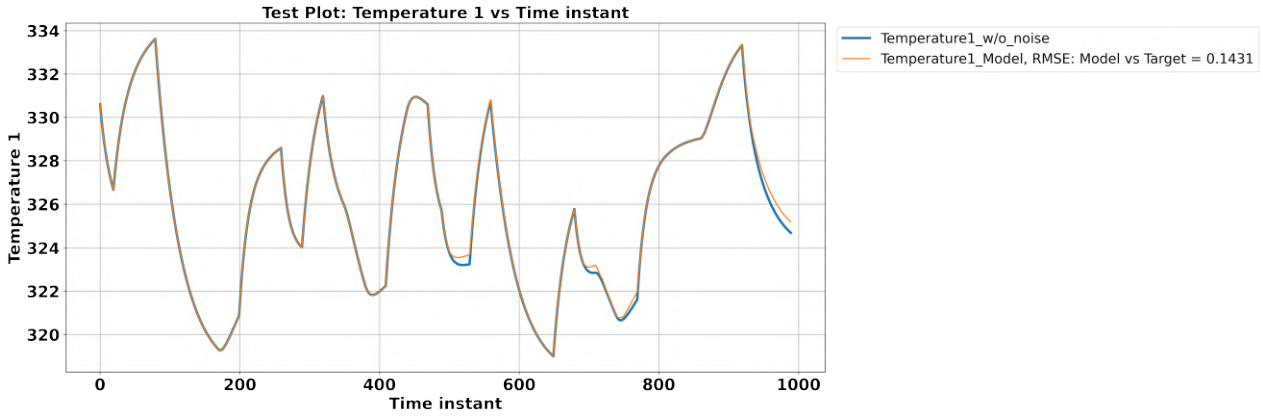
(a) Temperature 1 vs Time Instants



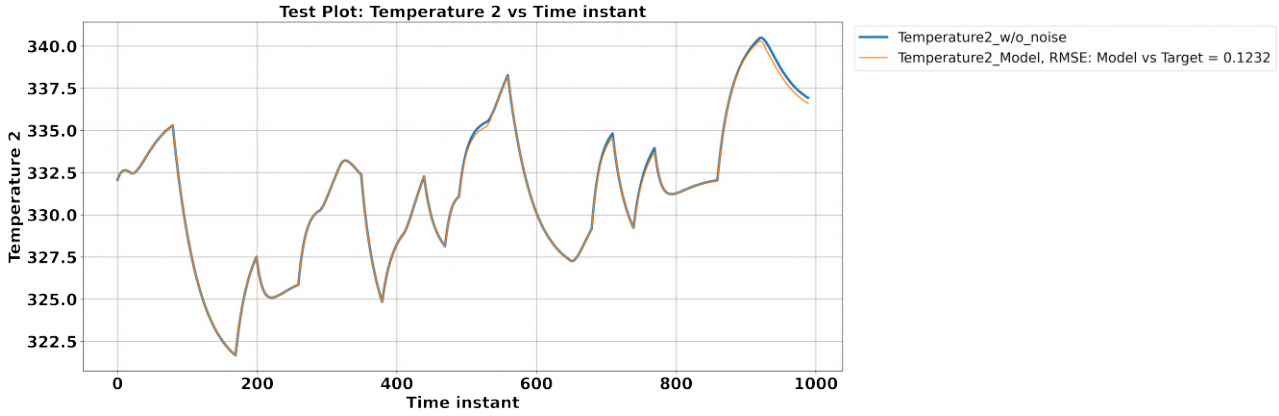
(b) Temperature 2 vs Time Instants

Figure E.3: LSTM one step predictions Validation: TCL States vs Time Instants

Results on the Test set



(a) Temperature 1 vs Time Instants



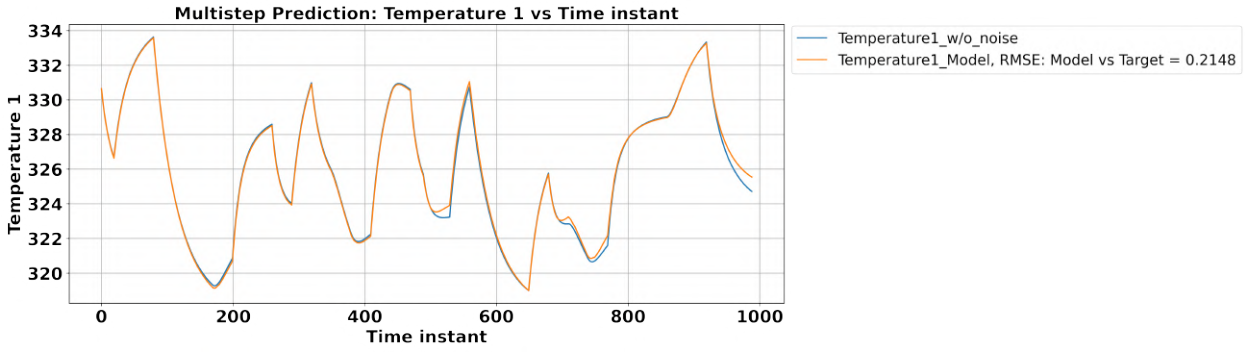
(b) Temperature 2 vs Time Instants

Figure E.4: LSTM one step predictions test: TCL States vs Time Instants

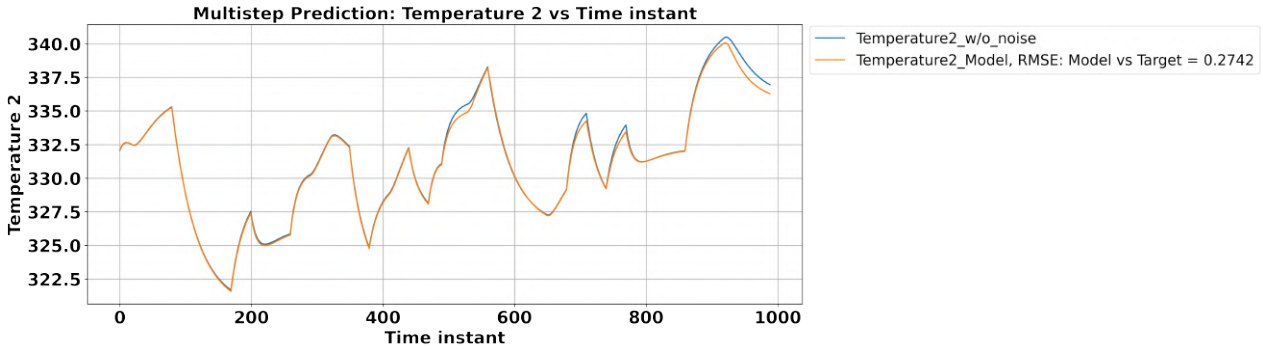
Fig (E.2) - (E.4) shows the plots for the states of the system vs Time for the training, validation and test data set respectively.

Multi step predictions for TC Lab-LSTM

After training the network using series-parallel mode now, the network was tested for multi-step prediction. The output of the previous instant is directly fed as input for the prediction of the next instance. Thus, this will help in giving predictions purely on estimated values done by the network. Therefore, tests data set was used to evaluate the model's performance for the multi-step prediction paradigm. Results were as follows:



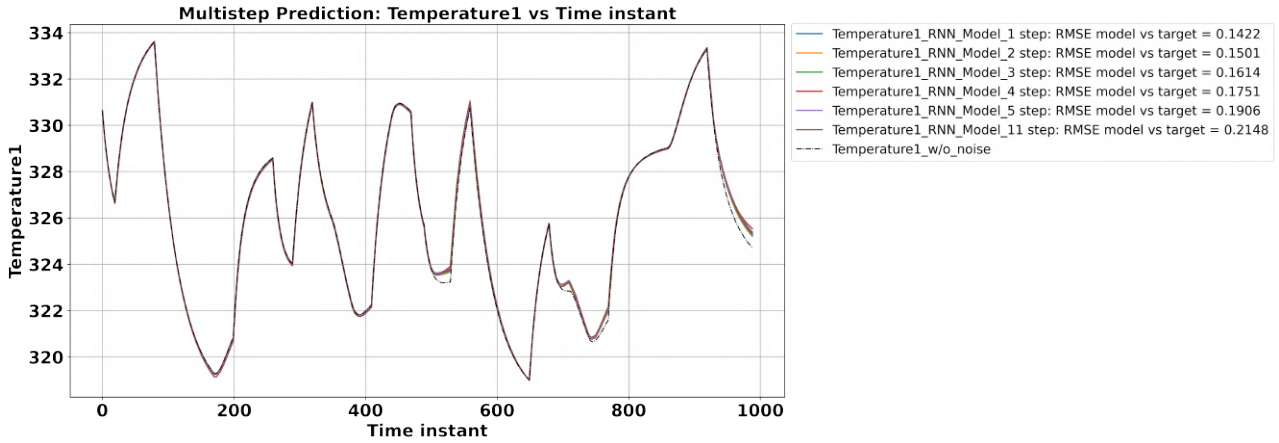
(a) Temperature 1 vs Time Instant



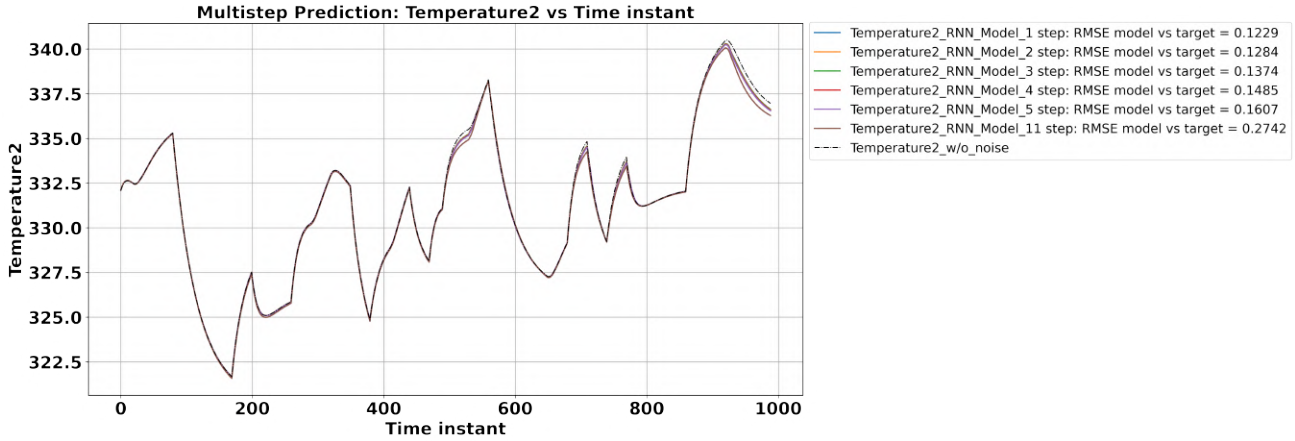
(b) Temperature 2 vs Time Instant

Figure E.5: Multi step predictions test: TCL States vs Time Instant

As we can see from the Fig. E.5, unlike RNN shown in 5.36, the LSTM model of the TCL system was able to predict the states accurately for multistep prediction with low RMSE values. This shows that the vanishing gradient problem faced in RNN was resolved using LSTM. Now this can be further used with estimator to give filtered values. Like in appendix D, various step predictions for LSTM has also been reported below:



(a) Temperature 1 vs Time Instant



(b) Temperature 2 vs Time Instant

Figure E.6: Different step predictions test: TCL States vs Time Instant

From Fig. E.6, we can show that the RMSE values for sequential step predictions do not diverge and remains low even for higher step predictions. Table E.1 give the architecture of the LSTM which has been used to model the TCL system

Table E.1: Summary of LSTM model of TCL system

T	**Learning rate**	**Neurons in each hidden layer**	**Number of hidden layers**	**Epochs**	**Loss Train**	**Loss Test**
10	0.0001	10	2	25000	1.3×10^{-6}	3.7×10^{-6}

References

Ambaw, A., 2005 01, *MODELING CHEMICAL ENGINEERING PROCESSES USING ARTIFICIAL NEURAL NETWORKS*, Ph.D. thesis

apmonitor, 2020, "Temperature control lab,"

Bhusan, M., and Patwardhan, S., 2018, "State estimation: Part 1, tech. rep. (i.i.t bombay)," *IEEE Transactions on Neural Networks*

Chang, W.-D., 2014, "Recurrent neural network modeling combined with bilinear model structure," *Neural Computing and Applications* **24**, 765–773.

Habtom, R., and Litz, L., 1997, "Estimation of unmeasured inputs using recurrent neural networks and the extended kalman filter," in *Proceedings of International Conference on Neural Networks (ICNN'97)*, Vol. 4 (IEEE). pp. 2067–2071.

Jazwinski, A., 2007, *Stochastic Processes and Filtering Theory*, Dover Books on Electrical Engineering Series (Dover Publications). ISBN 9780486462745

Jha, S., 2021, "Modelling and control of thermal energy systems, tech. rep. (i.i.t bombay),"

Kingma, D. P., and Ba, J., 2014, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*

Ku, C.-C., and Lee, K. Y., 1995, "Diagonal recurrent neural networks for dynamic systems control," *IEEE transactions on neural networks* **6**, 144–156.

Kumpati, S. N., Kannan, P., *et al.*, 1990, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on neural networks* **1**, 4–27.

Marlin, T., 1995, *Process Control: Designing Processes and Control Systems for Dynamic Performance*, Chemical engineering series (McGraw-Hill). ISBN 9780071138161

Mukherjee, T., 2020, "Approximate jacobian based extended kalman filter for large scale state estimation, tech. rep. (i.i.t bombay),"

Parlos, A. G., Menon, S. K., and Atiya, A., 2001, "An algorithmic approach to adaptive state filtering using recurrent neural networks," *IEEE Transactions on Neural Networks* **12**, 1411–1432.

Patwardhan, S., 2018, "Advance process control:, tech. rep. (i.i.t bombay),"

Shahriari-Kahkeshi, M., and Askari, J., 2011, "Nonlinear continuous stirred tank reactor (cstr) identification and control using recurrent neural network trained shuffled frog leaping algorithm," in *The 2nd International Conference on Control, Instrumentation and Automation* (IEEE). pp. 485–489.

Slišković, D., Grbić, R., and Hocenski, Ž., 2011, "Methods for plant data-based process modeling in soft-sensor development," *Automatika* **52**, 306–318.

Srinivasan, K., and Prakash, J., 2006, "Non-linear state estimation for continuous stirred tank reactor using neural network state filter," in *2006 Annual IEEE India Conference* (IEEE). pp. 1–4.

Stubberud, S. C., Lobbia, R. N., and Owen, M., 1995, "An adaptive extended kalman filter using artificial neural networks," in *Proceedings of 1995 34th IEEE Conference on Decision and Control*, Vol. 2 (IEEE). pp. 1852–1856.

Thosar, D., Mukherjee, T., Gilbile, P., and Bhushan, M., 2020, "Conventional and explicit approaches for simultaneous moving horizon estimation and model predictive control: A comparative evaluation," *IFAC-PapersOnLine* **53**, 356–361.

Varsamopoulos, S., Bertels, K., and Almudever, C., 2018 11, "Designing neural network based decoders for surface codes,"

Wang, Y., 2017, "A new concept using lstm neural networks for dynamic system identification," in *2017 American Control Conference (ACC)* (IEEE). pp. 5324–5329.

Yadaiah, N., and Sowmya, G., 2006, "Neural network based state estimation of dynamical systems," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings* (IEEE). pp. 1042–1049.

Acknowledgements

I wish to express my deep gratitude to my supervisor, Prof. Mani Bhushan, for his noble guidance and valuable suggestions. I'm thankful to him for mentoring me throughout, with a lot of patience and constant support. Your continuous guidance and encouragement throughout the year have given me the confidence and strength to complete the M.Tech thesis in the best possible manner.

I would also like to express my gratitude to Prof. Sharad Bhartiya, Prof. Sachin Patwardhan, Dr. Adithya Sagar, Rohit Tiberiwal, and Sarvesh Agrawal for clearing my doubts and guiding me in the right direction whenever I needed. I would also like to thank my senior colleague, Tathagata Mukherjee, for his valuable inputs and advice.

I am incredibly grateful to my parents, Mr. D.C Agrawal and Mrs. Neetu Agrawal, for their love, prayers, caring, and sacrifices to educate and prepare me for my future. I would also like to express my thanks to my sister, Mrs. Vidushi Agrawal, for believing in me and supporting me, especially during hard times.

I pay my gratitude to IIT Bombay for providing me with access to invaluable literature and my friends who have given me an unbiased and honest opinion whenever required. Finally, my thanks go to all the people who have supported me to complete the research work directly or indirectly.

Vinayak Agrawal
IIT Bombay
4 July 2021