University College of Southeast Norway

www.usn.no

FMH606 Master's Thesis 2017

Industrial IT and Automation

Recursive Subspace System Identification (RSSID) algorithms

Harrison Amaghu Idornigie

Faculty of Technology, Natural sciences and Maritime Sciences

Campus Porsgrunn

University College of Southeast Norway

www.usn.no

Course: FMH606 Master's Thesis, 2017

Title: Recursive Subspace System Identification (RSSID) algorithms

Number of pages: 72

Keywords: Recursive Subspace System Identification, Artificial Neural Network, Process Data, Modeling, Simulations, Identification Methods, Prediction error method.

Student: Harrison Idornigie

Supervisor: David Di Ruscio

External partner: None

Availability: Open

Approved for archiving: _____

(supervisor signature)

Summary:

The goal of system identification is to find mathematical equation that gives approximation to the actual behavior of real systems. In this thesis, a recursive subspace model identification algorithm is presented that recursively identifies both linear and nonlinear systems. Each recursion step consisted of two-stages: first, the innovation form of the stochastic system was estimated, then the model Matrices was estimated.

Much attention is paid to the computational cost and the performance of the models derived from the developed identification algorithm and a comparison to existing traditional methods as well as the neural network algorithm was made using various monte-carlo simulations on different laboratory data.

It is observed that the proposed algorithm performed better than some traditional methods in some conditions and was reasonable good on other conditions or process types and is therefore very reliable.

Preface

As a requirement for the fulfilment of the Master degree program in Industrial IT and Automation, this Master thesis on “Recursive Subspace System Identification (RSSID) algorithms” is written and submitted to University College of southeast Norway in Porsgrunn, Norway. This thesis report is prepared based on various literature review, experiments and simulations performed under the supervision of David Di Ruscio.

My sincere appreciation goes to my supervisor Dr. David Di Ruscio for his guidance and support during the entire period of the thesis work. He was very helpful and always supportive, providing better suggestions on how to carry out the task during the thesis work.

Special Thanks to God for his infinite grace, mercy, favor and provision of abundant resources.

Finally, Thanks to the university lecturers who have impacted me with the required knowledge to be able to complete this task. I am also very grateful to my friend Ivan Pirir and my family for their support in my academic career.

Harrison Amaghu Idornigie.

Porsgrunn, 10.05.2017

Nomenclature

SID	System Identification
ESSM	Extended state space model
SIM	Subspace identification methods
SISO	Single input Single Output
MIMO	Multiple Input Multiple Output
RANN	Recursive Artificial Neural Network
DAQ	Data acquisition
PRBS	Pseudo random binary signals
RPEM	Recursive prediction error method
RARX	Recursive Auto Regressive with exogenous input
RDSR	Recursive Deterministic and Stochastic system identification and Realization
RSSID	Recursive system identification
SVD	Singular values decomposition
LTIFD	Linear time invariant system of a finite order

Contents

Preface ..... **III**

Nomenclature ..... **IV**

1 Introduction **5**

1.1 Previous work **6**
1.2 Aim of the thesis. **9**
1.3 Methods and limitations **9**
1.4 Thesis outline **10**

2 Background **11**

2.1 Models **11**
2.2 Systems Description **11**
2.2.1 Synthetic noisy data from known model. ***11***
2.2.2 The Quadruple tank process ***12***
2.2.3 The Air heater process ***14***
2.3 Problem description **15**
2.4 Model Performance analysis **16**
2.4.1 Validation Analysis ***16***
2.4.2 Optimal Prediction Error ***17***

3 Modeling and Identification **19**

3.1 The Modeling procedure **19**
3.1.1 Data collection and systemization ***19***
3.1.2 Input signals ***21***
3.2 Data preprocessing **22**
3.3 Identification methods **23**
3.3.1 RDSR Algorithm ***23***
3.3.2 Using the System Identification Toolbox ***28***
3.3.3 Neural Network Approach ***31***
3.4 Identified models **34**
3.4.1 Models and Initial conditions ***34***
3.5 Comparison between RDSR and traditional methods **36**

4 Results **38**

4.1 Model Performance Analysis **38**
4.1.1 Synthetic Model ***39***
4.1.2 Quadruple tank ***42***
4.1.3 Air heater ***45***
4.2 Optimal Model Prediction **47**
4.2.1 Quadruple tank ***48***
4.2.2 Synthetic model ***49***

5 Concluding Remarks **51**

5.1 Conclusion **51**
5.2 Further works **52**

References **53**

Appendices **55**

A.1 Master's thesis proposal 55
A.2 Parameters for Quadruple tank models 56
A.3 Parameters for Synthetic models 58
A.4 Parameters for Air heater models 60
A.5 MATLAB codes 62

1 Introduction

System identification is an attempt to estimate a white, black or grey box model of a dynamic system based on observing the input-output data from the experiment. Zadeh (1962) defined system identification as: *"... the determination based on input and output, of a system (model) within a specified class systems (models), to which the system under test is equivalent (in terms of a criterion)".* (Chinarro, 2014)

Generally, a model tries to emulate the system behavior in a simplified way by choosing only the most significant system properties. Modeling techniques can therefore be classified as:

- Apriori modeling, **white-box** or morphological modeling, by making simple experiments to consider the physical or chemical laws involved.
- Aposteriori modeling or **black-box** modeling, by building a model based only on data (data-driven) without having previous knowledge of the system. The model describes how the outputs depend on the inputs, not how the system actually is, and characterizes the system dynamics (delays, speed, oscillations, etc.).
- **Grey box** modeling is a technique used when dynamics of the system's internal laws are not entirely known, so it is based on both insight into the system and on experimental data analysis.

The availability and reliability of the various design techniques of system identification has helped to expand the application fields beyond the scope of just industrial applications. As a result, system identification models have been applied in other diverse fields, such as, economy, environment, biology, psychology, biomedical research, hydrology, and glaciology.

The identification problem requires (L. Ljung, 1994):

1. a set of model structures,
2. a validation criterion and
3. an aim

The validation criteria and model structures will be presented over the course of this thesis. Some examples of identification aims could be (Chinarro, 2014):

- To design control strategies for a system (e.g., in optimizing an electrical microgrid operation).
- To analyze the properties of the system (e.g., quantity rates in a medication reaction).
- To forecast the evolution of the system (e.g., future climate prediction according a IPCC downscaling model)
- To identify hidden factors influencing a system (e.g., sun spots in the karst spring).
- To improve the internal knowledge of the system (e.g., the delay in the aquifer discharge with respect to precipitation events).
- To identify the interaction between coupled systems (e.g., climate and glaciers).

Since some unknown physical parameters of the dynamic processes are difficult to measure, identification is therefore needed. The finally obtained model could be in the form of state space, transfer function or polynomial function. It is therefore necessary to find a suitable identification algorithm for the parameter identification of the models since these models sometimes have characteristics of multi-variable and high-order and some parameters may be time-varying.

Various available algorithms for recursive implementations of system identification are presented in this report and studied in some details. Major focus would be placed on a Recursive DSR (the RDSR method) that was developed in the end of the 90'ies but only published in an internal report. A comprehensive comparison between the RDSR algorithm and other algorithms is made based on various monte-carlo simulations.

1.1 Previous work

Over the last few years, various recursive versions of linear subspace identification algorithms have been presented such as in (I. Goethals, 2004), (M. Lovera, 1998) and (G. Mercere, 2004.). In (L. Bako, 2009) it is shown that the Hammerstein subspace identification algorithm presented in (I. Goethals, 2005) can also be transformed into recursive form, allowing for its use in on-line applications. Recursive subspace model identification (RSMI) has been developed for decades however, most of RSMIs are only applied for open loop data

In (Jie Hou, 2015), a modified closed-loop PARSIM-E identification method is proposed to further improve the computational efficiency without sacrificing identification accuracy. The key idea lies with pruning redundant estimates on coupled parameters associated with the PARSIM-E identification method. Moreover, the proposed recursive PARSIM-E method can be used to update the product of the extended observability and controllability matrices for online parameter estimation.

In (Jie Hou, 2015), it is also stated that the subspace identification method has attracted more and more attention as a relatively new topic in the field of system identification. Its outstanding advantage is that the state space model which is equivalent to the original state space model can be directly estimated by the input and output data. The subspace identification has been widely used in multivariable systems since it was proposed. Subspace identification methods (SIMs) have been increasingly explored in the past two decades owing to the uniform parametrization for multiple-input-multiple-output (MIMO) systems. A number of SIMs, e.g. CVA (Larimore, 1990), MOESP(Verhaegen M, 1992), and N4SID (P.V. Overschee, 1994), have been widely recognized for practical applications.

Most of projection based SIMs use singular value decomposition (SVD) to extract the extended observability matrix. For on-line identification, the computational burden of SVD has become a notorious problem impeding efficient application of recursive SIMs (RSIMs).

To overcome the drawback, the projection approximation subspace tracking (PAST) and IV-PAST strategies were adopted to develop RSIMs. However, it was pointed out that although PAST can be used to reduce computational complexity, the resulting state-space model may not converge to a constant state-basis. To address the problem, a propagator was proposed to estimate the plant state directly(G. Mercere, 2006).

In (Xi Chen, 2012), the recursive subspace method for Wiener systems with general nonlinearity is considered. By the recursive method for the principle component analysis, the subspace of extended controllability matrix is recursively obtained. Then the matrix coefficients of the linear subsystem and the nonlinear function are also recursively estimated. Under rather mild conditions, all estimates are shown to be consistent. A simulation example is provided justifying the method proposed in the paper.

In (I. Houtzager, 2009) a subspace model identification algorithm is presented that could recursively track slowly time-varying linear systems operating both in open loop and closed loop. Much attention was also paid to the computational cost and tracking performance of their developed identification algorithm. The computational complexity was reduced by using array algorithms in solving these linear problems and also exploiting the structures in the vectors which resulted in a fast implementation of their developed recursive identification algorithm. Emphasis was made on the effectiveness of the proposed algorithm in comparison with existing methods with a simulation study on a time-varying closed-loop system.

In (YuePing Jiang, 2009), A recursive subspace identification algorithm was proposed for the closed-loop stochastic systems in state-space form. Each recursion step consisted of two-stages: first, the innovation of the stochastic system was estimated by the extended least squares (ELS) algorithm; then, a basis of the extended observability matrix was estimated by the stochastic approximation based principal component analysis (SABPCA) method by using the estimated innovation to replace the true one. The recursively estimated observability matrix converged to the true one up to a similarity transformation. The performance of the proposed algorithm was also illustrated via a simulation example.

In (Ping Wu, 2008), a new recursive subspace model identification was proposed which can be applied under open loop and closed loop data. The key technique of this derivation of the proposed algorithm was to bring the Vector Auto Regressive with eXogenous input (VARX) models into RSMI. Numerical studies on a closed loop identification problem showed the effectiveness of the proposed algorithm.

In (G. Mercere, 2006), the convergence properties of a recently developed recursive subspace identification algorithm of the MOESP class was investigated. The algorithm operated based on an extended instrumental variable (EIV) version of the propagator method for signal subspace estimation. It was proved that, under weak conditions on the input signal and the identified system, the considered recursive subspace identification algorithm converges to a consistent estimate of the propagator and of the state space system matrices by extension.

In (Liang Ma, 2016), the parameter estimation problem of ARMAX models for the Hammerstein systems was considered. The recursive maximum likelihood method, which could be applied to online identification and occupies small memory capacity, was proposed to deal with the problem. It was an approximation of the maximum likelihood method. The parameters of the linear and nonlinear parts of the Hammerstein model and the noise model could be directly obtained without using the over-parameterization technique. Finally, the proposed method was applied to a classic Hammerstein ARMAX system and was compared with RLS method. The research results show the effectiveness of the proposed method.

In (Guillaume Mercere, 2005), the problem of the recursive identification of MIMO state space models in the framework of subspace methods was considered. Two new algorithms, based on a recursive formulation of the MOESP identification class, were more precisely developed.

In (Kentaro Kameyama, 2005), a new subspace method for predicting time-invariant/varying stochastic systems was investigated in the 4SID framework. Using the concept of angle between past and current subspaces spanned by the extended observability matrices, the future subspace was predicted by rotating current subspace in the geometrical sense. To treat even time-varying system, a recursive algorithm was derived for implementation. The proposed algorithm was tested by simulation experiments.

The field of neural networks has a history of some five decades but has found solid application only in the past fifteen years, and the field is still developing rapidly. It is different from the fields of control systems or optimization where the terminology, basic mathematics, and design procedures have been firmly established and applied for many years. (Howard Demuth, 2002).

In (Kumpati S. Narendra, 1990), it was demonstrated that neural networks can be used effectively for the identification and control of nonlinear dynamical systems. The emphasis was on models for both system identification and control. Static and dynamic back-propagation methods for the adjustment of parameters were discussed. In the models that were introduced, both multilayer and recurrent networks are interconnected in novel configurations and therefore there was a real need to study them in a unified fashion. Simulation results revealed that the identification and adaptive control schemes suggested were practically feasible. Basic concepts and definitions were introduced and theoretical questions which should be addressed were also described.

In (K. J. Hunt, 1992), focus was placed on the promise of artificial neural networks in the realm of modelling, identification and control of nonlinear systems. The basic ideas and techniques of artificial neural networks are presented in familiar language and notations to control engineers. Applications of a variety of Neural architectures in control was also surveyed. They explored the links between the fields of control science and neural networks in a unified presentation and identified key areas for future research.

In (Yi Liu, 2010), it was stated that online identification of nonlinear systems is still an important while difficult task in practice. A simple online identification method called Selective Recursive Kernel Learning (SRKL) was proposed for multi-input–multi-output (MIMO) systems with the nonlinear autoregressive with exogenous input form. A two-stage RKL online identification framework was formulated, where the information contained by a sample can be introduced into and/or deleted from the model recursively. A sparsification strategy to restrict the model complexity was then developed to guarantee that all the output channels of the MIMO model were accurate simultaneously. A novel pruning approach based on the fast leave-one-out cross-validation criterion was explored to acquire generalization ability by determining and then deleting the useless information. Consequently, the model could adaptively adjust its structure to capture the process dynamics. The SRKL method was applied intensively to several nonlinear systems for multifold identification aims. The obtained results showed that SRKL was superior to traditional methods in different situations. The benefits of its accuracy, reliable performance and simplicity of implementation in practice indicated that the SRKL method is promising for online identification of nonlinear systems.

Importance of Recursive System Identification:

It may sometimes be necessary to estimate a model on line at the same time as the input-output data is received or you may need the model to make some decision on line, as in adaptive control, adaptive filtering, or adaptive prediction or it may be necessary to investigate some possible time variation in the system's properties during the collection of data. Terms such as adaptive parameter estimation, recursive identification, sequential estimation, and on-line algorithms are used for such algorithms. Chapter 11 in (Ljung, 1999) deals with such algorithms in more detail (Ljung, 2000).

Some common needs or applications of online estimation methods include:

- Adaptive control — to estimate a plant model which modifies the controller based on changes in the plant model.
- Online parameter estimation – To estimate the model parameter values at a time step, recursive algorithms use the current measurements and previous parameter estimates.
- Soft sensing — to generate a measurement value based on the estimated plant model, and use this measurement for feedback control or fault detection.
- Fault detection — to compare the online plant model with the idealized or reference plant model to detect a fault (anomaly) in the plant.
- Verification of the experiment-data quality before starting offline estimation — Before using the measured data for offline estimation, online estimation is sometimes performed for a few iterations. This online estimation can provide a quick check of whether the experiment used adequate excitation signals that captured the relevant system dynamics.

Since the final models identified by the recursive implementations is usually similar to that of their offline counterpart, the recursive algorithms are therefore efficient in terms of memory usage. Also, recursive algorithms usually have smaller computational demands. This efficiency makes them suited for online and embedded applications. See Recursive Algorithms for Online Parameter Estimation.(Ljung, 2000).

1.2 Aim of the thesis.

The main issues of this report are:

1. To perform a literature review of Recursive Subspace based System Identification (RSSID) algorithms. Give an overview and work out a comparison with traditional Recursive System Identification (RSID) methods such as the Recursive Prediction Error Method (RPEM) and if there are something to gain by using RSSID methods, identification of system order etc. Possibly also compare RSSID methods with (recursive) Neural Network methods.
2. To investigate the RDSR algorithm in some details and give a short and detailed description of the algorithm.
3. To perform simulation experiments and Monte Carlo simulations to investigate the quality of the parameter estimates.
4. To test on real data from one of our laboratory processes would be of interests. The quadruple tank process would be a candidate laboratory process.

1.3 Methods and limitations

The models are identified using the System identification and RDSR Toolboxes in MATLAB. The data measured from the different systems under consideration are divided into two datasets, one for creating the models and the other for validating the models. Throughout the thesis, the black-box technique is used.

Recursive identification requires online estimation. Online estimation differs from offline estimation in the following ways (Ljung, 2000):

- Model delays — It is easier to estimate model delays in offline estimation but online estimation provides limited ability to estimate delays. For polynomial model estimation, you can specify a known value of the input delay (nk).
- Data pre-processing — For offline estimation data pre-processing, it is possible to use functions such as `detrend`, `retrend`, `idfilt`, and the System Identification app but for online parameter estimation at the command line, you cannot use pre-processing tools in System Identification Toolbox™. Pre-processing is implemented according to the application.
- Resetting of estimation — It is not possible to reset offline estimation but online estimation lets you reset the estimation at a specific time step during estimation
- Enabling or disabling of estimation — It is not possible to selectively enable or disable offline estimation but online estimation lets you enable or disable estimation for chosen time spans.

Online parameter estimation is usually mathematically challenging due to the less information provided for the estimation procedure compared to the offline procedure. Implementing online identification is not a push-button approach in the sense that a good domain knowledge is required as well as extensive simulation testing and careful model design.

1.4 Thesis outline

This thesis report is organized as follows. In chapter 1, a brief introduction, some previous work done in this field and the general overview of problem considered in this work is presented.

Chapter 2 gives a brief description of the models and Systems under consideration.

In chapter 3, the various Recursive System Identification approaches to be investigated are presented giving a brief explanation of the different algorithms and how the measured or generated data is organized into data matrices which satisfy the models or matrix equations required for the method. How the system order and the model matrices can be extracted from the known data matrices is also presented.

Efficient and stable implementations as well as various Monte Carlo simulations and experimentations are performed and the results are presented and compared with other methods for performance, stability and accuracy. Results from real world Numerical examples are also presented in chapter 4.

A summary of the work done is presented in chapter 5 as well as some Concluding remarks made based on the results from the simulations and experimentations. Some recommendations for further works also follows in.

2 Background

This chapter gives a brief description of the models and Systems under consideration. It starts with a brief introduction to the system identification procedure and some of the aspects regarding the various identification approaches. The purpose of this is to explain some of the choices made during the identification process and to support the assumptions made regarding which inputs and outputs data are used.

2.1 Models

A model represents a mathematical relationship between a system's input and output variables. Typically, models of dynamic systems are described by differential or difference equations, transfer functions, state-space equations, and pole-zero-gain models. A linear model is often sufficient to accurately describe the system dynamics but if the linear model output does not adequately reproduce the measured output, a nonlinear model can be used.

Before building a nonlinear model of a system after the system fails the tests for linearity, transform the input and output variables such that the relationship between the transformed variables is linear. If the variable transformations that yield a linear relationship between input and output variables cannot be determined, nonlinear structures can be used.

Dynamic models can be represented both in continuous-time and discrete-time form. In a dynamic system, the values of the output signals depend on both the present values of its input signals and on the past behavior or outputs of the system (MathWorks, 2016)

2.2 Systems Description

For this thesis work, the three systems under consideration are:

- Synthetic noisy data.
- The Quadruple tank process.
- The Air heater process.

A brief overview of these processes is presented below.

2.2.1 Synthetic noisy data from known model.

A known MIMO system with $n = 3$ states, $m = 2$ outputs and $r = 2$ inputs is simulated to generate data for testing the different algorithms to see how close to the known system matrices the identified matrices from the algorithms can get. It is described by the following state space model:

$$\overbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_{k+1}}^{x_{k+1}} = \overbrace{\begin{bmatrix} -1.5 & 1 & 0.1 \\ -0.7 & 0 & 0.1 \\ 0 & 0 & 0.85 \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_k}^{x_k} + \overbrace{\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}}^{B} \overbrace{\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_k}^{u_k} + \overbrace{\begin{bmatrix} 0 & 0.1 \\ 0.1 & 0 \\ 0 & 0.2 \end{bmatrix}}^{C} \overbrace{\begin{bmatrix} v_1 \\ v_2 \end{bmatrix}_k}^{v_k} \tag{2.1}$$

$$\overbrace{\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_k}^{y_k} = \overbrace{\begin{bmatrix} 3 & 0 & -0.6 \\ 0 & 1 & 1 \end{bmatrix}}^{D} \overbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_k}^{x_k} + \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}^{E} \overbrace{\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_k}^{w_k} \tag{2.2}$$

Where the initial state is given by

$$x_{k=1} = [\ 0\ \ 0\ \ 0\ ]^T \tag{2.3}$$

A Pseudo Random Binary Signal (PRBS) experiment is performed in each of the two input variables of the system. The experiment is of length $N = 2000$ discrete time instants or samples. The experiments in each input channel is made constant over intervals of length 10 samples for input 1 and 20 samples for input 2. This gives the input data matrix

$$U = [\ u_1\ \ u_2\ ] \in \mathbb{R}^{N\times 2} \tag{2.4}$$

The process noise v_k and the measurement noise w_k are both white noises with zero mean.

The simulated output y_k^d of the system is defined as the output of the deterministic part of the above state space model described as

$$x_{k+1}^d = Ax_k^d + Bu_k \tag{2.5}$$

$$y_k^d = Dx_k^d + Eu_k \tag{2.6}$$

The simulation Error:

The simulated error is defined as

$$e_k^d = y_k - y_k^d \quad \forall \quad k = 1, \dots, N \tag{2.7}$$

The size of the error can be measured by some matrix norm of the covariance matrix of the error

$$\Delta^d = \frac{1}{N-1}\sum_{k=1}^{N} e_k^d \left(e_k^d\right)^T = \frac{1}{N}(E^d)^T E^d \tag{2.8}$$

Where $E^d = Y - Y^d$ is an $N\times m$ matrix of the error at the N time instants. After simulating the model, the modeling error is 4.0594 which shows that it is a reasonably good model.

A good model is a model which results in a small simulated error e_k^d. This is different from the prediction error $e_k = y_k - \bar{y}_k$ where $\bar{y}_k$ is the optimal prediction.

2.2.2 The Quadruple tank process

The quadruple tank process is a non-linear process with an experimental setup as seen in Figure 2.1. The non-linear state space model of the tank is derived from mass balances and Bernoulli's / Torricelli's law. The details of the derivation is not of interest to this report however the equations below shows the mass balance equations derived for the process (Ruscio, 2012).

$$A_1\dot{x}_1 = -q_1^{out} + q_3^{out} + q_1^{inn}, \quad (2.9)$$

$$A_2\dot{x}_2 = -q_2^{out} + q_4^{out} + q_2^{inn}, \quad (2.10)$$

$$A_3\dot{x}_3 = -q_3^{out} + q_3^{inn}, \quad (2.11)$$

$$A_4\dot{x}_4 = -q_4^{out} + q_4^{inn}. \quad (2.12)$$

The flow q^{out} out of each tank is modeled using the Bernoulli's / Torricelli's law. We can obtain the flow velocity $v\left[\frac{m}{s}\right]$ by equating the potential energy and Kinetic energy i.e. $mgh = \frac{1}{2}mv^2$ and solving for the velocity $v = \sqrt{2gh}$. We can multiply this with the area, a of the outlet hole of the tank to obtain the volumetric flow-rate, $q\left[\frac{m^3}{s}\right]$, out of the tank as $q = av = a\sqrt{2gh} = c\sqrt{h}$, i.e. the flow is proportional with the square root of the height where constant $c = a\sqrt{2g}$.

The flow out of each i^{th} tank is therefore given by:

$$q_i^{out} = a_i v_i = a_i\sqrt{2gh_i}. \quad (2.13)$$

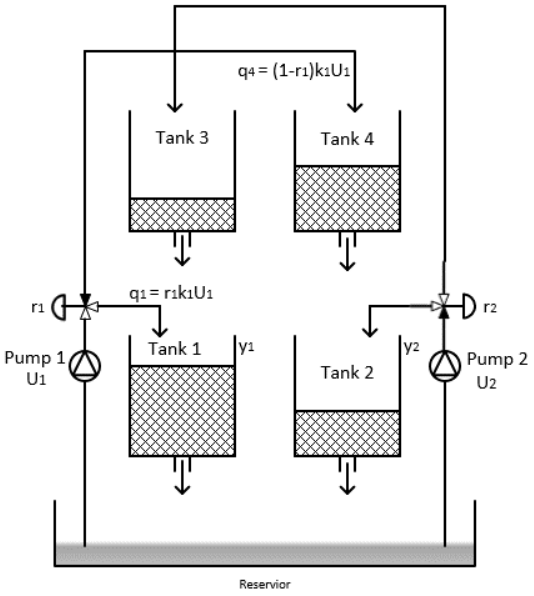

Figure 2.1: The Quadruple Tank Process

The flow $q_1 = k_1u_1$ from pump 1 can be divided into two paths, the flow $q_1^{inn} = \gamma_1 k_1 u_1$ into Tank 1 and a flow $q_4^{inn} = (1-\gamma_1)k_1u_1$ to Tank 4 such that the flow from pump 1 is $k_1u_1 = \gamma_1 k_1 u_1 + (1-\gamma_1)k_1u_1$. Where γ_1 is a valve parameter which may be fixed such that $0 < \gamma_1 < 1$.

Similarly, the flow $q_2 = k_2 u_2$ from the second pump can be divided into two, the flow $q_2^{inn} = \gamma_2 k_2 u_2$ into tank 2 and a flow $q_3^{inn} = (1-\gamma_2) k_2 u_2$ into Tank 3. Where $0 < \gamma_1 < 1$ and $0 < \gamma_2 < 1$ are fixed valve parameters.

It is also noteworthy that the system is non-minimum phase when choosing these parameters such that, $0 < \gamma_1 + \gamma_2 < 1$, and the system is minimum phase when, $1 < \gamma_1 + \gamma_2 < 2$. Therefore, a mass balance of the quadruple tank process gives the following state space model (Ruscio, 2012).

$$A_1 \dot{x}_1 = -a_1\sqrt{2gx_1} + a_3\sqrt{2gx_3} + \gamma_1 k_1 u_1, \quad (2.14)$$

$$A_2 \dot{x}_2 = -a_2\sqrt{2gx_2} + a_4\sqrt{2gx_4} + \gamma_2 k_2 u_2, \quad (2.15)$$

$$A_3 \dot{x}_3 = -a_3\sqrt{2gx_3} + (1-\gamma_2) k_2 u_2, \quad (2.16)$$

$$A_4 \dot{x}_4 = -a_4\sqrt{2gx_4} + (1-\gamma_1) k_1 u_1. \quad (2.17)$$

Where $A_i \; \forall \; i = 1, \dots, 4$ is the cross-section area of tank $a_i \; \forall \; i = 1, \dots, 4$ is the cross-section area of the outlet pipe of the tank i.

The input measurements are taken from the pump control signals U_1 and U_2 while the output measurements are taken from the levels in Tanks 1 and 2, Y_1 and Y_2 respectively.

2.2.3 The Air heater process

In cold regions, the rooms are usually very cold and uncomfortable for habitation. To make the buildings comfortable for living, the air heater is usually used to heat up the environment to conducive temperatures. One of such systems is analyzed in this report and various system identification approaches is tested on the system based on input/output data. For control purposes however, a mathematical model that has proven to describe the dynamic behavior of the air heater quite well in simulations is as follows (Finn Haugen, 2007):

$$T_{out} = T_{env} + T_{heat} \quad (2.18)$$

Where:

T_{env}: is the environmental (room) temperature.

T_{heat}: is the additive contribution to the total temperature T_{out} due to the heater. T_{heat} is given by the following "time-constant with time-delay" differential equation model:

$$\theta_t * \frac{d(T_{heat})}{dt} = -T_{heat} + Kh * u(t - \theta_d) \quad (2.19)$$

Where:

θ_t [s] is time-constant.

u [v] is the control signal to the heater.

Kh [K/V] is heater gain (K is Kelvin).

θ_d [s] is time-delay representing sluggishness of the heater.

For this report, we will not go into much details concerning the model development and the various model parameters however, it should be noted that the experiments were conducted in an environment with $T_{env} = 22℃$.

The physical structure of the air heater consists of the following items (Finn Haugen, 2007):

- One air fan
- One potentiometer (variable resistance) for manual adjustment of the voltage controlling the fan speed.
- One electric power cable (for connection to mains outlet, e.g. 220 V)
- Two temperature sensors, type Pt100, with measurement signal converter from resistance to current.
- One heating element (coil) for electric heating of air. Power (assuming 220 VAC) is 250 W. Heater: The supplied power is controlled by an external voltage signal in the range [0 V, 5 V] applied to a Pulse Width Modulator (PWM) which connects/disconnects the mains voltage to the heater.
- One electrical AC-DC converter from 220 VAC to 24 VDC.
- One Pulse-width modulator (PWM). The PWM signal is indicated by a lamp on the lab station. The PWM device requires 24 VDC power supply, which is produced by the AC/DC converter.
- Air Tube: The air pipe is made of plastic.

The input measurements are taken from the external control voltage signal to the heating element represented as U in the mathematical model while the output measurement is taken from the temperature sensor Pt100 represented as T_{out} in the mathematical model.

2.3 Problem description

As stated earlier, System Identification tries to estimate a model of a system based on observed input-output data. Several ways to describe a system and to estimate such descriptions exist. This section gives a brief account of some common approaches (Ljung, 2000).

The procedure to determine a model of a dynamical system from observed input-output data involves three basic ingredients:

- The input-output data
- A set of candidate models (the model structure)
- A criterion to select a model in the set. (the identification method)

The identification process amounts to repeatedly selecting a model structure, computing the best model in the structure, and evaluating this model's properties to see if they are satisfactory (Ljung, 2000). For recursive implementation, the cycle can be itemized as follows:

1. Design an experiment and collect input-output data from the process to be identified.

2. Select and define a model structure which is a set of candidate system types within which a model is to be found. A set of candidate model structures, are selected based on some criteria relating to the kind of model needed.
3. Compute the best model parameters in the model structure according to the input-output data and a given criterion of fit. The model maps the input $u(t)$ to the output $y(t)$ which is usually corrupted by noise or perturbations $\varepsilon(t)$.
4. Examine the obtained model's properties. The verification of the model consists of finding the vector that best minimizes the error between real data and predicted data: $e(t) = y(t) - \hat{y}(t)$.
5. If the model is good enough, then stop; else go back to Step 3 and repeat until satisfactory results are achieved.

System identification is carried out through stages of preparation, analysis and the identification as such (selection and optimization).

The analysis procedure tries to obtain the most details inside the system and take as much useful information from the time series as possible. e.g. finding out if it is a linear system or a nonlinear system, a time-invariant system or a time-variant system, a single input system or a multi input-multi output system, a continuous system or a discontinuous system, an open-loop system or a closed-loop system, etc.

The selection stage is the identification of the most suitable, identifiable model structure. The final identification stage which follows the most critical procedures of estimation computes the best model based on the data and a given criterion.

In **the validation** stage, the developed or identified model is analyzed to measures the ability of the model to explain the observed data and show how efficient the model is in terms of some criterion such as the general prediction error (maximum likelihood) or some efficiency criterion for parametric models. In recursive system identification, the process necessarily has to be iterative (Chinarro, 2014). This model validation process will be discussed in the next sections

2.4 Model Performance analysis

When a model is made, there is always a trade-off between accuracy and model complexity. A large model can reproduce a measured output arbitrarily well but we must verify that the model is relevant for other data that was not used for the estimation but was collected for the same system. This makes it necessary to analyze the created models.

2.4.1 Validation Analysis

To measure the quality of an identified model, we have to investigate the error by defining some criteria also known as the performance index such as:

- Goodness of fit in %
- MSE – Mean Squared Error

Fit %

Given the measured output y and the predicted or simulated output y. For N number of input-output pairs the goodness of fit is computed as:

$$fit = \left(1 - \frac{\sqrt{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}}{\sqrt{\sum_{i=1}^{N}\left(y_i - \frac{1}{N}\sum_{i=1}^{N} y_i\right)^2}}\right) \times 100 \quad \forall\ i = 1, \dots, N \tag{2.20}$$

Mean Squared Error (MSE):

In a recursive sense, the Mean squared error between the measured output and the simulated output at each observation *o* or sample is defined as

$$MSE_o = \frac{1}{N}\sum_{t=1}^{N}\left([y_o]_t - [\hat{y}_o{}^d]_t\right)^2 \quad \forall \quad o = 1, \dots, m \tag{2.21}$$

Where N is the number of observations (samples). This gives a vector of MSE for the m output channels, i.e.

$$MSE = \begin{bmatrix} MSE_1 \\ \vdots \\ MSE_m \end{bmatrix} \in \mathbb{R}^m \tag{2.22}$$

The closer to zero the MSE number gets, the better the model becomes.

2.4.2 Optimal Prediction Error

The identified state space model needs to be analyzed to see how well it can perform in future predictions from unknown data. One of the criteria used in this analysis is presented below.

The prediction Error

The prediction error of the model is computed by first simulating the optimal predictor (the Kalman filter), i.e.

$$x_{k+1} = Ax_k + Bu_k + K\overbrace{(y_k - Dx_k - Eu_k)}^{e_k} \tag{2.23}$$

$$\bar{y}_k = Dx_k + Eu_k \tag{2.24}$$

Which gives the prediction error as

$$e_k = y_k - \bar{y}_k \ \forall\ k = 1, \dots, N \tag{2.25}$$

The size of this error is measured as the size of the covariance matrix

$$\Delta = \frac{1}{N-1}\sum_{k=1}^{N} e_k e_k^T = \frac{1}{N-1} E^T E \tag{2.26}$$

Where $E = Y - \bar{Y}$ is an $N \times m$ matrix of the prediction error at the N discrete time instants. This is a true expected estimate of the exact covariance matrix $\Delta_0 = E(e_k e_k^T)$.

For systems with multiple outputs, the trace operator can be used (Ruscio, 2014), i.e.

$$V_N = trace(\Delta) \tag{2.27}$$

Similarly, the size of the simulated error covariance matrix can be measured as

$$V_N^d = trace(\Delta^d) \tag{2.28}$$

Scaled Prediction Error

The scaled prediction error between the measured output and the predicted output is given by

$$V = \frac{1}{N_{val}} \sum_{k=1}^{N_{val}} (y_k - \bar{y}_k)^T (y_k - \bar{y}_k) \tag{2.29}$$

Where $\bar{y}_k$ is the model predicted output. The closer to zero the value of V is, the better the predictions.

3 Modeling and Identification

The aim of this chapter is to present the identified models used for the identification process and explain how they are obtained. The methods and tools used are described including the data collection and systemization.

It may be necessary sometimes to estimate a model on line at the same time as the input-output data is received which leads us to recursive identification.

A typical recursive identification algorithm is given as

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)(y(t) - \hat{y}(t)) \tag{3.1}$$

where $\hat{\theta}(t)$ is the parameter estimate at time t, and $y(t)$ is the observed output at time t. Moreover, $\hat{y}(t)$ is a prediction of the value based on observations up to time $t-1$ and based on the current model (and possibly also earlier ones) at time $t-1$. The gain $K(t)$ determines how the current prediction error $y(t) - \hat{y}(t)$ affects the update of the parameter estimate (Ljung, 2000). Section 11 in (Ljung, 1999). In the following section, we will look more into some available recursive identification methods.

3.1 The Modeling procedure

For the identification of a system we need measurement data, a model set, an identification criterion and the tools to obtain the identified model. The outcome of the identification can be further fine-tuned by the user by choosing the right model set (model structure and order), this takes place after the measurements have taken place.

The final model should also be adequate for the intended purpose, which may be simulation, prediction, diagnostics, controller design or some other goal. For any of these applications it may be necessary to emphasize the accuracy of the model. The applied input signal has to assure that these frequencies are indeed excited. (Aarts, 2012).

3.1.1 Data collection and systemization

System identification primarily requires that the data captures the important dynamics of the system. A good experimental design would ensure that the right variables are measured with sufficient accuracy and duration to capture the dynamics to be modeled. In general, the experiment must have the following properties (MathWorks, 2016):

1. Use inputs that excite the system dynamics adequately.
2. Measure data long enough to capture the important time constants.
3. Set up data acquisition system to have good signal-to-noise ratio.
4. Measure data at appropriate sampling intervals or frequency resolution.(Ljung, 2000)

The measured data can be divided into two datasets, one for Model validation analysis while the other can be used for prediction analysis.

For Online Parameter Estimation at the Command Line, the workflow is thus:

1. Choose a model structure for your application.
   Ideally, you want the simplest model structure that adequately captures the system dynamics. There are many structures to select from such as the transfer function, State space, or non-linear networks but for the thesis work, we focus on the state space models and therefore we convert the identified polynomial models to the state space form for the sake of comparison

2. Create an online estimation System object for your model structure by using the following commands:
   recursiveARX — usually SISO or MISO ARX model

   The MATLAB syntax for implementing this is:
   `obj = recursiveARX`
   You can specify additional object properties such as the recursive estimation algorithm and initial parameter guesses.

3. Acquire input-output data in real time.
   Specify estimation output data, y, as a real scalar, and input data, u, as a real scalar or vector.

4. Preprocess the estimation data.
   Estimation data that contains deficiencies can lead to poor estimation results. Data deficiencies include drift, offset, missing samples, equilibrium behavior and outliers. Preprocess the estimation data as needed.

5. Update the parameters of the model using incoming input-output data.
   Online estimation algorithms estimate the parameters and states of a model when new data is available during the operation of the physical system. The System Identification Toolbox™ software uses linear, extended, and unscented Kalman filter algorithms for online state estimation. This toolbox, by default, uses recursive prediction error minimization algorithms for online parameter estimation.

   The `step` command is used to execute the specified recursive algorithm over each measurement of input-output data. The MATLAB syntax for implementing this is:
   `[A,B,yhat] = step(obj,y,u)`
   The output of the `step` command gives the estimated parameters (`A` and `B`), and estimated model output (`yhat`), at each set of input-output data.

6. Post-process estimated parameters.
   If necessary, its advised to post-process the estimated parameters by using a low-pass filter to smooth out noisy parameter estimates.

7. Validate the online estimation.

This involves performing several performance analyses.

8. Use the estimated parameters for your application.

3.1.2 Input signals

There are various approaches in applying the input signal for dynamic systems such as:

- RGS: Random, Gaussian Signal: discrete white noise with a flat spectrum.
- RBS: Random, Binary Signal.
- PRBS: Pseudo-Random, Binary Signal.
- SINE: sum of harmonic signals (sine functions).
- The Up-Down signal.

However, we will focus on the Pseudo Random Binary input Signals (PRBS) and also we will examine some criteria for measuring the quality of different experimental design. A great advantage of these signals is that it is easy to implement in practice and are therefore suitable for real identification experiments (Ruscio, 2014).

Pseudo-Random, Binary Signals:

This signal can be generated easily by a computer algorithm using n shift registers and modulo-2 addition for a given initial condition (Total number of samples, Total number of intervals, Minimum sample interval, Maximum sample interval, maximum input amplitude, Minimum input amplitude) and binary coefficients. The output is a deterministic periodic signal. It is called "pseudo-random" as it exhibits some properties that approximates a random signal therefore it is desirable that the signal is a "Maximum length PRBS" which means that the period of the signal is as large as possible. The Maximum period equals $M = 2^n - 1$ samples and a PRBS is a maximum length PRBS if the coefficients are chosen correctly (Aarts, 2012).

The binary signal $s(t)$ generated by this algorithm has output values 0 and 1, however, it can be transformed into a signal $u(t)$ shown in Figure 3.1 with amplitude c and mean m where the equation of the signal is

$$u(t) = m + c(-1 + 2s(t)) \tag{3.2}$$

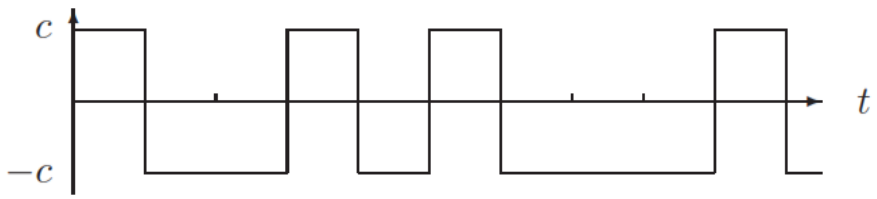

Figure 3.1: PRBS signal

Assume that an input signal series is given as

$$u_k \in \mathbb{R}^r \ \forall \ 1 \le k \le N \tag{3.3}$$

From the input series, we can measure the quality of the input signal by defining the input data matrix with $n+g$ block rows and $K = N - n - k$ block columns as:

Known data matrix of input variables

$$U_{k|n+g} \overset{\text{def}}{=} \overbrace{\begin{bmatrix} u_k & u_{k+1} & u_{k+2} & \cdots & u_{k+K-1} \\ u_{k+1} & u_{k+2} & u_{k+3} & \cdots & u_{k+K} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u_{k+n+g-2} & u_{k+n+g-1} & u_{k+n+g} & \cdots & u_{k+n+K+g-3} \\ u_{k+n+g-1} & u_{k+n+g} & u_{k+n+g+1} & \cdots & u_{k+n+K+g-2} \end{bmatrix}}^{} \in \mathbb{R}^{(n+g)r \times K} \tag{3.4}$$

The input signal u_k is said to be exciting of order n if and only if the matrix $U_{k|n+g}$ is non-singular (Ruscio, 1998). i.e.

$$rank\left(U_{k|n+g}\right) = (n+g)r \tag{3.5}$$

Where g is a prescribed model structure parameter with values

$$g = \begin{cases} 0 & if\ E = 0_{m \times r} \\ 1 & if\ E \neq 0_{m \times r} \end{cases}$$

Or if and only if the matrix

$$P_n = \frac{1}{K} U_{k|n+g} U_{k|n+g}^T \in \mathbb{R}^{(n+g)r \times (n+g)r} \tag{3.6}$$

is non-singular.

An optimal input design is an input signal with minimum condition number (cond ($U_{n|n+g}$) or cond (P_n)) subject to some constraints (Ruscio, 1998).

3.2 Data preprocessing

Estimation data that contains deficiencies can lead to poor estimation results. Data deficiencies include drift, offset, missing samples, equilibrium behavior, seasonality, and outliers. For recursive identification, it is difficult to account for trends and missing values due to the data being collected in the same time as the identification is taking place. This however, makes it possible for the model to also account for the noise and uncertainties in the data. Some further data preprocessing that may improve the quality of the online identification procedure includes checking for the following (Aarts, 2012):

- **Delays:** Delays may arise from several sources including the measurement equipment or the ZOH discretization. A known delay can be removed from the data collection process by compensation but it has to be assured that the delay is not overestimated as this would result in a-causal models.
- **Filtering:** Filtering is important before and after data acquisition. If necessary analog low-pass filters should be used as an anti-aliasing and/or noise filter during the

measurement. The cut-off frequency of an (analog) anti-aliasing filters should match the original sample frequency.

3.3 Identification methods

The 4 methods selected for comparison in this thesis work are described in the following subsections. These are selected primarily because they can handle multiple input systems as most of the systems under consideration are MIMO systems:

- Recursive Deterministic and Stochastic system identification and Realization (RDSR)
- Recursive Artificial Neural Network (RANN)
- Recursive Auto Regressive with exogenous input (RARX)
- Recursive Prediction Error Method (RPEM)

3.3.1 RDSR Algorithm

This algorithm is the recursive formulation of the (subspace) implementation of the Combined Deterministic and Stochastic system identification and Realization (DSR) algorithm presented in (Ruscio, 2012) which results in the name Recursive DSR (RDSR) algorithm.

This algorithm is based on a "short" sliding window illustrated in Figure 3.2 with input and output data vectors. The sliding window is considered short because it is much less than a typical length of time series used in the batch identification method (Ruscio, 1998).

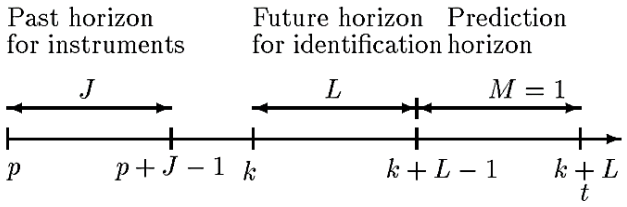

Figure 3.2: The sliding Window

Note that the present time instant in Figure 3.2 is $t = k + L$.

In the DSR algorithm, there are two important horizon parameters L and J needed by the algorithm however only L is important and necessarily needs to be specified in the RDSR algorithm.

- L is the horizon used to predict the number of states n in the system. The system order is bounded by $1 \leq n \leq Lm$ where m is the number of output variables. A rule of thumb is to choose L as small as possible, i.e. so that Lm is as close to n as possible.
- J is the past horizon used to define the instruments needed to remove noise from the data. For most normal noisy process data, we can put $J = L$ or $J < L$ for noise free (deterministic) data.

The number of samples in the input data window is $L + J + g$ and the number of samples in the output data window is $L + J + 1$. g is a discrete model parameter which is defined based on the direct input feed-through term (model matrix) E.

$$g = \begin{cases} 0 \; E = 0_{m \times r} \; i.e. & System\ is\ "proper" \\ 1 \; E\ is\ Estimated \; i.e. & System\ is\ "Strictly\ proper" \end{cases}$$

Let's take t as the present discrete time instant, the array with process data is stored for the output data window at each sample time as

$$Y^w_{t-(L+J)|L+J+1} = \begin{bmatrix} y^T_{t-(L+J)} \\ \vdots \\ y^T_t \end{bmatrix} \in \mathbb{R}^{(L+J+1) \times m} \tag{3.7}$$

For the input sequence, it is stored as

$$U^w_{t-(L+J)|L+J+g} = \begin{bmatrix} u^T_{t-(L+J)} \\ \vdots \\ u^T_{t+g-1} \end{bmatrix} \in \mathbb{R}^{(L+J+g) \times r} \tag{3.8}$$

Where the notations Y^w_t and U^w_t are used to represent the sliding window data matrices. It should however be noted that only a few samples are to be stored in the data windows.

Basic Matrix definitions for Subspace method

It is assumed that the process can be described by the following linear, discrete time invariant state space model (SSM) on innovation form.

$$x_{k+1} = Ax_k + Bu_k + CFv_k \tag{3.9}$$

$$y_k = Dx_k + Eu_k + Fv_k \tag{3.10}$$

where the integer $k \geq 0$ is discrete time, $x_k \in R^n$ is the state vector, $u_k \in R^r$ is the input vector, $v_k \in R^l$ is an external input white noise vector which satisfies $E(v_k v_k^T)^T = I$ i.e. it has unit covariance matrix and $y_k \in R^m$ is the output vector. $C_k = CF_kF_k^{-1}$ is the Kalman gain matrix.

The constant matrices in the SSM are of appropriate dimensions. A is the state transition matrix, B is the input matrix, C is the external input matrix, D is the output matrix and E is the direct input to output matrix and F is the direct external input to output matrix (Ruscio, 1998).

The following assumptions are stated:

- The pair (D, A) is observable.
- The pair $(A, [B \;\; C])$ is controllable.

With the assumptions above, the state vector x_k can be eliminated from (3.9). This results in the following Extended SSM model:

$$y_{k+1|L} = \tilde{A}_L y_{k|L} + \tilde{B}_L u_{k|L+g} + \tilde{C}_L v_{k|L+1} \tag{3.11}$$

Where the terms in the ESSM are given as:

$$\tilde{A}_L = \begin{bmatrix} 0 & I & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & I \\ A_1 & A_2 & \cdots & A_{L-1} & A_L \end{bmatrix} \in \mathbb{R}^{Lm \times Lm} \tag{3.12}$$

$$\tilde{B}_L = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 \\ B_0 & B_1 & \cdots & B_{L+g-2} & B_{L+g-1} & B_{L+g} \end{bmatrix} \in \mathbb{R}^{Lm \times (L+g)r} \tag{3.13}$$

$$\tilde{C}_L = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 \\ C_0 & C_1 & \cdots & C_{L-1} & C_L & C_{L+1} \end{bmatrix} \in \mathbb{R}^{Lm \times (L+1)m} \tag{3.14}$$

$$y_{k|L} = \begin{bmatrix} y_k \\ y_{k+1} \\ \vdots \\ y_{k+L-1} \end{bmatrix} \in \mathbb{R}^{Lm} \tag{3.15}$$

$$u_{k|L+g} = \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+L+g-2} \\ u_{k+L+g-1} \end{bmatrix} \in \mathbb{R}^{(L+g)m} \tag{3.16}$$

The extended external input vector $v_{k|L+1}$ is defined similarly. The integer L which defines the number of block rows in the extended state vectors must satisfy $L \geq L_{min}$ where the minimum number of block rows is defined by

$$L_{min} \stackrel{\text{def}}{=} \begin{cases} n - rank(D) + 1 & when\ m < n \\ 1 & when\ \ m \geq n \end{cases} \tag{3.17}$$

This ESSM (Extended State Space Model) model representation is more suitable for system Identification because it gives an equation where the states are eliminated therefore the system model can be recovered directly from the known inputs and outputs data (Ruscio, 1998).

In subspace identification, we want to find the system order n, the initial state vector x_0 and matrices in the ESSM (up to within a similarity transformation) A, B, C, D, E, F.

An essential step is the reconstruction of the (extended) observability matrix O_L which is estimated from the column space of a data matrix $Z_{k|L}$ (see (Ruscio, 2014)) defined as:

$$Z_{k|L} = O_L X_k \ \in \ \mathbb{R}^{Lm \times K} \tag{3.18}$$

Where

$$O_n = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{3.19}$$

The system order n is also identified as the dimension of the column space of $Z_{k|L}$.

$$rank(Z_{k|L}) = \dim[R(Z_{k|L})] = n \tag{3.20}$$

Once the rank and the observability matrix are known, it is easy to determine C and A. Output matrix C equals the top part of this matrix and this result can be used to compute A from the rest. Furthermore, the states x and the noise contributions can be estimated (Aarts, 2012).

In some cases, it is not desired to identify the direct input to output matrix E. If E is apriori known to be zero, then usually the other model matrices can be identified with higher accuracy if E is not estimated. For this reason, define the integer structure parameter g as follows

$$g \stackrel{\text{def}}{=} \begin{cases} 1 & when \quad E \neq 0_{m\times r} \\ 0 & when \quad E = 0_{m\times r} \end{cases} \tag{3.21}$$

Obtaining a good model of your system depends on how well your measured data reflects the behavior of the system. In practice, both high frequency process disturbances v_k and measurement noise w_k are present and has effect on the output series y_k.(Ruscio, 1998)

Online estimate is needed when the system parameters are time-varying, , in which case the subspace identification will encounter the problem of large computation complexity. This is because the subspace identification mainly uses QR decomposition and singular value decomposition (SVD) of linear algebra, and the calculation complexity of the SVD of the specific data matrix is great during the identification process. (Ruscio, 2014)

The R-DSR toolbox comes preloaded with several functions as m-files[1] to perform the required computations but only two will be considered to give an overview as to how the algorithm identifies the system recursively.

i. **dsr_upr.m** (updates the lower triangular R matrix)
   This function computes and updates the square lower triangular matrix ($R_t \in \mathbb{R}^{nr\times nr}$) from the input data window, the output data window and the previous R_{t-1} matrix. It should be noted that the new values for vectors u_t and y_t are fed into the data window in a first-in first-out strategy.
ii. **dsr_r2m.m** (Transforms the R_t matrix to State Space model matrices).
   This function computes the linear discrete time state space model matrices $(A_t, B_t, D_t, E_t, C_t, F_t)$ and the initial state vector $x_{t-(L+J)}$ from the square lower triangular matrix R_t.

After the first recursion (iteration), an estimate of the state space model matrices can be obtained but this model is obviously inaccurate due to measurement noise so from a stochastic

[1] MATLAB® Scripts

point of view, we need an infinite number of samples and recursions in order to identify a model with consistent and efficient parameter estimates however, from a deterministic point of view, the number of samples needed is finite. The RDSR algorithm needs at most

$$N_{min} = J + L + (L + g)r + n \tag{3.22}$$

samples to properly identify a (deterministic) system therefore, the state space model matrices should be computed for $t \geq N_{min}$ when using the recursive identification algorithm (Ruscio, 1998).

The final model, i.e. $(A_N, B_N, ...)$ generated from this RDSR algorithm is identical to the DSR model, i.e. the model computed from the batch algorithm.

The algorithm can therefore be summarized in the following steps (Ruscio, 2014).

i. Construct the Extended State Space Model from the input/output sequence
ii. Compute the QR decomposition from known data matrices in ESSM
iii. Separate the data into three parts:
    a. One part for analyzing and determination of system dynamics i.e. the system order n and the state equation system matrix $\tilde{A}_L$

$$R_{42} = \tilde{A}_L R_{32} \tag{3.23}$$

    b. One part for determination of the deterministic part of the system

$$R_{41} - \tilde{A}_L R_{42} = \tilde{B}_L R_{11} \tag{3.24}$$

    c. One part for determination of the stochastic part of the system

$$R_{44} = \tilde{C}_L E_L Q_4^T \tag{3.25}$$

$$R_{43} - \tilde{A}_L R_{33} = \tilde{C}_L E_L Q_3^T \tag{3.26}$$

iv. Extract the system matrices from sub-matrices in R
    a. System order: n and matrix D

$$R_{32} = \overbrace{[\,U_1 \;\; U_2] \begin{bmatrix} S_1 & 0 \\ 0 & S_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}}^{SVD} \quad \begin{cases} n & \text{non zero singular values.} \\ O_L = U_1 & \text{Extended observability matrix} \\ D = O_L(1{:}\,m, :) & \end{cases} \tag{3.27}$$

    b. Matrix A

$$R_{42} = \tilde{A}_L R_{32} \;\Big\{\; \tilde{A}_L = O A (O^T O)^{-1} O^T \tag{3.28}$$

    c. Matrices B and E

$$R_{41} - \tilde{A}_L R_{31} = \tilde{B}_L R_{11} \rightarrow cs\left(\begin{bmatrix} B \\ E \end{bmatrix}\right) = N^{+} cs(R_{41} - \tilde{A}_L R_{31}) \tag{3.29}$$

    d. Matrices C and F

$$R_{44} \rightarrow F \text{ and } E(\epsilon_k \epsilon_k^T) = F F^T \tag{3.30}$$

$$R_{43} - \tilde{A}_L R_{33} = \tilde{C}_L E_L Q_3^T \rightarrow C \tag{3.31}$$

The MATLAB® syntax for RDSR (Recursive DSR algorithm) is:

```
[A,B,D,E,C,F,x,s,R]=rdsr(Y,U,L,g,J,n);
```

where `Y` and `U` of size ($N \times m$) and ($N \times r$) are the given output and input time series data matrices respectively. N is the number of observations and m is the number of output variables, r is the number of input variables. `L` is the horizon used to predict the number of states which is also equal to the number of block rows in the extended observability matrix. `g` is the model structure parameter. `J` is the past horizon used to form instruments used to remove noise. `n` is the system order which is equal to the number of states.

On the output side, `A, B, D, E, C, F` are the State space model matrices. `x` is the state vector, $x_{t-(L+J)}$ for $t > L + J$. `s` is the array of size ($N \times Lm$) of singular values which can be used for system order decision. `R` - The parameters in the R_t matrix of size ($nr \times nr$) is stored in the t[th] row in the R matrix of size ($N \times (nr * nr)$) the sizes of the R matrices are presented in

Table 3.3.1.

Table 3.3.1: Sizes of the R matrices for the three systems under consideration

Process	***nr***
Synthetic Model (MIMO)	20
Quadruple tank (MIMO)	20
Air heater(SISO)	8

3.3.2 Using the System Identification Toolbox

For SISO systems, transfer function models provide a more compact representation of a system. Although the subspace models for SISO systems can be easily transformed to a transfer function, the result may be "suboptimal" as seen in the result section of this report. That means that another model structure may give better results. In this section, the Prediction-Error identification Methods (PEM) will be discussed that estimate the parameters of models directly in a transfer function format (Aarts, 2012).

The starting point for the PEM-models is the assumption that the unknown system G_0 can be represented by a linear time invariant system of a finite order (LTIFD system). The discrete time input signal $u(t)$ and output signal $y(t)$ are measured. The output $y(t)$ contains the response of the system on the input signal, plus there is a contribution from an unmeasurable disturbance $v(t)$, so we have

$$y(t) = G_0(z)u(t) + v(t) \tag{3.32}$$

Where the system is written as a transfer function $G_0(z)$. The disturbance $v(t)$ can originate from several sources like measurement noise, effects of non-measured inputs, process disturbances and non-linearity. This disturbance signal will be represented as a certain noise model written as

$$v(t) = H_0(z)e(t) \tag{3.33}$$

In which $e(t)$ is a white noise and $H_0(z)$ is a stable minimum phase transfer function. This leads to the assumed structure of the system

$$y(t) = G_0(z)u(t) + H_0(z)e(t) \tag{3.34}$$

It is assumed that the data from the real system can be considered to be generated by a process model $G_0(z)$ and a noise model $H_0(z)$ as expressed by this equation.

The goal therefore of the system identification method (RPEM) is to determine transfer functions $G(z)$ and $H(z)$ from the measured input and output data in a recursive manner.

As the name of the PEM-models suggest, the prediction error of the model is an important factor. This prediction error is the difference between the output $y(t)$ of the system as would be predicted by the model and the actual measurement. So first we need to know the prediction of our model (Aarts, 2012).

Model structures

The MATLAB® System Identification Toolbox provides the following functions that implement all common recursive identification algorithms for the Polynomial Black-Box Model structures defined in the general input-output form as

$$A(q)y(t) = \frac{B(q)}{F(q)}u(t - nk) + \frac{C(q)}{D(q)}e(t) \tag{3.35}$$

Where $A(q), B(q), C(q), D(q)$ and $F(q)$ are polynomials with their coefficients ordered by descending powers. q is the time shift operator while nk is the delay.

$$\begin{aligned} A(z) &= 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{n_a} z^{-na}, \\ B(z) &= b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{n_b} z^{-nb}, \\ C(z) &= 1 + c_1 z^{-1} + c_2 z^{-2} + \cdots + c_{n_c} z^{-nc}, \\ D(z) &= 1 + d_1 z^{-1} + d_2 z^{-2} + \cdots + d_{n_d} z^{-nd}, \\ F(z) &= 1 + f_1 z^{-1} + f_2 z^{-2} + \cdots + f_{n_f} z^{-nf}. \end{aligned} \tag{3.36}$$

The autoregressive component, A, is common between the measured and noise components. The polynomials B and F makes up the measured component while the polynomials C and D makes up the noise component.

These are called the **idpoly** model representations, in MATLAB® they are: **rarmax**, **rarx**, *Recursive Box-Jenkins* (**rbj**) models, *Recursive Prediction Error Method* (**rpem**) models, *Recursive pseudo-linear regression* (**rplr**) models, and *Recursive output error* (**roe**) models.

The details about the different algorithms listed above is beyond the scope of this report, see (Ljung, 2000) for more details. Table 3.2 shows an overview of the Different available polynomial model structures, see (Aarts, 2012). They all share the same basic syntax for MATLAB® implementation:

```
[thm,yh] = rfcn(z,nn,adm,adg)
```

Where `rfcn` specifies the recursive function being used, `z` contains the output-input data. `nn` is the chosen model structure, same as the one for the corresponding offline algorithm. The arguments `adm` and `adg` are optional and they select the adaptation mechanism and adaptation gain respectively. The output argument `thm` is a matrix that contains the current models at the different samples while the output argument `yh` is a column vector that contains the predicted values of $y(k)$, based on past observations and current model.

Table 3.2: Overview of Polynomial model structures

Name	equation	$G(z,\theta)$	$H(z,\theta)$
ARX	$A(z^{-1})\,y(t) = B(z^{-1})\,u(t) + e(t)$	$\frac{B(z^{-1},\theta)}{A(z^{-1},\theta)}$	$\frac{1}{A(z^{-1},\theta)}$
ARMAX	$A(z^{-1})\,y(t) = B(z^{-1})\,u(t) + C(z^{-1})\,e(t)$	$\frac{B(z^{-1},\theta)}{A(z^{-1},\theta)}$	$\frac{C(z^{-1},\theta)}{A(z^{-1},\theta)}$
OE	$y(t) = \frac{B(z^{-1})}{F(z^{-1})}\,u(t) + e(t)$	$\frac{B(z^{-1},\theta)}{F(z^{-1},\theta)}$	1
FIR	$y(t) = B(z^{-1})\,u(t) + e(t)$	$B(z^{-1},\theta)$	1
BJ	$y(t) = \frac{B(z^{-1})}{F(z^{-1})}\,u(t) + \frac{C(z^{-1})}{D(z^{-1})}\,e(t)$	$\frac{B(z^{-1},\theta)}{F(z^{-1},\theta)}$	$\frac{C(z^{-1},\theta)}{D(z^{-1},\theta)}$

It should be noted that **rarx** is a recursive variant of **arx**; similarly, **rarmax** is the recursive counterpart of **armax** and so on. Note also that **rarx** does not handle multi-output systems, and **rpem** does not handle state-space structures. More details can be found in (Ljung, 2000).

RPEM

The Prediction Error method is used to estimate the parameters and the RPEM is the recursive implementation where the accuracy of the predictions computed for the observed data is minimized with respect to the parameter vector, θ the cost function, i.e., a weighted norm of the prediction error $\varepsilon_F(t,\theta)$.

The general form of the cost function, as described in (Ljung, 1999) is given by:

$$V_N(\theta, Z^N) = \frac{1}{N}\sum_{t=1}^{N} l(\varepsilon_F(t,\theta)) \tag{3.37}$$

Where l is a scalar-valued function, Z^N is a vector containing the measured data and the prediction error is the difference between the observed and predicted outputs, i.e. $\varepsilon_F(t,\theta) = y(t,\theta) - \hat{y}(t,\theta)$.

In MATLAB® the default choice for l is in the quadratic norm: $l(\varepsilon) = \frac{1}{2}\varepsilon^2$ which is used in this study as well, this now makes the cost function to be minimized to be given by:

$$V_N(\theta, Z^N) = \frac{1}{2N}\sum_{t=1}^{N} \varepsilon_F^2(t,\theta) \tag{3.38}$$

And the final parameter estimate is the minimized argument of the function

$$\hat{\theta}_N = \operatorname{argmin}(V_N(\theta, Z^N)) \tag{3.39}$$

These algorithms are also prepared for true on-line applications, where the computed model is used for some on-line decision. This is accomplished by storing the update information and information about past data in some presentable form and then using that information as initial data for the next time step (Ljung, 2000).

3.3.3 Neural Network Approach

Over the past few years, Neural networks have been used to perform complex functions in various fields of application such as pattern recognition, system identification, classification, speech, vision and control systems. They can be trained to solve problems that are usually difficult for conventional computers or human beings.

The training process could be supervised or unsupervised but the supervised training methods are commonly used, other networks can be obtained from unsupervised training techniques or from direct design methods. Certain other kinds of linear networks and Hopfield networks can be designed directly.

In situations where the system has some quite complicated nonlinearities, which cannot be realized on physical grounds, nonlinear, black-box models could be a solution. Among the most used models of this character are the Artificial Neural Networks (ANN) (Howard Demuth, 2002).

Neural networks consist of simple elements operating in parallel. The ideas are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements as seen in Figure 3.3 (Howard Demuth, 2002). We can train a neural network to perform a function by adjusting the values of the connections (weights and biases) between elements.

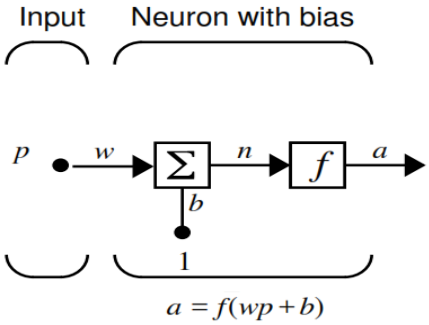

Figure 3.3: A Simple Neural Network Model

The input p is transmitted through a connection that multiplies its strength by the weights w, to form the product wp added to the bias, b. You may view the bias as simply being added to the product wp as shown by the summing junction in Figure 3.3 or as shifting the function f to the left by an amount b. The bias is much like a weight, except that it has a constant input. The transfer function net input n, is the sum of the weighted input wp and the bias b. This sum is the argument of the transfer function f typically a step function or a sigmoid function, which takes the argument n and produces the output a. Note that w and b are both the adjustable parameters of the neuron.

The central idea of neural networks is that such parameters can be adjusted so that the network exhibits some desired or interesting behavior. Thus, we can train the network to do a job by adjusting the weight or bias parameters, or perhaps the network itself will adjust these parameters to achieve some desired end. See (Howard Demuth, 2002) for more details.

The MATLAB® Neural Network Toolbox contains various functions to help in implementing the algorithms.

The network can be trained recursively (online) or in a batch. Batch training of a network proceeds by making weight and bias changes based on an entire set (batch) of input vectors. For recursive implementation, we use the Incremental training which changes the weights and biases of a network as needed after presentation of each individual input vector (Sample). Incremental training is sometimes referred to as “on line” or “adaptive” training. (Howard Demuth, 2002).

The algorithm has 3 main steps:

1. **Network initialization**
   In this stage, the network to be trained is created. The network used has 10 neurons in the hidden layer as seen in Figure 3.4 which is usually chosen arbitrarily, the number of neurons chosen for the input and output layers depends on the number of input signals in the input vector (P) and target signals in the target vector (T). Predictive models are used for system identification (or dynamic modelling), in which you build dynamic models of physical systems. These dynamic models are important for analysis, simulation, monitoring and control of a variety of systems, including manufacturing systems, chemical processes, robotics and aerospace systems (Howard Demuth, 2002).
   NARX (Nonlinear autoregressive with external input) networks can be trained to predict a time series given past values of the same time series, the feedback input, and

another time series, called the external or exogenous time series. The models are represented mathematically as:

$$y_i = f\left(y_{i-1}, y_{i-2}, \dots, y_{i-n_y}, u_{i-1}, \dots, u_{i-n_u}\right) + e_i, \quad i = 1,2, \dots, N \tag{3.40}$$

Where u_i and y_i denotes the scalar inputs and outputs and e_i are the additive measurement errors with uncorrelated zero-mean.

The success of NARX models is due both to their capability of capturing nonlinear dynamics and the availability of identification algorithms with a reasonably good computational cost.
Note that the output of the NARX network, $y(t)$, is fed back to the input of the network (through delays), since $y(t)$ is a function of $y(t-1), y(t-2), \dots, y(t-d)$. However, for efficient training this feedback loop can be opened since the true output is available during the training of the network, you can use the open-loop architecture shown in Figure 3.4, in which the true output is used instead of feeding back the estimated output.
This has two advantages. The first being that the input to the feedforward network is more accurate and secondly, the resulting network has a purely feedforward architecture, and therefore a more efficient algorithm can be used for training.

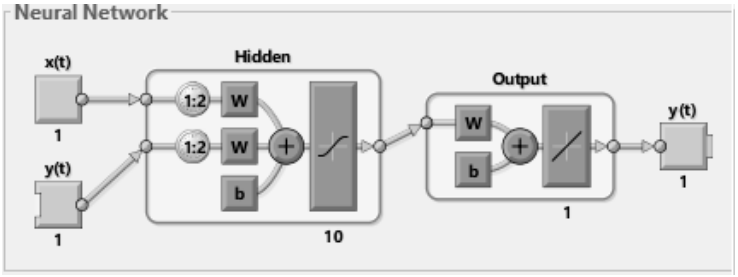

Figure 3.4: NARX Neural Network Model with 10 neurons in the hidden layer and 2 samples time delay.

Classical prediction error approaches for the identification of non-linear polynomial NARX/NARMAX models often yield unsatisfactory results for long-range prediction or simulation purposes, mainly due to incorrect or redundant model structure selection as seen in the results section. The Nonlinear Autoregressive with External (Exogenous) Input (NARX) models are predictive models and are therefore used in this thesis.

2. **Network Training/Adaptation.**
   The network built in the previous step is trained incrementally using the `adapt` function. It is often required to adjust the network learning rate and momentum for optimal adaptation. Here is the MATLAB® code syntax to train the network `net` recursively based on input/target signals `P` and `T`.

```
[net,a,e] = adapt(net,P,T);
```

The output of this function is the trained network `net`, the output `a` and the error `e`.

3. **Network Testing**
   Once the network is adapted, we can perform various model performance testing and plot its output signal and compare it to the target signal. It is important in this application to close the NARX model loop as seen in Figure 3.5 before testing on new inputs.

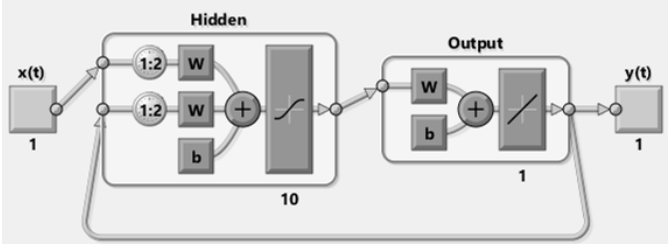

Figure 3.5: NARX Close loop by feeding the predictions back as one of the inputs.

Batch training is significantly faster and produces smaller errors than incremental training.

3.4 Identified models

Depending on the application of the model, accurate models are required. Having as precise models as possible might require a large amount of model parameters. Dealing with many parameters however, increases the model complexity, and thus might lead to prohibitive computational procedures caused by too large execution times. The main point in selecting appropriate models involves finding the right trade-off between model simplicity and accuracy

3.4.1 Models and Initial conditions

It is required that the input delays and model orders are provided as part of the initial conditions to estimate polynomial models. For recursive identification, it is assumed that the model delays and orders are unknown in advance therefore a guess or estimate is usually provided for the parameter values and initial parameter covariance matrix

To get initial model orders and delays for the systems, few measurements are taken from which several ARX or PEM models are estimated with a range of orders and delays and then the performance of these models are quickly compared. The model orders and delays values that corresponds to the best model performance is selected and used as an initial guess for further modeling.

The estimation procedure uses the ARX or PEM model structure, which includes the A and B polynomials, therefore, only estimates for the na, nb, and nk parameters are gotten as seen in Table 3.3.3. These results can however be used as initial guesses for the corresponding polynomial orders and input delays in other model structures.(Ljung, 2000).

After various attempts and comparisons of the identified PEM and ARX models, estimated using different guessed orders and delays, it is observed that combinations of *na* = 1-5 and *nb* = 2-4 generally give very good results with little variation in the model performance for the different combinations of these orders and delays. Higher orders (6-10) improves the performance with about 5-10% in all cases, but the increase in the model complexity due to the large number of parameters in the B-polynomials makes this increase insignificant. It is possible to under-fit (model order is too low) or over-fit (model order is too high) data by choosing an incorrect model order. Normally, the lowest-order model that adequately captures your system dynamics is desired. Under-fitting prevents algorithms from finding a good fit to the model, even if all other estimation settings are good, and there is good excitation of system dynamics. Over-fitting on the other hand leads to high sensitivity of parameters to the measurement noise or the choice of input signals It is therefore important that the simplest model structure that adequately captures the system dynamics is selected as shown in Table 3.3.3.

For the Neural network models, only the number of neurons and time delays are specified initially. For state space model identification, only the model order n is needed which is usually selected from prior knowledge of the system.

Table 3.3.3: Initial Model Selection

System.	***Model.***	***Orders.***			***Delays.***		
Synthetic model	RDSR	4			-		
	RANN	10 Neurons			2 samples		
	RARX	*na*	*nb*	*nc*	*nd*	*nf*	*nk*
		5	[5 5]	-	-	-	[0 0]
	RPEM	3	[3 3]	3	-	-	[1 1]
Quadruple tank	RDSR	4			-		
	RANN	10 Neurons			2 samples		
	RARX	*na*	*nb*	*nc*	*nd*	*nf*	*nk*
		5	[5 5]	-	-	-	[0 0]
	RPEM	2	[2 2]	2	-	-	[1 1]
Air heater	RDSR	2			-		
	RANN	10 Neurons			2 samples		
	RARX	*na*	*nb*	*nc*	*nd*	*nf*	*nk*
		1	1	-	-	-	1

RPEM	2	2	2	-	-	1

It is very important to also check that you have specified appropriate settings for the estimation algorithm. The forgetting factor algorithm is used in this thesis and the forgetting factor, λ, is chosen carefully. If λ is too small, the estimation algorithm assumes that the parameter value is varying quickly with time. Conversely, if λ is too large, the estimation algorithm assumes that the parameter value does not vary much with time. see (Ljung, 2000)

3.5 Comparison between RDSR and traditional methods

The RDSR method has some advantages and disadvantages compared to the prediction error method. One of the advantage is that the model structure of a state space system has no need for explicit definition of the model equations. One only has to select the order of the system matrix. In addition, the model structure is well suited for MIMO systems. Furthermore, the numerical and mathematical approach is elegant (robust, reliable) and efficient.

As the model structure offers little means to apply dedicated model equations, it may appear that the subspace model is sub-optimal but in many cases, the subspace model may be good enough for the intended application. (Aarts, 2012).

In general, the state-space model provides a more complete representation of the system, especially for MIMO systems, than polynomial models because the state-space model is similar to a first principle model. The identification procedure does not involve nonlinear optimization so the estimation reaches a solution regardless of the initial guess. Moreover, the parameter settings for the state-space model are simpler than polynomial models. You need to select only the order, or the number of states, of the model. The order can come from prior knowledge of the system. You also can determine the order by analyzing the singular values of the information matrix.

For MIMO systems, when the model order is high, the algorithm involved in the ARX model estimation is fast and efficient when the number of data points is very large. The state-space model estimation with a large number of data points is slow and requires a large amount of memory. If you must use a state-space model, for example in modern control methods, reduce the sampling rate of the signal in case the sampling rate is unnecessarily high.

The PEM method involves an iterative, nonlinear optimization in the identification procedure. This requires excessive computation time, and the minimization can get stuck at a false local minimum, especially when the order is high and the signal-to-noise ratio is low. However, you can use these models when the stochastic dynamics are important because they provide more flexibility for the stochastic dynamics. (ni.com, 2010)

In summary, three major aspects are considered in comparing the RDSR method with the RPEM method:

1) Parameterizations:
    - Classical RPEM approaches need a certain user-specified model parameterization, so-called canonical forms which could lead to numerically ill-

conditioned problems, meaning that the parameters in the canonical form model is extremely sensitive to small perturbations (Ruscio, 2014).
    - RDSR method need no parameterization.

2) Convergence:
    - The RPEM approach is iterative. Many hard to deal with problems such as problems with lack of convergence; no convergence; slow convergence; local minima; numerical instability;
    - RDSR method is non-iterative, there are no problems with convergence and the method is numerically robust (Ruscio, 2014).

3) Speed:
   RDSR method is generally faster than classical RPEM approaches (because the method is non-iterative).

4 Results

In this part of the study, efficient and stable implementations as well as various Monte Carlo simulations and experimentations are performed and the results are presented and compared using some basic validation criteria for model performance and accuracy as discussed in Section 2.4. Each MIMO and SISO model derived using the different algorithms under consideration are validated separately using the same validation dataset. Results from real world Numerical examples are also presented and the chapter ends with some summary/comments on the effects of using each model based on different applications.

After the linear parametric models as well as nonlinear model parameters are estimated, the model quality can be evaluated by comparing model response to measured response (MathWorks, 2016):

To perform the comparison, a model is estimated and used to simulate some time varying output, i.e., calculate the output $y(t)$ for some given input values to reproduce certain system behavior as closely as possible and then some performance tests are carried out. The estimated model can also be used to compute to some level of accuracy a qualified guess of future output values based on past observations of system's inputs and outputs.

Both simulation and prediction require initial conditions, which correspond to the states of the model at the beginning of the simulation or prediction.

All the toolboxes and algorithms under consideration in this thesis contains methods for estimating the initial conditions from available input and output measurements if the initial conditions are unknown.

4.1 Model Performance Analysis

Simulation means computing the model response using input data and some initial conditions. For online implementation, the sampling time of the model response is made to match the sampling time of the input data sequence used for the simulation.

Calculating the simulated output does not take the past and current outputs into consideration, only the inputs are used to compute the output which corresponds to a prediction horizon $k = \infty$.

Prediction forecasts the model response k steps ahead into the future using the current and past values of measured input and output values. k is called the prediction horizon, and corresponds to predicting output at each sampling time. To predict the model response k steps into the future from the current time t, the inputs up to time $t + k$ and outputs up to time t should be known.

The prediction abilities of models are usually more relevant than the simulation abilities especially when the purpose of the modelling is for control.

The polynomial models derived from the RARX and RPEM algorithms are converted to state space form for the sake of comparison of the derived matrices as presented in the Appendix section but it seems to be impossible to convert the RANN models to state space due to the recursive implementation, however, the results from all the systems and the derived models are presented in the following sections.

4.1.1 Synthetic Model

The synthetic model is a very complex and noisy system as explained in section 2.2.1, it is usually difficult to make accurate models from noisy data. Figure 4.1 shows the input data sequence sent into the model both for estimation and prediction. It's a MIMO system with 2 inputs and 2 outputs. The calculated values of the MSE and fit for this system is presented in Table 4.1 and for simulation, the RDSR and RPEM models produce the models with the least errors and fits as seen in Figure 4.2 but for prediction, the RPEM model outperforms the other models with a MSE value of 3.9417 and a fit of 51.09% for the first output y_1 and 82.95% for the second output y_2 as seen in Figure 4.5. The RDSR model performed relatively well too but the first output seemed to be noisier than the second output so the noise in the data affected its performance.

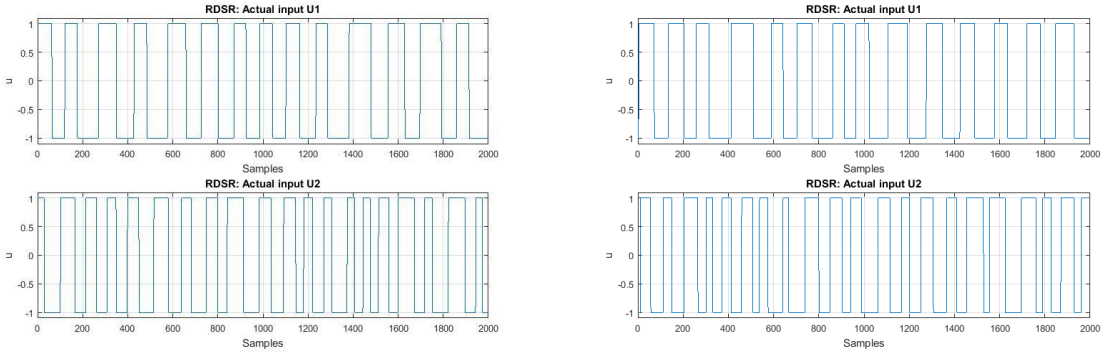

Figure 4.1 Input data for Synthetic model: **Left**: Estimation input sequence, **Right**: Prediction input sequence. The same sequence is used for all algorithms.

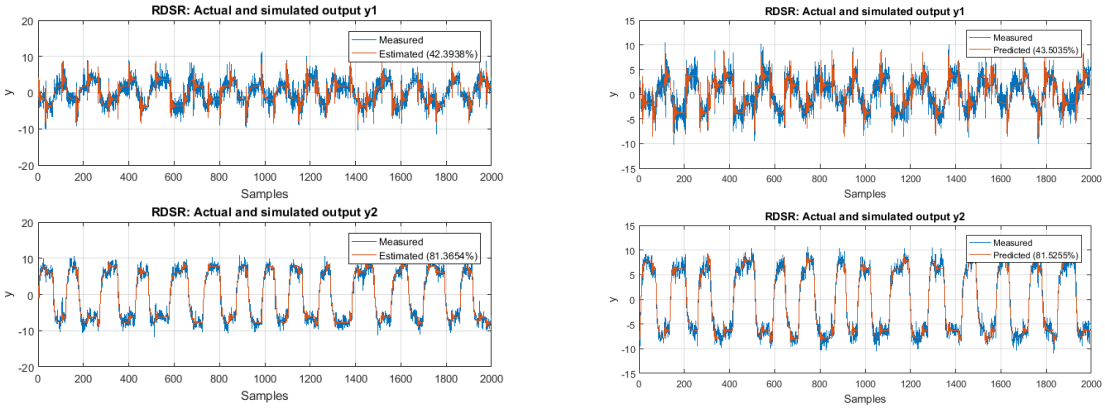

Figure 4.2: Performance of the RDSR model: **Left**: Real measurement and Simulated output for output y_1 (upper, fit: 42.39%) and y_2 (lower, fit: 81.36%) and **Right**: Real measurement and predicted output for output y_1 (upper, fit: 43.50%) and y_2 (lower, fit: 81.53%).

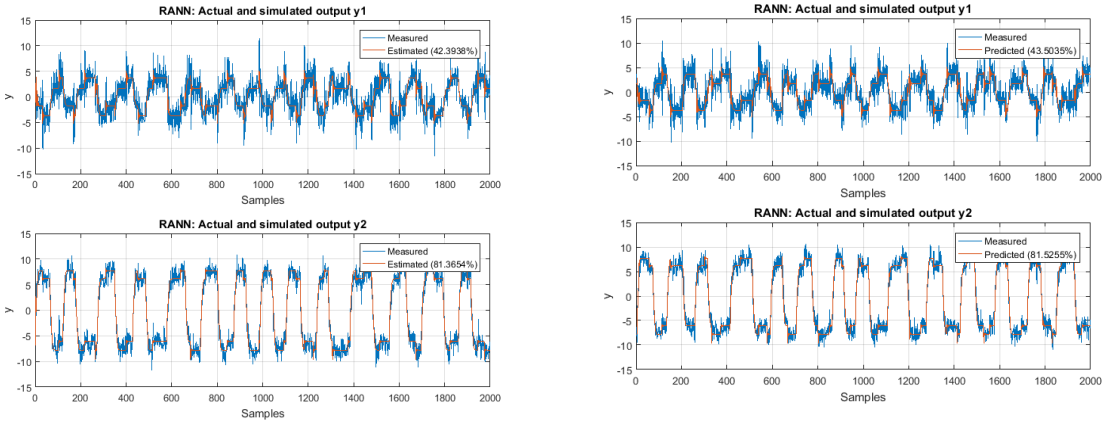

Figure 4.3: Performance of the RANN model: **Left**: Real measurement and Simulated output for output y_1 (upper, fit: 42.39%) and y_2 (lower, fit: 81.3654) and **Right**: Real measurement and predicted output for output y_1 (upper, fit: 43.50%) and y_2 (lower, fit: 81.52%).

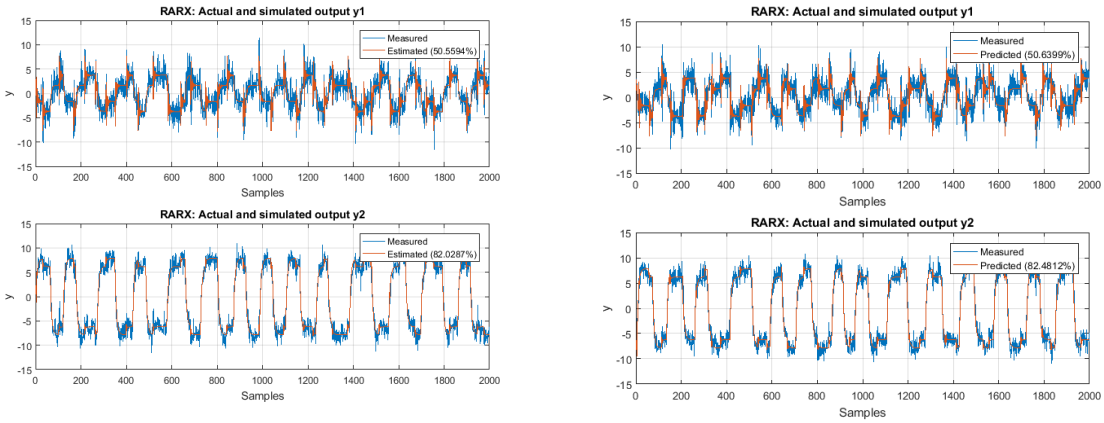

Figure 4.4: Performance of the RARX model: **Left**: Real measurement and Simulated output for output y_1 (upper, fit: 50.56%) and y_2 (lower, fit: 82.03%) and **Right**: Real measurement and predicted output for output y_1 (upper, fit: 50.64%) and y_2 (lower, 82.48%).

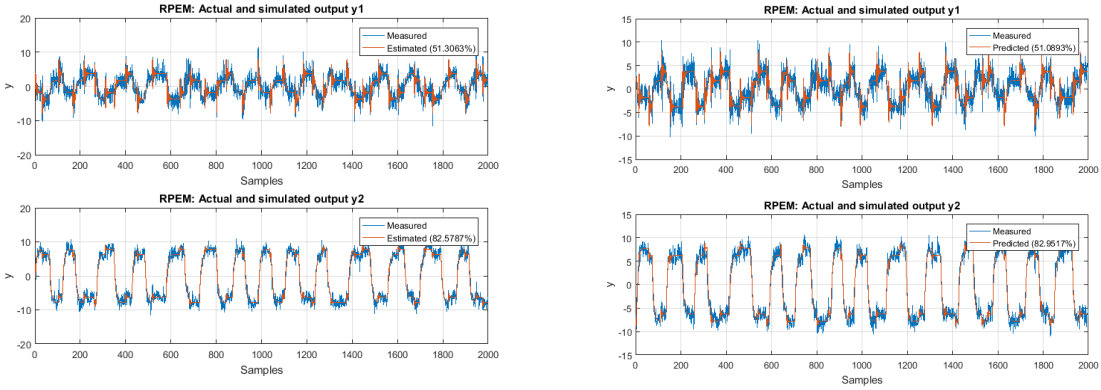

Figure 4.5: Performance of the RPEM model: **Left**: Real measurement and Simulated output for output y_1 (upper, fit: 51.31%) and y_2 (lower, fit: 82.57%) and **Right**: Real measurement and predicted output for output y_1 (upper, fit: 51.09%) and y_2 (lower, fit: 82.95%).

Table 4.1: Validation analysis and comparison for the Synthetic data, expressed in MSE and Goodness of fit

Type of Output	***Model***	***MSE***	***Fit (%)***	
			y_1	y_2
Simulated	RDSR	4.0350	42.3938	81.3654
	RANN	5.4087	42.1081	80.8608
	RARX	4.1892	50.5594	82.0287
	RPEM	4.0207	51.3063	82.5787
Predicted	RDSR	4.0167	43.5035	81.5255
	RANN	5.1533	42.8438	81.4879
	RARX	4.0606	50.6399	82.4812
	RPEM	3.9417	51.0893	82.9517

Generally, all the models performed very well with the RANN model at the bottom position in the rank due to the deficiency of not being able to accurately model the noise in the data but its performance can be improved since the model can always be retrained to get better or worse performance based on the choice of initial parameters.

The state space model parameters identified for this process by all the algorithms are presented in Appendix A.3

4.1.2 Quadruple tank

The quadruple tank process represents a nonlinear process as described in section 2.2.2 and it is a very complex and noisy system. Figure 4.6 shows the input data sequence sent into the process both for estimation and prediction. It's a MIMO system with 2 inputs and 2 outputs. The calculated values of the MSE and fit for this system is presented in Table 4.2 and for simulation, the RDSR models produce the best models with the least errors and fits as seen in Figure 4.7. For prediction, the RANN model outperforms the other models with a MSE value of 14.1287 and a fit of 35.159% for the first output y_1 and 33.237% for the second output y_2 as seen in Figure 4.8Figure 4.5. The RDSR model performed relatively well here too but the second output produced a negative value of fit which may be an indication of too much nonlinearity in the process and that affected its performance.

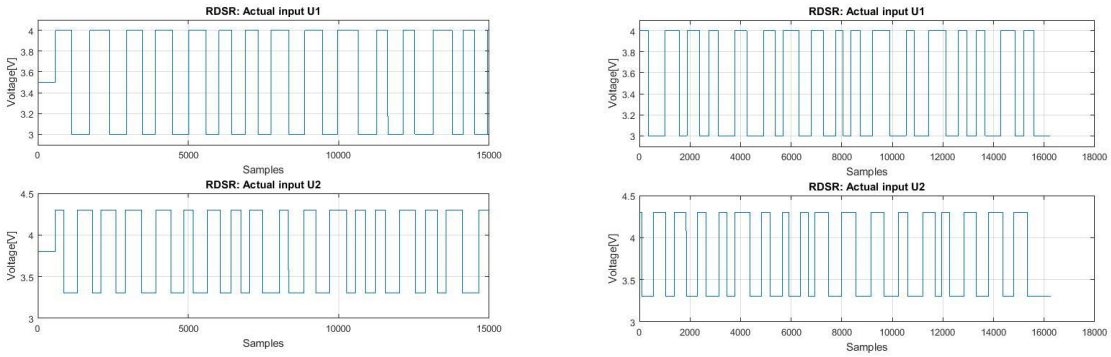

Figure 4.6 Input data for the quadruple tank: **Left**: Estimation input sequence, **Right**: Prediction input sequence. The same sequence is used for all the other algorithms.

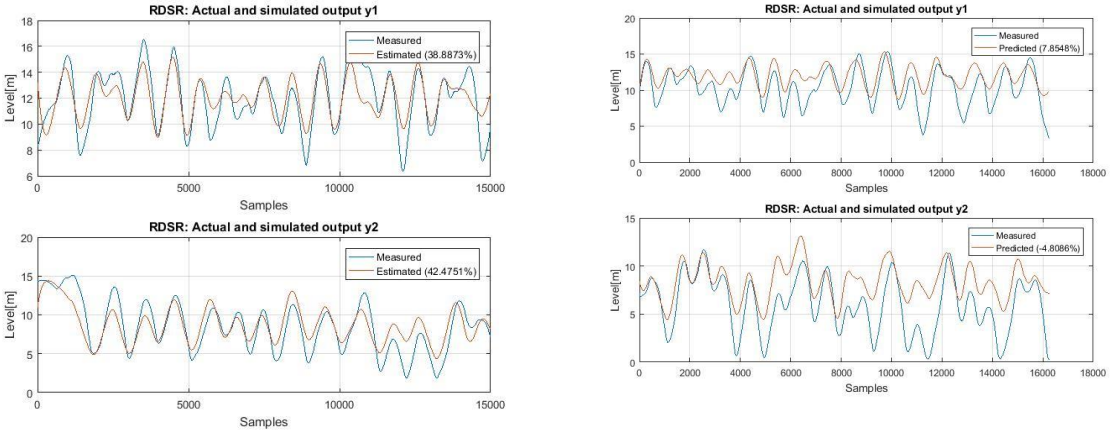

Figure 4.7: Performance of the RDSR model: **Left**: Real measurement and Simulated output for output y_1 (upper, fit: 38.89%) and y_2 (lower, fit: 42.48%) and **Right**: Real measurement and predicted output for output y_1 (upper, fit: 7.85%) and y_2 (lower, fit: -4.80%).

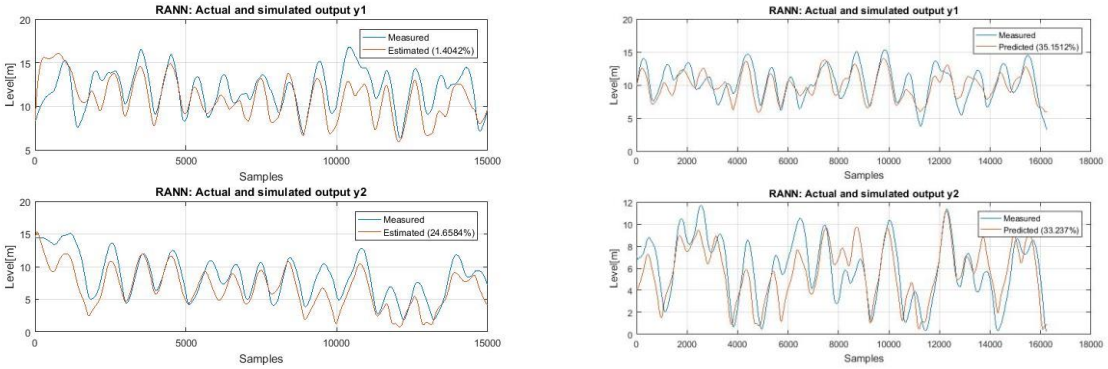

Figure 4.8: Performance of the RANN model: **Left**: Real measurement and Simulated output for output y_1 (upper, fit: 1.40%) and y_2 (lower, fit: 24.66%) and **Right**: Real measurement and predicted output for output y_1 (upper, fit: 35.15%) and y_2 (lower, fit: 33.24%).

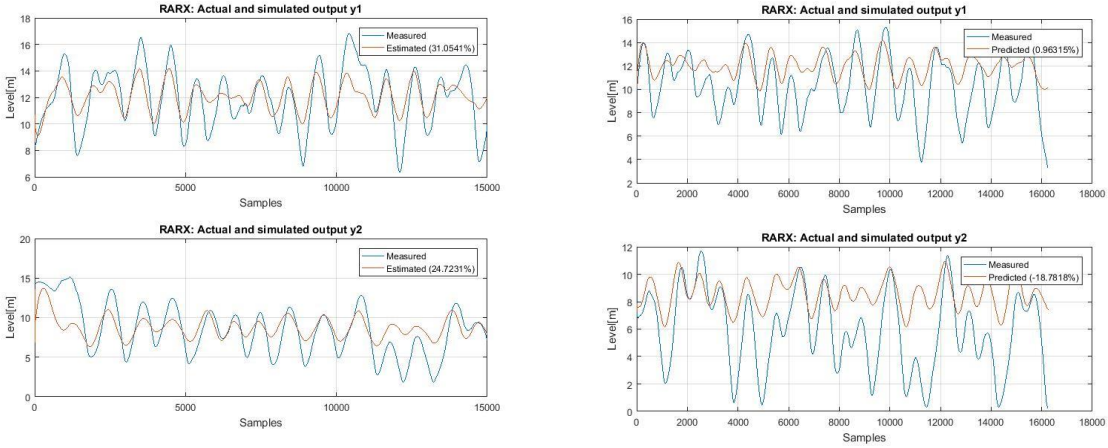

Figure 4.9 Performance of the RARX model: **Left**: Real measurement and Simulated output for output y_1 (upper, fit: 31.05%) and y_2 (lower, fit: 24.72%) and **Right**: Real measurement and predicted output for output y_1 (upper, fit: 0.965) and y_2 (lower, fit: -18.78%).

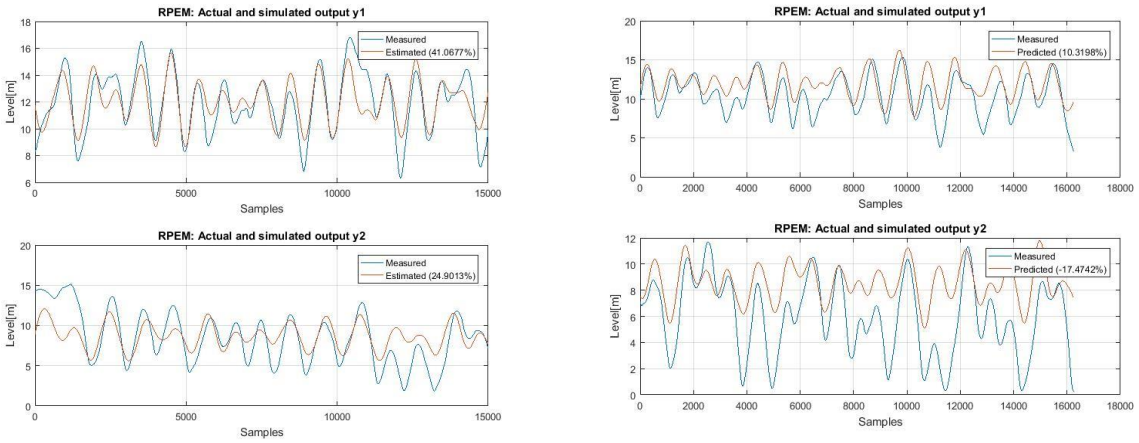

Figure 4.10 Performance of the RPEM model: **Left**: Real measurement and Simulated output for output y_1 (upper, fit: 41.06%) and y_2 (lower, fit: 24.90%) and **Right**: Real measurement and predicted output for output y_1 (upper, fit: 10.32%) and y_2 (lower, fit: -17.47%).

Table 4.2: Validation analysis for the Quadruple Tank process, expressed in MSE and model fit

Type of Output	***Model***	***MSE***	***Fit (%)***	
			y_1	y_2
Simulated	RDSR	5.3492	38.8873	42.4751
	RANN	17.8460	1.4042	24.6584
	RARX	8.3297	31.0541	24.7231
	RPEM	7.6541	41.0677	24.9013
Predicted	RDSR	14.2637	7.8548	-4.8086
	RANN	14.1287	35.1512	33.237
	RARX	17.6388	0.96315	-18.7818
	RPEM	16.2890	10.3198	-17.4742

Generally, all the models performed very well with relatively similar MSE values on the prediction analysis, however, the RANN model was generally more reliable as the other models were producing negative values for the fit on the second model output y_2.

It is possible that some of the sub optimality in the models could have arisen from poor choice of initial model parameters.

The state space model parameters identified for this process by all the algorithms are presented in Appendix A.2

4.1.3 Air heater

The air heater process represents a process as described in section 2.2.3. It's a SISO system. The calculated values of the MSE and fit for this system is presented in Table 4.3. For simulation, the RANN models produce the best models with the least errors and best fits as seen in Figure 4.12. For prediction, the RANN model outperforms the other models with a MSE value of 2.4755 and a fit of 66.56% for the output. The other models performed relatively well and in some similar fashion although the RDSR estimated output produced a negative value of fit.

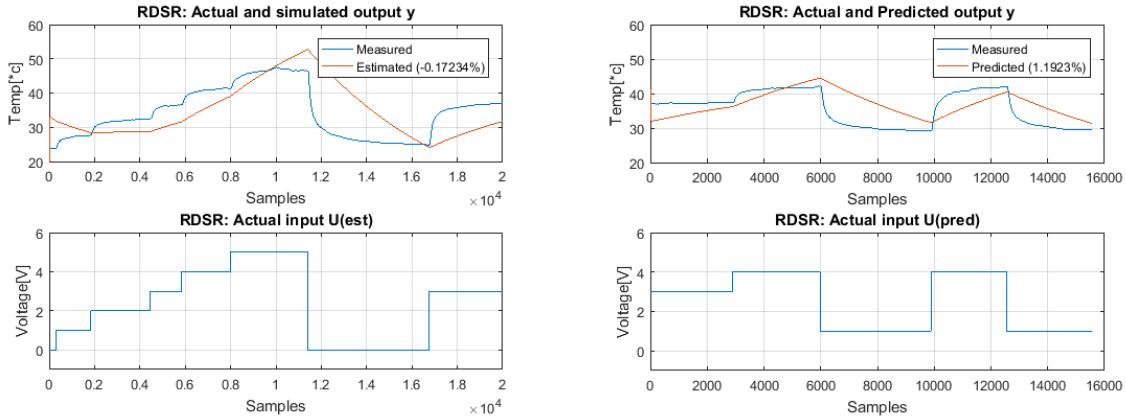

Figure 4.11 Performance of the RDSR model: **Upper Left**: Real measurement and Simulated output, **Lower Left**: Input sequence for the estimation. **Upper Right**: Real measurement and predicted output showing the goodness of fit in %, **Lower Right**: The input sequence for the prediction.

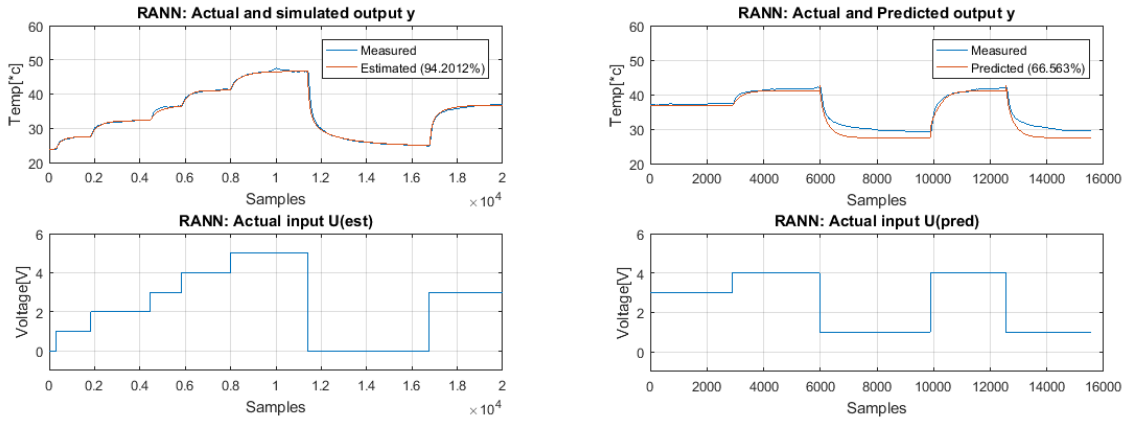

Figure 4.12 Performance of the RANN model: **Upper Left**: Real measurement and Simulated output, **Lower Left**: Input sequence for the estimation. **Upper Right**: Real measurement and predicted output showing the goodness of fit in %, **Lower Right**: The input sequence for the prediction.

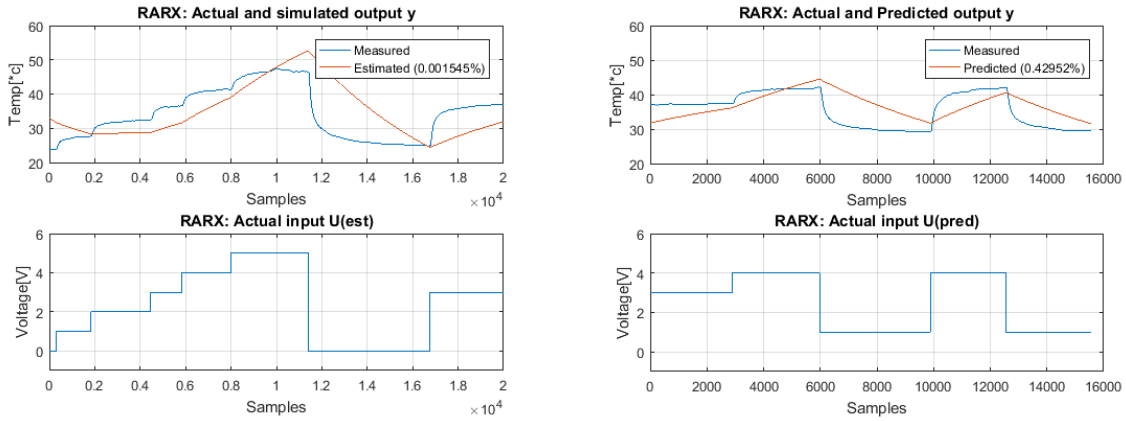

Figure 4.13 Performance of the RARX model: **Upper Left**: Real measurement and Simulated output, **Lower Left**: Input sequence for the estimation. **Upper Right**: Real measurement and predicted output showing the goodness of fit in %, **Lower Right**: The input sequence for the prediction.

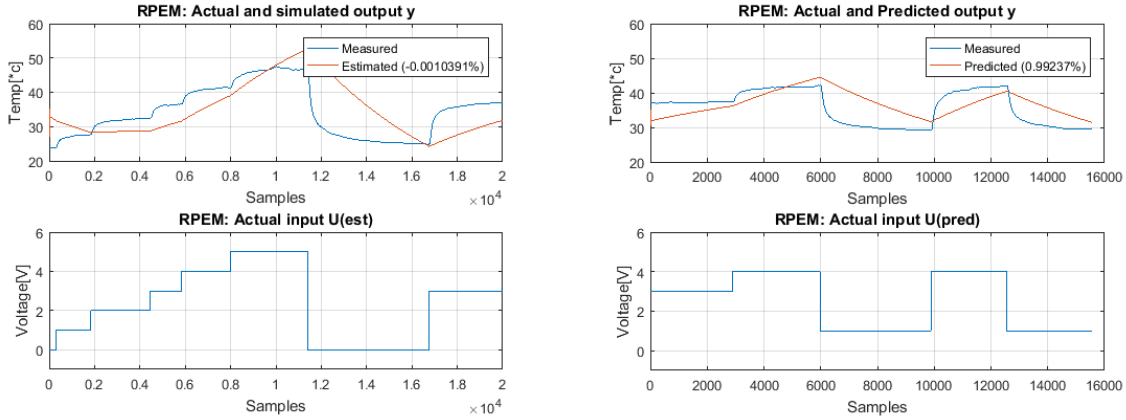

Figure 4.14 Performance of the RPEM model: **Upper Left**: Real measurement and Simulated output, **Lower Left**: Input sequence for the estimation. **Upper Right**: Real measurement and predicted output showing the goodness of fit in %, **Lower Right**: The input sequence for the prediction.

Table 4.3: Validation analysis for the Air Heater Model, expressed in MSE and model fit

Type of Output	*Model*	*MSE*	*Fit (%)*
Simulated	RDSR	53.5145	-0.1723
	RANN	0.1793	94.2012
	RARX	53.3289	0.0015
	RPEM	53.3316	0.0010
Predicted	RDSR	23.4755	1.1923
	RANN	2.6884	66.5630
	RARX	23.8394	0.4295
	RPEM	23.5706	0.9924

Generally, one can see that all the models performed very well with relatively similar MSE values on the prediction analysis, however, the RANN model was generally more reliable as the other models were producing similar values for the fit.

The state space model parameters identified for this process by all the algorithms are presented in Appendix A.4

4.2 Optimal Model Prediction

The predictions in the above section are based only on the deterministic model

$$x_{k+1} = Ax_k + Bu_k, \quad \text{given initial state } x_1 \tag{4.1}$$

$$y_k^d = Dx_k + Eu_k \tag{4.2}$$

Which is identified based on input and output data. y_k^d represents the part in the output which can be described by the deterministic inputs u_k . It therefore makes sense to assume that a better model for the output should contain both the stochastic and deterministic parts of the out.

We can often improve model prediction knowing that the optimal Kalman filter prediction, y_k, is approximately equal to the previous output y_{k-1}. The method determines both the deterministic part and the stochastic part of the model.

The algorithm gives exact results in the deterministic case and consistent results when the system is influenced by noise. For the stochastic part of the system, the system can be presented in the optimal prediction form with given initial conditions x_0. Assume that a combined deterministic and stochastic model with some model matrices A,B,D,E and Kalman filter gain matrix F and a set of input and output validation data are given (Ruscio, 2014). Simulation will then give us the optimal predictions as follows

$$x_{k+1} = Ax_k + Bu_k + K(y_k - Dx_k - Eu_k) \tag{4.3}$$

$$\bar{y}_k = Dx_k + Eu_k \tag{4.4}$$

Where $K = CF^{-1}$, the Kalman filter gain is constructed to minimize the covariance of the estimated error, the parameter in the Kalman filter is found such that the trace of the matrix $E\big((y_k - \bar{y}_k)(y_k - \bar{y}_k)^T\big)$ is minimized and usually also the prediction error criterion is minimized. The selection of a good model is based on some validation criteria of the prediction error criterion

$$V = trace(E\big((y_k - \bar{y}_k)(y_k - \bar{y}_k)^T\big)) \tag{4.5}$$

To simulate the optimal predictor, z_t can be specified to be zero. $K \in \mathbb{R}^{m\times m}$ is the Kalman filter gain matrix which is chosen as a diagonal matrix with parameters $0 \le k_u \le 1$, $\forall\, i = 1, \dots, m$. The following augmented state equation is used.

$$\overbrace{\begin{bmatrix} x_{t+1} \\ z_{t+1} \end{bmatrix}}^{\tilde{x}_{t+1}} = \overbrace{\begin{bmatrix} A & 0 \\ -KD & I_m - K \end{bmatrix}}^{\tilde{A}_t} \overbrace{\begin{bmatrix} x_t \\ z_t \end{bmatrix}}^{\tilde{x}_t} + \overbrace{\begin{bmatrix} B & 0 \\ -KE & K \end{bmatrix}}^{\tilde{B}_t} \overbrace{\begin{bmatrix} u_t \\ y_t \end{bmatrix}}^{\tilde{u}_t} \tag{4.6}$$

$$\hat{y}_t = \overbrace{\begin{bmatrix} D & I_m \end{bmatrix}}^{\tilde{D}_t} \overbrace{\begin{bmatrix} x_t \\ z_t \end{bmatrix}}^{\tilde{x}_t} + \overbrace{\begin{bmatrix} E & 0 \end{bmatrix}}^{\tilde{E}_t} \overbrace{\begin{bmatrix} u_t \\ y_t \end{bmatrix}}^{\tilde{u}_t} \tag{4.7}$$

4.2.1 Quadruple tank

When both the deterministic and stochastic parts of the system are included in the model, we get an optimal result as shown in Table 4.4 there is a 99.9% fit and the model predictions overlap the actual measurement as seen in Figure 4.15. Compared to the other Models, this optimal prediction produces almost accurate predictions

Table 4.4 Optimal Model performance

Model	***MSE***	***Fit (%)***	
		y_1	y_2
RDSR (optimal)	0.0000010945	99.8858	99.9411
RANN	14.1287	35.1512	33.237

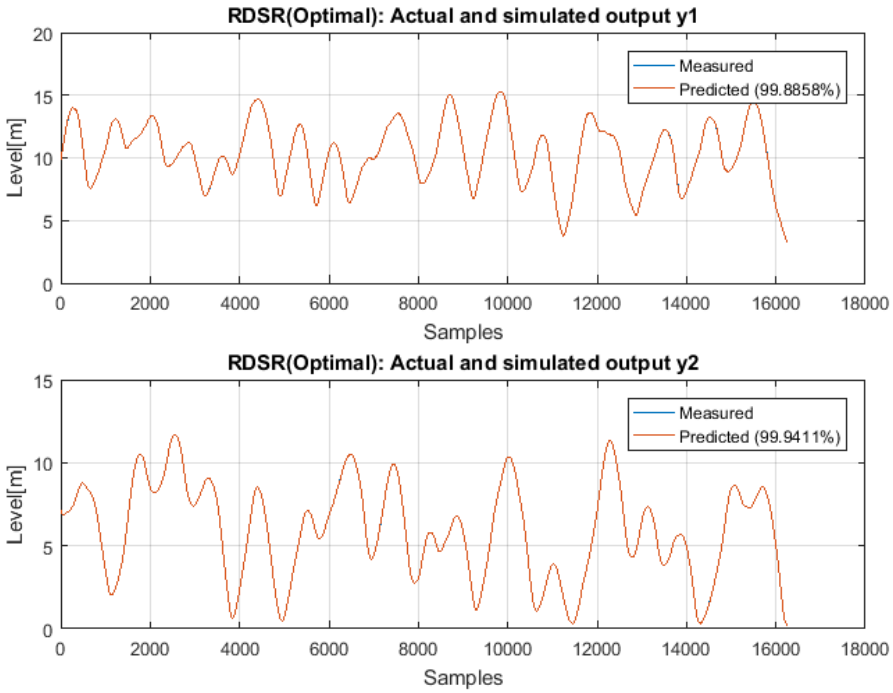

Figure 4.15: Optimal RDSR prediction: This figure illustrates the effect of using a Kalman filter for optimal prediction for the quadruple tank. See Figure 4.7 for comparison.

4.2.2 Synthetic model

For the synthetic model, both the deterministic and stochastic parts of the system are included in the model, we get an optimal result as shown in Table 4.5 there is a recognizable degree of improvement in the fit and the error is minimized with the model now able to represent some part of the noise in the system as seen in Figure 4.16.

Table 4.5: Optimal model performance

Model	***MSE***	***Fit (%)***	
		y_1	y_2
RDSR (optimal)	3.0917	58.6921	83.4726
RANN	5.1533	42.8438	81.4879

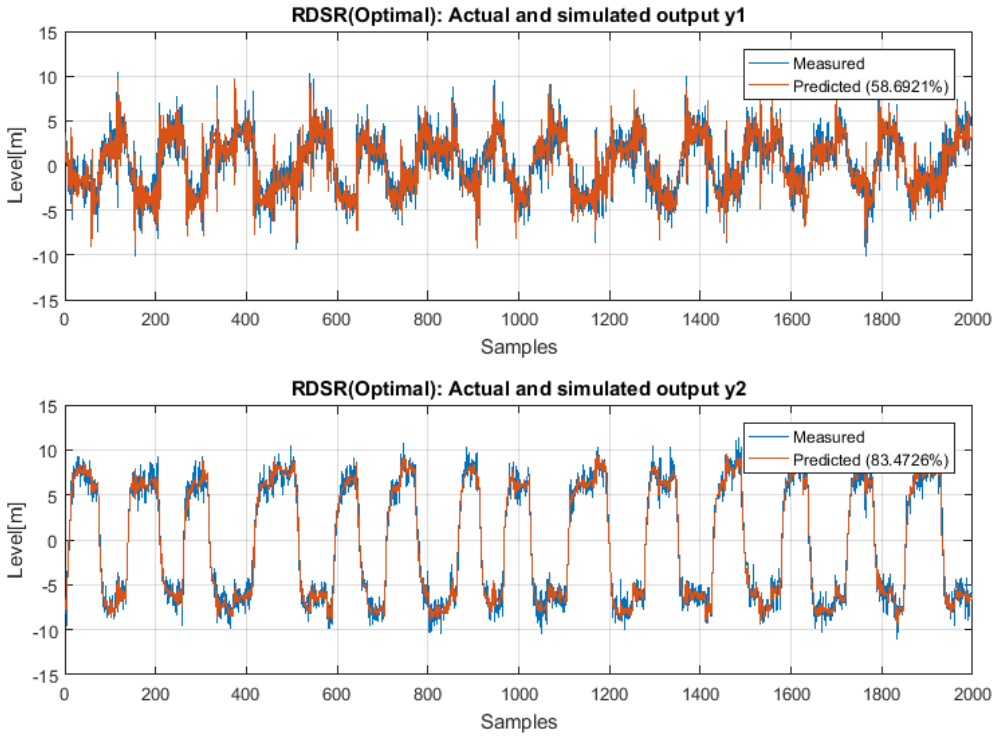

Figure 4.16: Optimal RDSR predictions for the synthetic model.

It can be seen from these results that simulation provides a better validation test for the model than prediction but the prediction ability of any modelling procedure is of more interest especially in control applications. However, the model validation procedure should be selected based on how you plan to use the model.

The RDSR method proves to be a stable and reliable algorithm for modelling both linear and non-linear processes. The speed of implementation is not put into consideration in this comparison since the implementation is dependent on the availability of input and output data due to the sampling time.

5 Concluding Remarks

The main ideas are highlighted in this chapter. The conclusion for this thesis work is presented and recommendations for further work is given. The main objective of this work is to study the use of recursive subspace system identification techniques to estimate system matrices and then the result of the simulations is compared with that of the traditional methods as well as the neural network approach. The MATLAB software has been used throughout this thesis work.

5.1 Conclusion

A comprehensive literature review of the recursive subspace identification algorithm was done and an overview was presented in the beginning sections of this report as well as a comparison with the traditional recursive system identification methods available in the MATLAB system identification and Neural network toolboxes. In comparing the RDSR method with other methods it was found that

1) Classical RARX and RPEM approaches need a certain user-specified model parameterization in canonical forms which could lead to numerically ill-conditioned problems because the parameters in the canonical form model is extremely sensitive to small perturbations meanwhile the RDSR method need no parameterization.
2) The RARX and RPEM approach is iterative which usually leads to many hard to deal with problems such as problems with lack of convergence; no convergence; slow convergence; local minima; numerical instability etc. meanwhile the RDSR method is non-iterative therefore there are no problems with convergence and the method is numerically robust
3) RDSR method is generally faster than classical RPEM approaches because the method is non-iterative.

Next, the RDSR algorithm was investigate in some details and a short and detailed description of the algorithm was presented in section 3.3.1 of this report. Various simulation experiments and Monte Carlo simulations performed to investigate the quality of the parameter estimates from all the chosen algorithms which are the RPEM, RARX, RANN and RDSR methods. Various tests on real data from two of our laboratory processes – the Air heater and the quadruple tank process were the candidate laboratory process, also some synthetic data generated from systems with known matrices showed that the RDSR algorithm is as reliable as the classical methods and even better in some instances.

Based on the results in chapter 4 as well as the information in other chapters, one can draw the following conclusions with focus on the predictability of the identified models since that is of more interest in testing the quality of identified models:

- For the Synthetic model:
  The RPEM method performed best (MSE: 3.9417) followed by the RDSR method (MSE: 4.0167) then the RARX method (MSE: 4.0606) and finally the RANN method (MSE: 5.1533).

- For the quadruple tank:
  The RANN method performed best (MSE: 14.1287) followed by the RDSR method (MSE: 14.2637) then the RPEM method (MSE: 16.2890) and finally the RARX method (MSE: 17.6388).

- For the Air heater process:
  The RANN method performed best (MSE: 2.6884) followed by the RDSR method (MSE: 23.4755) then the RPEM method (MSE: 23.5706) and finally the RARX method (MSE: 23.8394).

The optimal RDSR predictions were also implemented and the results compared with that of the RANN model output. The optimal predictions were achieved using the Kalman filtering technique and the results were almost as accurate as the raw measurement data with a 99.9% fit for the quadruple tank and a 58.7% and 83.5% fit for the synthetic data of which this deficiency was due to too much noisy system. Overall, the predictions were truly optimal.

The RDSR method has proven to be a very reliable and robust algorithm which performed very well in both MIMO and SISO systems even when the system is very noisy and somewhat unstable, it was still able to produce very good model matrices which could compete well with the traditional methods with high industrial standard.

5.2 Further works

The main tasks in this project were fulfilled with reasonably presentable results but it should be noted that it was a little difficult for the optimal prediction to be implemented for the SISO air heater process as the results of the implementation was always giving optimal prediction values up to infinity. Due to time constraints, this issue could not be thoroughly investigated to find out why the values were so high. Also, the optimal predictions were not implemented for the other RARX and RPEM methods for comparison. Thus, the following are the suggestions for further work in this line of study:

- Optimal RPEM and RARX implementation: It will be interesting to be able to implement optimal predictions for the RPEM and RARX algorithms since these also are very reliable system identification approaches.
- Neuro fuzzy system identification: An interesting field of study could be the use of Neuro fuzzy models to identify systems
- Recursive closed loop identification: Most of the simulations performed here are open loop identifications. I will suggest that entire control loops are identified as well.
- Online Model Validation and stability analysis: the model validation steps were performed offline after the models have been identified online, so I think it would be interesting to see models being identified and validated in the same time instant.

References

AARTS, R. G. K. M. 2012. *System Identification and Parameter Estimation,* Enschede, University of Twente.

CHINARRO, D. 2014. System Engineering Applied to Fuenmayor Karst Aquifer. *Springer Theses***,** 11-53.

FINN HAUGEN, E. F., RICARDO DUNIA, THOMAS F. EDGAR 2007. Demonstrating PID Control Principles using an Air Heater and LabVIEW. *CACHE News (Computer Aids for Chemical Engineering).*

G. MERCERE, M. L. 2006. CONVERGENCE ANALYSIS OF INSTRUMENTAL VARIABLE RECURSIVE SUBSPACE IDENTIFICATION ALGORITHMS. *14th IFAC Symposium on Identification and System Parameter Estimation.,* 39**,** 279-284.

G. MERCERE, S. L., AND M. LOVERA. 2004. Recursive subspace identification based on instrumental

variable unconstrained quadratic optimization. *Adaptive Control and Signal Processing,,* 18**,** 771-797.

GUILLAUME MERCERE, S. L., CHRISTIAN VASSEUR. 2005. SEQUENTIAL CORRELATION BASED PROPAGATOR ALGORITHM FOR RECURSIVE SUBSPACE IDENTIFICATION. *16th IFAC World Congress,* 38**,** 922-927.

HOWARD DEMUTH, M. B. 2002. *Neural Network Toolbox User's Guide,* Natick, MA, The MathWorks, Inc.

I. GOETHALS, K. P., J.A.K. SUYKENS, AND B. DE MOOR. 2005. Identification of MIMO Hammerstein models using least squares support vector machines. *Automatica,* 41**,** 1263-1272,.

I. GOETHALS, L. M., A. BENVENISTE, AND B. DE MOOR. 2004. Recursive output-only subspace identification for in-flight flutter monitoring. *Proceedings of the 22nd International Modal Analysis Conference (IMAC-XXII).*

I. HOUTZAGER, J. W. V. W., M. VERHAEGEN. 2009. Fast-array Recursive Closed-loop Subspace Model Identification. *15th IFAC Symposium on System Identification.,* 42**,** 96-101.

JIE HOU, F. C., TAO LIU. 2015. Recursive Closed-loop PARSIM-E Subspace Identification.

K. J. HUNT, D. S., R. ZBIKOWSKI, P. J. GAWTHROP 1992. Neural Networks for Control Systems - A Survey. *International Federation of Automatic Control.,* 28**,** 1083-1112.

KENTARO KAMEYAMA, A. O. 2005. RECURSIVE SUBSPACE PREDICTION OF LINEAR TIME-VARYING STOCHASTIC SYSTEMS. *16th IFAC world Congress.,* 38**,** 916-921.

KUMPATI S. NARENDRA, K. P. 1990. Identification and control of Dynamic Systems Using Neural Networks. *IEEE Transactions of Neural Networks.,* 1.

L. BAKO, G. M., S. LECOEUCHE, AND M. LOVERA. 2009. Recursive subspace identification of Hammerstein models based on least squares support vector machines. *IET Control Theory & Applications,* **,** 1209-1216.

L. LJUNG, T. G. 1994. *Modelling of dynamic systems,* Englewood Cliffs, Prentice Hall.

LARIMORE, W. E. 1990. Canonical variate analysis in identification filtering and adaptive control. *Proceedings of the 28th conference on decision and controls***,** 596-604.

LIANG MA, X. L. 2016. Recursive maximum likelihood method for the identification of Hammerstein ARMAX system. 40**,** 6523-6535.

LJUNG, L. 1999. *System Identification - Theory for the User.,* Upper Saddle River, N.J., Prentice Hall International.

LJUNG, L. 2000. *System Identification Toolbox User's Guide,* Natick, The MathWorks Inc,.

M. LOVERA, T. G., AND M. VERHAEGEN. 1998. Recursive subspace identification of linear and non-linear wiener state space models. *Automatica.,* 36**,** 1639-1650.

MATHWORKS. 2016. *System Identification Overview* [Online]. Available: https://se.mathworks.com/help/ident/gs/about-system-identification.html [Accessed].

NI.COM. 2010. *Selecting a model structure in the system identification process* [Online]. National Instruments. Available: http://www.ni.com/white-paper/4028/en/#toc4 [Accessed].

P.V. OVERSCHEE, B. D. M. 1994. N4SID: Two subspace algorithms for the identification of combined deterministic-stochastic system. *Automatica.***,** 75-93.

RUSCIO, D. D. 1998. *Advanced Process and quality control.,* Porsgrun, Norway, Telemark institute of Technology.

RUSCIO, D. D. 2012. System Identification and Optimal Estimation. *Exercise 8.* Telemark University College.

RUSCIO, D. D. 2014. *Subspace System Identification,* Porsgrunn, Norway, Telemark Institute of Technology.

VERHAEGEN M, D. P. 1992. Subspace model identification part 1. the output-error state-space model identification class of algorithms. 1187-1210.

XI CHEN, H.-T. F. 2012. Recursive Subspace Method for Wiener Systems Using Instrumental Variable Techniques. *16th IFAC Symposium on System Identification.,* 45**,** 1508-1513.

YI LIU, H. W., JIANG YU, PING LI. 2010. Selective recursive kernel learning for online identification of nonlinear systems with NARX form. *Journal of Process Control,* 20**,** 181-194.

YUEPING JIANG, H. F. 2009. Recursive Subspace Identification Algorithm for Closed-loop Stochastic Systems. *15th IFAC Symposium on System Identification.,* 42**,** 116-121.

Appendices

A.1 Master's thesis proposal

HSN University College of Southeast Norway

Faculty of Technology, Natural Sciences and Maritime Sciences, Campus Porsgrunn

FMH606 Master's Thesis

Title: Recursive Subspace System IDentification (RSSID) algorithms

HSN supervisor: David Di Ruscio

External partner: None

Task background:
Subspace System IDentification (SSID) is a relative matured field and there exists efficient algorithms as DSR, N4SID and MOESP etc. However,
algorithms for recursive implementations are not published and studied with the same details. However, a Recursive DSR (the RDSR method) where developed in the end of the 90'ies but only published in an internal report. It would be of grate interest to study the RDSR algorithm in detail, and possible other existing algorithms within this field.

Task description:

1. Perform a literature review of Recursive Subspace based System IDentification (RSSID) algorithms. Give an overview and work out a comparison with traditional Recursive System IDentification (RSID) methods as the Recursive Prediction Error Method (RPEM) and if there are something to gain by using RSSID methods, identification of system order etc. Possibly also compare RSSID methods with (recursive) Neural Network methods.
2. Investigate the RDSR algorithm in some details and give a short and detailed description of the algorithm.
3. Perform simulation experiments and Monte Carlo simulations to investigate the quality of the parameter estimates.
4. Testing on real data from one of our laboratory processes would be of interests. The quadruple tank process would be a candidate laboratory process.

Student category: IIA students

Practical arrangements:

Works at school

Signatures:

Student 152360@student.hit.no (date and signature): Harrison Idoniqué

31.01.2017 Harrison Idoniqué

Supervisor David Di Ruscio, (date and signature):

31/1-17 David Di Ruscio

Address: Kjølnes ring 56, NO-3918 Porsgrunn, Norway. **Phone:** 35 57 50 00. **Fax:** 35 55 75 47.

.

A.2 Parameters for Quadruple tank models

RDSR

```
m1_dsr =
  Discrete-time identified state-space model:
    x(t+Ts) = A x(t) + B u(t) + K e(t)
       y(t) = C x(t) + D u(t) + e(t)

  A =
              x1           x2          x3          x4
   x1      0.9992  -0.0003804       1.985     -0.2168
   x2   0.0009205       0.9992     -0.2174      -1.984
   x3  -1.087e-05   1.125e-05      0.9982    0.000668
   x4  -1.363e-06   8.375e-06   0.0002999      0.9968

  B =
              u1           u2
   x1   0.0003731   -0.002863
   x2  -0.0004917   -0.002078
   x3  -7.507e-05  -8.457e-06
   x4   2.453e-05  -6.204e-05

  C =
           x1       x2       x3       x4
   y1  -0.3613   0.6083    0.425   0.5647
   y2  -0.6082  -0.3611   0.5648  -0.4252

  D =
              u1           u2
   y1     0.00173  -0.0003209
   y2   0.0001133    0.005374

  K =
           y1       y2
   x1   -1.514   -2.651
   x2    2.522   -1.605
   x3  -0.2505  -0.3818
   x4  -0.3262   0.2881
```

RARX (MISO)

```
model1 =
  Discrete-time identified state-space model:
    x(t+Ts) = A x(t) + B u(t) + K e(t)
       y(t) = C x(t) + D u(t) + e(t)

A =
           x1        x2        x3        x4        x5
 x1         0         0         0         0   -0.4515
 x2       0.5         0         0         0   0.05206
 x3         0         1         0         0  -0.01738
 x4         0         0         1         0   -0.2048
 x5         0         0         0         1     1.396

B =
             u1          u2
 x1  -0.0008388   6.892e-05
 x2  -0.0001107    0.001041
 x3    0.001181  -0.0009092
 x4  -0.0001976   0.0001493
 x5  -0.0003302  -0.0001523

C =
     x1  x2  x3  x4  x5
 y1   0   0   0   0   1

D =
             u1          u2
 y1    0.001858  -0.0001527

K =
          y1
 x1   -0.4515
 x2   0.05206
 x3  -0.01738
 x4   -0.2048
 x5     1.396
```

```
model2 =
  Discrete-time identified state-space model:
    x(t+Ts) = A x(t) + B u(t) + K e(t)
       y(t) = C x(t) + D u(t) + e(t)

A =
           x1        x2        x3        x4        x5
 x1         0         0         0         0   -0.2685
 x2       0.5         0         0         0   0.05259
 x3         0         1         0         0   0.01797
 x4         0         0         1         0   -0.6513
 x5         0         0         0         1     1.715

B =
             u1          u2
 x1   1.304e-05   -0.001418
 x2   7.038e-05    0.001282
 x3   2.109e-05  -1.325e-05
 x4   0.0003315    -0.00297
 x5  -0.0003221    0.002378

C =
     x1  x2  x3  x4  x5
 y1   0   0   0   0   1

D =
             u1          u2
 y1  -4.854e-05    0.005282

K =
          y1
 x1   -0.2685
 x2   0.05259
 x3   0.01797
 x4   -0.6513
 x5     1.715
```

RPEM (MISO)

```
model1 =
  Discrete-time identified state-space model:
    x(t+Ts) = A x(t) + B u(t) + K e(t)
       y(t) = C x(t) + D u(t) + e(t)

A =
           x1        x2
 x1         0   -0.9975
 x2         1     1.997

B =
             u1          u2
 x1  -0.0003578   0.0007419
 x2   0.0003799  -0.0006615

C =
     x1  x2
 y1   0   1

D =
     u1  u2
 y1   0   0

K =
          y1
 x1   -1.107
 x2    1.284
```

```
model2 =
  Discrete-time identified state-space model:
    x(t+Ts) = A x(t) + B u(t) + K e(t)
       y(t) = C x(t) + D u(t) + e(t)

A =
          x1       x2
 x1        0   -0.998
 x2        1    1.998

B =
             u1          u2
 x1   0.0005534   0.0009239
 x2  -0.0004907  -0.0009422

C =
     x1  x2
 y1   0   1

D =
     u1  u2
 y1   0   0

K =
          y1
 x1   -1.266
 x2    1.621
```

A.3 Parameters for Synthetic models

RDSR

```
dddsr =
  Discrete-time identified state-space model:
    x(t+Ts) = A x(t) + B u(t) + K e(t)
       y(t) = C x(t) + D u(t) + e(t)

  A =
          x1       x2       x3       x4       x5       x6       x7       x8
   x1      0        0        0   0.2933        0        0        0        0
   x2    0.5        0        0   0.7572        0        0        0        0
   x3      0        1        0   0.4258        0        0        0        0
   x4      0        0        1  -0.9234        0        0        0        0
   x5      0        0        0        0        0        0        0   0.2933
   x6      0        0        0        0      0.5        0        0   0.7572
   x7      0        0        0        0        0        1        0   0.4258
   x8      0        0        0        0        0        0        1  -0.9234

  B =
          u1       u2
   x1 -0.1523  -0.5222
   x2 -0.3318  -0.8233
   x3 -0.3111    1.445
   x4 -0.0439  -0.1444
   x5  0.1819  -0.3244
   x6  0.5496  -0.5744
   x7  0.9162   0.4712
   x8  0.5438   0.5045

  C =
       x1  x2  x3  x4  x5  x6  x7  x8
   y1   0   0   0   2   0   0   0   0
   y2   0   0   0   0   0   0   0   2

D =
            u1          u2
 y1    -0.2152      0.2355
 y2  -0.004603  -0.009439

K =
      y1  y2
 x1    0   0
 x2    0   0
 x3    0   0
 x4    0   0
 x5    0   0
 x6    0   0
 x7    0   0
 x8    0   0
```

RARX (MISO)

1st output:

```
sysw1_arx =
  Discrete-time identified state-space model:
    x(t+Ts) = A x(t) + B u(t) + K e(t)
       y(t) = C x(t) + D u(t) + e(t)

A =
          x1        x2        x3        x4        x5
 x1        0         0         0         0     0.407
 x2      0.5         0         0         0  0.008422
 x3        0         1         0         0    0.1286
 x4        0         0         1         0    0.3764
 x5        0         0         0         1   -0.3908

B =
          u1        u2
 x1   0.01344    0.0341
 x2   -0.3246    0.5021
 x3   -0.1976    -1.641
 x4   -0.1257     1.469
 x5   -0.2842  -0.05298

C =
     x1  x2  x3  x4  x5
 y1   0   0   0   0   2

D =
          u1       u2
 y1  0.06604   0.1676

K =
          y1
 x1    0.2035
 x2  0.004211
 x3   0.06429
 x4    0.1882
 x5   -0.1954
```

2nd output:

```
sysq2_arx =
  Discrete-time identified state-space model:
    x(t+Ts) = A x(t) + B u(t) + K e(t)
       y(t) = C x(t) + D u(t) + e(t)

A =
          x1        x2        x3        x4        x5
 x1        0         0         0         0    0.4785
 x2      0.5         0         0         0    0.2037
 x3        0         1         0         0    0.2044
 x4        0         0         1         0    0.4675
 x5        0         0         0       0.5  0.01432

B =
          u1        u2
 x1   0.02254  0.001117
 x2    0.6485    0.2438
 x3    0.5356   -0.9153
 x4    0.9167  0.005804
 x5    0.4198    0.5058

C =
     x1  x2  x3  x4  x5
 y1   0   0   0   0   2

D =
          u1        u2
 y1   0.09419  0.004671

K =
          y1
 x1   0.2393
 x2   0.1018
 x3   0.1022
 x4   0.2338
 x5  0.00716
```

RPEM (MISO)

1st output:

```
m1q_pem =
  Discrete-time identified state-space model:
    x(t+Ts) = A x(t) + B u(t) + K e(t)
       y(t) = C x(t) + D u(t) + e(t)

  A =
          x1        x2        x3
 x1        0         0    0.6008
 x2        1         0    0.5742
 x3        0         1   -0.6404

  B =
          u1        u2
 x1  -0.09922    -1.274
 x2   -0.3103     1.427
 x3   -0.2404   0.06957

  C =
     x1  x2  x3
 y1   0   0   2

  D =
     u1  u2
 y1   0   0

  K =
          y1
 x1   0.3004
 x2  0.06192
 x3  -0.2169
```

2nd output:

```
mqq2_pem =
  Discrete-time identified state-space model:
    x(t+Ts) = A x(t) + B u(t) + K e(t)
       y(t) = C x(t) + D u(t) + e(t)

A =
          x1        x2        x3
 x1        0         0    0.4612
 x2        1         0    0.5757
 x3        0         1   -0.4673

B =
          u1        u2
 x1   0.3527   -0.6817
 x2   0.6875    0.2919
 x3   0.4797    0.5563

C =
     x1  x2  x3
 y1   0   0   2

D =
     u1  u2
 y1   0   0

K =
          y1
 x1    0.2306
 x2   0.06737
 x3  -0.0001269
```

A.4 Parameters for Air heater models

RDSR

```
mq1_dsr =
  Discrete-time identified state-space model:
    x(t+Ts) = A x(t) + B u(t) + K e(t)
       y(t) = C x(t) + D u(t) + e(t)

  A =
           x1        x2
   x1    1.869   -0.8687
   x2        1         0

  B =
           u1
   x1  0.03125
   x2        0

  C =
            x1         x2
   y1   0.02392   -0.01484

  D =
            u1
   y1  0.000385

  K =
       y1
   x1   0
   x2   0
```

RARX

```
mer1_arx =
  Discrete-time identified state-space model:
    x(t+Ts) = A x(t) + B u(t) + K e(t)
       y(t) = C x(t) + D u(t) + e(t)

  A =
          x1
   x1  0.9999

  B =
            u1
   x1  0.002131

  C =
       x1
   y1   1

  D =
       u1
   y1   0

  K =
          y1
   x1  0.9999
```

RPEM

```
m1_pemq =
  Discrete-time identified state-space model:
    x(t+Ts) = A x(t) + B u(t) + K e(t)
       y(t) = C x(t) + D u(t) + e(t)

  A =
           x1       x2
   x1       0   0.6554
   x2     0.5   0.6721

  B =
             u1
   x1   0.00315
   x2  0.001276

  C =
       x1  x2
   y1   0   1

  D =
       u1
   y1   0

  K =
          y1
   x1  0.6665
   x2   1.003
```

A.5 MATLAB codes

Code to simulate, plot and compare all algorithms

RDSR implementation:

```
%% load the data for both identification and prediction
clc, clear;

load('Yid.mat')
load('Uid.mat')

load('U_val.mat')
load('Y_val.mat')

%% %%%%%%%%%%%%% IDENTIFY MODEL WITH DSR TOOLBOX %%%%%
disp('Identify model with RDSR....(wait)')
% initialize dsr parameters
L = 2; J = L; g = 1; ff = 1; n = 4;

y=Yid; u=Uid;

% LOOP RECURSIVELY OVER ALL COLUMNS, INCLUDE ONE COLUMN AT A TIME.
[Ny,ny]=size(y);                    % Number of y variables ny.
[Nu,nu]=size(u);                    % Number of u variables nu.
N=min(Ny,Nu);                       % Number of samples in batch series.
nry=L+1+J;                          % Samples in output data window
Y^w_t=yt.
nru=L+g+J;                          % Samples in input data window U^w_t=ut.
nr=nry*ny+nru*nu;                   % The row and column size for R.
Lam=eye(nr)*ff;                     % Forgetting factor matrix.
R=zeros(nr);                        % Start with a square zero R matrix.

at=zeros(N,n*n);                    % Arrays to hold time varying matrices.
bt=zeros(N,n*nu);
dt=zeros(N,ny*n);
et=zeros(N,ny*nu);
ct=zeros(N,n*ny);
ft=zeros(N,ny*ny);
x0t=zeros(N,n);                     % Array to hold "initial" states.
Rt=zeros(N,nr*nr);                  % Array to hold the R_t matrix of size
(nr x nr).
sv=zeros(N,L*ny);                   % Array to hold singular values.

K=N-(J+L);                                              % # of columns to
update.
for i=1:K
   yt=y(i:i+nry-1,:);
   ut=u(i:i+nru-1,:);
   R=dsr_upr(Lam*R,yt,ut,L,g,J);                        % update the R
matrix
   if i>=nr

[a_dsr,b_dsr,d_dsr,e_dsr,cf_dsr,f_dsr,x0_dsr]=dsr_r2m(R,yt,ut,L,n,g,J,i+L+J
);    % compute model from R
   end
end
```

```
k_dsr=cf_dsr*inv(f_dsr); % Kalman gain matrix.

% make a state space
m1_dsr=idss(a_dsr,b_dsr,d_dsr,e_dsr,k_dsr,x0_dsr); %,Uid,x0_dsr); %
simulated output

% Validation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

z1=[Yid Uid]; % combine estimation data

% plot figures
ym_dsr=predict(m1_dsr,z1,inf);
fit = goodnessOfFit(ym_dsr,Yid,'NRMSE')
fitY = fit*100;
fitE = goodnessOfFit(ym_dsr,Yid,'MSE')

figure;
subplot(211), plot([Yid(:,1) ym_dsr(:,1)]);
title('RDSR: Actual and simulated output y1')
legend('Measured',strcat('Estimated (', num2str(fitY(1)),'%)'));
xlabel('Samples'); ylabel('Level[m]');grid on

subplot(212), plot([Yid(:,2) ym_dsr(:,2)]);
title('RDSR: Actual and simulated output y2')
legend('Measured',strcat('Estimated (', num2str(fitY(2)),'%)'));
xlabel('Samples'); ylabel('Level[m]');grid on

% subplot(221), plot(Uid(:,1)), title('RDSR: Actual input U1')
% xlabel('Samples'); ylabel('Voltage[V]'); grid on
%
% subplot(223), plot(Uid(:,2)), title('RDSR: Actual input U2')
% xlabel('Samples'); ylabel('Voltage[V]');grid on

% Prediction %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

z2 = [Y_val U_val]; % combine new dataset
y1m_dsr=predict(m1_dsr,z2,inf); % simulate the model

fit1 = goodnessOfFit(y1m_dsr,Y_val,'NRMSE') %calculate fit%
fiCt1 = fit1*100;
fitP1 = goodnessOfFit(y1m_dsr,Y_val,'MSE') %calculate MSE

% make plots
figure;
subplot(211), plot([Y_val(:,1) y1m_dsr(:,1)]);
title('RDSR: Actual and simulated output y1')
legend('Measured',strcat('Predicted (', num2str(fiCt1(1)),'%)'));
xlabel('Samples'); ylabel('Level[m]');grid on

subplot(212), plot([Y_val(:,2) y1m_dsr(:,2)]);
title('RDSR: Actual and simulated output y2')
legend('Measured',strcat('Predicted (', num2str(fiCt1(2)),'%)'));
xlabel('Samples'); ylabel('Level[m]');grid on

% subplot(211), plot(U_val(:,1)), title('RDSR: Actual input U1')
% xlabel('Samples'); ylabel('Voltage[V]'); grid on
%
% subplot(212), plot(U_val(:,2)), title('RDSR: Actual input U2')
% xlabel('Samples'); ylabel('Voltage[V]');grid on
```

Optimal RDSR:

```
% Optimal %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[Yp_dsr,Vpe_dsr,Xp]=opt_pred(a_dsr,b_dsr,d_dsr,e_dsr,k_dsr,Y_val,U_val,x0_d
sr);
Vpe_dsr

fiIt1 = goodnessOfFit(Yp_dsr,Y_val,'NRMSE');
fiIt111 = fiIt1*100

figure;
subplot(211), plot(Y_val(:,1)); hold on;
plot(Yp_dsr(:,1)); hold off
title('RDSR(Optimal): Actual and simulated output y1')
legend('Measured',strcat('Predicted (', num2str(fiIt111(1)),'%)'));
xlabel('Samples'); ylabel('Level[m]');grid on

subplot(212), plot(Y_val(:,2)); hold on;
plot(Yp_dsr(:,2)); hold off

title('RDSR(Optimal): Actual and simulated output y2')
legend('Measured',strcat('Predicted (', num2str(fiIt111(2)),'%)'));
xlabel('Samples'); ylabel('Level[m]');grid on
```

Kalman filter:

```
 function [yp,Vpe,xp] = opt_pred(a,b,d,e,K,Y,U,x0);

[m,n]=size(d); [N,r]=size(U);
% K=cf*inv(f);

%%%% Simulate the optimal predictor %%%%%%%%
%   x_{t+1} = (A - K D) x_t + (B - K E) u_t + K y_t
%   yp_t    = D x_t + E u_t

a1 = a-K*d;
b1 = [b-K*e K];
d1 = d;
e1 = [e,zeros(m,m)];
u1 = [U,Y];
xm=x0;

%Initialize output arrays
y=zeros(N,m); xp=zeros(N,n);

%Simulation loop for evaluation of the states
    for i=1:N
        % store xm in array x for the states
        xp(i,:)=xm';
        % update (integrate) the state vector
        xm = a1*xm + b1*u1(i,:)';
    end

%Compute the outputs
yp=xp*d1'+u1*e1';
```

```
%% 4. %%% COMPUTE PREDICTION ERROR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Vpe= (Y-yp)'*(Y-yp); Vpe=Vpe/(max(size(yp))-1); Vpe=trace(Vpe);
end
% END OPT_pred
```

RANN implementation:

```
%% %%%%%%%%%%%%%%%%%%%%%% IDENTIFY MODEL WITH ANN %%%%%%%
disp('Identify model with ANN....(wait)')
%
P=Uid'; T=Yid';
%lr = 0.1;
X=P;

N=length(Uid);

Ai = {[0;0;0;0;0;0;0;0;0;0] [0;0;0;0;0;0;0;0;0;0];-0.628604071828737 -
0.627272617156328};
Ai2 = {[0;0;0;0;0;0;0;0;0;0] [0;0;0;0;0;0;0;0;0;0];0.882471707976993
0.883036639810788};

Xi = {[3.50000000000000;3.80000000000000]
[3.50000000000000;3.80000000000000]};
Xi2 = {[3.50000000000000;3.80000000000000]
[3.50000000000000;3.80000000000000]};

Ym = zeros(2,N);

% Validation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% LOOP RECURSIVELY OVER ALL COLUMNS, INCLUDE ONE COLUMN AT A TIME.
for i = 1:N
    % first output
    [Y_pred1,Xf,Af] = myNeuralNetworkFunctionU1(X(1,i),Xi,Ai);
    Xi = Xf;
    Ai = Af;
    Ym(1,i) = Y_pred1;

    %second output
    [Y_pred2,Xf2,Af2] = myNeuralNetworkFunctionU2(X(2,i),Xi2,Ai2);
    Xi2 = Xf2;
    Ai2 = Af2;
    Ym(2,i) = Y_pred2;

end

ym_ann = Ym';
% fit = goodnessOfFit(y,yref,cost_func);

% z1=[Yid Uid];
% ym_dsr=predict(m1_dsr,z1,inf);
fitt = goodnessOfFit(ym_ann,Yid,'NRMSE')
fitt = fitt*100;
fitE = goodnessOfFit(ym_ann,Yid,'MSE')
```

```
figure;
subplot(211), plot([Yid(:,1) ym_ann(:,1)]);
title('RANN: Actual and simulated output y1')
legend('Measured',strcat('Estimated (', num2str(fitt(1)),'%)'));
xlabel('Samples'); ylabel('Level[m]');grid on

subplot(212), plot([Yid(:,2) ym_ann(:,2)]);
title('RANN: Actual and simulated output y2')
legend('Measured',strcat('Estimated (', num2str(fitt(2)),'%)'));
xlabel('Samples'); ylabel('Level[m]');grid on

% subplot(221), plot(Uid(:,1)), title('RANN: Actual input U1')
% xlabel('Samples'); ylabel('Voltage[V]'); grid on
%
% subplot(223), plot(Uid(:,2)), title('RANN: Actual input U2')
% xlabel('Samples'); ylabel('Voltage[V]');grid on

% Prediction %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
P=U_val'; T=Y_val';
%lr = 0.1;
X=P;

N=length(U_val);

for i = 1:N

    [Y_pred1,Xf,Af] = myNeuralNetworkFunctionU1(X(:,i),Xi,Ai);
    Xi = Xf;
    Ai = Af;
    Ym(:,i) = Y_pred1;

    [Y_pred2,Xf2,Af2] = myNeuralNetworkFunctionU2(X(:,i),Xi2,Ai2);
    Xi2 = Xf2;
    Ai2 = Af2;
    Ym(2,i) = Y_pred2;
end
y1m_ann = Ym';

% z2 = [Y_val U_val];
% y1m_dsr=predict(m1_dsr,z2,inf);
fitt1 = goodnessOfFit(y1m_ann,Y_val,'NRMSE')
fitt1 = fitt1*100;
fitP1 = goodnessOfFit(y1m_ann,Y_val,'MSE')

figure;
subplot(211), plot([Y_val(:,1) y1m_ann(:,1)]);
title('RANN: Actual and simulated output y1')
legend('Measured',strcat('Predicted (', num2str(fitt1(1)),'%)'));
xlabel('Samples'); ylabel('Level[m]');grid on

subplot(212), plot([Y_val(:,2) y1m_ann(:,2)]);
title('RANN: Actual and simulated output y2')
legend('Measured',strcat('Predicted (', num2str(fitt1(2)),'%)'));
xlabel('Samples'); ylabel('Level[m]');grid on

% subplot(221), plot(U_val(:,1)), title('RANN: Actual input U1')
% xlabel('Samples'); ylabel('Voltage[V]'); grid on
%
% subplot(223), plot(U_val(:,2)), title('RANN: Actual input U2')
```

```
% xlabel('Samples'); ylabel('Voltage[V]');grid on
```

RARX implementation:

```
%% %%%%%%%%%%%%%%%%%%%%%%%% RecursiveARX %%%%%%%%%%%%%%%%%%%%%%%%%%%
disp('Identify model with RARX....(wait)')

na = 5;
nb = [5 5];
nk = [0 0];
z1 =[Yid(:,1) Uid];
model1 = arx(z1,[na nb nk]);

arx_obj1 = recursiveARX([na nb nk]);
arx_obj2 = recursiveARX([na nb nk]);

N=length(Uid);
for i = 1:N
    y1 = Yid(i,1);
    y2 = Yid(i,2);

    [A1_rarx,B1_rarx,y1_arx_hat] = step(arx_obj1,y1,Uid(i,:)');
    [A2_rarx,B2_rarx,y2_arx_hat] = step(arx_obj2,y2,Uid(i,:)');

end
% u=u';

% Validation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% fit = goodnessOfFit(y,yref,cost_func);
z1_1=[Yid(:,1) Uid];
model1=idss(model1);

% sys1_arx = idpoly(A1_rarx,B1_rarx,[],[],[]);
% ym_arx_1=predict(sys1_arx,z1_1,inf);
ym_arx_1=predict(model1,z1_1,inf);

% figure;
% compare(z1_1,model1,sys1_arx)

fit_1 = goodnessOfFit(ym_arx_1,Yid(:,1),'NRMSE')
fit_1 = fit_1*100;
fitE_1 = goodnessOfFit(ym_arx_1,Yid(:,1),'MSE')

z1_2=[Yid(:,2) Uid];
model2 = arx(z1_2,[na nb nk]);
model2 =idss(model2);
% sys2_arx = idpoly(A2_rarx,B2_rarx,[],[],[]);

% figure;
% compare(z1_2,model2,sys1_arx)

% ym_arx_2=predict(sys2_arx,z1_2,inf);
ym_arx_2=predict(model2,z1_2,inf);

fit_2 = goodnessOfFit(ym_arx_2,Yid(:,2),'NRMSE')
fit_2 = fit_2*100;
fitE_2 = goodnessOfFit(ym_arx_2,Yid(:,2),'MSE')
```

```
figure;
subplot(211), plot([Yid(:,1) ym_arx_1]);
title('RARX: Actual and simulated output y1')
legend('Measured',strcat('Estimated (', num2str(fit_1),'%)'));
xlabel('Samples'); ylabel('Level[m]');grid on

subplot(212), plot([Yid(:,2) ym_arx_2]);
title('RARX: Actual and simulated output y2')
legend('Measured',strcat('Estimated (', num2str(fit_2),'%)'));
xlabel('Samples'); ylabel('Level[m]');grid on

% subplot(221), plot(Uid(:,1)), title('RARX: Actual input U1')
% xlabel('Samples'); ylabel('Voltage[V]'); grid on
%
% subplot(223), plot(Uid(:,2)), title('RARX: Actual input U2')
% xlabel('Samples'); ylabel('Voltage[V]');grid on

% Prediction %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

z2_1=[Y_val(:,1) U_val];
sys1_arx = idpoly(A1_rarx,B1_rarx,[],[],[]);
% y1m_arx_1=predict(sys1_arx,z2_1,inf);
y1m_arx_1=predict(model1,z2_1,inf);

fit1_1 = goodnessOfFit(y1m_arx_1,Y_val(:,1),'NRMSE')
fit1_1 = fit1_1*100;
fit1P_1 = goodnessOfFit(y1m_arx_1,Y_val(:,1),'MSE')

z2_2=[Y_val(:,2) U_val];
sys2_arx = idpoly(A2_rarx,B2_rarx,[],[],[]);

% y1m_arx_2=predict(sys2_arx,z2_2,inf);
y1m_arx_2=predict(model2,z2_2,inf);

fit1_2 = goodnessOfFit(y1m_arx_2,Y_val(:,2),'NRMSE')
fit1_2 = fit1_2*100;
fit1P_2 = goodnessOfFit(y1m_arx_2,Y_val(:,2),'MSE')

figure;
subplot(211), plot([Y_val(:,1) y1m_arx_1]);
title('RARX: Actual and simulated output y1')
legend('Measured',strcat('Predicted (', num2str(fit1_1),'%)'));
xlabel('Samples'); ylabel('Level[m]');grid on

subplot(212), plot([Y_val(:,2) y1m_arx_2]);
title('RARX: Actual and simulated output y2')
legend('Measured',strcat('Predicted (', num2str(fit1_2),'%)'));
xlabel('Samples'); ylabel('Level[m]');grid on

% subplot(221), plot(U_val(:,1)), title('RARX: Actual input U1')
% xlabel('Samples'); ylabel('Voltage[V]'); grid on
%
% subplot(223), plot(U_val(:,2)), title('RARX: Actual input U2')
% xlabel('Samples'); ylabel('Voltage[V]');grid on
```

RPEM implementation:

```
%% %%%%%%%%%%%%%%%%%%%%%% IDENTIFY MODEL WITH RPEM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp('Identify model with RPEM....(wait)')
data = iddata(Yid,Uid,1);
z1 = [Yid(:,1) Uid];
z2 = [Yid(:,2) Uid];
m1=pem(z1,'best');
M1=idpoly(m1);

mm2=pem(z2,'best');
MM2=idpoly(mm2);
% get(m)
na=0; nb=[3 3]; nc=3; nd=0; nf=[0 0]; nk=[1 1];
nn=[na nb nc nd nf nk];

EstimatedParameters = rpem(z1,nn,'ff',1);
EstimatedParameters2 = rpem(z2,nn,'ff',1);

rpem_time = toc;
p = EstimatedParameters(end,:)
p2 = EstimatedParameters2(end,:)

% 1st output
m1_pem = idpoly([1 p(1:na)],... % A polynomial
    [[zeros(1,nk(1)) p(na+1:na+nb(1))];... % B1 polynomial
    [zeros(1,nk(2)) p(na+nb(1)+1:na+nb(1)+nc)]],... % B2 polynomial  [1
p(na+nb(1)+1:na+nb(1)+nc)]) % C polynomial
    [1 p(na+nb(1)+nc+1:na+nb(1)+nc+2)]) % D polynomial
m1_pem.Ts = 1;

% 2nd output
m2_pem = idpoly([1 p2(1:na)],... % A polynomial
    [[zeros(1,nk(1)) p2(na+1:na+nb(1))];... % B1 polynomial
    [zeros(1,nk(2)) p2(na+nb(1)+1:na+nb(1)+nc)]],... % B2 polynomial  [1
p(na+nb(1)+1:na+nb(1)+nc)]) % C polynomial
    [1 p2(na+nb(1)+nc+1:na+nb(1)+nc+2)]) % D polynomial
m2_pem.Ts = 1;

% Validation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% fit = goodnessOfFit(y,yref,cost_func);
z1_1=[Yid(:,1) Uid];
% ym_pem_1=predict(m1_pem,z1_1,inf);
ym_pem_1=predict(M1,z1_1,inf);

fit_1 = goodnessOfFit(ym_pem_1,Yid(:,1),'NRMSE')
fit_1 = fit_1*100;
fitEE_1 = goodnessOfFit(ym_pem_1,Yid(:,1),'MSE')

z1_2=[Yid(:,2) Uid];
% ym_pem_2=predict(m2_pem,z1_2,inf);
ym_pem_2=predict(MM2,z1_2,inf);

fit_2 = goodnessOfFit(ym_pem_2,Yid(:,2),'NRMSE')
fit_2 = fit_2*100;
fitEE_2 = goodnessOfFit(ym_pem_2,Yid(:,2),'MSE')

figure;
subplot(211), plot([Yid(:,1) ym_pem_1]);
```

```
title('RPEM: Actual and simulated output y1')
legend('Measured',strcat('Estimated (', num2str(fit_1),'%)'));
xlabel('Samples'); ylabel('Level[m]');grid on

subplot(212), plot([Yid(:,2) ym_pem_2]);
title('RPEM: Actual and simulated output y2')
legend('Measured',strcat('Estimated (', num2str(fit_2),'%)'));
xlabel('Samples'); ylabel('Level[m]');grid on

% subplot(221), plot(Uid(:,1)), title('RPEM: Actual input U1)')
% xlabel('Samples'); ylabel('Voltage[V]'); grid on
%
% subplot(223), plot(Uid(:,2)), title('RPEM: Actual input U2')
% xlabel('Samples'); ylabel('Voltage[V]');grid on

% Prediction %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

z2_1=[Y_val(:,1) U_val];
% y1m_pem_1=predict(m1_pem,z2_1,inf);
model1=idss(M1)
y1m_pem_1=predict(M1,z2_1,inf);

fit1_1 = goodnessOfFit(y1m_pem_1,Y_val(:,1),'NRMSE')
fit1_1 = fit1_1*100;
fitP1_1 = goodnessOfFit(y1m_pem_1,Y_val(:,1),'MSE')

z2_2=[Y_val(:,2) U_val];
% y1m_pem_2=predict(m2_pem,z2_2,inf);
model2=idss(MM2)
y1m_pem_2=predict(MM2,z2_2,inf);

fit1_2 = goodnessOfFit(y1m_pem_2,Y_val(:,2),'NRMSE')
fit1_2 = fit1_2*100;
fitP1_2 = goodnessOfFit(y1m_pem_2,Y_val(:,2),'MSE')

figure;
subplot(211), plot([Y_val(:,1) y1m_pem_1]);
title('RPEM: Actual and simulated output y1')
legend('Measured',strcat('Predicted (', num2str(fit1_1),'%)'));
xlabel('Samples'); ylabel('Level[m]');grid on

subplot(212), plot([Y_val(:,2) y1m_pem_2]);
title('RPEM: Actual and simulated output y2')
legend('Measured',strcat('Predicted (', num2str(fit1_2),'%)'));
xlabel('Samples'); ylabel('Level[m]');grid on

% subplot(221), plot(U_val(:,1)), title('RPEM: Actual input U1')
% xlabel('Samples'); ylabel('Voltage[V]'); grid on
%
% subplot(223), plot(U_val(:,2)), title('RPEM: Actual input U2')
% xlabel('Samples'); ylabel('Voltage[V]');grid on
```