Use Authorization

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at Idaho State University, I agree that the Library shall make it freely available for inspection. I further state that permission for extensive copying of my thesis for scholarly purposes may be granted by the Dean of the Graduate School, Dean of my academic division, or by the University Librarian. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Signature: _____

Date: _____

Linear System Identification via Rational Function Approximation of Empirical Transfer Function Estimates

By

Anene Vitus Omeje

A thesis

Submitted in partial fulfilment

Of the requirements for the degree of

Master of Science in Measurement and Control Engineering

College of Science and Engineering

Idaho State University

Fall 2019

Committee Approval

To the Graduate Faculty:

The members of the committee appointed to examine the thesis of Anene Vitus Omeje find it satisfactory and recommend that it be accepted.

_____

Dr. Ken W. Bosworth, Ph.D.

Major Advisor

_____

Dr. Marco P. Schoen, Ph.D., P.E.

Committee Member

_____

Dr. Steve C. Chiu, Ph.D.

Graduate Faculty Representative (GFR)

Acknowledgement

I give my sincerest gratitude to my advisor Dr. Ken W. Bosworth for his continuous support during this research. His guidance, advice, enthusiasm and immense knowledge helped me in all the time of this thesis.

I would like to thank my committee member, Dr. Marco P. Schoen, who through his Advanced Measurement class introduced me to independent research, paper review and presentation.

In addition, a thank you to Dr. Steve Chiu, who was willing to be my Graduate Faculty Representative.

I would also like to thank Dr. Thom Baldwin, and all the other professors from whom I had an opportunity to learn.

Special thanks to my lovely wife Rebecca, whose motivation and emotional support kept me going through my graduate program.

Table of Contents

List of Figures

List of Tables

Linear System Identification via Rational Function

Approximation of Empirical Transfer Function Estimates

Thesis Abstract – Idaho State University (2019)

The most widespread approach to optimal prediction of discrete-time systems relies on prediction error methods (PEMs), for which a bulk of theoretical results is available. In this thesis research, we consider the problem of identifying the transfer function of discrete linear systems using frequency domain approach. An ARX model structure is used to represent known system models, and their approximate empirical transfer function estimates are generated using random input sequence. The main objective is to recover the original system models using the empirical transfer function estimates as major input to our main MATLAB® coded program. The effectiveness of this method is illustrated by comparing, with no bias, the test problems' results with those from MATLAB® system identification toolbox.

Key Words: Empirical transfer function estimates (ETFE), *ratdisk*, ARX model.

Chapter 1: Introduction

1.1 Overview

System identification is a technique that is used to obtain mathematical model of dynamic process based on the input and output signal from a process or plant. The model obtained can relate the input and output relationship involved in the experiment. Generally, a modeling approach starts with one or more experiments involving the system to obtain the input and output data, followed by model structure selection, model estimation and model validation.

Pedro Gonnet, Richardo Pachon and Lloyd Trefethen in their paper titled "*Robust Rational Interpolation and Least-Squares"* proposed an algorithm, with MATLAB® program implementation, that removes spurious poles in a function defined on the unit circle in the complex plane and produces a robust linearized least-squares approximants [7].

This thesis presents a method of identifying a discrete time linear system using the empirical transfer function estimates obtained from an ARX model representation of the rational function, with the help of a 58-line MATLAB® code named *Ratdisk;* written by PEDRO GONNET et al and published in 2011. An ARX model structure is used to represent the known system and coded in MATLAB®, where the approximate empirical transfer function estimate of this known system is supplied to the *Ratdisk* for identification. *Ratdisk*, a MATLAB® code, accepts the empirical transfer function, the numerator and the denominator orders of the system, number of samples, and then estimates the original transfer function of the system. Like every other black box model estimation, this method does not require much prior knowledge of the parameters that make up

the dynamics of the system, however the code is written such that the system order, or a guess of the system order be supplied as an input to the code.

1.2 System Identification Procedure

System identification is a procedure to build a mathematical model of the dynamics of a system from measured data. It could be considered as approximate modeling for a specific application based on available observed data and prior system knowledge [2]. Unlike modeling from first principles, which requires an in-depth knowledge of the system under consideration, system identification methods can handle a wide range of system dynamics without much knowledge of the actual system physics.

1.3 System Identification Basic Entities

The construction of model from a given measurement data involves three basic entities [1]:

i. A set of data
ii. A model structure
iii. A rule by which the model structure can be assessed using the data.

1.4 The system Identification Loop

The system identification procedure has a fundamental logical routine. First is data collection, next is choosing a model set and picking the best model in this set. It is common to obtain a first model that does not pass the model validation test [1]. In which case the various steps of the procedure be revised. Figure 1.1 describes in some detail the procedure for system identification.

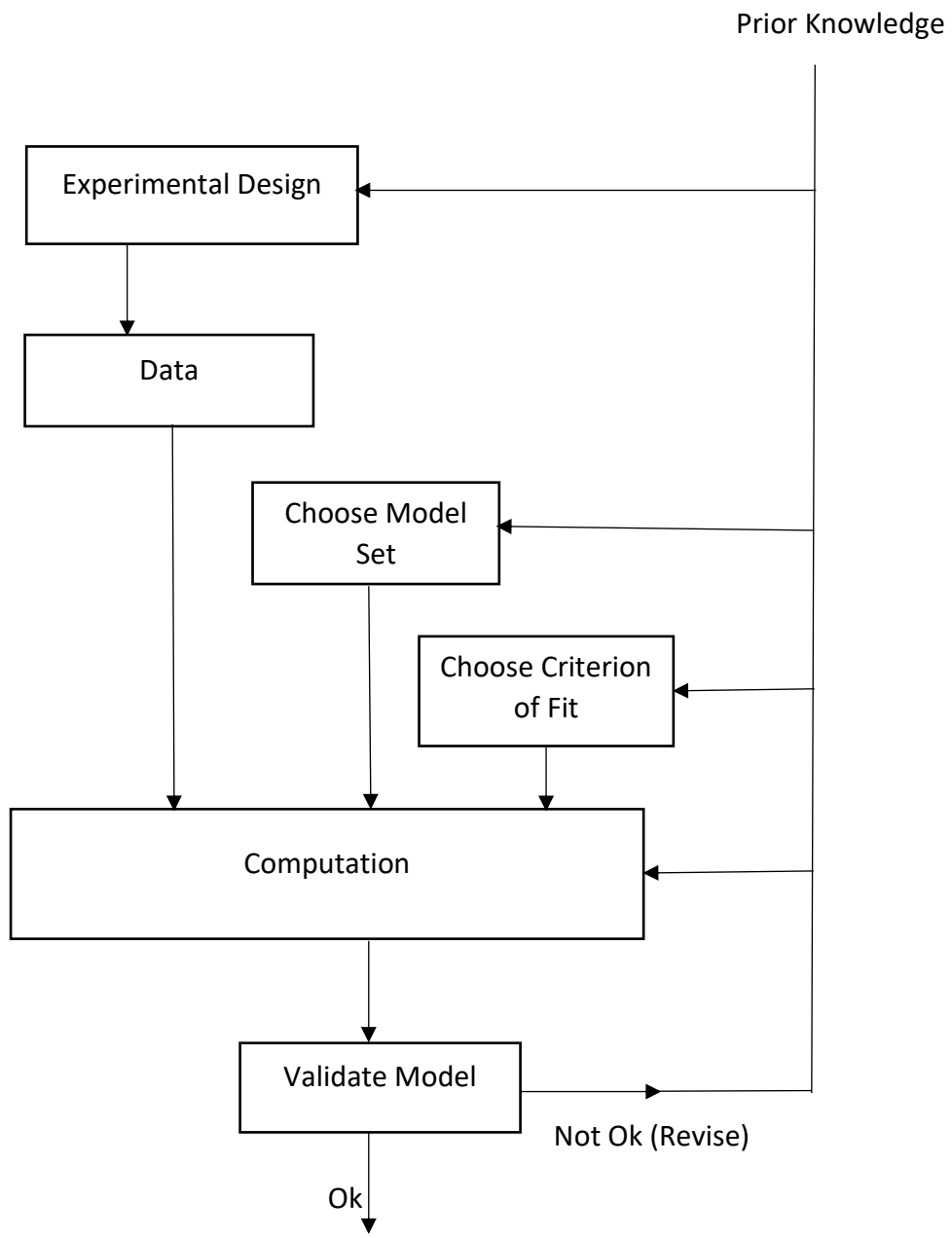

Figure 1.1: The System Identification Loop

1.5 Objectives:

- The objective of this research work is to use frequency domain approach via empirical transfer function estimates (ETFE) in identification of systems.
- Second objective is to analyze and compare the results of the systems identified using *Ratdisk* code and that obtained from MATLAB® system identification toolbox GUI.

1.6 Thesis Outline

Chapter two presents the different categories of system models. We discuss the Black Box Model, the General Linear Polynomial Model, and some other important equation error models used in system identification.

Chapter three introduces linear dynamic systems in the form of difference equations. We present a least-square solution approach to the ARX Model. We also discuss the validation of estimated models.

Chapter four presents the frequency domain analysis of rational functions. We explain the formation of the empirical transfer function estimates, the discrete Fourier transform, the fast Fourier transform, and the Z-transform.

Chapter five is a collection of test problems, the numerical results, and analysis.

Chapter six presents the conclusion and possible future research work.

Chapter 2: System Identification Models

Choosing a proper model structure is an important requirement before attempting system estimation. There are two major categories of models common in system identification: the black-box model and the grey-box model.

2.1 The Black-box Model

The black-box model assumes that the physical dynamics of systems are unknown and that parameters are adjustable. The black box is an abstraction representing a class of concrete open-loop systems which can be viewed solely in terms of their inputs and output reactions. Black-box modeling is useful when the primary interest is in fitting a set of data regardless of the original mathematical structure of the model [4]. A variety of parametric model structures are available to assist in modeling the black box. Parametric models describe systems in terms of differential equations and transfer functions. These models provide understanding of the physical system dynamics and compact structures. It is important to test several structures to determine the best model fit. A parametric model could define a continuous-time system or a discrete-time system. This research is focused on discrete-time systems identification.

2.2 The Grey-box Model

In Grey-box modeling, part of the information about the underlying dynamics or some of the physical parameters of the system are already known [3]. One can select the grey box model to specify these partially known physical parameters or some constraints. Then, the remaining unknown parameters of the partially known model will be determined using estimation methods.

Grey-box model identification involves optimization process to minimize the difference between the estimated output and the measured real output. Finding the global optimum depends on the limit range set and the initial values.

2.3 General Linear Model

This represents systems using (2.1) below, which is known as the general-linear polynomial model.

$$y(t) = G(q,\theta)u(t) + H(q,\theta)e(t) \tag{2.1a}$$

$$e(t) = \text{zero mean white noise} \tag{2.1b}$$

$u(t)$ and $y(t)$ are the input and output of the system respectively. $G(q,\theta)$ is the transfer function of the deterministic part of the system while $H(q,\theta)$ is the transfer function of the stochastic part of the system (q is backward time shift operator). θ is the parameter vector that ranges over the subset $\mathbf{R}^d$, where d is the dimension of θ[1].

$$\theta \in D_M \subset \mathbf{R}^d \tag{2.2}$$

D_M is set of values over which θ ranges in a model structure.

(2.1) to (2.2) represent a set of models. Depending on the purpose of the model, the focus is to select via estimation that member of the set that appears to be most suitable for the model.

One step output prediction of (2.1) is represented as follows [1]:

$$\hat{y}(t/\theta) = H^{-1}(q,\theta)G(q,\theta)u(t) + [1 - H^{-1}(q,\theta)]y(t) \tag{2.3}$$

$\hat{y}(t/\theta)$ is the estimate of the output $y(t/\theta)$

Models that specify G and H as in (2.1) and (2.3) could be called predictor model [1].

M(θ) is a model associated with the parameter θ .

The generalized linear model structure has five polynomials A, B, C, D, and F [3].

$$A(q)y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t) \tag{2.4}$$

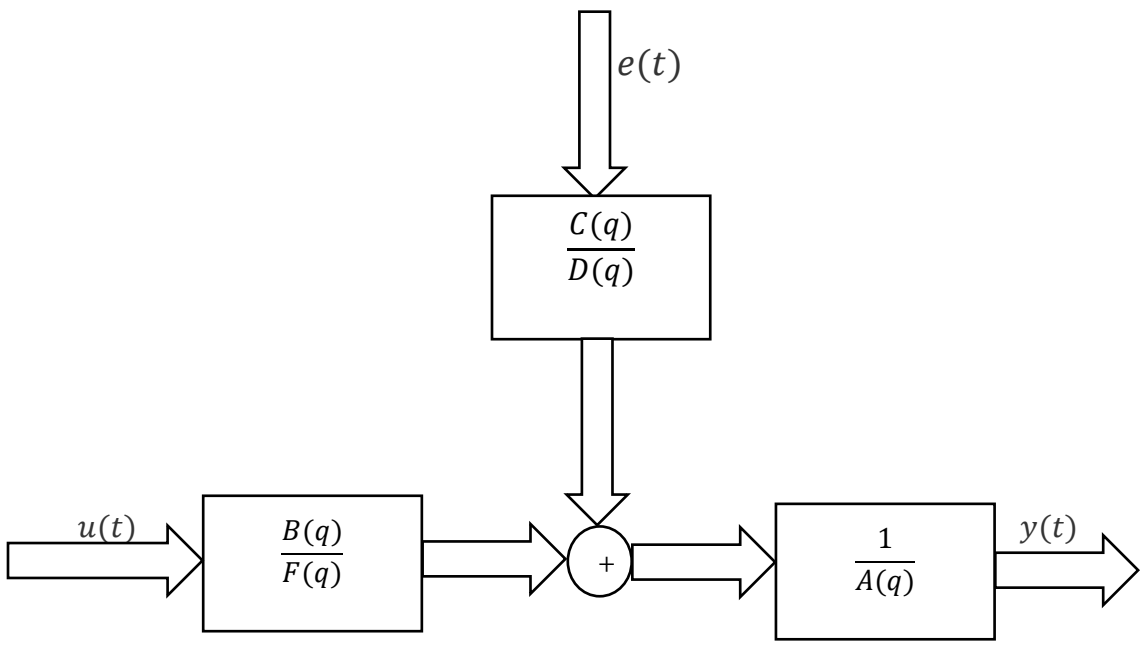

Figure 1.2 General Linear Polynomial Model

There are other simpler models that are subset of the general linear model. In practice, the general model is generally over parameterized. Following are other less complex model structures frequently used in system identification. These less complex models are obtained by setting some polynomials in (2.4) to unity [1],[3].

2.3.1 The AR Model

The Autoregressive model structure is a process model used in the generation of models where outputs depend only on previous outputs [1],[2]. It has no system inputs or disturbances. There is a limitation to the class of problems AR model can solve due to its simplicity [3]. It is a model for a signal, not a system [1]. The AR process is an example of a stochastic process, which has degrees of built in randomness – that is, being able to predict future trends well with past data, but we will never get 100 percent accuracy [3]. As a subset of the general linear model, $B(q)$, $C(q)$ and $D(q)$ are unity.

$$y(t) = \frac{1}{A(q)} e(t) \tag{2.5}$$

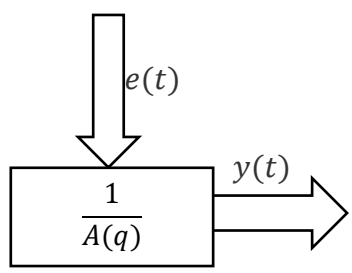

Figure 1.3 AR Model

2.3.2 The ARX Model

The ARX is an acronym for *AutroRegression* with *eXogenous input* [1]. It is the simplest model incorporating the stimulus signal [3]. The estimation of the ARX model is the most efficient of the polynomial estimation methods because it is the result of solving linear regression equations in analytic form [4]. The ARX model is used in this research work to accurately represent the benchmark systems and polynomials used in testing our proposed method of identification.

$$y(t) = \frac{B(q)}{A(q)}u(t) + \frac{1}{A(q)}e(t) \tag{2.6}$$

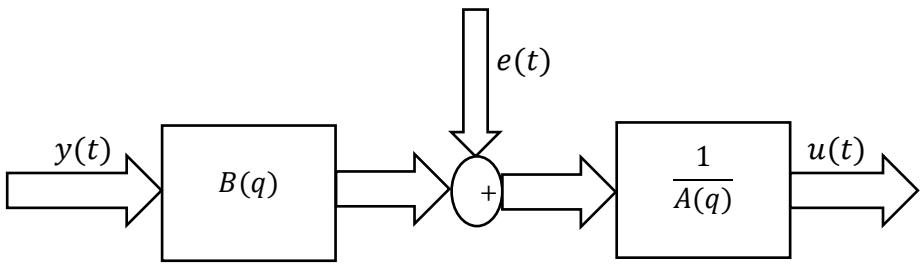

Figure 1.4 ARX Model Structure

2.3.3 ARMA Model

Autoregressive Moving Average (ARMA) is a representation of a stochastic process. The stochastic part of the general linear model (2.4) is an ARMA process. For ARMA models; $B \equiv 0, D \equiv F \equiv 1$ [1]. ARMA is also a stationary process.

$$v(t) = \frac{C(q)}{A(q)}e(t) \tag{2.7}$$

$v(t)$ is the stochastic part of the general linear model (2.4).

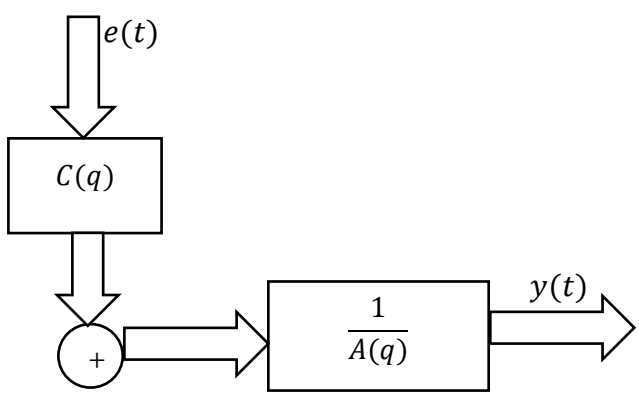

Figure 1.5 ARMA Model Structure

2.3.4 ARMAX Model

The Autoregressive Moving Average Exogenous (ARMAX) model structure is ARX with disturbance dynamics. It adds flexibility to ARX model by describing the equation error as a moving average of the disturbance term [1]. This model is useful when there is a dominating disturbance that enters early in the process, such as at the input [3].

$$y(t) = \frac{B(q)}{A(q)}u(t) + \frac{C(q)}{A(q)}e(t) \tag{2.8}$$

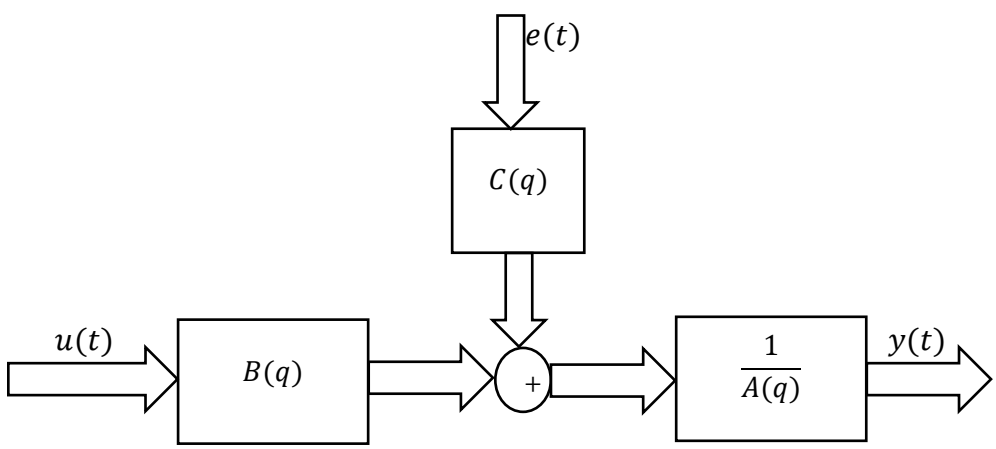

Figure 1.6 ARMAX Model Structure

2.3.5 Box-Jenkins Model

The Box-Jenkins (BJ) named after statisticians George Box and Gwilym Jenkins is a model structure that provides a complete model with disturbance properties modeled separately from system dynamics. This model type is mostly used when there are disturbances that enter late in the process [3].

$$y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t) \tag{2.9}$$

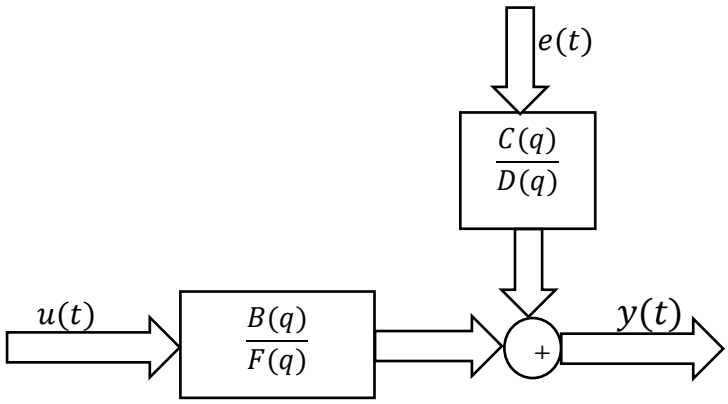

Figure 1.7 Box-Jenkins Model Structure

2.3.6 Output-Error Model (OE)

This model describes system dynamics separately and no parameters are used for modelling the disturbance characteristics.

$$y(t) = \frac{B(q)}{F(q)} u(t) + e(t) \tag{2.10}$$

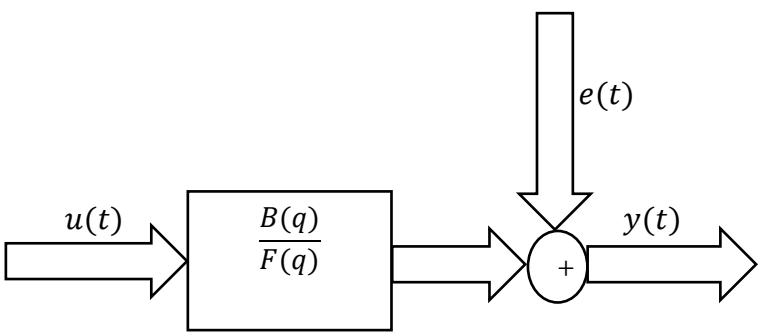

Figure 1.8 Output Error Model Structure

2.4 Transfer Function Model

The general-linear polynomial models discussed in 2.4 are used in stochastic control – where the deterministic and stochastic parts are described. Transfer function models are commonly

used to describe only the deterministic part of a system. In classical control engineering, the deterministic part of the system is more important than the stochastic part. Transfer function models can describe both continuous-time and discrete-time systems.

In the time domain, for linear time invariant (LTI) systems, we have

$y(t) = g(t) * u(t) + e(t)$ or

$y(k) = g(k) * u(k) + e(k)$

with $g(t), g(k)$, being the unit impulse response of the system, and * represents convolution (over the time range (0,t) or (0,k) since we assume causality).

In the frequency (Laplace or Z-transform) domain, we have

$Y(s) = G(s)U(s) + E(s)$

$Y(z) = G(z)U(z) + E(z)$

with upper case variables denoting when $\mathcal{L}\big(g(t)\big) = G(s)$ or $z\big(g(t)\big) = G(z)$.

$G(s)$ and $G(z)$ are the "Transfer" functions.

By an abuse of notation, which is generally accepted in the controls community,

we write:

$y(t) = G(s)u(t) + e(t)$

$y(k) = G(z)u(k) + e(k)$

as denoting Transfer function models.

2.4.1 State-Space Model

The classical parametric system identification model minimizes performance function based on sum of square errors. These methods work well in many cases. However, for complex systems with high order, multiple inputs and outputs, and large number of measurements; the classical methods suffer from several problems. They could experience local minima in the performance function which could result to no convergence to global minima. State-space model is preferred when the system has such complicated multiple-input and multiple-output (MIMO) physical systems. The following equations describe a state-space model.

$$x(t+1) = Ax(t) + Bu(t) + Ke(t)$$

$$y(t) = Cx(t) + Du(t) + e(t) \tag{2.11}$$

$x(t)$ is the state vector, $y(t)$ is the system output, $u(t)$ the system input and $e(t)$ is the stochastic error. A, B, C, D, and K are the system matrices. The dimension of the state vector $x(t)$ is the only parameter needed to provide for the state-space model. This classical method of representing linear dynamic systems became popular after Kalmam's (1960) work on prediction and linear quadratic control [1]. The State-space model is similar to "*first principle models*", and so, provides a more complete representation of system than the polynomial models [3].

Chapter 3: ARX Model and the Linear Least Squares Method

3.1 Linear dynamic System

A *Linear difference equation* is a common way of representing the relationship between input and output of the ARX model of a linear dynamic system [1].

$$y(t) + a_1 y(t-1) + \cdots + a_n y(t-n) = b_1 u(t-1) + \cdots + b_m u(t-m) + e(t) \tag{3.1}$$

$e(t)$ is white noise that enters as direct error in the difference equation.

A more useful way of presenting (3.1) is:

$$y(t) = -a_1 y(t-1) - \cdots - a_n y(t-n) + b_1 u(t-1) + \cdots + b_m u(t-m) + e(t) \tag{3.2}$$

Vector representation of (3.1) for more compact notation:

$$\theta = [a_1 \dots a_n \;\; b_1 \dots b_m]^T \tag{3.3}$$

$$\varphi(t) = [-y(t-1) \dots - y(t-n) \;\; u(t-1) \dots u(t-m)]^T \tag{3.4}$$

$$y(t) = \varphi^T(t)\theta$$

$$A(q) = 1 + a_1 q^{-1} + \cdots + a_n q^{-n}$$

$$B(q) = b_0 + b_1 q^{-1} + \cdots + b_m q^{-m}$$

Equation (3.1) corresponds to (2.1)-general linear model equation:

$$G(q,\theta) = \frac{B(q)}{A(q)}, \;\; H(q,\theta) = \frac{1}{A(q)} \tag{3.5}$$

$A(q)y(t)$ is the autoregressive part of the ARX model (3.1) while $B(q)u(t)$ is the exogeneous term [1].

The output $y(t)$ in this linear regression model (3.1) is defined in terms of the variables φ plus an unobserved error term e. Let the independent variable $t = 1, \dots, N$, and $y := [y(1), \dots, y(N)]^T$,

$e = [e(1), \dots, e(N)]^T$, $\theta = [a_1 \dots a_n \ b_1 \dots b_m]^T$as defined in (3.3) be a column vectors of the appropriate dimension.

$$\Phi_{t_j} := \varphi_j(t) \tag{3.6}$$

$j = 1, \dots, p$. p is the number of unknown parameters, $p = n + m$.

The model 3.1 can be expressed in matrix notation as

$$y = \Phi\theta + e \tag{3.7}$$

It can be seen however, that (3.1) is a description of a dynamic system with unknown parameter θ and an observation matrix, called the regressor matrix Φ, with known elements relating the parameters to the observations [2].

3.2 Least Squares Estimation

In regression analysis, least squares estimation, also called least squares fitting, is the technique of finding the best fit curve for a set of data. At the end of the eighteenth century, Karl Friedrich Gauss formulated the principles of the least squares while he was determining the orbits of planets and asteroids. He stated that the unknown parameters θ of a mathematical model should be chosen in a such a way that:

the sum of the squares of the differences between the actually observed and the computed values, multiplied by numbers that measure the degree of precision, is a minimum [11].

Gauss postulated that a good way of estimating the unknowns from a given measurement data is by demanding that the *prediction* errors, also called *residuals* $\varepsilon(t) \coloneqq y(t) - \varphi^T(t)\theta$, are small [2]. This is formally stated thus; that we shall choose the parameter vector θ such that the sum of squared prediction errors

$$J(\theta) := \sum_{t=1}^{N} \varepsilon^2\,(t) \;= \sum_{t=1}^{N} [y(t) - \vartheta^T(t)\theta]^2 \tag{3.8}$$

is minimal. The quantity $J(\theta)$ is a scalar function known as least-squares objective function [2]. Using the matrix theory $(\Phi\theta)^T = \theta^T\Phi^T$, equation (3.8) can be written as

$$J(\theta) \coloneqq \varepsilon^T\varepsilon = (y^T - \theta^T\Phi^T) \tag{3.9}$$

The scalar J is minimal if and only if the derivative of J with respect to θ is zero. In other words, J is a p-dimensional vector and the second derivative is positive. Equations (3.10) and (3.11) are standard results of vector-matrix derivatives, which were employed in deriving the normal equation as follows;

$$\frac{\partial a^T\theta}{\partial\theta} = a \tag{3.10}$$

$$\frac{\partial\theta^T A\theta}{\partial\theta} = (A + A^T)\theta \tag{3.11}$$

The terms in the scalar function J are also scalars, so that,

with $y, \Phi\theta \epsilon \mathbb{R}^N$, $y^T\Phi\theta = \theta^T\Phi^T y$ and then,

$$J(\theta) = y^T y - y^T \Phi\theta - \theta^T \Phi^T y + \theta^T \Phi^T \Phi\theta$$

$$= y^T y - 2\theta^T \Phi^T y + \theta^T \Phi^T \Phi\theta \tag{3.12}$$

Therefore, the derivative of J in (3.12) with respect to θ gives the following

$$\frac{\partial J(\theta)}{\partial \theta} = -2\Phi^T y + 2\Phi^T \Phi\theta \tag{3.13}$$

The derivative of $J(\theta)$ is zero if and only if

$$\Phi^T \Phi\hat{\theta} = \Phi^T y \tag{3.14}$$

Which are referred to as the *normal equations.* Mathematically (but not numerically), the ordinary least squares estimate can be deduced from (3.14) by multiplying both sides with $(\Phi^T \Phi)^{-1}$(assume Φ has full rank).

$$\hat{\theta} = (\Phi^T \Phi)^{-1} \Phi^T y \tag{3.15}$$

Equation (3.15) estimates the unknown parameters θ if and only if p X p matrix $\Phi^T \Phi$ is invertible. The next is to show that $\hat{\theta}$ gives a minimum of J.

Let $\theta = \hat{\theta} + \Delta\theta$ (3.16)

A substitution of (3.15) into (3.12) gives

$$J(\hat{\theta}) = J(\theta) - (\Delta\theta)^T \Phi^T \Phi(\Delta\theta) \tag{3.17}$$

Thus, $J(\theta)$ has a minimum at $\hat{\theta}$ if

$$(\Delta\theta)^T \Phi^T \Phi(\Delta\theta) > 0$$

Least-squares method of estimation could also be interpreted from the perspective of the dependency between the estimate and the number of output samples N. In a case where the number of output measurements is equal to the number of the unknown parameters p;

$$\hat{\theta} = \Phi^{-1} y \tag{3.18}$$

Equation (3.18) gives the parameter estimates $\hat{\theta}$ provided Φ^{-1} is invertible, in other words, the columns of the square matrix Φ are independent. The noise associated with the output in this case (of $N = p$) is directly reflected in the estimate $\hat{\theta}$[2]. In practice, N is mostly chosen much larger than the number of unknown parameters. If $N > p$, as is the case of this thesis main work, there are more equations than unknown parameters, and the estimate is that in (3.15);

$\hat{\theta} = (\Phi^T \Phi)^{-1} \Phi^T y$, where $(\Phi^T \Phi)^{-1} \Phi^T$ is called the pseudo inverse of Φ.

The third case is where $N < p$, the number of unknowns exceed the number of equations, and therefore, no unique solution exists.

Note that model (3.1) can be represented such that y(t) could be seen as being calculated from past data that depends on the parameters in θ .

$$\hat{y}(t/\theta) = \varphi^T(t)\theta = \theta^T \varphi(t) \tag{3.19}$$

$\hat{y}(t/\theta)$ is the output estimation.

3.3 Identifiability

The basic assumption in most identification exercise is identifiability. The problem of identifiability is whether the identification procedure will give a unique parameter θ, or whether the resulting model is equivalent to the true system [1]. In other words, identifiability is the ability to estimate a model uniquely, translating to:

- The model being unique with respect to its parameter
- Being able to resolve unambiguously between models of different structures with data evidence
- Being equipped with the ability to uniquely estimate the parameters of given model structure from a given data set.

In this research, we assume a-priori that our system model is estimated with a large enough data set with sufficiently "*rich*" inputs that identifiability is generally assumed.

3.4 Validation of Estimated Model

In order to validate the estimated output, the following approaches may be used [12]:

- Compare the estimated output to the actual output [12]. The difference between the estimated output and actual output are the residuals.
- Analysis of autocorrelation and cross-correlation of residuals with input.
- Analysis of model response in terms of impulse, step, or frequency response plots [12].
- Considering plots of poles and zeros of linear parametric model [12].

- Comparison between non-parametric and parametric models [12]. Nonparametric models would include impulse, step, and frequency response models; whereas parametric models would include linear polynomial models, state-space models, and non-linear parametric models [12].
- Comparison between models through Akaike Information Criterion, or Akaike Final Prediction Error [12].
- Making linear and non-linear plots for Hammerstein-Wiener and non-linear ARX models [12].

3.4.1 Residuals $(\boldsymbol{r_i})$ and Residual Sum of Squares $(\boldsymbol{RSS})$

Residuals are the differences between the observed data, and what is predicated by the least square solution of a regression equation. The RSS is the measure of discrepancy between the actual(observed) data and the identified model [7], [8].

$$r_i = y_i - \hat{y}_i \; ; \qquad (3.20)$$

$$i = 1, 2, \dots N$$

y_i = observed data points.

$\hat{y}_i$ = predicted output data points.

N = total number of data points

$$RSS = \sum_i^N (y_i - \hat{y}_i)^2 \qquad (3.21)$$

3.4.2 Mean Square Error (MSE)

The MSE is a risk function that measures the average squared difference between the estimated values and the actual value [7].

$$MSE = \frac{RSS}{N} \tag{3.22}$$

$$= \frac{\sum_i^N (y_i - \hat{y}_i)^2}{N} \tag{3.23}$$

3.4.3 Percentage Fit to Estimation Data

This is a measure of how well the response of the model fits the estimation data, expressed as a percentage.

$$Perc\_fit = \left(\left(1 - \frac{MSE}{y_{norm}}\right) * 100\right) \tag{3.24}$$

$$y_{norm} = \frac{\sum_i^N (y_i)^2}{N} \tag{3.25}$$

Chapter 4: Frequency Domain Analysis of rational functions

The focus of this identification method is based on discrete time and frequency domain method.

4.1 Impulse Response Analysis

For

$$y(t) = G(q)u(t) + v(t) \tag{4.1}$$

Subject to the impulse input $u(t) = \begin{cases} \alpha \ , \ t = 0 \\ 0 \ , \ t \neq 0 \end{cases}$ (4.2)

$G(q)$ is the true transfer function from u to y; with forward shift operator q

$v(t)$ is disturbance, maybe white noise [2].

From the definition of transfer function and impulse response, the output will be

$$y(t) = \alpha g(t) + v(t)$$

Assuming the noise is negligible, $\{g(t)\}$ could possibly be determined from an experiment with a pulse input. The impulse response coefficient estimate is

$$\hat{g}(t) = \frac{y(t)}{\alpha} \tag{4.3}$$

and the error term estimate is $\frac{y(t)}{\alpha}$. The shortcoming of this impulse response estimation is that many physical processes do not allow pulse inputs of such an amplitude (α) with insignificant error compare to the impulse response coefficients [1].

4.2 Step-Response Analysis

$$u(t) = \begin{cases} \alpha \ , \ t \geq 0 \\ 0 \ , \ t < 0 \end{cases} \tag{4.4}$$

Applying the step response (4.4) in (4.1) gives the output

$y(t) = \alpha \sum_{k=1}^{t} g(k) + v(t)$

Estimates of $g(k)$ could be obtained as

$$\hat{g}(t) = \frac{y(t) - y(t-1)}{\alpha} \tag{4.5}$$

Whereas the error should be $\frac{v(t) - v(t-1)}{\alpha}$.

Determining the impulse response coefficients using (4.5) would involve dealing with large errors.

4.3 Sine Wave Response

Linear time invariant (LTI) systems can be identified by the sinewave. Sinewave is represented as follows:

$$u(t) = \alpha \sin \omega t, \quad t = 0,1,2 \ \ldots \tag{4.6}$$

$G(j\omega)$ with j the complex number is the transfer function in frequency domain, otherwise known as frequency transfer function. It is the Fourier transform of the impulse response $g(t)$. This frequency function is found by simply substituting $j\omega$ for s in the Laplace transfer function $G(s)$[2].

For sampled systems, instead of the Laplace, Fourier transform and the discrete Fourier transform (DFT) of $g(t)$ are used; that is

$$G(e^{jw}) = \sum_{t=-\infty}^{\infty} g(t)e^{-jwt} \tag{4.6}$$

The DFT is the discrete version of the Fourier transformation.

4.4 Sine Wave Response Identification

Let $I_m(z)$ denote the imaginary part of z. That is, if $z = a + ib$, then $I_m(z) = b$.

In complex numbers, $\sin\omega t = e^{j\omega t}$, and $G(e^{j\omega})$ is a complex number which could be written as $|G(e^{j\omega})|e^{j\phi}$, where $|G(.)|$ is the magnitude and $\phi = \arg\big(G(.)\big)$.

With k in $(-\infty, \infty)$, the sinewave input (4.6) gives an output

$$y(t) = \alpha \sum_{k=-\infty}^{\infty} g(k) I_m(e^{j\omega(t-k)})$$

$$= \alpha I_m \sum_{k=-\infty}^{\infty} g(k)(e^{-j\omega(k-t)})$$

$$= \alpha I_m \left\{ e^{j\omega t} \sum_{k=-\infty}^{\infty} g(k) e^{-j\omega k} \right\}$$

$$= \alpha I_m \left\{ e^{j\omega t} \sum_{k=-\infty}^{\infty} G(e^{j\omega}) \right\}$$

$$= \alpha |G\big(e^{j\omega}\big)| \sin(\omega t + \phi) \tag{4.7}$$

The output is a sinewave of the same frequency as $u(t)$, but multiplied in magnitude by $|G(e^{j\omega})|$ and shifted in phase by ϕ. This result assumes that the input is an everlasting sinewave which is never true in practice. Therefore, whenever $u(t) = 0, t < 0$, an initial transient must be accepted in the response. This could be handled by neglecting the first part of the response [2].

4.5 Empirical Transfer Function Identification

4.5.1 Sine Wave Testing

The complex-valued function $G(e^{j\omega})$, $-\pi \leq \omega \leq \pi$, is the frequency function of a discrete - time LTI system. In this work, we directly identify frequency transfer function from data gotten from known systems. Sinewave testing and its process is one of the simplest methods of frequency function identification. The method of sinewave testing repeatedly uses a single frequency sinewave at a time and this makes it time-consuming. This shortcoming is overcome using multifrequency inputs – leading to discrete Fourier transforms of signals [2].

4.5.2 Discrete Fourier Transform (DFT) of Signals

The discrete Fourier transform is a transform like the Fourier transform but is used with digitized signals. It is the discrete version of the FT that views both the time domain and frequency domain as periodic.

The DFT of the signal $y(t)$, sampled at $t = 1, 2, \ldots N$, is given by

$$Y_N(\omega) = \frac{1}{\sqrt{N}} \sum_{t=1}^{N} y(t) e^{-j\omega t} \tag{4.8}$$

where $\omega = \omega_k = \frac{2\pi k}{N},\ k = 1,2,\ldots,N$. For specific k , $\frac{N}{k}$ is the period associated with the specific frequency ω_k.

In a similar way, the DFT of the randomly constructed input data $u(t)$ used in this work was made. The absolute square value of $Y(\omega_k)$, $|Y\left(\frac{2\pi k}{N}\right)|^2$, is a measure of the energy contribution of this frequency to the energy of the signal. A plot of $|Y(\omega)|^2$ as function of ω is called the periodogram of the signal $y(t)$.

4.5.3 Empirical Transfer Function Estimate

The empirical transfer function is the equivalent of the Laplace transfer function substituting $j\omega$ for s.

$$Y(j\omega) = G(j\omega)U(\omega) \tag{4.9}$$

This type of algebraic relationship is also applicable in sampled systems. For a given input $u(t)$ and an output signal $y(t)$, the following estimate of the transfer function is found:

$$\hat{G}\left(e^{j\omega}\right) = \frac{Y_N(\omega)}{U_N(\omega)} \tag{4.10}$$

Equation (4.10) is the empirical transfer function estimate (ETFE). This expression holds for any frequency which can be detected; which is the case for this thesis work. U_N and Y_N are respectively, series expansions of the input and the output signals in terms of sines and cosines. For each of the frequencies contained in $u(t)$ and $y(t)$, the relationship of (4.7) holds [2], which allows the reconstruction of both the magnitude and phase shift of the frequency function for N number of frequencies [2]. The DFT of these data vectors after component-wise

division gives $\hat{G}(e^{j\omega})$ consisting of $N/_2$ data points , for $\omega = \frac{2\pi k}{N}, k = 0, 1, \dots, N-1, \pi$ (rad/s) $[1], [2]$. It is assumed that $U_N(\omega) \neq 0$. When this does not hold for some frequencies, the ETFE is undefined at those frequencies [1].

4.5.4 Properties of the ETFE

I. It decays as $\frac{1}{\sqrt{N}}$ as in (4.8), N is a multiple of the periods.

II. The ETFE $G(e^{j\omega})$ is defined only for a fixed number of frequencies.

III. At those frequencies in which ETFE is defined, $G(e^{j\omega})$ is unbiased and its variance decays like $\frac{1}{N}$.

4.5.5 Algorithm for Identification of Frequency Function from Input and Output Data.

I. Generate random input signal $u(t)$, $t = 1,2, \dots N$.

II. Apply the input signal to the system, with a zero-order hold on the inputs.

III. Record the input signal $u(t)$ and the corresponding output signal $y(t)$.

IV. Take the DFT of input $u(t)$ and output $y(t)$, resulting in $N_N(\omega)$ and $Y_N(\omega)$ respectively.

V. Make a component-wise division of $Y_N(\omega)$ by $N_N(\omega)$ for $\omega = \frac{2\pi k}{N}, k = 0, 1, \dots, N-1, \pi$ (rad/s).

VI. Use elementary frequency response to get an estimate of the polynomial transfer function $G(z)$.

VII. Directly start from step 3 if a disturbance is noticed on the input signal [2].

4.5.6 Fast Fourier Transform (FFT)

The fast Fourier transform is an implementation of the DFT which produces the same results as the DFT, but in a much more efficient and faster way. It is said to be an algorithm used for fast and efficient computation of DFT.

4.6 The Z – Transform

It is discussed in 4.1 (Fourier analysis) that $G\left(e^{j\omega}\right)$ is the discrete Fourier transform (DFT) of the impulse response $g(t)$. In other words, $G\left(e^{j\omega}\right)$ is the transfer function of a sampled system evaluated at the point where $z = e^{j\omega}$.This number gives complete information about what happens under stationary condition; when the input is sinewave of frequency ω. The z-transform is basically the discrete time transfer function of $g(t)$ [2]. For sampled input-output data vector $u(t)$ and $y(t)$, the z-transform is defined as:

$$U(z) \coloneqq \sum_{k=0}^{\infty} u(k) z^{-k} \tag{4.11}$$

and

$$Y(z) \coloneqq \sum_{k=0}^{\infty} y(k) z^{-k} \tag{4.12}$$

respectively.

The z-transform is also the discrete counterpart of the Laplace transfer function. Consider the Laplace Transform

$$\mathcal{L}|f(x)| \coloneqq F(s) \coloneqq \int_0^{\infty} f(x) e^{-st}\, dt, \tag{4.13}$$

whose convolution transformation gives the continuous s-domain function

$$Y(s) = G(s)U(s) \tag{4.14}$$

In a similar way,

$$Y(z) = G(z)U(z) \tag{4.15}$$

The transfer function of (4.14) in z-domain is

$$G(z) = \frac{Y(z)}{U(z)} \tag{4.16}$$

$$\frac{Y(z)}{U(z)} := \frac{\sum_{k=0}^{m} b_k z^{-k}}{\sum_{k=0}^{n} a_k z^{-k}} \tag{4.17}$$

For causal system: $n \geq m;\ b_0 = 0,\ \ a_0 = 1.$

$$\frac{Y(z)}{U(z)} := \frac{\sum_{k=1}^{m} b_k z^{-k}}{1+\sum_{k=1}^{n} a_k z^{-k}} = \frac{b_1 z^{-1}+\cdots+b_m z^{-m}}{1+a_1 z^{-1}+\cdots+a_n z^{-n}} \tag{4.18}$$

Chapter 5: The Numerical Results

5.1 Collection of Test Problems

Rational functions of known systems are obtained from benchmark automatic control research papers. Because we are working on discrete-time, continuous-time transfer functions from these research papers are converted to their discrete-time equivalent using MATLAB®, with a sample time of 0.1 seconds.

Each of the test problems is analyzed in four different ways:

- With no additional random noise
- With input noise w_1
- With output noise w_2
- With input and output noise

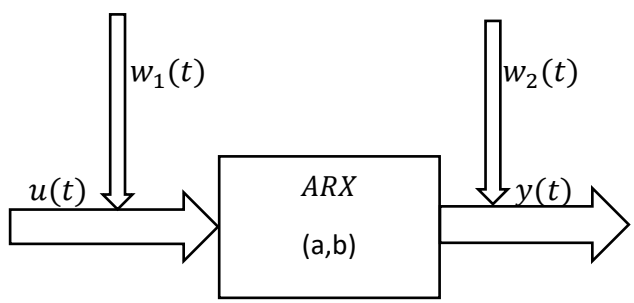

Figure 5.1 The ARX With Input and Out Noise

5.2 Test Function 1

Recovering Wiener-Hammerstein Nonlinear State-Space Models Using Linear Algebra (P. Dreesen, M. Ishteva, and J. Schoukens, 2015)-System 1 [12].

$$G\_true(z) = \frac{0.7071z+4.2426}{z^2+0.750z+0.125} \tag{5.1}$$

5.2.1 Test Function 1 With No Input & Output noise:

$$G\_ratdisk(z) = \frac{0.701z+4.2427}{z^2+0.7508z+0.1262} \tag{5.2}$$

$$G\_SysID(z) = \frac{0.3417z}{z^2+0.672z+0.142} \tag{5.3}$$

Table 5.1: System_ID & Ratdisk Results of Test Function 1 With No Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	-0.3396 + 0.1633i	-0.4966	-0.5
Eigenvalue 2	-0.3396 - 0.1633i	-0.2542	-0.25
Percentage fit to estimation	20.74	100	__
MSE	17.63	1.3853e-04	__

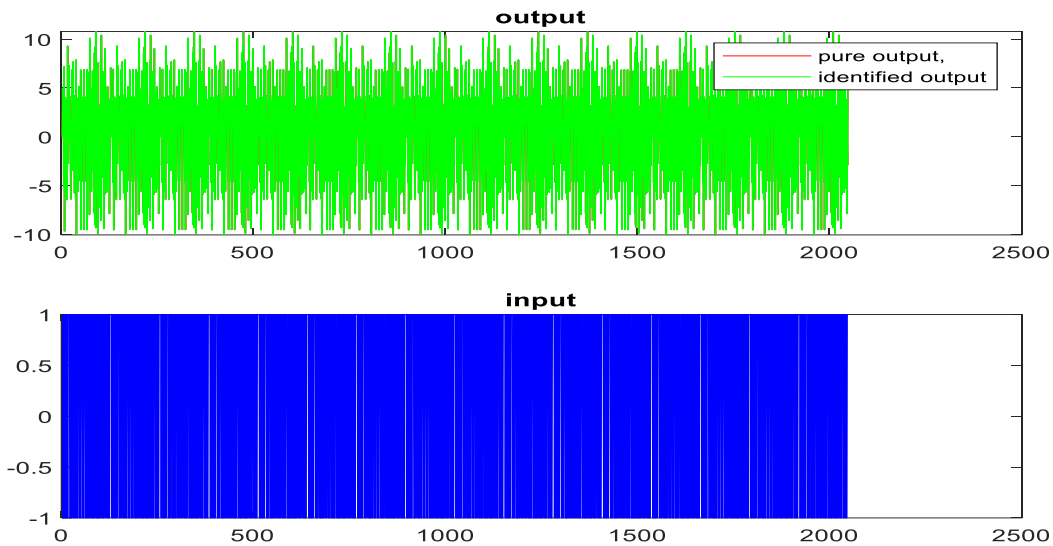

Figure 5.2 Ratdisk Simulated Output & Actual Output Compared – Test Function 1 With No Noise

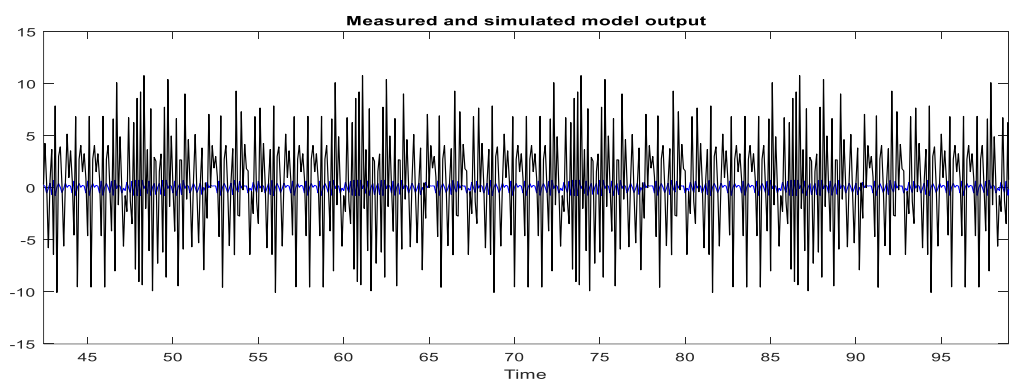

Figure 5.3 System_ID Simulated & Actual Output Compared – Test function 1 With No Noise.

5.2.2 Test function 1 With 8% Input Noise:

$$G\_ratdisk(z) = \frac{0.7013z+4.2431}{z^2+0.7508z+0.1263} \tag{5.4}$$

$$G\_SysID(z) = \frac{0.3449z}{z^2+0.6785z+0.1414} \tag{5.5}$$

Table 5.2: System_ID & Ratdisk Results of Test Function 1 With Input Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	-0.3393 + 0.1622i	-0.4965	-0.5
Eigenvalue 2	-0.3393 - 0.1622i	-0.2543	-0.25
Percentage fit to estimation	20.71	100	__
MSE	17.71	1.4029e-04	__

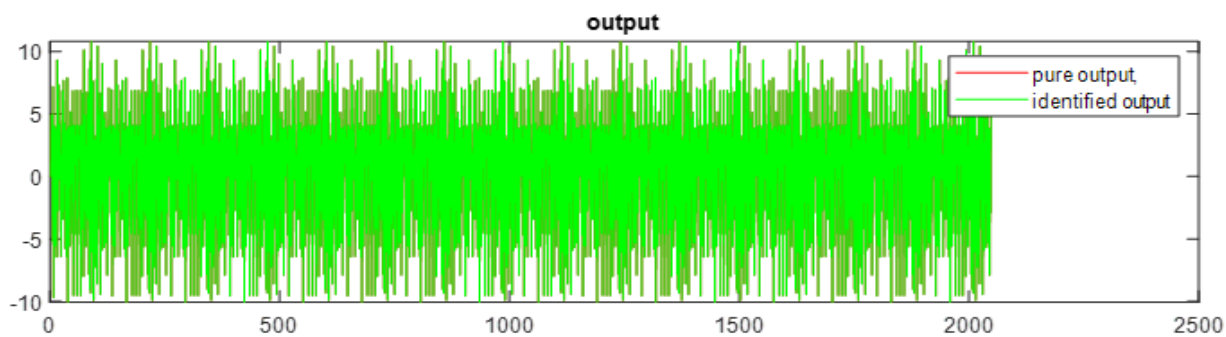

Figure 5.4 Ratdisk Output & Actual Output Compared – Test Function 1 With Input Noise

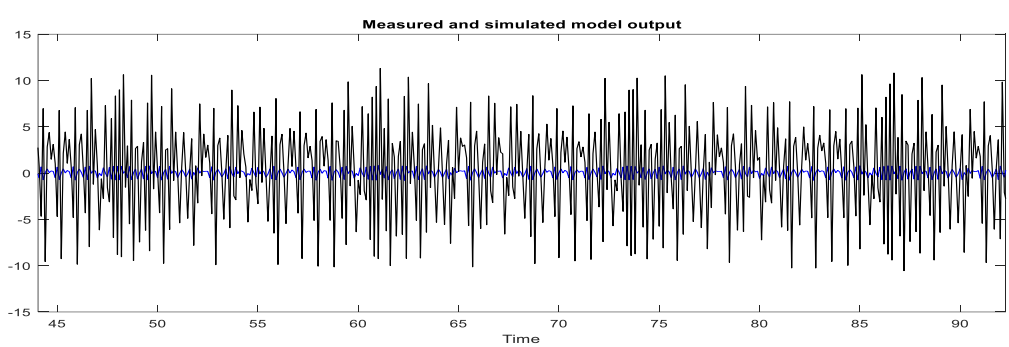

Figure 5.5 System_ID Simulated Output & Actual Output Compared – Test Function 1 With Input Noise

5.2.3 Test function 1 With 8% Output Noise:

$$G\_ratdisk(z) = \frac{0.6985z+4.2411}{z^2+0.7502z+0.1259} \tag{5.6}$$

$$G\_SysID(z) = \frac{0.3411z}{z^2+0.6784z+0.1415} \tag{5.7}$$

Table 5.3: System_ID & Ratdisk Results of Test Function 1 With Output Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	-0.3392 + 0.1626i	-0.4967	-0.5
Eigenvalue 2	-0.3392 - 0.1626i	-0.2535	-0.25
Percentage fit to estimation	20.7	100	__
MSE	17.64	1.7673e-04	__

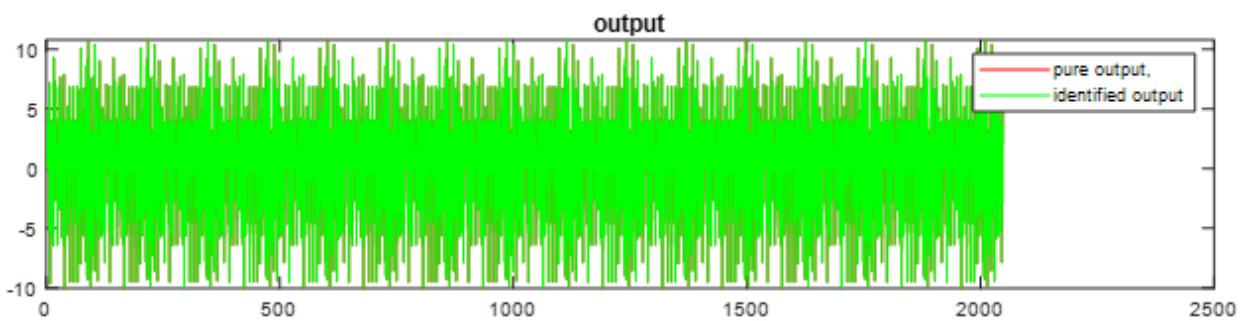

Figure 5.6 Ratdisk Output & Actual Output Compared – Test Function 1 With Output Noise

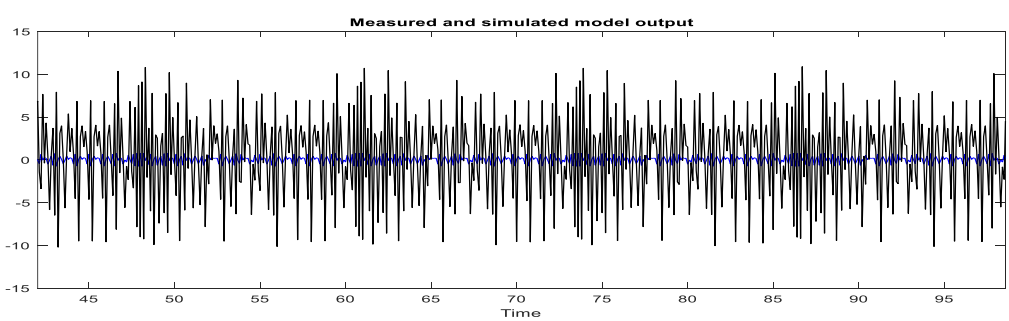

Figure 5.7 System_ID Simulated Output & Actual Output Compared – Test Function 1 With Output Noise

5.2.4 Test Function 1 With 8% Input Noise and 8% Output Noise:

$$G\_ratdisk(z) = \frac{0.6980z+4.2488}{z^2+0.7497z+0.1251} \tag{5.8}$$

$$G\_SysID(z) = \frac{0.3524z}{z^2+0.6753z+0.141} \tag{5.9}$$

Table 5.4: System_ID & Ratdisk Results of Test Function 2 With Input and Output Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	-0.3377 + 0.1643i	-0.4991	-0.5
Eigenvalue 2	-0.3377 - 0.1643i	-0.2506	-0.25
Percentage fit to estimation	20.56	100	__
MSE	17.92	3.6393e-04	__

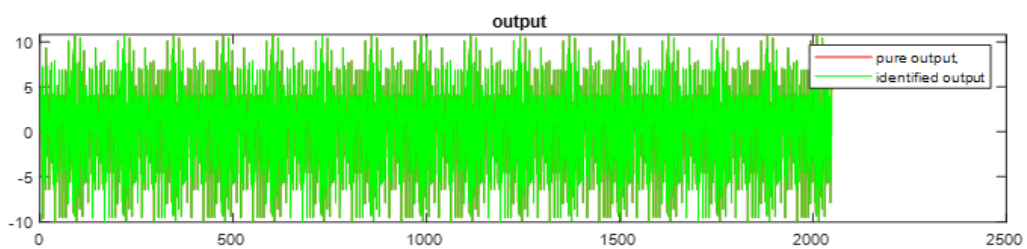

Figure Figure 5.8 Ratdisk Simulated & Actual Outputs Compared – Test Function 1 With Input and Output Noise

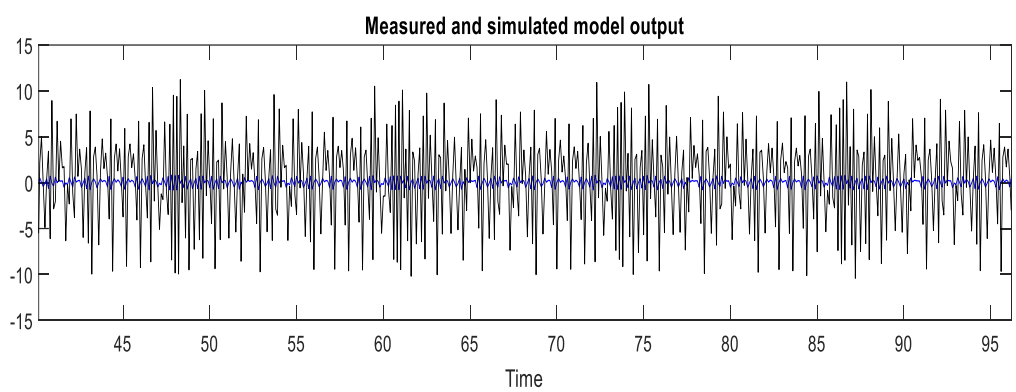

Figure 5.9 System_ID Simulated Output & Actual Output Compared – Test Function 1 With Input and Output Noise

5.3 Test Function 2

Recovering Wiener-Hammerstein Nonlinear State-Space Models Using Liner Algebra (P. Dreesen, M. Ishteva, and J. Schoukens, 2015)-System 2 [12]

$$G\_true(z) = \frac{8.4853z^2+25.4558z+16.9706}{z^3-1.5z^2+z-0.25} \tag{5.10}$$

5.3.1 Test Function 2 With No Input and Output Noise:

$$G\_ratdisk(z) = \frac{8.2861z^2+25.3759z+16.7805}{z^3-1.5027z^2+1.0034z-0.2520} \tag{5.11}$$

$$G\_SysID(z) = \frac{7.952z^2+21.38z}{z^3-1.915z^2+1.587z-0.5438} \tag{5.12}$$

Table 5.5: System_ID & Ratdisk Results of Test Function 2 With No Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	0.5717 + 0.6148i	0.4994 + 0.5007i	0.5000 + 0.5000i
Eigenvalue 2	0.5717 - 0.6148i	0.4994 - 0.5007i	0.5000 + 0.5000i
Eigenvalue 3	0.7716 + 0.0000i	0.5038 + 0.0000i	0.5000 + 0.0000i
Percentage fit to estimation	88.34	99.99	__
MSE	117.8	0.5110	__

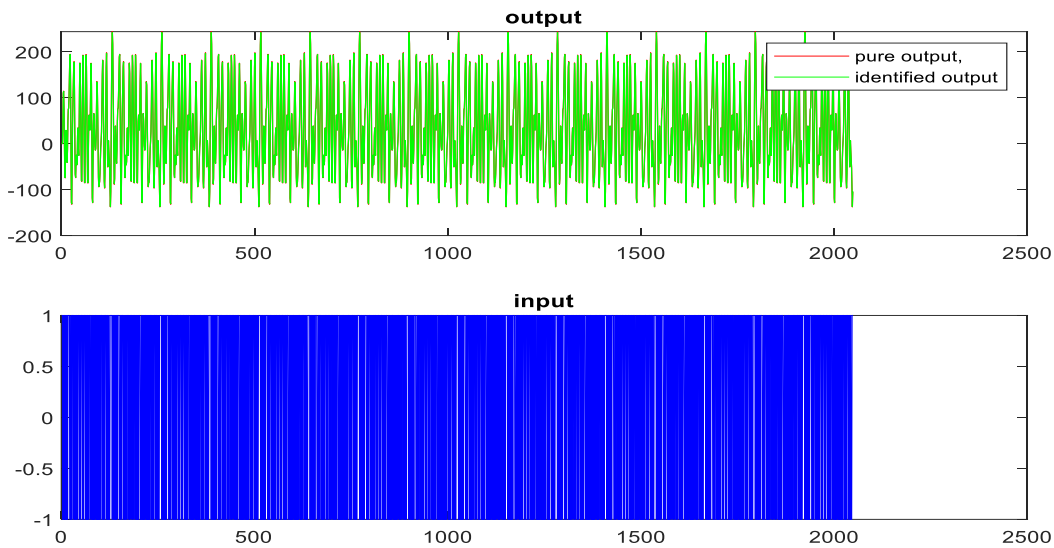

Figure 5.10 Ratdisk Simulated Output & Actual Output Compared – Test Function 2 With No Noise.

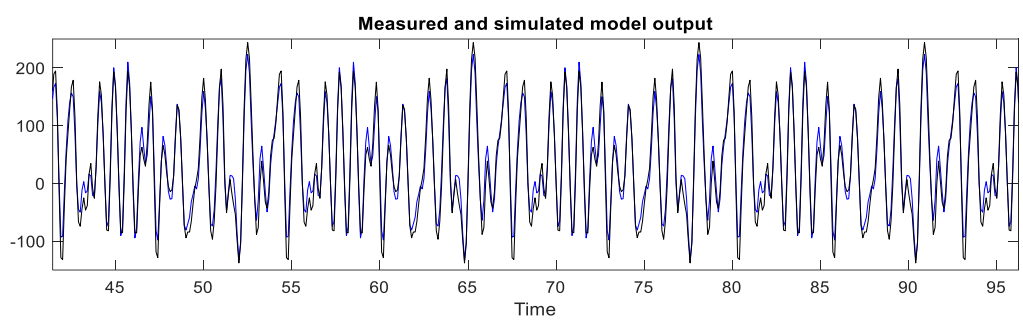

Figure 5.11 System_ID Simulated & Actual Output Compared – Test Function 2 With No Noise

5.3.2 Test Function 2 With 8% Input Noise:

$$G\_ratdisk(z) = \frac{8.2506z^2+25.25z+16.4721}{z^3-1.5161z^2+1.0193z-0.2579} \tag{5.13}$$

$$G\_SysID(z) = \frac{8.307z^2+21.59z}{z^3-1.912z^2+1.585z-0.5444} \tag{5.14}$$

Table 5.6: System_ID & Ratdisk Results of Test Function 2 With Input Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	0.5698 + 0.6165i	0.5035 + 0.5032i	0.5000 + 0.5000i
Eigenvalue 2	0.5698 - 0.6165i	0.5035 - 0.5032i	0.5000 + 0.5000i
Eigenvalue 3	0.7725+ 0.0000i	0.5091 + 0.0000i	0.5000 + 0.0000i
Percentage fit to estimation	88.59	99.99	__
MSE	193.4	1.0904	__

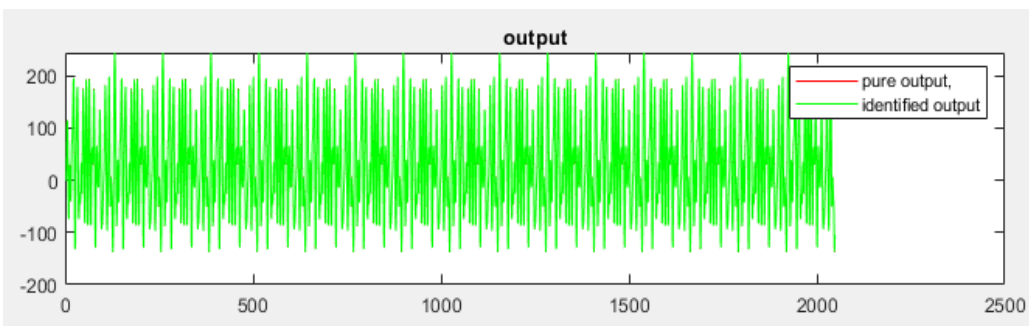

Figure 5.12 Ratdisk Output & Actual Output Compared – System 2 With Input Noise

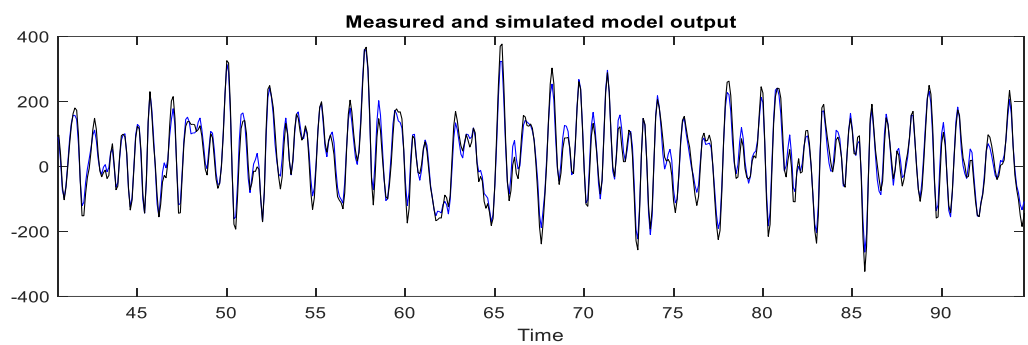

Figure 5.13 System_ID Simulated Output & Actual Output Compared – System 2 With Input Noise

5.3.3 Test Function 2 With 8% Output Noise:

$$G\_ratdisk(z) = \frac{8.2816z^2+25.3855z+16.7667}{z^3-1.5028z^2+1.0036z-0.2521} \tag{5.15}$$

$$G\_SysID(z) = \frac{7.956z^2+21.39z}{z^3-1.915z^2+1.586z-0.5437} \tag{5.16}$$

Table 5.7: System_ID & Ratdisk Results of Test Function 2 With Output Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	0.5709 + 0.6142i	0.4994 + 0.5008i	0.5000 + 0.5000i
Eigenvalue 2	0.5709 - 0.6142i	0.4994 - 0.5008i	0.5000 + 0.5000i
Eigenvalue 3	0.7732+ 0.0000i	0.5039 + 0.0000i	0.5000 + 0.0000i
Percentage fit to estimation	88.33	99.99	__
MSE	117.8	0.5109	__

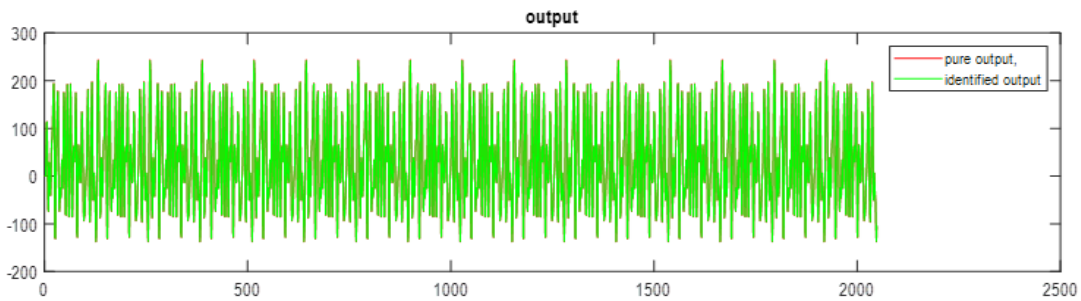

Figure 5.14 Ratdisk Output & Actual Output Compared – Test Function 2 With Output Noise

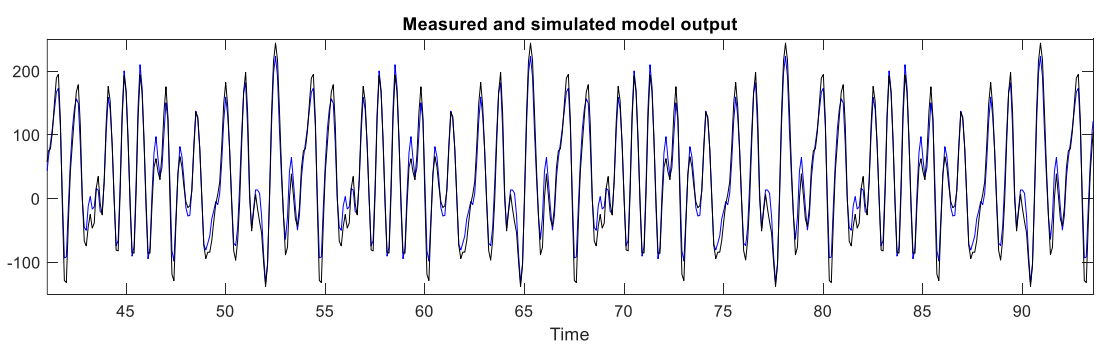

Figure 5.15 System_ID Simulated Output & Actual Output Compared – Test Function 2 With Output Noise

5.3.4 Test Function 2 With 8% Input Noise and 8% Output Noise:

$$G\_ratdisk(z) = \frac{8.2855z^2+25.3565z+16.7373}{z^3-1.5038z^2+1.0052z-0.2528} \tag{5.17}$$

$$G\_SysID(z) = \frac{7.94z^2+21.39z}{z^3-1.915z^2+1.587z-0.5437} \tag{5.18}$$

Table 5.8: System_ID & Ratdisk Results of Test Function 2 With Input and Output Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	0.5718 + 0.6147i	0.4996 + 0.5014i	0.5000 + 0.5000i
Eigenvalue 2	0.5718 - 0.6147i	0.4996 - 0.50148i	0.5000 + 0.5000i
Eigenvalue 3	0.7714+ 0.0000i	0.5046 + 0.0000i	0.5000 + 0.0000i
Percentage fit to estimation	88.32	99.99	__
MSE	118.32	0.5204	__

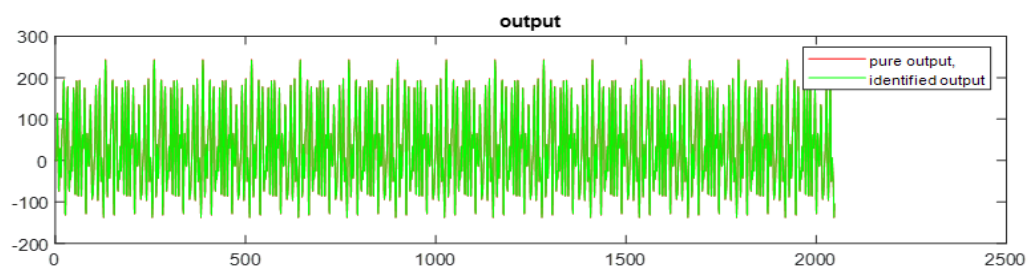

Figure 5.16 Ratdisk Output & Actual Output Compared – Test Function 2 With Input & Output Noise

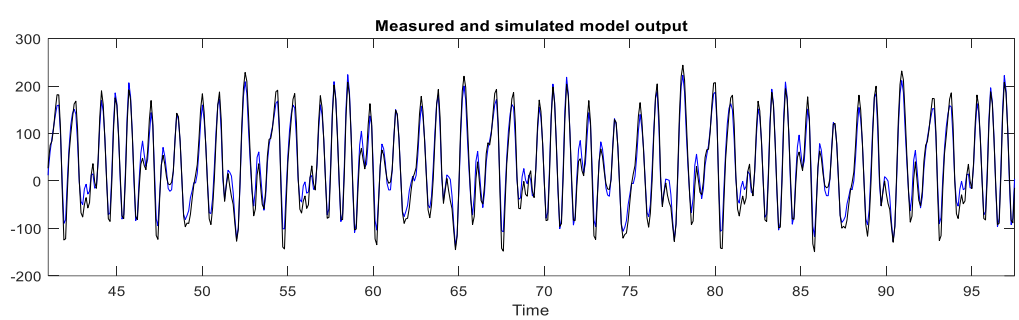

Figure 5.17 System_ID Simulated Output & Actual Output Compared – Test Function 2 With Input & Output Noise.

5.4 Test Function 3

Version 7.0 of the CONTSID Toolbox (A. Padilla, H. Garnier, and M. Gilson, 2015)-System 1 [5]

$$G\_true(z) = \frac{-0.7757z^3-1.367z^2+1.624z+0.6028}{z^4-1.198z^3+0.3238z^2-0.6383z-0.6015} \tag{5.19}$$

5.4.1 Test Function 3 With No Input & output noise:

$$G\_ratdisk(z) = \frac{-0.7708z^3-1.3649z^2+1.6257z+0.6071}{z^4-1.1969z^3+0.3223z^2-0.6377z+0.5992} \tag{5.20}$$

$$G\_SysID(z) = \frac{-0.7487z^3-1.405z^2+1.436z}{z^4-1.171z^3+0.5154z^2-0.7093z+0.5519} \tag{5.21}$$

Table 5.9: System_ID & Ratdisk Results of Test Function 3 With No Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	-0.2817+0.7716i	-0.3336 + 0.7439i	-0.3338 + 0.7444i
Eigenvalue 2	-0.2817-0.7716i	-0.3336 - 0.7439i	-0.3338 - 0.7444i
Eigenvalue 3	0.8672 + 0.2568i	0.9321 + 0.1806i	0.9328 + 0.1834i
Eigenvalue 4	0.8672 - 0.2568i	0.9321 - 0.1806i	0.9328 - 0.1834i
Percentage fit to estimation	86.91	99.96	__
MSE	0.1229	0.0028	__

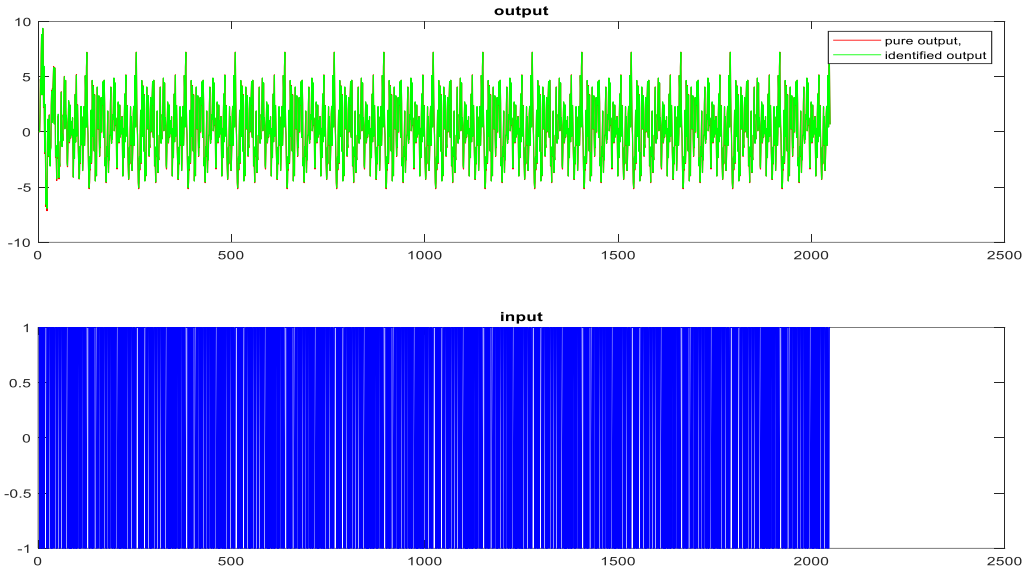

Figure 5.18 Ratdisk Simulated Output & Actual Output Compared – Test Function 3 With No Noise

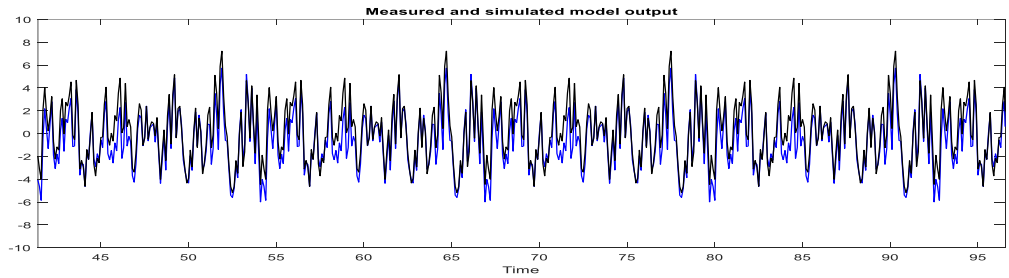

Figure 5.19 System_ID Simulated & Actual Output Compared – Test Function 3 With No Noise

5.4.2 Test Function 3 With 8% Input Noise:

$$G\_ratdisk(z) = \frac{-0.7707z^3-1.3645z^2+1.6271z+0.6071}{z^4-1.1974z^3+0.3226z^2-0.6381z+0.5995} \tag{5.22}$$

$$G\_SysID(z) = \frac{-0.749z^3-1.404z^2+1.439z}{z^4-1.173z^3+0.5169z^2-0.7105z+0.5525} \tag{5.23}$$

Table 5.10: System_ID & Ratdisk Results of Test Function 3 With Input Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	-0.2813+0.7718i	-0.3336 + 0.7440i	-0.3338 + 0.7444i
Eigenvalue 2	-0.2813-0.7718i	-0.3336 - 0.7440i	-0.3338 - 0.7444i
Eigenvalue 3	0.8678 + 0.2562i	0.9323 + 0.1805i	0.9328 + 0.1834i
Eigenvalue 4	0.8678 - 0.2562i	0.9323 - 0.1805i	0.9328 - 0.1834i
Percentage fit to estimation	86.97	99.96	__
MSE	0.1242	0.0030	__

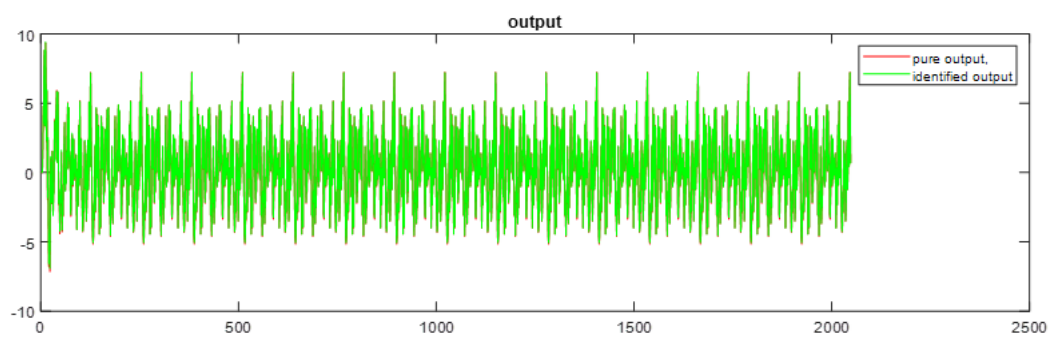

Figure 5.20 Ratdisk Output & Actual Output Compared – System 3 With Input Noise

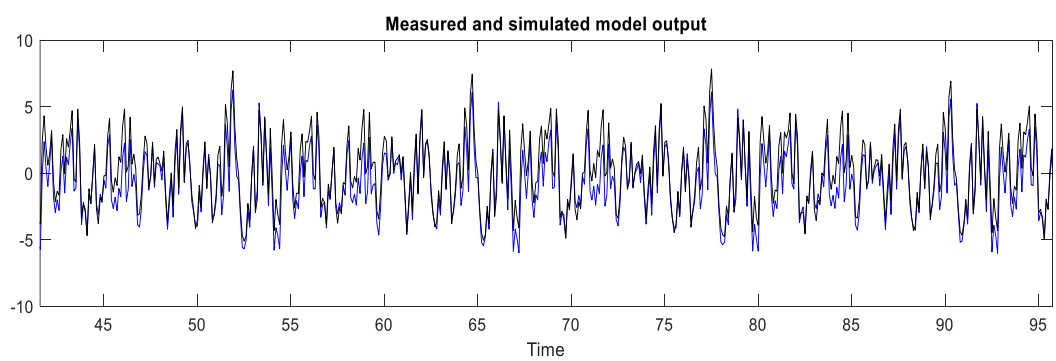

Figure 5.21 System_ID Simulated Output & Actual Output Compared – System 3 With Input Noise

5.4.3 Test Function 3 With 8% Output Noise:

$$G\_ratdisk(z) = \frac{-0.7777z^3-1.3649z^2+1.6179z+0.6199}{z^4-1.1935z^3+0.3174z^2-0.6373z+0.5997} \tag{5.24}$$

$$G\_SysID(z) = \frac{-0.7501z^3-1.413z^2+1.395z}{z^4-1.154z^3+0.507z^2-0.707z+0.5435} \tag{5.25}$$

Table 5.11: System_ID & Ratdisk Results of Test Function 3 With Output Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	-0.2829+0.7719i	-0.3351 + 0.7440i	-0.3338 + 0.7444i
Eigenvalue 2	-0.2829-0.7719i	-0.3351 - 0.7440i	-0.3338 - 0.7444i
Eigenvalue 3	0.8599 + 0.2545i	0.9319 + 0.1798i	0.9328 + 0.1834i
Eigenvalue 4	0.8678 - 0.2545i	0.9319 - 0.1798i	0.9328 - 0.1834i
Percentage fit to estimation	85.77	99.95	__
MSE	0.1451	0.0037	__

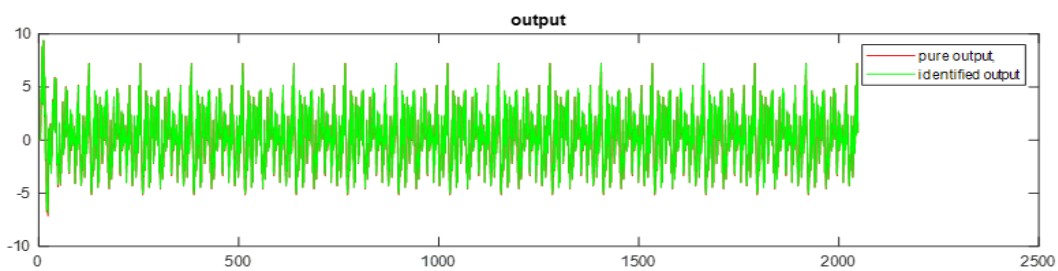

Figure 5.22 Ratdisk Output & Actual Output Compared – System 3 With Output No Noise

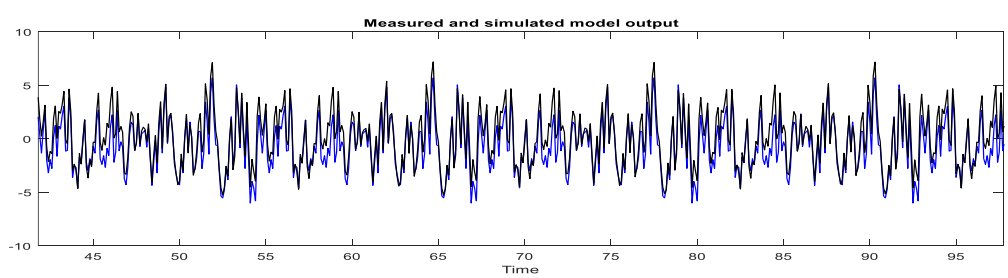

Figure 5.23 System_ID Simulated Output & Actual Output Compared – System 3 With Output Noise

5.4.4 Test Function 3 With 8% Input Noise and 8% Output Noise:

$$G\_ratdisk(z) = \frac{-0.7747z^3-1.3585z^2+1.6184z+0.6059}{z^4-1.1977z^3+0.3249z^2-0.6410z+0.6011} \tag{5.26}$$

$$G\_SysID(z) = \frac{-0.7462z^3-1.42z^2+1.398z}{z^4-1.153z^3+0.5061z^2-0.7054z+0.5432} \tag{5.27}$$

Table 5.12: System_ID & Ratdisk Results of Test Function 3 With Input & output Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	-0.2831+0.7714i	-0.3335 + 0.7451i	-0.3338 + 0.7444i
Eigenvalue 2	-0.2831-0.7714i	-0.3335 - 0.7451i	-0.3338 - 0.7444i
Eigenvalue 3	0.8596+ 0.2559i	0.9323 + 0.1813i	0.9328 + 0.1834i
Eigenvalue 4	0.8596 - 0.2559i	0.9319 - 0.1813i	0.9328 - 0.1834i
Percentage fit to estimation	85.87	99.98	__
MSE	0.1438	0.0015	__

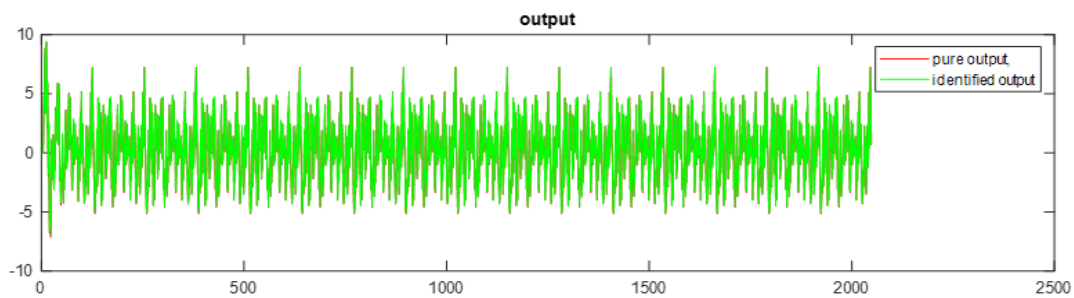

Figure 5.24 Ratdisk Output & Actual Output Compared – System 3 With Input & Output Noise

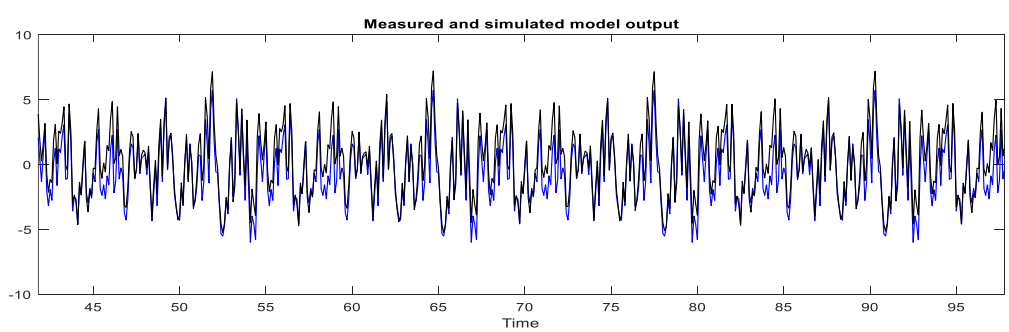

Figure 5.25 System_ID Simulated Output & Actual Output Compared – System 3 With Input & Output Noise

5.5 Test Function 4

Version 7.0 of the CONTSID Toolbox (A. Padilla, H. Garnier, and M. Gilson, 2015)-System 2 [5]

$$G\_true(z) = \frac{-0.7526z^3-1.329z^2+1.58z+0.5876}{z^4-1.194z^3+0.3258z^2-0.6458z-0.6059} \tag{5.28}$$

5.5.1 Test Function 4 With No Input & Output Noise:

$$G\_ratdisk(z) = \frac{-0.7478z^3-1.3271z^2+1.5811z+0.5914}{z^4-1.1927z^3+0.3244z^2-0.6453z+0.6034} \tag{5.29}$$

$$G\_SysID(z) = \frac{-0.7267z^3-1.368z^2+1.395z}{z^4-1.166z^3+0.5155z^2-0.7151z+0.5565} \tag{5.30}$$

Table 5.13: System_ID & Ratdisk Results of Test Function 4 With No Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	-0.2835+0.7745i	-0.3346 + 0.7471i	-0.3349 + 0.7476i
Eigenvalue 2	-0.2835-0.7745i	-0.3346 - 0.7471i	-0.3349 - 0.7476i
Eigenvalue 3	0.8665 + 0.2591i	0.9310 + 0.1834i	0.9319+ 0.1861i
Eigenvalue 4	0.8665 - 0.2591i	0.9310 - 0.1834i	0.9319 - 0.1861i
Percentage fit to estimation	86.82	99.96	__
MSE	0.1185	0.0025	__

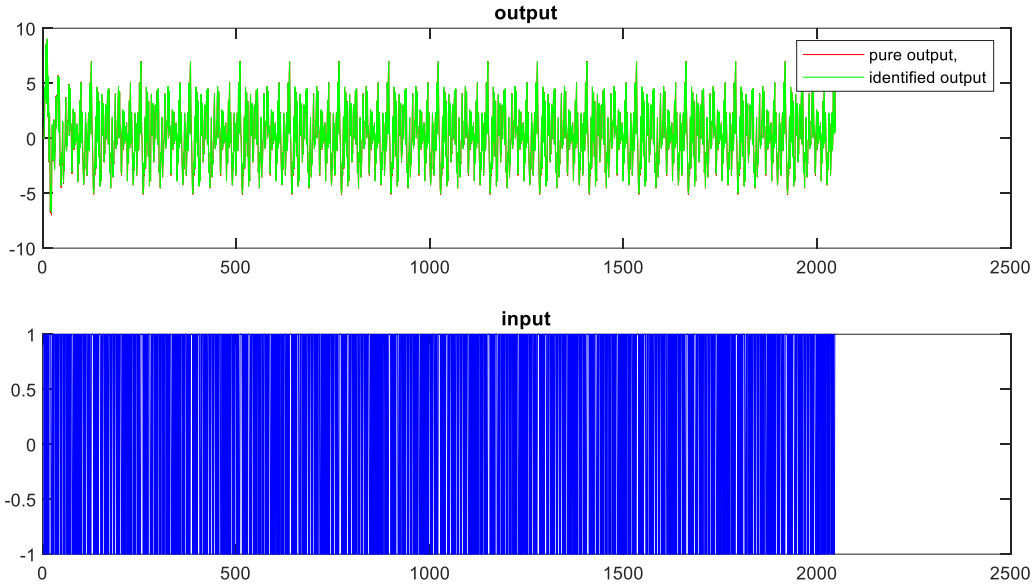

Figure 5.26 Ratdisk Simulated Output & Actual Output Compared – Test Function 4 With No Noise

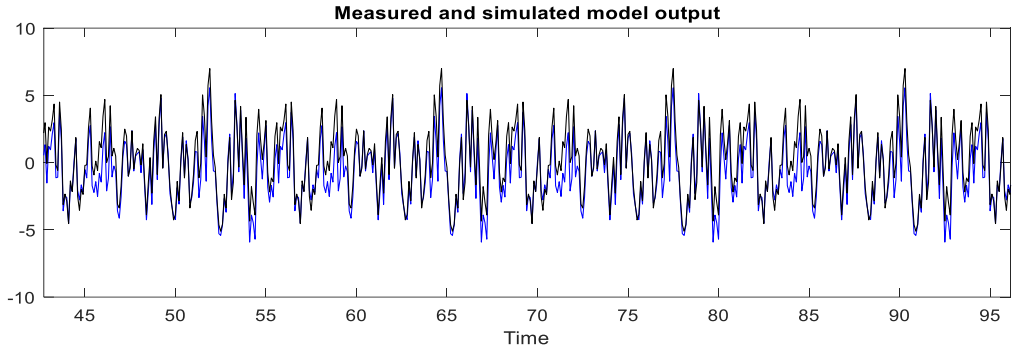

Figure 5.27 System_ID Simulated & Actual Output Compared – Test Function 4 With No Noise

5.5.2 Test Function 4 With 8% Input Noise:

$$G\_ratdisk(z) = \frac{-0.74758z^3-1.3271z^2+1.5818z+0.5911}{z^4-1.1927z^3+0.3244z^2-0.6455z+0.6032} \tag{5.31}$$

$$G\_SysID(z) = \frac{-0.7278z^3-1.366z^2+1.397z}{z^4-1.168z^3+0.5157z^2-0.7158z+0.5573} \tag{5.32}$$

Table 5.14: System_ID & Ratdisk Results of Test Function 4 With Input Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	-0.2835+0.7745i	-0.3346 + 0.7472i	-0.3349 + 0.7476i
Eigenvalue 2	-0.2835-0.7745i	-0.3346 - 0.7472i	-0.3349 - 0.7476i
Eigenvalue 3	0.8665 + 0.2591i	0.9309 + 0.1829i	0.9319+ 0.1861i
Eigenvalue 4	0.8665 - 0.2591i	0.9309 - 0.1829i	0.9319 - 0.1861i
Percentage fit to estimation	86.87	99.95	__
MSE	0.1195	0.0033	__

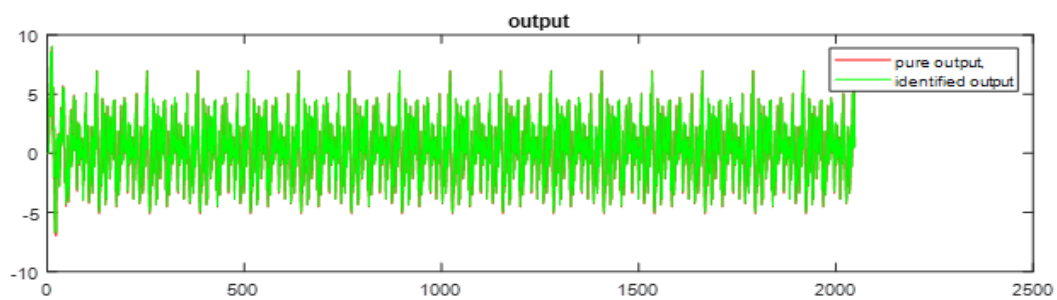

Figure 5.28 Ratdisk Simulated output & Actual Output Compared – Test Function 4 With Input Noise

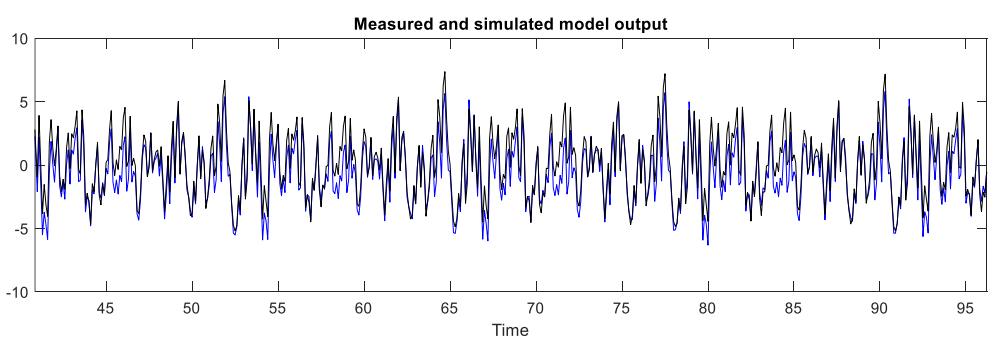

Figure 5.29 System_ID Simulated & Actual Output Compared – Test Function 4 With Input Noise

5.5.3 **Test Function 4 With 8% Output Noise:**

$$G\_ratdisk(z) = \frac{-0.7516z^3-1.3198z^2+1.5828z+0.5966}{z^4-1.1925z^3+0.3251z^2-0.6472z+0.6042} \tag{5.33}$$

$$G\_SysID(z) = \frac{-0.7318z^3-1.378z^2+1.358z}{z^4-1.149z^3+0.5076z^2-0.7111z+0.546} \tag{5.34}$$

Table 5.15: System_ID & Ratdisk Results of Test Function 4 With Output Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	-0.2840+0.7741i	-0.3347 + 0.7478i	-0.3349 + 0.7476i
Eigenvalue 2	-0.2840-0.7741i	-0.3347 - 0.7478i	-0.3349 - 0.7476i
Eigenvalue 3	0.8585 + 0.2570i	0.9309 + 0.1830i	0.9319+ 0.1861i
Eigenvalue 4	0.8585 - 0.2570i	0.9309 - 0.1830i	0.9319 - 0.1861i
Percentage fit to estimation	85.6	99.96	__
MSE	0.1412	0.0031	__

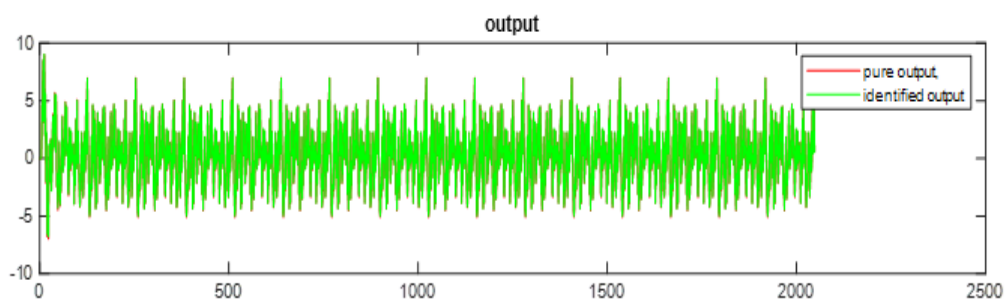

Figure 5.30 Ratdisk Simulated output & Actual Output Compared – Test Function 4 With Output Noise.

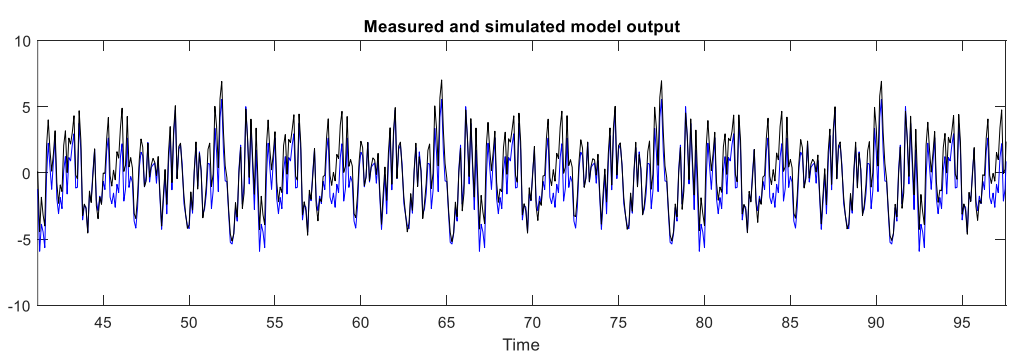

Figure 5.31 System_ID Simulated & Actual Output Compared – Test Function 4 With Output Noise.

5.5.4 Test Function 4 With 8% Input and 8% Output Noise:

$$G\_ratdisk(z) = \frac{-0.7494z^3-1.3286z^2+1.5714z+0.6065}{z^4-1.1892z^3+0.3203z^2-0.6447z+0.6034} \tag{5.35}$$

$$G\_SysID(z) = \frac{-0.7284z^3-1.383z^2+1.365z}{z^4-1.15z^3+0.506z^2-0.7117z+0.5477} \tag{5.36}$$

Table 5.16: System_ID & Ratdisk Results of Test Function 4 With Input & output Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	-0.2840+0.7741i	-0.3359 + 0.7472i	-0.3349 + 0.7476i
Eigenvalue 2	-0.2840-0.7741i	-0.3359 - 0.7472i	-0.3349 - 0.7476i
Eigenvalue 3	0.8585 + 0.2570i	0.9304 + 0.1831i	0.9319+ 0.1861i
Eigenvalue 4	0.8585 - 0.2570i	0.9304 - 0.1831i	0.9319 - 0.1861i
Percentage fit to estimation	85.74	99.95	__
MSE	0.1419	0.0033	__

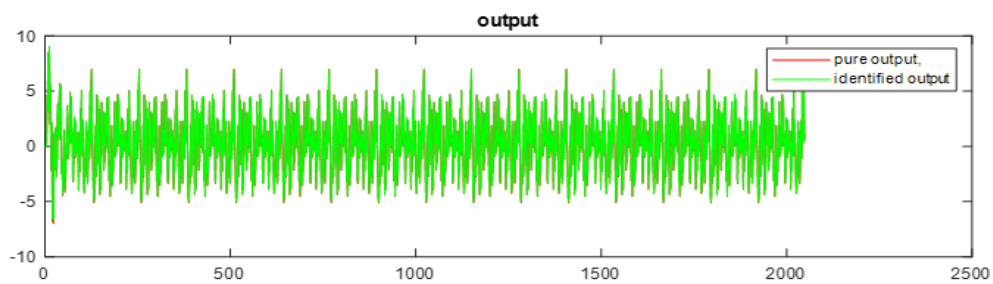

Figure 5.32 Ratdisk Simulated Output & Actual Output Compared – Test Function 4 With Input & Output Noise.

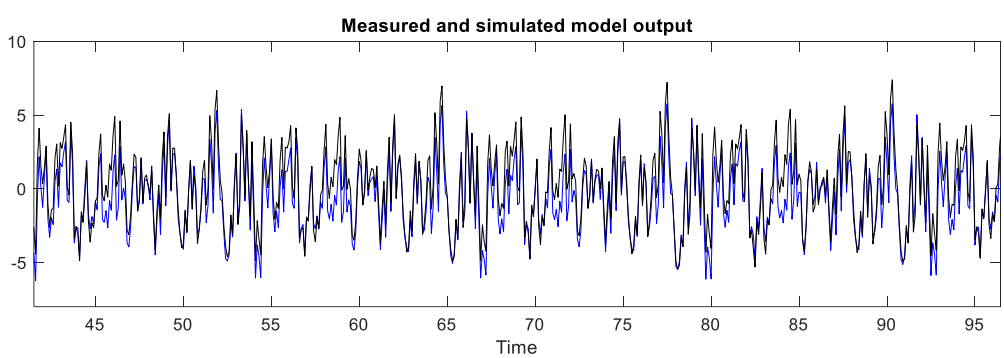

Figure 5.33 System_ID Simulated & Actual Output Compared – Test Function 4 with Input & Output Noise.

5.6 Test Function 5

Identification of block-oriented systems with rate saturation nonlinearity (A.Y.K. Yong, A.H. Tan, and C.L. Cham, 2015)-System 1 [13]

$$G\_true(z) = \frac{z+0.5}{z^2-1.5z+0.7} \tag{5.37}$$

5.6.1 Test Function 5 With No Input and Output Noise:

$$G\_ratdisk(z) = \frac{0.995z+0.5005}{z^2-1.4975z+0.6978} \tag{5.38}$$

$$G\_SysID(z) = \frac{0.9533z}{z^2-1.63z+0.832} \tag{5.39}$$

Table 5.17: System_ID & Ratdisk Results of Test Function 5 With No Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	0.8150 + 0.4096i	0.7488+0.3703i	0.7500+0.3708i
Eigenvalue 2	0.8150 - 0.4096i	0.7588-0.3703i	0.7500-0.3708i
Percentage fit to estimation	88.8	99.99	__
MSE	0.1789	0.0013	__

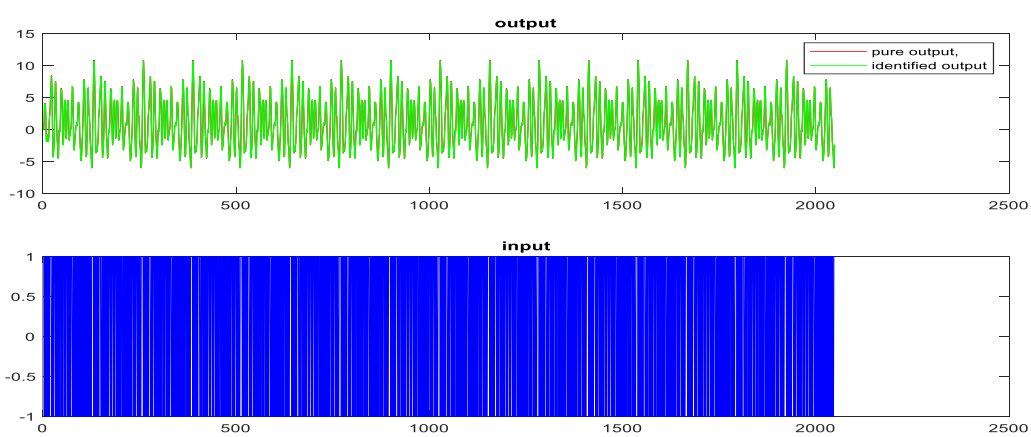

Figure 5.34 Ratdisk Simulated Output & Actual Output Compared – Test Function 5 With No Noise

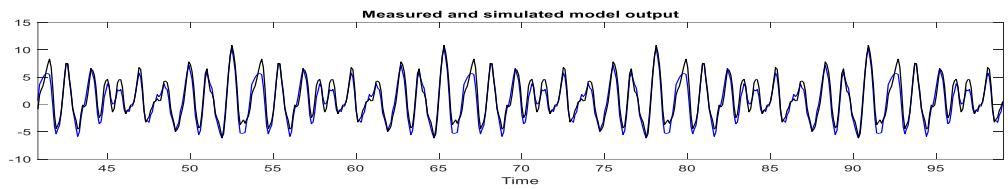

Figure 5.35 Ratdisk Simulated Output & Actual Output Compared – Test Function 5 With No Input & Output Noise.

5.6.2 Test Function 5 With 8% Input Noise:

$$G\_ratdisk(z) = \frac{0.995z+0.5005}{z^2-1.4975z+0.6978} \tag{5.40}$$

$$G\_SysID(z) = \frac{0.9531z}{z^2-1.63z+0.8323} \tag{5.41}$$

Table 5.18: System_ID & Ratdisk Results of Test Function 5 With Input Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	0.8150 + 0.4100i	0.7487+0.3702i	0.7500+0.3708i
Eigenvalue 2	0.8150 - 0.4100i	0.7487-0.3702i	0.7500-0.3708i
Percentage fit to estimation	88.78	99.99	__
MSE	0.1801	0.0015	__

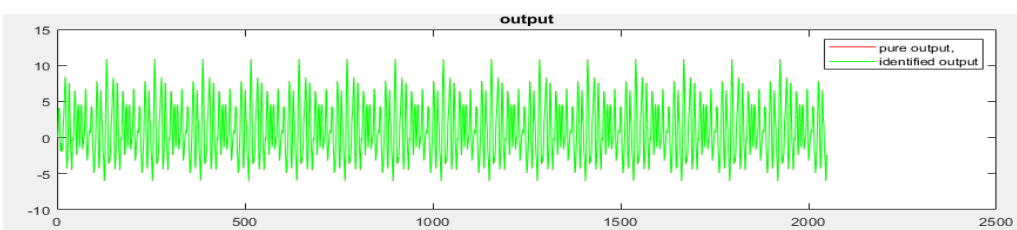

Figure 5.36 Ratdisk Simulated Output & Actual Output Compared – Test Function 5 With Input Noise

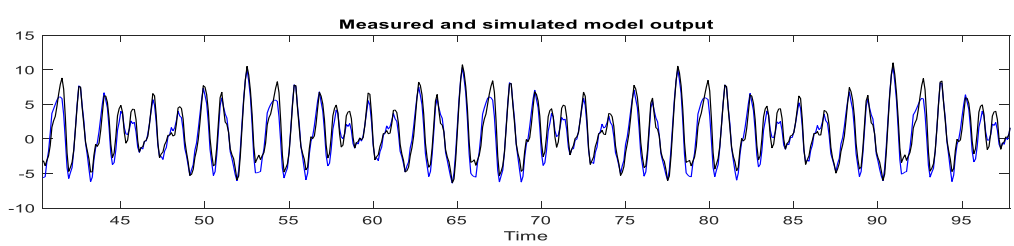

Figure 5.37 System_ID Simulated & Actual Output Compared – Test Function 5 With Input Noise

5.6.3 Test Function 5 With 8% Output Noise:

$$G\_ratdisk(z) = \frac{0.9938z+0.4986}{z^2-1.4980z+0.6981} \tag{5.42}$$

$$G\_SysID(z) = \frac{0.9494z}{z^2-1.626z+0.8282} \tag{5.43}$$

Table 5.19: System_ID & Ratdisk Results of Test Function 5 With Output Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	0.8130 + 0.4089i	0.7490+0.3703i	0.7500+0.3708i
Eigenvalue 2	0.8130 - 0.4089i	0.7490-0.3703i	0.7500-0.3708i
Percentage fit to estimation	88.87	99.99	__
MSE	0.2101	0.0016	__

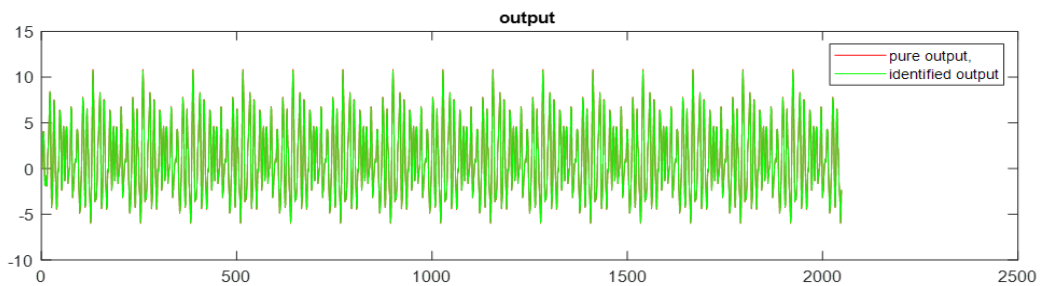

Figure 5.38 Ratdisk Simulated Output & Actual Output Compared – Test Function 5 With Output Noise

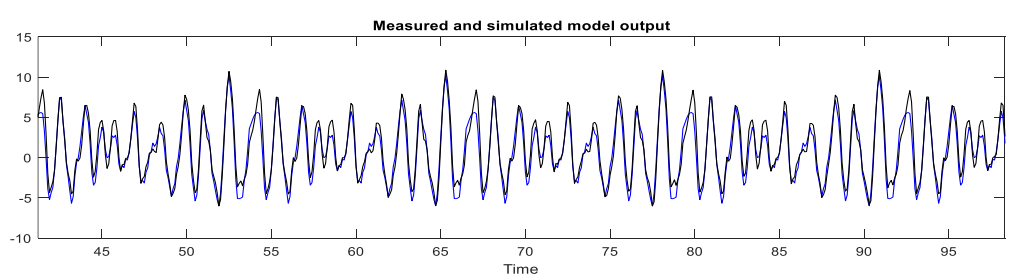

Figure 5.39 System_ID Simulated & Actual Output Compared – Test Function 5 With Output Noise

5.6.4 Test Function 5 With 8% Input and 8% Output Noise:

$$G\_ratdisk(z) = \frac{0.9994z+0.5019}{z^2-1.4972z+0.6976} \tag{5.44}$$

$$G\_SysID(z) = \frac{0.9549z}{z^2-1.626z+0.8286} \tag{5.45}$$

Table 5.20: System_ID & Ratdisk Results of Test Function 5 With Input & output Noise

	System ID tool	Ratdisk Method	True System
Eigenvalue 1	0.8130 + 0.4094i	0.7490+0.3703i	0.7500+0.3708i
Eigenvalue 2	0.8130 - 0.4094i	0.7490-0.3703i	0.7500-0.3708i
Percentage fit to estimation	87.96	100	__
MSE	0.2067	7.2046e-04	__

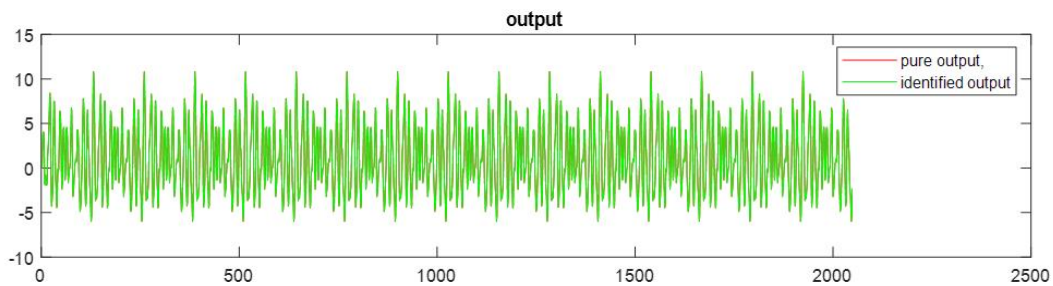

Figure 5.40 Ratdisk Simulated Output & Actual Output Compared – Test Function 5 With Input & Output Noise

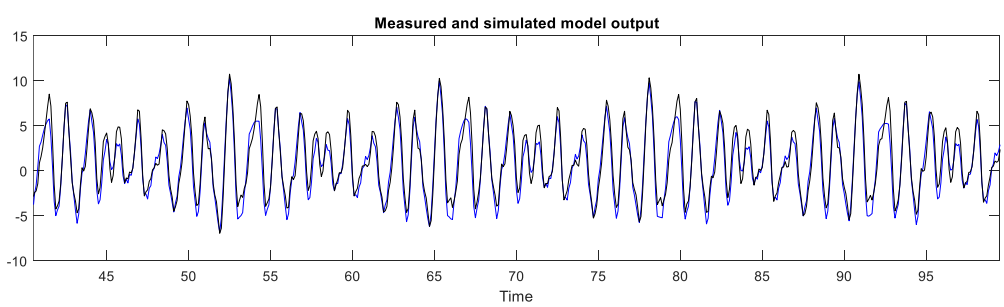

Figure 5.41 System_ID Simulated & Actual Output Compared – Test Function 5 With Input & Output Noise

Chapter 6: Conclusion & Future Work

This thesis research presents a technique for identification of system models via Empirical Transfer Function Estimates (ETFE). The analyses of the numerical results in chapter five shows that this technique gives better model estimation in all of our test functions than the MATLAB® system identification toolbox. It is seen from the tables of results that the percentage fit to estimation is over 99.5% for each of the functions tested using this technique whereas the corresponding percentage fit using system identification toolbox is below 88%. A comparison of the mean square errors (MSE) for each of the two techniques also shows that the errors associated with the system identification toolbox method is bigger than errors about the *ratdisk* method. The eigenvalues estimation via this proposed system identification technique is very close and almost the same in value as the true system's poles in each of the test functions in chapter five.

Analysis of model estimation from higher order non-minimum phase functions using this technique shows that it produced bad results. A further research that would focus on estimation of non-minimum phase systems of higher order is recommendable.

MATLAB® version 9.6.0.1072779 (R2019a) was used for the programming and analysis, and the programs executed on an Intel® Core ™ i7 CPU at 1.80Hz, 4 Core(s) laptop computer with 16.0 GB of memory.

References

[1] *System Identification, Theory For The User*, Second edition, Lennart Ljung, 1999.

[2] *System Identification, An Introduction*, Karel J. Keesman, 2011.

[3] National Instruments, *"Selecting a Model Structure in the System Identification Process"* Visited at http://www.ni.com/product-documentation/4028/en/ on September 25, 2019.

[4] System Identification with MATLAB®. Linear Models Identification: ARMAX, State Space and Transfer Function, A. Smith, 2018.

[5] A. Padilla, H. Garnier, M. Gilson, Universite de Lorraine, Nancy, France, "Version 7.0 of the CONTSID toolbox", 2015.

[6] Pedro Gonnet, Ricardo Pachon, Lloyd N. Trefethen, *Robust Rational Interpolation and Least-Squares*, Electronic Transactions on Numerical Analysis Volume 38, pp. 146 – 167, 2011.

[7] Least Squares Fitting with Applications, Per Christian Hansen, Victor Pereyra and Godela Scherer, 2013

[8] Shaldon M. Ross, *Probability and Statistics for Engineers and Scientists*, Fifth Edition, pp. 357 – 420.

[9] System Identification using Nuclear Norm and Tabu Search Optimization by Asif Ahmed, thesis work submitted to the College of Science and Engineering, Idaho State University, 2016.

[10] *Adaptive Control System* second edition by Karl J Astrom and Bjorn Wittenmark, 2008.

[11] MathWorks, "*Validating Models After Estimation,"* Visited at https://www.mathworks.com/help/ident/ug/validating-models-after-estimation.html on September 25, 2019.

[12] Philippe Dreesen, Mariya Ishteva, Johan Schoukens, Vrije Univerteit Brussel, Dept. ELEC, Brussels, Belgium, "*Recovering Wiener-Hammerstein Nonlinear State-Space Models Using Liner Algebra*", 2015.

[13] Alex Y. K. Yong, Ai Hui Tan, Chin Leei Cham, Multimedia University, Cyberjaya, Malaysia, "*Identification of Block-oriented Systems with Rate Saturation Nonlinearity*", 2015.

Appendix

Appendix A

Main Program Code

```
% File Name: main.m
% Date created : 09/05/2019
% Author : Anene Vitus Omeje, greatly assisted by Dr. Ken Bosworth

% script for ratdisk arx identification:
clc;
close all;
%clear all;
disp('various results based on switching prob. in rpbs')
resp = 'y';

% define a stable arx model
% The characteristic equation's coeff. are [ 1  a]
    a = [-1.5 1 -0.25];
    b =  [8.4853 25.4558 16.9706];


while resp =='y'
   disp('You can either create a new control sequence, or load one that has been saved')
   resp = input('Do you wish to create a new control sequence y/n: ','s');

   if resp == 'y'

    p = input('pls. enter switching prob. p in (0,1), p = ');
    N = input('pls. enter N(as power of 2): ');
    M = input('pls. enter M(as power of 2, less than N): ');
    n = input('pls. enter n(as numerator order of the ARX Model): ');
    m = input('pls. enter m(as denominator order of the ARX Model): ');
    u  = prbs(N,p);

   else
    fname = input('please enter the name of the file containing u: ','s');
    load(fname,'-mat');
    N = length(u);
    M = input('pls. enter M(as power of 2, less than N): ');
    n = input('pls. enter n(as numerator order of the ARX Model): ');
    m = input('pls. enter m(as denominator order of the ARX Model): ');

   end

    c=lower(input('Select option from -> (a) No noise (b) Input noise (c) output noise (d) both : ','s'));
    %
        uc = ones(N,M);
        uc = uc.*u;
    if c=='a' || c=='c'
        uc = reshape(uc,N*M,1);% this gives us M repetitions of the N-periodic signal
        ucT = uc;
    elseif c=='b' || c=='d'
        sigmai = input('enter std. dev. for input noise? : ');
        ucT = reshape(uc,N*M,1);
        uc = ucT+sigmai*randn(size(ucT));
    else
    end
```

```
    ucfft = fft(uc);
    figure(1)
    plot(1:N*M,abs(ucfft))
    title('fft of input')
    if c=='a' || c=='b'
                yc = arx_sim(a,b,N*M,uc);
    elseif c=='c' || c=='d'
         sigma2 = input('enter std. dev. for output noise? : ');
         yc = arx_sim(a,b,N*M,uc)+sigma2*randn(size(uc));
    else
    end
    figure(2)
    plot(1:N*M,yc)
    title('output from given model (noise: may be)')
    ycfft=fft(yc);
    etfN_M = ycfft./ucfft;
    figure(3)
    plot(1:N*M,abs(etfN_M),'r*')
    title('fft of output');
% now fetch the eft values at the N roots of unity, by taking every Mth
% data point in the N*M element etfN_M
    etf_N=zeros(N,1);
    for k = 1:N
        jj=(k-1)*M+1;
        etf_N(k) = etfN_M(jj);
    end
    % do a safety check: we may have accidentally missed one of the
    % N frequencies in our u : this would show up as an Inf value in
    % abs(etf_N)
    if max(abs(etf_N)) == Inf
        disp('bad input signal, missing at least one of our frequencies')
        continue % this will throw us back to the top of the while loop
    end


    figure(4)
    plot(1:N,abs(etf_N),'r*')
    title('magn. of etf');
    [r,num,den,mu,nu,poles] = ratdisk(etf_N,n,m,N-1)

    % so the numerator and denominator match our convention (highest
    % power to lowest power), we will convert them:
    % Note: num and den are column vectors. We want row vectors, so
    % transpose and then fliplr and normalize our denominator
    bratdisk = fliplr(num');
    aratdisk = fliplr(den');
    bratdisk = 1/aratdisk(1)*bratdisk
    aratdisk = 1/aratdisk(1)*aratdisk
    disp('poles of estimated transfer function:')
    roots(aratdisk)
    disp('zeros of estimated transfer function:')
    roots(bratdisk)
    disp('hit enter to continue')
    pause()
    ycT = arx_sim(a,b,N*M,ucT);
    ycP = arx_sim(aratdisk(2:length(aratdisk)),bratdisk,N*M,ucT);
    MSE = sum((ycT-ycP).^2)/(N*M)
```

```
    ynorm = sum((ycT).^2/(N*M));
    Perc_Predictn = 1 -(MSE/ynorm)
    figure(5)
    subplot(2,1,1);
    plot([1:length(ucT)],ycT,'r',[1:length(ucT)],ycP,'g')
    title('output');
    legend('pure output,','identified output');
    subplot(2,1,2);
    plot([1:length(ucT)],ucT,'b')
    title('input');

%Save input that gives better result
 resp = input('Do you want to save this input control sequence u ? y/n: ','s');
 if resp == 'y'
     disp('the filename should have a .mat extension')
     filename= input('Pls. enter the name of the file to save to : ','s');

     save(filename,'-mat','u')
 end
   resp = input('try again? y/n: ','s');
end
```

```
various results based on switching prob. in rpbs
You can either create a new control sequence, or load one that has been saved


Error using input
Cannot call INPUT from EVALC.

Error in main (line 20)
   resp = input('Do you wish to create a new control sequence y/n: ','s');
```

Published with MATLAB® R2019a

Appendix B

Ratdisk Code

```
% File Name: ratdisk.m
% Date created : January, 2011
% Author : P. Gonnet, R. Pachon,and L. N. Trefethen
% Reference: "Robust Rational Interpolation and Least-Squares", Electronic Transactions on
%Numerical Analysis Volume 38, pp. 146 ? 167, 2011.

function [r,a,b,mu,nu,poles,residues] = ratdisk(f,m,n,N,tol)
%  Input: Function f or vector of data at zj = exp(2i*pi*(0:N)/(N+1))
%          for some N>=m+n.  If N>>m+n, it is best to choose N odd.
%         Maximal numerator, denominator degrees m,n.
%         An optional 5th argument specifies relative tolerance tol.
%           If omitted, tsol = 1e-14.  Use tol=0 to turn off robustness.
% Output: function handle r of exact type (mu,nu) approximant to f
%         with coeff vectors a and b and optional poles and residues.
% P. Gonnet, R. Pachon, L. N. Trefethen, January 2011

if nargin<4, if isfloat(f), N=length(f)-1;
  else N=m+n; end, end                        % do interpolation if no N given
N1 = N+1;                                     % no. of roots of unity
if nargin<5, tol = 1e-14; end                 % default rel tolerance 1e-14
if isfloat(f), fj = f(:);                     % allow for either function
else fj = f(exp(2i*pi*(0:N)'/(N1))); end      %   handle or data vector
ts = tol*norm(fj,inf);                        % absolute tolerance
M = floor(N/2);                               % no. of pts in upper half-plane
f1 = fj(2:M+1); f2 = fj(N+2-M:N1);            % fj in upper, lower half-plane
realf = norm(f1(M:-1:1)-conj(f2),inf)<ts;     % true if fj is real symmetric
oddN = mod(N,2)==1;                           % true if N is odd
evenf = oddN & norm(f1-f2,inf)<ts;            % true if fj is even
oddf  = oddN & norm(f1+f2,inf)<ts;            % true if fj is odd
row = conj(fft(conj(fj)))/N1;                 % 1st row of Toeplitz matrix
col = fft(fj)/N1; col(1) = row(1);            % 1st column of Toeplitz matrix
if realf, row = real(row);                    % discard negligible imag parts
  col = real(col); end
d = xor(evenf,mod(m,2)==1);                   % either 0 or 1
while true                                    % main stabilization loop
  Z = toeplitz(col,row(1:n+1));               % Toeplitz matrix
  if ~oddf & ~evenf                           % fj is neither even nor odd
    [U,S,V] = svd(Z(m+2:N1,:),0);             % singular value decomposition
    b = V(:,n+1);                             % coeffs of q
  else                                        % fj is even or odd
    [U,S,V] = svd(Z(m+2+d:2:N1,1:2:n+1),0);   % special treatment for symmetry
    b = zeros(n+1,1); b(1:2:end) = V(:,end);  % coeffs of q
  end
  if N > m+n && n>0, ssv = S(end,end);        % smallest singular value
  else ssv = 0; end                           % or 0 in case of interpolation
  qj = ifft(b,N1); qj = qj(:);                % values of q at zj
  ah = fft(qj.*fj);                           % coeffs of p-hat
  a = ah(1:m+1);                              % coeffs of p
  if realf a = real(a); end                   % discard imag. rounding errors
  if evenf a(2:2:end) = 0; end                % enforce even symmetry of coeffs
  if  oddf a(1:2:end) = 0; end                % enforce odd symmetry of coeffs
  if tol>0                                    % tol=0 means no stabilization
    ns = n;                                   % no. of singular values
    if oddf|evenf, ns = floor(n/2); end
    s = diag(S(1:ns,1:ns));                   % extract singular values
    nz = sum(s-ssv<=ts);                      % no. of sing. values to discard
    if nz == 0, break                         % if no discards, we are done
    else n=n-nz; end
  else break                                  % no iteration if tol=0.
  end
end                                           % end of main loop
```

```
  end
end                                          % end of main loop
nna = abs(a)>ts; nnb = abs(b)>tol;           % nonnegligible a and b coeffs
kk = 1:min(m+1,n+1);                         % indices a and b have in common
a = a(1:find(nna,1,'last'));                 % discard trailing zeros of a
b = b(1:find(nnb,1,'last'));                 % discard trailing zeros of b
if length(a)==0 a=0; b=1; end                % special case of zero function
mu = length(a)-1; nu = length(b)-1;          % exact numer, denom degrees
r = @(z) polyval(a(end:-1:1),z)...           % function handle for r
       ./polyval(b(end:-1:1),z);
if nargout>5                                 % only compute poles if necessary
  poles = roots(b(end:-1:1));                % poles
  t = max(tol,1e-7);                         % perturbation for residue estimate
  residues = t*(r(poles+t)-r(poles-t))/2;    % estimate of residues
end
```

```
Not enough input arguments.

Error in ratdisk (line 17)
if nargin<4, if isfloat(f), N=length(f)-1;
```

Published with MATLAB® R2019a

Appendix C

ARX Simulation

```
% File Name: arx_sim.m
% Date created : 12/14/2018
% Author : Anene Vitus Omeje

function y = arx_sim(a,b,N,u,y0)
% arx_sim runs an arx model of the form:
% y(k) + a(1)y(k-1) + ... + a(n)y(k-n) = b(1)u(k-1) + ... + b(m)u(k-m)
% for N time steps k = 1,...,N with exogenous input u and optional IC y0
% usage:
% y = arx_sim(a,b,N,u,y0)
% y will be an N x 1 vector of system outputs
% or:
% y = arx_sim(a,b,N,u) in which case y0 = zeros(N,1) is assumed.
%
% Notes:
%  0) the coeff. vector a is NOT the coeff.'s of the char. poly,
%     rather [ 1 , a]  is.  I.e., we normalize the leading coeff.
%     of y(k) in the arx model to be 1.
%  1) it must be the case that m <= n or an error will be thrown.
%  2) u must be a vector with N values, or an error will be thrown.
%  3) N must be >= n, or an error will be thrown.
n = length(a);
m = length(b);
if m > n
    error('m must be <= n');
end
if N < n
    error('N must be >= n');
end
y = zeros(N,1);
if nargin == 5
    y(1:n) = y0;
end
for k = n+1:N
    y(k) = -dot(a,y(k-1:-1:k-n)) + dot(b,u(k-1:-1:k-m));
end
return
```

```
Not enough input arguments.

Error in arx_sim (line 22)
n = length(a);
```

Published with MATLAB® R2019a

Appendix D

Switching Probability Code

```
% File Name: prbs.m
% Date created : 12/14/2018
% Author : Anene Vitus Omeje

function u = prbs(N,p)
% prbs creates an psuedo-random binary sequence alternating between
% 1 and -1, with switching probability p in [0 ,1]
% The closer p is to 1, the more likely a switch in state will occur.
%
% usage: u = prbs(N,p)
% where: N: length of u (i.e., if t = 1:N, u = [u(1),...,u(N)]
%        p: switching probability in [0,1]
u = ones(N,1);
r = rand(N,1);
Mask = (r < p);
s = 1-2*Mask;
for k = 2:N
    u(k) = s(k)*u(k-1);
end
return
```

```
Not enough input arguments.

Error in prbs (line 13)
u = ones(N,1);
```

Published with MATLAB® R2019a

Appendix E

Continuous to Discrete-Time Conversion

```
% File Name: cont_2_disc.m
% Date created: 09/24/2019
% Author: Anene Vitus Omeje
% Reference: Mathworks doucumentation, visited at
%"https://www.mathworks.com/help/control/ug/continuous-discrete-conversion-methods.html"

%Conversion from continuous-time to dicrete-time
numa = input('pls. enter continous-time system numerator, numa = ');
deno = input('pls. enter continous-time system denomintor, deno = ');
sT = input('pls. enter sample time in seconds, sT = ');
Sys_cont = tf(numa,deno); % continuous-time tranfer function
sys_disc = c2d(Sys_cont,sT) % equivalent discrete-time transfer function
```

```
Error using input
Cannot call INPUT from EVALC.

Error in cont_2_disc (line 8)
numa = input('pls. enter continous-time system numerator, numa = ');
```

Published with MATLAB® R2019a