Technische Hochschule Rosenheim
Technical University of Applied Sciences

Development of a Transparent Identification Method for Multi-Input Multi-Output Systems

Master's Thesis 2020/21

Engineering Sciences (Master)

Faculty of Engineering

submitted by

Abhishek Shivarkar

Matr. No.: 918732

Date of Submission: 28th May 2021

First Examiner:	Prof. Dr.-Ing. Peter Zentgraf
Second Examiner:	Prof. Dr.-Ing. Frank King

Index

Acknowledgement

It gives me a great pleasure to present this master’s thesis in the **“Engineering Sciences”** coursework. In preparing this thesis a number of hands helped me directly and indirectly.

I am very much obliged to **Prof. Dr.-Ing Peter Zentgraf** at **Technische Hochschule Rosenheim** for giving me the opportunity to work on this topic. I would like to thank him for helping and giving proper guidance throughout the thesis work. His timely suggestions made it possible to complete this thesis. All efforts would have gone in vain without his valuable guidance.

I would also like to extend my special thanks to the staff of Measurement and Control System Laboratory for supporting this thesis.

Sincerely,

Abhishek Shivarkar (918732)

Abstract

This thesis is an extension of the method presented by Prof. Dr. Peter Zentgraf for modelling linear systems. Here, an effort has been made to extend it to multiple-input multiple-output systems. A transparent technique that simulates processes using input and output measurement data without the use of any complex optimization or iterative algorithms has been developed. The solution obtained using least squares technique is very simple and easy to understand. The main aim is to provide bachelor students of engineering with a tool to formulate transfer functions.

The algorithm thus programmed using MATLAB has been tested for artificially generated erroneous measurement data and also for measurements obtained from a practical application at the university. Inclusion of dead times and estimation of initial conditions makes it ideal to be used for various range of applications such as stable and unstable systems with and without damping, open and closed loop systems. Over-integration, normalizing and/or zeroing of different coefficients adds more degrees of freedom to improve the model quality.

1 Introduction

1.1 System Identification

Nowadays developing mathematical models of systems play an important role in engineering related design tasks. Right from vibration suppression to process control a model of the control system is required. It helps to determine the system parameters and/or to simulate the behaviour of the control system so as to verify a certain desirable performance before actually using the controller in a real application. Thus, such models are useful for simulations, prediction and forecasting, state estimation and understanding and analysing system properties. There are two ways to derive these models, either from physical principles or based on experimentation. Laws of nature i.e., physical conservation laws are applied to the simplified technical processes in physical or theoretical modelling. This type of modelling can be very time-consuming and often the system parameters remain unknown. While in experimental modelling, mathematical models of dynamic systems are derived using measured experimental data and this process is known as system identification.

The initial step in this process is to design experiments in an optimal way to obtain good experimental data. It also includes selection of measured variables and the choice of input signals. Then with trial and error as well as using the prior knowledge of the system (if available), a suitable model structure is chosen. In the next step, a cost function that reflects how well the model fits the experimental data is defined. Further, this cost function is optimised to estimate the numerical values of the model parameters present in the selected model structure. The optimisation process adjusts the simulated response of this experimental model structure to the measured response. The last step is to validate the model by testing it in search of any inadequacies and then deciding if it satisfies the necessary application needs.

Various system identification techniques [1] present so far have one or more of the following characteristics:

- They are applicable to only certain type of systems
- They assume the system is in equilibrium position
- They need complex mathematical optimization methods and/or iterative algorithms to determine the model parameters
- They identify the unknown system parameters in terms of state-space models or discrete-time z-transfer functions
- They require special test signals and are used only for open loop systems

1.2 Motivation

The complex mathematical optimization methods are usually not a part of the curriculum for the bachelor program in engineering sciences. Nor the concept of state-space modelling is introduced in the bachelor course of control engineering. Therefore, it becomes very difficult for these students to understand the system identification techniques from a theoretical perspective, and it ultimately becomes a hindrance to use them effectively.

The primary motivation for this thesis is to provide bachelor students of engineering sciences, who are newly introduced to the subject of control engineering, with a simple, fast, and transparent method to formulate transfer functions of the control systems.

1.3 Thesis Objectives

1. Devise a mathematical procedure to model multiple-input multiple-output linear systems.
2. Develop a MATLAB program for the derived identification procedure.
3. Design multiple test cases for testing the complete MATLAB program.
4. Develop an algorithm for parametric search of system dead times.
5. Develop an optimization function to evaluate system dead times and compare the results with the parametric search.
6. Test and analyze the identification procedure with a real system.
7. Compare and analyze the identification results with MATLAB System Identification Toolbox.

2 Literature Review

2.1 State-of-the-Art

System identification problems [2] are usually characterized by the following set of selection choices:

- The model structure to be selected
- The input signals to be applied
- The identification process to be used to calculate unknown model parameters
- The method to assess the model quality
- The validation process to scrutinize and gain confidence in the estimated model

This gives rise to the following setup to approach system identification problems:

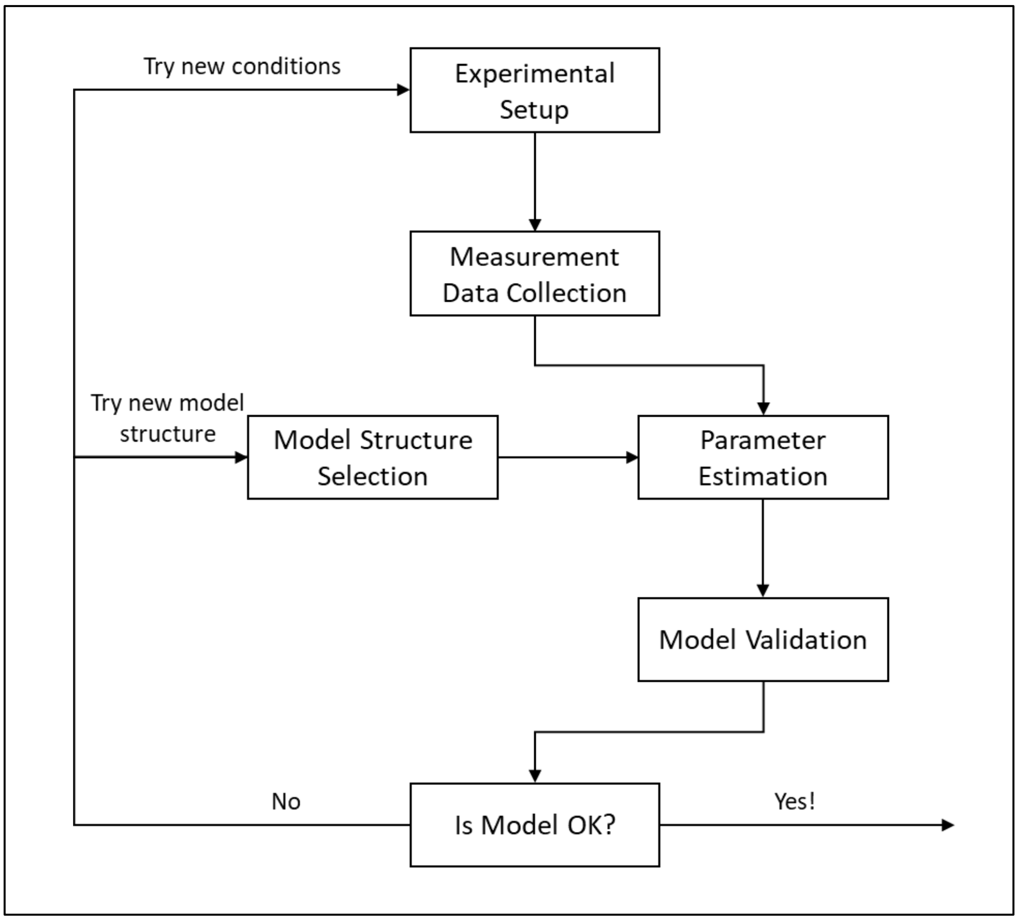
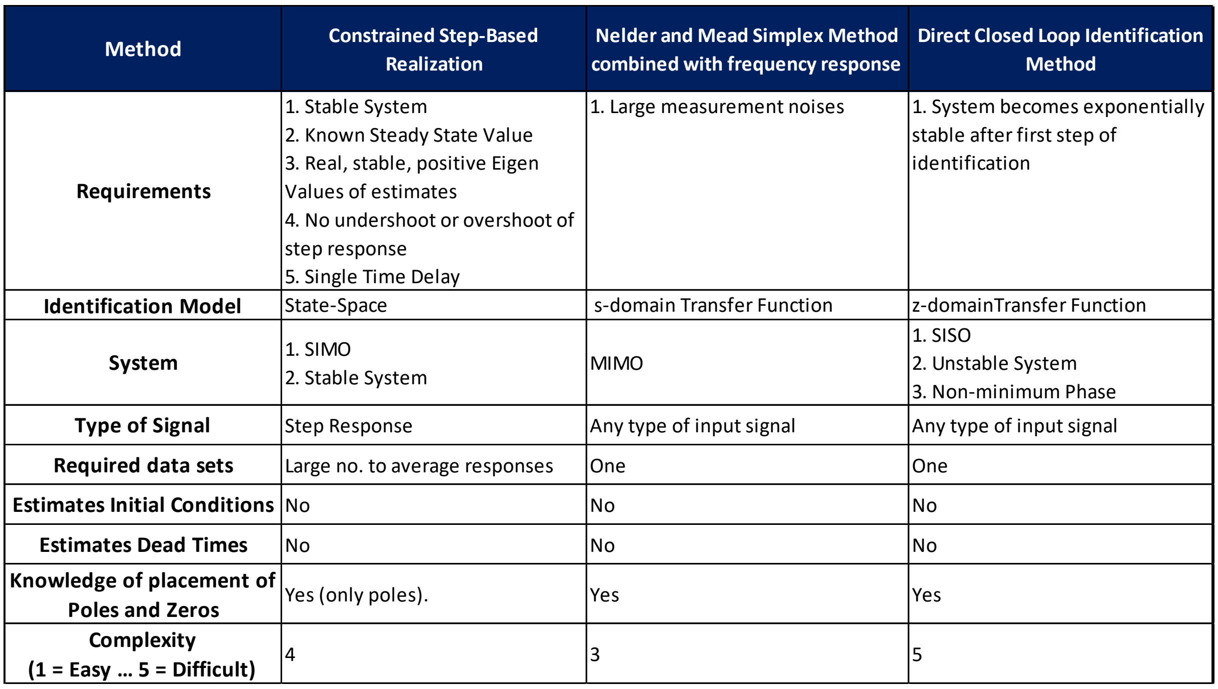


Figure 1: Overview of system identification process

The state-of-the-art method [3] to solve system identification problems is the kernel-based regularization method. This method is inspired from machine learning and is first described by Pillonetto and De Nicolao in 2010. In the first step, a kernel (model structure) is selected that includes the conditions for actual estimation problem. The kernels are defined by basis functions in terms of splines that are used in continuous function approximation techniques. The kernel designs like stable spline (SS) kernel [4] and diagonal corelated (DC) kernel [5] are widely used to determine the underlying model structure in this method. These kernel designs are obtained by two systematic ways, either from perspective of system theory or machine learning, which can embed various types of prior knowledge about the system. Once the kernel structure is selected, the next step is known as tuning that estimates the hyperparameters. It has no knowledge about the system and hence is a pure estimation problem of statistical nature. This regularization technique provides non-integer model orders which overcomes the difficulty faced by traditional methods to find the best model orders. A detailed implementation of the kernel-based regularization method is given in Chen and Ljung [6].

2.2 Survey

A short comparison is made between some identification methods used for linear time-invariant systems [7], [8], [9]. After listing some of the important characteristics of these methods, they are rated as per their complexity:



Method	Constrained Step-Based Realization	Nelder and Mead Simplex Method combined with frequency response	Direct Closed Loop Identification Method
Requirements	1. Stable System 2. Known Steady State Value 3. Real, stable, positive Eigen Values of estimates 4. No undershoot or overshoot of step response 5. Single Time Delay	1. Large measurement noises	1. System becomes exponentially stable after first step of identification
Identification Model	State-Space	s-domain Transfer Function	z-domainTransfer Function
System	1. SIMO 2. Stable System	MIMO	1. SISO 2. Unstable System 3. Non-minimum Phase
Type of Signal	Step Response	Any type of input signal	Any type of input signal
Required data sets	Large no. to average responses	One	One
Estimates Initial Conditions	No	No	No
Estimates Dead Times	No	No	No
Knowledge of placement of Poles and Zeros	Yes (only poles).	Yes	Yes
Complexity (1 = Easy ... 5 = Difficult)	4	3	5

Figure 2: Comparison between some methods for linear time-invariant systems

An approach in system identification was published by Prof. Dr. Peter Zentgraf [1] in 2019 for modelling single-input single-output linear systems. This paper discussed the use of linear ODEs with constant coefficients, Laplace Transforms, and the method of Least Squares to maintain a simple and clear approach. It also highlighted the advantages of identifying the system directly in the time-continuous Laplace domain (s-domain) with a practical example. If initial identification in z-domain is done and then a re-transformation procedure, like Euler or Tustin procedure [10], is used to transform into s-domain, then the coefficients of the transfer function lose their physical meaning. The final solution derived from method of Least Squares, first given by Gauss [11], uses only two analytical equations. The technique uses integration, first used by Golubev and Horowitz [12], instead of derivatives in the s-domain to filter out the noise and to make the resulting signals more deterministic. Test results for two real systems showed that this method calculates initial conditions and dead time of the system and can be used widely to identify different types of systems. In this thesis, an extension to include multi-input multi-output systems with the same method is developed.

3 Development of Identification Procedure

The primary objective is to determine the general system behaviour of the model from the measured data set of input signals $u(t)$ and output signals $y(t)$. Moreover, even for completely different input signals, this identified model should be able to predict the corresponding output behaviour without having to measure again. Therefore, for the sake of simplicity and without limiting the general validity, linear time-invariant ordinary differential equations (ODEs) are used. It is necessary that the input signals have enough dynamics so that the derivatives of input and output signals are not zero over a large period. Only then the coefficients of the ODE can be estimated with a numerical certainty. Also, the measurement errors are assumed to be zero-mean random numbers. The procedure described in this section is also easy to understand because it is very clearly derived with the help of Laplace transformation and Least Squares.

3.1 Pre-requisites

To understand the identification procedure from section 3.2 more effectively, only the theoretical concepts of Linear ODEs, Laplace transformations and the method of Least Squares from engineering mathematics are necessary. Therefore, a short review of these pre-requisites in relation with dynamic systems is provided in this section.

3.1.1 Linear Ordinary Differential Equations

The dynamic behaviour of a linear time-invariant system with an input variable $u(t)$ and output variable $y(t)$ can be described in the form of an ODE as follows:

$$b_0.y(t) + b_1.\dot{y}(t) + b_2.\ddot{y}(t) + \cdots + b_n.y^{(n)}(t) =$$
$$a_0.u(t) + a_1.\dot{u}(t) + a_2.\ddot{u}(t) + \cdots + a_m.u^{(m)}(t)$$

(1)

3.1.2 Laplace and Inverse Laplace Theorems

The general Laplace differentiation theorem is used for the transformation of input and output signals into Laplace domain. It is given by the following equation:

$$f^{(n)}(t) = s^n.F(s) - \sum_{k=0}^{n-1} f^{(k)}(0).s^{n-k-1}$$

(2)

Similarly, the inverse transformation of input and output signals from the Laplace domain into the time domain is given by the following integration theorem:

$$\frac{1}{s^{(n)}}.F(s) = f^{(-n)}(t)$$

(3)

3.1.3 Least Squares

The method of least squares from Gauss obtains an approximate solution to an overdetermined system of equations by minimizing the sum of the squares of the residuals from each equation. The general representation of the system of equations is as follows:

$$\underline{\tilde{y}} = H.\underline{x} + \underline{e} \tag{4}$$

where: $\underline{\tilde{y}} = [\tilde{y}_1 \quad \tilde{y}_2 \quad \cdots \quad \tilde{y}_m]^T$	: $m \times 1$ column vector of measurements
$\underline{x} = [x_1 \quad x_2 \quad \cdots \quad x_n]^T$	: $n \times 1$ column vector with unknown model parameters
$\underline{e} = [e_1 \quad e_2 \quad \cdots \quad e_m]^T$	: $m \times 1$ column vector with unknown estimation errors
$H = \begin{bmatrix} -h_1^T - \\ -h_2^T - \\ \vdots \\ -h_m^T - \end{bmatrix}$	: $m \times n$ Base matrix of the known base functions

For $H.x$ to be in good coincidence with the measurement values $\tilde{y}$, e should be as small as possible. This is achieved by minimizing the square-sum $J(x)$ with respect to x where:

$$J(x) = \frac{1}{2}.\underline{e}^T(x).\underline{e}(x) \tag{5}$$

The unknown model parameters can then be found out using the equation given below:

$$\underline{\hat{x}} = (H^T H)^{-1}.H^T.\underline{\tilde{y}} \tag{6}$$

3.2 Mathematical Description

Before proceeding into the extension of identification procedure to multi-input multi-output (MIMO) systems, it is necessary to review the equations of this procedure for single-input single-output (SISO) systems presented in section 3.2.1 (refer [1] for details).

3.2.1 Single-Input Single-Output Systems

A third order ODE is taken into consideration by substituting $m = 0$, $n = 3$ and $b_3 = 1$ in equation (1).

$$b_0.y(t) + b_1.\dot{y}(t) + b_2.\ddot{y}(t) + 1.\dddot{y}(t) = a_0.u(t) \tag{7}$$

The Laplace transforms are calculated with the help of equation (2) and $Y(s)$ is sorted which yields the following relation between the output variable, input variable and the initial conditions:

$$Y(s) = \underbrace{\frac{a_0}{s^3 + b_2.s^2 + b_1.s + b_0}U(s)}_{Part\ 1} + \underbrace{\frac{s^2.y(0) + s.\left(b_2.y(0) + \dot{y}(0)\right) + b_1.y(0) + b_2.\dot{y}(0) + \ddot{y}(0)}{s^3 + b_2.s^2 + b_1.s + b_0}}_{Part\ 2} \tag{8}$$

In the next step, the derivatives of the ODE are converted into integrals by multiplying with $\frac{1}{s^3}$ and then taking the Laplace inverse using equation (3). Further, the product of two unknown variables is converted into a single intermediate unknown variable for the applicability of Least Squares. The equation is as shown:

$$y(t) - y(0) + b_2.(y_i(t) - t.y(0)) + b_1\left(y_{ii}(t) - \frac{1}{2}.t^2.y(0)\right) + b_0.y_{iii}(\mathrm{t}) = \frac{1}{2}.t^2.y_{20} + t.y_{10} + a_0 u_{iii}(\mathrm{t}) \tag{9}$$

This equation is generalized for sampling times $t = [t_1\ t_2\ \dots\ t_q]$ and rearranged to the form of Least Squares.

$$\underbrace{\begin{bmatrix} y(t_1) - y(0) \\ \vdots \\ y(t_q) - y(0) \end{bmatrix}}_{\tilde{y}} =$$

$$\underbrace{\begin{bmatrix} t_1.y(0) - y_i(t_1) & \frac{1}{2}.t_1^2.y(0) - y_{ii}(t_1) & -y_{iii}(t_1) & u_{iii}(t_1) & \frac{1}{2}.t_1^2 & t_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ t_n.y(0) - y_i(t_q) & \frac{1}{2}.t_q^2.y(0) - y_{ii}(t_q) & -y_{iii}(t_q) & u_{iii}(t_q) & \frac{1}{2}.t_q^2 & t_q \end{bmatrix}}_{H} . \underbrace{\begin{bmatrix} b_2 \\ b_1 \\ b_0 \\ a_0 \\ y_{20} \\ y_{10} \end{bmatrix}}_{x} \tag{10}$$

Once the unknown coefficients in x are obtained by Least Squares, then the remaining initial conditions can be determined from the following linear system of equations:

$$\begin{bmatrix} y_{20} \\ y_{10} \end{bmatrix} = \begin{bmatrix} b_2 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{y}(0) \\ \ddot{y}(0) \end{bmatrix} \tag{11}$$

Therefore, the generalized equations for an ODE of order n as given in equation (1) are:

$$\underbrace{\begin{bmatrix} y(t_1) - y(0) \\ \vdots \\ y(t_q) - y(0) \end{bmatrix}}_{\tilde{y}} =$$

$$\underbrace{\begin{bmatrix} \frac{1}{1!}.t_1.y(0) - y_i(t_1) \dots \frac{1}{(n-1)!}.t_1^{n-1}.y(0) - y_{\underbrace{ii\dots i}_{n-1}}(t_1) & -y_{\underbrace{ii\dots i}_{n}}(t_1) & u_{\underbrace{ii\dots i}_{n}}(t_1) \dots u_{\underbrace{ii\dots i}_{n-m}}(t_1) & \frac{1}{(n-1)!}.t_1^{n-1} \dots \frac{1}{1!}.t_1 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{1}{1!}.t_q.y(0) - y_i(t_q) \dots \frac{1}{(n-1)!}.t_q^{n-1}.y(0) - y_{\underbrace{ii\dots i}_{n-1}}(t_q) & -y_{\underbrace{ii\dots i}_{n}}(t_q) & u_{\underbrace{ii\dots i}_{n}}(t_q) \dots u_{\underbrace{ii\dots i}_{n-m}}(t_q) & \frac{1}{(n-1)!}.t_q^{n-1} \dots \frac{1}{1!}.t_q \end{bmatrix}}_{H} . \underbrace{\begin{bmatrix} b_{n-1} \\ \vdots \\ b_0 \\ a_0 \\ \vdots \\ a_m \\ y_{(n-1)0} \\ \vdots \\ y_{10} \end{bmatrix}}_{x} \tag{12}$$

$$\begin{bmatrix} y_{(n-1)0} \\ \vdots \\ y_{20} \\ y_{10} \end{bmatrix} = \begin{bmatrix} b_2 & \cdots & b_{n-1} & 1 \\ \vdots & b_{n-1} & 1 & 0 \\ b_{n-1} & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{y}(0) \\ \ddot{y}(0) \\ \vdots \\ y^{(n-1)}(0) \end{bmatrix} \tag{13}$$

3.2.2 Extension to Multi-Input Multi-Output Systems

The mathematical development to achieve the primary objective of this thesis is based on similar lines of SISO. A dynamic system with measured input signals ($u_1(t)$, $u_2(t)$, … and so on) and measured output signals ($y_1(t)$, $y_2(t)$, … and so on) is considered. The identification method is developed initially for multiple-input single-output (MISO) system, and then it is extended for other outputs.

Without limiting the general validity, the linear time-invariant ODE considered for the dynamic behaviour of the MISO system can be described as follows:

$$\begin{gathered} b_0.y_1(t) + b_1.\dot{y}_1(t) + b_2.\ddot{y}_1(t) + \cdots + b_n.y_1^{(n)}(t) = \\ a_1.u_1(t) + a_{p+1}.\dot{u}_1(t) + a_{2p+1}.\ddot{u}_1(t) + \cdots + a_{mp+1}.u_1^{(m)}(t) \ + \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad\qquad \vdots \\ + a_p.u_p(t) + a_{2p}.\dot{u}_p(t) + a_{3p}.\ddot{u}_p(t) + \cdots + a_{(m+1)p}.u_p^{(m)}(t) \end{gathered} \tag{14}$$

Non-linear systems, where the unknown model parameters do not occur linearly, too can be transformed into a linear form by an off-integration or by replacement with intermediate variables and then this method with linear ODEs could be applied. Once the intermediate variables are calculated, they could be resolved to obtain the unknown non-linearly occurring model parameters (explained further in this section).

Again, without restriction of general validity, the further development is carried out using a third order ODE obtained by substituting $m = 0$, $n = 3$ and $p = 2$ in equation (14). Also, any one of the coefficients of the ODE can be freely chosen to be normalized or set to any other value and ideally it would not have any effect on the ODE. But due to measurement errors, the results change, and more number of possible solutions could be obtained (the derivation for this hidden degree of freedom is presented in section 3.3.2). Here, the coefficient b_3 is arbitrarily chosen and set to one. Therefore, the number of unknown model parameters is reduced by one.

$$b_0.y_1(t) + b_1.\dot{y}_1(t) + b_2.\ddot{y}_1(t) + 1.\dddot{y}_1(t) = a_1.u_1(t)+a_2.u_2(t) \tag{15}$$

This time domain ODE is transformed into the Laplace domain by applying Laplace transforms from equation (2). During this transformation, care should be taken with the initial values of input and output signals and its derivatives. At the start of an actual process, the initial values of the output signal and its derivatives $y_1(0)$, $\dot{y}_1(0)$, $\ddot{y}_2(0)$ would not necessarily be zero (i.e., when not started from equilibrium position). They could also have a non-zero value for $t < 0$ because they represent the state of system up to time $t = 0$ (i.e., when not started from rest position). As the effects till time $t = 0$ are incorporated into initial values of output and its derivatives, hence without limiting the general validity, the initial values of the input signal and its derivatives are assumed to have zero value for $t < 0$.

Accordingly, the following Laplace transformations are obtained:

$$b_0.y_1(t) \overset{\mathcal{L}}{\rightarrow} b_0.Y_1(s) \tag{16}$$

$$b_1.\dot{y}_1(t) \overset{\mathcal{L}}{\rightarrow} b_1.(s.Y_1(s) - y_1(0)) \tag{17}$$

$$b_2.\ddot{y}_1(t) \overset{\mathcal{L}}{\rightarrow} b_2.(s^2.Y_1(s) - s.y_1(0) - \dot{y}_1(0)) \tag{18}$$

$$1.\dddot{y}_1(t) \overset{\mathcal{L}}{\rightarrow} s^3.Y_1(s) - s^2.y_1(0) - s.\dot{y}_1(0) - \ddot{y}_2(0) \tag{19}$$

$$a_1.u_1(t) \overset{\mathcal{L}}{\rightarrow} a_1.U_1(s) \tag{20}$$

$$a_2.u_2(t) \overset{\mathcal{L}}{\rightarrow} a_2.U_2(s) \tag{21}$$

The equations from (16) to (21) are substituted in equation (15) to obtain the following equation:

$$\begin{gathered} b_0.Y_1(s) + b_1.(s.Y_1(s) - y_1(0)) + b_2.(s^2.Y_1(s) - s.y_1(0) - \dot{y}_1(0)) \\ +s^3.Y_1(s) - s^2.y_1(0) - s.\dot{y}_1(0) - \ddot{y}_2(0) = a_1.U_1(s) + a_2.U_2(s) \end{gathered} \tag{22}$$

The terms are rearranged and sorted for $Y_1(s)$ which gives the following equation:

$$\begin{gathered} Y_1(s) = \underbrace{\frac{a_1}{s^3 + b_2.s^2 + b_1.s + b_0} U_1(s)}_{Part\ 1} + \underbrace{\frac{a_2}{s^3 + b_2.s^2 + b_1.s + b_0} U_2(s)}_{Part\ 2} \\ + \underbrace{\frac{s^2.y_1(0) + s.\left(b_2.y_1(0) + \dot{y}_1(0)\right) + b_1.y_1(0) + b_2.\dot{y}_1(0) + \ddot{y}_1(0)}{s^3 + b_2.s^2 + b_1.s + b_0}}_{Part\ 3} \end{gathered} \tag{23}$$

Therefore, equation (23) can be used to simulate the system with non-zero initial conditions. But if the initial conditions, $y_1(0)$, $\dot{y}_1(0)$, $\ddot{y}_2(0)$, are zero, then the value of $Part$ 3 is zero and the output is only impacted by the input signals $U_1(s)$ and $U_2(s)$ through the transfer function in $Part$ 1 and $Part$ 2 respectively.

Now, the equation (22) has the highest power of s as 3, so to avoid all the derivatives in time domain, it is multiplied by $\frac{1}{s^3}$. A product with higher order is also possible and further solutions could be obtained (the derivation for this hidden degree of freedom is presented in section 3.3.1). Thus, the entire equation is converted into an integral relationship which is now easier to solve and is sorted by the unknown model parameters as follows:

$$\left(Y_1(s) - \frac{1}{s}.y_1(0) - \frac{1}{s^2}.\dot{y}_1(0) - \frac{1}{s^3}.\ddot{y}_2(0)\right) + b_2.\left(\frac{1}{s}.Y_1(s) - \frac{1}{s^2}.y_1(0)\right)$$
$$+b_2.\dot{y}_1(0)\left(-\frac{1}{s^3}\right) + b_1.\left(\frac{1}{s^2}.Y_1(s) - \frac{1}{s^3}.y_1(0)\right) + b_0.\frac{1}{s^3}.Y_1(s)$$
$$= a_1.\frac{1}{s^3}.U_1(s) + a_2.\frac{1}{s^3}.U_2(s) \tag{24}$$

The method of Least Squares cannot be applied directly when there is a product of two unknowns (here, $b_2.\dot{y}_1(0)$). Hence intermediate variables are needed to be introduced to keep the equation in the correct form for Least Squares. Here y_{120}, y_{110} are used:

$$\left(Y_1(s) - \frac{1}{s}.y_1(0)\right) + b_2.\left(\frac{1}{s}.Y_1(s) - \frac{1}{s^2}.y_1(0)\right) + b_1.\left(\frac{1}{s^2}.Y_1(s) - \frac{1}{s^3}.y_1(0)\right)$$
$$+b_0.\frac{1}{s^3}.Y_1(s) - \left(\underbrace{b_2.\dot{y}_1(0) + \ddot{y}_2(0)}_{=:y_{120}}\right).\frac{1}{s^3} - \underbrace{\dot{y}_1(0)}_{=:y_{110}}.\frac{1}{s^2}$$
$$= a_1.\frac{1}{s^3}.U_1(s) + a_2.\frac{1}{s^3}.U_2(s) \tag{25}$$

After calculation of b_2, y_{120} and y_{110} the initial conditions can be recalculated again (explained further in this section). $y_1(0)$ is assumed to be known. If in case that's not true then, additional intermediate variables are needed to be introduced where $y_1(0)$ has a product with another unknown parameter.

The equation (25) is transformed back into time domain using inverse Laplace transform from equation (3). The inverse Laplace for the functions with time ($t \geq 0$) is given by:

$$\frac{1}{s}.y_1(0) \xrightarrow{\mathcal{L}^{-1}} y_1(0) \tag{26}$$

$$\frac{1}{s^2}.y_1(0) \xrightarrow{\mathcal{L}^{-1}} t.y_1(0) \tag{27}$$

$$\frac{1}{s^3}.y_1(0) \xrightarrow{\mathcal{L}^{-1}} \frac{t^2}{2}.y_1(0) \tag{28}$$

Usually, the measurement data is split for estimation and verification. Hence, it is efficient to use two different time coordinates, t for measured data set and $\tilde{t}$ for the estimation data set. It is considered that at time instance $t = t_1$, the estimation starts and hence the variable $\tilde{t} = 0$ which also means $\tilde{t} = t - t_1$. This is helpful since the contribution of lower limit of integration is zero and it decreases the number of terms in the procedure. Moreover, the system simulation should also start at time zero seconds. The output signals $y(t)$ and the input signals $u(t)$ are changed to $y(\tilde{t})$ and $u(\tilde{t})$ respectively.

$$y_1(\tilde{t}) - y_1(0) + b_2.\left(y_{1i}(\tilde{t}) - \tilde{t}.y_1(0)\right) + b_1\left(y_{1ii}(\tilde{t}) - \frac{1}{2}.\tilde{t}^2.y_1(0)\right) + b_0.y_{1iii}(\tilde{t})$$
$$= \frac{1}{2}.\tilde{t}^2.y_{120} + \tilde{t}.y_{110} + a_1u_{1iii}(\tilde{t}) + a_2u_{2iii}(\tilde{t}) \tag{29}$$

This equation is shown below for different sampling times $\tilde{t} = [\tilde{t_1} \quad \tilde{t_2} \quad ... \quad \tilde{t_q}]$ in matrix form:

$$\begin{bmatrix} y_1(\tilde{t}_1) - y_1(0) \\ \vdots \\ y_1(\tilde{t}_q) - y_1(0) \end{bmatrix} + \begin{bmatrix} y_{1i}(\tilde{t}_1) - \tilde{t}_1 y_1(0) \\ \vdots \\ y_{1i}(\tilde{t}_q) - \tilde{t}_q y_1(0) \end{bmatrix}.b_2 + \begin{bmatrix} y_{1ii}(\tilde{t}_1) - \frac{1}{2}.\tilde{t}_1^2.y_1(0) \\ \vdots \\ y_{1ii}(\tilde{t}_q) - \frac{1}{2}.\tilde{t}_q^2.y_1(0) \end{bmatrix}.b_1 + \begin{bmatrix} -y_{1iii}(\tilde{t}_1) \\ \vdots \\ -y_{1iii}(\tilde{t}_q) \end{bmatrix}.b_0 =$$
$$\frac{1}{2}.\begin{bmatrix} \tilde{t}_1^2 \\ \vdots \\ \tilde{t}_q^2 \end{bmatrix}.y_{120} + \begin{bmatrix} \tilde{t}_1 \\ \vdots \\ \tilde{t}_q \end{bmatrix}.y_{110} + \begin{bmatrix} u_{1iii}(\tilde{t}_1) \\ \vdots \\ u_{1iii}(\tilde{t}_q) \end{bmatrix}.a_1 + \begin{bmatrix} u_{2iii}(\tilde{t}_1) \\ \vdots \\ u_{2iii}(\tilde{t}_q) \end{bmatrix}.a_2 \tag{30}$$

This equation is rearranged to a suitable form, to be solved by Least Squares as shown below:

$$\underbrace{\begin{bmatrix} y_1(\tilde{t}_1) - y_1(0) \\ \vdots \\ y_1(\tilde{t}_q) - y_1(0) \end{bmatrix}}_{\tilde{y}} =$$

$$\underbrace{\begin{bmatrix} \tilde{t}_1 . y_1(0) - y_{1i}(\tilde{t}_1) & \frac{1}{2} . \tilde{t}_1^2 . y_1(0) - y_{1ii}(\tilde{t}_1) & -y_{1iii}(\tilde{t}_1) & u_{1iii}(\tilde{t}_1) & u_{2iii}(\tilde{t}_1) & \frac{1}{2} . \tilde{t}_1^2 & \tilde{t}_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \tilde{t}_q . y_1(0) - y_{1i}(\tilde{t}_q) & \frac{1}{2} . \tilde{t}_q^2 . y_1(0) - y_{1ii}(\tilde{t}_q) & -y_{1iii}(\tilde{t}_q) & u_{1iii}(\tilde{t}_q) & u_{2iii}(\tilde{t}_q) & \frac{1}{2} . \tilde{t}_q^2 & \tilde{t}_q \end{bmatrix}}_{H} \underbrace{\begin{bmatrix} b_2 \\ b_1 \\ b_0 \\ a_1 \\ a_2 \\ y_{120} \\ y_{110} \end{bmatrix}}_{x} \tag{31}$$

The unknown model parameters in equation (31) are to be found out so that $\tilde{y}$ and $H.x$ are as equal as possible. By adding an error vector e for non-modellable random measurement errors as well as modelling errors, this can be solved by the Method of Least Squares (refer section 3.1.3).

Once the unknown coefficients and intermediate variables are obtained by Least Squares, then the initial conditions can be calculated back from the intermediate variables by inverting the following linear system of equation:

$$\begin{bmatrix} y_{120} \\ y_{110} \end{bmatrix} = \begin{bmatrix} b_2 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{y}_1(0) \\ \ddot{y}_1(0) \end{bmatrix} \tag{32}$$

Further, this procedure is generalised for an ODE with output and input signals with order n and m respectively and with p input signals. The two analytical equations obtained are as follows:

$$\underbrace{\begin{bmatrix} y_1(\tilde{t}_1) - y_1(0) \\ \vdots \\ y_1(\tilde{t}_q) - y_1(0) \end{bmatrix}}_{\tilde{y}} =$$

$$\underbrace{\begin{bmatrix} \frac{1}{1!}.\tilde{t}_1.y_1(0) - y_{1i}(\tilde{t}_1) \dots \frac{1}{(n-1)!}.\tilde{t}_1^{n-1}.y_1(0) - y_{1\underbrace{ii\dots i}_{n-1}}(\tilde{t}_1) & -y_{1\underbrace{ii\dots i}_{n}}(\tilde{t}_1) & u_{1\dots p\underbrace{i\dots ii}_{n}}(\tilde{t}_1) \dots u_{1\dots p\underbrace{i\dots ii}_{n-m}}(\tilde{t}_1) & \frac{1}{(n-1)!}.\tilde{t}_1^{n-1} \dots \frac{1}{1!}.\tilde{t}_1 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{1}{1!}.\tilde{t}_q.y_1(0) - y_{1i}(\tilde{t}_q) \dots \frac{1}{(n-1)!}.\tilde{t}_q^{n-1}.y_1(0) - y_{1\underbrace{ii\dots i}_{n-1}}(\tilde{t}_q) & -y_{1\underbrace{ii\dots i}_{n}}(\tilde{t}_q) & u_{1\underbrace{\dots pi\dots ii}_{n}}(\tilde{t}_q) \dots u_{1\dots p\underbrace{i\dots ii}_{n-m}}(\tilde{t}_q) & \frac{1}{(n-1)!}.\tilde{t}_q^{n-1} \dots \frac{1}{1!}.\tilde{t}_q \end{bmatrix}}_{H}.$$

$$\underbrace{\begin{bmatrix} b_{n-1} \\ \vdots \\ b_0 \\ a_1 \\ \vdots \\ a_{(m+1).p} \\ y_{1(n-1)0} \\ \vdots \\ y_{110} \end{bmatrix}}_{x} \tag{33}$$

$$\begin{bmatrix} y_{1(n-1)0} \\ \vdots \\ y_{120} \\ y_{110} \end{bmatrix} = \begin{bmatrix} b_2 & \cdots & b_{n-1} & 1 \\ \vdots & b_{n-1} & 1 & 0 \\ b_{n-1} & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{y}(0) \\ \ddot{y}(0) \\ \vdots \\ y^{(n-1)}(0) \end{bmatrix} \tag{34}$$

In a similar way, this procedure can be extended to the remaining outputs ($y_2(t)$, $y_3(t)$ and so on) and the coefficients of ODE and the initial conditions could be found out for the same.

3.3 Derivation of Additional Functionalities

Some hidden degrees of freedom were discussed in the previous section 3.2.2. A more detailed theoretical procedure of some additional functionalities along with these hidden degrees of freedom is presented here in this section. These additional functionalities can help to improve the model quality.

3.3.1 Over-Integration

Consider the same ODE used to present the mathematical procedure from the previous section. The equation (22) is now multiplied by a higher order, say $\frac{1}{s^4}$ instead of $\frac{1}{s^3}$. This not only avoids all the derivatives in time domain but also integrates the equation one more time.

$$\left(\frac{1}{s}.Y_1(s)-\frac{1}{s^2}.y_1(0)-\frac{1}{s^3}.\dot{y}_1(0)-\frac{1}{s^4}.\ddot{y}_2(0)\right)+b_2.\left(\frac{1}{s^2}.Y_1(s)-\frac{1}{s^3}.y_1(0)\right)$$

$$+b_2.\dot{y}_1(0)\left(-\frac{1}{s^4}\right)+b_1.\left(\frac{1}{s^3}.Y_1(s)-\frac{1}{s^4}.y_1(0)\right)+b_0.\frac{1}{s^4}.Y_1(s)$$

$$=a_1.\frac{1}{s^4}.U_1(s)+a_2.\frac{1}{s^4}.U_2(s) \tag{35}$$

Now, the same procedure is followed, firstly the product of two unknowns is substituted with intermediate variables and then inverse Laplace transformation is performed. The equation is rearranged to the following form:

$$\underbrace{\begin{bmatrix} y_{1i}(\tilde{t}_1)-\tilde{t}_1.y_1(0) \\ \vdots \\ y_{1i}(\tilde{t}_q)-\tilde{t}_q.y_1(0) \end{bmatrix}}_{\tilde{y}} =$$

$$\underbrace{\begin{bmatrix} \frac{1}{2}.\tilde{t}_1^2.y_1(0)-y_{1ii}(\tilde{t}_1) & \frac{1}{3!}.\tilde{t}_1^3.y_1(0)-y_{1iii}(\tilde{t}_1) & -y_{1iiii}(\tilde{t}_1) & u_{1iiii}(\tilde{t}_1) & u_{2iiii}(\tilde{t}_1) & \frac{1}{3!}.\tilde{t}_1^3 & \frac{1}{2}.\tilde{t}_1^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{2}.\tilde{t}_q^2.y_1(0)-y_{1ii}(\tilde{t}_q) & \frac{1}{3!}.\tilde{t}_q^3.y_1(0)-y_{1iii}(\tilde{t}_q) & -y_{1iii}i(\tilde{t}_q) & u_{1iiii}(\tilde{t}_q) & u_{2iiii}(\tilde{t}_q) & \frac{1}{3!}.\tilde{t}_q^3 & \frac{1}{2}.\tilde{t}_q^2 \end{bmatrix}}_{H} \underbrace{\begin{bmatrix} b_2 \\ b_1 \\ b_0 \\ a_1 \\ a_2 \\ y_{120} \\ y_{110} \end{bmatrix}}_{x} \tag{36}$$

Once the unknown model parameters in x are calculated using Least Squares, the initial conditions can be found out using the same equation (32). Further, the system behaviour can be simulated using the equation (23) in terms of transfer function. Therefore, in general an ODE with output order n can be over-integrated to obtain multiple possible solutions.

3.3.2 Normalizing Different Denominator Coefficients

Here, in the third order ODE, (which is obtained by substituting $m = 0$, $n = 3$ and $p = 2$ in equation (14)), b_2 (instead of b_3) is arbitrarily chosen and set to one.

$$b_0 . y_1(t) + b_1 . \dot{y}_1(t) + 1 . \ddot{y}_1(t) + b_3 . \dddot{y}_1(t) = a_1 . u_1(t) + a_2 . u_2(t) \tag{37}$$

The Laplace transformations are carried out to give the following equation:

$$\begin{gathered} b_0 . Y_1(s) + b_1 . (s . Y_1(s) - y_1(0)) + 1 . (s^2 . Y_1(s) - s . y_1(0) - \dot{y}_1(0)) \\ +b_3 . \left(s^3 . Y_1(s) - s^2 . y_1(0) - s . \dot{y}_1(0) - \ddot{y}_2(0)\right) = a_1 . U_1(s) + a_2 . U_2(s) \end{gathered} \tag{38}$$

The terms are rearranged and sorted for $Y_1(s)$ as shown below:

$$\begin{gathered} Y_1(s) = \underbrace{\frac{a_1}{b_3 . s^3 + 1 . s^2 + b_1 . s + b_0} U_1(s)}_{Part\ 1} + \underbrace{\frac{a_2}{b_3 . s^3 + 1 . s^2 + b_1 . s + b_0} U_2(s)}_{Part\ 2} \\ + \underbrace{\frac{s^2 . b_3 . y_1(0) + s . \left(1 . y_1(0) + b_3 . \dot{y}_1(0)\right) + b_1 . y_1(0) + 1 . \dot{y}_1(0) + b_3 . \ddot{y}_1(0)}{b_3 . s^3 + 1 . s^2 + b_1 . s + b_0}}_{Part\ 3} \end{gathered} \tag{39}$$

This equation is used to simulate the system behaviour once the unknown model parameters are calculated. The further procedure is same, i.e., equation (38) is multiplied by $\frac{1}{s^3}$, substituted by intermediate variables, and transformed by inverse Laplace theorems to obtain the following equation suitable to be solved by Least Squares:

$$\underbrace{\begin{bmatrix} y_{1i}(\tilde{t}_1) - \tilde{t}_1 . y_1(0) \\ \vdots \\ y_{1i}(\tilde{t}_q) - \tilde{t}_q . y_1(0) \end{bmatrix}}_{\tilde{y}} =$$

$$\underbrace{\begin{bmatrix} y_1(0) - y_1(\tilde{t}_1) & \frac{1}{2} . \tilde{t}_1^2 . y_1(0) - y_{1ii}(\tilde{t}_1) & -y_{1iii}(\tilde{t}_1) & u_{1iii}(\tilde{t}_1) & u_{2iii}(\tilde{t}_1) & \frac{1}{2} . \tilde{t}_1^2 & \tilde{t}_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ y_1(0) - y_1(\tilde{t}_q) & \frac{1}{2} . \tilde{t}_q^2 . y_1(0) - y_{1ii}(\tilde{t}_q) & -y_{1iii}(\tilde{t}_q) & u_{1iii}(\tilde{t}_q) & u_{2iii}(\tilde{t}_q) & \frac{1}{2} . \tilde{t}_q^2 & \tilde{t}_q \end{bmatrix}}_{H} \underbrace{\begin{bmatrix} b_3 \\ b_1 \\ b_0 \\ a_1 \\ a_2 \\ y_{120} \\ y_{110} \end{bmatrix}}_{x} \tag{40}$$

Here, b_2 (instead of b_3) is determined in the vector of unknown model parameters x. The initial conditions can be calculated back from the intermediate variables by inverting the following linear system of equation:

$$\begin{bmatrix} y_{120} \\ y_{110} \end{bmatrix} = \begin{bmatrix} 1 & b_3 \\ b_3 & 0 \end{bmatrix} \begin{bmatrix} \dot{y}_1(0) \\ \ddot{y}_1(0) \end{bmatrix} \tag{41}$$

Similarly, any coefficient of the output signal can be normalized, and the equations could be solved to obtain further possible solutions.

3.3.3 Zeroing Different Coefficients

In the case when one or many coefficients of the ODE are pre-decided to be zero, then the procedure remains same, but some terms are removed from the equations. In equation (15), say b_2 is arbitrarily chosen and set to zero, the ODE equation changes to:

$$b_0.y_1(t) + b_1.\dot{y}_1(t) + 1.\ddot{y}_1(t) = a_1.u_1(t) + a_2.u_2(t) \tag{42}$$

Then the Laplace transformations are carried out to give the following equation:

$$b_0.Y_1(s) + b_1.(s.Y_1(s) - y_1(0)) + s^3.Y_1(s) - s^2.y_1(0) - s.\dot{y}_1(0) - \ddot{y}_2(0) = \\ a_1.U_1(s) + a_2.U_2(s) \tag{43}$$

The terms are rearranged and sorted for $Y_1(s)$ which gives the following equation:

$$Y_1(s) = \underbrace{\frac{a_1}{s^3 + b_1.s + b_0} U_1(s)}_{Part\ 1} + \underbrace{\frac{a_2}{s^3 + b_1.s + b_0} U_2(s)}_{Part\ 2} \\ + \underbrace{\frac{s^2.y_1(0) + s.\dot{y}_1(0) + b_1.y_1(0) + \ddot{y}_1(0)}{s^3 + +b_1.s + b_0}}_{Part\ 3} \tag{44}$$

This equation is used to simulate the system behaviour once the unknown model parameters are calculated. The next procedure is same, i.e., equation (43) is multiplied by $\frac{1}{s^3}$, substituted by intermediate variables, and transformed by inverse Laplace theorem to obtain the following equation suitable to be solved by Least Squares:

$$\underbrace{\begin{bmatrix} y_1(\tilde{t}_1) - y_1(0) \\ \vdots \\ y_1(\tilde{t}_q) - y_1(0) \end{bmatrix}}_{\tilde{y}} =$$

$$\underbrace{\begin{bmatrix} \frac{1}{2}.\tilde{t}_1^2.y_1(0) - y_{1ii}(\tilde{t}_1) & -y_{1iii}(\tilde{t}_1) & u_{1iii}(\tilde{t}_1) & u_{2iii}(\tilde{t}_1) & \frac{1}{2}.\tilde{t}_1^2 & \tilde{t}_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{2}.\tilde{t}_q^2.y_1(0) - y_{1ii}(\tilde{t}_q) & -y_{1iii}(\tilde{t}_q) & u_{1iii}(\tilde{t}_q) & u_{2iii}(\tilde{t}_q) & \frac{1}{2}.\tilde{t}_q^2 & \tilde{t}_q \end{bmatrix}}_{H} \underbrace{\begin{bmatrix} b_1 \\ b_0 \\ a_1 \\ a_2 \\ y_{120} \\ y_{110} \end{bmatrix}}_{x}$$

(45)

Since, b_2 is chosen to be zero, it simply means that the column corresponding to it in the base matrix is removed and it is also absent in the vector of unknown model parameters x.The initial conditions can be calculated back from the intermediate variables by inverting the following linear system of equation:

$$\begin{bmatrix} y_{120} \\ y_{110} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{y}_1(0) \\ \ddot{y}_1(0) \end{bmatrix}$$

(46)

Therefore, in a similar way any coefficient of the output and/or input signals can be set to zero to obtain different possible models.

3.3.4 Adding Weights

At times, a particular range of input measurement data has a strong influence on the output producing a transient response. Such time periods of the measured data can be taken care of by assigning a special weighting factor in the identification procedure. The individual elements of the error vector e are weighted by a large value at the corresponding position of a diagonal weighting matrix W, so now the cost function becomes the weighted error square sum:

$$J(x) = \frac{1}{2}.\underline{e}^T(x).W.\underline{e}(x)$$

(47)

The $J(x)$ is minimized with respect to x and the unknown model parameters are found out using the equation:

$$\underline{\hat{x}} = (H^T.W.H)^{-1}.H^T.W.\underline{\tilde{y}}$$

(48)

3.3.5 Dead Time – Parametric Search

Dead time is the delay from when an input signal starts until when the output signal first responds. This section explains the theoretical procedure to calculate the dead time in a system.

Consider a system with two input signals u_1 and u_2 and one output signal y_1 as shown in following figure:

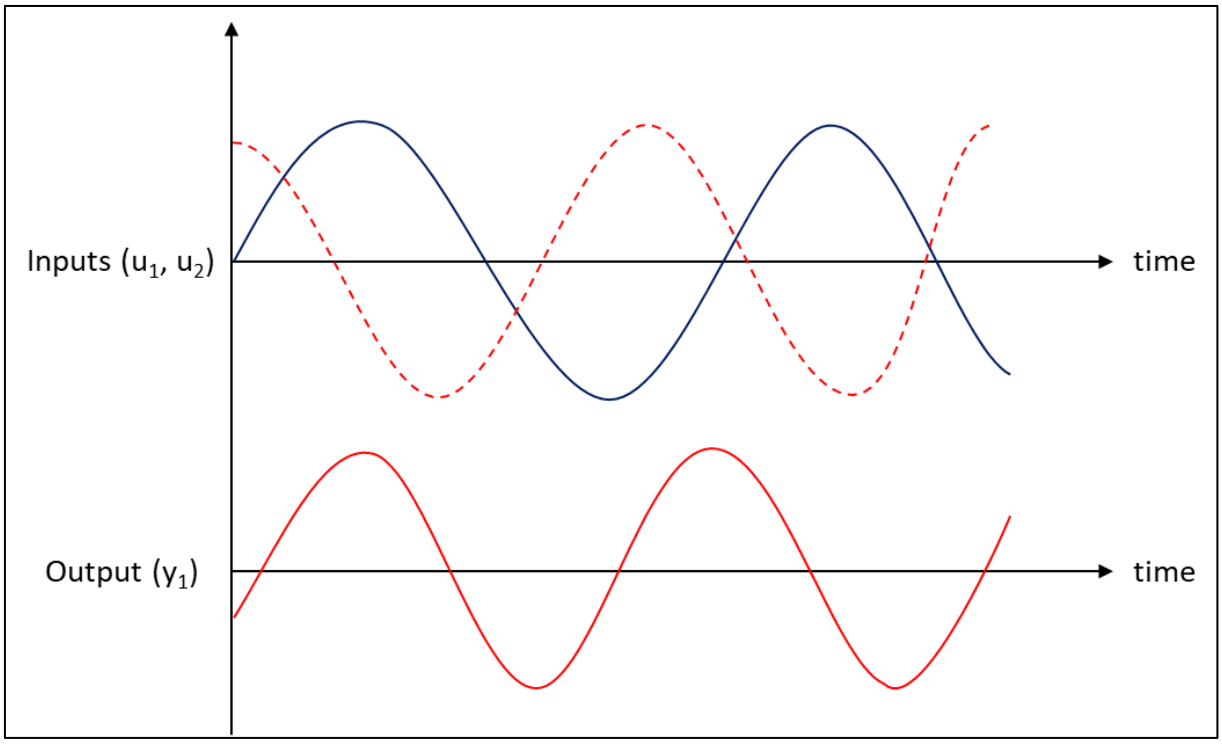


Figure 3: Time representation of a system with input and output signals

In this type of multi-input system two types of dead times usually appear namely, output time delays and the input time delays. Although both the delays cause the output to appear at a later point of time than the input, these delays can be calculated independently of each other.

A. Output Time Delays:

To calculate the output time delay, the output signal is shifted back in time step by step between pre-defined time bounds. At each step, the identification procedure (from section 3.2.2) is performed and using the Least Squares, the square-sum $J(x)$ is obtained from equation (5). This $J(x)$ is divided by number of measurements used for estimation and this value, say J_d, is recorded. At the end of this process, all J_d obtained at different steps are compared, and the solution is obtained at the step where J_d is the smallest. The time offset at this step is calculated to give the required output time delay. The following figure shows the concept of shifting the output signal into the past:

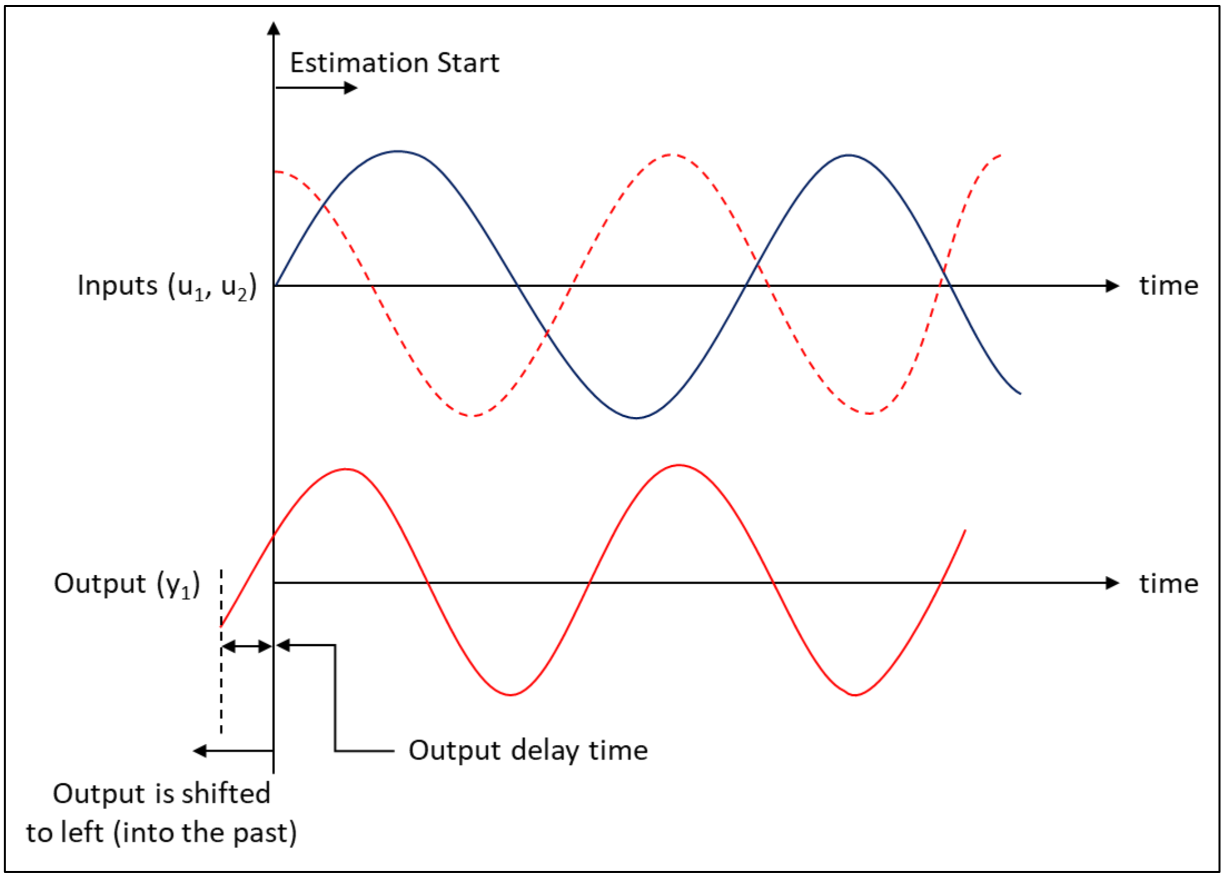


Figure 4: Concept of shifting the output signal into the past

B. Input Time Delays:

In case of multi-input systems, there are more than one input signals, that can have individual delay times. Therefore, each input signal is shifted individually into the future, step by step. But to apply this, a different start of estimation is arbitrarily chosen because there is no record of the data previous to the measurement range. At each step, the identification is done, and the square-sum $J(x)$ is calculated. Here too, the value J_d is calculated by dividing $J(x)$ by the number of measurements used for estimation. At the end of this process, the minimum among all J_d is found out and the step corresponding to it is checked. Now, at this step, the time offset from the start of measuring range is calculated separately for each input signal, which provides the individual input time delays. The following figure shows the concept of shifting the input signals into the future:

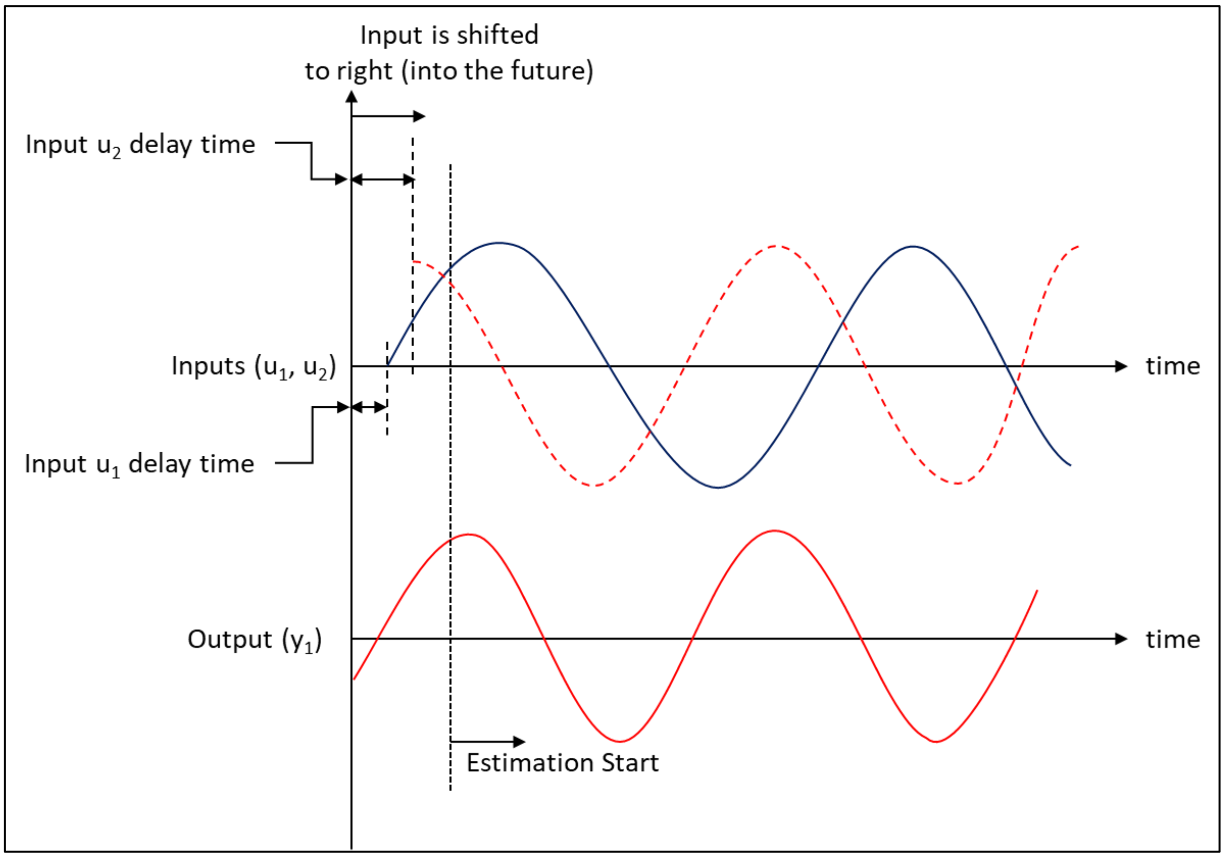


Figure 5: Concept of shifting the input signals into the future

It should be noted that in both the cases due to the shifting of signals, some measurement data is omitted, and the information is always lost.

This process for calculation of both output and input delay times can also be extended for multiple outputs ($y_2(t)$, $y_3(t)$ and so on) in a similar way.

3.4 Evaluation Criteria

Two criteria are defined for the evaluation of the identification procedure:

A. Fit Value:

$$fit = 100\%.\left(1 - \frac{norm(y_{Mess} - y_{Sim})}{norm\big(y_{Mess} - mean(y_{Mess})\big)}\right) \tag{49}$$

where, $norm$ is the magnitude of the vector. It is also called as Euclidean or 2-norm. Whereas $mean$ refers to the mean value of the vector. y_{Mess} and y_{Sim} refer to the measured and simulated output, respectively. This fit value [13] is basically normalized root mean square error between the measured and simulated output data (obtained from equation (23)) and acts as a criterion to determine the quality of identification.

B. Back Calculation Accuracy (RR) Value:

$$RR = 100\%.\left(1 - \frac{norm(c - c_e)}{norm(c)}\right) \tag{50}$$

where vector c comprises of the true coefficients and true initial conditions:

$$c = [a_1 \quad a_2 \quad b_2 \quad b_1 \quad b_0 \quad \dot{y}(0) \quad \ddot{y}(0)]^T$$

while vector c_e consists of the estimated coefficients (obtained from $\underline{\hat{x}}$ after minimization) and estimated initial conditions (from equation (32)). Thus, the RR value, also known as back calculation accuracy, is a measure of the deviations of the estimated coefficients from the true coefficients.

4 Programming Structure – MATLAB

For the identification procedure described in section 3, the entire programming and testing is performed in MATLAB Version: 9.9.0.1570001 (R2020b).

4.1 About MATLAB

MATLAB is a programming language that provides a numeric computing environment to develop algorithms and create models. It is profoundly used in engineering courses for academic and research purposes due to its following advantages over other programming languages:

- It is very easy to implement and test algorithms without the need of recompilation
- It has inbuilt functions and algorithms for faster development of program codes
- The debugging feature is very effective to find bugs and errors and makes it easier to improve the code
- It facilitates development of standalone desktop or web application with the help of App Designer and MATLAB Compiler.

These advantages added with the fact that MATLAB is already a part of the curriculum for the bachelor students of engineering courses helps to satisfy the thesis requirements and makes MATLAB suitable for programming this identification procedure.

4.2 Design Philosophy

Before the start of programming, a list of technical requirements to be achieved from the identification program is made. This list is split into two parts to provide a sufficient insight for programming.

A. Basic Requirements
1. Number of input and output signals are identifiable
2. Splitting of data into two parts viz. for estimation and simulation is possible
3. The numerator and denominator order of transfer function is selectable
4. Zero or non-zero initial conditions is selectable
5. Input, output, and difference graphs are plotted
6. Fit and RR value are calculated
7. Estimation with or without noise is possible

B. Additional Requirements
1. Degree of over-integration is selectable
2. Normalizing each denominator coefficient is possible
3. Setting coefficients equal to zero is possible
4. Estimation for various sample sizes is possible
5. Results in terms of coefficients of transfer function or poles-zeros is selectable
6. Variable start of estimation is possible
7. Measured data weightages are selectable
8. Dead times of each transfer function is estimated

To fulfil these program requirements a structured way of programming is adopted. Initially, only the basic requirements are programmed. The codes are written in function files. These function files do not provide output on their own, but for them to do so they need to be called out by other files by passing on some parameters as input. Each function file follows a specific format. It begins with a header which includes the file name, its purpose, input and output parameters of the function, name of the author, time of creation and a change log that has a note of all the modification activities done in the program. The remaining part of these files is the programmed algorithm intended for further use. Moreover, wherever required, every section of the code is properly commented with the necessary description for easier understanding. Once the basic requirements are programmed, testing is performed using the test files (that also follow a similar header format). Further, the in-built features of debugging and generating profiler report are used to solve the programming errors and improve the speed and quality of code. For each additional requirement that is programmed, the function files are modified by adding the required algorithm and the corresponding input-output parameters. These files are tested every time after modification using new test files. If required, the codes are debugged, and the run time is measured again. Also, after every modification, it is ensured that the previous test files and functionalities still run without an error. In this manner, a new requirement is programmed into the code without the loss of the old ones.

Following is the program structure of the identification procedure:

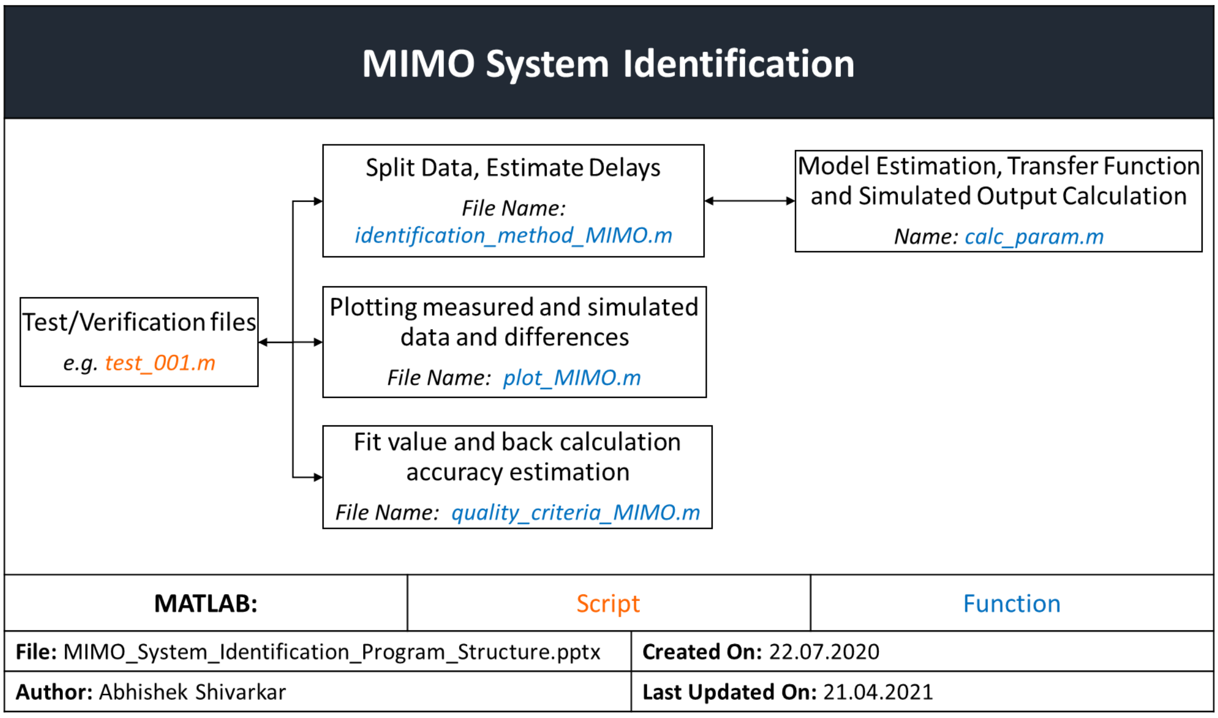

MIMO System Identification
Split Data, Estimate Delays
File Name: identification_method_MIMO.m
Model Estimation, Transfer Function and Simulated Output Calculation
Name: calc_param.m
Test/Verification files
e.g. test_001.m
Plotting measured and simulated data and differences
File Name: plot_MIMO.m
Fit value and back calculation accuracy estimation
File Name: quality_criteria_MIMO.m
MATLAB:
Script
Function
File: MIMO_System_Identification_Program_Structure.pptx
Created On: 22.07.2020
Author: Abhishek Shivarkar
Last Updated On: 21.04.2021

Figure 6: Program structure for identification of MIMO systems

The program structure shows the flow of commands between different files. These files are of two types, Script files (marked in orange colour) and Function files (marked in blue colour).

identification_method_MIMO.m is the main function file that has most of the technical requirements of the program (except graphs and evaluation criteria) incorporated into it. It basically splits the measured data and also holds the code to estimate input and output time delays. If the input parameters are selected separately for each MISO system, then this main function file runs as many times as the number of MISO systems, otherwise it runs just once if these parameters are same across all the outputs. This file saves all the variables in a structure data type known as *results* that enables the user for its efficient extraction. This main file passes on the necessary inputs to another function named *calc_param.m* (that is programmed inside the same main file) to estimate transfer function and simulate the model behaviour.

For plotting measurement and simulation graphs, the *plot_MIMO.m* is used. It has four different subplots that represent the input data (measured and estimated), output data (measured, estimated, and simulated), measured minus simulated output error and lastly the estimated minus simulated output error. The colour coding and the plot line type is maintained throughout to distinguish each plot from one another (refer test results in section 5.2 for the graphs).

The quality of model is evaluated in *quality_criteria_MIMO.m*. The output data (measured and simulated) and the unknown model parameters (true and estimated) are passed here as inputs to get the fit and RR values as outputs. If the true model parameters are unknown, then only fit value is calculated.

The test files (e.g., *test_001.m*) are basically scripts that test the program requirements. They generate artificial measurement data and pass on the input parameters to the main function file to identify the model. The results acquired are then passed as inputs to the plot and quality check function. (refer section 5 for more program testing related details)

Overall, this methodology was followed throughout this programming phase that helped to maintain simple, fast, and efficient programs.

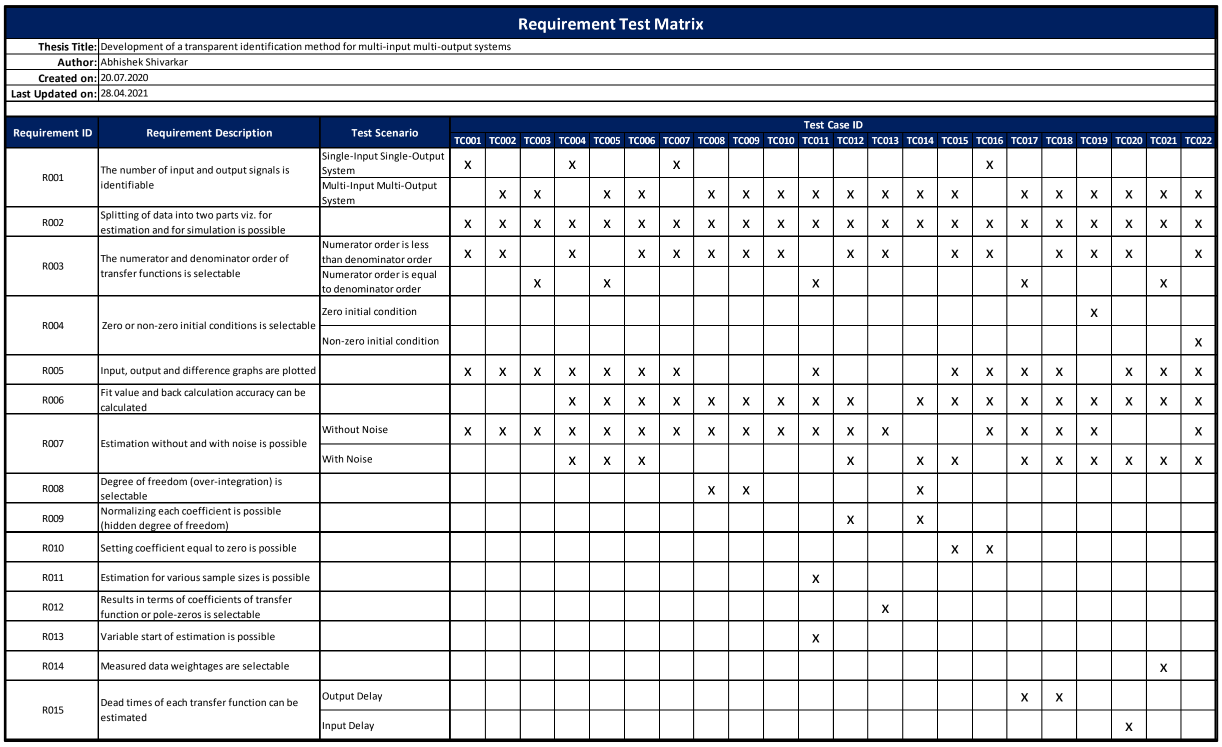
(All the MATLAB built in functions that are directly used are listed in Appendix A at the end of this document.)

5 Identification Program Testing

This section refers only to the tests that are conducted to verify the working of the identification method.

5.1 Testing Philosophy

Once the basic requirements are programmed, the testing phase starts. To perform efficient testing and to maintain track of all the test results, a testing methodology is adopted. Initially, each requirement is allotted an identification number and test scenarios (if available) are mentioned. Each test to be performed is also allotted a test case identification number. The requirements are listed row-wise (vertically), and test cases are listed column-wise (horizontally) to obtain the Requirement-Test Matrix (as shown below).



Requirement Test Matrix

Thesis Title:	Development of a transparent identification method for multi-input multi-output systems
Author:	Abhishek Shivarkar
Created on:	20.07.2020
Last Updated on:	28.04.2021

Requirement ID	Requirement Description	Test Scenario	Test Case ID																					
			TC001	TC002	TC003	TC004	TC005	TC006	TC007	TC008	TC009	TC010	TC011	TC012	TC013	TC014	TC015	TC016	TC017	TC018	TC019	TC020	TC021	TC022
R001	The number of input and output signals is identifiable	Single-Input Single-Output System	X			X			X									X						
		Multi-Input Multi-Output System		X	X		X	X		X	X	X	X	X	X	X	X		X	X	X	X	X	X
R002	Splitting of data into two parts viz. for estimation and for simulation is possible		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
R003	The numerator and denominator order of transfer functions is selectable	Numerator order is less than denominator order	X	X		X		X	X	X	X	X		X	X		X	X		X	X	X		X
		Numerator order is equal to denominator order			X		X						X						X				X	
R004	Zero or non-zero initial conditions is selectable	Zero initial condition																			X			
		Non-zero initial condition																						X
R005	Input, output and difference graphs are plotted		X	X	X	X	X	X	X				X				X	X	X	X		X	X	X
R006	Fit value and back calculation accuracy can be calculated					X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X
R007	Estimation without and with noise is possible	Without Noise	X	X	X	X	X	X	X	X	X	X	X	X	X			X	X	X	X			X
		With Noise				X	X	X						X		X	X		X	X	X	X	X	X
R008	Degree of freedom (over-integration) is selectable									X	X					X								
R009	Normalizing each coefficient is possible (hidden degree of freedom)													X		X								
R010	Setting coefficient equal to zero is possible																X	X						
R011	Estimation for various sample sizes is possible												X											
R012	Results in terms of coefficients of transfer function or pole-zeros is selectable														X									
R013	Variable start of estimation is possible												X											
R014	Measured data weightages are selectable																						X	
R015	Dead times of each transfer function can be estimated	Output Delay																	X	X				
		Input Delay																				X		

Figure 7: Requirement-Test Matrix

After the creation of this matrix, a series of tests are conducted by varying different parameters to verify the basic requirements that are programmed (The first seven tests shown in the matrix above refer to this activity). Once it is completed and the expected results are achieved, additional requirements are programmed one by one. After each requirement is programmed, new test cases are created for testing. A requirement might need one or more than one tests depending on the number of systems to be identified and the combination of input parameters that are selected. After each test, the requirements that are tested are marked with a cross in corresponding boxes of the matrix. It can be seen that the basic requirements being fundamental to the identification program get tested again and again with the additional requirements.

All these tests cases, follow the same principle that a system with known transfer function is used to artificially generate measurement data. With the help of this data, the parameters of this system are reidentified and a comparison is made to check the identification quality.

As mentioned in the design philosophy, after each modification in the program, it is ensured that the previous test files and the functionalities still run and provide the same results. For this purpose, a test-error maintenance is carried out. An error number is defined for the test cases and is recorded for each test. This error number is defined as:

$$Error\ number = 1 - \frac{Fit\ in\ \%}{100} \tag{51}$$

For tests, where multiple models are identified, the lowest of all the fit values is taken to calculate the error number. The smaller the number, more better is the identification. So, when the main program is modified for additional requirements, the old tests are run again, and the new error number is cross-checked with the previously recorded one to verify the results. If not the same, this error number should not be greater than the previous one after program modifications are done. Because a greater error number suggests a lower identification quality. Therefore, this helps to improve or at the least maintain the quality of the identification program after programming of each requirement.

Following table shows the error number for the respective test cases. An empty space suggests that the fit value was not calculated in that test.

Test Cases	Error Number	Test Cases	Error Number
001	-	012	0.471
002	-	013	-
003	-	014	0.432
004	0.035	015	0.095
005	0.020	016	0.011
006	0.025	017	0.023
007	0.063	018	0.029
008	0.032	019	0.095
009	0.337	020	0.027
010	0.071	021	0.112
011	0.016	022	0.034

Table 1: Error number for various test cases

5.2 Test Results

5.2.1 Single-Input Single-Output System

A test is performed to identify a single-input single-output system once the basic requirements are programmed. For the same, a system with input (u) and output signal (y) as shown in figure below is taken into consideration. The transfer function G of numerator order 0 and denominator order 2 is arbitrarily chosen to generate artificial measurement data. The values of these transfer functions can be referred in the figure below. The step size for generating data is selected as 0.01. In this case no noise component is added to the generated data.

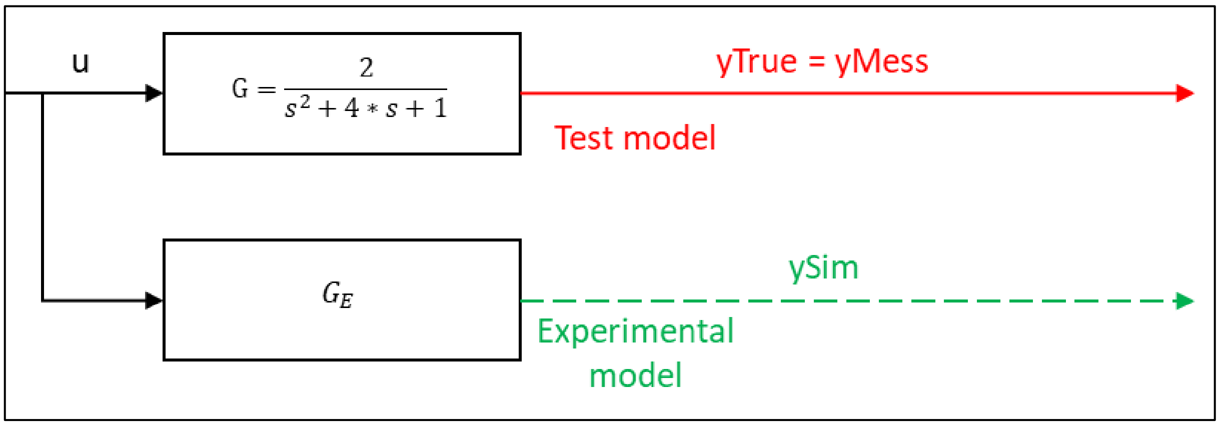


Figure 8: Signals and systems for testing

The same model order and estimation length from 0 *sec* to 7 *sec* is selected for identification. Here, the idea is to test the fundamental requirements i.e., if the system identifies number of input and output signals, can split the data into estimation and simulation, if it estimates the coefficients of transfer function as per the selected model order, evaluates the quality criteria and plots the required graphs. After identification, the estimated coefficients are obtained as shown below:

Parameter	True Values	Estimated Values
a_1	2	2.0039
b_1	4	4.0050
b_0	1	0.9936

Table 2: Comparison of true and estimated coefficients for SISO system

From the comparison table it is clear that, the system has correctly identified the number of input and output signals. For the selected model order, it has also identified the coefficients of the transfer function with an RR value of 99.80 % (almost 100 %).

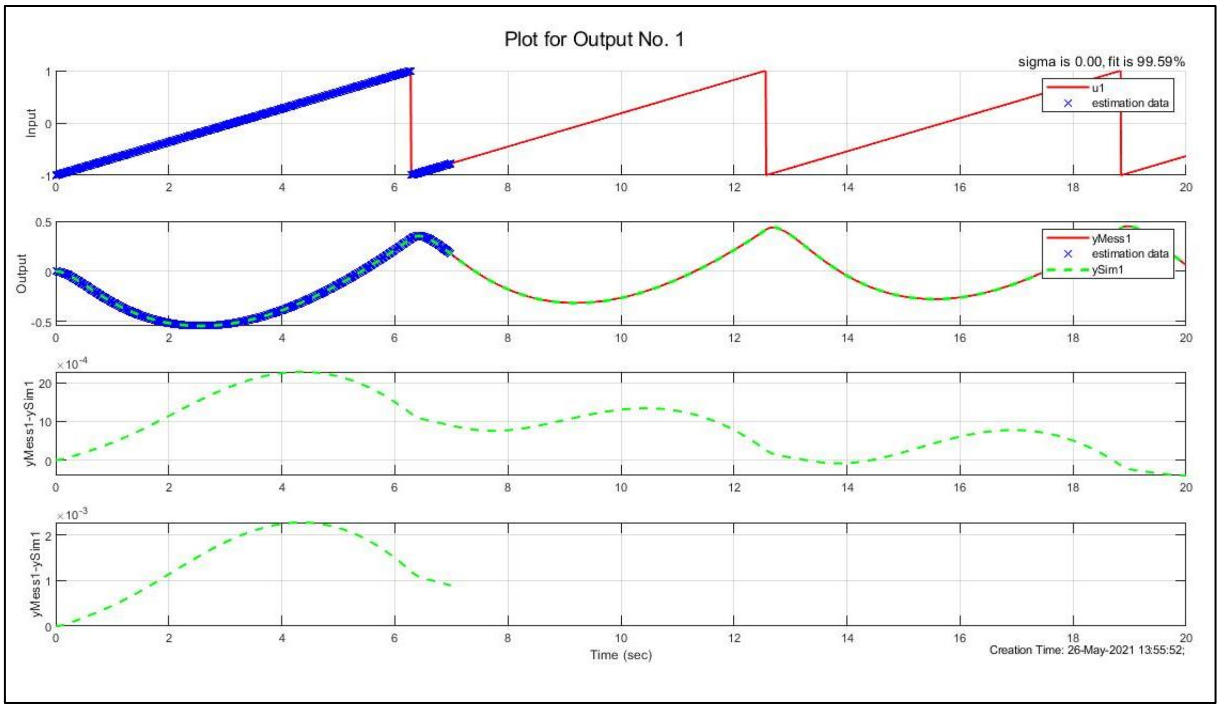


Figure 9: Measurement and simulation plots with identified transfer function

The graph shown above has four subplots. The first two represent the input signals and the output signals (both measured and simulated) respectively, while the last two represent the output error. The data used for estimation is marked in blue crosses. It can be verified from the plot that first 7 sec are used for estimation. The model fit is 99.59 % (almost 100 %).

Therefore, at no noise, the coefficients are recalculated with high accuracy and simulated response has almost no deviations. This also verifies the basic requirements included in the identification program.

5.2.2 Non-Zero Initial Conditions

The program here is tested for a system with measurements that does not start in equilibrium position. For the same, a two-input two-output system (u_1, u_2, y_1, y_2) as shown in figure below is taken into consideration. $G1$, $G2$, $G3$ and $G4$ are the transfer functions of numerator order 0 and denominator order 3 that are arbitrarily chosen to generate artificial measurement data. The values of these transfer functions can be referred in the figure below. Additionally, the following arbitrary initial conditions are also selected:

$$[y_1(0) \quad \dot{y}_1(0) \quad \ddot{y}_1(0)] = [1 \quad 0.04 \quad -0.2] \tag{52}$$

$$[y_2(0) \quad \dot{y}_2(0) \quad \ddot{y}_2(0)] = [0.5 \quad -0.15 \quad 0.3] \tag{53}$$

The step size for generating data is selected as 0.01. To simulate measurement errors, normally distributed random numbers with standard deviation from 0.00 to 0.05 in steps of 0.01 are added to the artificially generated measurement data.

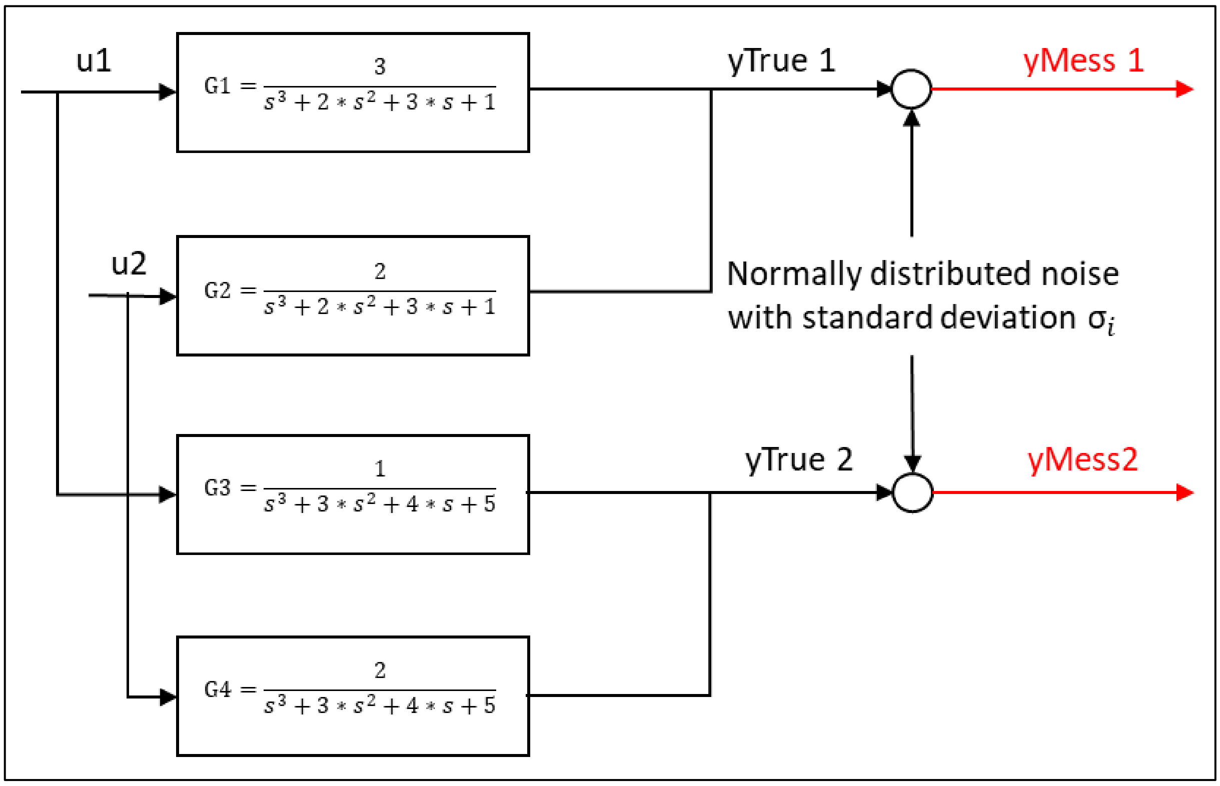

u1
u2
$G1 = \frac{3}{s^3 + 2 * s^2 + 3 * s + 1}$
$G2 = \frac{2}{s^3 + 2 * s^2 + 3 * s + 1}$
$G3 = \frac{1}{s^3 + 3 * s^2 + 4 * s + 5}$
$G4 = \frac{2}{s^3 + 3 * s^2 + 4 * s + 5}$
yTrue 1
yMess 1
Normally distributed noise with standard deviation σ_i
yTrue 2
yMess2

Figure 10: Signals and system for artificially generated measurement data

Now, the idea here is to find the coefficients of the transfer functions for the experimental model shown in figure below as well as to make the simulated response of the system as close to the output measurement data by identification procedure developed in this thesis. The same model order is selected for the identification program.

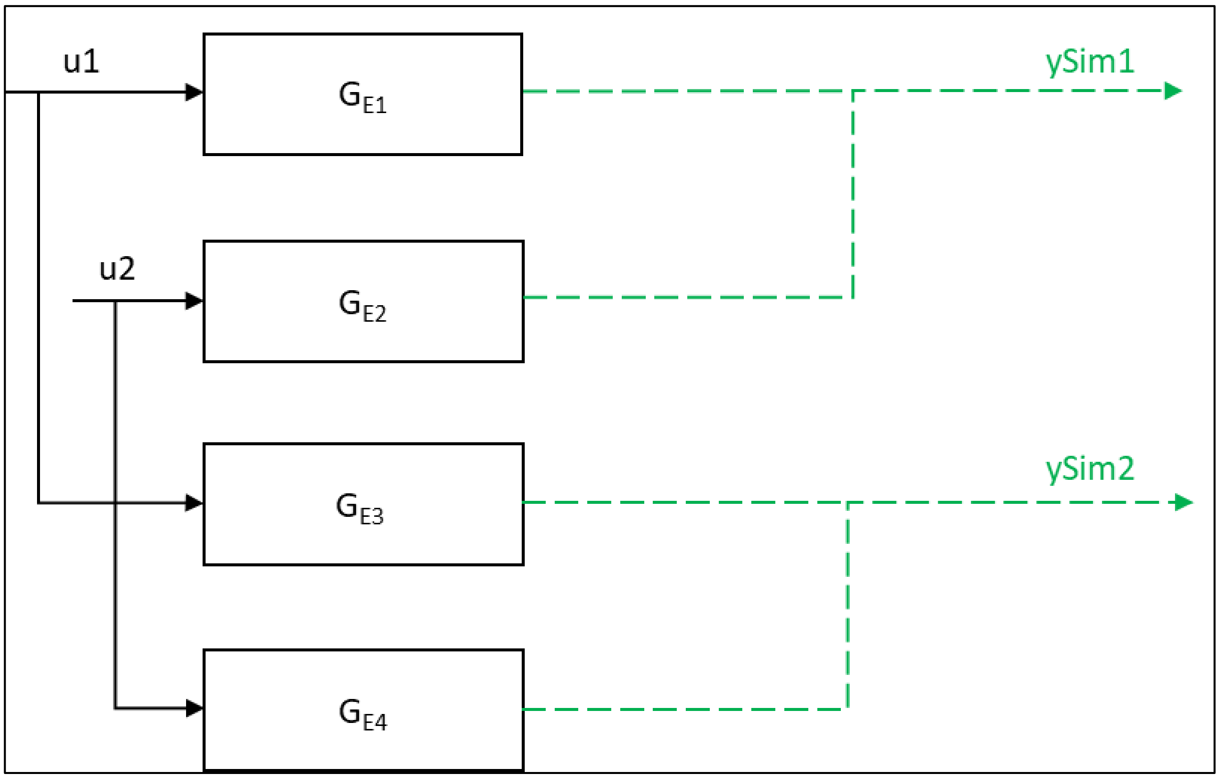


Figure 11: Experimental model for recalculation of transfer functions

Here the measurement and simulation plot obtained for output 1 for a standard deviation value of 0.05 is only shown:

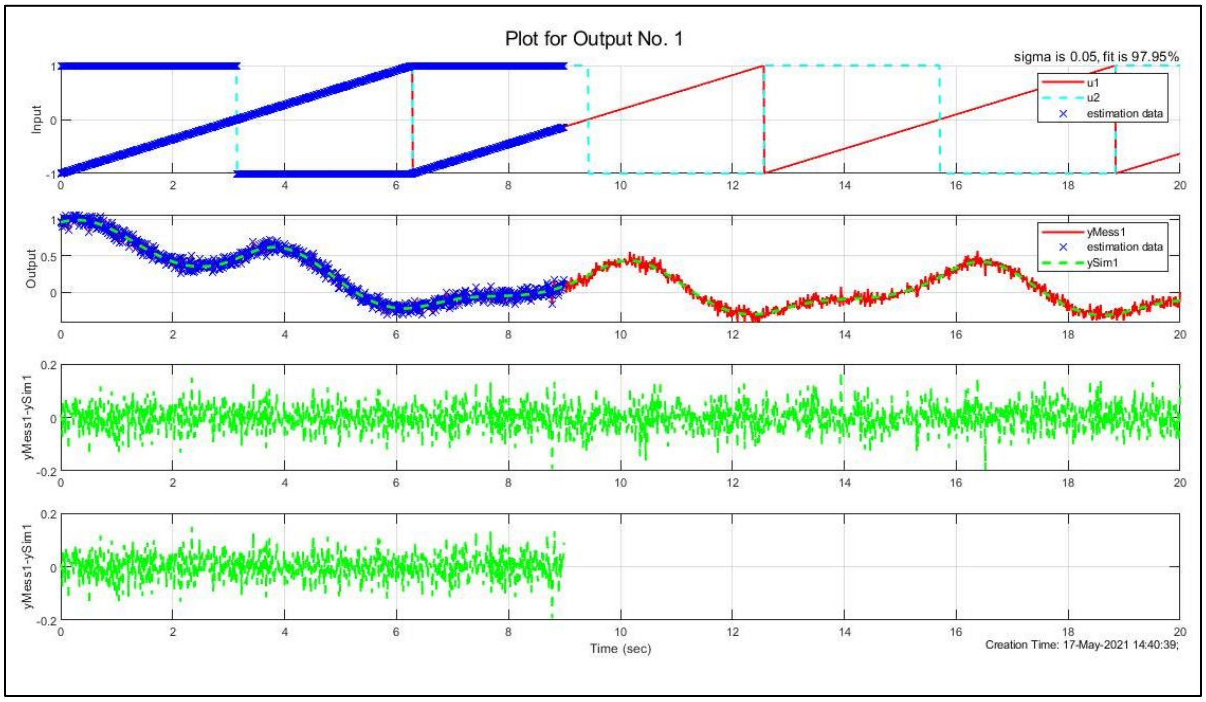


Figure 12: Measurement and simulation plots with identified transfer function

Following table shows a comparison between the true and estimated coefficients of the transfer functions and initial conditions for output 1:

Parameter	True Values	Estimated values with noise component σ					
		$\sigma = 0.00$	$\sigma = 0.01$	$\sigma = 0.02$	$\sigma = 0.03$	$\sigma = 0.04$	$\sigma = 0.05$
a_1	3	2.9929	2.9950	2.9970	2.9988	3.0005	3.0019
a_2	2	1.9918	1.9990	2.0061	2.0132	2.0201	2.0270
b_2	2	1.9881	1.9934	1.9986	2.0037	2.0086	2.0135
b_1	3	2.9741	2.9945	3.0149	3.0353	3.0555	3.0757
b_0	1	0.9899	0.9958	1.0018	1.0078	1.0138	1.0198
$\dot{y}_1(0)$	0.04	0.0397	0.0683	0.0970	0.1256	0.1544	0.1831
$\ddot{y}_1(0)$	-0.2	-0.2011	-0.2513	-0.3015	-0.3518	-0.4022	-0.4525

Table 3: Comparison of true and estimated model parameters at different levels of noise

Following figure shows the comparison of fit and RR value at different standard deviations of noise for output 1:

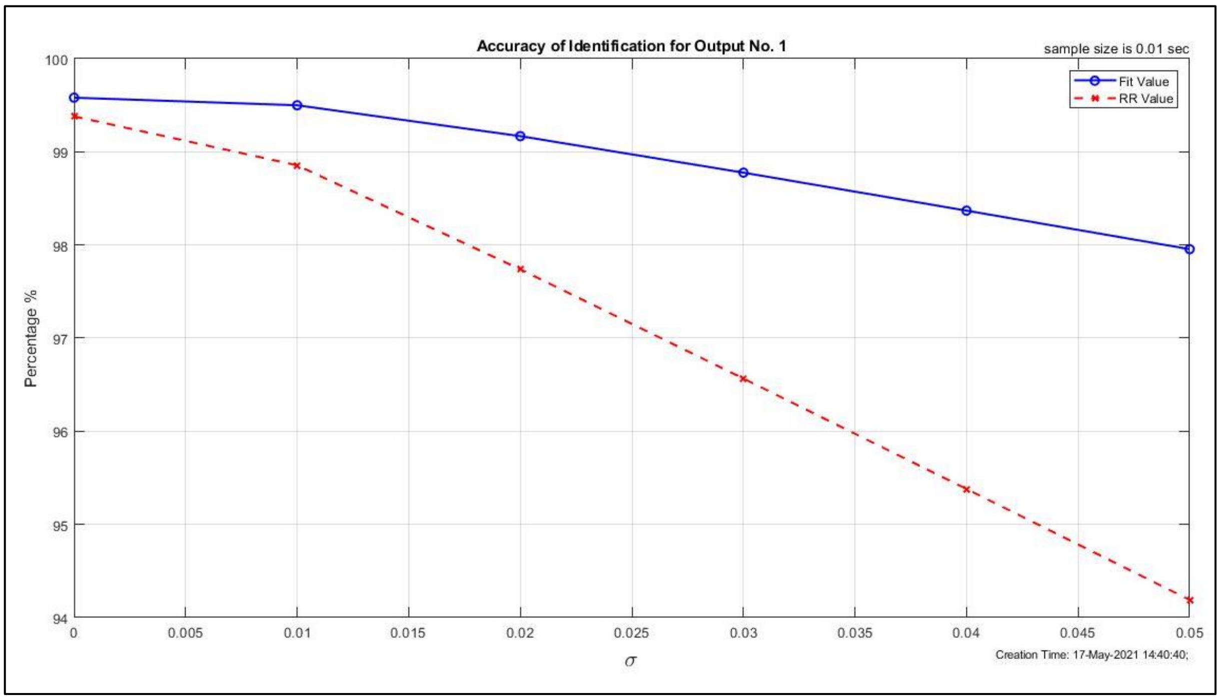


Figure 13: Accuracy comparison at different levels of noise

At standard deviation of 0.05 the fit value obatined is 97.95 % and RR value is 94.18 %. It is seen that with increase in magnitude of noise component, the accuracy decreases. Both the fit value and RR value seems to be very good even for higher noise. Therefore, this means that for noisy measurements the transfer functions obtained is numerically close. Further, a considerably good simulation with only slight deviations is obtained and the non-zero starting conditions are also estimated.

Similar results are generated for second output as shown below:

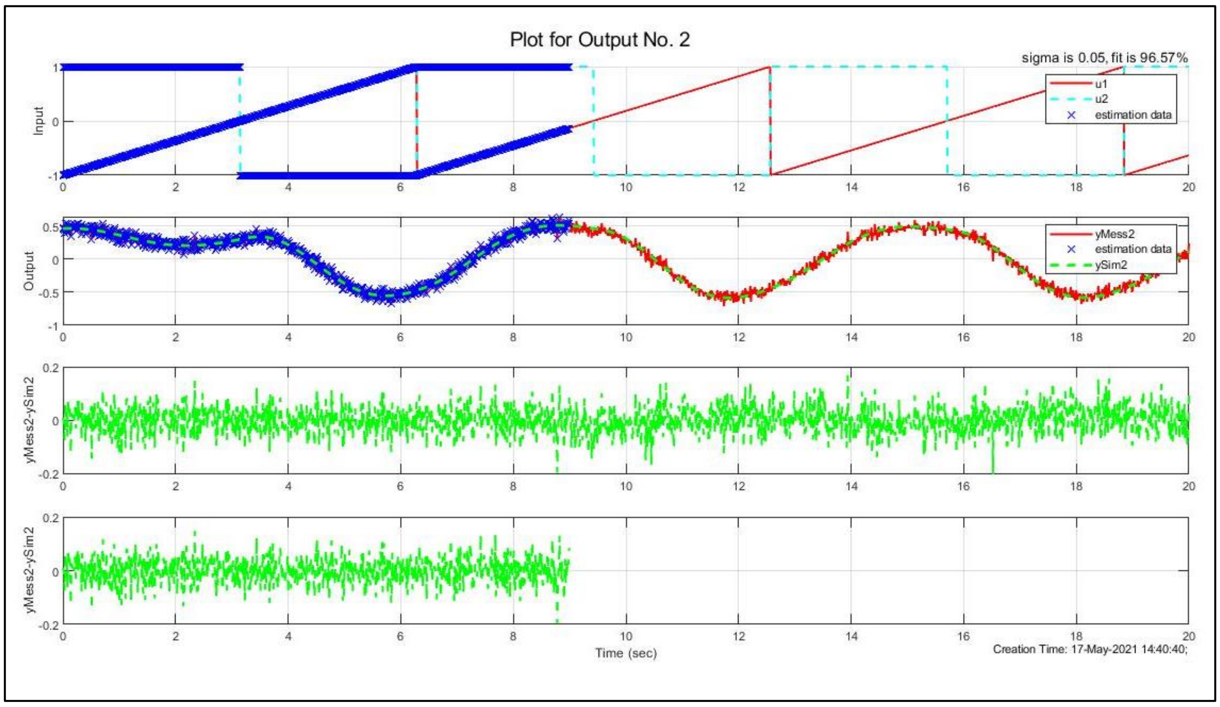


Figure 14: Measurement and simulation plots with identified transfer function

Parameter	True Values	Estimated values with noise component σ					
		$\sigma = 0.00$	$\sigma = 0.01$	$\sigma = 0.02$	$\sigma = 0.03$	$\sigma = 0.04$	$\sigma = 0.05$
a_1	1	0.9994	1.0410	1.0856	1.1333	1.1843	1.2387
a_2	2	1.9796	2.0357	2.0941	2.1548	2.2179	2.2834
b_2	3	2.9721	3.0418	3.1141	3.1889	3.2664	3.3467
b_1	4	3.9674	4.0196	4.0723	4.1253	4.1788	4.2327
b_0	5	4.9310	5.0316	5.1350	5.2414	5.3507	5.4631
$\dot{y}_2(0)$	-0.15	-0.1490	-0.1175	-0.0854	-0.0526	-0.0192	0.0149
$\ddot{y}_2(0)$	0.3	0.2913	0.2246	0.1535	0.0776	-0.0034	-0.0902

Table 4: Comparison of true and estimated model parameters at different levels of noise

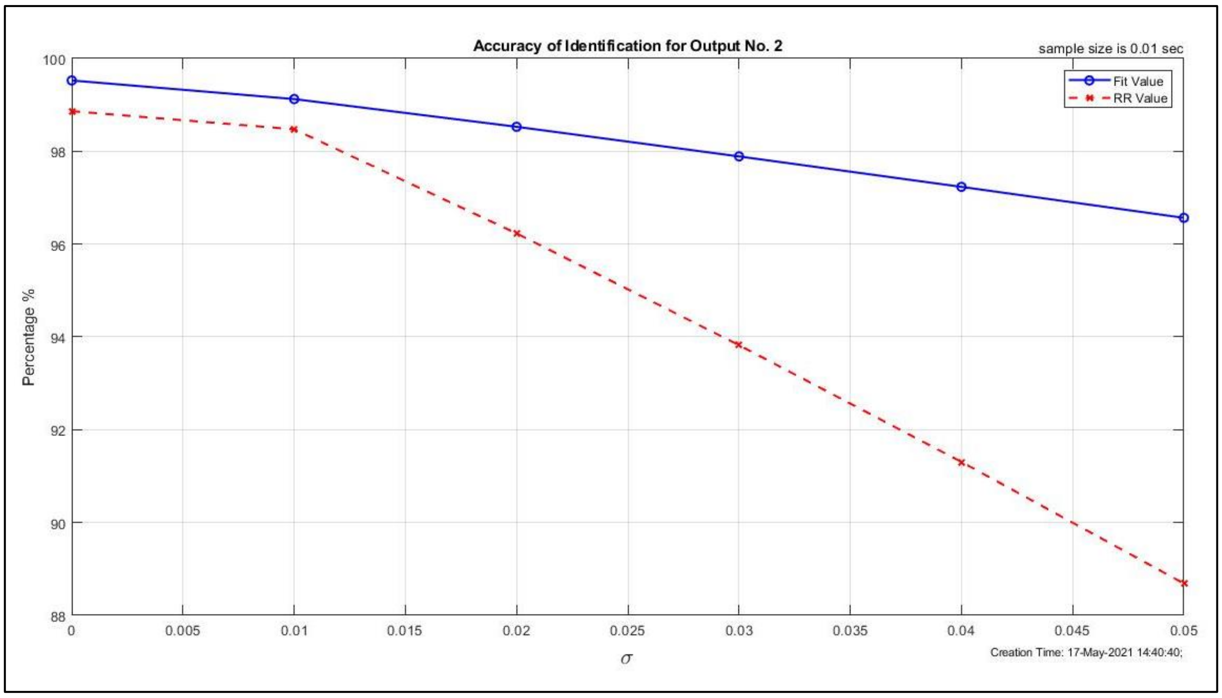


Figure 15: Accuracy comparison at different levels of noise

This verifies the identification program for estimating non-zero initial conditions with noisy measurement data for a multi-input multi-output system.

5.2.3 Over-Integration

The program here is tested for over-integration (refer section 3.3.1 for mathematical description). The same two-input two-output system (u_1, u_2, y_1, y_2) as the previous test (shown in figure 10), with the same transfer functions $G1$, $G2$, $G3$ and $G4$ of numerator order 0 and denominator order 3 are arbitrarily chosen to generate artificial measurement data. The step size for generating data is selected as 0.01. To simulate measurement errors, normally distributed random numbers with standard deviation from 0.00 to 0.05 in steps of 0.01 are added to the artificially generated measurement data.

The same model order is selected for identification and the results without and with over-integration (up to factor 2) are compared. Here to effectively compare the results between the combination of different standard deviation values and over-integration factors, the measurement and simulation plots are avoided and only an accuracy comparison based on the evaluation criteria is made. Moreover, from hereon, result for only one of the outputs is shown since the identification is independent of each other.

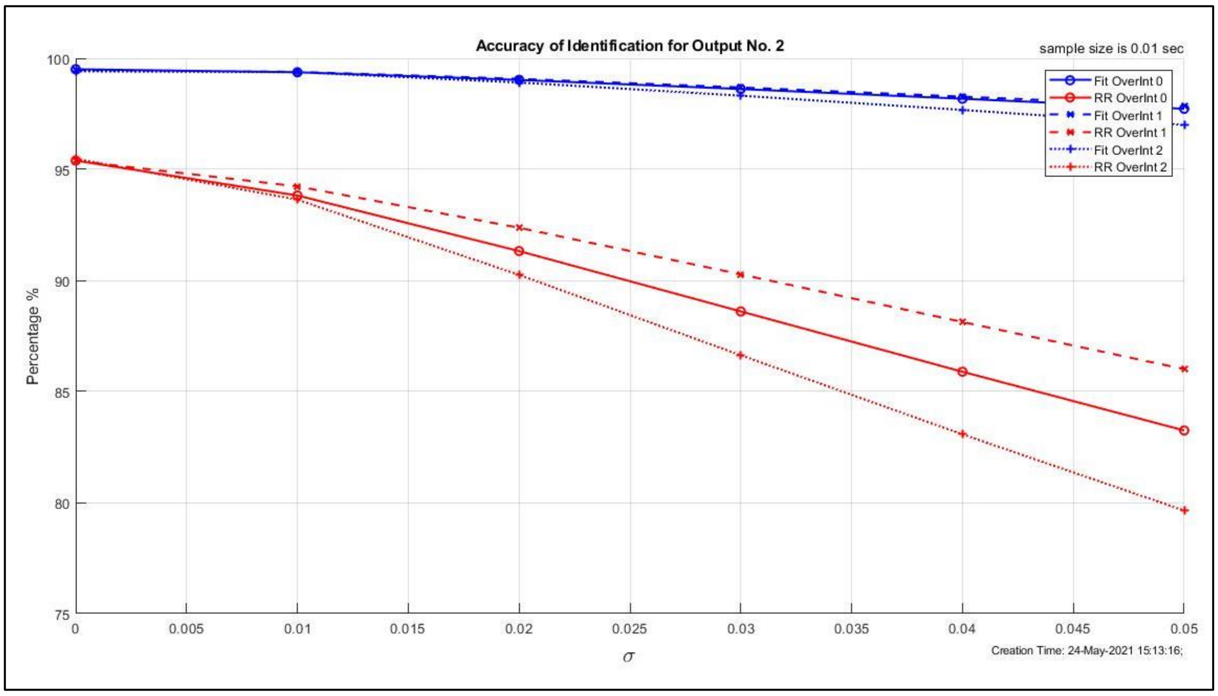


Figure 16: Accuracy comparison for different noise levels and over-integration factors

It is seen here that with over-integration factor of 1 the accuracy increases for the corresponding standard deviations (shown by dashed line). This effect may happen because numerical integration makes the simulated output signal more deterministic and less sensitive to linear dependence. On the contrary over-integrating it further decreases the accuracy (shown by dotted line). The cause for this may be the unavoidable numerical integration errors. Hence, over-integrating the signals may or may not improve the accuracy for various systems. But the system model can always be checked for over-integration and the best possible solution could be chosen from the available ones to improve the model quality.

5.2.4 Normalizing Different Coefficients

The program tested here is for normalizing different coefficients of denominator (refer section 3.3.2 for mathematical description). A two-input one-output system (u_1, u_2, y_1) with the transfer functions $G1$ and $G2$ of numerator order 0 and denominator order 3 is arbitrarily chosen to generate artificial measurement data. Also, these transfer functions have the coefficient of s^3 normalized. The true values of coefficients are provided in the comparison table later. The step size for generating data is selected as 0.001. To simulate measurement errors, normally distributed random numbers with standard deviation from 0.00 to 0.05 in steps of 0.01 are added to the artificially generated measurement data.

The same respective model order is selected for identification. All the four denominator coefficients are normalized one by one in the program for comparison of identification results.

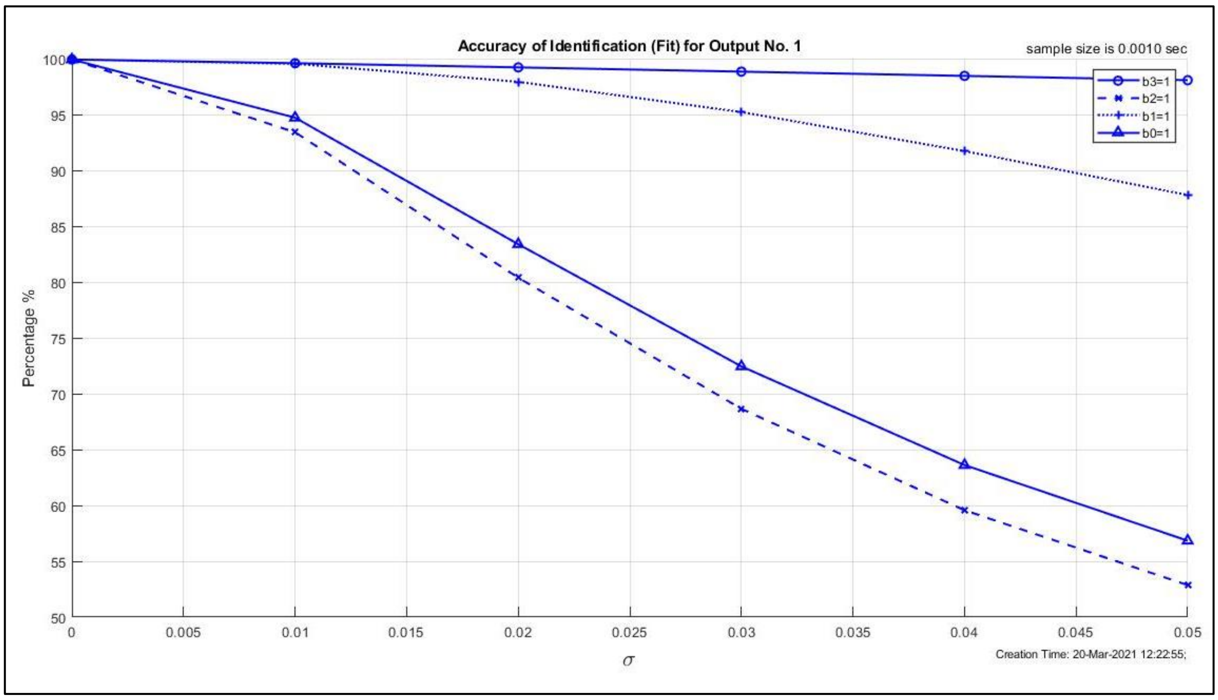


Figure 17: Accuracy comparison for different noise levels and normalizing each denominator coefficient

At no noise, the fit value is same for different normalization coefficients. But as the noise level increases the fit quality differs and that means simulated response changes as per the normalization of different coefficients. This is due to the modelling errors and unavoidable numerical integration errors in the input and output signals that gives slightly different over-determined system of equations. Also, it can be noted that for noisy measurement data, normalization of the highest order denominator coefficient gives the best possible fit quality compared to others (even when the true transfer function used to generate artificial measurement data is normalized by any coefficient in the denominator).

Here, accuracy in terms of RR value is avoided as it is already known that the coefficients calculated are not numerically same because of different normalizations. Following table compares the estimated coefficient values for normalization of different denominator coefficients at zero standard deviation:

Parameter	True Values	Estimated values with noise component $\sigma = 0.00$ and normalized coefficient			
		$b_3 = 1$	$b_2 = 1$	$b_1 = 1$	$b_0 = 1$
a_1	3	2.9974	1.5007	0.9995	0.7505
a_2	2	1.9981	1.0004	0.6663	0.5003
b_3	1 (norm.)	1	0.5007	0.3335	0.2504
b_2	2	1.9974	1	0.6661	0.5001
b_1	3	2.9989	1.5014	1	0.7508
b_0	4	3.9941	1.9996	1.3319	1

Table 5: Comparison of true and estimated model parameters for normalizing each denominator coefficient

If the estimated values in column $b_2 = 1$ are normalized with coefficient of b_3 (here, 0.5007) then the values obtained are numerically close to column $b_3 = 1$. And the same follows for other columns. This verifies the identification program for normalizing different denominator coefficients.

5.2.5 Setting Coefficients to Zero

The program tested here is for setting different coefficients to zero (refer section 3.3.3 for mathematical description). A two-input two-output system (u_1, u_2, y_1, y_2) with the transfer functions $G1$, $G2$, $G3$ and $G4$ (as shown in figure below), of numerator order 2 and denominator order 3 is arbitrarily chosen and some of the coefficients are arbitrarily set to zero to generate artificial measurement data. The step size for generating data is selected as 0.01. To simulate measurement errors, normally distributed random numbers with standard deviation of 0.05 are added to the artificially generated measurement data.

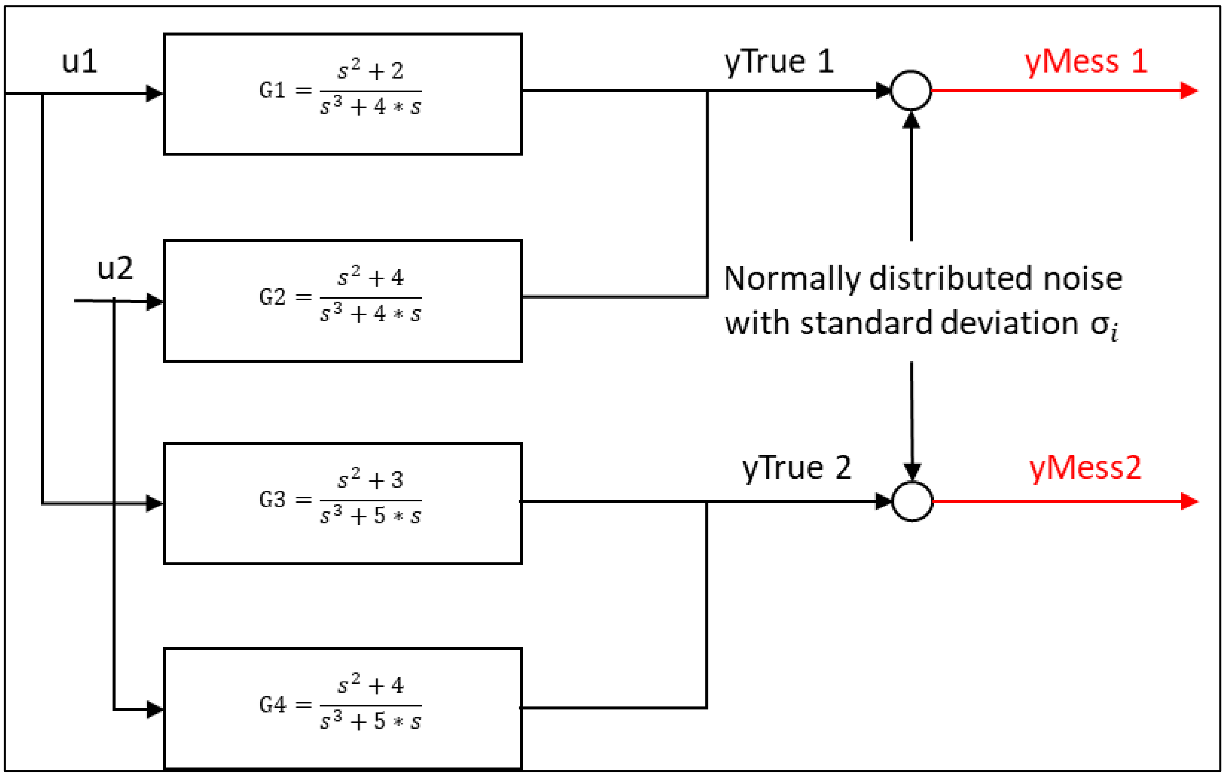

u1
$G1 = \frac{s^2+2}{s^3+4*s}$
yTrue 1
yMess 1
u2
$G2 = \frac{s^2+4}{s^3+4*s}$
Normally distributed noise with standard deviation σ_i
$G3 = \frac{s^2+3}{s^3+5*s}$
yTrue 2
yMess2
$G4 = \frac{s^2+4}{s^3+5*s}$

Figure 18: Signals and system for artificially generated measurement data by setting some coefficients to zero

As shown in figure above, to compare and verify, the same model order is selected, and the same coefficients are pre-set to zero for the identification process. The remaining unknown coefficients are estimated, and simulated response is plotted.

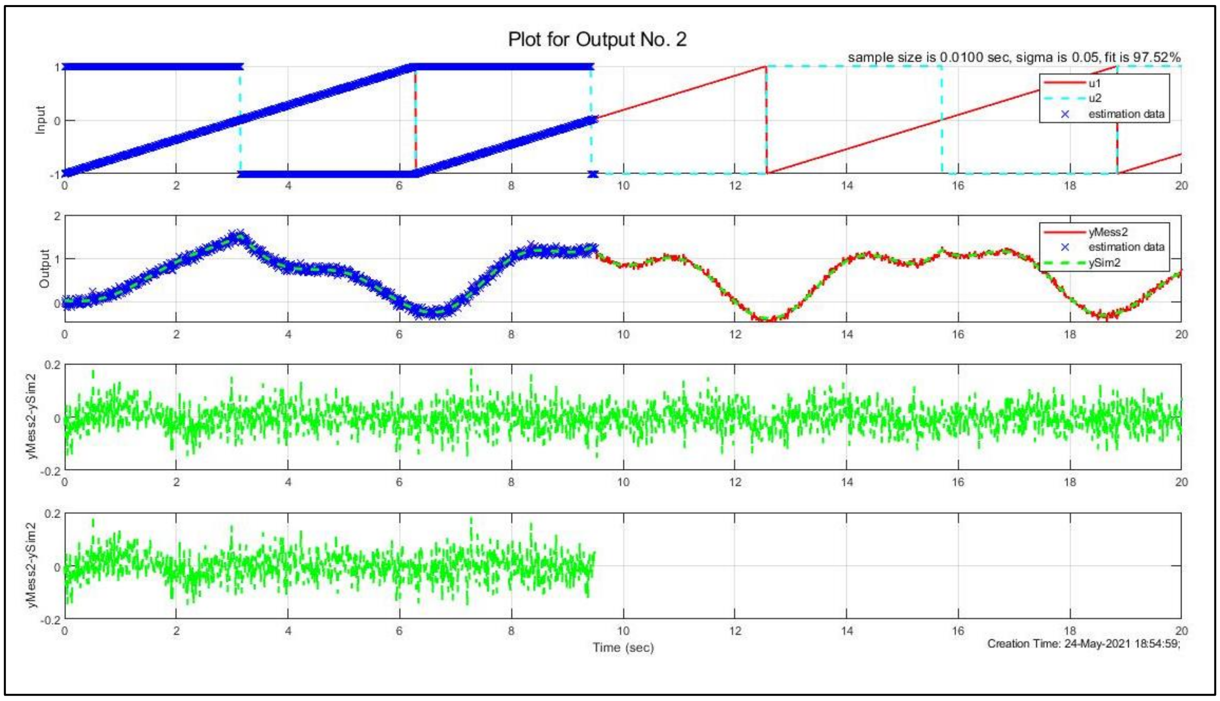


Figure 19: Measurement and simulation plots with identified transfer function by setting some coefficients to zero

A good fit value of 97.52 % and RR value of 96.21 % is obtained. Following table compares the true and estimated coefficients of the identified transfer function:

Parameter	True Values	Estimated values with noise component $\sigma = 0.05$
a_1	3	3.2532
a_2	4	4.0836
a_5	1	1.0245
a_6	1	0.9868
b_1	5	4.9480

Table 6: Comparison of true and estimated coefficients by setting some coefficients to zero

Similarly, by setting the coefficients corresponding to s^0 to zero, integrators and differentiators can be achieved in a transfer function.

5.2.6 Various Sample Sizes

The program here is tested by changing the sample size of estimated data range. The same single-input single-output system (u, y) (as shown in figure 8), with the same transfer function G of numerator order 0 and denominator order 2 is arbitrarily chosen to generate artificial measurement data. The step size for generating data is selected as 0.0001. No noise is added to this artificially generated measurement data.

Same model order is selected for identification. The sample size for estimation is selected as:

$$h = [0.0001 \quad 0.001 \quad 0.01 \quad 0.1] \tag{54}$$

This is achieved by selecting every n^{th} data for estimation where:

$$n = [1 \quad 10 \quad 100 \quad 1000] \tag{55}$$

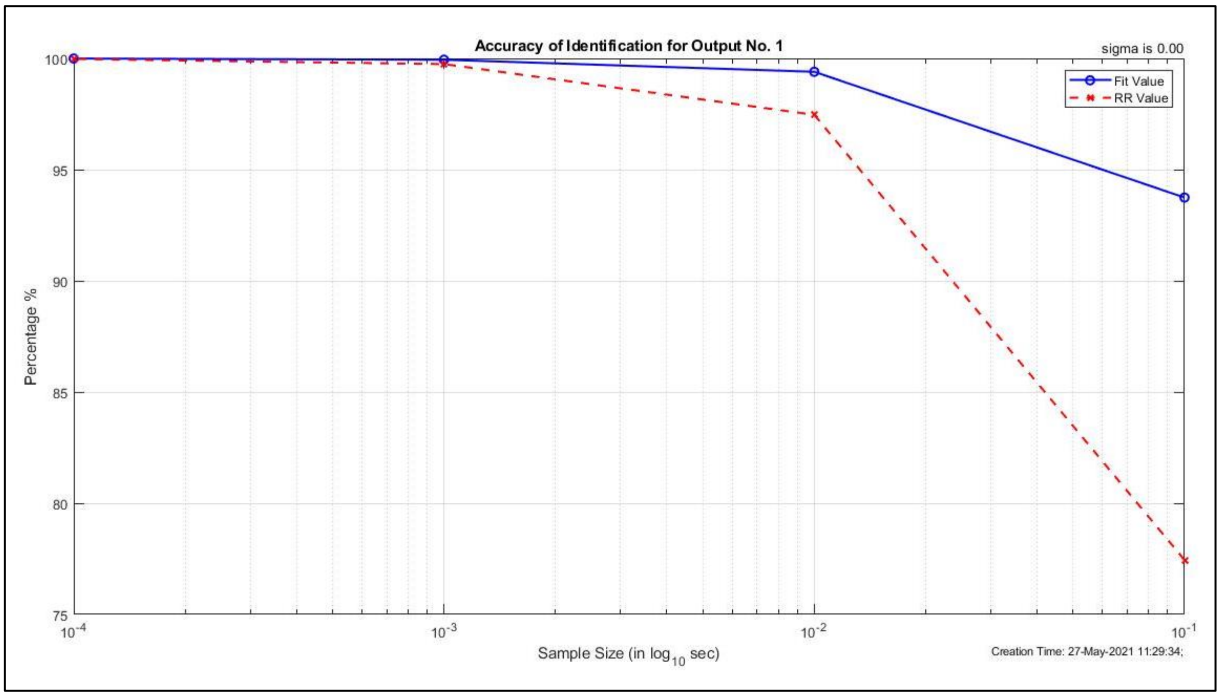


Figure 20: Accuracy for different sample sizes for estimation

Selecting n^{th} data among available measurements is in fact omission of some data points. This results in loss of some information that negatively affects the identification quality. This effect is seen in the figure above, where increasing sample size reduces the accuracy of identification.

The numerical errors that are present, sometimes prevent base matrix inversion. This process of selecting n^{th} data, also known as thinning-out data, reduces these numerical errors. Therefore, in such cases a solution can be obtained by thinning-out the data and compensating the model quality.

5.2.7 Result in Transfer Function or Poles-Zeros

The program tested here is to obtain results in terms of transfer function and poles-zeros. A two-input two-output system (u_1, u_2, y_1, y_2) with the transfer functions $G1$, $G2$, $G3$ and $G4$ (as shown in figure below), of numerator order 1 and denominator order 3 is arbitrarily chosen to generate artificial measurement data. The step size for generating data is selected as 0.01. No noise is added to this artificially generated measurement data.

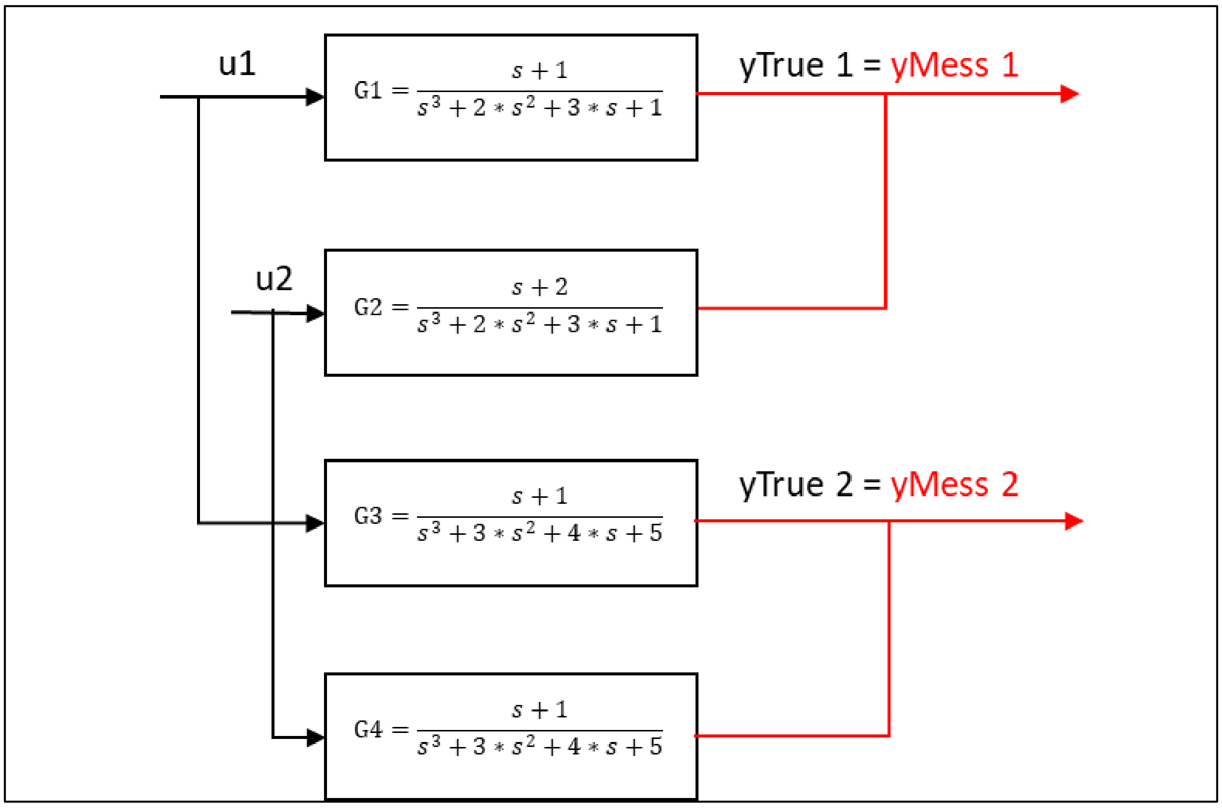
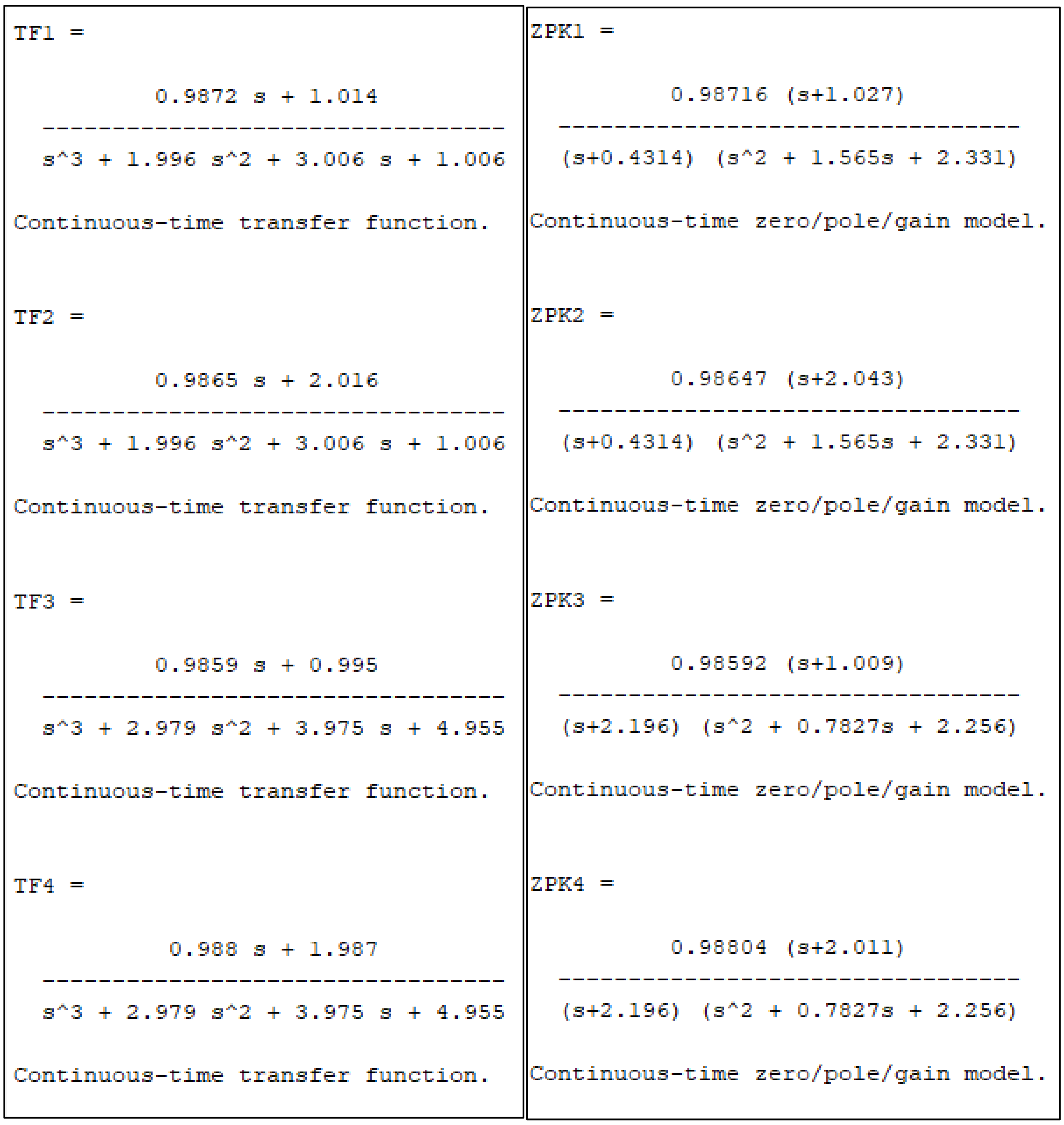

u1
u2
$G1 = \frac{s+1}{s^3+2*s^2+3*s+1}$
$G2 = \frac{s+2}{s^3+2*s^2+3*s+1}$
$G3 = \frac{s+1}{s^3+3*s^2+4*s+5}$
$G4 = \frac{s+1}{s^3+3*s^2+4*s+5}$
yTrue 1 = yMess 1
yTrue 2 = yMess 2

Figure 21: Signals and system for artificially generated measurement data to obtain result in transfer function and poles-zeros

The same model order is selected for identification. Input parameter is set as 0 to obtain the results in transfer function (polynomial form) or 1 to obtain it in poles-zeros (factorized form). A direct comparison is made between the results obtained in both the forms in the following figure:



```
TF1 =

          0.9872 s + 1.014
  ---------------------------------
  s^3 + 1.996 s^2 + 3.006 s + 1.006

Continuous-time transfer function.


TF2 =

          0.9865 s + 2.016
  ---------------------------------
  s^3 + 1.996 s^2 + 3.006 s + 1.006

Continuous-time transfer function.


TF3 =

          0.9859 s + 0.995
  ---------------------------------
  s^3 + 2.979 s^2 + 3.975 s + 4.955

Continuous-time transfer function.


TF4 =

           0.988 s + 1.987
  ---------------------------------
  s^3 + 2.979 s^2 + 3.975 s + 4.955

Continuous-time transfer function.
```

```
ZPK1 =

         0.98716 (s+1.027)
  ---------------------------------
  (s+0.4314) (s^2 + 1.565s + 2.331)

Continuous-time zero/pole/gain model.


ZPK2 =

         0.98647 (s+2.043)
  ---------------------------------
  (s+0.4314) (s^2 + 1.565s + 2.331)

Continuous-time zero/pole/gain model.


ZPK3 =

         0.98592 (s+1.009)
  ---------------------------------
  (s+2.196) (s^2 + 0.7827s + 2.256)

Continuous-time zero/pole/gain model.


ZPK4 =

         0.98804 (s+2.011)
  ---------------------------------
  (s+2.196) (s^2 + 0.7827s + 2.256)

Continuous-time zero/pole/gain model.
```

Figure 22: Resultant transfer functions obtained in both the forms

Therefore, the identification procedure is tested to provide the results in the polynomial as well as poles-zeros form of transfer function. These are also directly checked with the actual transfer function used to generate measurement data.

5.2.8 Different (Non-Zero) Start of Estimation Time

The program tested here is for a non-zero start of estimation. A two-input two-output system (u_1, u_2, y_1, y_2) with the transfer functions $G1$, $G2$, $G3$ and $G4$ (as shown in figure below), of numerator order 1 and denominator order 1 is arbitrarily chosen to generate artificial measurement data. The step size for generating data is selected as 0.01. To simulate measurement errors, normally distributed random numbers with standard deviation of 0.05 are added to the artificially generated measurement data.

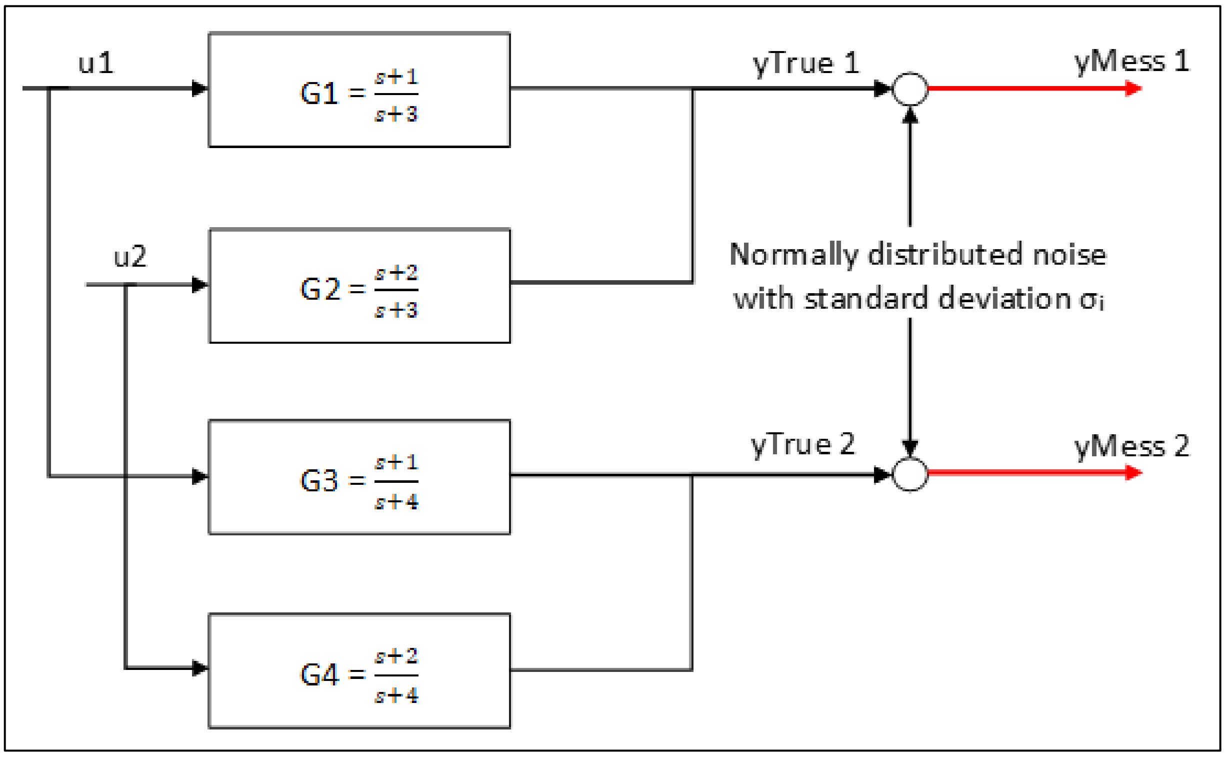


Figure 23: Signals and system for artificially generated measurement data for different (non-zero) estimation start

An arbitrary start (4.37 sec) and end (12.58 sec) for estimation time is selected, and the system is identified for the same model order. The following figure shows the measurement and simulation plot for output 1 of the identified model:

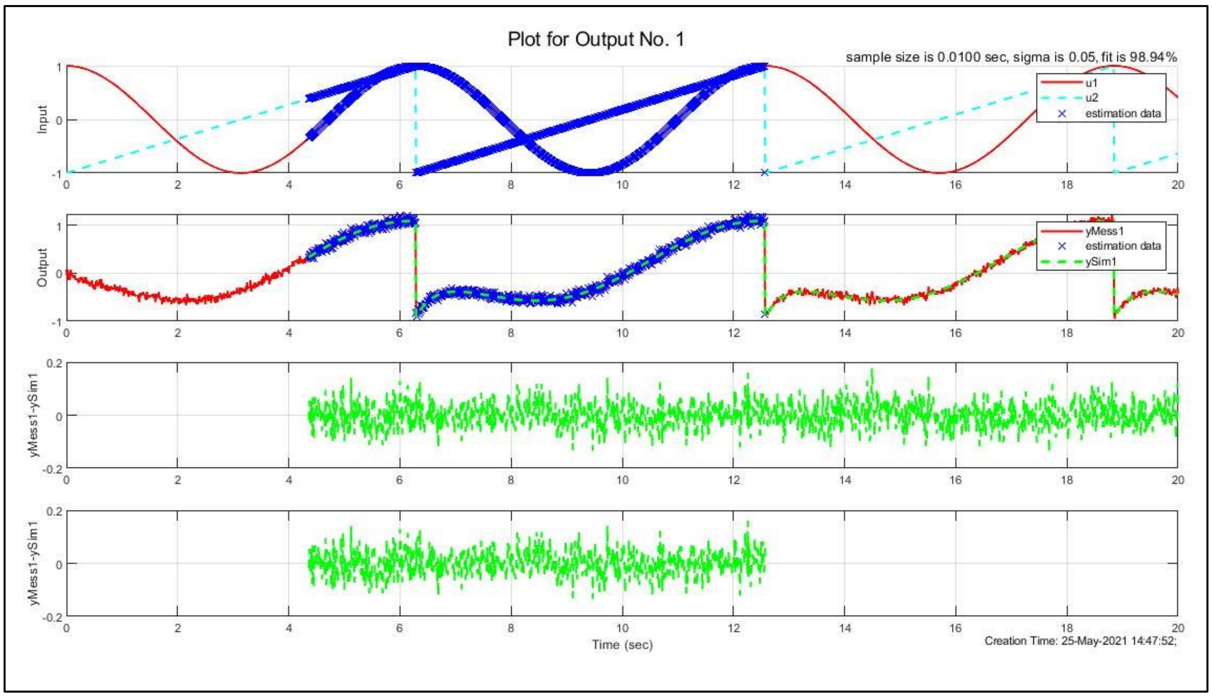


Figure 24: Measurement and simulation plots with identified transfer function for different (non-zero) estimation start

Parameter	True Values	Estimated values with noise component $\sigma = 0.05$
a_1	1	1.0257
a_2	2	2.0257
a_3	1	1.0705
a_4	1	1.0028
b_0	3	3.1099

Table 7: Comparison of true and estimated coefficients for different (non-zero) estimation start

The fit and RR value of 98.94 % and 96.61 % respectively shows that even when a different estimation data set is selected from the measurement data, the system can be modelled and simulated with a high quality. Sometimes when a part of the measured data does not have more dynamics i.e., the input or the output signals or their derivatives assume steady value, then corresponding columns in the base matrix become more and more similar and coefficients cannot be calculated. In such cases this functionality is useful to select a different measuring range for estimation and calculating the coefficients.

5.2.9 Adding Weights

A two-input two-output system (u_1, u_2, y_1, y_2) with the transfer functions $G1$, $G2$, $G3$ and $G4$ (as shown in figure 23), of numerator order 1 and denominator order 1 is arbitrarily chosen to generate artificial measurement data. The step size for generating data is selected as 0.01. To simulate measurement errors, normally distributed random numbers with standard deviation of 0.03 are added to the artificially generated measurement data.

The data is normally generated by applying equal weightage value to all the measurements. But for testing of this program, the identification is done by adding same weights to the measurements from time $4\,sec$ to $8\,sec$ (this is chosen arbitrarily). Following are the five different weight values that are taken one at a time for this specific time range:

$$W = [0.01 \quad 0.5 \quad 1 \quad 3 \quad 10] \tag{56}$$

A comparison is made between the model quality to verify it with the expected behaviour:

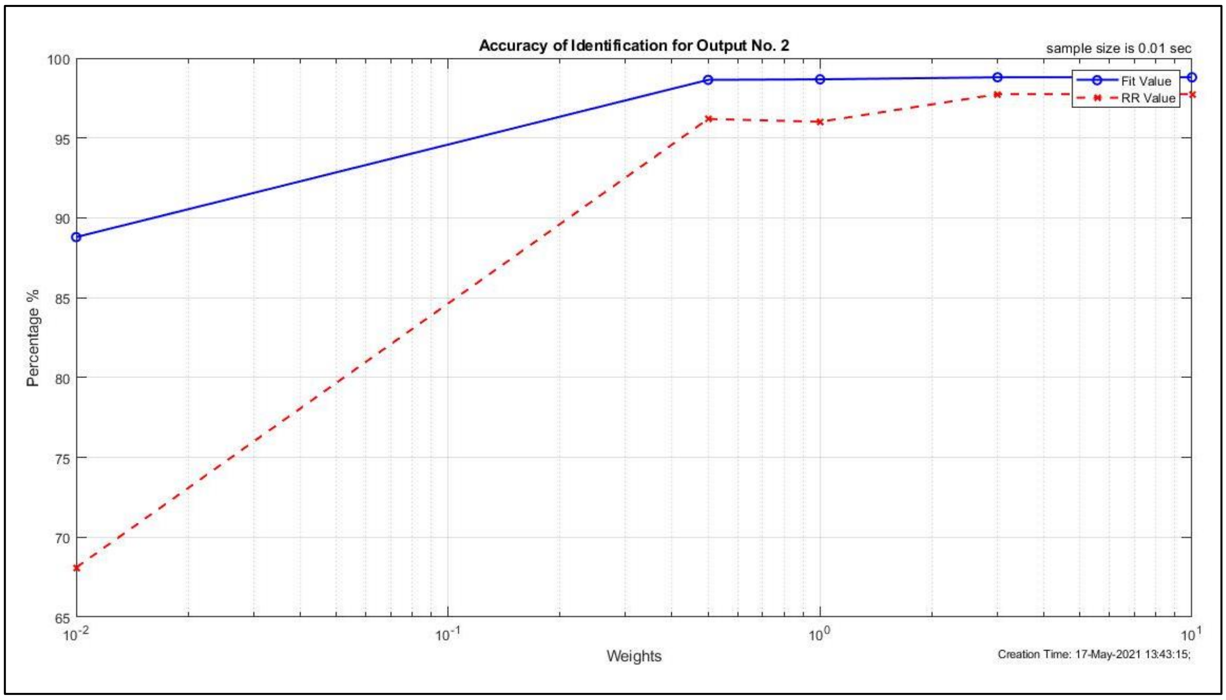


Figure 25: Accuracy comparison for different weights

It is seen that the accuracy improves by adding weightage values. This is in fact accordance with the theory (refer section 3.3.4 for the description), that increase in weights for individual elements of the error vector, reduces the error value (measured output minus simulated output) at those points. Furthermore, this improves the model quality. Hence, this verifies the identification program for applying additional weightage values to the measurements.

5.2.10 System with Dead Times

The tests for output and input delay time are performed separately (refer section 3.3.5 for the concept of time delays).

A. Output Time Delays

The program here is tested to calculate the output time delays. The same two-input two-output system (u_1, u_2, y_1, y_2) (as shown in figure 10), with the same transfer functions $G1$, $G2$, $G3$ and $G4$ of numerator order 0 and denominator order 3 are arbitrarily chosen to generate artificial measurement data. The step size for generating data is selected as 0.01. To simulate measurement errors, normally distributed random numbers with standard deviation of 0.03 are added to the artificially generated measurement data. The output time delays used for generating data are set as $D_1 = 1.2\,sec$ and $D_2 = 0.8\,sec$.

The same model order is selected for identification and the delay type to be estimated is set as 1 representing the output delay. Dead time bounds are also set as $[0\,sec \quad 2\,sec]$.

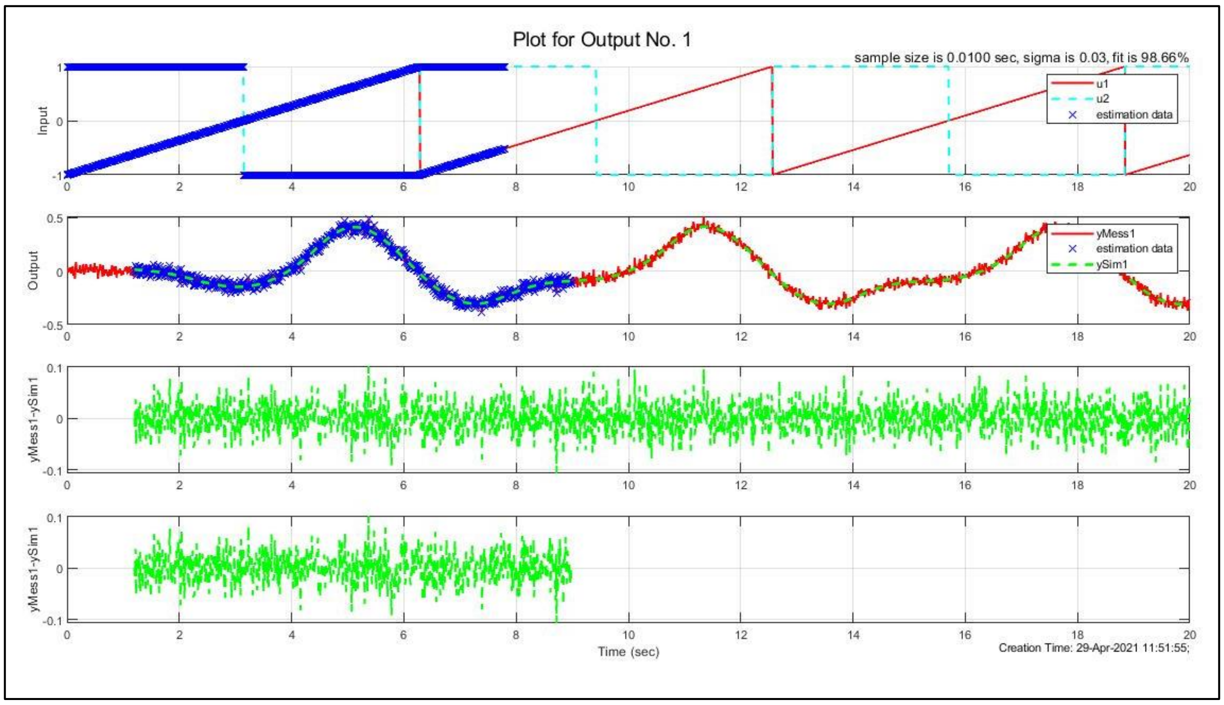


Figure 26: Measurement and simulation plots with identified transfer function and calculated delay for output 1

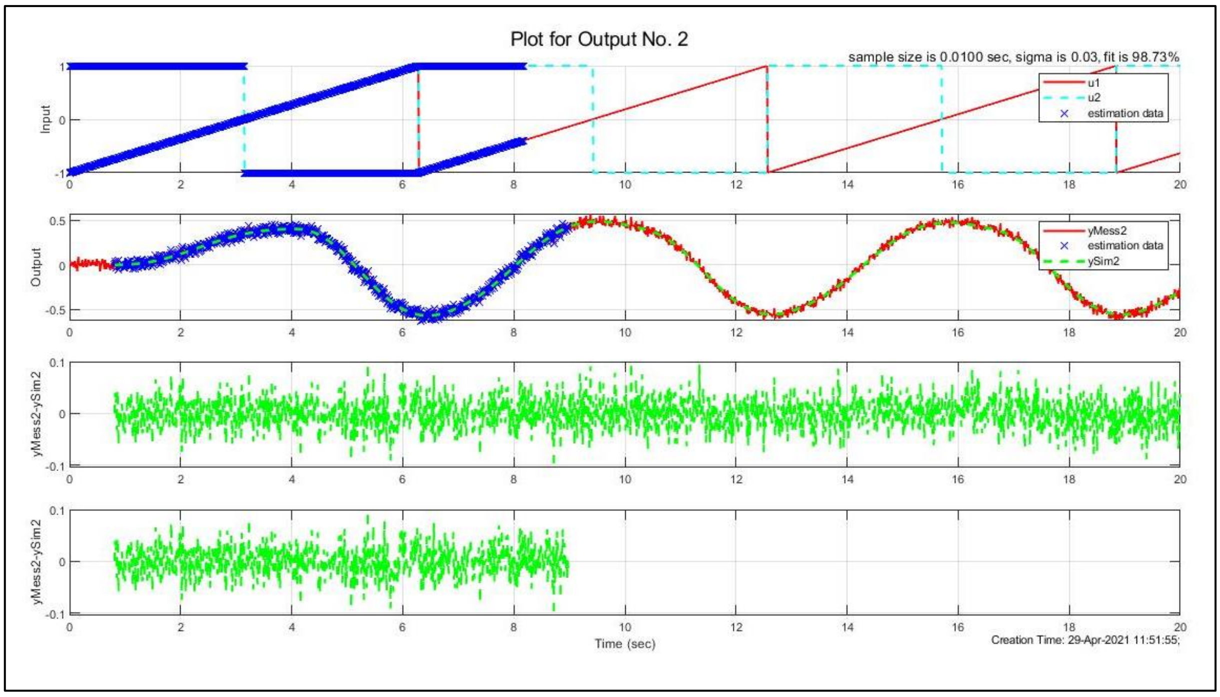


Figure 27: Measurement and simulation plots with identified transfer function and calculated delay for output 2

Parameter (unit in sec)	True Values	Calculated Values with $\sigma = 0.03$
D_1	1.2	1.2000
D_2	0.8	0.8100

Table 8: Comparison between the actual and calculated output delay times

By doing a parametric search the identification program calculates almost numerically identical output time delays as that of actual values. Moreover, a good quality model is also obtained with high accuracy values for both the outputs (Fit: 98.66 %, 98.73 % and RR: 97.76 %, 94.77%). This tests the program for calculating output delay times.

B. Input Time Delays

The program here is tested to calculate the input time delays. The same two-input two-output system (u_1, u_2, y_1, y_2) (as shown in figure 10), with the same transfer functions $G1$, $G2$, $G3$ and $G4$ of numerator order 0 and denominator order 3 are arbitrarily chosen to generate artificial measurement data. The step size for generating data is selected as 0.01. To simulate measurement errors, normally distributed random numbers with standard deviation of 0.03 are added to the artificially generated measurement data. The input time delays corresponding to each transfer functions used for generating data are set as $D_1 = 0.8\,sec$, $D_2 = 1.2\,sec$, $D_3 = 1\ sec$ and $D_4 = 1.5\,sec$.

The same model order is selected for identification and the delay type to be estimated is set as 2 representing the input delay. Dead time bounds are also set as $[0\,sec \quad 2\,sec]$. The estimation starts at the upper limit of this dead time bounds to compensate for the absence of data prior to the measurements.

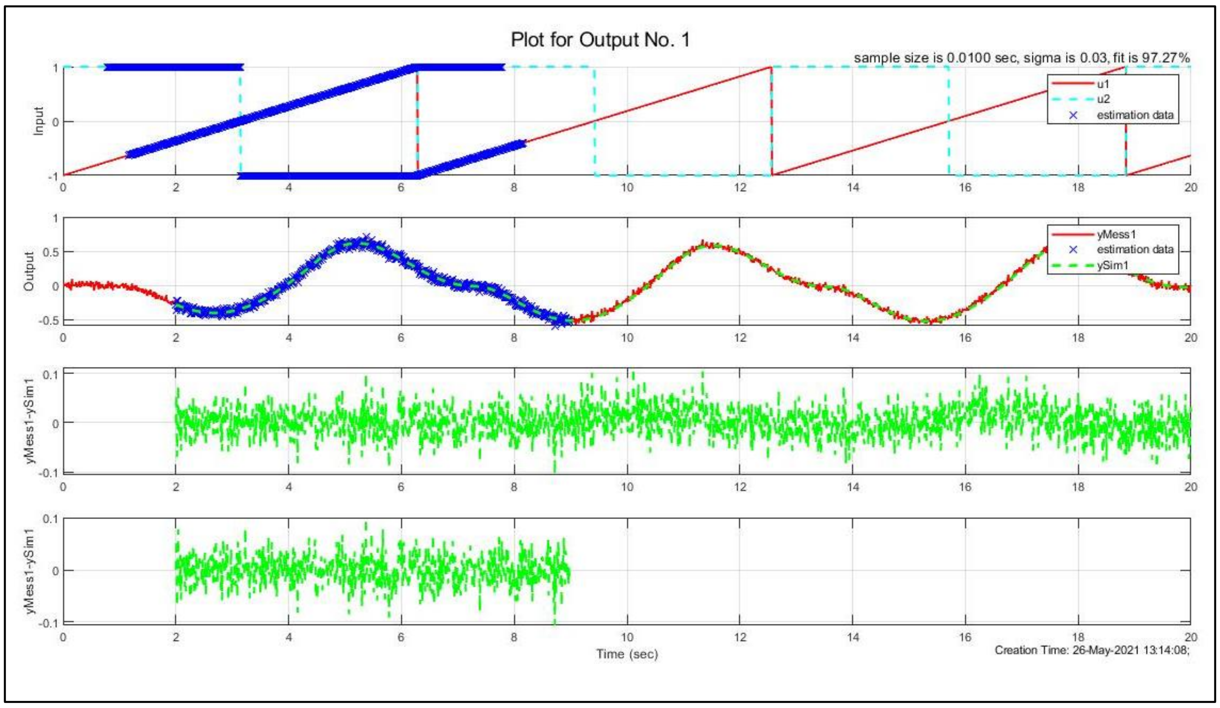


Figure 28: Measurement and simulation plots with identified transfer function and calculated input delays for output 1

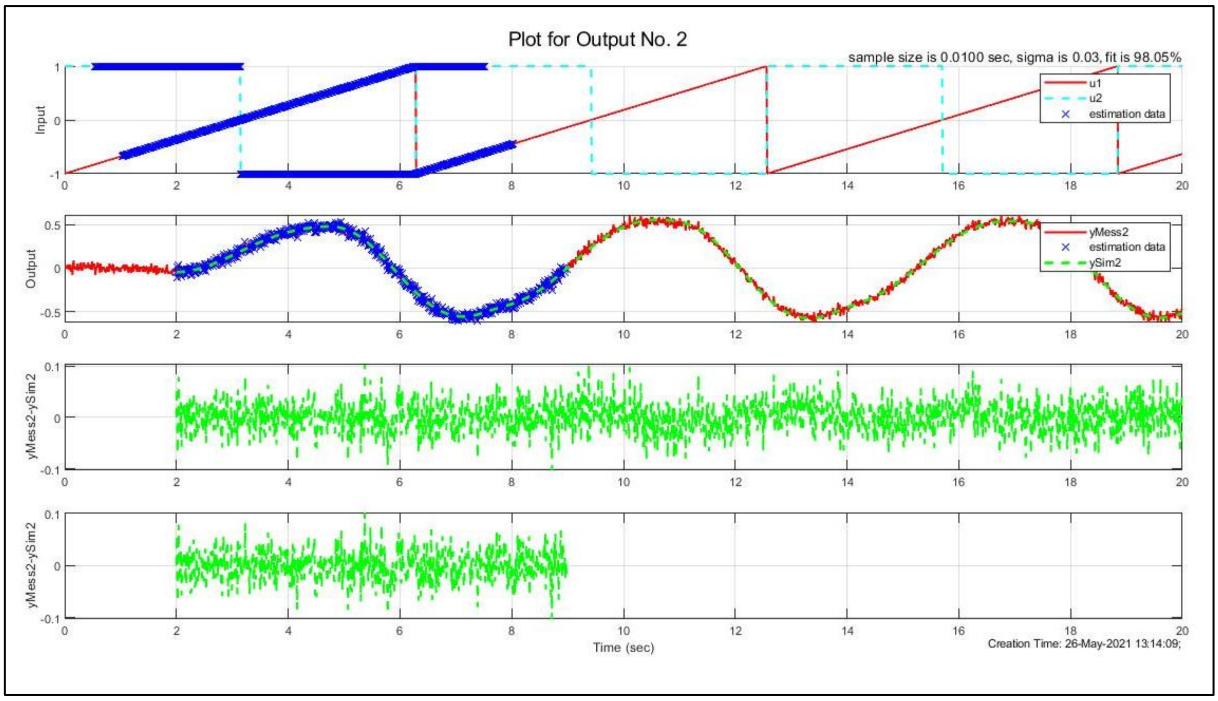


Figure 29: Measurement and simulation plots with identified transfer function and calculated input delays for output 2

Parameter (unit in sec)	True Values	Calculated Values with $\sigma = 0.03$
D_1	0.8	0.8300
D_2	1.2	1.2100
D_2	1	0.9700
D_2	1.5	1.4700

Table 9: Comparison between the actual and calculated input delay times

By doing a parametric search the identification program calculates almost identical input time delays as that of actual values. Moreover, a good quality model is also obtained with good accuracy values for both the outputs (Fit: 97.27 %, 98.05 % and RR: 93.21 %, 91.12%). This tests the program for calculating input delay times.

5.3 Conclusion

Each and every requirement of the identification program is tested with multiple tests thus verifying the identification method. Test results show that multi-input multi-output systems with noisy measurement data are identified. The program is also applicable for systems that do not start in equilibrium position because, the initial values are estimated. System with dead times are also identified with an accurate recalculation of delay times. Over-integration, normalizing different denominator coefficients and adding weightages provide more number of choices to select the best possible solution. Transfer function is obtained in both polynomial as well as factorized form. As seen from the results of the two evaluation criteria in these tests, the coefficients are recalculated, and the simulated response is generated with high accuracy. Thus, this method is applicable to open and closed loop systems and also for stable and unstable systems.

6 Performance Testing

This section tests the performance parameters like computational time, accuracy, and robustness of the identification method. It includes testing with the optimization functions to compute the dead times. The identification procedure is also tested with measurement data from a real system. A comparison is also made between the identification procedure and MATLAB System Identification Toolbox.

6.1 Output Delay Time using Optimization Function

Initially, for the estimation of dead times, a parametric search was carried out. After successfully testing the identification program for dead times, an optimization algorithm was developed. This algorithm is tested here using two MATLAB optimization functions, namely, *fmincon* and *ga* (genetic algorithm). Lower and upper dead time bounds are provided for both these optimizers to carry out a constrained minimization process between these limits. The values are set as $[0\,sec \quad 2\,sec]$ respectively. *ga* does not require an initial value for starting the optimization. For *fmincon* it is set as $0\,sec$. Now, the exact same system given in section 5.2.10 A is used to generate artificially generated measurement data for both the cases. And the remaining identification parameters are set to same values.

6.1.1 Optimization using *fmincon*

The simulation results obtained were as follows:

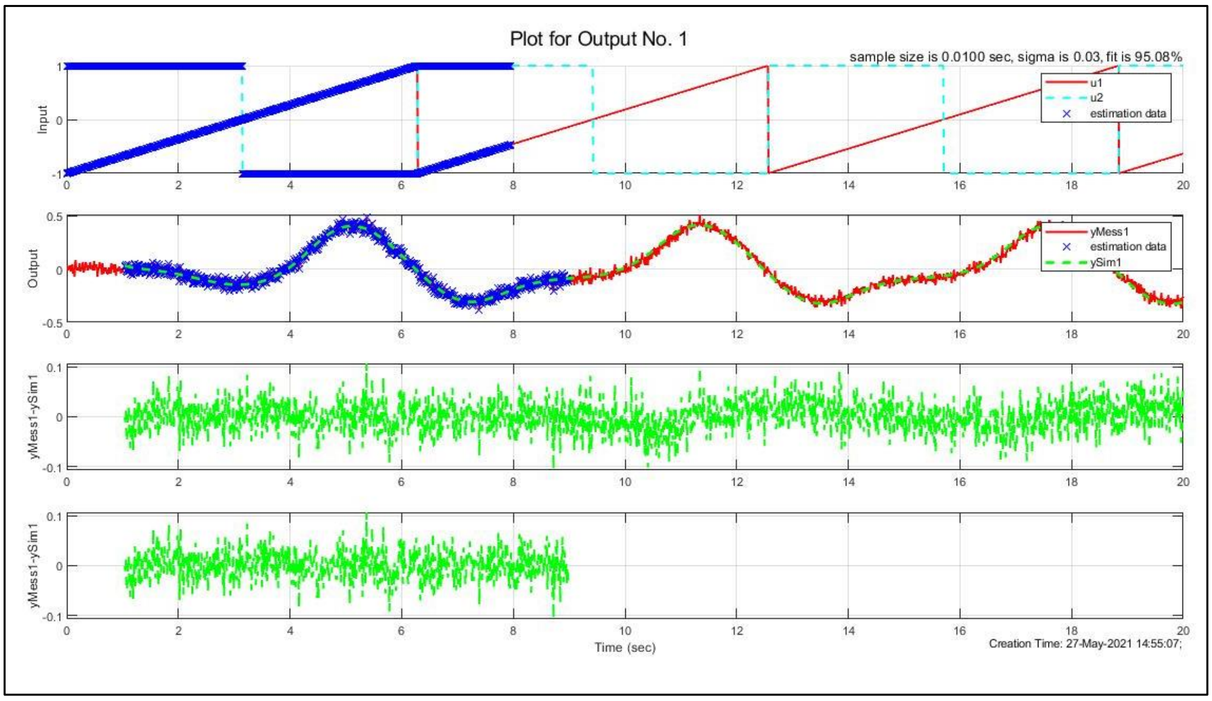


Figure 30: Measurement and simulation plots using fmincon to calculate delay time for output 1

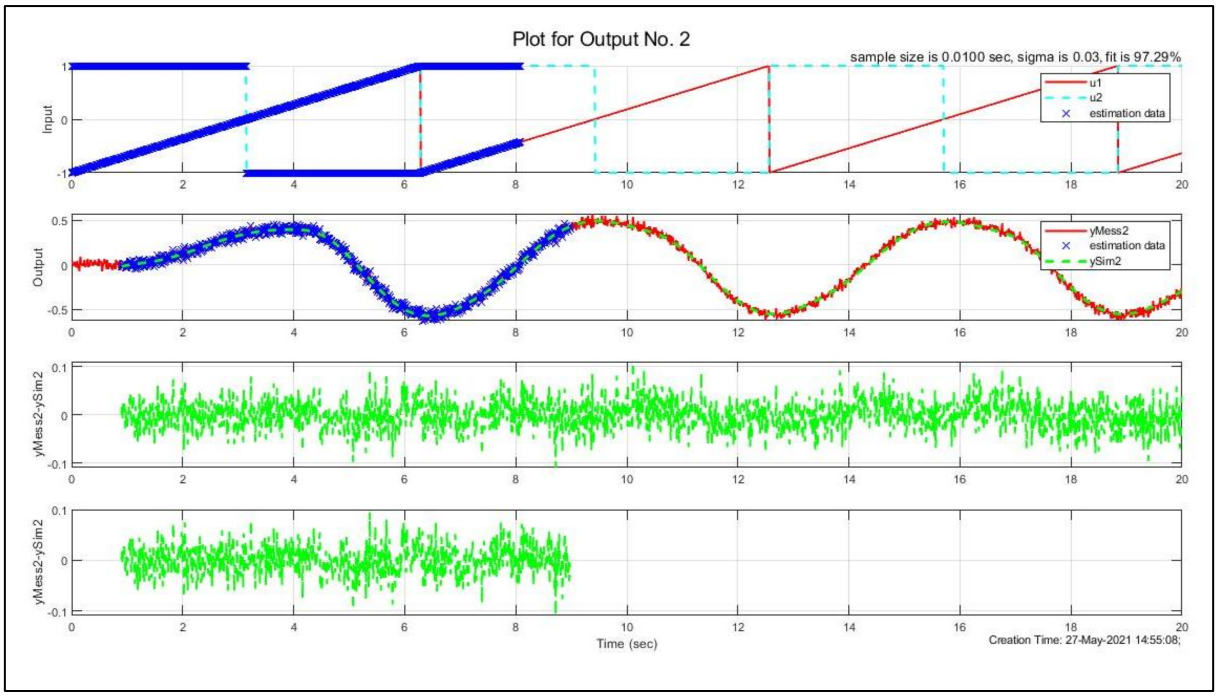


Figure 31: Measurement and simulation plots using fmincon to calculate delay time for output 2

Parameter (unit in sec)	True Values	Calculated Values with $\sigma = 0.03$
D_1	1.2	1.0400
D_2	0.8	0.9000

Table 10: Comparison between the actual and calculated output delay times using fmincon

The optimization using fmincon does not give the exact estimates of output time delays. Because it depends highly on the initial value that is provided for optimization. A better initial guess of this value could give better results. Although, the estimated coefficients obtained with an RR of 81.08 % and 89.38 % are not exact, still an acceptable fit quality is obtained for both the outputs (Fit: 95.08 %, 97.29 %).

6.1.2 Optimization using *ga*

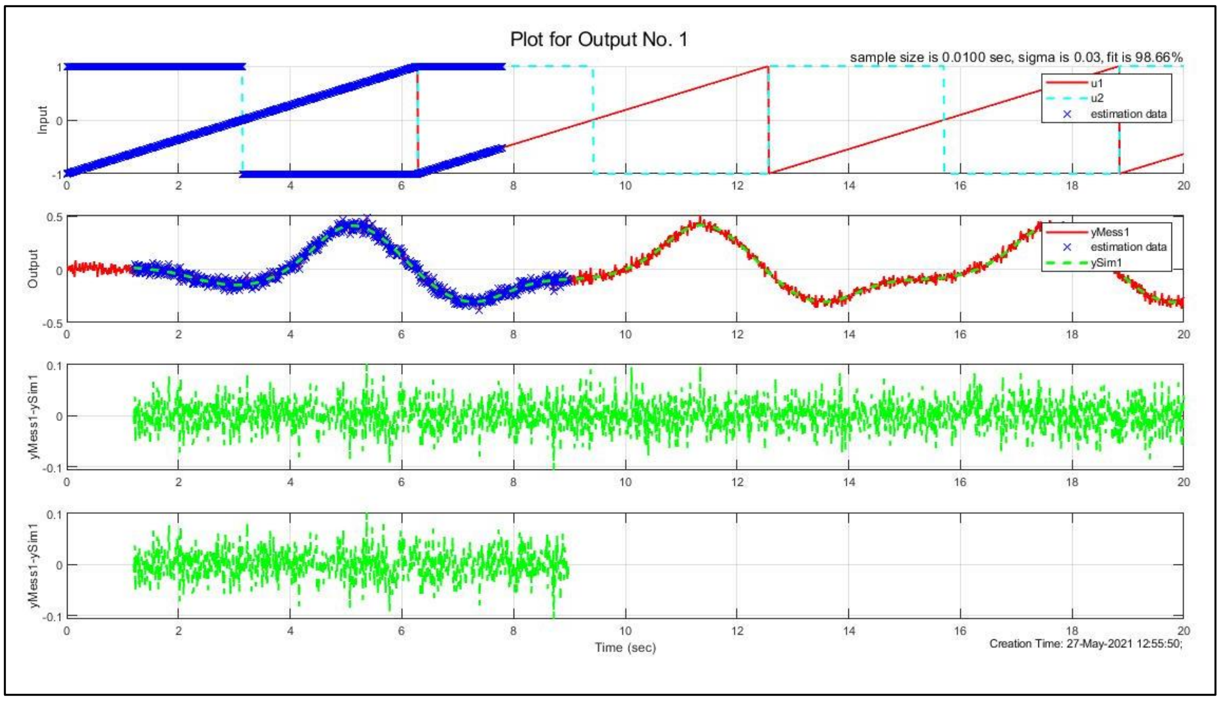


Figure 32: Measurement and simulation plots using ga to calculate delay time for output 1

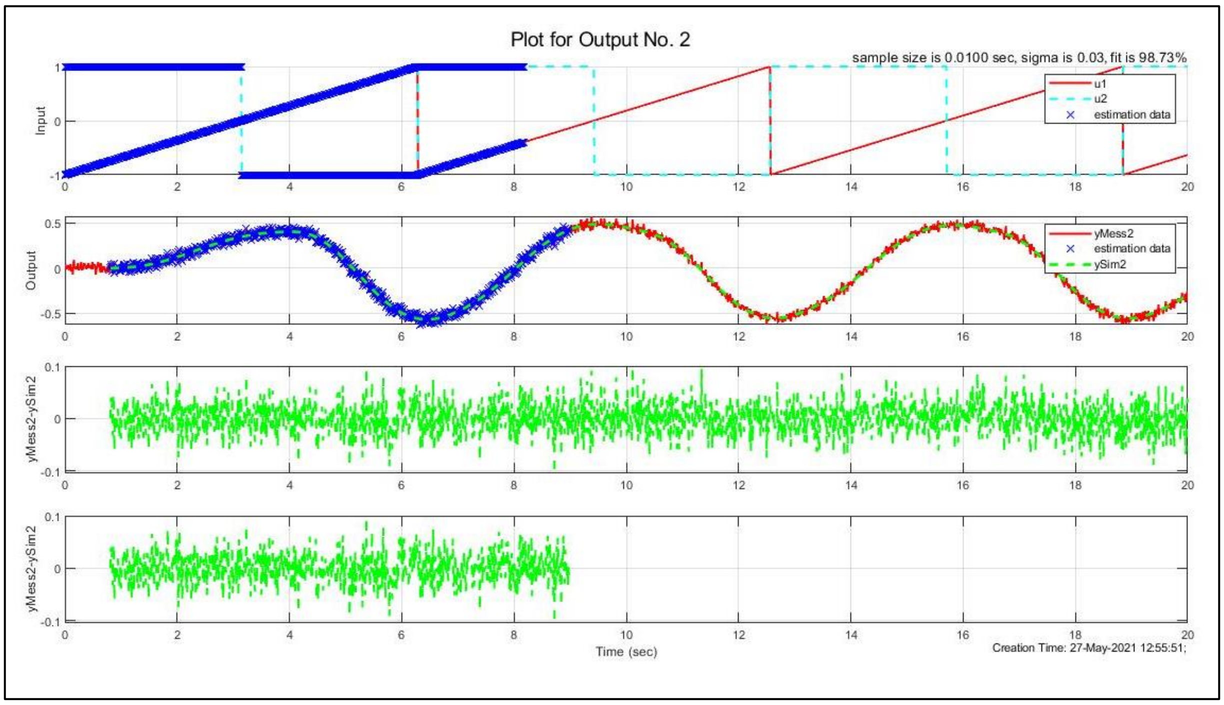


Figure 33: Measurement and simulation plots using ga to calculate delay time for output 2

Parameter (unit in sec)	True Values	Calculated Values with $\sigma = 0.03$
D_1	1.2	1.2000
D_2	0.8	0.8100

Table 11: Comparison between the actual and calculated output delay times using ga

The optimization using *ga* gives an almost identical estimate of output time delays. Moreover, a good quality model is also obtained with high accuracy values for both the outputs (Fit: 98.66 %, 98.73 % and RR: 97.76 %, 94.77%). The time delay and accuracy values obtained here are numerically same with the values of parametric search.

6.1.3 Accuracy vs Computational Time Comparison

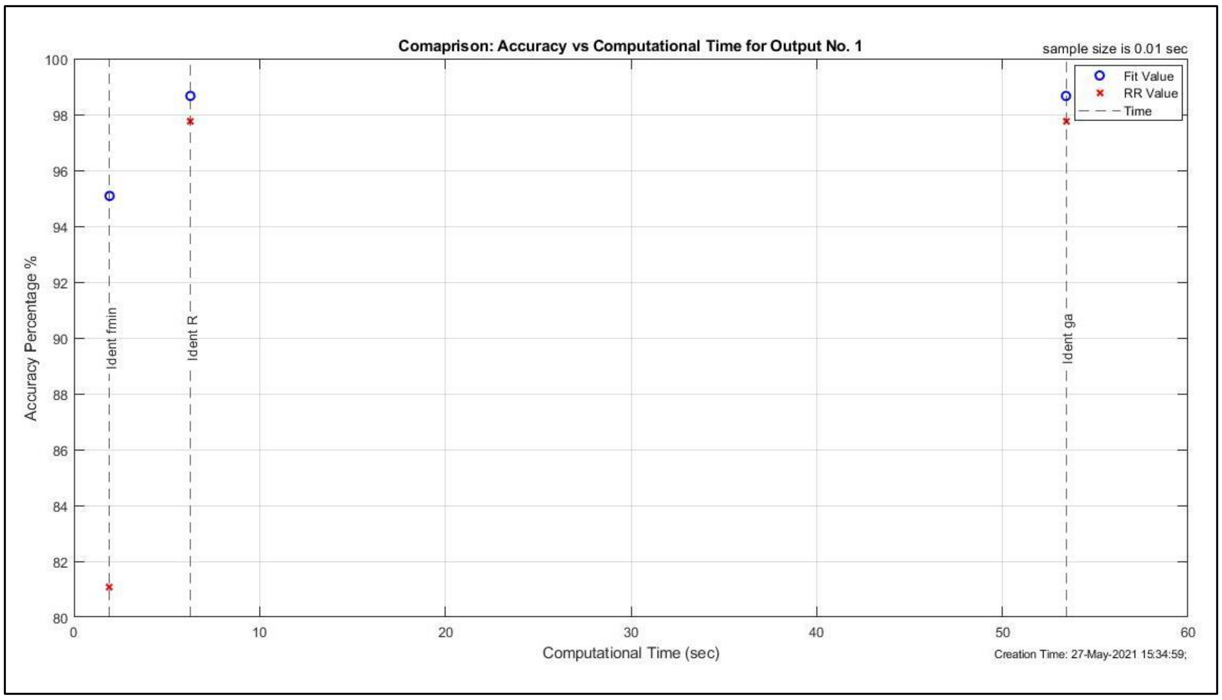


Figure 34: Comparison for output time delay estimation using the identification procedure with parametric search and the two optimizers

Although the *ga* optimizer gives the same result as the parametric search, it takes a lot of time to estimate the model parameters and dead time. On the other hand, *fmincon* is faster than both the methods but the estimated dead time and the identified coefficients are not accurate. In comparison with both the optimizers, the identification procedure with parametric search, has high accuracy and a much lower computational time than *ga*.

6.2 Identification of a Real System

Here the method is tested with measurement data obtained from a real system. Data sets are generated from a single-input single-output system with three different input signals. A single output is generated by adding the output of all the three signals together. This three-input one-output system whose true coefficients and initial conditions are unknown is tested using the identification procedure.

Initially, numerator order 0 and denominator order 2 is selected and initial conditions are calculated. The coefficient of the highest derivative of output is set to 1 for the normalization and the remaining coefficients are determined. The result with this setup is as shown:

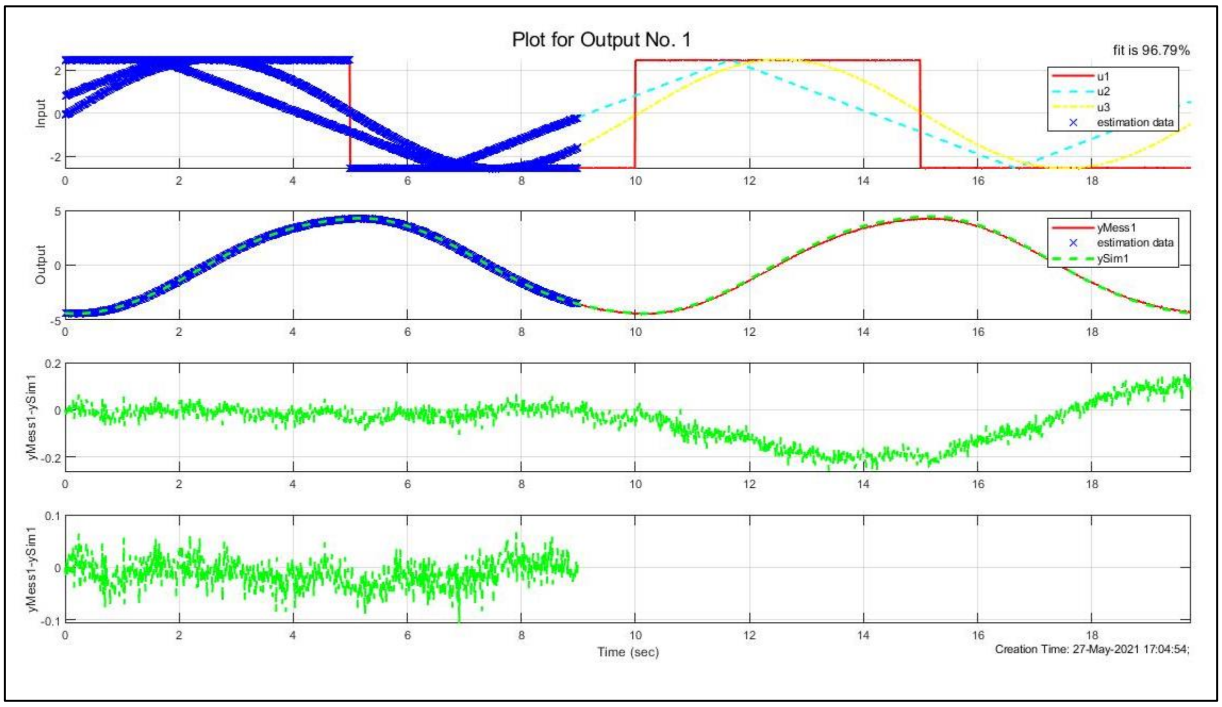


Figure 35: Measurement and simulation plots with identified transfer function of numerator order 0 and denominator order 2

A good fit of 96.79 % is obtained with the selected parameters for identification. Since the true coefficients are unknown, model quality is checked only using the fit value i.e., the difference between the measurement and simulation (shown in subplot 3 of above figure). During identification it is necessary that the noise components are filtered out instead of getting modelled. Therefore, this difference signal needs to be uncorrelated to get a better identification.

To obtain a better fit, first the order of numerator is increased and then the denominator order. This is done until there is no significant improvement in the fit value. Keeping the other parameters same, a fit of 99.20 % (almost 100 %) is achieved with numerator order 0 and denominator order 3. This is shown in the figure below:

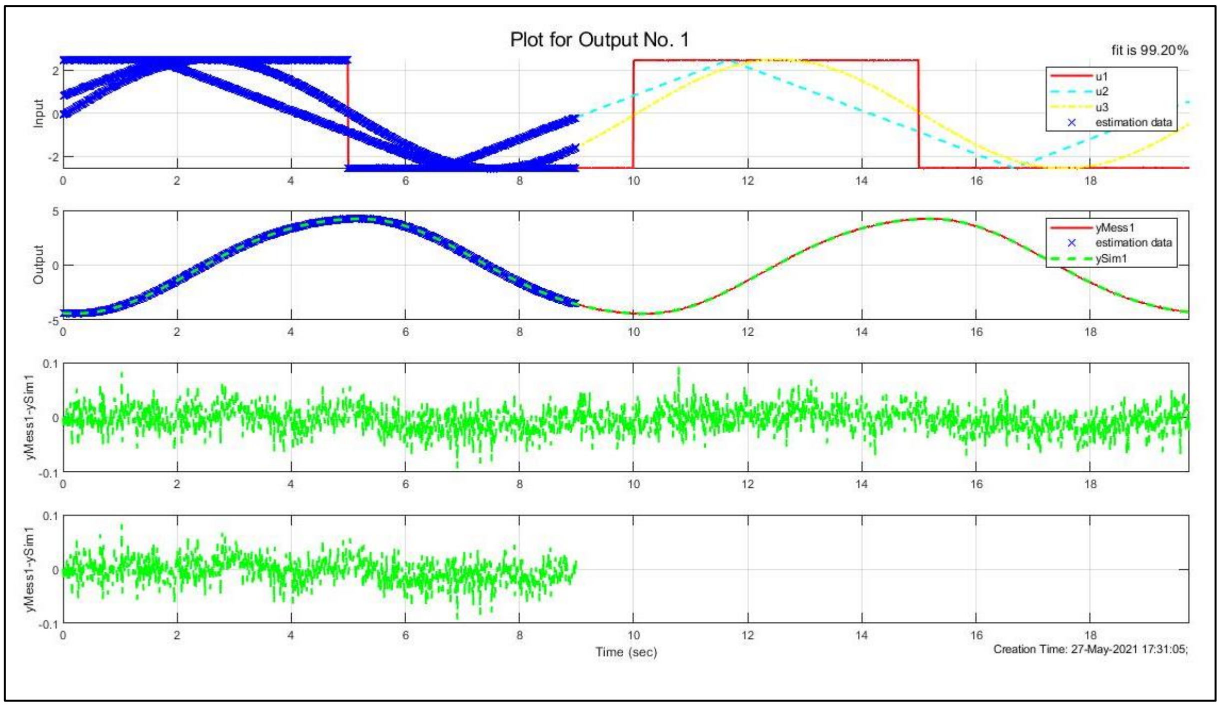


Figure 36: Measurement and simulation plots with identified transfer function of numerator order 0 and denominator order 3

Here, the best possible solution is obtained with a high model quality. In the figure above, the difference signal looks almost like uncorrelated white noise. If in case the model quality is still not good, then the estimation range or the normalization coefficient can be altered, and the fit value could be checked. The system can also be checked by setting different coefficients to zero or by searching for dead times. As seen from the previous section, the time required to compute the solution is very less and thus many models can be generated very quickly from the same measurement data set. These models can then be evaluated for the fit criterion and the model with highest fit can be selected as the best result.

Hence, the method is successfully tested with measurement data from a real system.

6.3 Comparison with System Identification Toolbox

In this section the identification method is compared with the System Identification Toolbox provided by MATLAB. In the comparisons below, it is to be noted that, a suffix R to the variables, denote the identification procedure described in this thesis, while a suffix M denotes the MATLAB System Identification Toolbox. A two-input one-output system (u_1, u_2, y) is chosen here (as shown if figure below) with arbitrary transfer functions $G1$, $G2$ with numerator order 0 and denominator order 3 to generate artificial measurement data. The values of these transfer functions can be referred in the figure below. The step size for generating data is selected as 0.01. The other test specific parameters required are defined in subsequent sections.

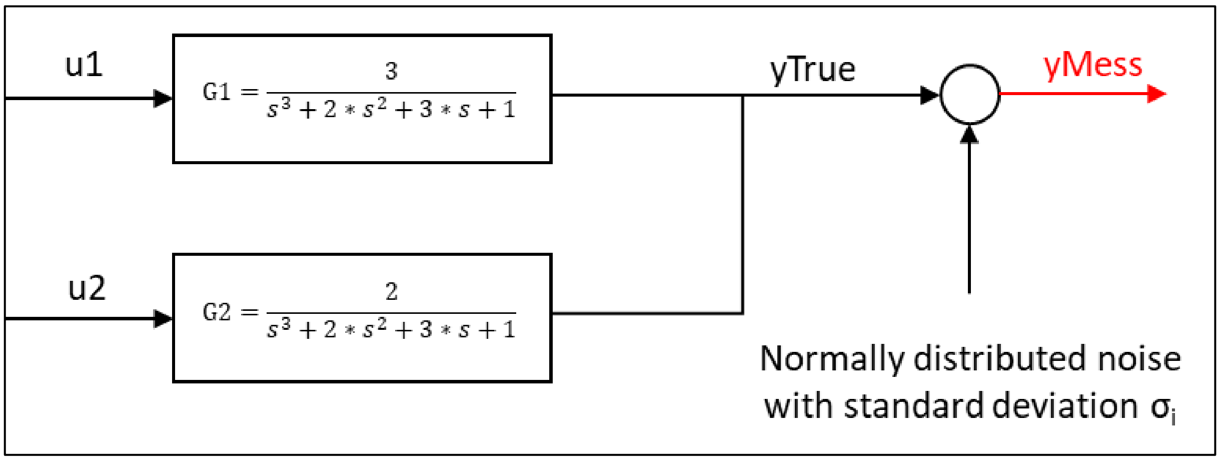

u1
$G1 = \frac{3}{s^3 + 2 * s^2 + 3 * s + 1}$
yTrue
yMess
u2
$G2 = \frac{2}{s^3 + 2 * s^2 + 3 * s + 1}$
Normally distributed noise with standard deviation σ_i

Figure 37: Signals and system for artificially generated measurement data for comparison

6.3.1 Zero Initial Conditions

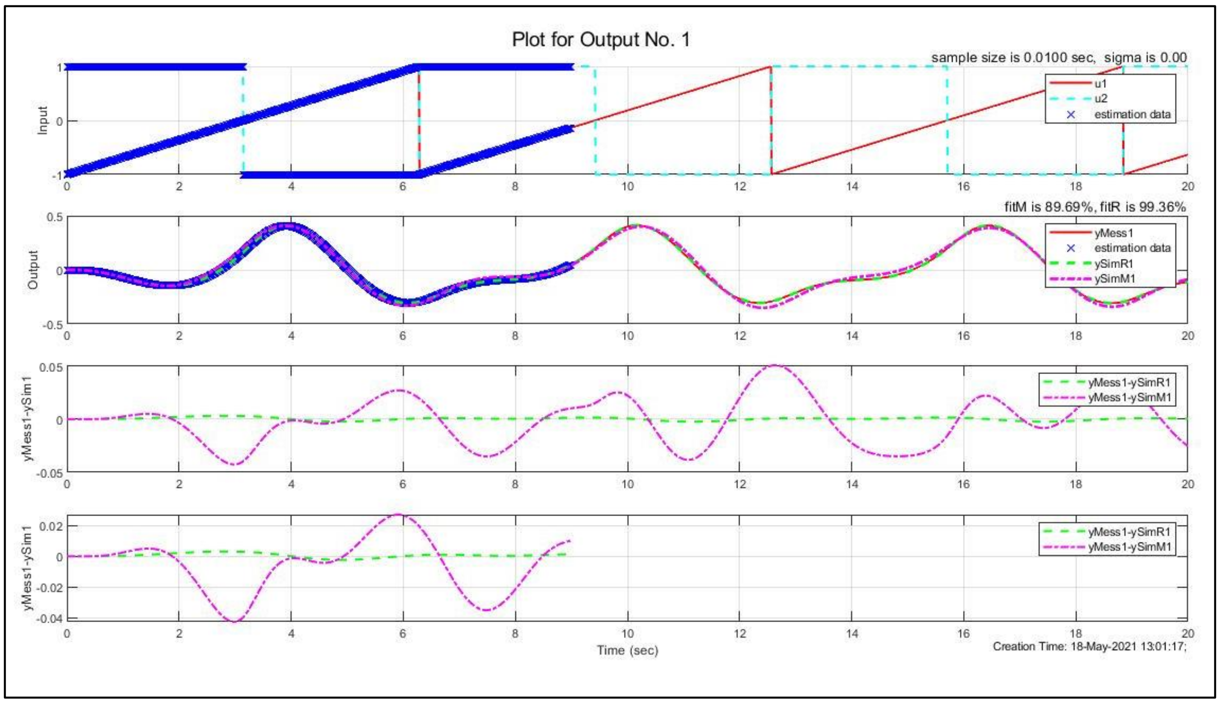


Figure 38: Measurement and simulation plots with zero initial conditions and no noise

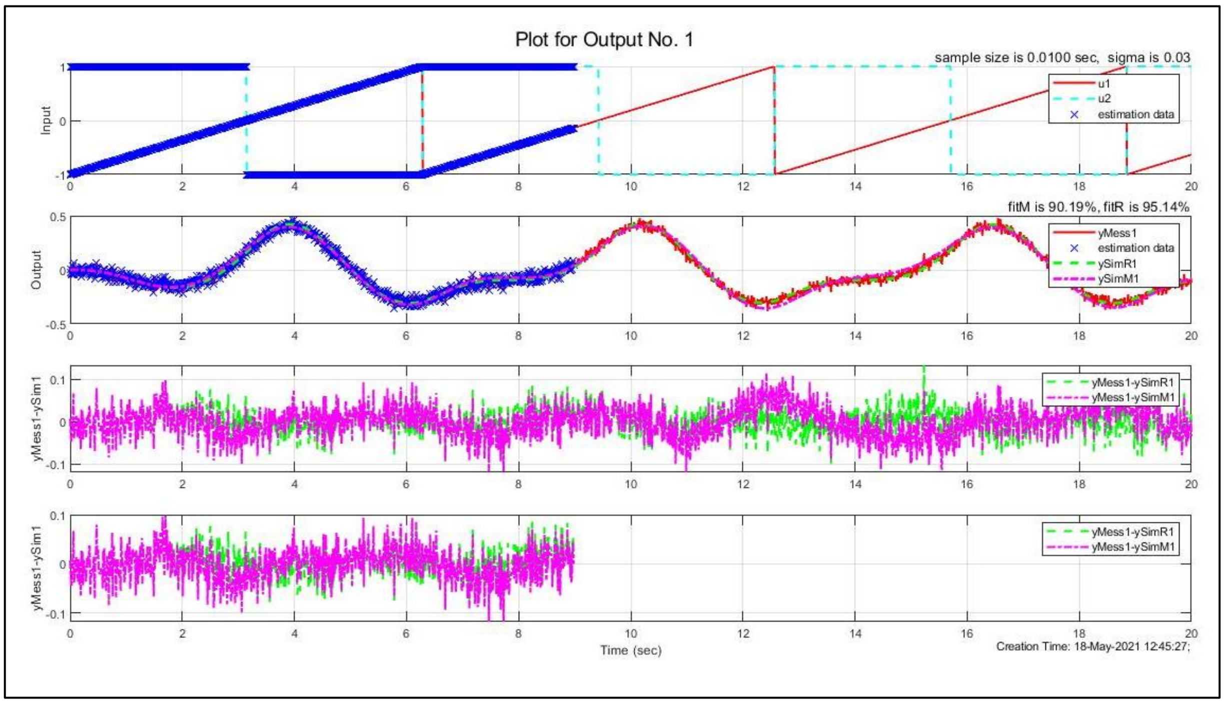


Figure 39: Measurement and simulation plots with zero initial conditions and standard deviation of 0.03

Quality Criteria		Identification Method	System Identification Toolbox
Fit	$\sigma = 0.00$	99.36 %	89.69 %
	$\sigma = 0.03$	95.14 %	90.19 %
RR	$\sigma = 0.00$	99.27 %	43.52 %
	$\sigma = 0.03$	93.77 %	41.84 %

Table 12: Comparison of quality criteria with zero initial conditions

A direct comparison shows that the identification method gives a better fit than the system identification toolbox regardless of noise levels. Moreover, the recalculated coefficients of the transfer function using the toolbox deviate strongly from the actual values thus losing their physical significance in practical applications. On the other hand, the RR value obtained from the identification procedure is good which means the recalculated coefficients can be used again to calculate back other operating parameters.

6.3.2 Output Delay Time

For the same system, the output time delays used for generating data is set as $D_1 = 1.2\ sec$. A standard deviation of 0.03 is used. The system is then identified using both the methods.

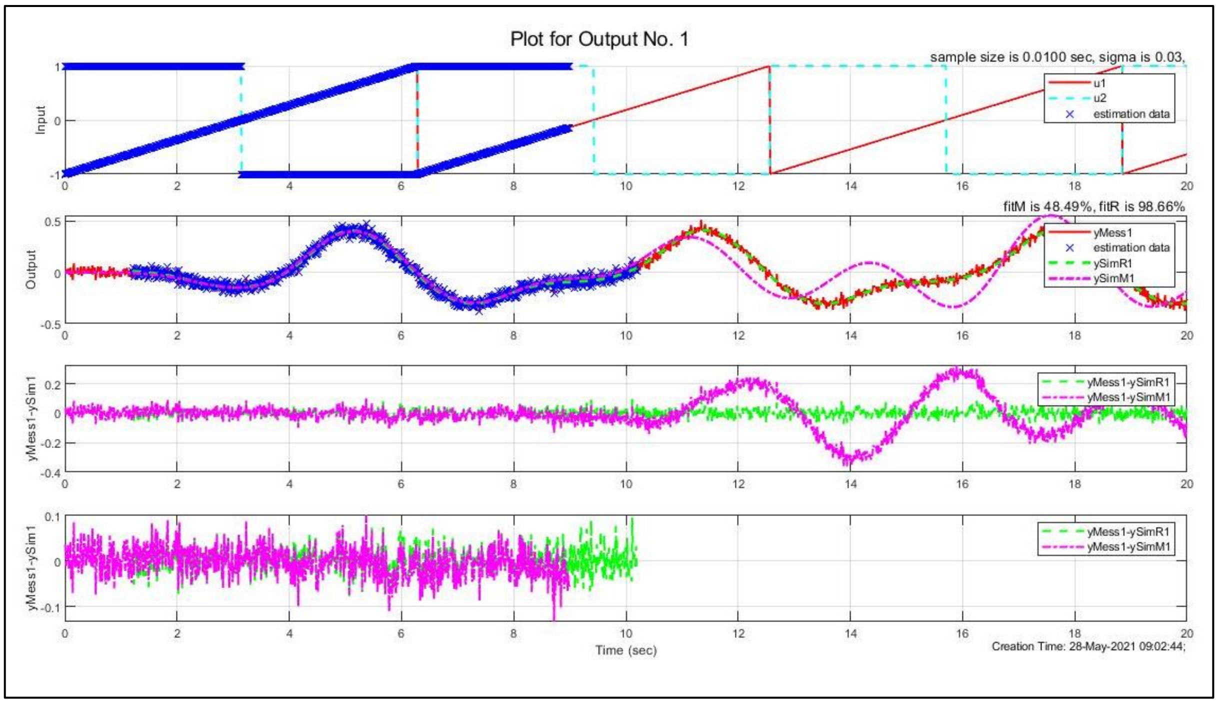


Figure 40: Measurement and simulation plots with output delay time

In case of time delays, the simulation behaviour obtained from toolbox is poor with a fit of only 48.49 %. The simulated values deviate highly from the measurement data which was not part of the estimation range. Also, the recalculated coefficients (RR: -24.30%) and the time delay value ($0.07\ sec$) obtained are not at all identical with the true values. In comparison, the identification procedure gives good results with high model quality (Fit 98.66 %) and recalculates the exact coefficients (RR: 97.82 %) and time delay ($1.2000\ sec$).

6.3.3 Accuracy vs Computational Time Comparison

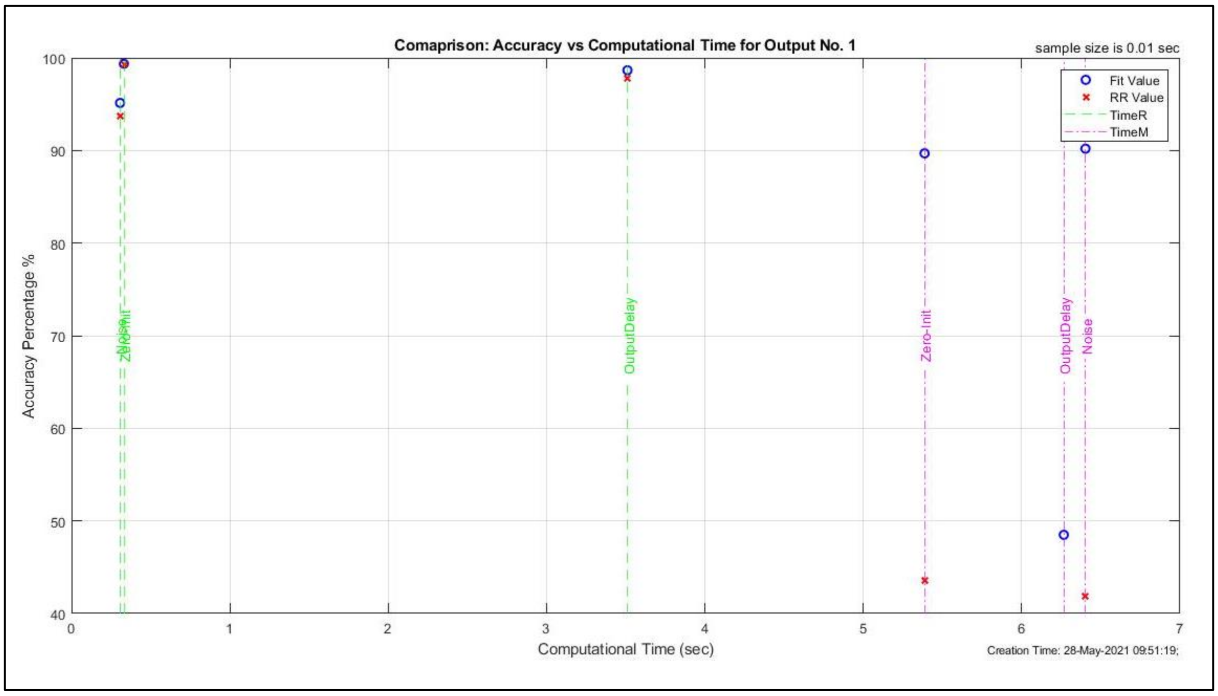


Figure 41: Comparison using the identification procedure in this thesis and MATLAB System Identification Toolbox

The green dashed lines represent the results from identification method described here, while the magenta dashed, and dot lines represent the results from the toolbox. The accuracy and computational time difference between the two methods is clearly visible. The method developed here requires less time and gives more accurate results. Whereas the toolbox takes more computational time and identifies with lesser accuracy.

6.4 Conclusion

The optimization functions were successfully tested to estimate the output time delay. On comparison with the parametric search results, the optimization with *fmincon* had a slight drop in accuracy but was found to be faster than the parametric search. This will speed up the process to identify systems with output delay times. Efforts in guessing the initial value of the optimizer can help to obtain better results. The identification method was also tested with measurement data from a real system. As the method requires a low computational time, a high-quality fit was quickly obtained by changing model orders and other parameters. This shows that the method has a wide range of application with real systems. Then three comparisons were made with System Identification Toolbox of MATLAB. In each case, the identification method developed here proved to be more efficient, in terms of both accuracy and computational time, than the toolbox.

7 Summary and Future Scope

To summarize, an identification method to model multi-input multi-output systems, described only from the theoretical concepts of bachelor course of engineering sciences was presented. A complete MATLAB program satisfying all the technical requirements for identification was developed and successfully tested with the help of multiple test cases. An algorithm for parametric search was developed to evaluate system dead times. For system with output delay times, the results of the parametric search were compared with the optimization function. The identification method was tested for measurement data generated by a real system and good quality results were obtained. Finally, the identification method proved to be more accurate and fast when compared to the MATLAB System Identification Toolbox.

At present, the extension of the optimization function to evaluate the input delay time is being worked on to further increase the speed of identification with dead times. The future scope of this method would be to include plant properties, like stability, instability, and minimum phase, in quality criterion minimization so that these properties are not lost in the solutions. Another interesting work would be to solve the least square problem that has a rank-deficient base matrix. This would help overcome the limitation of this method when the base matrix is non-invertible. Further, a web-application is to be created for this method using the MATLAB App Designer to test real-time system data.

8 References

[1] P. Zentgraf, “Ein neues Verfahren zur Modellierung linearer Systeme,” *ATP Magazine,* 2019.

[2] L. Ljung, “System Identification: An Overview,” *Springer,* 2015.

[3] C. T. M. B. Ljung L, “A shift in paradigm for system identification,” *International Journal of Control,* pp. 173-180, 2019.

[4] D. N. G. Pillonetto G, “A new kernel-based approach for linear system identification,” *Automatica,* pp. 81-93, 2010.

[5] C. A. D. N. G. Pillonetto G, “Prediction error identification of linear systems: A nonparametric Gaussian regression approach,” *Automatica,* pp. 291-305, 2011.

[6] L. L. Chen T, “Implementation of algorithms for tuning parameters in regularized least squares problems in system identification,” *Automatica,* pp. 2213-2220, 2013.

[7] d. C. R. Miller D, “Identification of Linear Time-Invariant Systems va Constrained Step-Based Realization,” in *16th IFAC Symposium on System Identification,* Brussels, Belgium, 2012.

[8] T. H. L. H. Fuh C, “Parameter Identification of Linear Time-Invariant Systems with Large Measurement Noises,” in *12th World Congress on Intelligent Control and Automation,* Guilin, China, 2016.

[9] V. M. Chou C, “Closed-Loop Identification of Linear Time-Invariant Systems Under Linear Time-Varying Feedback,” in *IFAC Dynamics and Control of Process Systems,* Corfu, Greece, 1998.

[10] W. W. Lutz J, Taschenbuch der Regelungstechnik, 2014.

[11] J. J. Crassidi J, Optimal Estimation of Dynamic Systems, Chapman and Hall/CRC, 2004.

[12] H. I. Golubev B, “Plant rational transfer approximation from input-output data,” *International Journal of Control,* 1982.

[13] “Goodness of fit between test and reference data,” The Mathworks Inc., [Online]. Available: https://www.mathworks.com/help/ident/ref/goodnessoffit.html. [Accessed 2020].

Appendix

A. MATLAB Functions

all
ceil
cell
cell2mat
circshift
cumtrapz
error
exp
factorial
find
flip
fliplr
floor
fmincon
ga
grid
hold
iddata
isempty
isnan
legend
length
lsim
mean
min
nargin
norm
ones
plot
profile
sawtooth
sgtitle
sim
sine
size
sprint
square
struct
subplot
text
tf
tfest
tf2zp
xlabel
xlim
ylabel
zeros
zpk