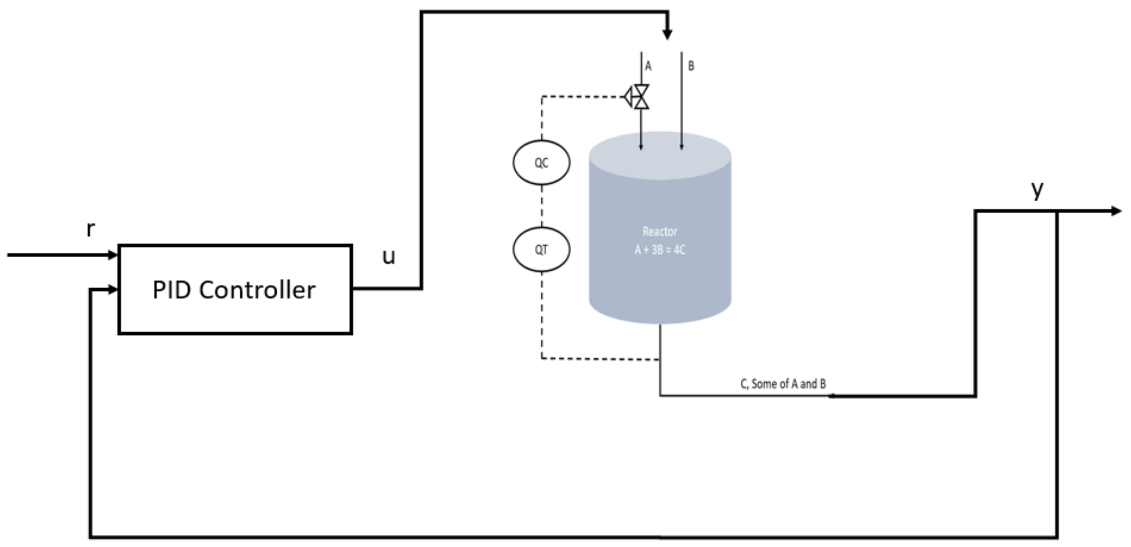
FMH606 Master's Thesis 2021

Industrial IT and Automation

Identification of process models in closed loop



Sai Sandesh Paidipamula

Faculty of Technology, Natural sciences and Maritime Sciences
Campus Porsgrunn

University of South-Eastern Norway

www.usn.no

Course: FMH606 Master's Thesis, 2020

Title: Identification of process models in closed loop

Number of pages: 119

Keywords: Open loop, Closed loop, Open loop identification, Closed loop identification, model identification, Simulink, MATLAB, Identification of process models

Student:	Sai Sandesh Paidipamula
Supervisor:	Bjørn Glemmestad
External partner:	Yara Digital Production, Bjørn Glemmestad

Summary:

In control engineering, there are two types of models known: Open loop and Closed loop. In the open loop model, the input is altered manually by an actuator based on the output desired. But in a closed loop system, the output is automatically adjusted to the setpoint using the feedback. The controller manipulates the input to reduce the error between the setpoint and the desired output, and this is done by tuning the controller appropriately. In the demonstration case, the process model is first constructed for analysis in both open and closed loop. A PRBS signal is applied to excite the systems to get the process data.

Identification of process models has advanced leaps and bounds in recent years. An excellent example would be the DSR toolbox, present in MATLAB, which provides system matrices in state space model of any process. It is proven that model identification is imperative, as it shows information of a process model in a compact form.

Open loop identification has established that the identified model shows the same information as the original process model. But when system identification is applied to a Closed loop model, it shows a strange behavior.

For a Closed loop, the identified model does not show the same behavior as the original process model. The input and output chose for identification are not just influenced by model process alone, but also with some external disturbance created by the controller. This disturbance is created by the controller in the Closed loop system.

Preface

The report contains the master thesis work done in the 4th semester of Master of Science in Industrial IT and Automation. The thesis focuses on “Model identification of process models in closed-loop”.

The reader should have had prior knowledge of control engineering, system identification and MATLAB, as they have been used in all or most parts of the work. The thesis gives little introduction to Control theory, open and closed loop systems, and system identification. A demonstration case is used here to show how the open and closed loop models are used in model identification and compare how they represent the data when compared to the data obtained from the original model. From the discussions in this thesis work I believe that it is possible for one to get a better understanding of how data identification of a closed loops system is not going to provide the same information as the original model.

I could not have completed the thesis work without the guidance of my supervisor, Professor Bjørn Glemmestad, I would like to convey my sincere gratitude for the constant support, knowledgeable insight and assistance, and timely motivation. Finally, I would like to thank my friends and family for helping me through this period.

The software and tools used for completing this thesis are:

-MATLAB

-MATLAB Simulink

-MS Office

Bangalore, 19 May 2021

Sai Sandesh Paidipamula

Contents

Table of Figures

Nomenclature

Symbol	Abbreviation/Description	Unit
u or u(s)	input	
y or y(s)	output	
Kp	Proportional Gain	
Ti	Integral Time	s
Td	Derivative Time	s
τ	Time Delay	s
e	error	
H(s)	Transfer function	
x	state variable	
x0	initial state at t=0	
A	State matrix	
B	Input matrix	
D	Output matrix	
E	Feed through matrix	
F	Innovation noise covariance matrix	
K	Kalman Gain	
n	System order	
l	number of inputs	
m	number of outputs	
r_C	Reaction rate	
k	Reaction constant	
x_A	Concentration of A	Kg/s or ton/hour
x_B or B_conc	Concentration of B	Kg/s or ton/hour
x_C	Concentration of C	Kg/s or ton/hour
B_in	Inlet feed of B	Kg or ton
dmdt	Change of mass wrt to time	Kg/s or ton/hour
A_in	Inlet feed of A	Kg or ton
f_out	mass flowing out of the reactor	Kg/s or ton/hour
B_in_o	Feed of B in open loop	Kg or ton
raw_m_B_o	Concentration of B in open loop	Kg or ton
m_A	Mass of A	Kg or ton
m_B	Mass of B	Kg or ton
m_C	Mass of C	Kg or ton
y_o	open loop output data	
u_o	open loop input data	
B_in_c	Feed of B in closed loop	Kg or ton
y_c	closed loop output data	
u_c	closed loop input data	

dt	Sampling time	s
T	Temperature	Celsius
P	Power	Watt
PRBS	Pseudo Random Binary Signal	
MIMO	Multiple-Input Multiple-Output	
SISO	Single-Input Single-Output	
SSM	State Space Model	
ODE	Ordinary Differential Equation	
DSR	Deterministic and Stochastic systems and Realization	
SVD	Single Value Decomposition method	

1 Introduction

System identification is applied to make a smaller, compressed model of any given process that gives the same information as the process. In control engineering, the aim is to design a system that reduces human interaction and operates a plant automatically. There are a variety of techniques available to control a process. A control loop can be seen in almost every situation, for example, while cooking, the cook observes the amount of heat required and through a sensory feedback (which can be visual, smell or touch) and adjusts the heat accordingly; or a driver driving a car with respect to how fast he wants to move. These are examples of Open loop control, as they are based on a feedback. In a Closed loop system, the process is made automatic. There exist many methods for PID tuning, although they vary mainly in choosing the desired characteristics of the system. Even after using a variety of tuning parameters, the system could still show unexpected behaviour, showing that there is no set of parameters that can be chosen. The choosing of the parameters can be made based on the necessity; high gain is used for performance and low gain for less input usage and a robust model.

In a feedback closed loop control, the input is controlled based on output required, and a change in input is applied until the measured output reaches the desired output. To have an automatic control, a PID controller is generally used.

The purpose of the experiment is to demonstrate the difference in responses of the open and closed loop model identification. In the open loop identification testing, an open loop system without a controller is used, whose input and the output are used for system identification. Using the system matrices that were found using the DSR method of system identification, the identified model is checked for consistent responses compared to the original model. In the closed loop, the same is expected to be found, using the input and the output when the controller is in operation.

1.1 Background

In control theory, subspace identification is used to identify the state from the available input-output data. The concept of system identification was first known to be found in statistics.

In 1985-1995, subspace identification methods were improved upon to operate on input-output. A breakthrough development was made for operating directly on the available inputs and outputs and avoiding the estimating the samples of covariance functions prior to system matrix realization (J-N, 1985) [1].

PID controller is a foundational control technology for process manufacturers, since the early 1900s. The first real PID-type controller was developed by Elmer Sperry in 1911. The first theoretical analysis of a PID controller was published by Nicolas Minorsky in 1922. A former pneumatic controller with completely tuneable was implemented by Taylor Instrumental

Company in the year 1933.1731 Various tuning rules are documented in the 'Handbook of PI & PID Controller Tuning Rules' from 2009 [2].

Control theory is classified into Classical and Modern control theory. Classical control theory emerged in the 1930s and 1940s[3]. Here physical systems and control design in the frequency domain are expressed in transfer functions such as Laplace transforms. Bode plots, Nyquist plots and root locus are the tools for analysing systems and design controllers. Modern control theory emerged around 1960. Here time domain systems are described in state-space representations. State-space is commonly used to model multiple-input and multiple-output (MIMO) systems, such as spacecraft, aircraft, automobiles, marine vessels etc.

The model identification can be applied to a system of any order, and this is possible with the advances made in computing the system with new ideas and algorithms found using the input and output from the simulated model. These models generally contain a lot of important information about the system. This information can be used for system identification to make compact forms, for example state space models. The models made generally contain most of the information of the model and can be replaced as the original model for analysis or other usage. It is very interesting to know that the Open loop models show almost the exact information when system identification is done, whereas there is a lot of uncertainty in finding the same for Closed loop system identified models.

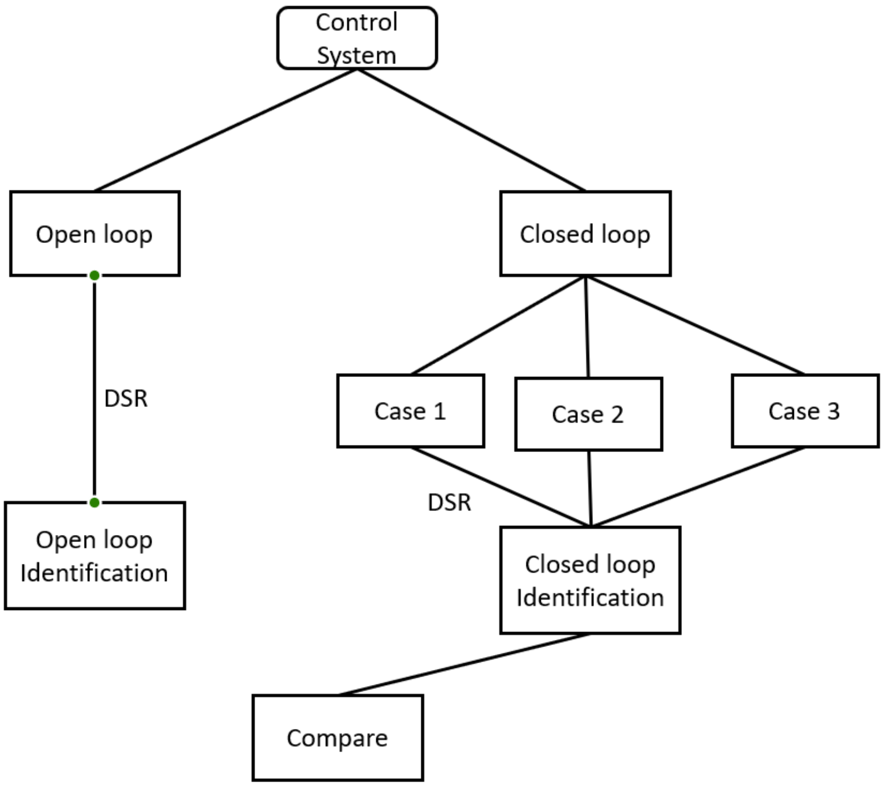


Figure 1.1: Flow of thesis

1.2 Objectives and goals

The goals of the thesis as mentioned in the task description are:

- Literature survey on model identification methods for industrial plants and processes controlled in closed loop.
- Describe a demonstration case for identification in both open and closed loop and implement the same in a programming language, MATLAB.
- Give an overview of the open loop model, and how model identification is done.
- Give an overview of the closed loop model, for possible scenarios (3 cases), and identify these models.
- Compare the open and closed loop models identified.

1.3 Report structure

The structure of the thesis is:

- Chapter 1 contains the introduction with the background and objectives of the thesis.
- Chapter 2 shows the literature survey done on model identification and closed loop identification.
- Chapter 3 is the background and theory of Control engineering with a few explanations of control theory and Modeling the system.
- Chapter 4 contains background and theory of system identification with some details about state space models.
- Chapter 5 contains the demonstration case done for open loop, closed loop and comparing the outputs. It also gives a summary of the understanding of the simulation done.
- Chapter 6 mentions how the thesis can be used for future study.
- Chapter 7 is a short conclusion of the thesis.

2 Literature Survey

In [4], where the author finds a CS-PSO fusion identification algorithm, to identify the optimal model of the thermal process based on the process input and output data. In this algorithm, the conventional particle swarm optimization algorithm is employed to identify the model parameters of the typical thermal process, where the cuckoo searching algorithm is used to optimize its velocity parameters, and thus the identification accuracy is improved. The effectiveness of this approach is validated by the extensive simulation results about the ultra-supercritical unit and the circulating fluidized bed thermoelectric unit. In order to talk about PSO-CS Algorithm, the CS algorithm has a long history, which can jump out of the local optimal value to find the global optimal variables, and by using CS algorithm we can optimize the speed parameters of PSO algorithm which can improve the identification accuracy. A Levy flight which is a typical random walk mechanism is used, where a class of non-Gaussian stochastic processes and is related to the Levy stable distribution.

The idea of the PSO-CS fusion algorithm is that the velocity parameter v in the particle swarm optimization algorithm is optimized by the cuckoo search algorithm.

Another interesting Process model identification is disclosed in [5],where a method of controlling and managing a process control system having a plurality of control loops. This method includes implementing a plurality of control routines to control operation of the plurality of control loops. The plurality of control routines may include at least one non-adaptive control routine. Operating condition data is then collected in connection with the operation of each control loop of the plurality of control loops, and a respective process model is identified for each control loop of the plurality of control loops from the respective operating condition data collected for each control loop of the plurality of control loops. In some cases, the identification of the respective process models may be automatic.

In some process control systems, the controller can include and executing one or more modules, each of which receives inputs from and/or provides outputs to other module and performs some process operation, such as measuring or detecting a process parameter, controlling a device, or performing a control operation, such as the implementation of a proportional-derivative-integral (PID) control routine. Process controllers are typically programmed to execute a different algorithm, sub-routine, or control loop for each of a number of different loops defined for or contained within a process. In general, each such control loop includes one or more input blocks, a single output control block and an output block. The function blocks that implement such control routines, have been configured in accordance with several control techniques.

The Time-domain approach in [6] initially provides introduction to the main aspects of existing time-domain methods and software for identifying linear continuous-time models of dynamical systems from sampled input or output data. Further it demonstrates these approaches via simulated and real data examples. In order to identify a continuous-time model from time-domain sampled data, two main time-domain approaches namely, the 'indirect' approach, a DT model is identified first using DT identification methods, and then converted into a CT model using a knwon algorithm for discrete to continuous-time conversion. And the other approach is 'direct' approach where the CT model is identified directly from DT data. This paper mainly concentrates on Direct approach.

The software in the Continuous-Time System Identification (CONTSID) toolbox contains most of the parametric modelling methods, which allow one to directly identify CT models of linear time-invariant SISO, MIS0 and MIMO systems from uniformly and non-uniformly sampled data. CONTSID toolbox is designed as an add on to the SID toolbox. The other toolbox is the Computer Aided Program for Time series Analysis and Identification of Noisy systems (CAPTAIN) is a more general toolbox intended not only for the identification of DT and CT transfer function models but also for the extrapolation, interpolation and smoothing of non-stationary and nonlinear time series. CT identification algorithms are all based on Refined Instrumental Variable (RIV) estimation.

The main advantage of the continuous-time methods is that they provide differential equation models whose parameters can be interpreted immediately in physically meaningful terms. In this paper various illustrative simulations and real examples are taken. One among that is rainfall flow modelling where it concerns the modelling of the daily effective rainfall-flow data. In this Effective rainfall is a nonlinear transformation of measured rainfall that is a function of the soil-water storage in the catchment and provides a measure of the rainfall that is effective in causing flow variations.

A Discrete-Time state feedback control design provides information on how to design a discrete-time control system for linear objects with delay using a state feedback controller with an observer in [7]. Discrete-time models of control objects with time delay have the finite dimension of the state space model. The results of research are illustrated with a MATLAB example. The continuous model of the control object has to be transformed into the discrete-time form. The state space of continuous linear control objects with delay has infinite dimension. This feature leads to difficulties in control algorithms designing for such objects. The discrete-time models of continuous linear control objects with delay are that the state space of such models has the finite dimension. This became possible by losing some information about the behaviour of the control object due to the sampling of processes in time. So, control algorithms based on discrete-time models of control objects can be synthesized without any specific methods using a lot of known methods of state feedback control designing.

In [8],a refined IV method was found, where it focuses on instrumental variable techniques which are used to identify closed loop plant models relying on simple linear algorithms. For closed loop identification, initially a basic IV estimator was used and later a tailor made IV algorithm was proposed. Recently a new technique based on identification of a realistic BOX Jenkins model was proposed. But the downside to this model is it uses non-linear algorithms. This problem can be overcome by the use of the optimal Refined Instrumental Variable (RIV) method of estimation. The study of identification of a non-linear-in-the-parameters Box-Jenkins TF model within a closed loop environment, by using the optimal Refined Instrumental Variable technique modified to handle the closed loop situation (RIVCL) is observed in this paper. It has shown that a minimal value of the associated parametric error covariance matrix can be achieved by the RIVCL choice of instruments and prefilters. This proposed method is compared to recently suggested estimators and RIVCL algorithm provides the best efficiency.

3 Background and theory on Control Engineering and Modelling

3.1 Control Engineering

Control system engineering deals with the principles of control theory, to design a system which gives the desired output in a very controlled manner. It is often implemented to analyse, design, and optimize complex systems consisting of highly integrated coordination of mechanical, electrical, electronic, or pneumatic elements. This chapter introduces PID controller, control loops and other measures used to determine the performance and robustness of these control loops.

3.1.1 Control Loop

In a control system to achieve automatic control, the control loop consists of all the components that are required to improve the process value (measured output) to be equal to a desired set point[9]. The main steps involved in a controller are:

- Measure the current value of the measured value to be controlled.
- Compare with the given set-point.
- The difference is calculated, and a corrective signal is sent.
- The change takes place and step one is repeated again.

The above steps are repeated until the error, or the difference between the measured value and the desired set point value is zero.

A process model can be determined using an excitation. A PRBS signal is used to determine the Chemical process experiment here. And later a step response method is used to check the system identification part for confirming that the model identified is good. More is explained in further topics.

3.1.2 Open loop

A control system which is operated manually, and completely independent of the output of the system is an open loop system.

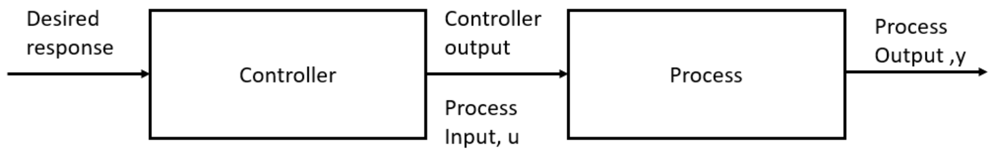


Figure 3.1: Open loop system Block diagram

The above image of an open loop control system, the output is completely independent of what operation is done with the controller, as shown in Figure 3.1. Typical examples of open loop systems in daily life are, using an Electric hand dryer, Automatic Washing machine, a Toaster, a Coffee maker, or temperature control of water in shower. Using an open loop system can be advantageous, as it is very simple to construct, the input and output variables are easily measurable, and can be used for future analysis. The downside to it is that the operation is manual and does not happen automatically. Hence, open loop control is also called Manual control[9].

3.1.3 Closed loop

A control system where the output is affected by the input, by adjusting the input automatically so the output reaches a desired value is called a closed loop control system. Usually, an open loop control system with a feedback control is called a closed loop control[10]. Using the feedback, the controller adjusts the input to make the measured output equal to the desired output. Hence, the closed loop control is also called Automatic control.

The process variable is the system parameter that needs to be controlled such as temperature, pressure, or flow rate, in a typical control system. The process variable and feedback to the control system is measured by a sensor. The set point is said to be the desired or command value for the process variable. To determine the desired output to drive the system (plant), the difference between the process variable and set point is used by the control system algorithm, at any given moment. So, the continuously repeated process of reading sensors to provide constant feedback and calculating the desired output at a fixed loop rate is called a closed loop control system as shown in Figure 3.2.

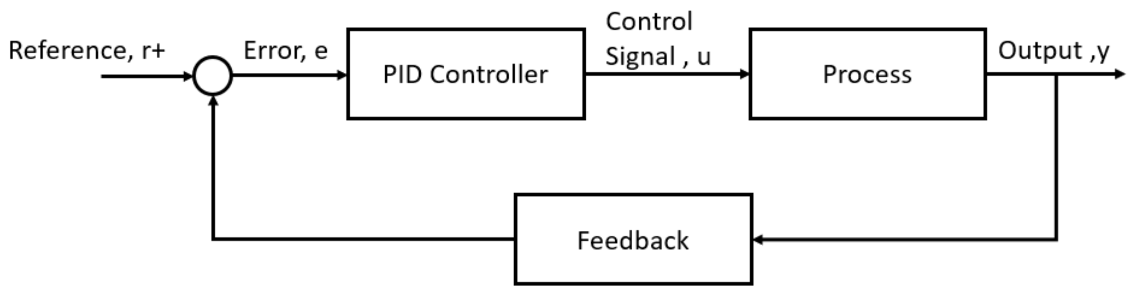


Figure 3.2: A closed loop feedback block diagram

The term closed loop control implies the use of a feedback control action in order to reduce any errors within the system and its feedback distinguishes the main differences between an open loop and a closed loop system. Main characteristics of a closed loop control system are as follows:

- Reduces errors by adjusting the input of the system automatically.
- Improves stability of an unstable system.
- Increases or decreases the system sensitivity.
- Enhances robustness against external disturbances to the process.
- Produces a reliable performance.

Control system performance[11] is often measured by applying a step function to the setpoint command variable, and then measuring the response of the process variable. The response is taken by measuring defined waveform characteristics. Rise time is the amount of the time the system takes to go from 10% to 90% of the steady-state, or final value. Settling time is defined as time required for the settlement of process variable to within a certain percentage of the final value.

3.1.4 PID controller parameters

PID controllers are widely used in applications for industrial process control. About 95% of closed loop operations of the industrial automation sector use PID controllers. PID stands for Proportional-Integral-Derivative. The three controllers are combined in order to produce a control signal. A control loop feedback device is used to regulate all the process variables.

3.1.4.1 Proportional Response

The ratio of output response to the error signal is called proportional gain (Kp). However, if the proportional gain is too large, the process variable will begin to oscillate. And if Kp is increased further, the oscillations become larger, and the system becomes unstable and may oscillate out of control.

3.1.4.2 Integral Response

The integral component sums the error term over time. Even a small error term causes the integral component to increase slowly as a result. The integral response continually increases over time unless the error is zero, therefore, the effect is to drive the steady state to zero. The difference between the desired output and setpoint is called steady-state error.

$$ki = \ kp\,Ti \tag{3.1}$$

3.1.4.3 Derivative Response

The derivative component decreases the output if the measured output increases rapidly. If the derivative time Td parameter is increased, it causes the control system to react more strongly to the changes in the error term and in turn increases the speed of the overall control system response.

$$kd = \frac{Kp}{Td} \tag{3.2}$$

3.1.5 PID Tuning

PID tuning is setting necessary gains for the process with P, I and D to get an optimal response from a control system. It is done by using the coefficients known as the controller gains, which can be formulated in different ways. It is not always necessary to include all the three terms of controller as they might reduce the performance. For a typical PID controller, the response is given by (3.3)

$$u(t) = kp\, e(t) + \frac{Kp}{Ti}\int_0^t e(t)d\tau + kp\, Td\, \frac{de}{dt} \tag{3.3}$$

Using equations (3.1) and (3.2) in (3.3) gives:

$$u(t) = kp\, e(t) + \ ki\int_0^t e(t)d\tau + kd\frac{de}{dt} \tag{3.4}$$

Some common controller types and characteristics are as follows:

I. **P Controller:** A Proportional controller or P-Controller produces the control output proportional to the current error. It is a form of feedback control. The difference between set point and the process variable is called an error. The error value multiplied by the proportional gain (Kp) determines the output response. Increasing the value of proportional gain Kp results in an increase of speed of response. However, the process variable starts oscillating at a higher rate if Kc is increased beyond the normal range, leading to instability of the system. This particular controller is provided with manual reset in order to reduce the error when used alone. However, zero error state cannot be achieved by this controller as it produces deviation from the set point. Hence, P controller will always have a steady state error.

II. **I Controller:** The purpose of Integral controller or I-Controller is to reduce the steady state error of the system. It is a form of feedback control. As a result, the smallest of the error values produces a high integral response. The integral response decreases if the error is negative and increases if the error is positive. The speed of response is slow when I-controller alone is used, which improves the steady state response i.e., it removes any deviations that may exist. The speed of the response is increased by decreasing the integral gain Ki.

III. **D-Controller**: A Derivative controller or D-Controller measures the rate of change of speed per unit of time and produces the output proportional to the rate of change. The rate of change of derivative output is equal to the rate of change of error multiplied by a derivative constant. Unlike only P-controller and only I-controller, D-controller is a form of feed forward control which measures only the change in error. This controller is used when the process variable starts changing at a high rate of speed. In such a case, the derivative controller moves the final control device in a direction so as to counteract the rapid change of a process variable. Therefore, derivative controllers cannot be used alone for any control applications. If the derivative term is large, the D-controller responds to the changes in the process variable. The very benefit of D controllers is to resist change in the system, the most important of these being oscillations. In most controllers, derivative response depends only on process variables rather than the error. Also, most control systems use less derivative time td, as the derivative response is very sensitive to the noise in the process variable leading to produce extremely high output even for a small amount of noise.

IV. **Proportional-Integral (PI) Controller:**
PI-controller lacks the D derivative of the PID system. PI Controller is also a form of feedback controller as it provides a faster response time than only I-controller due to the addition of proportional action. It stops the fluctuations in the system and returns the system to its set point. The PI-controller response time is faster than the only I-controller, and it is still up to 50% slower than the only P-controller. Therefore, PI controllers are often combined with D-controllers to increase the response time.

V. **Proportional-Derivative (PD) Control:**
Another combination of controllers is the PD-controller, which lacks the I-controller of the PID system. It is a combination of feedforward and feedback control, as it operates on both the current process conditions and also the predicted process conditions. The control output is a linear combination of the error signal and its derivative in PD-controller.

VI. **Proportional-Integral-Derivative (PID) Control:**

Proportional-Integral-Derivative controller is a combination of all three types of control methods. It combines the advantages of each type of control, hence proving to be the most accurate and stable controller. It includes a quicker response time because of the only P-controller, along with the decreased offset from the combined derivative and integral controllers. The offset is removed by the addition of an I-controller. The addition of a D-controller greatly increases the controller’s response as it predicts disturbances to the system by measuring the change in error. Although the PID controller seems to be the most adequate controller, it is also the most expensive controller. PID controller correlates the controller output to the error, integral of the error, and derivative of the error.

A PID controller cannot be used in the process which includes noise, since the noise would interfere with the predictive, feedforward aspect because of the use of a derivative controller. However, the PID controller is used when the process requires no offset and a faster response time.

3.1.6 PID tuning goals

There are so many various methods for PID tuning[12], despite less tuning parameters, is the fact that the choice depends on the desired characteristics of the system. Different tuning methods can help to achieve different system behaviour, and some tuning methods also have tuning parameters that can be chosen for this purpose. The main trade-off is between high controller gains for performance and low gains for robustness and less input usage. Following properties are desired to obtain in the tuned control loop:

Set point tracking – The purpose of this is to track the change or error in the controller. It equalizes the set point and process variable, so that the operation begins with no error.

Disturbance rejection – It ensures to keep the desired output at the set point despite disturbances to the process.

Robustness – The ability to handle uncertainty.

Low input usage – High performance may demand a high degree of input usage, which can be expensive and wear out or damage the actuator.

3.2 Modelling

First principle and Data-Driven Models:

To build the first principle of modelling a system, it requires time, money, and adequate skill and this depends on the type of model to be designed. The first principle is applicable to limited areas like batch processes and continuous processes in process industries. First-principle models are generally made using engineering, which are defined based on physical laws[13]. Using these laws, balances are made for the model- mass balance, energy balance, etc. There are also other models that are measured based on quantity, physical laws, or device implementation knowledge, instead of relying on just data.

Data driven models or Compiled models are condensed and easier. Usually, data driven from empirical models are used because they are a simplified model of a bigger model, making it simpler to understand.

Data driven models are designed from the data taken from the input and output of the process or simulation models. As these models are converted to simpler form from the original data, they cannot be used for accuracy even if they represent the model well. It is wiser to use Empirical models[14] for faster collection of data and build the model, which minimizes the time spent of analysis and development time. Unlike the first-principles models which can extrapolate out of the test data used, the empirical models cannot use data other than that used for testing to develop the model. Another drawback of the Empirical models is that the model must be built all over if there are any changes in operations or the variables used. Ultimately, while choosing variables, first-principles modelling experts are asked to choose which variables are suitable for building the Empirical models.

In First-principles models, a lot of variables are used which are not seen while building empirical type models, variables that are used to measure physical properties in a model are not used in building Empirical models. The best way to aid the empirical method is to use the first-principle model to preprocess data and then use this data in the empirical method for better extrapolation, a hybrid approach will be more efficient.

Simplified versions of more complex models are made using the first-principles models are very complicated and making simpler versions of these models will make it compact and easy for understanding and access relevant data. The downside to this is that there will be some unavoidable loss of data that will be lost forever.

3.3 Mathematical Model of Control System

A mathematical model of any system is a smaller version of representing a process. There are three ways to represent a process model as mathematical models[15]:

1. Differential equation model
2. Transfer function model
3. State space model

3.3.1 Differential equation model

In a differential equation model, a system is described according to the differential equations and then uses them to make the model of the system.

3.3.2 Transfer function model

The Transfer function model simply uses the differential equations and then converts it using the Laplace transform to represent it in the s-domain.

A transfer function H(s) is given by Figure 3.3:

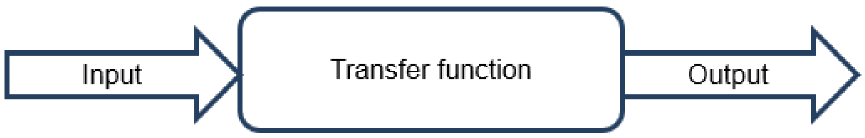


Figure 3.3: Transfer function

In short, in this type of model, the Transfer function, H(s), multiplied by the input, u(s), gives the output, y(s).

$$H(s) = \frac{y(s)}{u(s)} \tag{3.5}$$

3.3.3 State-Space Representation:

A mathematical model of a physical system, a state-space representation in control engineering, uses a set of data of inputs, outputs and state variables and relates it in first order differential equations. State variables evolve over time and change with imposed input variables, whereas output variables depend on the state variables.

The “state space” is the Euclidean space, within which state vectors can be represented on the axes, expressing the variables as vectors allowing abstract inputs, outputs, and states. If the system is linear, time-variant and finite-dimensional, a matrix form of differential equation is written (Katalin, 2001). While the state space method can be efficiently applied to research systems with modulation or without it, it also provides a compact way to analyse systems with multiple inputs and outputs[16].

Having a varied field of application, ranging from economics to statistics to computer science and electrical engineering and even neuroscience, state-space models can be used to identify trends and cycles and to identify factors or unobserved time-based series. (Kalman,1960; Harvey, 1990) [17]

4 Background and theory system identification

4.1 System Identification

A mathematical model using this system could describe physical as well as economical processes, ranging from an object falling under the influence of gravity to the stock markets reacting to a pandemic. Control systems rely extensively on system identification, in which a data driven model integrates with a controller design to form a formal controller optimality proof.

An important factor influencing the system identification is the quality of the input. Designing optimal experimental designs to yield maximally precise outputs is focused on. (goodwin 1977, walter 1997) [18]

System identifications are of two types: White box and Black box model.

4.1.1 White Box:

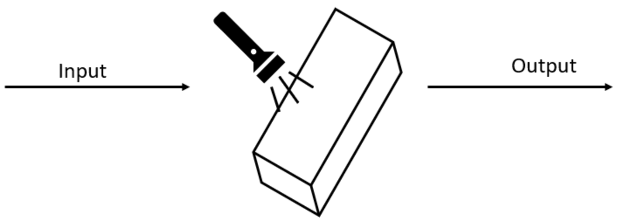


Figure 4.1: White box model

A “White box”, in areas of computing, engineering and science, is a system where a logical explanation for the inner workings and components are readily available for understanding[19].

4.1.2 Black Box:

A “Black box” is a system whose implementation is “opaque” or unclear. The inputs and outputs are known without knowledge of its internal workings as in the case of an algorithm or an engine. The term seems to have first come into use when Wilhelm Cauer in 1941 (cauer, 1941), published about electronic circuits responding to signals applied to their ports. Although there is documentation of earlier uses of such two terminal components as early as 1921 or even earlier (Belevitch, 1962), the theory was perfected during the 1960s. (Mario, 1963) [19]

Having been widely used in computing and mathematics, the black box theory is sufficiently explored in engineering and technology as well. More broadly, the black box theory can be used even in psychological factors influencing the mind and its behaviours.

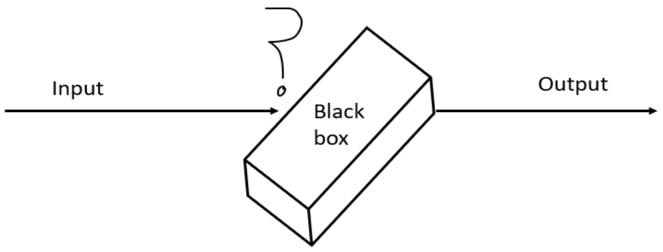


Figure 4.2: Black box model

Norbert Wiener in 1961 (Weiner, 1961) described a black box as an unknown system that can be identified using techniques of system identification. It uses statistical methods to determine a mathematical relation between the inputs and outputs. The design of experiments to generate data to fit such models and examine the influence of external influences is known as black box system identification. [19]

The “explanatory principle” hypotheses a causal relation between the input and the output where both entities are distinct and the system appears non-openable or black to the observer, but the inputs and outputs are observable or relatable.(glanville, 2009)

4.2 State space models

State space methods are used for representing a system and designing a controller. The classical transfer functions represent only a small portion of the output related to the Laplace transform of the input whereas an SSM can represent any system. It is a dynamic model, and it consists of 1st order DE[20] [21]. Hence, if a system has its order = k, then there will exist k different differential equations as shown in the form below.

$$\begin{aligned} \dot{x}1 &= \frac{dx1}{dt} = f1(\mathrm{x}1, \mathrm{x}2, \dots, \mathrm{xk}, \mathrm{u}1, \mathrm{u}2, \dots, \mathrm{ul}, \mathrm{t}) \\ \dot{x}2 &= \frac{dx2}{dt} = f1(\mathrm{x}1, \mathrm{x}2, \dots, \mathrm{xk}, \mathrm{u}1, \mathrm{u}2, \dots, \mathrm{ul}, \mathrm{t}) \\ &\dots\dots\dots\dots \\ \dot{x}k &= \frac{dxk}{dt} = f1(\mathrm{x}1, \mathrm{x}2, \dots, \mathrm{xk}, \mathrm{u}1, \mathrm{u}2, \dots, \mathrm{ul}, \mathrm{t}) \end{aligned} \tag{4.1}$$

The states and the inputs are put in a vector form, as shown below, and then the SSM can be made.

$$x = \begin{matrix} x1 \\ x2 \\ \cdots \\ \cdots \\ xk \end{matrix} \qquad u = \begin{matrix} u1 \\ u2 \\ \cdots \\ \cdots \\ ul \end{matrix} \tag{4.2}$$

$$\dot{x} = \frac{dx}{dt} = f1(\mathrm{x}, \mathrm{u}, \mathrm{t}) \qquad (4.3)$$

The SSM differential equations for linear systems are as below, where a and b are functions of time,

$$\dot{x}1 = \frac{dx1}{dt} = a11x1 + \cdots + a1kxk + b11u1 + \cdots + b1lul$$

$$\dot{x}2 = \frac{dx2}{dt} = a21x1 + \cdots + a2kxk + b21u1 + \cdots + b2lul$$

$$\ldots\ldots\ldots\ldots\ldots \qquad (4.4)$$

$$\dot{x}k = \frac{dxk}{dt} = ak1x1 + \cdots + akkxk + bk1u1 + \cdots + bklul$$

The state equation for a linear time variant model is used for controlling a process and is of the form:

$$\dot{x} = \frac{dx}{dt} = Ax + Bu \qquad (4.5)$$

Where A is the State matrix, order $k*k$, and the eigenvalues of A = poles of the system.

And B is the input matrix, order $k*l$, where l is the number of inputs.

They can be written as

$$A = \begin{bmatrix} a1,1 & \cdots & a1,k \\ \vdots & \ddots & \vdots \\ ak,1 & \cdots & ak,k \end{bmatrix} \qquad B = \begin{bmatrix} b1,1 & \cdots & b1,l \\ \vdots & \ddots & \vdots \\ bk,1 & \cdots & ak,l \end{bmatrix} \qquad (4.6)$$

The output equation has the same structure as the state equation,

$$y = Dx + Eu \qquad (4.7)$$

Where D is the output matrix, with order $m*k$, m is the number of outputs, E is the Feed through matrix, with order *m*1*.

4.2.1 System Definition

Let us assume that the system can be described as a linear, discrete-time invariant, strictly proper, combined deterministic and stochastic state space model, given by

$$Xk+1 = A\,xk + B\,uk \qquad \{Initial\ state\ x0 \tag{4.8}$$

$$yk = D\,xk + E\,uk \tag{4.9}$$

where $k \geq 0$, is discrete time, $x_k \in \mathbb{R}^n$ state vector, $n \geq 1$ the system order, $u_k \in \mathbb{R}$ control input, $y_k \in \mathbb{R}$ measured output, $v_k \in \mathbb{R}^n$ measurement noise. A- State transition matrix, B- external input matrix, D- output matrix.

Assume:

1. The pair (A,D) is observable, and pair(A,B) is controllable.
2. The system is stable.
3. For a nonlinear SSM, all the vectors and functions are assumed to be continuous.
4. Data might be from real systems.

PID controller ideally has the transfer function:

$$H(s) = Kp(1 + 1/Tis + Tds) \tag{4.10}$$

Where, Kp is the proportional gain, Ti is the integral time constant, and Td is the derivative time constant.

4.2.2 State model

In a multi-input and multi-output (MIMO) system, the output equation and the state equation collectively represent the dynamic system. Generally, in a MIMO system, the two equations (4.8) and (4.9) are derived for analysis. And writing any equation for this type of model is a state model[22]. To write an output equation, the output variables are used.

4.2.3 Introduction to State Space Analysis

In a state space system [23, p. 1], a group of variables are used which summarize the history of the system in order to predict the future status of the system. In any dynamic system, the initial conditions are known from the history of the system, and this data is used to predict the future values of the system. The State variables are the smallest set of variables that determine the state of a system.

The need for a state space system- Even with so many traditional methods like finding transfer function, the advantage of using an SSM is that it is faster than the other techniques, and there is no necessity for the SSM model to have all the conditions of the model, it only requires the initial condition. Using which, the future values can be estimated, and the system can be analysed. SSM is also known to provide accurate analysis results. Analysis of a MIMO system is also made easy in this method when compared to methods using the Transfer function. As there is a use of state vectors, all the inputs can be combined and placed as the state vector, making it easier for analysis as the data can be reused. SSM is applicable to all dynamic systems. The controllability of the control system is a very important aspect, especially for a MIMO model, and SSM shows the extent of how much the system can be controlled and how observable the system is. The biggest disadvantage is that, as SSM is usable on a MIMO system, but the system becomes equally complicated.

4.2.4 Plant Identification:

When a dynamic representation of the system is not available, a dynamic model using system identification techniques is explored, where the system is excited by a measurable signal and the output corresponding to it is collected. The resulting data is used to obtain a state-space model or a transfer function.

A Simulink model allows to simulate input-output data instead of measuring it. The system response to a stimulus is simulated and then a dynamic model is estimated based on the resultant output data. Once this data is obtained, a feedback controller can be designed to meet the objectives of the behaviour of the system.

A system identification toolbox software allows the use of a PID tuner to identify and control designs in a single interface. The tool allows identification of input-output data to identify models or to simulate the data and then identify one or more models.

4.2.5 Process Models

In chemical plants, a model relating the effect of a measurable input variable on an output quantity is often required in the form of a SISO plant. The overall system may be MIMO in nature, but the experimentation or simulation is carried out to measure the incremental effect of one input variable on a selected output. Such a proxy is obtained by collecting or simulating input-output data and deriving a process model (low order transfer function with unknown delay) from it[24].

4.3 DSR Toolbox

DSR is a method for system identification of combined Deterministic and Stochastic systems and Realization (DSR), the detailed work on how the toolbox was originally made is found in [25].

It is a subspace identification to identify the process model using known input and output data, and the system order. With DSR, the subspace model is given on a plate by running just one line of command. DSR toolbox is system identification simplified and made as a toolbox in MATLAB.

The DSR toolbox is used to obtain system matrices directly when the input and output data are provided as arguments, along with a few other optional arguments. The purpose of using this toolbox is, it is a well-known method that provides the system matrices that represent any process model in a state space model, the matrices A, B, D and E, explained in Page 32. The explanation of the arguments required are explained in Appendix B.

The DSR toolbox works on basis of Realization theory, where with the system order n as the input, it uses SVD (Single Value Decomposition method) to find the observability matrix and controllability matrix, obviously not seen while using the toolbox, which is used to find the system matrices. [17].

The installation of the Toolbox is available for free; the downloaded zip file of the toolbox needs to be added in the MATLAB path. The necessary instructions for installation are available in the webpage [26].

To calculate a discrete state space model, a simple way to use the below command in MATLAB:

$$[A, B, D, E, C, F, x0] = dsr\ (Y, U, L) \tag{4.11}$$

The dsr command takes the arguments as Y = Output data, U = Input data and L = a positive integer, where 1<= n<=L*m. L here is the maximum system order the user can specify for the system order.

5 Demonstration case

5.1 Modelling and Simulation

A Chemical process:

In this demonstration case, A chemical reactor is used as shown in Figure 5.1, with Two feeds reactant A and reactant B, that react to form the product C. The reactants are not converted completely, hence there will be some presence of all the chemicals A, B and C at the outlet. The reactor kinetics for this model is given by:

$$r_C = k * x\_A * x\_B$$

Where,

r_C is the reaction rate for the formation of the Product C,

k is the reaction rate constant,

x_A is the concentration of A, and

x_B is the concentration of B.

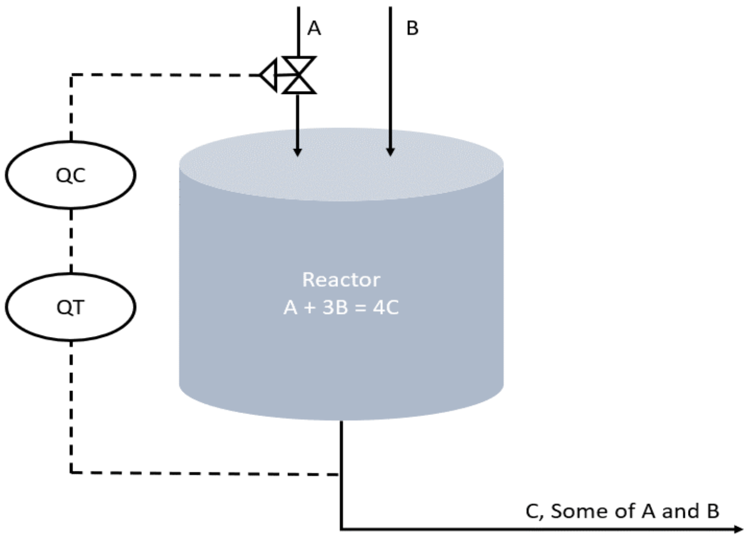


Figure 5.1: Chemical Plant model

It is known that the reactor needs to have a volume as it needs to have the process dynamics. The model has a Quality controller (QC) that allows control of the feed of reactant B. The Control will be made with the help of the measurements (QT) taken from the outlet, that checks the concentration of B (B_conc) and finally to manipulate the feed of reactant B (B_in).

After the setup is made Two types of Simulators are built, one for Open loop, where there is no controller, and second for Closed loop, with a controller in this chapter. Later system identification is done on both the models and finally a comparison of the models is done.

5.1.1 Chemical Reactor:

In a normal Industrial chemical production, there are Two factors on focus: The Chemical Kinetics and the Reactor design. The aim is to have maximum efficiency with the safest design. The reaction can be exothermic or endothermic or may have a constant temperature-Isothermal reactor. But the main focus here is not involved in Energy balance, and the experiment is assumed to be an Adiabatic reactor- no heat interchange. The main focus here would be to use the mass balance. In a chemical Reactor, the simplest way to define it, the Reactants react with each other with specific conditions to form the products. The Chemical reactors are classified based on the mode of operation:

1. Batch reactor
2. Semi batch reactor
3. Flow reactor
    a. Tubular reactor
    b. CSTR- Continuous Stirred Tank Reactor

5.1.2 CSTR- Continuous Stirred Tank Reactor

<u>Theory</u>:

The contents are well stirred and uniform throughout the reactor. The CSTR, usually is run in a steady state to have a mixed consistency and have an even concentration, reaction rate and other factors. With this system, it can also be said that the concentration is the same even at the exit points.

Steady state: In any process, the state of the reaction where the state variables are observed to be constant even with the process trying to change these state variables.

Time constant: After a step change is made, the time taken by the CSTR to reach 63% of the time taken for the process to reach steady state.

<u>Assumptions</u>:

1. The density and the heat capacity are assumed constant.
2. The reacting mixture is assumed to be mixed well.
3. The heat losses to the outside from the reactor is neglected.
4. The overall heat transfer coefficient is assumed constant.

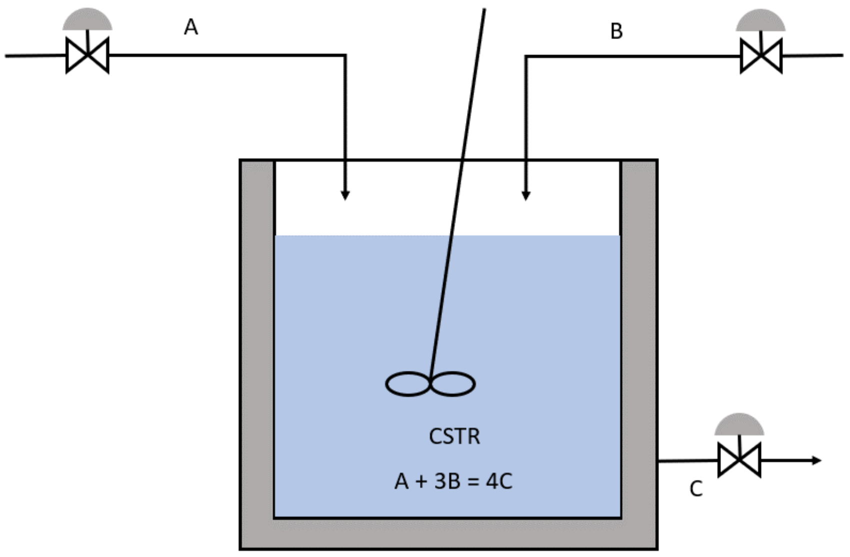


Figure 5.2: CSTR model

Figure 5.2 is a model made for the CSTR model. There are 3 ODE’s in this model and the below equations are written in the state space form. These equations were made using simple dynamic mass balance. The accumulation of the mass(dm/dt) is the mass left in the reactor after reaction, Or,

$$Accumulated = inflow - outflow - reacted \tag{5.1}$$

The State equations for the CSTR are:

$$dmdt\_A = A\_in - x\_A * f\_out - 0.25 * r\_C \tag{5.2}$$

$$dmdt\_B = B\_in - x\_B * f\_out - 0.75 * r\_C \tag{5.3}$$

$$dmdt\_C = r\_C - x\_C * f\_out \tag{5.4}$$

Where,

dmdt_A is the accumulation of mass of component A (for example kg/s or ton/hour).

dmdt_B is the accumulation of mass of component B (for example kg/s or ton/hour).

dmdt_C is the accumulation of mass of component C (for example kg/s or ton/hour).

A_in is the mass flow feed into the reactor of A (for example kg/s or ton/hour).

B_in is the mass flow feed into the reactor of B (for example kg/s or ton/hour).

x_A is the mass fraction of A inside the reactor (mass of A divided by total mass)

x_B is the mass fraction of B inside the reactor (mass of B divided by total mass) x_C is the mass fraction of C inside the reactor (mass of C divided by total mass)

f_out is the mass flow out of the reactor, which is the sum of the inflows.

r_C is the reaction rate to form component C.

The model is designed so that 1 mol of chemical A is reacting with 3 moles of B to form 4 moles of C, simply assuming the molecular weight for A and B to preserve the mass balance that dictates how much of Product C is generated.

A MATLAB function is written to represent the equations 5.2, 5.3 and 5.4 are written in Appendix A.

5.2 Open loop simulation

Input : The feed B-B_in_o

Output: Conc of B-raw_m_B_o

Input, u
Process
Output ,y

Figure 5.3: Open loop model

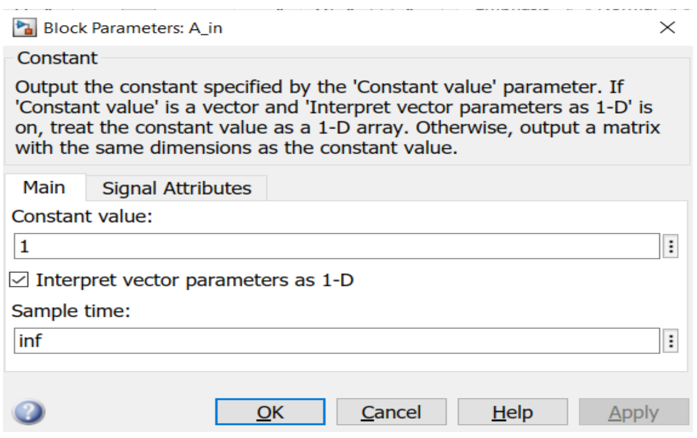
5.2.1 Construction of Simulink model

In the open loop, the input is the feed of B, denoted as B_in_o in this section for reference, and the output is the concentration of B, named as raw_m_B_o. The idea here is to build a simulator in MATLAB Simulink which will have the same behaviour of a real-life setup. The concentration of B inside the reactor is given by:

$$conc.B = y = m\_B \,/\, (m\_A + m\_B + m\_C) \tag{5.5}$$

5.2.1.1 Block A_in

To make the inlet for the chemical A to enter the reactor, an integer block parameter with a constant value of 1 is made. This block is called A_in shown in Figure 5.4.


Block Parameters: A_in

Constant

Output the constant specified by the 'Constant value' parameter. If 'Constant value' is a vector and 'Interpret vector parameters as 1-D' is on, treat the constant value as a 1-D array. Otherwise, output a matrix with the same dimensions as the constant value.

Main | Signal Attributes

Constant value: 1

☑ Interpret vector parameters as 1-D

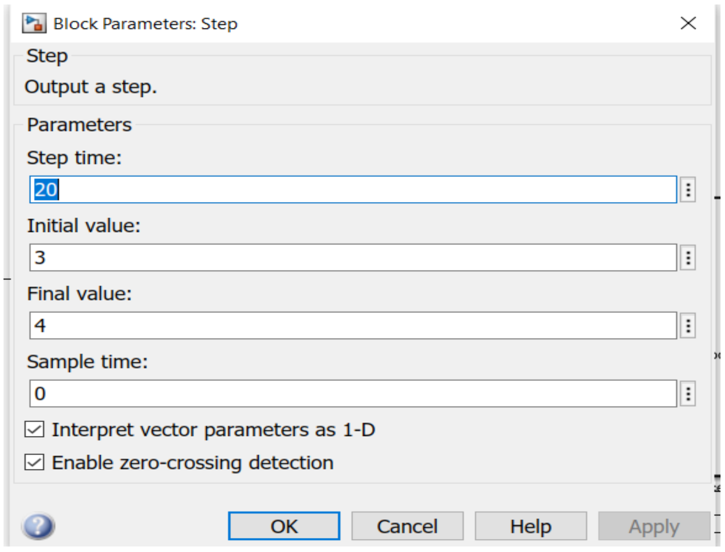
Sample time: inf

Figure 5.4: Constant Block for A_in

To make the second inlet, for chemical B, a step response block is placed to observe the behaviour of the system. This Feed of B is the input of the system and will be used later for calculations and analysis. This set of data will be named as B_in, and B_in_o saying that the data is from an open simulation.

5.2.1.2 Step Change:

A step change is given to the system to observe the behaviour of the model after a disturbance is applied. The Step change is applied after 20 seconds allowing the model to reach a steady state before applying the step. A Step block as shown in Figure 5.5 is made in Simulink is used to show this response from 3 to 4.


Block Parameters: Step

Step

Output a step.

Parameters

Step time: 20

Initial value: 3

Final value: 4

Sample time: 0

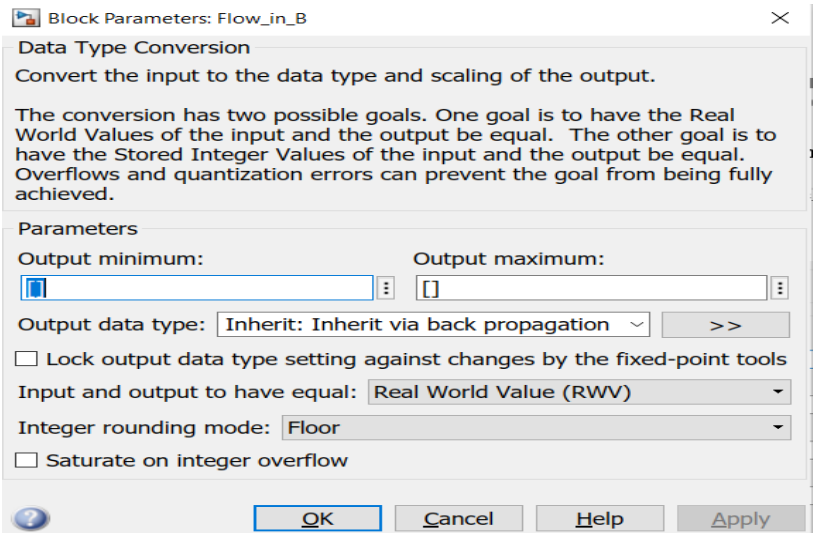
☑ Interpret vector parameters as 1-D

☑ Enable zero-crossing detection

Figure 5.5: Step change block

5.2.1.3 Covert Block

The Convert Block Figure 5.6 is used to make sure the input to the model is of the required data type. The input and output of this Block are set to Real World Values.


Block Parameters: Flow_in_B

Data Type Conversion

Convert the input to the data type and scaling of the output.

The conversion has two possible goals. One goal is to have the Real World Values of the input and the output be equal. The other goal is to have the Stored Integer Values of the input and the output be equal. Overflows and quantization errors can prevent the goal from being fully achieved.

Parameters

Output minimum: []

Output maximum: []

Output data type: Inherit: Inherit via back propagation

☐ Lock output data type setting against changes by the fixed-point tools

Input and output to have equal: Real World Value (RWV)

Integer rounding mode: Floor

☐ Saturate on integer overflow

Figure 5.6: Convert block

5.2.1.4 A MATLAB function block

A MATLAB function block, Figure 5.7 is used to write a piece of code inside and connect the inputs and the outputs directly as ports. The main working of the CSTR model is written, and the code available in Appendix A. In this function block, a MATLAB function is written, where the differential equations of the chemical reactor shown in equations 5.2, 5.3 and 5.4 are written. The function has inputs A_in, B_in, m_A, m_B and m_C, and the Outputs from the function are differentials of the masses of A, B and C with respect to time given by dmdt_A, dmdt_B and dmdt_C respectively. The function is written in Appendix A.

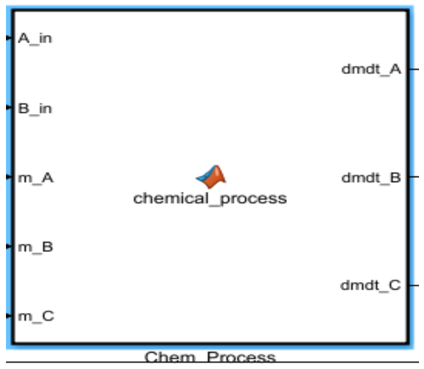


Figure 5.7: MATLab function block

As the function block here gives the derivatives of masses of A, B and C, these values need to be changed to get the current values. Hence, each of these outputs have an integrator block placed to obtain m_A, m_B and m_C. These values can be used as the input for the MATLAB function block, and the outputs for the next time step are determined this way, the same way a dynamical system works.

5.2.1.5 Integrator block

The first integrator block is made for the dmdt_A. Using an integrator block will convert this value to m_A, which is the m_A inside the chemical reactor after the current time step. The initial condition is set to 5. The Lower saturation limit is set to 0 as the value of A being fed can never be negative in real life.

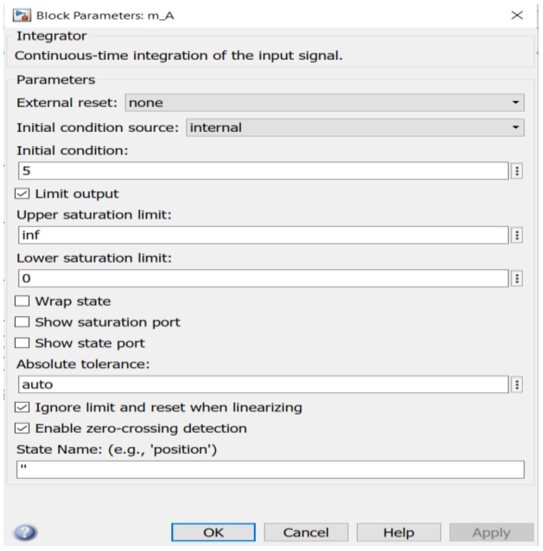
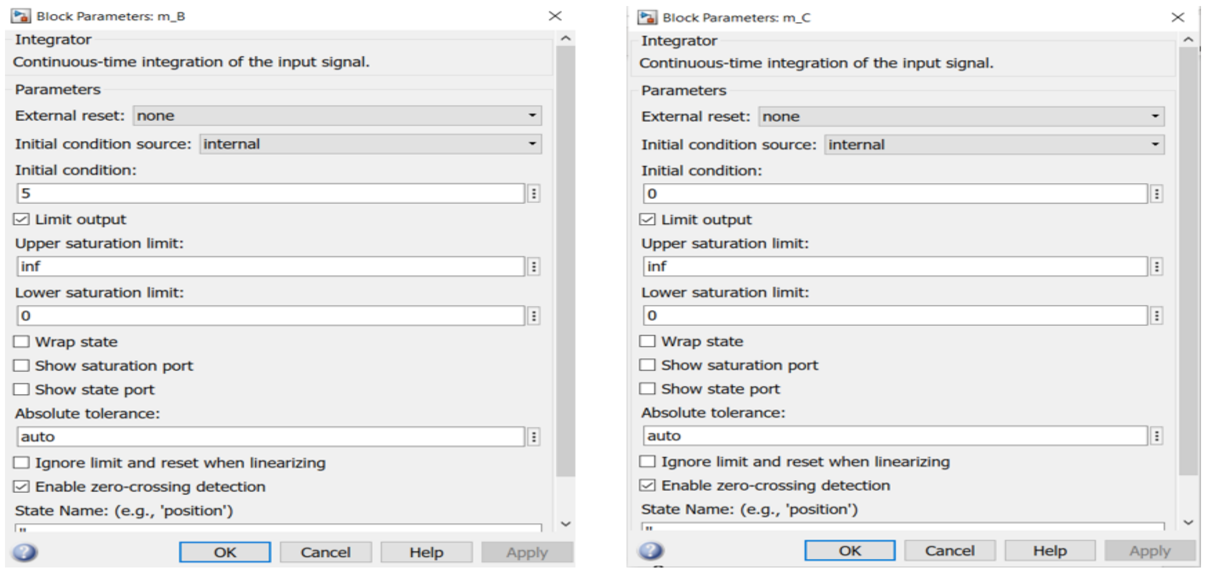

Block Parameters: m_A
Integrator
Continuous-time integration of the input signal.
Parameters
External reset: none
Initial condition source: internal
Initial condition:
5
☑ Limit output
Upper saturation limit:
inf
Lower saturation limit:
0
☐ Wrap state
☐ Show saturation port
☐ Show state port
Absolute tolerance:
auto
☑ Ignore limit and reset when linearizing
☑ Enable zero-crossing detection
State Name: (e.g., 'position')
''

Figure 5.8: Integrator block for m_A

In the same way, the integrator is made for m_B and m_C as well in Figure 5.9. The initial condition is 5 for m_B, and ror m_C, initial condition is 0, as the Product is not formed at time =0, and the lower limit is set as 0 for both.



Block Parameters: m_B

Integrator

Continuous-time integration of the input signal.

Parameters

External reset: none

Initial condition source: internal

Initial condition:

5

☑ Limit output

Upper saturation limit:

inf

Lower saturation limit:

0

☐ Wrap state

☐ Show saturation port

☐ Show state port

Absolute tolerance:

auto

☐ Ignore limit and reset when linearizing

☑ Enable zero-crossing detection

State Name: (e.g., 'position')

''

OK Cancel Help Apply

Block Parameters: m_C

Integrator

Continuous-time integration of the input signal.

Parameters

External reset: none

Initial condition source: internal

Initial condition:

0

☑ Limit output

Upper saturation limit:

inf

Lower saturation limit:

0

☐ Wrap state

☐ Show saturation port

☐ Show state port

Absolute tolerance:

auto

☐ Ignore limit and reset when linearizing

☑ Enable zero-crossing detection

State Name: (e.g., 'position')

''

OK Cancel Help Apply

Figure 5.9: Integrator block for m_B and m_C

5.2.1.6 Find Concentration of B

The output desired for analysis is the concentration of B, B_conc, denoted in this section as raw_m_B_o for the open loop simulation. As mentioned earlier, in Equation 5.5, the below design is made to achieve the concentration of B.

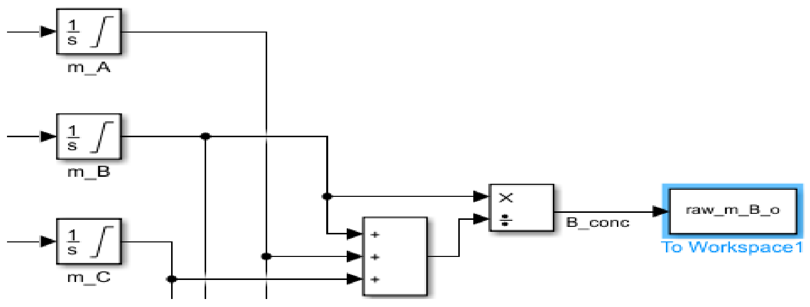
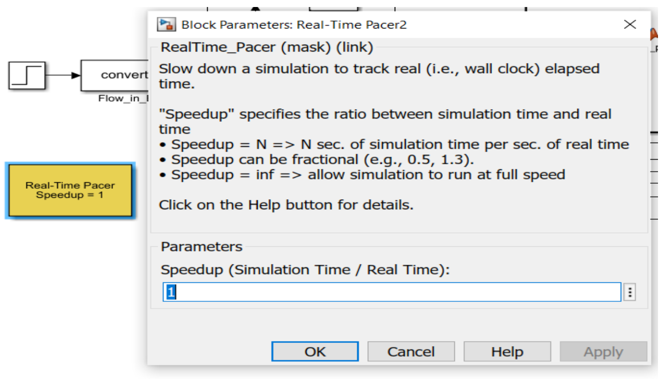


Figure 5.10: Concentration of B

The masses of A, B and C, m_A + m_B + m_B, are first added, and then the mass of B, m_B is divided by this to get the concentration of B, raw_m_B_o shown in Figure 5.10.

5.2.1.7 Real-Time pacer

A real time pacer is added in the model to make observations as if the simulation is running in real time. The model runs in real time and the speed can be altered according to the requirement of observation.


Block Parameters: Real-Time Pacer2

RealTime_Pacer (mask) (link)

Slow down a simulation to track real (i.e., wall clock) elapsed time.

"Speedup" specifies the ratio between simulation time and real time
• Speedup = N => N sec. of simulation time per sec. of real time
• Speedup can be fractional (e.g., 0.5, 1.3).
• Speedup = inf => allow simulation to run at full speed

Click on the Help button for details.

Parameters

Speedup (Simulation Time / Real Time): 1

Figure 5.11: Real time pacer

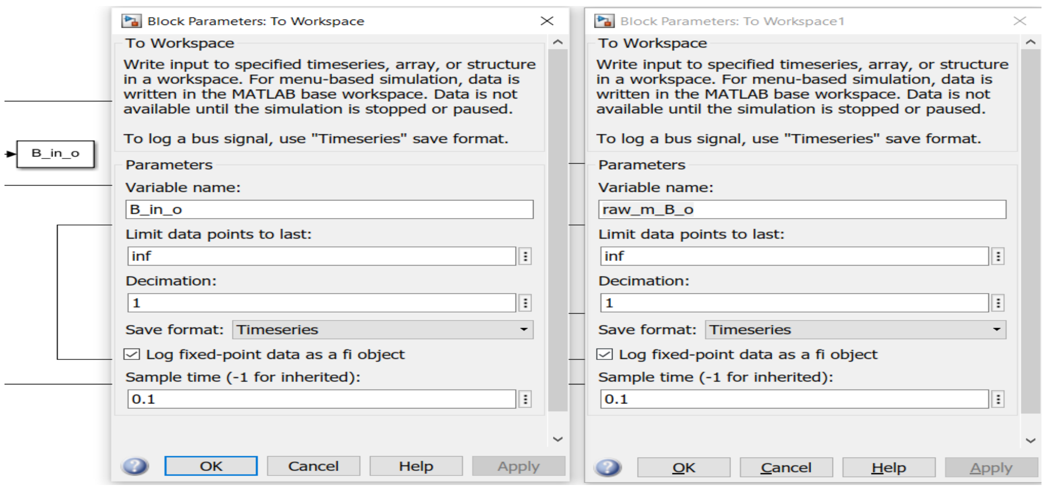
5.2.1.8 Scope

A scope block is finally added to show real time plotting of all the variables mentioned above. The scope block allows us to observe the plotting of these modules.

5.2.1.9 ToWorkspace

A ToWorkspace block enables the Simulink model to save the data of the variables with respect to the time and can be used in the MATLAB workspace for further analysis.

There are Two ToWorkspace blocks here, Figure 5.12: B_in_o- the feed of B in open loop and B_conc- raw_m_B_o. The data is saved for further calculations and to use in model identification. The format for saving is Timeseries, as the values stored in the variable will have the time stored in reference to the corresponding value at that time. The Sample time is generally set to 0.1, this gives the recordings of the variables with the time step of 0.1 in the workspace.


Block Parameters: To Workspace

To Workspace

Write input to specified timeseries, array, or structure in a workspace. For menu-based simulation, data is written in the MATLAB base workspace. Data is not available until the simulation is stopped or paused.

To log a bus signal, use "Timeseries" save format.

Parameters

Variable name: B_in_o

Limit data points to last: inf

Decimation: 1

Save format: Timeseries

☑ Log fixed-point data as a fi object

Sample time (-1 for inherited): 0.1

Block Parameters: To Workspace1

To Workspace

Write input to specified timeseries, array, or structure in a workspace. For menu-based simulation, data is written in the MATLAB base workspace. Data is not available until the simulation is stopped or paused.

To log a bus signal, use "Timeseries" save format.

Parameters

Variable name: raw_m_B_o

Limit data points to last: inf

Decimation: 1

Save format: Timeseries

☑ Log fixed-point data as a fi object

Sample time (-1 for inherited): 0.1

Figure 5.12: ToWorkspace block

The final setup for the open loop was designed as shown below.

5.2.1.10 Simulink design

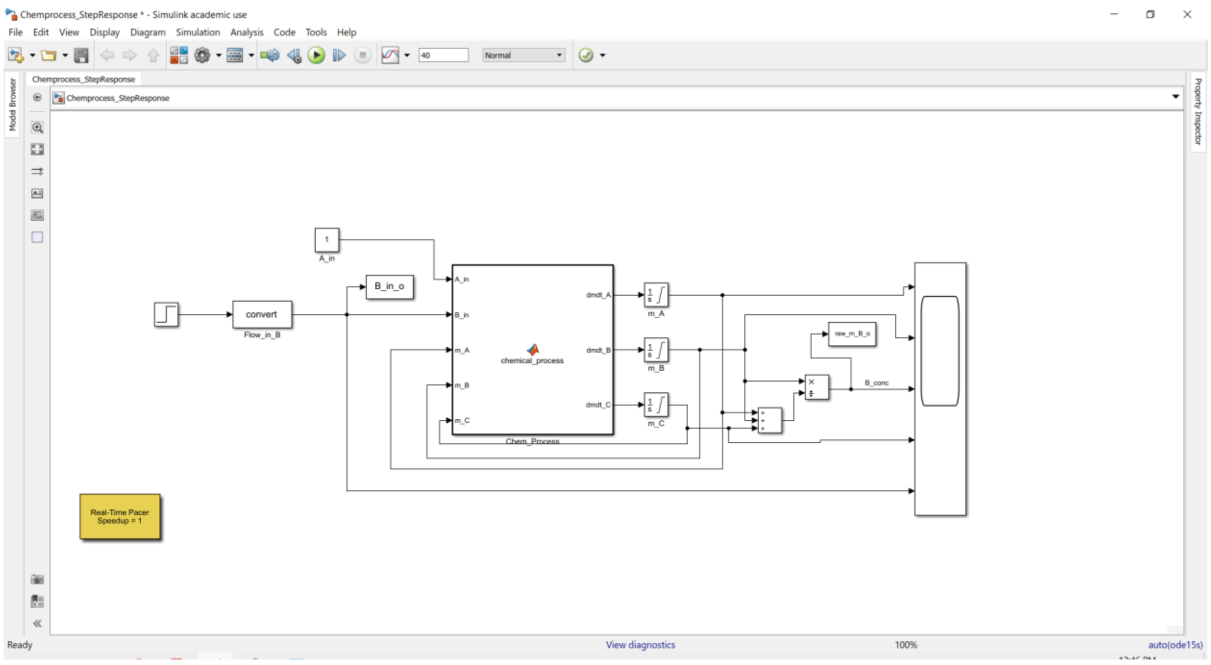


Figure 5.13: Simulink model designed

The next image Figure 5.14 shows the behavior of the open loop system made.

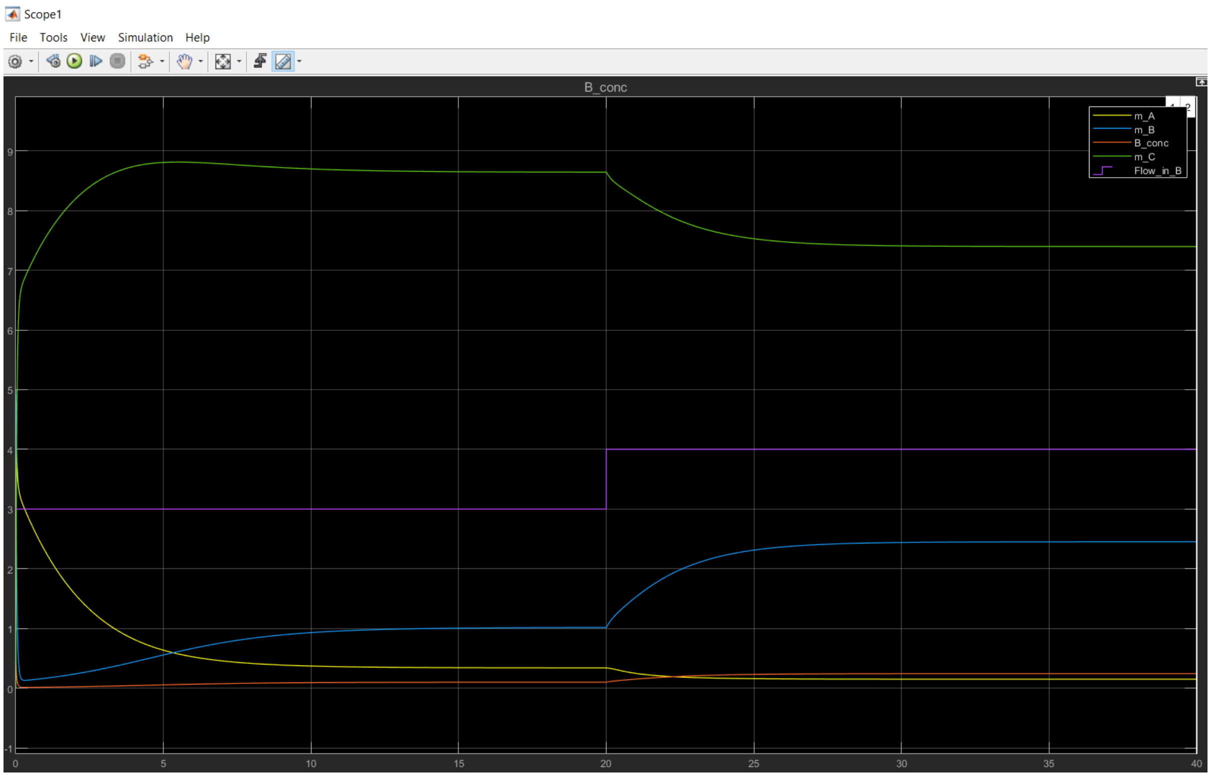


Figure 5.14: Simulink response

The legend on the Top left corner in Figure 5.14: Simulink response. The simulator first runs for 20 seconds and then a step response is applied to the Flow_in_B, which is the inlet feed of B from 3 to 4, the Purple line here. The Blue line represents the mass of B, m_B showing a step response. The output and variable in focus is the concentration of B, B_conc in the Red line, showing a smaller step response. More inference is seen later in the Chapter.

5.2.2 Open loop system behavior

To see how the system works, A PRBS signal is applied in place of the Step Response. After many trials, the sample time was decided to be 2 seconds to allow m_B to reach its steady state and have more accurate data.

5.2.2.1 PRBS as input:

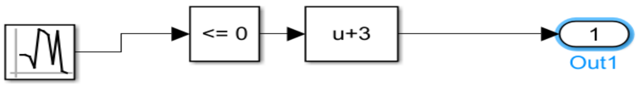


Figure 5.15: PRBS for B_in

The logic behind the PRBS signal is, a Random number block is used, which is connected to "Compare to constant" Block is used, where it compares if the value is lesser than 0, the output is 0, if the value is greater than 0, then the output is 1; it works on a True or False logic. A Bias of 3 is added next as the PRBS is required from 3 to 4(u+3 is the lower limit). All of this is placed as a subsystem to make the model compact as shown in Figure 5.15.

This subsystem is named as "PRBS_Binary_Signal" and connected to the main Chemical process model Figure 5.16.

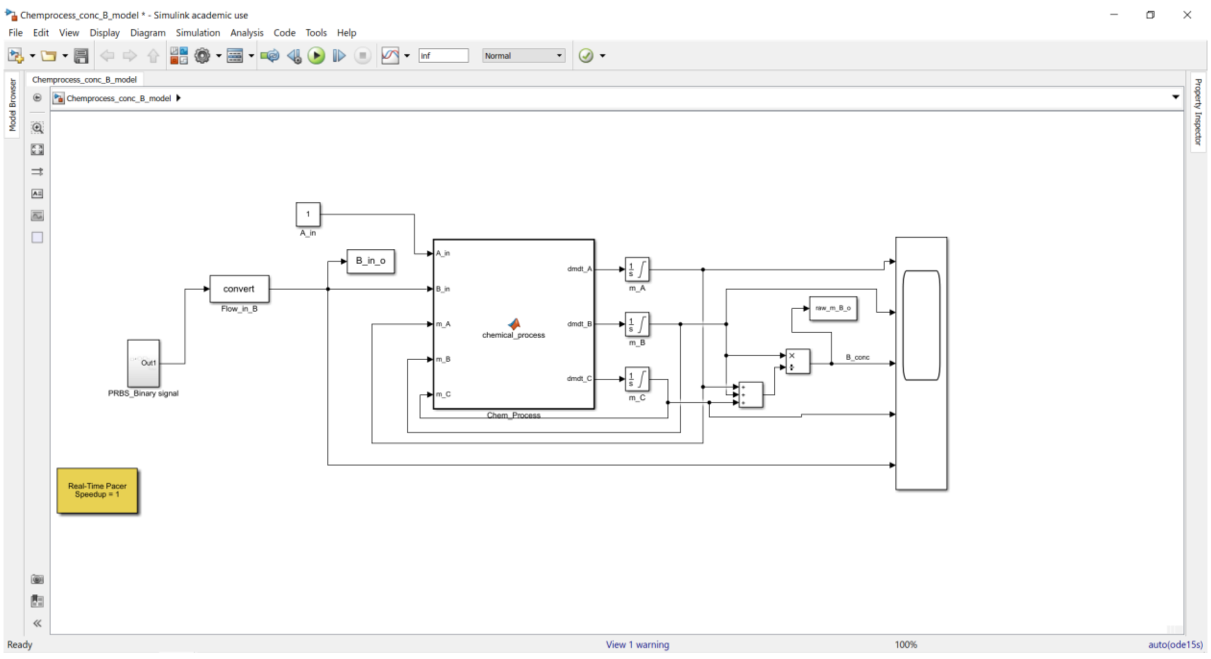


Figure 5.16: Open loop PRBS model

The response for the simulation is shown below in Figure 5.17. The plot legend and the colours assigned to each variable remain the same as the above step response setup.

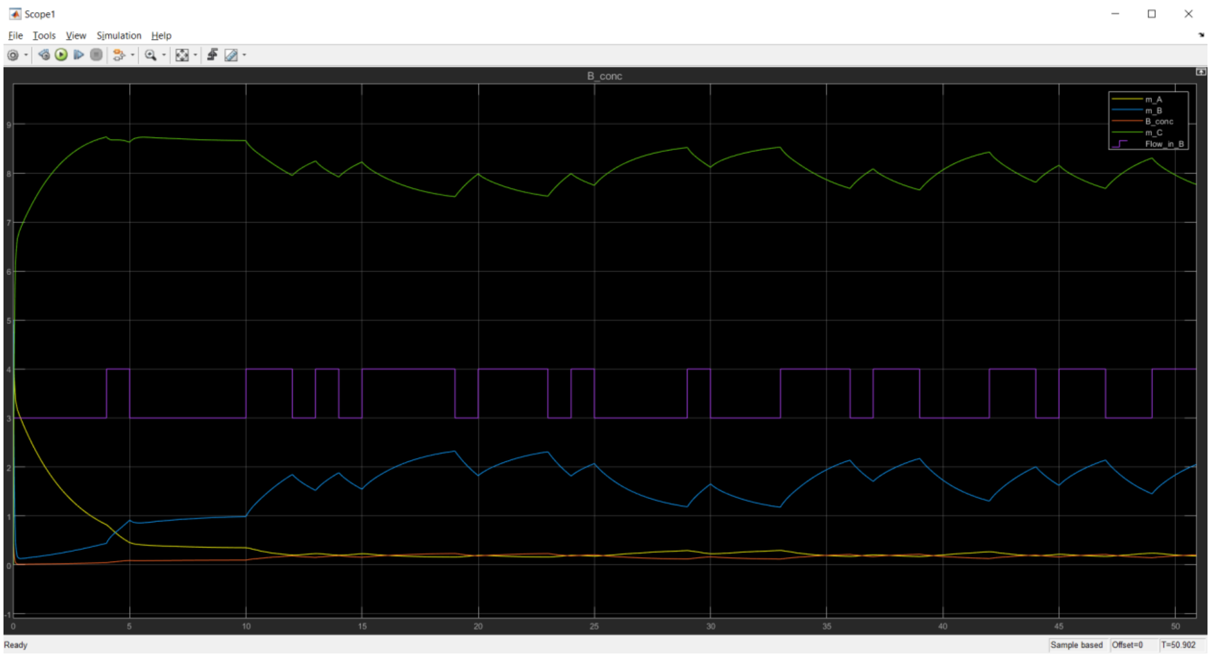


Figure 5.17: PRBS response for open loop

In comparison with the Step response above, the input signal Flow_in_B, the Purple line, shows a PRBS signal and all the other variables show the behaviour accordingly. The m_A is utilized the least around 0.3 represented by the Yellow line. The Blue line m_B shows the m_B in the reactor and ranges around 1 and 2 with changes in the PRBS signal and the m_C between 8.5 and 9.5. The output of the model, concentration of B, B_conc in Red colour, is seen between 0.1 and 0.2.

5.2.2.2 Saving the Simulink results

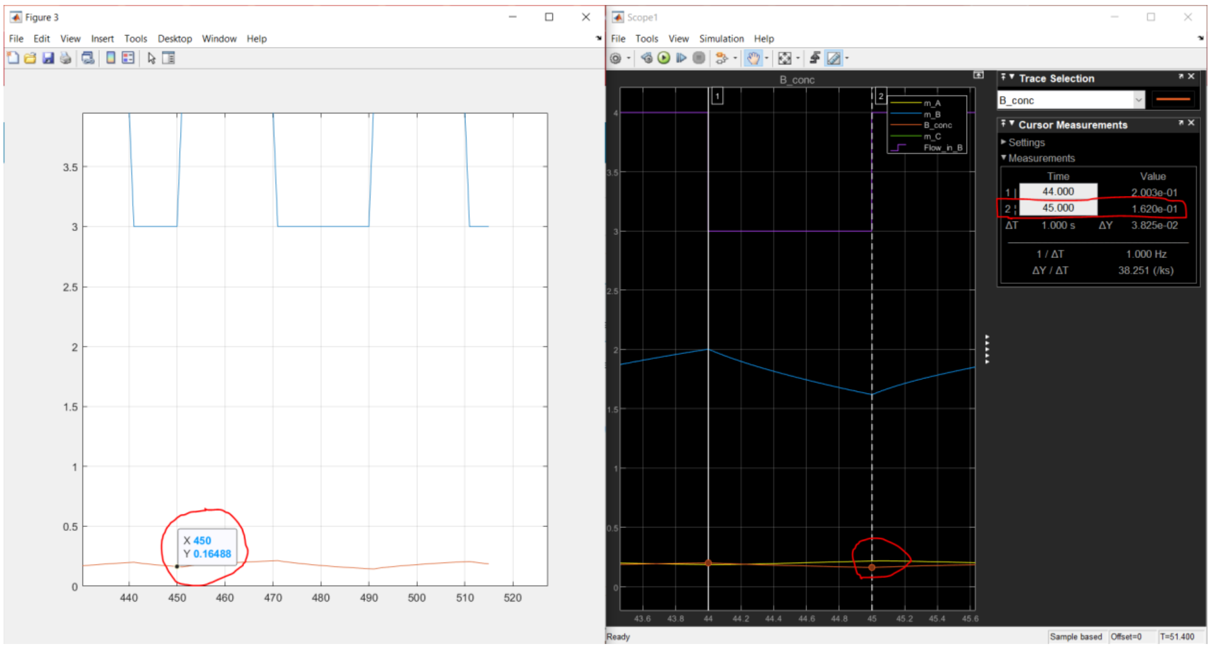
This data is first saved into variables as input u_o and output y_o from the Simulink.SimulationOutput. The Two sets of data are saved into excel sheets using the command “xlswrite(‘Open_input.xlsx',u_o)” and “xlswrite('Open_output.xlsx',y_o)”, which could be reused in the future for analysis. To load the same the command “u_o = readtable('Open_input.xlsx', 'range', 'B1:B515');” and “y_o = readtable('Open_output.xlsx', 'range', 'E1:E515');” are used. And u_o = u_o{:,:}; is used to write the data into an array.

5.2.2.3 Checking for Synchronization

To check for synchronization of data, the data saved and loaded is the same, the input u_o and output y_o are used to plot using the below commands:

```
% synchronize u and y...
>> figure(2)
>> clf
>> plot([u_o, y_o])
>> grid on
```

The plot obtained from the above command is in the RHS and the plot from the simulator is in the RHS of the below image. For comparison, the cursors are placed in both the plots at time = 45s = 450th time step.



```
        Time        Value
1 |    44.000      2.003e-01
2 ¦    45.000      1.620e-01
ΔT     1.000 s  ΔY  3.825e-02

       1 / ΔT      1.000 Hz
      ΔY / ΔT      38.251 (/ks)
```

Figure 5.18: Synchronization

In Figure 5.18 above, at 45 seconds, y = 0.16 in both the plots. Confirming that the output is saved correctly, and the data are in synchronization.

5.2.2.4 Model Identification:

The input and output data from above are used for identification using the DSR toolbox, Description given in Appendix B. The DSR toolbox is used to determine the system matrices as given by the equation:

$$x_{t+1} = A\,x_t + B\,u_t + C\,e_t, \quad x_{t=0}=x0, \tag{5.6}$$

$$y_t \quad = D\,x_t + E\,u_t + e_t, \tag{5.7}$$

As mentioned in the System identification section,

A = State matrix,

B = Input matrix,

D = Output matrix,

E = Direct feed through matrix,

C = CF*inv(F), (Kalman gain, K =inv(A)*C and C=A*K),

As a requirement to use the DSR toolbox, the means of the input and output are used as shown in the below commands:

```
>> u_bias = mean(u_o);
>>y_bias = mean(y_o);
```

Using the input, output, and their biases in order to be scaled and system order n = 1, the below command is given to obtain the system matrices:

```
>> [A,B,D,E,CF,F,x0] = dsr(y_o-y_bias, u_o-u_bias,1)
Ordering the given input output data
QR decomposition

A =    0.9579

B =    0.0069

D =     1

E =   3.4993e-04

CF =    0.0192

F =   6.4137e-04

x0 =    0.3446
```

5.2.2.5 Making a step response

After obtaining the system matrices for the open loop with system order n = 1, the step response is plotted for u= 3 to 4. A step response is applied at time = 500. Here u will be the input and y will be the output for plotting purposes to compare later with the original open loop Simulink model.

```
>> dt = 0.1;
>> t = 0:dt:100;
n = length(t);
>> u = ones(1,n)*3;
steptime = 500;
u(steptime+1:n) = ones(n-steptime,1)*4;
u = u - u_bias;
```

5.2.2.6 Generating the output using the system matrices

Next the equations (5.6) and (5.7) are used in the code to generate the output of the step response with the system matrices A, B, D and E as shown in the equations.

```
x = zeros(1,n);
y = zeros(1,n);
>>
for i = 1:n-1
```

```
    x(:,i+1) = A*x(:,i) + B*u(:,i);
    y(:,i) = D*x(:,i) + E*u(:,i);
end
```

The subtracted biases are now added for scaling, to make sure the plotting is appropriate.

```
% Add bias/scaling to get original values
u = u + u_bias;
y = y + y_bias;
>>figure(3)
clf
plot(t,[y', u'])
grid on
```

The above code can be found in Appendix C.

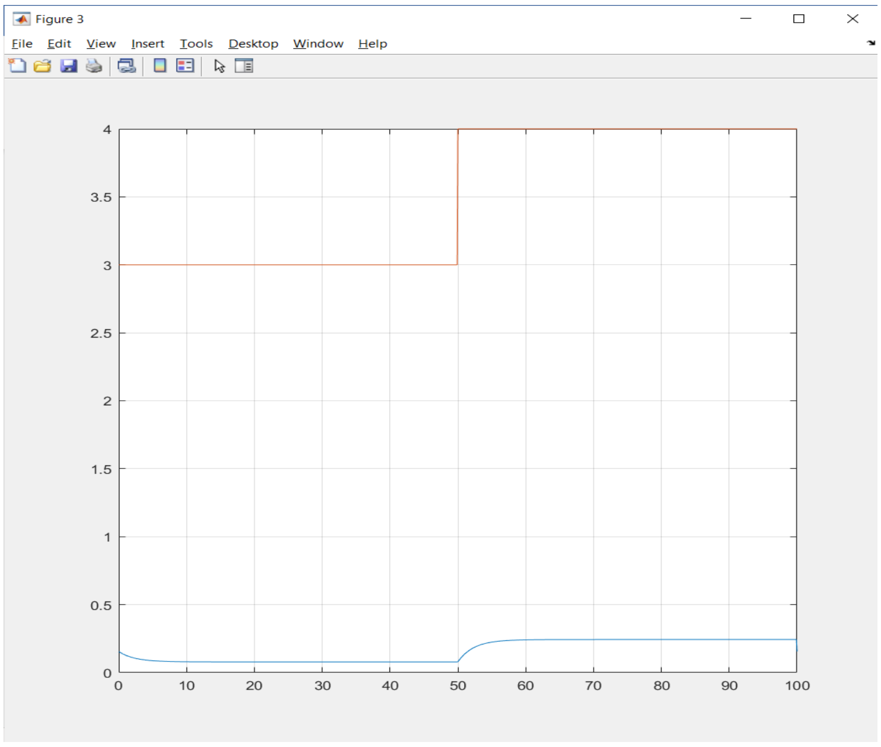


Figure 5.19: Step response from system matrices

The step response obtained using the Simulink model and then compared with the step response using the system matrices found in Figure 5.19.

5.2.2.7 Comparing Simulink model with identification model

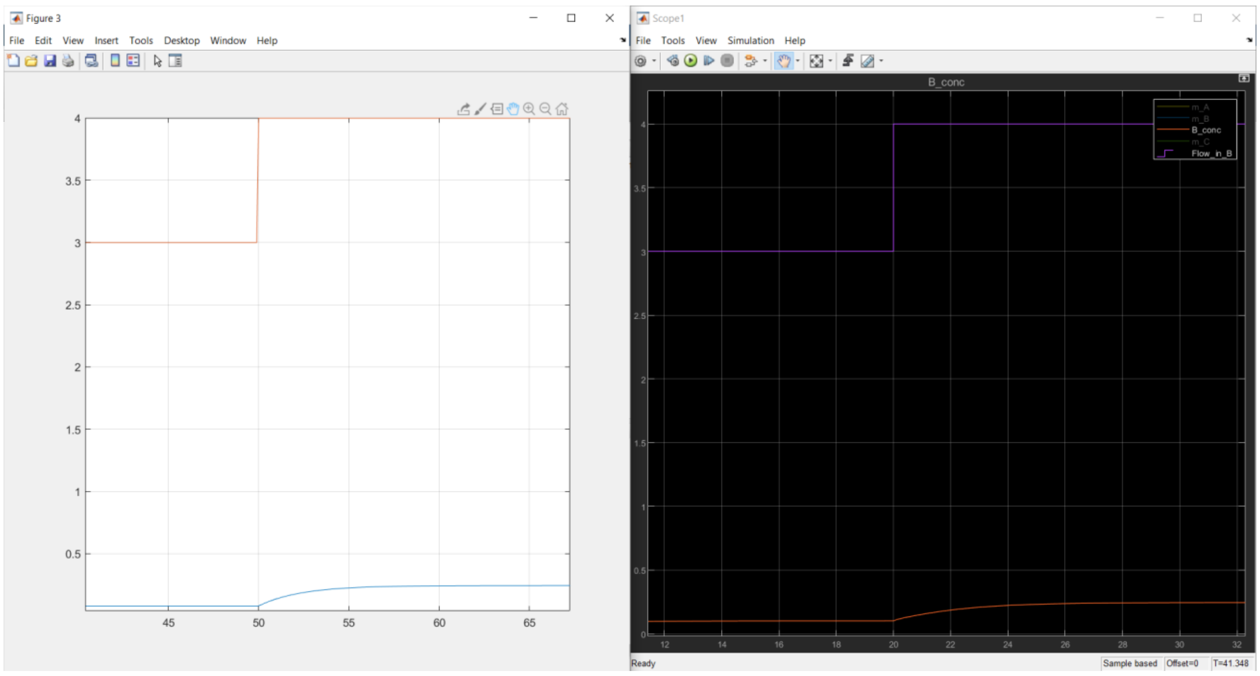


Figure 5.20: Compare Simulink with system identification model

The output curve reaches 100% after 10 seconds in both models. The gain is 0.2 and the time constant is 2.8 seconds in both the plots. Hence confirming that the system identification model represents the exact behaviour as the original model as shown in Figure 5.20.

Similarly, the system matrices are found for when system order n=2 and n=3 and then step responses are made for comparison.

5.2.2.8 System order n = 2

The only changes to Appendix C in command given are n = 2 in dsr and defining x, as shown below.

```
[A,B,D,E,CF,F,x0] = dsr(y_o-y_bias, u_o-u_bias,2)
System order ? .................... ( 1 ) =? 2

A =     0.9663   -1.7385
       -0.0007    0.7648

B =    -0.0103
       -0.0004

D =   -0.7194   -0.6946
```

```
E =  -9.5405e-06

CF =  -0.0017
       0.0003

F =   1.0754e-04

x0 =  0.1786
     -0.0175

And x = zeros(2,n);
```

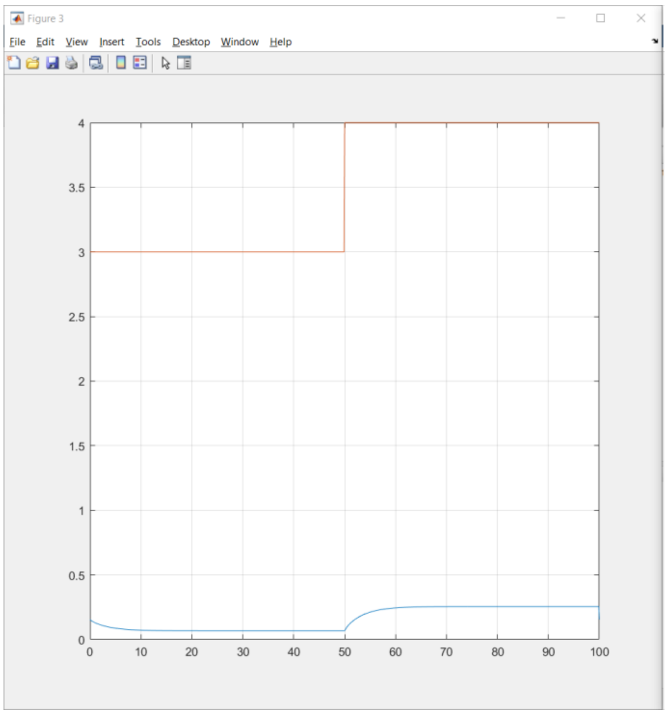


Figure 5.21: Step response for n=2

The output curve in Figure 5.21 reaches 100% after 10 seconds. The gain is 0.2 and the time constant is 2.8 seconds for this model as well.

5.2.2.9 System order n = 3

And the final trial is done for system order n= 3. And the same change in command applies while using the dsr command and defining x to obtain the system matrices in Appendix C.

```
[A,B,D,E,CF,F,x0] = dsr(y_o-y_bias, u_o-u_bias,3)
System order ? .................... ( 1 ) =? 3

A =    0.9669    1.1247    1.5245
```

```
       0.0010    0.8548    2.8010
      -0.0001    0.0069    0.7476

B =   -0.0119
       0.0008
      -0.0001

D =   -0.5971    0.7136   -0.3665

E =   5.5888e-06

CF =  1.0e-03 *

   -0.9200
   -0.3159
   -0.0151

F =   1.0699e-04

x0 =  0.2236
      0.0200
      -0.0079

And x = zeros(3,n);
```

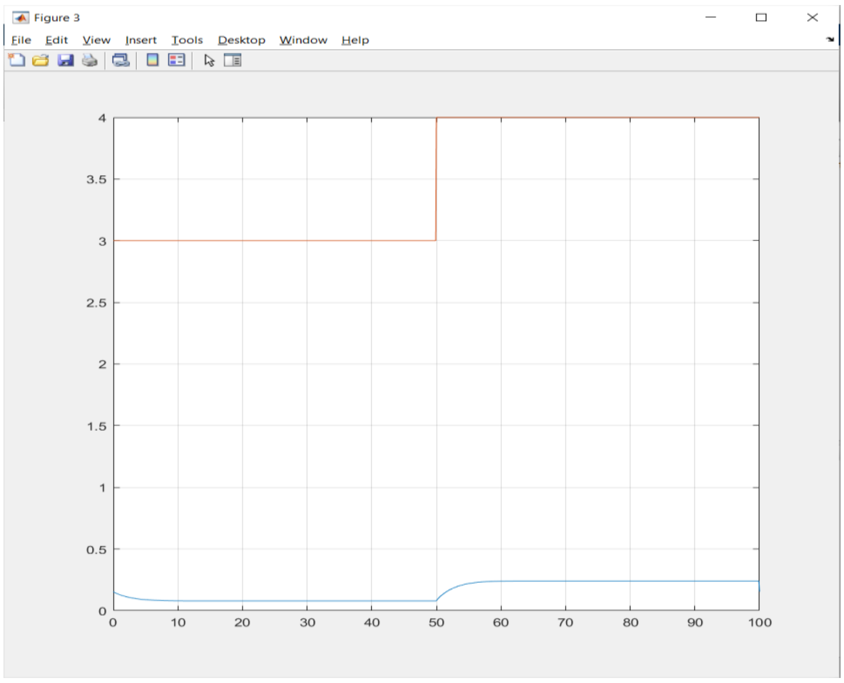


Figure 5.22. Step response for n=3

The output curve in Figure 5.22 reaches 100% after 10 seconds. The gain is 0.2 and the time constant is 2.8 seconds for the model identified using the system matrices formed using order n=3.

The system matrices obtained from the system identification are summarised in Table 5.1.This allows to make confirmation that the open loop identification using the dsr method for system orders n =1, 2 and 3 show the same behaviour as the original model. There is always scope for exploring if this is suitable for higher order systems.

Table 5.1: Open loop system matrices

Open loop							
	A			**B**	**D**			**E**
n = 1	0.9579			0.0069	1			3.50E-04
n = 2	0.9663 -0.0007	-1.7385 0.7648		-0.0103 -0.0004	-0.7194	-0.6946		-9.54E-06
n = 3	0.9669 0.001 -0.0001	1.1247 0.8548 0.0069	1.5245 2.801 0.7476	-0.0119 0.0008 -0.0001	-0.5971	0.7136	-0.3665	5.59E-06

Open loop model identification holds good for representing the same information as the original models from where the input and output data are used. In other words, it can be said that the information given by the identified system can be used instead of using the original data. As told in previous sections, the model identification comprises data in a compact form and is easier for analysis.

5.3 Closed loop:

In this section, the closed loop model is going to be constructed in MATLAB Simulink. The results of the Simulink model are used to identify the system model for further analysis. The aim of this construction is to be able to control the concentration of B, B_conc, by being able to control the feed of B, B_in_c, which is the input of the model. This is also to make a comparison in the later part of the chapter between the Open loop response above and the results from Closed loop response here. The model will be made to replicate a real-world setup of a feedback type system.

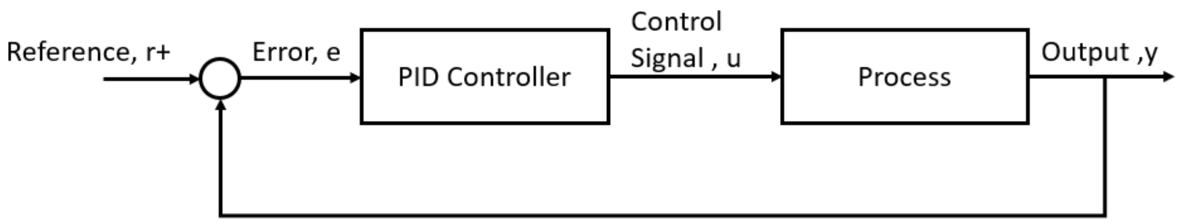


Figure 5.23: Closed loop

In this closed loop system, a PID controller is used in Figure 5.23: Closed loops. The input for the controller is the difference between the setpoint and the B_conc. The controller here is used to control the feed of reactant A_in to the reactor ultimately controlling the B_conc. In Figure 5.24, the idea for building the closed loop model for the chemical reactor is seen.

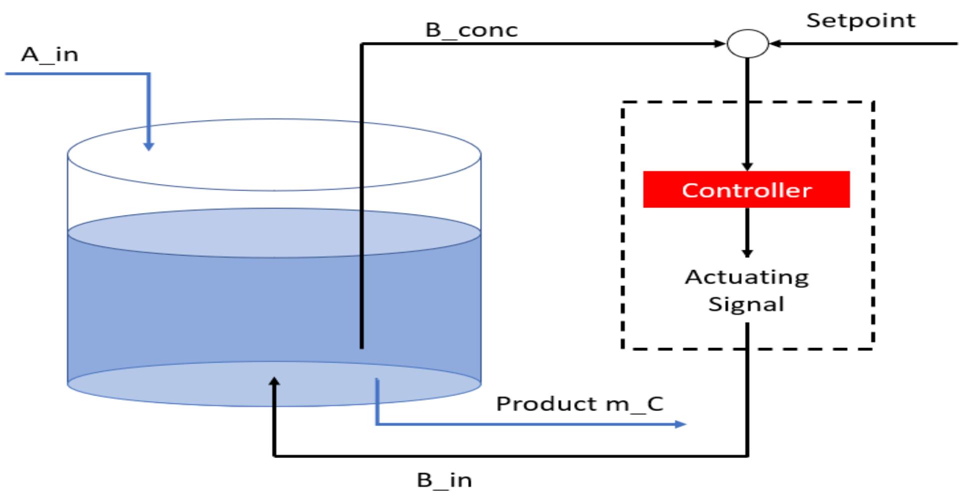


Figure 5.24: Idea of closed loop model

5.3.1.1 A MATLAB function block

Inside this function block in Figure 5.25, the main differential equations of the CSTR model are written as code, where the input and output are directly connected as feed or ports, available in Appendix A. The differential equations of the chemical reactor shown in equations 5.2, 5.3 and 5.4 are written with the inputs as A_in, B_in, m_A, m_B and m_C, and the Outputs are differentials of the masses of A, B and C with respect to time given by dmdt_A, dmdt_B and dmdt_C respectively. The same model as the Open loop is used here as well.

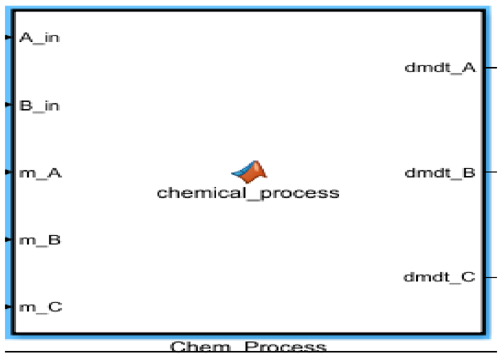


Figure 5.25: MATLAB function block

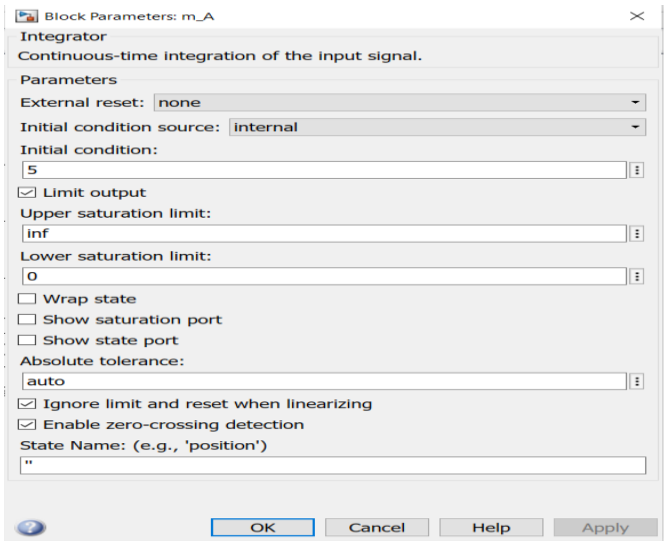
5.3.1.2 Step Change:

A step change is given to the system to observe the behaviour of the model after a disturbance is applied. The Step change is applied after 20 seconds allowing the model to reach a steady state before applying the step.

5.3.1.3 Integrator block

An integrator block to receive each of the values of m_A, m_B and m_C. This will be used as the input values for the next time step.

The initial condition of m_A = 5 Figure 5.26, and Lower saturation =0 as the value should not go below 0 in real life.



Block Parameters: m_A

Integrator

Continuous-time integration of the input signal.

Parameters

External reset: none

Initial condition source: internal

Initial condition:

5

☑ Limit output

Upper saturation limit:

inf

Lower saturation limit:

0

☐ Wrap state

☐ Show saturation port

☐ Show state port

Absolute tolerance:

auto

☑ Ignore limit and reset when linearizing

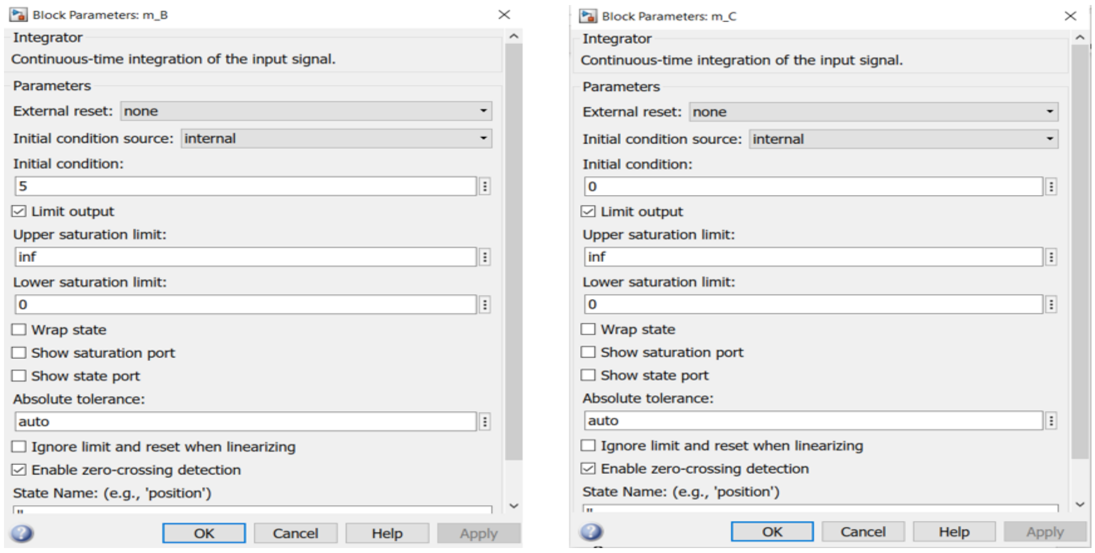
☑ Enable zero-crossing detection

State Name: (e.g., 'position')

''

Figure 5.26: Integrator block for m_A

For m_B and m_C, two more integrator blocks Figure 5.27 are used with the same initial conditions as for open loop.



Block Parameters: m_B

Integrator

Continuous-time integration of the input signal.

Parameters

External reset: none

Initial condition source: internal

Initial condition:

5

☑ Limit output

Upper saturation limit:

inf

Lower saturation limit:

0

☐ Wrap state

☐ Show saturation port

☐ Show state port

Absolute tolerance:

auto

☐ Ignore limit and reset when linearizing

☑ Enable zero-crossing detection

State Name: (e.g., 'position')

''

Block Parameters: m_C

Integrator

Continuous-time integration of the input signal.

Parameters

External reset: none

Initial condition source: internal

Initial condition:

0

☑ Limit output

Upper saturation limit:

inf

Lower saturation limit:

0

☐ Wrap state

☐ Show saturation port

☐ Show state port

Absolute tolerance:

auto

☐ Ignore limit and reset when linearizing

☑ Enable zero-crossing detection

State Name: (e.g., 'position')

''

Figure 5.27. Integrator block for m_B and m_C

5.3.1.4 Find Concentration of B

The output desired for analysis is the concentration of B, B_conc in Closed loop simulation. As mentioned earlier, in Equation 5.5, the below design is made to find the concentration of B Figure 5.28.

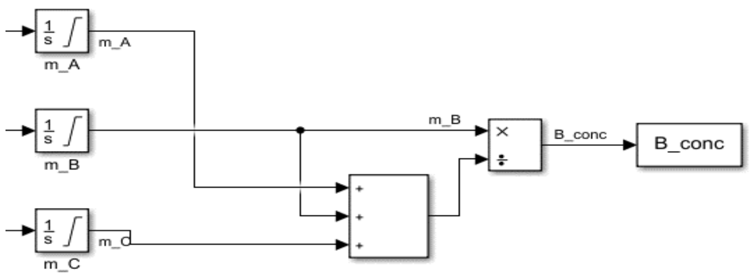
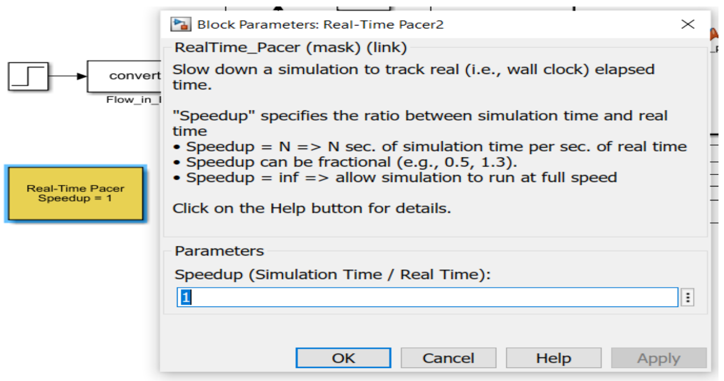


Figure 5.28: Concentration of B

The masses of A, B and C, m_A + m_B + m_B, are first added, and then the mass of B, m_B is divided by this to get the concentration of B, B_conc.

5.3.1.5 Real-Time pacer

A real time pacer[27] is added in the model to make observations as though the simulation is running in real time. The model runs in real time and the speed can be altered according to the requirement of observation.



Block Parameters: Real-Time Pacer2

RealTime_Pacer (mask) (link)

Slow down a simulation to track real (i.e., wall clock) elapsed time.

"Speedup" specifies the ratio between simulation time and real time
• Speedup = N => N sec. of simulation time per sec. of real time
• Speedup can be fractional (e.g., 0.5, 1.3).
• Speedup = inf => allow simulation to run at full speed

Click on the Help button for details.

Parameters

Speedup (Simulation Time / Real Time):

1

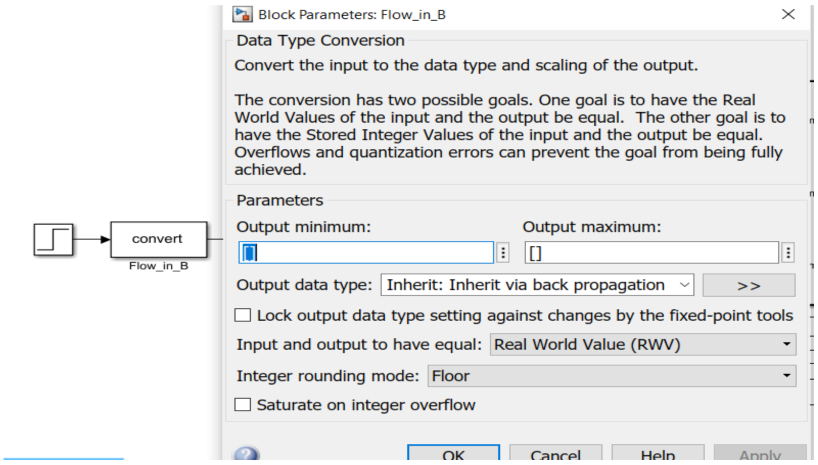
Figure 5.29: Real time pacer

5.3.1.6 Scope

The scope block allows us to observe the plotting of necessary modules.

5.3.1.7 Convert block

A convert block, Figure 5.30 is used after the step response to make sure the data type is not integer type as the values sent into the model need to be real world values (negative or decimal values).



Block Parameters: Flow_in_B

Data Type Conversion

Convert the input to the data type and scaling of the output.

The conversion has two possible goals. One goal is to have the Real World Values of the input and the output be equal. The other goal is to have the Stored Integer Values of the input and the output be equal. Overflows and quantization errors can prevent the goal from being fully achieved.

Parameters

Output minimum: []

Output maximum: []

Output data type: Inherit: Inherit via back propagation

☐ Lock output data type setting against changes by the fixed-point tools

Input and output to have equal: Real World Value (RWV)

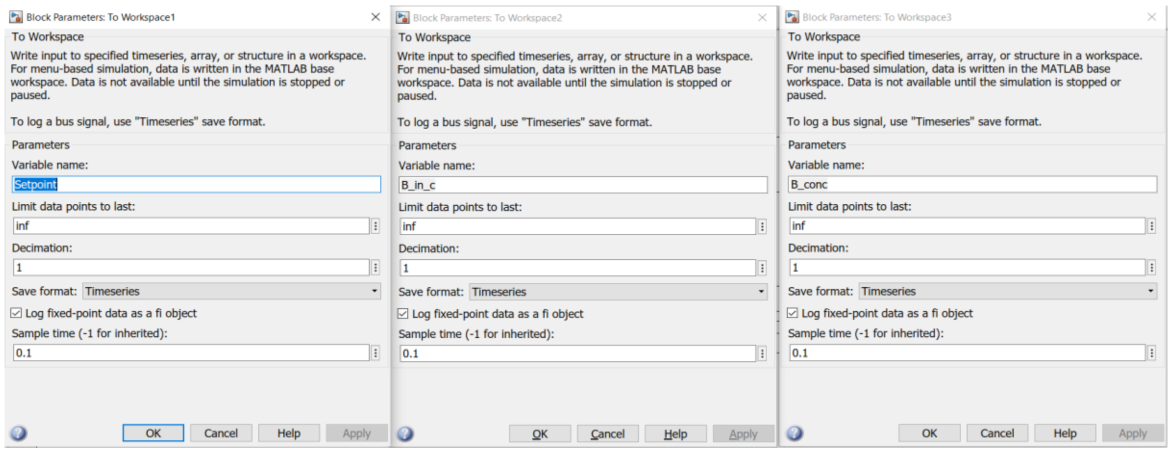
Integer rounding mode: Floor

☐ Saturate on integer overflow

Figure 5.30: Convert block

5.3.1.8 To Workspace

There are three To Workspace blocks here: Setpoint, B_in_c- the feed of B in closed loop and B_conc. The data is saved for further calculations and to use in model identification.


Block Parameters: To Workspace1

To Workspace

Write input to specified timeseries, array, or structure in a workspace. For menu-based simulation, data is written in the MATLAB base workspace. Data is not available until the simulation is stopped or paused.

To log a bus signal, use "Timeseries" save format.

Parameters

Variable name: Setpoint

Limit data points to last: inf

Decimation: 1

Save format: Timeseries

☑ Log fixed-point data as a fi object

Sample time (-1 for inherited): 0.1

Block Parameters: To Workspace2

To Workspace

Write input to specified timeseries, array, or structure in a workspace. For menu-based simulation, data is written in the MATLAB base workspace. Data is not available until the simulation is stopped or paused.

To log a bus signal, use "Timeseries" save format.

Parameters

Variable name: B_in_c

Limit data points to last: inf

Decimation: 1

Save format: Timeseries

☑ Log fixed-point data as a fi object

Sample time (-1 for inherited): 0.1

Block Parameters: To Workspace3

To Workspace

Write input to specified timeseries, array, or structure in a workspace. For menu-based simulation, data is written in the MATLAB base workspace. Data is not available until the simulation is stopped or paused.

To log a bus signal, use "Timeseries" save format.

Parameters

Variable name: B_conc

Limit data points to last: inf

Decimation: 1

Save format: Timeseries

☑ Log fixed-point data as a fi object

Sample time (-1 for inherited): 0.1

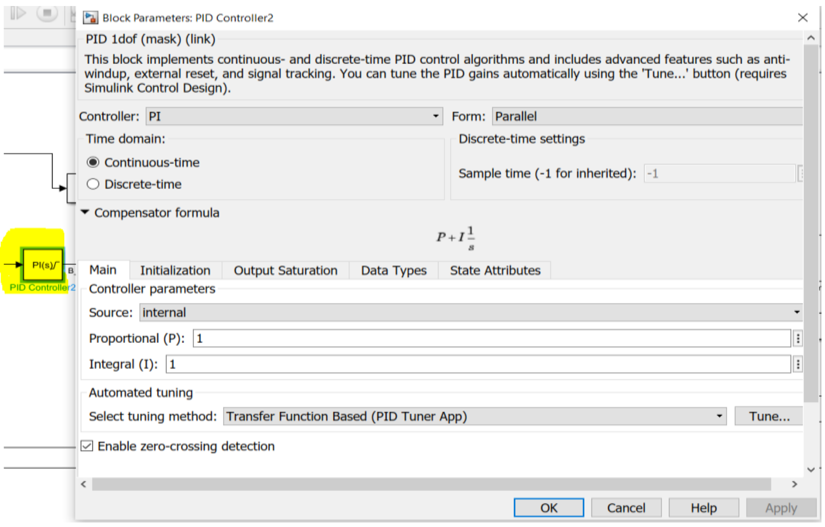
Figure 5.31: To Workspace blocks

5.3.1.9 Setpoint

A setpoint is used to use this as a reference for the controller to change the output, to ultimately make the error = 0.

5.3.1.10 PID controller

To complete a closed loop system, a PID controller block Figure 5.32, is placed, and the tuning is done in the next section. The PID controller is tuning the B_conc with the setpoint.


Block Parameters: PID Controller2

PID 1dof (mask) (link)

This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (requires Simulink Control Design).

Controller: PI | Form: Parallel

Time domain: [✓ Continuous-time] Discrete-time

Discrete-time settings: Sample time (-1 for inherited): -1

Compensator formula

$$P + I\frac{1}{s}$$

Main | Initialization | Output Saturation | Data Types | State Attributes

Controller parameters

Source: internal

Proportional (P): 1

Integral (I): 1

Automated tuning

Select tuning method: Transfer Function Based (PID Tuner App) Tune...

☑ Enable zero-crossing detection

Figure 5.32: PID configuration

5.3.2 Final Simulink Model:

Although having all the elements needed to build the Simulink model, there are various ways to analyze the data. The below image is one of the cases to be experimented with, more cases are explained further in the Chapter.

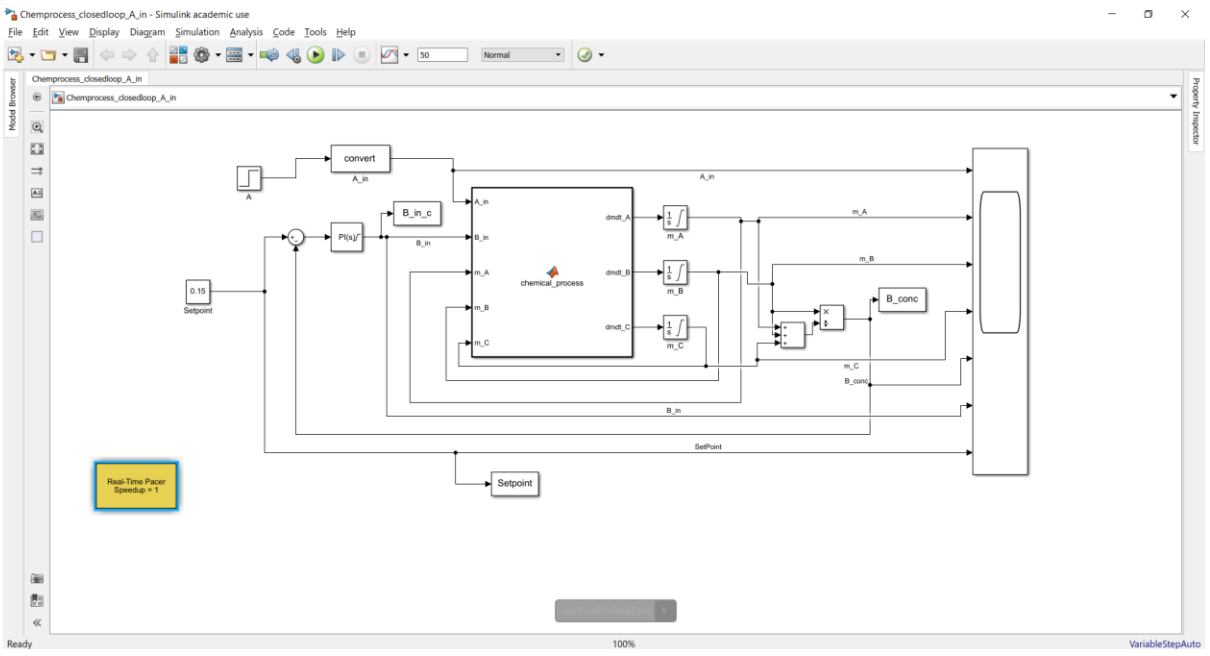

Chemprocess_closedloop_A_in - Simulink academic use
File Edit View Display Diagram Simulation Analysis Code Tools Help
Chemprocess_closedloop_A_in
convert
A_in
A
B_in_c
B_in
PI(s)
0.15
Setpoint
A_in
B_in
m_A
m_B
m_C
chemical_process
dmdt_A
dmdt_B
dmdt_C
m_A
m_B
m_C
B_conc
m_C
B_conc
SetPoint
Setpoint
Real-Time Pacer Speedup = 1
Ready
100%
VariableStepAuto

Figure 5.33: Closed loop Simulink model

This setup has a step response for the feed of A, A_in from 1 to 1.1 is shown in Figure 5.33, and the setpoint for the B_conc is kept constant at 0.15.

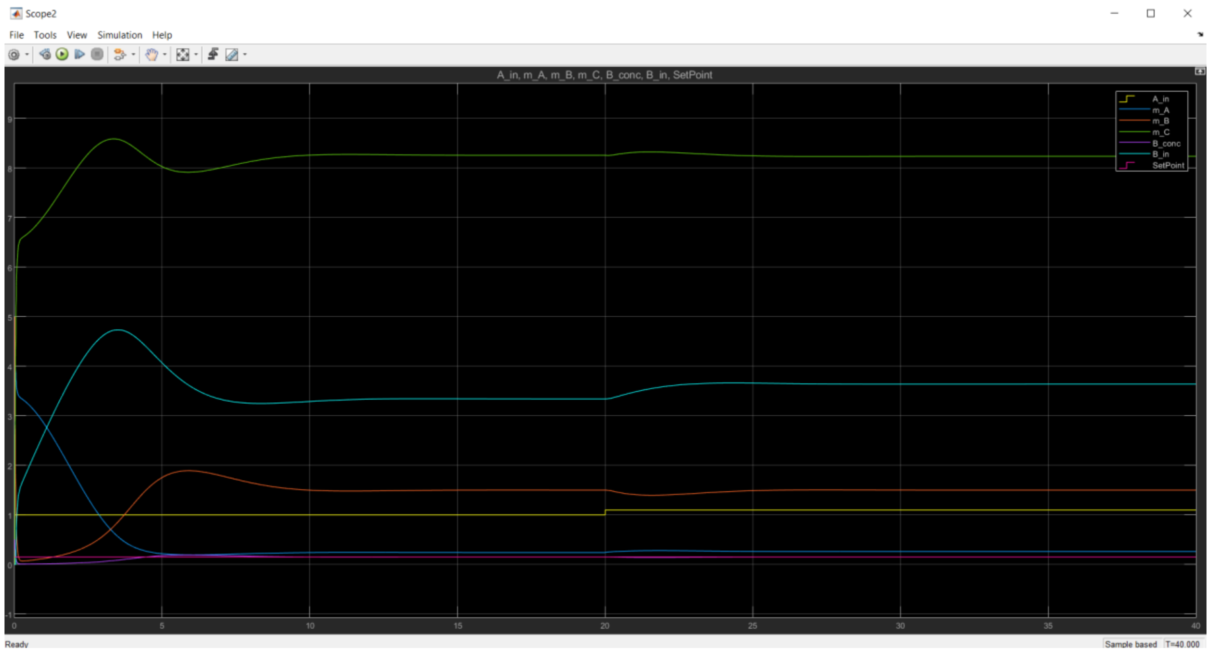


Figure 5.34: Step response for closed loop

The response in Figure 5.34 shows how the system reacts to the step response applied at time = 20 seconds.

The simulator first runs for 20 seconds and then a step response is applied to the A_in, which is the inlet feed of A from 1 to 1.1, the Yellow line. The Red line represents the mass of B, m_B showing a step response. The input used for analysis here is the feed of B, B_in, shown in the light Blue line here. The output and variable in focus is the concentration of B, B_conc in the Purple line, showing a very small step response. The next image shows a closer response of B_conc.

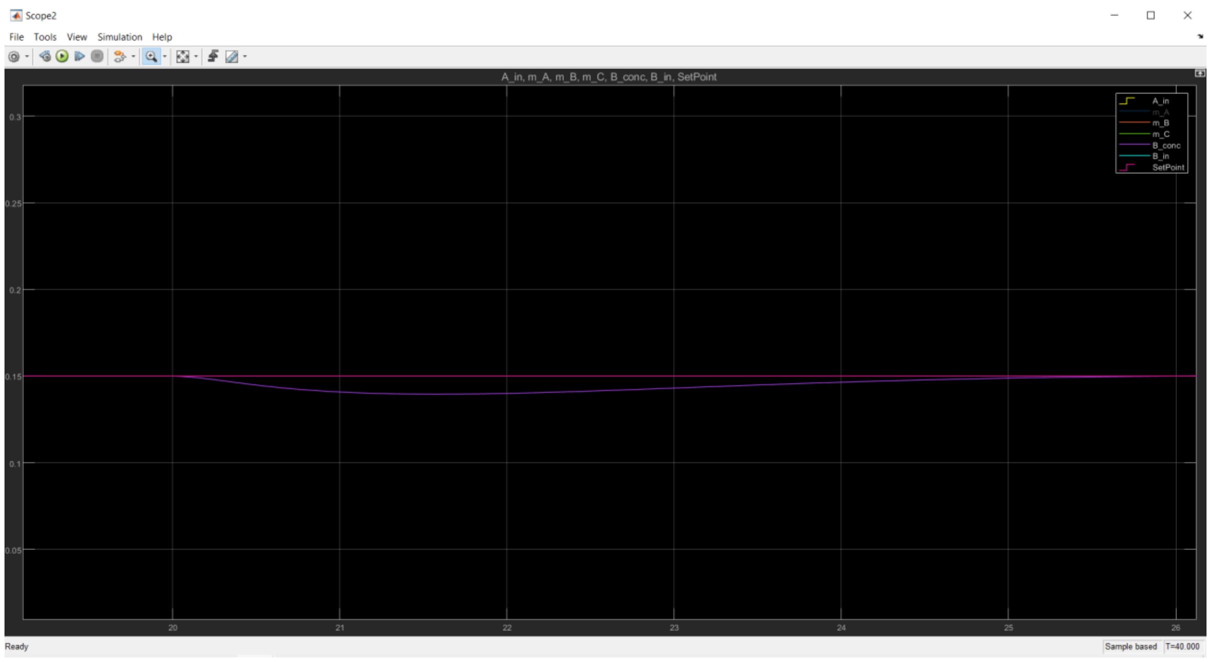


Figure 5.35. B_conc response

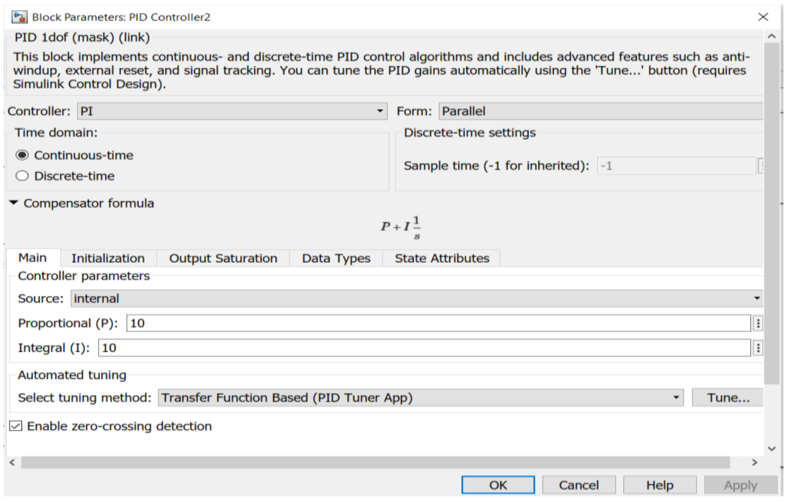
Figure 5.35shows that B_conc, which is controlled by the PID controller, is going down from 0.15 to 0.137 and reaches back to the setpoint at 0.15 after 5 seconds.

The controller tuning is shown in the next section.

5.3.3 **Controller tuning**:

A trial-and-error Step response method is chosen for the closed loop controller tuning[28].

The tuning of the controller by changing values of Proportional(P) and Integral(I) Values as shown below:


Block Parameters: PID Controller2

PID 1dof (mask) (link)

This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (requires Simulink Control Design).

Controller: PI Form: Parallel

Time domain: [✓ Continuous-time] Discrete-time

Discrete-time settings

Sample time (-1 for inherited): -1

Compensator formula

$$P+I\frac{1}{s}$$

Main | Initialization | Output Saturation | Data Types | State Attributes

Controller parameters

Source: internal

Proportional (P): 10

Integral (I): 10

Automated tuning

Select tuning method: Transfer Function Based (PID Tuner App)

[✓] Enable zero-crossing detection

Figure 5.36: Controller tuning

The formula used in the Simulink PID block is given by:

$$Compensator\ formula = P + I\ (1/s) \tag{5.8}$$

To test which values of P and I give the best results, trial and error of different values were done, and the below responses were observed:

5.3.3.1 P = 7 and I = 10

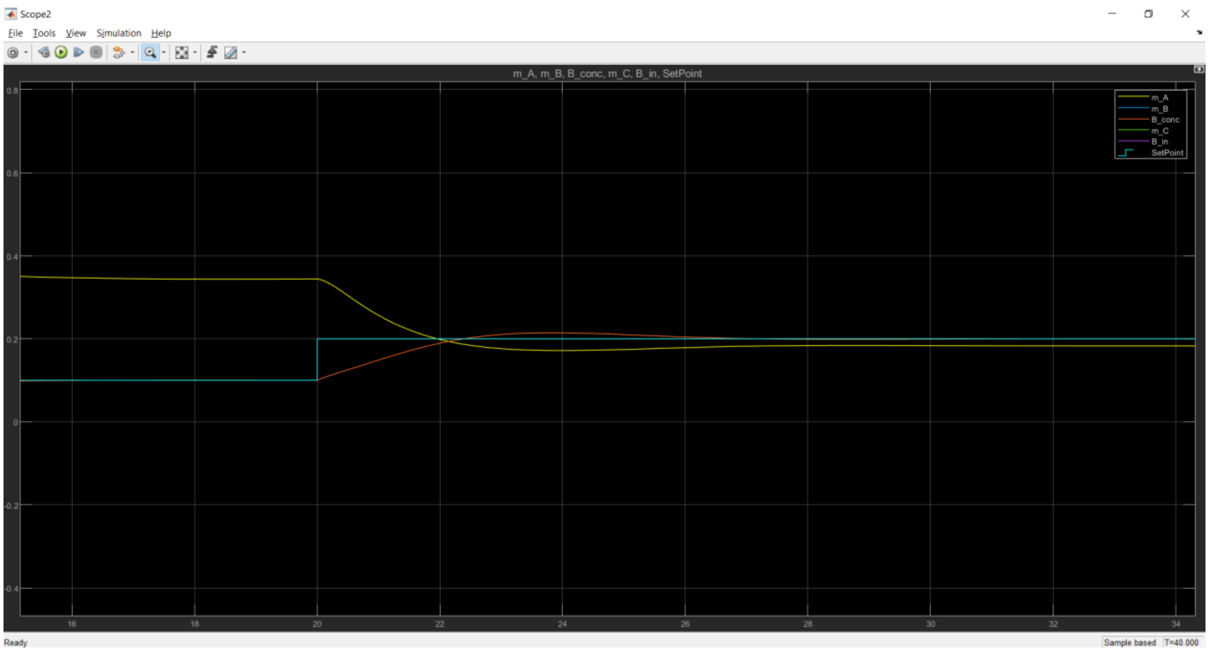


Figure 5.37: Response for P= 7 and I = 10

The Controller is showing a smooth response. It takes close to 7 seconds to reach 100% , which is the steady state, when the setpoint is changing from 0.1 to 0.2.

5.3.3.2 P = 5 and I = 10

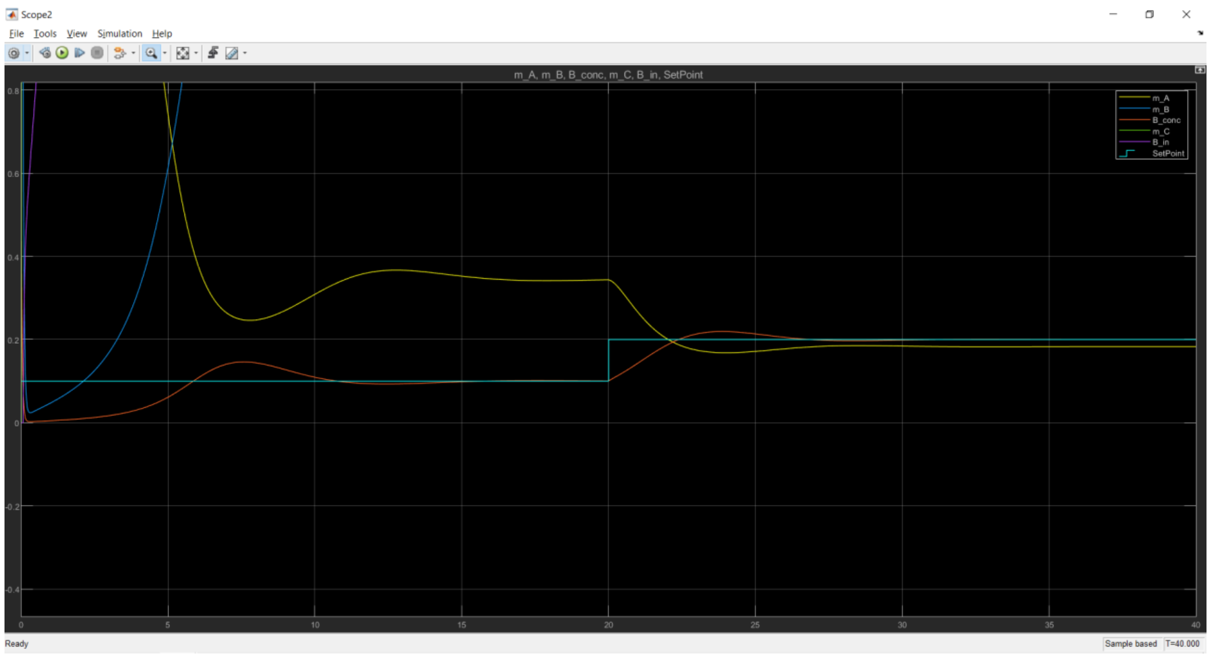


Figure 5.38: Response for P= 5 and I = 10

At P = 5 the controller is taking more time to reach the steady state, around 10 seconds.

5.3.3.3 P = 5 and I = 5

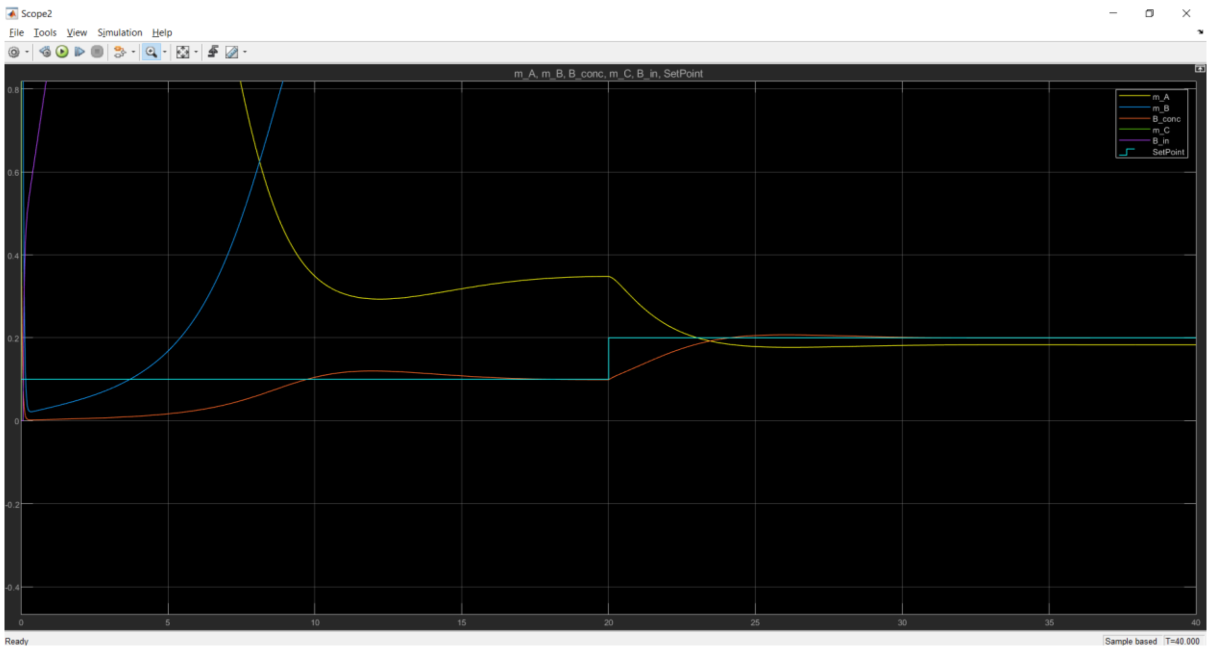


Figure 5.39: Response for P= 5 and I = 5

At I= 5, the controller is smoother with less overshoot than the previous trials, but it takes longer to reach the steady state.

5.3.3.4 P = 8 and I = 8

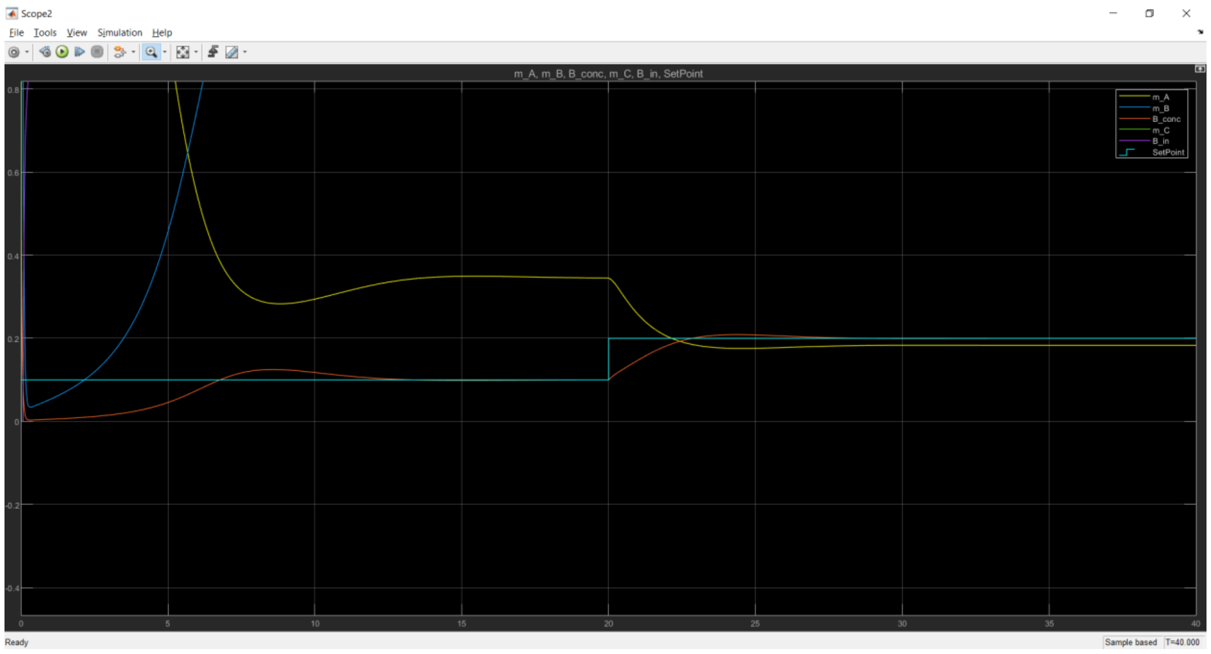


Figure 5.40: Response for P= 8 and I = 8

At P = 8 and I = 8, the response is the smoothest so far. The controller has less overshoot and reaches the steady state around 7 seconds.

5.3.3.5 P = 10 and I = 10

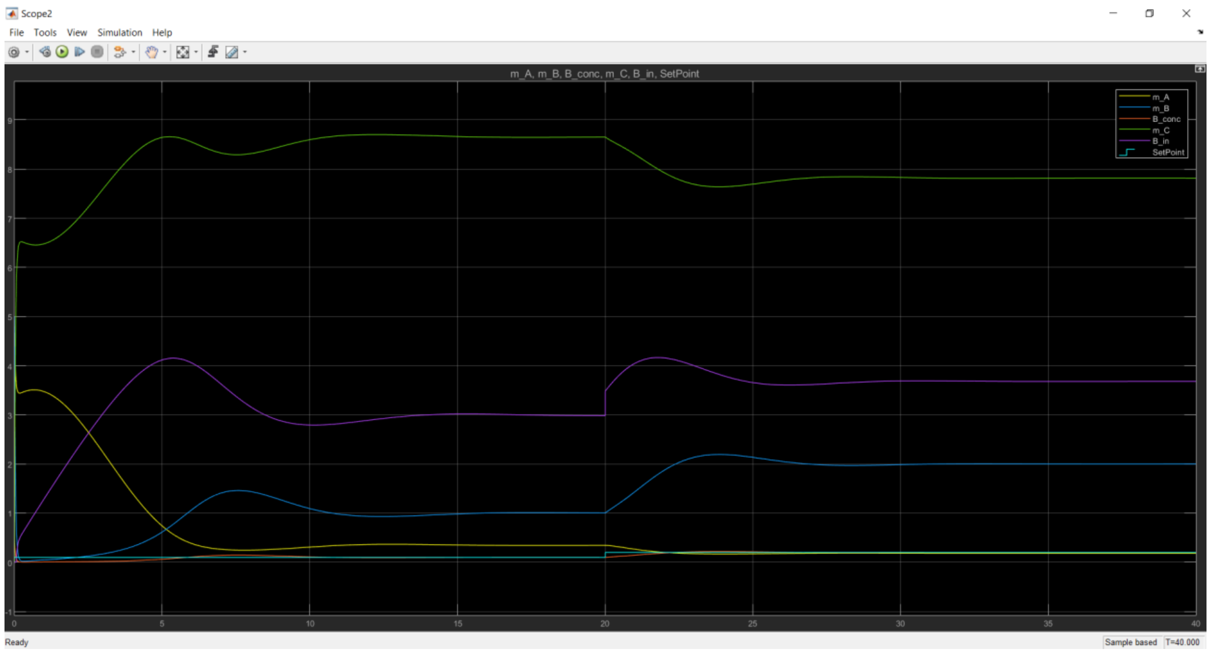


Figure 5.41: Response for P= 10 and I = 10

Figure 5.41 is made for the controller at P = 10 and I = 10.

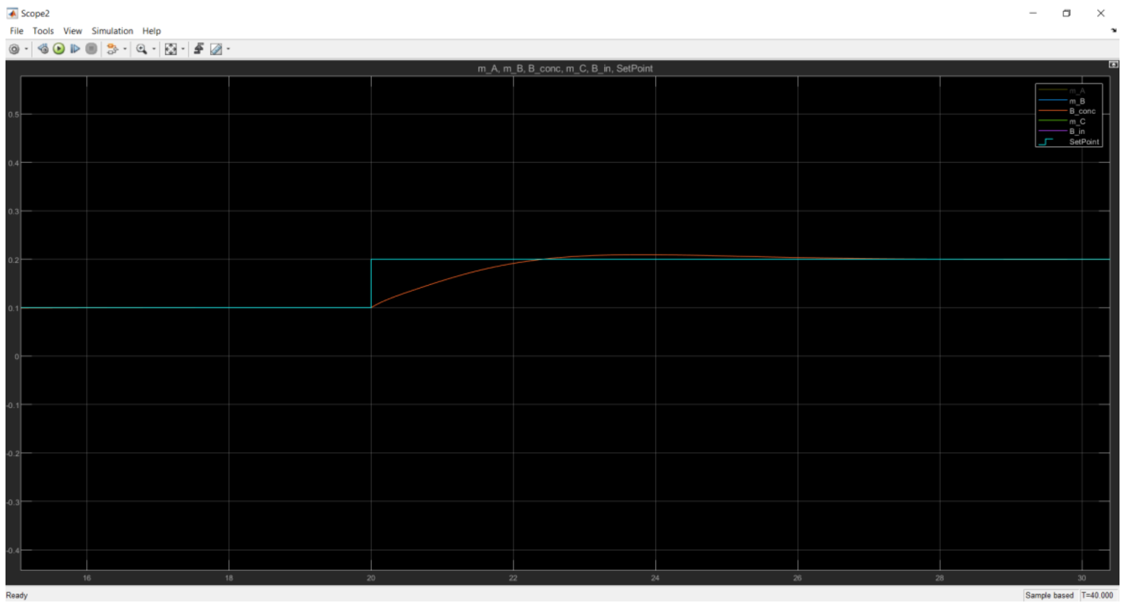


Figure 5.42: B_conc for P= 10 and I = 10

Figure 5.42 above is plotted with the Setpoint and B_conc v/s time. There is a very small overshoot observed. The final value reaches the setpoint perfectly, the steady state error is Zero. And the controller takes around 7 seconds to reach the steady state for the step change from 0.1 to 0.2. This set of values looks ideal, but more trials were made to check how the controller behaves.

5.3.3.6 P = 20 and I = 10

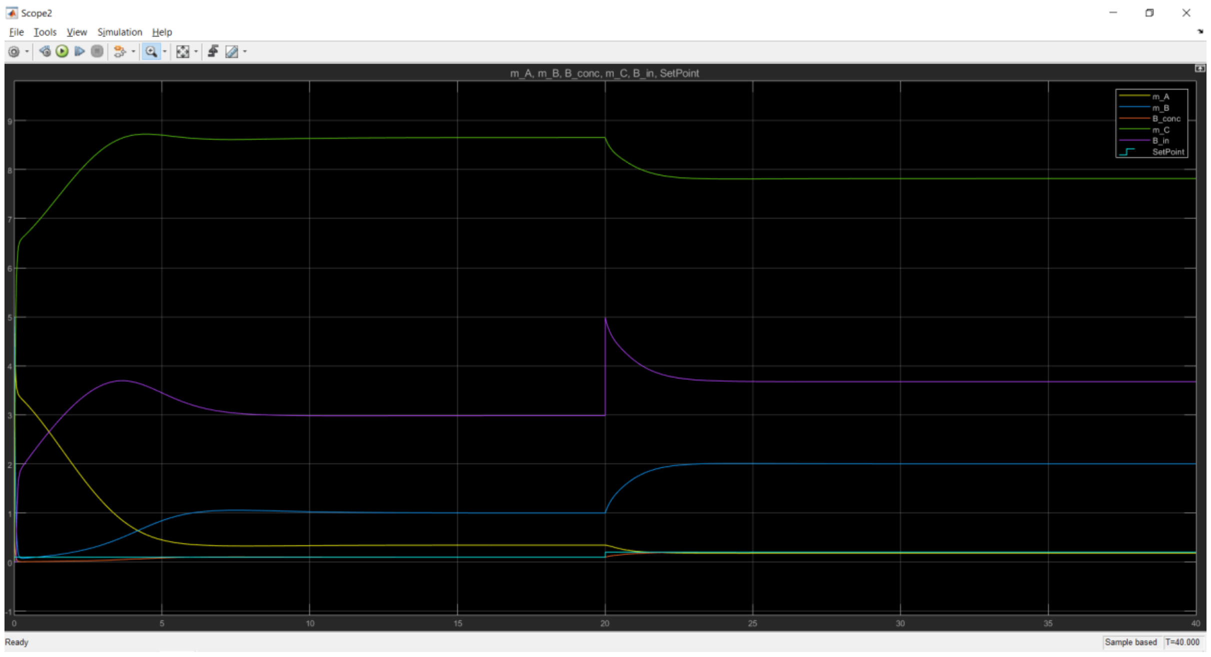


Figure 5.43: Response for P= 20 and I = 10

At P = 20, the controller is very aggressive. To have a closer look at the set point change and the B_conc;

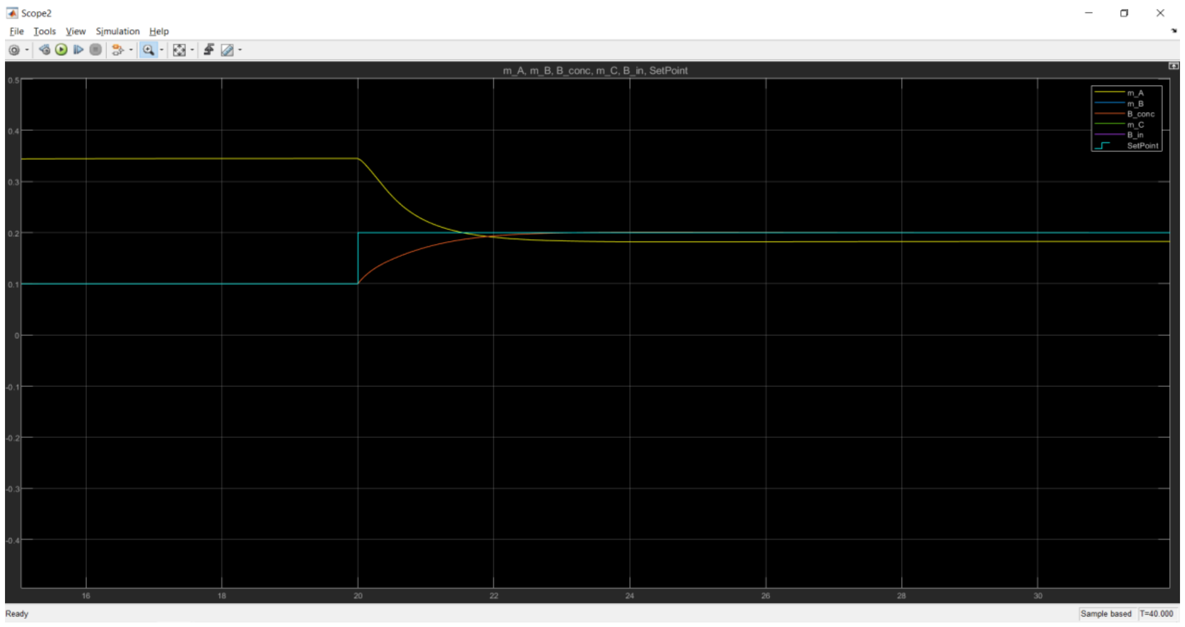


Figure 5.44: B_conc for P= 20 and I = 10

The B_conc is reaching the steady state much sooner and smoothly.

5.3.3.7 P = 7 and I = 1

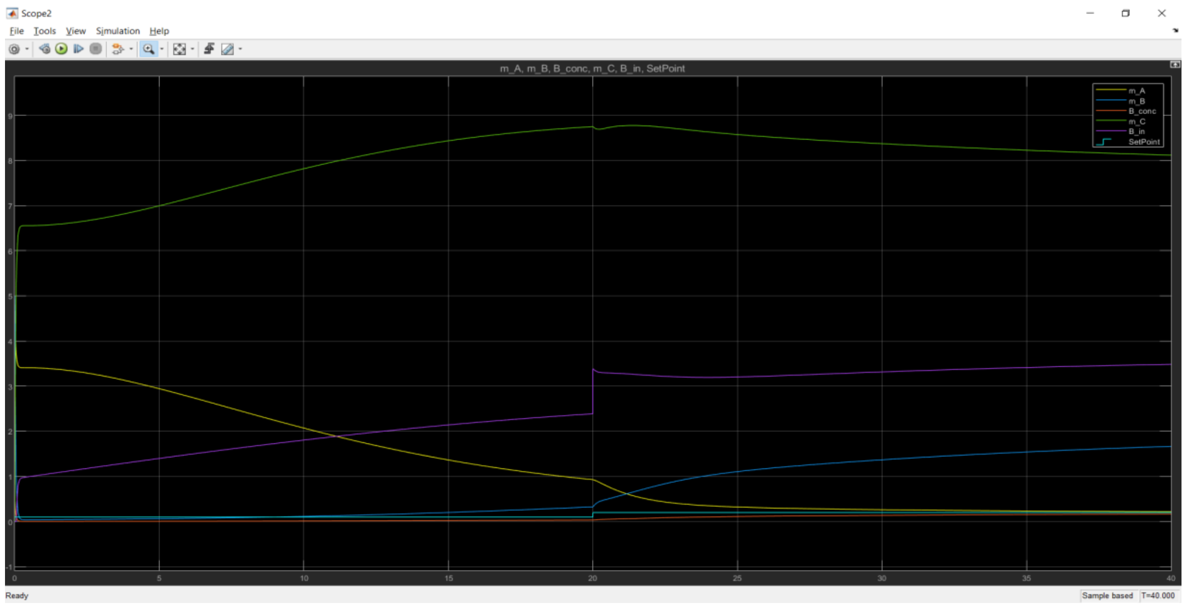


Figure 5.45: Response for P= 7 and I = 1

At P = 7 and I = 1, the controller has less Integral action and takes a lot of time to reach the steady state, maybe not suitable for this model.

5.3.3.8 P = 10 and I = 100

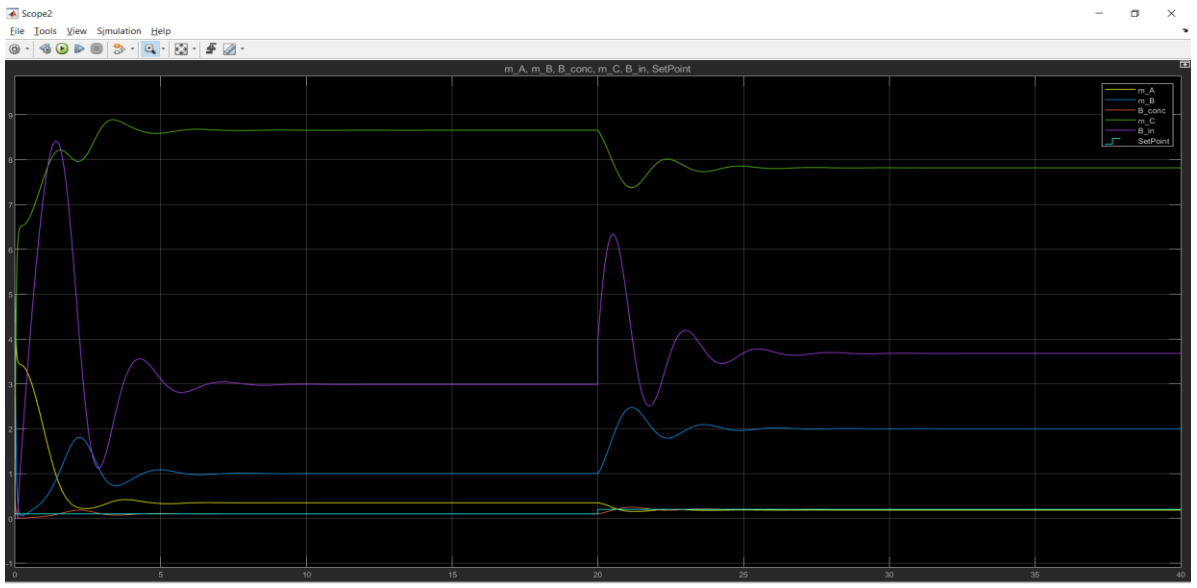


Figure 5.46: Response for P= 10 and I = 100

Super aggressive behaviour of the controller, there is a big overshoot seen in the feed of B, B_in and hence not recommended.

5.3.3.9 P = 100 and I = 10

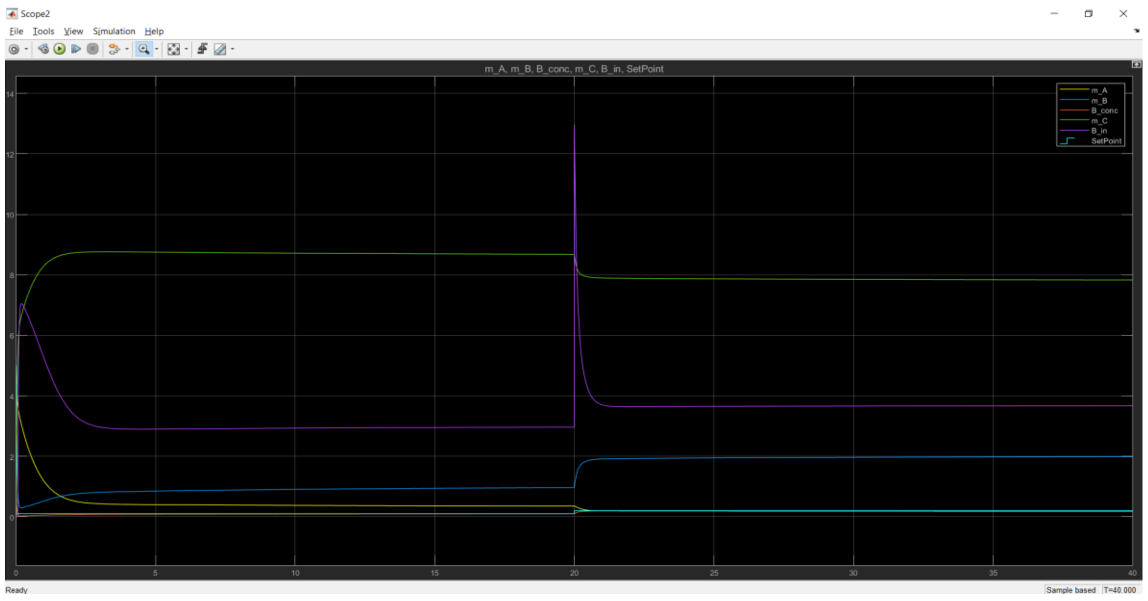


Figure 5.47: Response for P= 100 and I = 10

At P= 100 and I = 10, the controller shows extremely aggressive behaviour, it should be avoided as it will cause disasters.

From the above trial and error for a closed loop system, the final controller tuning was made and the simulation is continued for the values chosen: P = 10 and I = 10.

There are 3 cases being tested for the closed loop simulation, they are categorised by different variables used as input and output to have a confirmation of which entities of the system could be identified to best represent the Original Simulink model.

1. Case 1: Input: Setpoint and Output: B_conc
2. Case 2: Input: B_in_c and Output: B_conc
3. Case 3: Input: B_in_c and Output: B_conc by creating disturbance in A_in

5.4 Case 1: Setpoint and B_conc

Input: Setpoint

Output: B_conc

5.4.1 Feed of A, A_in

The Feed A_in is set as a constant in this model, where A_in =1.

5.4.2 Setpoint: PRBS signal

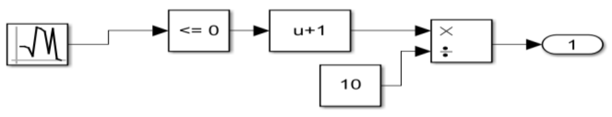


Figure 5.48: PRBS for Setpoint

Adding a PRBS signal was needed as the output is the Concentration of B, B_conc. Hence, PRBS signal from 1 to 2 is generated(u+1 is that lower limit), and then divide it by 10 to attain a signal from 0.1 to 0.2 shown in Figure 6.47.

The PID controller is used to control the feed of B, B_in as the same was done in the Open loop as well. In the first case, the setpoint is controlled, this is done because the PID controller is controlling the feed B_in, not allowing us to have control over it.

5.4.3 Simulink design

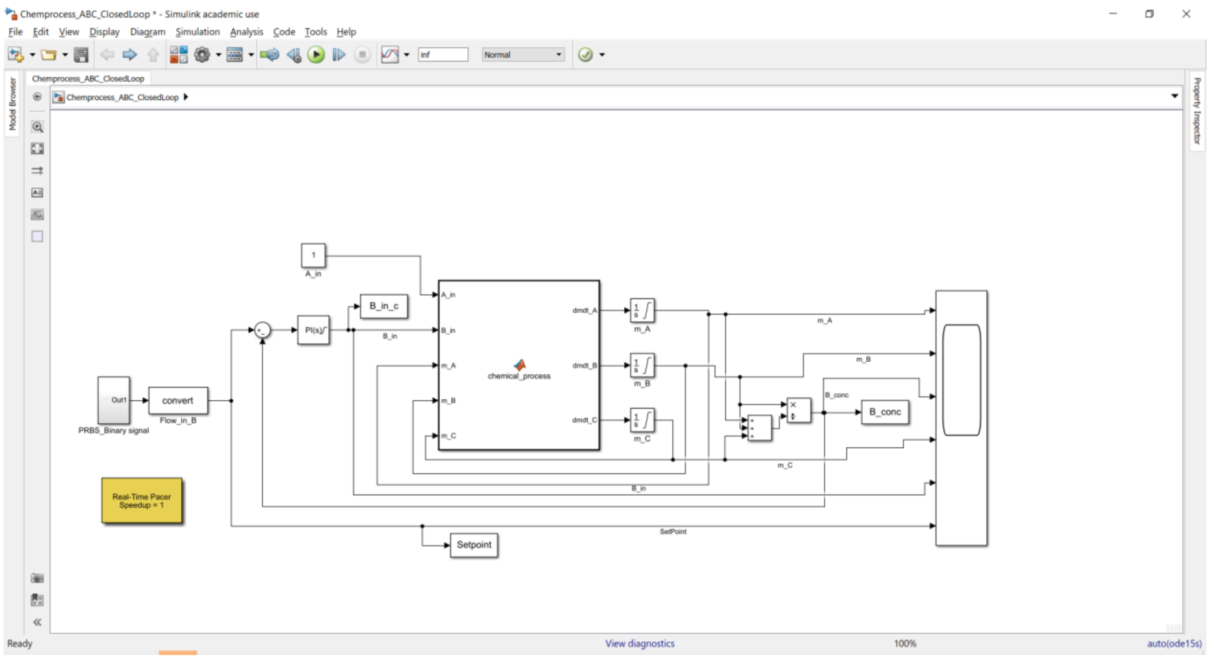


Figure 5.49: Closed loop PRBS model

Figure 5.50 shows the response of the closed loop system constructed with the new PRBS signal to the setpoint for B_conc from 0.1 to 0.2. The plot legend and the colours assigned to each variable.

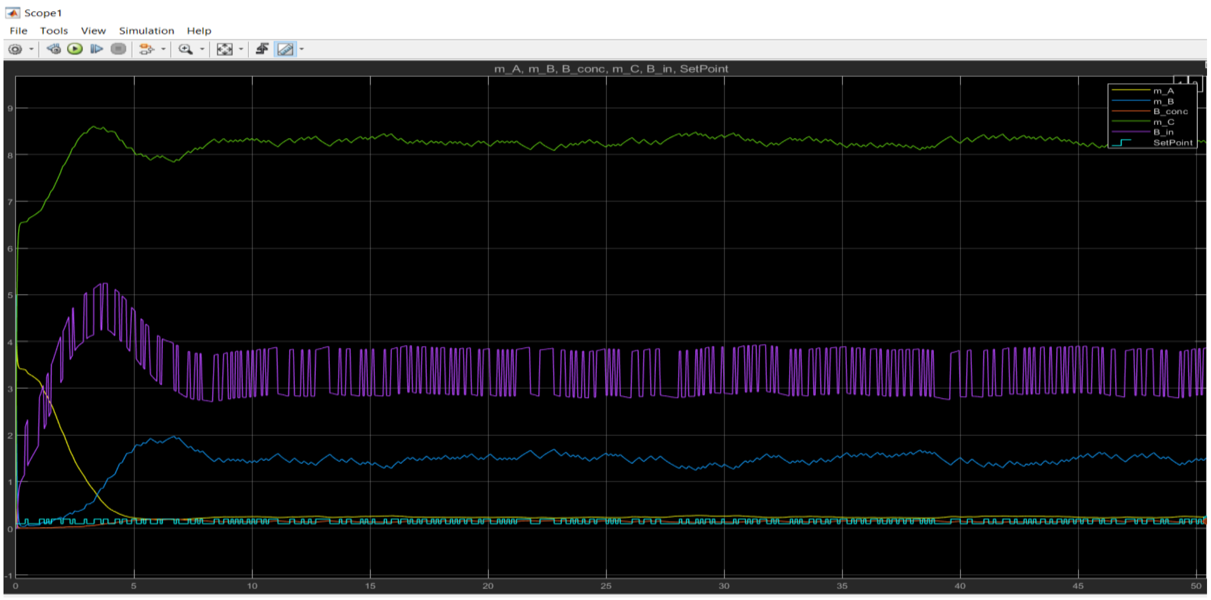


Figure 5.50: Response for Closed loop PRBS model

As the PRBS signal changes too soon, m_B does not reach the steady state before the consecutive change of signal, the model is changed to have a PRBS signal that changes less frequently (so the system almost stabilizes for a few time steps) as shown in Figure 5.51.

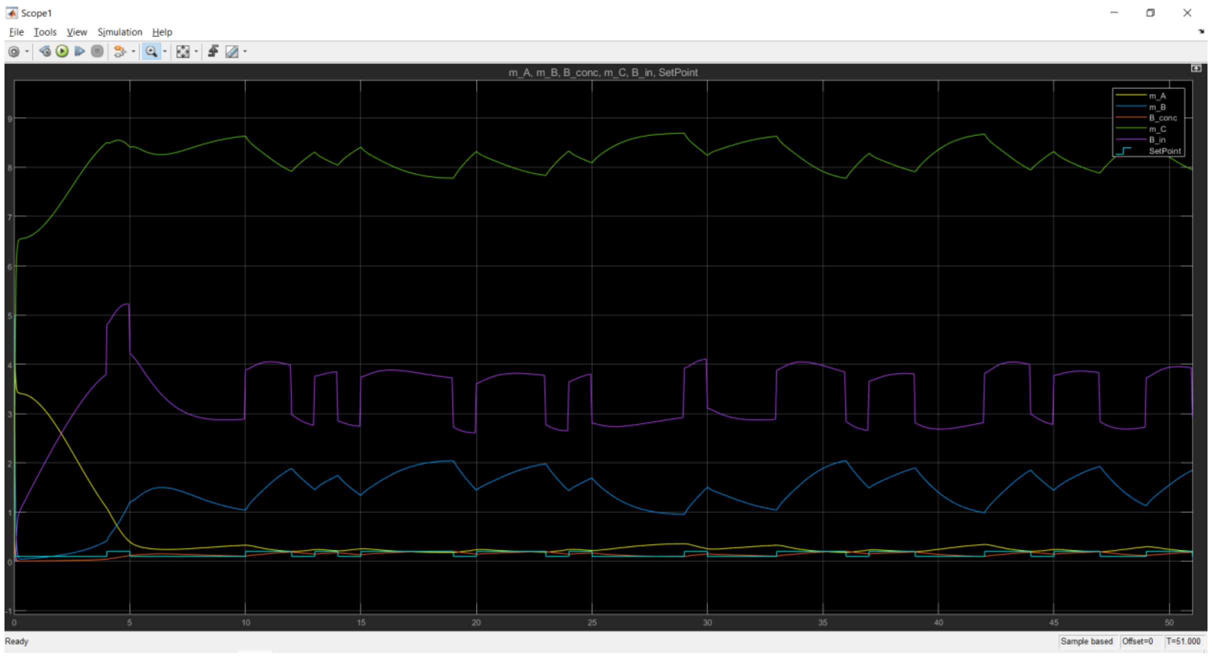


Figure 5.51: PRBS with sample time = 2 seconds

The input signal is the Setpoint here in light blue colour, showing a PRBS signal. The B_in is ranging around 2.8 to 4 when the disturbance is applied. The m_A is utilized around 0.3 represented by the Yellow line. The Blue line m_B shows the m_B in the reactor and ranges around 1 and 2, with changes in the PRBS signal and the m_C is around 8. The output of the model, concentration of B, B_conc in Red colour, is seen between 0.1 and 0.2.

5.4.3.1 Saving the Simulink results into variables

This input and output data are saved into variables u_c and y_c respectively using the below commands:

```
u_c = out.Setpoint.data;
y_c = out.B_conc.data;
```

5.4.3.2 Checking for Synchronization

The below commands are given to plot the loaded values of u_c and y_c:

```
% synchronize u and y...
>> figure(2)
```

```
clf
plot([u_c y_c])
grid on
```

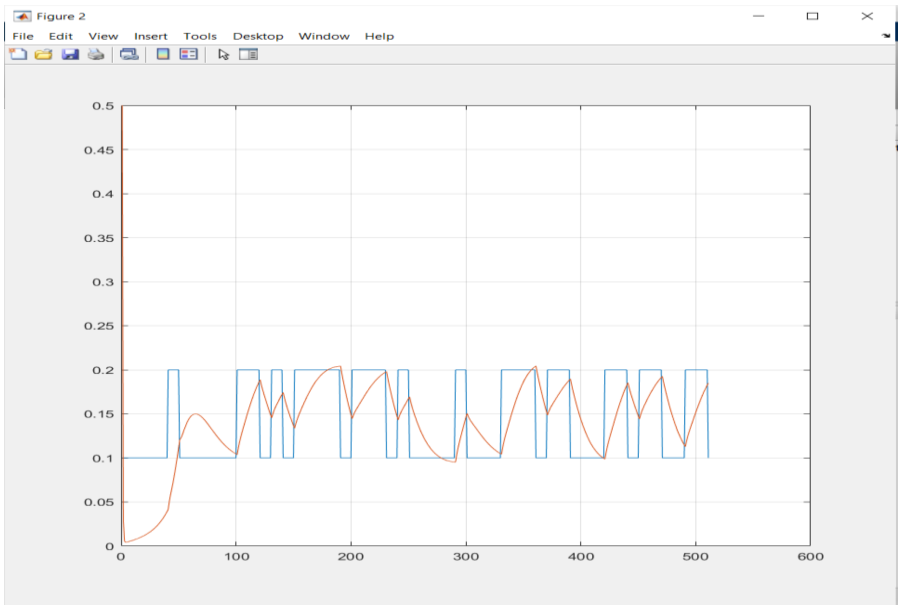


Figure 5.52: Input and output values loaded

To compare this plot with the original model, Figure 5.53 shows both plots placed next to each other and there are cursors with measurements for comparison.

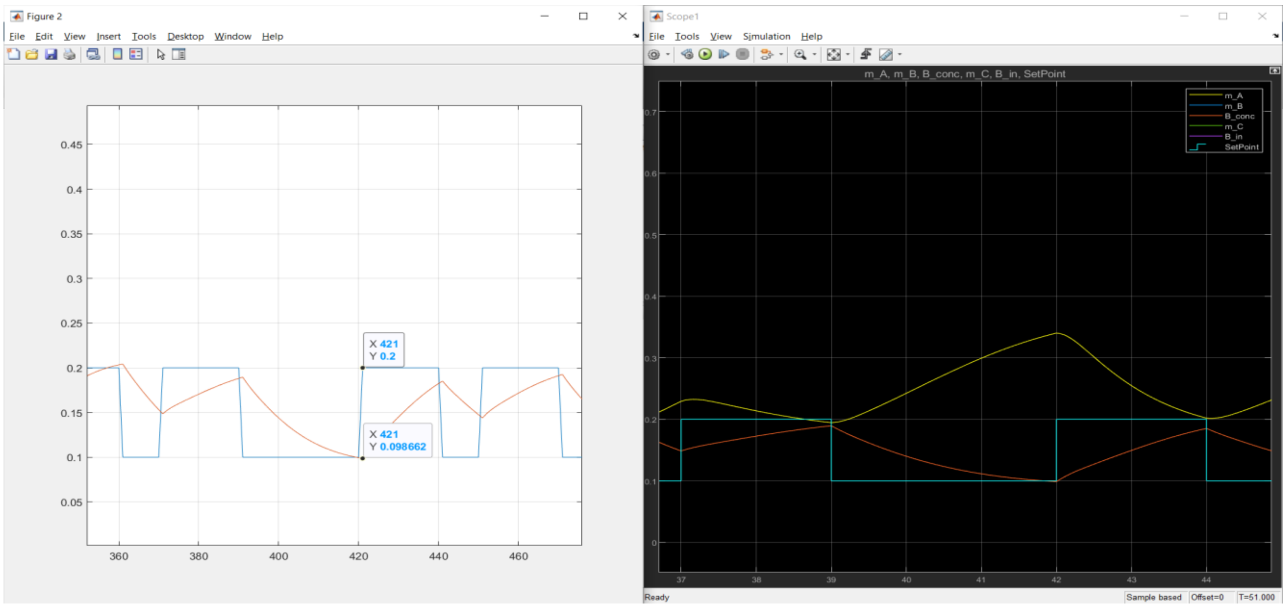


Figure 2
File Edit View Insert Tools Desktop Window Help
X 421
Y 0.2
X 421
Y 0.098662
Scope1
File Tools View Simulation Help
m_A, m_B, B_conc, m_C, B_in, SetPoint
m_A
m_B
B_conc
m_C
B_in
SetPoint
Ready
Sample based
Offset=0
T=51.000

Figure 5.53: Synchronization

At 42 seconds, y = 0.0987 in the RHS and the same measurements are seen at time = 42.1 in the LHS for u = 0.1. This is observed as the plotting in LHS has started from 1s and not 0. But this will not affect the result. Hence confirming that the output is saved correctly, and the synchronization problem is resolved.

5.4.3.3 Model Identification:

To only use the data from after the system has reached steady state, the data of the first 10 seconds is not used for system identification.

To identify the Closed loop system to find the system matrices for the input = Setpoint and output = B_conc. Using the input, output, and their biases in order to be scaled and system order n = 1, the below command is given to obtain the system matrices:

For n = 1:

```
>> u_bias = mean(u_c);
y_bias = mean(y_c);
>> [A,B,D,E,CF,F,x0] = dsr(y_c-y_bias, u_c-u_bias,1)
Ordering the given input output data
QR decomposition
A =    0.9415

B =    0.0747

D =     1

E =   -0.0022

CF =    0.0010

F =   1.5099e-04

x0 =   -0.0427
```

5.4.3.4 Making a step response

After obtaining the system matrices for the Closed loop, the step response is plotted for u= 0.1 to 0.2 at time step = 500. This is to make sure the comparison is made the same way as it was done for the original Simulink model.

```
>>dt = 0.1;
>> t = 0:dt:100;
n = length(t);
u = ones(1,n)*1;
steptime = 500;
u(steptime+1:n) = ones(n-steptime,1)*1.1;
u = u - u_bias;
```

5.4.3.5 Generating the output using the system matrices

The system matrices are now placed in a state space model(SSM) form and then a step response is plotted using the matrices A, B, D and E.

```
x = zeros(1,n);
y = zeros(1,n);
>> for i = 1:n-1
     x(:,i+1) = A*x(:,i) + B*u(:,i);
     y(:,i) = D*x(:,i) + E*u(:,i);
end
```

The biases are now added back to u and y for scaling.

```
% Add bias/scaling to get original values
>> u = u + u_bias;
y = y + y_bias;
figure(3)
clf
plot(t,[y', u'])
grid on
```

The above code can be found in Appendix C.

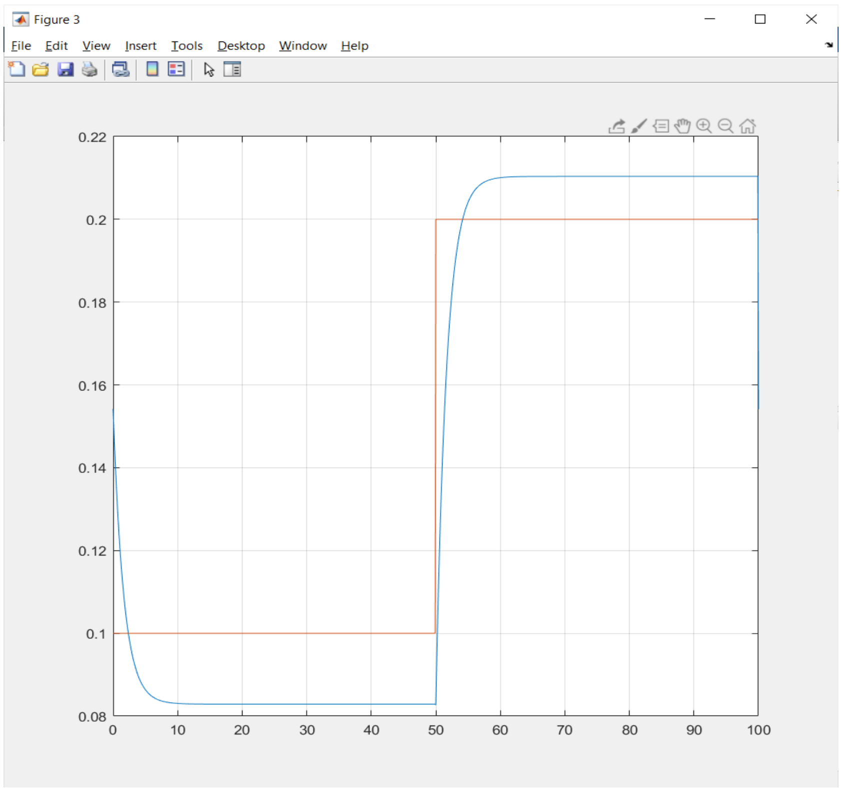


Figure 5.54: Step response using system matrices

Figure 5.54 is the step response obtained using the system matrices.

5.4.3.6 Comparing Simulink model with identification model

A step response for the setpoint from 0.1 to 0.2 is applied to the Simulink model, replacing the PRBS signal to compare the above response.

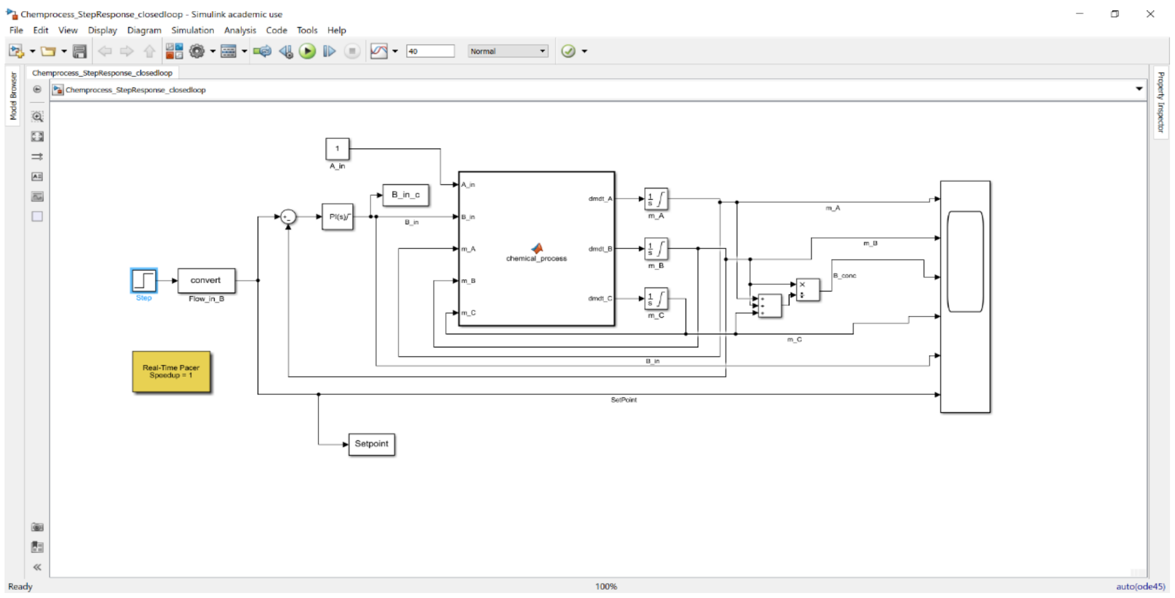

Chemprocess_StepResponse_closedloop - Simulink academic use
File Edit View Display Diagram Simulation Analysis Code Tools Help
1
A_in
B_in_c
PI(s)
B_in
A_in
B_in
m_A
m_B
m_C
chemical_process
dmdt_A
dmdt_B
dmdt_C
m_A
m_B
m_C
B_conc
Step
convert
Flow_in_B
Real-Time Pacer Speedup = 1
SetPoint
Setpoint
Ready
100%
auto(ode45)

Figure 5.55: Closed loop Simulink model

Figure 5.56 was seen for the step response. The step change was applied after 20 seconds.

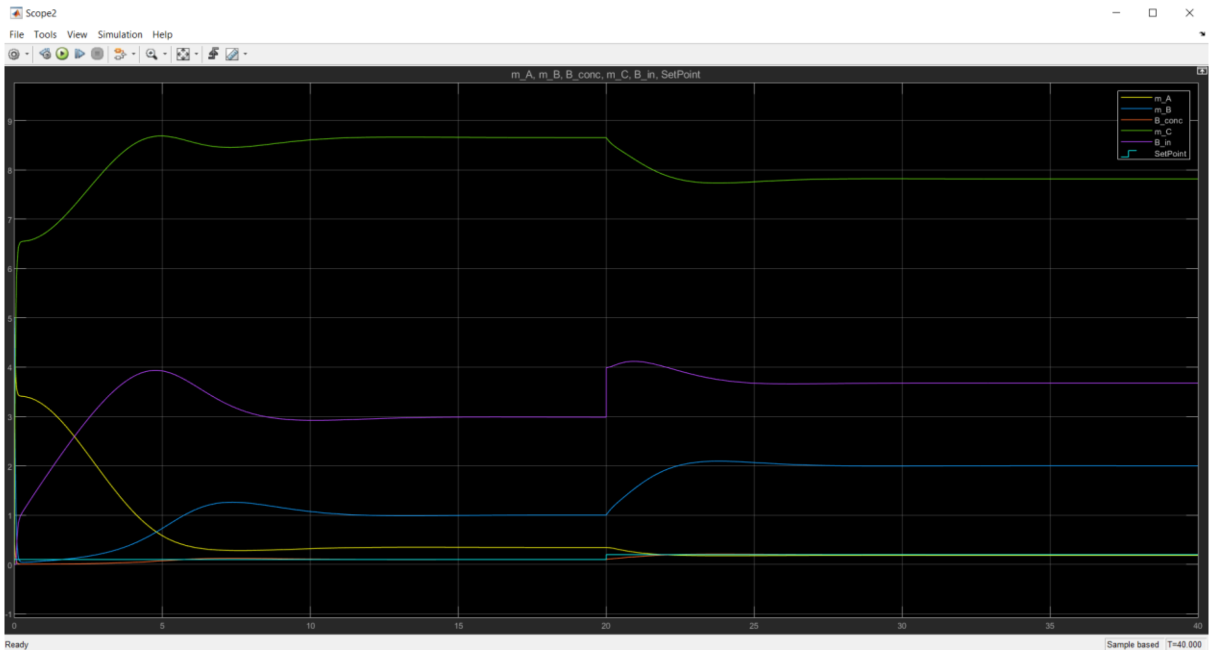


Figure 5.56: Closed loop response

To compare the response from the Simulink step response and the step response from the system identification model, the two responses are placed next to each other.

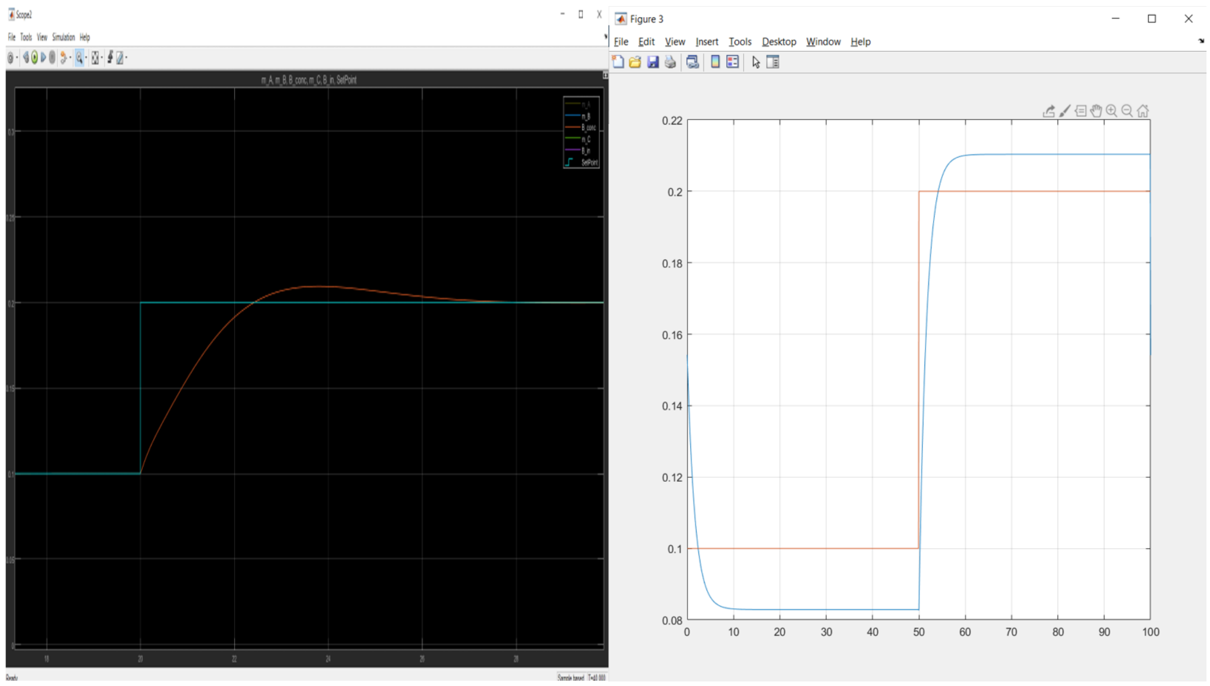

Scope2
m_A, m_B, B_conc, m_C, B_in, SetPoint
Figure 3
File Edit View Insert Tools Desktop Window Help
0.22
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0
10
20
30
40
50
60
70
80
90
100

Figure 5.57: Comparing Simulink model with identified model for n = 1

The LHS is the Simulink model, the step change in Setpoint is made from 0.1 to 0.2, for which the response of B_conc is captured. The controller takes around 7 seconds to reach steady state here. The RHS is the System identification model, where the controller also takes around 7 seconds to reach study state, but the B_conc does not reach the Setpoint here. The system identification model does not show the same behavior as the original model.

5.4.3.7 System order n = 2

Similarly, the system matrices are found for when system order n=2 and n=3 and then step responses are made for comparison Appendix C.

For system order = 2:

```
>> [A,B,D,E,CF,F,x0] = dsr(y_c-y_bias, u_c-u_bias,2)
Ordering the given input output data
QR decomposition
System order ? .................... ( 1 ) =? 2


A =   0.9578   -1.5198
     0.0001    0.5426
```

```
B =  -0.1034
     -0.0034


D =  -0.7221   -0.6917


E =  -0.0012


CF = -0.0041
      0.0009


F =    1.2431e-04


x0 = -0.1849
      -0.3269
```

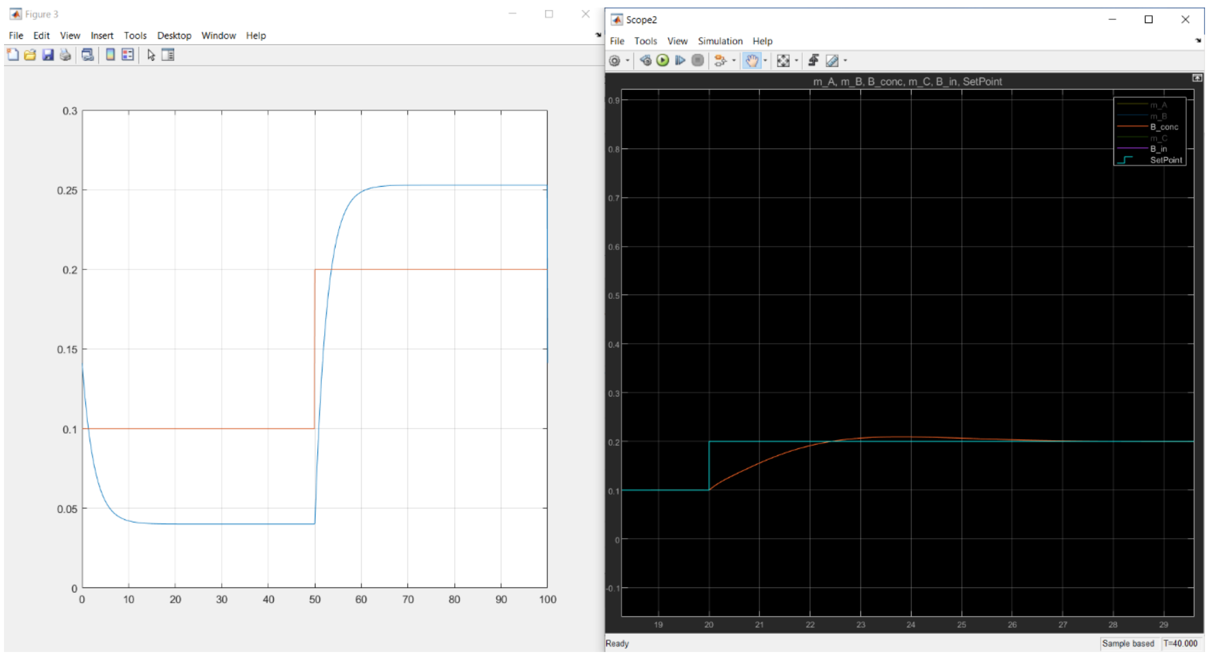


Figure 3
File Edit View Insert Tools Desktop Window Help
Scope2
File Tools View Simulation Help
m_A, m_B, B_conc, m_C, B_in, SetPoint
m_A
m_B
B_conc
m_C
B_in
SetPoint
Ready
Sample based
T=40.000

Figure 5.58: comparison for n=2

The step response curve reaches 100% after 7 seconds in the RHS, the simulation model response and 14 seconds in the RHS model, which is the step response applied to the system identification model. The figure in the LHS is the step response using the system matrices after the system identification step, whereas the plot on the RHS is the step response used on the Simulink model.

The step response shown in these two plots are clearly different and thus it can be told that the model identified after a system identification will not show an accurate behaviour of the original model. The gain in LHS is 0.04-0.25= 0.21, 63% of 0.21 is 0.132. 0.04+0.132=0.172, which is found at 52.5s and the time constant is 2.5 second. And in the RHS, the gain is 0.1-0.2 = 0.1, 63% of 0.1 is 0.063. 0.1+0.063=0.163, which is found at 21.17 and the time constant is 1.17 second. This clearly indicates that the step response made for the system is showing bad and thus meaning that model identification for a closed loop system is giving bad results.

The output for the system order n = 2 is showing the same behaviour as it was for n = 1. Meaning, the identified model does not have the same information as the original model.

5.4.3.8 System order n = 3

For the system order n= 3, the same change applies to Appendix C- using the dsr command and defining x to obtain the system matrices.

```
>>[A,B,D,E,CF,F,x0] = dsr(y_c-y_bias, u_c-u_bias,3)
Ordering the given input output data
QR decomposition
System order ? .................... ( 1 ) =? 3

A = 0.9588    1.1786    1.3727
     -0.0003    0.9622    2.6815
     -0.0002    0.0006    0.5770


B = -0.1227
     0.0068
     -0.0013


D = -0.6013    0.6964   -0.3917


E = 1.2740e-04


CF = -0.0012
     -0.0005
     -0.0000
```

```
F =    1.2297e-04


x0 = -0.0833
      0.3463
     -0.1748
```

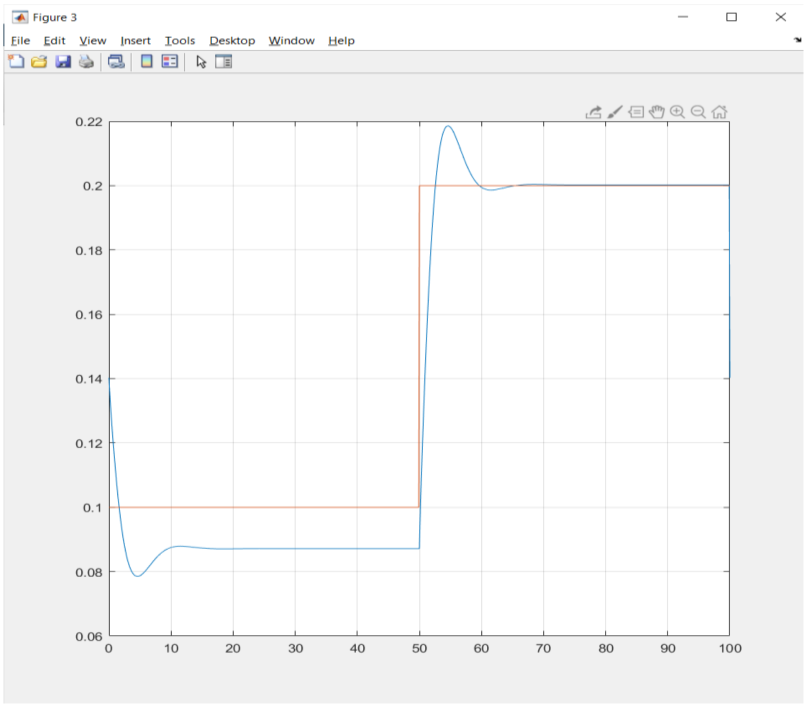


Figure 5.59: Step response for n=3

The output curve reaches 100% after 10 time steps. The gain is 0.11 with a little more overshoot, and the time constant is around 2 seconds for the model identified using the system matrices formed using order n=3.

Case 1: Input: Setpoint and Output: B_conc								
	A			B	D			E
n = 1	0.9415			0.0747	1			-2.20E-03
n = 2	0.9578	-1.5198		-0.1034	-0.7221	-0.6917		-1.20E-03
	0.0001	0.5426		-0.0034				
n = 3	0.9588	1.1786	1.3727	-0.1227	-0.613	0.6964	-0.3917	1.27E-04
	-0.003	0.9622	2.6815	0.0068				
	-0.0002	0.0006	0.577	-0.0013				

The above table for the Closed loop identification is made using the dsr method for system orders n =1, 2 and 3, with input as the setpoint and the concentration of B, B_conc as the outpoint. Although models made with n = 1 and n = 2 show the similar behaviour as and the model with n = 3 shows different, but better response, they vary from the original model built in Simulink. Higher system orders can be checked with the same models.

In other words, it can be said that the information given out after the system identification step does not give a good detail about the model used originally.

5.5 Case 2: B_in_c and B_conc

Input: B_in_c

Output: B_conc

Using the same model as used in Case 1, but using the variables B_in_c as the input for this model instead.

5.5.1 Feed of A, A_in

The feed of A is set to a constant value, A_in = 1, same as in 6.4.1

5.5.2 Setpoint: PRBS signal

Same as in 6.4.2, setpoint has the PRBS signal, and Setpoint = 0.1 to 0.2 with a sample time of 2 seconds.

5.5.3 Simulink design

Same design as shown in 6.4.3 and 6.4.4

5.5.3.1 Saving the Simulink results into variables

This input and output data are saved into variables u_c and y_c respectively using the below commands:

```
u_c = out.B_in_c.data;

y_c = out.B_conc.data;
```

To only use the data from after the system has reached steady state, the data from the first 10 seconds is not used for system identification.

5.5.3.2 Checking for Synchronization

The below commands are given to plot the loaded values of u_c and y_c:

```
% synchronize u and y...
```

```
>> figure(2)
clf
plot([u_c y_c])
grid on
```

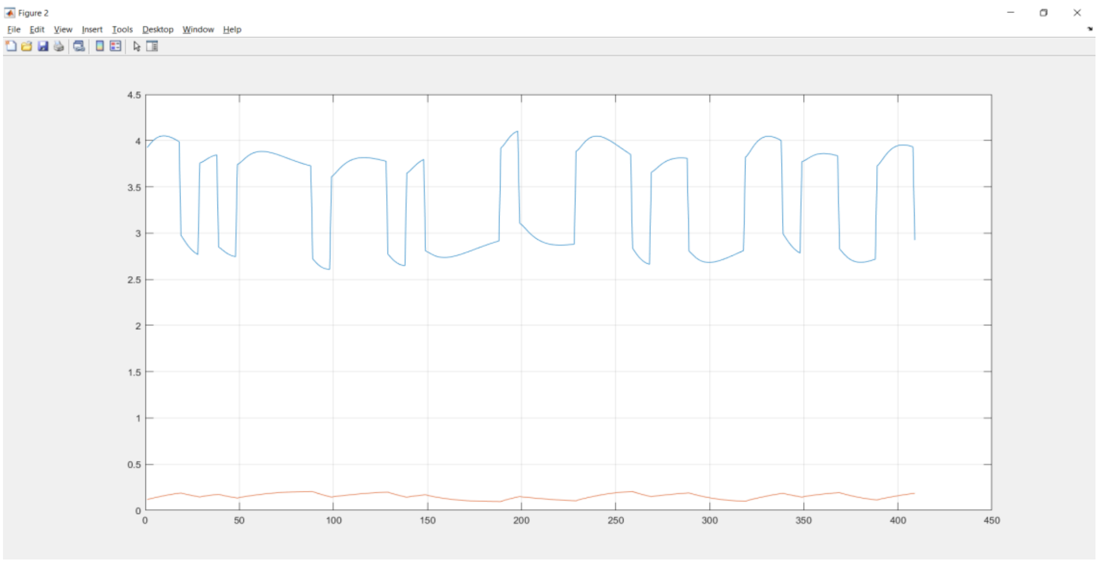


Figure 5.60: Input and Output

The model is compared with the Simulink model to compare for synchronization shown in Figure 5.61.

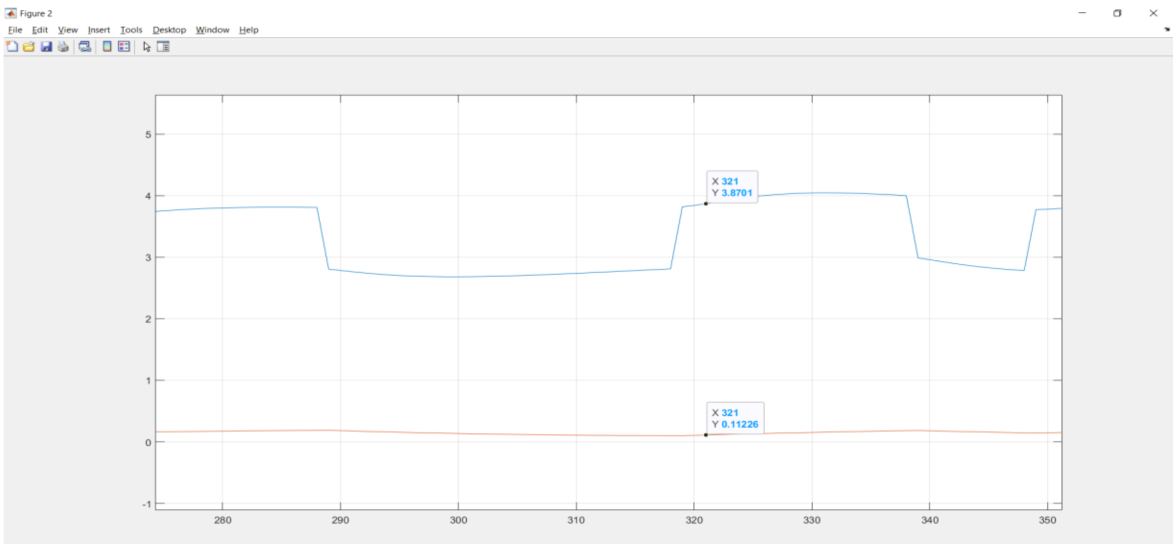


Figure 5.61: Synchronization checking

At 321st time step, the B_conc is at 0.112 and B_in_c, u is at 3.87, which is the same as the values in the original Simulink model at time =42 seconds, when the setpoint is at 0.2. The change in time as a whole will not affect the final result, as both u and y have uniformly moved.

5.5.3.3 Model Identification:

To identify the Closed loop system, the system matrices for the input = B_in_c and output = B_conc. Using the input, output, and their biases in order to be scaled and system order n = 1, the below command is given to obtain the system matrices:

For n = 1:

```
>> u_bias = mean(u_c);
y_bias = mean(y_c);
>> [A,B,D,E,CF,F,x0] = dsr(y_c-y_bias, u_c-u_bias,1)
Ordering the given input output data
QR decomposition
A =    0.9472

B =    0.0067

D =     1

E =  -3.7417e-06

CF =   6.9694e-04

F =   1.0416e-04

x0 =   -0.0366
```

5.5.3.4 Making a step response

As the input here is B_in_c and generating a step response for this is not the same, hence a step change is given as the step change for Setpoint for having the same behavior for B_in_c and B_conc. Same as in Case 1. The step response is plotted for Setpoint = 0.1 to 0.2 at time step = 500. The comparison will be made for the output B_conc = y.

```
>>dt = 0.1;
```

```
>> t = 0:dt:100;
n = length(t);
u = ones(1,n)*0.1;
steptime = 500;
u(steptime+1:n) = ones(n-steptime,1)*0.2;
u = u - u_bias;
```

5.5.3.5 Generating the output using the system matrices

The system matrices are now placed in a state space model(SSM) form and then a step response is plotted using the matrices.

```
x = zeros(1,n);
y = zeros(1,n);
>> for i = 1:n-1
     x(:,i+1) = A*x(:,i) + B*u(:,i);
     y(:,i) = D*x(:,i) + E*u(:,i);
end
```

The biases are now added back to u and y for scaling.

```
% Add bias/scaling to get original values
>> u = u + u_bias;
y = y + y_bias;
figure(3)
clf
plot(t,[y', u'])
grid on
```

The above code can be found in Appendix C.

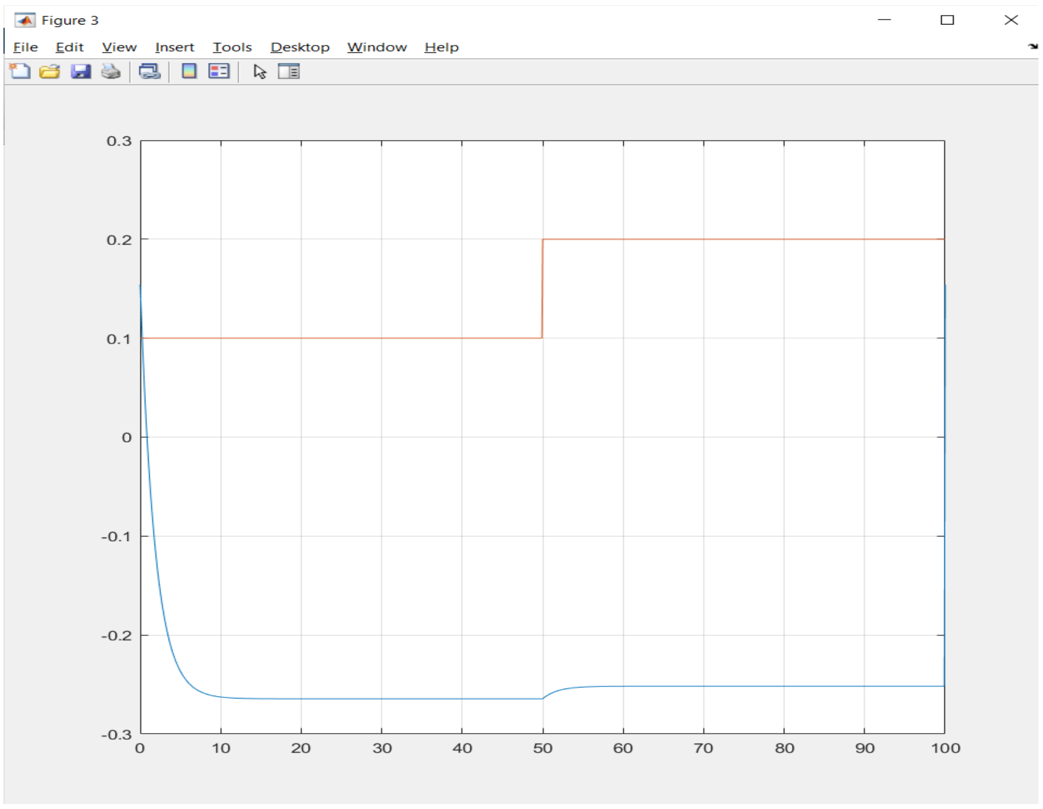


Figure 5.62: Step response for n=1

5.5.3.6 Comparing Simulink model with identification model

A step response for the setpoint from 0.1 to 0.2 is applied after 20 seconds to the Simulink model, replacing the PRBS signal to compare the above response. Same as in Case 1, the step response is plotted for Setpoint = 0.1 to 0.2 at time step = 500. The comparison will be made for the output B_conc = y.

To compare the response from the Simulink step response and the step response from the system identification model, the two responses are placed next to each other Figure 5.63.

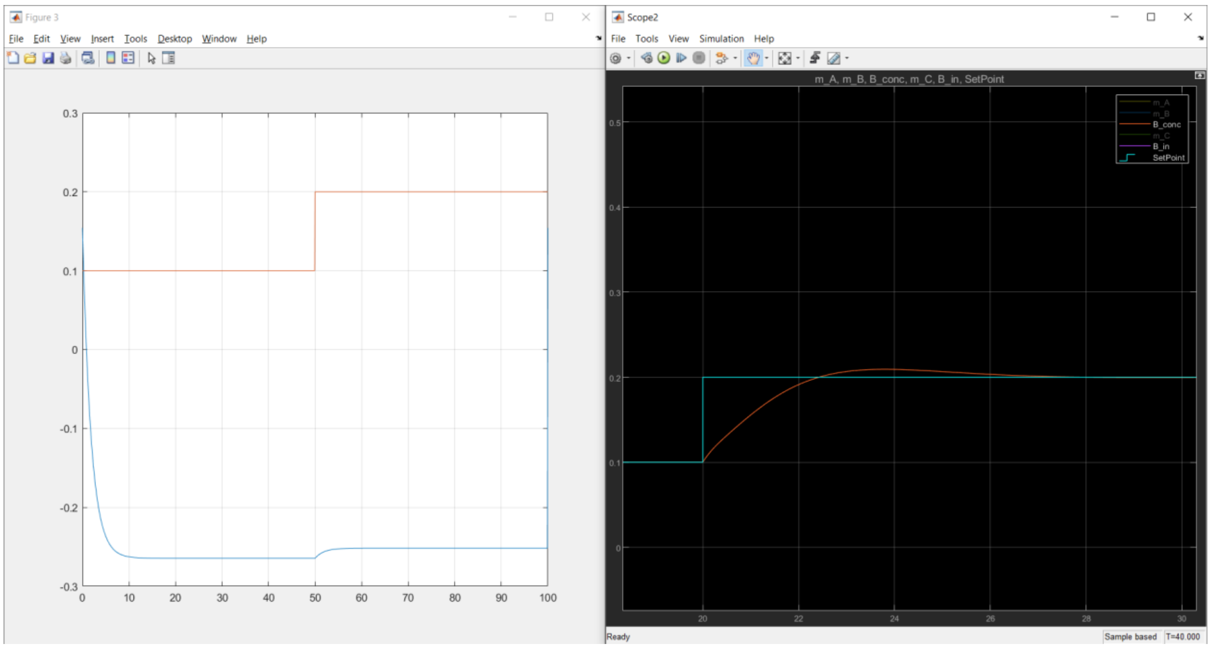


Figure 5.63: Step response comparison for identified model and Simulink model

The RHS is the Simulink model, the step change in Setpoint is made from 0.1 to 0.2, for which the response of B_conc is captured. The controller takes around 7 seconds to reach a steady state here. The LHS is the System identification model, where the controller does not reach steady state; the B_conc does not reach the Setpoint here. The system identification model does not show the same behavior as the original model.

Similarly, the system matrices are found for when system order n=2 and n=3 and then step responses are made for comparison shown in Appendix C.

5.5.3.7 System order n = 2

For system order = 2:

```
>> [A,B,D,E,CF,F,x0] = dsr(y_c-y_bias, u_c-u_bias,2)
Ordering the given input output data
QR decomposition
System order ? ..................... ( 1 ) =? 2
A =    0.9491   -1.6104
      -0.0018    0.6443


B =   -0.0105
      -0.0004
```

```
D =   -0.7259   -0.6878


E =   -2.2031e-05


CF =   1.0e-03 *
      -0.4187
       0.1027


F =    7.8280e-05


x0 =   0.0509
      -0.0005
```

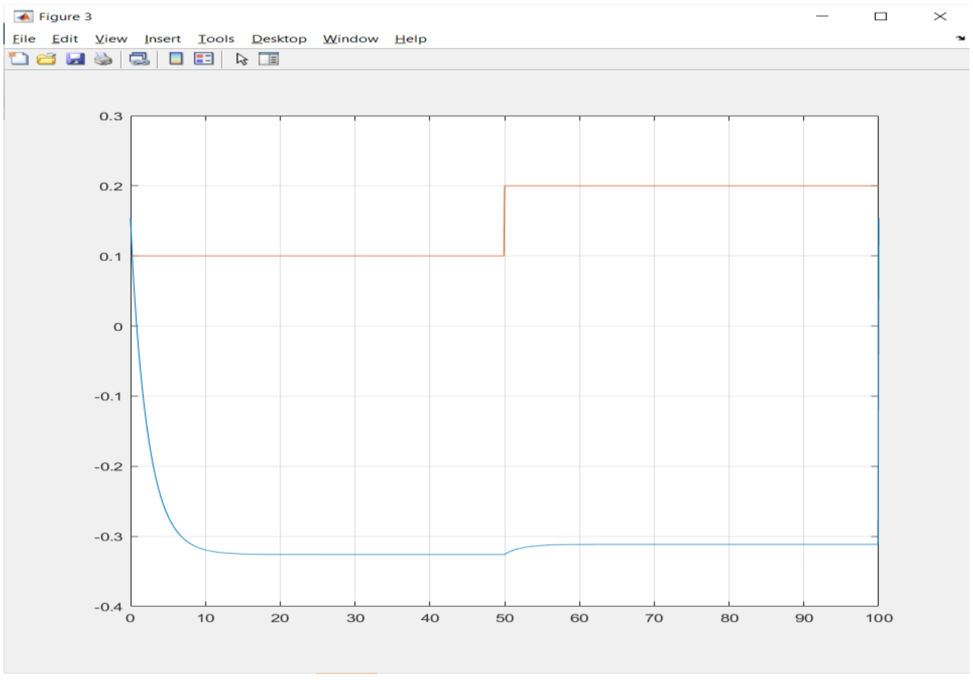


Figure 5.64: Step response for n=2

In the System identification model for n = 2, the controller does not reach steady state, showing similar behavior as for n = 1 from the previous section; the B_conc does not reach the Setpoint here. The system identification model does not show the same behavior as the original model. The output for the system order n = 2 is showing the same behaviour as it was for n = 1.

5.5.3.8 System order n = 3

For the system order n= 3, the same change applies in Appendix C to obtain the system matrices.

```
>>[A,B,D,E,CF,F,x0] = dsr(y_c-y_bias, u_c-u_bias,3)
Ordering the given input output data
QR decomposition
System order ? ..................... ( 1 ) =? 3
A = 0.9504    1.0187   -1.4480
    0.0024    0.6748   -2.6866
    0.0000    0.0025    0.8401

B =-0.0122
    0.0007
    0.0000

D =-0.6075    0.7277    0.3184

E = 1.7271e-06

CF =1.0e-03 *
   -0.4371
   -0.1343
    0.0120

F = 7.7640e-05

x0 =0.0615
    0.0010
   -0.0001
```

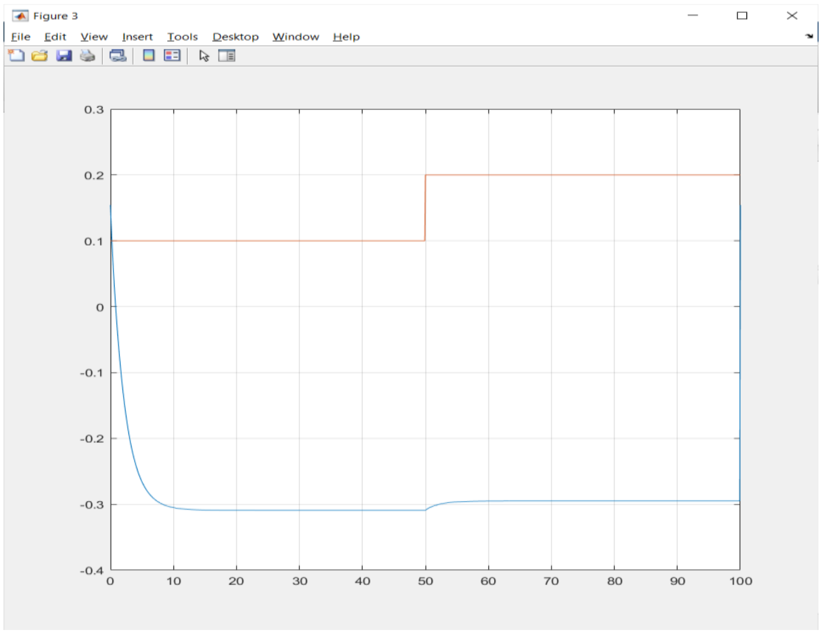


Figure 5.65: Step response for n = 3

For the System identification model for n = 3, the controller does not reach steady state, showing similar behavior as for n = 1 and 2 from the previous sections; the B_conc does not reach the Setpoint here as well. The system identification model does not show the same behavior as the original model.

Table 5.2: Closed loop model: Case 2 System matrices

Case 2: Input: B_in_c and Output: B_conc								
	A			**B**	**D**			**E**
n = 1	0.9472			0.0067	1			3.74E-06
n = 2	0.9491	-1.6104		-0.0105	-0.7259	-0.6878		-2.20E-05
	-0.0018	0.6443		-0.0004				
n = 3	0.9504	1.0187	-1.448	-0.122	-0.6075	0.7277	0.3184	1.73E-06
	0.0024	0.6748	-2.6866	0.0007				
	0	0.0025	0.8401	0				

The above table for the Closed loop identification is made using the dsr method for system orders n =1, 2 and 3, with input as the setpoint and the concentration of B, B_conc as the outpoint. All Three models made with n = 1, n = 2 and n = 3 show similar behaviour, but they vary from the original model built in Simulink. Higher system orders can be checked with the same models.

In other words, it can be said that the information given out after the system identification step does not give a good detail about the model used originally.

5.6 Case 3: B_in_c and B_conc with disturbance in A_in

Input: B_in_c (operated through change in A_in)

Output: B_conc

5.6.1 PRBS signal

First, starting with a PRBS signal for A_in ranging from 1 to 1.1.

A PRBS signal was needed as the output is the Conc of B now. Hence, PRBS signal from 1 to 1.1 is generated(u+0 is that lower limit), and then divide it by 10 to attain a signal from 0 to 0.1, and then adding a constant value 1 will make a signal from 1 to 1.1.

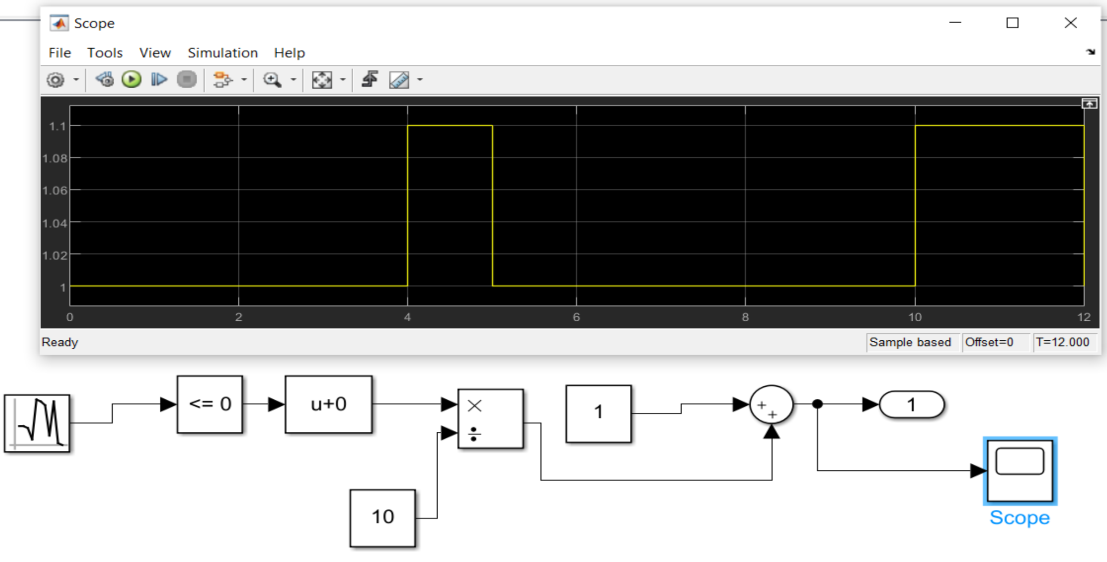
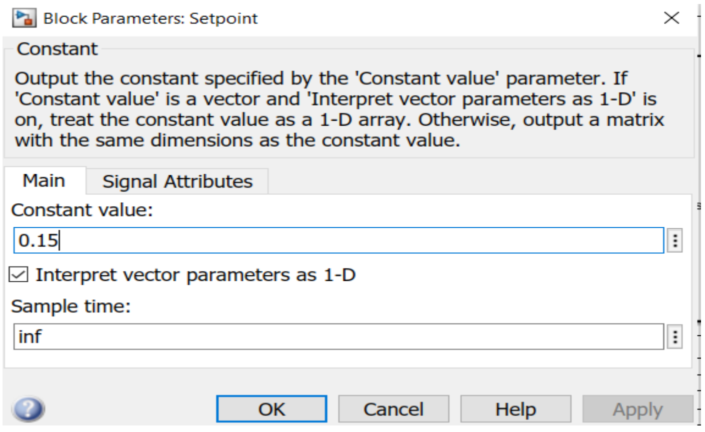


Figure 5.66: PRBS for A_in for case 3

PRBS signal for A_in from 1 to 1.1. By applying a PRBS signal to the A_in with a change in signal for every 1 second (10 readings/units), the disturbance signal is used to observe the behavior of the constructed closed loop response for case 3.

5.6.2 Setpoint

The setpoint is set as a constant with the value = 0.15. After tuning the controller, replace the setpoint with a constant value = 0.15. The Setpoint for the B_conc is set to a constant value = 0.15.



Block Parameters: Setpoint

Constant

Output the constant specified by the 'Constant value' parameter. If 'Constant value' is a vector and 'Interpret vector parameters as 1-D' is on, treat the constant value as a 1-D array. Otherwise, output a matrix with the same dimensions as the constant value.

Main | Signal Attributes

Constant value: 0.15

☑ Interpret vector parameters as 1-D

Sample time: inf

OK | Cancel | Help | Apply

Figure 5.67: Constant block for setpoint

The PID controller is used to have a control over the feed B_in as the same was done in the Open loop as well. In order to achieve this, the feed A_in is controlled, this is done because the PID controller is controlling the feed B_in, not allowing us to have control over it. This finally controls the B_conc (the output).

5.6.3 Simulink design

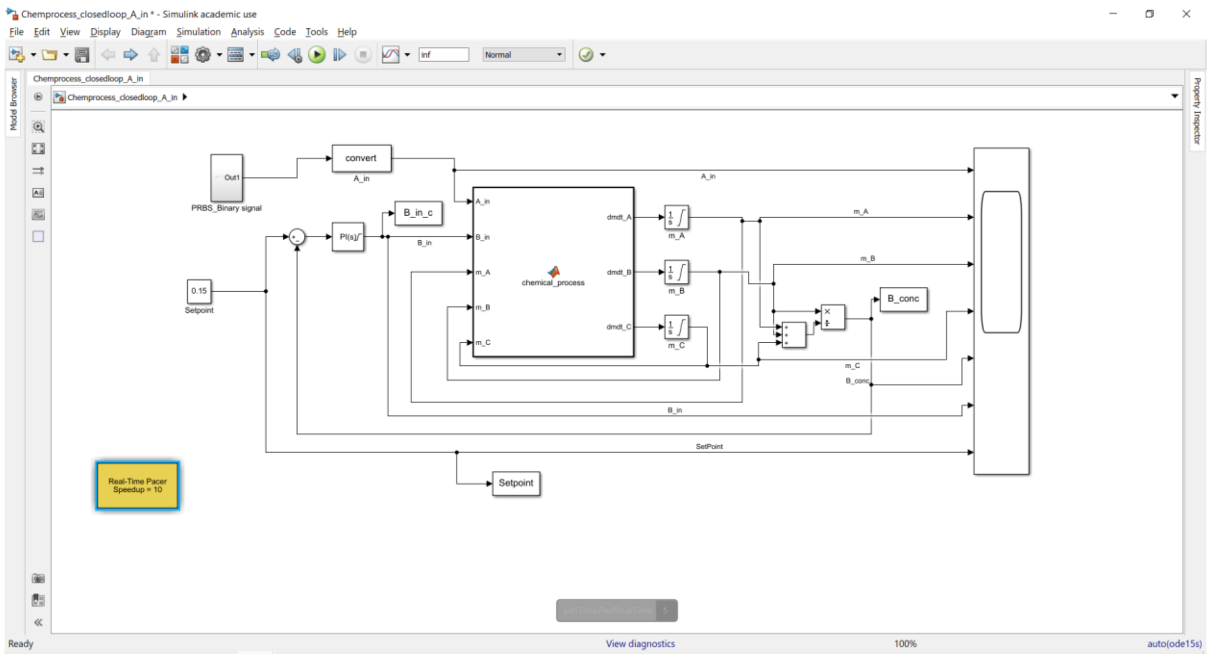

Chemprocess_closedloop_A_in * - Simulink academic use
File Edit View Display Diagram Simulation Analysis Code Tools Help
inf
Normal
Chemprocess_closedloop_A_in
Out1
PRBS_Binary signal
convert
A_in
B_in_c
B_in
PI(s)
0.15
Setpoint
chemical_process
dmdt_A
dmdt_B
dmdt_C
m_A
m_B
m_C
B_conc
SetPoint
Setpoint
Real-Time Pacer Speedup = 10
Ready
View diagnostics
100%
auto(ode15s)

Figure 5.68: Simulink model for case 3

The next image shows the response of the closed loop system constructed with the new PRBS signal and the constant setpoint. The plot legend and the colours assigned to each variable are the same as the above step response setup.

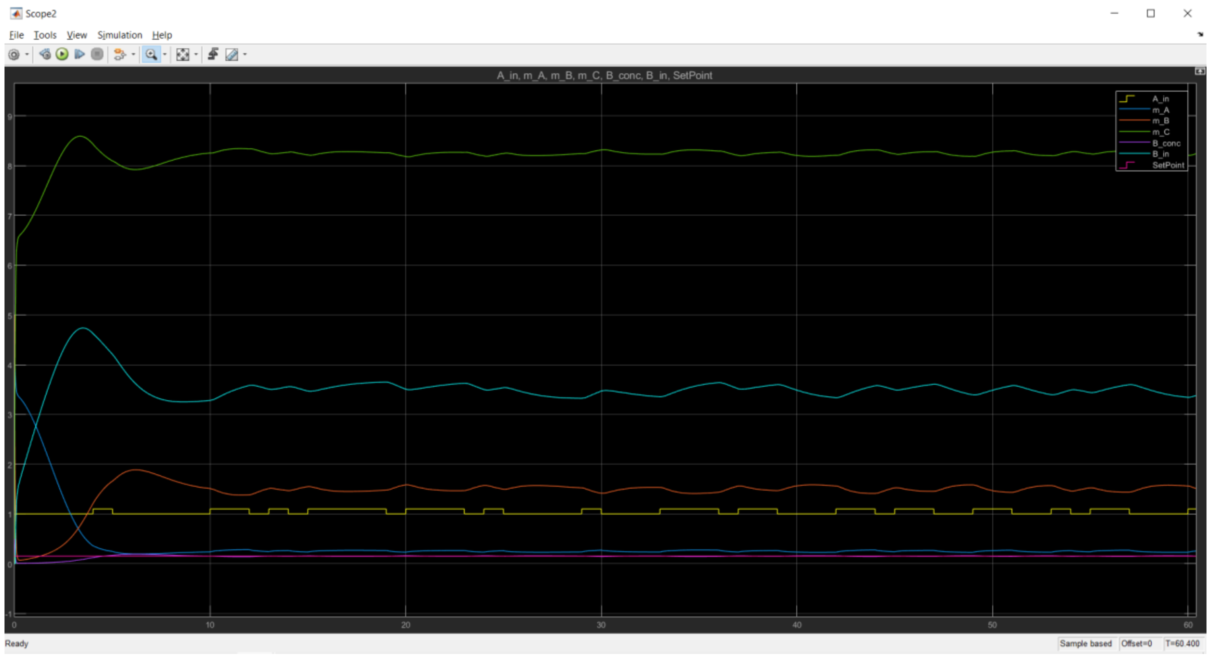


Figure 5.69: Response for A_in change model

The input signal A_in, the Yellow line, shows a PRBS signal and all the other variables show the behaviour accordingly. The B_in is ranging around 3.4 to 3.7 when the disturbance is applied. The m_A is utilized the least around 0.3 represented by the Blue line. The Red line m_B shows the m_B in the reactor and ranges around 1.4 and 1.6 with changes in the PRBS signal and the m_C is above 8. The output of the model, concentration of B, B_conc in Purple colour, is seen between 0.1 and 0.2, as the setpoint is a constant at 0.15, in Pink.

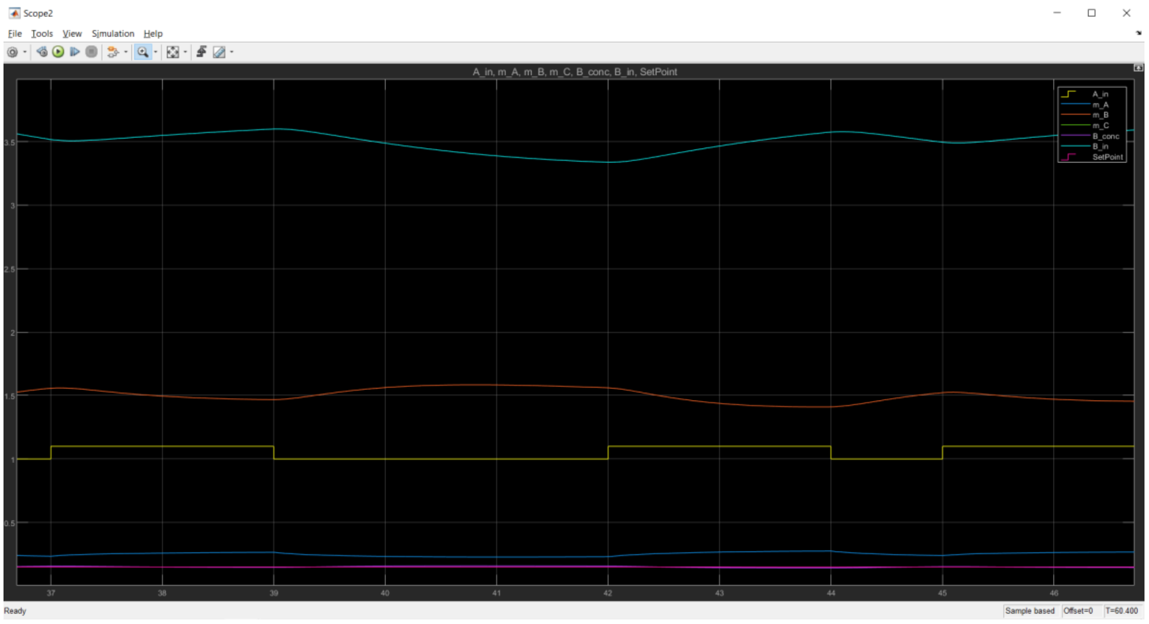


Figure 5.70: PRBS signal in focus

A closer look of the PRBS signal is shown in Figure 5.70

5.6.3.1 Saving the Simulink results

This data is first saved into variables as input u_c and output y_c from the out from the Simulink.SimulationOutput. Both sets of data are saved into excel sheets using the command “xlswrite(‘Closed_input.xlsx',u_c)” and “xlswrite('Closed_output.xlsx',y_c)”, which could be reused in the future for analysis.

To load the saved data from an excel sheet, the command “u_c = readtable('Closed_input.xlsx', 'range', 'B101:B605');” and “y_c = readtable('Closed_output.xlsx', 'range', 'E101:E605');” are used.

Here, the data of the first 10 seconds is not used for system identification, as the system is reaching the steady state after the first 10 seconds. This removal of initial data is made to ensure there is no disturbance and there is accurate data for model identification.

5.6.3.2 Checking for Synchronization

To check for synchronization of data with the Simulink model, the input u_c and output y_c is used to plot and compare using the below commands:

```
% synchronize u and y...
>> figure(2)
clf
plot([u_c y_c])
grid on
```

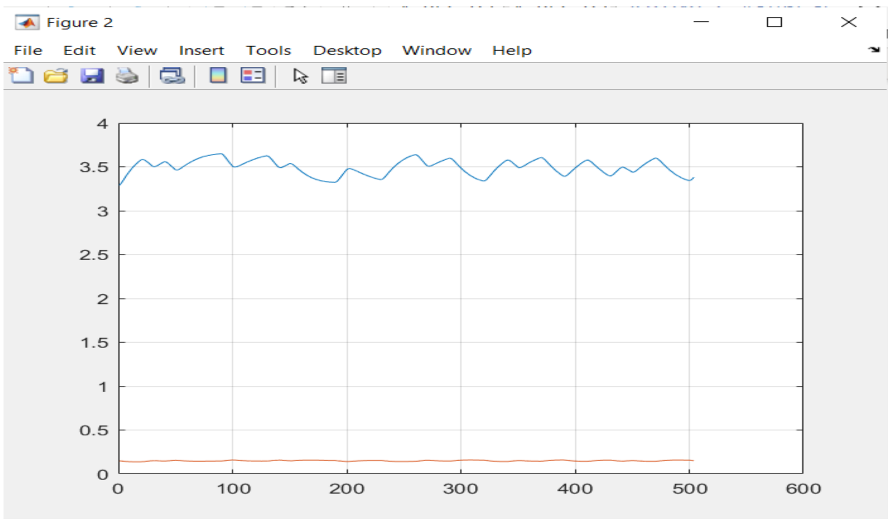


Figure 5.71: input and output plot confirmation

This gives a plot of the input u_c and y_c as Figure 5.71. Next, to compare this plot with the original model, Figure 5.72 shows the cursors with measurements.

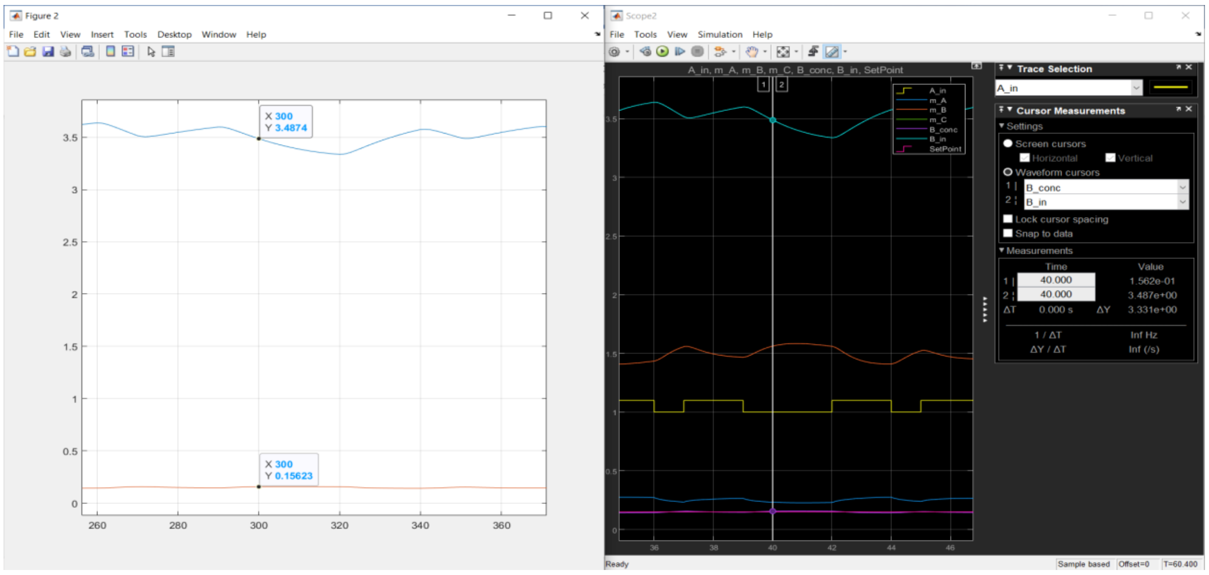


Figure 2
File Edit View Insert Tools Desktop Window Help
X 300
Y 3.4874
X 300
Y 0.15623
Scope2
File Tools View Simulation Help
A_in, m_A, m_B, m_C, B_conc, B_in, SetPoint
A_in
m_A
m_B
m_C
B_conc
B_in
SetPoint
Ready
Sample based
Offset=0
T=60.400

Trace Selection: A_in

Cursor Measurements

Settings
- Screen cursors: Horizontal, Vertical
- Waveform cursors: 1 B_conc, 2 B_in
- Lock cursor spacing
- Snap to data

Measurements

	Time		Value
1	40.000		1.562e-01
2	40.000		3.487e+00
ΔT	0.000 s	ΔY	3.331e+00
	1 / ΔT		Inf Hz
	ΔY / ΔT		Inf (/s)

Figure 5.72: Synchronization

As the first 10 seconds data is removed as it is a disturbance, the comparison will be made for 300th reading in LHS and 40th second in RHS between B_in and B_conc. It is seen than in LHS, the Red line, is the B_conc, and the value is 0.1562, which is the same in the RHS at the 40th second. The same is seen for B_In as 3.487 in both the plots, confirming the data is synchronized as expected.

5.6.3.3 Model Identification:

The input and output data from above are used for identification of the model using the DSR toolbox, Description given in Appendix B.

The DSR toolbox is used to identify the Closed loop system to find the system matrices. Using the input, output, and their biases in order to be scaled and system order n = 1, the below command is given to obtain the system matrices:

```
>> u_bias = mean(u_c);

y_bias = mean(y_c);

>> [A,B,D,E,CF,F,x0] = dsr(y_c-y_bias, u_c-u_bias,1)

Ordering the given input output data

QR decomposition

A =     0.9062


B =     0.0088


D =      1
```

```
E =   -0.0938


CF =   3.5397e-05


F =   6.9029e-07


x0 =-0.0191
```

5.6.3.4 Making a step response

After obtaining the system matrices for the Closed loop, the step response is plotted for u= 1 to 1.1, similar to the one used for A_in. This is to make sure the comparison is made the same way as it was done for the original Simulink model. A step response is applied at time = 500.

```
>>dt = 0.1;
>> t = 0:dt:100;
n = length(t);
u = ones(1,n)*1;
steptime = 500;
u(steptime+1:n) = ones(n-steptime,1)*1.1;
u = u - u_bias;
```

5.6.3.5 Generating the output using the system matrices

Next the equations 6.1 and 6.2 are written as MATLAB code to generate the output of the step response with the system matrices A, B, D and E as shown in the equations.

```
x = zeros(1,n);
y = zeros(1,n);
>> for i = 1:n-1
     x(:,i+1) = A*x(:,i) + B*u(:,i);
     y(:,i) = D*x(:,i) + E*u(:,i);
end
```

The subtracted biases are now added for scaling, to make sure the plotting is appropriate.

```
% Add bias/scaling to get original values
>> u = u + u_bias;
```

```
y = y + y_bias;
figure(3)
clf
plot(t,[y', u'])
grid on
```

The above code can be found in Appendix C.

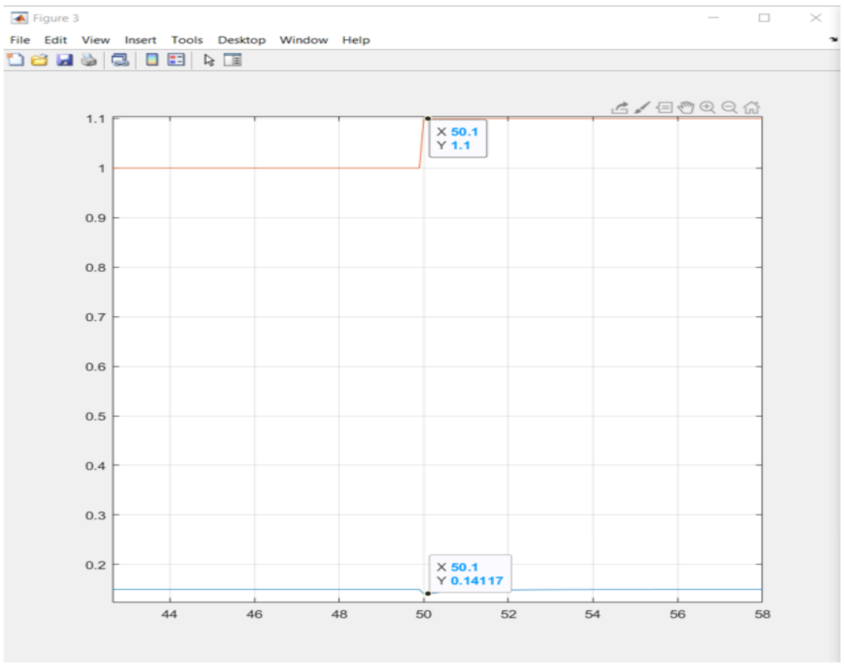
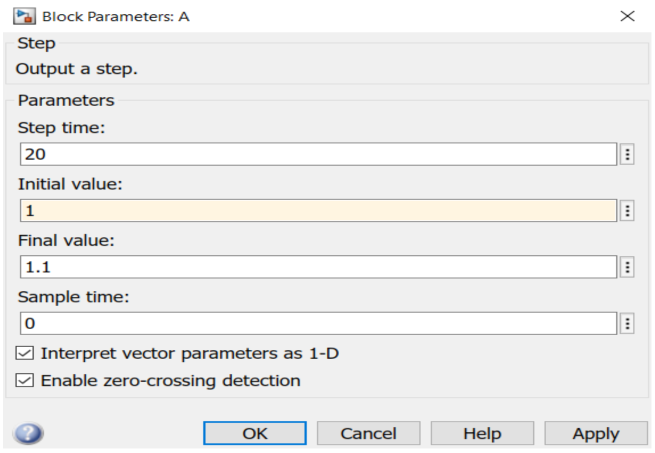


Figure 5.73: Step response for Case 3 n=1

Figure 5.73 is the step response obtained using the system matrices.

5.6.3.6 Comparing Simulink model with identification model

To compare this, a step response model was made, replacing the PRBS signal Figure 5.74.



Block Parameters: A

Step

Output a step.

Parameters

Step time:

20

Initial value:

1

Final value:

1.1

Sample time:

0

☑ Interpret vector parameters as 1-D

☑ Enable zero-crossing detection

Figure 5.74: Step change block of A_in

The A_in is being controlled by creating a disturbance with a step response with a step change from 1 to 1.1 at time = 20 seconds.

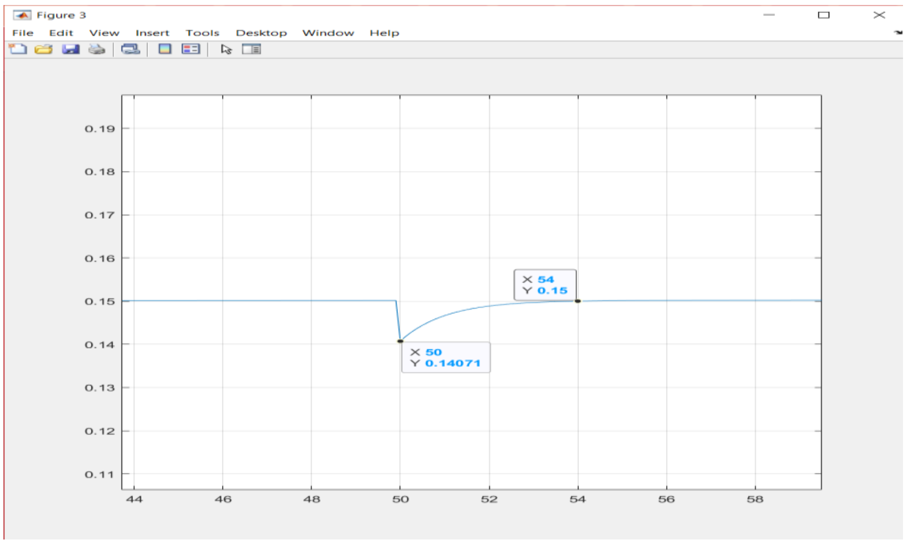
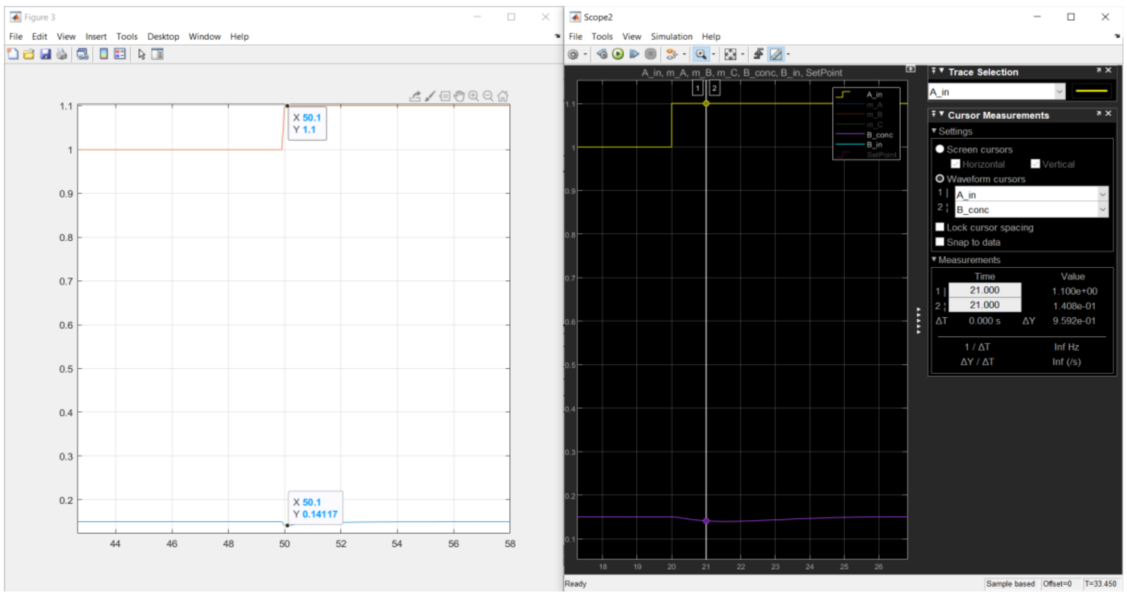


Figure 5.75: System identified model step response

Figure 5.75 is the step response of the model made from the system matrices in the above section, which is also the LHS part in Figure 5.76: Comparison of identified model and Simulink model. The curve drops to 0.14 in 1 second, and the curve looks more like a negative overshoot. And it reaches 0.15, the steady state exactly after 4 seconds. As the controller brings back the B_conc to the original value, the gain is finally Zero here.



	Time	Value
1	21.000	1.100e+00
2	21.000	1.408e-01
ΔT	0.000 s	ΔY 9.592e-01
1 / ΔT		Inf Hz
ΔY / ΔT		Inf (/s)

Figure 5.76: Comparison of identified model and Simulink model

Whereas the output curve reaches 100% around 7 seconds in the original model, in RHS in Figure 5.76. The response drops from 0.15 to 0.1172, but as the controller brings back the B_conc to the original value, the gain is finally Zero here in the RHS.

Similarly, the system matrices are found for when system order n=2 and n=3 and then step responses are made for comparison.

5.6.3.7 System order n = 2

The only changes in Appendix C are n = 2 in dsr and defining x, as shown below.

```
>> [A,B,D,E,CF,F,x0] = dsr(y_c-y_bias, u_c-u_bias,2)
Ordering the given input output data
QR decomposition
System order ? ..................... ( 1 ) =? 2


A =     0.9063   -1.9049
       -0.0001    0.9985


B =    -0.0120
       -0.0000


D =    -0.7410   -0.6715
```

```
E =   -0.0939


CF =   1.0e-05 *
      -0.1920
       0.0372


F =   5.7297e-08


x0 =  0.0258
      0.0000
```

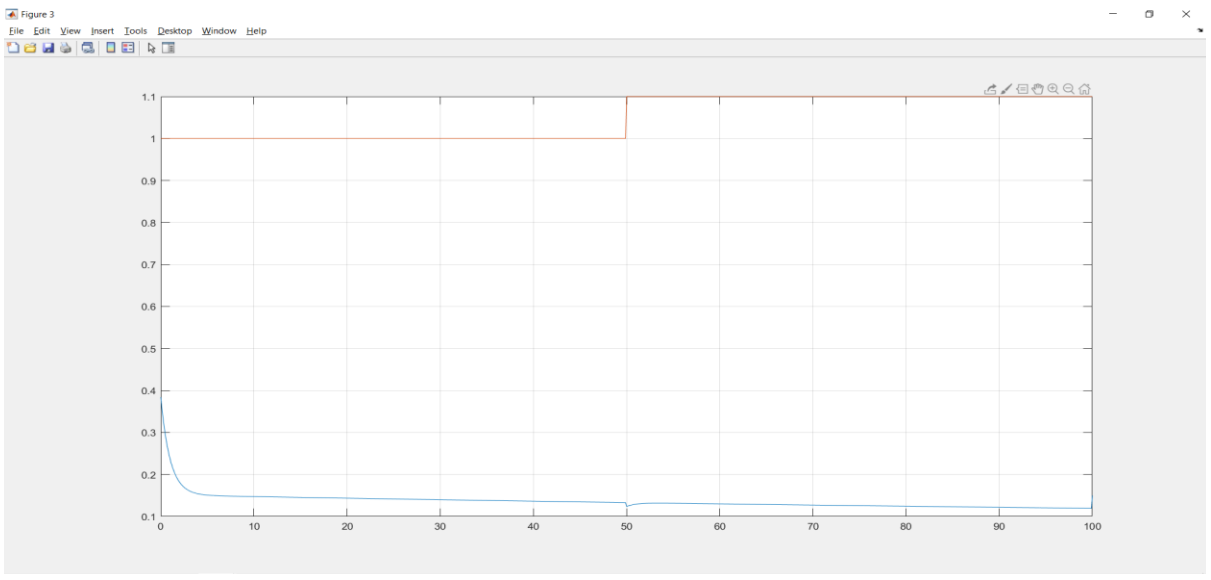


Figure 5.77: Step response for n=2

The output here is very strange, and maybe the model identification model for n = 2 is not reaching a steady state, it is gradually decreasing.

5.6.3.8 System order n = 3

And the third trial is done for system order n= 3 to Appendix C. The same change in command applies while using the dsr command and defining x to obtain the system matrices.

```
>>[A,B,D,E,CF,F,x0] = dsr(y_c-y_bias, u_c-u_bias,3)
Ordering the given input output data
```

```
QR decomposition
System order ? ..................... ( 1 ) =? 3

A =     0.9064    1.1677    0.3653
        0.0000    0.9995    1.4808

       -0.0000    0.0003   -0.1421
B =    -0.0143
        0.0005
       -0.0004

D =    -0.6329    0.6701   -0.3878

E =    -0.0943

CF =    1.0e-06 *
        -0.3032
        -0.0694
         0.0009

F =     4.7405e-08

x0 =    0.0302
       -0.0001
        0.0001
```

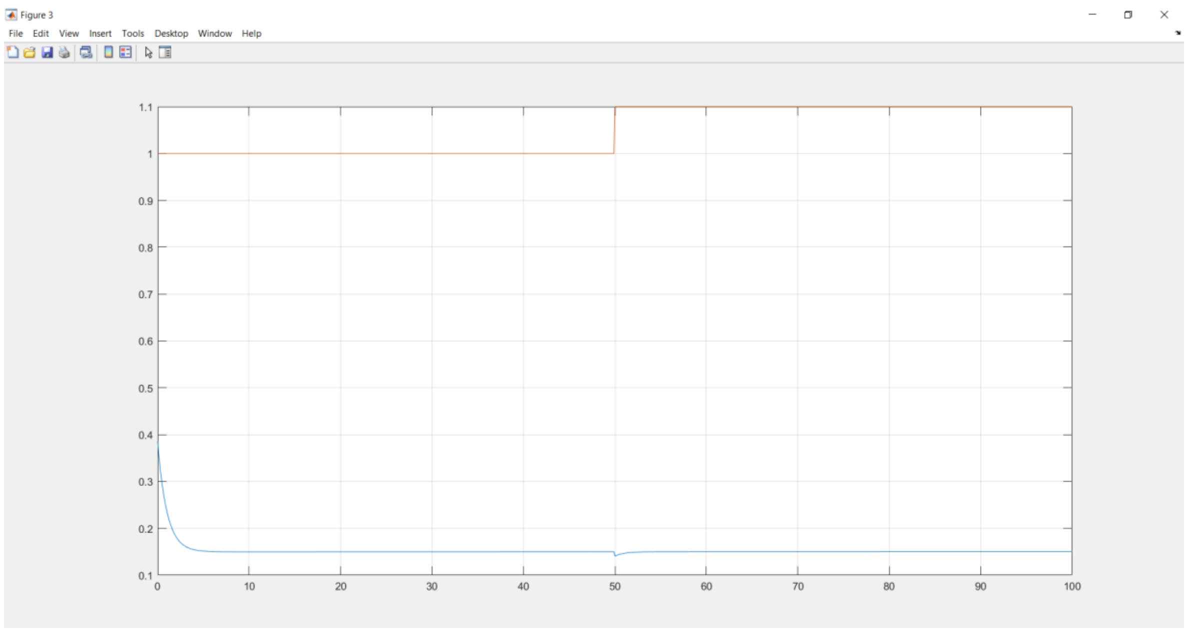


Figure 5.78: Step response for n = 3

The output curve reaches 100% after 4 time steps. As the controller brings back the B_conc to the original value, the gain is finally Zero here for the model identified using the system matrices formed using order n=3.

Table 5.3: Closed loop model: case 3 system matrices

Case 3: Input: B_in_c and Output: B_conc by creating disturbance in A_in								
	A			B	D			E
n = 1	0.9062			0.0088	1			-1.91E-02
n = 2	0.9063	-1.9049		-0.012	-0.741	-0.6715		-9.39E-01
	-0.0001	0.9985		0				
n = 3	0.9064	1.1677	0.3653	-0.0143	-0.6329	0.6701	-0.3878	-9.43E-02
	0	0.9995	1.4808	0.0005				
	0	0.0003	-0.1421	-0.0004				

Table 5.3 that the Closed loop identification using the dsr method for system orders n =1, 2 and 3 show the similar behaviour as with each other, but they vary from the original model built in Simulink. The models are shown only for system orders 1, 2 and 3. Higher system orders can be checked with the same models.

Table 5.4: summary of all 3 cases in closed loop

Case 1: Input: Setpoint and Output: B_conc								
	A			B	D			E
n = 1	[0.9415]			0.0747	1			-2.20E-03
n = 2	0.9578	-1.5198		-0.1034	-0.7221	-0.6917		-1.20E-03
	0.0001	0.5426		-0.0034				
n = 3	0.9588	1.1786	1.3727	-0.1227	-0.613	0.6964	-0.3917	1.27E-04
	-0.003	0.9622	2.6815	0.0068				
	-0.0002	0.0006	0.577	-0.0013				
Case 2: Input: B_in_c and Output: B_conc								
	A			B	D			E
n = 1	0.9472			0.0067	1			3.74E-06
n = 2	0.9491	-1.6104		-0.0105	-0.7259	-0.6878		-2.20E-05
	-0.0018	0.6443		-0.0004				
n = 3	0.9504	1.0187	-1.448	-0.122	-0.6075	0.7277	0.3184	1.73E-06
	0.0024	0.6748	-2.6866	0.0007				
	0	0.0025	0.8401	0				
Case 3: Input: B_in_c and Output: B_conc by creating disturbance in A_in								
	A			B	D			E
n = 1	0.9062			0.0088	1			-1.91E-02
n = 2	0.9063	-1.9049		-0.012	-0.741	-0.6715		-9.39E-01
	-0.0001	0.9985		0				
n = 3	0.9064	1.1677	0.3653	-0.0143	-0.6329	0.6701	-0.3878	-9.43E-02
	0	0.9995	1.4808	0.0005				
	0	0.0003	-0.1421	-0.0004				

From the above models' analysis, it can be told that Closed loop identification does not show the same behaviour as the original model. The identified model does not provide the same information as the original model.

5.7 Comparing the Open loop response with the closed loop response:

To compare the open loop and the closed loop responses, it is best to use case 3 from the closed loop trial. In the open loop identification testing, without a controller, the input was the feed of B, B_in and B_conc was the output. And the system provides good response with the system matrices that were found using the DSR method of system identification. In the closed loop, the same is expected to be found, i.e., the input is B_in and B_conc is the output, and the controller is in operation. The purpose of the experiment is to demonstrate that the response from the identification is completely different from using a controller with no controller in the system.

5.7.1 Comparison of Open loop and closed loop Simulink models

The result of step responses from both the Simulink models are used in Figure 5.80: Compare open and closed response.

In LHS, the Open loop step response, the step change is applied for Feed B_in from 3 to 4 at 20 seconds and the output is the B_conc.

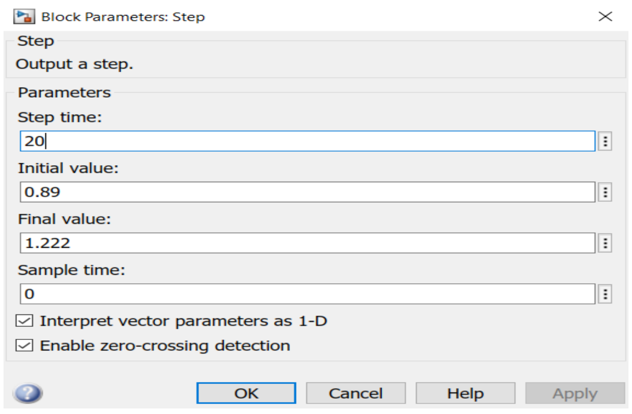

Block Parameters: Step
Step
Output a step.
Parameters
Step time: 20
Initial value: 0.89
Final value: 1.222
Sample time: 0
☑ Interpret vector parameters as 1-D
☑ Enable zero-crossing detection

Figure 5.79: Step change for A_in in closed loop

In RHS, the Closed loop step response is made for A_in from 0.89 to 1.22, this is made because a change in response is observed for the Feed B_in, which is the input, the controller has control over the B_conc and there is no direct control over it. Hence the A_in is changed to attain a change in B_in from 3 to 4. This was made possible using trial and error for A_in and the closest values are 0.89 to 1.22 as shown in Figure 5.79.

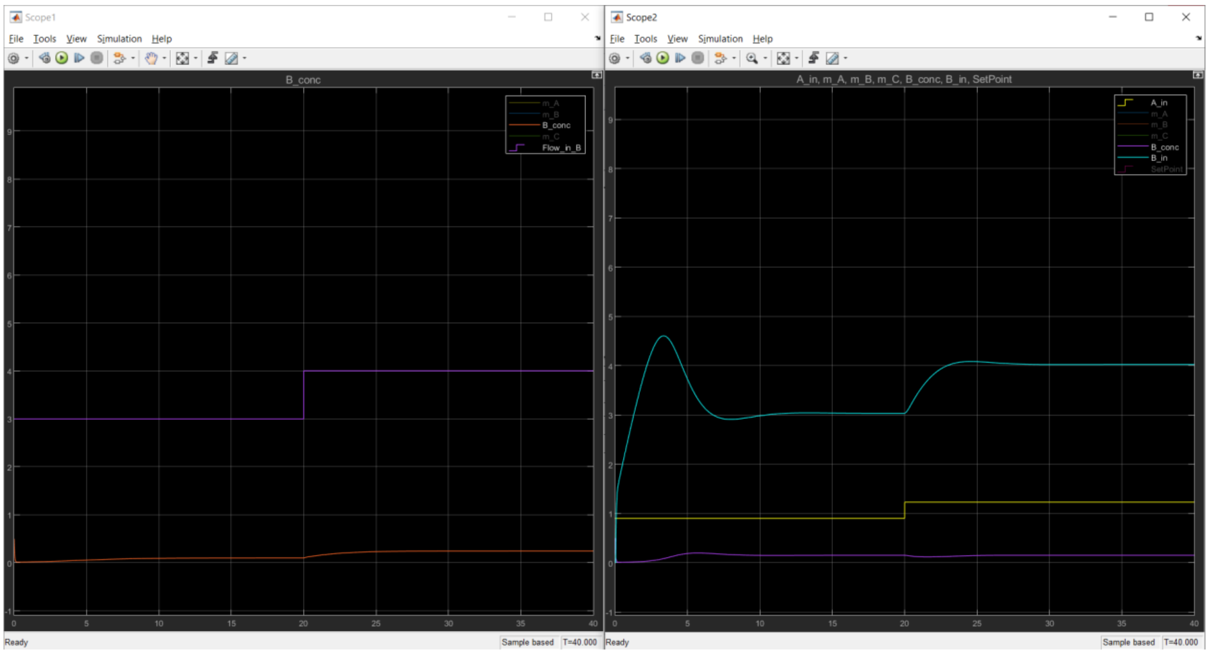
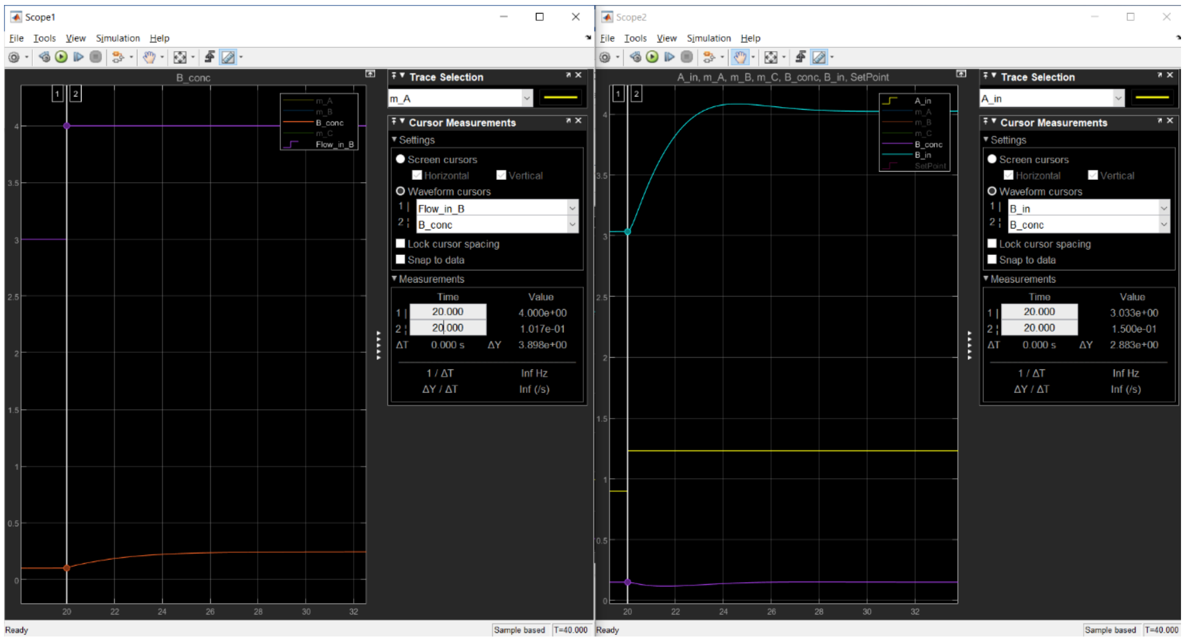


Figure 5.80: Compare open and closed response

Even with the step change applied as mentioned above, the comparison cannot be very accurate. In Figure 5.81: Comparison with step change, a step change is applied at time = 20, but in RHS, the Closed loop response, the input here is B_in, is a result of a step change applied to A_in, and hence the controller takes a time of 10 seconds to change the B_in from 3 to 4 and reach the steady state. Whereas in LHS, the step change in the open loop is directly applied to B_in and hence B_in changes from 3 to 4 in an instant.


Scope1
File Tools View Simulation Help
B_conc
Scope2
A_in, m_A, m_B, m_C, B_conc, B_in, SetPoint
Ready
Sample based
T=40.000

Scope1:

Trace Selection: m_A

Cursor Measurements

Settings

Screen cursors: Horizontal, Vertical

Waveform cursors
1 | Flow_in_B
2 ¦ B_conc

Lock cursor spacing

Snap to data

Measurements

```
      Time          Value
1 |   20.000        4.000e+00
2 ¦   20.000        1.017e-01
ΔT    0.000 s   ΔY  3.898e+00

      1 / ΔT        Inf Hz
      ΔY / ΔT       Inf (/s)
```

Scope2:

Trace Selection: A_in

Cursor Measurements

Settings

Screen cursors: Horizontal, Vertical

Waveform cursors
1 | B_in
2 ¦ B_conc

Lock cursor spacing

Snap to data

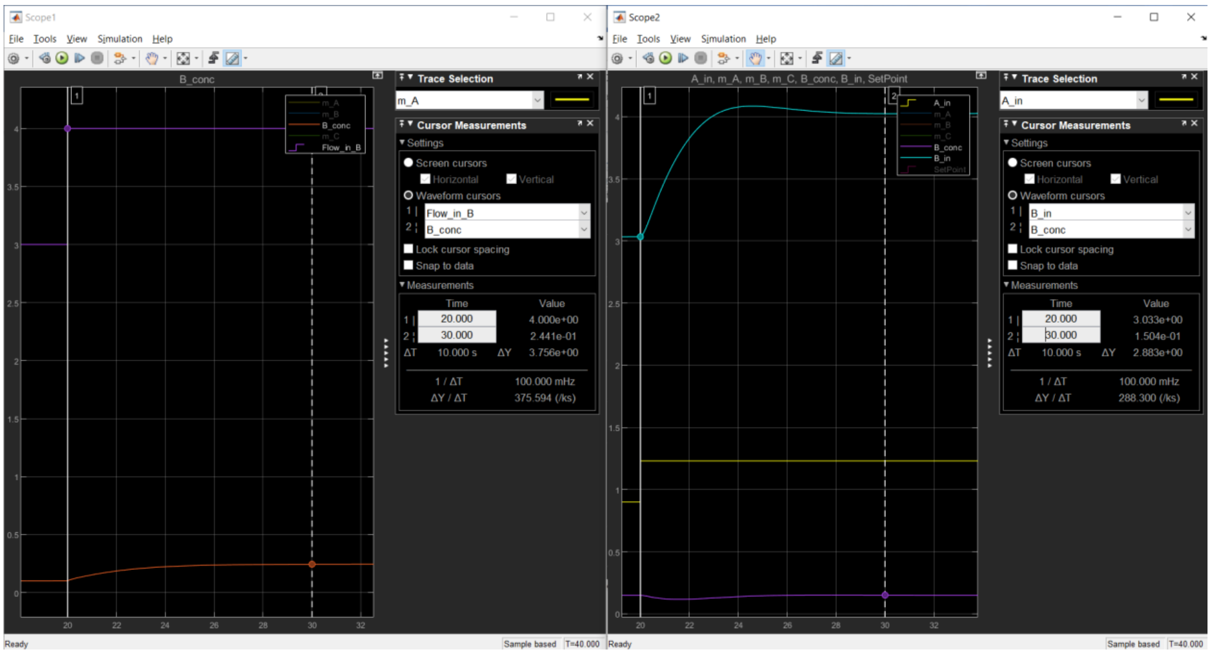
Measurements

```
      Time          Value
1 |   20.000        3.033e+00
2 ¦   20.000        1.500e-01
ΔT    0.000 s   ΔY  2.883e+00

      1 / ΔT        Inf Hz
      ΔY / ΔT       Inf (/s)
```

Figure 5.81: Comparison with step change

For both models, the B_conc, which is the output, in LHS, the open loop shows a clean step response with gain 0.2 and the time constant 2.8 Seconds. And in RHS, there is a change in B_conc, where the value drops to 0.1175 at its peak (lower), and, as the controller adjusts the B_conc to reach the same value as B_conc had before the step change was applied, as the controller brings back the B_conc to the original value, the gain is finally Zero here. The steady state for B_conc in open loop reaches after 10 seconds, and for closed loop, it reaches around 7 seconds.



Scope1

Trace Selection: m_A

Cursor Measurements

Settings
- Screen cursors (Horizontal, Vertical)
- Waveform cursors: 1 Flow_in_B; 2 B_conc
- Lock cursor spacing
- Snap to data

Measurements

	Time		Value
1	20.000		4.000e+00
2	30.000		2.441e-01
ΔT	10.000 s	ΔY	3.756e+00
	1 / ΔT		100.000 mHz
	ΔY / ΔT		375.594 (/ks)

Scope2

Trace Selection: A_in

Cursor Measurements

Settings
- Screen cursors (Horizontal, Vertical)
- Waveform cursors: 1 B_in; 2 B_conc
- Lock cursor spacing
- Snap to data

Measurements

	Time		Value
1	20.000		3.033e+00
2	30.000		1.504e-01
ΔT	10.000 s	ΔY	2.883e+00
	1 / ΔT		100.000 mHz
	ΔY / ΔT		288.300 (/ks)

Figure 5.82: Comparison after reaching steady state

To have a better comparison, both open and closed loop responses were constructed in the same Simulink model as shown in Figure 5.83. The upper model is the open loop and the lower is Closed loop model.

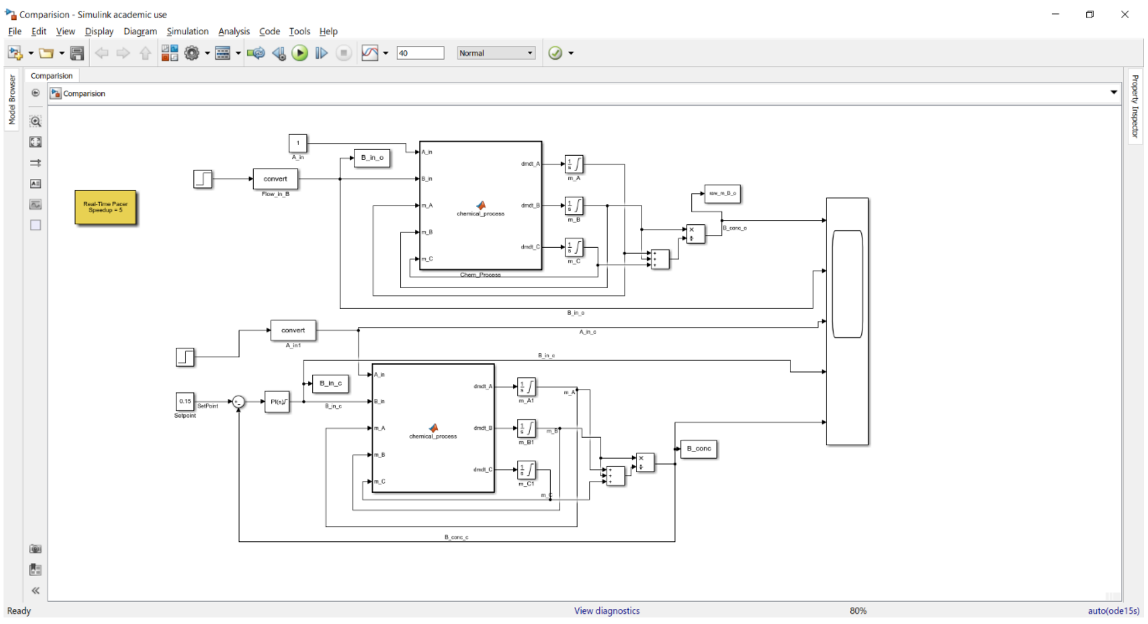
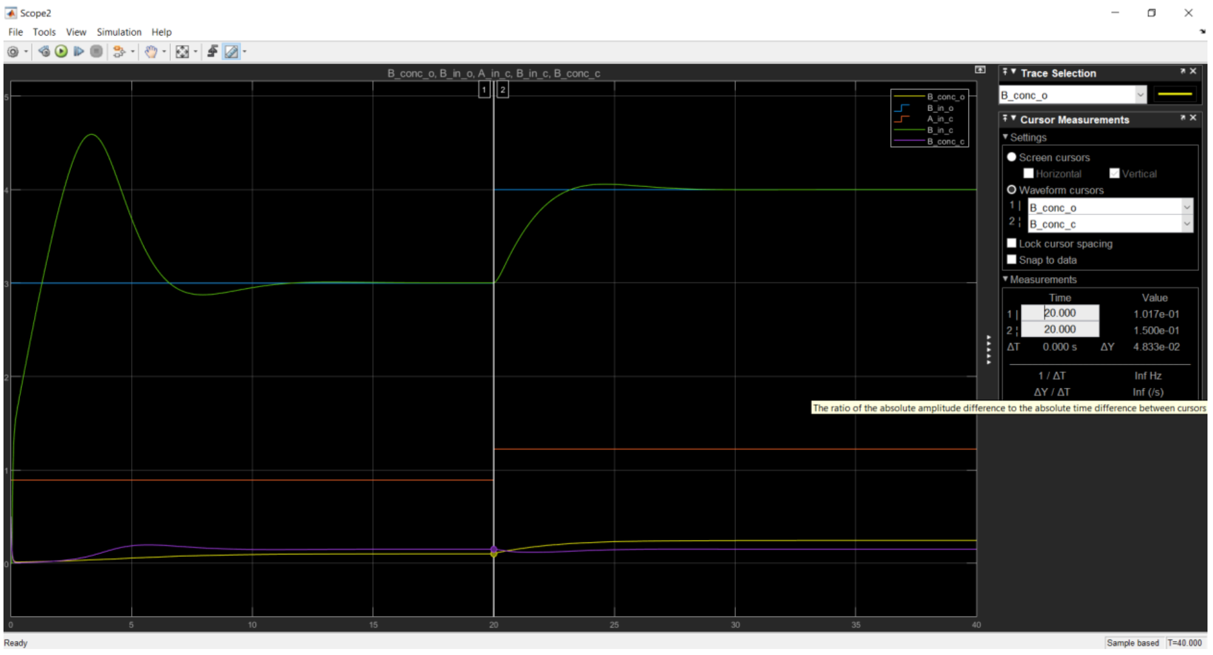


Figure 5.83: Combined Simulink model for comparing

The response of the combined Simulink model is shown in the below image.



Trace Selection
B_conc_o

Cursor Measurements
Settings
Screen cursors
Horizontal
Vertical
Waveform cursors
1 B_conc_o
2 B_conc_c
Lock cursor spacing
Snap to data

Measurements

	Time		Value
1	20.000		1.017e-01
2	20.000		1.500e-01
ΔT	0.000 s	ΔY	4.833e-02
	1 / ΔT		Inf Hz
	ΔY / ΔT		Inf (/s)

The ratio of the absolute amplitude difference to the absolute time difference between cursors

Figure 5.84: Step change for combined model

In Figure 5.84, B_conc_o is the concentration of B for open loop and B_conc_c is for closed loop, similarly the Feed of B are B_in_o and B_in_c, respectively. A step change is applied at 20 seconds, and the same response is seen as above in Figure 5.81

5.7.2 Comparison of Open loop and closed loop identified models

A combined table of the system matrices are shown below in Table 5.5.

Table 5.5: Open loop and Closed loop case 3 system matrices comparison

Open loop								
	A			**B**	**D**			**E**
n = 1	0.9579			0.0069	1			3.50E-04
n = 2	0.9663	-1.7385		-0.0103	-0.7194	-0.6946		-9.54E-06
	-0.0007	0.7648		-0.0004				
n = 3	0.9669	1.1247	1.5245	-0.0119	-0.5971	0.7136	-0.3665	5.59E-06
	0.001	0.8548	2.801	0.0008				
	-0.0001	0.0069	0.7476	-0.0001				
Closed loop: Case 3: Input: B_in_c and Output: B_conc by creating disturbance in A_in								
n = 1	0.9062			0.0088	1			-1.91E-02
n = 2	0.9063	-1.9049		-0.012	-0.741	-0.6715		-9.39E-01
	-0.0001	0.9985		0				
n = 3	0.9064	1.1677	0.3653	-0.0143	-0.6329	0.6701	-0.3878	-9.43E-02
	0	0.9995	1.4808	0.0005				
	0	0.0003	-0.1421	-0.0004				

The comparison for models with system order n= 1 in open loop identified model and closed loop identified model, for case 3 is shown in Figure 5.85.

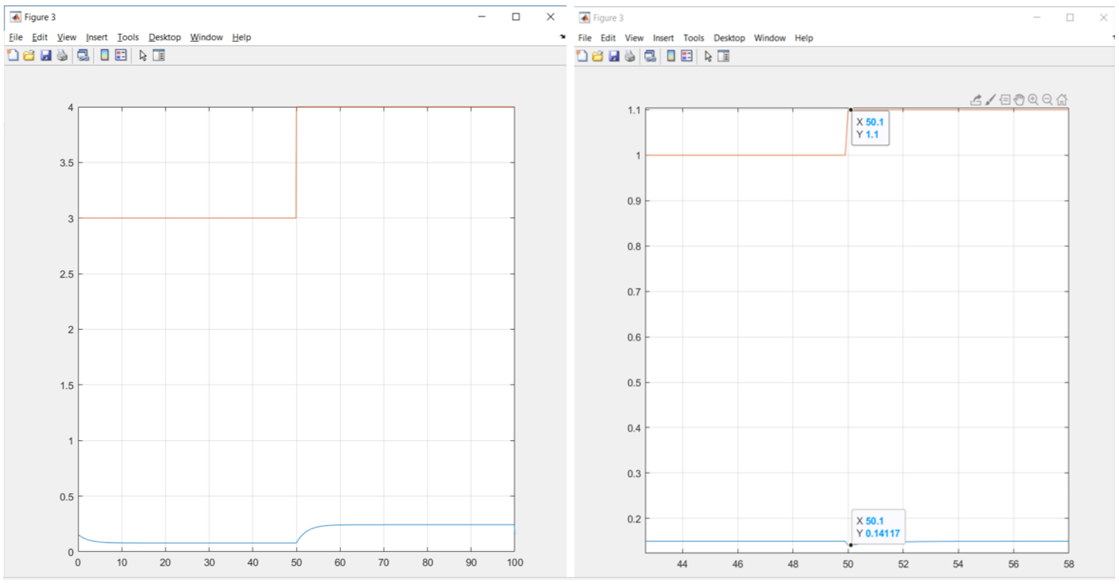


Figure 5.85: Identified model comparison

For both models, the B_conc, which is the output, in LHS, the open loop shows a clean step response with gain 0.2 and the time constant 2.8 Seconds. In RHS, there is a change in

B_conc, where the value drops to 0.078 at its peak (lower), and after the step change is applied, the B_conc rises to 0.218 at steady state. The response is completely different from the response expected as the set point changes from 0.1 to 0.2, the gain is RHS is Zero as discussed in the Closed loop section. The steady state for B_conc in open loop and closed loop reaches after 10 seconds.

The above response makes it very clear that closed loop data of any model is not sufficient to show the same information of any process model. And hence, the open loop should be used for model identification.

5.8 Another Example: An Electric Air heater

To control the temperature of a room, using a heater, how much electric power must be supplied or increased in order to increase the temperature by one degree. First, the mathematical model is found for the above problem specified. Similar to the reactor model in the demonstration case, how much Feed of B, B_in is needed to increase a certain amount of concentration of B, B_conc. To start, first an open loop for this system is made.

5.8.1 Open loop:

Case 1: No change in outdoor temperature or no disturbances are present, then, increase the power by say 100Watt, and wait till the room temperature is stable and find the ratio of change in Temperature and change in Power at steady state (dT/dP). In the same way, the PRBS signal is applied in the reactor model, where the signal moves up and down alternately, to capture the response and a dynamic model is available for m_B.

Case 2: Disturbances, external disturbances like outdoor temperature changes, or a window is slightly opened: Then the same process is repeated, make a step change in the Power (input), and after waiting for the system to reach steady state. After repeating the process for a several times and then taking an average of the answer, similar to identification routine. The process model of the reactor is already available, and the results for plotting are shown in the Open loop Simulation section. It is shown that the B_conc changes with change in feed of B, B_in. After the system matrices are identified using the system identification, it was seen to be the same as the Simulink model. This is to verify that the Open loop is correct, and the model identification is matching the Simulink model.

5.8.2 Closed loop:

The difference in a closed loop to the open loop is that the open loop uses a sensor (a thermostat for the air heater example). The Power is not changed manually, the Temperature is automatically changed when a Temperature setpoint is given and keeps the temperature constant. In the same way, a controller is placed in the closed loop Simulink model. And the Controller keeps the B_conc constant. The ultimate observation to be made here is the response of the change in temperature to the change in power. Or for the Simulink demonstration case made, the response of the concentration of B, B_conc, made to the change in input, feed of B, B_in.

Case 1: No change in outdoor temperature or no disturbances are present. No change in T, and no change in P, so there is no dP and dT, meaning there is no information, just some noise. In the Reactor model, there will be constant B_in and B_conc. So, if there is no change in the input, then the model does not provide any information as the system is in a steady state. Only by using a step change to the input, or the setpoint, a model can be made.

Case 2: Disturbances, external disturbances like outdoor temperature changes, or a window is slightly opened: First, the temperature will drop a little, and then the thermostat will increase the voltage accordingly and compensate the temperature change by adjusting the Power to make the temperature equal to the desired setpoint value. No, if the thermostat is working properly, then the temperature is kept fairly constant, (meaning dT almost =0), but there is a constant change in the input P (or dP). Hence, the effect is ultimately shown wrongly (dT/dP = 0), which is not true, it is completely wrong. Similarly, the feed of B, B_in is being adjusted by the controls in the reactor. Although there is a change in the B_in to compensate for the change in B_conc, the effect shown is incorrect. Or it can be said that the Closed loop response does not show the

In the First case, Open loop, if the Power is changed by 100W, how much of the output has changed will be seen. But in the second case, Closed loop, having a change in the Power, the temperature will not change. There will be no observation seen in the output change, and this change is not known. And the mode identified will show a completely different response when compared to the original model, because there are disturbances that are unknown.

In other words, if an observer is provided with only the Closed loop response, then the observer will feel that there is no effect or no relation between the input and the output. Or the mathematical model made from the Closed loop response will not show the actual response that the original process model.

6 Scope for future work

The possible weaknesses that have been found in the study are listed in this section. Some solutions to overcome them are provided. Although the open loop identified model showed reasonable outcome and provided the same model as the original, the closed loop identification evidently did not provide the necessary information to represent its process model. Therefore, a general solution cannot be defined for identification of a closed loop model, as confirmed by the demonstration case.

There are a few cases where system identification can be applied to find an open loop data with a closed loop response. This will, in the end, give the same information as the process model. Knowing the disturbances and the controller dynamics could help better understand the process model although other conditions could also have been bearing an influence. These techniques will have their own disadvantages though a few papers would argue against it.

In [29], the author has spoken about how to overcome the challenges faced in closed loop identification. There are 3 approaches proposed:

The Direct approach where the feedback is ignored, and open loop data is used as input and output measurements for model identification. The second method is the Indirect approach, where the closed loop transfer function is identified, and the open loop parameters are drawn from it. The third approach is called the Joint Input-Output approach, where both the input and output are jointly used bearing the output with an influence of a setpoint, and then open loop parameters are determined.

In [30], DSR_e method is introduced, which can be used on both open loop and closed loop identification. In this method, both the signal and the innovations part are used for identification and the Direct feed through matrix $E = 0$. There are 2 steps in the algorithm used here: First is to split the future output into a signal part and an innovations part. The second step is where deterministic subspace system identification step is applied to find the system order.

The challenges found in the thesis can be addressed using the proposed methods. Although these need not necessarily be the only methods available, and these methods found can be used as practical alternatives on real plants.

7 Conclusion

Literature survey was conducted on model identification for industrial plants and for processes controlled in Closed-loop models.

Demonstration cases for both open and closed loop models were presented in MATLAB, these models were identified and compared with the original process models. Also, both simulated models and the identified models of open and closed loop were compared based on the response with the same input and output system variables.

Using an open loop model for identification, in most cases at least, the identified model gave the same amount of information as the original, and also showed the same process behavior. The newly identified model can therefore be used to define the process model.

When a closed loop system is used to identify the system, the information obtained is not just about the process model, but also about the controller, which is very different from the process behavior. There are a few other complicated methods that can be used to identify the process model using these closed loop responses when there are disturbances applied.

The data from the simulation models, both in open and closed loop, obtained from saved process variables make them available for analysis, which can be used as historical data. Although, the analysis could not identify the information of process models that were responsible for the behavior of the closed loop response, the only possible explanation would be to say that the input and output data were influenced by the controller and hence the model identification was not successful for the closed loop model. It is not possible however to estimate the extent of influence of the disturbance by the controller.

The model made in this thesis uses the time unit in seconds, but in real plants, the time used is in minutes or hours, hence it is proportionally faster than what is seen in the chemical or process industry, but the results are the same. The model could have been recalibrated according to the need and test situation. The model made for simulations will work efficiently even if simulated for the time unit as hours. It can be said that the simulation model explained here is a system made for extremely small reactors or micro reactors, as seen in a pharmaceutical company.

The study makes it very clear that closed loop data of any model is not sufficient to show all the information of any process model. A general solution cannot be described for closed loop identification, and one should try to avoid closed loop identification. The best possible method would be to get open loop data and then identify the model.

References

[1] ‘Subspace identification method’, *Wikipedia*. Jan. 15, 2021. Accessed: May 03, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Subspace_identification_method&oldid=1000497204

[2] ‘A brief history of feedback control - Chapter 1’. https://lewisgroup.uta.edu/history.htm (accessed May 19, 2021).

[3] ‘Control theory’, *Wikipedia*. Apr. 23, 2021. Accessed: May 19, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Control_theory&oldid=1019410890

[4] C. Huang, T. Zhang, X. Dang, and H. Jiang, ‘Model identification of typical thermal process in thermal power plant based on PSO-CS fusion algorithm’, in *2018 Chinese Control And Decision Conference (CCDC),* Shenyang, China, Jun. 2018, pp. 3847–3852. doi: 10.1109/CCDC.2018.8407791.

[5] ‘Process model identification in a process control system_US7444191.pdf’.

[6] H. Garnier and P. Young, ‘Time-domain approaches to continuous-time model identification of dynamical systems from sampled data’, in *Proceedings of the 2004 American Control Conference*, Boston, MA, USA, 2004, pp. 667–672 vol.1. doi: 10.23919/ACC.2004.1383680.

[7] A. A. Abdullin, A. G. Mamatov, and K. S. Gorshkov, ‘Discrete-time state feedback control design for linear objects with delay’, in *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus),* Moscow, Jan. 2018, pp. 851–856. doi: 10.1109/EIConRus.2018.8317222.

[8] M. Gilson, H. Garnier, P. Young, and P. Van den Hof, ‘A REFINED IV METHOD FOR CLOSED-LOOP SYSTEM IDENTIFICATION’, *IFAC Proc. Vol.*, vol. 39, no. 1, pp. 903–908, 2006, doi: 10.3182/20060329-3-AU-2901.00143.

[9] Electrical4U, ‘What is a Control System? (Open Loop & Closed Loop Control Systems Explained) | Electrical4U’, *https://www.electrical4u.com/*. https://www.electrical4u.com/control-system-closed-loop-open-loop-control-system/ (accessed May 19, 2021).

[10] F. A. Haugen, ‘Automatic Control’, p. 870.

[11] ‘Book: Chemical Process Dynamics and Controls (Woolf)’, *Engineering LibreTexts,* May 19, 2020. https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Book%3A_Chemical_Process_Dynamics_and_Controls_(Woolf) (accessed May 17, 2021).

[12] Tutorials Point (India) Ltd., *Mathematical Model of Control System,* (Jan. 19, 2018). Accessed: May 19, 2021. [Online Video]. Available: https://www.youtube.com/watch?v=ZAoIhtZsRX8

[13] Jun 12 and 2008, ‘First-Principle Versus Data-Driven Models’, *Control Global*. https://www.controlglobal.com/articles/2008/200/ (accessed May 19, 2021).

[14] ‘Compiled models vs. first principles models for fault detection and diagnosis’. https://gregstanleyandassociates.com/whitepapers/FaultDiagnosis/Compiled-Models/compiled-models.htm (accessed May 19, 2021).

[15] Tutorials Point (India) Ltd., *State Model*, (Jan. 19, 2018). Accessed: May 19, 2021. [Online Video]. Available: https://www.youtube.com/watch?v=DSvBXXnZv34

[16] ‘State-space representation’, *Wikipedia*. Apr. 08, 2021. Accessed: May 03, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=State-space_representation&oldid=1016595996

[17] ‘main_00.pdf’. Accessed: May 18, 2021. [Online]. Available: https://davidr.no/iia2217/pensum/main_00.pdf

[18] ‘System identification’, *Wikipedia*. Jan. 23, 2021. Accessed: Feb. 08, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=System_identification&oldid=1002219949

[19] ‘Black box’, *Wikipedia*. Mar. 19, 2021. Accessed: May 03, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Black_box&oldid=1012946656

[20] ‘State-space model - MATLAB - MathWorks India’. https://in.mathworks.com/help/control/ref/ss.html (accessed Apr. 12, 2021).

[21] A. Ghafooripour, A. A. Aghakoochak, and H. Kiamehr, *AN OVERVIEW OF SYSTEM IDENTIFICATION METHODS AND APPLICATIONS PART II: THEORY, TYPE OF TESTED STRUCTURES, HISTORY AND PROSPECTIVE OF SYSTEM IDENTIFICATION*. 2000.

[22] Tutorials Point (India) Ltd., *Introduction to State Space Analysis*, (Jan. 19, 2018). Accessed: May 19, 2021. [Online Video]. Available: https://www.youtube.com/watch?v=BgaTRpitlGY

[23] Complexity Explorer, *5.1 What is a Dynamical System?*, (Aug. 27, 2018). Accessed: May 19, 2021. [Online Video]. Available: https://www.youtube.com/watch?v=6bGVtoqtWWw

[24] ‘System dynamics and State-space Representations’. https://mec560sbu.github.io/2016/09/11/Systems_Dynamics/ (accessed May 19, 2021).

[25] D. Di Ruscio, ‘Combined Deterministic and Stochastic System Identification and Realization: DSR - A Subspace Approach Based on Observations’, *Model. Identif. Control Nor. Res. Bull.*, vol. 17, no. 3, pp. 193–230, 1996, doi: 10.4173/mic.1996.3.3.

[26] ‘D’. http://www.davidr.no/d-sr/d-sr_e.html (accessed Mar. 24, 2021).

[27] https://home.usn.no/roshans/mpc/ (Accessed May 19, 2021).

[28] ‘86589_pid-controller-tuning-in-simulink.pdf’. Accessed: May 18, 2021. [Online]. Available: https://in.mathworks.com/content/dam/mathworks/tag-team/Objects/p/86589_pid-controller-tuning-in-simulink.pdf

[29] H. Guidi, ‘Open and Closed-loop Model Identification and Validation’, p. 243.

[30] D. Di Ruscio, ‘Closed and Open Loop Subspace System Identification of the Kalman Filter’, *Model. Identif. Control Nor. Res. Bull.*, vol. 30, no. 2, pp. 71–86, 2009, doi: 10.4173/mic.2009.2.3.

References

Appendices

Appendix A Chemical_process

```
function [dmdt_A, dmdt_B, dmdt_C] =
chemical_process(A_in,B_in,m_A,m_B,m_C)

%Reaction constant
k_r = 1000;

%Calculate fractions of the masses inside the reactor
Total_mass = m_A + m_B + m_C;    %Kg
x_A = m_A / Total_mass;
x_B = m_B / Total_mass;
x_C = m_C / Total_mass;

%Outflow
f_out = A_in + B_in;

%Calculte reaction rate
r_C = k_r * x_A * x_B;

%State equations
dmdt_A = A_in - x_A*f_out - 0.25*r_C;    %Kg/s
dmdt_B = B_in - x_B*f_out - 0.75*r_C;    %Kg/s
dmdt_C = r_C - x_C*f_out;                %Kg/s
```

Appendix B DSR explanation

```
function [a,b,d,e,cf,f,x0,sn]=dsr(y,u,L,g,k,bmet,n)
% DSR  Deterministic and Stochastic system identification
and Realization
%       [A,B,D,E,CF,F,x0]=dsr(Y,U,L)
%       [A,B,D,E,CF,F,x0]=dsr(Y,U,L,g)
%       [A,B,D,E,CF,F,x0]=dsr(Y,U,L,g,J,M,n)
%       PURPOSE:
%       Estimate the system order (n) and the matrices
(A,B,D,E,CF,F)
%       in the following discrete time combined
deterministic and
%       stochastic dynamic model on innovations form
%
%       x_{t+1} = A x_t + B u_t + C e_t,    x_{t=0}=x0,
%       y_t     = D x_t + E u_t + e_t,
%
%       where C     = CF*inv(F),        (Kalman gain, K
=inv(A)*C and C=A*K),
%       Delta = E(e_t e_t^T) = F*F',   (Innovations noise
covariance matrix).
%
%       ON INPUT:
%       Y          - Output time series matrix of size (N x
m)
%                    where N is the number of observations
and
%                    m is the number of output variables.
%       U          - Input time series matrix of size (N x
r)
%                    where r is the number of input
variables.
%       L          - Number of block rows in extended
observability matrix.
%                    Choose L .geq. 1. This means that one
can estimate
%                    system order (n) bounded by, 0 < n
.leq. L*m.
%       OPTIONAL INPUT PARAMETERS:
%       g          - g=1 default, g=0 force E to be the
zero matrix.
%       J          - Past horizon used to define
instruments. Default, J=L.
%       M          - Default M=1. See tutorial.
%       n          - Optional specification of model order,
0 < n .leq. L m.
%       ON OUTPUT:
%       A,B,D,E    - Model system matrices.
%       CF, F      - C=CF*inv(F) is the Kalman filter gain
matrix.
%       F          - Delta = F*F' is the innovation noise
covariance matrix.
%       x0         - Initial values for the state vector,
x_t, i.e. state at t=0.
%
%                                        COPYRIGHT 1996,
1999, DDIR
%                                        License belong
to:
%                                        Product id: 10
000
%---------------------------------------------------------
----------------

% DATE: 9, january 2004. Better method for setting the
system order.
%        1. november 1996
% Notes:
% 1. Choose L as close to the observability index (n-d+1)
as possible
%    L >= n-d+1, n >=d whenever the input is poor with
frequencies
%    where d=rank(y_t) usually equal to m.
%    (however, this is not necessary)
% 2. In case that E=0, then chose parameter g=0.
% 3. k, bmet, n. Optional parameters for advanced use.
%    (k=L default, bmet=1
% 4. Algorithm: Di Ruscio (1996), A Method for ...,
%               In "Computer Aided Time series Modeling",
%               Ed: M. Aoki, Springer Verlag.
% 5. J must satisfy J > 0 in order to define the
instruments.
%---------------------------------------------------------
----------------
```

Appendix C Identification.m

```
save('data_c', 'y_c', 'u_c')

clear all
load('data')
dt = 0.1;

% synchronize u and y...

figure(2)
clf
plot([u_o, y_o])
grid on

% Do identification for normalized/scaled data
 u_bias = mean(u_o);
 y_bias = mean(y_o);
 [A,B,D,E,CF,F,x0] = dsr(y_o-y_bias, u_o-u_bias,1)

% Make step response
t = 0:dt:100;
n = length(t);

% u for step response (scaling)
u = ones(1,n)*3;
steptime = 500;
u(steptime+1:n) = ones(n-steptime,1)*4;
u = u - u_bias;
x = zeros(1,n);
y = zeros(1,n);

for i = 1:n-1
    x(:,i+1) = A*x(:,i) + B*u(:,i);
    y(:,i) = D*x(:,i) + E*u(:,i);
end

% Add bias/scaling to get original values
u = u + u_bias;
y = y + y_bias;
figure(3)
clf
```

```
plot(t,[y', u'])
grid on
```