University College of Southeast Norway

www.usn.no

FMH606 Master's Thesis 2017

M.Sc. Industrial IT and Automation

Closed Loop Transient Response System Identification

Abass Mwadini Abass

Faculty of Technology, Natural sciences and Maritime Sciences
Campus Porsgrunn

Course: FMH606 Master's Thesis, 2017

Title: Closed Loop transient response System Identification.

Number of pages: 132

Keywords: SID, Transient response, Closed-loop, Back calculation, Auto-tuning.

Student: Abass Mwadini Abass

Supervisor: David Di Ruscio

External partner: Christer Dalen

Availability: Open

Approved for archiving: ______________________________

(supervisor signature)

Summary:

Closed-loop Transient step response analysis SID technique is known for its simplicity and effectiveness in accommodating practical constraints and operational restrictions of particular system, and thus giving customized models for the particular process. Different algorithms have therefore been developed within the field the closed-loop step-response transient analysis.

The objective of this thesis is to review the most common transient step response SID algorithms, and analyse the algorithm's performance in model estimation and PID controller auto-tuning.

Nine different transient step-response SID algorithms are reviewed and studied by means of simulating different numerical examples which represent processes with various dynamic characteristics. Among the discussed algorithms, five (YS, JR, JL, CLC and MF) are primarily for FOPDT and the remaining (JS, JSDR, DR and DR1) are for SOPDT. The simulation takes also into account the stochastic framework where the systems are influenced with measurement noise, and weighs up the performance of the algorithms based on accuracy in estimating model parameters, robustness of PID-setting, and strength in withstanding the measurement noise.

The study reveals that JL has the strongest performance among FOPDT algorithms which also provides the most robust PI-controller auto-tuning. For SOPDT algorithms, the study points out the DR1 algorithm as superb and the most robust when not exposed to measurement noise.

Preface

This thesis work marks the end of the two years master program of M.Sc. Industrial IT and Automation (IIA) at University College of Southeast Norway (USN). The thesis's main theme is about System Identification methods for Closed-loop Transient response analysis, and was initiated and supervised by associate professor in process control Dr. David Di Ruscio, and co-supervised by Christer Dalen.

The execution of the thesis has been tough and challenging, but thanks to the support and advice from my supervisors, without them this thesis would not be possible to materialize. I would therefore dedicate special thanks to my main supervisor Dir. David Di Ruscio for the fundamental background knowledge I acquired through his Control Theory and System Identification (SID) courses, and for the tremendous support during the entire period of the execution. Similarly, I would like to forward my sincere thanks and appreciation to my co-supervisor for his time, his huge contribution, clarifications, and ideas ever since the beginning of this thesis.

Secondly, I would like to thank all from the Faculty of Technology, Natural Science and Marine Science that has been there for me, encouraged me and ensured that my two years at USN has been refreshing, interesting and successful. Your contribution can never be overrated.

Finally, to my parents, my wife, son and my family in general. Your support, encouragement and faith in me has always been my driving force and source of my motivation. I admire and appreciate you all.

Porsgrunn, 15th May 2017

Abass Mwadini Abass

Abbreviation

CLC	Cheng-Liang Chen
DR	Dalen & Di Ruscio method
DR1	Dalen & Di Ruscio method 1
ESSM	Extended State Space Model
FOPDT	First-Order process Plus Delay Time
GM	Gain Margin [dB]
HF, HR	Half-rule model reduction technique
JL	Jietae Lee method
JR	Jutan & Rodriguez method
JS	Jahanshahi & Skogestad method
JSDR	Modified JS method – by DR
LTIFD	Linear Time Invariant system of Finite Order
MF	Mamat & Fleming method
P	Proportional - controller
PI	Proportional Integral - controller
PID	Proportional Integral and Derivative - controller
PM	Phase Margin [°]
SID	System Identification
SNR	Signal to Noise Ratio
SOPDT	Second-Order process Plus Delay Time
SSM	State space model
YS	Yuwana & Seborg method

Nomenclature

A	Magnitude of set point change
$h_p(s)$	Process transfer function in Laplace domain
$h_c(s)$	Controller's transfer function in Laplace domain
K	Open loop gain $K = K_c K_p$
K_c	Controller's gain for step test
K_p	Process static gain (steady state gain)
K_{pi}, K_{pid}	PI-controller and PID-controller gain respectively
R_0, R	Set point initial and final value respectively
s	Laplace transformation variable
t	Time [s]
T	System's Time constant [s]
T_i	Integral time [s]
T_d	Derivative time [s]
t_{p1}	Time taken for the closed-loop response to reach the first peak [s]
t_{p2}	Time taken for the closed-loop response to reach the second peak [s]
t_u	Time taken for the closed-loop response to reach first lowest point [s]
y_{m1}	Close-loop response value at t_u
y_{p1}	Close-loop response value at t_{p1}
y_{p2}	Close-loop response value at t_{p2}
y_∞	Steady state response of the closed-loop
Δt	Half period of the oscillation for closed-loop step response [s]
τ_p	Identified process time delay [s]
$\top_p$	Identified process time constant [s]
ζ	Damping ratio of the closed-loop response

Contents

1 Introduction

Open-loop transient step response SID provides, in some cases, sufficient result. However, the approach is reported to perform poorly or even impossible to execute when the open-loop process is unstable [1]. The fact that most industrial systems works under feedback control favours and calls for closed-loop transient step response SID. With the closed-loop SID, the practical constraints and operational restrictions of the particular system or process can be accommodated during identification experiment, which ensures safety and less time consumption [1].

In the closed-loop step response analysis, the experimentation and data collection is performed on-line in an output feedback system, whereby small step change in controller's set point $r(t)$ is triggered to produce response $y(t)$ from the closed-system [2]. The closed-loop response is thereafter used in fitting a chosen model structure of the system by means of back calculation [3]. Different methods and algorithm have therefore been developed to back calculate and fit the estimated models.

The main objective of this thesis is to review the most common transient step response SID algorithms, as well as analyse the algorithm's performance in model estimation and PID controller auto-tuning by means of simulations of numerical example using MATLAB software. The thesis report consist of mainly five chapters ranging from Introduction, System Identification theory, Closed-loop Transient step-change response SID, Discussion to Conclusion.

The Introduction chapter introduces the research topic in its domain, and familiarizes the reader with background information that has given rise to this thesis work. The chapter goes on further to highlight the objectives and clarify the scope of the thesis, and provides orientation of previous works done in the field of Closed-loop transient response system identification. Chapter 2 builds up theoretical understanding and provides the reader with the necessary background information and knowledge of main concepts within system identification and the techniques associated with it, and thereafter channels the reader's attention into the main subject matter of the thesis.

Chapter 3 presents practical implementation and results of the individual tasks in sequential order arranged according to their subtopic, as highlighted in section 1.2. In this chapter, nine common transient step response SID methods and algorithms are discussed, analysed and tested by means of simulation using different numerical examples that represent systems with various response dynamics. The chapter, which also serve as the main chapter, gives the reader an insight into on-line PID-controller tuning based on identified parameters from the discussed algorithms, and sums the overall performance of algorithms by using Monte Carlo simulation technique.

The Discussion chapter weighs up the result against previous work and literature, and provides the reader with an objective interpretation, explanation and analysis of the results obtained in chapter 3. Finally, the Conclusion chapter provides summary of the findings and some suggestions based on findings.

1.1 Previous works

The closed loop transient response system identification is a relative old field with many proposed, proven and documented methods and algorithms started back in early 80s. YS was the most pioneering work that paved the way for further development and improvement of the technique. The YS utilized the critical points observed in closed-loop response of feedback system controlled by means of P-controller [2], to back-calculate the open-loop model of first-order plus time delay. YS used first-order Padé approximation of delay term, the move that gave rise to JR method. The JR method extended the Padé approximation of the delay term into second-order to improve the algorithms ability in dealing with systems containing large dead time [4].

In 1989, a new method JL was developed by matching dominant poles of the feedback system with the typical second-order process response, and hence moved away from utilizing Padé approximation [5]. In the same year, another method called CLC was developed based on critical magnitude ratio and crossover frequency of the system and its relationship to the typical second-order response transfer function [6, 7]. The CLC was later updated to counteract the steady state offset by utilizing the commonly used industrial controller (PI-controller), hence leading to a new method MF [8]. Several other updates and new methods for FOPDT have been developed since then.

In recent decades, identification of second-order process models has really gained momentum. Plenty of proposals and later proven algorithms have been suggested and documented, including estimation of unstable second-order model from unstable higher-order process, a method called JS [9], which was modified into JSDR, and later updated generalized to form a method known as DR which incorporate both unstable and stable second-order process model estimation regardless of the original systems dynamics behaviour. By updating and improving the DR’s systems response in time domain, a new method called DR1 initiated [10].

In 2010, a new method algorithm specializing in estimation of second-order inverse response process from transient response was presented. The algorithm provides simple identification procedures by avoiding solving nonlinear equations, and estimates, instead, model parameters in sequential way [11].

The closed-loop transient step response is an interesting and still maturing field. Several previous works, other than mentioned above, have been carried out reported from different holds. Among the main objective of this thesis is to review some most common algorithms for both FOPDT and SOPDT.

1.2 Scope and Limitation of the project

The main objective of this thesis is to review the most common system identification methods and algorithms developed for closed-loop step response transient analysis, and investigate the performance of such algorithms in model estimation of first-order and as well as second-order plus time delay process models. Having that in mind, the scope is further broken down into the following tasks reflecting the thesis description given in Appendix A.

- Conducting literature review of most common step response transient analysis SID methods.

- Discussing and illustrating how the first and second-order models are identified based on the back calculation techniques using the methods reviewed in the point above.
- Investigating and illustrating estimation of time delay of systems with different dynamics.
- Demonstrating and discussing how the methods or algorithms can be applied in designing and on-line auto tuning of PID controller.
- Performing simulations and detailed comparison of the algorithms in stochastic framework both when and when not influenced measurement noise.
- Applying and demonstrating the performance of the algorithms on the quadruple tank laboratory process.

Unfortunately, the expected quadruple tank laboratory process has not been available during the entire execution period of the thesis due to delayed production of the process, neither any other process experimental data were provided as an alternative. Therefore, this thesis is limited to simulation of only numerical examples collected from some of previous works.

2 System Identification Theory

Many systems and processes are complicated by nature. Industries strive in controlling the systems and process within defined operating conditions aiming for safe, effective and beneficial operation. To tune and maintain the systems into the desired conditions requires good understanding of system's dynamics and properties. This is where the system identification comes into place, a diverse field of techniques to reveal the system's behaviour as well as better understanding of cause-effect relationships of particular system [12] [13].

This chapter builds up theoretical understanding and familiarize the reader with the main concepts of system identification and techniques associated with the subject matter. The chapter provides the necessary background information to be used in later chapters.

2.1 Introduction to System Identification

System identification can be referred as technique used in determination of mathematical models of dynamic systems based on observed set of experimental data [1] [3]. The identified systems are normally in specified form (model structure), with free parameters adjusted based on defined prediction criterion, which takes into account the difference between the measured output and the output from resulted model [13] [14].

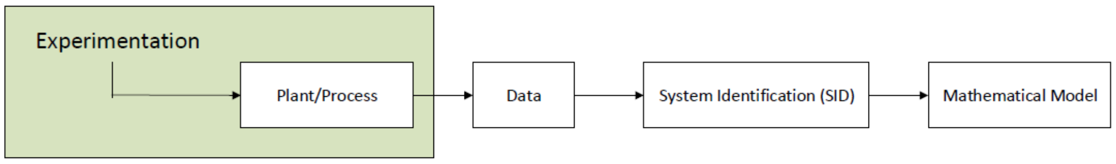


Figure 2-1: Overview of System Identification (SID) approach

Figure 2-1 gives an overview of the system identification technique whereby an experiment is conducted to observe and collect data necessary for estimation of the system dynamics by fitting into mathematical model of a pre-specified model structure [15].

Generally, the model should be compact and adequate to best reflect the system in question by including all system dynamics, and hence serves the purpose it is intended for. In system and control engineering, such purpose may include simulation, design of system's controller, prediction algorithms, etc. [15] [1]. However, applications of system identification are widespread, and have even gone beyond the traditional technical fields, as for example in biology and environmental sciences where the identified models enhance better scientific knowledge and trends in objects [15] [1].

2.2 Systems and Models of dynamic systems

2.2.1 Dynamic System

System is an object or collection of objects with known boundaries, containing different variables and the properties of the interest [16, 17]. The variables interacts with each other and/or with the surrounding environment to produce observable signals. Here, the surrounding environment is joint term for the external (outside system boundary) factors affecting the system, which includes both user's manipulated signal (input $u(t)$), and non-manipulated signal known as disturbance ($w(t)$ and $v(t)$) [3, 16]. Normally, the observable signals (output $y(t)$) define the behaviour in which the system is modelled upon it [3, 16, 17].

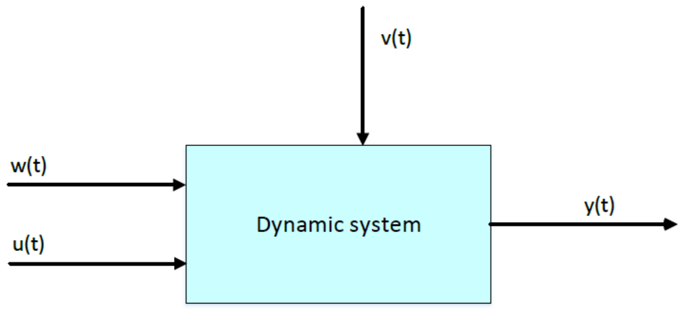


Figure 2-2: Typical dynamic system with inputs, disturbances and output

Figure 2-2 gives graphical presentation of typical dynamic system consisting of input signals $u(t)$, measurable disturbances $w(t)$, non-measurable disturbances $v(t)$, and output signal$y(t)$ [3].

A system is considered as dynamic if its variable(s) varies with respect to time [17]. The variations may be due to external excitement of the system through inputs and disturbances, or internally due to change in system operating conditions [17, 18]. However all systems are dynamic as they normally experience movement and exhibit inertia against changes [16, 17].

2.2.2 Models of dynamic system

A model is a representation of the system including inputs and outputs, which reflect the essential behaviour of the system [16]. Being representation means it is extremely rare to find model with exactly similar dynamics as its original system since it is impossible to include all factors influencing the system. However, with models, experiments or simulations on various operational conditions can be conducted, and hence provides valuable platform in studying the dynamics of the original system as well as designing, testing and tuning controller for the particular systems [1, 16-18].

2.2.2.1 Types of models

In system identification, models are estimated based on data obtained from experimentation on the real system [3]. The term system in this context also includes single or set of processes working together toward common goal [16].

Different classification of models are available, as for example empirical and mechanistic model. Empirical models are typically based on cause-and effect data obtained from direct observation and extensive data record on the particular system. On the other hand, the mechanistic models are based on underlying physical principles, as for example balance laws and/or natural constants [17, 19].

2.2.2.1.1 White-box models

White-box models, also known as analytical models, are exclusively mechanistic models, i.e. the models are result of first principles (physical and chemical laws) thus relationships between variables are clearly known [13]. With the white-box models, simple experiments may be needed to inquire into the principles involved [13, 20]. Among examples of white-box (analytical models) are aircraft flight model, carbon dioxide concentration in a ventilated room, etc. [17].

2.2.2.1.2 Black-box models

Black-box models refers to models exclusively based on experimental data, without any previous knowledge of the system's structure and variables [13, 20, 21]. These type of models are obtained using advanced algorithms, and focus on describing the relationship between the inputs and outputs of the system rather than the actual structure and variables of the system. Among examples of black-box models are neural network models, neuro-fuzzy models, etc. [13, 20, 21].

2.2.2.1.3 Grey-box models

Grey-box models are referred as intermediate case of modelling whereby the model structure is generically known and described using first principles [13, 22]. However, the internal particularities of the system are not entirely known, hence experimental data analysis is needed to estimate the parameters and remaining elements of the system [13, 22].

$$G(s) = k\frac{\mathrm{e}^{-\tau s}}{s^2 + a_1 s + a_0} \tag{2.1}$$

Among good examples of grey-box models are subspace models, transfer functions models, etc. Their model structure are previously known [13, 22]. For example, the typical second-order transfer-function process model as given by Equation (2.1), where k, τ, a_1 and a_2 are parameters to be estimated through experimentation and identification techniques.

Generally, the system identification techniques are used to estimate black-box or grey-box model of dynamic systems [13]. However, in this thesis, the focus will be on identification of grey-box models, with the transfer function as chosen model structure for further study on closed-loop transient response SID techniques.

2.3 Practical steps in System Identification

Implementation of the system identification techniques involves series of sequential and iterative steps, starting from experimentation and data collection, during which the system is excited with input signals to produce output data [1, 3]. The data are then fitted into the chosen model structure, and the model's unknown parameters are estimated using identification algorithm suitable for that particular model. The final step involves testing and validation of the resulted model against the real system [1, 3].

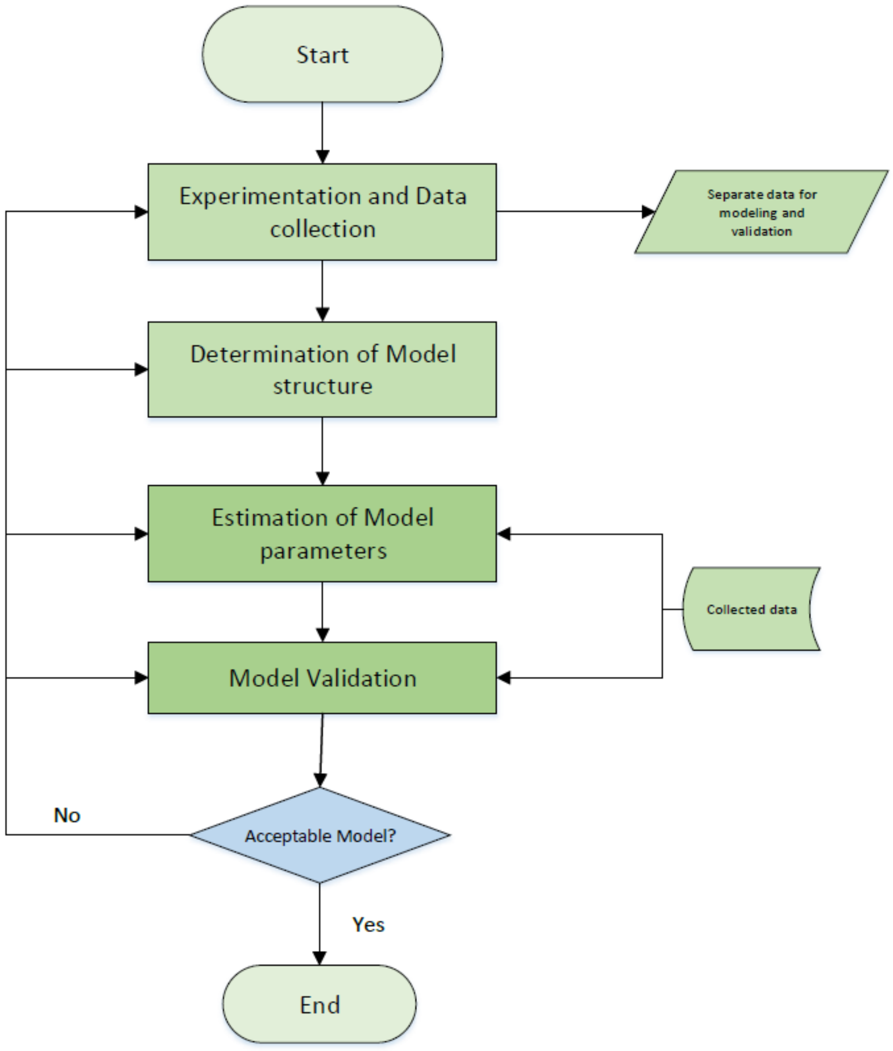

Start
Experimentation and Data collection
Separate data for modeling and validation
Determination of Model structure
Estimation of Model parameters
Collected data
Model Validation
No
Acceptable Model?
Yes
End

Figure 2-3: Practical procedure of system identification

Figure 2-3 gives graphical overview of the sequential steps used in system identification [1]. As seen on the figure, the practical implementation is rather iterative and may require several repetitions of different steps before the estimated model become acceptable [1]. The individual steps are further discussed in sub-sections below.

2.3.1 Experimentation and Data collection

Experimentation and data collection is among the most delicate stages in SID. Inputs signals are manipulated to trigger response from the system. The signals should cover wide enough range and carefully chosen to ensure maximum information [16]. It is natural to believe that prior knowledge of the system leads to better design and planning of experiment [3, 16, 17]. Among examples of inputs signals are step change, sinusoid, impulse signal, random signal, etc. This thesis will focus on step change in controller's reference value as the chosen input signal.

2.3.1.1 Experimental Design

Conducting experiments is not only time-consuming, but often involves use of resources and material that may be expensive in nature. A thoroughly plan and experimental design is essential to ensure effective and successful experiments [16, 17]. The minimum requirement in designing experiment for system identification include a prior decision in signals and variables of interest, timeframe of measurement, sampling time, number of observations, signals to manipulate and how the manipulation shall be done [1].

2.3.1.2 Pre-treatment of data

Pre-treatment of the data collected from experimentation is inevitable, as the raw data are less likely ready for immediate application in SID algorithms. There are several challenges the collected data may encounter during experimentation. These includes, but not limited to; [1]

- High-frequency noisy signals
- Presence of outliers and data missing
- Drift and offset, and low-frequency noisy signals

To minimize the effect the deficiencies, some special measures need to be employed, including the use of low-pass filter of suitable filter constant to withhold signals with frequency higher than the target signal. Additionally, it is recommended to let the noise model cancel out the effect of low-frequency disturbances [1].

2.3.2 Determination of model structure

Next to pre-treatment of experimental data is determination of model structure to be utilized for system identification. This is crucial stage, as the structure of chosen model influences the quality of final identification result. Choosing suitable model structure requires good understanding of identification techniques and adequate knowledge of the system [1]. When determining the model structure, one should first outline the model type, for example non-parametric model of transfer-function type, and then decide the size of the model for example transfer-function model of second-order process plus delay, and finally choose model parameterization [1].

It is strongly recommended that the model structure should cover as many relevant linear and non-linear phenomenon as possible [20]. However, being too flexible could minimize the ability of the model to accommodate the physical insight of the particular system [20].

2.3.3 Model fitting and parameter estimation

Fitting and parameterizing the model structure chosen from the above stage, involves feeding of estimation data (part of collected experimental data) into suitable identification algorithm based on the identification method and the chosen model structure. The goal is to obtain well conditioned parameterization that accomplish given quality criteria, which limit numerical error influence on input-output behaviour of the system [3, 13, 20].

In closed-loop transient response SID, the change in controller's set-point acts as input part of the estimation data, while the system response retains the output role. Often, the estimation data is structured as in Equation (2.2) before being imported to the closed-loop SID algorithm [10].

$$R = \begin{bmatrix} r_1^T \\ \vdots \\ r_N^T \end{bmatrix}, and\ \ Y = \begin{bmatrix} y_1^T \\ \vdots \\ y_N^T \end{bmatrix} \tag{2.2}$$

2.3.4 Model Validation

Models estimated from SID must undergo qualitative check known as validation, a practical process of assessing models performance. The assessment is conducted by feeding validation dataset (separate dataset, other than training dataset used for model estimation) into the model and compare the model results against certain performance criteria, for example general prediction error, spectra analysis, etc. [13, 20, 22].

Once the model passes the performance criteria, the model attain generalization status. Generalization is a qualitative measure of model's ability in providing accurate predictions and explanation of systems output from inputs different to those used during model development [13, 22].

2.4 System Identification Approaches

Different system identification methods have been presented and documented in diverse literature. Based on their identified model structure, the methods are grouped into two categories, known as parametric and nonparametric system identification approach [1, 13, 15]. The two approaches works independently of each other, however often combined to exploit the strength of both simultaneously. For example, starting directly on parametric models may appear complicated and gives mislead results, thus using nonparametric approach (for example impulse response) helps in relieving specific dynamic of the system and hence assist in detecting the mislead results [1, 13, 15].

2.4.1 Parametric methods

In parametric system identification, the models rely on a parameter vector [1, 13]. The parameters within the vector are calculated and varied in respect to the defined prediction criteria to produce models. Among common examples of parametric system identification methods are least-square method, prediction error method, instrument-variable method, etc. [1, 13, 23].

2.4.1.1 Least-Square Method

The least-square method is capable of estimating models of predefined structure for both static and dynamic systems [24]. The method is based on standard regression model that is given as,

$$y = [\varphi_1 \quad \cdots \quad \varphi_n] \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}, \text{ which can be written as } \quad y = \varphi\theta \tag{2.3}$$

where φ is known as regression vector containing regression variables, y is the observed variable with known value, and θ denotes the unknown parameter vector to be estimated [24]. Consider m number of observed values, the corresponding extended equation becomes,

$$\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} \varphi_{11} & \cdots & \varphi_{1n} \\ \vdots & \ddots & \vdots \\ \varphi_{m1} & \cdots & \varphi_{mn} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}, \text{ which can be written as } \; Y = \Phi\theta \tag{2.4}$$

Equation (2.4) is used to estimate the vector θ containing values of the unknown parameters [24].

2.4.1.2 Prediction-Error Method (PEM)

The prediction-error method is believed to exploit the system’s dynamic structure in best possible way. Its focus is rather shifted to accuracy of the predictions calculated from observations [1, 25].

$$y(t) = G_0(z)u(t) + v(t) \tag{2.5}$$

Consider an unknown system G_0 described by LTIFD as given in Equation (2.5) [15]. The system consist of measured input $u(t)$ and output $y(t)$ signals, in addition to unmeasurable disturbance $v(t)$. Assume that $v(t)$ is power spectrum signal expressed as in Equation (2.6), where $e(t)$ denotes white noise with variance σ_e^2 and that $H_0(z)$ is stable, monic and minimum phase transfer function [15].

$$v(t) = H_0(z)e(t) \tag{2.6}$$

The system model is then estimated using Equation (2.7).

$$y(t) = G_0(z)u(t) + H_0(z)e(t) \tag{2.7}$$

However, the expression is only valid if the disturbance can be generated from the white noise, implying that $u(t)$ and $v(t)$ cannot be correlated. This underlines the importance of verifying the compatibility of the predefined model structure with PEM assumptions [15].

2.4.1.3 Instrument-Variable Method

The Least-square method is easy and simple to implement. However, the method's ability in producing consistent parameters is limited to only certain condition due to correlation between the regression vector and observed data vector [1, 26]. Counteracting this deficiency has given rise to Instrument-Variable method, by introducing a general correlation $Z(t)$ containing elements known as instruments. The instrument matrix replaces the regression vector from the LS as seen in Equation (2.3) [1].

$$\hat{\theta} = [\sum_{t=1}^{N} Z(t)\Phi^T(t)]^{-1}[\sum_{t=1}^{N} Z(t)\mathrm{y}(t)] \tag{2.8}$$

2.4.1.4 Subspace Identification

Subspace system identification adds to the techniques used to estimate parametric models with limited number of mathematic parameters, for example state space model realization [15]. The method's basic step is to identify the order of the observed system and observability matrix through combining the knowledge of impulse response, and computation of rank and decomposition of matrix often by singular value decomposition [15, 27].

$$x_{k+1} = Ax_k + Bu_k + Cv_k \tag{2.9}$$

$$y_k = Dx_k + Eu_k + Fv_k \tag{2.10}$$

Consider the state-space model (SSM) given in Equation (2.9) and (2.10). Using the concept of observability and triangular Toeplitz matrix, the SSM can be extended into

$$y_{k+1|L} = \tilde{A}_{Ly_{k}|L} + \tilde{B}_{Lu_{k}|L+g} + \tilde{C}_{Lv_{k}|L+1} \tag{2.11}$$

Where the matrices in Equation (2.11) are given as,

$$\tilde{A}_L \stackrel{\text{def}}{=} O_L A (O_L^T O_L)^{-1} O_L^T \tag{2.12}$$

$$\tilde{B}_L \stackrel{\text{def}}{=} [O_L B \quad H_L^d] - \tilde{A}_L [H_L^d \quad 0_{Lm\times r}] \tag{2.13}$$

$$\tilde{C}_L \stackrel{\text{def}}{=} [O_L C \quad H_L^s] - \tilde{A}_L [H_L^s \quad 0_{Lm\times l}] \tag{2.14}$$

The model in Equation (2.11) can be identified directly from the sliding window of known data, unless when the inputs is affected in persistent excitation. In such situation, one rather should consider to identify only the minimal order of Equation (2.11) [14].

2.4.2 Non-parametric methods

Nonparametric system identification methods are generally simple and effective technique in estimating models of systems. Common for nonparametric methods is that the techniques yield models in form of curves or function, which contains characteristic behaviour of the system. The term nonparametric reflects the mathematical nature and the absence of physical parameters in models obtained from the approach. However, the models may still consist of mathematical parameter(s) to be identified [1, 13, 15].

Nonparametric models exist in time-domain, for example models from impulse response and step change response, and as well as in frequency domain, for example models from correlation analysis, spectral analysis and frequency analysis [15].

2.4.2.1 Correlation analysis

Correlation analysis method utilizes white noise signal as input. The goal is to estimate the weighting function mapping output against the input using normalized cross-covariance function between them [1, 3].

$$y(t) = \sum_{k=1}^{\infty} h(k)u(t-k) + \mathrm{v(t)} \tag{2.15}$$

Consider the typical model description for correlation analysis as given by Equation (2.15), whereby $h(k)$ is the weighting sequence and $\mathrm{v(t)}$ is disturbance. If the input is quasi-stationary sequence, which is independent of the disturbance, then the covariance function is given as, [1, 3]

$$r_{yu}(\tau) = \sum_{k=0}^{\infty} h(k)r_u(t-\tau) \tag{2.16}$$

Since the input is white noise, the covariance function can be estimated from the data by solving Equation (2.17) and (2.18) [1, 3].

$$\hat{r}_{yu}(\tau) = \frac{1}{N}\sum_{t=1-\min(\tau,0)}^{N-\max(\tau,0)} y(t+\tau)u(t) \qquad \tau = 0, \pm1, \pm2, \ldots \tag{2.17}$$

$$\hat{r}_u(\tau) = \frac{1}{N}\sum_{t=1}^{N-\tau} u(t+\tau)u(t) \qquad \hat{r}_u(-\tau) = \hat{r}_u(\tau) \qquad \tau = 0,1,2,\ldots \tag{2.18}$$

The estimate of weighting matrix $h(k)$ can be obtained by solving Equation (2.19).

$$\hat{r}_{yu}(\tau) = \sum_{k=0}^{\infty} \hat{h}(k)\hat{r}_u(\tau-k) \tag{2.19}$$

Generally, pure white noise simplify the solution for the weighting matrix [1, 3].

2.4.2.2 Spectral analysis

A frequency domain analysis in which the frequency response of system is derived from cross-spectrum of the impulse response signal. The derived frequency response contain information about gain and phase of the system from inputs of different frequencies [1, 13].

Consider the model description in Equation (2.15), applying discrete Fourier transformation on the model yields to spectral densities as given in Equation (2.20).

$$\emptyset_{yu}(\omega) = H(e^{-i\omega})\emptyset_u(\omega) \tag{2.20}$$

Where,

$$\emptyset_{yu}(\omega) = \frac{1}{2\pi}\sum_{t=-\infty}^{\infty} r_{yu}(\tau)e^{-i\tau\omega} \tag{2.21}$$

$$\emptyset_u(\omega) = \frac{1}{2\pi}\sum_{t=-\infty}^{\infty} r_u(\tau)e^{-i\tau\omega} \tag{2.22}$$

$$H\left(e^{-i\omega}\right) = \sum_{k=0}^{\infty} h(k)e^{-ik\omega} \tag{2.23}$$

The transfer function $H\left(e^{-i\omega}\right)$ can be estimated using the Equation (2.23). However, it is worth to mention that $\emptyset_{yu}(\omega)$ yields complex solutions [1, 13].

2.4.2.3 Frequency analysis

The key input in frequency-analysis system identification is sinusoidal signal, in which change in amplitude and phase creates base for the system response. Here, the frequency of the input signal and its corresponding system response is identical [1]. Consider a sinusoidal input to the system, given as

$$u(t) = \mathrm{a}\sin(\omega t) \tag{2.24}$$

If the system is asymptotically stable, its output at the steady state is given as in Equation (2.25).

$$y(t) = bsin(\omega t + \theta) \tag{2.25}$$

Where b is the response amplitude and θ is phase shift given as in Equation (2.26).

$$b = a|G(i\omega)| \text{ and } \theta = arg[G(i\omega)] \tag{2.26}$$

In practice, it less likely to achieve accurate estimate of the parameters b and θ due to presence of noise signal during experimentation [1].

2.4.2.4 Transient response analysis

This nonparametric identification technique uses transient response of a system excited with input signal of either step or impulse form. The choice of input signal depends on the nature of the process, for example, the impulse signal is commonly used in processes involving injection of substance as form of input [1].

The techniques is quite flexible as it can accommodate both direct, indirect and joint input-output identification. By direct identification means no feedback involved, only input $u(t)$ and output $y(t)$ data are used for model fitting. Indirect identification is performed in a feedback loop, and it requires knowledge of the loop's controller (controller transfer function), controller's set point $r(t)$, and system's response $y(t)$. The third approach is also applied in closed-loop and requires input signal $u(t)$ in addition to the same information needed for indirect identification [1, 10].

2.4.2.4.1 Impulse response

Consider an approximated impulse signal given in Equation (2.27) as input to the system, where α denotes the impulse length. It is assumed that the signal resembles ideal input and satisfies $\int u(t)dt$ [1].

$$u(t) = \begin{cases} -1/\alpha, & 0 \leq t < \alpha \\ 0, & \alpha \leq t \end{cases} \tag{2.27}$$

The impulse input approximation (Equation (2.27)) works just fine if the magnitude of α is less compared to the time constant of system in question. With larger α, the signal becomes distorted [1]. When the input signal $u(t)$ is triggered into the system, the system responds by producing output $y(t)$ as given in Equation (2.28), [15]

$$y(t) = \alpha g_0(t) + v(t) \tag{2.28}$$

whereby the system impulse can be estimated for all values of $\top \geq 0$ (i.e. for high SNR) as

$$\hat{g}(\top) = \frac{y(\top)}{\alpha} \tag{2.29}$$

The impulse response method is adored for its simplicity, however, systems tend to exhibit nonlinear dynamics when subjected to inputs with high amplitude.

2.4.2.4.2 Step response

The step-response technique uses step input to trigger transient response from system, which contains characteristic dynamics (time delay, dominating time constant, static gain, damping ratio, etc.) of the process. These dynamics are essential when it comes to designing and tuning controller for the system [1, 3]. Consider a system is excited with step input signal given by Equation (2.30).

$$u(t) = \begin{cases} \beta, & t \geq 0 \\ 0, & t < 0 \end{cases} \tag{2.30}$$

The system's response to the input is as given in Equation (2.31), where β is magnitude of the step input provided to the system [3].

$$y(t) = \beta \sum_{k=1}^{t} g_0(k) + v(t) \tag{2.31}$$

It follows that the system estimate $g_0(k)$ can be identified for all values of $\top \geq 0$ as, [3]

$$\hat{g}(\top) = \frac{y(\top) - y(\top - 1)}{\beta} \tag{2.32}$$

The step response transient analysis method is subjected to relatively large error term, making it not well suited for identification of definite impulse response [3].

2.4.2.4.2.1 Open-Loop Transient Response SID

Open-loop transient identification, also known as direct identification utilizes the step input $u(t)$ and the observed response $y(t)$ to fit the system into simple (first and second) order models described in transfer function.

First order open-loop Transient response

Consider the Laplace transformed expression for the output signal $y(t)$ given as,

$$Y(s) = G(s)U(s) \tag{2.33}$$

where $U(s)$ is the step input signal in Laplace domain [18]. The system model *G(s)* is thereafter estimated using the standard transfer function for first order process given as [18]

$$G(s) = k\frac{e^{-\tau s}}{1 + \top s} \tag{2.34}$$

where k is the static gain of the system, $\top$ is the time constant (i.e. time taken for the output *y(t)* to reach 63% of the maximum system response) of the system, and the τ denotes time delay (dead-time) of the system [18]. The parameters in Equation (2.34) can be deduced directly from the system response as illustrated in Figure 2-4.

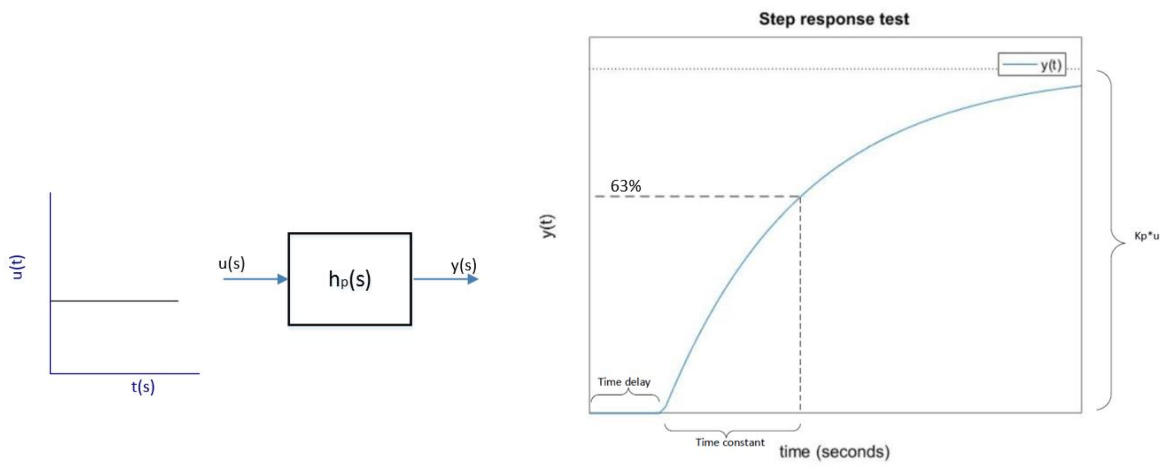


Figure 2-4: Open-Loop step response technique for identification of first order system

Open-loop Transient response for oscillating processes

For the second order process, the system is identified using the typical transfer function for damped system given as, [1]

$$G(s) = k\frac{{\omega_0}^2 e^{-\tau s}}{s^2 + 2\zeta\omega_0 s + {\omega_0}^2} \tag{2.35}$$

where the parameters k and τ are identical to those in Equation (2.34) discussed above, ω_0 denotes the natural frequency of the system response, while ζ represents the relative damping which characterizes the response of the system. The systems experiences oscillations at $0<\zeta<1$, becomes critically damped at $\zeta = 1$, and is said to be undamped when $\zeta < 0$. [1]

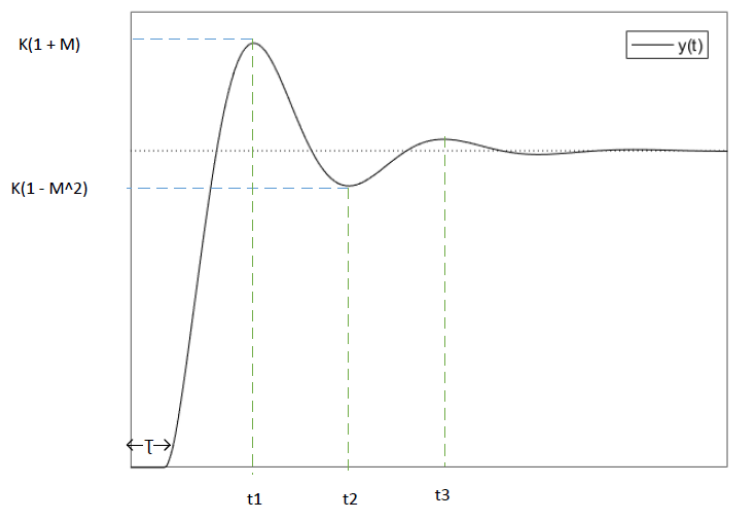


Figure 2-5: Open-Loop step response for oscillating systems [1]

The model parameters are obtained by solving Equation (2.36) and (2.37) based on critical points in the system step response as shown in Figure 2-5.

$$\zeta = \frac{-logM}{\sqrt{\pi^2 + (logM)^2}} \tag{2.36}$$

$$\omega_0 = \frac{2\pi}{T\sqrt{1-\zeta^2}} \tag{2.37}$$

The variables K and M can be deduced from the expression presented on the response graph, while T is the periodic time of the oscillation (t_3 - t_1) [1].

2.4.2.4.2.2 Step change Closed-loop Transient Response SID

In closed-loop step response analysis, the experimentation and data collection is performed on-line in an output feedback system, whereby small step change in controller's set point $r(t)$ is triggered to produce response $y(t)$ from the closed-system [2]. This closed-loop response is thereafter used in fitting transfer function model of the system [3].

The practice requires knowledge of the controller forming the feedback system, in addition to $r(t)$ and $y(t)$ [10]. P-controller is the mostly used controller type in closed-loop step-response analysis, however, different algorithms have been developed based on PI-controller not only because of its status as commonly used industrial controller, but also as a measure in counteracting the steady state offset associated with the P-controller [8]. The step-change closed-loop transient response system identification is the main theme of this thesis. From now onward, the attention will be solely tuned into this technique of system identification.

2.5 Persistent excitation

Quality of the system data collected during experimentation has huge influence in obtaining distinctive models with best possible reflection of the original system. The data should be sufficiently rich and should cover whole frequency range within the region of the interest (i.e. where the model need to be accurate), a phenomenon known as persistent excitation [28, 29].

For the input signal to be persistently exciting of order n, the following two conditions should be met [1].

1. The limit for $r_u(\tau)$ given by Equation (2.38) exists
2. The matrix $R_u(n)$ given by Equation (2.39) is positive definite.

$$r_u(\tau) = \lim_{N\to\infty}\frac{1}{N}\sum_{t=1}^{N} u(t+\tau)u^T(t) \tag{2.38}$$

$$R_u(n) = \begin{bmatrix} r_u(0) & r_u(1) & \cdots & r_u(n-1) \\ r_u(-1) & r_u(0) & & \vdots \\ \vdots & & & \\ r_u(1-n) & \cdots & & r_u(0) \end{bmatrix} \tag{2.39}$$

Normally, inputs composed of only set of frequencies are applied under experimentation, resulting into limited ability of SID techniques as only models of certain orders can be distinguished [28]. Usually, persistent excitation in open loop SID guarantees distinctive models with respect to structure of the preferred model. It is however, not the case when closed-loop SID is concerned. In some cases, the closed-loop SID fails to distinguish unlike models of a system in feedback loop by only using input-output data collected during experimentation, as illustrated below [29].

Consider a system given by,

$$x(t+1) = ax(t) + bu(t) + v(t) \tag{2.40}$$

where $u(t)$ is input signal to the system, and $v(t)$ represent random white noise acting on the feedback system. Assuming that the system is controlled by P-controller with proportional gain K_c, gives the input signal $u(t)$ as,

$$u(t) = K_c x(t) \tag{2.41}$$

Applying the feedback rule, and substituting Equation (2.41) into Equation (2.40) gives,

$$x(t+1) = (a + bK_c)x(t) + v(t) \tag{2.42}$$

hence proving that all models $\langle \hat{m}, \hat{n} \rangle$ having following model structure

$$\hat{m} = m - \alpha k \text{ and } \hat{n} = n + \beta \tag{2.43}$$

of the system (Equation (2.40)) gives identical input-output behaviour of the system and hence cannot be distinguished by just using the observed (input-output) data [29].

2.6 Monte Carlo Simulation

Monte Carlo simulation is a wide known technique used in assessing performance of algorithms especially when achieving exact computed result seems impossible [30]. The technique can be performed in many ways. It follows, however a fixed pattern in the following steps; [30]

- Defining input domain
- Generating random inputs within the input domain
- Calculating algorithm's results by deterministic computation using the inputs
- Aggregating individual result into final results
- Validating the result against predefined performance criteria, if present.

This simulation technique is applied in comparing the performance and robustness of the chosen algorithms for set-point change transient response of closed-loop system later in chapter 3.4.

$$P = \frac{1}{m-1}\sum_{k=1}^{m} n(\hat{\theta}_k - \theta_0)(\hat{\theta}_k - \theta_0)^T \tag{2.44}$$

The algorithms performance is measured against the performance criterion given in Equation (2.44), where $\hat{\theta}$ denotes the identified parameters, θ_0 stands for the real parameters, while *n* and *m* represent number of samples and simulation runs respectively [1].

3 Closed-Loop Transient Step-change Response SID

A closed-loop step-change change transient response SID can be described as an on-line identification approach whereby an output feedback system is excited with simple step change in controller's reference value (set point) to trigger response from the closed system. The approach, also known as indirect identification, is normally performed in the following steps [2, 3, 31];

1. Switch controller in P-mode, and set suitable controller gain K_c to produce underdamped response. PI-mode is used for the MF algorithm, hence suitable integral time T_i shall also be provided.
2. A closed-loop is identified using set point $r(s)$, controller $h_c(s)$, system output $y(s)$, and critical points on $y_{p1}, y_{m1}, y_{p2}, t_{p1}, t_u, t_{p2}$ and Δt as illustrated in Figure 3-6.
3. Open-loop (process model) parameters are determined using back calculation.

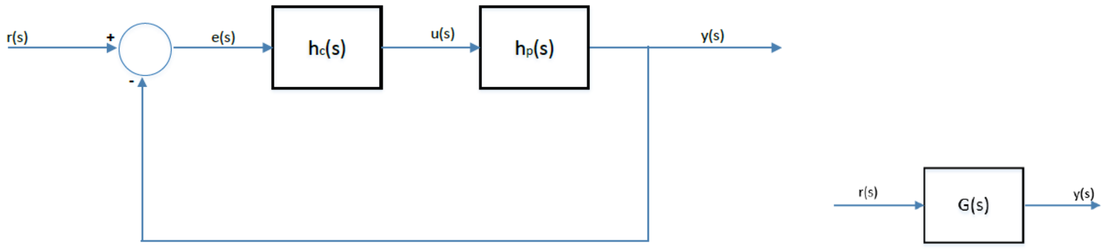


Figure 3-1: A standard and an accumulated feedback system for set point test

Back Calculation

The set-point experiment conducted on the closed-loop produces set of data for transient response as shown in Figure 3-3. The data obtained are used in estimation and mapping of the parameters in the feedback transfer function model, which are later used in back calculation.

Back calculation is determination of open-loop process model dynamics from the estimated closed-loop response using set of mathematical expressions (algorithms). These algorithms relate the second-order feedback response parameters (given by Equation (3.6), (3.32), (3.37) and (3.59)) with parameters for standard transfer function of the process $h_p(s)$ as given by Equation (3.1), (3.58) and (3.72).

For successful back calculation which leads to distinctive transfer models of first and second orders, the system excitation should at least fulfil requirements for persistent excitation of sufficiently higher order [29].

3.1 Closed-loop Transient response SID methods/algorithms

Since early 80s, plenty of efforts has been dedicated in finding robust and reliable process models from set point response of the closed-loop systems, resulting into miscellaneous algorithms with various strengths and limitations when applied to processes with different dynamics. In this chapter, nine common methods/algorithms will be reviewed. Whereby five of them are only for first-order process SID.

3.1.1 Yuwana and Seborg method (YS)

The YS method has paved the way and provided fundamental approach in tuning controllers on-line using single experiment test performed in feedback system under operation. The experiment is conducted by observing, and analysing the closed-loop response to the small step changes in controller's set point implemented on the system.

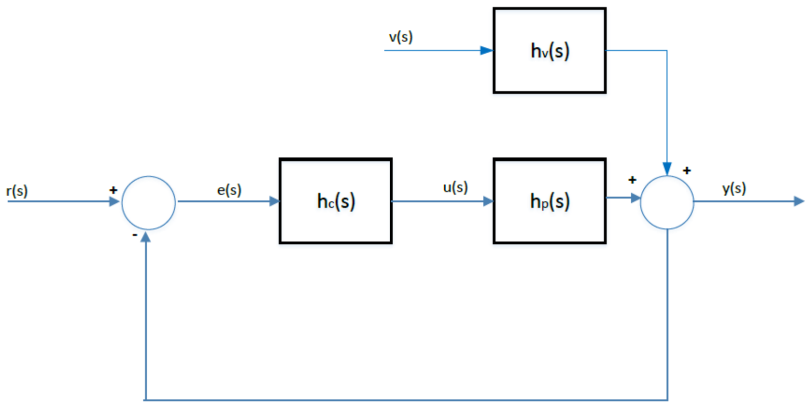


Figure 3-2: Block diagram for simplified feedback control system

The method is particularly meant for simple first order process plus dead time as given by Equation (3.1) whereby the parameters are estimated directly from the closed system response data.

$$\mathrm{h}_{p(s)} = \frac{\mathrm{K}_p \mathrm{e}^{-\tau_p s}}{1 + \top_p s} \tag{3.1}$$

The K_p in the Equation (3.1) denotes process gain, also known as steady state gain, τ_p denotes process time delay (dead time), while $\top_p$ denotes process time constant (time taken for the process to reach 63% of the maximum response) [18].

3.1.1.1 Method built-up

Consider a closed loop system (a feedback system) as shown in Figure 3-2. The system which is composed of first order process model with time delay, is controlled by proportional controller, a special form of PID-controller whereby the Integral and Derivative parts of the controller are made inapplicable. It is highly recommended that the value of the proportional gain is carefully chosen and large enough to excite the system to form overshoot i.e. oscillatory behaviour [2].

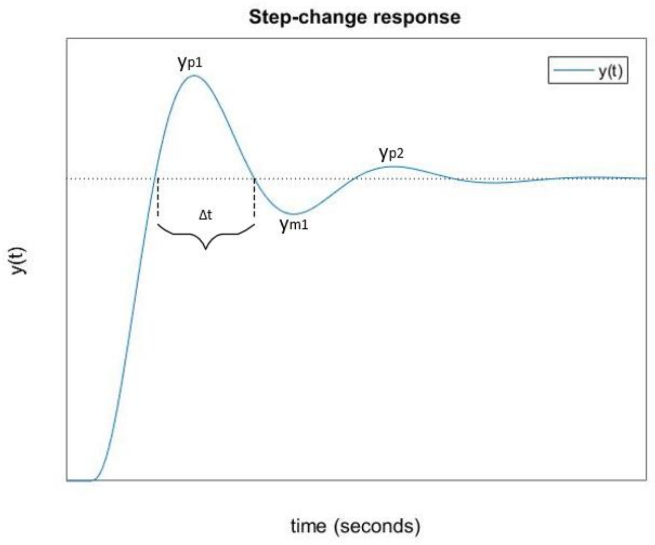


Figure 3-3: Closed-loop response to set point step-change

The YS method is based on the following assumptions

- The disturbances (noise) acting on the system are negligible, i.e. $v(s) = 0$.
- The process and the closed loop transfer function are unknown.

Based on the assumption, the closed loop transfer function for the step change shown in Figure 3-3 is given as in Equation (3.2) [2].

$$\frac{y_{(s)}}{r_{(s)}} = \frac{h_{c(s)} h_{p(s)}}{1 + h_{c(s)} h_{p(s)}} \tag{3.2}$$

Where $h_{c(s)}$ denotes controller's transfer function, $r_{(s)}$ represents step change in controller's set-point (reference value), while $y_{(s)}$ denotes the closed system output. Since the controller used in YS's transient response system identification is a simple proportional controller (P-controller), the controller transfer function is therefore reduced to [2],

$$h_{c(s)} = K_c \tag{3.3}$$

Substituting and simplifying the Equations (3.2) and (3.3) yields loop transfer function expression containing primarily of an accumulated gain, time constant and dead time as given in Equation (3.4) [2].

$$\frac{y_{(s)}}{r_{(s)}} = \frac{K_c K_p e^{-\tau_p s}}{1 + \top_p s + K_c K_p e^{-\tau s}} \tag{3.4}$$

YS employed analytical method in estimation of the model parameters from the closed loop response, by introducing first order Padé-approximation of the dead time given as, [2]

$$e^{-\tau s} \cong \frac{1 - 0.5\tau s}{1 + 0.5\tau s} \tag{3.5}$$

The YS then eliminates the delay term in the denominator of the system response transfer-function by substituting the Equation (3.5) into the denominator of the Equation (3.4), resulting into step change response as expressed in Equation 3.6.

$$\frac{y_{(s)}}{r_{(s)}} = (\frac{K_c K_p}{K_c K_p + 1}) * \frac{(1 + 0.5\tau_s)e^{-\tau s}}{\top^2 s^2 + 2\zeta\top s + 1} \tag{3.6}$$

Where the parameters $\top$ and ζ denotes the natural frequency and the relative damping (also known as damping ratio) of the system. Their magnitudes depend on the process time constant $\top_p$ and its time delay τ_p, and are given as [2]

$$\top = \sqrt{\frac{\top_p \tau}{2(K_c K_p + 1)}} \tag{3.7}$$

$$\zeta = \frac{\top_p + \frac{\tau}{2}(1 - K_c K_p)}{\sqrt{2\top_p \tau (K_c K_p + 1)}} \tag{3.8}$$

The value of ζ influences the oscillatory behaviour of the closed-loop system response. When 0 < ζ < 1, it yields an underdamped response as shown in Figure 3-3. The process model parameters are deduced by the back-calculation method using Equations (3.9), (3.10) and (3.11) as given below [2],

$$K_p = {}^{y_\infty}/_{K_c(A - y_\infty)} \tag{3.9}$$

$$\tau_p = \frac{2\Delta t\sqrt{(1 - \zeta_r{}^2) * (1 + K_c K_p)}}{\pi\left[\zeta_r\sqrt{1 + K_c K_p} + \sqrt{\zeta_r{}^2(K_c K_p + 1) + K_c K_p}\right]} \tag{3.10}$$

$$\top_p = \frac{\Delta t}{\pi}\left(\zeta_r\sqrt{1 + K_c K_p} + \sqrt{\zeta_r{}^2(K_c K_p + 1)}\right)\left(\sqrt{(1 - \zeta_r{}^2) * (1 + K_c K_p)}\right) \tag{3.11}$$

The value of A in Equation (3.9) represents the magnitude of the set point applied to the feedback system, y_∞ denotes the system response at the steady state, while the value of ζ_r can be calculated using two different expressions as provided in Equation (3.12) and (3.13).

$$\zeta_r = \frac{-\ln\left[\frac{y_\infty - y_{m1}}{y_{p1} - y_\infty}\right]}{\left[\sqrt{\pi^2 + \left\{\ln\left(\frac{y_\infty - y_{m1}}{y_{p1} - y_\infty}\right)\right\}^2}\right]} \tag{3.12}$$

$$\zeta_r = \frac{-\ln\left[\frac{y_{p2} - y_\infty}{y_{p1} - y_\infty}\right]}{\left[\sqrt{4\pi^2 + \left\{\ln\left(\frac{y_{p2} - y_\infty}{y_{p1} - y_\infty}\right)\right\}^2}\right]} \tag{3.13}$$

The quantities y_{p1}, y_{p2}, y_{m1}, and y_∞ are directly measured from the step response data as illustrated in Figure 3-3. Some closed-loop systems takes longer time to reach the steady state. To deal with such situation, YS calculated the steady state response based on the magnitude of the two first oscillations as [2],

$$y_\infty = \frac{y_{p2}y_{p1} - {y_{m1}}^2}{y_{p1} + y_{p2} - 2y_{m1}} \tag{3.14}$$

As a result, the YS method derived an expression for the time-domain transient response based on step change of controller's set point as given in Equation (3.15).

$$y_{(t')} = A\frac{\mathrm{K}_c\mathrm{K}_p}{\mathrm{K}_c\mathrm{K}_p + 1} * \left[1 - De^{-\frac{\zeta t'}{\top}}\sin(Et' + \phi)\right] \tag{3.15}$$

YS defined the parameters in the transient response as,

$$t' = t - \tau \tag{3.16}$$

$$D = \frac{\sqrt{1 - \frac{\zeta\tau}{\top} + \frac{1}{4}(\frac{\tau}{\top})^2}}{\sqrt{1 - \zeta^2}} \tag{3.17}$$

$$E = \frac{\sqrt{1 - \zeta^2}}{\top} \tag{3.18}$$

$$\phi = \tan^{-1}\left[\frac{\top\sqrt{1 - \zeta^2}}{\zeta\top - 0.5\tau}\right] \tag{3.19}$$

The YS method estimates, in addition, the time t_k at which the critical points (maximum overshoot and undershoot) of the response takes place, as given by Equation (3.20). k is a positive integer representing the order at which those critical points appear [2].

$$t'_k = \frac{\tan^{-1}\left(\frac{1-\zeta^2}{\zeta}\right) + k\pi - \phi}{\sqrt{1-\zeta^2}/\tau} \tag{3.20}$$

3.1.1.2 Strengths and shortcomings

The development of YS method proved to influential for closed-loop system identification, as it omits the need for trial and error test, and hence less time consuming when it comes to controller tuning. However, the performance of the method is worsens when applied to processes with large time delay (dead time), truly due to the implementation of low order Padé approximation of the delay term [4].

3.1.2 Jutan and Rodriguez’s method (JR)

The JR method is basically an extension of the YS method, where by the impact of the Padé approximation in the process model is examined and improved by applying higher order approximation [4].

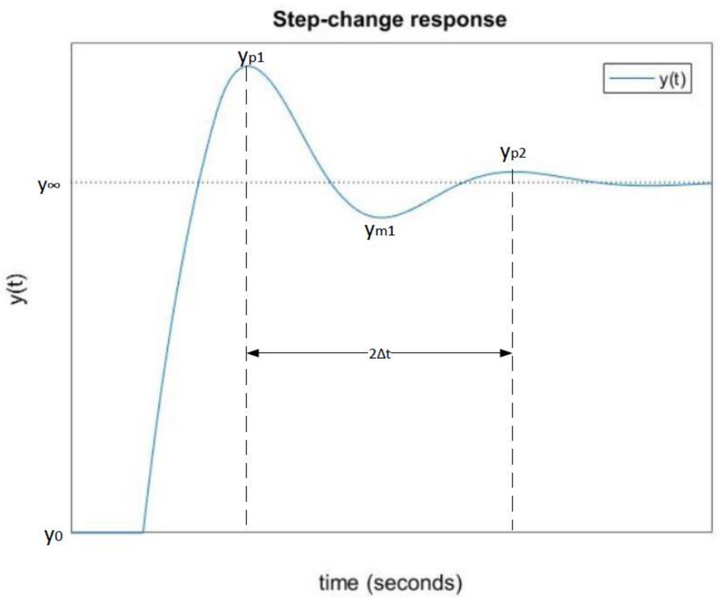


Figure 3-4: Set-point response data curve as defined in [4]

3.1.2.1 Method built-up

A set point response as shown in Figure 3-4 is produced when a closed-loop feedback system is excited with the step change in the reference value. Using Laplace transformation, the response is expressed as in Equation (3.4). At this stage, the delay term is extended into second order Padé approximation in functional form (Equation (3.21)), which includes constants γ_1, γ_2, and δ, to be fitted by for example nonlinear least squares [4].

$$e^{-\tau s} \cong \frac{1 + \gamma_1 s + \gamma_2 \tau^2 s^2}{1 + \delta \tau s} \tag{3.21}$$

The introduction of the expanded delay term expression influences the process model parameters to be estimated. As a result, the Equation (3.7), (3.8), (3.9), (3.10), and (3.11) are respectively replaced by Equation (3.22), (3.23), (3.24), (3.26), and (3.25) in order to accommodate the constant terms within the delay expression (refer to Equation (3.21)) [4].

$$\top = \sqrt{\frac{\delta\top_p\tau + \gamma_2\mathrm{K}_c\mathrm{K}_p\tau^2}{\mathrm{K}_c\mathrm{K}_p + 1}} \tag{3.22}$$

$$\zeta = \frac{\top_p + \tau(\delta+\gamma_1\mathrm{K}_c\mathrm{K}_p)}{\sqrt{2(1+\mathrm{K}_c\mathrm{K}_p)(\top_p\tau + \gamma_2\mathrm{K}_c\mathrm{K}_p\tau^2)}} \tag{3.23}$$

The JR calculates back the process model parameters using the response data y_{p1}, $y_{p2}, y_{m1}, \Delta t,$ and y_∞ from step-change excited into the feedback system as illustrated in Figure 3-4. The steady state gain K_p, the process time constant $\top_p$, and the process time delay τ_p can be estimated as given in the below expressions [4];

$$K_p = \frac{|y_\infty - y_0|}{K_c(|R - R_0| - |y_\infty - y_0|)} \tag{3.24}$$

$$\top_p = \frac{-(2\gamma_2\mathrm{K}_c\mathrm{K}_p\alpha\beta + \alpha\delta) \pm \sqrt{(2\gamma_2\mathrm{K}_c\mathrm{K}_p\alpha\beta + \alpha\delta)^2 - 4\left(\beta^2\gamma_2\mathrm{K}_c\mathrm{K}_p + \beta\delta\right)(1+\mathrm{K}_c\mathrm{K}_p)(\gamma_2\mathrm{K}_c\mathrm{K}_p\alpha^2 - \top^2(1+\mathrm{K}_c\mathrm{K}_p))}}{2\left(\beta^2\gamma_2\mathrm{K}_c\mathrm{K}_p + \beta\delta\right)} \tag{3.25}$$

$$\tau = \alpha + \beta\top_p \tag{3.26}$$

Where the variables α and β are provide as,

$$\alpha = \frac{2\zeta\top(1+\mathrm{K}_c\mathrm{K}_p)}{\delta + \gamma_1\mathrm{K}_c\mathrm{K}_p} \tag{3.27}$$

$$\beta = -(\delta + \gamma_1\mathrm{K}_c\mathrm{K}_p)^{-1} \tag{3.28}$$

Moreover, the JR identified the similar expression for system response in time-domain (Equation (3.15)) as defined in [2]. However, due to application of an extended Padé approximation, the magnitudes of variables $D, E,$ and ϕ, change to [4],

$$D = \sqrt{1 - \frac{2\delta\zeta\tau}{\top} + \delta^2\left(\frac{\tau}{\top}\right)^2} \tag{3.29}$$

$$E = \frac{\sqrt{1-\zeta^2}}{\top} \tag{3.30}$$

$$\phi = \tan^{-1}\left[\frac{\top\sqrt{1-\zeta^2}}{\zeta\top - \delta\tau}\right] \tag{3.31}$$

3.1.2.2 Strengths and shortcomings

The JR method expands the system identification for closed-loop to yield improved and acceptable parameters also when dealing with processes containing large time delay. Introducing the second order Padé approximation matches the behaviour of the closed-loop, due to its second-order nature of the response of the feedback system. The method also identifies and update the expression for variable E due to calculation flaw appeared in the YS method [4]. However, solving the higher-order Padé approximation using nonlinear function, adds up to calculation complexity as it requires least square fitting to estimate the constant parameters in Equation (3.21) [4].

3.1.3 Jietae Lee's method (JL)

The JL method seeks to improve the closed-loop transient response for the process models with significant large dead time. It is an extension of the YS method whereby the Padé approximation for the time-delay term is replaced with a model reduction technique that requires solving nonlinear equation iteratively [5].

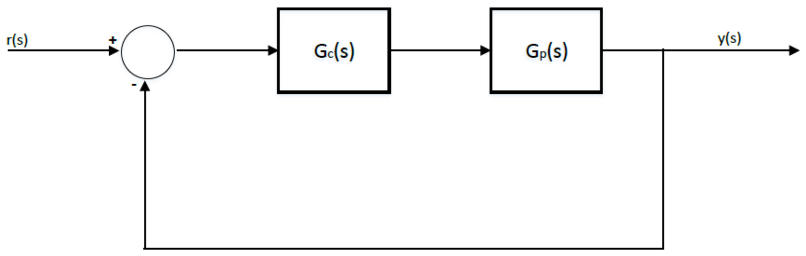


Figure 3-5: Standard feedback system - block diagram

3.1.3.1 Method built-up

Consider a second order closed-loop transfer function with time delay as expressed in Equation (3.32). A Step change in the controllers (P-controller) reference value gives system response as shown in Figure 3-3.

$$\frac{y_{(s)}}{r_{(s)}} = \frac{K(qs+1)e^{-\tau_s}}{\top^2 s^2 + 2\zeta\top s + 1} \tag{3.32}$$

The JL method chooses, instead of direct approximation of the delay term, to map and equate the dominant poles of the transfer function of the closed-loop model in Equation (3.4) to poles of the observed process transfer function as given by Equation (3.32) [5]. A further comparison of steady state gains for the both transfer functions gives the magnitude of the process gain K_p as given in [2] by Equation (3.9). It also leads to a trigonometrical expression for the error term $e^{-\alpha\tau}$, a new process time-constant expression $\top_p$ based on the process time delay τ_p and damping ratio ζ, and as well as new expression for natural frequency of the system $\top$. The new mathematical expressions are as given in Equation (3.33), (3.34) and (3.35) respectively [5].

$$\mathrm{e}^{-\alpha\tau} = \frac{\mathrm{K}_c\mathrm{K}_p}{\beta}\left[\alpha\sin(\beta\tau) - \beta\cos(\beta\tau)\right] \tag{3.33}$$

$$\tau_{k+1} = \frac{1}{\beta}\left[v + \tan^{-1}\left(\frac{\beta\mathrm{e}^{-\alpha\tau_k}}{\mathrm{K}_c\mathrm{K}_p\sqrt{\alpha^2+\beta^2}\cos(\beta\tau_k - v)}\right)\right] \tag{3.34}$$

$$\top_p = \frac{1}{\alpha}\left[1 + \mathrm{K}_c\mathrm{K}_p\mathrm{e}^{-\alpha\tau}\cos(\beta\tau)\right] \tag{3.35}$$

Where the value of quantities α, β and v are given as

$$\alpha = \frac{\zeta}{\top}\ ,\ \beta = \frac{\sqrt{1-\zeta^2}}{\top},\ \text{and}\ \ v = \tan^{-1}(\beta/\alpha) \tag{3.36}$$

The method opens for different nonlinear iterative approaches, for example a fixed point iteration method, to be utilized in solving Equation (3.33), with the first positive value from the calculation as preferred magnitude of the process time delay [5].

3.1.3.2 Strengths and shortcomings

By introducing the nonlinear iterative approach for the time delay term, the system response for the process with large time delay is improved. The method provides therefore an alternative to the Padé approximation in estimation of process model parameters. However, the approach also adds mathematical complexity due to iterative expressions [5].

3.1.4 Cheng-Liang Chen's method (CLC)

The CLC method is an acknowledgement of JL's modification of the delay term expression involving the iterative nonlinear expression. The main concept is to observe and demonstrate the validity of the JL method on underdamped processes. The method makes it possible to deduct the system critical data, i.e. phase cross-over frequency ω_c and system's gain margin GM, directly from the closed-loop response rather than using the characterized (standard) low-order parametric model [6, 7].

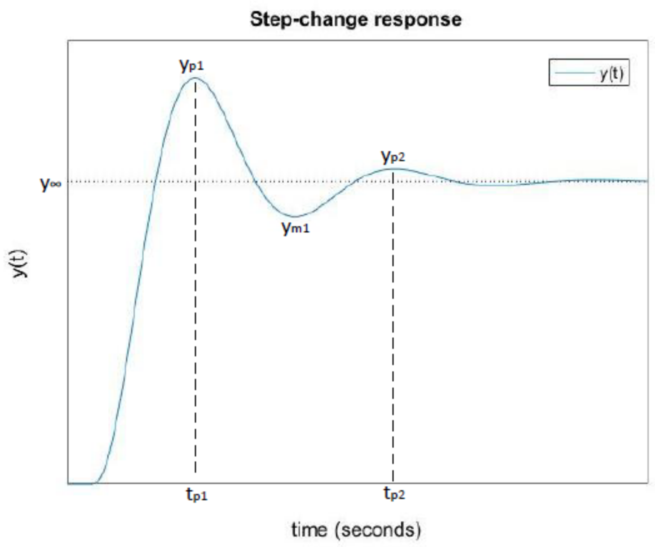


Figure 3-6: Standard underdamped feedback response

3.1.4.1 Method built-up

Consider a process controlled by proportional controller in a closed-loop system with large enough feedback gain (i.e. underdamped closed-loop system). Figure 3-6 shows the response to the step change of the controller's reference value as given in the expression below [6],

$$\frac{y_{(s)}}{r_{(s)}} = \frac{K e^{-\tau_s}}{\top^2 s^2 + 2\zeta \top s + 1} \tag{3.37}$$

Instead of using the closed-loop response (Equation (3.37)) to estimate the parameters for process model (i.e. open-loop parametric model), Chen went on defining ultimate gain (gain margin, $|G_{cl}(i\omega_c)|$) and ultimate frequency (cross-over frequency, ω_c) for the open loop system which are essential in tuning the controller [6]. The phase cross-over frequency is estimated by solving this nonlinear expression.

$$-\tau\omega_{\ c} - \tan^{-1}\left(\frac{2\zeta\top\omega_{\ c}}{\sqrt{1-\top^2\omega_{\ c}{}^2}}\right) = -\pi \tag{3.38}$$

While the magnitude (GM) at this particular frequency is given as

$$|G_{cl}(i\omega_c)| = \frac{K}{\sqrt{(1-\top^2\omega_{\ c}{}^2)^2 + (2\zeta\top\omega_{\ c})^2}} \tag{3.39}$$

Both the closed-loop ($G_{cl}(s)$) and the open-loop ($G_c(s)G_p(s)$) experience the same ultimate frequency [6]. The corresponding open-loop ultimate gain is given as

$$\left|G_c(i\omega_c)G_p(i\omega_c)\right| = \frac{|G_{cl}(i\omega_c)|}{\sqrt{1+2|G_{cl}(i\omega_c)| + |G_{cl}(i\omega_c)|^2}} \tag{3.40}$$

Applying reciprocal on the Equation (3.40) equals to the gain margin for the system with the controller gain K_c, thus giving the ultimate gain for the feedback controller as [6],

$$K_{cu} = K_c GM \tag{3.41}$$

The process model parameters in the Equation (3.1) are thereafter identified from the resulted ultimate frequency and closed-loop controller gain as follows,

$$\top_p = \frac{1}{\omega_u}\sqrt{{K_{cu}}^2{K_p}^2 - 1} \tag{3.42}$$

$$\tau = \frac{1}{\omega_u}\left[\pi - \tan^{-1}\left(\top_p \omega_u\right)\right] \tag{3.43}$$

While the value of K_p can be estimated as proposed in [2]. I addition to that, Chen utilized the measurable quantities y_{p1}, y_{m1}, y_{p2} and y_∞ from the underdamped step-change response of the closed-loop (Figure 3-5) in estimation of the parameters $(K, \top, \zeta)$ for the feedback transfer function as follows [6],

For closed-loop feedback gain,

$$K = {}^{y_\infty}/_{A} \tag{3.44}$$

expression for natural frequency is given as,

$$\top = \frac{\left(t_{m1} - t_{p1}\right)\sqrt{1-\zeta^2}}{\pi} \tag{3.45}$$

and the damping ratio can be estimated as,

$$\zeta = \frac{-\ln(H)}{\sqrt{\pi^2 + (\ln H)^2}} \tag{3.46}$$

where the magnitude of the variable H is estimated from the critical points in the response as given in below[6],

$$H = \frac{1}{3}\left[\frac{y_{p1} - y_\infty}{y_\infty} + \frac{y_\infty - y_{m1}}{y_{p1} - y_\infty} + \frac{y_{p2} - y_\infty}{y_\infty - y_{m1}}\right] \tag{3.47}$$

The expression involves the steady state response value for the system (y_∞). However, waiting for systems to reach the steady state is unnecessary, as the value can be calculated using YS's Equation (3.14). Thus, less time consuming also when processes with long time constants are concerned [6].

3.1.4.2 Strength and Shortcoming

The CLC method eliminates the need for low order approximation of the delay term in the feedback transfer function. However, it adds difficulties as it involves solving non-linear equation. Direct determination of the ultimate gain and phase cross-over frequency, however, simplify the concept of on-line tuning since the control parameters can be easily tuned using Nichol's method [6, 7].

3.1.5 Mamat and Fleming's method (MF)

Similar to the CLC, the MF method's main concept is based on determination of the ultimate gain and phase cross-over frequency of the open-loop system directly from the closed-loop system response [8]. However, the MF uses a widely used PID type (PI-controller) in a closed-loop with the process to be identified. The application of the PI-controller eliminates or minimizes the steady-state offset occurring when P-controller is used during the step-change experimentation [8].

3.1.5.1 Method built-up

Consider the closed-loop feedback system as in Figure 3-5 and its corresponding response to a step-change in the PI-controller's set point when the controller's parameters K_c and $T\ _i$ are wisely chosen such that the system possesses underdamped behaviour as shown in Figure 3-6. The controller's transfer function, the process model, and the closed-loop transfer function are as given in Equation (3.48), (3.1) and (3.37) respectively [8].

$$h_c(\mathrm{s}) = \mathrm{K}_c(1 + \frac{1}{T_i s}) \tag{3.48}$$

The values of parameters of the closed–loop transfer function are estimated in time domain using the following expressions.

$$\top = \frac{\left(t_{p2} - t_{p1}\right)\sqrt{1-\zeta^2}}{2\pi} \tag{3.49}$$

$$\zeta = \sqrt{\frac{\left(-\frac{1}{2\pi}\ln\left[\frac{y_{p2} - y_\infty}{y_{p1} - y_\infty}\right]\right)^2}{1 + \left(-\frac{1}{2\pi}\ln\left[\frac{y_{p2} - y_\infty}{y_{p1} - y_\infty}\right]\right)^2}} \tag{3.50}$$

$$\tau = \frac{S_c}{y_\infty} - 2\zeta\top \tag{3.51}$$

Where the quantity S_c can be solved analytically using integral Equation (3.52). As for the feedback gain *K*, the MF's method makes the use of Equation (3.44) given by [6].

$$S_c = \int_0^\infty [y_\infty - y(t)]\,\mathrm{dt} \tag{3.52}$$

The MF method determines the frequency response of the feedback system using the time-domain parameters $K, \zeta, \tau,$ and $\top$ estimated above. At this stage, the ultimate gain and the phase crossover frequency of closed-loop system are determined. To obtain the dynamic of the process alone $H_p(j\omega)$, the MF then eliminates the dynamics of the PI-controller $H_c(j\omega)$ from the closed-loop dynamics $H_{cl}(j\omega)$ [8]. At the phase crossover frequency ω_c, which is the ultimate frequency both for the open and closed-loop transfer function, the open-loop gain can therefore be given as,

$$\left|H_cH_p(i\omega_c)\right| = \frac{|H_{cl}(i\omega_c)|}{1 + |H_{cl}(i\omega_c)|} \tag{3.53}$$

whereby the ultimate frequency (phase cross-over frequency ω_c), can be determined by solving a similar nonlinear expression given as Equation (3.54). For the closed-loop dynamics, its corresponding ultimate gain ($|H_{cl}(i\omega_c)|$, the magnitude at ω_c) is given by Equation (3.39) [8].

$$\tau\omega_{\ c} + \tan^{-1}\left(\frac{2\zeta\top\omega_{\ c}}{1 - \top^2\omega_{\ c}{}^2}\right) = \pi \tag{3.54}$$

The process model parameters $\top_p, \tau_p$, and K_p are then back-calculated using the given expressions below

$$\top_p = \frac{1}{\omega_c}\sqrt{\frac{K_c{}^2K_p{}^2(1 + T_i{}^2\omega_c{}^2) - |H_{cl}(i\omega_c)|^2T_i{}^2\omega_c{}^2}{|H_{cl}(i\omega_c)|\omega_c{}^2T_i}} \tag{3.55}$$

$$\tau_p = \frac{1}{\omega_c}\left[\tan^{-1}(T_i\omega_c) + \tan^{-1}\left(\frac{1}{\top\omega_c}\right)\right] \tag{3.56}$$

$$K_p = \frac{T_i}{K_cS_c}y_\infty \tag{3.57}$$

3.1.5.2 Strengths and shortcoming

Using the PI-controller during transient response test experiment improves the response of the identified process model due to elimination of the steady-state offset, an error that occurs when P-controller is used instead. However, the PI-controller increases the complexity of the method due to the increased non-linear parameter's expressions [8]. During testing, the parameters for PI-controller must be wisely chosen such that feedback systems exhibit oscillatory behaviour. Choosing the optimal parameter values requires therefore prior knowledge or/and experience with the process in question [8].

3.1.6 Jahanshahi and Skogestad's method (JS)

The JS method extends the concept of step-change transient response system identification to also include higher-order process models dynamics, i.e. the models consisting of one or more zeros and multiple poles in transfer function. To minimize the identification complexity associated with higher-order models (due to increased number of parameters to be estimated), the higher-order models are truncated using square root method and so reduced into second-order process model [9].

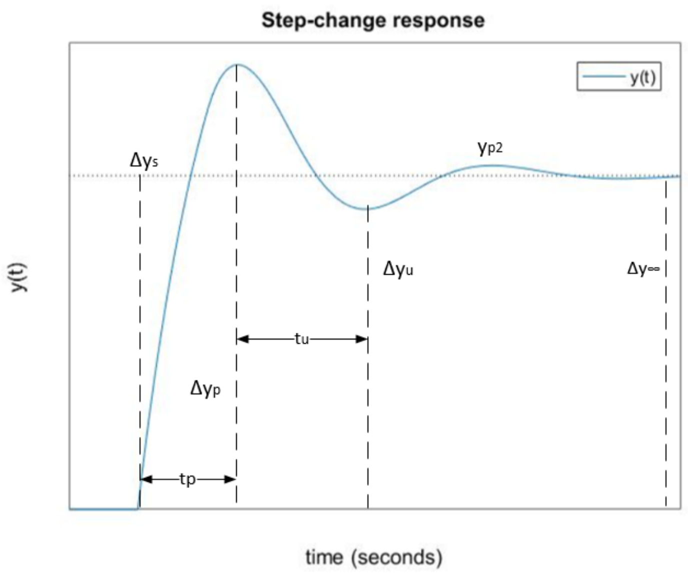


Figure 3-7: Experimental closed-loop step response for stabilized

If the original process model contains both stable and unstable poles (poles located on the right side of the complex plane), the unstable poles are preferred since the poles have more contribution in the process dynamic based on Henkel singular value approach [9]. JS controls the reduced process model with P-controller to form a feedback system ready for online step-change experimentation [9].

3.1.6.1 Method built-up

Consider a typical unstable second-order transfer function model containing a single zero, two poles, and zero time delay as in Equation (3.58),

$$\mathrm{h}_{p(s)} = \frac{b_1 s + b_0}{s^2 - a_1 s + a_0} \tag{3.58}$$

where a_1 and $a_0 > 0$. A P-controller is connected to the process to form a feedback system with the closed-loop transfer function as given in Equation (3.59) [9].

$$\frac{\mathrm{y}_{(s)}}{r_{(s)}} = \frac{K_{c0}(b_1 s + b_0)}{s^2 + (-a_1 + K_{c0} b_1)s + (a_0 + K_{c0} b_0)} \tag{3.59}$$

The response is compared with the typical closed-loop transfer function as given in Equation (3.60) whereby the parameters $K, \zeta, \tau,$ and $\top$ are the estimated from the measurable response data $\Delta y_p, \Delta y_u, \Delta y_s, \Delta y_\infty, t_p$ and t_p as illustrated in Figure 3-8.

$$\frac{\mathrm{y}_{(s)}}{r_{(s)}} = \frac{K_2(1 + \top_z s)}{\top^2 s^2 + 2\zeta \top s + 1} \tag{3.60}$$

At this stage, the JS method applies back calculation approach to determine the steady-state gain K_p and other parameters of the process model as given below [9],

$$K_p = \frac{\Delta y_\infty}{K_{c0}|\Delta y_s - \Delta y_\infty|} \tag{3.61}$$

$$a_0 = \frac{1}{\top^2(1 + K_{c0}K_p)} \tag{3.62}$$

$$b_0 = K_p a_0 \tag{3.63}$$

$$b_1 = \frac{K_2 \top_z}{K_{c0}\top^2} \tag{3.64}$$

$$a_1 = \frac{-2\zeta}{\top} + K_{c0} b_1 \tag{3.65}$$

The JS went on defining the time-domain step-change response as similar to Equation (3.66) given in [2] where the expression for the damping ratio ζ and the quantity E are identical to Equation (3.17) and Equation (3.18) respectively.

$$y_{(t)} = \Delta y_s K_2 \left[1 + D e^{-\frac{\zeta t}{\top}} \sin(Et + \phi)\right] \tag{3.66}$$

However, the phase shift ϕ and quantity D are influenced by presence of pole in the open-loop transfer function and are given as,

$$D = \frac{\sqrt{1 - \frac{2\zeta\top_z}{\top} + (\frac{\top_z}{\top})^2}}{\sqrt{1 - \zeta^2}} \tag{3.67}$$

$$\phi = \tan^{-1}\left[\frac{\top\sqrt{1 - \zeta^2}}{\zeta\top - \top_z}\right] \tag{3.68}$$

The rest of the parameters $\top$, K_p and $\top_z$ are determined using the following expressions [9]

$$\top = \frac{t_u\sqrt{1 - \zeta^2}}{\pi} \tag{3.69}$$

$$K_2 = \frac{\Delta y_\infty}{\Delta y_s} \tag{3.70}$$

$$\top_z = \zeta\top + \sqrt{\zeta^2\top^2 - \top^2[1 - D^2(1 - \zeta^2)]} \tag{3.71}$$

3.1.6.2 Strengths and shortcoming

The method paves the way for identification of high-order process models regardless of the dynamic nature (stability) of the process. However, the method has inherited some of the flaws in earlier method [9, 10].

3.1.7 Dalen and Di Ruscio's method (DR)

The DR method is based on investigation on the creditability and validity of the JS's proposed method in practical application. Their assessment resulted into pointing out number of flaws, and suggested some possible changes aimed at improving the JS method [10].

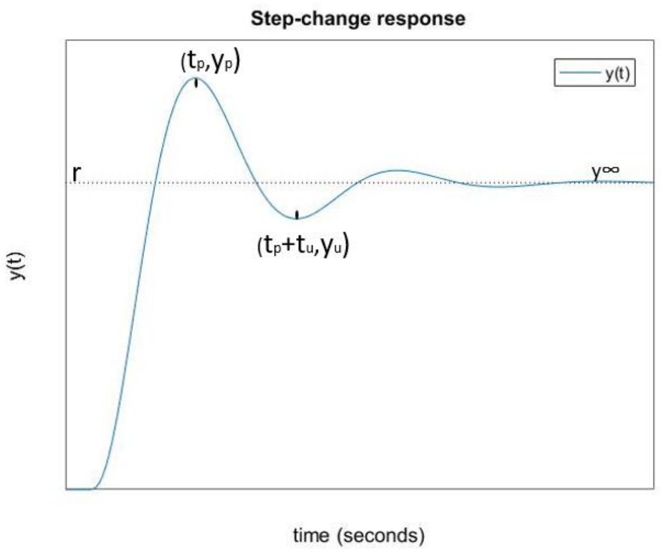


Figure 3-8: Closed loop response to set-point step change [10]

The study inspired the authors to come up with the DR method proposal for identification of second order process transfer-function model with a single zero as given in Equation (3.72).

3.1.7.1 Method built-up

Consider a closed-loop system consisting of a second order process model controlled by P-controller with large enough gain K_c.

$$\mathrm{h}_{p(s)} = \frac{b_1 s + b_0}{s^2 + a_1 s + a_0} \tag{3.72}$$

When a step change in the controller's reference value is excited into the feedback loop, the system responds as shown in Figure 3-8 and its closed-loop transient response in time domain is given as, [10]

$$y(t) = KR[1 - e^{-\frac{\zeta}{\top}t}(\cos(\omega t) + c\sin(\omega t))] \tag{3.73}$$

Where ω is angular velocity which can be estimated as

$$\omega = \frac{\sqrt{1-\zeta^2}}{\top} \tag{3.74}$$

While the value of quantity c in Equation (3.73) is based on time duration for the response to reach the first peak t_p, and is given as

$$c = \frac{\cos(t_p\omega) + \frac{e^{\frac{t_p\zeta}{\top}}(y_p - RK)}{RK}}{\sin(t_p\omega)} \tag{3.75}$$

Whereby the value of t_p is estimated as,

$$t_p = \frac{\frac{\pi}{2} + \tan^{-1}\left(\frac{c\zeta + \top\omega}{\zeta - c\top\omega}\right)}{\omega} \tag{3.76}$$

Even though the proposed algorithm is primarily time-domain based response, the DR method estimates also the zero $\top_z$ in the open-loop transfer function as,

$$\top_z = \zeta\top - c\top\sqrt{1 - \zeta^2} \tag{3.77}$$

$$\top_z = \zeta\top - c\top^2\omega \tag{3.78}$$

whereby the relative damping ζ is estimated using Equation (3.12) [10].

3.1.7.2 Strengths and shortcoming

The DR method highlights the limitation of the JS method, and suggests possible improvement of the method, as well as providing a simplified method with non-complex solution, which estimate transient response of process directly into time-domain. However, the method is meant for high (second and/or higher) order process models [10].

3.1.8 Dalen and Ruscio's method 1 (DR1)

The DR1 is primarily identical to the DR method. The only difference is on the value of the step response of the process model in time-domain (Equation (3.73)), whereby a different value of variable *c* is suggested as,

$$c = \frac{\zeta + \top\omega\tan(t_p\omega)}{\top\omega - \zeta\tan(t_p\omega)} \tag{3.79}$$

The DR1 algorithm is however, more prone to noise compared to the DR [10]

3.2 Method analysis and Simulation Study

This section presents individual and sequential evaluation of the algorithms and performance of the methods presented in chapter 3.1. To start with, method's mathematical expressions are first weighed up against the mathematical development techniques (formula deriving techniques), and thereafter validated against findings in the proceeding methods.

To evaluate the methods, a simulation study using numerical examples is conducted for each method. The objective is to determine methods' validity, performance and robustness of the response under the step change experimentation. Five different values of P-controller gain K_c

are used during simulation to observe K_c sensitivity on process model parameters to be identified. The simulation study is performed in MATLAB software using the programming codes available in Appendix B and C.

3.2.1 Simulated examples

For this phase of simulation, three non-noisy, different numerical examples (representing systems of different dynamics) are used, particularly to evaluate the identified parameters. More examples, also with added measurement noise, are to be simulated later during online tuning of PID controller in section 3.3.

During simulation, the step response identification methods YS, JR, JL, CLC and MF are used to calculate first order approximation of Equation (3.80), (3.81) and (3.82). While JS, JSDR, DR and DR1 are applied to develop second order approximation of the similar examples. Additionally, the half-rule model reduction technique discussed in [32] and denoted by HR (or HF in some cases), is used to verify the first-order approximation method results.

Example 1

Equation (3.80) represents a third-order process model containing three poles (T_1=2, T_2=1 and T_3=0.5) plus time delay ($\tau = 1$). The model has a dominant pole T_1 (i.e. $T_1 > \tau$). The example is identical to example 1 in [2].

$$H_p(s) = \frac{e^{-s}}{(2s+1)(s+1)(0.5s+1)} \tag{3.80}$$

Example 2

Equation (3.81) represents a firth-order process model containing five identical poles of T=1, with no time delay (i.e. τ =0). The model will provide indication of algorithm performance when applied to higher order process. It is identical to example 2 in [2].

$$H_p(s) = \frac{1}{(s+1)^5} \tag{3.81}$$

Example 3

Equation (3.82) represents a third-order process model containing one dominant T_1=2 and two identical poles T_2=T_3=1, plus a time delay of τ =3 which is larger than the dominant pole .It is identical to example 3 in [2], and particularly chosen to reveal the performance the algorithms for processes with significant large dead time.

$$H_p(s) = \frac{e^{-3s}}{(s+1)^2(2s+1)} \tag{3.82}$$

3.2.2 Choosing K_c and T_i for step change experimentation/simulation

As condition for applicability, the algorithms require oscillatory, but stable closed-loop system response [2, 4, 6, 8-10]. It follows that the values of K_c and T_i must be carefully chosen to ensure that damping ratio interval $0 < \zeta < 1$ is maintained during set-point test.

This underline the importance of understanding the process behaviour prior to experimentation. In most practical cases, the process dynamics and the suitable value of K_c are not known in advance. It may therefore takes several trial and error attempts before landing in suitable value of K_c and T_i [33].

3.2.3 Step Response SID for first order processes

In order to have consistent study, each example is simulated using the same K_c values for all the five algorithms to be investigated as shown in Table 3.1.

Table 3.1: K_c values used for simulation

Example	K_{c1}	K_{c2}	K_{c3}	K_{c4}	K_{c5}
1	2.00	1.50	1.00	0.75	0.50
2	2.00	1.50	1.00	0.75	0.50
3	1.25	1.00	0.75	0.50	0.25

3.2.3.1 YS method

3.2.3.1.1 Method Analysis

In their calculation for the ζ of closed-loop response (Equation 7 in [2]), YS ignored the possibility of having two values for the ζ as ± sign on the denominator is left out as pointed in [4]. Additionally, YS provides two separate expressions for ζ, Equation (3.12) and (3.13) based on critical points ($y_{p1}, t_{p1}, y_{m1}, t_{m1}, y_{p2}$ and y_∞) obtained from the step response of the closed-loop (Figure 3-3), and recommends the average of the two equations as applicable value of ζ. However, the Equation (3.12) is found to give better-signal- to noise properties as outlined in [4].

The process model gain K_p given by [2] does not consider responses with non-zero initial values (i.e. for $y(0) \neq 0$), neither the dynamic set-point change R, applied before the system reaches steady state due to the previous step change, R_0 [4]. An alternative expression to incorporate the dynamic set-point change in both directions, i.e. increment and decrement change, is proposed as Equation (3.24) in [4].

YS developed Equation (3.10) and (3.11) for respective back calculation of τ_p and $\top_p$ [2] respectively. Based on their derivation technique, the expressions are incorrect and should rather be given as below; [4, 10]

$$\top_p = \frac{\Delta t \beta_1 \beta_2}{\pi} \text{ and } \tau_p = \frac{2\Delta t \beta_2}{\pi \beta_1} \tag{3.83}$$

Where β_1 and β_2 are given as

$$\beta_1 = \zeta\sqrt{K_c K_m + 1} + \sqrt{\zeta^2(K_c K_m + 1) - 1 + K_c K_m} \text{ and } \beta_2 = \sqrt{(1-\zeta^2)(K_c K_m + 1)} \tag{3.84}$$

As consequence, the expression would yield complex number solutions for $\top_p$, for all values of ζ such that the open-loop gain K is, [7].

$$K < \frac{1-\zeta^2}{1+\zeta^2}$$

Finally, the YS's time-domain based transient estimation $y(t)$ (Equation (A-1) in [2]), contains a wrong expression for E. The correct value of E should read as provided in [7, 10] as

$$E = \frac{\sqrt{1-\zeta^2}}{\top}$$

According to [2], the method's applicability is restricted to process with no significant dead time due to low order approximation of delay term (Equation (3.5)). JR tested the validity of the claim by simulation with extended delay term (Equation 3.21) [4].

3.2.3.1.2 Simulation Results

Example 1

Figure 3-9 shows three different step responses of the closed-loop made of P-controller and the original third-order process model Equation (3.80) at left. The corresponding step response of first-order model approximated using YS algorithm is given on right graph of the same figure. Five various controller gain values (refer to Table 3.1) are used to investigate the algorithm's performance and sensitivity on various K_c, as presented in Table 3.2.

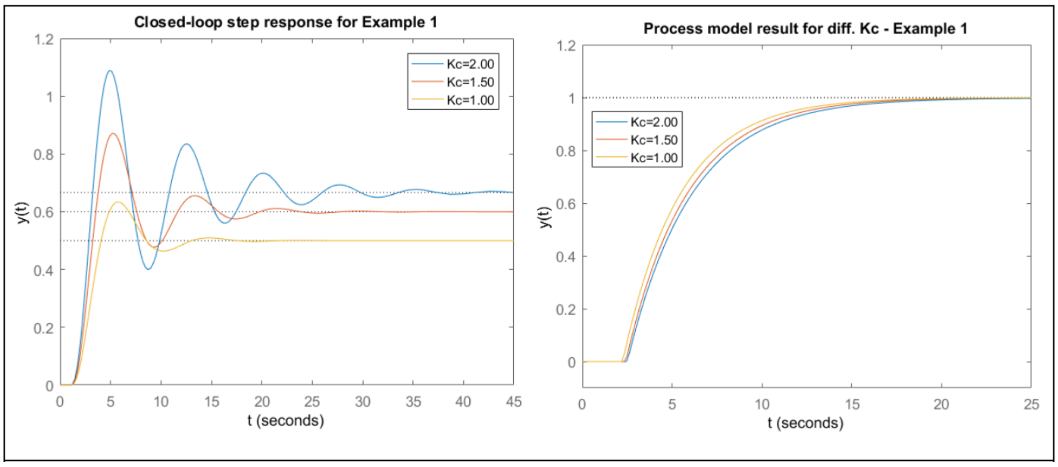


Figure 3-9: Third-order original process and its first-order approx. by YS - Example 1

The K_c values chosen were within the boundary for underdamped stable process. The higher the value, the bigger the oscillation that counteract to higher $\top_p$ and higher τ_p. Comparing to half-rule (HR) based model approximation, the algorithm aligned toward the HR when the magnitude of the overshoot decreased.

Table 3.2: Process model parameters by YS – example 1

K_c	K_p	$\top_p$	τ_p
2.00	0.9987	3.56	2.51
1.50	1	3.382	2.42
1.00	1	3.181	2.25
0.75	1	3.03	2.08
0.50	1	3.012	1.94
HR	**1**	**2.5**	**2.0**

However, for this particular process, the value of $\top_p$ converged towards 3.012, while the magnitude of τ_p kept on decreasing until at lowest value K_c= 0.14 before which the algorithm failed to gives solutions. The highest values of K_c before which the method produced unrealistic and non-number solutions was observed to be at $K_c = 3.07$, with $\top_p$ and τ_p equals to 4.12 and 2.67 respectively.

Example 2

Similar values of K_c as used for this firth order process model, as given by Equation (3.81). Figure 3-10 shows the step response of the feedback system containing the original process on the left graph, and its corresponding approximated first-order model response using YS method.

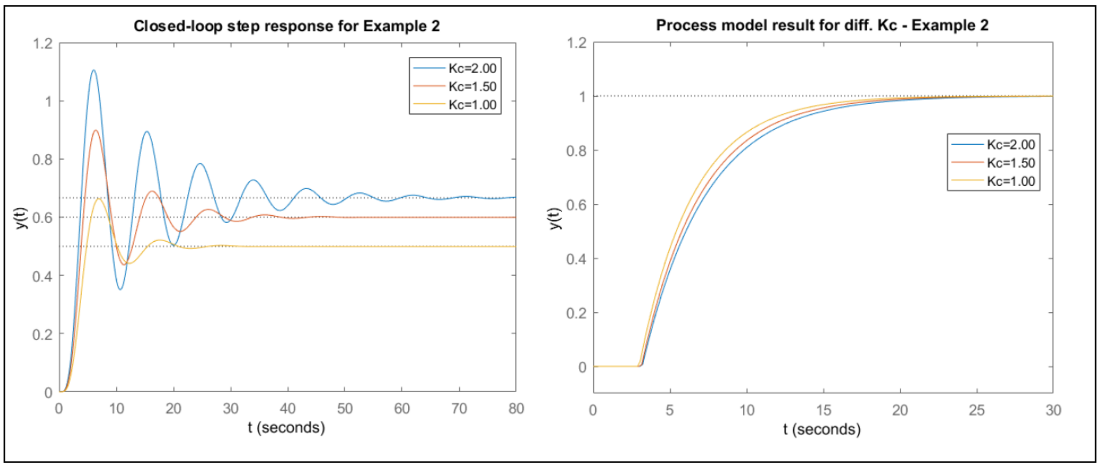


Figure 3-10: Firth-order original process and its first-order approx. by YS - Example 2

The chosen K_c values were inside the desired ζ limit. Again, as the K_c decreased, the estimated time constant $\top_p$ moved closer to the HR's value, while the effective delay τ_p deviated more from the HR's value, as shown in the Table 3.3.

Table 3.3: Process model parameters by YS – example 2

K_c	K_p	$\top_p$	τ_p
2.00	1	4.114	3.19
1.50	1	3.814	3.14
1.00	1	3.495	2.99
0.75	1	3.314	2.84
0.50	1	3.188	2.66
HR	**1**	**1.5**	**3.5**

However, with YS algorithm for this particular process, the values of $\top_p$ and τ_p converged towards 2.941 and 1.91 respectively at the lowest value of $K_c = 0.08$ before the algorithm failed to produce solutions. The highest values of K_c before which the method gave unrealistic and non-number solutions was observed to be at $K_c = 2.88$, with $\top_p$ and τ_p corresponding to 4.581 and 3.18 respectively.

Example 3

Five various K_c values (1.25, 1.0, 0.75, 0.5 and 0.25) are used to estimate first-order model from the process model given by Equation (3.82). Figure 3-11 shows the step response of the closed-loop system consisting of the original model at left, and its corresponding step response of the first-order approximation at right.

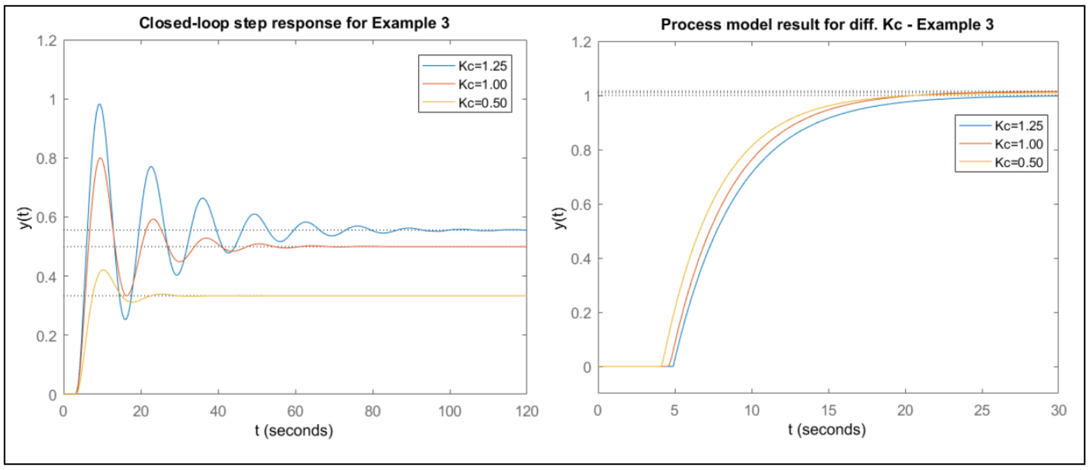


Figure 3-11: Third-order original process and its first-order approx. by YS - Example 3

As illustrated in Figure 3-11, high values of K_c were associated with bigger oscillations (see Figure 3-11) which in turn gave increasing magnitude of model parameters ($\top_p$ and τ_p) as seen in Table 3.4. In respect to HR, the algorithm aligned the model toward the HR when low values of K_c were applied. However, effective delay τ_p deviated more from the HR's value as the K_c kept decreasing. It should be noted that the K_c values chosen were within the boundary for underdamped stable process.

Table 3.4: Process model parameters by YS – example 3

$\boldsymbol{K_c}$	$\boldsymbol{K_p}$	$\boldsymbol{\top_p}$	$\boldsymbol{\tau_p}$
1.25	1	4.09	4.9
1.00	1.016	3.843	4.67
0.75	1	3.763	4.38
0.50	1.01	3.579	4.17
0.25	1	3.555	3.17
HR	1	2.5	4.5

Using low K_c in this example, the estimated model parameters $\top_p$ and τ_p converged towards 3.547 and 2.85 respectively, with $K_c = 0.17$ as the lowest before failed to produce solutions. On the other hand, the highest values of K_c before which the method gave unrealistic or non-number solutions was observed to be at $K_c = 1.72$, with $\top_p$ and τ_p corresponding to 4.413 and 5.11 respectively.

3.2.3.2 JR method

3.2.3.2.1 Method Analysis

JR used Marquardt optimization to map and correlate values of the constants γ_1, γ_2, and δ for the second order delay term approximation (Equation (3.21)) as -0.6143, 0.1247, and 0.3866 respectively [4]. The γ_1 in Equation 27 in [4] increases sensitivity of the $\top_p$ and τ_p toward low values of the open-loop gain ($K = K_c K_p$). When $K \ll 1$ the negative solution from the square root in Equation (3.25) is preferred in order to avoid unusually low $\top_p$ and hence negative τ_p. [4].

The JR method is applicable solely to stable but oscillatory closed-loop systems containing at least two overshoots with peak values y_{p1} and y_{p2} as shown in Figure 3-4, meaning that K_c shall always be high enough to allow oscillations [4].

3.2.3.2.2 Simulation Results

Example 1

Using the JR algorithm, the third-order process model in Equation (3.80) is reduced into first-order with four different P-controller gain values as outlined in Table 3.1. The step response for the original and the approximated model is as shown in Figure 3-12.

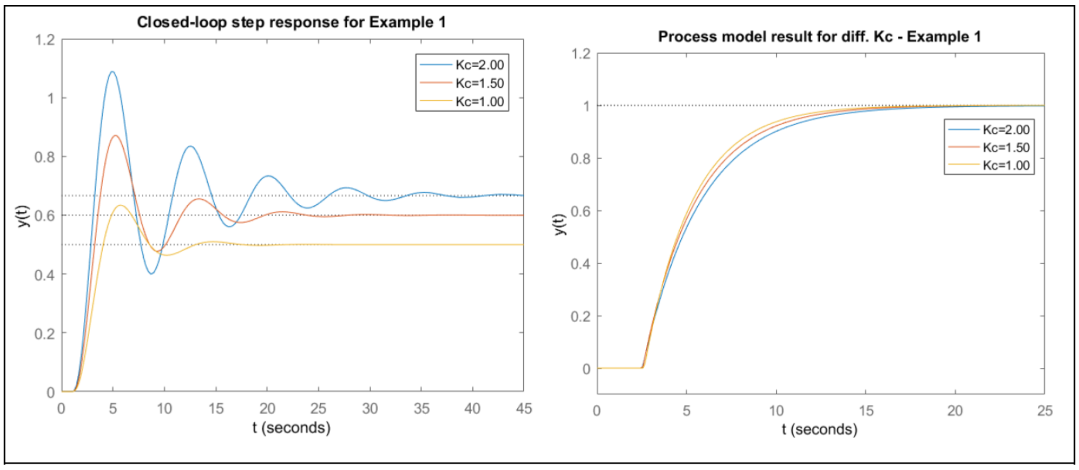


Figure 3-12: Third-order original process and its first-order approx. by JR - Example 1

Again the higher the K_c value, the bigger the oscillation that counteracted to higher $\top_p$ and higher τ_p. A similar tendency that was also observed with YS algorithm. The estimated model parameters moved toward HR's based model approximation, when the magnitude of the overshoot decreased, i.e. with low K_c values.

Table 3.5: Process model parameters by JR – example 1

K_c	K_p	$\top_p$	τ_p
2.00	0.9987	3.119	2.45
1.50	1	2.928	2.54
1.00	1	2.658	2.64
0.75	1	2.498	2.69
0.50	1	5.2933	-34.24
HR	**1**	**2.5**	**2.0**

For this particular process, the values of $\top_p$ and τ_p converged towards 2.416 and 2.72 respectively at the lowest value of K_c= 0.63 before the algorithm gave negative effective delay solutions. The highest registered value of K_c before which the algorithm produced non-number solutions was observed to be at $K_c = 3.07$, with $\top_p$ and τ_p equals 3.443 and 2.29 respectively.

Example 2

Figure 3-13 shows the step response (on the left graph) of the original process given by Equation (3.81) and its corresponding approximated first-order model results using JR method. The four K_c values used for estimation are as listed in Table 3.1.

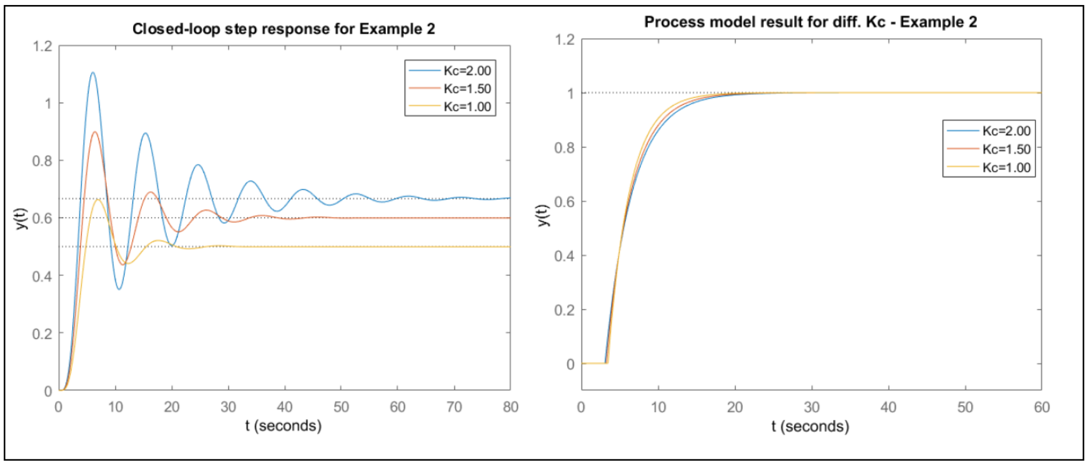


Figure 3-13: Firth-order original process and its first-order approx. by JR - Example 2

As seen in Table 3.6, the dominant time constant $\top_p$ of estimated first-order process, increased with increasing value of K_c, contrary to the estimated effective process dead time which decreased with increasing size of K_c.

Table 3.6: Process model parameters by JR – example 2

K_c	K_p	$\top_p$	τ_p
2.00	1	3.519	3.08
1.50	1	3.197	3.25
1.00	1	2.788	3.46
0.75	1	2.56	3.66
0.50	1	5.367	-35.27
HR	1	1.5	3.5

However, the estimated parameters moved closer toward the HR's calculated values as the oscillations got smaller due to decreasing value of K_c. The value of $\top_p$ and τ_p converged towards 2.419 and 3.77 respectively at the lowest value K_c = 0.63 before the algorithm gave negative effective time delay and/or unrealistic solutions. While the highest registered

K_c before which the algorithm gave non-number solutions was observed to be at $K_c = 2.88$ with $\top_p$ and τ_p equals 4.042 and 2.92 respectively.

Example 3

The JR algorithm is used to estimate a first-order model approximation from the originally third-ordered process model with significant dead time given by Equation (3.82). Figure 3-14 shows step response of the closed-loop system and its approximated first-order model respectively.

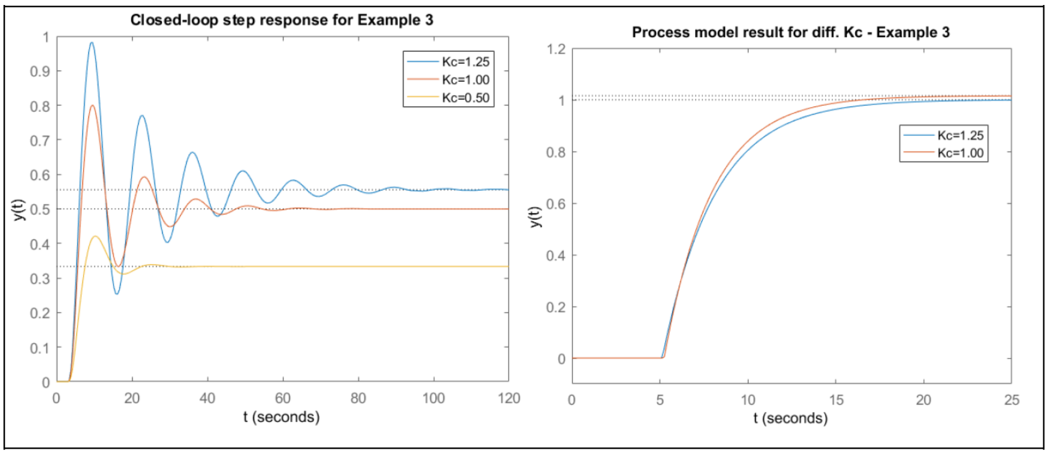


Figure 3-14: Third-order original process and its first-order approx. by JR - Example 3

As presented in Table 3.7, the dominant time constant $\top_p$ of estimated first-order process increased with increasing value of K_c hence experienced more deviation from the corresponding HR's value. While on the other hand, the identified effective process dead time decreased with increasing magnitude of K_c which made it to move closer toward the HR's calculated values.

Table 3.7: Process model parameters by JR – example 3

K_c	K_p	$\top_p$	τ_p
1.25	1	2.985	5.13
1.00	1.016	2.715	5.24
0.75	1	2.548	5.55
0.50	1	5.2426	-35.52
0.25	1	-0.0544	13.8693
HR	**1**	**2.5**	**4.5**

The value of $\top_p$ and τ_p converged towards 2.397 and 5.69 respectively at the lowest value of K_c= 0.63 before reaching negative time delay and/or negative time constant. The highest registered K_c before which the algorithm gave non-number solutions was observed to be at $K_c = 1.72$, with $\top_p$ and τ_p equals 3.307 and 4.91 respectively.

3.2.3.3 JL method

3.2.3.3.1 Method Analysis

JL mapped dominant poles of the closed-loop transfer function to the corresponding poles in observed step response of the underdamped system response (refer to Equation (3.32) and Figure 3-4) [5]. Based on pole matching, JL derived an iterative, non-linear expression for the process time delay τ_p given as Equation (3.33). JL recommends fewest iteration possible, mostly about five iterations, for faster and sufficient results [5]. He also suggested initial value for the τ_p (refer to Equation 12 in [5]), though with no explanation. However, [7] suggests the observable delay in closed-loop response as alternative and possibly better starting point, based on his findings.

The process gain K_p which is inherited from [2] cannot deal with non-zero initial response (i.e. for $y(0) \neq 0$), neither the dynamic set-point change R, as highlighted in [4] and in 3.2.3.1.1 section. An alternative expression to incorporate the dynamic set-point change in both directions (increment and decrement) as well as non-zero initial response is given as Equation (3.24) [4].

The method requires an oscillatory closed-loop system containing at least one overshoot with peak value y_{p1} and one undershoot y_{m1} as shown in Figure 3-6, meaning that K_c should be chosen large enough to allow oscillations [5].

3.2.3.3.2 Simulation results

Example 1

In this example, a first-order process model approximation from the third-order model given as Equation (3.80) is produced using JL algorithm with four different values of K_c as given in the fourth row in Table 3.1. Figure 3-15 show the step response of the P-controlled feedback system composed of the third-order model at left, and its corresponding step response of the estimated first-order model at right.

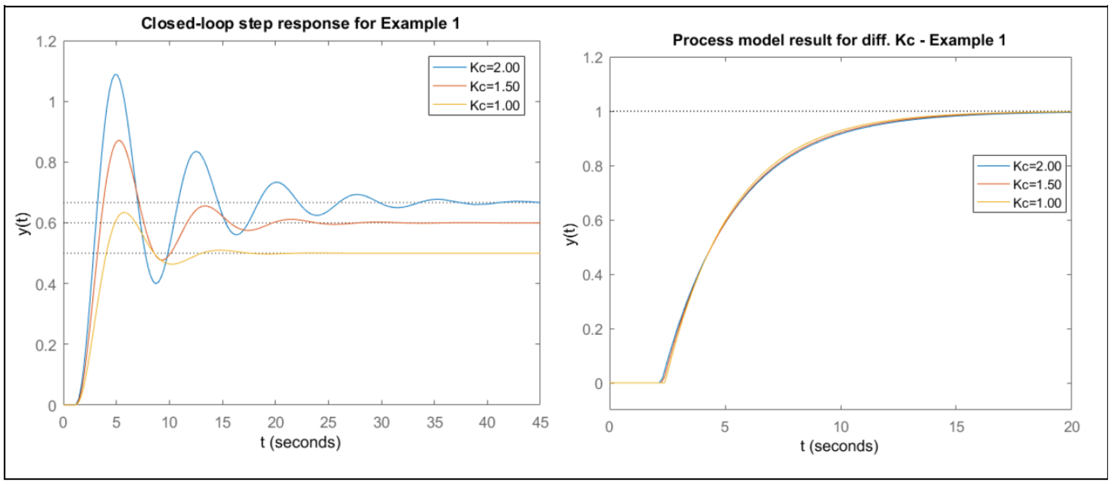


Figure 3-15: Third-order original process and its first-order approx. by JL - Example 1

Similar tendency was observed as shown in Table 3.8 as the dominant time constant $\top_p$ of estimated first-order process increased with increasing value of K_c, hence experienced more deviation from the corresponding HR's value. The effective dead time, however, decreased with increasing magnitude of K_c while at the same time got closer the HR's calculated τ_p.

Table 3.8: Process model parameters by JL – example 1

K_c	K_p	$\top_p$	τ_p
2.00	0.9987	3.114	2.24
1.50	1	3.031	2.31
1.00	1	2.879	2.4
0.75	1	2.726	2.42
0.25	1	2.656	2.58
HR	1	2.5	2.0

The values of $\top_p$ and τ_p converged towards 2.44 and 3.15 respectively at the lowest value of $K_c = 0.14$ before the algorithm failed to gives solutions. The highest registered value of K_c before which the algorithm gave non-number solutions was observed to be at $K_c = 3.08$, with $\top_p$ and τ_p equals to 3.153 and 2.06 respectively.

Example 2

The performance of JL algorithm is further tested in reducing this firth-order process model given by Equation (3.81) into lowest order model using similar P-controller gains as in example 1. The controller together with the original model forms a closed-loop system with step response as shown in Figure 3-16 at left, with corresponding step response of the its first-order approximation at right.

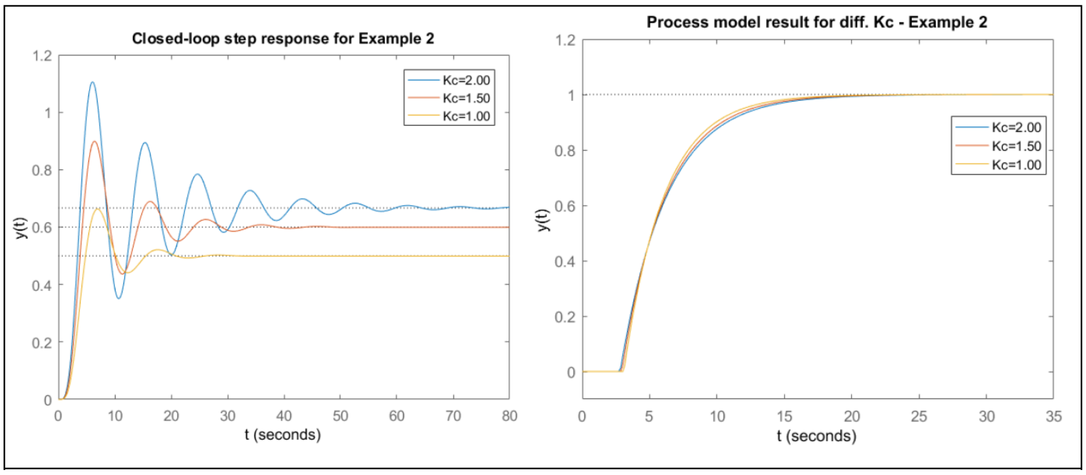


Figure 3-16: Firth-order original process and its first-order approx. by JL - Example 2

Table 3.9 shows the estimated time constant $\top_p$, and the effecting time delay τ_p of the resulted model, whereby the deviation between the parameters obtained from JL and HR decreased with lower values of K_c. While $\top_p$ increased with increasing K_c. The magnitude of τ_p was registered to move in the opposite direction.

Table 3.9: Process model parameters by JL – example 2

K_c	K_p	$\top_p$	τ_p
2.00	1	3.444	2.81
1.50	1	3.24	2.92
1.00	1	2.975	3.08
0.75	1	2.79	3.17
0.50	1	2.611	3.37
HR	1	1.5	3.5

Generally, The values of $\top_p$ and τ_p converged towards 2.033 and 4.31 respectively at the lowest value of $K_c = 0.08$ before the algorithm failed to give solutions. The highest registered value of K_c before which the algorithm gives non-number solutions was observed to be K_c=2.88, with $\top_p$ and τ_p corresponding to 3.704 and 2.63 respectively.

Example 3

Figure 3-17 shows the step response for the closed-loop system composed of a P-controller and the process model Equation (3.82), and its corresponding step response of the approximated first-order model using JL method.

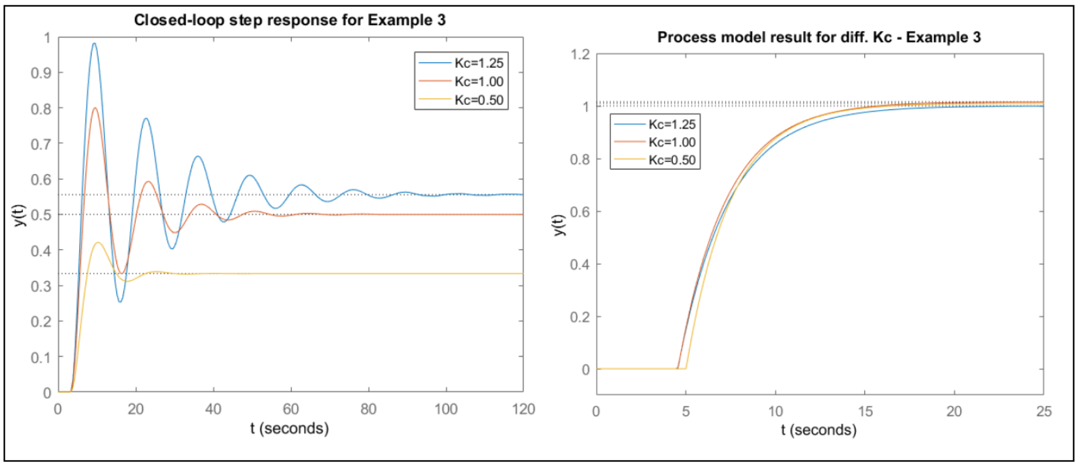


Figure 3-17: Third-order original process and its first-order approx. by JL - Example 3

When the K_c increased, the magnitude of τ_p decreased and aligned more toward the HR calculated parameters τ = 4.5. On the contrary, the $\top_p$ experienced lowest deviation from HR's calculated values when K_c moved toward 0.75, but at the same time increased its magnitude when K_c increased.

Table 3.10: Process model parameters by JL – example 3

$\boldsymbol{K_c}$	$\boldsymbol{K_p}$	$\boldsymbol{\top_p}$	$\boldsymbol{\tau_p}$
1.25	1	2.818	4.55
1.00	1.016	2.684	4.56
0.75	1	2.72	4.67
0.50	1	2.674	4.77
0.25	1	2.519	4.94
HR	1	2.5	4.5

The values of $\top_p$ and τ_p finally converged to around 2.467 and 5.07 respectively at the lowest value of $K_c = 0.17$ before the algorithm became unable to produce feasible solutions. The highest registered value of K_c before which the algorithm gave non-number solutions was observed to be at $K_c = 1.72$, with $\top_p$ and τ_p equals 2.867 and 4.46 respectively.

3.2.3.4 CLC method

3.2.3.4.1 Method Analysis

CLC utilized the similarity of the ultimate frequency ω_c between the closed-loop transfer function G_{cl} and the open-loop transfer function G_cG_p. The CLC went on further to relate the ultimate gains G_{cl} and G_cG_p for the two loops (open and closed) to form the ultimate feedback controller gain K_{cu}, given as in Equation (3.41) [6].

CLC also defined the non-linear equation for w_c as given in Equation (3.38) [CLC]. However, based on mathematical development, Taiwo, in [7], indicated that the expression should rather be as given in Equation (3.54).

In contrast to the algorithms discussed earlier, CLC provided an updated estimation of the relative damping ζ for the closed-loop response, which involves critical response values at first y_{p1} and second peak y_{p2}, and the minimum value at response undershoot y_{m1}. The method is therefore applicable to stable and oscillatory closed-loop systems containing at least two overshoots [6].

3.2.3.4.2 Simulation results

Example 1

The graph on the left-hand side of Figure 3-18 shows the closed-loop response formed using different values of the controller gain K_c ranging from 1.0 to 0.25 within the boundary of the stable underdamped response. The figure gives also the corresponding first-order model result approximated using CLC algorithm.

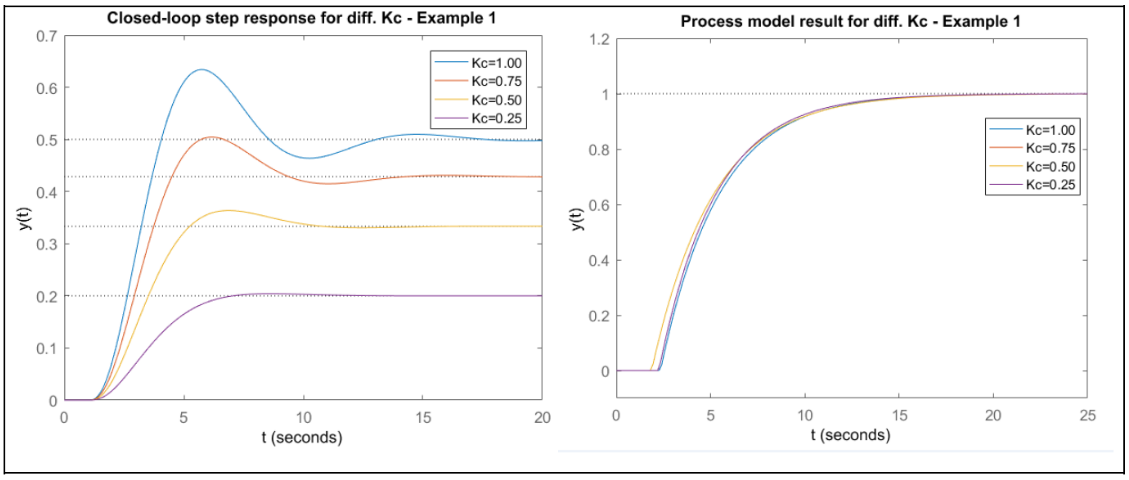


Figure 3-18: Third-order original process and its first-order approx. by CLC - Example 1

Increased magnitude of K_c created bigger oscillations which gave longer time constant τ_p as seen in Table 3.11. The process time-delay τ_p decreased with increasing magnitude of K_c, and appeared to have its maximum values towards lowest K_c. However, the delay seemed also to increase in length around $K_c = 1$. Comparing to HR's based model approximation, the algorithm aligned toward the HR with decreasing values of K_c.

Table 3.11: Process model parameters by CLC – Example 1

K_c	K_p	$\top_p$	τ_p
2.00	1	11.49	0.583
1.50	0.9999	10.31	0.609
1.00	1	3.073	2.34
0.75	1	3.249	1.87
0.5	1	3.249	1.87
0.25	1	2.99	2.25
HR	**1**	**2.5**	**2.0**

It follows that, the values of $\top_p$ converged towards 2.827, while the magnitude of τ_p kept on increasing to 2.57 at lowest value of $K_c = 0.14$, before which the algorithm failed to give solutions. The highest values of K_c before which the method yielded unrealistic and non-number solutions was observed to be at $K_c = 3.07$, with $\top_p$ and τ_p equals 12.09 and 0.612 respectively.

Example 2

This firth-order process model example is chosen to test the performance of the CLC algorithm on higher order models, using similar P-controller gains as in the example above. Figure 3-19 shows the closed-loop step response for various controller gains as seen in the legend, with corresponding step response of the its first-order approximation at right.

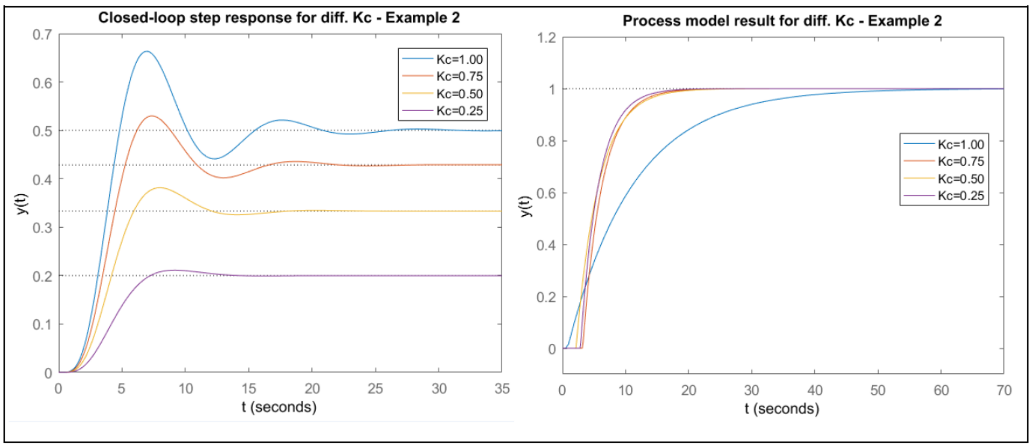


Figure 3-19: Firth-order original process and its first-order approx. by CLC - Example 2

As seen in the Table 3.12, the dominating time-constant of the approximated process increased with increasing K_c. While the delay-time τ_p decreased with increasing magnitude of K_c, and appeared to have its maximum values towards lowest K_c. However, the delay seemed also to increase around $K_c = 0.75$, a similar trend as observed in the example 1 above.

Table 3.12: Process model parameters by CLC – Example 2

K_c	K_p	$\top_p$	τ_p
2.00	0.9996	15.64	0.66
1.50	1	11.29	0.807
1.00	1	10.41	0.803
0.75	1	3.101	3.21
0.50	1	3.6	2.15
0.25	1	2.88	2.84
HR	**1**	**2.5**	**2.0**

With CLC algorithm, the values of $\top_p$ converged towards 2.507 while τ_p moved toward 3.42 at the lowest value of $K_c = 0.08$ before the algorithm failed to produce solutions due to overdamped closed response. The highest values of K_c before which the method produced non-number solutions was observed to be at $K_c = 2.88$, with $\top_p$ and τ_p corresponding to 17.15 and 0.66 respectively.

Example 3

In this example, the CLC algorithm is utilized to estimate a first-order model from the process model given by Equation (3.82). Figure 3-20 shows the step response of the feedback system (with different K_c values) at left, and its corresponding step response of the approximated model at right.

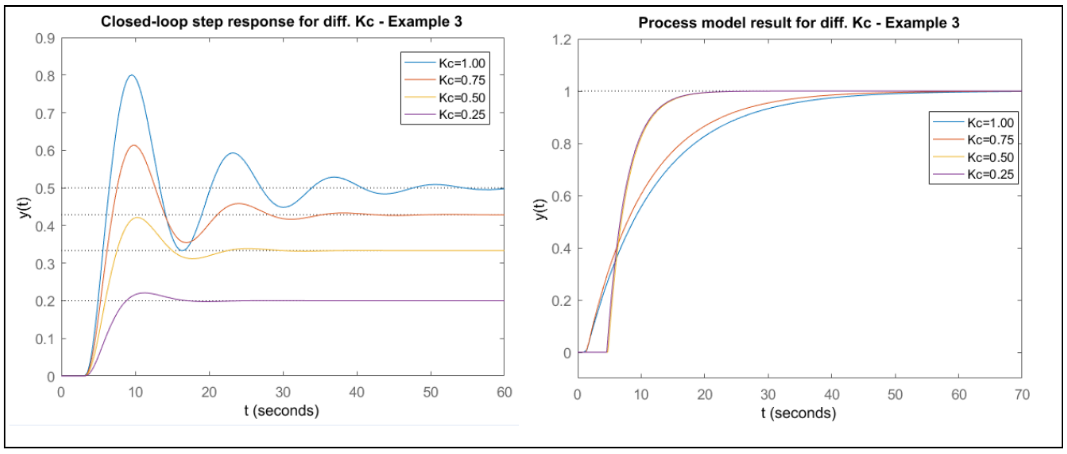


Figure 3-20: Third-order original process and its first-order approx. by CLC - Example 3

High values of K_c are associated with bigger oscillations and longer time-constant $\top_p$ of the approximated process. Lower K_c gave generally longer process delay τ_p. However, the delay seemed to reach highest value around $K_c = 0.5$, and so just to decrease again slightly toward the lower values as seen in Table 3.13. In respect to HR, the algorithm aligned the model toward the HR when low values of K_c were applied. However, effective delay τ_p deviated more from the HR's value as the K_c kept decreasing.

Table 3.13: Process model parameters by CLC – Example 3

K_c	K_p	$\top_p$	τ_p
1.25	0.9999	11.45	1.36
1.00	1	10.62	1.36
0.75	1	9.269	1.39
0.50	1	3.019	4.77
0.25	1	3.015	4.59
OL	**1**	**2.5**	**2.0**

Generally, the estimated model parameters $\top_p$ and τ_p converged towards 3.023 and 4.44 respectively, with $K_c = 0.17$ as the lowest before failed to produce solutions. On the other hand, the highest values of K_c before which the method gave unrealistic non-number solutions due to unstable dynamics of the feedback system was observed to be at $K_c = 1.72$, with $\top_p$ and τ_p corresponding to 12.24 and 1.41 respectively.

3.2.3.5 MF method

3.2.3.5.1 Method analysis

MF replaced the P-controller Equation 3.3 with the most commonly used industrial controller, PI-controller, for the purpose of counteracting the steady-state offset occurring during step change experimentation [8].

Similar to the CLC, the MF method utilized the identical cross-over frequency ω_c and estimated the ultimate open-loop magnitude from closed loop's ω_c for further back calculation of the process model parameters $\top_p$, K_p, τ_p as given in Equation (3.55), (3.57) and (3.56) respectively [8]. MF pointed also the inconsistence of the non-linear expression for the ultimate frequency (refer to Equation 8 in [6]), and updated it as given in Equation (3.54). The parameters K_c and T_i should be defined to reflect the particular system prior to the step change experimentation such the relative damping satisfies $0 < \zeta < 1$ [8].

3.2.3.5.2 Simulation results

Three examples are simulated using MF algorithm with varying controller gain K_c but similar value of integral time ($T_i = 3$). The integral time is randomly chosen without any prior knowledge of its effect on the processes to be simulated.

Example 1

In this example, a first-order model is approximated form the third-order process with time delay, as given by Equation (3.80), using the MF algorithm. Figure 3-21 shows step responses of the feedback system controlled by a PI-controller with various values of K_c as seen on the left. The respective step-responses of the first-order estimated model are given on the right side.

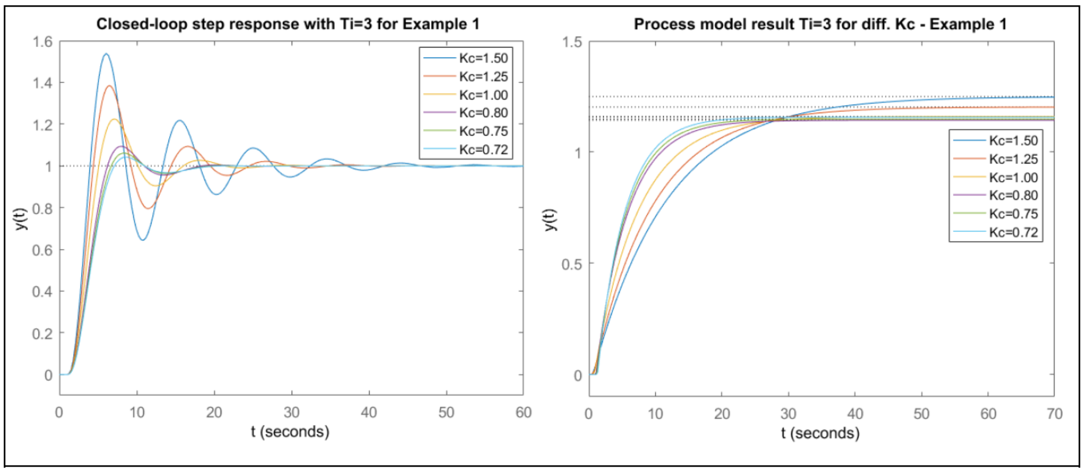


Figure 3-21: Third-order original process and its first-order approx. by MF - Example 1

As seen in the Table 3.14, the process model exhibited longer dominating time-constants when the proportional gain of the PI-controller increased. The delay time, however, decreased with increasing value of K_c. With respect to HR's estimated model, the lower values of K_c appeared to align identified model parameter toward HR's parameter values.

Table 3.14: Process model parameters by MF – Example 1

K_c	K_p	$\top_p$	τ_p
1.50	1.25	11.2	0.619
1.25	1.203	8.897	0.685
1.00	1.16	6.545	0.812
0.80	1.143	4.724	1.05
0.75	1.149	4.397	1.17
HR	**1**	**2.5**	**2.0**

In general, the identified model parameters $\top_p$ and τ_p converged towards 4.124 and 1.29 respectively at the lowest value of $K_c = 0.72$ before the algorithm failed to provide solutions due less overshoot (less than two oscillations) experienced by closed-loop response. On the other hand, the highest value of controller gain before which the algorithm produced non-number is $K_c = 2.10$ corresponding to 2.413, 52.62 and 0.293 for $\boldsymbol{K_p}$, $\top_p$ and τ_p respectively.

Example 2

This firth-order process model given by Equation (3.81) will provide indication in performance and robustness of the MF algorithm on higher order processes. Figure 3-22 on the left, shows the step-response of feedback system with various values of the controller gain K_c, and its respective step-response of the identified first-order models.

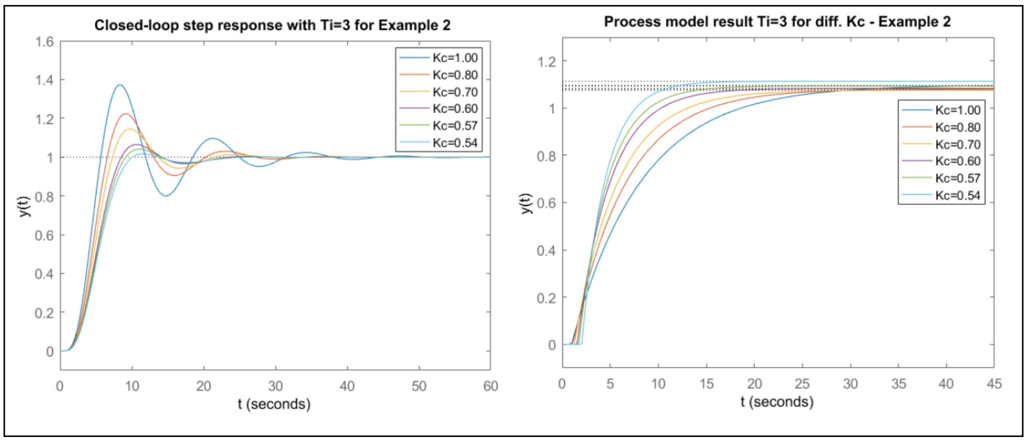


Figure 3-22: Firth-order original process and its first-order approx. by MF - Example 2

The time-constant $\top_p$ of the identified first-order process followed similar trend as in the example 1 above, whereby the increasing controller gain K_c was associated with longer time constant, while the process dead-time increased when K_c increased as seen in Table 3.15.

Table 3.15: Process model parameters by MF – Example 2

$\boldsymbol{K_c}$	$\boldsymbol{K_p}$	$\boldsymbol{\top_p}$	$\boldsymbol{\tau_p}$
1.00	1.097	7.278	0.967
0.80	1.081	5.486	1.12
0.70	1.076	4.452	1.26
0.60	1.083	3.408	1.55
0.57	1.093	2.968	1.73
HR	**1**	**1.5**	**3.5**

Generally, lower values of K_c moved the model toward HR's approximated parameters though failed to come closest enough to HR. The algorithm appeared to give increased magnitude of K_p and $\boldsymbol{\tau_p}$ at the lowest value of $Kc = 0.54$ corresponding to 1.157 and 4.124 respectively, while $\top_{\boldsymbol{p}}$ decreased unexpectedly. The highest values of K_c before which the method produced non-number solutions was observed to be at $K_c = 1.77$, with K_p, $\top_p$ and τ_p corresponding to 0.6192, 2.904 and 1.52 respectively.

Example 3

Figure 3-23 shows the step-response of a feedback system composed of a PI-controlled third-order process with relatively large time-delay, and the corresponding step response of first-order model approximated using MF algorithm. The controller's K_c used under simulation are as seen in Table 3.16, the integral time remains constant at $T_i = 3$ throughout the simulation.

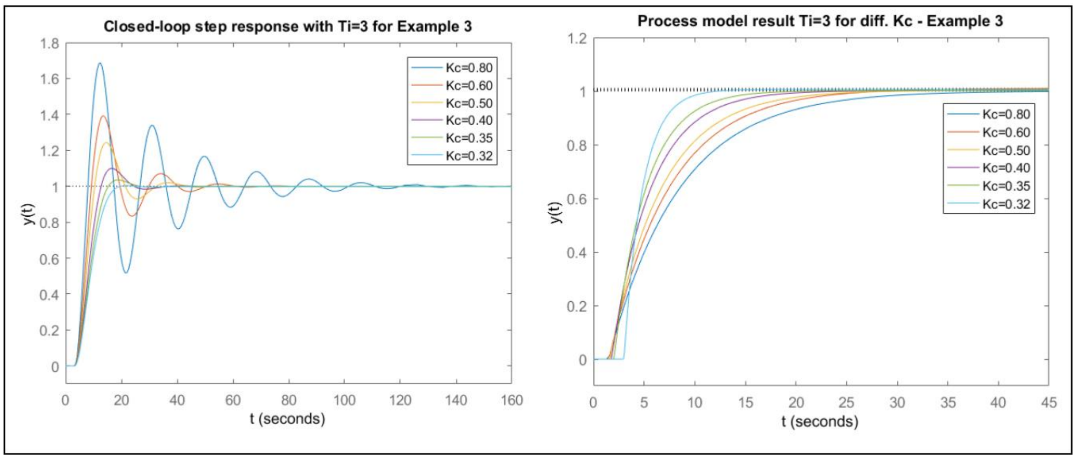


Figure 3-23: Third-order original process and its first-order approx. by MF - Example 3

Based on the simulation result, lower K_c values gave more accurate approximation as the identified parameters resembles the HR's calculated parameters. While the delay time τ_p showed some sort of increment with decreasing K_c, the time constant moved in the opposite direction.

Table 3.16: Process model parameters by MF – Example 3

$\boldsymbol{K_c}$	$\boldsymbol{K_p}$	$\boldsymbol{\top_p}$	$\boldsymbol{\tau_p}$
0.80	1	6.911	1.51
0.60	1.01	5.91	1.54
0.50	1.003	5.043	1.61
0.40	1.002	3.846	1.78
0.35	1.002	3.11	2.03
HR	**1**	**2.5**	**4.5**

The estimated parameters $K_p,\ \top_p$ and τ_p, converged toward 1.006, 1.835 and 3.01 at lowest value of K_c before which the algorithm's limitation on producing solution was reached. The highest value of K_c before which the closed-loop response became unstable was registered at $K_c = 1.00$ which corresponded to 0.5657, 1.126 and 2.66 for $K_p,\ \top_p$ and τ_p respectively.

3.2.4 Step Response SID for second order processes

This subsection presents analysis and simulation study on second-order process SID algorithms utilizing similar examples as the ones used during simulation for the first-order process model identification. The examples are simulated using the similar K_c values for all the four algorithms (JS, JSDR, DR and DR1) as given in Table 3.17

Table 3.17: Controller gains for simulation of different examples

Example	K_{c1}	K_{c2}	K_{c3}
1	2.00	1.50	1.00
2	2.00	1.50	1.00
3	1.00	0.75	0.50

3.2.4.1 JS and JSDR method

3.2.4.1.1 Method analysis

JS identifies second order unstable process model given as Equation (3.58) by stabilizing the corresponding closed-loop system using P-controller. By utilizing the stable closed-loop transfer function Equation (3.60), the process model parameters b_0, b_1, a_0 and a_1 are back calculated, which gives unstable process model when a_1> 0. [9]

JS's expression for step response in time domain (i.e. Equation A.2 in [9]) is incorrect, even though the method refers to the correct expression given by Equation (3.15). The mistake is also highlighted in [10].

DR substituted $\sqrt{1-\zeta^2}$ term into the $sin(\omega t + \phi)$ term and suggested an updated expression for the $\top_z$ as given in Equation (3.77), which gave rise to JSDR. In many cases, the JS is observed to yield complex solutions when $\top_z$ < 0, suggesting that $\top_z$ sign should be known in advance to avoid the complex solutions, as highlighted in [10]. DR suggested the real part of $\top_z$ as alternative solutions [10].

3.2.4.1.2 Simulation results

Example 1

The JS and JSDR algorithms are used to identify the second-order model from the third-order process given by Equation (3.80). Figure 3-24 on left, shows the step response of the feedback system made of the original process (third-order process) and P-controller of different proportional gain K_c, and the corresponding step-response of a second-order model produced by JS algorithm when $K_c = 1$ is presented on right.

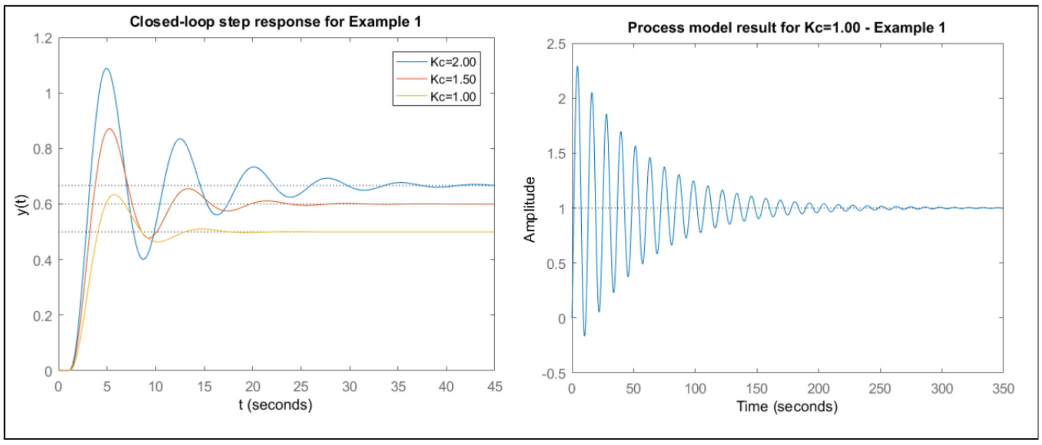


Figure 3-24: Third-order original process and its first-order approx. by JS - Example 1

Table 3.18 presents the resulted model parameters obtained from the JS on the left side, and JSDR on the right side of the table. As seen in the table, the JS algorithm produced negative magnitude of a_1 and hence unstable second-order model for K_c values of 2.0 and 1.5. This explains to why Figure 3-24 contains step response from the estimated process model only when $K_c = 1.0$ which was barely stable and exhibited no inverse response.

Table 3.18: Process model parameters by JS and JSDR respectively – Example 1

	JS				JSDR			
K_c	b_0	b_1	a_0	a_1	b_0	b_1	a_0	a_1
2.0	0.2327	0.2023	0.2327	-0.1625	0.2327	-0.1216	0.2327	0.4853
1.5	0.2498	0.3194	0.2498	-0.0924	0.2498	-0.1647	0.2498	0.6338
1.0	0.286	0.547	0.286	0.03511	0.286	-0.2559	0.286	0.838

The JSDR algorithm gave positive a_1 and hence stable second-order process for the all three values of K_c as seen in the Table 3.18. With JSDR, the estimated model showed inverse response dynamics as shown in Figure 3-25, reflecting the time-delay of the original system.

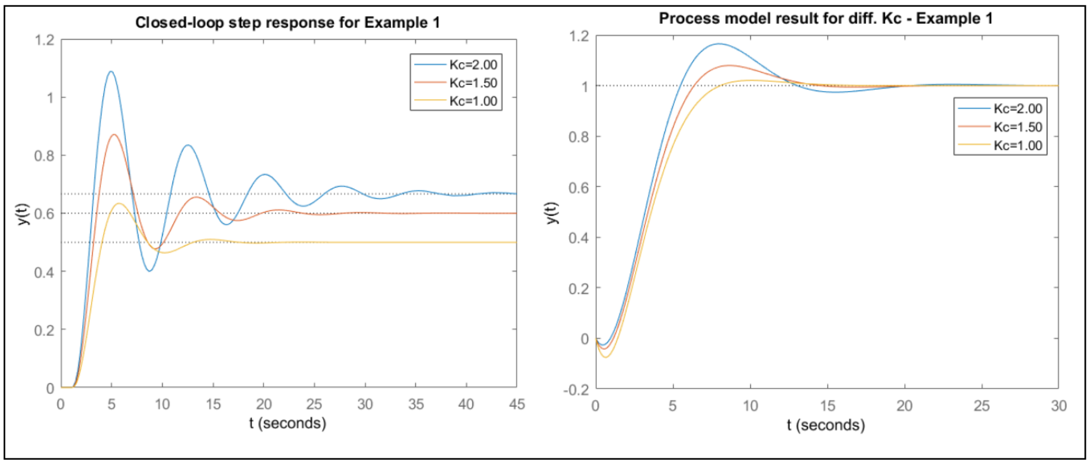


Figure 3-25: Third-order original process and its first-order approx. by JSDR - Example 1

Example 2

The performance of JS and JSDR is further tested by estimating a second-order model, this time from a firth-order process model of identical time-constant and with no time delay as given by Equation (3.81).

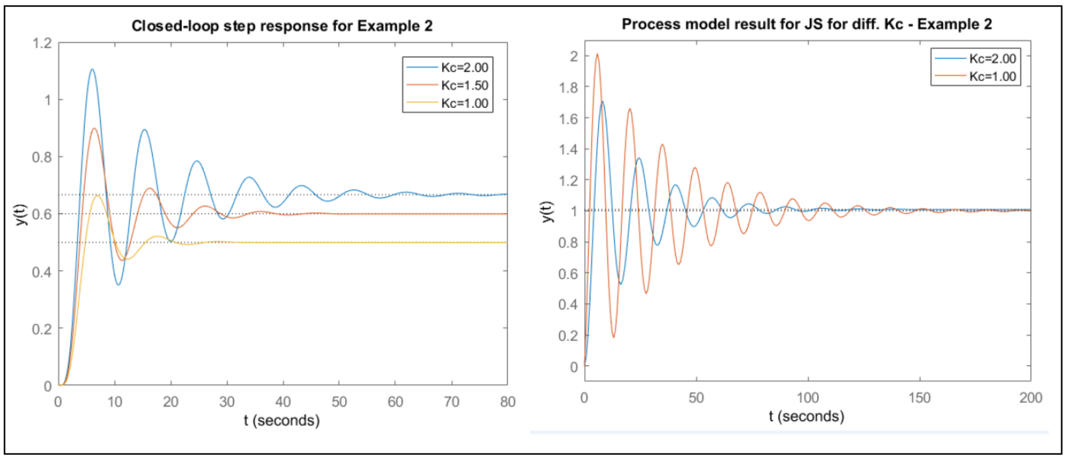


Figure 3-26: Firth-order original process and its first-order approx. by JS - Example 2

Figure 3-26 shows the step-response for both the feedback system composed of the original process and P-controller at different values of controller gain K_c, and the approximated model result from JS algorithm on the right.

Table 3.19: Process model parameters by JS and JSDR respectively – Example 2

	JS				JSDR			
K_c	b_0	b_1	a_0	a_1	b_0	b_1	a_0	a_1

2.0	0.1509	0.02261	0.1496	0.09006	0.1509	0.02261	0.1496	0.09006
1.5	0.1705	0.1817	0.1707	-0.02441	0.1705	-0.0825	0.1707	0.3719
1.0	0.1871	0.319	0.1871	0.0588	0.1871	-0.1301	0.1871	0.5079

As seen in Table 3.19, the JS algorithm managed to produce unstable second-order process model only when K_c=1.5. The two other values of K_c gave positive a_1, which in turn produced stable process dynamics. It should be noted that, Figure 3-26 shows only the stable process model estimated.

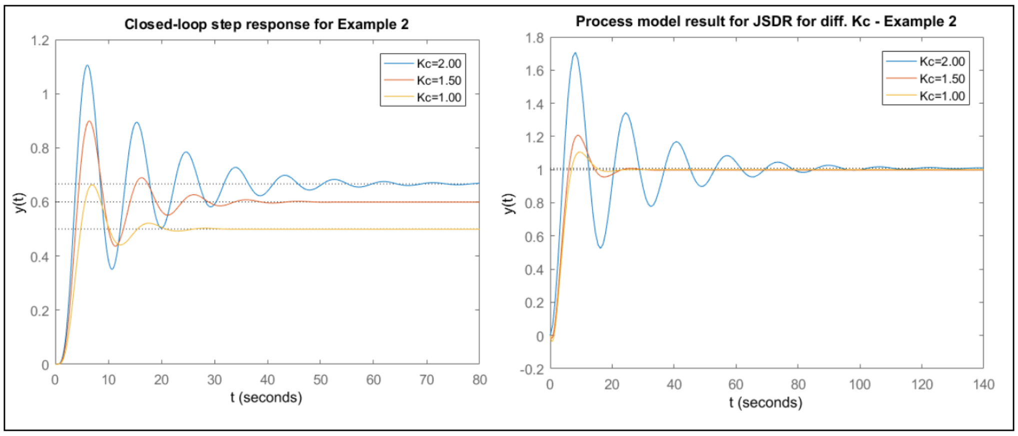


Figure 3-27: Firth-order original process and its first-order approx. by JSDR - Example 2

On the other hand, the JSDR approximated models with stable dynamics for all three chosen values of K_c. The models showed, in addition, inverse response of the system for 2 values of K_c as illustrated on the right graph in Figure 3-27. From the graph, it can be seen that the lower the K_c value, the more the oscillatory behaviour of the system.

Example 3

In this example, a third-order process characterized with significant time delay (Equation (3.82)) is reduced into second-order process model using JS and JSDR algorithms. Figure 3-28 gives graphical representation of the closed-loop response of the system in series with the P-controller with different sizes of K_c, on the left. Its corresponding identified second-order model is given on the right graph.

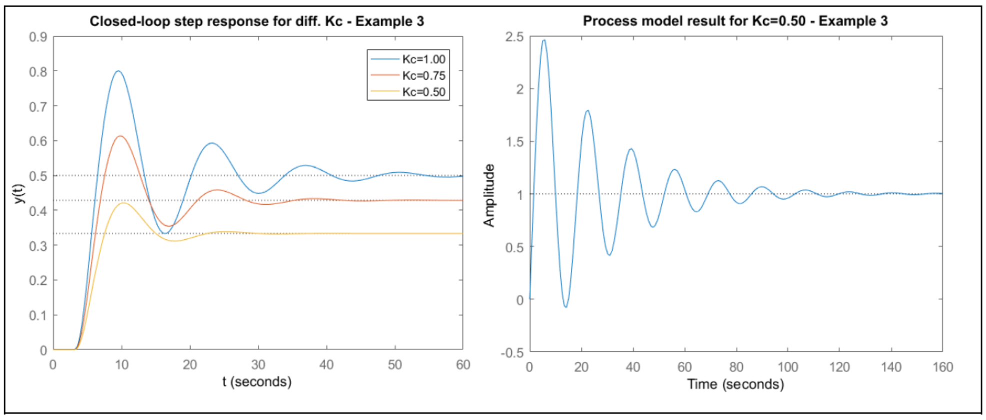


Figure 3-28: Third-order original process and its first-order approx. by JS - Example 3

Only at the lowest K_c (0.5), the JS algorithm managed to produce a stable second-order process model as shown in Figure 3-28. The model exhibited direct (no inverse response) but oscillatory behaviour due to positive values of b_1 as seen under JS section in Table 3.20. Based on the results presented in the table, the stability of the models approximated by JS algorithm increased with decreasing magnitude of the controller gain K_c.

Table 3.20: Process model parameters by JS and JSDR respectively – Example 3

	JS				JSDR			
K_c	b_0	b_1	a_0	a_1	b_0	b_1	a_0	a_1
1.0	0.107	0.2766	0.108	-0.1002	0.107	-0.1888	0.108	0.3652
0.75	0.1214	0.391	0.1215	-0.03427	0.1214	-0.2431	0.1215	0.4413
0.5	0.1402	0.6013	0.1402	0.07277	0.1402	-0.3523	0.1402	0.5496

Contrary to JS, the JSDR produced stable second-order models for all three values of controller gain as shown on Figure 3-29. The identified models experienced inverse response due to negative b_1 (refer to Table 3.20 under the JSDR column) as reflection to time-delay existed in the original process. Based on the Figure 3-29, the magnitude of the inverse response, appeared to increase along with increasing size of K_c.

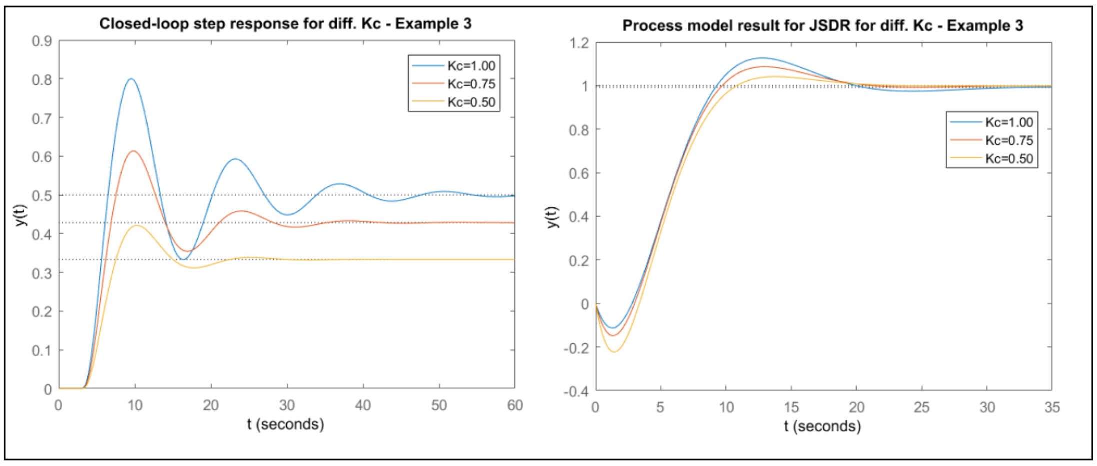


Figure 3-29: Third-order original process and its first-order approx. by JSDR - Example 3

3.2.4.2 DR and DR1 method

3.2.4.2.1 Method analysis

DR emphasized in generalizing the identification of second order process models (Equation (3.58)) proposed in [9] to also apply for other second-order models regardless of their stability behaviour [10]. The DR also correlated the time-domain response provided in [2] to better reflect response of the second-order process model as given by Equation (3.72).

The DR eliminated the complex solution problem experienced in [9] by proposing two alternative expressions for τ_z, Equation (3.77) and (3.78). Knowing the sign of τ_z before hand is therefore no longer necessary [10].

As highlighted earlier, DR and DR1 methods differs slightly in step response in time domain expression $y(t)$ as given by Equation (3.73). The DR1 is more sensitive to noise because of having different value of c as given by Equation (3.79) [10].

3.2.4.2.2 Simulation results

Example 1

This simulation for example 1 is meant to observe the performance of DR and DR1 algorithms in approximating a second-order model from the third-order process with delay as given by Equation (3.80). Figure 3-30 gives graphical view of the step-response of P-controlled third-order process on left, and its corresponding step-response of the identified model on right, using DR algorithm.

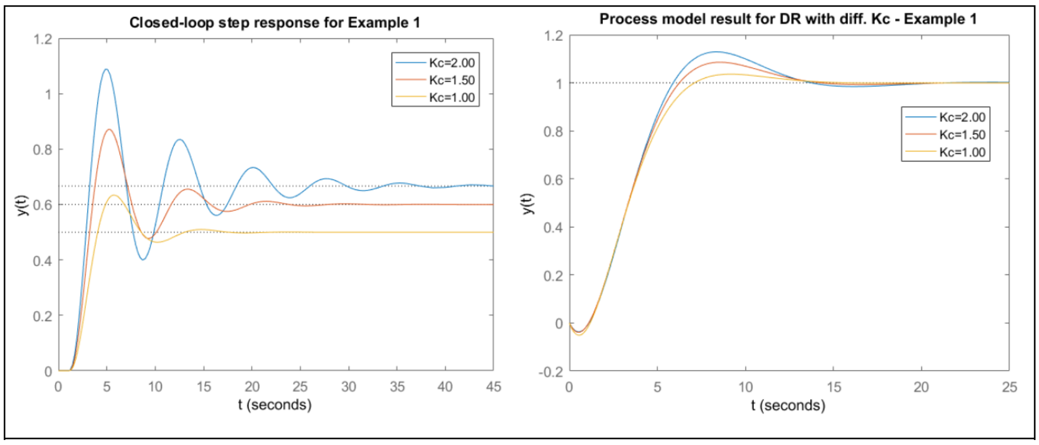


Figure 3-30: Third-order original process and its first-order approx. by DR - Example 1

For all the three K_c values used for step experiment, the DR algorithm produced stable, with oscillatory dynamics models, which also exhibited inverse response as seen in Figure 3-30. The observed dynamic characteristics of the identified model are due to positive and negative magnitude of parameters a_1 and b_1 respectively as seen under DR column in Table 3.21.

Table 3.21: Process model parameters by DR and DR1 respectively – Example 1

	DR				**DR1**			
K_c	b_0	b_1	a_0	a_1	b_0	b_1	a_0	a_1
2.0	0.2327	-0.1459	0.2327	0.5344	0.2327	-0.4455	0.2327	1.133
1.5	0.2498	-0.1586	0.2498	0.6218	0.2498	-0.5103	0.2498	1.152
1.0	0.286	-0.2009	0.286	0.783	0.286	-0.8474	0.286	1.429

Using DR1 algorithm on same original process model, gave also stable dynamics with inverse response for all three values of K_c as presented in Table 3.21. The step-response of the approximated models show however, no oscillatory behaviour as illustrated in Figure 3-31. The static gain of the identified model was identical for all three values of K_c.

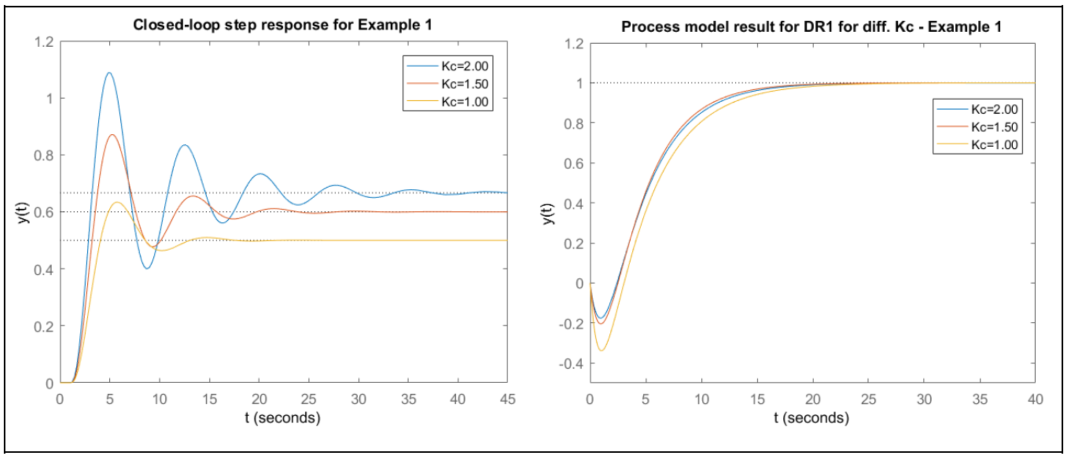


Figure 3-31: Third-order original process and its first-order approx. by DR1 - Example 1

Example 2

Example 2 involves an overdamped process of firth-order without input delay as given by Equation (3.81). The objective is to deduce second-order model approximation through step-response transient analysis using DR and DR1 algorithms. Figure 3-32 on left, shows the step-response of an output feedback system composed of the original process and P-controller of various gains K_c, and its corresponding step-response of the approximated model located on the right, using DR algorithm.

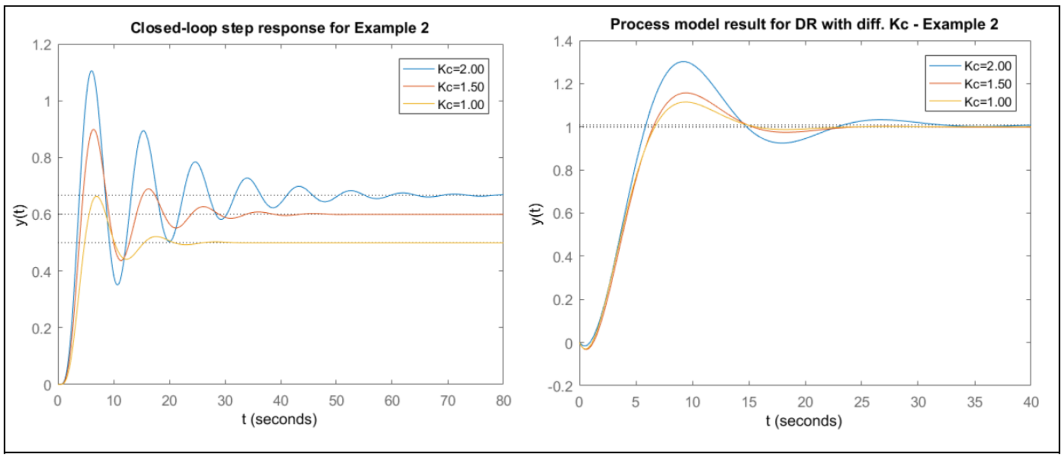


Figure 3-32: Firth-order original process and its first-order approx. by DR - Example 2

With DR algorithm, the identified model possessed stable dynamics with some sort of inverse response for all three values of K_c, due to positive and negative magnitude of parameters a_1 and b_1 respectively (refer to the DR column in Table 3.22), as illustrated graphically on the right graph in Figure 3-32. Based on the graph, the lower values of K_c produced process model with oscillatory dynamics.

Table 3.22: Process model parameters by DR and DR2 respectively – Example 2

	DR				DR1			
K_c	b_0	b_1	a_0	a_1	b_0	b_1	a_0	a_1
2.0	0.1509	-0.07532	0.1496	0.2859	0.1509	-0.3033	0.1496	0.7419
1.5	0.1705	-0.1168	0.1707	0.4233	0.1705	-0.5275	0.1707	1.039
1.0	0.1871	-0.1177	0.1871	0.4955	0.1871	-0.6251	0.1871	1.003

On the other hand, the DR1 algorithm produced process model with only overdamped response for all the values of K_c used. With DR1, the approximated model exhibited stable dynamics with inverse response as shown on the right graph in Figure 3-33.

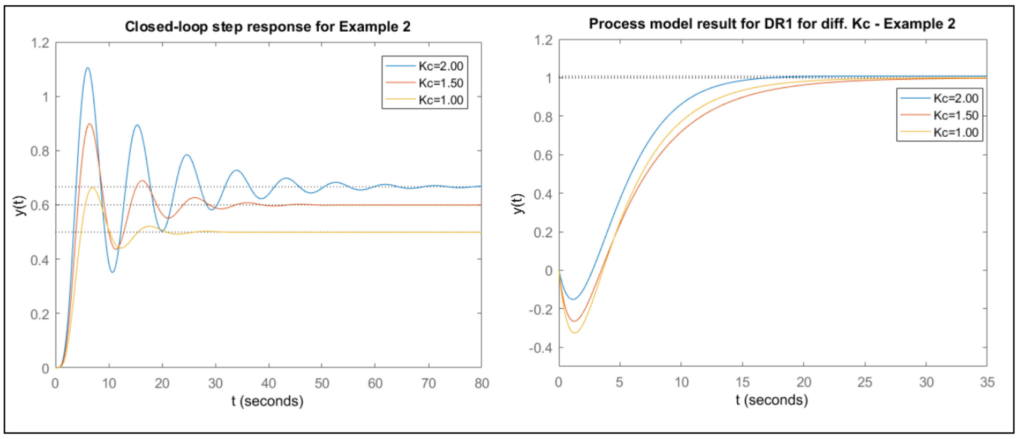


Figure 3-33: Firth-order original process and its first-order approx. by DR1 - Example 2

Both DR and DR1 produces model approximation possessing inverse response for all the chosen value of K_c. However, the magnitude of the inverse response experienced when using DR decreases with decreasing size of K_c, while for DR1, the inverse response increases and hence moves in opposite direction.

Example 3

In this simulation, a third-order process with significant large time delay is reduced into second-order process model using transient step-response analysis algorithms DR and DR1. The left graph in Figure 3-34 shows an output feedback response of the original process controlled by P-controller with various controller gains. Its corresponding step-response of the identified model using DR algorithm is presented on right graph in Figure 3-34.

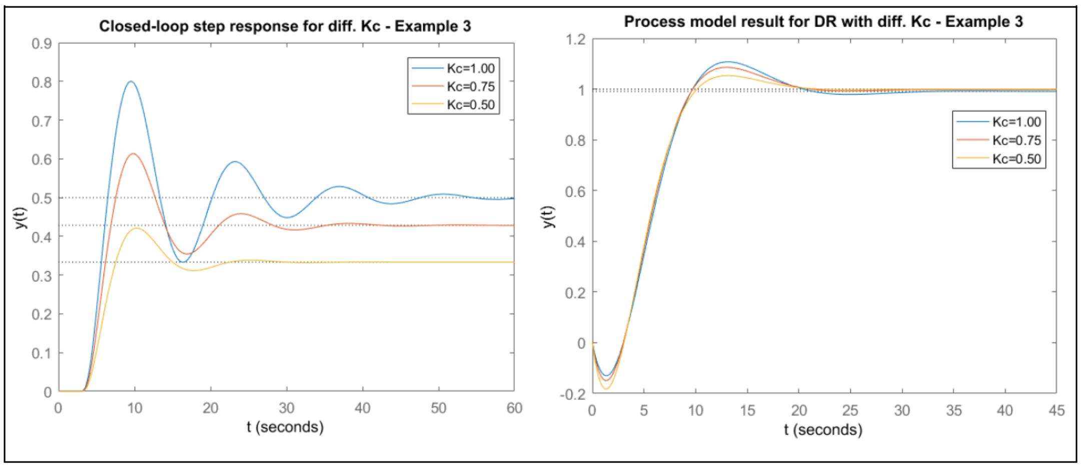


Figure 3-34: Third-order original process and its first-order approx. by DR - Example 3

The DR algorithm produced stable second-order process model containing inverse response, whereby high values of K_c were associated with increasing magnitude of the inverse response (b_1) as seen under DR column in Table 3.23. With DR algorithm and for all test values of K_c, the resulted model approximation experienced small overshoot and very slight oscillatory dynamics as illustrated on right graph in Figure 3-34.

Table 3.23: Process model parameters by DR and DR2 respectively – Example 3

	DR				**DR1**			
K_c	b_0	b_1	a_0	a_1	b_0	b_1	a_0	a_1
1.00	0.1047	-0.6481	0.1287	0.9712	0.1047	-4.359	0.1287	4.682
0.75	0.123	-0.393	0.1283	0.6091	0.123	-7.005	0.1283	5.568
0.50	0.1402	-0.3051	0.1402	0.5258	0.1402	13.3	0.1402	-6.279

Using DR1 algorithm on the same example produced quite different results as shown on the step-response of the identified model on the right graph in Figure 3-35. At K_c=0.5, the DR1 algorithm identified negative value for the parameter a_1 (refer to Table 3.23) and hence identified a model with unstable dynamics.

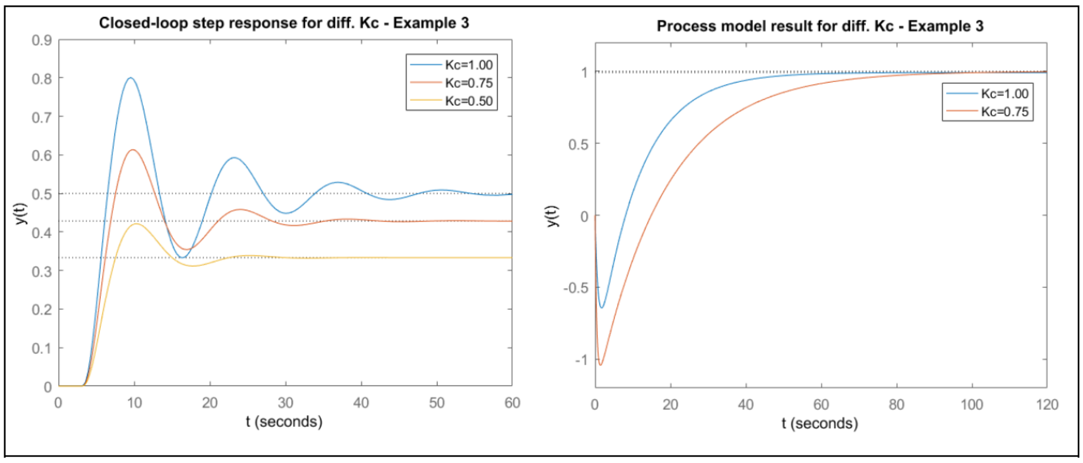


Figure 3-35: Third-order original process and its first-order approx. by DR1 - Example 3

3.3 Using Transient response SID algorithms in auto tuning PID

The closed-loop response system identification provides simple and effective on-line auto-tuning and adjustment of PID controller parameters to facilitate safe, flexible, and optimal process operation. During the on-line PID tuning, small step change in controller's reference value is applied to an output feedback system made of the process in question, and P-controller or in some cases PI-controller. The step change experimentation produces a closed-loop response as in Figure 3-6 from which the critical response values $y_{p1}, y_{m1}, y_{p2}, y_{\infty}, t_{p1}, t_{m1}$ and t_{p1} are deduced. To obtain the controller parameters, the below steps should be followed;[2, 31]

1. Estimation of closed-loop response model parameters $(\top, \zeta, K, K_2, t_z, b_0, b_1, a_0, a_1)$ from the critical response points $(y_{p1}, y_{m1}, y_{p2}, y_{\infty}, t_{p1}, t_{m1}$ and $t_{p1})$ using online transient response SID methods (YS, JR, JL, CLC, MF, JS, DR, DR1, etc.).
2. Back calculation of the process model parameters $(K_p, \tau, \top_p, b_0, b_1, a_0, a_1)$ from the estimated closed-loop response parameters.
3. Calculating PID controller parameters (K_c, T_i, T_d) from the identified parameters of the process model using known PID settings, for example SIMC, Nicholas Ziegler's settings, etc.

The focus of this section is on the third mentioned step, since the first two steps are already addressed in detail in section 3.2. Under this section, the SIMC's PID settings are used to calculate the PID-controller parameters and investigate the performance and robustness of the identification algorithms discussed in section 3.1.

The SIMC's settings are model based, and do not require to drive the closed-loop around its stability limit, which is clear advantage over both Good margin and Nichols Ziegler's methods. SIMC's settings are in addition more applicable and yield better loop stability as compare to Ziegler's method [24].

3.3.1 PID Controller

PID controller is an industrial standard and widely used controller, which stands for Proportional Integral and Derivative controller. The controller has received massive recognition due to its simplicity, robustness and flawless applicability within the control loop. The PID controller is an error-based type of controller whereby the control action depends on magnitude of the deviation between the controller's set point and the measured process output [18, 31].

$$u(s) = K_{pid}\frac{(T_i s + 1)(T_d s + 1)}{T_i s}e(s) \tag{3.85}$$

The P-term tends to react faster, though alone cannot bring the process into steady state since it amplifies the error. Opposite to the P-term, the I-term can alone bring the process into steady state. The D-term facilitates faster control while at the same time stabilize the system, even though the term has a tendency of amplifying measurement related error [18].

Equation (3.85) presents the PID transfer function in cascade (serial) form, whereby $e(s)$ denotes the error which is the difference between the reference value $r(s)$ and the process measured output $y(s)$. To tune PID-controller means defining contribution of each active controller-terms (P, I, D), by adjusting the value of K_{pid}, T_i and T_d respectively. Whereby; [24]

- K_{pid} is controller's proportional gain
- T_i is controller's integral time
- T_d is controller's derivative time

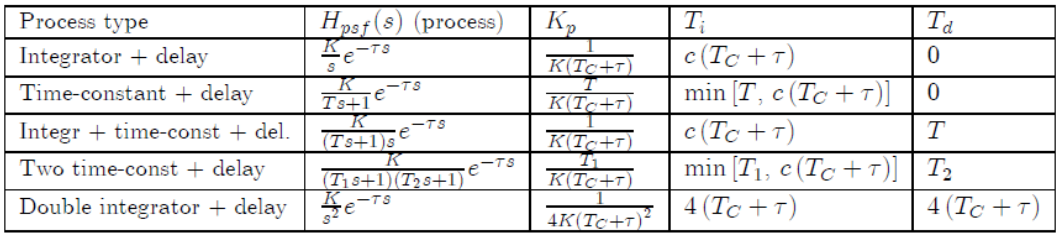
It follows that a PI-controller is special type of PID-controller in which the derivative part of the controller is deactivated (i.e. T_d = 0), while a P-controller has both Integral and Derivative terms deactivated (i.e. T_i=T_d=0) [34].

3.3.1.1 SIMC PID settings

The SIMC settings are widely used model-based PID tuning whereby the controller parameters are analytically derived from process models [18]. The method follows two steps as given below; [18, 32]

1. Obtain a first or a second order process model containing time delay
2. Derive model based controller parameters, with PI and PID-controller as starting point for the first-order and second-order process respectively.

Figure 3-36 gives an overview of SIMC PID settings for various types of the process [18]. The method recommends the value for the constant $c = 4$, while the tuning parameter T_c should be kept equivalent to the effective time delay τ of the process model [35].



Process type	$H_{psf}(s)$ (process)	K_p	T_i	T_d
Integrator + delay	$\frac{K}{s}e^{-\tau s}$	$\frac{1}{K(T_C+\tau)}$	$c\,(T_C+\tau)$	0
Time-constant + delay	$\frac{K}{Ts+1}e^{-\tau s}$	$\frac{T}{K(T_C+\tau)}$	$\min\left[T,\, c\,(T_C+\tau)\right]$	0
Integr + time-const + del.	$\frac{K}{(Ts+1)s}e^{-\tau s}$	$\frac{1}{K(T_C+\tau)}$	$c\,(T_C+\tau)$	T
Two time-const + delay	$\frac{K}{(T_1s+1)(T_2s+1)}e^{-\tau s}$	$\frac{T_1}{K(T_C+\tau)}$	$\min\left[T_1,\, c\,(T_C+\tau)\right]$	T_2
Double integrator + delay	$\frac{K}{s^2}e^{-\tau s}$	$\frac{1}{4K(T_C+\tau)^2}$	$4\,(T_C+\tau)$	$4\,(T_C+\tau)$

Figure 3-36: SIMC's PID-controller settings for different process types [18]

3.3.1.2 PID controller for oscillating systems

For oscillating processes, for example Equation (3.97), different settings other than presented on section 3.3.1.1 applies. Consider a step response of typical second-order transfer function model given by Equation (3.85), where τ, ζ and k represent time delay, damping ratio, and static gain of the process response respectively. The variable $\top_0$ determines speed of the system response, which is equivalent to the reciprocal of the natural frequency of the system ω_0 [34, 36].

$$h_p(s) = k\frac{e^{-\tau s}}{\top_0^2 s^2 + 2\zeta\top_0 s + 1} \tag{3.85}$$

The system's damping ratio determines the oscillatory behaviour of the response, whereby the system produce oscillating dynamics when $0 < \zeta < 1$. Under this condition, the PID-settings for the process are given as, [34]

$$h_c(s) = K_p(1 + \frac{1}{T_i s} + T_d s) \tag{3.86}$$

where the tuning parameters proportional gain (K_p), integral time (T_i), and derivative time (T_d) are given respectively as,

$$K_p = \frac{2\zeta\top_0}{k(T_c + \tau)} \tag{3.87}$$

$$T_i = 2\zeta\top_0 \tag{3.88}$$

$$T_d = \frac{\top_0}{2\zeta} \tag{3.89}$$

These PID-settings are only applicable for underdamped process, with damping ratio ranging from 0 to 1 as outlined above [34, 36].

3.3.1.3 Stability – Gain and Phase Margin

Gain and Phase Margins can be referred as measure for stability of feedback system. The margins give indication on how much the loop transfer function can withstand changes before the asymptotically stable system becomes marginally stable [24]. Consider a loop transfer function in complex domain given as

$$L(j\omega) = h_c(j\omega)h_p(j\omega) \tag{3.90}$$

where ω is the response frequency. Usually, the Gain Margin (GM) is expressed in decibel (dB) which is calculated by using Equation (3.91), while the Phase Margin (PM) is measured in degrees, and be calculated using Equation (3.92), whereby ω_{180} and ω_c are known as phase crossover and amplitude crossover frequency respectively [24].

$$GM\ [dB] = -|L(j\omega_{180})|\ [dB] \tag{3.91}$$

$$PM = 180° + \angle L(j\omega_c) \tag{3.92}$$

The GM indicates the maximum gain the loop can have before the system becomes unstable, while PM indicates the maximum phase lag function of the loop that can be reduced before the loop become unstable [24]. Generally, the larger the values (GM and PM), the better the stability, though with the expense of increased sluggishness in systems dynamics. The reasonable ranges are therefore given as, [24]

$$6dB \leq GM \leq 12dB \text{ and } 30° \leq PM \leq 60°$$

3.3.2 Auto-tuning PID using back calculated model parameters and SIMC

3.3.2.1 Simulation examples

Seven more examples (Equation (3.93) – (3.97)) are involved, in addition to the examples (Equation (3.80) – (3.82)) used earlier during individual method analysis in section 3.2, to fully investigate the performance of algorithms in on-line tuning of the PID-controller. The examples covers process models of various dynamics, low and higher-order, as well as models with significant dead time.

Example 4

Equation (3.93) is stable first-order process model with relatively large time delay $\tau = 4.5$ and dominant (i.e. $T > \tau$) time constant of $T = 9$.

$$h_p(s) = \frac{1}{9s+1}e^{-4.5s} \tag{3.93}$$

Example 5

Example 5 represents a stable first-order process model with dominant time constant of $T = 1$. The process consist of, however, minor time-delay of $\tau = 0.5$ as given by Equation (3.94).

$$h_p(s) = \frac{1}{s+1}e^{-0.5s} \tag{3.94}$$

Example 6

Example 6 is a simple and stable first-order process model with time constant T identical to the model's dead time τ as given by Equation (3.95) where $T = \tau = 1$.

$$h_p(s) = \frac{1}{s+1}e^{-s} \tag{3.95}$$

Example 7

Equation (3.96) represent a stable first-order process model experiencing a time delay τ larger than the process time constant T. The magnitude of process parameters T and τ are given as 1 and 2 respectively.

$$h_p(s) = \frac{1}{s+1}e^{-2s} \tag{3.96}$$

Example 8

A stable second-order process model containing two poles, both positioned on the left side of the complex plane, as given by Equation (3.97). The process experiences an input delay of τ =1.

$$h_p(s) = \frac{1}{9s^2 + 2.4s + 1}e^{-s} \tag{3.97}$$

3.3.2.2 Simulation study

The mode of simulation under this section follows the categories of the closed-loop transient response SID methods presented earlier in section 3.1.

Lowest-order model examples (Equations (3.93) – (3.96)) are simulated for PID auto-tuning using combination of the SIMC's PI-controller settings and the first-order step-response SID methods (YS, JR, JL, and CLC) only. The FOPDT's methods are, additionally used to simulate on-line PID-controller tuning for the all other process model examples. It should be noted that, the MF's method for FOPDT is purposely not included in PID-auto tuning simulation experiment due to the difficultness in finding a proper combination of the step experiment parameters K_c and T_i during model identification as highlighted in section 3.2.3.

For simulation of the second-order step-response SID methods (JS, JSDR, DR and DR1), only the models with higher model dynamics (Equation (3.80) – (3.82), and Equation (3.97)) are used. The simulation study was performed in MATLAB software using the programming codes available in Appendix B and C.

3.3.2.2.1 PID auto-tuning using FOPDT methods and SIMC settings

As mentioned above, the simulation study under this section is performed using the four discussed transient response SID algorithms, namely YS, JR, JL and CLC, in addition to the half-rule model reduction technique (HR), which functions as reference for performance measurement.

Example 1

Four various transient step-response identification algorithms (YS, JR, JL and CLC) are used to first, to estimate parameters for first-order process model approximated from the third-order process with input time delay as given in Equation (3.80) using test P-controller gain K_c=1.0, and then provide basis for designing and tuning PI-controller with the help of SIMC tuning rules.

Table 3.24: PI Auto-tuning results with different algorithms for K_c=1.0 - Example 1

	Without noise					With noise				
	K_p	$\top_p$	τ_p	K_{pi}	T_i	K_p	$\top_p$	τ_p	K_{pi}	T_i
YS	1	3.181	2.25	0.7079	3.1814	1.021	20.63	23.85	0.4236	20.626
JR	1	2.658	2.64	0.5043	2.6580	1.291	10.27	12.83	0.3099	10.266
JL	1	2.879	2.4	0.5999	2.8792	1.021	14.97	23.36	0.3138	14.967
CLC	1	3.099	2.44	0.6353	3.0989	1.021	-21.79	-15.18	0.7035	-121.4
HR	1	2.5	2.0	0.6250	2.5	1	2.5	2.0	0.6250	2.5

Table 3.24 gives overview of the identified model parameters together with corresponding PI-tuned parameter results obtained both without and with measurement noise induced into closed-loop system during step test experimentation. As seen in the Table 3.24, all the algorithm performed poorly when the output feedback system of the original process subjected to the noise. Based on the result, JR algorithm produced the least error among the four, though still very far from expected (HR) results.

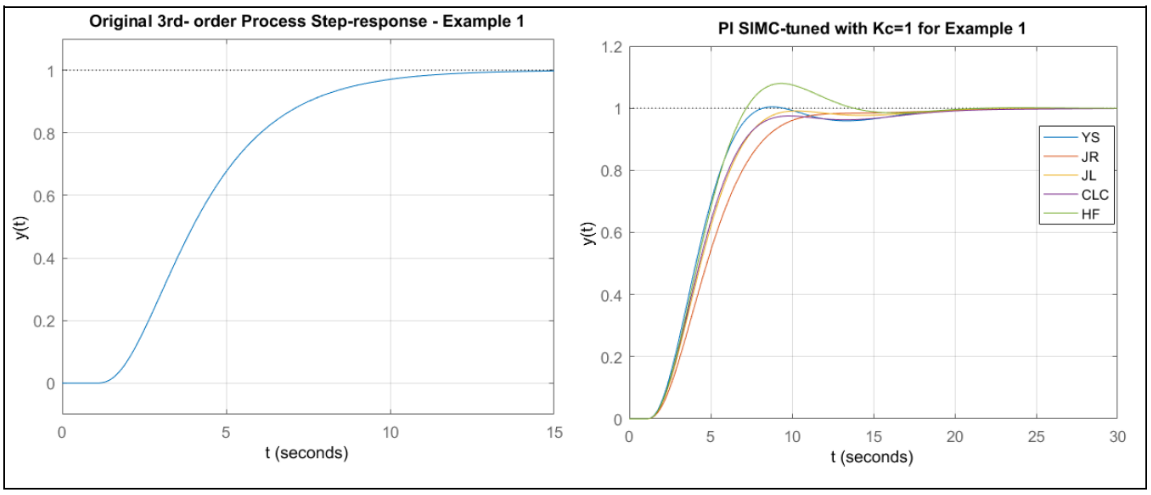


Figure 3-37: PI Auto-tuning result using different algorithms for K_c=1.0 - Example 1

Figure 3-37 gives graphical overview of the step-response of the original process on left, and the corresponding PI-tuned process results when no noise involved during set-point test. Generally, all four algorithms produced stable and relatively robust closed-loop response. HR and YS were the only algorithm that produced overshoot in the feedback loop as shown in Figure 3-37, while JR, JL and CLC produced overdamped closed-loop response. Among the algorithms, JL gave the best possible robust PI-parameters compared to others.

Example 2

In this example, the algorithms are supposed to estimate suitable PI-controller parameters for a firth-order process with no input delay as given in Equation (3.81). A controller gain of K_c= 0.75 is used during step test on the feedback loop, first without and then with white Gaussian noise induced. Table 3.25 presents model and PI-tuning parameters results with and without noise for the four algorithms in addition to HR.

Table 3.25: PID Auto-tuning with different algorithms for K_c= 0.75 - Example 2

	Without noise					**With noise**				
	K_p	$\top_p$	τ_p	K_{pi}	T_i	K_p	$\top_p$	τ_p	K_{pi}	T_i
YS	1	3.3136	2.8415	0.5831	3.3136	1.012	7.896	13.9	0.2808	7.8962
JR	1	2.56	3.66	0.3495	2.5595	0.8422	1.502	12.3	0.0728	1.5019
JL	1	2.79	3.17	0.4394	2.7900	1.0123	3.13	14.7	0.1054	3.1298
CLC	1	3.101	3.21	0.4835	3.1011	1.0123	-14.29	-5.3970	1.3075	-43.18
HR	1	1.5	3.5	0.2143	1.5	1	1.5	3.5	0.2143	1.5

As seen in Table 3.25, with noise, all four algorithms were unable to produce PI-tuning parameters that could provide smooth and stable closed-loop response. The worst was CLC algorithm, which returned both negative time-constant, negative input delay and negative integral time for the PI-controller.

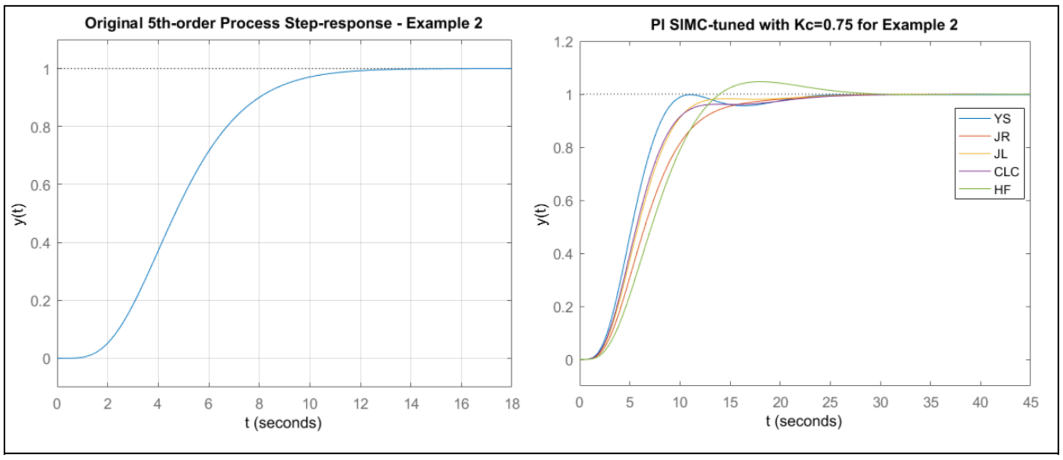


Figure 3-38: PI Auto-tuning result using different algorithms for K_c =0.75 - Example 2

However, when no noise involved, the algorithms regained their strength and were able to provide PI-settings that gave stable and robust closed-loop response as illustrated on the right graph in Figure 3-38. JR, JL and CLC algorithms produced overdamped feedback response, while YS experienced very slight oscillation. The speed of the closed-loop response produced varied depending on the algorithm used. YS appeared to have highest response speed, while JR had the lowest response speed among the four.

Example 3

The algorithms are further tested in designing and tuning robust PI-controller for a third-order process with large time delay as given by Equation (3.82). During step test, a controller gain of $K_c = 1.0$ is used to the feedback system composed of the original process and P-controller. The test was performed in two conditions, first without and later with Gaussian white noise subjected to the feedback loop. Table 3.26 shows identified model and PI-tuning parameters result for $K_c = 1.0$ both without and with the measurement noise.

Table 3.26: PID Auto-tuning with different algorithms for K_c=1.0 - Example 3

	Without noise					**With noise**				
	K_p	$\top_p$	τ_p	K_{pi}	T_i	K_p	$\top_p$	τ_p	K_{pi}	T_i
YS	1	3.948	4.72	0.4180	3.9478	1.0260	3.155	3.98	0.3863	3.1555
JR	1	2.801	5.33	0.2626	2.8009	1.238	3.102	4.6	0.2725	3.1024
JL	1	2.795	4.63	0.3016	2.7954	1.026	2.131	3.86	0.2689	2.1314
CLC	1	10.62	1.36	3.9124	10.6186	1.017	13.71	1.08	6.2391	8.6429
HR	1	2.5	4.5	0.2778	2.5	1	2.5	4.5	0.2778	2.5

When exposed to noise, the JR algorithms provided almost identical process time-day (τ_p = 4.6) as HR. Its identified time-constant, however, deviated more from HR when noise was involved. The rest of algorithms gave relatively poor result, with CLC the worst among them.

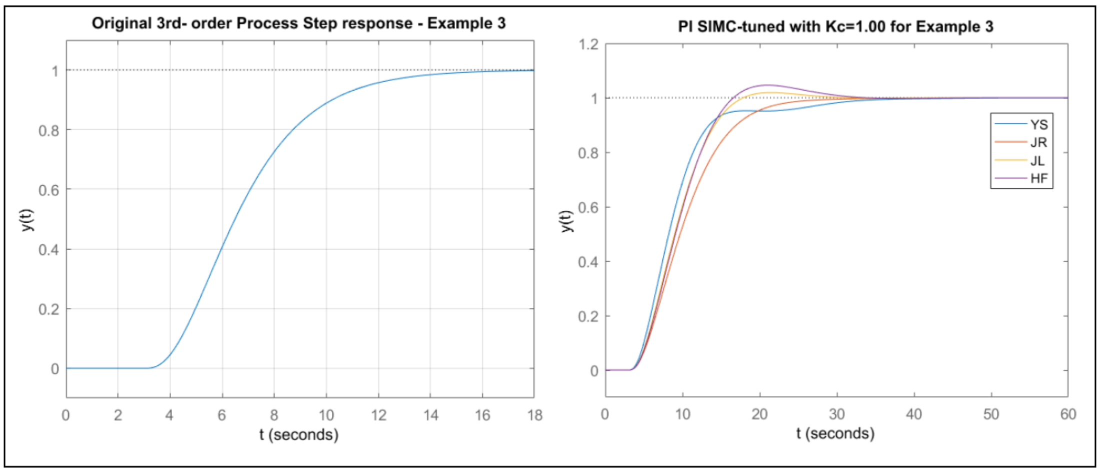


Figure 3-39: PI Auto-tuning results using different algorithms for K_c=1.0 - Example 3

Figure 3-39 shows step-response of the original process on the left, and its corresponding step-response (on the right) when PI-tuned without imposed noise, using different algorithms. Notice that when CLC was used to tune the PI-controller, the closed loop became destabilized and underwent unstable dynamics, and therefore not included in the graph. A slight overshoot was observed when the controller was tuned with parameters from JL and HF (also referred as HR). The fastest response speed was attained when the system was tuned with YS parameters, while the slowest response was produced when JR is used. In general, all the three algorithms (YS, JR and JL) produce robust controlled system response, which reached the steady state almost simultaneously.

Example 4

The algorithms are supposed, in this example, to design and tune PI-controller for a first-order model as given in Equation (3.93). Figure 3-40 shows step-response of the process to be controlled.

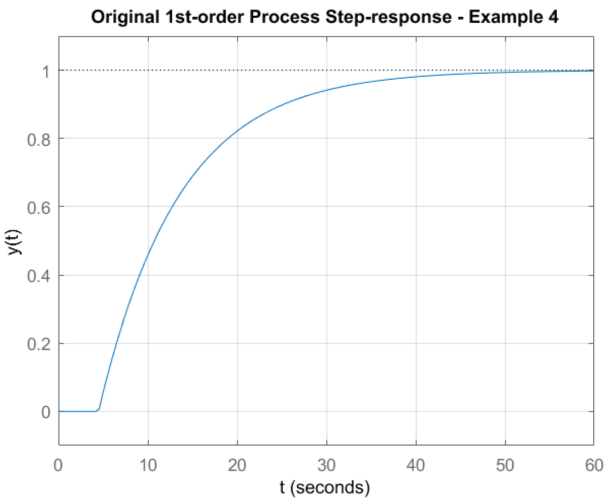


Figure 3-40: Step response of the original process - Example 4

Two different controller gain values (2.0 and 1.0) were used for all four algorithms during step test experiment. The model and PI-parameters results are as presented in Table 3.27. Generally, when using JL algorithm, the identified model parameters were closest to the real process parameter (HR), regardless of the size of the controller gain K_c. The CLC model parameter results deviated most in both values of K_c, however, at K_c=1.0 the CLC results were noticeably improved. Based on the Table 3.27, the identified time delay of the approximated process is favoured when K_c=1.0.

Table 3.27: PI auto-tuning with different algorithms for K_c=2.0 and K_c=1.0 - Example 4

	$K_c = 2.00$					$K_c = 1.00$				
	K_p	$\top_p$	τ_p	K_{pi}	T_i	K_p	$\top_p$	τ_p	K_{pi}	T_i
YS	0.9998	9.32	4.79	0.9785	9.3197	1	9.036	3.78	1.1946	9.0360
JR	0.9998	8.704	4.9	0.8892	8.7035	1	8.222	4.59	0.8949	8.2221
JL	0.9998	9.005	4.52	0.9969	9.0047	1	8.952	4.53	0.9890	8.9519
CLC	0.9998	25.01	1.41	8.8991	11.2422	1	10.07	3.71	1.3554	10.0651
HR	1	9	4.5	1	9	1	9	4.5	1	9

Figure 3-41 presents step-response of the feedback system composed of the original process and PI-controller tuned using different transient response SID algorithm for K_c=2.0 and K_c=1.0 respectively. At K_c=2.0, the PI-tuning parameter obtained from the CLC algorithm could not stabilize the feedback system, and therefore not seen in the graph. The YS, JR and JL provided robust set-point tracking control with minimal overshoot that brought the system quickly into steady state. The JL and HF (HR) provided almost similar response when K_c=2.0.

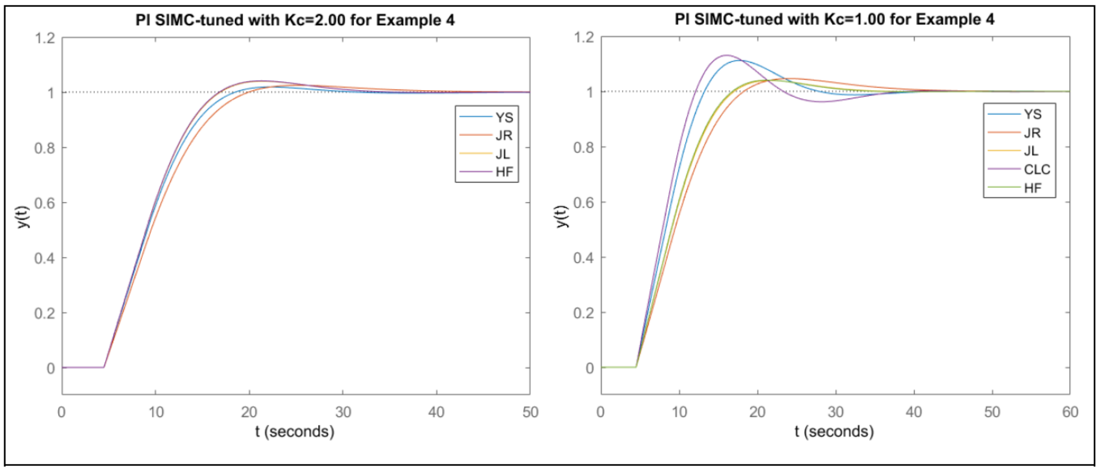


Figure 3-41: PI auto-tuning with different algorithms for K_c=2.0 and K_c=1.0 - Example 4

Noticeable overshoot, and slight oscillations were observed when YS and CLC algorithm were used for K_c=1.0. The CLC exhibited, in addition, fastest response-speed among the four, while JR was the slowest. In general, all four algorithms stabilized the system and brought the system into the steady state almost simultaneously. The most robust settings for this particular example at K_c=1.0 were originated from HF and JL algorithm.

Example 5

This example represents a first order process containing input delay, with dominating time constant. Figure 3-42 shows step-response of the original process as given by Equation (3.94).

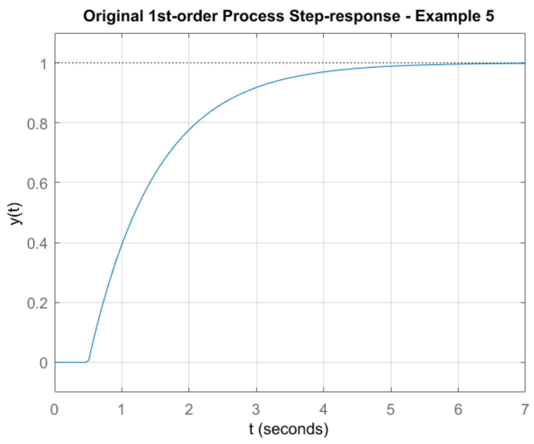


Figure 3-42: Step response of the original process - Example 5

The process is to be controlled in a feedback system with PI-controller tuned by using YS, JR, JL and CLC identified model and PI-parameters. Table 3.28 gives overview of the parameter results obtained from all the algorithms when using test controller gain of K_c=2.5 and K_c=2.0 respectively.

Table 3.28: PI auto-tuning with different algorithms for K_c=2.5 and K_c=2.0 - Example 5

	$K_c = 2.50$					$K_c = 2.00$				
	K_p	$\top_p$	τ_p	K_{pi}	T_i	K_p	$\top_p$	τ_p	K_{pi}	T_i
YS	0.9977	1.138	0.603	0.9457	1.1379	0.9996	1.047	0.535	0.9789	1.0472
JR	0.9977	1.054	0.583	0.9059	1.0536	0.9996	0.012	0.574	0.8895	1.0211
JL	0.9977	1.067	0.538	0.9941	1.0668	0.9996	1.012	0.507	0.9973	1.0119
CLC	0.9977	1.107	0.576	0.9625	1.1068	0.9996	1.059	0.552	0.9602	1.0588
HR	1	1	0.5	1	1	1	1	0.5	1	1

Based on the result presented in the table, all four algorithms produced better model parameter result when K_c=2. The model parameters obtained from JL for both values of K_c were the most closest to the real process parameter (HR). A general finding for this particular example was that, all the four algorithms produced sufficiently good result regardless of the K_c value used.

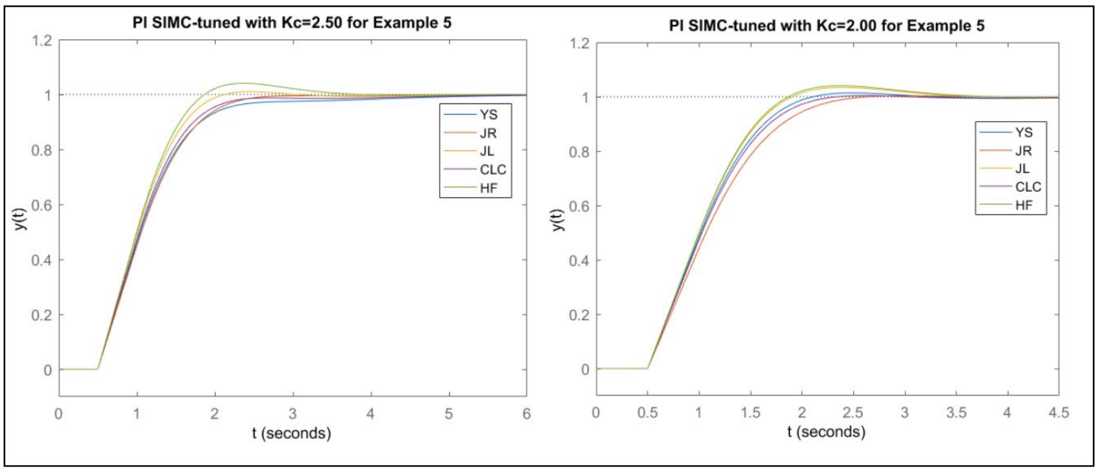


Figure 3-43: PI auto-tuning with different algorithms for K_c=2.5 and K_c=2.0 - Example 5

Figure 3-43 shows step-response of the PI-tuned feedback system with different algorithms at K_c=2.5 and K_c=2.0 respectively. All the algorithms led to stable and robust system control with good set-point tracking behaviour for both values of K_c. Notice that, only HF and the JL algorithm produced response with slight overshoot in both condition. In general, the algorithms settled the system faster into steady state when K_c=2.0.

Example 6

In this simulation example, a PI-controller is designed and tuned to control a first-order process with identical time-constant and input delay. Figure 3-44 shows step-response of the process to be controlled as given by Equation (3.95).

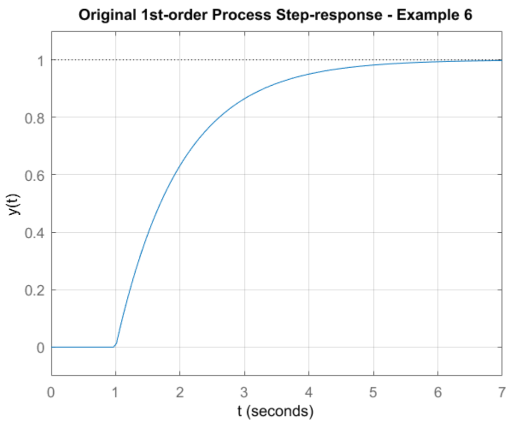


Figure 3-44: Step response of the original process - Example 6

Four different SID algorithms (YS, JR, JL and CLC) were applied during step experiment with K_c=1.0 and K_c=0.75. Table 3.29 presents the identified model and corresponding PI-setting results for both values of K_c.

Table 3.29: PI auto-tuning with different algorithms for K_c=1.0 and K_c=0.75 - Example 6

	$K_c = 1.00$					$K_c = 0.75$				
	K_p	$\top_p$	τ_p	K_{pi}	T_i	K_p	$\top_p$	τ_p	K_{pi}	T_i
YS	1.001	1.126	0.934	0.6023	1.1255	1	1.162	0.895	0.6496	1.1624
JR	1.001	0.9587	1.15	0.4177	0.9587	1	0.935	1.16	0.4015	0.9350
JL	1.001	0.9693	0.967	0.5009	0.9693	1	1.014	1.02	0.4971	1.0139
CLC	1.001	2.159	0.387	2.789	2.1595	1	1.097	1.03	0.5329	1.0965
HR	1	1	1	0.5	1	1	1	1	0.5	1

Based on the result in the table, the identified model parameters obtained when using CLC were very poor and experienced largest deviation from the real parameters when K_c=1. As a result, the PI-controller with CLC setting gave unstable system dynamics. The YS and JR algorithm had better model parameters when K_c=1, which in turn gave stable and robust feedback response as shown in Figure 3-45.

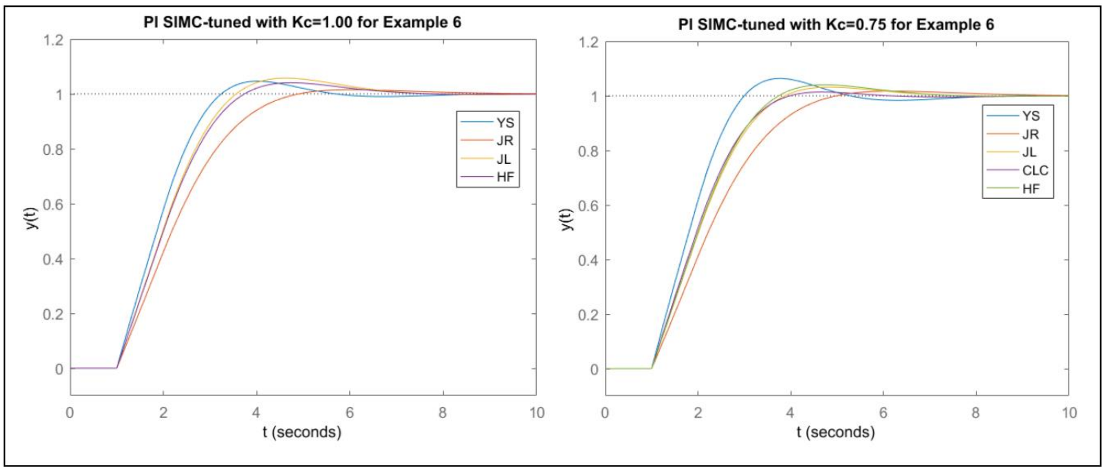


Figure 3-45: PI auto-tuning with different algorithms for K_c=1.0 and K_c=0.75 - Example 6

Generally, JL provided best PI-setting for the process when K_c=1.0, while CLC failed even to stabilize the systems. When K_c= 0.75, the CLC together with JL algorithm provided the most robust control settings among the four algorithms. The YS settings on other hand, gave response with some minor overshoot for both values of K_c.

Example 7

A first-order process with significant large time delay (as twice as long as its time-constant) given by Equation (3.96), is to be controlled and tuned by PI-controller containing parameter settings obtained by using YS, JR, JL and CLC algorithms. Figure 3-46 shows step-response of the original process.

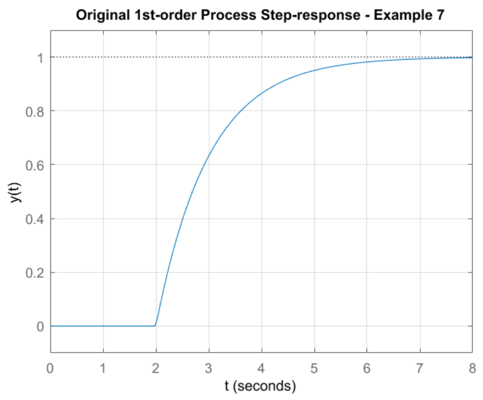


Figure 3-46: Step response of the original process - Example 7

Two different values of K_c (1.25 and 0.75) are applied to all four algorithms during step experiment, and the results for both values of K_c are as presented in Table 3.30.

In both cases, the JL algorithm provided best possible model parameters τ_p and τ_p, closest to the real (HR) process parameters, which in turn produced almost similar PI-settings as obtained when using half-rule technique. In contrast to JL, the CLC algorithm provided poorest model and PI-parameter setting for both values of K_c. At K_c=1.25, the CLC could not provide suitable enough setting for stable and robust closed-loop response.

Table 3.30: PI auto-tuning with different algorithms for K_c=1.25 and K_c=0.75 - Example 7

	$K_c = 1.25$					$K_c = 0.75$				
	K_p	$\top_p$	τ_p	K_{pi}	T_i	K_p	$\top_p$	τ_p	K_{pi}	T_i
YS	1.002	1.585	2.09	0.3792	1.5854	1.002	1.441	1.81	0.3977	1.4406
JR	1.002	1.126	2.2	0.2553	1.1264	1.002	0.9647	2.35	0.2046	0.9647
JL	1.002	0.9907	1.93	0.2566	0.9907	1.002	0.9829	1.91	0.2567	0.9829
CLC	1.002	3.258	0.76	2.1408	3.258	1.002	1.045	1.196	0.2660	1.0453
HR	1	1	2	0.25	1	1	1	2	0.25	1

As shown in Figure 3-47, the PI-settings with YS algorithm gave the fastest response in both cases. The algorithm exhibited also some sort of overshoot when K_c=0.75. Some small overshoots were also detected when JL algorithm was used for both values of K_c as seen in Figure 3-47. The JR algorithm, unlike others, produced PI-setting with slowest response dynamics, and with no overshoot for both values of K_c, without compromising the time taken to bring the system to steady state.

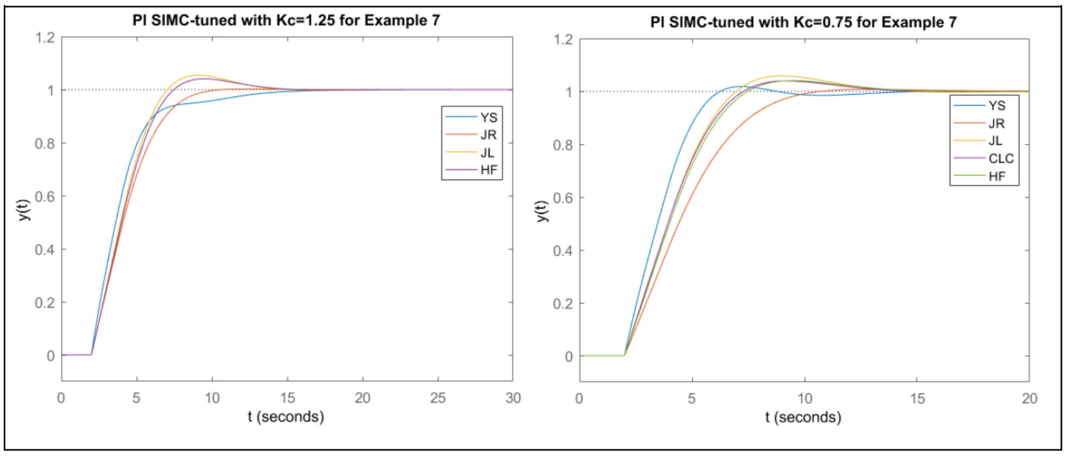


Figure 3-47: PI auto-tuning with different algorithms for K_c=1.25 and K_c=0.75 - Example 7

Example 8

A second-order system (Equation (3.97)) with oscillating dynamics is to be auto-tuned with PI-controller composed of settings obtained from YS, JR, JL and CLC algorithms. Figure 3-48 shows open-loop response of the original process. The process has two complex poles, and input time delay of magnitude 1. Two test gain values are used during step test, and the results are as presented in Table 3.31.

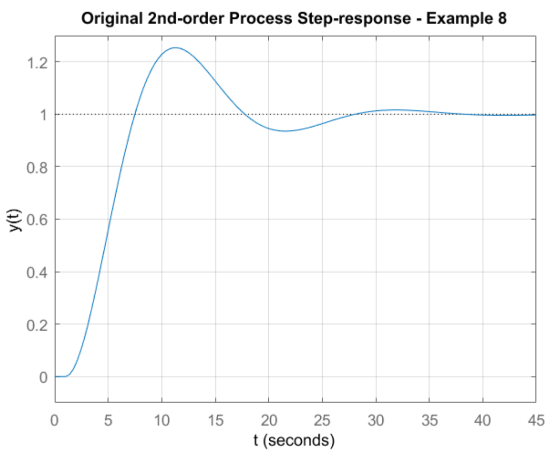


Figure 3-48: Step response of the original process - Example 8

Note that, none of the algorithms managed to provide similar or close enough identified model parameters $\top_p$ and τ_p as that of HR. In addition to that, the deviation between the identified parameters across the algorithms were unusually large. The CLC algorithm gave the worst identified model parameters which led to completely poor PI-settings and hence unstable system dynamics for both values of K_c as presented in Table 3.31. The identified model result from YS, JR and JL algorithms in both cases (K_c=1.25 and K_c=1.00) indicated that the first order approximation of the original process has $\top_p$ and τ_p values of approximately 3 and 5 respectively.

Table 3.31: PI auto-tuning with different algorithms for K_c=1.25 and K_c=1.0 - Example 8

	$\boldsymbol{K_c = 1.25}$					$\boldsymbol{K_c = 1.00}$				
	$\boldsymbol{K_p}$	$\boldsymbol{\top_p}$	$\boldsymbol{\tau_p}$	$\boldsymbol{K_{pi}}$	$\boldsymbol{T_i}$	$\boldsymbol{K_p}$	$\boldsymbol{\top_p}$	$\boldsymbol{\tau_p}$	$\boldsymbol{K_{pi}}$	$\boldsymbol{T_i}$
YS	1	4.151	4.7	0.4412	4.1514	0.9999	3.916	4.93	0.3970	3.9162
JR	1	3.124	4.99	0.3129	3.1238	0.9999	2.713	5.5	0.2445	2.7132
JL	1	2.994	4.39	0.3409	2.9937	0.9999	2.655	4.82	0.2755	2.6551
CLC	1	33.84	0.452	37.4189	3.6174	0.9999	33.49	0.452	37.0475	3.6163
HR	1	1.8	1.6	0.5625	1.800	1	1.8	1.6	0.5625	1.800

When it comes to PI-controller auto-tuning and control of the process, all the three algorithms (YS, JR and JL) provided better and more robust closed-loop system response compared to response observed when using PI-controller tuned with HR settings. With HR (also referred as HF) settings, the response evinced oscillations and was barely stable as seen in Figure 3-49.

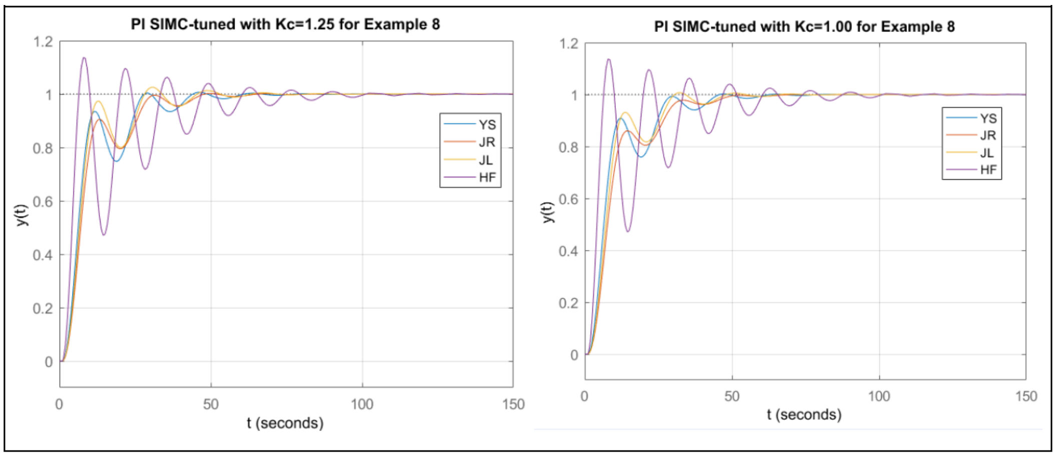


Figure 3-49: PI auto-tuning with different algorithms for K_c=1.25 and K_c=1.0 - Example 8

Both YS's, JR's and JL's PI-settings demonstrated oscillating response, and took quite long time to bring the system to steady state. The closed-loop responses for all the three algorithms shows similar patterns, however, the JR and JL are the most robust among the three.

3.3.2.2.2 PID auto-tuning using SOPDT methods

Simulation study using four process examples is used to assess the performance of the discussed second-order SID algorithms (JS, JSDR, DR and DR1) on auto-tuning PID-controller. A step experiment was performed under two conditions, first without, and later with white Gaussian noise imposed into closed-loop system composed of the process and P-controller at a given controller gain (K_c) size. Thereafter, the SID algorithms were applied to identify the model and PID-parameters for the process in question. The resulted PID-settings were then applied to control the original process. Additionally, half-rule model-reduction technique (HR) was used to tune the PID-controller. These HR's settings will serve as reference for performance measurement.

Example 1

In this example, the algorithms are supposed to auto-tune a PID-controller for a third-order process with input time delay as given by Equation (3.80). Figure 3-50 shows step-response of the original process on the left, and its corresponding step-response, on the right graph, of the noise induced closed-loop controlled by P-controller with K_c=2.0.

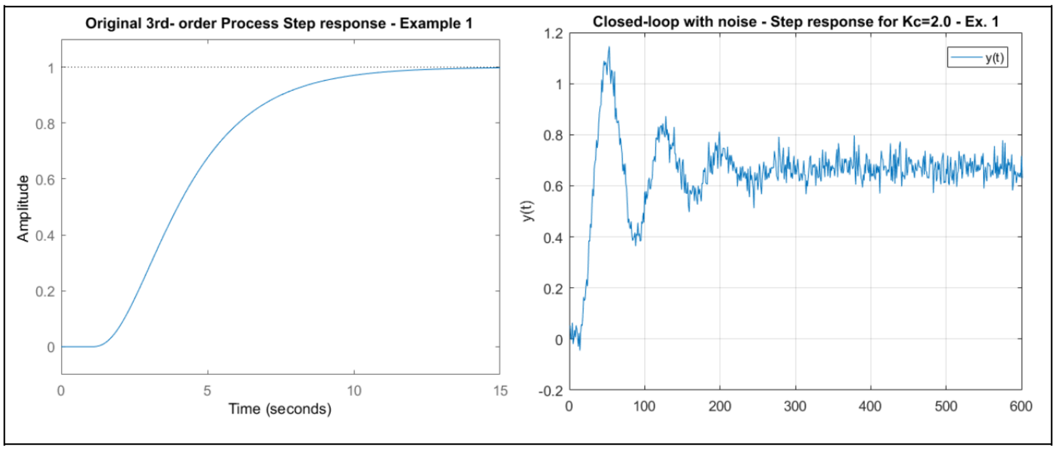


Figure 3-50: Process step-response and corresponding feedback response with P-controller at K_c=2.0

Table 3.32 presents the result obtained from model identification and PID parameter auto-tuning. Based on the identified model results, the JS algorithm yielded negative damping ratio both when with and when without noise, leading to unstable process dynamic in both cases. The PID-setting obtained by using the JS algorithm also failed to provide stable closed-loop dynamics when applied to the original process. Similar to JS, the DR1 algorithm gave also negative damping ratio and hence unstable dynamics but only when exposed to measurement noise.

Table 3.32: PID Auto-tuning SOPDT with different algorithms for K_c=2.0 - Example 1

	Without noise					**With noise**				
	⊤0	ζ_p	K_{pid}	T_i	T_d	⊤0	ζ_p	K_{pid}	T_i	T_d
JS	2.0593	-0.213	0.4487	-4.83	2.059	1.7907	-0.788	-1.921	0.5244	9.7063
JSDR	2.0593	0.5631	1.87	2.32	1.83	1.7907	1.2551	1.1007	3.6058	0.8893
DR	2.0593	0.5936	1.79	2.44	1.73	1.7907	1.3272	1.1146	3.9391	0.8141
DR1	2.0593	1.1867	1.28	4.89	0.868	1.7907	-2.827	0.8925	-10.13	-0.317
HR	1.5811	1.0277	0.8	2.0	1.25	1.5811	1.0277	0.8	2.0	1.25

When no noise involved, the JSDR, DR and DR1 gave stable second-order model approximation of the original process with similar static gain. Their estimated models displayed also inverse response as shown on the left graph in Figure 3-51. The JSDR and DR resulted models had damping ratio within $0 < \zeta_p < 1$ range, hence exhibited some sort of oscillation and overshoot. Similar to the HR, the DR1 algorithm identified the approximated model with $\zeta_p > 1$ (refer to *without noise* column in Table 3.32) leading to overdamped response.

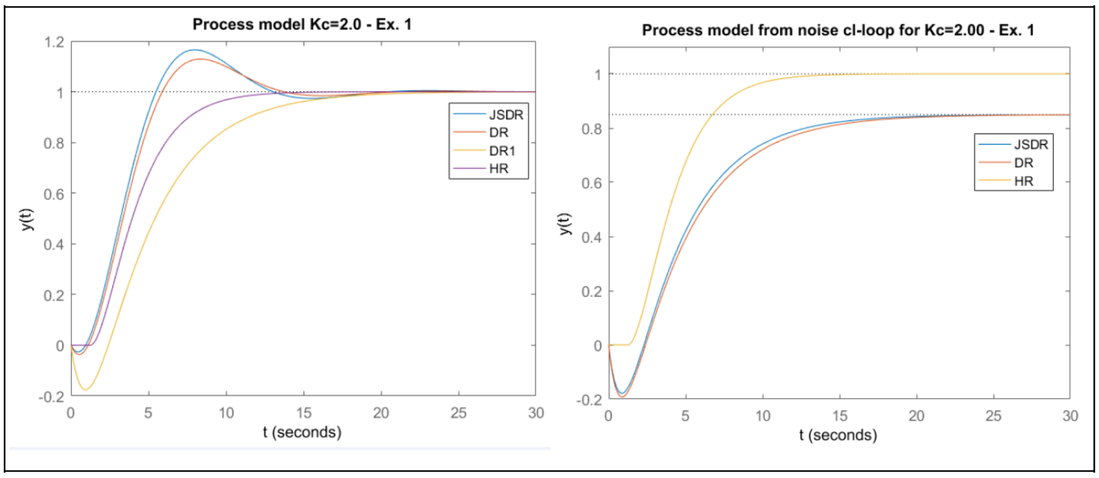


Figure 3-51: Step-response of identified model without and with noise – Example 1

Using the DR1 algorithm to auto-tune the PID controller for the process when the random Gaussian noise involved, gave negative integral and negative derivative time as seen in Table 3.32, and hence invalid PID-settings for the process. The JSDR and DR algorithms, however, could provide PID-settings with more stable response dynamics when subjected to the random noise as illustrated in Figure 3-52.

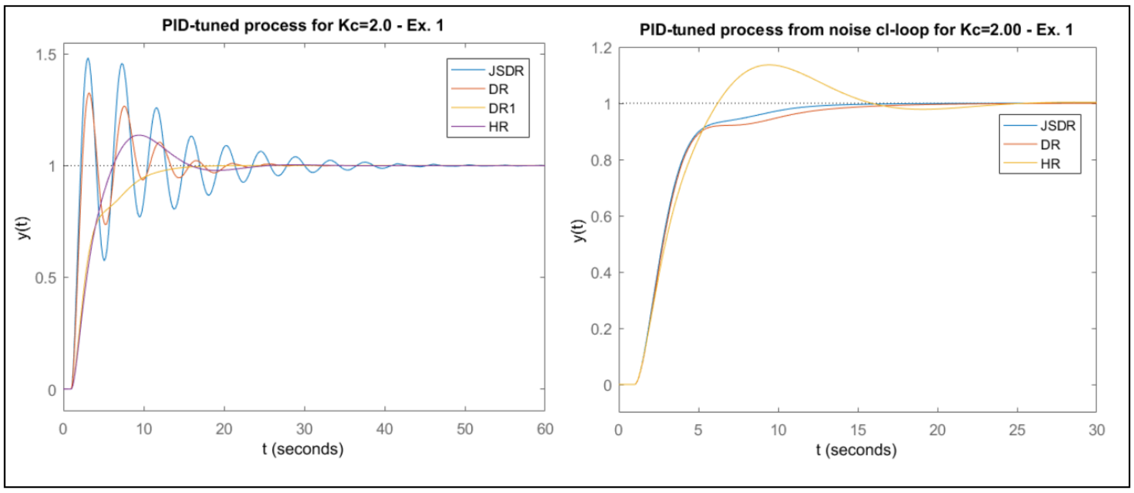


Figure 3-52: Closed-loop response of original process and PID tuned controller without and with noise – Example 1

When not subjected to measurement noise, the DR1 algorithm gave PID-settings with most robust and stable dynamics as seen on the left graph in Figure 3-52, while the both JSDR and DR algorithms produced controller settings with oscillatory dynamics which took long time to reach the steady state. Note that, among the three algorithms (JSDR, DR and DR1), the JSDR produced the least robust and yet stable closed-loop response as seen in Figure 3-52.

Example 2

This example is meant to assess the performance of the algorithms in auto-tuning PID-controller for a firth-order process containing five identical time-constants with no time delay as given in Equation (3.81). Figure 3-53 shows step-response of the original process on left, and its corresponding response of the feedback system composed of P-controller with K_c=1.5.

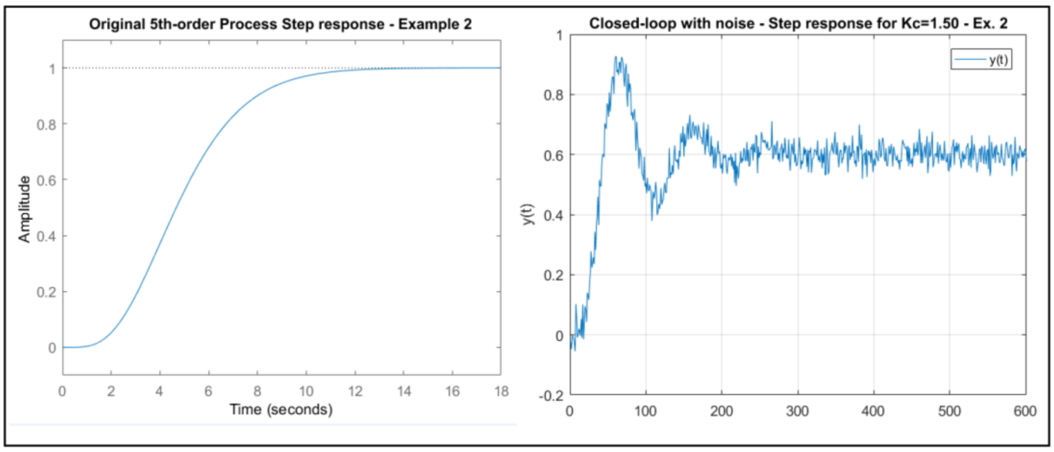


Figure 3-53: Process step-response and corresponding feedback response with P-controller at K_c=1.5 – Ex. 2

The PID-auto tuning results in both cases (without and with the masurement noises) are presented in Table 3.33. In both cases, the JS algorithm identified a second-order model approximation with negative relative frequency ζ_p, and hence unstable dynamics. As a result, the algorithm gave invalid PID-setting due to negative integral and derivative time as seen in Table 3.33.

Table 3.33: PID Auto-tuning SOPDT with different algorithms for K_c=1.5 - Example 2

	Without noise					**With noise**				
	τ_0	ζ_p	K_{pid}	T_i	T_d	τ_0	ζ_p	K_{pid}	T_i	T_d
JS	2.3729	-0.174	0.2516	-0.824	-0.82	2.2631	-0.154	0.2689	-0.698	-7.341
JSDR	2.3729	0.6592	1.57	3.13	1.8	2.2631	0.5466	1.5139	2.4741	2.0702
DR	2.3729	0.6577	1.573	3.12	1.8	2.2631	0.5483	1.5092	2.4816	2.0639
DR1	2.3729	1.3432	1.01	6.37	0.883	2.2631	0.9559	1.0542	4.3267	1.1838
HR	1.2247	1.0206	0.3	1.5	1.0	1.2247	1.0206	0.3	1.5	1.0

When no noise involved, the JSDR, DR and DR1 identified a second-order model with stable dynamics and inverse response. Having ζ_p of 0.6592 and 0.6577 respectively, models from

JSDR and DR experienced overshoot and minimal inverse response compared to model obtained from DR1, which possessed no overshoot and had relatively large magnitude of the inverse response as shown on the left graph in Figure 3-54. In general, when not subjected to the noise, the step-response of models from all the three algorithms (JSDR, DR and DR1) converged toward similar steady-gain as that of HR.

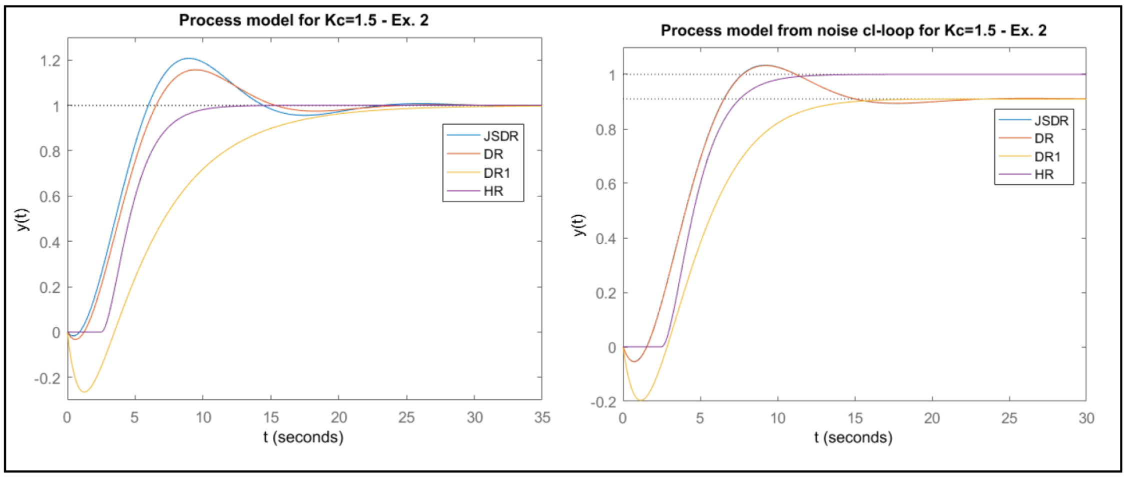


Figure 3-54: Step-response of identified model without and with noise – Example 2

With noise, the JSDR and DR provided almost identical model with similar response containing same magnitude of both the overshoot and the inverse response as seen on right graph in Figure 3-54. The model from DR1 is almost critically damped and experienced the largest inverse response as illustrated on the same graph in Figure 3-54. Notice that, the steady-state gain obtained with JSDR, DR and DR1 differs from HR when the closed-loop was subjected to the measurement noise.

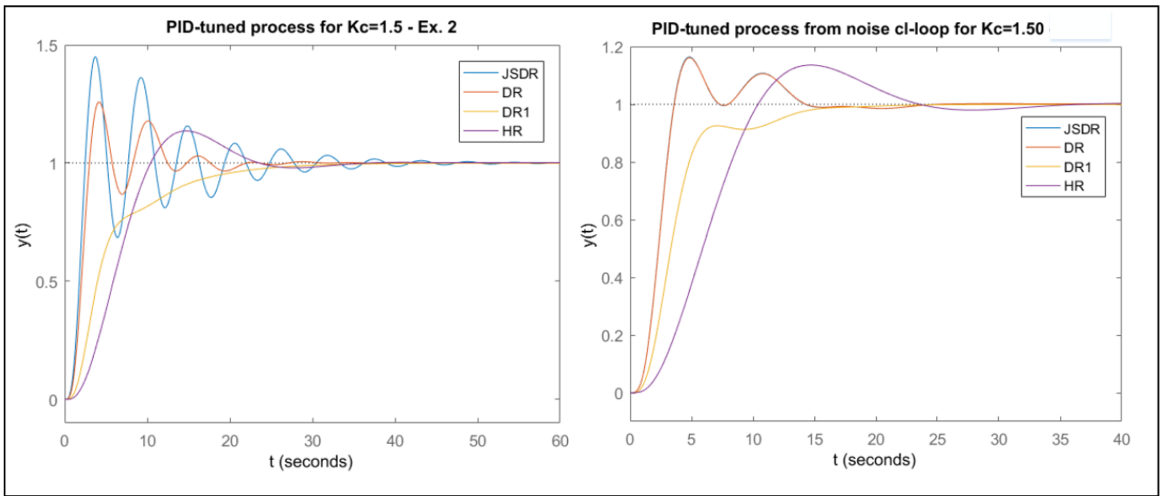


Figure 3-55: Closed-loop response of original process and PID tuned controller without and with noise – Ex. 2

Figure 3-55 shows PID- controller auto tuning result obtained from JSDR, DR, and DR1 algorithms without and with the white Gaussian noise applied to the closed-loop during step experimentation, on left and right respectively. When without noise, the DR1 algorithm produced the most stable feedback response, while the JSDR produced the least robust closed-loop dynamics as seen in the left graph in Figure 3-55. When subjected to the noise, the DR1 still managed to provide the most robust PID-tuned response, while the JSDR and DR produced almost exactly similar settings and hence identical PID-tuned response as shown on the right graph in Figure 3-55.

Example 3

A third-order process with large time delay is to be controlled by PID-controller with tuning parameters obtained from JS, JSDR, DR and DR1 algorithms during step test experiment conducted on the feedback loop which consist of the original process and P-controller with K_c=0.75. Figure 3-56 shows the step-response of the process and its corresponding closed-loop system when subjected to Gaussian's random white noise.

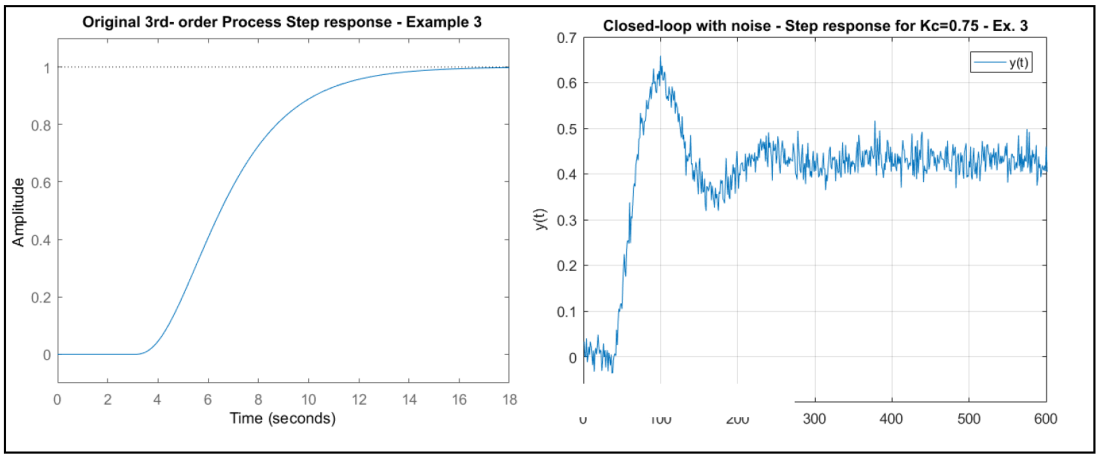


Figure 3-56: Process step-response and corresponding feedback response with P-controller at K_c=0.75 - Example 3

Similar trend as in previous examples was observed for JS algorithm, The algorithm gave second-order model approximation with unstable dynamics in both cases due to negative relative damping, which resulted into negative T_i and T_d therefore invalid PID-settings as seen in Table 3.34. When without noise, the JSDR and DR gave relatively identical identified model with overshoot and similar magnitude of the inverse-response. The DR1 produced overdamped response with similar static gain as others, but with extremely large inverse-response as seen on the left graph in Figure 3-57.

Table 3.34: PID Auto-tuning SOPDT with different algorithms for K_c=0.75 - Example 3

	Without noise					**With noise**				
	τ_0	ζ_p	K_p	T_i	T_d	τ_0	ζ_p	K_p	T_i	T_d

JS	2.7923	-0.1917	0.114	-1.0705	7.2835	2.3021	-0.641	0.2262	-2.9513	-1.796
JSDR	2.7923	0.8859	0.743	4.95	1.58	2.3021	1.3136	0.4552	4.9853	1.0631
DR	2.7923	0.8504	0.775	4.75	1.64	2.3021	1.6164	0.4530	6.6446	0.7976
DR1	2.7923	7.7736	0.397	43.4	0.18	2.3021	0.1355	-0.1775	0.6238	8.4966
HR	1.7321	1.0104	0.2857	2.0	1.5	1.7321	1.0104	0.2857	2.0	1.5

When the noise was applied to the feedback system, the DR1 produced barely stable oscillatory response, however, with shorter inverse response as compared to response when noise was involved as shown on right graph in Figure 3-57. JSDR and DR gave an identified model with overdamped response and larger inverse-response as compared to DR1. In general, when exposed to the noise, all the three algorithms (JSDR, DR and DR1) converged into steady-state with static gain lower than the gain obtained with HR as illustrated in Figure 3-57.

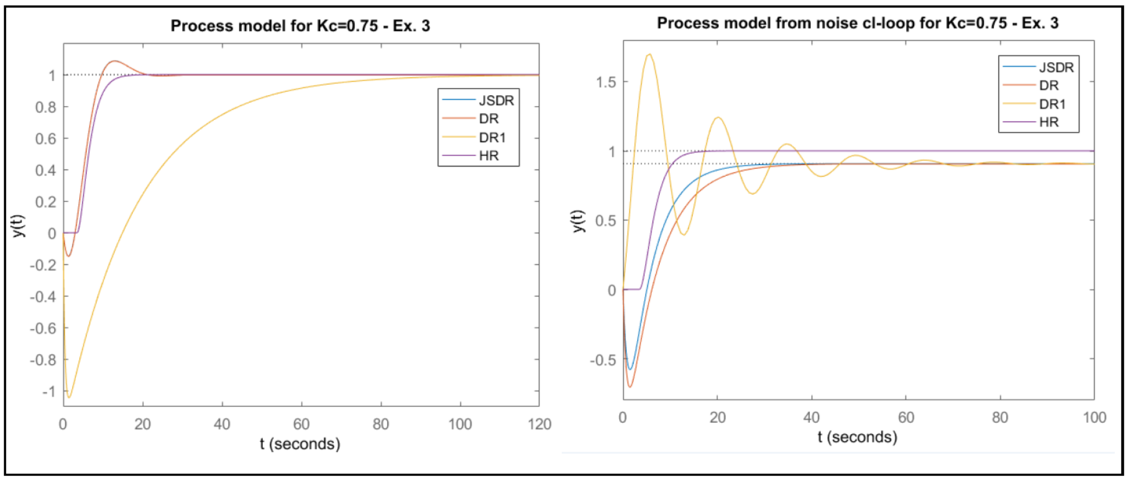


Figure 3-57: Step-response of identified model without and with noise – Example 3

Figure 3-58 shows closed-loop response with PID-controller tuned using settings obtained from SID algorithms (JS, JSDR, DR and DR1), both without and with the random white noise injected into the system.

The PID-controller settings obtained with JSDR and DR algorithm when no noise involved were quite similar and gave identical controlled loop dynamics containing oscillations, overshoots and small inverse response as shown on the left graph in Figure 3-58. The responses were the quickest and reached the steady state faster than HR’s settings. The DR1 algorithm, unlike others, exhibited no overshoots and took long time to bring the process into steady state.

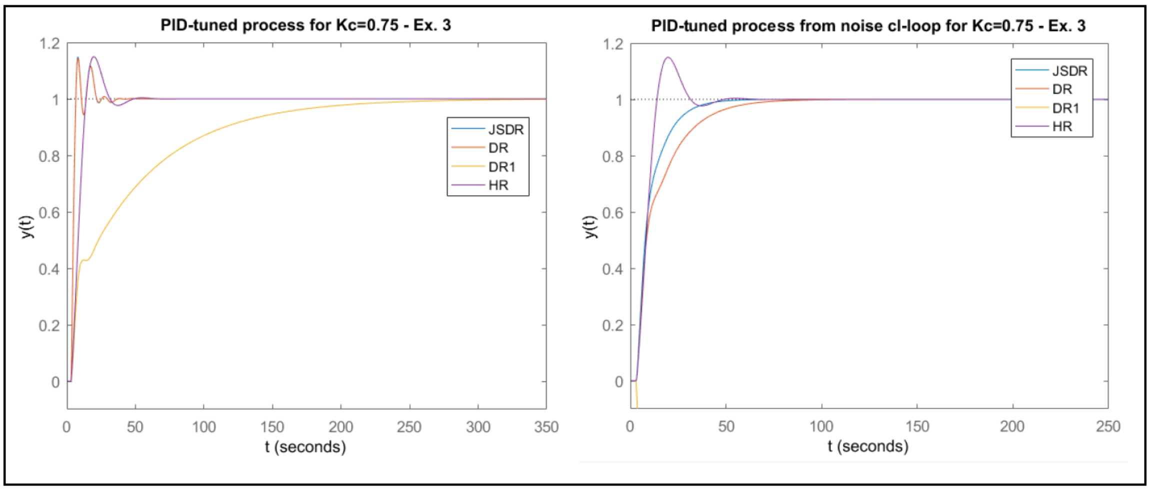


Figure 3-58: Closed-loop response of original process and PID tuned controller without and with noise – Example 3

When exposed to the noise, both JSDR and DR algorithm's PID settings provided stable and non-oscillatory response dynamics, though the JSDR gave settings with the most robust response among the two as illustrated on the right graph in Figure 3-58. On the other hand, the DR1 algorithm produced negative controller gain (K_p=-0.1775), leading to direct action mode and hence worst set point tracking behaviour as seen on right graph in Figure 3-58.

Example 8

Example 8 represents a second-order process with oscillatory dynamics and an input time delay of magnitude one, as given in Equation (3.97). The process is supposed to be controlled and auto-tuned using PID-controller with parameter settings based on identified model obtained by using JS, JSDR, DR and DR1 algorithms with K_c=0.5. Figure 3-59 gives overview of the process response on left, and its corresponding closed loop response (on right) when random noise of white Gaussian type was induced into the feedback system.

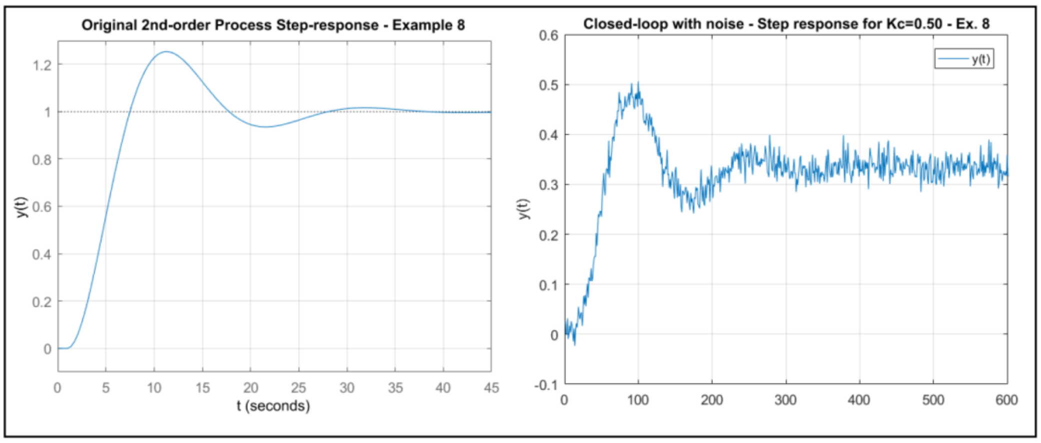


Figure 3-59: Process step-response and corresponding feedback response with P-controller at K_c=0.50 - Example 8

Based on the simulation results presented in Table 3.35, the JS algorithm, when subjected to the noise, produced second-order model with negative damping ratio and hence unstable response. Consequently, the negative damping led to negative T_i and T_d thus invalid PID-settings.

On contrary, JSDR, DR and DR1 identified model with stable dynamics and inverse response when subjected to the noise as illustrated in Figure 3-60. The JSDR and DR models possessed almost identical underdamped and inverse response, while the DR1's model was almost critically damped and had the largest magnitude of inverse response as seen in Table 3.35 and illustrated on right graph in Figure 3-60.

Table 3.35: PID Auto-tuning SOPDT with different algorithms for K_c=0.50 - Example 8

	Without noise					**With noise**				
	τ_0	ζ_p	K_p	T_i	T_d	τ_0	ζ_p	K_p	T_i	T_d
JS	3.0563	0.1575	-0.258	0.9625	9.7051	2.7613	-0.066	0.0397	-0.3658	-20.85
JSDR	3.0563	0.3594	1.8276	2.1966	4.2523	2.7613	0.6572	0.5363	3.6293	2.1009
DR	3.0563	0.3423	2.6672	2.0923	4.4645	2.7613	0.6388	0.5545	3.5280	2.1612
DR1	3.0563	0.4069	1.0517	2.4875	3.7551	2.7613	0.9521	0.3958	5.2583	1.4500
HR	3.0	0.4	1.2	2.4	3.37	3.0	0.4	1.2	2.4	3.37

All the algorithms produced stable models with oscillatory dynamics when with no noise involved. The JS's model exhibited no inverse response and contained more oscillations then others. The JSDR, DR and DR1 estimated models with both oscillation and inverse response, however, the DR1 model experienced the largest magnitude of inverse response and the smallest overshoot among the four algorithms as illustrated on the left graph in Figure 3-60.

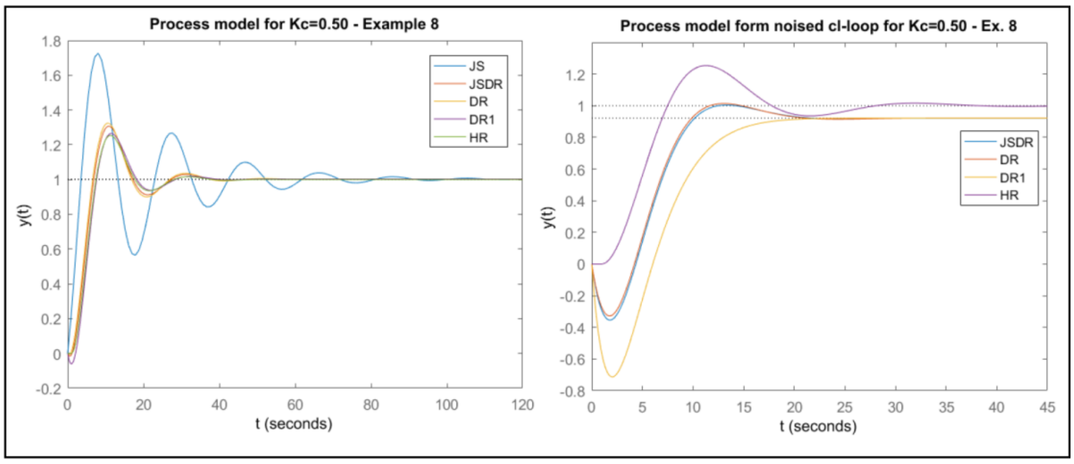


Figure 3-60: Step-response of identified model without and with noise – Example 8

Figure 3-61 illustrates the response of the closed-loop system controlled by PID with settings originated from estimated models obtained by using JS, JSDR, DR and DR1 algorithms. The graph on the right side shows the response when no noise was involved, while the graph on the left side displays the closed-loop response when the feedback system was subject to the noise.

When not subjected with the noise, the JS algorithm produced a combination of negative proportional gain and high integral time (refer to Table 3.35) leading to unstable closed-loop dynamics. DR and JSDR produced stable but oscillatory feedback response, while the DR1 gave the most robust PID settings with high response speed and no oscillations as illustrated on the right graph in Figure 3-61.

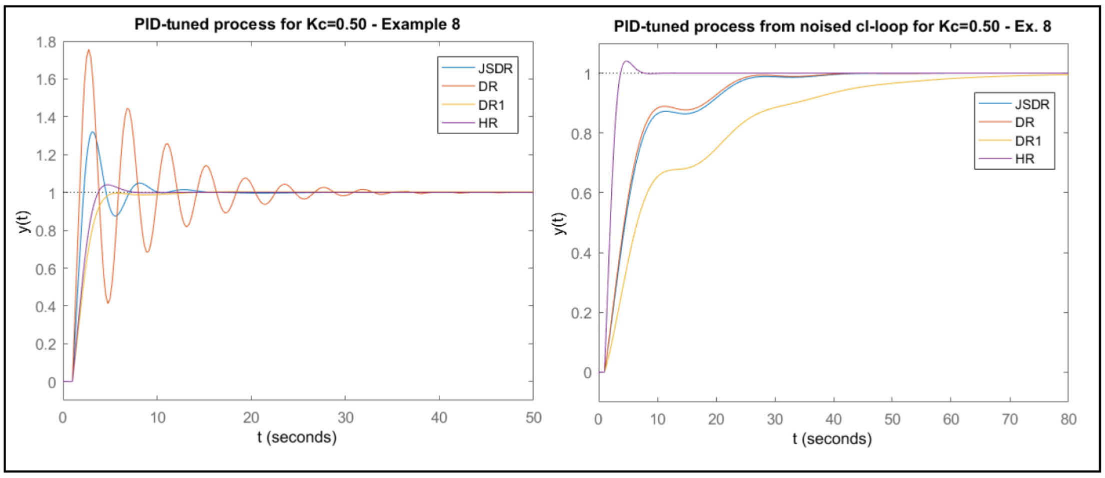


Figure 3-61: Closed-loop response of original process and PID tuned controller without and with noise – Example 8

With noise, the JSDR and DR algorithms produced almost identical settings reflecting similar closed-loop dynamics as shown on the right graph in Figure 3-61. The DR1 settings produced the feedback response with slowest speed, and used the longest time to bring the response to steady state.

3.4 Monte Carlo simulations on SID algorithms in stochastic framework

In this section, Monte Carlo simulation are performed aimed at further assessment and comparison of the performance of the algorithms discussed in this thesis. The technique is applied in simulation of different examples (i.e. examples representing different response dynamics) to both first and second-order algorithms. It should be noted that the simulation involves only numerical examples, due to unavailability of experimental data as stated in section 1.2.

A random signal of white Gaussian noise type with SNR of 25dB was added to signal (closed-loop response). Total of 100 simulation (*m*) were conducted for *n*=1, and with Equation (2.44) as performance criterion. The algorithms are assessed and validated against either HR parameters or real parameters, i.e. parameters in the numerical examples of the same order as the order of the algorithm to be applied on the example. The numerical results obtained at the end of simulation determines the strength of algorithm, whereby small numerical values are associated with good performance. Generally, the smaller the value, the stronger the performance [10].

The Monte Carlo simulation was performed in MATLAB software using the programming codes available in Appendix D. Note that, similar code structure is applicable for the Monte Carlo simulation for SOPDT algorithms. The FOPDT algorithms codes are off coarse replaced with relevant SOPDT algorithms codes as given in Appendix C.

3.4.1 Monte Carlo Simulation for SID algorithms for FOPDT

Three examples (example 1, 2 and 7) with two different values of K_c are simulated by applying the four FOPDT algorithms (YS, JR, JL and CLC).

3.4.1.1 Example 1

This numerical example represents a third-order process with three different poles (T_1 = 2, T_2 = 1, T_3 = 0.5) and an input time delay τ of one second as given by Equation (3.80). The real (reference) parameters for simulations for this particular example are K_p,τ_p and τ_p obtained by using HR technique corresponding to 1.0, 2.5, and 2.0 respectively. Two values of K_c (1.75 and 1.25) are applied during simulation, to enable observation of influence of K_c on performance of the algorithms.

Figure 3-62 gives graphical overview of the estimated static gain K_p of the identified first order model using $K_c = 1.75$ and $K_c = 1.25$ respectively. Based on the distribution of the parameter (K_p), the JR algorithm (blue coloured cross) underwent the most deviation, while the other algorithm gave relatively stable K_p.

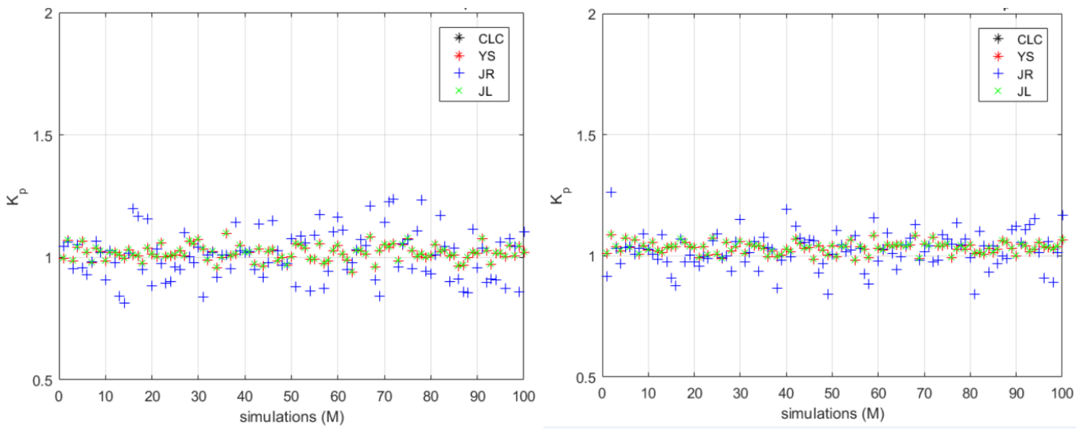


Figure 3-62: Monte-Carlo's K_p estimate using $K_c = 1.75$ and $K_c = 1.25$ respectively – Example 1

When it comes to the time constant $\top_p$, the CLC algorithm displayed very high variation and hence highest deviations among others, as illustrated in Figure 3-63. The YS experienced also some noticeable variations, while the JL showed the most stable prediction of the $\top_p$.

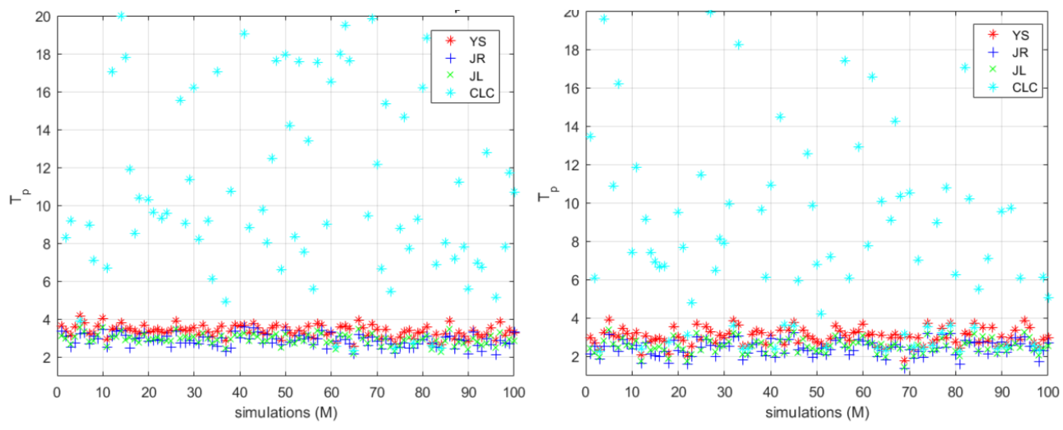


Figure 3-63: Monte-Carlo's $\top_p$ estimate using $K_c = 1.75$ and $K_c = 1.25$ respectively – Example 1

Figure 3-64 gives graphical overview of distribution of the identified time delay τ_p for the two K_c values used during simulation. For both values of K_c, there is clear separation between the CLC's values and the rest. The CLC algorithm displayed better precision for τ_p compared to $\top_p$ as illustrated in Figure 3-63, though with low accuracy in both cases. Note that at K_c=1.25, the YS, JR and JL precision were minimized.

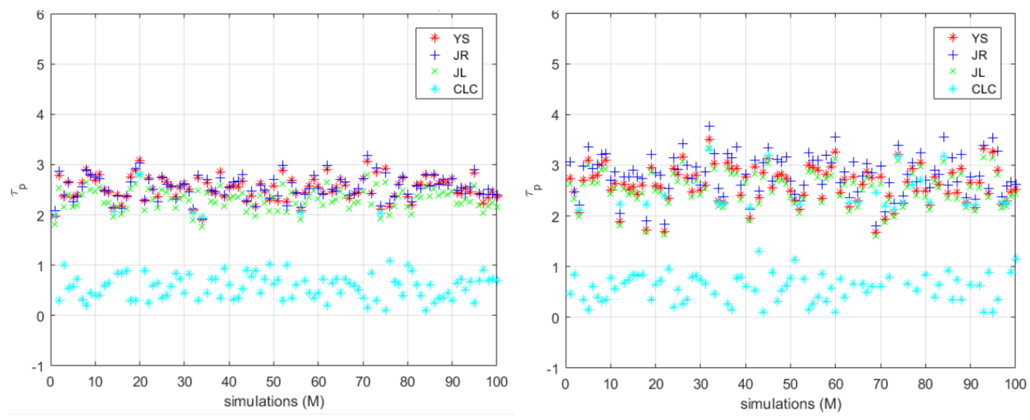


Figure 3-64: Monte-Carlo's τ_p estimate using $K_c = 1.75$ and $K_c = 1.25$ respectively – Example 1

Table 3.36 gives overall performance result of the tested algorithms for both values of K_c. The JL algorithm displayed strongest performance in both cases, while the CLC's performance in both cases were the poorest and extremely unreliable due to the huge registered difference from the real parameters as seen in the Table 3.36. The JR's performance is rated second when K_c=1.75, and was the third strongest when K_c=1.25.

Table 3.36: Algorithm's performance - Example 1

Algorithm	Performance	
	K_c =1.75	K_c = 1.25
YS	1.2886	1.0664
JR	0.7047	1.0692
JL	0.4346	0.5447
CLC	850.96	460.44

3.4.1.2 Example 2

Example 2 represents a firth-order process with identical poles of T=1 and no input time delay as given by Equation (3.81). The reference values for this example are HR's parameters K_p,$\top_p$ and τ_p with corresponding values 1.0, 1.5, and 3.5 respectively. During simulation, two values of K_c are used, which are 1.25 and 0.75.

The distribution of the static gain K_p during simulation showed similar trend as for example 1 above. The JR algorithm displayed the largest deviation for both size of K_c, though still with acceptable limit as illustrated in Figure 3-65.

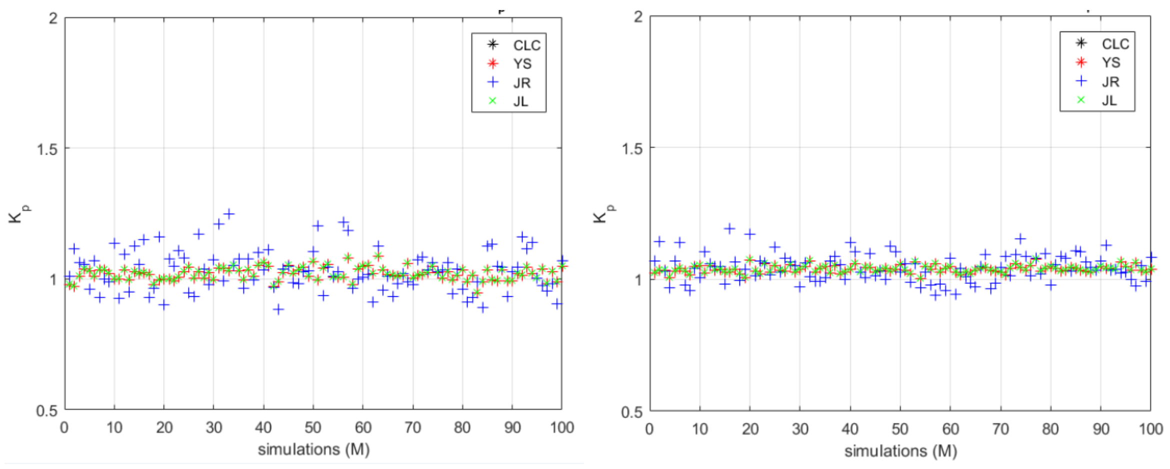


Figure 3-65: Monte-Carlo's K_p estimate using K_c = 1.25 and K_c = 0.75 respectively – Example 2

For the time constant τ_p, the CLC algorithm showed again highest deviations amongst the algorithms in both cases as illustrated in Figure 3-66. At K_c=0.75, the precision of the rest of algorithms experienced some decrements in strength.

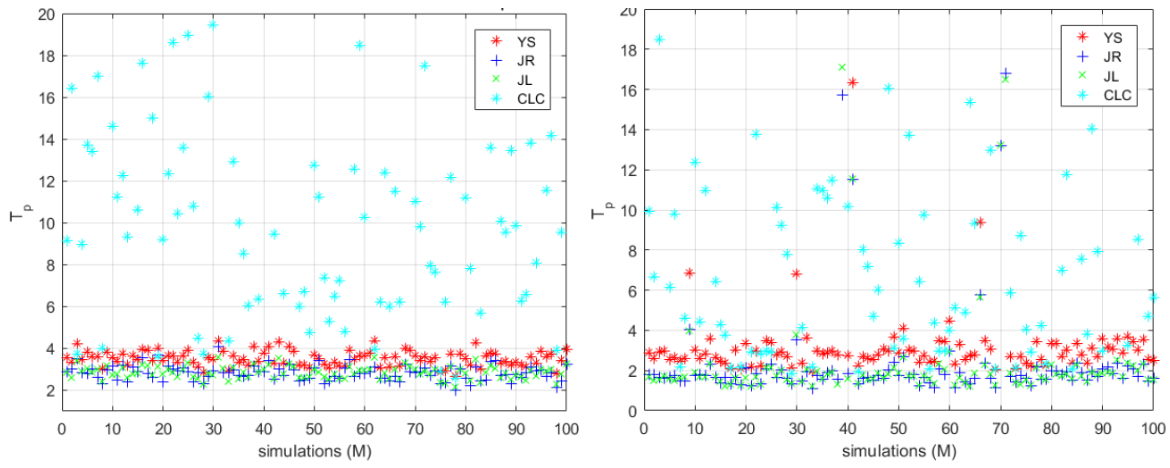


Figure 3-66: Monte-Carlo's τ_p estimate using $K_c = 1.25$ and $K_c = 0.75$ respectively – Example 2

Figure 3-67 gives overview of distribution of time delay τ_p during the simulations for both values of K_c. As seen in the graphs, the CLC algorithm identified the τ_p mostly between 0 and 2, while the other three algorithms gave higher values of τ_p ranging from 2 and above, and hence displayed clear separation among them.

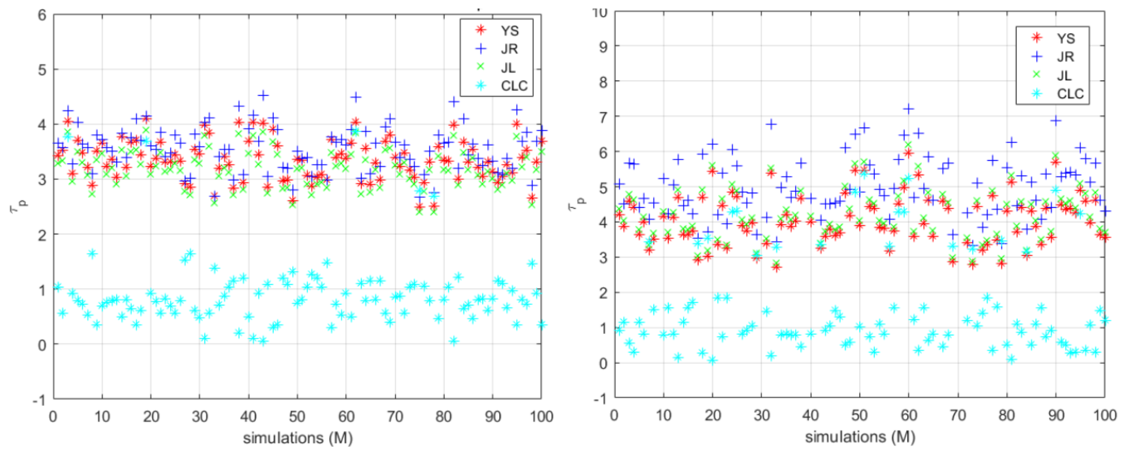


Figure 3-67: Monte-Carlo's τ_p estimate using $K_c = 1.25$ and $K_c = 0.75$ respectively – Example 2

Table 3.37 gives overview of the overall performance of the algorithms for both values of K_c. Based on the result, the CLC algorithm experienced least strong and very poor performance in both conditions. The JL and JR showed almost identical strength when K_c=1.25, while the YS algorithm took the third place. When K_c=0.75, the JL displayed strongest performance followed by YS and JR in sequence.

Table 3.37: Algorithm's performance - Example 2

Algorithm	Performance	
	K_c =1.50	K_c = 1.00
YS	4.5006	3.5485
JR	2.1494	1.5236
JL	2.1362	1.2680
CLC	1555	765.6

3.4.1.3 Example 7

Example 7 represents a first-order process with static gain K_p, time-constant τ_p and input delay time τ_p of 1, 1 and 2 respectively. The process parameters are, off course, the natural reference values used for validation of the simulation result, whereby two different values of 1.0 and 0.75 for K_c were used.

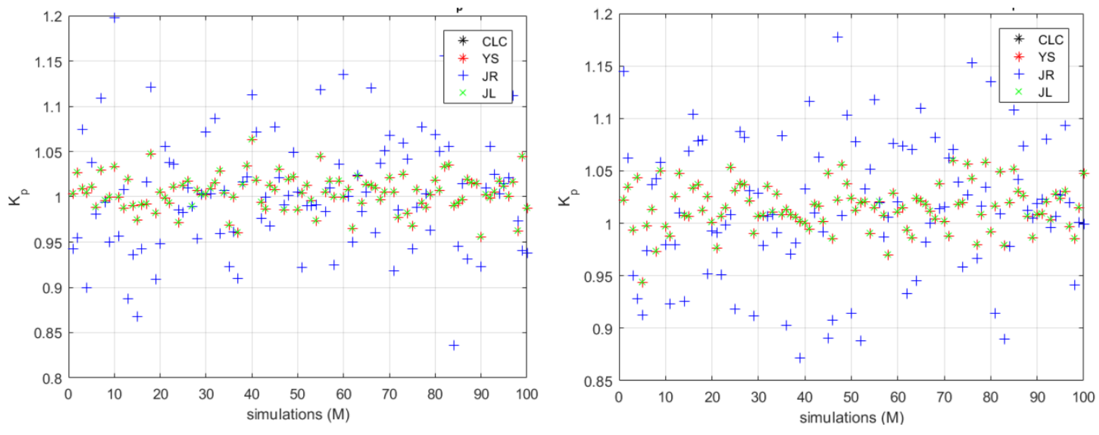


Figure 3-68: Monte-Carlo's K_p estimate using K_c = 1.0 and K_c = 0.75 respectively – Example 7

Figure 3-68 gives graphical result and distribution of the estimated static gains by the three algorithms for K_c = 1.0 and K_c = 0.75 respectively. All the algorithms identified the gain around 1, however, the JR showed the least precision among the four.

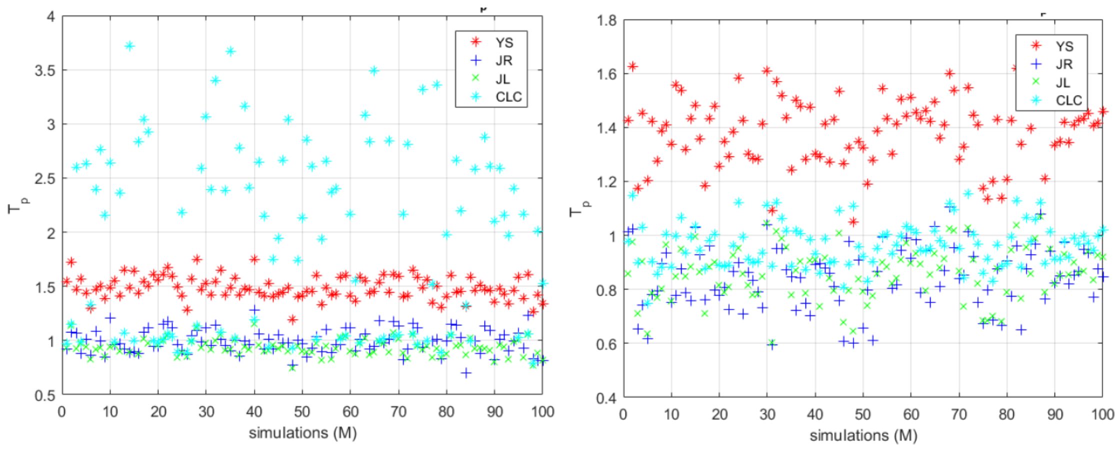


Figure 3-69: Monte-Carlo's T_p estimate using K_c=1.0 and K_c=0.75 respectively – Example 7

For the time constant T_p, the CLC algorithm exhibited largest deviations and poor precision for K_c= 1.0. However, at K_c= 0.75 the algorithms improved the estimations and identified the T_p value around 1 as shown in Figure 3-69. The YS algorithm showed poor accuracy, though with relatively stable precision in both cases as seen Figure 3-69.

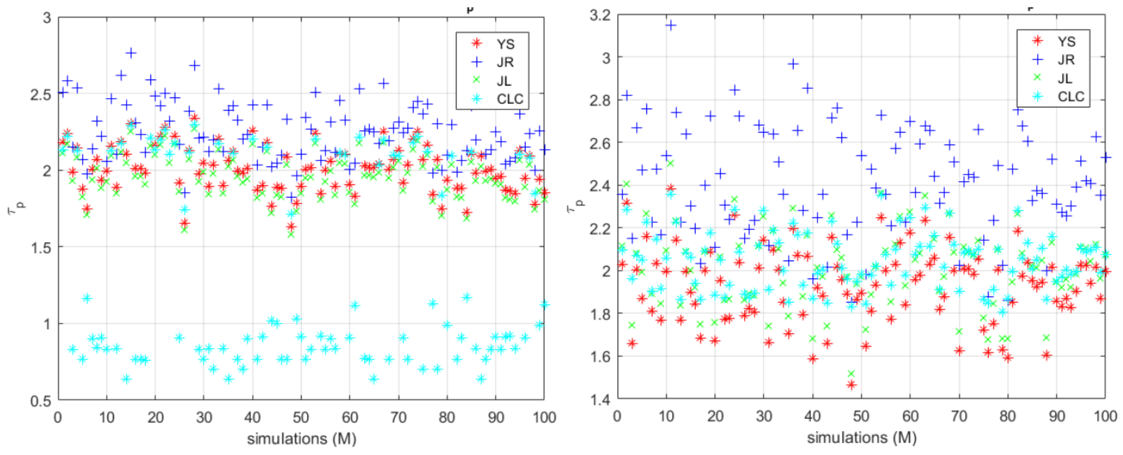


Figure 3-70: Monte-Carlo's τ_p estimate using K_c=1.0 and K_c=0.75 respectively – Example 7

The CLC displayed similar trend in estimation of time delay τ_p, which is poor accuracy and low precision at K_c=1.0, but improved performance at K_c=0.75. At K_c=0.75, the JR's identified τ_p experienced largest deviation among the four algorithms as seen in Figure 3-70.

Table 3.38: FOPDT algorithms' performance - Example 7

Algorithm	Performance	
	K_c =1.00	K_c = 0.75

YS	0.2717	0.2044
JR	0.1167	0.2813
JL	0.0335	0.0656
CLC	2.5231	0.0281

Table 3.38 presents overview of the overall performance of the algorithms for both values of K_c. When K_c=1.00, the JL algorithm displayed strongest performance amongst the algorithms, followed sequentially by JR and YS. The CLC had the least strong performance at K_c=1.0. However, the algorithm displayed the strongest performance when K_c=0.75, followed by JL, YS and JR at last place.

3.4.2 Monte Carlo Simulation for SID algorithms for SOPDT

For the second-order SID algorithms, five numerical examples (example 1, 2, 3, 8 and 9) representing systems of different dynamic characteristics are simulated to assess and compare the algorithm's performances among each other.

3.4.2.1 Example 1

The example is identical to example 1 used for Monte Carlo simulation for FOPDT. The reference parameters for simulations for this particular example are b_0, b_1, a_0 and a_1 obtained by HR technique corresponding to 0.4, -0.5, 0.4 and 0.96 respectively. The value of K_c used during simulation is 2.5.

Figure 3-71 gives graphical overview of the identified parameters b_1 and b_0 respectively. Generally, all the algorithms experienced high precision around zero for the parameter b_1, however, the DR1 algorithm was least precise among the four. When it comes to b_0, the algorithms provided almost identical estimation for all the 100 numbers of simulations conducted as shown in Figure 3-71.

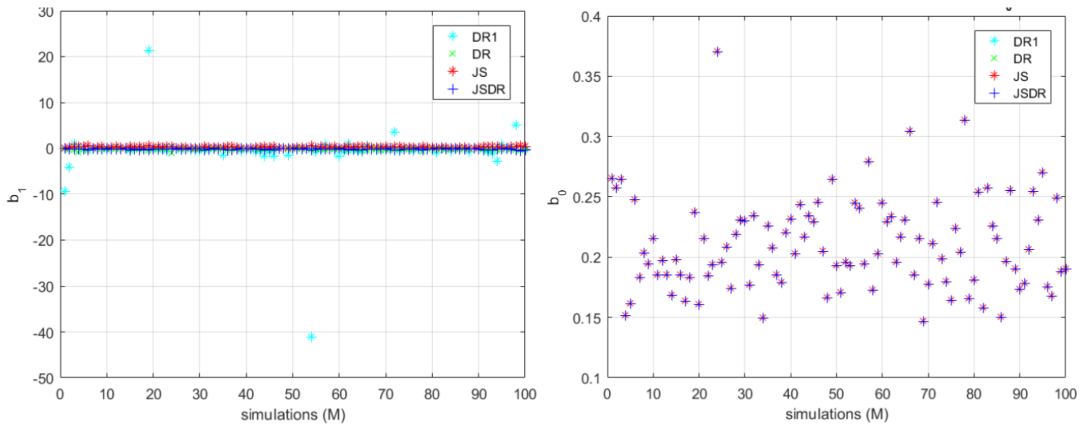


Figure 3-71: Monte-Carlo's simulation result for estimated parameters b_1 and b_0 respectively – Example 1

When identifying the parameter a_1, the DR1 algorithm displayed huge deviation in very few numbers of simulations. Otherwise, the similar trend is spotted for the estimated parameters a_1 and a_0 as shown in the Figure 3-72.

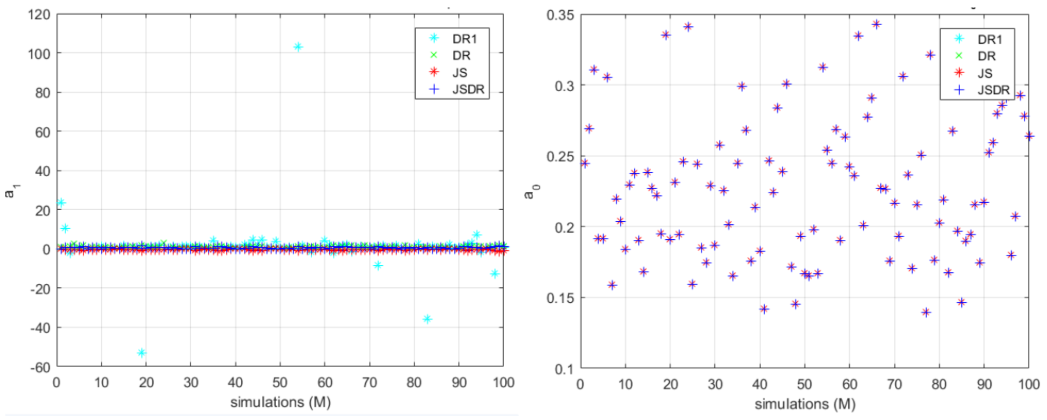


Figure 3-72: Monte-Carlo's simulation result for estimated parameters a_1 and a_0 respectively – Example 1

Table 3.39 presents overall performance of the algorithms based on the Monte Carlo simulation of this particular example with K_c=2.5. The DR algorithm displayed the strongest performance among the four, while the JSDR followed closely at the second place. Based on the simulation result, the DR1 had the least strong performance as seen in Table 3.39.

Table 3.39: SOPDT algorithms' performance - Example 1

Algorithm	Performance with K_c=2.5
JS	2.5232

JSDR	0.5098
DR	0.4802
DR1	183.71

3.4.2.2 Example 2

This example is identical to the example 2 simulated for FOPDT algorithms in section 3.4.1.2. The reference parameters are similarly b_0, b_1, a_0 and a_1 obtained by HR technique corresponding to 0.6667, -1.6667, 0.6667 and 1.6667 respectively. The value of K_c used during simulation is 2.0.

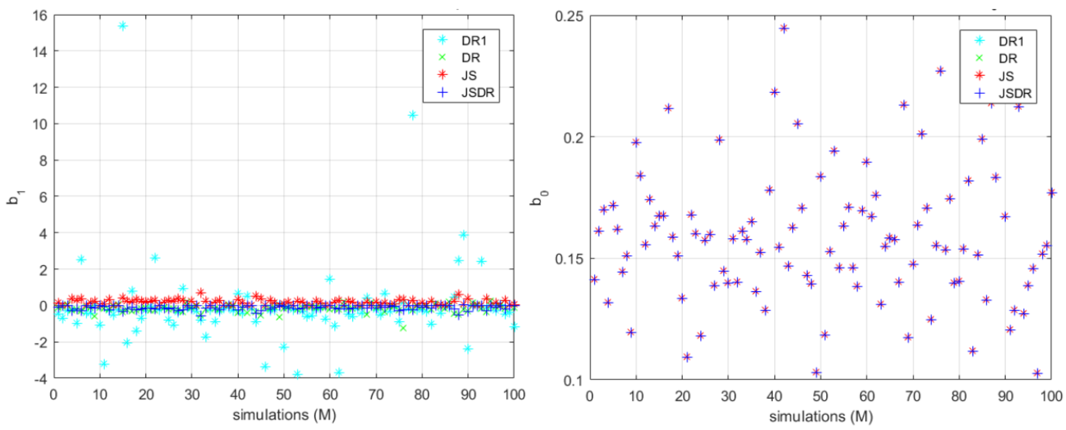


Figure 3-73: Monte-Carlo's simulation result for estimated parameters b_1 and b_0 respectively – Example 2

Figure 3-73 shows graphical distribution of the identified parameter b_1 on the left and the parameter b_0 on the right. The DR1 algorithm displayed, again, least precision as it showed noticeably huge variation in some simulations. The other three algorithms estimated b_1 mostly around zero with high precision. For b_0, the algorithms produced identical values of the parameter for almost all 100 simulations.

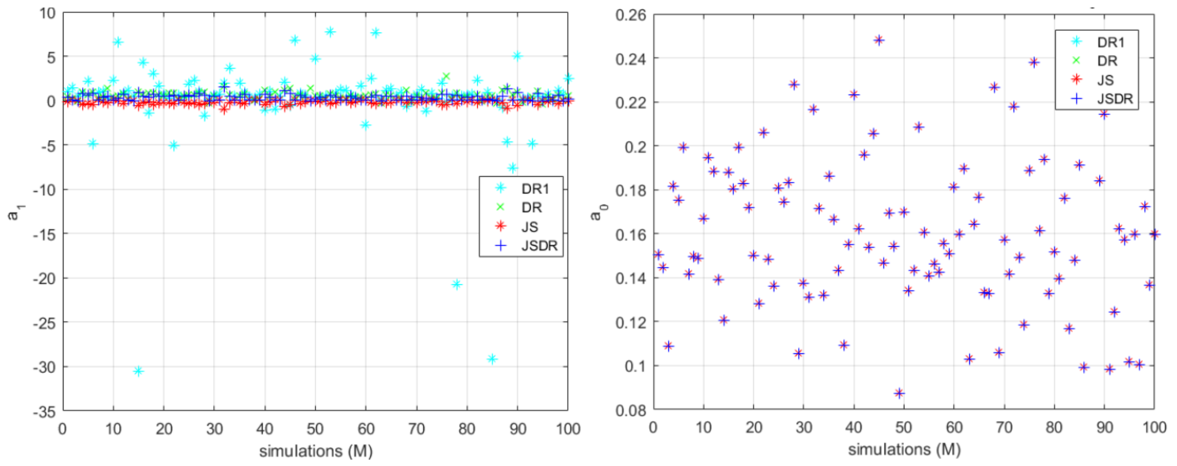


Figure 3-74: Monte-Carlo's simulation result for estimated parameters a_1 and a_0 respectively – Example 2

Similar trend is observed when it comes to estimation of parameters a_1 and a_0 using the Monte Carlo simulation as shown in Figure 3-74. The only noticeable difference is that the a_1 value identified using DR1 deviated more toward negative value compared to the deviation displayed by b_1.

Table 3.40: SOPDT algorithms' performance - Example 2

Algorithm	Performance with K_c=2.0
JS	7.3215
JSDR	4.8721
DR	4.4109
DR1	41.5358

Compared to the HR identified parameters, the overall performances of the algorithms in this particular numerical example are as given in Table 3.40. The DR exhibited the strongest performance amongst all, though followed closely by JSDR at second, and JS at third place. The DR1 appeared again to be the least strong as seen in Table 3.40.

3.4.2.3 Example 3

This example represents a third order process containing significant large (larger than the dominating time constant) τ_p=3, and three time-constants T_1=2, T_2=T_3=1, as given by Equation (3.82). A K_c=1.0 was used for simulations, with the HR's second-order calculated parameter b_0, b_1, a_0 and a_1 as reference (real) values used for validation. The values are given in their respective order as 0.3333, -1.1667, 0.3333 and 1.1667.

The distribution and variation of the identified parameters b_0, b_1, a_0 and a_1 obtained as a result of simulations showed exactly similar trend as the one observed in simulation results for example 2 as illustrated in Figure 3-73 and Figure 3-74.

Table 3.41: SOPDT algorithms' performance - Example 3

Algorithm	Performance with K_c=1.0
JS	3.9054
JSDR	1.8675
DR	1.5416
DR1	17.7001

Table 3.41 presents the overall performance result of all the algorithms tested on the example 3. As seen in the table, the DR algorithm emerged with the strongest performance of all, yet again closely followed by JSDR, and JS in the respective order. The DR1 algorithm displayed the least strong performance of all.

3.4.2.4 Example 8

The example represents a second-order process with underdamped response and an input delay τ_p=1. The Monte Carlo simulation for this example are performed at K_c=0.75, and with the model's real data as reference data for validation of the simulation results obtained using the four algorithms (JS, JSDR, DR and DR1). These validation parameters b_0, b_1, a_0 and a_1 corresponds respectively to 0.1111, -0.1111, 0.1111 and 0.2667.

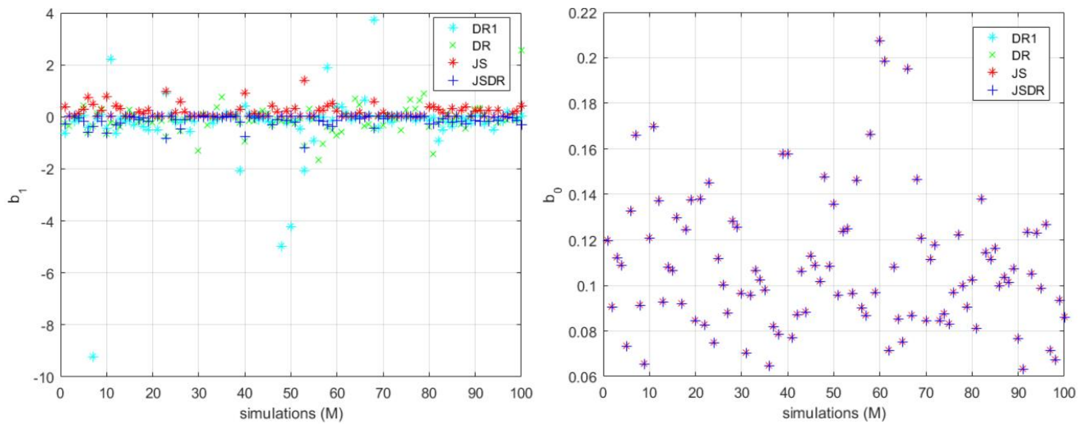


Figure 3-75: Monte-Carlo's simulation result for estimated parameters b_1 and b_0 respectively – Example 8

The DR1, DR and JS algorithms experienced some sort of variation and less precision in identifying the parameter b_1 as illustrated in Figure 3-75. However, when it comes to estimation of parameter b_1, all the algorithms gave identical results in all simulations as also observed in previous examples.

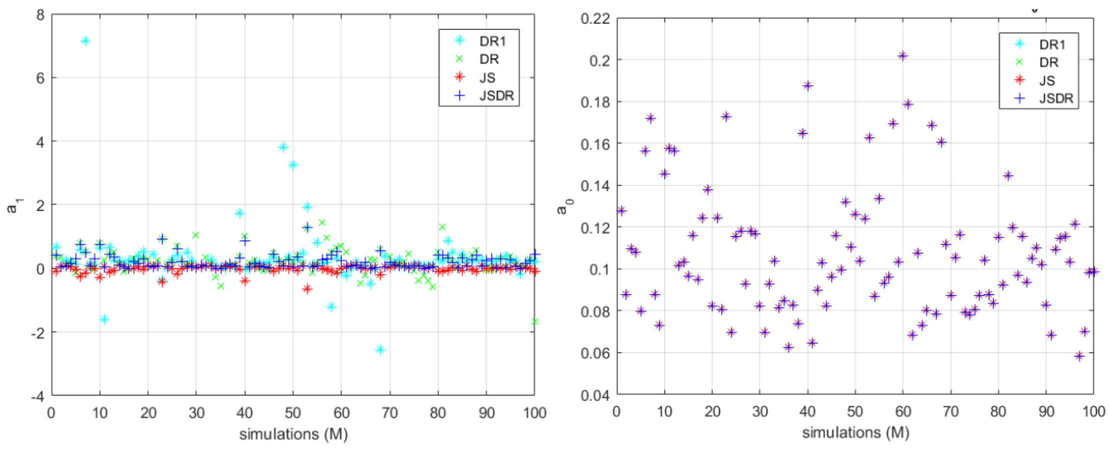


Figure 3-76: Monte-Carlo's simulation result for estimated parameters a_1 and a_0 respectively – Example 8

Figure 3-76 gives result overview of the identified parameter a_1 and a_0. In most simulations, the algorithms estimated the parameter a_1 within the interval of -0.5 to 0.5. However, some few deviations were detected whereby the DR1 stood for the most of it as shown in left graph in Figure 3-76. The algorithms estimate for the parameter a_0 were identical and appeared to be within the range of 0.06 – 0.2.

Table 3.42: SOPDT algorithms' performance - Example 8

Algorithm	Performance with K_c=0.75
JS	0.2322
JSDR	0.0983
DR	0.4223
DR1	2.5695

Table 3.42 presents the overall performance result for the algorithms against the HR parameters for this particular example and at K_c=0.75. Based on the results, the JSDR emerged with the strongest performance, followed by JS and DR. The DR1 displayed the least strong performance of all as seen in the Table 3.42.

3.4.2.5 Example 9

The process model given by Equation (3.98) represents a stable but oscillatory second-order process with no input time-delay. Both of the process' poles are located on the left hand side of the complex plane. The process contains also a zero, situated at the right hand side of the plane.

$$h_p(s) = \frac{0.1518s + 0.03928}{s^2 + 0.1067s + 0.0485} \tag{3.98}$$

Monte Carlo simulations were performed using the JS, JSDR, DR and DR1 algorithms with K_c=0.2. The reference parameters for performance validation b_0, b_1, a_0 and a_1 are the real model parameters given as 0.0393, 0.1518, 0.0485 and 0.1067 respectively.

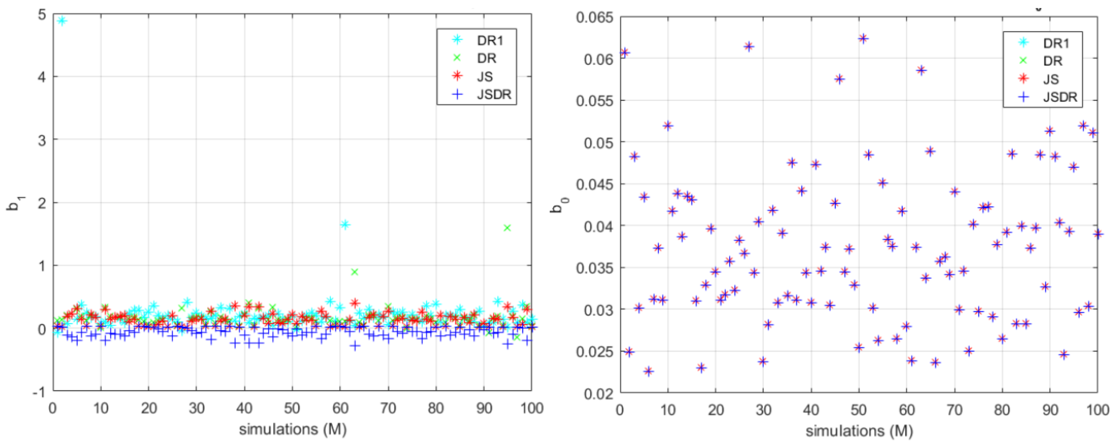


Figure 3-77: Monte-Carlo's simulation result for estimated parameters b_1 and b_0 respectively – Example 9

Figure 3-77 gives graphical overview of the identified parameter b_1 and b_0 displayed on the left and the right graph respectively. The algorithms estimated the value of b_1 ranging from around -0.5 to 0.5 with relatively good precision as seen on left graph in Figure 3-77. The algorithms were again consistent and identical when identifying the parameter b_0 as shown on right graph in Figure 3-77.

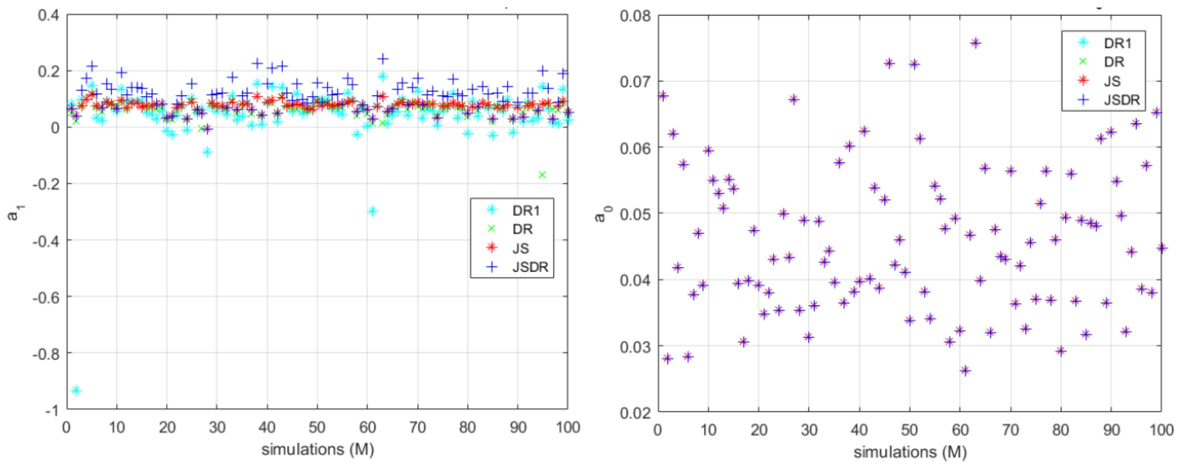


Figure 3-78: Monte-Carlo's simulation result for estimated parameters a_1 and a_0 respectively – Example 9

The same pattern is followed when it came to identification of the parameter a_1 and a_0. Based on the right graph in Figure 3-78, the algorithms estimated a_1 mostly within the interval of 0 – 0.2, though with varied precision and accuracy among them. For the parameter a_0, the result ranged mostly from 0.03 to 0.07. The algorithms produced identical values of a_0 in same simulation index.

Table 3.43: SOPDT algorithms' performance - Example 9

Algorithm	Performance with K_c=0.2
JS	0.0122
JSDR	0.0529
DR	0.0378
DR1	0.2752

Table 3.43 gives overview of the overall performance of the algorithms when applied to example 9 with K_c=0.2. Generally, all performance of the entire assessed algorithm were quite impressive, and did not differ much among them. However, the JS algorithm showed strongest performance followed by DR, JSDR, and DR1 in the name sequence as seen in the Table 3.43.

4 Discussion

This chapter weighs up the results obtained from the intensive simulation study conducted as presented in sections 3.2, 3.3 and 3.4. The chapter provides objective interpretation, explanation and analysis of the results greater perspective as a whole, and compare the findings against the literature, in the outlined sequence.

4.1 Choice of suitable K_c

Apart from the MF algorithm, all the Transient analysis SID methods presented in this thesis utilizes the simplicity of P-controller in studying the response of an output feedback system, leading to identification of open-loop system by means of back calculation. However, the simplicity of P-controller comes with its price, namely by introducing steady state offset in the system response [8, 24]. The offset does not prevent to system to attain stable and consistent response, but it makes the system unable to exactly align its response with controller's reference value.

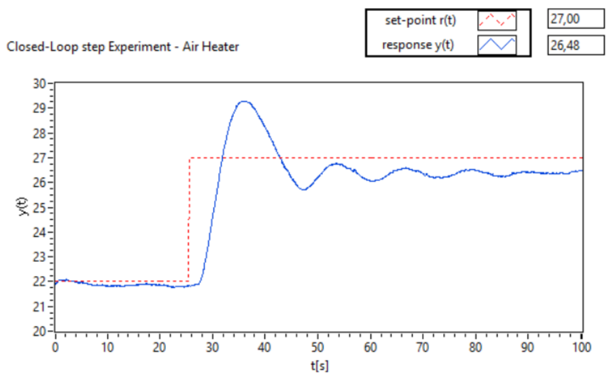


Figure 4-1: Steady state offset in air-heater's response with P-controller

Figure 4-1 illustrates the steady state offset observed during step test experimentation conducted on an air heater controlled by P-controller at $K_c = 1.25$. An offset of around 0.50° Celsius prevent the system response (blue coloured) aligning with the set-point (dotted red line).

Moreover, the algorithms presented, are based on underlying assumption that the size of the controller gain K_c shall be chosen wisely to produce underdamped closed-loop response (i.e. with relative damping ζ in interval $0<\zeta<1$) [2, 4-6, 8-10]. The simulations performed in chapter 3.2 has proven the inapplicability of the algorithms when ζ of the closed loop response lies outside the required interval. The algorithms fails to produce solutions, and gives complex solutions when $\zeta > 1$ and $\zeta < 0$ respectively.

The size of K_c has huge influence on the performance of algorithm. Even when the chosen K_c satisfies $0<\zeta<1$, still cannot guarantee validity of the results. As shown and illustrated during simulation study, the algorithms reacts different to size of K_c. In general, all the presented algorithms favour small values of K_c, which again are associated with small overshoot. The

CLC algorithm produces best possible results when the closed-loop response undergoes minimal overshoot with magnitude of normally not larger than 0.15. For the JR algorithm, however, an overshoot with magnitude smaller than 0.075 gives poor results. Depending on nature of the response, the JL algorithm has proven to cover the largest range of overshoot of about 0.031 – 0.625 in magnitude, and still being able produce good model parameters. The size of K_c influenced also the performance of the algorithms for second-order models.

4.2 Persistent excitation and Back calculation

In closed-loop system identification, the persistent excitation alone does not always guarantee sufficiently informative data for the identification purpose, which lead to distinguishable models based on only input-output description of the system [29]. Using transfer function as chosen model structure, however, minimized the probability of ending up with non-distinctive models when the back calculation was applied.

The simulation study in chapter 3.2 has proven that, when no noise is involved, the back calculation techniques can approximate models that can be distinguished based on only the input-output description of the system. The finding underlines the need of proper filtering (i.e. removing white noise, high and low frequency disturbances) of the signal data obtained from the step experimentation before being imported into the SID algorithms.

4.3 Step-response Transient SID algorithms for FOPDT

Five algorithms for step-response transient analysis SID for FOPDT have been discussed in this study. The algorithms strength and performance have been assessed though simulation study of different examples representing processes of different dynamics and response behaviour.

Starting with the YS algorithm, the simulation results obtained in subchapter 3.2.3.1.2 and 3.3.2.2.1 captures repeated trend in YS's performance. Generally, the algorithm estimates sufficiently good model parameters for systems with first-order dynamics as seen in simulation results for examples 4, 5, 6 and 7. The algorithm yields less good model results when applied to systems with significant large time delay as illustrated in simulation result for examples 3, 4, 6 and 7. The algorithm's performance worsens with increasing order of the system, as shown in simulation results for examples 1, 2, 3 and 8. Moreover, the algorithm's ability in identifying model parameter is even worsened when applied to higher-order system with oscillatory dynamics.

Under the similar conditions, the JR algorithm identified model parameters that gave somehow better reflection of the original system as compared to YS algorithms as proved by the simulation results presented in section 3.2.3.2 and 3.3.2.2.1. However, the JR algorithm identified a slightly higher time delay, which in most cases deviated further from HR parameters as seen in almost all the simulation results. Additionally, the JR algorithm yields negative time delay and/or negative time constant for the estimated model when the underdamped closed-loop response of the original system moves closer to overdamped response as discussed in chapter 4.1.

Among the five algorithms, the JL algorithm identified model parameters with best possible dynamics description and reflection of the original system, based on the simulation results obtained in sub-chapter 3.2.3.3 and 3.3.2.2.1. The algorithm exhibits good ability in coping with various conditions for example low- and high-order system, system with oscillatory dynamics, as well as system with significant large dead time as seen in simulation results for examples three, four, six and seven. Unlike other, the algorithm accepts the largest range of magnitude of the overshoot occurring in closed loop response without compromising much on validity and quality of the estimated parameters.

CLC-algorithm's ability to identify models that match original system dynamics is highly dependent on the size of overshoots the feedback response. Based on the simulation results in sections 3.2.3.4.2 and 3.3.2.2.1, the algorithm's best performance happened when the overshoots are extremely low. Even when applied to closed-loop response with minimal overshoots, the algorithm still struggled to provide good sets (combination) of model parameters. In most examples, the algorithm identified quite acceptable gain and time constant, but with poor (in respect to HR) identified dead time, or vice versa. In respect with the original system dynamics, the CLC algorithm's coped better with first-order systems compared to ones with higher-order dynamics.

Unlike other, the MF algorithm utilizes PI-controller in estimation of models from closed-loop system response (as highlighted in section 3.1.5), requiring pre-setting of both the test gain and integral time prior to step test experimentation. Finding best combination of controller test parameters has proven to be very challenging, as it requires sufficient knowledge of the system dynamics. Similar to CLC, the MF also appeared to perform best when applied to feedback system having response with very small overshoots. The simulation results obtained in section 3.2.3.5.2 shows however an interesting finding. The algorithm produced relatively acceptable and better time-constant when the original system dynamics contains experience input time delay, while the algorithm gave relatively acceptable estimated time delay when the original system dynamics contains no input delay.

4.4 Step-response Transient SID methods for SOPDT

For second-order with time-delay SID, four algorithms (JS, JSDR, DR and DR1) have been discussed and assessed by means of simulation study of different examples representing systems with different dynamics as presented in chapter 3.1 and 3.2.4.

Staring with the JS algorithm, the simulation result in section 3.2.4.1.2 has revealed the highest (among the discussed algorithms for SOPDT) sensitivity to size of the controller test gain K_c, i.e. sensitivity on the magnitude of the overshoot displayed in closed-loop response. Even when chosen gain satisfied the underdamped criterion (i.e. $0<\zeta<1$), the algorithm identified model parameters with completely different dynamics. For example, the algorithm gave model with unstable dynamics in example 1 for K_c values 2 and 1.5, but also produced model parameter with stable dynamics form the same original system when K_c=1. The similar trend is also traced in example 2. Moreover, when applied to systems with time delay, the algorithm failed to reflect the dead time existing in the original process's response, neither displayed inverse response.

The JSDR is primarily a modified JS algorithm. The JSDR showed improved performance and minimized sensitivity on K_c and hence avoided yielding models with complete different dynamics, as proved with simulation results in section 3.2.4.1.2 and 3.3.2.2.2. Additionally, the JSDR estimated models incorporated inverse response to reflect the input delay experienced in systems response as illustrated, among others, in Figure 3-25, Figure 3-27 and Figure 3-29.

Based on the simulation results obtained under similar conditions as for JS and JSDR algorithms, the DR algorithm provided, generally, identified parameters with better and closer response to original dynamics of the systems as shown in simulation results for example 1, 2 and 3 in section 3.2.4.2.2. The resulted model dynamics display almost constant inverse response despite the variation in size of the K_c used during step experimentation. In all simulated examples, the DR algorithm identified models showed underdamped response regardless of the nature of response in the original system. The phenomenon is highlighted in the simulation results obtained in section 3.2.4.2.2, whereby the DR algorithm identified models with oscillatory behaviour while the original system response is overdamped (i.e. contains no oscillations). However, it should be noted that, the amount (size) of the oscillations decreases with decreasing size of K_c.

The DR1 algorithm, on the other hand, identified model parameters, which provided best possible reflection of the original system response. For example, the estimated model obtained from overdamped systems (examples 1, 2 and 3 in section 3.2.4.2.2) also displayed overdamped response, while the algorithm also managed to match the underdamped response of the original system (example 8 in chapter 4.4) with the response of the equivalent dynamics. Additionally, the DR1 algorithm, identified models with inverse response to counteract the dead time experienced by systems, as illustrated in simulation results in section 3.2.4.2.2.

4.5 Time delay and Effect of noise in identification of delay

When no noise involved, the identification of system's time delay during estimation of first-order models (using SID algorithms for first-order models) appeared to be straightforward, since all the algorithms (YS, JR, JL, CLC, and MF) have incorporated time delay τ_p as direct output. The quality or validity of identified time delay has proven to depend on strength of individual algorithm and size of controller gain (and test integral time for MF) as seen in simulation results presented in section 3.2.3. Generally, the JL algorithm identified, in most cases, the most valid delay time (with respect to HR). The JR comes second, while YS, CLC and MF makes the remaining sequence.

Identifying the time delay for second-order approximated models is not as straightforward as for first-order models. The second-order algorithms discussed (JS, JSDR, DR and DR1) do not have delay time as integrated direct output form the algorithms. However, the JSDR, DR and DR1 produced model parameters with inverse response to reflect the time delay in the original system, whereby the magnitude of time delay is appeared to be equivalent to the time taken for the inverse response to change direction (i.e. turn into direct response) as illustrated in Figure 4-2.

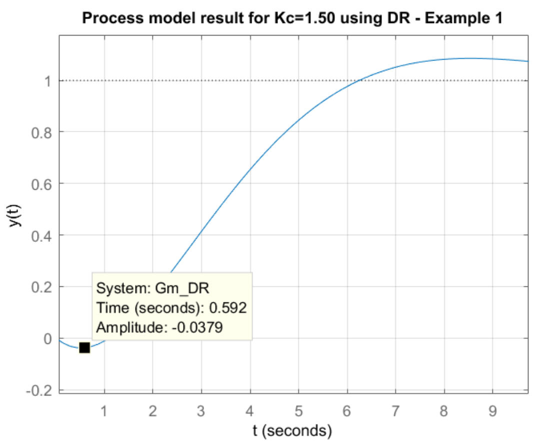


Figure 4-2: Black square indication response turning point

Figure 4-2 shows the point (just graphical approximation) at which the inverse response turns direct response of the identified system. The magnitude of the dead time existing in original system has proven to adhere with the negative ratio of the identified parameters b_1 and b_0 (i.e. $\tau_p = -\frac{b_1}{b_0}$). Based on the simulation results obtained in 3.2.4, the DR provided best possible estimations of time delay, followed by JSDR, and DR1. In almost all simulated examples, the JS algorithm failed complete to identify the input delay experienced in the original systems.

On the other hand, when the random white noise was induced into closed-loop during step test, the algorithms' strength in estimating correct dead time decreased noticeably. Their estimations become no longer reliable as shown in simulation results in chapter 3.3.2.2. For the individual first-order algorithms, JR showed the least effect, followed by YS and then JL. The CLC algorithm appeared to be the most affected among the four. For the second-order algorithms, the DR algorithm showed least effect, though only in few cases (simulated examples) the algorithm provided acceptable time delay. The DR1 comes at second place, however, with unreliable time delay in most of time. Only in few occasions, the JSDR managed to provide estimate for time delay, not necessarily acceptable results though, while JS should not even counted on.

4.6 Using Transient response SID algorithms in auto tuning PID

The PID auto-tuning mechanism in this study was based SIMC settings derived from the model parameters estimated using the discussed system identification algorithms. The effectiveness of the PID-settings depended therefore on the validity and the quality of the estimated model parameters.

Among the discussed first-order SID algorithms, the JL produced, in most cases, the most robust PI-settings that gave stable feedback dynamics with impressive set-point tracking ability. The JL's PI-settings experienced good response speed with no or very minimum overshoot regardless of original systems dynamics behaviour as shown in simulation results in section 3.3.2.2.1. The simulation results for examples 1, 2 and 3 in the same section show also the algorithms good ability in coping with noise signals.

When the PI-settings from YS algorithm were used, the closed-loop response exhibited, in most cases, stable dynamics though with some oscillations. The algorithm produced the fastest response in expense of robust control. However, the algorithm poor ability in coping with noise signal as seen in Table 3.24, Table 3.25 and Table 3.26.

The JR’s PI-settings gave quite robust control and stabilized the closed-loop system. However, the algorithm produced the slowest response with in most cases no overshoot. The CLC algorithms PI-settings were the most unreliable, and many cases produced unstable feedback response as seen in section 3.3.2.2.1. The poor settings are mainly due to the large controller gains K_c used during step-test experimentation, since the algorithm is extremely sensitive to both size of K_c and noise.

Table 4.1: Gain Margin (GM) and Phase Margin (PM) for FOPDT algorithms

	Example 1				**Example 6**				**Example 8**			
K_c	**1.25**		**1.00**		**1.25**		**0.75**		**1.00**		**0.50**	
	GM	**PM**	**GM**	**PM**	**GM**	**PM**	**GM**	**PM**	**GM**	**PM**	**GM**	**PM**
YS	10.11	67.68	9.95	66.61	9.448	65.68	8.34	60.26	8.89	89.78	10.12	87.89
JR	11.79	68.72	12.22	68.84	11.19	65.45	11.49	64.28	10.00	84.75	-	-
JL	10.61	66.03	10.98	66.91	9.95	61.76	10.06	62.14	8.76	82.65	10.88	80.22
CLC	10.66	68.32	10.53	67.89	3.76	164.7	9.82	63.76	19.61	-118	21.77	-104
HR	9.99	59.46	9.99	59.46	9.94	61.35	9.94	61.35	4.35	21.72	4.35	21.72

Table 4.1 gives an overview of Gain Margins (GM) and Phase Margins (PM) for three chosen example with PI-settings obtained by the four algorithms. Note that, the CLC’s PI-settings for both K_c in example 8, and the example 6 when K_c=1.25 gave unstable dynamics. Note also that, JR gave no results for example 8 when K_c=0.5 due to negative integral time and hence invalid PI-settings.

Generally, the JR’s PI-settings gave both largest GM and largest PM, meaning that the JR’s settings had largest stability margins hence most flexible to changes. The JL experienced the second largest GM, hence good flexibility in loop transfer-function gain. The YS experienced, on the contrary, the smallest margins and hence least flexible.

When it comes to second-order algorithms, the DR1 algorithm has proved to be unreliable when the closed-loop response is subjected to noise during step-test. Either provided poor PID-setting or failed to provide PID-settings at all. Based on the simulation results in 3.3.2.2.2, the DR and JSDR when subjected to noise, produced PID-settings with almost identical dynamics behaviour, in most cases. Note that, the JS failed totally, both when and when no noise was involved.

Generally, when no noise was involved, the DR1 provided the most robust PID-settings with underdamped dynamics (i.e. no overshoots) but with slowest response speed. The DR1's PID-settings showed best tracking behaviour as illustrated on the left graphs in Figure 3-52, Figure 3-55, Figure 3-58 and Figure 3-61. The PID-settings from JSDR and DR gave stable closed-loop dynamics with fastest response speed and oscillations. The JSDR had, in most cases, higher oscillation compare to DR.

Table 4.2: Gain Margin (GM) and Phase Margin (PM) for SOPDT algorithms

	Example 1				**Example 2**				**Example 8**			
K_c	**1.25**		**1.00**		**1.25**		**0.75**		**1.00**		**0.50**	
	GM	**PM**	**GM**	**PM**	**GM**	**PM**	**GM**	**PM**	**GM**	**PM**	**GM**	**PM**
JS	-	-	26.21	-139	∞	-153	∞	-133	21.85	-148	24.66	-123
JSDR	6.00	61.68	6.44	60.63	3.00	26.08	8.68	59.13	5.80	47.04	5.32	43.22
DR	5.38	60.44	4.66	55.22	3.68	33.26	7.39	55.52	4.67	40.29	1.66	16.44
DR1	13.68	94.98	14.56	84.66	15.67	74.50	17.43	99.89	9.56	61.55	11.06	65.75
HR	12.77	53.38	12.77	53.38	15.46	54.41	15.46	54.41	9.94	61.35	9.94	61.35

Table 4.2 gives an overview of Gain Margins (GM) and Phase Margins (PM) for three chosen example with PID-settings obtained by the four algorithms when no noise was involved. Note that, the JS algorithm produced unstable response for all tested example with the specified size of K_c.

Generally, the DR1's PID-settings gave both largest GM and largest PM, hence most flexible due to largest stability margins. The JSDR experienced the second largest GM and PM, hence good flexibility in both loop transfer-function gain phase lag function of the loop. The DR experienced, on the contrary, the smallest margins and hence least flexible.

4.7 Strength and Performance and Effect of measurement noise

Generally, the results from Monte Carlo simulations for the first-order model identification algorithms in section 3.4.1 are consistent, and validates the algorithms' performances indicated during individual simulation. The results underlines the JL as the algorithm with the overall strongest performance, followed closely by JR, YS and CLC in the respective order.

However, the results also revealed minimized performance strength encountered by the algorithms when identifying higher-order systems as shown in Table 3.36, Table 3.37 and Table 3.38. The similar trend is also revealed for second-order algorithms as seen in section 3.4.2. The reduced performance is primarily due to algorithms limitations, but is also believed to be associated with the use of HR calculated parameters as reference (real) parameters for the validation criterion. It should be noted that the HR parameters are themselves based on estimation.

The performance results for the second-order algorithms were not as consistent as for the first-order algorithms. Generally, the DR algorithm encountered the strongest performance, followed by closely by JSDR, JS in their respective order. Based on Monte Carlo simulation results, the DR1 displayed the least strong performance of all. However, the Monte Carlo simulations used random inputs of white Gaussian noise, meaning that the performance result of the DR1 algorithm may be misleading since the algorithm is highly sensitive to- and influenced by the noise.

Generally, all the algorithms have displayed reduced strength when exposed to measurement noise in stochastic framework. The extent to which the algorithms are affected by the measurement noise differs. JS and DR1 has proved to be the most prone and delicate to noise amongst the SOPDT algorithm. Their practical application are therefore more suited in deterministic framework rather than stochastic.

5 Conclusion

Five and four algorithms for step-response transient analysis SID for first and second-order models respectively, have been reviewed and their performance been assessed through simulations of numerical examples representing systems of various response dynamics.

JL algorithm's impressive good performance is highlighted, which is also found to withstand largest range of the magnitude of the overshoots (i.e. largest variation of the P-controllers test gain K_c) without compromising much on the quality of estimated parameters. The JR algorithms is also proven to be robust in identifying parameters first-order models plus time delay, and auto-tuning of PI-controller for stable, robust and good set-point tracking behaviour, while the YS algorithm is found to give satisfactory results.

The study has also proven the reduced strength of the YS algorithm when applied to systems with significant large delay time. Moreover, the enormous sensitivity of CLC algorithm to the size of controller's test gain, which is directly associated with size of overshoot of feedback system, has been highlighted. The CLC's strength and good performance is found to be highly favoured with small magnitudes of overshoot, preferably under 0.15. Unfortunately, the MF algorithm has not been properly assessed. Finding the suitable combination of test parameters T_i and K_c for the MF algorithm, has proven to be challenging as it requires several trial and error step-experiments.

For second-order SID algorithms, the study has proven the solid performance and robustness of DR1 algorithm in identifying second-order models, as well as auto-tuning of PID-controller for best possible stable and tracking behaviour of the system. Both the DR and JSDR are found to provide quite acceptable model parameters, with PID-settings that gives stable but oscillatory dynamics, while the JS algorithm is proven to be the lest reliable.

The simulation results has also visualized the effect noise on the performance of algorithms. All the algorithms are proven to display reduced strength when subjected to noise. The DR1 has shown to be the most affected and highly unreliable when applied to noisy closed-loop response. The JS is the second most affected, while DR and JSDR have shown almost identical strength and proven to be the least affected and hence the most reliable in noisy systems.

References

[1] T. Söderström and P. Stoica, *System identification*. New York: Prentice Hall, 1989.

[2] M. Yuwana and D. E. Seborg, "A new method for on-line controller tuning," *AIChE Journal,* vol. 28, pp. 434-440, 1982.

[3] L. Ljung, *System identification : theory for the user*, 2nd ed. ed. Upper Saddle River, N.J: Prentice Hall PTR, 1999.

[4] A. Jutan and E. S. Rodriguez, "Extension of a new method for on-line controller tuning," *The Canadian Journal of Chemical Engineering,* vol. 62, pp. 802-807, 1984.

[5] J. Lee, "On-line PID controller tuning from a single, closed-loop test," *AIChE Journal,* vol. 35, pp. 329-331, 1989.

[6] C.-L. Chen, "A simple method for on-line identification and controller tuning," *AIChE Journal,* vol. 35, pp. 2037-2039, 1989.

[7] O. Taiwo, "Comparison of four methods of on-line identification and controller tuning," *IEE Proceedings D: Control Theory and Applications,* vol. 140, pp. 323-327, 1993.

[8] R. Mamat and P. J. Fleming, "Method for on-line identification of a first order plus dead-time process model," *Electronics Letters,* vol. 31, pp. 1297-1298, 1995.

[9] E. Jahanshahi and S. Skogestad, "Anti-slug control solutions based on identified model," *Journal of Process Control,* vol. 30, pp. 58-68, 2015.

[10] C. Dalen and D. Di Ruscio, "On closed loop transient response system Identification," *Modeling, Identification and Control,* vol. 37, pp. 213-223, 2016.

[11] P. Balaguer, V. Alfaro, and O. Arrieta, "Second order inverse response process identification from transient step response," *ISA Transactions,* vol. 50, pp. 231-238, 2011.

[12] T. Williams, *Modelling Complex Projects*. Hoboken: John Wiley & Sons, Ltd., 2003.

[13] D. Chinarro and SpringerLink, *System Engineering Applied to Fuenmayor Karst Aquifer (San Julián de Banzo, Huesca) and Collins Glacier (King George Island, Antarctica)*. Cham: Springer, 2014.

[14] D. Di Ruscio, "Subspace System Identification - Theory and applications," Sixth ed. Porsgrunn: Telemark University College, 2014.

[15] R. Aarts, "System Identification and Parameter Estimation," 2011/2012 ed. Enschede: University of Twente, 2011.

[16] P. Fritzson, *Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica*. Hoboken: Wiley, 2011.

[17] B. Lie, "Modeling of Dynamic Systems," ed. Porsgrunn: University College of Telemark, 2015.

[18] F. Haugen, *Basic dynamics and control*, [2nd ed.]. ed. Skien: TechTeach, 2010.

[19] H. Viumdal, S. Mylvaganam, and D. Di Ruscio, "System identification of a non-uniformly sampled multi-rate system in aluminium electrolysis cells," *Modeling, Identification and Control,* vol. 35, pp. 127-146, 2014.

[20] L. Ljung, "Perspectives on system identification," *Annual Reviews in Control,* vol. 34, pp. 1-12, 2010.

[21] H.-P. Halvorsen, "System Identification and Estimation," ed. Porsgrunn: University College of Southeast Norway, 2016.

[22] D. J. Murray-Smith and D. J. Murray-Smith, *Modelling and Simulation of Integrated Systems in Engineering : Issues Of Methodology, Quality, Testing And Application*. Burlington: Elsevier Science, 2012.

[23] V. Valdivia, A. Barrado, A. Lazaro, C. Fernandez, and P. Zumel, "Black-box modeling of DC-DC converters based on transient response analysis and parametric identification methods," ed, 2010, pp. 1131-1138.

[24] F. Haugen, *Advanced dynamics and control*. Skien: Techteach, 2010.

[25] L. Ljung, "Prediction error estimation methods," *Circuits, Systems and Signal Processing,* vol. 21, pp. 11-21, 2002.

[26] S. Paluri, S. Patluri, and M. Mossberg, "A Study of Impulse Response System Identification," ed, 2007.

[27] L. Bernt, R. David Di, E. Rolf, G. Bjørn, H. Maths, H. Finn, *et al.*, "Modeling, Identification and Control at Telemark University College," *Modeling, Identification and Control,* vol. 30, pp. 133-147, 2009.

[28] A. K. Tangirala, *Principles of system identification. Theory and practice*. Boca Raton, FL: CRC Press, 2015.

[29] U. Forssell, "Closed-loop identification : methods, theory, and applications," no. 566, Department of Electrical Engineering, Linköping University, Linköping, 1999.

[30] M. J. Quinn, *Parallel programming : in C with MPI and OpenMP,* International ed. ed. Boston: McGraw-Hill Higher Education, 2003.

[31] M. Shamsuzzoha, S. Skogestad, and I. J. Halvorsen, "On-Line PI Controller tuning Using Closed-Loop Setpoint Response," 2010.

[32] S. Sigurd, "Simple analytic rules for model reduction and PID controller tuning," *Modeling, Identification and Control,* vol. 25, pp. 85-120, 2004.

[33] A. Jutan, "A comparison of three closed-loop PID tuning algorithms," *AIChE Journal,* vol. 35, pp. 1912-1914, 1989.

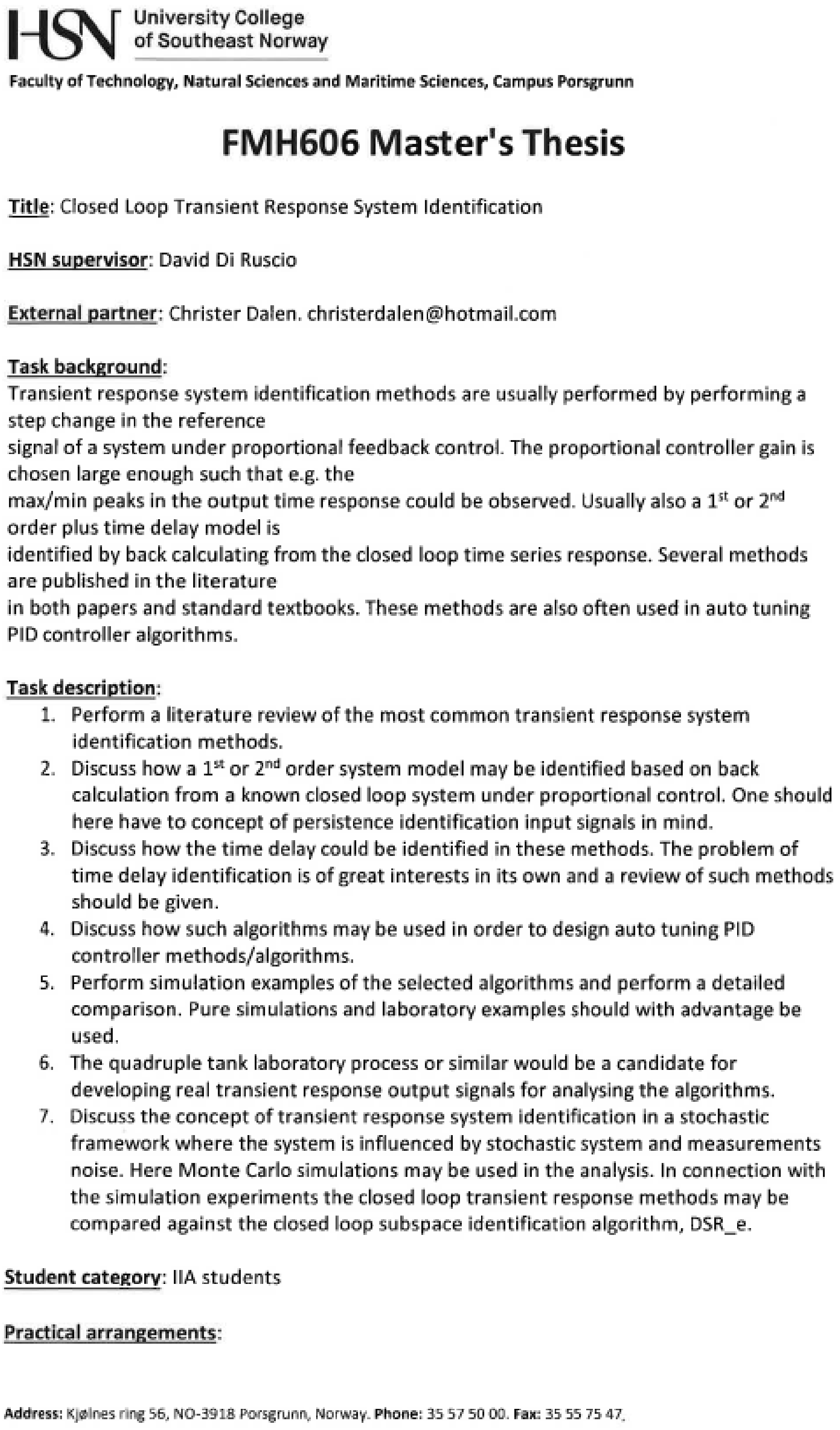
[34] D. Di Ruscio, "System Theory, State Space Analysis and Control Theory," in *Lecture Notes in Control Theory*, August 2015 ed. Porsgrunn: Telemark University College, 2015.

[35] S. Skogestad and C. Grimholt, "The SIMC Method for Smooth PID Controller Tuning," in *PID Control in the Third Millennium: Lessons Learned and New Approaches*, R. Vilanova and A. Visioli, Eds., ed London: Springer London, 2012, pp. 147-175.

[36] K. Ogata, *MATLAB for control engineers*. Upper Saddle River, N.J: Pearson Prentice Hall, 2008.

Appendices

Appendix A: Thesis's Task Description



University College of Southeast Norway

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

FMH606 Master's Thesis

Title: Closed Loop Transient Response System Identification

HSN supervisor: David Di Ruscio

External partner: Christer Dalen. christerdalen@hotmail.com

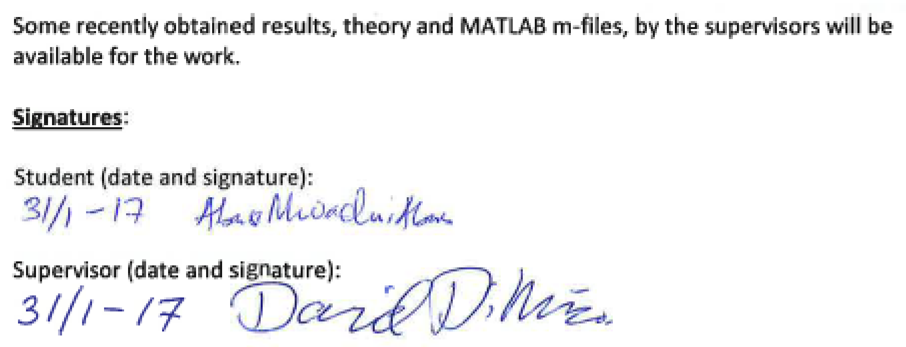
Task background:
Transient response system identification methods are usually performed by performing a step change in the reference
signal of a system under proportional feedback control. The proportional controller gain is chosen large enough such that e.g. the
max/min peaks in the output time response could be observed. Usually also a 1st or 2nd order plus time delay model is
identified by back calculating from the closed loop time series response. Several methods are published in the literature
in both papers and standard textbooks. These methods are also often used in auto tuning PID controller algorithms.

Task description:
1. Perform a literature review of the most common transient response system identification methods.
2. Discuss how a 1st or 2nd order system model may be identified based on back calculation from a known closed loop system under proportional control. One should here have to concept of persistence identification input signals in mind.
3. Discuss how the time delay could be identified in these methods. The problem of time delay identification is of great interests in its own and a review of such methods should be given.
4. Discuss how such algorithms may be used in order to design auto tuning PID controller methods/algorithms.
5. Perform simulation examples of the selected algorithms and perform a detailed comparison. Pure simulations and laboratory examples should with advantage be used.
6. The quadruple tank laboratory process or similar would be a candidate for developing real transient response output signals for analysing the algorithms.
7. Discuss the concept of transient response system identification in a stochastic framework where the system is influenced by stochastic system and measurements noise. Here Monte Carlo simulations may be used in the analysis. In connection with the simulation experiments the closed loop transient response methods may be compared against the closed loop subspace identification algorithm, DSR_e.

Student category: IIA students

Practical arrangements:

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



Some recently obtained results, theory and MATLAB m-files, by the supervisors will be available for the work.

Signatures:

Student (date and signature):
31/1 -17 [illegible]

Supervisor (date and signature):
31/1-17 [illegible]

Appendix B: Individual SID algorithm for FOPDT

YS - Algorithm

```
function [Kp,tau,Tp,Kc_pi,Ti_pi] = sim_pi_yuwana(yp1,ym1,yp2,dT,Kc,A)
%% Transient response SID & Autotuning - Yuwana & Seborg
% Using SIMC settings for 1st order process
% Author: Abass Mwadini Abass
% Date: 10.04.2017 - OK
%**********************************************
%Defining steady state response and zeta
y_inf = ((yp2*yp1-ym1^2)/(yp1+yp2-2*ym1));          % Steady st. res. Eq.13
v_1 = (y_inf-ym1)/(yp1-y_inf);                      % Overshoot
v_2 = (yp2-y_inf)/(yp1-y_inf);                      % Overshoot
zeta_1 = -log(v_1)/sqrt(pi^2+log(v_1)^2);             % Damping ratio Eq.11
zeta_2 = -log(v_2)/sqrt(4*pi^2+log(v_2)^2);           % Damping ratio Eq.12
zeta = mean([zeta_1 zeta_2]);                         % Average zeta

% Back calculating model parameters
Kp = y_inf/(Kc*(A-y_inf));                              % Process gain Eq.8
const_1 = zeta*sqrt(Kc*Kp+1)+sqrt(zeta^2*(Kc*Kp+1)+Kc*Kp);
const_2 = sqrt((1-zeta^2)*(Kc*Kp+1));

Tp = (dT/pi)*const_1*const_2;               % Process T constant Eq.9
tau = (2*dT/pi)*const_2/const_1;            % Process Time delay Eq.10
Tp = real(Tp); tau = real(tau);
[Kc_pi,Ti_pi]= simc_pi(Kp,Tp,tau);             % Skogestad Tuning
```

JR - Algorithm

```
function [Kp,tau,Tp,Kc_pi,Ti] = sim_pi_jutan(yp1,ym1,yp2,dT_j,Kc,R0,R,y0)
%% Transient response SID & Autotuning - Jutan & Rodriguez
% Using SIMC settings for 1st order process
% Author: Abass Mwadini Abass
% Date: 10.04.2017
%**********************************************
% Defining constants
gamma_1=-0.6143; gamma_2=0.1247; delta=0.3866;
%Defining variables
y_inf = (yp2*yp1-ym1^2)/(yp1+yp2-2*ym1);            % Steady st. resp Eq.16
Kp = abs(y_inf-y0)/(Kc*(abs(R-R0)-abs(y_inf-y0)));  % Process gain Eq.9
K = Kc*Kp;                                          % Open-loop gain

% Estimating the damping ratio
alpha_1 = (y_inf-ym1)/(yp1-y_inf);
zeta = -log(alpha_1)/sqrt(pi^2+log(alpha_1)^2);    % Damping ratio Eq.14
T = dT_j*sqrt(1-zeta^2)/pi;                        % closed-loop freq. Eq.8

% Back calculation of model parameters
alpha = 2*zeta*T*(1+K)/(delta+gamma_1*K);            % Eq.26
beta = -1/(delta+gamma_1*K);                         % Eq.27
A_1 = beta^2*gamma_2*K + beta*delta;                 % Eq.23
B_1 = 2*gamma_2*K*alpha*beta + alpha*delta;          % Eq.24
C_1 = gamma_2*K*alpha^2 - T^2*(1+K);                 % Eq.25

Tp = (-B_1+sqrt(B_1^2-4*A_1*C_1))/(2*A_1);           % Eq.21
tau = alpha+beta*Tp;                                 % Time delay Eq.22
[Kc_pi,Ti]= simc_pi(Kp,Tp,tau);
```

JL - Algorithm

```
function [Kp,tau,Tp,Kc_pi,Ti] = sim_pi_lee(yp1,ym1,yp2,dT,Kc,A,y)
%% Transient response SID - Jietae Lee
% Using SIMC settings for 1st order process
% Author: Abass Mwadini Abass
% Date: 10.04.2017
%**************************************************************************
```

```
%Defining variables
y_inf = (yp2*yp1-ym1^2)/(yp1+yp2-2*ym1);       % Steady state res. Eq.13 YS
zeta = -log((y_inf-ym1)/(yp1-y_inf))/...
    sqrt(pi^2+log((y_inf-ym1)/(yp1-y_inf))^2);         % Damping ratio Eq.3

T = (dT/pi)*sqrt(1-zeta^2);                              % Cl.-loop fr. Eq.4
alpha=zeta/T; beta=sqrt(1-zeta^2)/T; v=atan(beta/alpha);     % Lee variables

% Back calculating process model parameters
Kp = y_inf/(Kc*(A-y_inf));                               % Process gain Eq.9
tau(1) = (v+pi/4)/beta;
for i = 1:((length(y)+1)/12)
    tau(i+1) = (1/beta)*(v+atan(beta*exp(-
alpha*tau(i))/(Kc*Kp*sqrt(alpha^2+beta^2)...
        *cos(beta*tau(i)-v))));                  % process Time delay Eq. 12
    tau(i) = tau(i+1);
end
tau = tau(end);
Tp = (1/alpha)*(1+Kc*Kp*exp(alpha*tau)*cos(beta*tau));       % Eq.11
[Kc_pi,Ti]= simc_pi(Kp,Tp,tau);                              % SIMC PI Tuning
```

CLC - Algorithm

```
function [Kp,tau,Tp,Kc_pi,Ti] = sim_pi_chen(yp1,ym1,yp2,tp1,tm1,Kc,A)
%% Transient response SID & Auto Tuning - Cheng-Liang Chen
% Using SIMC settings
% Author: Abass Mwadini Abass
% Date: 10.04.2017
%*************************************************************************
%Defining variables
y_inf = (yp2*yp1-ym1^2)/(yp1+yp2-2*ym1);          % Steady state res. Eq.6
K = y_inf/A;                                      % Closed-loop gain Eq.2
% Estimating the damping ratio (zeta)
H = (1/3)*(((yp1-y_inf)/y_inf)+((y_inf-ym1)/(yp1-y_inf))...
    +((yp2-y_inf)/(y_inf-ym1)));                       % Overshoot Eq.7
zeta = -log(H)/sqrt(pi^2+log(H)^2);                    % Damp- ratio Eq.3
T = ((tm1-tp1)*sqrt(1-zeta^2))/pi;                     % Cl.loop freq Eq.4
d = 2*tp1-tm1;                                         % Cl.loop delay Eq.5

% Estimating crossover frequency (omega_c)
w_c = 0;
w_c = fsolve(@f_omega,w_c,optimset('Display','off'));
function func = f_omega(omega_c)
    func = -d*omega_c-atan(2*zeta*T*omega_c/...
        sqrt(1-T^2*omega_c^2))+pi;                    % Eq 8
end

% Calculating ultimate loop gains, gain margin, and ultimate controler gain
G_cl = K/sqrt((1-T^2*w_c^2)^2+(2*zeta*T*w_c)^2);   % Cl.ult. Eq.9
G_op = G_cl/sqrt(1+2*G_cl+G_cl^2);                         % Op.ult. Eq.10
GM = 1/G_op;
K_cu = Kc*GM;                                  % Ult. controller gain Eq.11

% Estimating transfer model parameters
Kp = y_inf/(Kc*(A-y_inf));                         % Process gain Eq.15
Tp = (1/w_c)*sqrt((K_cu^2*Kp^2 -1)); % Eq.13
tau = (1/w_c)*(pi-atan(Tp*w_c)); % Eq.14

Tp = real(Tp); tau = real(tau);
[Kc_pi,Ti]= simc_pi(Kp,Tp,tau);                    % SIMC Tuning rules
end
```

MF – Algorithm

```
function [Kp,tau,Tp,Kc_pi,Ti_pi] = sim_pi_mamat(yp1,ym1,yp2,tp1,tp2,Kc,A,y,Ti,Tt)
%% Transient response SID & PID Auto Tuning - Mamat & Flemming
% Using SIMC Tuning rules
% Author: Abass Mwadini Abass
% Date: 10.04.2017
%**********************************************
%Defining variables
y_inf = (yp2*yp1-ym1^2)/(yp1 + yp2 -2*ym1);             % Steady state
K = y_inf/A;                                            % Cl. loop gain Eq.4
phi = -log((yp2-y_inf)/(yp1-y_inf))/(2*pi);             % Overshoot Eq.8
zeta = sqrt(phi^2/(1+phi^2));                           % Damping ratio Eq.5
T = (tp2-tp1)*sqrt(1-zeta^2)/(2*pi);                    % Cl. loop freq Eq.6
S_c = trapz(Tt,(y_inf*ones(length(y),1)-y));            % Eq. 9
d = (S_c/y_inf)-2*zeta*T;                               % Resp. delay Eq.7

% Estimating crossover frequency (omega_c)
w_c = 0;
w_c = fsolve(@f_omega,w_c,optimset('Display','off'));
function func = f_omega(omega_c)
    func = d*omega_c+atan(2*zeta*T*omega_c/...
        (1-T^2*omega_c^2))-pi;                           % Eq 12
end

% Ultimate loop gains, gain margin, and ultimate controler gain
M = K/sqrt((1-T^2*w_c^2)^2+(2*zeta*T*w_c)^2); %Cl.lp ult.G Eq.13
% Estimating process model parameters
Kp = Ti*y_inf/(Kc*S_c);                                     % Gain Eq.16
Tp = sqrt((Kc*Kp)^2*(1+Ti^2*w_c^2)-M^2*Ti^2*w_c^2)...
    /(M*w_c^2*Ti);                                       % T.const. Eq.14
tau = (1/w_c)*(atan(Ti*w_c)+atan(1/(T*w_c)));               % T delay Eq.15
Tp = real(Tp); tau = real(tau);
[Kc_pi,Ti_pi]= simc_pi(Kp,Tp,tau);                          % SIMCS settings

end
```

Utility – Critical points from closed-loop response

```
function [y,yp1,ym1,yp2,tp1,tm1,tp2,dT,dT_j,Tt]= response_data1(Fs)
T = 0:0.1:60;
% Defining the measure quantities based on the response graph
[y,Tt]=step(Fs,T);                                  % Step response
[yp1,ip] = max(y);                                 % Response value at 1st overshoot
tp1 = Tt(ip);                                      % Response Time at 1st overshoot
[ym1,im] = min(y(ip:end));                         % Response value at 1st overshoot
tm1 = Tt(ip+im-1);                                 % Response Time at 1st undershoot
[yp2,iu]= max(y((ip+im-1):end));                   % Response value at 2nd overshoot
tp2 = Tt(ip+im+iu-1);                              % Response Time at 2nd overshoot
dT = tm1-tp1;                                      % Half period Eq. A-11
dT_j = (tp2-tp1)/2;                                % Half period Jutan Reponse graph
```

Appendix C: Individual SID Algorithms for SOPDT

JS – Algotithm

```
function [K_2,T_z,T,a_0,a_1,b_0,b_1] = sim_jahanshahi(yp1,ym1,tp1,tm1,Kc,R,y)
%% Transient response SID - Jahanshahi & Skogestad
% Author: Abass Mwadini Abass
% Date: 07.04.2017
%**********************************************
% Defining constants
y_inf=y(end);y_s=R;y_p=yp1;y_u=ym1;t_p=tp1;t_u=tm1;

% Damping ratio & gain
v = (y_inf-y_u)/(y_p-y_inf);                          % Eq. A.9
zeta = -log(v)/sqrt(pi^2+log(v)^2);                   % Eq. A.8
T = t_u*sqrt(1-zeta^2)/pi;                            % Eq. A.10
K_2 = y_inf/y_s;                                      % Eq. A.11
phi = atan((1-zeta^2)/zeta)-(t_p*sqrt(1-zeta^2)/T);   % Eq. A.12
E = sqrt(1-zeta^2)/T;                                 % Eq. A.13
D_0 = (y_p-y_inf)/y_inf;                              % Eq. A.14
D = D_0/(exp(-zeta*t_p/T)*sin(E*t_p+phi));            % Eq. A.16
T_z = zeta*T+sqrt(zeta^2*T^2-T^2*(1-D^2*(1-zeta^2))); % Eq. A.17

% back calculation
Kp = y_inf/(Kc*(abs(y_s-y_inf)));                     % Eq. A.18
a_0 = 1/(T^2*(1+Kc*Kp));                              % Eq. A.19
b_0 = Kp*a_0;                                         % Eq. A.20
b_1 = K_2*T_z/(Kc*T^2);                               % Eq, A.21
a_1 = -2*zeta/T+Kc*b_1;                               % Eq. A.22
% Non-complex numbers
a_0 = real(a_0);
a_1 = real(a_1);
b_0 = real(b_0);
b_1 = real(b_1);
```

JSDR – Algorithm

```
function [K,T_z,T,a_0,a_1,b_0,b_1] = sim_modJahanshahi(yp1,ym1,tp1,tm1,Kc,R,y)
%% Transient response SID - Modified Jahanshahi & Skogestad
% Author: Abass Mwadini Abass
% Date: 07.04.2017
%**********************************************
% Defining constants
y_inf=y(end);y_s=R;y_p=yp1;y_u=ym1;t_p=tp1;t_u=tm1;

% Damping ratio & gain
v = (y_inf-y_u)/(y_p-y_inf);                          % Eq.10
zeta = -log(v)/sqrt(pi^2+log(v)^2);                   % Eq.9
T = t_u*sqrt(1-zeta^2)/pi;                            % Eq.16
K = y_inf/y_s;                                        % Eq.11
omega = sqrt(1-zeta^2)/T;                             % Eq.7
phi = atan(sqrt(1-zeta^2)/zeta)-t_p*omega;            % Eq.13

E = sqrt(1-zeta^2)/T;                                 % Eq.A.4
D_0 = (y_p-y_inf)/y_inf;                              % Eq.14
D = D_0/(exp(-zeta*t_p/T)*sin(E*t_p+phi));            % Eq.A.16
T_z = zeta*T+sqrt(zeta^2*T^2-T^2*((zeta^2-1)*D^2+1)); % Eq.20 A
%T_z = zeta*T-sqrt(zeta^2*T^2-T^2*((zeta^2-1)*D^2+1));  % Eq.20 B
T_z = real(T_z);

% back calculation
Kp = y_inf/(Kc*abs(y_s-y_inf));                       % Eq. A.18
a_0 = 1/(T^2*(1+Kc*Kp));                              % Eq. A.19
b_0 = Kp*a_0;                                         % Eq. A.20
b_1 = K*T_z/(Kc*T^2);                                 % Eq, A.21
a_1 = -2*zeta/T+Kc*b_1;                               % Eq, A.21
```

DR – Algorithm

```
function [K,T_z,T,a_0,a_1,b_0,b_1] = sim_dalen(yp1,ym1,tp1,tm1,Kc,R,y)
%% Transient response SID - Dalen & Russio
% Author: Abass Mwadini Abass
% Date: 07.04.2017
%**********************************************
% Defining constants
y_inf=y(end);y_s=R;y_p=yp1;y_u=ym1;t_p=tp1;t_u=tm1;

% Damping ratio & gain
v = (y_inf-y_u)/(y_p-y_inf);                        % Eq.10
zeta = -log(v)/sqrt(pi^2+(log(v)^2));               % Eq.9
T = t_u*sqrt(1-zeta^2)/pi;                          % Eq.16
K = y_inf/y_s;                                      % Eq.11

omega = sqrt(1-zeta^2)/T;
gamma = exp(t_p*zeta/T)*(y_p-R*K)/(R*K);
c = (-cos(t_p*omega)+gamma)/sin(t_p*omega);         % Eq.23

T_z = zeta*T-c*T*sqrt(1-zeta^2);                    % Eq.25
%Kp = y_inf/(Kc*(abs(y_s-y_inf)));                  % Eq. A.18

% Back calclulations
b_0 = K/(T^2*Kc);                                 % Eq. A.20
b_1 = T*b_0;                                      % Eq, A.21
a_0 = 1/T^2-Kc*b_0;                               % Eq. A.19
a_1 = -2*zeta/T+Kc*b_1;                           % Eq. A.22
```

DR1 - Algorithm

```
function [K,T_z,T,a_0,a_1,b_0,b_1] = sim_dalen2(yp1,ym1,tp1,tm1,Kc,R,y)
%% Transient response SID - Dalen & Russio 2
% Author: Abass Mwadini Abass
% Date: 07.04.2017
%**********************************************
% Defining constants
y_inf=y(end);y_s=R;y_p=yp1;y_u=ym1;t_p=tp1;t_u=tm1;

% Damping ratio & gain
v = (y_inf-y_u)/(y_p-y_inf);                        % Eq.10
zeta = -log(v)/sqrt(pi^2+(log(v)^2));               % Eq.9
T = t_u*sqrt(1-zeta^2)/pi;                          % Eq.16
K = y_inf/y_s;                                      % Eq.11

omega = sqrt(1-zeta^2)/T;                           % Eq.7
c = (zeta+T*omega*tan(t_p*omega))/...
    (T*omega-zeta*tan(t_p*omega));                  % Eq.28

T_z = zeta*T-c*T*sqrt(1-zeta^2);                    % Eq.25
Kp = y_inf/(Kc*(abs(y_s-y_inf)));                   % Eq. A.18

% Back calclulations
a_0 = 1/(T^2*(1+Kc*Kp));                          % Eq. A.19
b_0 = Kp*a_0;                                     % Eq. A.20
b_1 = K*T_z/(Kc*T^2);                             % Eq, A.21
a_1 = -2*zeta/T+Kc*b_1;                           % Eq. A.22
```

Appendix D: Monte Carlo Simulation

```
%% MONTE CARLO - FOPDT
% Random Gaussian white noise of y = awgn(y,30,'measured')
% Example 1 for Kc=1.75 and Kc=1.25
% Example 2 for Kc=1.25 and Kc=0.75
% Example 7 for Kc=1.00 and Kc=0.75
%**************************************************************************

% Initail parameters
T=0:0.1:60; R0=0; R=1; Y=step(Fs,T);
N=1; M=100; z_vec=zeros(M,1);cr_sum=0;

% Initiating parameter vectors
kp_ys=z_vec; tau_ys=z_vec; tp_ys=z_vec;            % For Yuwana & Seborg
kp_jr=z_vec; tau_jr=z_vec; tp_jr=z_vec;            % For Jutan & Rodriguez
kp_jl=z_vec; tau_jl=z_vec; tp_jl=z_vec;            % For Lee
kp_clc=z_vec; tau_clc=z_vec; tp_clc=z_vec;         % For Chen

% For validation
diff_param_ys = zeros(M,3);diff_param_jr = zeros(M,3);
diff_param_jl = zeros(M,3);diff_param_clc = zeros(M,3);
sum_ys=0;sum_jr=0;sum_jl=0;sum_clc=0;
real_param = [Kp_hf,Tp_hf,tau_hf];

for i = 1:M;

    % Calculate critical points on the response
    [y,yp1,ym1,yp2,tp1,tm1,tp2,dT,dT_j,Tt]= response_data1_monte(Fs,T);
    % Yuwana & Seborg
    [Kp_YS,tau_YS,Tp_YS] = sim_pi_yuwana(yp1,ym1,yp2,dT,Kc,A); % YS
    kp_ys(i)=Kp_YS; tau_ys(i)=tau_YS; tp_ys(i)=Tp_YS;    % Storing YS param
    diff_param_ys(i,:) =[Kp_YS,Tp_YS,tau_YS]-real_param; % For validation
    sum_ys = sum_ys+diff_param_ys(i,:)*diff_param_ys(i,:)';
    % Jutan & Rodriguez
    y0 = y(1);
    [Kp_JR,tau_JR,Tp_JR] = sim_pi_jutan(yp1,ym1,yp2,dT,Kc,R0,R,y0); % JR
    kp_jr(i)=Kp_JR; tau_jr(i)=tau_JR; tp_jr(i)=Tp_JR;    % Storing JR param
    diff_param_jr(i,:) =[Kp_JR,Tp_JR,tau_JR]-real_param; % For validation
    sum_jr = sum_jr+diff_param_jr(i,:)*diff_param_jr(i,:)';
    % Jitae Lee
    [Kp_JL,tau_JL,Tp_JL] = sim_pi_lee(yp1,ym1,yp2,dT,Kc,A,y); % JL
    kp_jl(i)=Kp_JL; tau_jl(i)=tau_JL; tp_jl(i)=Tp_JL;    % Storing JL param
    diff_param_jl(i,:) =[Kp_JL,Tp_JL,tau_JL]-real_param; % For validation
    sum_jl = sum_jl+diff_param_jl(i,:)*diff_param_jl(i,:)';
    % Cheng-Liang Chen
    [Kp_CLC,tau_CLC,Tp_CLC] = sim_pi_chen(yp1,ym1,yp2,tp1,tm1,Kc,A);
    kp_clc(i)=Kp_CLC; tau_clc(i)=tau_CLC; tp_clc(i)=Tp_CLC; % Stor. CLC par
    diff_param_clc(i,:) =[Kp_CLC,Tp_CLC,tau_CLC]-real_param; % For validation
    sum_clc = sum_clc+diff_param_clc(i,:)*diff_param_clc(i,:)';
end
P_ys = sum_ys*N/(M-1),P_jr = sum_jr*N/(M-1),
P_jl = sum_jl*N/(M-1),P_clc = sum_clc*N/(M-1),

figure()
k_i = 1:1:100;
plot(k_i,tp_ys,'*r',k_i,tp_jr,'+b',k_i,tp_jl,'xg',k_i,tp_clc,'*c')
title('Process parameter estimate for time-constant T_p');
xlabel('simulations (M)'); ylabel('T_p'); legend('YS','JR','JL','CLC')
grid on, %ylim([0 20]);

figure()
plot(k_i,tau_ys,'*r',k_i,tau_jr,'+b',k_i,tau_jl,'xg',k_i,tau_clc,'*c')
title('Process parameter estimate for time delay \tau_p');
xlabel('simulations (M)'); ylabel('\tau_p'); legend('YS','JR','JL','CLC')
grid on, %ylim([-1 10]);

figure()
```

```
plot(k_i,kp_clc,'*k',k_i,kp_ys,'*r',k_i,kp_jr,'+b',k_i,kp_jl,'xg')
title('Process parameter estimate for static gain K_p');
xlabel('simulations (M)'); ylabel('K_p'); legend('CLC','YS','JR','JL')
grid on, %ylim([0.5 2]);
```

Utility for Monte Carlo Simulation

```
function [y,yp1,ym1,yp2,tp1,tm1,tp2,dT,dT_j,Tt]= response_data1_monte(Fs,T)

% Defining the measure quantities based on the response graph
[y,Tt]=step(Fs,T);                        % Step response
y = awgn(y,30,'measured');                % Random Gausssian white noise
[yp1,ip] = max(y);                        % Response value at 1st overshoot
tp1 = Tt(ip);                             % Response Time at 1st overshoot
[ym1,im] = min(y(ip:end));                % Response value at 1st overshoot
tm1 = Tt(ip+im-1);                        % Response Time at 1st undershoot
[yp2,iu]= max(y((ip+im-1):end));          % Response value at 2nd overshoot
tp2 = Tt(ip+im+iu-1);                     % Response Time at 2nd overshoot
dT = tm1-tp1;                             % Half period Eq. A-11
dT_j = (tp2-tp1)/2;                       % Half period Jutan Reponse graph
```