NTNU – Trondheim
Norwegian University of
Science and Technology

SPECIALIZATION PROJECT 2017

TKP4580

PROJECT TITLE:

Dynamic Extremum Seeking Control Using ARX Model

By

Jeongrim Ryu

Supervisor for the project: Sigurd Skogestad

Co-Supervisor for the project: Dinesh Krishnamoorthy

Date: 15/12/2017

Preface

This project was written for TKP4580 Chemical engineering, specialization project at Norwegian University of Science and Technology in Autumn 2017.

This project investigates extremum seeking control, which is also called self-optimizing control. Previously studied classical extremum seeking control strategies were struggling to find an extremum or a maximum point by gradient estimation using a local linear static model. These approaches tend to have a slow dynamic to find the new extremum after being disturbed. Thus, a new method is introduced in this project to estimate the gradient using a local linear dynamic model, which is ARX model.

Hammerstein and Wiener models are used to describe the plant in a simple way, and plants with different kinds of transfer functions are studied to implement the dynamic extremum seeking control. Simulation results are tuned by trial and error method, and several issues of this model are tried to be solved. The best simulation results are compared to results obtained by the classical extremum seeking control, validating the performance of the dynamic extremum seeking control.

This project is greatly interesting since not only it is deeply related to the Advanced process control course but also the implementation of the ARX model for gradient estimation makes a clear improvement on the extremum seeking control. I would like to appreciate my supervisor, Sigurd Skogestad, and co-supervisor, Dinesh Krishnamoorthy, for offering me the opportunity to do this project. I am grateful for sharing their knowledge and answering my questions throughout this project.

12-2017, Trondheim, Norway

Jeongrim Ryu

Contents

List of Tables

List of Figures

1 Introduction

In oil production, gas lift on oil wells is one of the most popular techniques in artificial lift methods to improve oil production rate[1]. In specific, gas is compressed first and injected into the tubing. Due to the fact that gas mixture has a lower density than the fluid density, this newly formed multi-phase flow by gas lift decreases down-hole pressure and consequently increases the oil production rate. A simple sketch of a gas lift oil well is illustrated in Figure 1 [1].

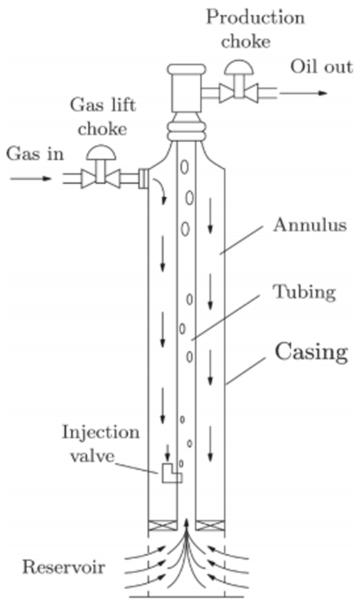

Figure 1: A gas lift oil well

However, excessive gas injection has a counter-active effect on oil production rate. This is because increasing the gas injection rate produces more frictional pressure drop[2]. This leads to the relationship between the gas injection rate and the production rate having a maximum point, so-called an "extremum". Aliev et al.[3] published their study on mathematical modeling of gas lift process, and the dependence of oil production rate on the gas injection rate is clearly shown in Figure 2[3].

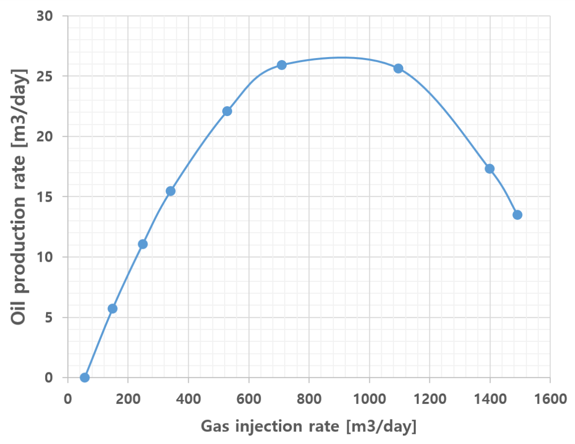

Figure 2: The dependence of oil production rate on the gas injection rate

With this background, this kind of problem where the input-to-output relationship has an extremum and keeping the output around the maximum value is the main issue is called extremum seeking which is a model-free real-time optimization method firstly developed in 1922[5]. Extremum seeking controller has developed towards faster and exact convergence with sufficient stability[2][5][6][7].

A classic extremum seeking controller optimizes the steady state gradient in real time using a locally linear static model. The main drawback is that using transient measurements for the estimation of steady-state gradient. For accurate steady-state gradient estimation, the process has to settle down to steady state before it can be used. Almost all extremum seeking algorithm today assume a local linear static model to estimate the steady state gradient around the current operating time which makes the convergence to the optimum very slow. This issue is the motivation to start this project to figure out a new algorithm to make faster convergence in the extremum seeking controller.

The main idea to achieve the objective is using a local linear dynamic model to estimate the steady state gradient providing a faster update of the estimated gradient instead of a local linear static model. Among diverse dynamic models, ARX model is introduced and it is named as a "Dynamic extremum seeking controller".

Plant models with the first-order and second-order transfer functions are used to see how the implemented ARX model works. For each of them, a large number of simulations are performed with different tuning parameters such as a controller gain, a window size of the input and output data vector for the gradient estimation, and a perturbation wave frequency by the trial and error method.

In the simulation of dynamic extremum seeking algorithm, one problem happened in the second-order transfer function system that the estimated steady-state gradient does not follow the true gradient. This is solved by introducing a PRBS perturbation wave instead of a sinusoidal wave. The other problem is numerical spikes problem which is resulted from the ARX model since the change in the output is so small near the extremum that the ARX model has problems on estimating parameters making numerical errors. This is solved by combining the ARX model and the least square method.

Comparing the simulation results of the dynamic extremum seeking controller obtained after solving the problems with the classic extremum seeking controller validates the successful performance of the dynamic ESC as expected.

Additionally, several future works are suggested at the end of this project. The first work is introducing a disturbance on the tuned systems and monitoring the performance of the dynamic extremum seeking controller. The second work is the implementation of plant system to a more realistic case study. The last try is replacing a simple integral controller with a model predictive controller(MPC) in a multi-variable system.

2 Theory

2.1 Classic extremum seeking controller

The concept of extremum seeking controller(ESC) was emerged in 1922 by Leblanc who introduced the adaptive control method for the first time[5]. In 2000, Krstic and Wang[5] well provided "the first rigorous proof of stability for an extremum seeking feedback scheme" and the ESC obtained a lot of popularity after that[6]. These days, the ESC using gradient estimation is one of the most popular approaches. In specific, the classic ESC suggested by Krstic and Wang[5] optimizes the steady state performance of the output in real time by adding an external dither signal on the input.

Figure 3 shows the basic structure of the classic ESC process which is a gradient-based model in a discrete time setting with a sampling time T_s[6]. Overall, the process includes a non-linear plant model, high pass filter(HPF), low pass filter(LPF), gradient estimation unit, integral controller($C(s)$) and a sinusoidal signal($asin\omega t$). The main procedure to find the extremum is estimating the gradient based on y values for each step, and using a controller(integral controller in this case) with a set-point of zero.

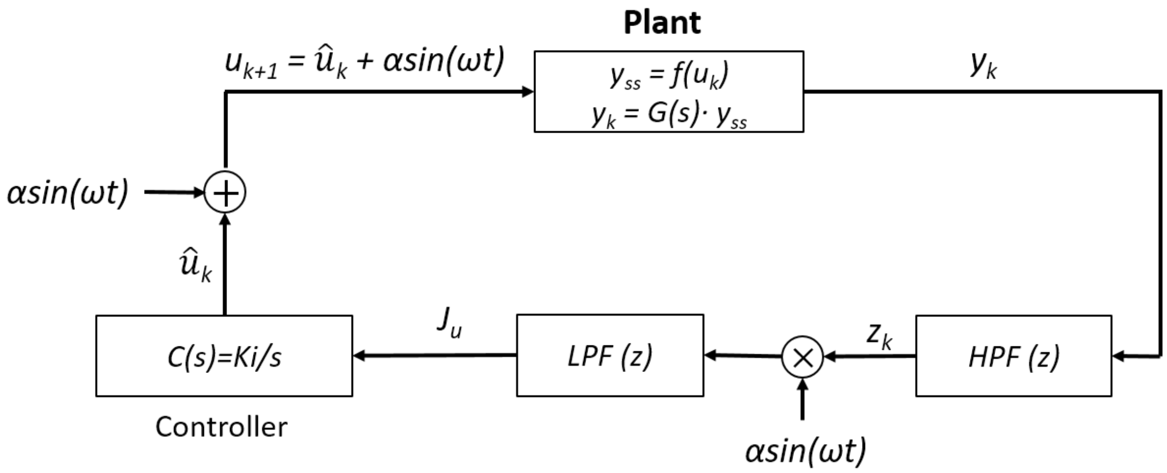

Figure 3: Classic ESC scheme implemented in discrete time

In terms of the plant model block, it consists of two parts, $y_{ss} = f(u_k)$ having a static non-linearity and a linear dynamic system $G(s)$[5]. This kind of plant model is called Hammerstein and Wiener models[4] as shown in Figure 4. The static gain of the plant($\frac{\delta y}{\delta u}$) should be maintained to be constant in one of the two blocks.

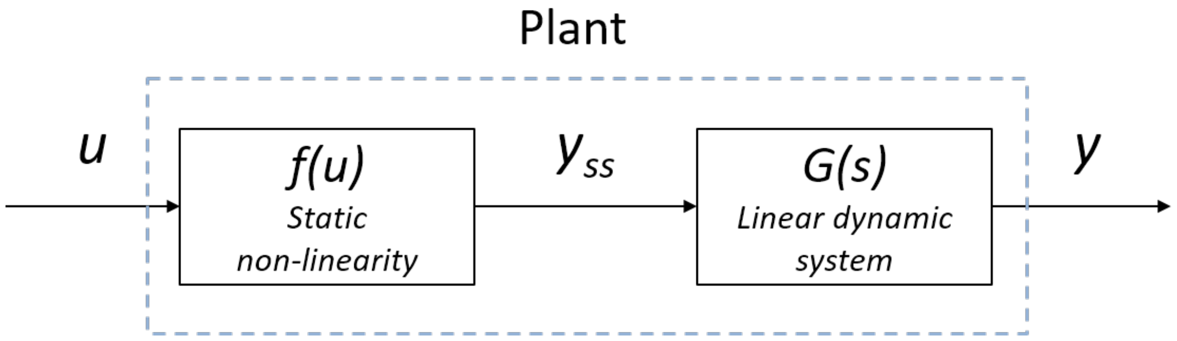

Figure 4: Hammerstein and Wiener models

Equation (1) to (3) is provided by Krishnamoorthy et al.[6], explaining the high pass filter, the low pass filter, and the estimated optimizing variable. When the output y enters the high pass filter, signals with higher frequency than a cut-off time period T_h can only pass the filter removing a low-frequency part, which can also be explained as the DC component of y is subtracted and y_k is moved to have zero mean[5]. Then $asin\omega t$ is multiplied resulting in a signal having two sinusoidals, and it enters the low pass filter with a cut-off time constant T_l where a DC component of the two sinusoidals J_u comes out[5] which is an estimated gradient. Finally, the integral controller generates an estimated input $\hat{u_k}$ and a new input is updated with sine perturbations.

$$z_k = \frac{T_h}{T_s + T_h}[z_{k-1} + y_k - y_{k-1}] \tag{1}$$

$$J_u = (1 - \frac{T_s}{T_s + T_l})J_{u,k-1} + \frac{T_s}{T_s + T_l} z_k \alpha sin(\omega t) \tag{2}$$

$$\hat{u}_k = u_{k-1} + T_s K_i J_u \tag{3}$$

In terms of a relationship between frequencies, the dither frequency ω should be positioned between the high and low pass filter frequencies. Thus, the dither time constant(T_D) is located between the time constants for the two filters, T_h and T_l.

However, there are some drawbacks in the classic ESC[6]. The first disadvantage in the classic ESC is its slow transients to the new optimum after being disturbed. This slow dynamic is due to the fact that the input(u_k) and output(y_k) data are dynamic data but the classic ESC uses a local linear static model to estimate a steady state gradient. Specifically, the local linear static model uses only a part data reached a steady state and throws others away so that it cannot efficiently estimate a nonlinear dynamic plant system. As already mentioned in the chapter 1, this is the motivation to start this project. The second advantage it that the classic ESC loses its robustness when there are disturbances with large amplitudes which changes frequently, which is well modified by Krishnamoorthy et al.[6] by introducing a disturbance rejection block.

Even though various kinds of the ESC have been introduced such as using 1st-order least-square method for gradient estimation[7], all of them has the same disadvantage caused from applying a static model to estimate a dynamic system.

Thus, introducing a dynamic model to estimate a nonlinear dynamic system in the ESC will achieve much faster transients to the new optimum. This new approach is named as a "Dynamic extremum seeking controller" or simply a dynamic ESC.

2.2 Dynamic extremum seeking controller

Overall, the dynamic ESC is based on the same idea with the classic ESC that the gradient is estimated first, and a controller updates a new input value which can achieve the zero set-point for the gradient.

The main difference of the dynamic ESC from the classic ESC is the gradient estimation method. In detail, the slow transient to a new optimum in the classic ESC results from the gradient estimation part. As already mentioned in the chapter 2.1, the data of u_k and y_k are dynamic but the classic ESC uses a static model. Based on this fact, using a dynamic model instead of a static model is considered since it can handle the input and output data more efficiently for the gradient estimation. To realize this idea, a local linear time-invariant model is considered in the dynamic ESC which is expected to achieve faster update of the estimated gradient.

Figure 5 illustrates a schematic control structure for the dynamic ESC. As the classic ESC suggested by Krstic and Wang[5], Hammerstein and Wiener models are used to describe a plant in a simple way. In terms of the perturbation, $\alpha sin\omega t$ is applied firstly, however, it is replaced with a pseudo-random binary sequence wave in second order system which will be explained in the chapter 3.2.

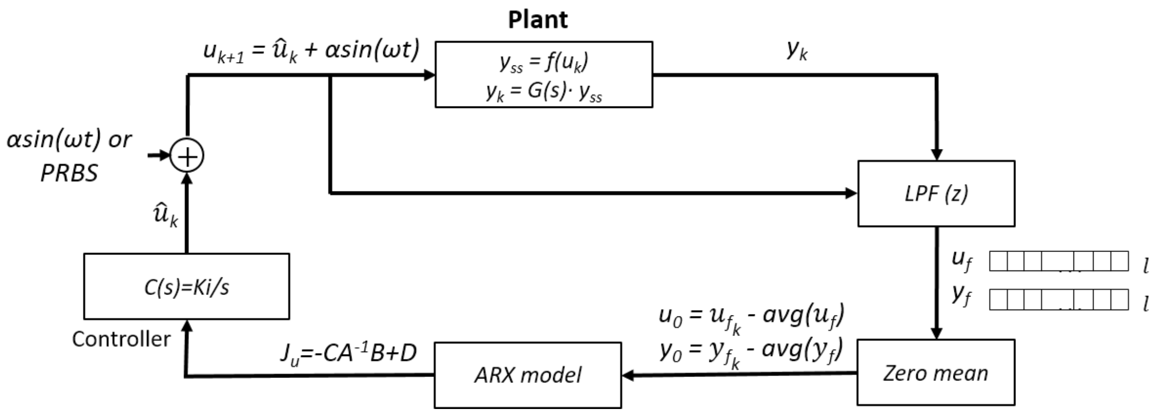
Plant
$u_{k+1} = \hat{u}_k + \alpha sin(\omega t)$
$y_{ss} = f(u_k)$
$y_k = G(s) \cdot y_{ss}$
y_k
$\alpha sin(\omega t)$ or PRBS
$\hat{u}_k$
LPF (z)
u_f
y_f
l
l
C(s)=Ki/s
Controller
$u_0 = u_{f_k}$ - avg(u_f)
$y_0 = y_{f_k}$ - avg(y_f)
$J_u = -CA^{-1}B+D$
ARX model
Zero mean

Figure 5: Dynamic ESC scheme using ARX method

Algorithm 1 The dynamic ESC using ARX model

```
input: $y_k$,$u_k$
1: for $k = 1 \to n$ do
2:     Low pass filter to remove noise
3:         $y_{fk} \leftarrow G_{LPF}(s)y_k$
4:         $u_{fk} \leftarrow G_{LPF}(s)u_k$
5:     Moved to have a zero mean
6:         $y_0 \leftarrow y_{fk} - avg(y_{fk})$
7:         $u_0 \leftarrow u_{fk} - avg(u_{fk})$
8:     Gradient estimation with the ARX model using data sets of $y_0$ and $u_0$
9:         $data \leftarrow iddata(y_0, u_0, T_s)$
10:        $[A(q) \quad B(q)] \leftarrow arx[data, [1, 1, 0]]$    for 1st-order systems
11:        $[A(q) \quad B(q)] \leftarrow arx[data, [2, 2, 0]]$    for 2nd-order systems
12:        $[A_d \quad B_d \quad C_d \quad D_d] \leftarrow idss[A(q) \quad B(q)]$
13:        $[A \quad B \quad C \quad D] \leftarrow d2c[A_d \quad B_d \quad C_d \quad D_d]$
14:        $J_u \leftarrow -CA^{-1}B + D$
15:    Integral Controller
16:        $\hat{u_k} \leftarrow u_k + K_i J_u$
17:    Perturbation
18:        $u_{k+1} \leftarrow \hat{u_k} + \alpha sin\omega t$
19: end for
```

A basic algorithm for the dynamic ESC is displayed in Algorithm 1 which is describing the dynamic ESC procedures performed in Figure 5. Specifically, the input and output values in the plant are filtered out by low pass filter and sets of data with a window size(l) are built up. They pass the zero mean unit where the average value is subtracted from the data sets and they are moved to have a zero mean. After that, the ARX model estimates parameters in a dynamic discrete time system, and converting it into a dynamic continuous time state-space system gives A, B, C, and D matrices. Based on the matrices, the steady state gradient is calculated.

Main advantages of the dynamic ESC are (1) it uses all dynamic data for gradient estimation, (2) it is possible to estimate a controller gain and time constants in transfer functions, (3) it is possible to obtain true gradient values rather than just the sign of the gradient, and (4) it converges faster to a new extremum.

A deep understanding of the dynamic ESC requires some knowledge on the ARX model which is the most important idea and basis for this project, which will be stated in the chapter 2.3.

2.3 ARX model

Selecting models of dynamical systems is well identified by Ljung[9]. A basic linear dynamic model with additive disturbance in discrete time system is specified as:

$$y(t) = G(q)u(t) + H(q)e(t) \tag{4}$$

where

$$G(q) = \sum_{k=1}^{\infty} g(k)q^{-k}, \quad \text{and} \quad H(q) = 1 + \sum_{k=1}^{\infty} h(k)q^{-k}$$

In Equation (4), $G(q)$ describes a relationship between the input $u(t)$ and the output $y(t)$, and it is called the transfer function. The other term $H(q)e(t)$ is an additive term at the output comes from the disturbances where $e(t)$ is white noise.

In a finite number of values, $g(k)$ and $h(k)$ can be considered as coefficients which should be determined, and they can simply denoted by the vector θ. Thus the model can be described as:

$$y(t) = G(q, \theta)u(t) + H(q, \theta)e(t) \tag{5}$$

ARX model is a family of transfer function models which parametrizing $G(q, \theta)$ and $H(q, \theta)$. It starts with specifying the input and output relationship as a linear difference equation:

$$y(t) + a_1 y(t-1) + ... + a_{n_a} y(t-n_a) = b_1 u(t-1) + ... + b_{n_b} u(t-n_b) + e(t) \tag{6}$$

where

$$\theta = [a_1 \quad a_2 \quad ... \quad a_{n_a} \quad b_1 \quad b_2 \quad ... \quad b_{n_b}]^T$$

Additionally, by introducing some terms:

$$A(q) = 1 + a_1 q^{-1} + ... + a_{n_a} q^{-n_a}, \quad \text{and} \quad B(q) = b_1 q^{-1} + ... + b_{n_b} q^{-n_b}$$

This makes it possible to represent G and H as rational functions:

$$G(q, \theta) = \frac{B(q)}{A(q)}, \quad H(q, \theta) = \frac{1}{A(q)} \tag{7}$$

The polynomial parameter(θ) estimation is performed by a least square method. When the ARX model is estimated, the dynamic discrete-time system model can be converted into the dynamic discrete state-space model and dynamic continuous time state-space system in sequence, which enables to calculate the gradient easily from a state space representation:

$$\dot{x} = Ax + Bu, \quad \text{and} \quad y = Cx + Du \tag{8}$$

Where $\cdot$ is a notation for differentiation with respect to time.

Thus, the estimated gradient($\hat{J}_u$) can be calculated by:

$$\frac{dy}{du} = -CA^{-1}B + D \tag{9}$$

Based on this ARX model structure, how estimated A, B, C and D matrices in Equation 8 are linked with variables in transfer functions will be explained, both for the first and second-order transfer functions.

2.3.1 First-order transfer function

First-order ARX model is used for estimating a system containing a first-order transfer function as Equation (10).

$$G(s) = \frac{k}{\tau s + 1} \tag{10}$$

And the corresponding ARX model is

$$y(t) + a_1 y(t-1) = b_1 u(t-1) + e(t) \tag{11}$$

where

$$\theta = [a_1 \quad b_1]^T$$

Parameters a_1 and b_1 are the two estimated variables by the 1st order ARX model, and it gives A, B, C, and D matrices, which are $[1 \times 1]$ in this case.

In regards to the plant model structure, the static non-linearity function $f(u)$ gives an input y_{ss} for $G(s)$. Thus, the relationship between y and y_{ss} can be stated in Fourier domain as:

$$y = G(s) \cdot y_{ss} = \frac{1}{\tau s + 1} y_{ss} \tag{12}$$

By a transformation from Fourier domain to time domain yields:

$$\tau\dot{y} + y = y_{ss} \tag{13}$$

By rearrangement,

$$\dot{y} = \frac{y_{ss} - y}{\tau} \tag{14}$$

Put $y = x$,

$$\dot{x} = \frac{y_{ss} - x}{\tau} = -\frac{1}{\tau}x + \frac{k}{\tau}y_{ss} \tag{15}$$

Equation (8) can be written as a matrix form:

$$\begin{bmatrix} \dot{x} \\ y \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x \\ y_{ss} \end{bmatrix} \tag{16}$$

and comparing Equation (16) with $y = x$ and Equation (15) yields

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau} & \frac{k}{\tau} \\ 1 & 0 \end{bmatrix} \tag{17}$$

In conclusion, the first-order ARX model estimates 2 parameters a_1 and b_1, or the gain of the system k and time constants τ in its final analysis.

2.3.2 Second-order transfer function

Second-order ARX model is used for estimating a system containing a second-order transfer function as Equation (18).

$$G(s) = \frac{\tau_a s + 1}{(\tau_1 s + 1)(\tau_2 s + 1)} = \frac{\gamma s + 1}{\alpha s^2 + \beta s + 1} \tag{18}$$

And the corresponding second-order ARX model is

$$y(t) + a_1 y(t-1) + a_2 y(t-2) = b_1 u(t-1) + b_2 u(t-2) + e(t) \tag{19}$$

where

$$\theta = [a_1 \quad a_2 \quad b_1 \quad b_2]^T$$

Parameters a_1, a_2, b_1, and b_2 are the four estimated variables by the 2nd order ARX model, and it gives A,B,C, and D matrices.

Similar with the 1st order transfer function, y_{ss} is the input for G(s). In Fourier domain,

$$y = G(s) \cdot y_{ss} = \frac{\gamma s + 1}{\alpha s^2 + \beta s + 1} y_{ss} \tag{20}$$

Equation (20) can be rearranged with respect to $\frac{y}{y_{ss}}$, and $z(s)$ is introduced for simplifying a conversion to time domain,

$$\frac{y}{y_{ss}} = \frac{(\gamma s + 1) z(s)}{(\alpha s^2 + \beta s + 1) z(s)} \tag{21}$$

Converting Equation (21) to time domain yields:

$$y_{ss} = \alpha z + \beta \dot{z} + z \tag{22}$$

and

$$y = r\dot{z} + z \tag{23}$$

Put

$$x_1 = \dot{z}, \quad \text{and} \quad x_2 = z$$

where the relationship between $\dot{x_2}$ and x_1 becomes

$$\dot{x_2} = x_1 \tag{24}$$

Then, a set of equations can be obtained by applying x_1 and x_2 in Equation (22) and (23) as

$$y_{ss} = \alpha\dot{x_1} + \beta x_1 + x_2 \tag{25}$$

$$y = \gamma x_1 + x_2 \tag{26}$$

Equation (25) can be rearranged in terms of $\dot{x_1}$ as

$$\dot{x_1} = -\frac{\beta}{\alpha}x_1 - \frac{1}{\alpha}x_2 + \frac{u}{\alpha} \tag{27}$$

Meanwhile, Equation (15) can be written in a second-order matrix form as

$$x = \begin{bmatrix}\dot{x_1}\\ \dot{x_2}\end{bmatrix} = A\begin{bmatrix}x_1\\ x_2\end{bmatrix} + By_{ss} = \begin{bmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}b_1\\ b_2\end{bmatrix}y_{ss} \tag{28}$$

$$y = C\begin{bmatrix}x_1\\ x_2\end{bmatrix} + Dy_{ss} = c_1x_1 + c_2x_2 + dy_{ss} \tag{29}$$

Thus, comparing Equation (24), (26), and (27) with Equation (28) and (29) provides A,B,C, and D matrices as

$$A = \begin{bmatrix}-\frac{\beta}{\alpha} & -\frac{1}{\alpha}\\ 1 & 0\end{bmatrix} = \begin{bmatrix}-\frac{\tau_1+\tau_2}{\tau_1\cdot\tau_2} & -\frac{1}{\tau_1\cdot\tau_2}\\ 1 & 0\end{bmatrix} \tag{30}$$

$$B = \begin{bmatrix}\frac{1}{\alpha}\\ 0\end{bmatrix} = \begin{bmatrix}\frac{1}{\tau_1\cdot\tau_2}\\ 0\end{bmatrix} \tag{31}$$

$$C = \begin{bmatrix}\gamma & 1\end{bmatrix} = \begin{bmatrix}\tau_a & 1\end{bmatrix} \tag{32}$$

$$D = 0 \tag{33}$$

In conclusion, the second-order ARX model estimates 4 parameters a_1, a_2, b_1, and b_2. At the same time, it can also be explained to estimate the system gain k and time constants τ_a, τ_1, and τ_2 in its last analysis.

Therefore, from the analysis on the linkage between estimated variables and parameters in transfer functions, the second advantage of the dynamic ESC, "it is possible to estimate a controller gain and time constants in transfer functions", is well explained.

3 Simulation results of the dynamic ESC

As shown in Figure 2, the relationship between the oil production rate and the gas injection rate can be simplified as a quadratic equation. This relationship represents a static non-linear part of the plant in Figure 5, $y_{ss} = f(u)$. Specifically, Equation (34) can be used as a simple static non-linear quadratic system with unknown parameters a, b, and c, which has an extremum at $(u, y_{ss}) = (a, b)$ when the c is negative.

$$f(u) = c(u - a)^2 + b = y_{ss} \tag{34}$$

The constants in the Equation (34) are dependent not only on different wells but also on operation situations and disturbances. In this project, the $f(u)$ is not describing a real system, and the constants are manually selected to have a clear maximum and small values for the input and output. This is because when the values for u or y is large, it requires too much computational time to converge. Using a simple model will provide a clear and fast result and it will also help to develop the dynamic ESC model faster. Thus, the a, b, and c are set to be 20, 40, and -0.1 respectively as Equation (35).

$$f(u) = -0.1(u - 20)^2 + 45 = y_{ss} \tag{35}$$

Matlab is used as a simulation tool, and CasADI is used for differential equation solver. The initial value for the input and output is set as $(u, y_{ss}) = (15, 42.5)$ to obtain results fast enough, and a sampling time of $T_s = 1sec$ is used for all simulation cases.

Figure 6 shows an example of a simulation result. In y-axis, the output, input, and gradient are plotted, and x-axis means number of iterations. The green solid lines of u and y mean the input and output value containing noise. As explained in the chapter

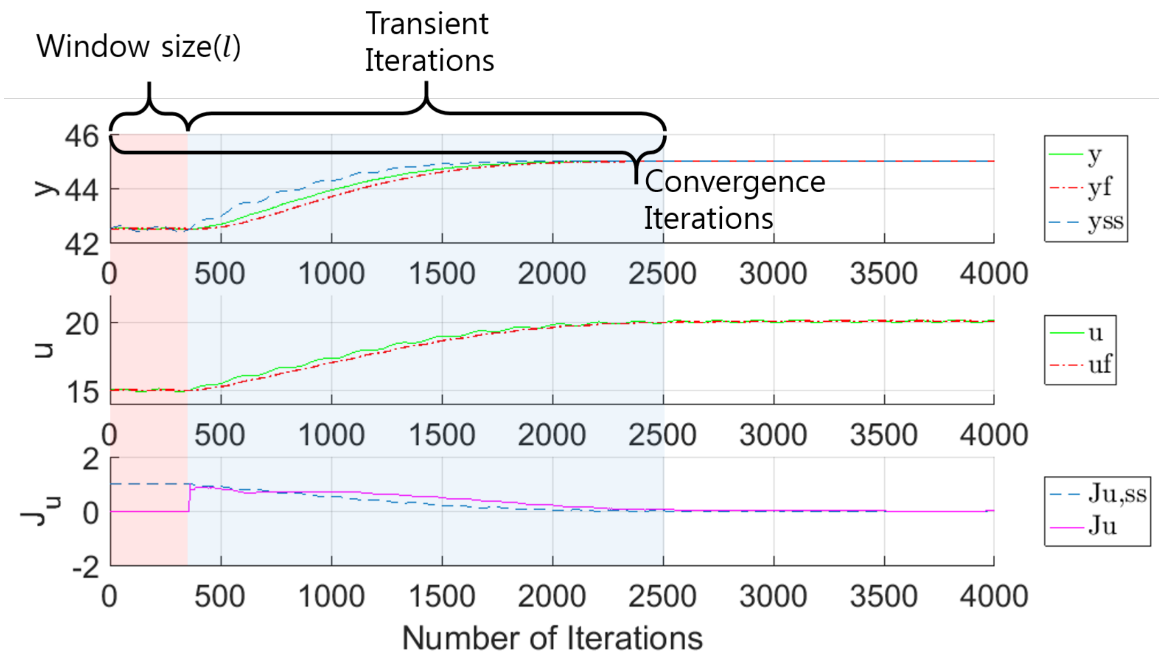

Figure 6: Schematic explanation for convergence iterations and transient iterations

2.2, the noise is filtered out by the low pass filter, providing u_f and y_f illustrated with red dash-dot lines. The u_f and y_f data are moved to have a zero mean, and the steady-state gradient is estimated which is plotted by a purple line. The blue dash lines y_{ss} and $J_{u,ss}$ means a steady state value or a true value of y and J_u calculated from the Equation (35). The true values make it possible to check if the dynamic ESC algorithm is working properly as intended or not.

Additionally, the window size or the data vector size for the gradient estimation is indicated in the Figure 6 with red box meaning that the gradient estimation starts after building up the data vectors with a chosen size. Since the data will be built up continuously in real systems, one does not need to wait to build up data like this situation. Thus, the pure iteration number needed to converge into a new optimum should subtract the window size from the overall convergence iterations as the blue box in Figure 6.

When judging the point where the system converged into its extremum, the noise filtered output value(y_f) is used. It is performed manually by zooming up the y figure.

In this chapter, monitoring if the dynamic ESC algorithm is working well is the first focus. For this, plant processes with different dynamics are tested, especially a first-order transfer function, a second-order transfer function without zero, and a second-order transfer function with zero are applied adding more complexity.

After making the dynamic ESC algorithm in different plant models, controller tuning is performed by a Trial and Error method. Detailed tuning procedures are provided in Appendix A, and controller tuning for the classic ESC has also performed in Appendix B to compare them in the chapter 4. In the main report part, only some important simulation results are explained to prevent confusion. The tuning parameters in this dynamic ESC are listed in Table 1 for different transfer function cases which are tested one by one.

Table 1: Tuning parameters in the dynamic ESC

	Tuning parameters
Case 1 $G(s) = \frac{1}{174s+1}$	· Integral controller gain (K_i) · Window size (l) · Sine wave time constant (T_D)
Case 2 $G(s) = \frac{1}{(174s+1)(60s+1)}$	· Integral controller gain (K_i) · Window size (l) · PRBS calculation step*
Case 3 $G(s) = \frac{40s+1}{(174s+1)(60s+1)}$	· Integral controller gain (K_i) · Window size (l) · PRBS calaculation step*

*it indicates a length of steps where the next binary sequence is generated.

The integral controller gain has the largest effect on the number of iterations because larger controller gain gives faster control initially. The window size or data vector length has the second largest influence especially on the robustness, and the sine wave frequency is adjusted lastly. The PRBS wave is applied later instead of the sine wave in the second-order transfer function system for solving an issue, which will be explained in the chapter 3.2.

3.1 Plant process with first-order transfer function

In this chapter, a simple first-order transfer function is used to describe the dynamic of a plant. Equation (36) implies a typical first-order transfer function where the gain is 1 because the system gain is already provided in the static non-linear part $f(u)$. For the time constant, it is set to be 174. As already mentioned in section 2.3, the first-order ARX model is used to estimate parameters. The detailed tuning process is provided in the Appendix A.1. The amplitude of the sinusoidal wave perturbation(α) is fixed as 0.1 because it did not have a significant effect on the system.

$$G(s) = \frac{1}{\tau s + 1} = \frac{1}{174s + 1} \tag{36}$$

Table 2 shows the simulation result with different tuning parameters for the first-order system. One thing to consider in parameter tuning is that the window size is set to be multiples of the dither time constant.

Table 2: Simulation result of a dynamic ESC with a plant model containing first-order transfer function.

No.	Controller gain(K_i)	Dither time constant(T_D)	Window size (l)	Convergence iterations	Transient iterations	True extremum
1	0.001	180	$2*180$	7200 $\uparrow$	7200 $\uparrow$	O
2	0.002	180	$2*180$	7200 $\uparrow$	7200 $\uparrow$	O
3	0.003	180	$2*180$	4200	3840	O
4	0.004	180	$2*180$	3000	2640	O
5	0.005	180	$2*180$	2500	2140	O
6	0.006	180	$2*180$	2000	1640	offset*
7	0.007	180	$2*180$	1800	1440	offset*
8	0.008	180	$2*180$	1700	1340	offset*
9	0.009	180	$2*180$	1600	1240	offset*
10	0.005	30	$1*30$	2400	2370	O
11	0.005	30	$6*30$	2500	2380	O
12	0.005	30	$10*30$	2800	2500	O
13	0.005	180	$1*180$	2600	2420	O
14	0.005	180	$2*180$	2500	2140	O
15	0.005	180	$3*180$	2600	2060	O
16	0.005	180	$4*180$	3100	2380	O
17	0.005	180	$8*180$	–	–	X
18	0.005	$4*180$	$4*180$	3000	2280	O
29	0.005	$4*180$	$8*180$	–	–	X

* It converged to the optimum, but in the control input, there was a small offset

In specific, simulations number $1-9$ are performed to check the effect of the integral controller gain. A small controller gain requires a large number of iterations while a large controller gain needs a small number of iterations. This tendency is illustrated in Figure 7.

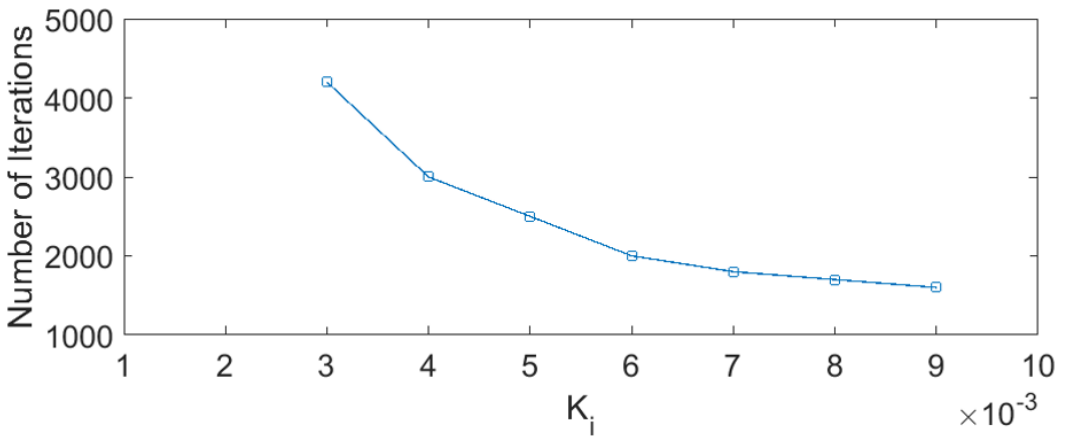

Figure 7: Simulation result for first-order system using a dynamic ESC: K_i varies

However, simulations number $6-9$ with controller gains of $0.006-0.009$ have a small offset in the control input with a small magnitude. In detail, Figure 8 describes a simulation number 5 when $K_i = 0.005$. In this case, it takes around 2500 iterations overall to converge into the true extremum including the window size. Although numerical spikes round 3000 iterations make the result deviate from the optimum, this problem will be solved in chapter 4 later, and it is enough to focus on if the system primarily reaches the extremum or not in this stage.

In Figure 9 with $K_i = 0.006$, the control input slightly passes the extremum and converges to $(20.5, 45)$. This does not make a serious problem in the estimation of the output, however, this tendency becomes larger when a larger controller gain is applied. Based on these observations, the controller gain is chosen to be 0.005 for the next simulations.

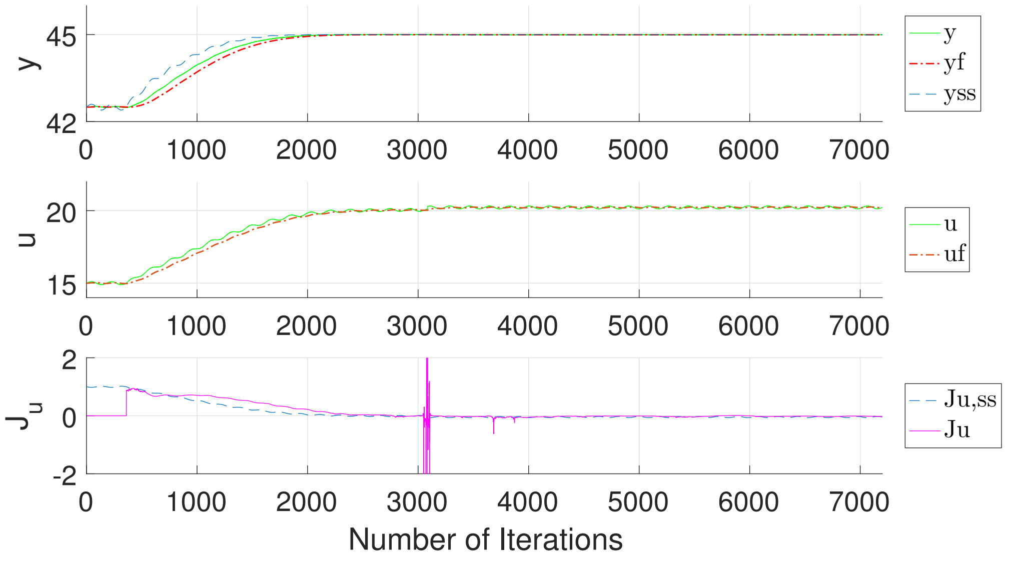

Figure 8: Simulation result for first-order system using a dynamic ESC: $K_i = 0.005$, $T_D = 180$, and $l = 2 * 180$.

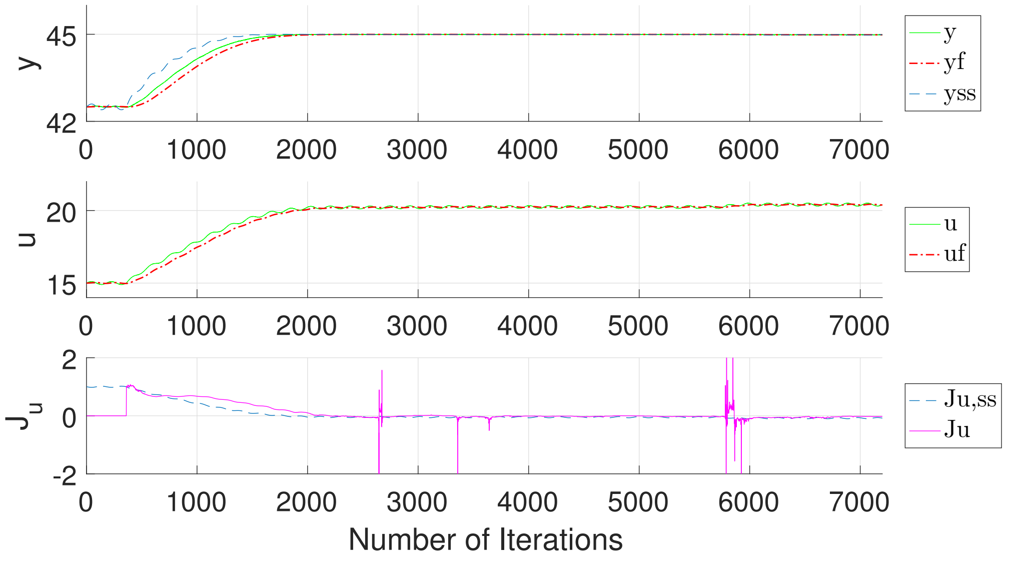

Figure 9: Simulation result for first-order system using a dynamic ESC: $K_i = 0.006$, $T_D = 180$, and $l = 2 * 180$.

The window size can be figured out from simulations number $10 - 20$, by comparing simulations where all other tuning parameters but the dither time constant are same. In detail, simulation sets of (10,11,12), (13,14,15,16,17), and (17,18,19) in Table 2 can provide the effect of the window size on the result. From the iteration numbers in Table 2, it can be concluded that there is no clear tendency depending on the window size. However, in terms of robustness, small window sizes make noisy behavior triggering frequent numerical spikes while excessively large window sizes always make serious deviations. This is well illustrated in Figure 10 and 11 which are the two extreme cases, a simulation number 13 and 17 respectively.

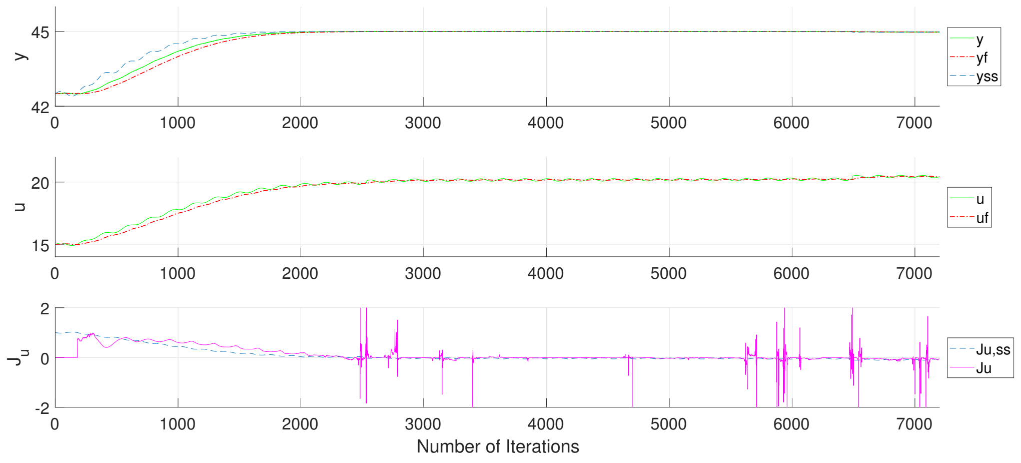

Figure 10: Simulation result for first-order system using a dynamic ESC: $K_i = 0.005$, $T_D = 180$, and $l = 180$.

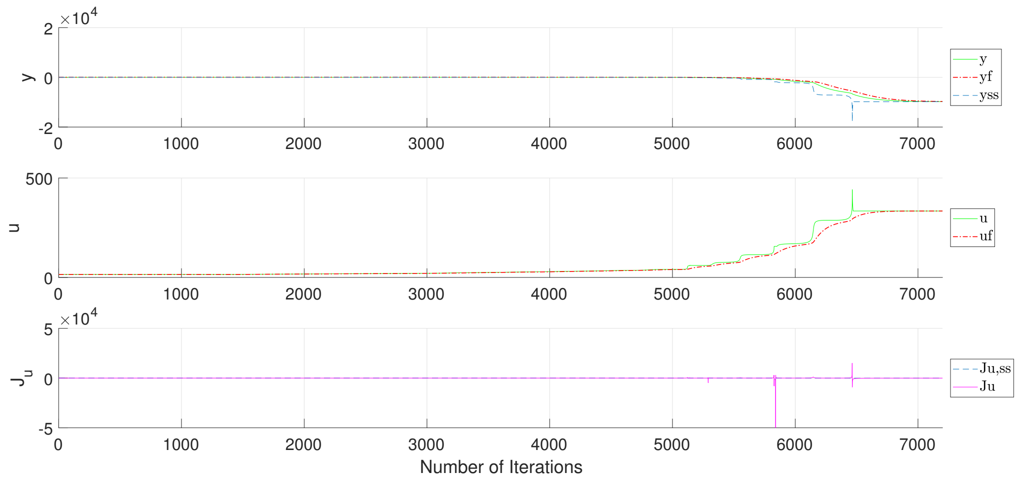

Figure 11: Simulation result for first-order system using a dynamic ESC: $K_i = 0.005$, $T_D = 180$, and $l = 8 * 180$.

The influence of the dither time constant can be obtained by comparing simulations number 11 with 13, or 16 with 18 where all other tuning parameters but the dither time constant are same. From the transient iteration numbers in Table 2, it is concluded that the dither time constant does not have a recognizable effect on the system since they yield a more or less same number of transient iterations. Additionally, it does not influence the robustness as well.

Based on the dynamic ESC tuning procedure, the simulation number 16 in Table 2 is concluded to be the best-optimized result, which is also shown in Figure 12.

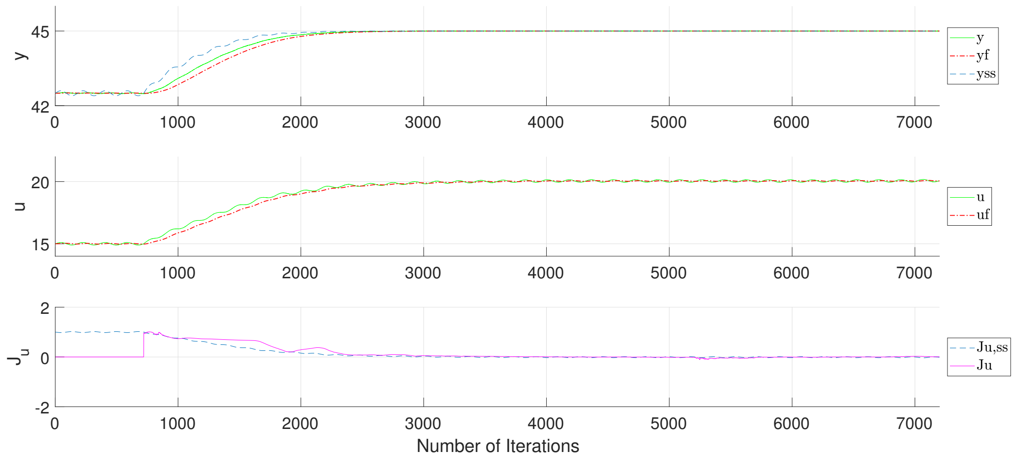

Figure 12: Simulation result for first-order system using a dynamic ESC: $K_i = 0.005$, $T_D = 180$, and $l = 4 * 180$.

In Figure 12, it is clearly indicated that the estimated gradient J_u follows the true gradient, verifying the fact that the dynamic ESC using ARX model is successfully implemented as intended in the first-order transfer function system. It converges to the true optimum around 3100 iterations overall, which means it takes 2380 iterations for a transition.

However, numerical spikes problem happens occasionally as illustrated in Figure 8, 9, 10 and 11. Since it sometimes makes serious deviations especially when the controller gain is too large, or when the window size is too small or large. These are studied more in the Appendix A.1. In conclusion, this problem has to be solved for providing a robustness on the dynamic ESC process which will be discussed in 4.

3.2 Plant process with second-order transfer function

In this chapter, plant dynamic models with second-order transfer functions are applied.

$$G(s) = \frac{\tau_a s + 1}{(\tau_1 s + 1)(\tau_2 s + 1)} = \frac{1}{(174s + 1)(60s + 1)} \tag{37}$$

$$G(s) = \frac{\tau_a s + 1}{(\tau_1 s + 1)(\tau_2 s + 1)} = \frac{40s + 1}{(174s + 1)(60s + 1)} \tag{38}$$

Equation (37) and (38) are the second-order transfer functions used in this project, which are second-order transfer function without zero($\tau_a = 0$) and with zero($\tau_a = 40$) respectively. To give a different dynamic from the first time constant of 174, the second time constant is set to be 60. The two transfer functions be separately explained in subsection 3.2.1 and 3.2.2. The second-order ARX model is used to estimate parameters.

In the second-order system, transfer function without zero is simulated first. However, a problem arouses from the starting point that J_u does not follow the true gradient. Figure 13 shows a simulation result where the integral controller is not active, which is performed to check if the gradient estimation part works properly, and it indicates that J_u is estimated as 0 while the true gradient is 1.

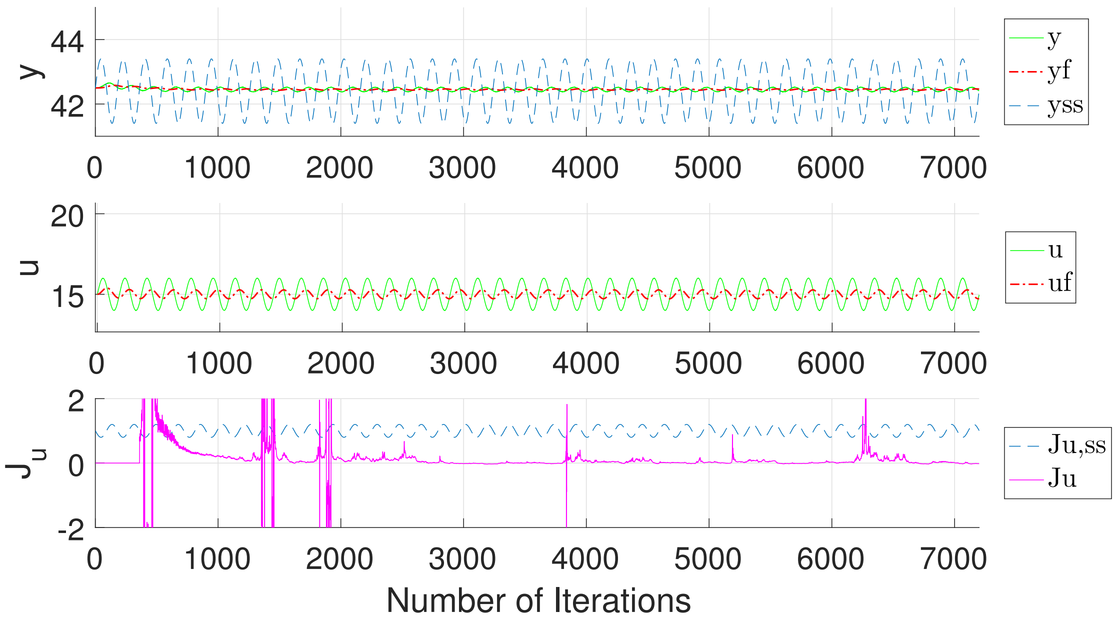

Figure 13: Simulation result for second-order without zero system($\tau_a = 0$) using a dynamic ESC: $K_i = 0$.

Based on the fact that there was no such a problem in the first-order system, an approach for figuring out the cause of this matter comes from the difference between the first and second-order system. Specifically, when we think of step responses for different transfer functions, the dynamic in the second-order system has a secondary time lag response as shown in Figure 14.

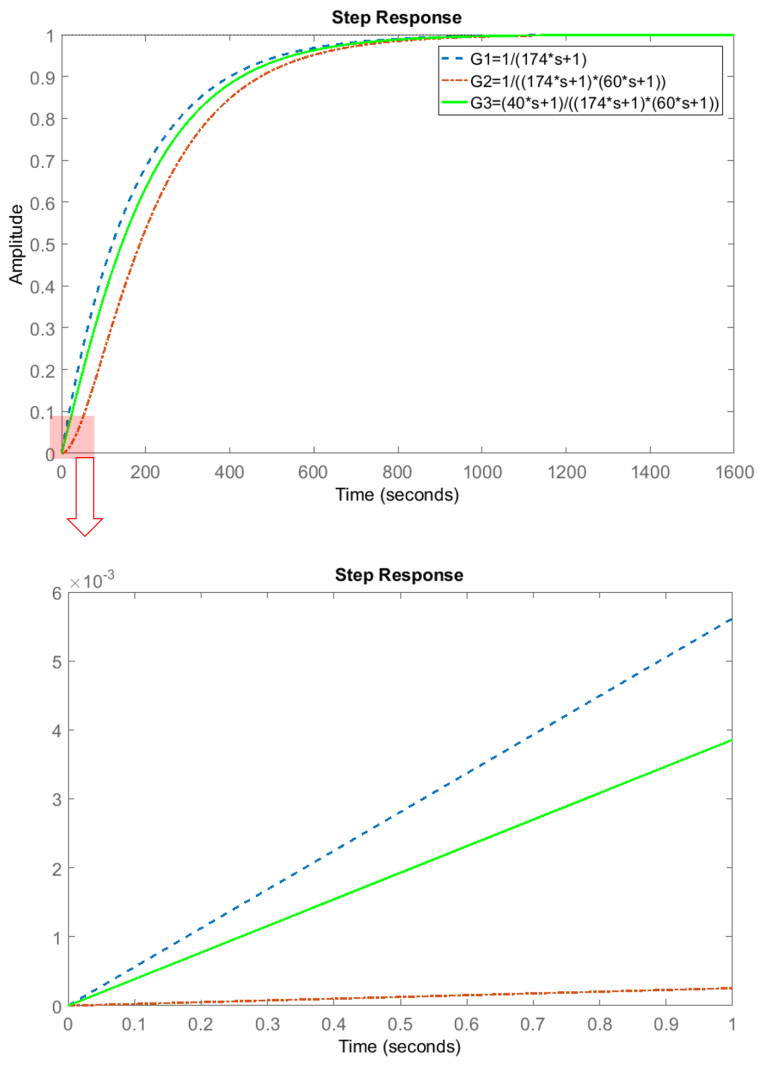

Figure 14: Step responses of first and second-order transfer functions

The red dash-dot line is a step response to the second-order transfer function without zero($\tau_a = 0$). With the figure below with zoomed-in in axis, it is shown that there is almost no change in the response at the starting point due to the secondary time lag response. Thus, sine waves with one frequency are discussed not to be sufficient to obtain full information about y, so that the ARX model could not estimate the gradient.

To solve this problem, a pseudo-random binary sequence(PRBS) is introduced instead of a simple sine wave and it becomes possible to make the estimated gradient follow the true gradient. The PRBS generates a binary sequence with a length of N, which is set to be 1 in this project. Because the PRBS wave has a diverse range of frequencies, it is more efficient to make the ARX model estimate four parameters in the second-order system as well. Specifically, lines $17 - 19$ in Algorithm 1 can be replaced with Algorithm 2 as stated below.

Algorithm 2 The dynamic ESC using ARX model in second-order system

```
···
1: Perturbation
2:         $prbs \leftarrow idinput(1)$
3:         $u_{k+1} \leftarrow \hat{u_k} + prbs$
```

Tuning parameters are similar to the first-order transfer function system, however, the only difference is determining how often to calculate the new PRBS instead of determining the dither time constant.

3.2.1 The second-order transfer function without zero ($\tau_a = 0$)

Table 3 shows the simulation results when tuning parameters vary. As almost of all tendency depending on the tuning parameters are explained with simulation plots in 3.1, only the best-tuned result will be illustrated. Detailed procedures for determining the tuning parameters for this second-order transfer function without zero can be obtained in Appendix A.2

Table 3: Simulation result of a dynamic ESC with a plant model containing second-order transfer function without zero($\tau_a = 0$)

No.	Controller gain(K_i)	PRBS calculation steps	window size(l)	Convergence iterations	Transient iterations	True extremum
1	0.005	30	$4 * 180$	2500	1780	O
2	0.005	60	$4 * 180$	3000	2280	O
3	0.005	90	$4 * 180$	3000	2280	O
4	0.005	180	$4 * 180$	3000	2280	O
5	0.001	30	$4 * 180$	10000	9280	O
6	0.002	30	$4 * 180$	6000	5280	O
7	0.003	30	$4 * 180$	4000	3280	O
8	0.004	30	$4 * 180$	3000	2280	O
9	0.005	30	$4 * 180$	2500	1780	O
10	0.006	30	$4 * 180$	2400	1680	offset*
11	0.007	30	$4 * 180$	2300	1580	offset*
12	0.008	30	$4 * 180$	2200	1480	offset*
13	0.005	30	$1 * 180$	1100	920	offset*
14	0.005	30	$2 * 180$	1400	1040	O
15	0.005	30	$3 * 180$	1600	1060	O
16	0.005	30	$4 * 180$	2500	1780	O
17	0.005	30	$5 * 180$	3000	2100	O

* It converged to the optimum, but in the control input, there was a small offset

In detail, simulation $1 - 4$ are obtained with different PRBS calculation steps. Overall, the PRBS tuning parameter does not have a huge influence on the iterations needed to reach a new optimum. All the results converge to a right extremum at the first time around $2500 - 3000$ iterations, but numerical spikes always happen in this second-order model making recognizable deviations. Recalculating the PRBS wave every 30 step is chosen for next simulations.

Simulations number $5 - 12$ are performed to figure out the controller gain which gives fast and smooth convergence. Because the results converge to a point passing the maximum with larger K_i value than 0.005, it is chosen for the best controller gain.

In terms of the window size, a small window size such as simulations 13,14 could not properly estimate the gradient making noisy ARX model fitting. The increased window size provides much better results while some numerical spikes lead them to deviate later as well.

Based on the dynamic ESC tuning, the best-tuned result is obtained in a simulation number 9(or 16) when $K_i = 0.005$, PRBS=30, and $l = 4 * 180$ which is shown in Figure 15. Even though numerical spikes deviates the result from the maximum point, J_u well follows the true gradient, so that fixing the numerical peaks will achieve well-converging simulation result later in section 4.

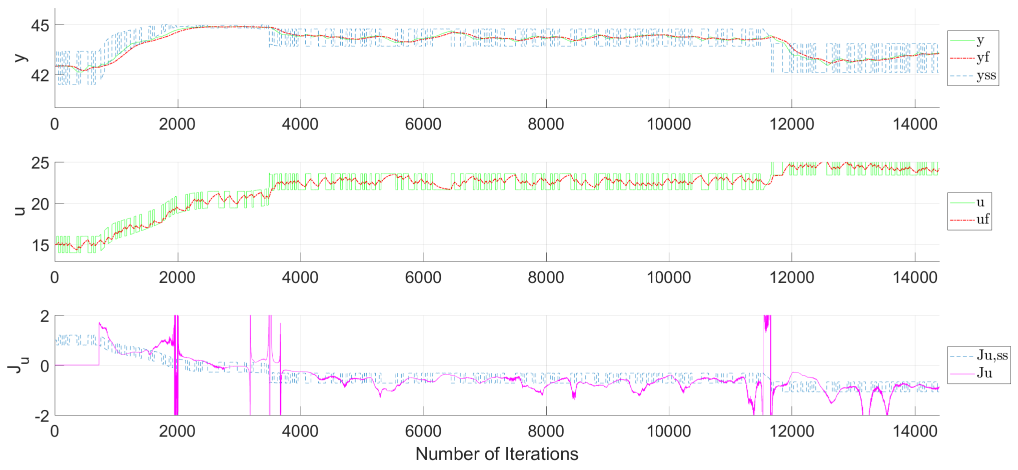

Figure 15: Simulation result for second-order without zero($\tau_a = 0$) system using a dynamic ESC: $K_i = 0.005$, PRBS= 30, and $l = 720$

3.2.2 Second-order transfer function with zero ($\tau_a = 40$)

Similarly with the previous result, the simulation results in the second-order transfer model with zero are suggested in Table 4. Detailed simulation plots and tuning procedures are suggested in Appendix A.3

Table 4: Simulation result of a dynamic ESC with a plant model containing second-order transfer function with zero($\tau_a = 40$)

No.	Controller gain(K_i)	PRBS calculation steps	window size(l)	Convergence iterations	Transient iterations	True extremum
1	0.005	30	$4 * 180$	3000	2280	O
2	0.005	60	$4 * 180$	2500	1780	O
3	0.005	90	$4 * 180$	3000	2280	O
4	0.005	180	$4 * 180$	2500	1780	X
5	0.001	30	$4 * 180$	13000	12280	O
6	0.002	30	$4 * 180$	7500	6780	O
7	0.003	30	$4 * 180$	6000	5280	O
8	0.004	30	$4 * 180$	5000	4280	O
9	0.005	30	$4 * 180$	3500	2780	O
10	0.006	30	$4 * 180$	2500	1780	offset*
11	0.007	30	$4 * 180$	2000	1480	offset*
12	0.008	30	$4 * 180$	1500	780	offset*
13	0.005	30	$1 * 180$	3000	2820	O
14	0.005	30	$2 * 180$	3000	2640	O
15	0.005	30	$3 * 180$	3500	2960	O
16	0.005	30	$4 * 180$	3500	2780	O
17	0.005	30	$5 * 180$	3000	2100	O

* It converged to the optimum, but in the control input, there was a small offset

In Table 4, simulations number 1−4 are obtained with different PRBS calculation steps. There is no increasing or decreasing tendency of the iteration numbers depending on the PRBS calculation steps. However, with the PRBS tuning parameters of 90 and 180 have relatively unstable behavior. Thus, the PRBS is set to be calculated every 30 iterations.

Simulations number $5 - 12$ show results with different K_i values. When the controller gain is increased to 0.006, relatively large numerical peaks happen and u and y pass the extremum having small offsets. This tendency becomes larger in simulations with larger controller gains. Based on the simulation result, $K_i = 0.005$ is chosen for next simulations in this second-order with zero model as well.

Next simulations $13 - 17$ are performed to see the effect of the window size of data sets. It is concluded that moderately large values of the gradient estimation window sizes give more robust results while too small window sizes make a lot of noise in the ARX estimation part and too large window size has a disadvantage since it may have problems on detecting the influence of a disturbance when a disturbance occurs.

Figure 16 shows the best-tuned simulation result, which is the simulation number 9(or 16). In detail, it smoothly converges to its maximum value with 2780 number of transition iterations. Although there are small numerical peaks around 1800 iterations, it is not problematic because it does not make a deviation.

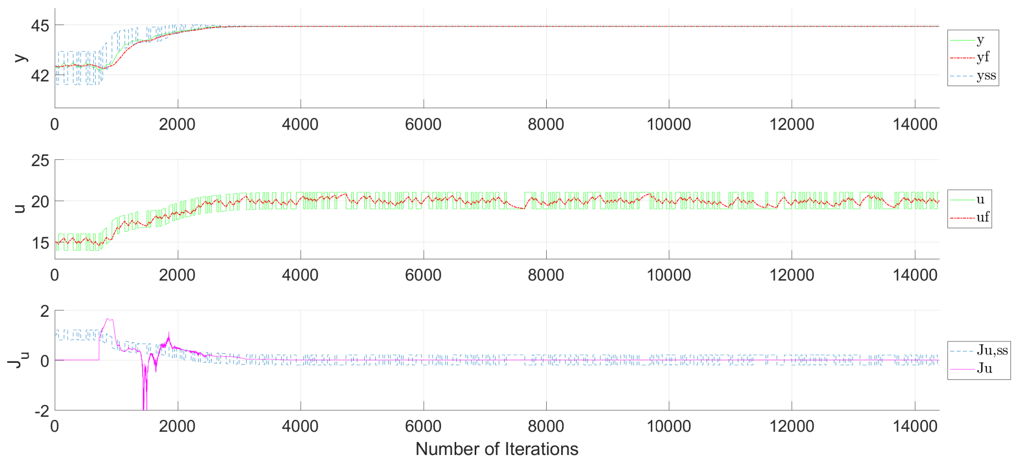

Figure 16: Simulation result for second-order with zero system($\tau_a = 40$) using a dynamic ESC: $K_i = 0.005$, PRBS$= 30$, and $l = 720$

However, in some of the other results, numerical spikes occasionally happen after convergence as the plant models with other transfer functions.

3.3 Application of optimal profit

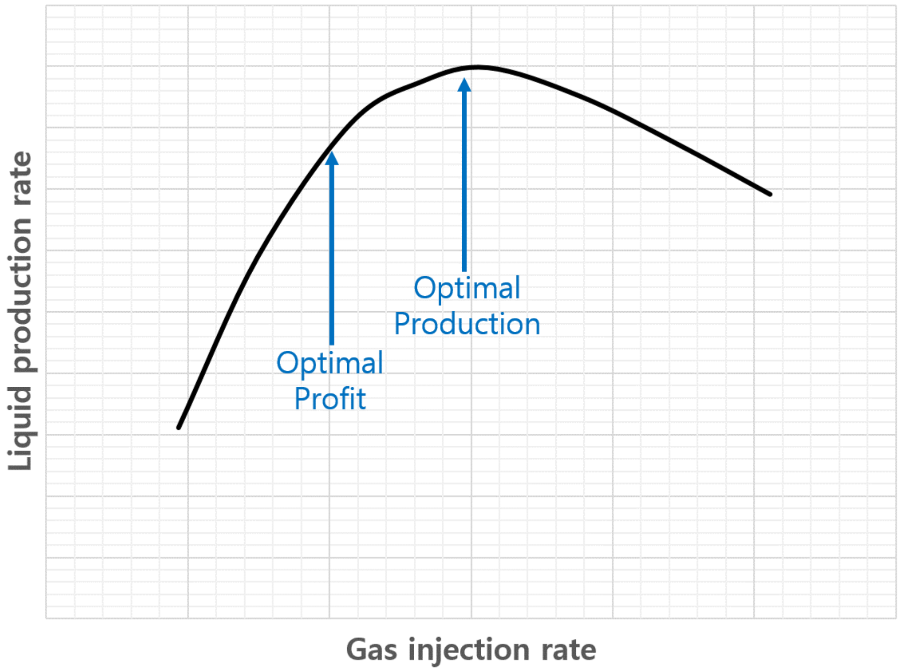

Figure 17: Optimal profit and production rate

There is one important thing which should be considered in the oil production-gas injection rate dependency as shown in Figure 17[10]. When one considers costs, the gas compression process requires a lot of energy which relatively costs a lot, however, there is no huge improvement in the oil production rate near the extremum. This means that the optimal profit will be obtained at a point somewhere before it reaches its maximum.

Thus, the dynamic ESC is performed with an assumption that the optimal profit is earned when the gradient is 0.2. Figure 18 illustrates a result when second-order transfer function with zero model is applied, and other tuning parameters are same as the best result previously obtained. It states that this dynamic ESC can also be applied to find an optimal profit point where the gradient is not zero.

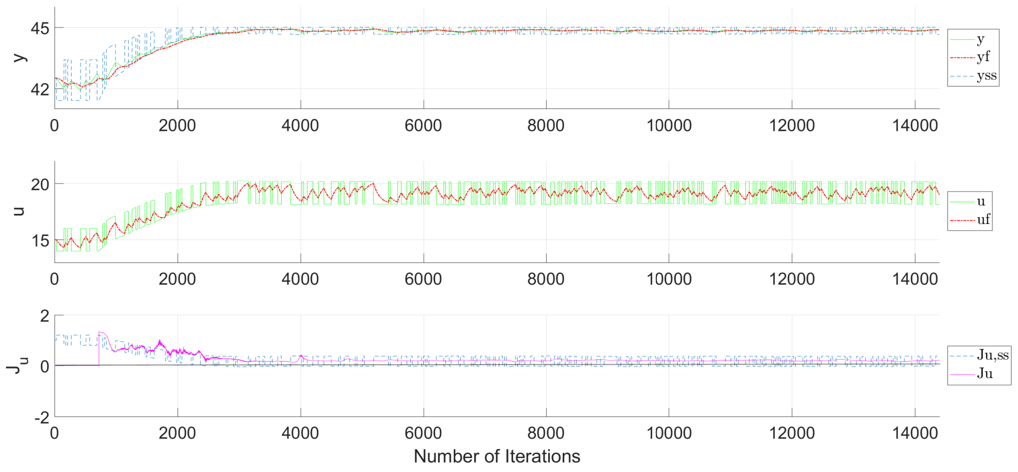

Figure 18: Simulation result applying an optimal profit point for second-order with zero($\tau_a = 40$) system using a dynamic ESC: $K_i = 0.005$, PRBS$= 30$, and $l = 720$

3.4 Discussion

The simulation results for plants with different transfer functions show that the dynamic ESC algorithm is successfully implemented, not only in the first-order system but also in the second-order system. In detail, the estimated gradient J_u follows the true gradient $J_{u,ss}$ and correspondingly u and y converges to the true extremum. Different tuning parameters are applied and the best-tuned results are provided by the Trial and Error method. The integral controller gain has a large effect on the iteration number to converge, while the iteration number is almost independent of the dither wave time constant or the PRBS recalculation steps. The window size has an influence on the robustness of the system, in specific, a small window size makes noisy ARX estimation or unstable behavior and a large window size makes huge deviation by numerical spikes.

Both in the first and second-order models, numerical spikes happen and sometimes they make a deviation from the true extremum. After some observations, it became clear that the numerical spikes are prone to firstly happen right after the estimated steady-state reaches zero, and occasionally even after convergence. Therefore, the ARX model is discussed as a possible reason to explain such phenomena.

As already mentioned in section 2.3, the first-order ARX model is

$$y(t) + a_1 y(t-1) = b_1 u(t-1) + e(t) \tag{39}$$

where

$$\theta = [a_1 \quad b_1]^T$$

As Equation (39), the first-order ARX model estimates two parameters a_1 and b_1 using least square method for each of them. When we consider a point near the extremum where J_u is close to zero, there is a sinusoidal change in the input u, but almost no change in the output y which makes only one parameter can be estimated as depicted in Figure 19. However, the ARX model still tries to estimate two parameters, and this is the cause of numerical peaks.

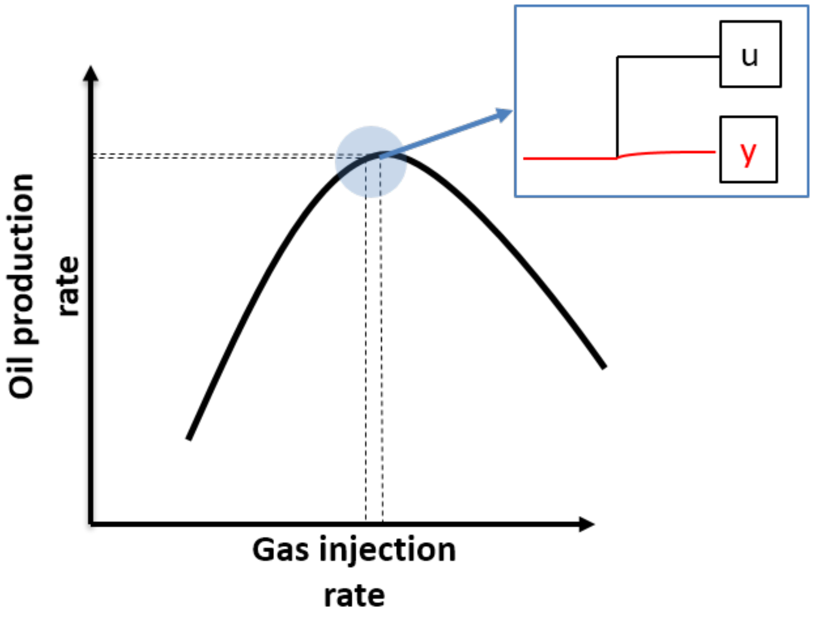

Figure 19: The input and output relationship when the gradient is close to zero

In second-order system, the numerical peaks becomes more problematic than the first-order system. In specific, the second-order ARX model is:

$$y(t) + a_1 y(t-1) + a_2 y(t-2) = b_1 u(t-1) + b_2 u(t-2) + e(t) \tag{40}$$

where

$$\theta = [a_1 \quad a_2 \quad b_1 \quad b_2]^T$$

Now the second-order ARX model estimates four parameters stated in θ vector. However, as already issued in the first-order model, change in the input but no change in y make only one parameter can be estimated as shown in Figure 19 while the second-order ARX model tries to adjust four parameters causing numerical peaks problem.

Since the reason of the numerical spikes is expected to be the ARX model, a method suggested solving this issue is combining a simple first-order least square(LS) method with the ARX model in gradient estimation. In specific, the ARX model can be switched with the LS method when J_u is close to zero in the region where numerical peaks may occur. This enables J_u to converge to zero fast enough with the local linear dynamic model, and when it is sufficiently close to zero, the switched LS method gives stability on the gradient estimation avoiding numerical peaks. Based on this idea, simulations with different thresholds of the switching point are performed in section 4.

4 Numerical spikes modification and validation

A new algorithm to solve the numerical spikes problem in the dynamic ESC is suggested in Algorithm 3. The ARX model is switched to a first-order Lease Square method when the estimated steady-state gradient is sufficiently close to zero.

4.1 Plant process with first-order transfer function

The strategy to combining the ARX model and the LS method is first tried in the first-order system. In detail, the result is illustrated in Figure 20, which is simulated with the same tuning parameters with Figure 12.

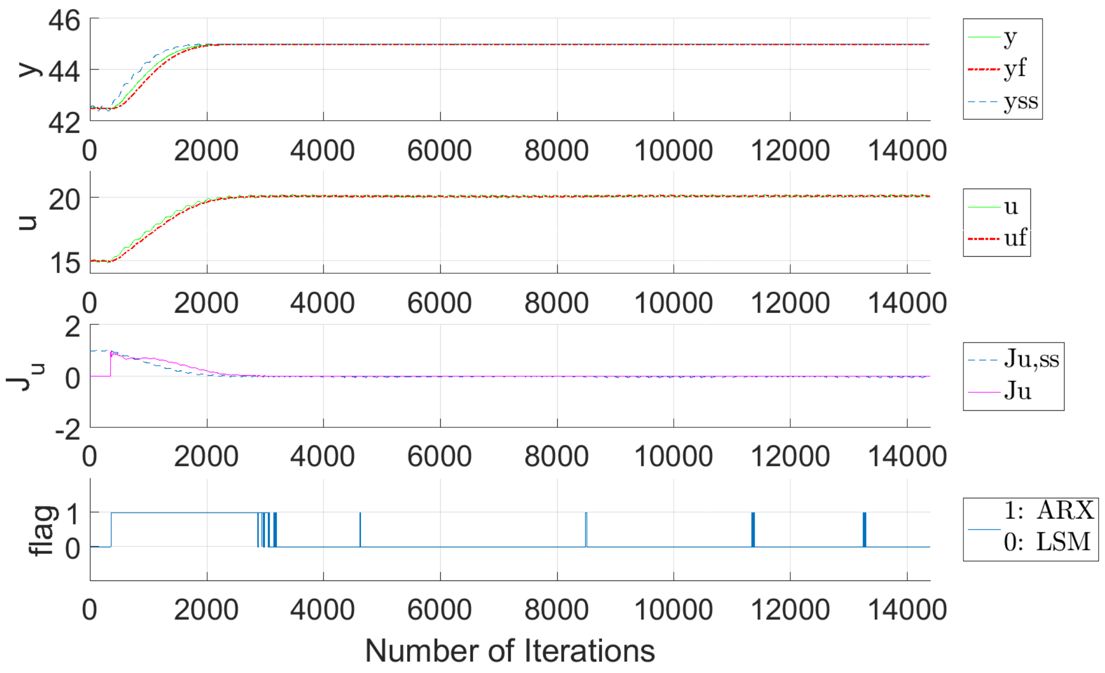

Figure 20: Modified dynamic ESC simulation result for first-order system: $K_i = 0.005$, $T_D = 180$, and $l = 360$ with a tolerance of 0.010

Figure 20 shows that the strategy of switching the ARX model to the LS method is well implemented and the numerical spikes are removed. The last subplot is providing

Algorithm 3 The dynamic ESC combining ARX model and LS method

```
input: $y_k$,$u_k$
1: for $k = 1 \to n$ do
2:     Low pass filter to remove noise
3:         $y_{fk} \leftarrow G_{LPF}(s)y_k$
4:         $u_{fk} \leftarrow G_{LPF}(s)u_k$
5:     Moved to have a zero mean
6:         $y_0 \leftarrow y_{fk} - avg(y_{fk})$
7:         $u_0 \leftarrow u_{fk} - avg(u_{fk})$
8:     if $J_{u,k-1} >$ Threshold then
9:         Gradient estimation with the ARX model using data sets of $y_0$ and $u_0$
10:            $data \leftarrow iddata(y_0, u_0, T_s)$
11:            $[A(q) \quad B(q)] \leftarrow arx[data, [1, 1, 0]]$   for 1st-order systems
12:            $[A(q) \quad B(q)] \leftarrow arx[data, [2, 2, 0]]$   for 2nd-order systems
13:            $[A_d \quad B_d \quad C_d \quad D_d] \leftarrow idss[A(q) \quad B(q)]$
14:            $[A \quad B \quad C \quad D] \leftarrow d2c[A_d \quad B_d \quad C_d \quad D_d]$
15:            $J_u \leftarrow -CA^{-1}B + D$
16:     else
17:         Gradient estimation with the LS method using data sets of $y_{fk}$ and $u_{fk}$
18:            $X \leftarrow [u_{fk} \quad ones(size(u_{fk}))]$
19:            $b \leftarrow (X'X)^{-1}X'y_{fk}$
20:            $J_u \leftarrow b(1, 1)$
21:     end if
22:     Integral Controller
23:         $\hat{u_k} \leftarrow u_k + K_iJ_u$
24:     Perturbation
25:         $prbs \leftarrow idinput(1)$
26:         $u_{k+1} \leftarrow \hat{u_k} + prbs$
27: end for
```

information of which of the two methods is used for gradient estimation in each step, specifically, the flag equals to 1 when the ARX model is used and 0 when the LS method is used. The gradient is estimated by the ARX model at the first time, and when J_u becomes close to the threshold of 0.010, the ARX model and LS method conflict and then the gradient is mostly estimated by the LS method later. The procedure to find a proper threshold is provided in Appendix A.1.3.

Figure 21 is comparing simulation results for the first-order system obtained by the dynamic and classic ESC. The blue solid lines illustrate the simulation with classic ESC model, and the tuning process is provided in Appendix B.1. For the dynamic ESC, y_f and u_f are stated with red dash-dot lines and J_u with a purple line. While the classic ESC converges to the extremum around $5 \cdot 10^4$ iterations, the dynamic ESC only requires 3100 iterations, and when the window size is subtracted, it only takes 2380 iterations to the new extremum. This is meaningful because only around 4% of the transient iteration steps are needed for convergence with the application of the dynamic ESC compared to the classic ESC.

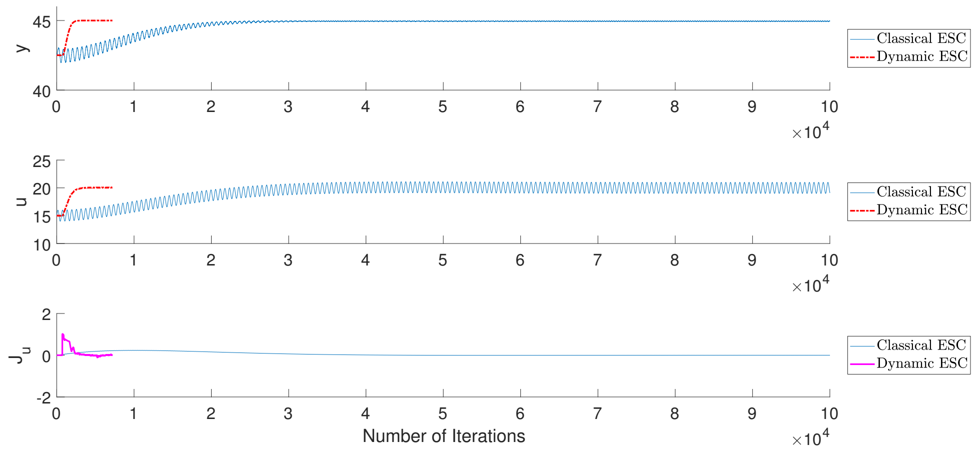

Figure 21: Simulation result comparison of a dynamic and classic ESC for first-order system

4.2 Plant process with second-order transfer function

4.2.1 Second-order transfer function without zero ($\tau_a = 0$)

In the plant model with second-order transfer function without zero system, the numerical spikes problem was the most problematic making all simulation results deviate from its extremum after convergence. However, as shown in Figure 22, combining the ARX model with LS method can solve the problem in this second-order system as well with the threshold of 0.03. The tuning parameters are the same as the simulation case in Figure 15, and detailed modifying steps are suggested in Appendix A.2.4

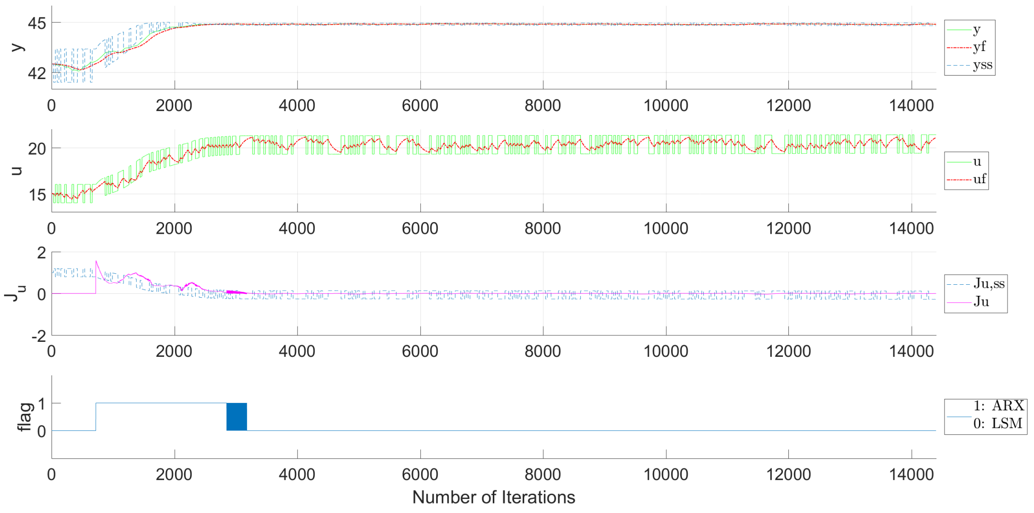

Figure 22: Modified dynamic ESC simulation result for second-order without zero($\tau_a = 0$) system: $K_i = 0.005$, PRBS= 30, and $l = 720$ with a tolerance of 0.03

However, it sometimes gives worse results then Figure 22 when simulating the same conditions multiple times. It may be because the PRBS is different in different simulations since it is random, and sometimes the calculation step is too small and gives some noise like behavior. Thus, increasing the PRBS recalculation step can be considered in a future work.

It is possible to compare the dynamic ESC with the classic ESC in Figure 23, and the tuning for the classic ESC is provided in Appendix B.1. In short, the classic ESC requires $5 \cdot 10^4$ iterations while the dynamic ESC needs only 2500 iterations or 1780 transient iterations which are 3.6% of the classic ESC for a transition to a new extremum.

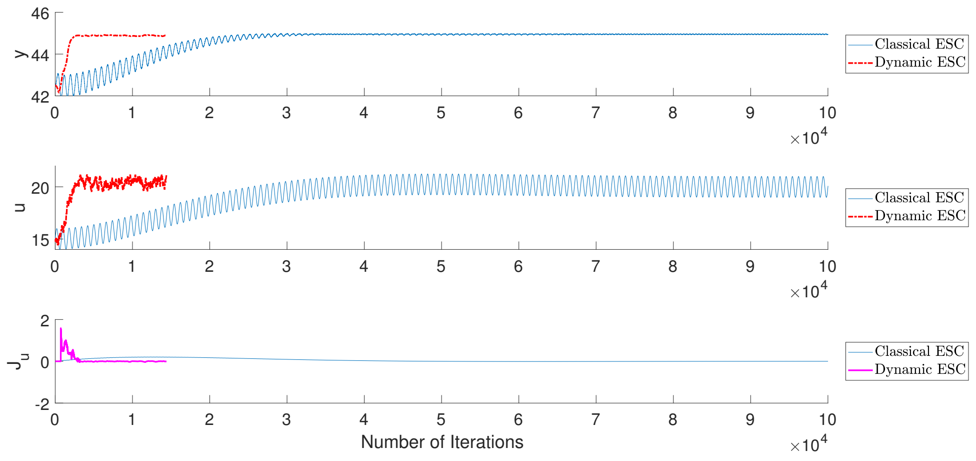

Figure 23: Simulation result comparison of a dynamic and classic ESC for second-order without zero($\tau_a = 0$) system

4.2.2 Second-order transfer function with zero ($\tau_a = 40$)

In the second-order transfer function with zero system, the numerical spikes were not a huge issue compared to other systems, but the same method is applied as illustrated in Figure 24. Because the peaks were not serious, it was possible to fix it with only with a small threshold of 0.001, which can be compared with Figure 16.

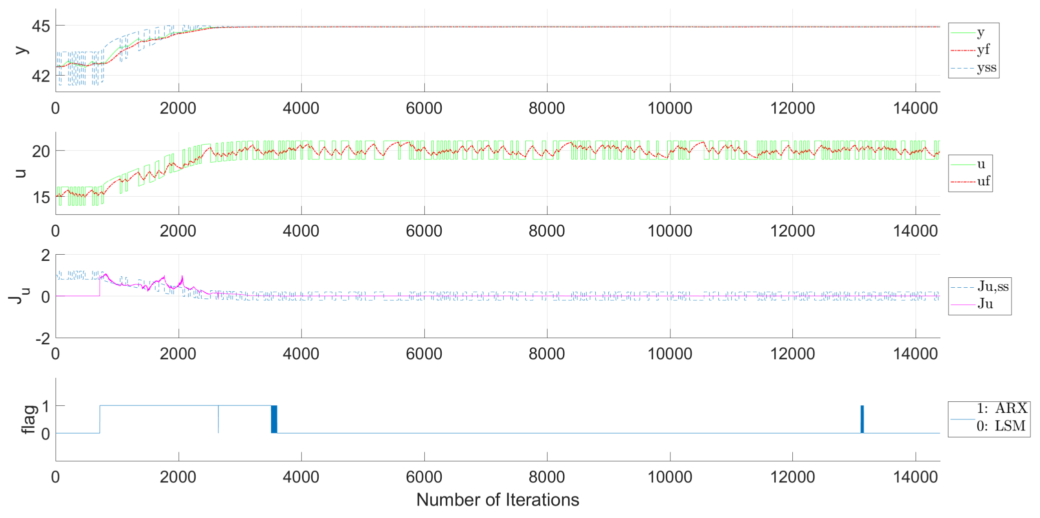

Figure 24: Modified dynamic ESC simulation result for second-order with zero($\tau_a = 40$) system: $K_i = 0.005$, PRBS$= 30$, and $l = 720$ with a tolerance of 0.001

In Figure 25, the classic ESC takes around $5 \cdot 10^4$ iterations as well, while the dynamic ESC requires 3500 iterations or 2780 transient iterations(5.6% of the classic ESC).

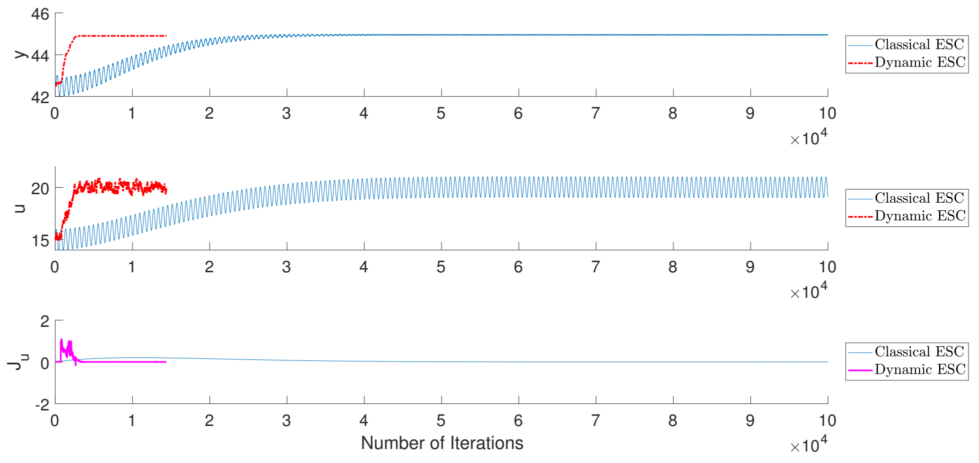

Figure 25: Simulation result comparison of a dynamic and classic ESC for second-order with zero($\tau_a = 40$) system

Overall, the classic ESC requires a significant number of iterations compared to the dynamic ESC, verifying the superior ability of the dynamic ESC developed in this project.

5 Discussion

With the motivation of figuring out a dynamic ESC algorithm which is expected to achieve a faster convergence to a new extremum, the ARX model is introduced to estimate the steady-state gradient which is a local linear dynamic model. In this discussion section, the problem-solving processes of the dynamic ESC will be focused and summarized.

First of all, a problem caused when the second-order transfer function without zero is applied. Specifically, the estimated steady-state gradient could not follow the true gradient during simulations.

The approach to explain this behavior comes from the step responses of different transfer functions since there was no such a problem in the first-order system. In detail, the step response of the second-order system has a secondary time lag, and the sinusoidal wave is not enough to obtain the full information of the output. With this approach, the PRBS wave is introduced instead of a simple sinusoidal perturbation to provide multiple frequencies, and it could solve the problem.

The second problem is detected in all of the three transfer function systems that numerical spikes happen and sometimes they make deviations. The ARX model accounts for this problem with the observation that the spikes are prone to firstly happen right after the estimated steady-state gradient becomes close to zero. In specific, when there is a change in the input near the extremum, the output almost does not change making the estimation of only one parameter possible while the ARX model still tries to estimate two parameters in the first-order system or four parameters in the second-order system.

To fix this phenomenon, the combination of the ARX model with the LS method is introduced with the idea that the ARX model estimates gradient faster in the early stages, and it is switched to the LS method when the estimated gradient is close enough to zero. Although the LS method is a kind of a local linear static model, it could not affect the iteration numbers to converge since most of the important part is estimated by the ARX model. This implementation is concluded to work efficiently making the dynamic ESC more robust.

6 Conclusion

In this project, the slow control performance of the classic ESC provided the biggest motivation to develop the dynamic ESC. The main idea of this algorithm is introducing a local linear dynamic model for the gradient estimation in place of a local linear static model. One of the simple time-variant models is the ARX model, and it is implemented in this project.

There were two main problems solved in this project. The first problem happened when the second-order transfer function is applied. In detail, the sinusoidal wave perturbation could not obtain the full information of y. In other words, a sinusoidal wave with only one frequency is not sufficient to make the ARX model to estimate four parameters requiring diverse frequencies in the Fourier domain. Thus, PRBS(pseudo-random binary sequence) with an amplitude of 1 is applied instead of it in the second-order system.

The second problem happened in the ARX model. In specific, it is possible to estimate only one parameter near the extremum since there is almost no change in the output. However, the ARX model estimates multiple parameters altogether and this makes numerical problems. This problem is solved by switching the ARX model to LS method when the gradient is sufficiently close to zero.

The dynamic ESC has tuning parameters which are the controller gain, perturbation time constant(or PRBS calculation steps), and a window size of data sets for estimating the gradient using the ARX model. Different sets of tuning parameters are tested in all of the three transfer function cases and obtained the best-tuned results. This is an important step because wrong tuning parameters give unwanted behaviors such as having small offsets in the input from the true extremum, causing frequent numerical spikes, deviating from the true optimum, and making the ARX estimation noisy which are well described in the appendix A.

In the comparison with the simulation results of the classic ESC, it was shown that the iteration step was hugely reduced in the dynamic ESC, especially only $3.6 - 5.6\%$ of iterations were needed for the dynamic ESC. It verifies that the dynamic ESC is more efficient and faster control strategy than the classic ESC.

Overall, it is concluded that the dynamic ESC is successfully implemented in the first- and second-order systems. The estimated steady-state gradient follows the true gradient, and the system converged into the true extremum. Based on this study, several future works are suggested which will be done in Spring 2018.

First of all, since this project is aiming to find an algorithm for the dynamic ESC using the ARX model, the initial point is set to be $(15, 42.5)$ and make it converge to (20.45). Thus, the next step is adding a disturbance on the system which makes the relationship of the input and output changes, and monitoring if the (u,y) converges to a new optimal point.

Moreover, in this project, the Hammerstein and Wiener models are used to describe the plant process as suggested in the classic ESC modeling[2] where a static non-linearity($f(u)$) and a linear dynamic system($G(s)$) are divided. However, in reality, a process where the non-linearity and the dynamic parts are combined together is more general. An example of this process which can be used in ESC is a parallel heat exchangers with stream split provided by Jaschke and Skogestad (2014). In the process, the objective is maximizing the output temperature and it has a quadratic relationship with the split, which is similar to the relationship of the gas injection rate and the oil production rate. This heat exchangers process will be used to implement the dynamic ESC to check its real application in the future. This process has an advantage that there are dynamic changes in the output even near the extremum so that the numerical spikes problem are expected not to happen.

Finally, the model predictive controller can be applied replacing a simple integral controller when the process is a multi-variable system. Because a better controller gives a better control, it is expected to improve the dynamic ESC algorithm.

References

[1] Eikrem, G. O., Aamo, O. M., & Foss, B. A. (2008). On instability in gas lift wells and schemes for stabilization by automatic control. *SPE Production & Operations*, 23(02), 268-279.

[2] Peixoto, A. J., Pereira-Dias, D., Xaud, A. F., & Secchi, A. R. (2015). Modelling and extremum seeking control of gas lifted oil wells. *IFAC-PapersOnLine*, 48(6), 21-26.

[3] Aliev, F. A., & Jamalbayov, M. A. (2015, October). Theoretical Basics of Mathematical Modeling of the Gas Lift Process in the Well-Reservoir System (Russian). In *SPE Russian Petroleum Technology Conference*. Society of Petroleum Engineers.

[4] Abd-Elrady, E., & Gan, L. (2008). Identification of Hammerstein and Wiener models using spectral magnitude matching. *IFAC Proceedings Volumes*, 41(2), 6440-6445.

[5] Krstić, M., & Wang, H. H. (2000). Stability of extremum seeking feedback for general nonlinear dynamic systems. *Automatica*, 36(4), 595-601.

[6] Krishnamoorthy, D., Pavlov, A., & Li, Q. (2016). Robust Extremum Seeking Control with application to Gas Lifted Oil Wells. *IFAC-PapersOnLine*, 49(13), 205-210.

[7] Hunnekens, B. G. B., Haring, M. A. M., van de Wouw, N., Nijmeijer, H. (2014, December). A dither-free extremum-seeking control approach using 1st-order least-squares fits for gradient estimation. In Decision and Control (CDC), *2014 IEEE 53rd Annual Conference on* (pp. 2679-2684). IEEE.

[8] Choi, J. Y., Krstic, M., Ariyur, K. B., & Lee, J. S. (2002). Extremum seeking control for discrete-time systems. *IEEE Transactions on automatic control*, 47(2), 318-323.

[9] Ljung, L. (1999). System identification: Theory for the user, ptr prentice hall information and system sciences series. *ed: Prentice Hall, New Jersey.*

[10] C. J. (2016, February 18). Maximizing Production through Optimized Gas Lifting. Retrieved December 13, 2017, from https://www.emersonprocessxperts.com/2016/02/maximizing-production-optimized-gas-lifting

Appendix

Appendix A Dynamic ESC tuning and numerical spikes modification

A.1 Plant process with first-order transfer function

A.1.1 Changes in controller gain

Table A5: Simulation result of a dynamic ESC with a plant model containing first-order transfer function when K_i varies.

No.	Controller gain(K_i)	Dither time constant(T_D)	Window size (l)	Convergence iterations	Transient iterations	True extremum
1	0.001	180	$2 * 180$	7200 ↑	7200 ↑	O
2	0.002	180	$2 * 180$	7200 ↑	7200 ↑	O
3	0.003	180	$2 * 180$	4200	3840	O
4	0.004	180	$2 * 180$	3000	2640	O
5	0.005	180	$2 * 180$	2500	2140	O
6	0.006	180	$2 * 180$	2000	1640	offset*
7	0.007	180	$2 * 180$	1800	1440	offset*
8	0.008	180	$2 * 180$	1700	1340	offset*
9	0.009	180	$2 * 180$	1600	1240	offset*

* It converged to the optimum, but in the control input, there was a small offset

Table A5 shows the simulation result with various controller gains while other tuning parameters are fixed. Overall, with the increase in K_i value, the transient to reach a new extremum tents to significantly decrease. Results of simulations 1, 5, 6, and 9 are listed below.

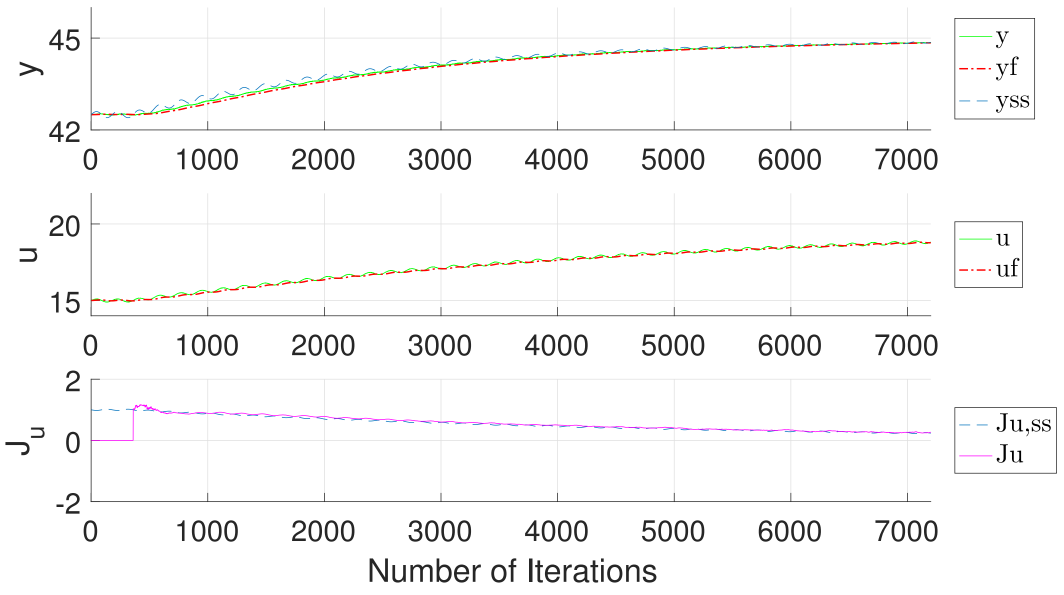

Figure A26: Simulation result for first-order system using a dynamic ESC: $K_i = 0.001$, $T_D = 180$, and $l = 2 * 180$.

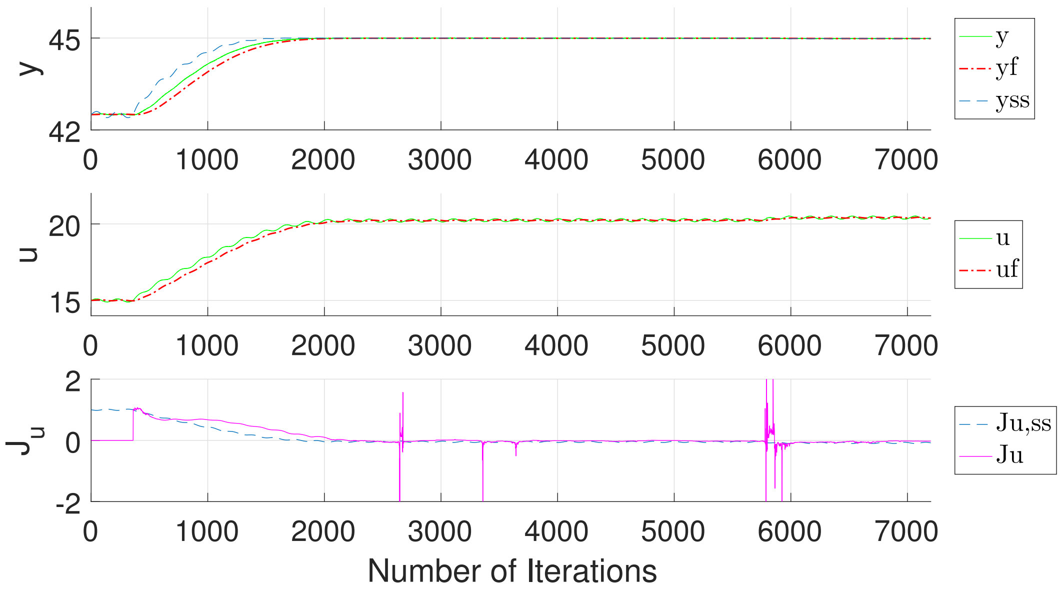

Figure A28: Simulation result for first-order system using a dynamic ESC: $K_i = 0.006$, $T_D = 180$, and $l = 2 * 180$.

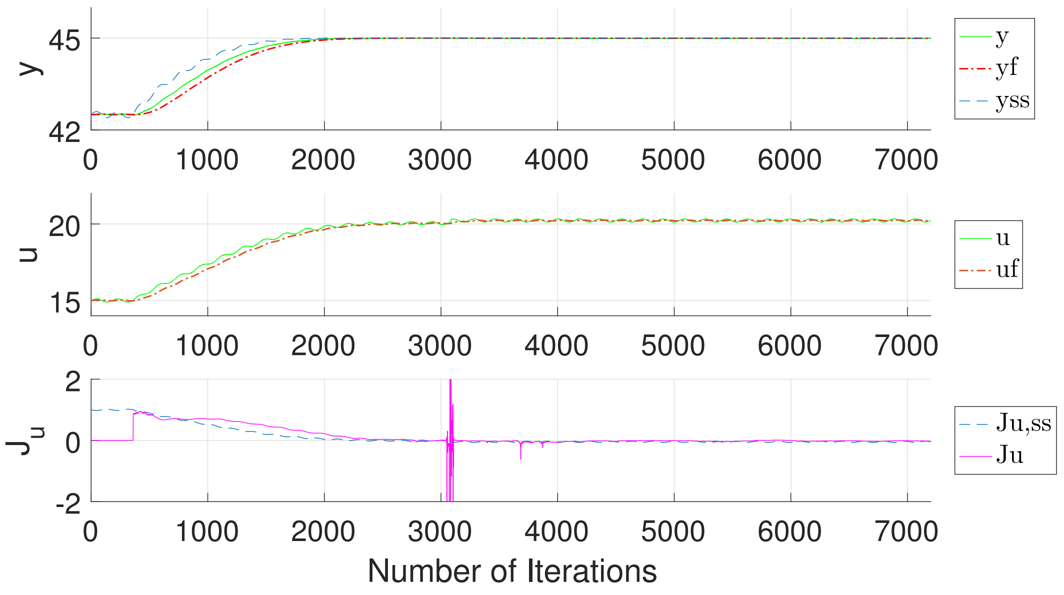

Figure A27: Simulation result for first-order system using a dynamic ESC: $K_i = 0.005$, $T_D = 180$, and $l = 2 * 180$.

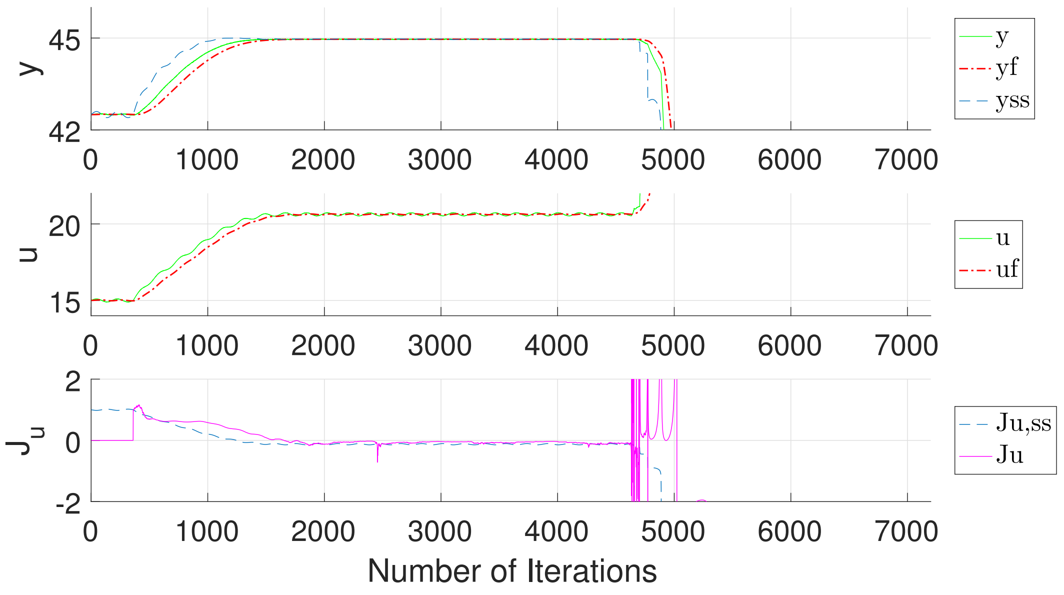

Figure A29: Simulation result for first-order system using a dynamic ESC: $K_i = 0.009$, $T_D = 180$, and $l = 2 * 180$.

Figure A26 is obtained when $K_i = 0.001$. It is clearly indicated that the estimated gradient J_u follows the true gradient, saying that the ARX model is working as intended. The dynamic with this small integral controller is so slow that it needs more iteration to converge to the maximum where the gradient becomes zero.

Figure A27 is obtained when $K_i = 0.005$. Around 2400 number of iterations, it converges to the true extremum at $(20, 45)$. However, there is an issue on this Figure A27 that the result slightly deviates from its maximum around 3100 iterations caused by aggressive peaks in J_u. This problem is well modified in section 4, and the process to fix it will be shown at the end of this section.

For $K_i = 0.006$, the result converges around 2000 iterations to a point slightly passing the extremum as illustrated in Figure A28. Moreover, in Figure A29 obtained when $K_i = 0.009$, it also converges to point having a small offset from the true extremum with 1500 iterations as the case with $K_i = 0.006$. However, the situation is much worse with $K_i = 0.009$ and the numerical peaks in J_u make a serious deviation around 4800 iterations.

Since the simulation results from $K_i = 0.006$ to $K_i = 0.009$ pass the extremum, $K_i = 0.005$ is chosen for next simulations to see how other tuning parameters effect on the number of iterations.

A.1.2 Changes in dither time constant and window size

Table A6 shows simulation results when K_i is fixed as 0.005 and only various sets of T_D and l are tested.

Table A6: Simulation result of a dynamic ESC with a plant model containing first-order transfer function when dither time constant and window size vary.

No.	Controller gain(K_i)	Dither time constant(T_D)	Window size (l)	Convergence iterations	Transient iterations	True extremum
1	0.005	30	$1 * 30$	2400	2370	O
2	0.005	30	$6 * 30$	2500	2380	O
3	0.005	30	$10 * 30$	2800	2500	O
4	0.005	30	$20 * 30$	2800	2200	O
5	0.005	180	$1 * 180$	2600	2420	O
6	0.005	180	$2 * 180$	2500	2140	O
7	0.005	180	$3 * 180$	2600	2060	O
8	0.005	180	$4 * 180$	3100	2380	O
9	0.005	180	$8 * 180$	–	-	X
10	0.005	$4 * 180$	$4 * 180$	3000	2280	O
11	0.005	$4 * 180$	$8 * 180$	–	-	X

Comparing simulation number 2 and 5 or 8 and 10 in Table A6 states situations that different dither time constants are applied while other tuning parameters are same. Based on the fact that the Transient iterations are similar, it can be concluded that the number of iterations is almost independent of the dither time constant. Simulations number 2 and 5 are stated in Figure A30 and A31 respectively.

In terms of the window size, the transient iterations depending on the window size do not have a clear tendency as well. However, it becomes meaningful in regards to robustness when we have a look at simulation figures. In detail, figures for simulations number 5 to 9 are provided to give the explanation where the window size only varies in Figure A31 to A34.

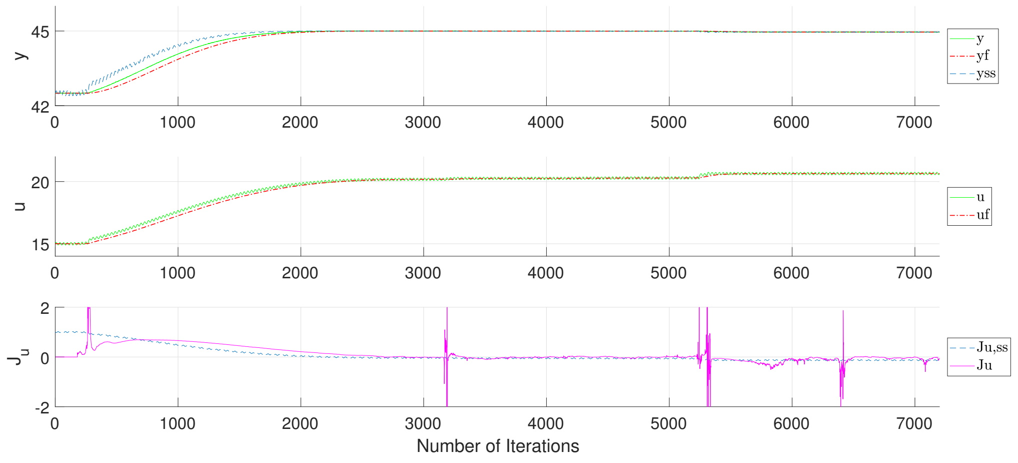

Figure A30: Simulation result for first-order system using a dynamic ESC: $K_i = 0.005$, $T_D = 30$, and $l = 180$.

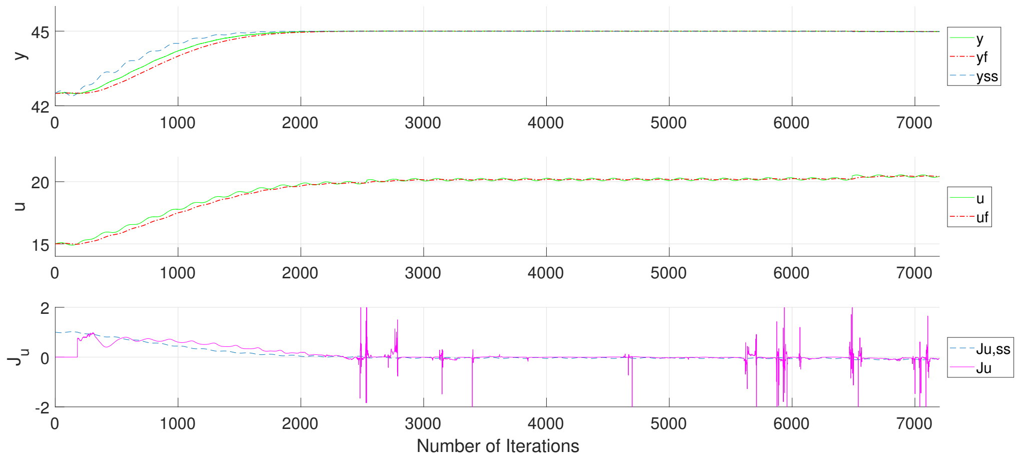

Figure A31: Simulation result for first-order system using a dynamic ESC: $K_i = 0.005$, $T_D = 180$, and $l = 180$.

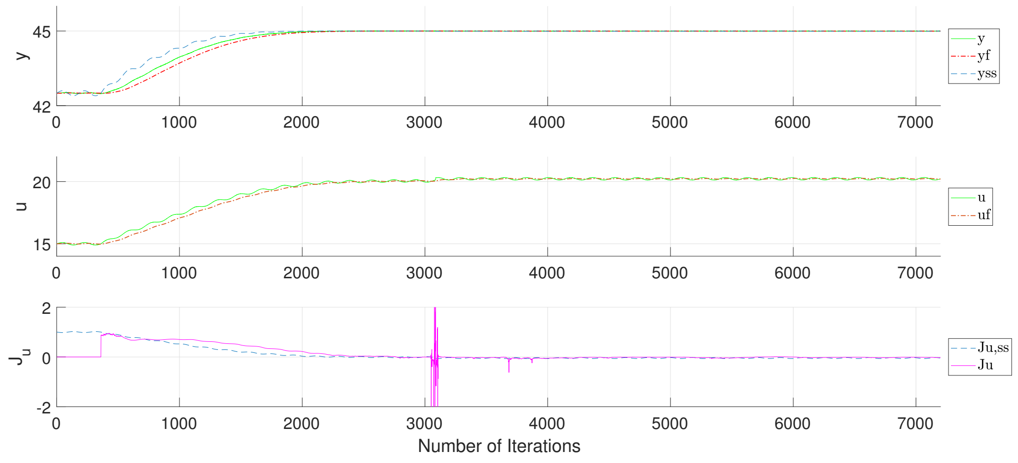

Figure A32: Simulation result for first-order system using a dynamic ESC: $K_i = 0.005$, $T_D = 180$, and $l = 2 * 180$.

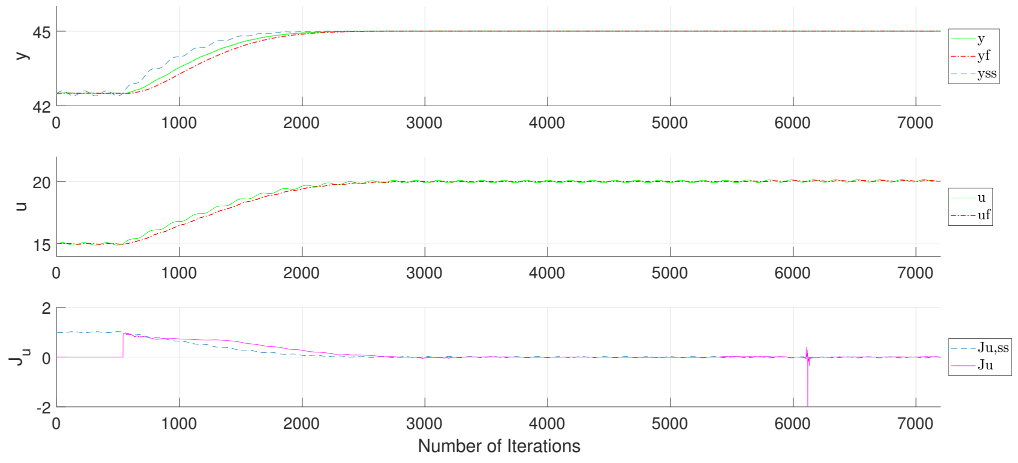

Figure A33: Simulation result for first-order system using a dynamic ESC: $K_i = 0.005$, $T_D = 180$, and $l = 3 * 180$.

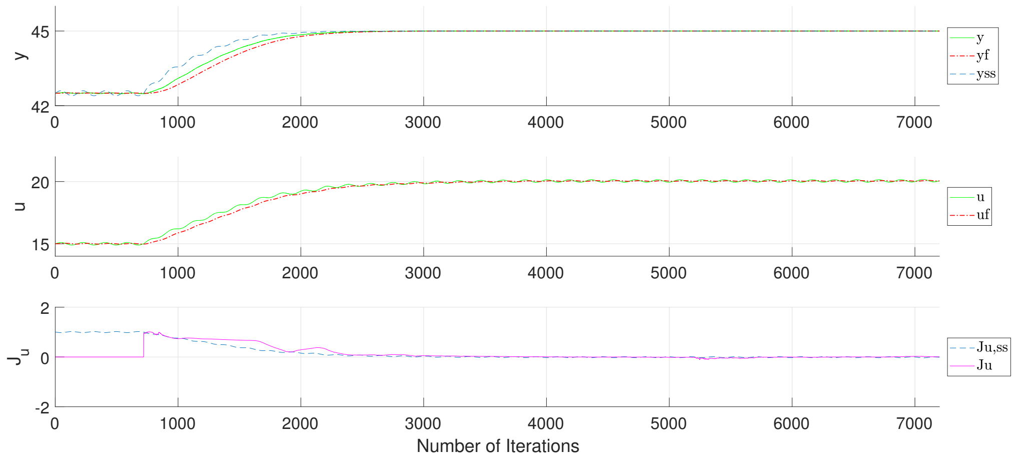

Figure A34: Simulation result for first-order system using a dynamic ESC: $K_i = 0.005$, $T_D = 180$, and $l = 4 * 180$.

Figure A31 is obtained when the window size equals to 180. After J_u reaches zero, a lot of small numerical peaks happens, and around 6500 iterations peaks with amplitudes of 50 make recognizable deviations from the true extremum.

When the window size increased to 360 as illustrated in Figure A32, small numerical peaks almost disappear and relatively big peaks with an amplitude of 32 about 3000 iterations make deviations.

With the increased window as 540, only small numerical peaks around 600 iterations occur with amplitudes of 3 as depicted in Figure A33, and even in the 720 window size, the result becomes stable without rigid peaks as in Figure A34.

Based on these observations, steady-state gradient estimation with a sufficient number of data provides stable and robust results. This tendency is also obtained with simulation number 1 to 4 that numerical peaks almost disappear in simulation number 4.

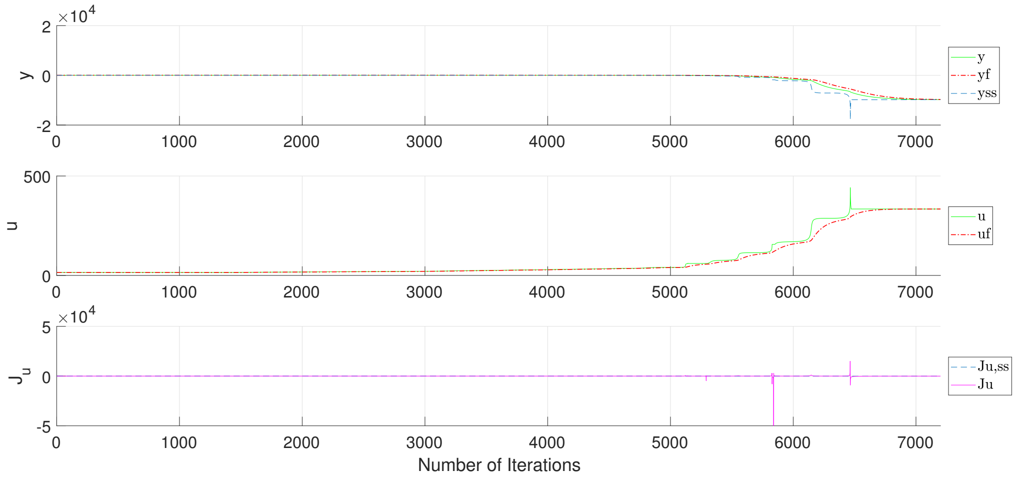

Figure A35: Simulation result for first-order system using a dynamic ESC: $K_i = 0.005$, $T_D = 180$, and $l = 8 * 180$.

However, as shown in Figure A35, too large window size gives huge numerical peaks which surely trigger big deviations. Because the result hits the extremum only for a second, it was not possible to get a number of iterations to converge. The same result is obtained in simulation number 11 so that it is recommended a moderate length of the window size in this first-order system.

A.1.3 Numerical spike modification

Figure A36 is obtained where the ARX model is used when $J_u > 0.002$ and the LS method is used when $J_u < 0.002$. This result shows that the $J_u = 0.002$ were not able to remove the numerical spikes. This is because $J_u = 0.002$ is still causing problems in the ARX model when it estimates parameters to calculate the steady-state gradient.

Therefore, it is tested with larger thresholds. When the switching point is set to be 0.008, the serious numerical spikes are removed as shown in Figure A37. Although there are small spikes in the result, they are not seriously affecting the overall result.

When it is simulated with a threshold of 0.010, it is clear that the numerical peaks are all removed as illustrated in Figure A38. Compared to Figure A36 and A37, there are less conflicts between the ARX model and LS method. As intended, the estimated gradient converges fast to zero at the initial stage with the ARX model, so that the combination of the two methods does not effect on the overall iteration number to converge. Therefore, this approach is concluded to work well to fix the numerical problem in the first-order system and will be tried as well in the second-order system.

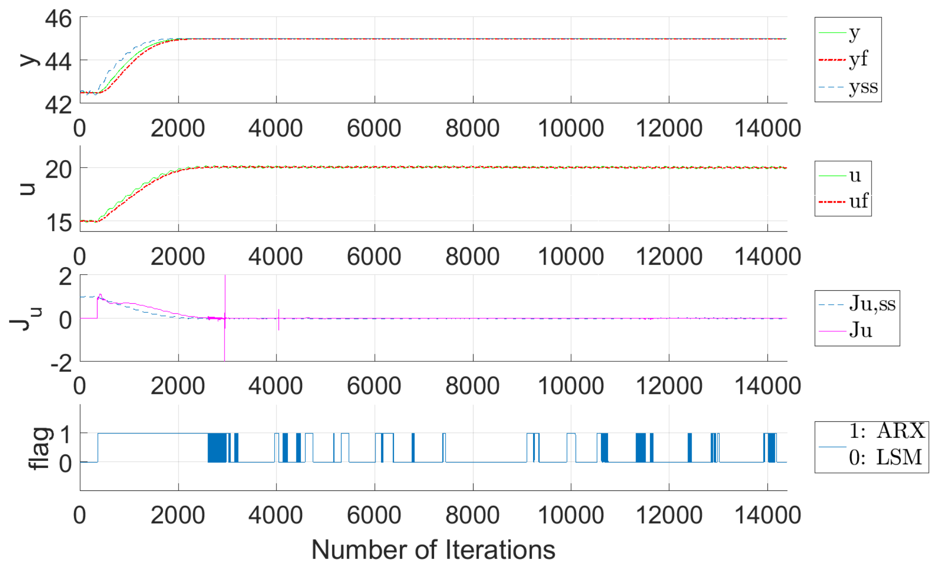

Figure A36: Modified dynamic ESC simulation result for first-order system: $K_i = 0.005$, $T_D = 180$, and $l = 360$ with a tolerance of 0.002.

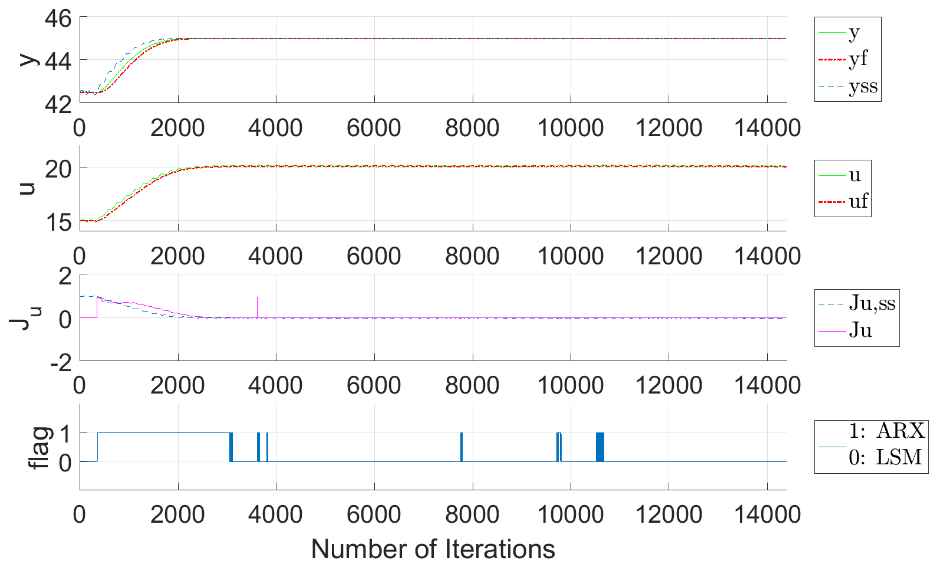

Figure A37: Modified dynamic ESC simulation result for first-order system: $K_i = 0.005$, $T_D = 180$, and $l = 360$ with a tolerance of 0.008.

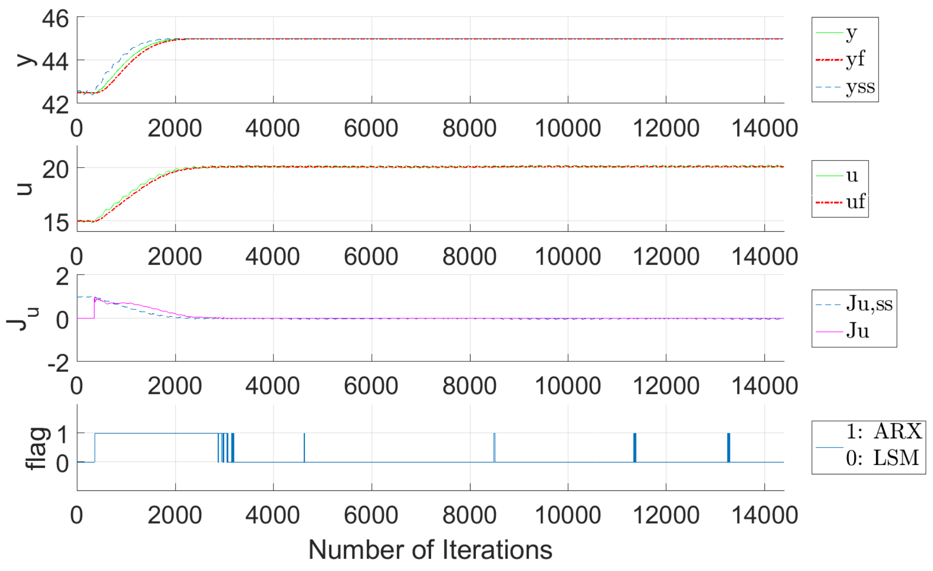

Figure A38: Modified dynamic ESC simulation result for first-order system: $K_i = 0.005$, $T_D = 180$, and $l = 360$ with a tolerance of 0.010.

A.2 Plant process with second-order transfer function without zero($\tau_a = 0$)

A.2.1 Changes in PRBS calculation step

Table A7: Simulation result of a dynamic ESC with a plant model containing second-order transfer function without zero when PRBS varies.

No.	Controller gain(K_i)	PRBS calculation steps	window size(l)	Convergence iterations	Transient iterations	True extremum
1	0.005	30	$4 * 180$	2500	1780	O
2	0.005	60	$4 * 180$	3000	2280	O
3	0.005	90	$4 * 180$	3000	2280	O
4	0.005	180	$4 * 180$	3000	2280	O

Table A7 shows the simulation result when PRBS calculation step is changed from 30 to 180. Overall, there is no huge difference on the transient iteration number depending on PRBS calculation steps.

Figure A39 to A42 are obtained for each PRBS calculation steps. Even though the numerical spikes are problematic in all cases arousing deviations, all of the results converge to the right extremum at the first time around $2500 - 3000$ iterations. The PRBS recalculation step is set to be every 30 because the amplitude of numerical peaks with larger PRBS step tends to be large.

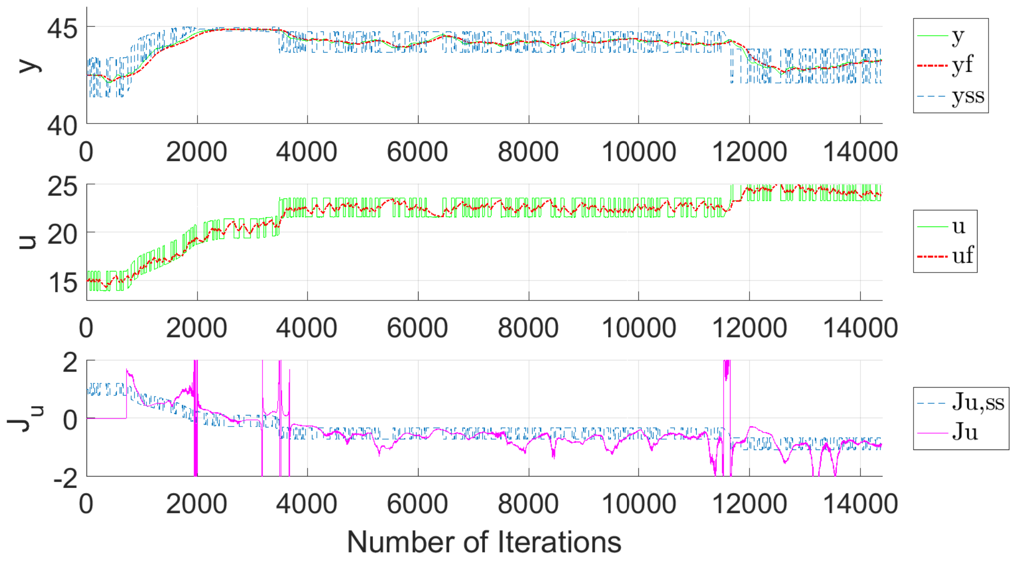

Figure A39: Simulation result for second-order without zero($\tau_a = 0$) system using a dynamic ESC: $K_i = 0.005$, PRBS= 30, and $l = 720$

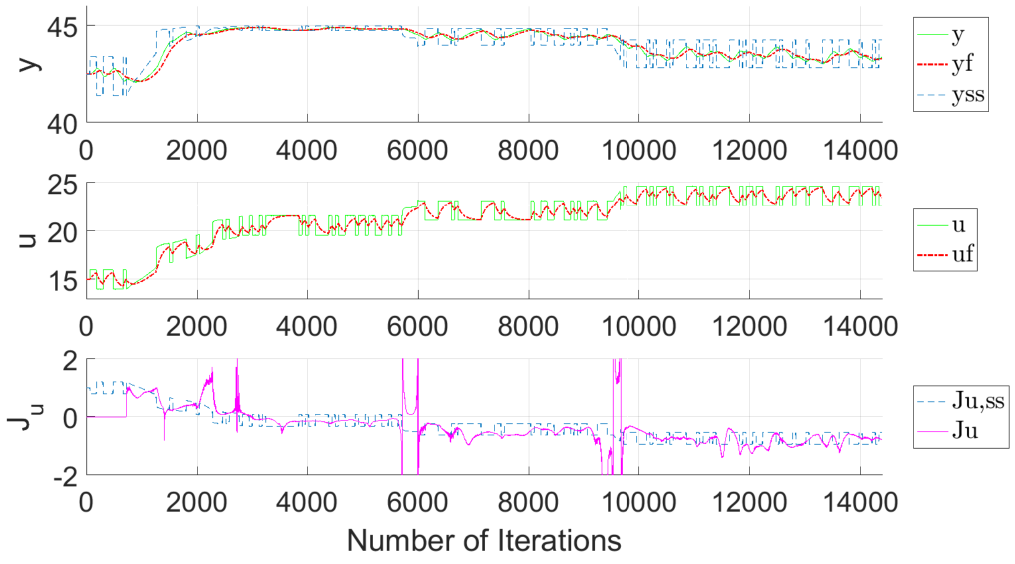

Figure A40: Simulation result for second-order without zero($\tau_a = 0$) system using a dynamic ESC: $K_i = 0.005$, PRBS= 60, and $l = 720$

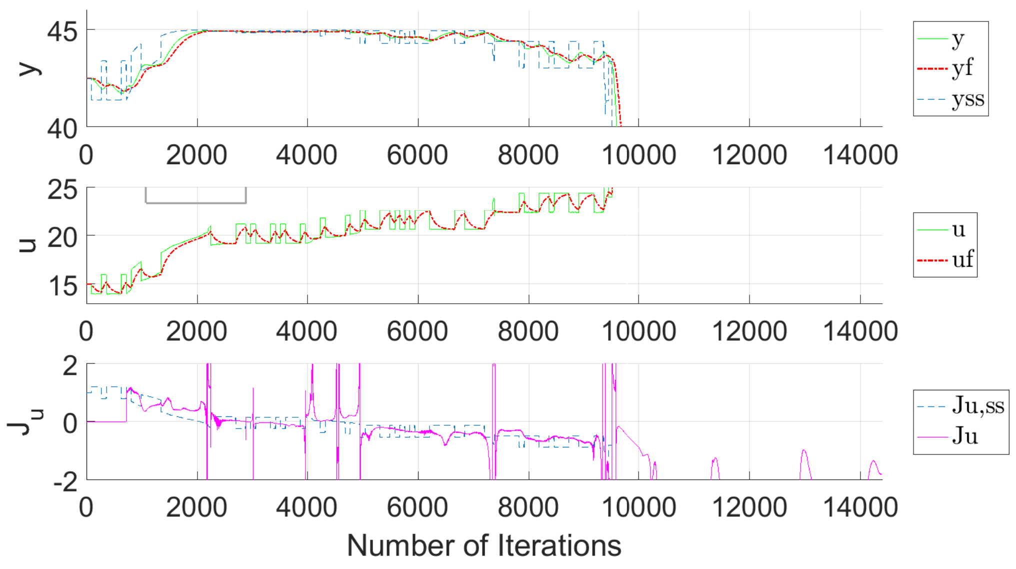

Figure A41: Simulation result for second-order without zero($\tau_a = 0$) system using a dynamic ESC: $K_i = 0.005$, PRBS= 90, and $l = 720$

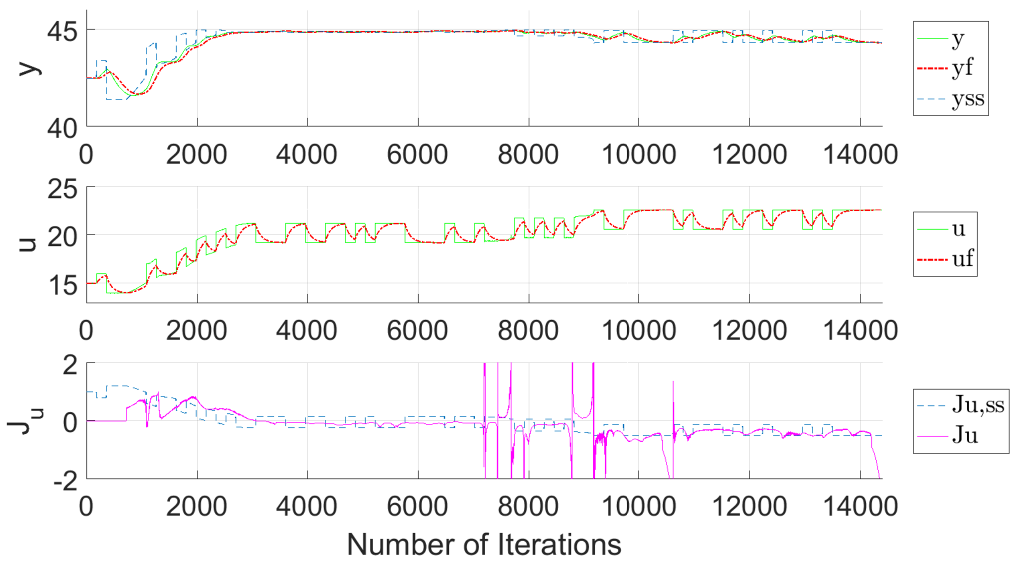

Figure A42: Simulation result for second-order without zero($\tau_a = 0$) system using a dynamic ESC: $K_i = 0.005$, PRBS= 180, and $l = 720$

A.2.2 Changes in controller gain

Table A8: Simulation result of a dynamic ESC with a plant model containing second-order transfer function without zero when K_i varies.

No.	Controller gain(K_i)	PRBS calculation steps	window size(l)	Convergence iterations	Transient iterations	True extremum
1	0.001	30	$4 * 180$	10000	9280	O
2	0.002	30	$4 * 180$	6000	5280	O
3	0.003	30	$4 * 180$	4000	3280	O
4	0.004	30	$4 * 180$	3000	2280	O
5	0.005	30	$4 * 180$	2500	1780	O
6	0.006	30	$4 * 180$	2400	1680	offset*
7	0.007	30	$4 * 180$	2300	1580	offset*
8	0.008	30	$4 * 180$	2200	1480	offset*

* It converged to the optimum, but in the control input, there was a small offset

Table A8 illustrates the results with different applications of the integral gain values. Based on the previous PRBS simulations. PRBS is set to be calculated every 30 iteration, and the window size is fixed as 720.

Figure A43 is obtained with $K_i = 0.005$. With the increase of K_i values, the number of iteration to converge decreased to 2500, which is a quarter of iteration with $K_i = 0.001$. After convergence, the result deviates from the extremum by numerical spikes around 3500 and 11500 iterations, so it is necessary to remove them to achieve robustness.

When $K_i = 0.006$, the results start to converge passing the extremum with small offsets due to a large controller gain. Figure A44 shows the simulation result when $K_i = 0.008$, which converges to $u = 21$ and deviates later by numerical spikes. Thus, K_i is set to be 0.005 in next simulations.

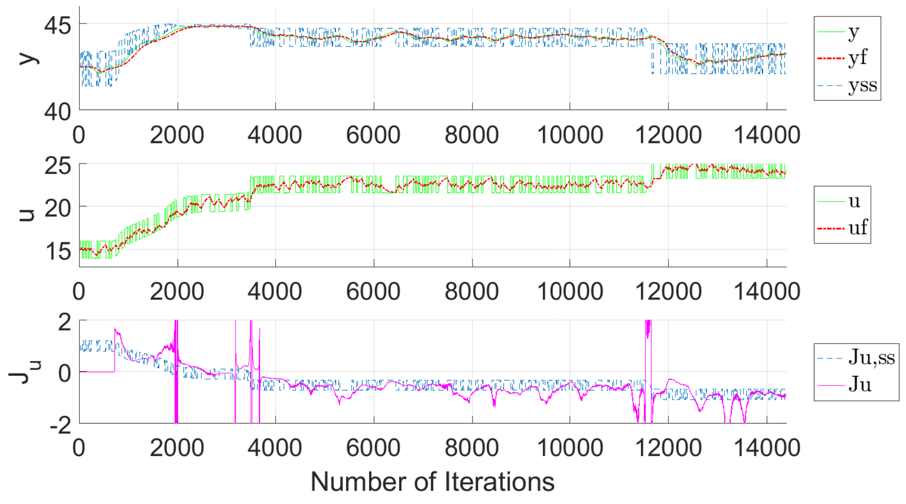

Figure A43: Simulation result for second-order without zero($\tau_a = 0$) system using a dynamic ESC: $K_i = 0.005$, PRBS= 30, and $l = 720$

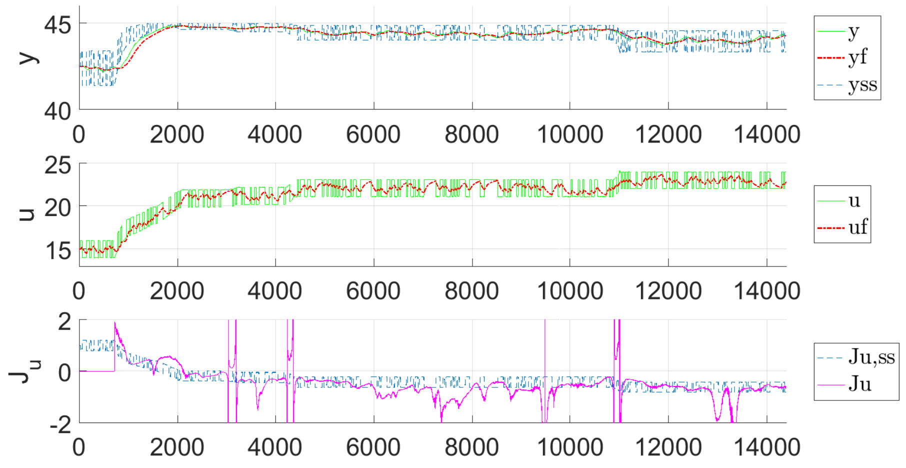

Figure A44: Simulation result for second-order without zero($\tau_a = 0$) system using a dynamic ESC: $K_i = 0.008$, PRBS= 30, and $l = 720$

A.2.3 Changes in window size

Table A9 provides simulation results when the gradient estimation window size varies. Other parameters, K_i, and PRBS are set to be 0.005 and 30 respectively.

Figure A45 to A49 are plots for the simulation results. They show that a small window size results in an aggressive behavior with frequent numerical peaks while a large window size gives a more robust result. The best-tuned result is chosen to be simulation number 4 and numerical spikes modification is performed in the next section.

Table A9: Simulation result of a dynamic ESC with a plant model containing second-order transfer function without zero when the window size varies

No.	Controller gain(K_i)	PRBS calculation steps	window size(l)	Convergence iterations	Transient iterations	True extremum
1	0.005	30	$1 * 180$	1100	920	
2	0.005	30	$2 * 180$	1400	1040	
3	0.005	30	$3 * 180$	1600	1060	
4	0.005	30	$4 * 180$	2500	1780	
5	0.005	30	$5 * 180$	3000	2100	

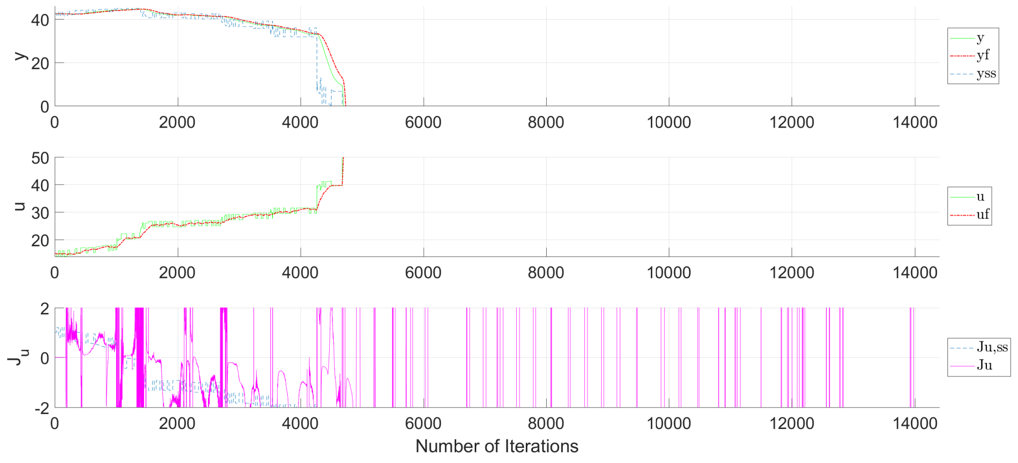

Figure A45: Simulation result for second-order without zero($\tau_a = 0$) system using a dynamic ESC: $K_i = 0.005$, PRBS= 30, and $l = 180$

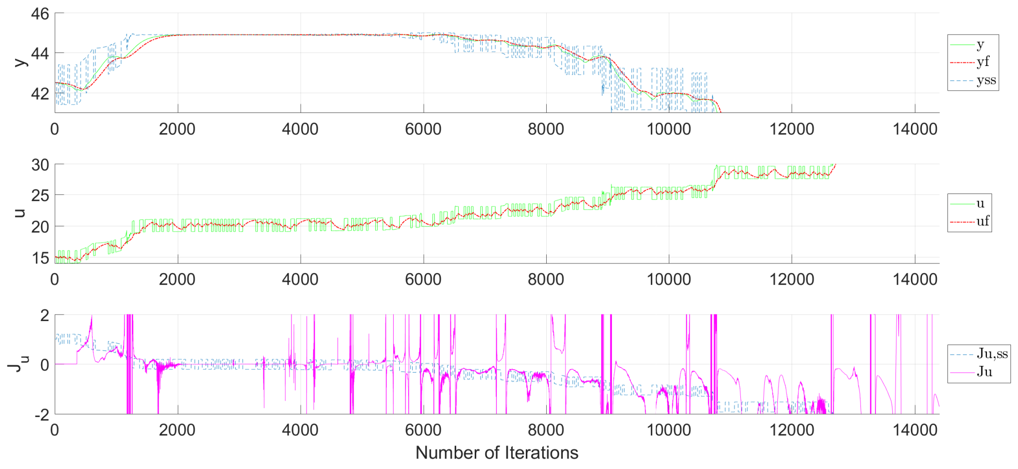

Figure A46: Simulation result for second-order without zero($\tau_a = 0$) system using a dynamic ESC: $K_i = 0.005$, PRBS= 30, and $l = 360$

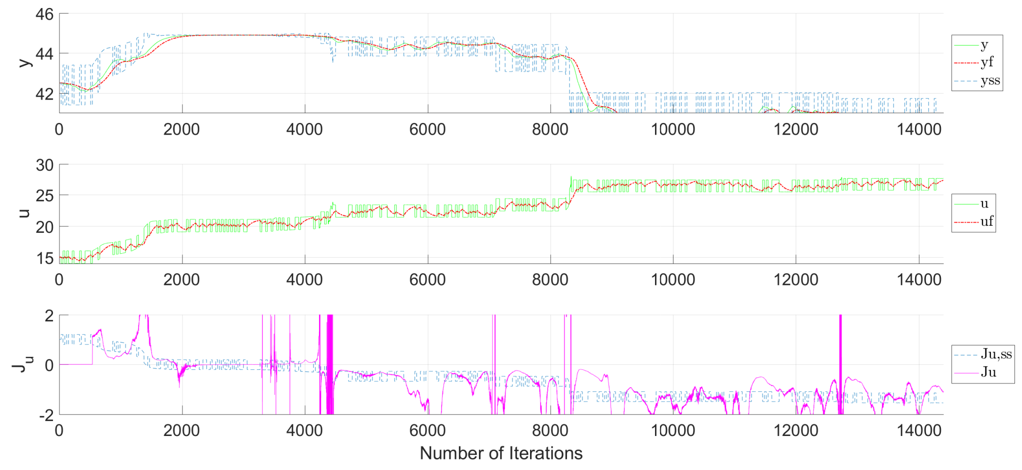

Figure A47: Simulation result for second-order without zero($\tau_a = 0$) system using a dynamic ESC: $K_i = 0.005$, PRBS$=30$, and $l = 540$

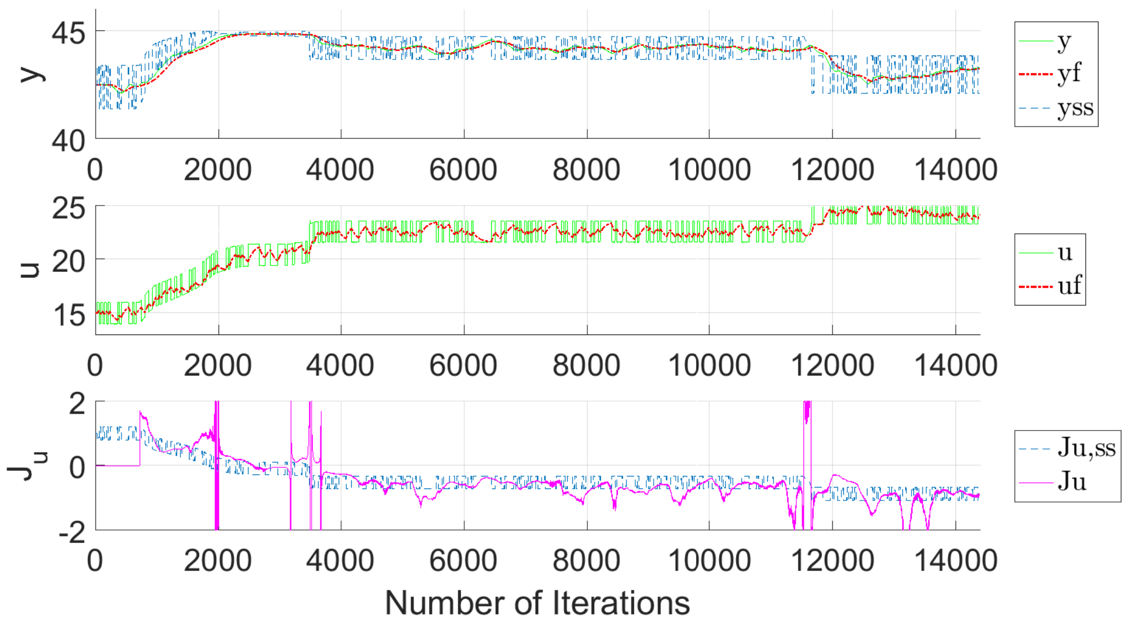

Figure A48: Simulation result for second-order without zero($\tau_a = 0$) system using a dynamic ESC: $K_i = 0.005$, PRBS$=30$, and $l = 720$

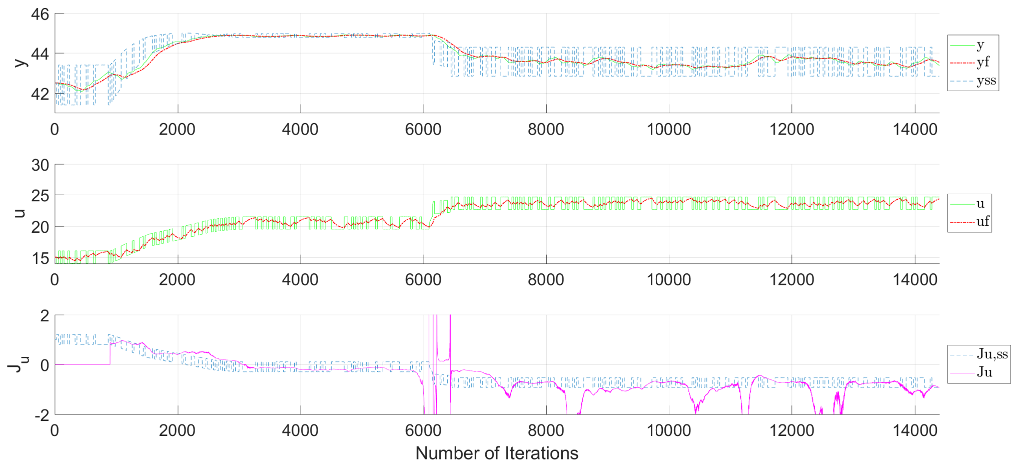

Figure A49: Simulation result for second-order without zero($\tau_a = 0$) system using a dynamic ESC: $K_i = 0.005$, PRBS= 30, and $l = 900$

A.2.4 Numerical spikes modification

Figure A50 is obtained with a threshold of 0.001. It is not enough to remove the numerical spikes and in some region, the ARX model and LS method conflicts. With an increased tolerance of 0.016 as shown in Figure A51, the situation becomes better while the numerical peaks still exist.

In Figure A52 with a relatively higher tolerance of 0.030, serious numerical matters have disappeared. Although the ARX model and LS method conflicts around 3000 iterations, it is acceptable.

However, when simulations in the same condition are performed multiple times, it rarely but sometimes gives a worse result than Figure A52. A possible reason for this is that the PRBS frequency becomes different in every simulation and sometimes the frequency is too small and gives a noise like behavior. Thus, it can be considered to use higher PRBS frequencies such as 60 in the future work.

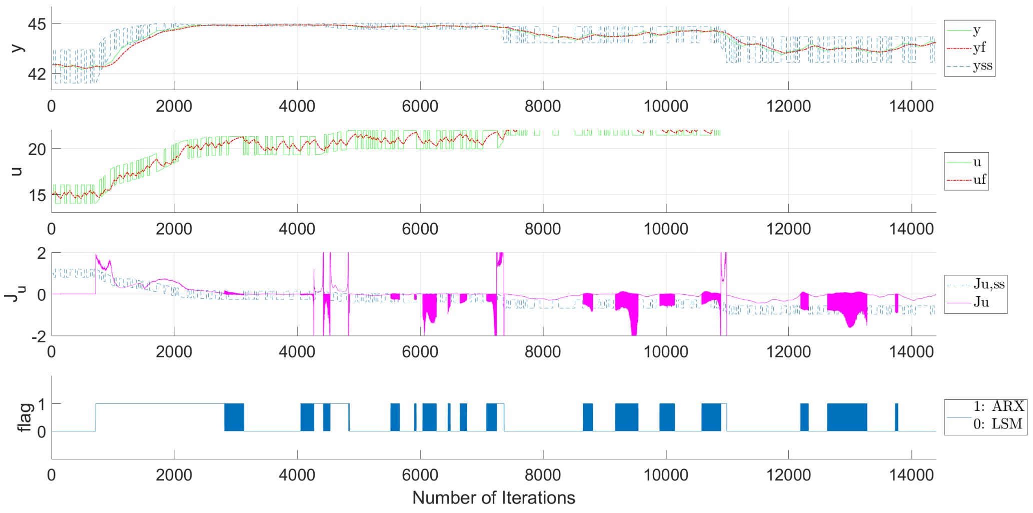

Figure A50: Modified dynamic ESC simulation result for second-order without zero($\tau_a = 0$) system: $K_i = 0.005$, PRBS= 30, and $l = 720$ with a tolerance of 0.001.

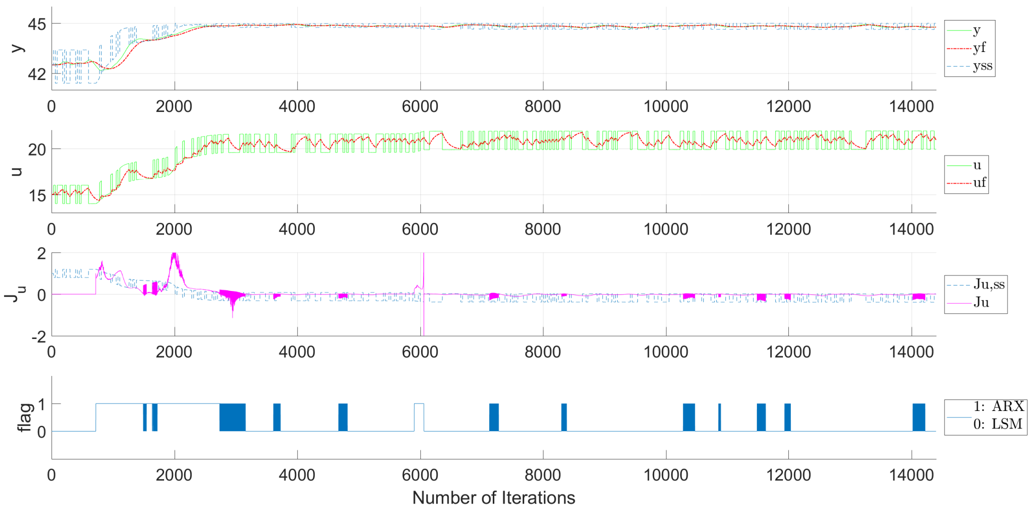

Figure A51: Modified dynamic ESC simulation result for second-order without zero($\tau_a = 0$) system: $K_i = 0.005$, PRBS= 30, and $l = 720$ with a tolerance of 0.016.

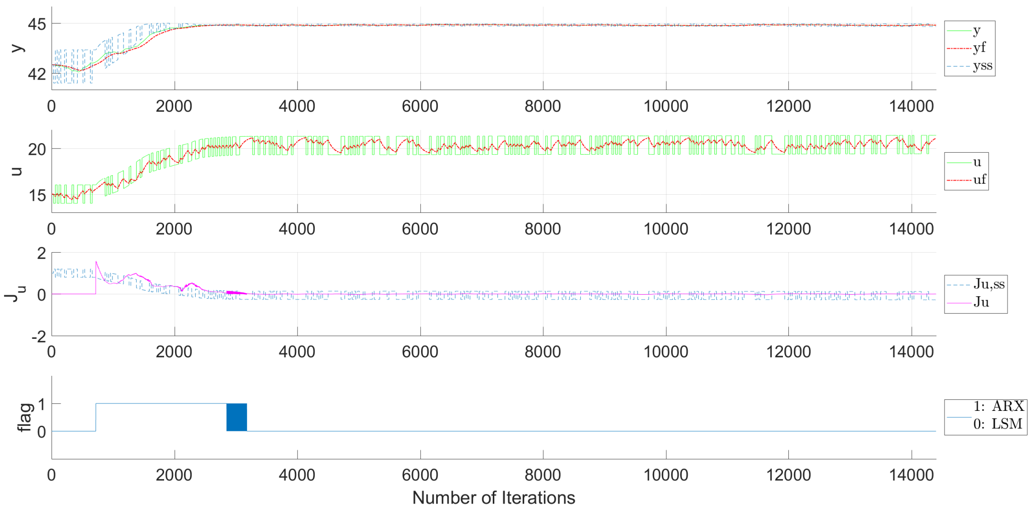

Figure A52: Modified dynamic ESC simulation result for second-order without zero($\tau_a = 0$) system: $K_i = 0.005$, PRBS= 30, and $l = 720$ with a tolerance of 0.030.

A.3 Plant process with second-order transfer function with zero($\tau_a = 40$)

A.3.1 Changes in PRBS

Table A10: Simulation result of a dynamic ESC with a plant model containing second-order transfer function with zero when PRBS varies

No.	Controller gain(K_i)	PRBS calculation steps	window size(l)	Convergence iterations	Transient iterations	True extremum
1	0.005	30	$4*180$	3000	2280	O
2	0.005	60	$4*180$	2500	1780	O
3	0.005	90	$4*180$	3000	2280	offset*
4	0.005	180	$4*180$	2500	1780	offset*

* It converged to the optimum, but in the control input, there was a small offset

According to Table A10, it also indicates that the PRBS calculation steps are not influential on the iteration number to converge.

Figure A53 and A54 shows the result when PRBS is recalculated every 30 and 60 steps respectively. Both of them converges well to the maximum point, and a small numerical peak happens around 4000 iterations in Figure A54, which is not seriously problematic.

However, with PRBS calculation steps of 90 and 180 provided in Figure A55 and A56, both of them easily deviate from the true extremum. Therefore, the PRBS recalculation step is set to be 30.

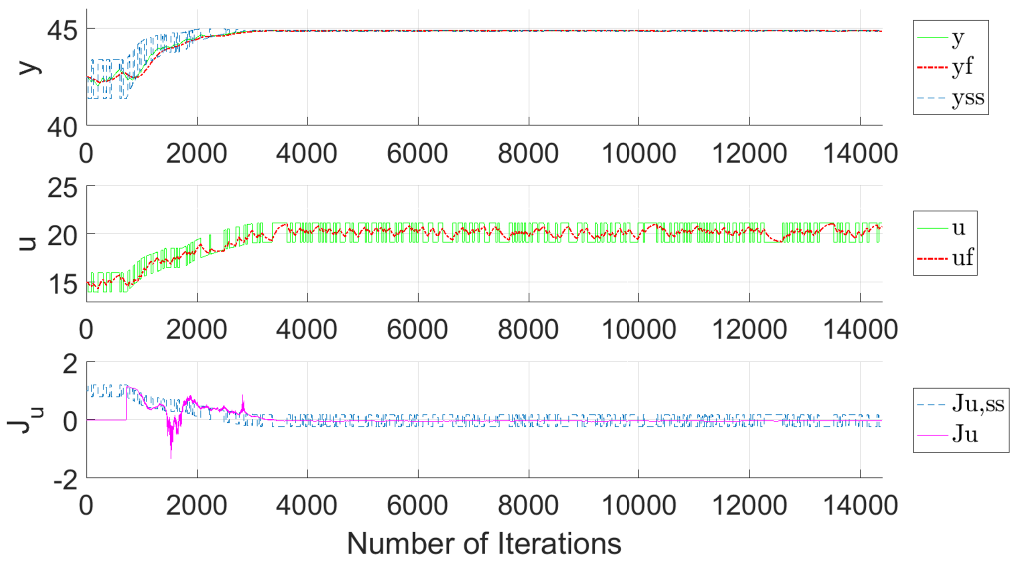

Figure A53: Simulation result for second-order with zero($\tau_a = 40$) system using a dynamic ESC: $K_i = 0.005$, PRBS= 30, and $l = 720$

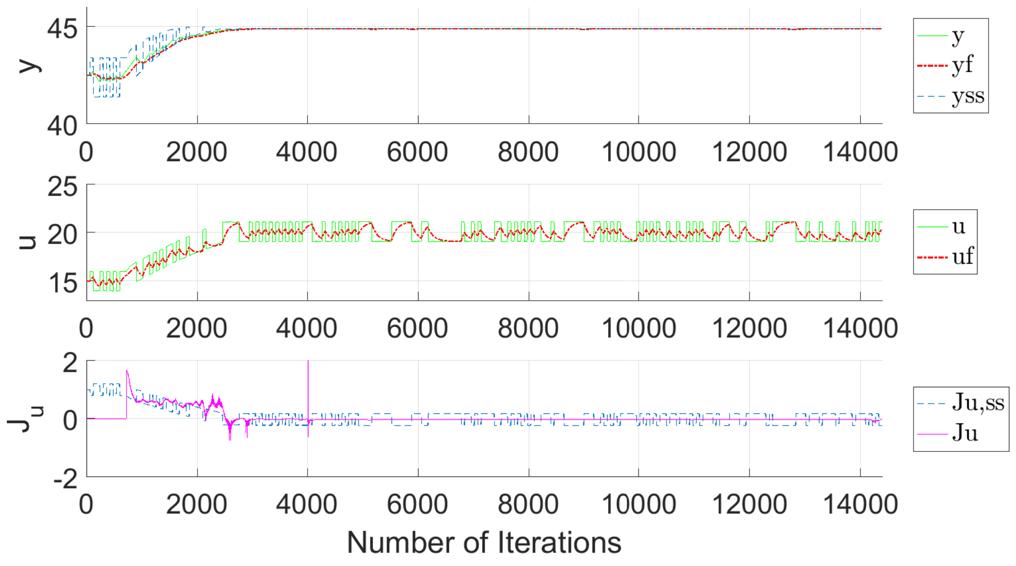

Figure A54: Simulation result for second-order with zero($\tau_a = 40$) system using a dynamic ESC: $K_i = 0.005$, PRBS= 60, and $l = 720$

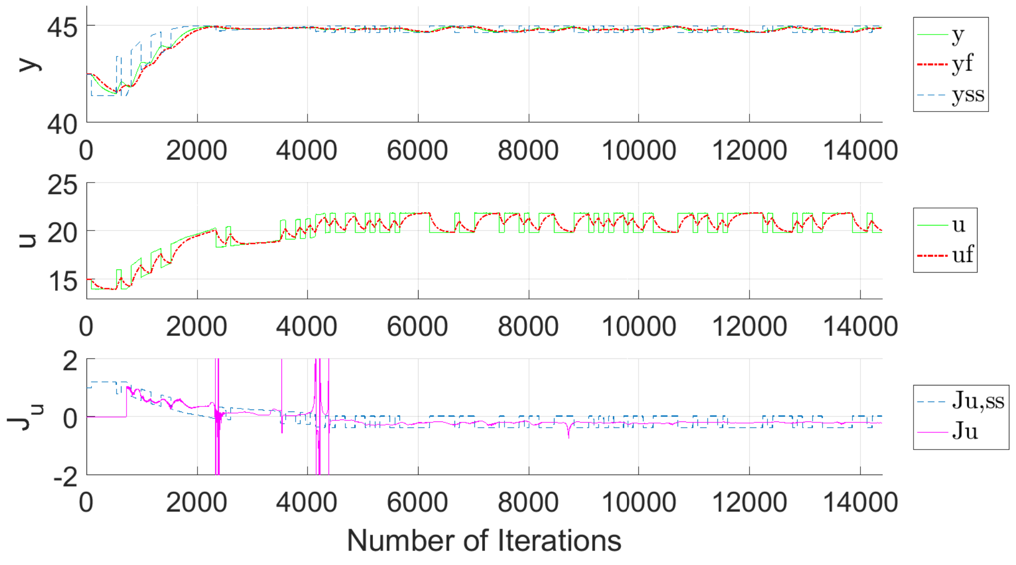

Figure A55: Simulation result for second-order with zero($\tau_a = 40$) system using a dynamic ESC: $K_i = 0.005$, PRBS= 90, and $l = 720$

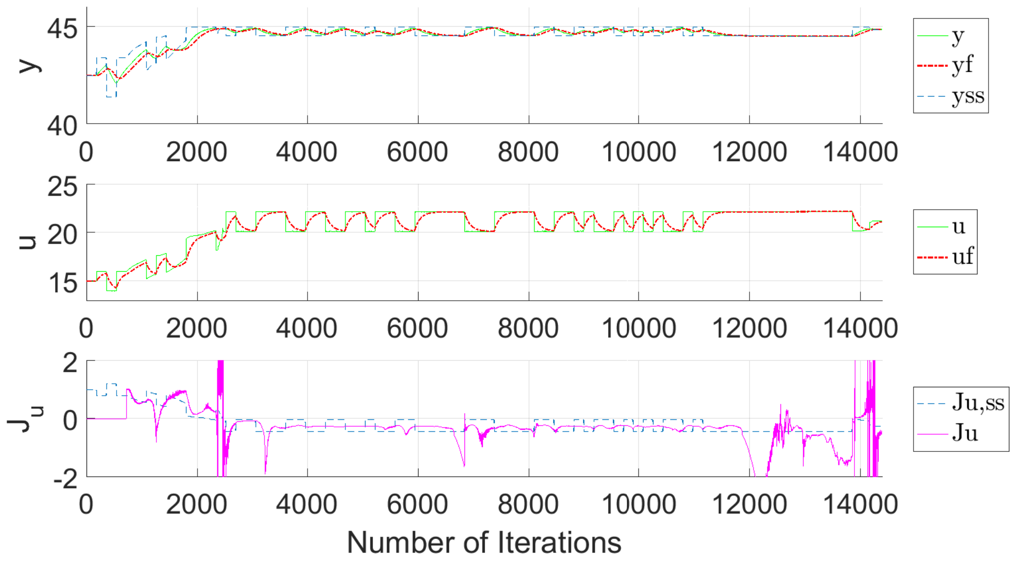

Figure A56: Simulation result for second-order with zero($\tau_a = 40$) system using a dynamic ESC: $K_i = 0.005$, PRBS= 180, and $l = 720$

A.3.2 Changes in controller gain

Table A11 states the results with different controller gains in a plant model with the second-order system containing zero.

Table A11: Simulation result of a dynamic ESC with a plant model containing second-order transfer function with zero when K_i varies

No.	Controller gain(K_i)	PRBS calculation steps	window size(l)	Convergence iterations	Transient iterations	True extremum
1	0.001	30	$4 * 180$	13000	12280	O
2	0.002	30	$4 * 180$	7500	6780	O
3	0.003	30	$4 * 180$	6000	5280	O
4	0.004	30	$4 * 180$	5000	4280	O
5	0.005	30	$4 * 180$	3500	2780	O
6	0.006	30	$4 * 180$	2500	1780	offset*
7	0.007	30	$4 * 180$	2000	1480	offset*
8	0.008	30	$4 * 180$	1500	780	offset*

* It converged to the optimum, but in the control input, there was a small offset

For small K_i, the results are almost stable with slow dynamics so that they do not deviate from the maximum. Figure A57 shows the result obtained when $K_i = 0.005$ which converges around 3500 iterations with only small numerical spikes.

When the controller gain is increased to 0.006 as Figure A58, relatively large numerical peaks happen and u and y pass the extremum. This tendency is detected in simulations with larger controller gains 0.007 and 0.008 as well.

Based on the simulation results, $K_i = 0.005$ is chosen for next simulations in this second-order with zero model as well.

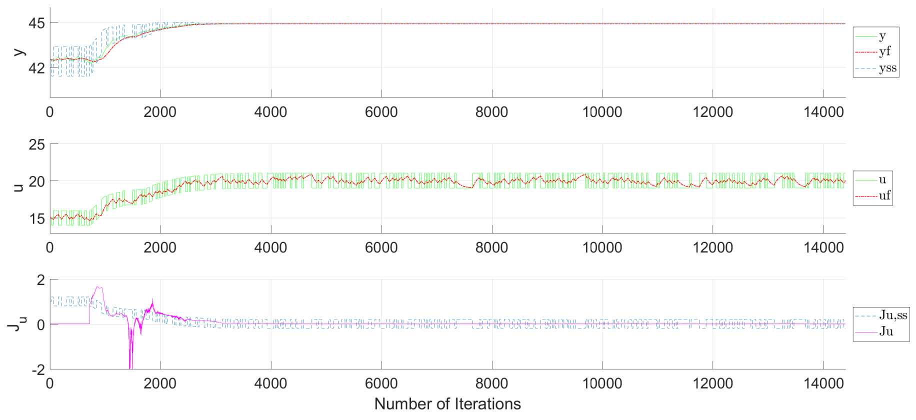

Figure A57: Simulation result for second-order with zero($\tau_a = 40$) system using a dynamic ESC: $K_i = 0.005$, PRBS$= 30$, and $l = 720$

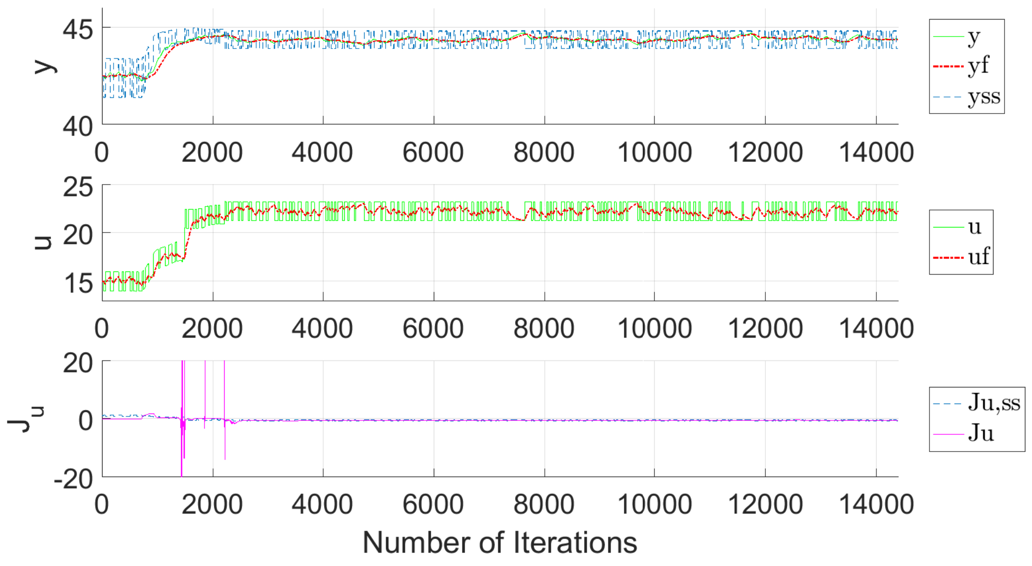

Figure A58: Simulation result for second-order with zero($\tau_a = 40$) system using a dynamic ESC: $K_i = 0.006$, PRBS$= 30$, and $l = 720$

A.3.3 Changes in window size

Table A12 indicates simulation results when the window size varies.

Table A12: Simulation result of a dynamic ESC with a plant model containing second-order transfer function with zero when window size varies

No.	Controller gain(K_i)	PRBS calculation steps	window size(l)	Convergence iterations	Transient iterations	True extremum
1	0.005	30	$1 * 180$	3000	2820	O
2	0.005	30	$2 * 180$	3000	2640	O
3	0.005	30	$3 * 180$	3500	2960	O
4	0.005	30	$4 * 180$	3500	2780	O
5	0.005	30	$5 * 180$	3000	2100	O

Figure A59 to A63 depicts the simulations number $1 - 5$. In this second-order with zero system, there is a clear tendency that increasing the window size reduces a noisy behavior in its early stage and numerical spikes in the ARX model. Based on the figures, the best result is chosen as Figure A62.

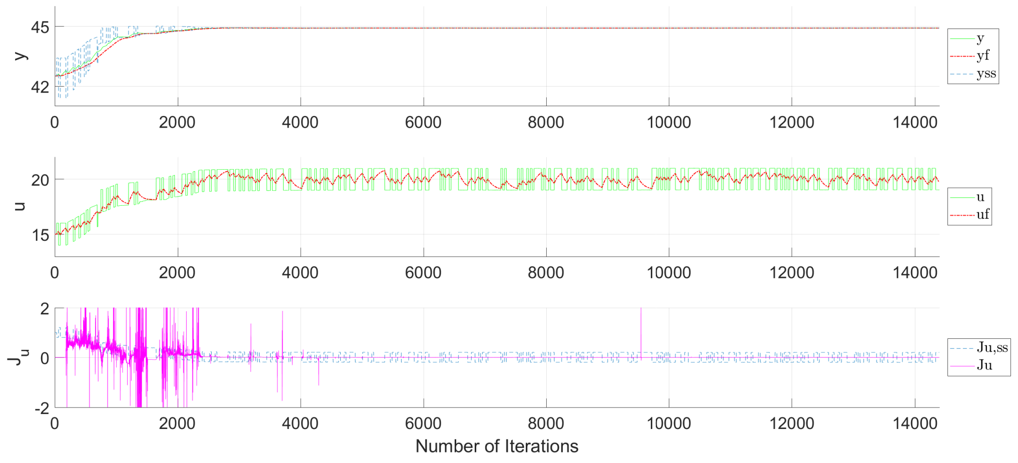

Figure A59: Simulation result for second-order with zero($\tau_a = 40$) system using a dynamic ESC: $K_i = 0.005$, PRBS= 30, and $l = 180$

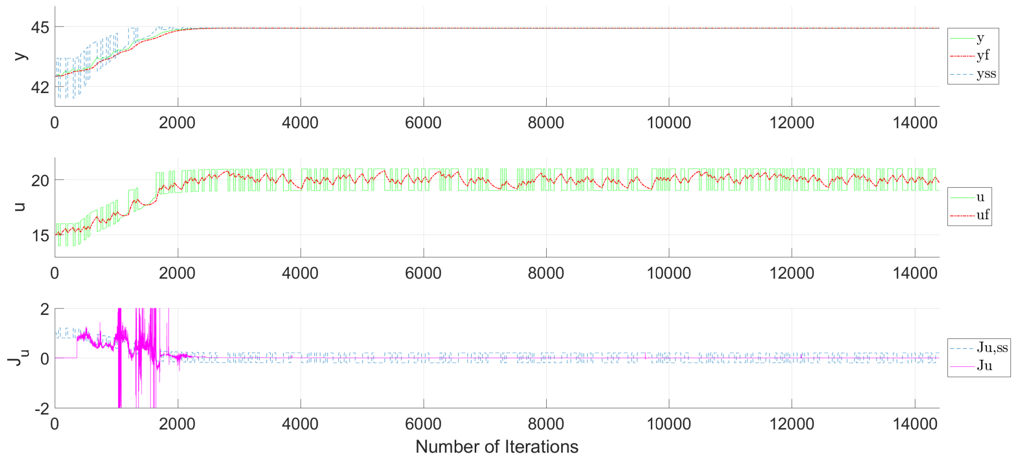

Figure A60: Simulation result for second-order with zero($\tau_a = 40$) system using a dynamic ESC: $K_i = 0.005$, PRBS= 30, and $l = 360$

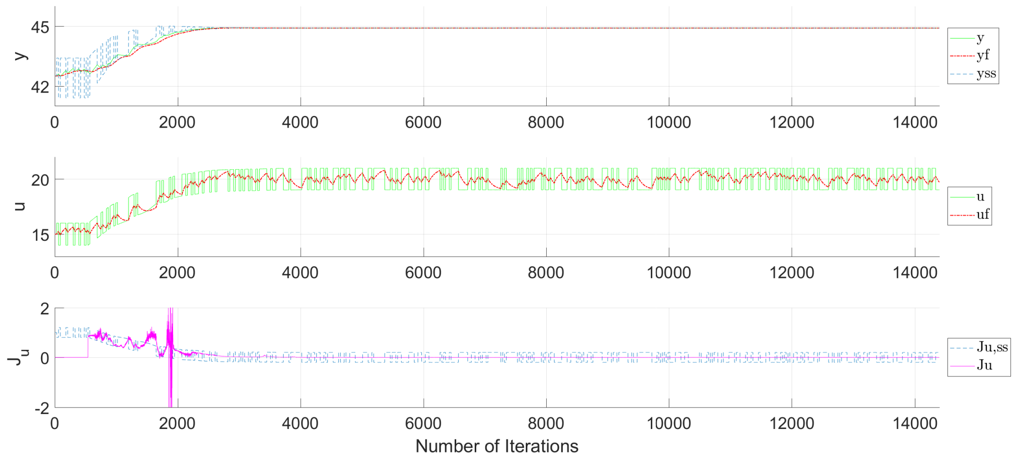

Figure A61: Simulation result for second-order with zero($\tau_a = 40$) system using a dynamic ESC: $K_i = 0.005$, PRBS= 30, and $l = 540$

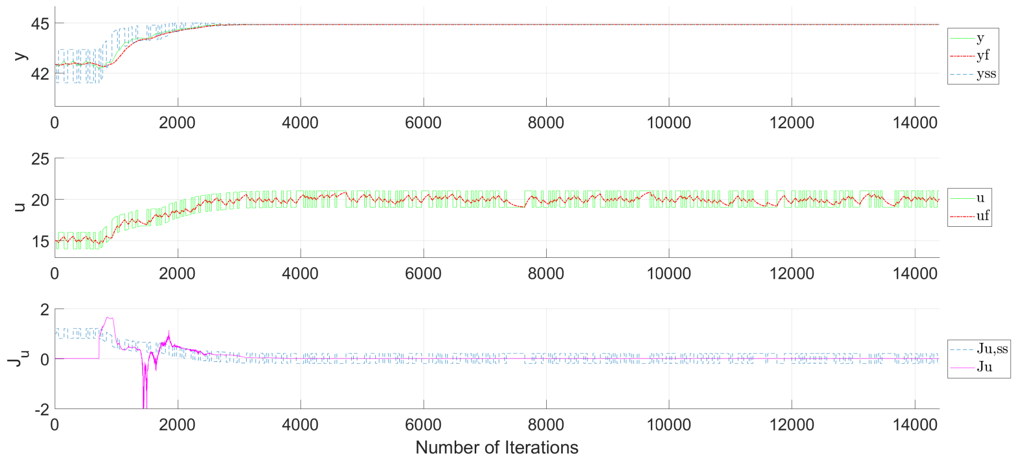

Figure A62: Simulation result for second-order with zero($\tau_a = 40$) system using a dynamic ESC: $K_i = 0.005$, PRBS= 30, and $l = 720$

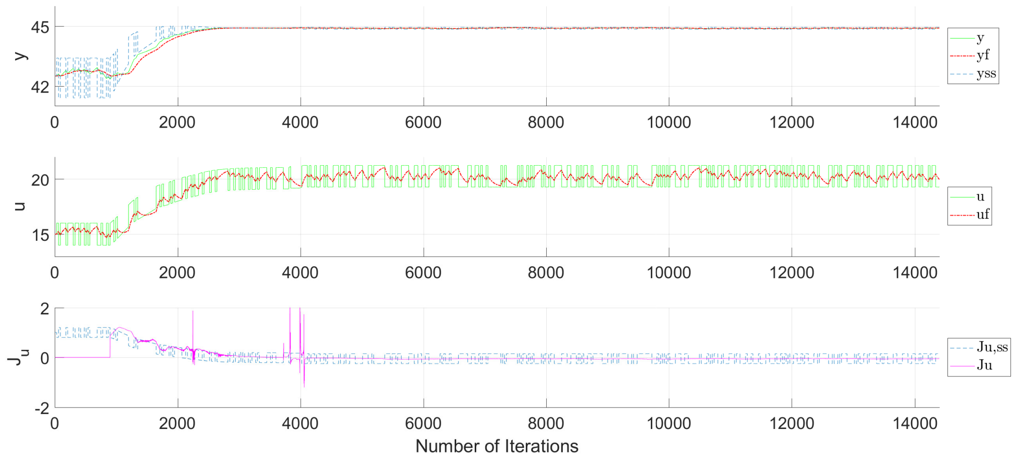

Figure A63: Simulation result for second-order with zero($\tau_a = 40$) system using a dynamic ESC: $K_i = 0.005$, PRBS= 30, and $l = 900$

A.3.4 Numerical spikes modification

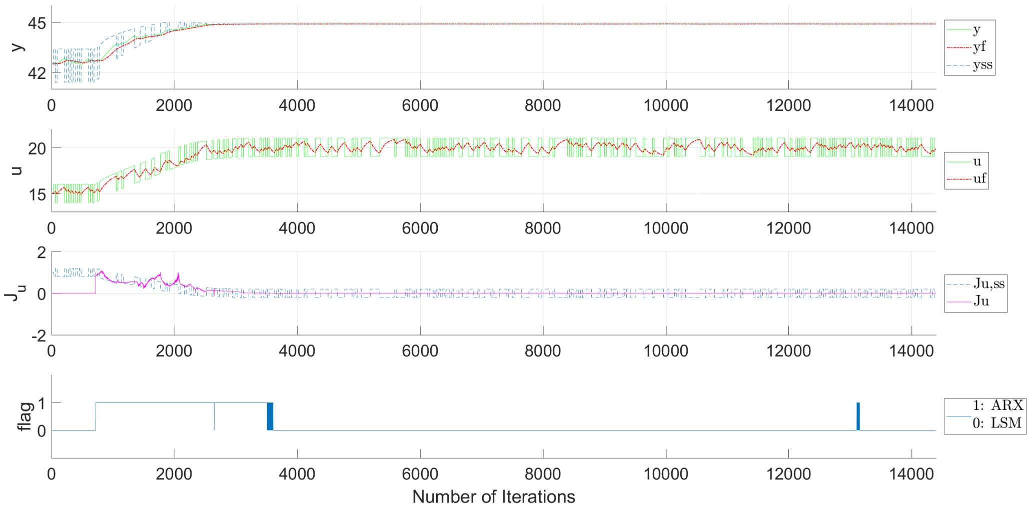

Figure A64: Modified dynamic ESC simulation result for second-order with zero($\tau_a = 40$) system: $K_i = 0.005$, PRBS= 30, and $l = 720$ with a tolerance of 0.001.

Although the numerical spikes problem is not serious in this second-order with zero system, the modification approach is also implemented. Figure A64 shows the result that the same method can be applied in this system as well to prohibit unwanted numerical problems.

Appendix B Classic ESC tuning

In the classic ESC, there are mainly two tuning parameters, the controller gain K_i and a dither time period T_D. Specifically, the dither time period is located in the middle of a high pass filter cut-off time period T_h and a low pass filter cut-off time period T_l which are set to be $10T$ and $0.2T$ respectively. To tune the parameters, simulations with different K_i values are studied first.

B.1 Plant process with first-order transfer function

The best tuned result is obtained with $K_i = 0.001$ and $T_D = 600$ which corresponds to Figure B70

B.1.1 Changes in controller gain

Table B13: Simulation result of a classic ESC with a plant model containing first-order transfer function when K_i varies

No.	Controller gain(K_i)	Dither time constant(T_D)	$T_h = 0.2T$	$T_l = 10T$	Convergence iterations
1	0.0005	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
2	0.0006	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
3	0.0007	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
4	0.0008	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
5	0.0009	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
6	0.0010	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
7	0.0020	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
8	0.0030	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
9	0.0040	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
10	0.0050	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$

Figure B65 is indicating the simulation number 1 with $K_i = 0.0005$ and $T_D = 1800$. With these tuning parameters, there is a small overshoot in the control input u. With the increased controller gain, the dynamic becomes faster making larger initial slope of u and y, but overshoot becomes larger as shown in Figure B66, B67 and B68. Moreover, they show more oscillatory behavior.

In specific, in Figure B66, the y value is oscillating at least around its extremum, but in Figure B67 and B68, the y value oscillates more like the u value. Thus, the controller gain is set to be 0.001.

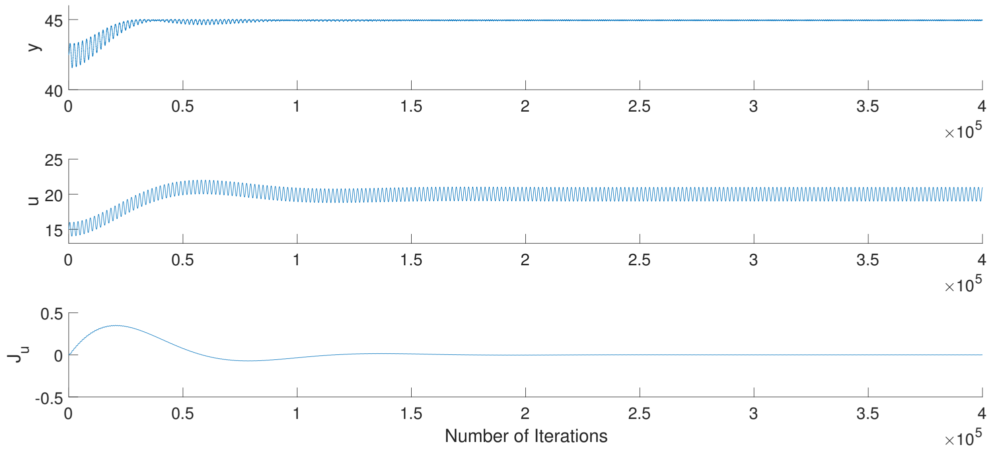

Figure B65: Simulation result for first-order system using a classic ESC: $K_i = 0.0005$ and $T_D = 1800$

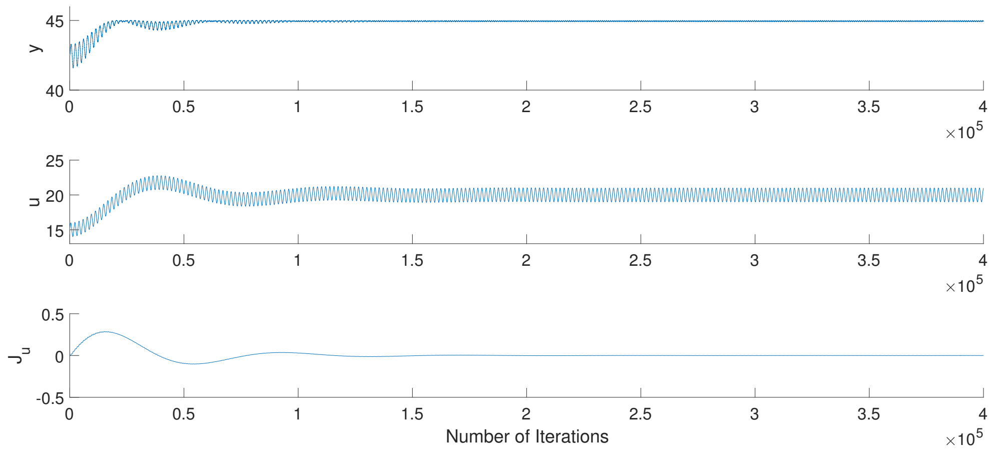

Figure B66: Simulation result for first-order system using a classic ESC: $K_i = 0.0010$ and $T_D = 1800$

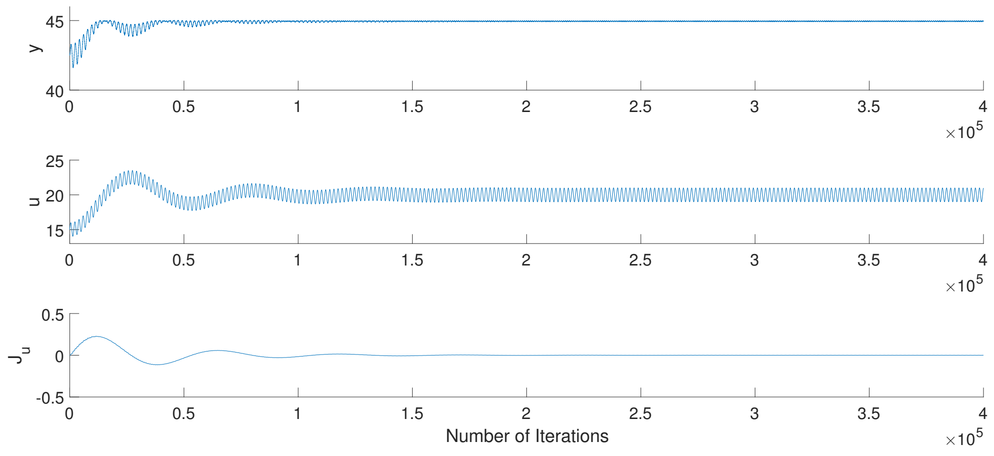

Figure B67: Simulation result for first-order system using a classic ESC: $K_i = 0.0020$ and $T_D = 1800$

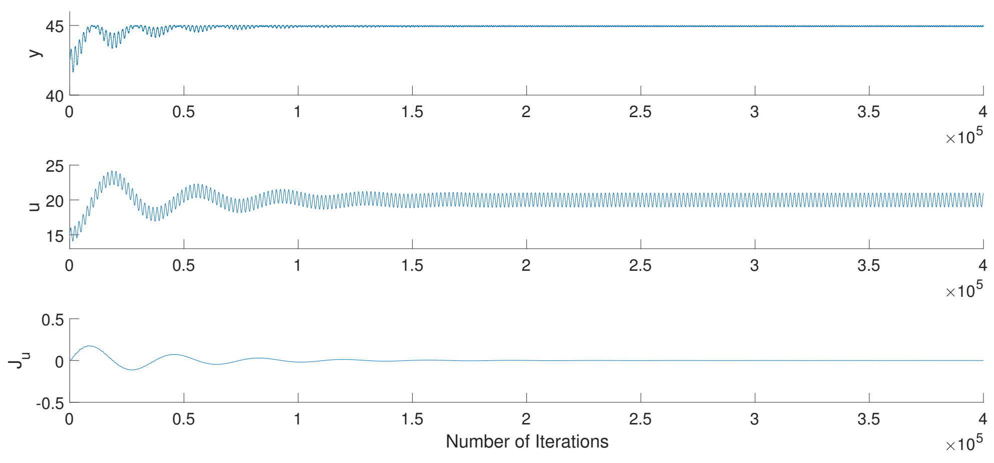

Figure B68: Simulation result for first-order system using a classic ESC: $K_i = 0.0040$ and $T_D = 1800$

B.1.2 Changes in dither time constant

Table B14: Simulation result of a classic ESC with a plant model containing first-order transfer function when T_D varies

No.	Controller gain(K_i)	Dither time constant(T_D)	$T_h = 0.2T$	$T_l = 10T$	Convergence iterations
1	0.0010	200	40	2000	$1.5 \cdot 10^5$
2	0.0010	400	80	4000	$1.0 \cdot 10^5$
3	0.0010	600	120	6000	$0.5 \cdot 10^5$
4	0.0010	800	160	8000	$1.0 \cdot 10^5$
5	0.0010	1000	200	10000	$1.0 \cdot 10^5$
6	0.0010	1200	240	12000	$1.5 \cdot 10^5$
7	0.0010	1400	280	14000	$1.0 \cdot 10^5$
8	0.0010	1600	320	16000	$1.5 \cdot 10^5$
9	0.0010	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$

Table B14 show simulation results with different T values. From the table, one can check that $T_D = 600$ gives the smallest iteration number to converge.

Simulation number 2, 3, 4, and 5 in Table B14 are provided as figures below. With a small dither time constant value, there is no oscillatory behavior as shown in Figure B69, but the dynamic with a smaller dither time constant is relatively slow requiring a large number of iterations. In contrast, the increased dither time constant value results in oscillations as Figure B71 and B72 resulting in a large iteration number. Therefore, T_D is set to be 600 where the smallest iteration is needed without oscillation.

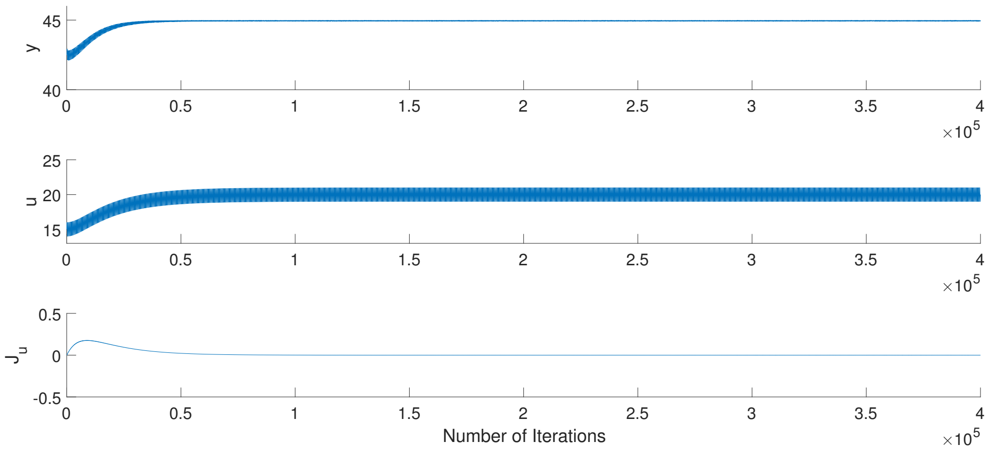

Figure B69: Simulation result for first-order system using a classic ESC: $K_i = 0.0010$ and $T_D = 400$

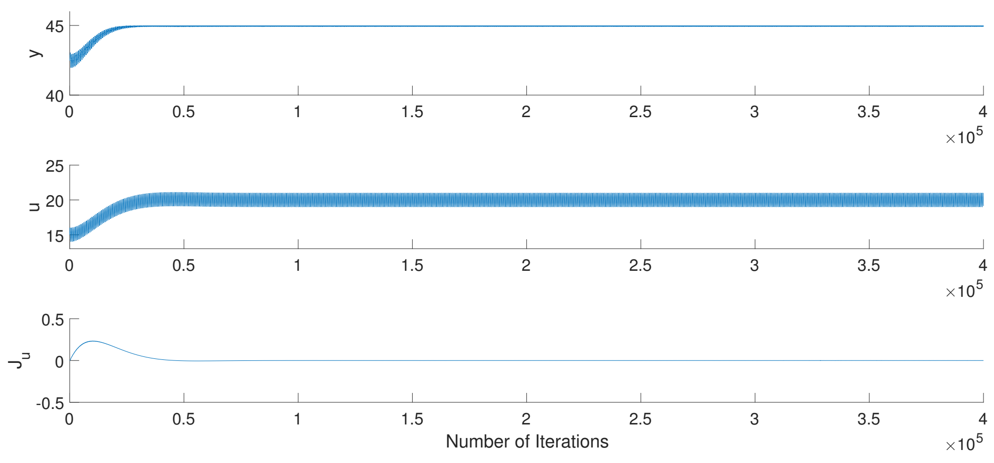

Figure B70: Simulation result for first-order system using a classic ESC: $K_i = 0.0010$ and $T_D = 600$

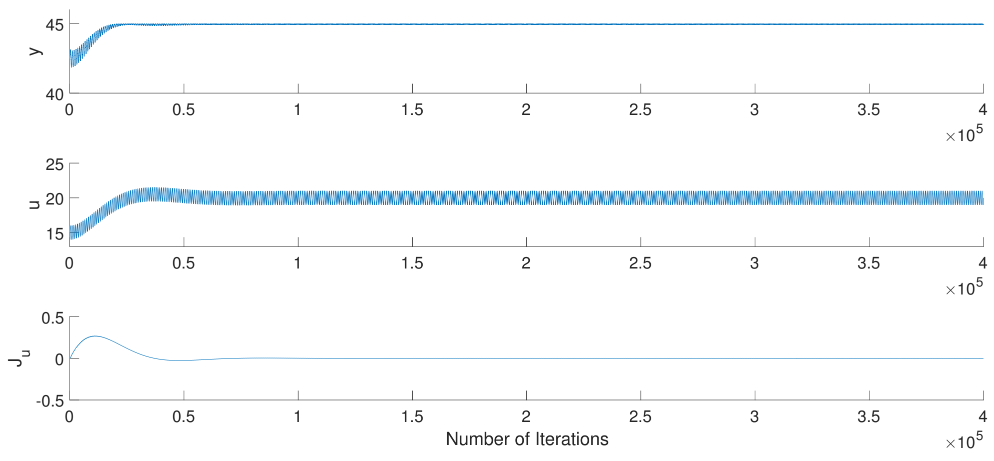

Figure B71: Simulation result for first-order system using a classic ESC: $K_i = 0.0010$ and $T_D = 800$

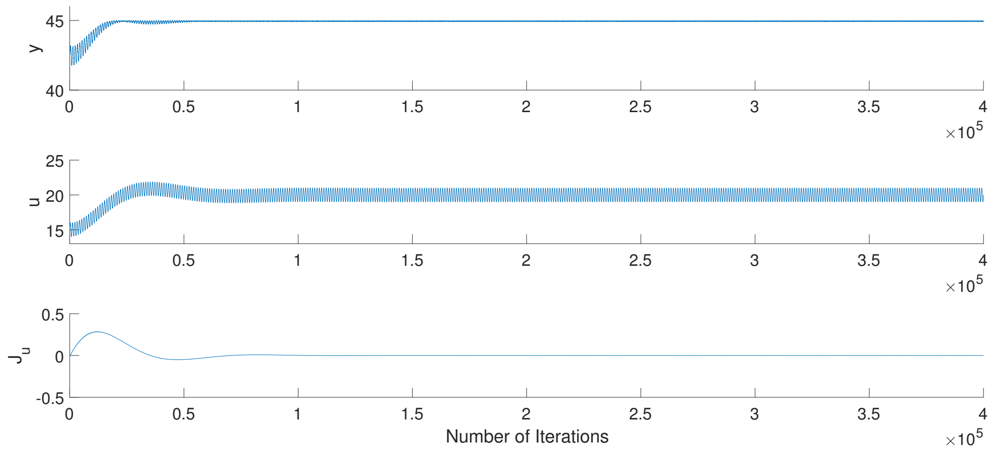

Figure B72: Simulation result for first-order system using a classic ESC: $K_i = 0.0010$ and $T_D = 1000$

B.2 Plant process with second-order transfer function without zero($\tau_a = 0$)

The best tuned result is obtained with $K_i = 0.001$ and $T_D = 800$ which corresponds to Figure B79

B.2.1 Changes in controller gain

The simulation result in this second-order without zero system is more or less similar to that of a first-order system. Figure B73 to B76 shows overshoots in the estimation of u as well. Additionally, simulations with larger K_i values are fast initially but the oscillatory behavior becomes serious as well. Thus, the overall iteration numbers to converge are similar. Based on these facts, $K_i = 0.0010$ is chosen as well which has a fast dynamic and less an oscillatory behavior.

Table B15: Simulation result of a classic ESC with a plant model containing second-order transfer function without zero when K_i varies

No.	Controller gain(K_i)	Dither time constant(T_D)	$T_h = 0.2T$	$T_l = 10T$	Convergence iterations
1	0.0005	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
2	0.0006	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
3	0.0007	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
4	0.0008	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
5	0.0009	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
6	0.0010	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
7	0.0020	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
8	0.0030	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
9	0.0040	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
10	0.0050	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$

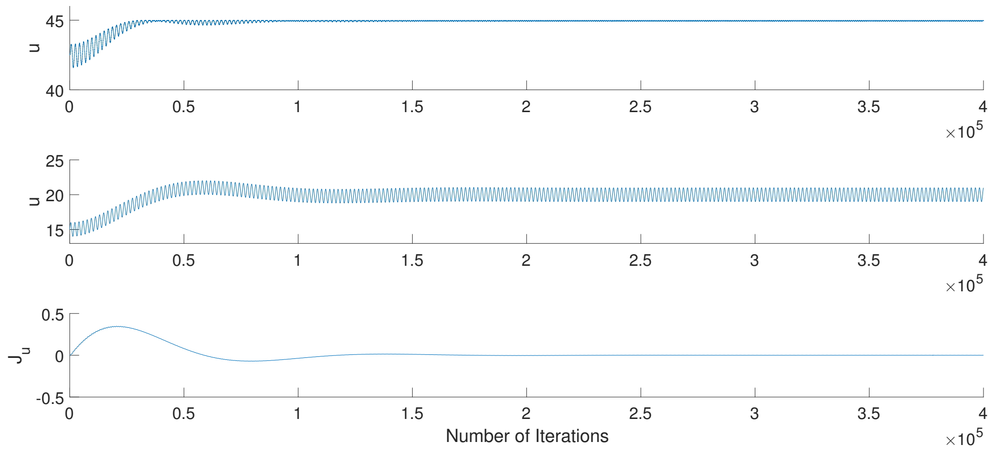

Figure B73: Simulation result for second-order without zero($\tau_a = 0$) system using a classic ESC: $K_i = 0.0005$ and $T_D = 1800$

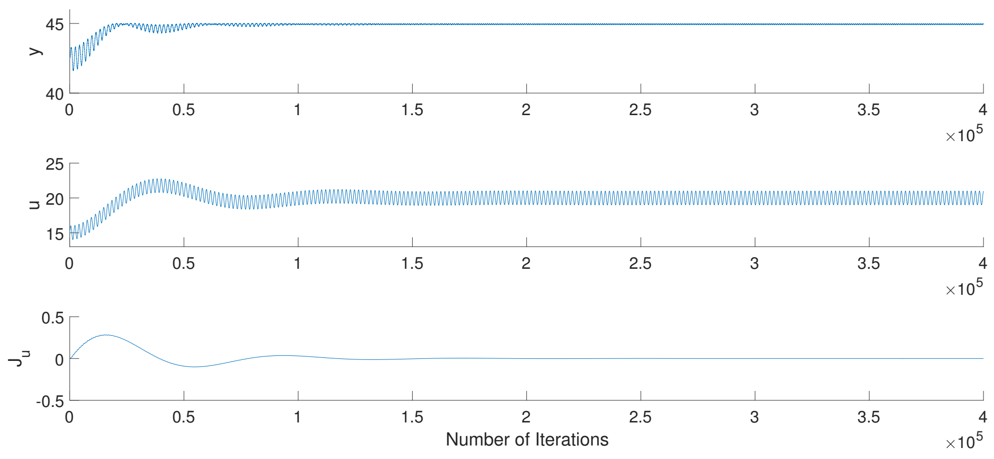

Figure B74: Simulation result for second-order without zero($\tau_a = 0$) system using a classic ESC: $K_i = 0.0010$ and $T_D = 1800$

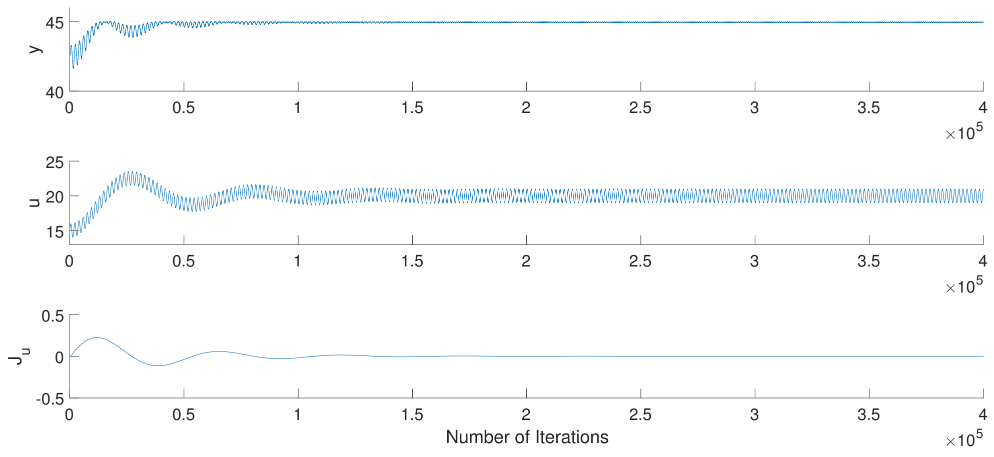

Figure B75: Simulation result for second-order without zero($\tau_a = 0$) system using a classic ESC: $K_i = 0.0020$ and $T_D = 1800$

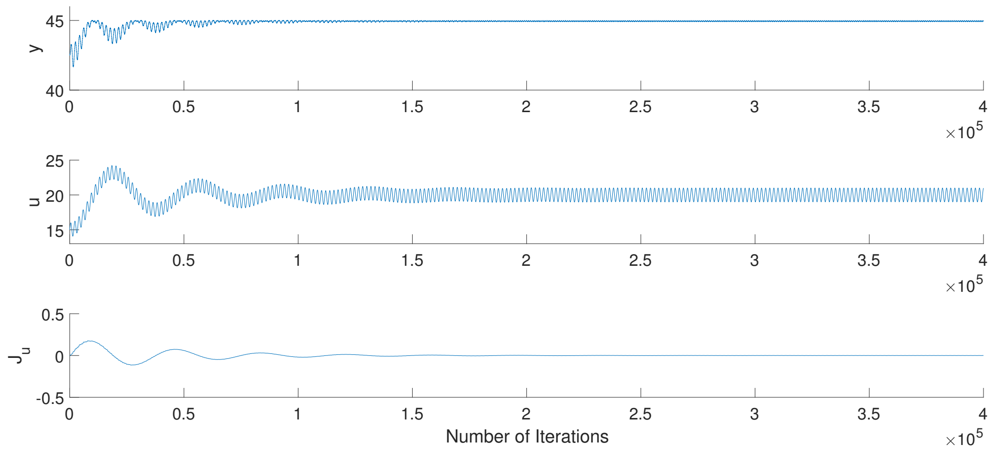

Figure B76: Simulation result for second-order without zero($\tau_a = 0$) system using a classic ESC: $K_i = 0.0040$ and $T_D = 1800$

B.2.2 Changes in dither time constant

Table B16: Simulation result of a classic ESC with a plant model containing second-order transfer function without zero when T_D varies

No.	Controller gain(K_i)	Dither time constant(T_D)	$T_h = 0.2T$	$T_l = 10T$	Convergence iterations
1	0.0010	400	80	4000	$3.5 \cdot 10^5$
2	0.0010	600	120	6000	$1.0 \cdot 10^5$
3	0.0010	800	160	8000	$0.5 \cdot 10^5$
4	0.0010	1000	200	10000	$1.0 \cdot 10^5$
5	0.0010	1200	240	12000	$1.5 \cdot 10^5$
6	0.0010	1400	280	14000	$1.5 \cdot 10^5$
7	0.0010	1600	320	16000	$1.5 \cdot 10^5$
8	0.0010	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$

Similarly with the previous tuning processes, Figure B77 to B80 concludes that

$T = 800$ provides the smallest iterations to converge. In short, as the first-order system, small T values has a slow dynamic while large T values have an oscillatory behavior, and both of them requires a large number of iterations.

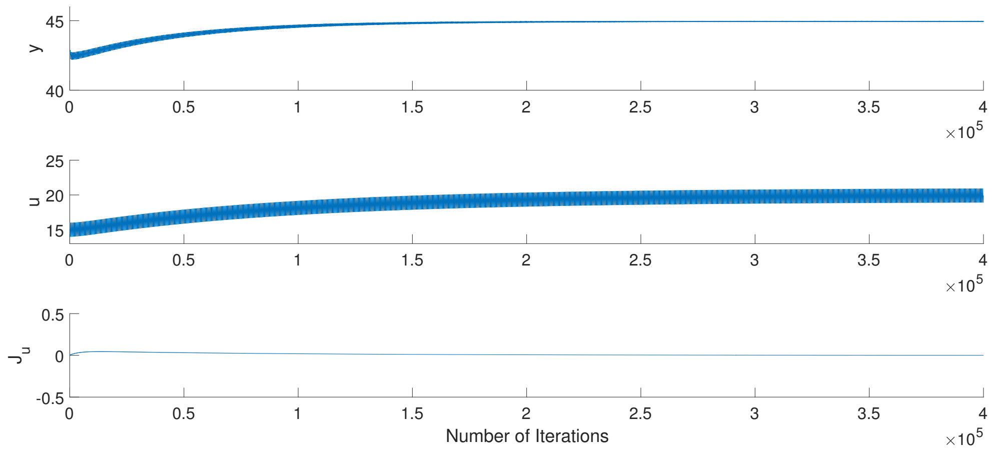

Figure B77: Simulation result for second-order without zero($\tau_a = 0$) system using a classic ESC: $K_i = 0.0010$ and $T_D = 400$

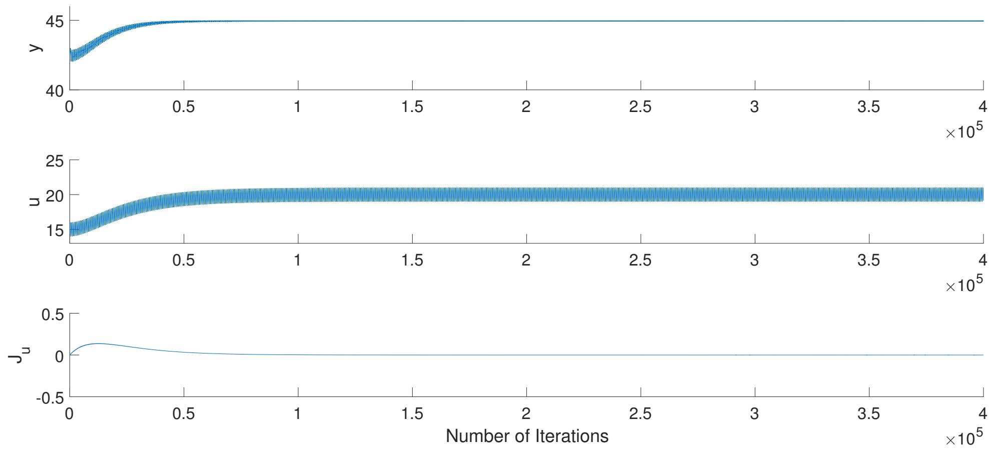

Figure B78: Simulation result for second-order without zero($\tau_a = 0$) system using a classic ESC: $K_i = 0.0010$ and $T_D = 600$

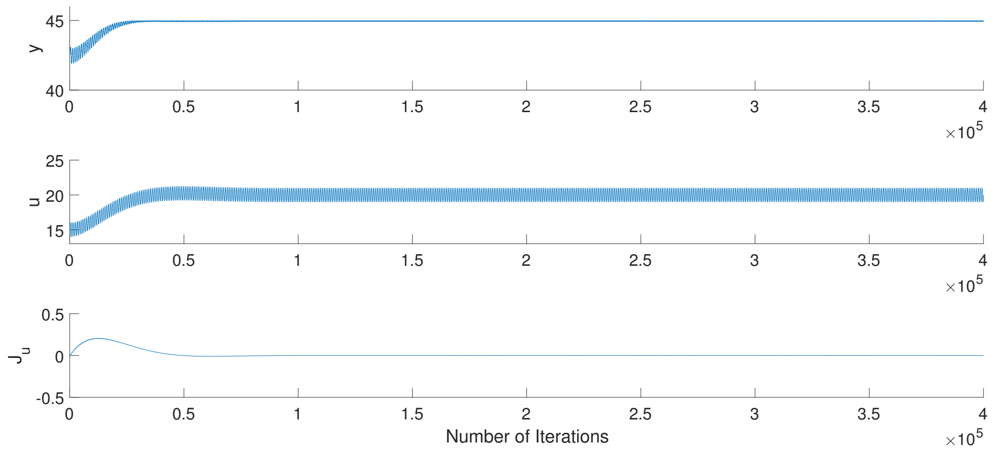

Figure B79: Simulation result for second-order without zero($\tau_a = 0$) system using a classic ESC: $K_i = 0.0010$ and $T_D = 800$

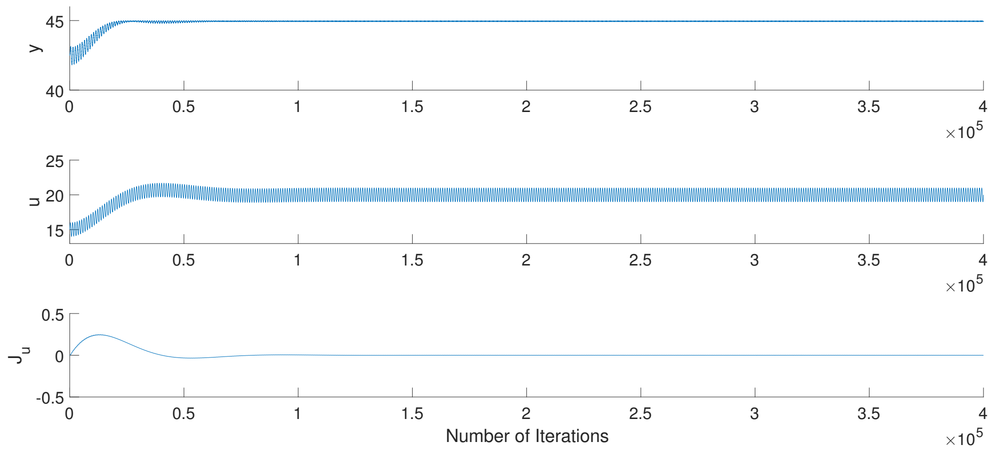

Figure B80: Simulation result for second-order without zero($\tau_a = 0$) system using a classic ESC: $K_i = 0.0010$ and $T_D = 1000$

B.3 Plant process with second-order transfer function with zero($\tau_a = 40$)

The best tuned result is obtained with $K_i = 0.001$ and $T_D = 600$ which corresponds to Figure B86

B.3.1 Changes in controller gain

Table B17: Simulation result of a classic ESC with a plant model containing second-order transfer function with zero when K_i varies

No.	Controller gain(K_i)	Dither time constant(T_D)	$T_h = 0.2T$	$T_l = 10T$	Convergence iterations
1	0.0005	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
2	0.0006	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
3	0.0007	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
4	0.0008	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
5	0.0009	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
6	0.0010	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
7	0.0020	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
8	0.0030	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
9	0.0040	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
10	0.0050	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$

Simulation results for 1, 6, 7, and 9 in Table B17 are provided in Figure B81 to B84. Because the result becomes much oscillatory with large controller gain, the observation on these figures results in a K_i set to be 0.0010 as well.

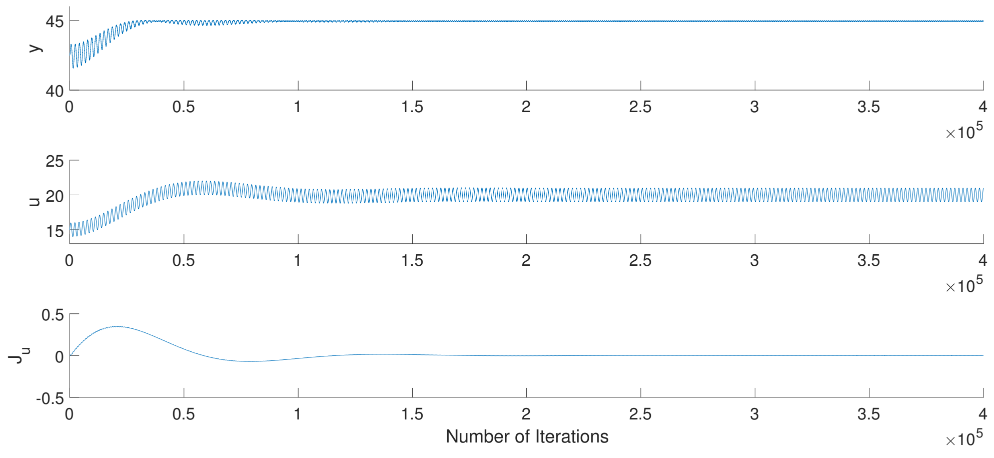

Figure B81: Simulation result for second-order with zero($\tau_a = 40$) system using a classic ESC: $K_i = 0.0005$ and $T_D = 1800$

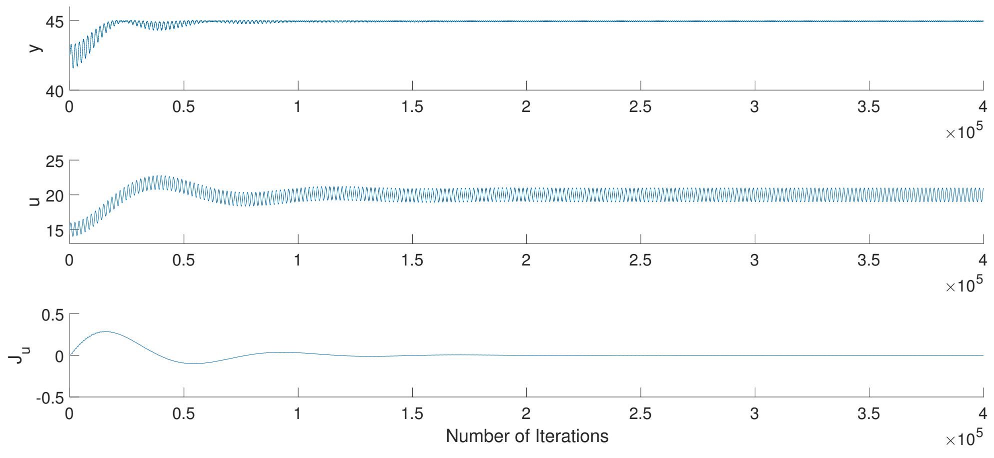

Figure B82: Simulation result for second-order with zero($\tau_a = 40$) system using a classic ESC: $K_i = 0.0010$ and $T_D = 1800$

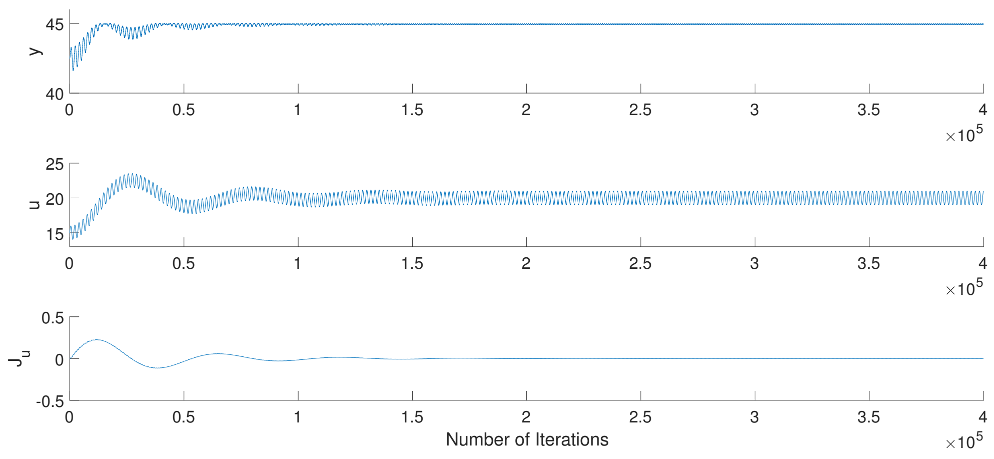

Figure B83: Simulation result for second-order with zero($\tau_a = 40$) system using a classic ESC: $K_i = 0.0020$ and $T_D = 1800$

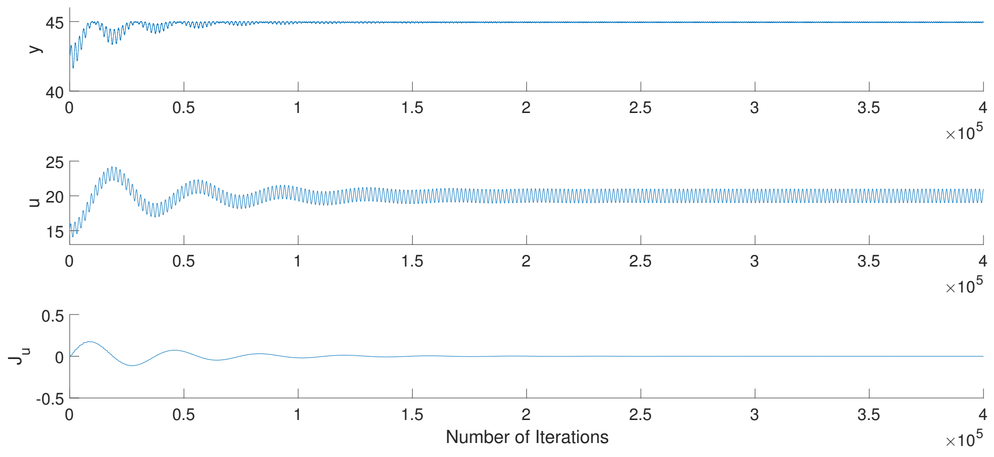

Figure B84: Simulation result for second-order with zero($\tau_a = 40$) system using a classic ESC: $K_i = 0.0040$ and $T_D = 1800$

Table B18: Simulation result of a classic ESC with a plant model containing second-order transfer function with zero when T_D varies

No.	Controller gain(K_i)	Dither time constant(T_D)	$T_h = 0.2T$	$T_l = 10T$	Convergence iterations
1	0.0010	400	80	4000	$1.0 \cdot 10^5$
2	0.0010	600	120	6000	$0.5 \cdot 10^5$
3	0.0010	800	160	8000	$1.0 \cdot 10^5$
4	0.0010	1000	200	10000	$1.0 \cdot 10^5$
5	0.0010	1200	240	12000	$1.5 \cdot 10^5$
6	0.0010	1400	280	14000	$1.5 \cdot 10^5$
7	0.0010	1600	320	16000	$1.5 \cdot 10^5 - 2 \cdot 10^5$
8	0.0010	1800	360	18000	$1.5 \cdot 10^5 - 2 \cdot 10^5$

B.3.2 Changes in dither time constant

In this case, $T = 600$ provides the smallest number of iterations to converge. Different simulation results are illustrated in Figure B85 to B88 verifying that $T = 600$ gives smooth non-oscillatory behavior and converges faster.

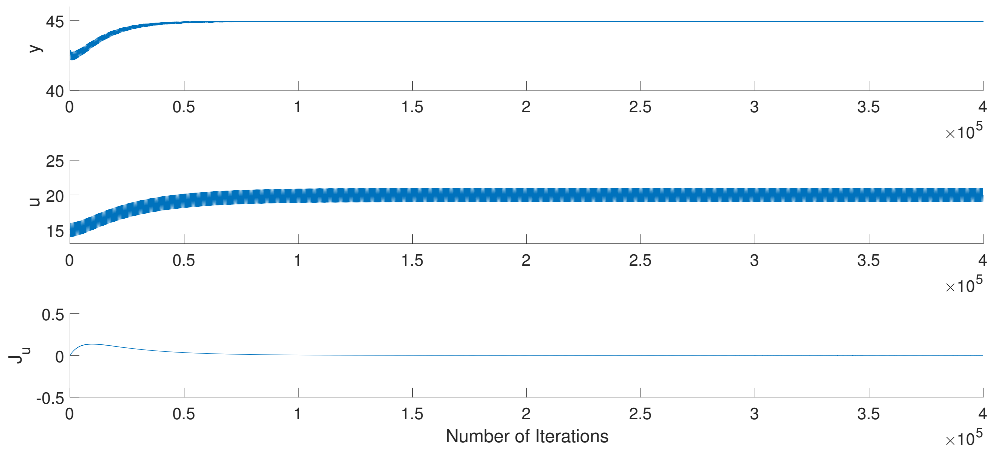

Figure B85: Simulation result for second-order with zero($\tau_a = 40$) system using a classic ESC: $K_i = 0.0010$ and $T_D = 400$

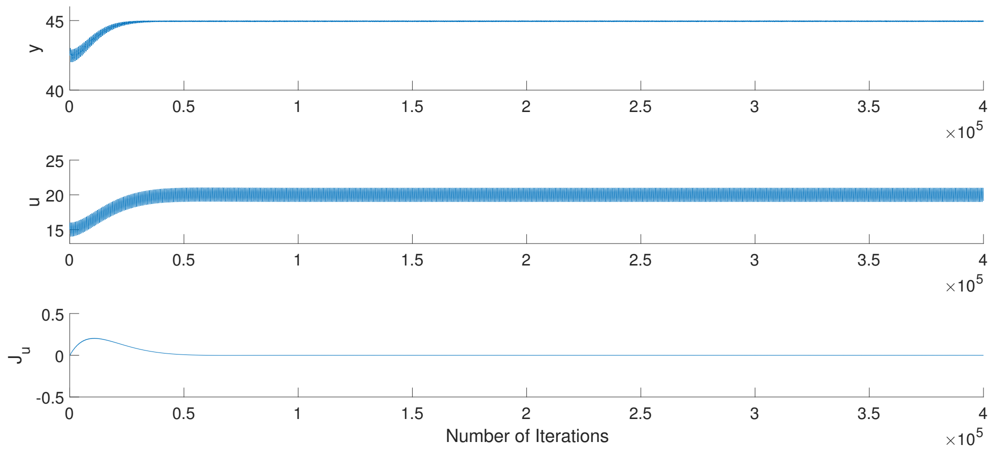

Figure B86: Simulation result for second-order with zero($\tau_a = 40$) system using a classic ESC: $K_i = 0.0010$ and $T_D = 600$

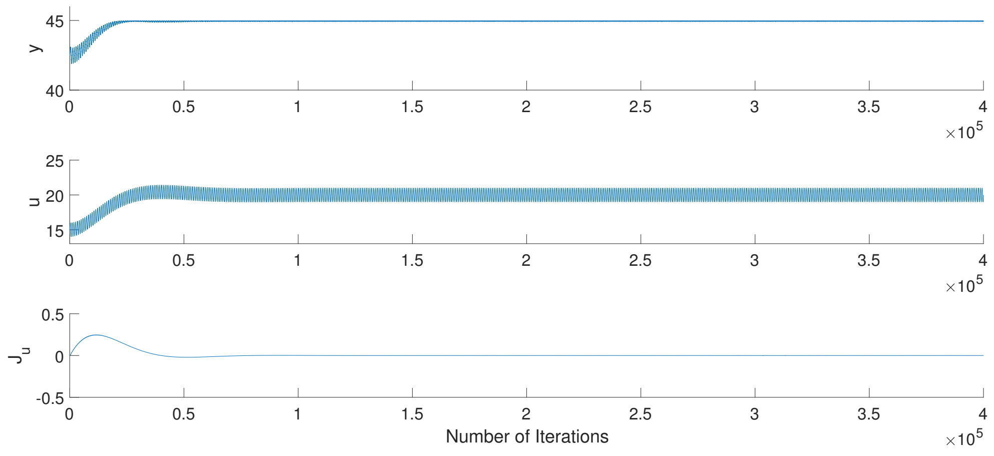

Figure B87: Simulation result for second-order with zero($\tau_a = 40$) system using a classic ESC: $K_i = 0.0010$ and $T_D = 800$

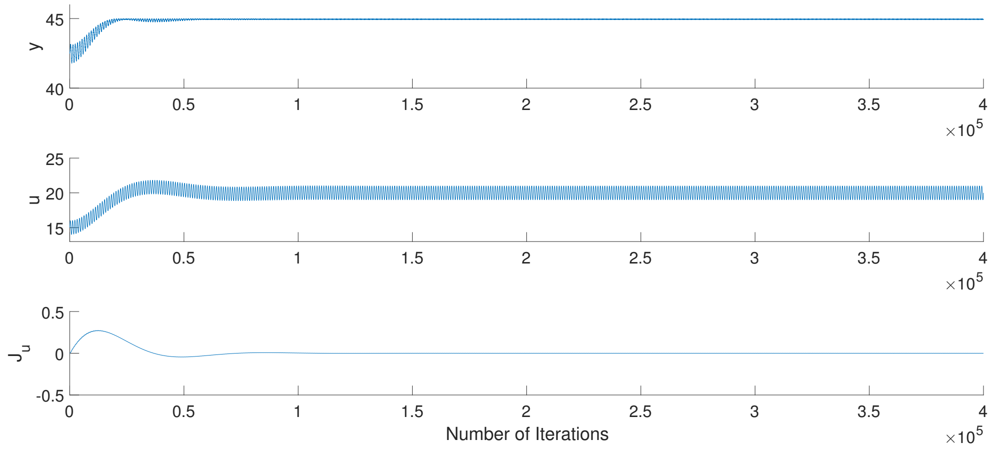

Figure B88: Simulation result for second-order with zero($\tau_a = 40$) system using a classic ESC: $K_i = 0.0010$ and $T_D = 1000$

Appendix C Matlab Script

C.1 Dynamic ESC

C.1.1 Plant process with first-order transfer function

```
clear
clc

addpath ('D:\matlab\casadi-matlabR2014b-v3.2.3')
addpath('D:\matlab')
import casadi.*

Ts = 1; % Sample time
y = MX.sym('y');
u = MX.sym('u');
tau = (174/Ts);

dx1 = ((-0.1*u^2+4*u+5) - y)/tau;

ode = struct('x',y,'p',u,'ode',dx1,'quad',y);
opts = struct('tf',Ts);

F = integrator('F','cvodes',ode,opts);

xf = 42.5;
u_in0 = 15;
u_in = u_in0;

Ki = 0.005;                     % Controller gain
TD = 1*180;                     % Dither time constant
```

```
l = 4*180;                    % Window size

nIter = 2*3600;

h = waitbar(0,'Simulation in Progress...');
aL = Ts/(Ts+ 100);
sim.uf = u_in;
sim.yf = xf;

ARX = 1;

for sim_k  = 1:nIter
    waitbar(sim_k /nIter,h,sprintf('Time: %0.0f min',sim_k*Ts/60))

    Fk = F('x0',xf,'p',u_in);
    xf = full(Fk.xf);

    sim.y(sim_k) = xf + 0.005*randn(1);
    sim.u(sim_k) = u_in + 0.005*randn(1);
    sim.ySS(sim_k) = -0.1*u_in^2+4*u_in+5;
    sim.JuSS(sim_k) = -0.2*u_in+4;
    if sim_k>1      % Noise filtering
        sim.uf(sim_k) = (1-aL)*sim.uf(sim_k-1) + aL*sim.u(sim_k);
        sim.yf(sim_k) = (1-aL)*sim.yf(sim_k-1) + aL*sim.y(sim_k);
    end
    if sim_k > l
        ymeas = sim.yf(sim_k-l:sim_k)';
        umeas = sim.uf(sim_k-l:sim_k)';
        if ARX
            Y0 = ymeas - mean(ymeas); % Zero mean
            U0 = umeas - mean(umeas); % Zero mean

            data = iddata(Y0,U0,Ts);
            sysARX = arx(data,[1,1,0]);
            sysD = idss(sysARX);
            sys = d2c(sysD);
            Ju_hat = (-sys.C*(sys.A\sys.B) + sys.D);
            Ju(sim_k) = Ju_hat;

        end
        if sim_k > l && (mean(Ju(sim_k-10:sim_k)) >0.001) % ...
            I-controller
            u_in0 =   u_in0  + Ki*Ju_hat ;
        else
            u_in0 = u_in0 ;
        end
    end
        u_in = u_in0+ 0.1*sin(2*pi*(1/(TD))*sim_k);  % add perturbation

end
close(h);

sim.Ju = Ju;

%%

figure(3411)
subplot(311)
hold all
plot(sim.y)
plot(sim.yf)
plot( sim.ySS,'--')
set(gca,'FontSize',20);
grid on
```

```
ylabel 'y'

subplot(312)
hold all
plot(sim.u)
plot(sim.uf,'--')
set(gca,'FontSize',20);
grid on
ylabel 'u'

subplot(313)
hold all
plot(sim.JuSS,'--')
plot(sim.Ju)
set(gca,'FontSize',20);
grid on
ylabel 'J_u'
```

C.1.2 Plant process with second-order transfer function

```
clear
clc

addpath ('D:\matlab\casadi-matlabR2014b-v3.2.3')
addpath('D:\matlab')
import casadi.*

Ts = 1; % Sample time
y = MX.sym('y');
u = MX.sym('u');
```

```
x1 = MX.sym('x1');
x2 = MX.sym('x2');
tau = (174/Ts);

tau_1 = 174;
tau_2 = 60;
tau_a = 40;                  % tau_a = 0 and 40 are studied

[a, b, c, d] = tf2ss([0 tau_a 1],[tau_1*tau_2 tau_1+tau_2 1]);

% dx/dt=Ax+Bu, y=Cx+Du, x=[x1 x2]', A=2X2 matrix
dx1 = a(1,1)*x1+a(1,2)*x2+b(1,1)*(-0.1*u^2+4*u+5);
dx2 = a(2,1)*x1+a(2,2)*x2+b(2,1)*(-0.1*u^2+4*u+5);
y = c(1,1)*x1+c(1,2)*x2+d*(-0.1*u^2+4*u+5);

ode = struct('x',vertcat(x1,x2),'p',u,'ode',vertcat(dx1,dx2),'quad',y);
opts = struct('tf',Ts);

F = integrator('F','cvodes',ode,opts);

xf = [0; 42.5/c(1,2)];
u_in0 = 15;

u_in = u_in0;

Ki = 0.005;                  % Controller gain
PRBS = 30;                   % PRBS calculation steps
l = 4*180;                   % Window size

nIter = 4*3600;

h = waitbar(0,'Simulation in Progress...');
```

```
aL = Ts/(Ts+ 100);
sim.uf = u_in;
sim.yf = c*xf;



ARX = 1;
pbrs = 0;

for sim_k  = 1:nIter
    waitbar(sim_k /nIter,h,sprintf('Time: %0.0f min',sim_k*Ts/60))

    Fk = F('x0',xf,'p',u_in);
    xf = full(Fk.xf);

    sim.y(sim_k) = c*xf + 0.005*randn(1);
    sim.u(sim_k) = u_in + 0.005*randn(1);
    sim.ySS(sim_k) = -0.1*u_in^2+4*u_in+5;
    sim.JuSS(sim_k) = -0.2*u_in+4;
    if sim_k>1      % Noise filtering
        sim.uf(sim_k) = (1-aL)*sim.uf(sim_k-1) + aL*sim.u(sim_k);
        sim.yf(sim_k) = (1-aL)*sim.yf(sim_k-1) + aL*sim.y(sim_k);
    end
    if sim_k > l
        ymeas = sim.yf(sim_k-l:sim_k)';
        umeas = sim.uf(sim_k-l:sim_k)';
        if ARX
            Y0 = ymeas - mean(ymeas); % Zero mean
            U0 = umeas - mean(umeas); % Zero mean


            data = iddata(Y0,U0,Ts);
            sysARX = arx(data,[2,2,0]);
            sysD = idss(sysARX);
            sys = d2c(sysD);
            Ju_hat = (-sys.C*(sys.A\sys.B) + sys.D);
            Ju(sim_k) = Ju_hat;

        end
        if sim_k > l && (mean(Ju(sim_k-10:sim_k)) >0.001)  % ...
           I-controller
            u_in0 =   u_in0  +Ki*Ju_hat ;
        else
            u_in0 = u_in0 ;
        end
    end

if rem(sim_k,PRBS)==0
    pbrs = 1*idinput(1);
end
u_in = u_in0+ pbrs;     % add perturbation
end
close(h);

sim.Ju = Ju;

%%

figure(3411)
subplot(311)
hold all
plot(sim.y)
plot(sim.yf)
plot( sim.ySS,'--')
```

```
set(gca,'FontSize',20);
grid on
ylabel 'y'

subplot(312)
hold all
plot(sim.u)
plot(sim.uf,'--')
set(gca,'FontSize',20);
grid on
ylabel 'u'

subplot(313)
hold all
plot(sim.JuSS,'--')
plot(sim.Ju)
set(gca,'FontSize',20);
grid on
ylabel 'J_u'
ylim([-2 2])
```

C.2 Modified Dynamic ESC combining ARX model and LS method

C.2.1 Plant process with first-order transfer function

```
clear
clc

```

```
addpath ('D:\matlab\casadi-matlabR2014b-v3.2.3')
addpath('D:\matlab')
import casadi.*

Ts = 1; % Sample time
y = MX.sym('y');
u = MX.sym('u');
tau = (174/Ts);

dx1 = ((-0.1*u^2+4*u+5) - y)/tau;

ode = struct('x',y,'p',u,'ode',dx1,'quad',y);
opts = struct('tf',Ts);

F = integrator('F','cvodes',ode,opts);

xf = 42.5;
u_in0 = 15;
u_in = u_in0;

Ki = 0.005;                  % Controller gain
TD = 180;                    % Dither time constant
l = 2*180;                   % Window size

nIter = 4*3600;

h = waitbar(0,'Simulation in Progress...');
aL = Ts/(Ts+ 100);
sim.uf = u_in;
sim.yf = xf;

ARX = 1;
```

```

for sim_k  = 1:nIter
    waitbar(sim_k /nIter,h,sprintf('Time: %0.0f min',sim_k*Ts/60))

    Fk = F('x0',xf,'p',u_in);
    xf = full(Fk.xf);

    sim.y(sim_k) = xf + 0.005*randn(1);
    sim.u(sim_k) = u_in + 0.005*randn(1);
    sim.ySS(sim_k) = -0.1*u_in^2+4*u_in+5;
    sim.JuSS(sim_k) = -0.2*u_in+4;
    if sim_k>1      % Noise filtering
        sim.uf(sim_k) = (1-aL)*sim.uf(sim_k-1) + aL*sim.u(sim_k);
        sim.yf(sim_k) = (1-aL)*sim.yf(sim_k-1) + aL*sim.y(sim_k);
    end
    if sim_k > l
        ymeas = sim.yf(sim_k-l:sim_k)';
        umeas = sim.uf(sim_k-l:sim_k)';

        Ju(l) = 1;  %initialization

        if Ju(sim_k-1) > 0.010    % ARX model
            Y0 = ymeas - mean(ymeas); % Zero mean
            U0 = umeas - mean(umeas); % Zero mean

            data = iddata(Y0,U0,Ts);
            sysARX = arx(data,[1,1,0]);
            sysD = idss(sysARX);
            sys = d2c(sysD);
            Ju_hat = (-sys.C*(sys.A\sys.B) + sys.D);
            Ju(sim_k) = Ju_hat;
            flag(sim_k) = 1;
        else                          % LS method
            Y = ymeas;
            U = umeas;
            X = [U ones(size(U))];
            b = (inv(X'*X))*(X'*Y);
            Ju_hat = b(1,1);
            Ju(sim_k) = Ju_hat;
            flag(sim_k) = 0;
        end

        if sim_k > 1 && (mean(Ju(sim_k-10:sim_k)) >0.001)  % ...
           I-controller
            u_in0 =   u_in0  + Ki*Ju_hat ;
        else
            u_in0 = u_in0 ;
        end
    end
        u_in = u_in0+ 0.1*sin(2*pi*(1/(TD))*sim_k);  % add perturbation
end

close(h);

sim.Ju = Ju;
%%

figure(3412)
subplot(411)
hold all
plot(sim.y)
plot(sim.yf)
plot( sim.ySS,'--')
set(gca,'FontSize',20);
```

```
grid on
ylabel 'y'

subplot(412)
hold all
plot(sim.u)
plot(sim.uf,'--')
set(gca,'FontSize',20);
grid on
ylabel 'u'

subplot(413)
hold all
plot(sim.JuSS,'--')
plot(sim.Ju)
set(gca,'FontSize',20);
grid on
ylabel 'J_u'
ylim([-2,2])

subplot(414)
hold all
plot(flag)
set(gca,'FontSize',20);
grid on
ylabel 'flag'
```

C.2.2 Plant process with second-order transfer function

```
clear
```

```
clc

addpath ('D:\matlab\casadi-matlabR2014b-v3.2.3')
addpath('D:\matlab')
import casadi.*

Ts = 1; % Sample time
y = MX.sym('y');
u = MX.sym('u');
x1 = MX.sym('x1');
x2 = MX.sym('x2');
tau = (174/Ts);

tau_1 = 174;
tau_2 = 60;
tau_a = 40;                  % tau_a = 0 and 40 are studied

[a, b, c, d] = tf2ss([0 tau_a 1],[tau_1*tau_2 tau_1+tau_2 1]);


% dx/dt=Ax+Bu, y=Cx+Du, x=[x1 x2]', A=2X2 matrix
dx1 = a(1,1)*x1+a(1,2)*x2+b(1,1)*(-0.1*u^2+4*u+5);
dx2 = a(2,1)*x1+a(2,2)*x2+b(2,1)*(-0.1*u^2+4*u+5);
y = c(1,1)*x1+c(1,2)*x2+d*(-0.1*u^2+4*u+5);

ode = struct('x',vertcat(x1,x2),'p',u,'ode',vertcat(dx1,dx2),'quad',y);
opts = struct('tf',Ts);

F = integrator('F','cvodes',ode,opts);

xf = [0; 42.5/c(1,2)];
u_in0 = 15;
```

```
u_in = u_in0;

Ki = 0.005;                    % Controller gain
PRBS = 30;                     % PRBS calculation steps
l = 4*180;                     % Window size

nIter = 4*3600;

h = waitbar(0,'Simulation in Progress...');
aL = Ts/(Ts+ 100);
sim.uf = u_in;
sim.yf = c*xf;



ARX = 1;
pbrs = 0;

for sim_k  = 1:nIter
    waitbar(sim_k /nIter,h,sprintf('Time: %0.0f min',sim_k*Ts/60))

    Fk = F('x0',xf,'p',u_in);
    xf = full(Fk.xf);

    sim.y(sim_k) = c*xf + 0.005*randn(1);
    sim.u(sim_k) = u_in + 0.005*randn(1);
    sim.ySS(sim_k) = -0.1*u_in^2+4*u_in+5;
    sim.JuSS(sim_k) = -0.2*u_in+4;
    if sim_k>1
        sim.uf(sim_k) = (1-aL)*sim.uf(sim_k-1) + aL*sim.u(sim_k);
        sim.yf(sim_k) = (1-aL)*sim.yf(sim_k-1) + aL*sim.y(sim_k);
    end
    if sim_k >Gradient
        ymeas = sim.yf(sim_k-Gradient:sim_k)';
        umeas = sim.uf(sim_k-Gradient:sim_k)';

        Ju(Gradient) = 1;           %initialization

        if Ju(sim_k-1) > 0.030      % ARX model
            Y0 = ymeas - mean(ymeas);
            U0 = umeas - mean(umeas);

            data = iddata(Y0,U0,Ts);
            sysARX = arx(data,[2,2,0]);
            sysD = idss(sysARX);
            sys = d2c(sysD);
            Ju_hat = (-sys.C*(sys.A\sys.B) + sys.D);
            Ju(sim_k) = Ju_hat;
            flag(sim_k) = 1;
        else                        % LS method
            Y = ymeas;
            U = umeas;
            X = [U ones(size(U))];
            b = (inv(X'*X))*(X'*Y);
            Ju_hat = b(1,1);
            Ju(sim_k) = Ju_hat;
            flag(sim_k) = 0;
        end

        if sim_k > Gradient && (mean(Ju(sim_k-10:sim_k)) >0.001)   ...
            % I-controller
            u_in0 =   u_in0  +Ki*Ju_hat ;
        else
            u_in0 = u_in0 ;
        end
    end

if rem(sim_k,PRBS)==0
    pbrs = 1*idinput(1);
end
u_in = u_in0+ pbrs;  % add perturbation
end
close(h);

sim.Ju = Ju;

%%

figure(1)
subplot(411)
hold all
plot(sim.y)
plot(sim.yf)
plot( sim.ySS,'--')
set(gca,'FontSize',20);
grid on
ylabel 'y'

subplot(412)
hold all
plot(sim.u)
plot(sim.uf,'--')
set(gca,'FontSize',20);
grid on
ylabel 'u'

```

```
subplot(413)
hold all
plot(sim.JuSS,'--')
plot(sim.Ju)
set(gca,'FontSize',20);
grid on
ylabel 'J_u'
ylim([-2 2])

subplot(414)
hold all
plot(flag)
set(gca,'FontSize',20);
grid on
ylabel 'flag'
```

C.3 Classic ESC

C.3.1 Plant process with first-order transfer function

```
clear
clc

addpath ('D:\matlab\casadi-matlabR2014b-v3.2.3')
addpath('D:\matlab')
import casadi.*

Ts = 1; % Sample time
y = MX.sym('y');
u = MX.sym('u');
```

```
tau = (174/Ts);

dx1 = ((-0.1*u^2+4*u+5) - y)/tau;

ode = struct('x',y,'p',u,'ode',dx1,'quad',y);
opts = struct('tf',Ts);

F = integrator('F','cvodes',ode,opts);

xf = 42.5;
u_in0 = 15;
u_in = u_in0;

nIter = 4*1e5;

h = waitbar(0,'Simulation in Progress...');

TD = 800;                  % dither time constant
Th = (0.2)*TD;             % High Pass Filter time constant
Tl = 10*TD;                % Low Pass Filter time constant
a = 1;                     % sine amplitude
Ki = 0.005;                % Controller gain

%intitalisation
z(1) = 0;
x(1) = 0;
u_esc(1) = 15;
y_esc(1) = 42.5;
sim.u(1) = 15;
sim.y(1) = 42.5;
J(1) = y_esc(1);
u_hat(1) = u_esc(1);
```

```
sine(1) = 0;
f = 1/TD;

% filter coefficients
al = Ts/(Ts + Tl);
ah = Th/(Ts + Th);

ARX = 1;
pbrs = 0;


for sim_k  = 2:nIter
    waitbar(sim_k /nIter,h,sprintf('Time: %0.0f min',sim_k*Ts/60))

    Fk = F('x0',xf,'p',u_in);
    xf = full(Fk.xf);

    sim.y(sim_k) = xf + 0.005*randn(1);
    sim.u(sim_k) = u_in + 0.005*randn(1);

    y_esc(sim_k) = sim.y(sim_k);
    J(sim_k) = sim.y(sim_k);
    u_esc(sim_k) = sim.u(sim_k);

    z(sim_k) = ah*z(sim_k-1) + ah*J(sim_k) - ah*J(sim_k-1); ...
                     % High pass filter
    x(sim_k) = ((1-al)*x(sim_k-1) + ...
        al*z(sim_k)*sin(2*pi*f*Ts*(sim_k)));  % correlation and Low ...
        pass filter
    u_esc(sim_k) = u_esc(sim_k-1) + Ts*Ki*x(sim_k).*2/a; ...
                         % I-controller
    u_hat(sim_k) = u_esc(sim_k) + a*sin(2*pi*f*Ts*(sim_k)); ...
                        % estimated optimal input + dither
    sine(sim_k)= sin(2*pi*f*Ts*(sim_k));

    u_in = u_hat(sim_k);


end
close(h);
%% Plotting

hold all
figure(1)
subplot(311)
hold all
plot(y_esc)
set(gca,'FontSize',20);
ylabel 'y'

subplot(312)
hold all
plot(u_hat)
set(gca,'FontSize',20);
ylabel 'u'

subplot(313)
hold all
plot(x.*2/a)
set(gca,'FontSize',20);
ylabel 'J_u'
```

C.3.2 Plant process with second-order transfer function

```
clear
clc

addpath ('D:\matlab\casadi-matlabR2014b-v3.2.3')
addpath('D:\matlab')
import casadi.*

Ts = 1; % Sample time
y = MX.sym('y');
u = MX.sym('u');
x1 = MX.sym('x1');
x2 = MX.sym('x2');
tau = (174/Ts);

tau_1 = 174;
tau_2 = 60;
tau_a = 40;                   % tau_a = 0 and 40 are studied

[a,b,c,d] = tf2ss([0 tau_a 1] , [tau_1*tau_2 tau_1+tau_2 1]);

% dx/dt=Ax+Bu, y=Cx+Du, x=[x1 x2]', A=2X2 matrix
dx1 = a(1,1)*x1+a(1,2)*x2+b(1,1)*(-0.1*u^2+4*u+5);
dx2 = a(2,1)*x1+a(2,2)*x2+b(2,1)*(-0.1*u^2+4*u+5);
y = c(1,1)*x1+c(1,2)*x2+d*(-0.1*u^2+4*u+5);

ode = struct('x',vertcat(x1,x2),'p',u,'ode',vertcat(dx1,dx2),'quad',y);
opts = struct('tf',Ts);

F = integrator('F','cvodes',ode,opts);
```

```

xf = [0; 42.5/c(1,2)];
u_in0 = 15;
u_in = u_in0;

nIter = 4e5;

h = waitbar(0,'Simulation in Progress...');

TD = 1800;                    % dither time constant
Th = (0.2)*TD;                % High Pass Filter time constant
Tl = 10*TD;                   % Low Pass Filter time constant
a = 1;                        % sine amplitude
Ki = 0.001;                   % Controller gain

%intitalisation
z(1) = 0;
x(1) = 0;
u_esc(1) = 15;
y_esc(1) = 42.5;
sim.u(1) = 15;
sim.y(1) = 42.5;
J(1) = y_esc(1);
u_hat(1) = u_esc(1);
sine(1) = 0;
f = 1/TD;

% filter coefficients
al = Ts/(Ts + Tl);
ah = Th/(Ts + Th);

ARX = 1;
```

```
pbrs = 0;

for sim_k  = 2:nIter
    waitbar(sim_k /nIter,h,sprintf('Time: %0.0f min',sim_k*Ts/60))

    Fk = F('x0',xf,'p',u_in);
    xf = full(Fk.xf);

    sim.y(sim_k) = c*xf + 0.005*randn(1);
    sim.u(sim_k) = u_in + 0.005*randn(1);

    y_esc(sim_k) = sim.y(sim_k);
    J(sim_k) = sim.y(sim_k);
    u_esc(sim_k) = sim.u(sim_k);

    z(sim_k) = ah*z(sim_k-1) + ah*J(sim_k) - ah*J(sim_k-1); ...
                     % High pass filter
    x(sim_k) = ((1-al)*x(sim_k-1) + ...
        al*z(sim_k)*sin(2*pi*f*Ts*(sim_k)));  % correlation and Low ...
        pass filter
    u_esc(sim_k) = u_esc(sim_k-1) + Ts*Ki*x(sim_k).*2/a; ...
                        % I-controller
    u_hat(sim_k) = u_esc(sim_k) + a*sin(2*pi*f*Ts*(sim_k)); ...
                     % estimated optimal input + dither
    sine(sim_k)= sin(2*pi*f*Ts*(sim_k));

    u_in = u_hat(sim_k);

end
close(h);
%% Plotting

```

```
hold all
figure(1)
subplot(311)
hold all
plot(y_esc)
set(gca,'FontSize',20);
ylabel 'y'

subplot(312)
hold all
plot(u_hat)
set(gca,'FontSize',20);
ylabel 'u'

subplot(313)
hold all
plot(x.*2/a)
set(gca,'FontSize',20);
ylabel 'J_u'
```