Frida Bakken Myrvang

# Implementation of Extremum Seeking Control in an Experimental Lab-Rig

Master's thesis in Chemical Engineering and Biotechnology
Supervisor: Johannes Jäschke
Co-supervisor: Jose Otavio Assumpcao Matias
June 2021

Master's thesis

Frida Bakken Myrvang

# Implementation of Extremum Seeking Control in an Experimental Lab-Rig

Master's thesis in Chemical Engineering and Biotechnology
Supervisor: Johannes Jäschke
Co-supervisor: Jose Otavio Assumpcao Matias
June 2021

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Chemical Engineering

# Abstract

In this thesis, *Extremum seeking control* (ESC) is implemented in an experimental lab rig that represents a gas lifted well network. ESC is a purely data-driven, unconstrained, optimization method, where the plant gradients are estimated directly from measurements of the inputs and the objective function, and used to optimize the system. It is an alternative to traditional model-based optimization, such as Real-Time Optimization. Since no model is required, ESC solves some of the challenges related to model-based optimization, such as building and obtaining models of the system. Some of the main challenges with ESC lies in the estimation of the gradients and the parameter tuning, in addition to handling constraints with an unconstrained optimization method.

The goal of this thesis is to investigate the potential industrial application for ESC. This includes comparing the performance of different gradient estimation methods in ESC, and to study how constraint handling affects the performance of ESC. In the experimental lab-rig, ESC is implemented with three different gradient estimation techniques, *Least Square Estimation* (LSE), ARX-models and *Fast Fourier Transform* (FFT). In order to study these and the performance of ESC, three cases are considered. Two of the case studies are simple preliminary studies of a constrained and an unconstrained optimization problem. These are used to validate the gradient estimations, ESC's ability to optimize the system, and the proposed constraint handling strategy. The last case study is a more realistic optimization of the gas-lifted well, where noisy well production flowrates are included in the objective function and where constraints must be handled, which makes the gradient estimation more challenging.

From the results it was clear that ESC was able to drive the system to its optimum, in all cases, without violating any constraints. However, constraint handling and noisy measurements make the ESC implementation more challenging. Noisy measurements degraded the performance of all gradient estimation methods, while the constraint handling strategy only has a negative effect on the FFT. From the case studies, ESC seems to be most suited for unconstrained systems, with less measurement noise, but it is also applicable to other systems. The constant perturbation of the inputs, required in ESC, can make the method unsuited for chemical industrial applications. Thus, its benefits should be proven, both regarding economic performance and the simplicity of the method, before implementing ESC in a real chemical system.

# Sammendrag

I denne oppgaven implementeres *Extremum seeking control* (ESC) i en eksperimentell lab-rig som representerer et gassløftnettverk med tre brønner. ESC er en fullstendig datadrevet optimeringsmetode, der gradientene til målfunksjonen blir estimert fra datamålinger av inputene og målfunksjonen og brukt for å optimere systemet. Det er et alternativ til tradisjonell modellbasert optimering, som sanntidsoptimalisering. Siden metoden ikke behøver en modell løser den noen av utfordringene i modellbasertoptimering, som å lage og oppdatere en modell av systemet. Noen av hovedutfordringene med ESC er estimeringen av gradienten og å finne gode verdier for parameterne som må bestemmes. I tillegg tar ikke ESC i utgangspunktet hensyn til restriksjonene i optimeringsproblemet, noe som kan være utfordrende.

Målet i denne oppgaven er å undersøke den potensielle muligheten til å bruke ESC for industrielle formål. Dette inkluderer å sammenligne ulike metoder for å estimere gradienter i ESC, og å studere hvordan det å inkludere restriksjoner påvirker ESC. I den eksperimentelle lab-riggen er ESC implementert med tre forskjellige gradientestimeringsmetoder, minste kvadraters metode, ARX-modeller og fast Fourier transformasjon (FFT). For å undersøke disse, og ESC, er det utført 3 casestudier. To av de er enkle innledende studier, brukt for å validere gradientestimeringen, ESC sin evne til å optimere systemet og den foreslåtte metoden for å håndtere restriksjoner. Den siste casestudien er en mer realistisk optimering av et gassløftnettverk, der målinger av oljeraten fra brønnene, med forstyrrelser, er inkludert i målfunksjonen og restriksjoner må håndteres. Dette gjør estimeringen av gradientene mer utfordrende.

Fra resultatene var det klart at ESC var i stand til å optimere systemet, i alle casene, uten å bryte noen restriksjoner. Likevel gjorde restriksjoner og målinger med forstyrrelser implementasjonen av ESC mer utfordrende. Målinger med forstyrrelser degraderte ytelsen til alle gradientestimeringsmetodene, mens å inkludere restriksjoner bare hadde en negativ effekt på FFT. Fra casestudiene ser det ut som ESC er mest egnet for systemer uten restriksjoner, med lite målingsforstyrrelser, men metoden kan også benyttes i andre systemer. Den konstante oscillering av inputene kan gjøre metoden uegnet i kjemisk industri, og det bør derfor være klare fordeler, både med tanke på økonomisk ytelse og metodens enkelhet, før ESC skal implementeres i ekte kjemiske systemer.

# Preface

This thesis was written in the spring of 2021, as the final part of my M.Sc in Chemical Engineering at the Norwegian University of Science and Technology.

I would like to thank my supervisor Johannes Jäschke for the opportunity to work on this topic for my specialisation project and master's thesis. Also, a big thank you to my co-supervisor Jose Otavio Assumpcao Matias for all the appreciated help and support during my work.

Finally, I would like to thank my fellow students and friends at Chemical Engineering and Biotechnology for making these five years an unforgettable time!

## Declaration of Compliance

I, Frida Bakken Myrvang, hereby declare that this is an independent work according to the exam regulations of the Norwegian University of Science and Technology (NTNU).
**Signature:**

Frida B. Myrvang

**Place and Date:** Trondheim - Gløshaugen, June 2021

# Table of Contents

# List of Tables

# List of Figures

# Nomenclature

*Acronyms*

| | |
|---|---|
| CV | Controlled variable |
| DOF | Degree of freedom |
| ESC | Extremum-seeking control |
| FFT | Fast Fourier transform |
| LSE | Least-square estimation |
| LSESC | Least-square extremum-seeking control |
| MV | Manipulated variable |
| OF | Objective function |
| PID | Proportional–integral–derivative |
| PRBS | Pseudo binary random sequence |
| RTO | Real time optimization |
| SISO | Single input single output |
| SOC | Self-optimizing control |

| ***Symbol*** | ***Definition*** | ***Unit*** |
|---|---|---|
| $a$ | Amplitude of dither signal | [sL/min] |
| $\mathbf{d}$ | Disturbances | [-] |
| $F_s$ | Samples per seconds | [s$^{-1}$] |
| $\mathbf{f}$ | Steady-state model equations | [-] |
| $\mathbf{g}$ | Operational constraints | [-] |
| $J$ | Objective function in Case Study 1 | [(sL)$^2$min$^{-2}$] |
| $J$ | Objective function in Case Study 2 and 3 | [\$] |
| $\hat{\mathbf{J}}_\mathbf{u}$ | Estimated gradients in Case Study 1 | [sL/min] |
| $\hat{\mathbf{J}}_\mathbf{u}$ | Estimated gradients in Case Study 2 and 3 | [\$ min/sL] |
| $j$ | Unit imaginary number | [-] |
| $K_I$ | Integral gain in Case Study 1 | [-] |
| $K_I$ | Integral gain in Case Study 2 and 3 | [\$$^{-1}$] |
| N | Buffer length | [-] |

|  |  |  |
|---|---|---|
| $n_a$ | ARX-parameter (number of poles) | [-] |
| $n_b$ | ARX-parameter (number of zeros) | [-] |
| P | Number of periods within the buffer length | [-] |
| **u** | Inputs | [sL/min] |
| $u_{max}$ | Maximum input value | [sL/min] |
| $u_{min}$ | Minimum input value | [sL/min] |
| $u_{max}^{tot}$ | Maximum total input usage | [sL/min] |
| $w_{gl_i}$ | Gas lift rate in well $i$ | [sL/min] |
| $w_{ro_i}$ | Oil rate from reservoir | [kg/s] |
| **x** | State dependant variables | [-] |
| $\mathcal{J}$ | Fourier transformation of $J$ | [-] |
| $\mathcal{U}$ | Fourier transformation of $u$ | [-] |
| $\alpha_o$ | Price of oil | $[\frac{\$}{\text{kg/s}}]$ |
| $\alpha_{gl}$ | Cost of compressing gas for gas lift injection | $[\frac{\$}{\text{sL/min}}]$ |
| $\omega$ | Frequency | [s$^{-1}$] |
| $\Phi$ | Regressor vector | [-] |
| $\theta$ | Parameters to be estimated | [-] |
| $\phi_J$ | Phase information of OF | [rad] |
| $\phi_u$ | Phase information of inputs | [rad] |

Chapter 1

# Introduction

## 1.1 Motivation

Production optimization approaches generally seek to maximize the production, with minimal cost. A common way to address this is *Real-time optimization* (RTO) [22]. RTO is the on-line calculation of the optimal set-points for the process that allow the profit to be maximized, while satisfying the operational constraints [22].

There are many techniques to solve the RTO problem. Traditionally, a model-based optimization technique is used to calculate the optimal set-points of the process [22]. If a good model of the system is available and an appropriate algorithm to solve the problem is used, this will likely give good results. However, the plant model often represents an over-simplification of the process, and the model prediction can differ from the actual process behavior due to plant-model mismatch, uncertainty in parameters and unmeasured disturbances [17]. The calculated set-points are optimal for the simplified model, but necessarily optimal for the plant [6]. Making an accurate plant model can be complicated, time demanding, and a continuous work in process, since the system most likely will change over time [20]. In addition, solving complicated models online requires computational capacity [21].

On the other hand, model-free optimization methods are becoming more popular. These can counteract the effect of both expected and unexpected disturbances [6]. As implied, the model equations are not taken into account, which can be time- and cost-saving, when it comes to both making the model and calculating the optimal set-points online. Also, mismatch in the plant model and uncertainty in the parameters is not an issue, since no model is used.

One of these model-free optimization methods is *Extremum Seeking Control* (ESC) [16]. ESC is an unconstrained optimization technique, where the gradients of the objective function with respect to the inputs are estimated directly from data measurements. The process can then be optimized by driving the estimated gradients to zero, using integral control [2].

The gradient estimation step of the classical ESC approach consists of adding a sinusoidal dither to the inputs, and use high-pass and low-pass filters to extract the gradient [2]. However, implementation of a sinusoidal dither, in a real system, can be a challenge. Estimation techniques that do not require this can be an advantage in certain systems. In addition, the high- and low-pass filter have tuning parameters, and the performance of ESC relies heavily on the tuning parameters value [6]. Obtaining these can be time consuming, so choosing gradient estimation techniques with fewer tuning parameters can be advantageous. Alternatively, other data-based gradient estimation techniques can also be used, such as *Least Square Estimation* [10], *Fast Fourier transform* [11] and gradient estimation from fitting an ARX-model to the data [15].

As traditional model-based RTO, ESC implementations also have some challenges:

- The main challenge lies in the estimation of the plant gradients, especially if there are noisy measurements.
- Different gradient estimation techniques may suit different types of problems and systems, choosing the most adequate can be hard.
- Tuning the gradient estimation parameters, to get good performance, can be a challenging task.
- Constantly perturbing the inputs in the system may be undesirable.
- ESC is a slow control approach.
- It can be difficult to know if the system is actually operating at the optimum, since the optimality conditions cannot be checked as in model-based optimization schemes.
- It is an unconstrained optimization technique, so handling constraints is not straightforward.

ESC is not widely used in chemical processes. In the literature, ESC is mostly studied in chemical systems using simulations. For example, bio reactors [30, 26, 25, 18], non-isothermal reactors [9, 7], the CANON process [28], heat exchanger networks [31] and gas lifted oil wells [14]. However, only few ESC implementations have been reported. Among them test rigs for axial-flow compressor [8], the redox processes in wastewater chemical treatment [19] and the deammonification process [27]. Implementing ESC in lab-scale setups is an important step towards applying ESC in real chemical processes in the industry.

## 1.2 Scope

In this master thesis, the model-free optimization method, ESC, is implemented in an experimental lab rig. The lab rig represents a gas lift subsea oil well network, where the production optimization objective is to maximize the total oil production.

The ESC approach is implemented with three different gradient estimation techniques, which are

- Least square estimation
- Gradient estimation from fitting an ARX-model
- Fast Fourier transform

There are performed three different case studies. Two of the case studies are simple preliminary studies, where only the inputs are a part of the optimization problem. These are used to validate the gradient estimation techniques, the optimization, and the proposed constraint handling strategy. The last case study is a more realistic optimization of a gas-lifted well, where noisy oil rate measurements are included in the objective function, making the gradient estimation more challenging.

The main contributions of this thesis are

- Implementing ESC in a lab-scale setup, which is important for investigating ESC for potential industrial application.
- Comparing the performance of three different gradient estimation techniques in ESC.

- Developing a constraint handling strategy for the production optimization problem of interest.

Chapter 2

# General Concepts and Theory

In this chapter, the general concepts and theory are explained. This includes a general overview of optimization problems and the standard ESC framework, a detailed description of the gradient estimation methods used in the thesis (LSE, ARX-models and FFT) and guidelines on how to tune their parameters, and an explanation of the proposed ESC constraint handling strategy.

## 2.1 Building an Optimization Problem

Optimization problems are central in many disciplines, including chemical processes operation. These can be stated and presented in a standard form, consisting of an objective function, that needs to be optimized, and a set of constraints that must be satisfied.

The constraint set can contain upper and lower limits of the problem variables, limitations regarding safety, operation, or environmental regulations, and relations between variables, which are usually represented by a system model [5].

In this thesis, a process where the optimal operating condition can be described by the following optimization problem is considered

$$\begin{aligned} &\min_{\mathbf{x},\mathbf{u}} \quad J(\mathbf{x},\mathbf{u},\mathbf{d}) \\ \text{s.t} \quad &\mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}) = 0 \\ &\mathbf{g}(\mathbf{x}, \mathbf{u}, \mathbf{d}) \leq 0 \end{aligned} \tag{2.1}$$

where $J$ is the objective function to be optimized, **u** is the vector of decision variables used to optimize $J$, while **x** and **d** are the state dependent variables and disturbances, respectively. **f** and **g** represent the constraints that must be satisfied, where **f** are the steady state model equations and **g** are the operational limitations.

## 2.2 Extremum-Seeking Control

Extremum-seeking control (ESC) is a model-free, real-time optimization method and adaptive control approach. Since there is no need for the model equations, these are not considered in the optimization problem. Equation 2.1 can therefore be simplified as:

$$\begin{aligned} &\min_{\mathbf{x},\mathbf{u}} \quad J(\mathbf{u},\mathbf{d}) \\ \text{s.t} \quad &\mathbf{g}(\mathbf{u}, \mathbf{d}) \leq 0 \end{aligned} \tag{2.2}$$

Instead of using an optimization algorithm to solve the problem above, in ESC we first estimate the plant gradients. In order to do this, the inputs are perturbed, and the effect on the objective function is observed [16]. The estimated gradients can be written as

$$\mathbf{J_u} = \left[\frac{\partial J}{\partial u_1} \ \frac{\partial J}{\partial u_2} \ \cdots \ \frac{\partial J}{\partial u_{n_u}}\right] = [J_{u,1} \ \ J_{u,2} \ \ldots \ J_{u,n_u}] \tag{2.3}$$

where $n_u$ is the number of decision variables.

When the gradients are estimated, $n_u$ integral controllers are used to drive them to the desired value. If the problem is unconstrained (i.e **g(u,d)**:=Ø), which are the types of problems ESC is designed for, the gradients are driven to zero. This satisfies the first-order necessary condition of optimality [5].

In the unconstrained case the integral control, in discrete time, can be written as

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \mathbf{K}_I \hat{\mathbf{J}}_\mathbf{u} \tag{2.4}$$

where $\mathbf{K}_I$ is the diagonal gain matrix. The integral gain determines how aggressively the input changes [2]. Integral control is used since it is desirable to have slow control.

## 2.3 ESC Approaches

There are many different ESC approaches. The main similarity between the approaches is that a data widow containing input and objective function measurements is used to estimate the gradients, before using integral control to optimize the system. However, how these gradients are estimated can differ. A simple figure of a general ESC scheme is shown below, in Figure 2.1.

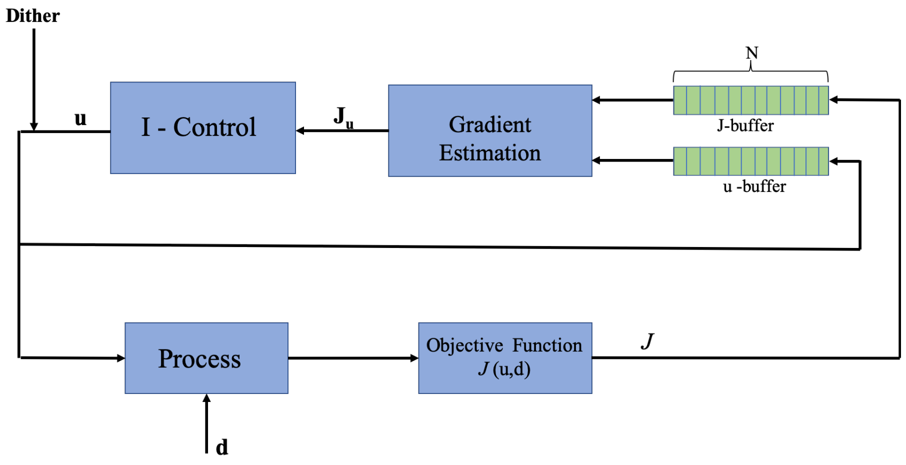


**Figure 2.1:** A simple, general, ESC scheme.

The classical approach consists of adding a sinusoidal wave, with a given frequency, as a dither to the input, resulting in a sinusoidal system response of the objective function. A high-pass filter is then used to subtract the static bias from $J$, which gives a sinusoidal response with zero mean. The product of this response and the sinusoidal input have a mean amplitude that is approximately the steady-state objective function gradient, and can be extracted using a low-pass filter[2].

Alternatively least square estimation [10], ARX-models [15] or the Fast Fourier transform [11] can be used to estimate the gradients. These are used and explored in this thesis, and are explained in details in the next sections.

### 2.3.1 Least Square Extremum-Seeking Control

In least square extremum seeking control (LSESC), least squares method is used to estimate the gradients of the objective function with relation to the inputs [10]. As explained in Section 2.2, a moving window of the last N data measurements of **u** and $J$ is used for this purpose. These buffers are given by

$$\mathbf{J} = [J(1)\ \ J(2)\ ...\ J(N)]^T \tag{2.5}$$

$$\mathbf{U} = [\mathbf{u}(1)^T\ \ \mathbf{u}(2)^T\ ...\ \mathbf{u}(N)^T]^T \tag{2.6}$$

where J(N) is the newest sample and J(1) the oldest. These buffers are used to fit a linear model of the objective function, as a function of the inputs. At every time step, the estimated linear model is given by the following equation

$$J = \hat{\mathbf{J}}_{\mathbf{u}}^T \mathbf{u} + \hat{m} \tag{2.7}$$

where $\hat{\mathbf{J}}_{\mathbf{u}}$ is the vector of the estimated gradients from **u** to J, and $\hat{m}$ is the bias term. A visualization of the one dimensional case is shown in Figure 2.2.

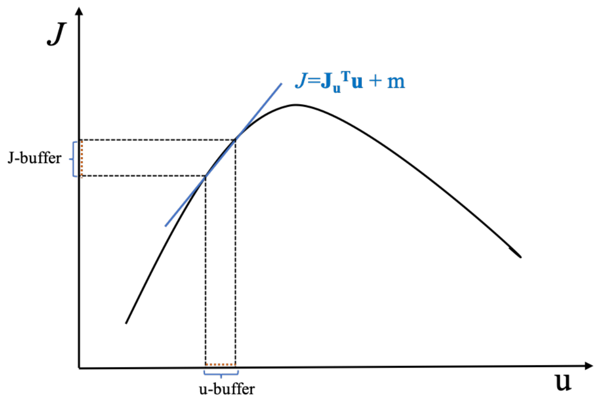


**Figure 2.2:** Visualization of the linear model fit of the objective function as a function of the inputs, from the last $N$ samples of data.

As mentioned, the most common dither in ESC is a sinusoidal wave. In LSESC, other dithers can also be used to perturb the input [10], such as a square wave or a PRBS. Another advantage, compared to the classical approach, is that there are fever tuning parameters since there are no high pass and low pass filters. In LSESC, only the dither signals, the integral gains, and the buffer length need tuning.

### 2.3.2 ESC Using ARX model

Another way of estimating the gradients is to use the last $N$ measurements of the inputs and the cost function to identify an ARX model, and convert the ARX polynomials to a continuous time state-space system [15].

For the SISO case, the identified ARX model is on the following form,

$$\begin{aligned} J(t) = &- a_1 J(t-1) - ... - a_{n_a} J(t-n_a) \\ &+ b'_1 u(t-1) + ... + b'_{n_b} u(t-n_b) + e(t) \end{aligned} \tag{2.8}$$

where $n_a$ is the number of poles and $n_b$ the number of zeros. If dead time is considered the dead time, $n_c$, must also be included in the equation. $n_a$, $n_b$, and potentially $n_c$, are tuning parameters that must be determined. The ARX polynomials are given by

$$A_{poly}(q) = 1 + a_1 q^{-1} + ... + a_{n_a} q^{-n_a} \tag{2.9}$$

$$B_{poly}(q) = b'_1 + ... + b'_{n_b} q^{-n_b} \tag{2.10}$$

where $q^{-1}$ is the unit delay operator.

Given the buffer of $N$ data samples, at time $k$, of the cost function, **J** = $[J(1)\ J(2)\ ...\ J(N)]$ and the input, **u** = $[u(1)\ u(2)\ ...\ u(N)]$, where $J(N)$ is newest sample and $J(1)$ is the oldest sample. The ARX coefficients, given by

$$\theta = [a_1\ ...\ a_{n_a}\ b'_1\ ...\ b'_{n_b}]^T \tag{2.11}$$

can be estimated using least squares estimation

$$\hat{\theta} = arg\ \underset{\theta}{\text{min}} ||\psi - \Phi\theta||_2^2 \tag{2.12}$$

where $\psi$ is given by

$$\psi = [J(N)\ \ J(N-1)\ ...\ J(n+1)]^T \tag{2.13}$$

where $n$ = max$(n_a, n_b)$. **Φ** is given by,

$$\left[\begin{array}{ccc|ccc} -J(N-1) & \dots & -J(N-1-n_a) & u(N-1) & \dots & u(N-1-n_b) \\ -J(N-2) & \dots & -J(N-1-n_b) & u(N-2) & \dots & u(N-2-n_b) \\ \vdots & & \vdots & \vdots & & \vdots \\ -J(n) & \dots & -J(n+1+n_a) & u(n) & \dots & u(n+1-n_b) \end{array}\right]$$

After computing the parameters $\theta$, the following ARX polynomial is obtained

$$J(t) = \frac{B_{poly}(q)}{A_{poly}(q)} u(t) \tag{2.14}$$

and the gradient can be estimated as

$$J_u = A_{poly}^{-1} B_{poly} \tag{2.15}$$

For the derivation of the ARX model in a multi-variable case see [1]. For simplicity when computing the gradients in the multiple input case, the ARX model is converted to a continuous time state-space system on the following form

$$\begin{aligned} \dot{x} &= Ax + Bu \\ J &= Cx + Du \end{aligned} \tag{2.16}$$

The steady state gain can then be found by setting $\dot{x} = 0$, which gives

$$J = (-CA^{-1}B + D)u = \hat{\mathbf{J}}_{\mathbf{u}} u \tag{2.17}$$

where $\hat{\mathbf{J}}_{\mathbf{u}}$ are the estimated gradients. After the gradients are estimated, integral control can be used to drive the system to its optimum, as explained in Section 2.2.

Similarly to LSESC, non-sinusoidal dithers can be applied when the ARX method is used. However, there are some tuning parameters, other than the tuning of the dithers, integral gains, and the buffer length. The model orders, $n_a$ and $n_b$, must be determined, and if dead time is considered, $n_c$, that is the number of input samples that occur before the input affects the output, must also be determined.

### 2.3.3 ESC Using Fast Fourier Transform

The fast Fourier transform (FFT) can also be used to estimate gradients. The motivation of combining both methods comes from the ability of FFT to decompose a signal, dependent on time, into different frequency components, with an amplitude and a phase. Since in ESC the system is perturbed by inputs with different frequencies, FFT becomes an intuitive approach for analysing the effect of different frequency components in the output signal, here the objective function measurements [11].

When a signal is transformed into the frequency domain by the FFT, it is possible to extract the frequency components of interest. These are the frequencies of the dither signals. When slow sinusoidal perturbations are added to the inputs, the objective function will also vary with the same frequencies, but with a different amplitude depending on the system gain. The power spectrum, which shows the amplitude of the variations, at different frequencies in the signal, can be used to estimate the steady-state gradient of the objective function with respect to each of the input signals [11].

Figure 2.3 shows a simple example of how an objective function, dependent on 2 two inputs, varies with the same frequencies as the perturbation frequencies of the inputs. The corresponding power spectrum are also included in the figure. The power spectrum for the objective function shows peaks in the frequencies of the two inputs $\omega_1$ and $\omega_2$. Also, it shows that the power associated with $\omega_2$, which indicates the effect of the second input in the objective function is larger than the effect of the first input. This means that the gain from the second input is larger than the gain from the first one.

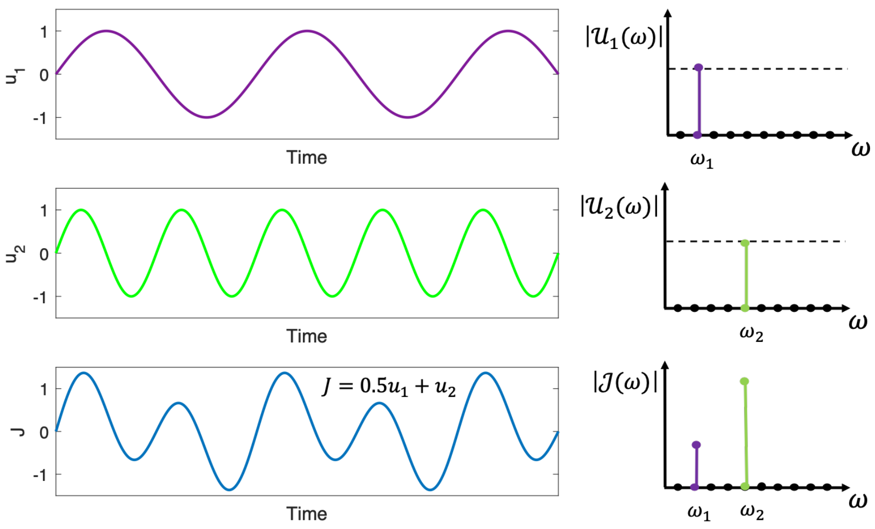


**Figure 2.3:** A simple example on how an objective function, $J$ (the blue line), vary with the same frequencies as the perturbation frequency of two inputs, $u_1$ (sinusoidal signal with frequency $\omega_1$ - green line) and $u_2$ (sinusoidal signal with frequency $\omega_2$ - purple line). The right side plots are the power spectrum of the objective function and the inputs.

The FFT-based ESC approach is implemented as follows. As for the two other gradient estimation techniques, a moving window of the $N$ last measurements of the objective function

$$[J(1) \;\; ... \;\; J(N)]^T$$

and the inputs

$$[\mathbf{u}(1)^T \;\; ... \;\; \mathbf{u}(N)^T]^T$$

are used to estimate the gradients. Before the FFT is performed, the signals must be detrended so they have zero mean [11]. When the FFT is performed, $N$ frequency components are extracted from the objective function and the inputs, as shown in Equation 2.18 and 2.19 respectively.

$$\mathcal{J}(l) = \sum_{k=0}^{N-1} J_0(k) e^{-j\frac{2\pi}{N}lk} \qquad \forall\, l = 0, ..., N-1 \tag{2.18}$$

$$\mathcal{U}_i(l) = \sum_{k=0}^{N-1} u_{i,0}(k) e^{-j\frac{2\pi}{N}lk} \qquad \forall\, l = 0, ..., N-1 \tag{2.19}$$

$J_0$ is the detrended objective function measurement and $u_{i,0}$ is the detrended measurement of input $i$. $j$ is the unit imaginary number and $l$ is the frequency components.

The amplitude of the objective function and the input signals, for all frequencies $l = 1, ..., N/2$, can be obtained from the single-sided amplitude spectrum, $2|\mathcal{J}(l)|$ and $2|\mathcal{U}_i(l)|$, and the the frequency components of interest can be extracted [11]. The single sided amplitude spectrum is used so that the negative frequencies computed by the FFT are discarded. The frequency, $\omega_l$, of frequency component $l$ is determined by

$$\omega_l = F_s \cdot l/N \tag{2.20}$$

where $F_s$ is the number of samples per second and $N$ is the number of samples.

The objective function gradient w.r.t the $i^{th}$ input is found by extracting the frequency component that corresponds to the perturbation frequency of input $i$, that is $\omega_i$. The magnitude of the gradient is determined by the relation between the amplitude of $\mathcal{J}$ and $\mathcal{U}_i$, at $\omega_i$

$$\left|\frac{\partial J}{\partial u_i}\right| = \left|\frac{\mathcal{J}(\omega_l = \omega_i)}{\mathcal{U}(\omega_l = \omega_i)}\right| \qquad \forall\, i = 1, ..., n_u \tag{2.21}$$

The amplitude spectrum is always a positive number, so in order to determine the sign of the gradient, the phase information of the input signal, $\phi_J(\omega_i)$, with respect to the phase information of the input signal, $\phi_{u_i}(\omega_i)$, is used [11]. An illustration of this is shown below, in Figure 2.4.

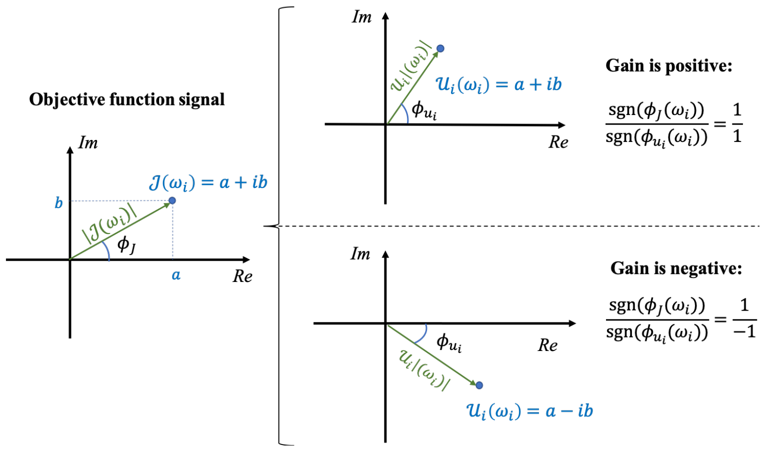

Objective function signal
Im
Re
b
a
$\mathcal{J}(\omega_i) = a + ib$
$|\mathcal{J}(\omega_i)|$
$\phi_J$
Im
Re
$|\mathcal{U}_i(\omega_i)|$
$\mathcal{U}_i(\omega_i) = a + ib$
$\phi_{u_i}$
Gain is positive:
$\frac{\text{sgn}(\phi_J(\omega_i))}{\text{sgn}(\phi_{u_i}(\omega_i))} = \frac{1}{1}$
Im
Re
$\phi_{u_i}$
$|\mathcal{U}_i(\omega_i)|$
$\mathcal{U}_i(\omega_i) = a - ib$
Gain is negative:
$\frac{\text{sgn}(\phi_J(\omega_i))}{\text{sgn}(\phi_{u_i}(\omega_i))} = \frac{1}{-1}$

**Figure 2.4:** Illustration of how the sign of the gradients is determined by the phase information of the input signal and the objective function signal.

The gradients are estimated by

$$\frac{\partial J}{\partial u_i} = \left|\frac{\mathcal{J}(\omega_i)}{\mathcal{U}(\omega_i)}\right| \text{sgn}\left[\frac{\phi_J(\omega_i)}{\phi_{u_i}(\omega_i)}\right] \tag{2.22}$$

After the gradients are estimated, integral control can be used in order to optimize the system, like in the other ESC approaches.

As the LSESC approach, the gradient estimation technique has no additional tuning parameters. Only the parameters for the dithers, the integral gains, and the buffer length must be tuned. But in this approach the buffer length should be an integer multiple of the time period of the input perturbations [11]. Also, the method is based on a sinusoidal perturbation of the inputs.

## 2.4 Parameter Tuning

For all the gradient estimation methods described above, the buffer length, $N$, the integral gains, $\mathbf{K_I}$, and the amplitude, $a$, and the frequency, $\omega$ of the dither signals need to be tuned.

There are three time scales to consider when tuning these parameters,

- The convergence to the optimum - integral gain (slow)
- The perturbation frequency - dither signal (medium)
- The controlled plant dynamics - response from input to output (fast)

To have a well functioning extremum-seeking controller, there should be a separation between these time scales [24]. The time scale of the controlled plant dynamics should the fastest, the perturbation of the inputs the medium, and the convergence to the optimum should be the fastest.

The integral gain must be chosen small enough so that the convergence to the optimum is slower than the perturbation frequency. On the other hand, the integral gain should not be too small, because this will lead to very slow optimum tracking, and the plant will operate suboptimally for an unnecessarily long period.

The input perturbation frequencies must be chosen small enough, so this time scale is slower than the time scale for the controlled plant dynamics. Also, if there are multiple inputs, the dither frequencies must be different. This is to be able to estimate the gradient with respect to the individual inputs. If the frequencies are the same, the inputs change uniformly and it is not possible to separate the effects of the individual inputs [24]. The integral gain and the perturbation amplitude, however, can be the same for all inputs. The amplitudes of the dither signals should be chosen large enough such that their effect on the objective function can be clearly identified. If the objective function measurements are noisy, the amplitude should be larger. However, there is a trade-off that should be taken care of because unnecessarily large input perturbations are not desirable from a process operation point of view.

The buffer length should be large enough to encompass the input changes. A larger buffer length could give a better estimation of the gradients, since there is more data used in the estimation. However, if a new disturbance affects the system, the buffer data will become outdated. Then, if a larger buffer is being used, the data related to the old disturbance will affect the accuracy of the estimated gradients for a longer period.

In addition, some gradient estimation techniques, such as the classical approach and the ARX method, require external tuning parameters. These also need to be tuned to fit the system and give a good estimation of the gradients.

## 2.5 Constraint Handling in Extremum Seeking Control

ESC is an unconstrained optimization technique. However, there are methods to handle constraints. One approach is relaxing the constraints, where constraint deviations are penalised. By relaxing the constraints, the constraint problem can be converted to an unconstrained problem [4]. Hence, a constrained optimization problem given by

$$\begin{aligned} &\min_{\mathbf{u}} \quad J(\mathbf{u},\mathbf{d}) \\ &\text{s.t} \qquad \mathbf{g}(\,\mathbf{u},\mathbf{d}) \leq 0 \end{aligned} \tag{2.23}$$

can be converted to

$$\min_{\mathbf{u}} \quad J + \sum_{i=1}^{n_g} p_i g_i \tag{2.24}$$

where $p_i$ is some weighting factor that penalises violating constraint $g_i$, for $i = 1, ..., n_g$, where $n_g$ is the number of constraints. When a constraint is violated, it has a positive effect on the objective function, which the optimizer would want to counteract.

Another approach that can be used to handle constraints, is to use active constraint control, along with self-optimizing control (SOC) for the unconstrained degrees of freedom [12]. SOC can be explained using the concept of self optimizing variables, which are defined as a set of CVs, that, ever when kept at a constant setpoint, the process still operates at an acceptable loss in face of disturbances [23]. When using this approach, one MV is used to handle each active constraint. The rest of the MVs are used to control the self-optimizing CVs to a constant set-point. For guidelines on how to choose self-optimizing variables, see [23]. The study conducted by Skogestad (2000) also points out that the ideal self-optimizing CV is the steady-state gradient from the input to the objective function $J_u$. This is because at the optimum, the gradient should be zero, and the self optimizing CVs can be controlled to a constant set-point of zero [23].

In this thesis, the second mentioned constraint handling approach is used. How this is implemented is further described in Section 5.2.2.

Chapter 3

# System Description and Lab-setup

In this thesis, ESC is implemented in an experimental lab-rig. This chapter contains a description of the lab-setup and a short presentation of the case studies carried out in this thesis.

## 3.1 Gas Lifted Well Network

The system the experimental lab-rig represents is a gas lifted well network with 3 wells. A simple sketch of the system is shown on the next page, in Figure 3.1.

The overall goal in the system is to maximize the production of oil from the reservoirs to the top facilities. If the pressure in the reservoirs is not high enough, the production can be increased by using artificial lift methods. In the system of interest, gas lift techniques are used. The idea of gas lift consists of injecting gas at the bottom of the wells. Then, the bulk density of the system decreases as well as the hydrostatic pressure. Since the reservoirs and wells are located several meters under the sea level, the effects of a lower hydrostatic pressure difference are significant and the reservoir outflow increases even with a constant reservoir pressure. However, if too much gas is injected, the frictional pressure drop effects start to dominate and the reservoir outflow decreases [13]. There is clearly a trade-off in the choice of the gas injection, which can be handled by production optimization methods such as ESC.

The gain in the oil production, by injecting gas, is not linear, it depends on the outflow from the reservoir. An increase in the outflow will decrease gain.

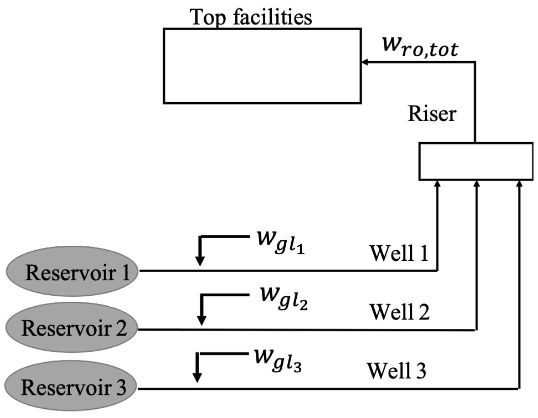


**Figure 3.1:** Simple figure of the gas lifted well network. $w_{gl_i}$ is the gas lift rate in well $i$ and $w_{ro,tot}$ is the total oil production from the 3 wells.

## 3.2 The Experimental Lab-rig

As seen in Figure 3.2, the experimental rig emulates the system presented in the previous section. The rig consists of reservoirs, wells, and risers. The top facilities are represented by a separation tank. The experimental lab-rig is a simplification of this system, and uses water and air, instead of oil and gas, but this does not have any influence on the gas lift phenomenon. A simple flowsheet of the experimental lab rig is shown in Figure 3.2.

The reservoir consists of a tank, a pump, and three control valves. The pump rotation can be regulated and the pump outlet pressure is measured. The opening of the valves can be manipulated to represent different reservoir productivity indexes. However, in our case, the valve openings of the three wells are set to the same value, such that the reservoirs behave equally, and are kept constant during the experiments. The flows before the valve openings are measured.

The wells are represented by three parallel hoses of 1.5m with an inner diameter of 2cm. Air is injected by three air flow controllers, approximately 10 cm after the reservoirs valves. These flow rates are the MVs, **u**, used in the three optimization problems presented in the next section.

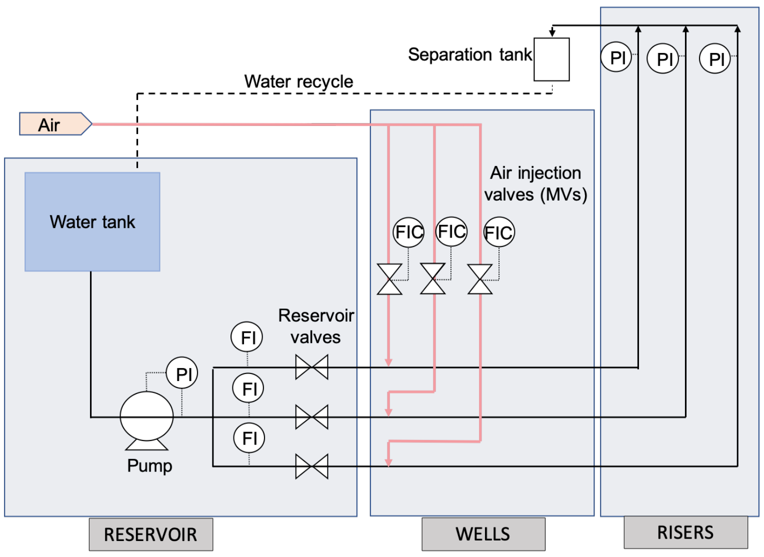


**Figure 3.2:** Simple flowsheet of the experimental lab rig.

The risers are composed of three 2.2m high hoses with the same diameter as the hoses in the well. The pressures can be measured at the top of the riser. The liquid is recirculated back to the tank in the reservoir and the air is vented out.

## 3.3 Presentation of Case Studies

In this thesis three case studies are performed. Two of the case studies are simple toy examples used to validate the gradient estimation, ESCs ability to optimize the system, and the constraint handling strategy. The last case study is a more realistic optimization problem of a gas lifted well network.

- **Case Study 1** is used to optimize a simple unconstrained problem, where the objective is only a function of the inputs. Since the optimum can be analytically derived and the gradients at the optimum are known to be equal to zero, the gradient estimation and ESC ability to drive the inputs to their optimal values is studied. Also, the performances of the three gradient estimation techniques (LSE, ARX-based, and FFT) are compared.
- **Case Study 2** is used to optimize a simple constrained problem. Again, the ob-

jective function is only a function of the inputs and the optimum can be computed beforehand. The goal of the second case study is to identify if the proposed constraint handling strategy affects ESC's ability to track the system optimum, and how it interacts with the three gradient estimation techniques.

- **Case Study 3** is used to optimize a gas lifted well network. It is a constrained problem, and the constraint handling strategy from Case Study 2 is used. In this case, the well production flowrates are included in the objective function instead of only the inputs. Thus, the system optimum cannot be directly calculated from the objective function. Moreover, the objective function measurements become noisy, which makes the gradient estimation more challenging compared to the two other case studies.

Chapter 4

# Case Study 1: An Unconstrained Optimization Problem

In Case Study 1, ESC is used to optimize an unconstrained optimization problem. The aim of this case study is to see if the gradient estimation from the experimental lab rig is sufficient to drive the system to the unconstrained optimum using ESC. The objective function is chosen so that the optimum value of the inputs and the plant gradients can be analytically computed, which makes it easy to verify the gradient estimation and the control to the optimum. The three gradient estimation techniques are tested in individual experiments.

## 4.1 The Optimization Problem

The optimization problem for this case is given by

$$\min_{\mathbf{u}} J = (u_1 - 3)^2 + (u_2 - 2.5)^2 + (u_3 - 1.5)^2 \tag{4.1}$$

where, $\mathbf{u} = [u_1\ u_2\ u_3]$, are the system manipulated variables. Clearly, the optimal solution to this problem is

$$\mathbf{u_{opt}} = [3\ \ 2.5\ \ 1.5] \tag{4.2}$$

and the gradients can be computed by

$$\mathbf{J_u} = \begin{bmatrix} \partial J/\partial u_1 \\ \partial J/\partial u_2 \\ \partial J/\partial u_3 \end{bmatrix} = \begin{bmatrix} 2(u_1 - 3) \\ 2(u_2 - 2.5) \\ 2(u_3 - 1.5) \end{bmatrix} \tag{4.3}$$

## 4.2 Methodology

In this section the methodology for implementing ESC, for Case Study 1, is explained. This includes the dither, the gradient estimation, the control, and the chosen tuning parameters for this specific case. The general scheme for building the extremum seeking controller, from Chapter 2.2, is shown below.

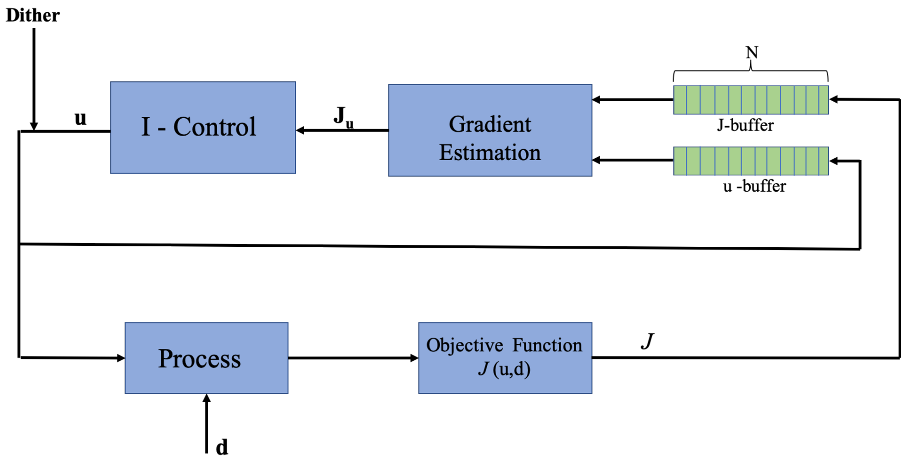


**Figure 4.1:** A simple, general, ESC scheme.

### 4.2.1 The Dither

In order to estimate the gradients, a dither must be added to each of the three inputs. In this case study, a square wave dither is used. A square wave is a periodic waveform, where the amplitude alternates between a fixed minimum and maximum, with the same duration at minimum and maximum. This type of dither is used in the lab rig because it is easier to implement than a sinusoidal dither, but still has many of the same properties as a sinusoidal wave, such as being periodical, with a given frequency, and having a given amplitude. In discrete-time the inputs with the added dither can be written like

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \mathbf{a} \cdot \mathbf{sq.wave}_k = \begin{bmatrix} u_{1,k} \\ u_{2,k} \\ w_{3,k} \end{bmatrix} + \begin{bmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & a_3 \end{bmatrix} \begin{bmatrix} \text{sq.wave}(\omega_1)_k \\ \text{sq.wave}(\omega_2)_k \\ \text{sq.wave}(\omega_3)_k \end{bmatrix} \tag{4.4}$$

where **a** is the amplitudes of the dither signals, and $\omega_i$ is the frequency of the square wave added to input $u_i$. A figure of a square wave is shown in Figure 4.2. $P$ is the number of periods within the buffer, $N$.

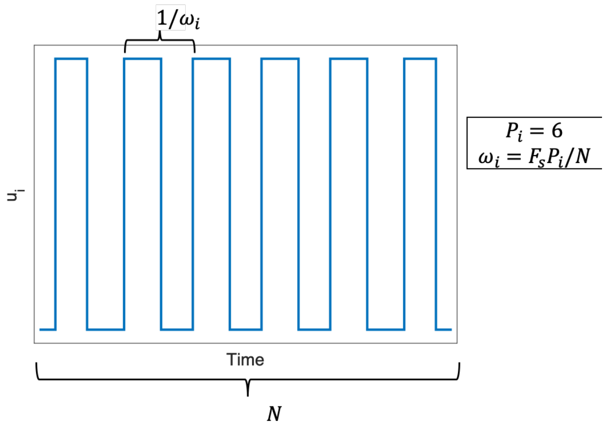


**Figure 4.2:** Explanation of the square wave.

When the FFT is used to estimate the gradients, the buffer length must be an integer multiple of the perturbation time period [11], i.e $P$ must be an integer number. Therefore, when $N$ and $P$ are determined, the frequency comes as a result of this. The inputs must have an individual perturbation frequency, in order to estimate the gradient with respect to each input. The amplitudes, on the other hand, are set equal. This makes sense since the inputs are the same type of variable, the gas lift rate.

### 4.2.2 Gradient Estimation

As explained in the theory, the $N$ last measurements of the inputs and the objective function are used to estimate the gradients. For the LSE and ARX approach this is done exactly as explained in Section 2.3.1 and 2.3.2, respectively. For the FFT, on the other hand, some issues occur when estimating the gradients. As seen from Equation 2.22, the magnitude and the sign of the gradient, using FFT, are estimated separately. Since we are working with real data and we use a square wave instead of a sinusoidal wave, small deviations can result in a wrong sign in the estimated gradients, even though the estimation of the magnitude is good.

To counteract sudden sign changes in the estimated gradient from the FFT, the gradient used in the control is a mean of the $N_{mean}$ last estimated gradients. The overall change in the inputs will be the same, but the control will be smoother. This is not done for the LSE and the ARX approach.

### 4.2.3 Control

Since this is an unconstrained problem, there is no need to use any constraint handling strategy in this ESC implementation, and the optimal operating strategy, $\mathbf{u_{opt}}$, can be found by controlling the estimated gradients to a constant set-point of zero. This is done by using integral control, with the estimated gradients as CVs. The integral control can be written as in Equation 2.4, from Section 2.

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \mathbf{K}_I \hat{\mathbf{J}}_\mathbf{u} = \begin{bmatrix} u_{1,k} \\ u_{2,k} \\ u_{3,k} \end{bmatrix} + \begin{bmatrix} k_{I1} & 0 & 0 \\ 0 & k_{I2} & 0 \\ 0 & 0 & k_{I3} \end{bmatrix} \begin{bmatrix} \hat{J}_{u1} \\ \hat{J}_{u2} \\ \hat{J}_{u3} \end{bmatrix} \tag{4.5}$$

The same integral gain is used for all the inputs. This is done because they are inputs of the same kind, with the same unit of measurement, and sampled at the same rate.

### 4.2.4 Tuning Parameters

The tuning parameters for Case Study 1 are shown in Table 4.1, below. For simplicity, the tuning parameters are set equal in all experiments. This gives all three methods the same baseline for estimating the gradients, and the only variation in the experiments is the gradient estimation techniques. The aim with the experiments is to compare the estimation techniques, which is easier to do when the tuning parameters are set the same. However, it is possible that another set of tuning parameters could have improved the performance of one or more of the methods.

**Table 4.1:** Tuning parameters in Case Study 1.

| **Parameter** | **Value LSE** | **Value ARX** | **Value FFT** |
|---|---|---|---|
| Sampling time [s] | 2 | 2 | 2 |
| $N$ [-] | 60 | 60 | 60 |
| $N_{mean}$ [-] | - | - | 10 |
| $a$ [sL/min] | 0.1 | 0.1 | 0.1 |
| $P_1$ [-] | 2 | 2 | 2 |
| $P_2$ [-] | 3 | 3 | 3 |
| $P_3$ [-] | 5 | 5 | 5 |
| $\omega_1$ [s$^{-1}$] | 1/60 | 1/60 | 1/60 |
| $\omega_2$ [s$^{-1}$] | 1/40 | 1/40 | 1/40 |
| $\omega_3$ [s$^{-1}$] | 1/24 | 1/24 | 1/24 |
| $K_{I1}$ [-] | 0.005 | 0.005 | 0.005 |
| $K_{I2}$ [-] | 0.005 | 0.005 | 0.005 |
| $K_{I3}$ [-] | 0.005 | 0.005 | 0.005 |
| $n_a$ | - | 1 | - |
| $n_b$ | - | [1 1 1] | - |

## 4.3 Results

In this section, the results, from implementing ESC to the experimental lab rig, are presented. Three different experiments are performed, one with each of the estimation techniques. These are presented separately, since the dithers are not exactly identical in the experiments, due to experimental uncertainties. In all experiments the inputs start from a suboptimal point, **u** = [1 1 1], which is arbitrary chosen.

### 4.3.1 LSE

Figure 4.4 shows the result from using LSE as the gradient estimation approach. The left side plots show the three inputs and the objective function. The plots on the right side show the value of the estimated gradients, in addition to the real value of the gradients, which is computed using Equation 4.3. The estimation of the gradients do not start before the buffer is filled with measurements. The buffer length is 60, and the sampling time is 2 s, so the estimation, and the control, do not start before 2 minutes has passed.

The figure shows that the inputs and the cost function are driven to their optimum. There are some delay between the estimated gradients and the real value, but from the experiment it is clear that the gradient estimation from LSE is sufficient to drive the system to its optimum and the gradients to zero.

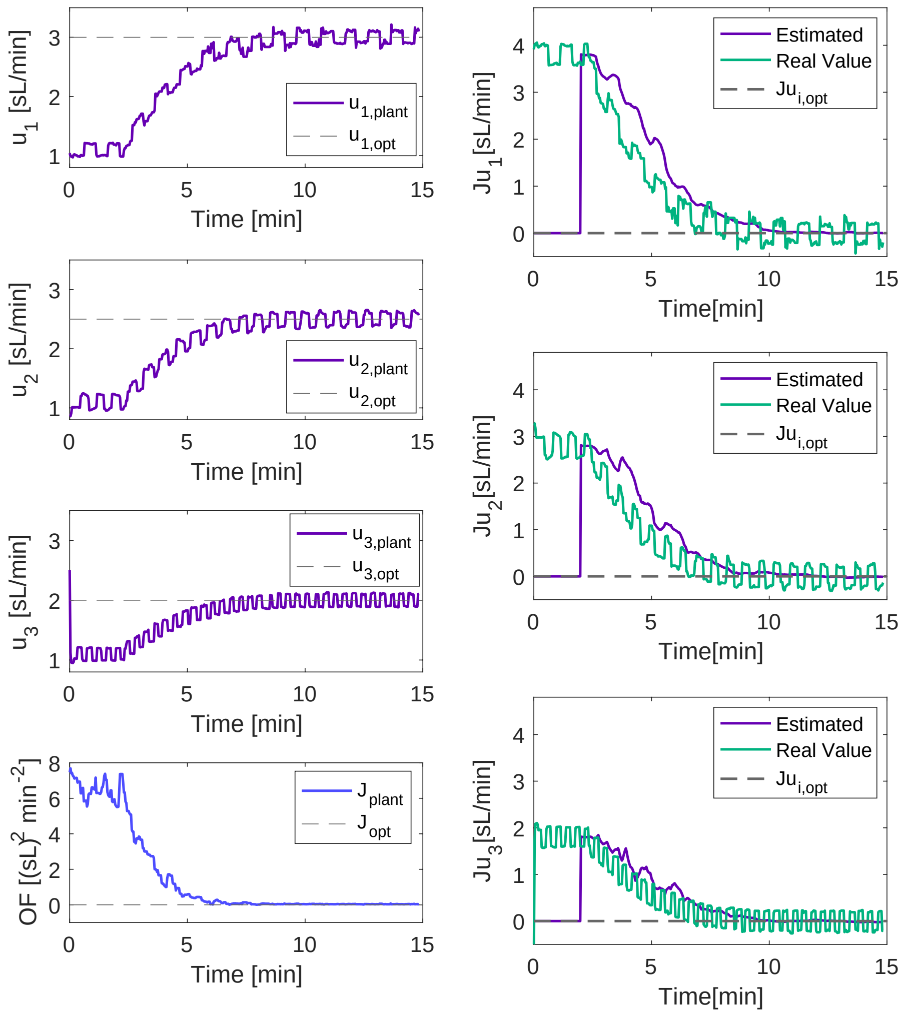


**Figure 4.3:** Result of the unconstrained optimization using LSE. The plots to the left show the inputs and the cost function, and their respective optimums. The plots to the right show the estimated gradients from the experimental lab rig, in addition the real value of the gradients.

### 4.3.2 ARX

The results, using the ARX approach, are shown in Figure 4.4. The figure has the same setup as Figure 4.3, and again, the estimation of the gradients and the control start after 2 minutes. The inputs and the objective function are driven to their optimal set-point. The estimated gradients are driven to zero, but there are some delay between the estimated gradient and the real one. The result are very similar to the experiment with LSE as the gradient estimation technique, in Figure 4.3.

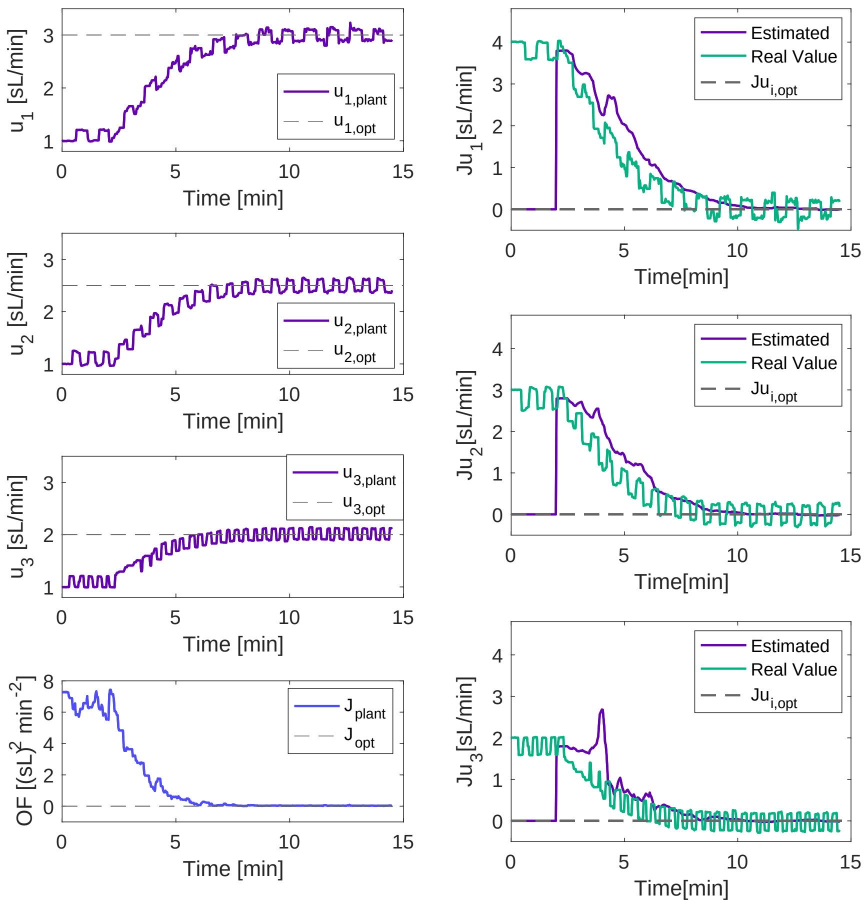


**Figure 4.4:** Result of the unconstrained optimization using ARX. The plots to the left show the inputs and the cost function, and their respective optimums. The plots to the right show the estimated gradients from the experimental lab rig, in addition the real value of the gradients.

The figure shows that there was an issue with the dither in $u_3$, around t = 3 min. For some unknown reason, the dither was not executed. From the estimation of $J_{u3}$, this dither issue had a negative effect on the gradient estimation. However, it did not seem to affect the optimization towards the optimal set-point, but it clearly shows the importance of the dither in the gradient estimation.

### 4.3.3 FFT

Figure 4.5 shows the result using FFT as gradient estimation method. Again, the figure is set up like Figure 4.3. In this experiment it takes an additional 20 seconds before the gradients estimation and the control starts, because the $N_{mean}$=10 last estimated gradients are used in the control. Consequently, the controller starts slightly later.

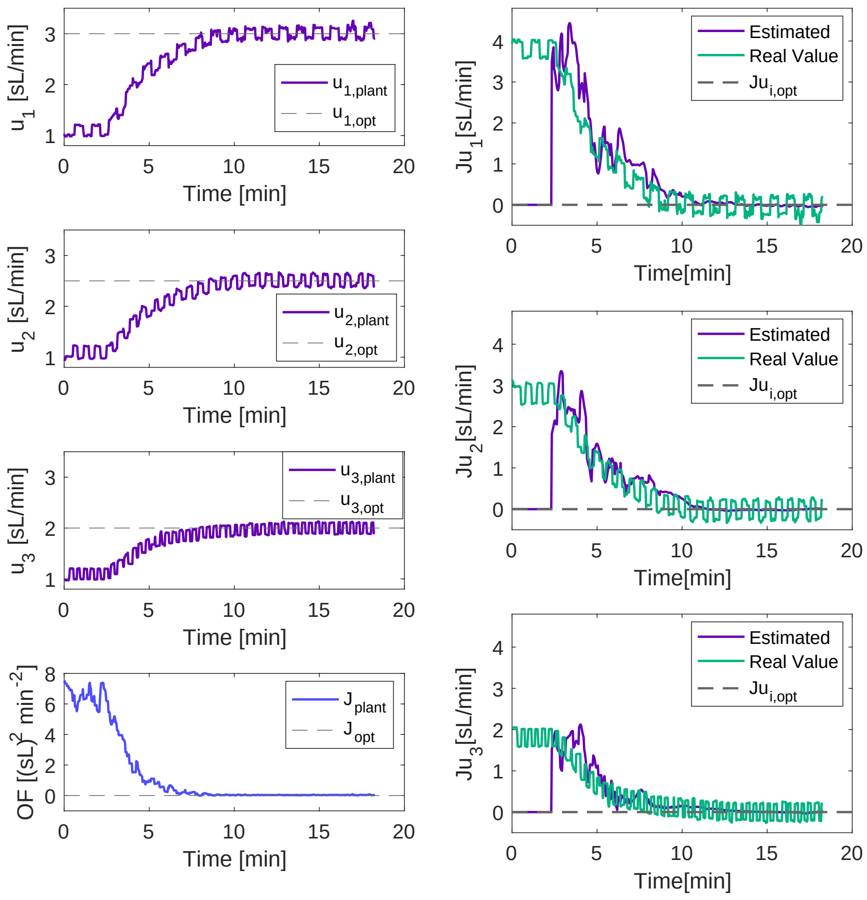


**Figure 4.5:** Result of the unconstrained optimization, using FFT. The plots to the left show the inputs and the cost function, and their respective optimums. The plots to the right show the estimated gradients from the experimental lab rig, in addition the real value of the gradients.

Figure 4.5 shows that there is some mismatch between the estimated gradient and the real gradient. Compared with the LSE and ARX, the estimated gradients are less smooth. Also, ESC reacts slower when FFT is the gradient estimation method. As a consequence, it takes more time for the system to reach the optimum set-point.

Compared with the LSE and ARX, the FFT used more time to drive the input to their optimum, despite having the same integral gains. One reason for this is that the average of the 10 last estimated gradients is used to control the system. Another explanation could be the negative spikes in the estimated gradient. These will decrease the average gradient, used in the control, and the inputs will move slower against the optimum.

Figure 4.6 show how the non averaged gradients look like. We know that the gradients should be positive, but the sign estimation clearly oscillates in some points. The main reason for this behavior is the square wave as the input dither and not a perfect sinusoidal wave, in addition to noisy input data. As seen from Equation 2.22, the sign and amplitude of the gradient are estimated separately. From these results, these sign oscillations are an effect that degrades the FFT performance, and makes it less reliable. The figure clearly shows why the average of the $N_{mean}$ = 10 last estimated gradients was used in the estimation.

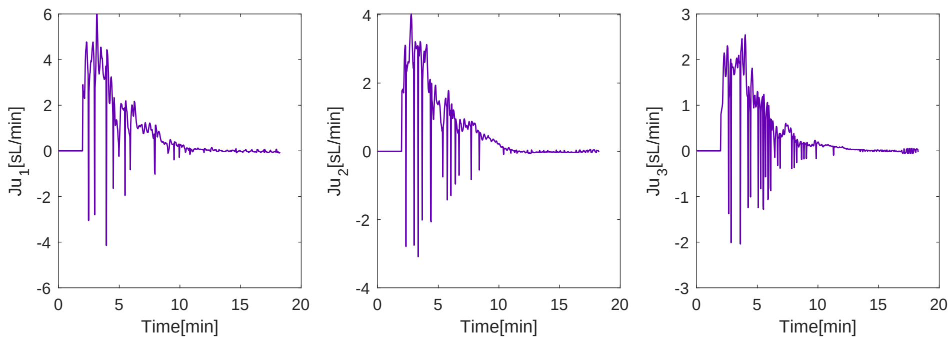


**Figure 4.6:** Non-averaged estimated gradients from experiment with FFT.

## 4.4 Conclusion

From this case study, it is clear that ESC combined with any of the three gradient estimation methods is able to track the system optimum. There was some deviation between the real value of the gradients and the estimated gradients, for all three experiments. However, the system was still driven to its optimum.

Overall, the performance was good in all three experiments. To better compare the methods, a performance analysis, where the loss in each experiment is compared, is performed. The FFT starting time is adjusted, such that the control starts at the same time in the analysis. A plot of the performance analysis is shown in Figure 4.7

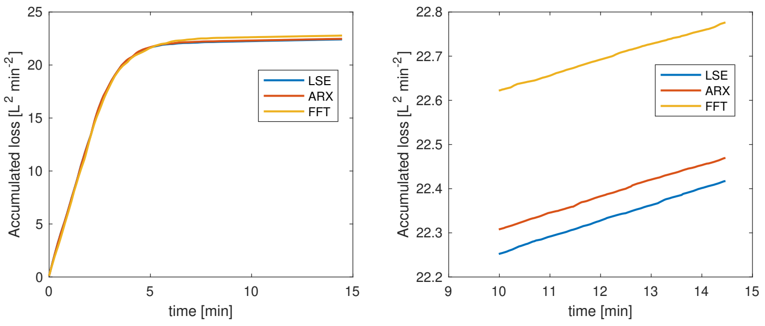


**Figure 4.7:** Performance analysis where the loss in the experiments are compared. The plot on the right side is a zoomed version of the left-hand side plot. The right hand side is zoomed in version of the left-hand side plot.

The results confirm the previous conclusion that the FFT performance was a bit worse than the ARX and LSE, even when the 20 second delay is disregarded. The performance for the LSE and ARX is almost identical, but by zooming in at the end of the plot we can see that the performance of the LSE was slightly better. The reason for this is most likely due to arbitrary differences in the rig, such as the dither issue in the ARX run.

Chapter 5

# Case Study 2: Simple Constrained Problem

Case Study 2 is designed for testing the proposed constraint handling strategy, in addition to verifying the gradient estimated and the control to the plant optimum. The objective function and the constraints only depend on the inputs, as in Case Study 1. So, again, the estimated gradients and the optimum can be easily computed. Moreover, we can detect if the proposed strategy has any effect on ESC capacity of tracking the true plant optimum. To check if the constraint handling is general enough, we change some parameters in the objective function after a given time, which represent a disturbance in the system, leading to a new optimal operating point.

## 5.1 Optimization Problem

$$
\begin{aligned}
&\max_{\mathbf{u}} J = \begin{cases} u_1 + 2u_2 + 3u_3 & t = 0 \\ 3u_1 + u_2 + 2u_3 & t = 17 \text{ min} \end{cases} \\
\text{s.t} \quad & \sum_{i=1}^{3} u_i \leq 7.5 \\
& 1 \leq u_i \leq 4 \quad \forall i \in \mathcal{N} = \{1, 2, 3\}
\end{aligned}
\tag{5.1}
$$

The optimization problem in Case Study 2 is given by Equation 5.1. The system is subject to constraints on the maximum and minimum value of the inputs, which can represent operational limits input. In addition, there is a constraint on the sum of the inputs. This is a maximum input usage constraint, and can represent a limited amount of resources that have to be distributed among the system units.

Note that, in this case, $J$ is a linear function of the inputs, therefore the optimum will always be on the constraints. When the OF changes, we need to be able to trade the active constraint set. The OF itself is simpler in this case, if compared to Case Study 1. However, the change in the OF and the constraints make this case more complicated.

From Equation 5.1, one can see that to maximize $J$, given the first OF, $u_3$ should be maximized, and that the remaining resources, after satisfying the minimum constraint on $u_1$, should be distributed to $u_2$. Given the second OF, after t = 17 min, $u_1$ should be maximized, and the remaining resources, after satisfying the minimum constraint on $u_2$, should be distributed to $u_3$. Thus, the optimal solution is given by

$$\mathbf{u_{opt}} = \begin{cases} [1.0 \;\; 2.5 \;\; 4.0] & t = 0 \\ [4.0 \;\; 1.0 \;\; 2.5] & t = 17 \text{ min} \end{cases} \tag{5.2}$$

The real value of the gradients can be found directly from the objective function, and are given by

$$\mathbf{J_u} = \begin{cases} [1 \;\; 2 \;\; 3] & t = 0 \\ [3 \;\; 1 \;\; 2] & t = 17 \text{ min} \end{cases} \tag{5.3}$$

## 5.2 Methodology

In this section the building blocks for the implementation of ESC in Case Study 2 are presented. This includes the dither, the gradient estimation, the control, the constraint handling, and the tuning parameters chosen for this specific case.

### 5.2.1 The Dither and Gradient Estimation

In this case study there is applied the same kind of dither as in the previous case. The amplitude is the same, but the frequencies are smaller in this case study. The buffer length is larger, and new $P$s are determined, resulting in slightly different frequencies. When it comes to the data pre-processing for the gradient estimation, the same approach is used. For the FFT, the mean of the $N_{mean}$ last estimated gradients is fed to the controller, to prevent miscalculating the sign of the estimated gradient used to control the system.

### 5.2.2 The Control and Constraint Handling

The control must be handled differently in the constrained case. The approach presented in Section 2.5 - active constraint control and SOC for the unconstrained degrees of freedom [12] - is used. Since there are constraints both on the minimum and maximum value of the inputs, in addition to a constraint on the sum of the inputs, different constraints can become active, depending on the system disturbances, which makes the constraint handling more complicated. Therefore, the approach in [12] is used with some additional logic.

Before the constraints become active, the controlled variables are

$$\begin{aligned} CV_1 &:= \hat{J}_{u,1} \\ CV_i &:= (\hat{J}_{u,i} - \hat{J}_{u,i-1}) \;\; \forall i = 2, 3 \end{aligned} \tag{5.4}$$

The input with the largest value is chosen as $v_1$, and used for controlling $CV_1$. All CVs are controlled to a constant set-point of zero, and the gradients are driven towards equal value. Since $CV_1 := \hat{J}_{u,1}$ is controlled to zero, this approach indirectly drives all gradients to zero.

Since the objective function is linear and all the inputs have positive gains with relation to $J$, it is clear that the maximum gas availability constraint (sum of the inputs) is active at the optimum. This active constraint should be controlled with the input with the largest flow, i.e $v_1$, and $CV_1$ is changed to

$$CV_1 := \sum_{i=1}^{n_w} u_i \tag{5.5}$$

which is controlled to a constant set-point of $u_{max}^{tot}$. Meaning that $u_1$ is kept at a constant value of

$$u_1 = u_{max}^{tot} - \sum_{i=2}^{n_w} u_i \tag{5.6}$$

The main changes from the logic proposed in [12] lie in the fact that we want to increase the input with the largest gradient first. A step-by-step approach of the constraint handling is presented in the next section, including a figure with an overview of the approach.

**Remark 1:** *The constraint handling strategy is based on the assumption that the estimated gradients are positive numbers.*

**Remark 2:** *The strategy only allows two of the inputs to be increased at a time, even if all gradients are positive.*

**Remark 3:** *One could argue that using the standard approach in ESC, as done in Case Study 1, in addition to checking the constraints, could be an easier and simpler approach. When the integral control is performed, if the minimum or maximum constraint on a input is violated, the input is set to the constraint. If the maximum gas capacity constraint is violated, $u_{tot} - u_{tot}^{max}$, should be subtracted from the input with the smallest gradient.*

## Constraint Handling: Step-by-step

In the explanation of the step-by-step strategy an illustrative example is used, where $u_1$ corresponds to the input with the largest value, and where the gradient of $u_2$ is larger than the gradients of $u_3$, i.e $J_{u2} > J_{u3}$. A visual overview of the constraint handling strategy is shown below in Figure 5.1.

1. **Check value of flows:** The input with the largest flow should be used to control the active constraint. This can change during operation, and must be checked.

2. **Check the value of the gradient for the two other inputs:** This is done because the order of the value of the gradients, $J_{u2}$ and $J_{u3}$, should be taken into consideration when performing the integral control, to make sure that the inputs with largest gradients are increased.

3. **Treat problem as unconstrained:** New input, $\mathbf{u_{k+1}}$, is calculated by treating the problem as an unconstrained problem, using the approach described in Equation 5.4. Which MV that is largest should be taken into account, in addition to the values of the gradients, compared to each other. Thus, in our example, The MV with the largest flow, $u_1$, should always be used for controlling $CV_1 := \hat{J}_{u1}$ to

$$u_{1,k+1}, = u_{1,k} + k_1 \hat{J}_{u_1} \tag{5.7}$$

How the two other inputs should be controlled depends on the values of $\hat{J}_{u_2}$ and $\hat{J}_{u_3}$, as discussed in step 2. Figure 5.1 shows how the other CV's are computed.

This is done to ensure that $u_2$, in our example, is increased more than $u_3$. This is desirable since $u_2$ has a larger gradient.

4. **Check minimum and maximum constraint on inputs:** Both $u_{min}$ and $u_{max}$ must be checked. If one of the inputs is below the minimum or above the maximum, the input should be changed to the minimum or maximum, respectively.

5. **Check constraint with maximum input capacity:** If the constraint is violated, $u_1$ could be used to control the constraint by setting

$$u_1 = u_{max}^{tot} - u_2 - u_3 \tag{5.8}$$

However, this can lead to two issues

- $u_1$ is decreased, even though $J_{u1}$ is the largest gradient:
- The minimum constraint on $u_1$ can be violated:

For solving these issues, the following intermediary steps are considered. If $J_{u1} > J_{u_2}$, meaning that $J_{u1}$ is the largest gradient, $u_1$ should not be decreased. In this case, the following should be obtained

$$u_2 + u_3 = u_{max}^{tot} - u_1 \tag{5.9}$$

If $J_{u,1} < J_{u_2}$, we can decrease $u_1$ in favour for $u_2$ and $u_3$. In this case, the following should be done

$$u_2 + u_3 = u_{max}^{tot} - u_{min} \tag{5.10}$$

In both cases, the constraint on maximum input capacity will not be violated. In both scenarios, a certain amount, $a$, must be subtracted from $u_2$ and/or $u_3$, this amount is

$$\begin{aligned} a &= (u_2 + u_3) - (u_{max}^{tot} - u_1) \qquad \text{if } J_{u1} > J_{u2} \\ a &= (u_2 + u_3) - (u_{max}^{tot} - u_{min}) \qquad \text{if } J_{u1} < J_{u_2} \end{aligned} \tag{5.11}$$

In our illustrative example, since $J_{u_2} > J_{u_3}$, the amount should be subtracted from $u_3$ first. However, the minimum constraint on $u_3$ can not be violated. So if

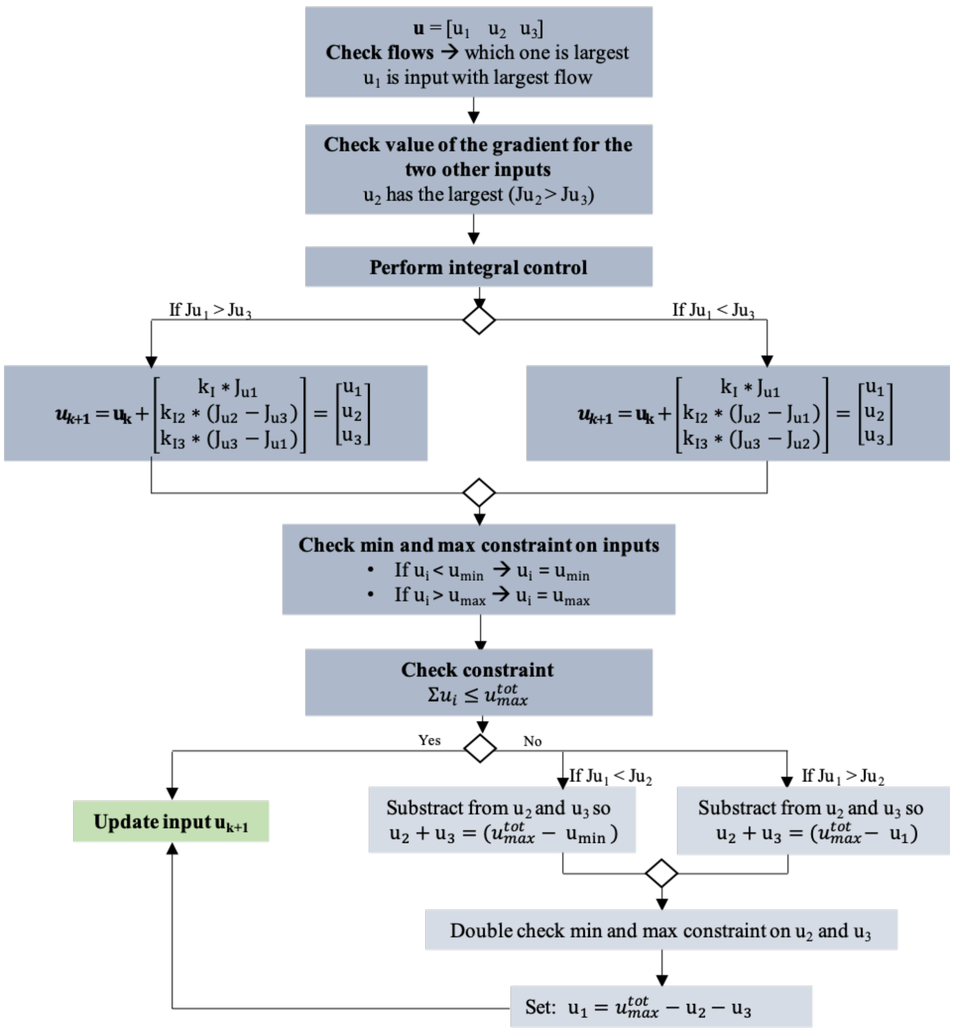
$$a > u_3 - u_{min} \tag{5.12}$$

$u_3 - u_{min}$ is subtracted from $u_3$, and the remaining part of $a$ is subtracted from $u_2$.

And $u_1$ can now be set to

$$u_1 = u_{max}^{tot} - u_2 - u_3 \tag{5.13}$$

6. **Update the input:** Now, all the constraints are handled, and the input,$\mathbf{u_{k+1}}$, can be updated.



$\mathbf{u} = [u_1 \quad u_2 \quad u_3]$
**Check flows** → which one is largest
$u_1$ is input with largest flow
**Check value of the gradient for the two other inputs**
$u_2$ has the largest ($Ju_2 > Ju_3$)
**Perform integral control**
If $Ju_1 > Ju_3$
If $Ju_1 < Ju_3$
$u_{k+1} = u_k + \begin{bmatrix} k_I * J_{u1} \\ k_{I2} * (J_{u2} - J_{u3}) \\ k_{I3} * (J_{u3} - J_{u1}) \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$
$u_{k+1} = u_k + \begin{bmatrix} k_I * J_{u1} \\ k_{I2} * (J_{u2} - J_{u1}) \\ k_{I3} * (J_{u3} - J_{u2}) \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$
**Check min and max constraint on inputs**
If $u_i < u_{min}$ → $u_i = u_{min}$
If $u_i > u_{max}$ → $u_i = u_{max}$
**Check constraint**
$\Sigma u_i \leq u_{max}^{tot}$
Yes
No
**Update input $u_{k+1}$**
If $Ju_1 < Ju_2$
If $Ju_1 > Ju_2$
Substract from $u_2$ and $u_3$ so $u_2 + u_3 = (u_{max}^{tot} - u_{min})$
Substract from $u_2$ and $u_3$ so $u_2 + u_3 = (u_{max}^{tot} - u_1)$
Double check min and max constraint on $u_2$ and $u_3$
Set: $u_1 = u_{max}^{tot} - u_2 - u_3$

**Figure 5.1:** Flow diagram of the proposed constraint handling strategy. For illustration purposes, we used the example where $u_1$ is the MV with the largest flow, while $u_2$ and $u_3$ are the to other MVs with smaller flows, where $J_{u2} > J_{u3}$. However, it can easily be adapted to the different cases.

### 5.2.3 Tuning Parameters

The values for the constraints are shown in Table 5.1, along with the values for the tuning parameters in ESC. Again, the tuning parameters are chosen to be the same in all experiments.

**Table 5.1:** Tuning parameters and variables for Case Study 2.

| Parameter | Value LSE | Value ARX | Value FFT |
|---|---|---|---|
| Sampling time [s] | 2 | 2 | 2 |
| $N$ | 120 | 120 | 120 |
| $N_{mean}$ | - | - | 20 |
| $u_{max}^{tot}$ [sL/min] | 7.5 | 7.5 | 7.5 |
| $u_{max}$ [sL/min] | 4.0 | 4.0 | 4.0 |
| $u_{min}$ [sL/min] | 1.0 | 1.0 | 1.0 |
| $a$ [sL/min] | 0.1 | 0.1 | 0.1 |
| $P_1$ [-] | 7 | 7 | 7 |
| $P_2$ [-] | 6 | 6 | 6 |
| $P_3$ [-] | 5 | 5 | 5 |
| $\omega_1$ [s$^{-1}$] | 1/34.29 | 1/34.29 | 1/34.29 |
| $\omega_2$ [s$^{-1}$] | 1/40 | 1/40 | 1/40 |
| $\omega_3$ [s$^{-1}$] | 1/48 | 1/48 | 1/48 |
| $K_{I_1}$ [\$$^{-1}$] | 0.005 | 0.005 | 0.005 |
| $K_{I_2}$ [\$$^{-1}$] | 0.005 | 0.005 | 0.005 |
| $K_{I_3}$ [\$$^{-1}$] | 0.005 | 0.005 | 0.005 |
| $n_a$ | - | 4 | - |
| $n_b$ | - | [4 4 4] | - |

## 5.3 Results and Discussion

In this section the results from case study 2 are presented. The three approaches for estimating gradients are applied in three individual experiments, and presented separately. In the experiments, the input start from a suboptimal point, $\mathbf{u}$ = [1 1 1], which is arbitrary chosen.

### 5.3.1 LSE

Figure 5.2 shows the result, using LSE as gradient estimation technique. The left side plots show the three inputs and the input summation. The plots on the right side show the objective function and the three gradients, both the estimated and the real value. The buffer length is 120, so the control does not start before 4 minutes has passed.

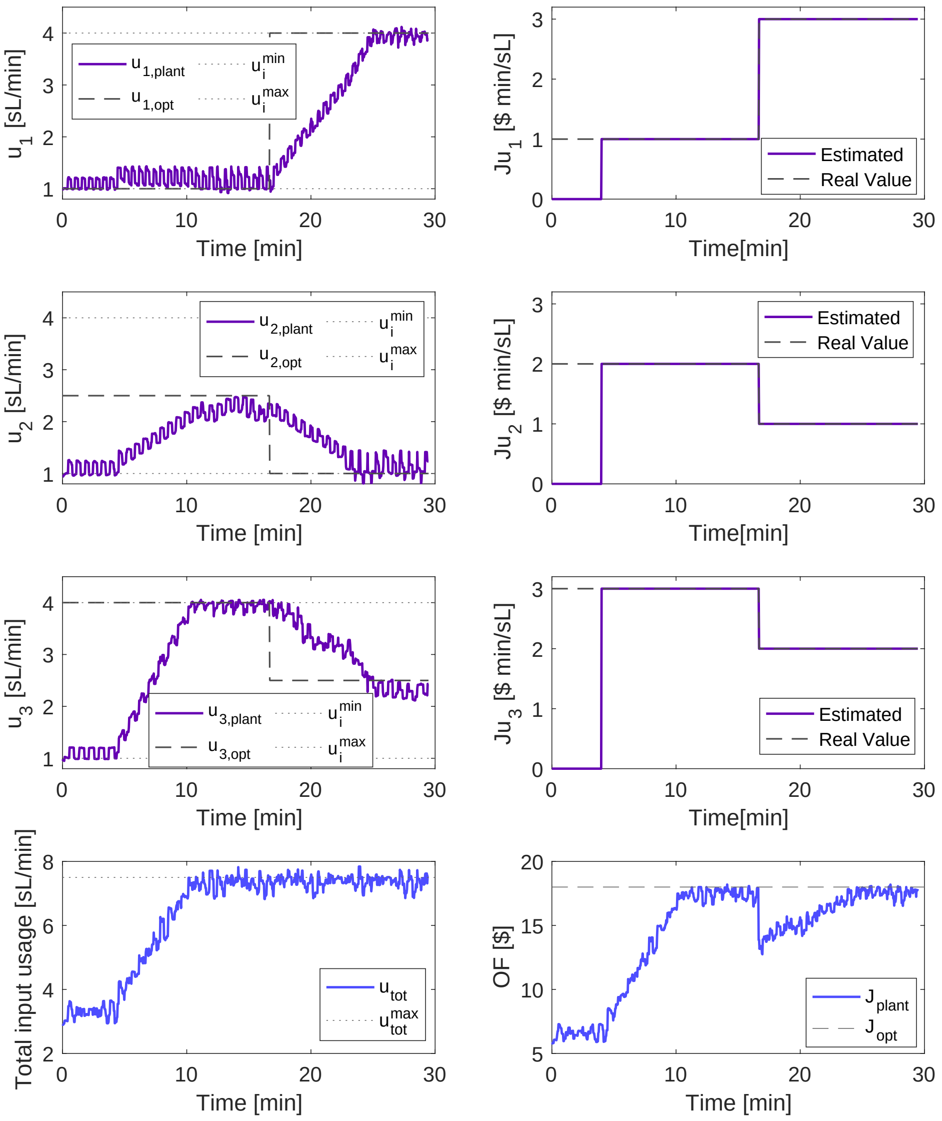


**Figure 5.2:** Results of the simple constrained optimization problem using LSE. The figure shows the inputs, objective function, the total input, and the input optimal value, in addition to the estimated gradients and their real value.

The figure shows that the gradient estimation was perfect during the whole experiment. Consequently, ESC was able to track the plant optimum. This was done without going above $u_{max}$ or below $u_{min}$, or going above $u^{tot}_{max}$. Note that, the constraints are slightly violated at some points. The reason is that ESC is an upper-layer control, it determines the set-points of the PI controller that manipulated the inputs. Since the input signal is noisy, **u** is not at the set point at all times, which can cause small violations of the constraints.

From the figure it is seen that after the second input reaches the optimum, its value is slightly decreased by ESC. By looking at the gradients it should not decrease at that point, but it is a weakness with the constraint handling strategy. The total input usage depends on the phase of the dither, this is to try and prevent that constraints disturb the dither. By looking at the plots of the inputs, it is clear that it was only partly successful. The same behavior in $u_2$ can be seen in all the following experiments, in Figure 5.3, 5.4, 6.2, 6.3 and 6.4, where the constraint handling strategy is used.

The figure shows that the inputs used a shorter time tracking the first optimum, starting from a sub-optimal point, than to find the new optimum. This is probably because when the optimum changes, at 17 min, the constraint on the maximum gas capacity is already active, resulting in more restrictive changes in the inputs, caused by the constraint handling strategy.

### 5.3.2 ARX

Figure 5.3 shows the result using ARX as the gradient estimation technique. The figure has the same setup as Figure 5.3. The gradients are estimated perfectly in this experiment as well, and the inputs are driven to their two optima without violating the constraints. The results are identical to the experiment with LSE as the gradient estimation technique. The only deviations are small differences in the inputs, due to process noise and small unmeasured disturbances in the rig. Their set-points are the same, since the estimated gradients and the integral gains are the same. As mentioned, we can see the same improper behavior in $u_2$ due to the constraint handling as it reaches the first optimum, like in the previous case.

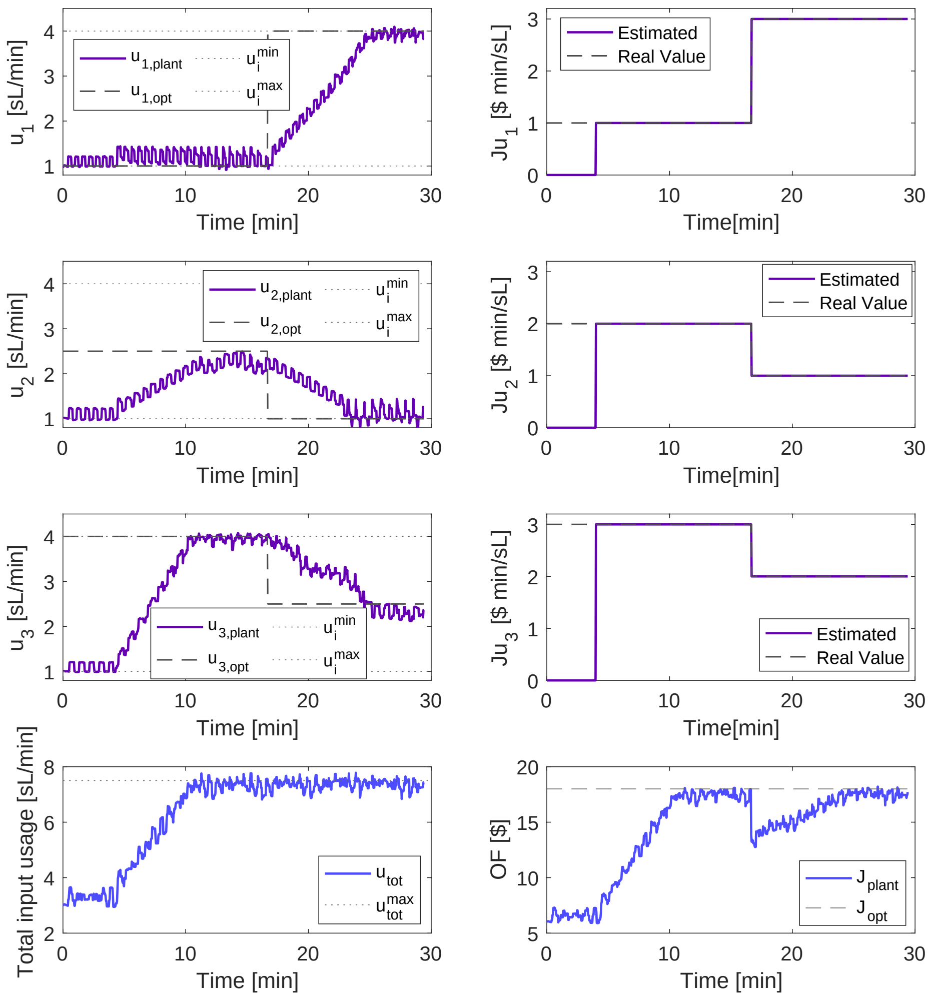


**Figure 5.3:** Results of the simple constrained optimization problem using ARX. The figure shows the inputs, objective function, the total inputs, and the input optimal value, in addition to the estimated gradients and their real value.

## 5.3.3 FFT

Figure 5.4 show the results with FFT as gradient estimation technique. The figure has the same setup as Figure 5.3, and shows that the gradient estimation is not perfect, like in the two previous experiments with LSE and ARX. However, the estimation is accurate enough and allows ESC to drive the inputs to their optimum value. Actually, as long as the order, and the sign, of the estimated gradients are correct, the inputs will be driven to their optimum. Again, the behavior in $u_2$ can be seen as it reaches its first optimum.

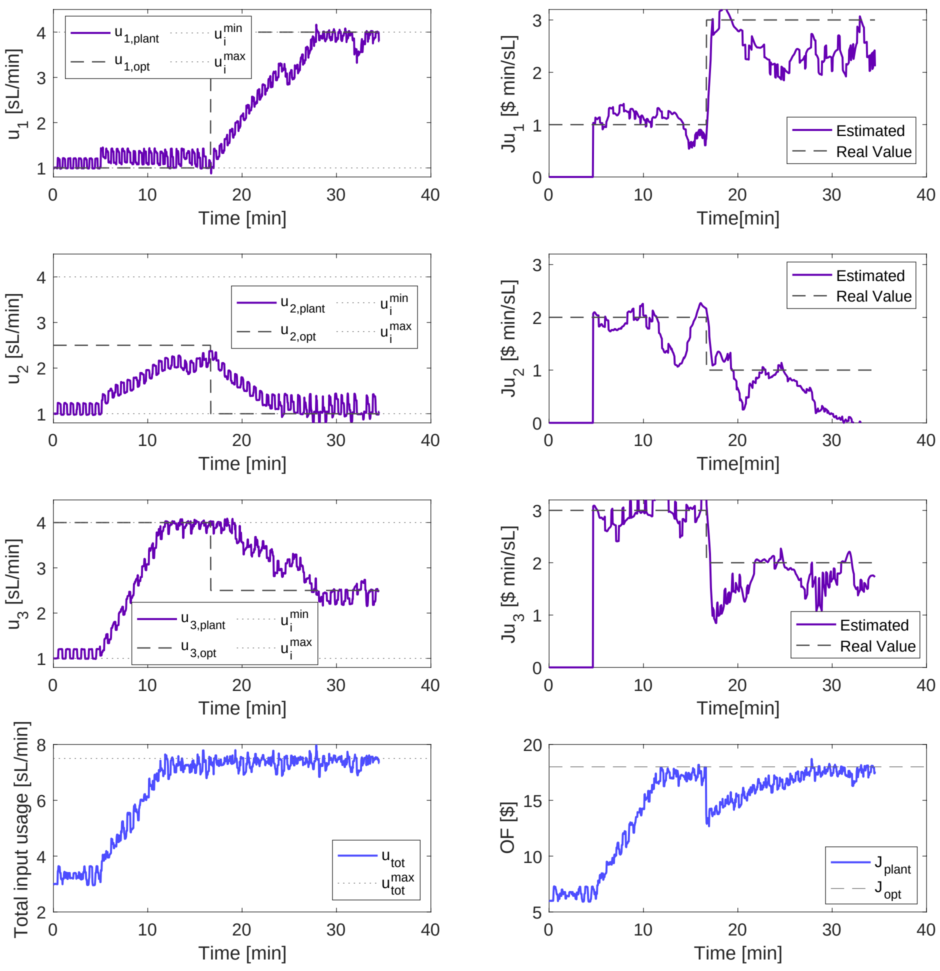


**Figure 5.4:** Result of the simple constrained optimization problem using FFT. The figure shows the inputs, objective function, the total of the inputs, and their optimums, in addition to the estimated gradients and their real value.

Compared with the two previous experiments, ESC needs more time before reaching both the first and the second optimal operating point. The first optimum is not reached for $u_2$ during the 17 minutes. The second optimum is reached around 28 minutes, while for the LSE and ARX it is reached around 26 minutes. The reason is mostly due to poorly estimated gradients, which have lower values, resulting in slower control. For the first optimum, a second reason is that the $N_{mean} = 20$ last estimated gradients are used in the control. Thus, the FFT-ESC starts 40s (20x2s) after the other two strategies.

After the disturbance, the gradient estimation accuracy degrades. One reason for this is that the control affect the periodical change of the dither more when the maximum input usage is reached. If the control did not affects the dither, one can assume that the gradient estimation would be better.

Figure 5.5 shows the non-averaged gradients. As mentioned, the sign of the estimated gradients using FFT is very sensitive to signal imperfections (noise and sinusoidal approximation by a square wave), which clearly can be seen from this figure. As seen in Case Study 1, a small amount of these negative spikes is not a significant problem for the optimization and the results, due to the averaging of the estimated gradient. However, if they occur more often they can potentially lead to totally wrong estimates of the gradients. This is the case at the end of the experiment for the second gradient. Note that $J_{u2}$ is significantly lower than the true value in Figure 5.4. However, since $u_2$ should be kept at the minimum according to the constraint handling strategy, this deviation did not affect the overall results. If it had been one of the other two gradients, the inputs would start to be driven in the wrong direction.

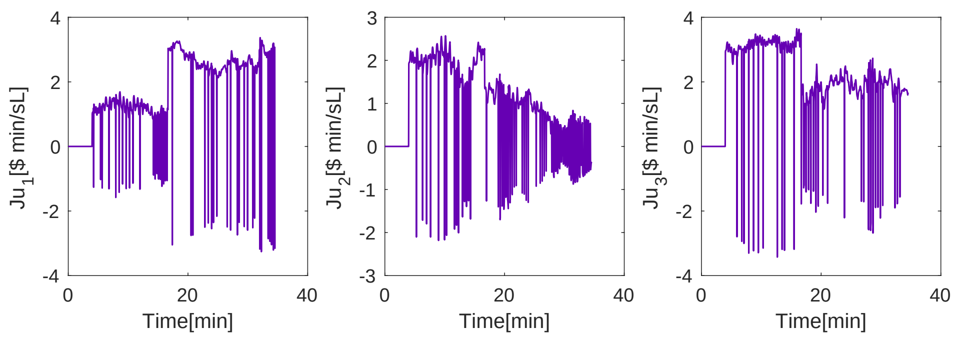


**Figure 5.5:** Non-averaged estimated gradients from experiment with FFT.

## 5.4 Conclusion

In this case study it is clear that LSE and ARX were the best approaches. Both approaches were able to provide gradient estimation with no deviation from their true value, even after the disturbance. The estimation using FFT was not as good. However, it was good enough to drive the system to its optimum, before and after the disturbance, but it took some more time to reach them.

The constraint handling and control worked, in the sense that the constraints, $u_{min}$, $u_{max}$ and $u_{max}^{tot}$, were not violated, and the system was able to track both the first and the second optimum. The constraint handling strategy did not seem to affect the gradient estimation for the ARX and LSE. On the other hand, the strategy seemed to have a negative effect on the FFT. The main problem was that the constraint handling, when the disturbance occurred, affected the periodicity in the inputs. The FFT is completely dependent on these periodic changes to estimate the gradient. Thus, a different constraint handling strategy, in which the periodicity of the inputs is prioritized, could potentially improve the accuracy of the FFT gradient estimation.

Compared to the previous case study, the gradient estimation by ARX and LSE were more accurate in this case study. The LSE and ARX are based on a linear approximation of the system to compute the gradients. Thus, it makes sense that these approaches will give a better result when approximating a linear objective function. The estimation from the FFT was better in the previous case, where the constraint handling strategy did not disturb the periodic input changes.

A performance analysis, where the losses in each experiment is compared, is shown in Figure 5.6. The FFT starting time is adjusted, such that the control starts at the same time in the analysis. The results are similar to the performance analysis for Case Study 1, where the performance of the ARX and LSE are almost identical, and better than for the FFT, even when the delay is disregarded. However, the difference in performance is larger in this case, which makes sense since the constraint handling degraded the performance of the FFT.

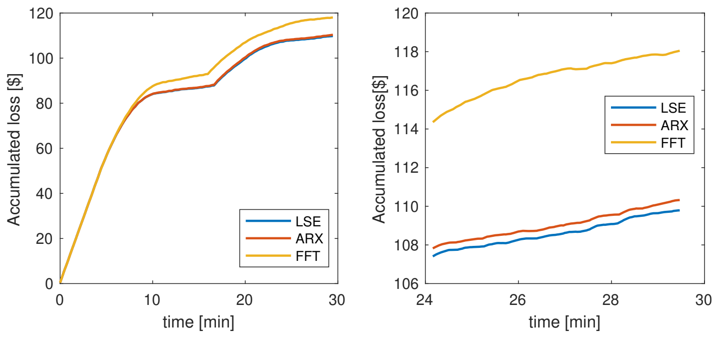


**Figure 5.6:** Performance analysis where the loss in the experiments are compared. The plot on the right side is a zoomed version of the left-hand side plot. The right hand side is zoomed in version of the left-hand side plot.

# Chapter 6

# Case Study 3: Optimizing a Gas Lifted Well Network

Case Study 3 is a more complicated and realistic optimization problem, where well production flowrates are included in the objective function. These measurements are noisier than the input measurements, which makes the gradient estimation more challenging. In this case study the real value of the gradients and the optimal value of the inputs is not known, but the objective function is chosen so that there is an indication on how they should be compared to each other. The constraints are the same as in the constrained toy example in Case Study 2.

## 6.1 Optimization Problem

$$\begin{aligned} \max_{\mathbf{u}} \quad & J = 5w_{ro_1} + 10w_{ro_2} + 20w_{ro_3} - \alpha_{gl}\sum_{i=1}^{3} u_i \\ \text{s.t} \quad & \sum_{i=1}^{3} u_i \leq 7.5 \\ & 1 \leq u_i \leq 4 \quad \forall i \in \mathcal{N} = \{1,2,3\} \end{aligned} \tag{6.1}$$

The overall objective is to maximize the production of oil, in an economical way, and the optimization problem is given by Equation 6.1, where $\alpha_{gl}$ is the cost of compressing the gas for gas lift injection and $w_{ro_i}$ is the oil flowrate from well $i$. Note that the constraints in this problem are the same as in Case Study 2. Thus the same constraint handling strategy can be used.

The inputs are still the gas lift rate in each well.

$$\mathbf{u} = [u_1 \;\; u_2 \;\; u_3]^T = [w_{gl_1} \;\; w_{gl_2} \;\; w_{gl_3}]^T$$

The real value of the gradients is not known in this case, neither is the optimum. From objective function and engineering insight, it is expected that, upon convergence, $u_1$ should be at the upper bound 4 sL/min, $u_3$ at the lower bound 1 sL/min, and the rest of the capacity (2.5 sL/min) allocated to $u_2$.

For the same reason, it is also expected that the first gradient should be the largest, and the third one the smallest. However, this order can change depending on how we approach the optimum. For example, it is known that injecting gas lift in a well will increase the flowrate from that well, up until a certain point where the pressure drop effects dominate the gain associated with lower hydrostatic pressure. Therefore, if one of the inputs approaches this turning point faster, its gradient is expected to decrease faster as well.

## 6.2 Methodology

This section presents the building blocks for the implementation of ESC in Case Study 3. This includes the dither, the gradient estimation, the control, the constraint handling, and the tuning parameters.

### 6.2.1 Gradient Estimation

As mentioned, in this case, noisy measurements of the objective function need to be handled. Figure 6.1, show the 200 measurements of the objective function in Case Study 2 and Case Study 3, before the control has started, where the only change in the inputs are from the dithers. Compared to the objective function in Case Study 2, the measurements are very noisy in this case. In Case Study 2 the input changes can clearly be observed directly from looking at the objective function, which can not be done in Case Study 3. A consequence of these noisy measurements is that it is harder to estimate the gradients.

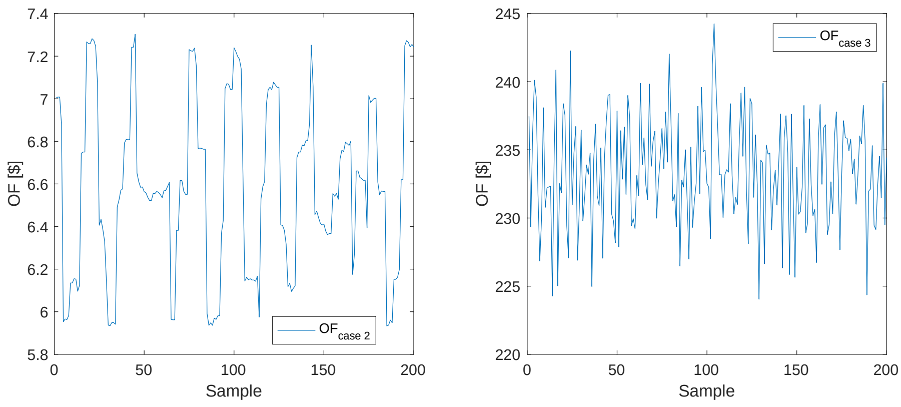


**Figure 6.1:** Comparison of the objective function measurements in Case Study 2 and 3. The plot on the left is the OF from Case Study 2 and the plot on the right is the OF from Case Study 3.

To better estimate the gradients, a larger amount of data is used, so the buffer length is remarkably increased. This is to better catch the periodical changes in the objective function from the dithers, and not only the noise. From the author's experience, having a longer buffer length can make the estimation more stable. Experiments with a shorter buffer length are shown in Appendix D.

Also, the approach used for the FFT in the previous cases, using the $N_{mean}$ previous estimated gradients, is used for the LSE and ARX approach in this case study. This is done to smooth out variations and rapid changes in the estimated gradients. Rapid changes in the gradients lead to rapid changes in the input, and more unstable control, which is undesired. Rapid changes in the input can also be bad for the gradient estimation itself. The perturbation that is added in order to estimate the gradients, can become insignificant compared to the fast changes in the inputs caused by the integral control. It is therefore important to try and smooth out these variations.

### 6.2.2 Dither, Control and Constraint Handling

The same kind of dither as in the two previous case studies was applied. The buffer length is increased, so the frequency of the perturbations are slightly different since the number of periods within the buffer, $P$, is chosen and frequencies come as a result of this.

As mentioned, the constraints in this case study are the same as in Case Study 2. Thus, the control and constraint handling are performed in the same way. The only difference is that the integral gains are slightly decreased, since we expect that the estimated gradients to have larger values in this case, which was concluded from exploratory offline experiments (not shown here).

## 6.2.3 Tuning Parameters

Table 6.1 shows the tuning parameters used in this case study, along with some operational parameters. Again, the same parameters are chosen in all three experiments.

**Table 6.1:** Tuning parameters and variables for case study 3

| Parameter | Value LSE | Value ARX | Value FFT |
|---|---|---|---|
| $\alpha_{gl}$ [\$ min/sl] | 0.5 | 0.5 | 0.5 |
| $u_{max}^{tot}$ [sL/min] | 7.5 | 7.5 | 7.5 |
| $u_{max}$ [sL/min] | 4.0 | 4.0 | 4.0 |
| $u_{min}$ [sL/min] | 1.0 | 1.0 | 1.0 |
| Sampling time [s] | 2 | 2 | 2 |
| $N$ [-] | 900 | 900 | 900 |
| $N_{mean}$ [-] | 100 | 100 | 100 |
| $a$ [sL/min] | 0.1 | 0.1 | 0.1 |
| $P_1$ [-] | 30 | 30 | 30 |
| $P_2$ [-] | 36 | 36 | 36 |
| $P_3$ [-] | 45 | 45 | 45 |
| $\omega_1$ [s$^{-1}$] | 1/60 | 1/60 | 1/60 |
| $\omega_2$ [s$^{-1}$] | 1/50 | 1/50 | 1/50 |
| $\omega_3$ [s$^{-1}$] | 1/40 | 1/40 | 1/40 |
| $K_{I_1}$ [\$$^{-1}$] | 0.005 | 0.005 | 0.005 |
| $K_{I_2}$ [\$$^{-1}$] | 0.005 | 0.005 | 0.005 |
| $K_{I_3}$ [\$$^{-1}$] | 0.005 | 0.005 | 0.005 |
| $n_a$ | - | 3 | - |
| $n_b$ | - | [2 2 2] | - |

## 6.3 Results and Discussion

In this section the results of the optimization of the gas lifted well network are presented. The result from each gradient estimation technique is presented separately. Since the real value of the gradients is unknown in this case, the gradients are estimated offline as well, by using the data measurements from the rig after running the experiment. This gives a better basis for comparing the methods when there is no real value to compare with. Also, because of the noisy measurements, it is difficult to determine if the difference in the experiments are caused by using different gradient estimation technique or the rig itself. In all experiments the inputs start from a suboptimal point, **u** = [1 1 1], which is arbitrary chosen.

### 6.3.1 LSE

The result from Case Study 3 using LSE is shown in Figure 6.2. The left side plots show the three inputs, that is the gas lift rates, and the total gaslift. The plot on the right side show the objective function and the gradients of the objective function with respect to the inputs. In the plots of the gradients, the estimated gradient from the rig is shown, in addition to the offline estimation of the gradients, using ARX and FFT. The buffer length is 900, and the $N_{mean}$ = 100 last estimated gradients are used in the control, so the control does not start before 33 minutes and 20 seconds has passed.

The figure shows that the inputs are driven to their expected optimums, but around 45 minutes there is a deviation in the expected result. For a short time period $J_{u2} > J_{u3}$, and $u_2$ increases while $u_3$ decreases. This is most likely because of a disturbance in the rig, and from the objective function, there is no visible loss as a consequence of this. The suboptimal behavior of $u_2$, as it reaches the optimum, can still be observed. The objective function increases as the control starts, which is the intention. The value of $J_{u2}$ and $J_{u3}$ decrease as the control starts and $u_2$ and $u_3$ increases, which is expected. However, this effect is more clear in the offline estimation using the ARX.

By comparing the different gradient estimation methods, the ARX and LSE have a very similar results, except from $J_{u2}$ and $J_{u3}$ the first 5 minutes after the gradient estimation starts. From the experience from Case Study 1 and Case Study 2, it is expected that the FFT would have lower values of the gradient, due to the oscillation in the estimated sign. However, this only the case for $J_{u1}$, where it either has a very similar value, or a lower value than the LSE and ARX.

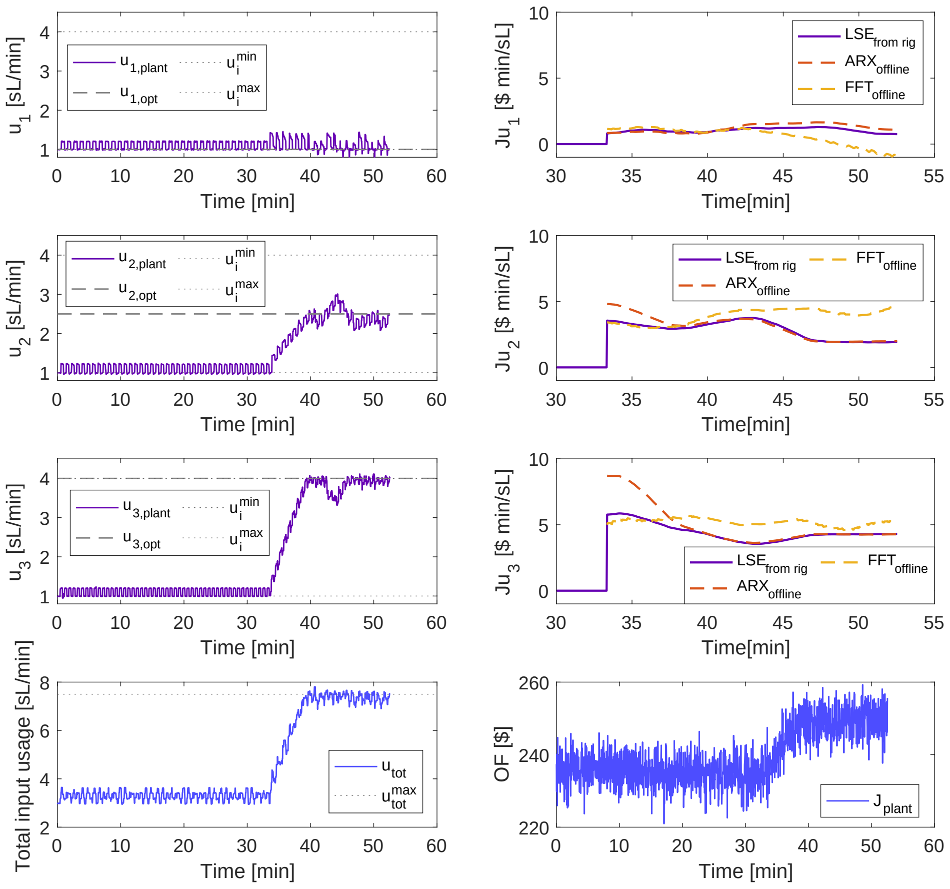


**Figure 6.2:** Result of the optimization of the gas lifted well network using LSE. The figure show the inputs, objective function, the total of the inputs, and their optimums, in addition to the estimated gradients from the rig and the offline estimation of the gradients using ARX and FFT.

### 6.3.2 ARX

Figure 6.3 shows the result using ARX as the gradient estimation approach. The figure has the same setup as Figure 6.2. In the plot of the estimated gradients, the offline estimation of the gradients using LSE and FFT is also shown. The inputs are driven to the expected values from the objective function, but again we see the improper behavior of $u_2$ caused by the constraint handling strategy as it reaches the expected optimum. The values of the gradient compared to each other is as expected, with $J_{u3} > J_{u2} > J_{u1}$. It is also clear that when the control starts, the objective function increases, which is the goal.

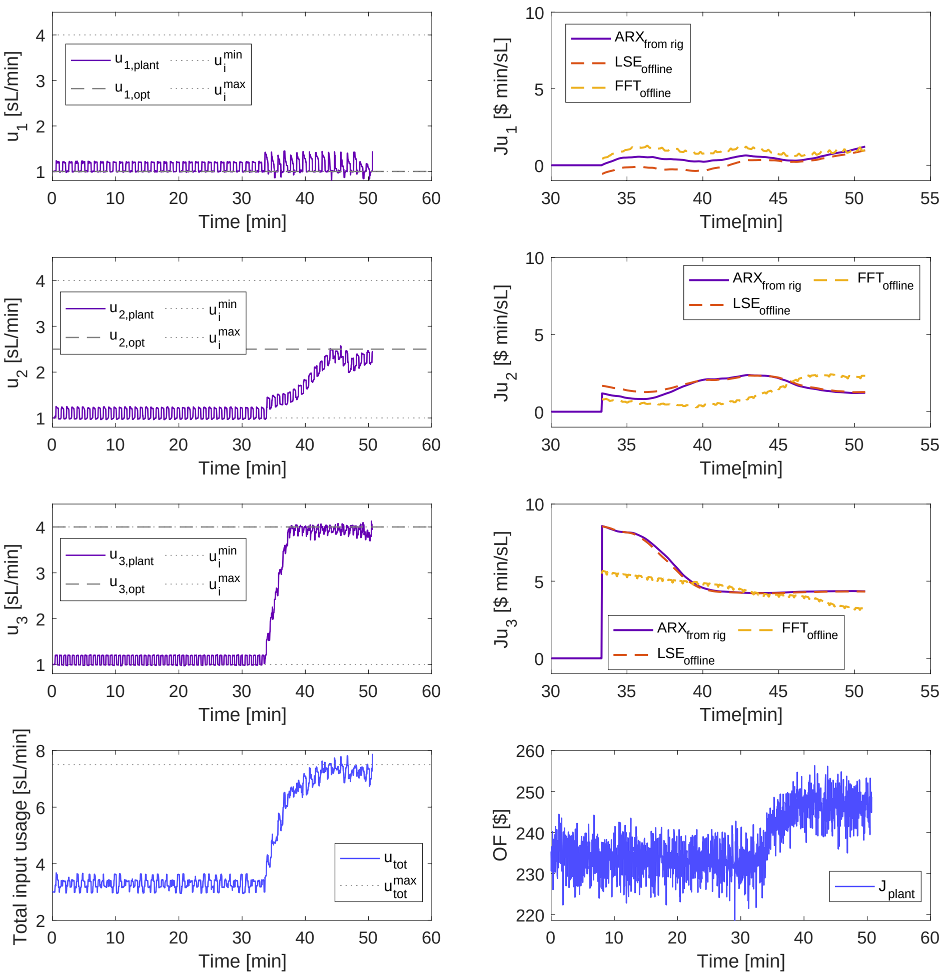


**Figure 6.3:** Result of the optimization of the gas lifted well network using ARX. The figure shows the inputs, objective function, the total of the inputs, and their optimums, in addition to the estimated gradients from the rig and the offline estimation of the gradients using LSE and FFT.

The value of $J_{u3}$ decreases as $u_3$ goes up to the maximum input value, which is expected, and is stabilized after that. The values of $J_{u1}$ and $J_{u3}$ present more oscillations. However, $J_{u2}$ seems to be decreasing when $u_2$ reaches the expected optimal value, around 45 minutes.

By comparing the estimated gradients from the different estimation techniques, the ARX and LSE give similar results. One would expect that the FFT would have lower values of the gradient, since it has negative values at some points, but this is not the case for the first gradient. Also, for $J_{u2}$ and $J_{u3}$, this is not the case at all times.

### 6.3.3 FFT

The result using FFT is shown in Figure 6.4. The figure has the same setup as Figure 6.2. The plot of the estimated gradients also shows the offline estimation of the gradients using LSE and ARX. Again, the inputs are driven to their expected values, and the order of the gradients is as expected.

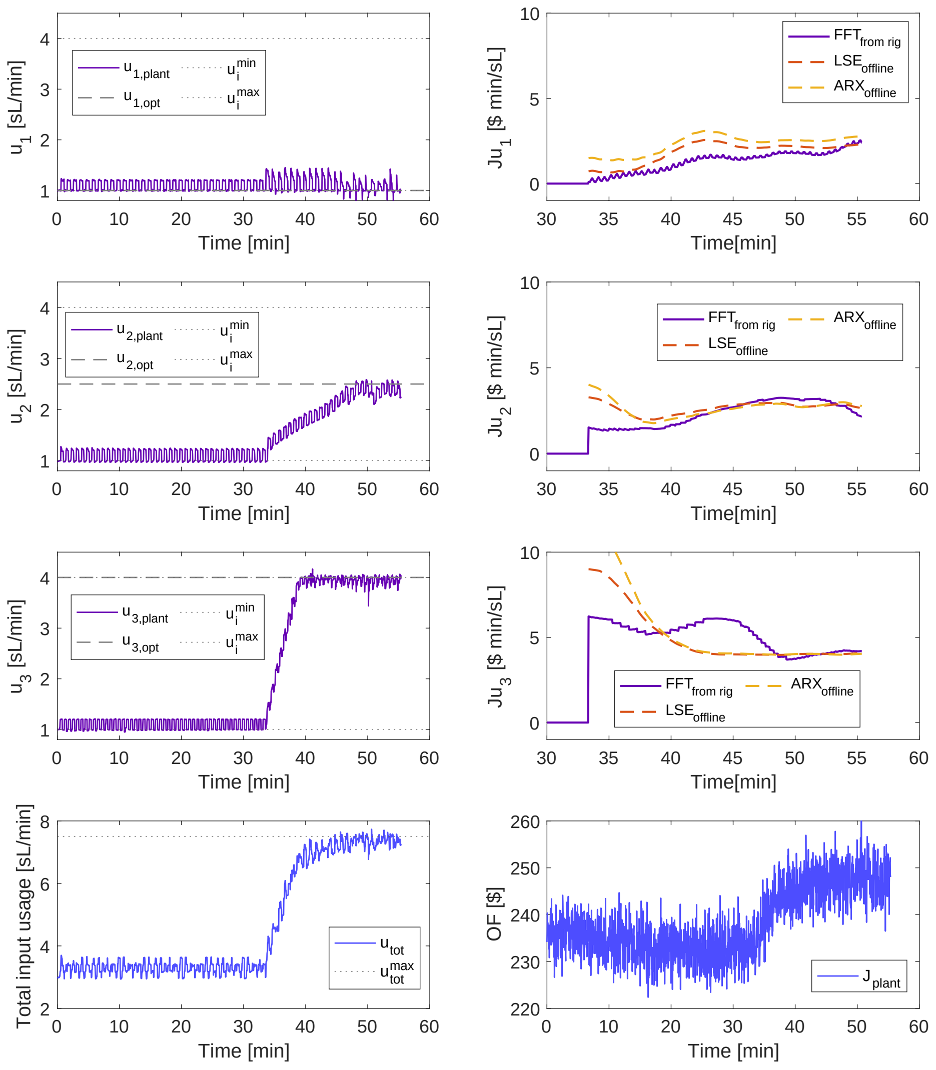


**Figure 6.4:** Result of the optimization of the gas lifted well network using FFT. The figure show the inputs, objective function, the total of the inputs, and their optimums, in addition to the estimated gradients from the rig and the offline estimation of the gradients using LSE and FFT.

The value of $J_{u3}$ does not decrease when $u_3$ is decreased, as in the other experiments. However, the value of $J_{u3}$ from the offline estimation with LSE and ARX decrease as $u_3$ increases. Nevertheless, the estimated values from all three methods converge to the same point.

Figure 6.5 shows the non-averaged estimated gradients from the FFT. It clearly has many negative spikes, which seem to occur periodically. The exception is $J_{u2}$ between 40 and 50 min, where the gradient is positive almost all the time. The periodicity of the sign oscillations seems to be connected with the frequencies of the dithers added to the inputs. $\omega_1 < \omega_2 < \omega_3$, i.e the negative spikes occurs with the highest frequency in $J_{u3}$ and with the lowest frequency for $J_{u1}$. The same periodical change is not found in the other case studies, and it is difficult to explain why this happens in this case, and not the others.

From Figure 6.5 it is also seen that the magnitude of the first gradient is larger than the second one at the end of the experiment. This means that if the signs were estimated correctly, the first input would have been increased. This can either come from a disturbance in the rig, that causes the first gradient to be larger, or it can be that the FFT estimates the magnitude of the gradient poorly. As seen from Figure 6.4, the periodical change in $u_1$ is bad from around 45 min, which can be the cause for poor estimation of the magnitude of $J_{u1}$.

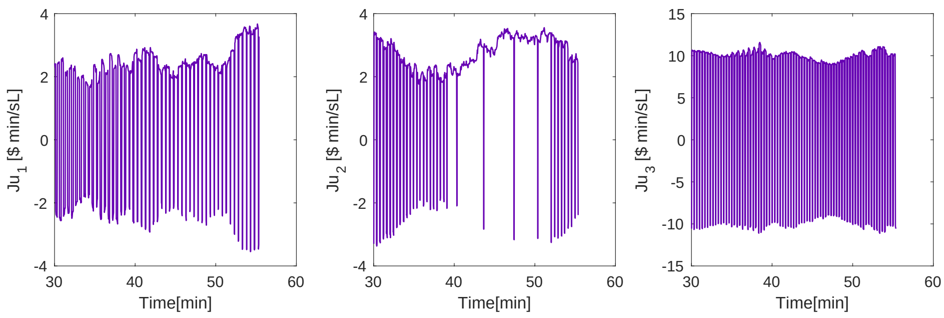


**Figure 6.5:** Non-averaged estimated gradients from experiment with FFT.

## 6.4 Conclusion

Since the real value of the gradients is unknown, it is hard to compare the gradient estimation techniques and say which performs better. However, the estimated gradients from the LSE and ARX have very similar results in all the experiments in Case Study 3, which can also be observed from Case Study 1 and 2. On the other hand, FFT has different behavior than the LSE and ARX. The reason can be the large number of negative spikes seen in the FFT estimation (Figure 6.5). However, from the same figure, it can be seen that the absolute value of the gradients from the FFT is different than the gradients estimated by the LSE and ARX in Figure 6.4. Which behavior that is most correct, is difficult to know. We know that the LSE and ARX had better results in the other case studies. However, the FFT may better with noisy measurements, since it only extracts the frequency of the dither signal, and consequently disregards the measurement noise that is represented in other frequency components.

When it comes to the control and constraint handling, the inputs are driven to the expected optimums and the constraint handling strategy performs as in Case Study 2, and is sufficient enough for the problem. Because of the long buffer length in this case, the control does not start before 33 minutes and 20 seconds has passed, which can cause a significant loss. If the controller is going to be on for a long time, this loss may be irrelevant.

Chapter 7

# Conclusions

## 7.1 Constraint Handling

From the three case studies, it is clear that ESC can be used to drive a system to its optimum, even with constraints. However, having constraints makes it more complicated. In an unconstrained problem, ESC is more straightforward to implement, only the tuning parameters (e.g. integral gain values, dither configuration, gradient estimation parameters, etc.) have to be determined. But in a constrained problem, the type of constraints and system must be taken into consideration when creating the constraint handling strategy. The more constraints and inputs, the more challenging will be to develop the strategy.

The constraint handling procedure in this thesis had some drawbacks, such as only being able to increase 2 of the inputs at the same time, and that it was designed only for estimated gradients with positive values. A better logic model could have been developed, but the constraint handling was not the main goal and, for the case studies in this thesis, its performance was adequate.

## 7.2 Gradient Estimation Techniques

When it comes to the different gradient estimation techniques the ARX and LSE had very similar performance. Despite LSE being a very simple way of estimating gradients, it worked well, also in Case Study 1 where the objective function was non-linear. A reason why the LSE and ARX performers so good, is that the system is fast. It reaches steady-state almost immediately after the signal to change the inputs is sent. Since the system response is practically instantaneous, the dynamic transition phase from one steady-state to the other does not influence the gradient estimation methods' performance. As a consequence the order of the ARX polynomials is low. Also, the ARX polynomials with order one, are similar to the linear models estimated by the LSE method. In more complicated cases, where the dynamic transitions play an important role, one could expect that the ARX would perform better than the LSE. Even though ARX is also a linear model, it is also a time series model. Therefore, it considers not only the input-output relation, but also how the time affects this relation. However, since both methods had very similar performance in the case studies, LSE is preferable due to the fact that it contains fewer tuning parameters than ARX.

Despite being a new approach to estimate gradients in ESC, the FFT gave promising results. Even though it is designed for a sinusoidal perturbation of the inputs, it worked properly with square waves dithers. The main issue with this approach was that the amplitude and the sign of the gradients were estimated separately, and because of noisy data, the sign of the gradient oscillated. This can potentially make the method unreliable since it is crucial to have the right sign of the gradients when controlling the system. However, if the sign of the gradients in the system is known, this approach can be good for estimating only the amplitude of the gradients.

From the three case studies it was clear that noisy measurements made the gradient estimation more challenging. Thus, ESC seems to be more suited for systems with less measurement noise. One possible solution could be to denoise the data measurements before estimating the gradients.

## 7.3 Implementing ESC in Real Chemical Systems

Even though ESC seems like a very simple control approach, it is not necessarily easy to implement. A clear drawback with ESC is that a significant number of parameters have to be tuned. Although there are some vague guidelines in the literature on how they should be tuned, this task is still challenging. The parameters affect each other, and with several inputs, the task becomes even harder. When trying to tune the parameters, it is smart to have offline data, or a simulation of the system, to test different tunings of the parameters before implementing.

To the best of the author’s knowledge, ESC has not been applied to any industrial chemical process operations at this point. The constant input oscillation is not something that normally is desired in chemical systems. ESC methods are more suitable for electrical systems[3], they have less inertia and less risk associated with oscillating inputs. However, if we can show that ESC can be reliably implemented and systematically improve the economic performance of different systems, it could be possible to convince plant operators and engineers to implement them in real chemical systems.

# Chapter 8

# Future Work

The gradient estimation methods studied in this thesis will be applied in the context of Modifier Adaptation (MA). MA is a hybrid (data and model-based) production optimization method, where a simplified nominal model is combined with plant measurements and plant gradient estimates such that the true system optimum can be found. Even when the simplified model presents a high degree of plant-model mismatch. According to the literature, the greatest challenge in MA applications is obtaining the plant gradients estimates. Thus, the methods developed in this thesis, which were already validated for the experimental rig, will be used in a MA implementation on the rig. This future work can be valuable to a preliminary evaluation of if MA methods can be used for optimizing subsea oil wells networks.

The main drawback of ESC implementation observed in this thesis was the necessity of applying a dither to the inputs. Despite needed for system excitation, these input oscillations can make the acceptance of ESC in the chemical engineering community harder. Therefore, as future work, ESC schemes without steady-state oscillation (e.g. [29] will be studied and implemented on the rig.

# Bibliography

[1] S. Bittanti. *Model Identification and Data Analysis*. Wiley, 2019.

[2] S. L. Brunton and L. J. N. Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamics Systems, and Control*. Cambrige University Press, 2019.

[3] S. L. Brunton, C. W. Rowley, S. R. Kulkarni, and C. Clarkson. Maximum Power Point Tracking for Photovoltaic Optimization Using Ripple-Based Extremum Seeking Control. *IEEE Transactions on Power Electronics*, 25(10), 10 2010.

[4] R. Dechter and J. Pearl. Network-Based Heuristics for Constraint-Satisfaction Problems. In *Search in Artificial Intelligence*. Springer New York, New York, NY, 1988.

[5] S. Dutta. *Optimization in Chemical Engineering*. Cambridge University Press, 2016.

[6] G. François, B. Srinivasan, and D. Bonvin. Comparison of six implicit real-time optimization schemes. 2013.

[7] M. Guay, D. Dochain, and M. Perrier. Adaptive extremum-seeking control of non-isothermal continuous stirred tank reactors. *Chemical Engineering Science*, 60(13), 7 2005.

[8] Hsin-Hsiung Wang, S. Yeung, and M. Krstic. Experimental application of extremum seeking on an axial-flow compressor. *IEEE Transactions on Control Systems Technology*, 8(2), 2000.

[9] N. Hudon, M. Perrier, M. Guay, and D. Dochain. Adaptive extremum seeking control of a non-isothermal tubular reactor with unknown kinetics. *Computers & Chemical Engineering*, 29(4), 2005.

[10] B. G. Hunnekens, M. A. Haring, N. Van De Wouw, and H. Nijmeijer. A dither-free extremum-seeking control approach using 1st-order least-squares fits for gradient estimation. *53rd IEEE Conference on Decision and Control*, pages 2679–2684, 2014.

[11] D. Krishnamoorthy. On the Design and Analysis of Multivariable Extremum Seeking Control using Fast Fourier Transform. Technical report, 2021.

[12] D. Krishnamoorthy, K. Fjalestad, and S. Skogestad. Optimal operation of oil and gas production using simple feedback control structures. *Control Engineering Practice*, 91, 2019.

[13] D. Krishnamoorthy, B. Foss, and S. Skogestad. Real-time optimization under uncertainty applied to a gas liftedwell network. *Processes*, 4(4), 2016.

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

[15] D. Krishnamoorthy and S. Skogestad. A fast and robust extremum seeking scheme using transient measurements. Technical report, 2019.

[16] M. Krstich and H.-H. Wang. Stability of extremum seeking feedback for general nonlinear dynamic systems. In *Automatica*, volume 36, pages 595–601, 2000.

[17] A. Marchetti, B. Chachuat, and D. Bonvin. Modifier-adaptation methodology for real-time optimization. *Industrial and Engineering Chemistry Research*, 48(13):6022–6033, 2009.

[18] N. Marcos, M. Guay, D. Dochain, and T. Zhang. Adaptive extremum-seeking control of a continuous stirred tank bioreactor with Haldane's Kinetics. *Journal of Process Control*, 14(3), 2004.

[19] E. Martínez. Extremum-seeking control of redox processes in wastewater chemical treatment plants. 2007.

[20] J. Matias and J. Jäschke. Online model maintenance in real-time optimization methods. *Computers and Chemical Engineering*, 145, 2021.

[21] A. D. Quelhas, N. J. C. de Jesus, and J. C. Pinto. Common vulnerabilities of RTO implementations in real chemical processes. *Canadian Journal of Chemical Engineering*, 91(4):652–668, 2013.

[22] D. E. Seborg, T. F. Edgar, D. A. Mellichamp, and F. J. Doyle III. *Process Dynamics and Control*. John Wiley & Sons, Inc, 3rd edition, 2011.

[23] S. Skogestad. Plantwide control: the search for the self-optimizing control structure. *Journal of Process Control*, 10(5):487–507, 2000.

[24] J. Straus, D. Krishnamoorthy, and S. Skogestad. On combining self-optimizing control and extremum-seeking control – Applied to an ammonia reactor case study. *Journal of Process Control*, 78:78–87, 2019.

[25] M. Titica, D. Dochain, and M. Guay. Adaptive Extremum Seeking Control of Fed-Batch Bioreactors. *European Journal of Control*, 9(6), 2003.

[26] M. Titica, D. Dochain, and M. Guay. Real-Time Optimization of Fed-Batch Bioreactors via Adaptive Extremum-Seeking Control. *Chemical Engineering Research and Design*, 81(9), 2003.

[27] O. Trollberg. Extremum Seeking Control Applied to a Deammonification Process . Technical report, Uppsala universitet, Uppsala, 2011.

[28] O. Trollberg, B. Carlsson, and E. W. Jacobsen. Extremum seeking control of the CANON process—Existence of multiple stationary solutions. *Journal of Process Control*, 24(2), 2014.

[29] L. Wang, S. Chen, and H. Zhao. A novel fast extremum seeking scheme without steady-state oscillation. In *Proceedings of the 33rd Chinese Control Conference*. IEEE, 2014.

[30] H.-H. Wangr, M. Krstic, and G. Bastin. Optimizing Bioreactors BY Extremum Seeking. *Int. J. Adapt Control Signal Process*, 13:651–669, 1999.

[31] B. Zitte, B. Hamroun, F. Couenne, and I. Pitault. Extremum-Seeking Based Distributed Optimization of Heat Exchangers Network. *IFAC-PapersOnLine*, 51(23), 2018.

Appendix A

# MATLAB Codes in the Lab-Rig

The experiemntal lab rig uses LabVIEW to have an interface between the rig and MATLAB. The MATLAB codes used in the experimental lab to estimate the gradients and calculate new set-points are presented in this appendix. This includes a main file, an initialization file and files for the gradient estimation techniques.

## A.1 Main File

This is the main file ran by LabVIEW. There is one main file for the unconstrained case, `LabViewMain_unconstrained.m`, and one for the constrained case, `LabViewMain_constrained`

### LabViewMain_unconstrained.m

```
% Main program
% Run Initialization file first

%%%%%%%%%%%%%%%%%%%%%%%
% Get Variables
%%%%%%%%%%%%%%%%%%%%%%%
% % setpoint of liquid flowrate
% ys1 = vector(1);
% ys2 = vector(2);
% ys3 = vector(3);
```

```matlab

% disturbances
% Opening —> 0 (fully closed) to 1 (fully open)
% measurement already in between [0, 1]
cv101 = P_vector(1);
cv102 = P_vector(2);
cv103 = P_vector(3);

%pump rotation [%]
pRate = P_vector(4);

% always maintain the inputs greater than 0.5
%inputs computed in the previous MPC iteration
% Note that the inputs are the setpoints to the gas flowrate PID's
% fic104sp = O_vector(1);
% fic105sp = O_vector(2);
% fic106sp = O_vector(3);
fic104sp = P_vector(5);
fic105sp = P_vector(6);
fic106sp = P_vector(7);
%current inputs of the plant
u0old=[P_vector(5);P_vector(6);P_vector(7)];

% cropping the data vector
nd = size(I_vector,2);
dataCrop = (nd - BufferLength + 1):nd;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MOST RECENT MEASUREMENT IS THE LAST ONE! %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% liquid flowrates [L/min]
fi101 = I_vector(1,dataCrop);
fi102 = I_vector(2,dataCrop);
fi103 = I_vector(3,dataCrop);

% actual gas flowrates [sL/min]
fic104 = I_vector(4,dataCrop);
fic105 = I_vector(5,dataCrop);
fic106 = I_vector(6,dataCrop);

% pressure @ injection point [mbar g]
pi105 = I_vector(7,dataCrop);
pi106 = I_vector(8,dataCrop);
pi107 = I_vector(9,dataCrop);

% reservoir outlet temperature [oC]
```

```
ti101 = I_vector(10,dataCrop);
ti102 = I_vector(11,dataCrop);
ti103 = I_vector(12,dataCrop);

% DP @ erosion boxes [mbar]
dp101 = I_vector(13,dataCrop);
dp102 = I_vector(14,dataCrop);
dp103 = I_vector(15,dataCrop);

% top pressure [mbar g]
% for conversion [bar a]-->[mbar g]
% ptop_n = ptop*10^-3 + 1.01325;
pi101 = I_vector(16,dataCrop);
pi102 = I_vector(17,dataCrop);
pi103 = I_vector(18,dataCrop);

% reservoir pressure [bar g]
% for conversion [bar g]-->[bar a]
% ptop_n = ptop + 1.01325;
pi104 = I_vector(19,dataCrop);

dw = size(pi104,2);


if ~exist('kk','var')
    kk = 1;
    checkPoint = 0;
    Ju_hat_buffer = [];
    u_buffer = [];
    w_ro_buffer = [];
else
    kk = kk + 1;
    checkPoint = 1;
end

if kk>N
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Gradient Estimation - Chose FFT, ARX or LSE %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Ju_hatkk = FFT_gradient(J_buffer, u_buffer, w);
    %Ju_hatkk = ARX_gradient(J_buffer', u_buffer', na, nb, nk);
    %Ju_hatkk = LSE_gradient(J_buffer, u_buffer);

    %If the gradient estimation does not give a result
    if isnan(Ju_hatkk)==1
        Ju_hatkk = [0; 0; 0; 0]; % 4th one is the bias term
    end

    Ju_hat_buffer=[Ju_hat_buffer, [Ju_hatkk(1:3)]];
    if size(Ju_hat_buffer,2)>N_mean
        Ju_hat_buffer = Ju_hat_buffer(:,2:end);
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Optimizing SS model %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%

    if kk>(N+N_mean)
    %Ju_hat = Ju_hatkk;          %LSE and ARX
    Ju_hat = mean(Ju_hat_buffer'); %FFT

    ficsp = [fic104sp,fic105sp,fic106sp];
    C_vector(1) = ficsp(1) + KI(1) * Ju_hat(1);
    C_vector(2) = ficsp(2) + KI(2) * Ju_hat(2);
    C_vector(3) = ficsp(3) + KI(3) * Ju_hat(3);

    else
        C_vector = [fic104sp;fic105sp;fic106sp];
    end

else
    C_vector = [fic104sp;fic105sp;fic106sp];
    Ju_hat = [0;0;0];
    Ju_hatkk = [0;0;0];
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Adding dither signal %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

input = [C_vector(1) + dither1(kk), C_vector(2)+ dither2(kk), C_vector(3)
    + dither3(kk)];

O_vector = input;

%%%%%%%%%%%%%%%%%%%%
%Send Variable %
%%%%%%%%%%%%%%%%%%%%
SS = kk;
if ~isnan(Ju_hat) % gradient estimation OK
    Estimation = 1;
else
    % gradient estimation with problems
    Estimation = 0;
end
Optimization = 1;
Parameter_Estimation = [Ju_hat(1),Ju_hat(2),Ju_hat(3),Ju_hatkk(1),Ju_hatkk(2),Ju_hatkk(3)];
State_Variables_Estimation = [fi101(end),fi102(end),fi103(end),fic104(end),fic105(end),fic106(end)];
State_Variables_Optimization = [fi101(end-1),fi102(end-1),fi103(end-1),fic104(end-1),fic105(end-1),fic106(end-1)];
Optimized_Air_Injection = [O_vector(1),O_vector(2),O_vector(3)];

%%%%%%%%%%%%%%%%%%%%%%%
% Create the buffer %
%%%%%%%%%%%%%%%%%%%%%%%
u_buffer = [u_buffer, [State_Variables_Estimation(4); State_Variables_Estimation(5); State_Variables_Estimation(6)]];
w_ro_buffer = [w_ro_buffer, [State_Variables_Estimation(1); State_Variables_Estimation(2); State_Variables_Estimation(3)]];

if size(u_buffer ,2)>N
    u_buffer = u_buffer(:, 2:end);
    w_ro_buffer = w_ro_buffer(:,2:end);
end

J_buffer = -((u_buffer(1,:)-3).^2 + (u_buffer(2,:)-2.5).^2 + (u_buffer(3,:)-2).^2);
Result = J_buffer(end);
```

## LabViewMain_constrained.m

```
% Main program
% Run Initialization file first

%%%%%%%%%%%%%%%%%%
% Get Variables
%%%%%%%%%%%%%%%%%%
% % setpoint of liquid flowrate
% ys1 = vector(1);
% ys2 = vector(2);
% ys3 = vector(3);

% disturbances
% Opening --> 0 (fully closed) to 1 (fully open)
% measurement already in between [0, 1]
cv101 = P_vector(1);
cv102 = P_vector(2);
```

```
cv103 = P_vector(3);

%pump rotation [%]
pRate = P_vector(4);

% always maintain the inputs greater than 0.5
%inputs computed in the previous MPC iteration
% Note that the inputs are the setpoints to the gas flowrate PID's
% fic104sp = O_vector(1);
% fic105sp = O_vector(2);
% fic106sp = O_vector(3);
fic104sp = P_vector(5);
fic105sp = P_vector(6);
fic106sp = P_vector(7);
%current inputs of the plant
u0old=[P_vector(5);P_vector(6);P_vector(7)];

% cropping the data vector
nd = size(I_vector,2);
dataCrop = (nd - BufferLength + 1):nd;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MOST RECENT MEASUREMENT IS THE LAST ONE! %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% liquid flowrates [L/min]
fi101 = I_vector(1,dataCrop);
fi102 = I_vector(2,dataCrop);
fi103 = I_vector(3,dataCrop);

% actual gas flowrates [sL/min]
fic104 = I_vector(4,dataCrop);
fic105 = I_vector(5,dataCrop);
fic106 = I_vector(6,dataCrop);

% pressure @ injection point [mbar g]
pi105 = I_vector(7,dataCrop);
pi106 = I_vector(8,dataCrop);
pi107 = I_vector(9,dataCrop);

% reservoir outlet temperature [oC]
ti101 = I_vector(10,dataCrop);
ti102 = I_vector(11,dataCrop);
ti103 = I_vector(12,dataCrop);

% DP @ erosion boxes [mbar]
dp101 = I_vector(13,dataCrop);
```

```
dp102 = I_vector(14,dataCrop);
dp103 = I_vector(15,dataCrop);

% top pressure [mbar g]
% for conversion [bar a]-->[mbar g]
% ptop_n = ptop*10^-3 + 1.01325;
pi101 = I_vector(16,dataCrop);
pi102 = I_vector(17,dataCrop);
pi103 = I_vector(18,dataCrop);

% reservoir pressure [bar g]
% for conversion [bar g]-->[bar a]
% ptop_n = ptop + 1.01325;
pi104 = I_vector(19,dataCrop);

dw = size(pi104,2);


% for the initial run, after the controller is activated
% to find the right index of the pre-created array dither
if ~exist('kk','var')
    kk = 1;
    checkPoint = 0;
    Ju_hat_buffer = [];
    u_buffer = [];
    w_ro_buffer = [];
else
    kk = kk + 1;
    checkPoint = 1;
end

%maximum input usage dependant on the dither
w_gl_tot= w_gl_TOT - (dither1(kk) + dither2(kk) + dither3(kk))/2;

if kk>N
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % GRADIENT ESTIMATION - Chose FFT, ARX or LSE
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Ju_hatkk = FFT_gradient(J_buffer, u_buffer, w);
    %Ju_hatkk = ARX_gradient(J_buffer', u_buffer', na, nb, nk);
    Ju_hatkk = LSE_gradient(J_buffer, u_buffer);

    %If the gradient estimation does not give a result
    if isnan(Ju_hatkk)==1
        Ju_hatkk = [0; 0; 0; 0]; % 4th one is the bias term
    end
```

```matlab

    %Gradient buffer
    Ju_hat_buffer=[Ju_hat_buffer, [Ju_hatkk(1:3)]];
    if size(Ju_hat_buffer,2)>N_mean
        Ju_hat_buffer = Ju_hat_buffer(:,2:end);
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Optimizing SS model %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if kk>(N+N_mean)
    Ju_hat = mean(Ju_hat_buffer');

    %Find max index with max flow--> this one should be used for
    %the active constraint
    ficsp = [fic104sp,fic105sp,fic106sp];
    idx = [1 2 3]; %Index list
    [M,I]=max([fic104sp,fic105sp,fic106sp]);
    idx = idx(idx~=I); %Remove I from index list

    %Check which of the 2 remaning that has largest grad(abs value)
    [M2, I2] = max([Ju_hat(idx(1)), Ju_hat(idx(2))]);
    I2 = idx(I2); %actual index of the input with largest gradient
    idx = idx(idx~=I2);

    % I: index with largest flow
    % I2: index with largest abs value of grad, of the other 2
    % idx: index with smallest abs value of grad, of other 2

    C_vector(I) = ficsp(I) + KI(I) * Ju_hat(I);
    if Ju_hat(I)>Ju_hat(idx)
        C_vector(I2) = ficsp(I2) + KI(I2)* (Ju_hat(I2) - Ju_hat(idx));
        C_vector(idx) = ficsp(idx) + KI(idx)*(Ju_hat(idx) - Ju_hat(I) );
    else
        C_vector(I2) = ficsp(I2) + KI(I2)* (Ju_hat(I2) - Ju_hat(I));
        C_vector(idx) = ficsp(idx) + KI(idx)*(Ju_hat(idx) - Ju_hat(I2) );

    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Constarint handling %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for i=1:3
            if C_vector(i)<w_gl_min + dither(kk,i)
                C_vector(i)=w_gl_min + dither(kk,i);
            elseif C_vector(i) > w_gl_max - dither(kk,i)
```

```
                C_vector(i) = w_gl_max - dither(kk,i);
            end
    end

    if Ju_hat(I) > Ju_hat(I2)
        if C_vector(I2)+C_vector(idx)>w_gl_tot-C_vector(I)
                tot = C_vector(I2)+C_vector(idx);
                subs = tot-(w_gl_tot-C_vector(I));
                if C_vector(idx)>subs+(w_gl_min +dither(kk-1,idx))
                    C_vector(idx) = C_vector(idx)-subs;
                else
                    C_vector(idx) = w_gl_min + dither(kk-1,idx);
                    C_vector(I2) = w_gl_tot - C_vector(I) - C_vector(idx);
                end
        end
    else
        if C_vector(I2)+C_vector(idx)>w_gl_tot-(w_gl_min + dither(kk-1,I))
                tot = C_vector(I2)+C_vector(idx);
                subs = w_gl_tot-w_gl_min;
                if C_vector(idx)>subs+(w_gl_min + dither(kk-1,idx))
                    C_vector(idx) = C_vector(idx)-subs;
                else
                    C_vector(idx) = w_gl_min + dither(kk-1,idx);
                    C_vector(I2) = w_gl_tot - C_vector(I) - C_vector(idx);
                end
        end
    end

    if sum(C_vector)>w_gl_tot
            C_vector(I) = w_gl_tot - C_vector(I2)-C_vector(idx);
    end

    else
        C_vector = [fic104sp;fic105sp;fic106sp];
    end

else
    C_vector = [fic104sp;fic105sp;fic106sp];
    Ju_hat = [0;0;0];
    Ju_hatkk = [0;0;0];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Adding dither signal %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
input = [C_vector(1) + dither1(kk), C_vector(2)+ dither2(kk), C_vector(3)
```

```
    + dither3(kk)];
O_vector = input;

%%%%%%%%%%%%%%%%%%%%
%Send Variable %
%%%%%%%%%%%%%%%%%%%%
SS = kk;
if ~isnan(Ju_hat) % gradient estimation OK
    Estimation = 1;
else
    % gradient estimation with problems
    Estimation = 0;
end
Optimization = 1;
Parameter_Estimation = [Ju_hat(1),Ju_hat(2),Ju_hat(3),Ju_hatkk(1),Ju_hatkk
    (2),Ju_hatkk(3)];
State_Variables_Estimation = [fi101(end),fi102(end),fi103(end),fic104(end)
    ,fic105(end),fic106(end)];
State_Variables_Optimization = [fi101(end-1),fi102(end-1),fi103(end-1),
    fic104(end-1),fic105(end-1),fic106(end-1)];
Optimized_Air_Injection = [O_vector(1),O_vector(2),O_vector(3)];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Creating the buffers %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
u_buffer = [u_buffer, [State_Variables_Estimation(4);
    State_Variables_Estimation(5); State_Variables_Estimation(6)]];
w_ro_buffer = [w_ro_buffer, [State_Variables_Estimation(1);
    State_Variables_Estimation(2); State_Variables_Estimation(3)]];
if size(u_buffer ,2)>N
    u_buffer = u_buffer(:, 2:end);
    w_ro_buffer = w_ro_buffer(:,2:end);
end
%Case 2
%J_buffer = 1*u_buffer(1,:) + 2*u_buffer(2,:) + 3*u_buffer(3,:);

%Case 3
J_buffer = 5*w_ro_buffer(1,:) + 10*w_ro_buffer(2,:) + 20*w_ro_buffer(3,:)
    - 0.5*(u_buffer(1,:) + u_buffer(2,:) + u_buffer(3,:));

Result = J_buffer(end);
```

## A.2 Initialization File

The initialization file is ran before the main file. The file sets the paramaters used in the experiment and creates the dither signals.

```
%clear
%clc
%GAINS
KI1 = 0.003;%
KI2 = 0.003;
KI3 = 0.003;
KI = [KI1, KI2, KI3];

%Constraints
w_gl_min=1;
w_gl_max=4;
w_gl_TOT=7.5;
a = 0.2;

%BUFFER PARAMETERS
N = 900;          %size buffer
N_mean = 100;     %size gradient buffer

%ARX-param
na =3;
nb=[2, 2, 2];
nk=[0 0 0];

%number of periods within the buffer (P)
w1=10*3;
w2=12*3;
w3=15*3;
w = [w1 w2 w3];

%CREATING DITHER SIGNAL
sq_wave1 = zeros(1,N);
sq_wave2 = zeros(1,N);
sq_wave3 = zeros(1,N);

for i=1:2*w1
    if (-1)^i == -1
        sq_wave1(round(N/(2*w1)*i)) = 1 ;
    else
        sq_wave1(round(N/(2*w1)*i)) = -1 ;
    end
```

```
end
for i=1:2*w2
    if (-1)^i == -1
        sq_wave2(round(N/(2*w2)*i)) = 1 ;
    else
        sq_wave2(round(N/(2*w2)*i)) = -1 ;
    end
end
for i=1:2*w3
    if (-1)^i == -1
        sq_wave3(round(N/(2*w3)*i)) = 1 ;
    else
        sq_wave3(round(N/(2*w3)*i)) = -1 ;
    end
end

sq_wave1 = a*repmat(sq_wave1,1,100);
sq_wave2 = a*repmat(sq_wave2,1,100);
sq_wave3 = a*repmat(sq_wave3,1,100);

dither1 = repmat(sq_wave1,[1, 100]);
dither2 = repmat(sq_wave2,[1, 100]);
dither3 = repmat(sq_wave3,[1, 100]);

dither = [dither1', dither2', dither3'];
```

## A.3 Gradient Estimation Files

There is one file for each estimation technique, `LSE_gradient.m`, `ARX_gradient.m` and `FFT_gradient.m`, in addition to `movingavg.m` used in the FFT approach.

### LSE_gradient.m

```
function Ju_hat = LSE_gradient(J_buffer, u_buffer)
N_buffer = length(J_buffer);
x = u_buffer';
y = J_buffer';

phi = [x ones(N_buffer,1)];
theta = inv(phi'*phi)*phi'*y;

```

```
Ju_hat = theta;
```

## ARX_gradient.m

```
function Ju_hat = ARX_gradient(J_buffer, u_buffer, na, nb, nk)

data = iddata(J_buffer-mean(J_buffer), u_buffer-mean(u_buffer),1);
m_arx = arx(data,[na, nb, nk]);
%Get the state-space model
sysd = idss(m_arx); %dymanic system
sysc = d2c(sysd); %continous sytem

A = sysc.A;
B = sysc.B;
C = sysc.C;
D = sysc.D;
Ju_hat = -C*pinv(A)*B + D;
Ju_hat = Ju_hat';
```

## FFT_gradient.m

```
function Ju_hat = FFT_gradient(J_buffer, u_buffer, w)
    N = length(J_buffer);
    L = N/4;
    NFFT = N;
    Fs = NFFT;
    w1 = w(1);
    w2 = w(2);
    w3 = w(3);

    J_mean = J_buffer - movingavg(J_buffer,L/2,0);
    u1_mean = u_buffer(1,:) - movingavg(u_buffer(1,:),L/2,0);
    u2_mean = u_buffer(2,:) - movingavg(u_buffer(2,:),L/2,0);
    u3_mean = u_buffer(3,:) - movingavg(u_buffer(3,:),L/2,0);

    Jhat = fft(J_mean)/N;
    u1hat = fft(u1_mean)/N;
    u2hat = fft(u2_mean)/N;
    u3hat = fft(u3_mean)/N;

    f = Fs/2*linspace(0,1,NFFT/2+1);
    i1 = find(f==w1);
    i2 = find(f==w2);
    i3 = find(f==w3);

    Jmag = 2*abs(Jhat(1:NFFT/2+1));
    Jph = angle(Jhat)*180/pi;

    U1mag = 2*abs(u1hat(1:NFFT/2+1));
    U1ph = angle(u1hat)*180/pi;
    U2mag = 2*abs(u2hat(1:NFFT/2+1));
    U2ph = angle(u2hat)*180/pi;
    U3mag = 2*abs(u3hat(1:NFFT/2+1));
    U3ph = angle(u3hat)*180/pi;

    Ju_hat1 = sign(Jph(i1).*U1ph(i1))*Jmag(i1)/U1mag(i1);
    Ju_hat2 = sign(Jph(i2).*U2ph(i2))*Jmag(i2)/U2mag(i2);
    Ju_hat3 = sign(Jph(i3).*U3ph(i3))*Jmag(i3)/U3mag(i3);

    Ju_hat = [Ju_hat1; Ju_hat2; Ju_hat3];
```

## movingavg.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Author: Dinesh Krishnamoorthy %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = movingavg(x,n,order)
warning off
if nargin<3
    order = 0;
end
ne = numel(x);
n2 = round(n/2);
y = zeros(size(x));
for i=1:ne
    if i<n2 % before half of the window
        X  = x(1:n);
        ie = i; % Evaluation point
    elseif i>=(ne-n2) % less than half of the window left
        X  = x(end-n+1:end);
        ie = i-(ne-n);
        %ie = i-(ne-n2);
    else
        X  = x(i-n2+1:i+n2);
        ie = n2;
    end
    if order == 0
        y(i) = mean(X);
    else
        p = polyfit([1:n]',X(:),order);
        y(i) = polyval(p,ie);
    end
end
warning on
```

Appendix B

# MATLAB Codes for Reading and Plotting the Results From the Lab-Rig

The codes in this appendix are the code for reading the data and plotting the result from the experimental lab rig. There is one file for each case study, `PlottingDataCase1.m`, `PlottingDataCase2.m` and `PlottingDataCase3.m`

## PlottingDataCase1.m

```
clear
clc
close all

%% Import the data
%% LSE
i = 826;
name = [pwd,'/2021-04-28_135606_LSE_case1_test1Matlab.txt'];
%% ARX
%i = 1271;
%name = [pwd,'/2021-04-28_141632_ARX_case1_test1Matlab.txt'];
%% FFT
%i = 2984;
```

```
%name = [pwd,'/2021-04-28_152159_FFT_case1_test2Matlab.txt'];

opts2 = delimitedTextImportOptions("NumVariables", 25);
opts2.DataLines = [2, Inf];
opts2.Delimiter = " ";
opts2.VariableNames = ["SS", "EstimationError", "OptmizationError", "OF",
    "Thetaw1", "Thetaw2", "Thetaw3", "Valvew1", "Valvew2", "Valvew3", "
    WroEstimatedw1", "WroEstimatedw2", "WroEstimatedw3", "PrhEstimatedw1",
     "PrhEstimatedw2", "PrhEstimatedw3", "WroOptimzedw1", "WroOptimzedw2",
     "WroOptimzedw3", "PrhOptimzedw1", "PrhOptimzedw2", "PrhOptimzedw3", "
    u0w1", "u0w2", "u0w3"];
opts2.VariableTypes = ["double", "double", "double", "double", "double", "
    double", "double", "double", "double", "double", "double", "double", "
    double", "double", "double", "double", "double", "double", "double", "
    double", "double", "double", "double", "double", "double"];
opts2.ExtraColumnsRule = "ignore";
opts2.EmptyLineRule = "read";
opts2.LeadingDelimitersRule = "ignore";
opts2 = setvaropts(opts2, ["SS", "EstimationError", "OptmizationError", "
    OF", "Thetaw1", "Thetaw2", "Thetaw3", "Valvew1", "Valvew2", "Valvew3",
     "WroEstimatedw1", "WroEstimatedw2", "WroEstimatedw3", "PrhEstimatedw1
    ", "PrhEstimatedw2", "PrhEstimatedw3", "WroOptimzedw1", "WroOptimzedw2
    ", "WroOptimzedw3", "PrhOptimzedw1", "PrhOptimzedw2", "PrhOptimzedw3",
     "u0w1", "u0w2", "u0w3"], "DecimalSeparator", ",");

direct = pwd;
expData = readtable(name, opts2);

%% Convert to output type
matlabData = table2array(expData);

%% Clear temporary variables
clear opts2

%% Data from the rig
Ju1_rig_avg = matlabData(i:end,5);
Ju2_rig_avg = matlabData(i:end,6);
Ju3_rig_avg = matlabData(i:end,7);
Ju1_rig = matlabData(i:end,8);
Ju2_rig = matlabData(i:end,9);
Ju3_rig = matlabData(i:end,10);
u1_rig = matlabData(i:end,20);
u2_rig = matlabData(i:end,21);
u3_rig = matlabData(i:end,22);
OF_rig = matlabData(i:end,4);
w_ro1_rig = matlabData(i:end,17);
```

```
w_ro2_rig = matlabData(i:end,18);
w_ro3_rig = matlabData(i:end,19);

u_rig = [u1_rig, u2_rig, u3_rig];
w_ro_rig = [w_ro1_rig, w_ro2_rig, w_ro3_rig];
%% Plotting
tlist2 = linspace(0,2*length(u1_rig),length(u1_rig));
tlist = tlist2/60;

dJdu1= -2*(u1_rig-3);
dJdu2= -2*(u2_rig-2.5);
dJdu3= -2*(u3_rig-2);

figure(1)
subplot(4,2,1)
plot(tlist,u1_rig, 'Color', [0.4 0 0.7], 'LineWidth', 1.5),hold on
yline(3,'--')
legend('u_{1,plant}','u_{1,opt}')
ylim([0.8 3.5])
xlabel('Time [min]')
ylabel('u_1 [sL/min]')
set(gca,'FontSize',13)

subplot(4,2,3)
plot(tlist,u2_rig, 'Color', [0.4 0 0.7], 'LineWidth',1.5),hold on
yline(2.5,'--')
legend('u_{2,plant}','u_{2,opt}')
ylim([0.8 3.5])
xlabel('Time [min]')
ylabel('u_2 [sL/min]')
set(gca,'FontSize',13)

subplot(4,2,5)
plot(tlist,u3_rig, 'Color', [0.4 0 0.7], 'LineWidth', 1.5),hold on
yline(2,'--')
legend('u_{3,plant}','u_{3,opt}')
ylim([0.8 3.5])
xlabel('Time [min]')
ylabel('u_3 [sL/min]')
set(gca,'FontSize',13)

subplot(4,2,7)
plot(tlist, -OF_rig,'Color', [0.3 0.3 1],'LineWidth', 1.5),hold on
yline(0, '--')
legend('J_{plant}', 'J_{opt}')
ylim([-1 8])
```

```
xlabel('Time [min]')
ylabel('OF [(sL)^2 min^{-2}]')
set(gca,'FontSize',13)

subplot(3,2,2)
plot(tlist,Ju1_rig_avg, 'Color', [0.4 0 0.7], 'LineWidth', 1.5),hold on
plot(tlist,dJdu1,'Color', [0 0.7 0.5],'LineWidth', 1.5), hold on
yline(0,'--', 'LineWidt',1.5)
ylim([-0.5 4.8])
%ylabel('$\textbf{Ju}_1 [\frac{\textbf{\$}}{\textbf{b}}]$','Interpreter','
    latex')
ylabel('Ju_1[sL/min]')
xlabel('Time[min]')
legend('Estimated', 'Real Value', 'Ju_{i,opt}');
set(gca,'FontSize',13)

subplot(3,2,4)
plot(tlist,Ju2_rig_avg,'Color', [0.4 0 0.7],'LineWidth', 1.5),hold on
plot(tlist,dJdu2,'Color', [0 0.7 0.5],'LineWidth', 1.5),hold on
yline(0,'--', 'LineWidt',1.5)
ylim([-0.5 4.8])
ylabel('Ju_2[sL/min]')
xlabel('Time[min]')
legend('Estimated', 'Real Value', 'Ju_{i,opt}');
set(gca,'FontSize',13)

subplot(3,2,6)
plot(tlist, Ju3_rig_avg,'Color', [0.4 0 0.7],'LineWidth', 1.5), hold on
plot(tlist,dJdu3,'Color', [0 0.7 0.5],'LineWidth', 1.5),hold on
yline(0,'--', 'LineWidt',1.5)
ylim([-0.5 4.8])
ylabel('Ju_3[sL/min]')
xlabel('Time[min]')
legend('Estimated', 'Real Value', 'Ju_{i,opt}')
set(gca,'FontSize',13)

%% Figure for FFT gradients
figure(2)
subplot(1,3,1)
plot(tlist,Ju1_rig,'Color', [0.4 0 0.7],'LineWidth', 1)
ylabel('Ju_{1}[sL/min]')
xlabel('Time[min]')
set(gca,'FontSize',12)

subplot(1,3,2)
plot(tlist,Ju2_rig,'Color', [0.4 0 0.7],'LineWidth', 1)
```

```
ylabel('Ju_{2}[sL/min]')
xlabel('Time[min]')
set(gca,'FontSize',12)

subplot(1,3,3)
plot(tlist,Ju3_rig,'Color', [0.4 0 0.7],'LineWidth', 1)
ylabel('Ju_{3}[sL/min]')
xlabel('Time[min]')
set(gca,'FontSize',12)
```

## PlottingDataCase2.m

```
clear
clc
close all

%% Import the data
%% LSE
i = 509;
endd = 883+i;
name = [pwd,'/2021-05-05_125639_LSE_case2Matlab.txt'];
%% ARX
%i = 1393;
%endd = 881+i;
%name = [pwd,'/2021-05-05_132756_ARX_case2Matlab.txt'];
%% FFT
%i = 2287;
%endd = 1035+i;
%name = [pwd,'/2021-05-05_140010_FFT_case2Matlab.txt'];

opts2 = delimitedTextImportOptions("NumVariables", 25);
opts2.DataLines = [2, Inf];
opts2.Delimiter = " ";
opts2.VariableNames = ["SS", "EstimationError", "OptmizationError", "OF",
    "Thetaw1", "Thetaw2", "Thetaw3", "Valvew1", "Valvew2", "Valvew3", "
    WroEstimatedw1", "WroEstimatedw2", "WroEstimatedw3", "PrhEstimatedw1",
     "PrhEstimatedw2", "PrhEstimatedw3", "WroOptimzedw1", "WroOptimzedw2",
     "WroOptimzedw3", "PrhOptimzedw1", "PrhOptimzedw2", "PrhOptimzedw3", "
    u0w1", "u0w2", "u0w3"];
opts2.VariableTypes = ["double", "double", "double", "double", "double", "
    double", "double", "double", "double", "double", "double", "double", "
    double", "double", "double", "double", "double", "double", "double", "
    double", "double", "double", "double", "double", "double"];
opts2.ExtraColumnsRule = "ignore";
opts2.EmptyLineRule = "read";
opts2.LeadingDelimitersRule = "ignore";
```

```
opts2 = setvaropts(opts2, ["SS", "EstimationError", "OptmizationError", "
    OF", "Thetaw1", "Thetaw2", "Thetaw3", "Valvew1", "Valvew2", "Valvew3",
     "WroEstimatedw1", "WroEstimatedw2", "WroEstimatedw3", "PrhEstimatedw1
    ", "PrhEstimatedw2", "PrhEstimatedw3", "WroOptimzedw1", "WroOptimzedw2
    ", "WroOptimzedw3", "PrhOptimzedw1", "PrhOptimzedw2", "PrhOptimzedw3",
     "u0w1", "u0w2", "u0w3"], "DecimalSeparator", ",");

direct = pwd;
expData = readtable(name, opts2);

%% Convert to output type
matlabData = table2array(expData);

%% Clear temporary variables
clear opts2

%% Data from the rig
Ju1_rig_avg = matlabData(i:endd,5);
Ju2_rig_avg = matlabData(i:endd,6);
Ju3_rig_avg = matlabData(i:endd,7);
Ju1_rig = matlabData(i:endd,8);
Ju2_rig = matlabData(i:endd,9);
Ju3_rig = matlabData(i:endd,10);
u1_rig = matlabData(i:endd,20);
u2_rig = matlabData(i:endd,21);
u3_rig = matlabData(i:endd,22);
OF_rig = matlabData(i:endd,4);
w_ro1_rig = matlabData(i:endd,17);
w_ro2_rig = matlabData(i:endd,18);
w_ro3_rig = matlabData(i:endd,19);

u_rig = [u1_rig, u2_rig, u3_rig];
w_ro_rig = [w_ro1_rig, w_ro2_rig, w_ro3_rig];

%% Plotting
tlist2 = linspace(0,2*length(u1_rig),length(u1_rig));
tlist = tlist2/60;

dJdu1 = [1*ones(500,1); 3*ones(length(Ju1_rig)-500,1)];
dJdu2 = [2*ones(500,1); 1*ones(length(Ju1_rig)-500,1)];
dJdu3 = [3*ones(500,1); 2*ones(length(Ju1_rig)-500,1)];
u1opt = [1*ones(500,1); 4*ones(length(Ju1_rig)-500,1)];
u2opt = [2.5*ones(500,1); 1*ones(length(Ju1_rig)-500,1)];
u3opt = [4*ones(500,1); 2.5*ones(length(Ju1_rig)-500,1)];

figure(1)
```

```
subplot(4,2,1)
plot(tlist,u1_rig, 'Color', [0.4 0 0.7], 'LineWidth', 1.5),hold on
plot(tlist,u1opt, '--', 'Color',[0.3 0.3 0.3],'LineWidth',1), hold on
yline(1, ':'), hold on
yline(4, ':')
lgd = legend('u_{1,plant}', 'u_{1,opt}', 'u_{i}^{min}', 'u_{i}^{max}');
lgd.FontSize = 11;
lgd.NumColumns = 2;
ylim([0.8 4.5])
xlabel('Time [min]')
ylabel('u_1 [sL/min]')
set(gca,'FontSize',13)

subplot(4,2,3)
plot(tlist,u2_rig, 'Color', [0.4 0 0.7], 'LineWidth',1.5),hold on
plot(tlist,u2opt, '--', 'Color',[0.3 0.3 0.3],'LineWidth',1)
yline(1, ':'), hold on
yline(4, ':')
lgd = legend('u_{2,plant}', 'u_{2,opt}', 'u_{i}^{min}', 'u_{i}^{max}');
lgd.FontSize = 11;
lgd.NumColumns = 2;
ylim([0.8 4.5])
xlabel('Time [min]')
ylabel('u_2 [sL/min]')
set(gca,'FontSize',13)

subplot(4,2,5)
plot(tlist,u3_rig, 'Color', [0.4 0 0.7], 'LineWidth', 1.5),hold on
plot(tlist,u3opt, '--', 'Color',[0.3 0.3 0.3],'LineWidth',1)
yline(1, ':'), hold on
yline(4, ':')
lgd = legend('u_{3,plant}', 'u_{3,opt}', 'u_{i}^{min}', 'u_{i}^{max}');
lgd.FontSize = 11;
lgd.NumColumns = 2;
ylim([0.8 4.5])
xlabel('Time [min]')
ylabel('u_3 [sL/min]')
set(gca,'FontSize',13)

subplot(4,2,7)
plot(tlist,sum(u_rig'),'Color', [0.3 0.3 1], 'LineWidth', 1.5), hold on
yline(7.5, ':')
legend('u_{tot}', 'u_{tot}^{max}')
xlabel('Time [min]')
ylabel('Total input usage [sL/min]')
set(gca,'FontSize',13)
```

```matlab

subplot(4,2,8)
plot(tlist,OF_rig, 'Color', [0.3 0.3 1],'LineWidth', 1.5),hold on
yline(3*4+2*2.5+1, '--')
legend('J_{plant}', 'J_{opt}')
xlabel('Time [min]')
ylabel('OF [$]')
set(gca,'FontSize',13)

subplot(4,2,2)
plot(tlist,Ju1_rig_avg, 'Color', [0.4 0 0.7], 'LineWidth', 1.5),hold on
plot(tlist,dJdu1, '--', 'Color',[0.3 0.3 0.3],'LineWidth',1)
ylabel('Ju_{1} [$ min/sL]')
xlabel('Time[min]')
ylim([0 3.2])
legend('Estimated', 'Real Value');
set(gca,'FontSize',13)

subplot(4,2,4)
plot(tlist,Ju2_rig_avg,'Color', [0.4 0 0.7],'LineWidth', 1.5),hold on
plot(tlist,dJdu2, '--', 'Color',[0.3 0.3 0.3],'LineWidth',1)
ylabel('Ju_{2} [$ min/sL]')
xlabel('Time[min]')
ylim([0 3.2])
legend('Estimated', 'Real Value');
set(gca,'FontSize',13)

subplot(4,2,6)
plot(tlist,Ju3_rig_avg,'Color', [0.4 0 0.7],'LineWidth', 1.5), hold on
plot(tlist,dJdu3, '--', 'Color',[0.3 0.3 0.3],'LineWidth',1)
ylabel('Ju_{3} [$ min/sL]')
xlabel('Time[min]')
ylim([0 3.2])
legend('Estimated', 'Real Value')
set(gca,'FontSize',13)

%% Figure for FFT gradients
figure(2)
subplot(1,3,1)
plot(tlist,Ju1_rig,'Color', [0.4 0 0.7],'LineWidth', 1)
ylabel('Ju_{1}[$ min/sL]')
xlabel('Time[min]')
set(gca,'FontSize',12)

subplot(1,3,2)
plot(tlist,Ju2_rig,'Color', [0.4 0 0.7],'LineWidth', 1)
```

```
ylabel('Ju_{2}[$ min/sL]')
xlabel('Time[min]')
set(gca,'FontSize',12)

subplot(1,3,3)
plot(tlist,Ju3_rig,'Color', [0.4 0 0.7],'LineWidth', 1)
ylabel('Ju_{3}[$ min/sL]')
xlabel('Time[min]')
set(gca,'FontSize',12)
```

## PlottingDataCase3.m

```
clear
clc
close all

%% Import the data
%% LSE
%i = 1;
%endd = 1575+i;
%name = [pwd,'/2021-06-01_case3_LSE_largeN.txt'];
%% ARX
%i = 1;
%endd = 1520+i;
%name = [pwd,'/2021-05-19_093835_ARX_case3_largeN2Matlab.txt'];
%% FFT
i = 1;
endd = 1660+i;
name = [pwd,'/2021-05-21_135057_FFT_case3_largeN2Matlab.txt'];

opts2 = delimitedTextImportOptions("NumVariables", 25);
opts2.DataLines = [2, Inf];
opts2.Delimiter = " ";
opts2.VariableNames = ["SS", "EstimationError", "OptmizationError", "OF",
    "Thetaw1", "Thetaw2", "Thetaw3", "Valvew1", "Valvew2", "Valvew3", "
    WroEstimatedw1", "WroEstimatedw2", "WroEstimatedw3", "PrhEstimatedw1",
     "PrhEstimatedw2", "PrhEstimatedw3", "WroOptimzedw1", "WroOptimzedw2",
     "WroOptimzedw3", "PrhOptimzedw1", "PrhOptimzedw2", "PrhOptimzedw3", "
    u0w1", "u0w2", "u0w3"];
opts2.VariableTypes = ["double", "double", "double", "double", "double", "
    double", "double", "double", "double", "double", "double", "double", "
    double", "double", "double", "double", "double", "double", "double", "
    double", "double", "double", "double", "double", "double"];
opts2.ExtraColumnsRule = "ignore";
opts2.EmptyLineRule = "read";
opts2.LeadingDelimitersRule = "ignore";
```

```
opts2 = setvaropts(opts2, ["SS", "EstimationError", "OptmizationError", "
    OF", "Thetaw1", "Thetaw2", "Thetaw3", "Valvew1", "Valvew2", "Valvew3",
     "WroEstimatedw1", "WroEstimatedw2", "WroEstimatedw3", "PrhEstimatedw1
    ", "PrhEstimatedw2", "PrhEstimatedw3", "WroOptimzedw1", "WroOptimzedw2
    ", "WroOptimzedw3", "PrhOptimzedw1", "PrhOptimzedw2", "PrhOptimzedw3",
     "u0w1", "u0w2", "u0w3"], "DecimalSeparator", ",");

direct = pwd;
expData = readtable(name, opts2);

%% Convert to output type
matlabData = table2array(expData);

%% Clear temporary variables
clear opts2

%% Data from the rig
Ju1_rig_avg = matlabData(i:endd,5);
Ju2_rig_avg = matlabData(i:endd,6);
Ju3_rig_avg = matlabData(i:endd,7);
Ju1_rig = matlabData(i:endd,8);
Ju2_rig = matlabData(i:endd,9);
Ju3_rig = matlabData(i:endd,10);
u1_rig = matlabData(i:endd,20);
u2_rig = matlabData(i:endd,21);
u3_rig = matlabData(i:endd,22);
OF_rig = matlabData(i:endd,4);
w_ro1_rig = matlabData(i:endd,17);
w_ro2_rig = matlabData(i:endd,18);
w_ro3_rig = matlabData(i:endd,19);

u_rig = [u1_rig, u2_rig, u3_rig];
w_ro_rig = [w_ro1_rig, w_ro2_rig, w_ro3_rig];

%% Offline gradient estimation
N = 900;
N_mean = 100;
P1 = 30;
P2 = 36;
P3 = 45;

Ju_hatListLSE = [];
Ju_hat_buffer_ListLSE = [];
Ju_hatListARX = [];
Ju_hat_buffer_ListARX = [];
Ju_hatListFFT = [];
```

```
Ju_hat_buffer_ListFFT = [];

Na = 3;
Nb = [2,2,2];
Nk = [0 0 0];

L = N/4;
NFFT = N;
Fs = NFFT;
%%
for i=N:size(u_rig,1)
    i
    u_buffer = u_rig(i-(N-1):i,:);
    J_buffer = OF_rig(i-(N-1):i);

    %% ARX
    data = iddata(J_buffer-mean(J_buffer),u_buffer-mean(u_buffer),1);
    m_arx = arx(data,[Na,Nb,Nk]);
    sysd = idss(m_arx); %dymanic system
    sysc = d2c(sysd); %continous sytem
    A = sysc.A;
    B = sysc.B;
    C = sysc.C;
    D = sysc.D;
    Ju_hat = -C*pinv(A)*B + D;
    Ju_hat = Ju_hat';

    if length(Ju_hatListARX)>N_mean
        Ju_hat_buffer = Ju_hatListARX(:,i-N-N_mean:i-N);
        Ju_hat_buffer_ListARX = [Ju_hat_buffer_ListARX, mean(Ju_hat_buffer
    ')'];
    end

    Ju_hatListARX = [Ju_hatListARX, Ju_hat(1:3)];
    %}
    %% LSE
    Ju_hat = LSE_gradient(J_buffer', u_buffer');

    if length(Ju_hatListLSE)>N_mean
        Ju_hat_buffer = Ju_hatListLSE(:,i-N-N_mean:i-N);
        Ju_hat_buffer_ListLSE = [Ju_hat_buffer_ListLSE, mean(Ju_hat_buffer
    ')'];
    end

    Ju_hatListLSE = [Ju_hatListLSE, Ju_hat(1:3)];
    %}
    %% FFT
    J_mean = J_buffer - movingavg(J_buffer,L/2,0);
    u1_mean = u_buffer(:,1) - movingavg(u_buffer(:,1),L/2,0);
    u2_mean = u_buffer(:,2) - movingavg(u_buffer(:,2),L/2,0);
    u3_mean = u_buffer(:,3) - movingavg(u_buffer(:,3),L/2,0);

    Jhat = fft(J_mean)/NFFT;
    u1hat = fft(u1_mean)/NFFT;
    u2hat = fft(u2_mean)/NFFT;
    u3hat = fft(u3_mean)/NFFT;

    f = Fs/2*linspace(0,1,NFFT/2+1);
    i1 = find(f==P1);
    i2 = find(f==P2);
    i3 = find(f==P3);

    Jmag = 2*abs(Jhat(1:NFFT/2+1));
    Jph = angle(Jhat)*180/pi; %phase information
    U1mag = 2*abs(u1hat(1:NFFT/2+1));
    U1ph = angle(u1hat)*180/pi; %phase information
    U2mag = 2*abs(u2hat(1:NFFT/2+1));
    U2ph = angle(u2hat)*180/pi;
    U3mag = 2*abs(u3hat(1:NFFT/2+1));
    U3ph = angle(u3hat)*180/pi;

    Ju_hat1 = sign(Jph(i1).*U1ph(i1))*Jmag(i1)/U1mag(i1);
    Ju_hat2 = sign(Jph(i2).*U2ph(i2))*Jmag(i2)/U2mag(i2);
    Ju_hat3 = sign(Jph(i3).*U3ph(i3))*Jmag(i3)/U3mag(i3);

    Ju_hat = [Ju_hat1; Ju_hat2; Ju_hat3];

    if length(Ju_hatListFFT)>N_mean
        Ju_hat_buffer = Ju_hatListFFT(:,i-N-N_mean:i-N);
        Ju_hat_buffer_ListFFT = [Ju_hat_buffer_ListFFT, mean(Ju_hat_buffer
    ')'];
    end

    Ju_hatListFFT = [Ju_hatListFFT,(Ju_hat(1:3))];
end

%% Plotting
tlist2 = linspace(0,2*length(u1_rig),length(u1_rig));
tlist = tlist2/60;

figure(1)
subplot(4,2,1)
```

```
plot(tlist,u1_rig, 'Color', [0.4 0 0.7], 'LineWidth', 1),hold on
yline(1, '--', 'Color',[0.3 0.3 0.3],'LineWidth',1), hold on
yline(1, ':'), hold on
yline(4, ':')
lgd = legend('u_{1,plant}', 'u_{1,opt}', 'u_{i}^{min}', 'u_{i}^{max}');
lgd.FontSize = 11;
lgd.NumColumns = 2;
ylim([0.8 4.5])
xlabel('Time [min]')
ylabel('u_1 [sL/min]')
set(gca,'FontSize',13)

subplot(4,2,3)
plot(tlist,u2_rig, 'Color', [0.4 0 0.7], 'LineWidth',1),hold on
yline(2.5, '--', 'Color',[0.3 0.3 0.3],'LineWidth',1)
yline(1, ':'), hold on
yline(4, ':')
lgd = legend('u_{2,plant}', 'u_{2,opt}', 'u_{i}^{min}', 'u_{i}^{max}');
lgd.FontSize = 11;
lgd.NumColumns = 2;
ylim([0.8 4.5])
xlabel('Time [min]')
ylabel('u_2 [sL/min]')
set(gca,'FontSize',13)

subplot(4,2,5)
plot(tlist,u3_rig, 'Color', [0.4 0 0.7], 'LineWidth', 1),hold on
yline(4, '--', 'Color',[0.3 0.3 0.3],'LineWidth',1)
yline(1, ':'), hold on
yline(4, ':')
lgd = legend('u_{3,plant}', 'u_{3,opt}', 'u_{i}^{min}', 'u_{i}^{max}');
lgd.FontSize = 11;
lgd.NumColumns = 2;
ylim([0.8 4.5])
xlabel('Time [min]')
ylabel('u_3 [sL/min]')
set(gca,'FontSize',13)

subplot(4,2,7)
plot(tlist,sum(u_rig'),'Color', [0.3 0.3 1], 'LineWidth', 1), hold on
yline(7.5, ':')
legend('u_{tot}', 'u_{tot}^{max}')
xlabel('Time [min]')
ylabel('Total input usage [sL/min]')
set(gca,'FontSize',13)

```

```matlab
subplot(4,2,8)
plot(tlist ,OF_rig, 'Color', [0.3 0.3 1],'LineWidth', 1),hold on
legend('J_{plant}')
xlabel('Time [min]')
ylabel('OF [$]')
set(gca,'FontSize',13)

subplot(4,2,2)
plot(tlist(901:end),Ju1_rig_avg(901:end), 'Color', [0.4 0 0.7], 'LineWidth', 1.5),hold on
plot(tlist(1001:end),Ju_hat_buffer_ListLSE(1,:),'--','LineWidth', 1.5), hold on
plot(tlist(1001:end),Ju_hat_buffer_ListARX(1,:),'--','LineWidth', 1.5), hold on
%plot(tlist(1001:end),Ju_hat_buffer_ListFFT(1,:),'--','LineWidth', 1.5), hold on
ylabel('Ju_{1} [$ min/sL]')
xlabel('Time[min]')
ylim([-1 10])
lgd = legend('FFT_{from rig}','LSE_{offline}', 'ARX_{offline}');
lgd.FontSize = 11;
set(gca,'FontSize',13)

subplot(4,2,4)
plot(tlist(901:end),Ju2_rig_avg(901:end),'Color', [0.4 0 0.7],'LineWidth', 1.5),hold on
plot(tlist(1001:end),Ju_hat_buffer_ListLSE(2,:),'--','LineWidth', 1.5), hold on
plot(tlist(1001:end),Ju_hat_buffer_ListARX(2,:),'--','LineWidth', 1.5), hold on
%plot(tlist(1001:end),Ju_hat_buffer_ListFFT(2,:),'--','LineWidth', 1.5), hold on
ylabel('Ju_{2} [$ min/sL]')
xlabel('Time[min]')
ylim([-1 10])
lgd = legend('FFT_{from rig}','LSE_{offline}', 'ARX_{offline}');
lgd.FontSize = 11;
lgd.NumColumns = 2;
set(gca,'FontSize',13)

subplot(4,2,6)
plot(tlist(901:end),Ju3_rig_avg(901:end),'Color', [0.4 0 0.7],'LineWidth', 1.5), hold on
plot(tlist(1001:end),Ju_hat_buffer_ListLSE(3,:),'--','LineWidth', 1.5), hold on
plot(tlist(1001:end),Ju_hat_buffer_ListARX(3,:),'--','LineWidth', 1.5),
    hold on
%plot(tlist(1001:end),Ju_hat_buffer_ListFFT(3,:),'--','LineWidth', 1.5),
    hold on
ylabel('Ju_{3} [$ min/sL]')
xlabel('Time[min]')
ylim([-1 10])
lgd = legend('FFT_{from rig}','LSE_{offline}', 'ARX_{offline}');
lgd.FontSize = 11;
lgd.NumColumns = 2;
set(gca,'FontSize',13)
%% Figure for FFT gradients
figure(2)
subplot(1,3,1)
plot(tlist(901:end), (Ju1_rig(901:end)),'Color', [0.4 0 0.7],'LineWidth',
    1)
ylabel('Ju_{1} [$ min/sL]')
xlabel('Time[min]')
set(gca,'FontSize',12)

subplot(1,3,2)
plot(tlist(901:end),(Ju2_rig(901:end)),'Color', [0.4 0 0.7],'LineWidth',
    1)
ylabel('Ju_{2} [$ min/sL]')
xlabel('Time[min]')
set(gca,'FontSize',12)

subplot(1,3,3)
plot(tlist(901:end),(Ju3_rig(901:end)),'Color', [0.4 0 0.7],'LineWidth',
    1)
ylabel('Ju_{3} [$ min/sL]')
xlabel('Time[min]')
set(gca,'FontSize',12)
```

# Appendix C

# Performance Analysis

## C.1 Performance Analysis Case Study 1

The code for the performance analysis, in Case Stduy 1, is shown in the code below. The loss is calculated as the deviation from the optimal solution, which is zero.

```
clear all
clc
opts2 = delimitedTextImportOptions("NumVariables", 25);
opts2.DataLines = [2, Inf];
opts2.Delimiter = " ";
opts2.VariableNames = ["SS", "EstimationError", "OptmizationError", "OF",
    "Thetaw1", "Thetaw2", "Thetaw3", "Valvew1", "Valvew2", "Valvew3", "
    WroEstimatedw1", "WroEstimatedw2", "WroEstimatedw3", "PrhEstimatedw1",
     "PrhEstimatedw2", "PrhEstimatedw3", "WroOptimzedw1", "WroOptimzedw2",
     "WroOptimzedw3", "PrhOptimzedw1", "PrhOptimzedw2", "PrhOptimzedw3", "
    u0w1", "u0w2", "u0w3"];
opts2.VariableTypes = ["double", "double", "double", "double", "double", "
    double", "double", "double", "double", "double", "double", "double", "
    double", "double", "double", "double", "double", "double", "double", "
    double", "double", "double", "double", "double", "double"];
opts2.ExtraColumnsRule = "ignore";
opts2.EmptyLineRule = "read";
opts2.LeadingDelimitersRule = "ignore";
opts2 = setvaropts(opts2, ["SS", "EstimationError", "OptmizationError", "
    OF", "Thetaw1", "Thetaw2", "Thetaw3", "Valvew1", "Valvew2", "Valvew3",
     "WroEstimatedw1", "WroEstimatedw2", "WroEstimatedw3", "PrhEstimatedw1
    ", "PrhEstimatedw2", "PrhEstimatedw3", "WroOptimzedw1", "WroOptimzedw2
    ", "WroOptimzedw3", "PrhOptimzedw1", "PrhOptimzedw2", "PrhOptimzedw3",
     "u0w1", "u0w2", "u0w3"], "DecimalSeparator", ",");

direct = pwd;

name1 = [pwd,'/2021-04-28_135606_LSE_case1_test1Matlab.txt'];
name2 = [pwd,'/2021-04-28_141632_ARX_case1_test1Matlab.txt'];
name3 = [pwd,'/2021-04-28_152159_FFT_case1_test2Matlab.txt'];
LSE = readtable(name1, opts2);
ARX = readtable(name2, opts2);
FFT = readtable(name3, opts2);
matlabData1 = table2array(LSE);
matlabData2 = table2array(ARX);
matlabData3 = table2array(FFT);

OF_rig_LSE = matlabData1(826:end,4);
OF_rig_ARX = matlabData2(1271:end,4);
OF_rig_FFT = matlabData3(2984:end,4);

J_loss_LSE = zeros(434,1);
J_loss_LSE(1)= -OF_rig_LSE(1)/60;
J_loss_ARX = zeros(434,1);
J_loss_ARX(1)= -OF_rig_ARX(1)/60;
J_loss_FFT = zeros(434,1);
J_loss_FFT(1)= -OF_rig_FFT(11)/60;
J_loss_FFT2 = J_loss_FFT;

for i=2:434
    J_loss_LSE(i) = J_loss_LSE(i-1) + (-OF_rig_LSE(i)/30);
    J_loss_ARX(i) = J_loss_ARX(i-1) + (-OF_rig_ARX(i)/30);
    J_loss_FFT(i) = J_loss_FFT(i-1) + (-OF_rig_FFT(i+10)/30);
end

tlist2 = linspace(0,2*434,434);
tlist = tlist2/60;

subplot(1,2,1)
plot(tlist, J_loss_LSE, 'LineWidth',1.5),hold on
plot(tlist, J_loss_ARX, 'LineWidth',1.5),hold on
plot(tlist, J_loss_FFT, 'LineWidth',1.5),hold on
legend('LSE','ARX','FFT')
xlabel('time [min]')
ylabel('Accumulated loss [(sL)^2 min^{-2}]')
subplot(1,2,2)
plot(tlist(300:end), J_loss_LSE(300:end), 'LineWidth',1.5),hold on
plot(tlist(300:end), J_loss_ARX(300:end), 'LineWidth',1.5),hold on
plot(tlist(300:end), J_loss_FFT(300:end), 'LineWidth',1.5),hold on
legend('LSE','ARX','FFT')
xlabel('time [min]')
ylabel('Accumulated loss [(sL)^2 min^{-2}]')
```

## C.2 Performance Analysis Case Study 2

The code for the performance analysis, in Case Study 2, is shown in the code below. The loss is calculated as the deviation from the optimal solution, which is 18.

```
clear all

opts2 = delimitedTextImportOptions("NumVariables", 25);
opts2.DataLines = [2, Inf];
opts2.Delimiter = " ";
opts2.VariableNames = ["SS", "EstimationError", "OptmizationError", "OF",
    "Thetaw1", "Thetaw2", "Thetaw3", "Valvew1", "Valvew2", "Valvew3", "
    WroEstimatedw1", "WroEstimatedw2", "WroEstimatedw3", "PrhEstimatedw1",
     "PrhEstimatedw2", "PrhEstimatedw3", "WroOptimzedw1", "WroOptimzedw2",
     "WroOptimzedw3", "PrhOptimzedw1", "PrhOptimzedw2", "PrhOptimzedw3", "
    u0w1", "u0w2", "u0w3"];
opts2.VariableTypes = ["double", "double", "double", "double", "double", "
    double", "double", "double", "double", "double", "double", "double", "
    double", "double", "double", "double", "double", "double", "double", "
    double", "double", "double", "double", "double", "double"];
opts2.ExtraColumnsRule = "ignore";
opts2.EmptyLineRule = "read";
opts2.LeadingDelimitersRule = "ignore";
opts2 = setvaropts(opts2, ["SS", "EstimationError", "OptmizationError", "
    OF", "Thetaw1", "Thetaw2", "Thetaw3", "Valvew1", "Valvew2", "Valvew3",
     "WroEstimatedw1", "WroEstimatedw2", "WroEstimatedw3", "PrhEstimatedw1
    ", "PrhEstimatedw2", "PrhEstimatedw3", "WroOptimzedw1", "WroOptimzedw2
    ", "WroOptimzedw3", "PrhOptimzedw1", "PrhOptimzedw2", "PrhOptimzedw3",
     "u0w1", "u0w2", "u0w3"], "DecimalSeparator", ",");

direct = pwd;

name1 = [pwd,'/2021-05-05_125639_LSE_case2Matlab.txt'];
name2 = [pwd,'/2021-05-05_132756_ARX_case2Matlab.txt'];
name3 = [pwd,'/2021-05-05_140010_FFT_case2Matlab.txt'];

LSE = readtable(name1, opts2);
```

```
ARX = readtable(name2, opts2);
FFT = readtable(name3, opts2);
matlabData1 = table2array(LSE);
matlabData2 = table2array(ARX);
matlabData3 = table2array(FFT);

OF_rig_LSE = matlabData1(509:end,4);
OF_rig_ARX = matlabData2(1393:end,4);
OF_rig_FFT = matlabData3(2287:end,4);

Jopt = 4*3 + 2*2.5 + 1;
J_loss_LSE = zeros(884,1);
J_loss_LSE(1)= (Jopt-OF_rig_LSE(1))/30;
J_loss_ARX = zeros(884,1);
J_loss_ARX(1)= (Jopt-OF_rig_ARX(1))/30;
J_loss_FFT = zeros(884,1);
J_loss_FFT(1)= (Jopt-OF_rig_FFT(21))/30;
J_loss_FFT2 = J_loss_FFT;

for i=2:884
    J_loss_LSE(i) = J_loss_LSE(i-1) + (Jopt-OF_rig_LSE(i))/30;
    J_loss_ARX(i) = J_loss_ARX(i-1) + (Jopt-OF_rig_ARX(i))/30;
    J_loss_FFT(i) = J_loss_FFT(i-1) + (Jopt-OF_rig_FFT(i+20))/30;
end

tlist2 = linspace(0,2*884,884);
tlist = tlist2/60;

subplot(1,2,1)
plot(tlist, J_loss_LSE, 'LineWidth',1.5),hold on
plot(tlist, J_loss_ARX, 'LineWidth',1.5),hold on
plot(tlist, J_loss_FFT, 'LineWidth',1.5),hold on
legend('LSE','ARX','FFT')
xlabel('time [min]')
ylabel('Accumulated loss [$]')

subplot(1,2,2)
plot(tlist(725:end), J_loss_LSE(725:end), 'LineWidth',1.5),hold on
plot(tlist(725:end), J_loss_ARX(725:end), 'LineWidth',1.5),hold on
plot(tlist(725:end), J_loss_FFT(725:end), 'LineWidth',1.5),hold on
legend('LSE','ARX','FFT')
xlabel('time [min]')
ylabel('Accumulated loss[$]')
```

Appendix D

# Experiments With Shorter Buffer Length in Case Study 3

Figure D.1 and D.2 show the results from Case Study 3, with $N = 300$, using LSE and ARX respectively. By comparing the figures with Figure 6.2 and 6.3, it is clear that the gradient estimation with a longer buffer length preforms better.

Note that in the ARX experiment, the objective function is given by,

$$J = 20w_{ro,1} + 10w_{ro,2} + 5w_{ro,3} + 0.5(u_1 + u_2 + u_3) \tag{D.1}$$

while in the LSE experiment, the objective function is the same as in Case Study 3, that is

$$J = 5w_{ro,1} + 10w_{ro,2} + 20w_{ro,3} + 0.5(u_1 + u_2 + u_3) \tag{D.2}$$

The plot on the left side show the inputs, and the sum of the inputs, that is the total gaslift. The right side plots show the estimated gradients, in addition to the objective function.

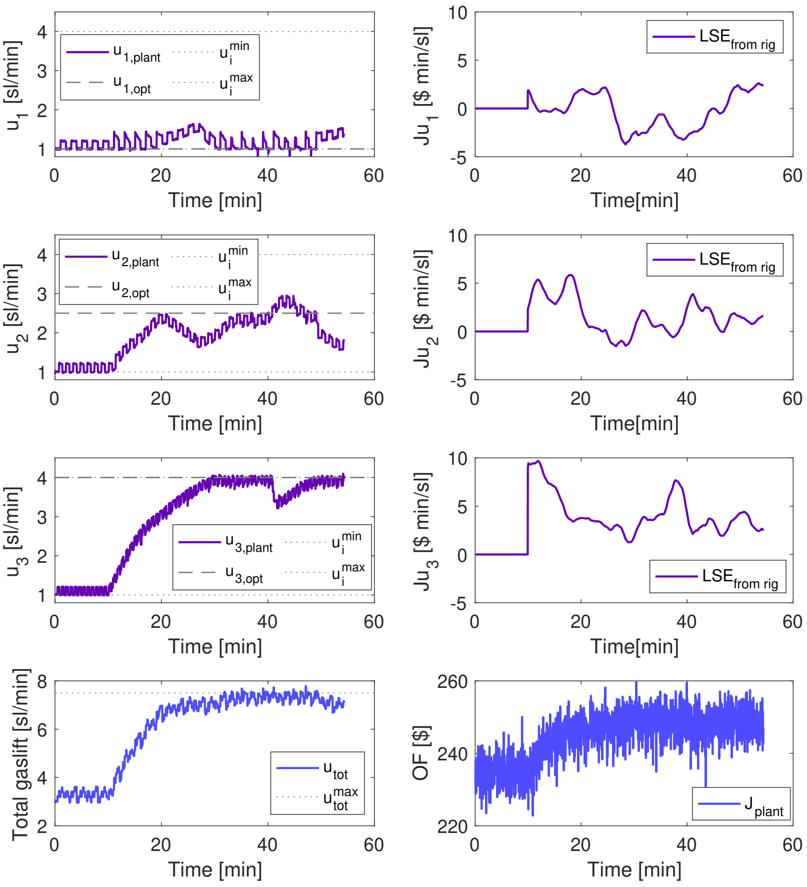


**Figure D.1:** Result of the optimization of the gas lifted well network using LSE and a shorter buffer length. The figure show the inputs, objective function, the total of the inputs, and their optimums..

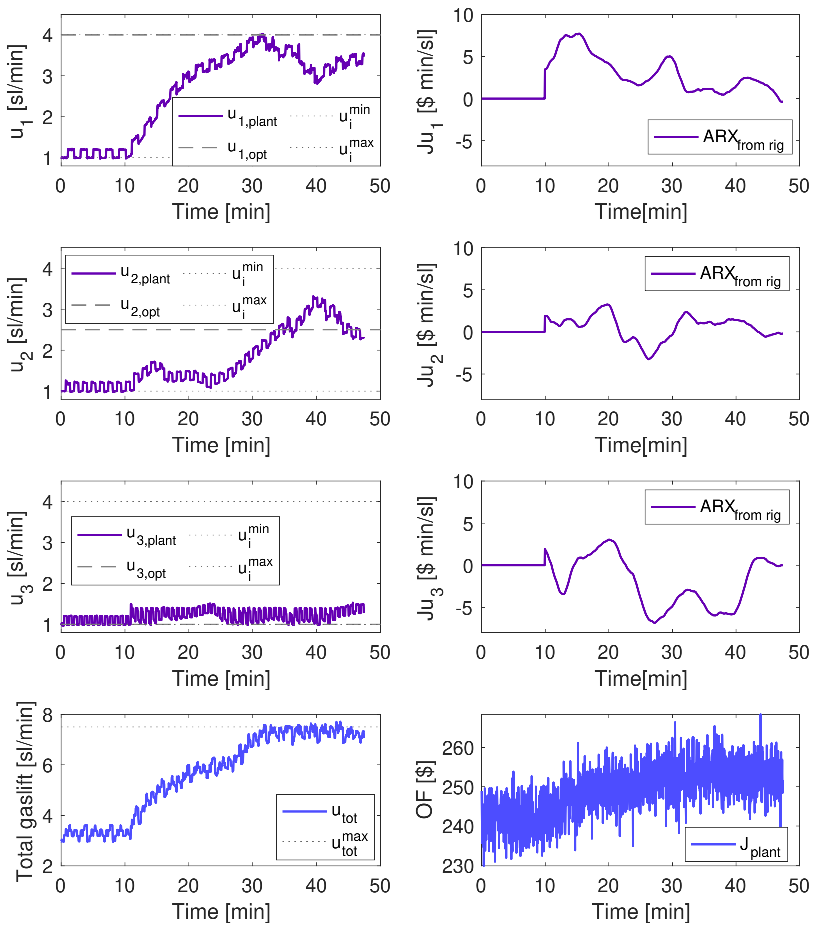


**Figure D.2:** Result of the optimization of the gas lifted well network using ARX and a shorter buffer length. The figure show the inputs, objective function, the total of the inputs, and their optimums.

Frida Bakken Myrvang

Implementation of Extremum Seeking Control in an Experimental Lab-Rig