NTNU
Kunnskap for en bedre verden

Department of Chemical Engineering

TKP4580 Chemical Process Technology, Specialization project

Modifier Adaptation: implementation in Experimental Lab Rig

Author:
Maren Sofie Lia

Supervisor:
Johannes Jäschke

Co-supervisor:
Jose Otavio Assumpcao Matias

December, 2021

Abstract

The main objective of this specialization project is to study the implementation of Real Time Optimization via Modifier Adaptation in an experimental lab rig, which represents a subsea oil well network with gaslift. RTO aims to maximize the revenue and minimize the operational costs of a process plant in real time. The RTO layer is followed by a faster control layer that accounts for fast corrective actions. Implementation of traditional RTO is challenging, mainly because it requires rigorous first-principles based models. Incorrect model structures lead to severe plant-model mismatch, which can result in the algorithm converging to the incorrect optimum. Modifier Adaptation can be a possible solution to these challenges. In contrast to traditional RTO, the MA method relies on a fixed nominal steady-state process model. The advantage of MA lies in its proven ability to converge to plant optimum despite plant-model mismatch. This is ensured by modifiers, which modifies the optimization problem in such a way that plant optimum is achieved. However, there is a gap in the literature when it comes to MA implementations. Only a few MA implementations in real systems are reported in the literature. This project is attempting to contribute to fill this gap.

Since plant optimality is ensured by the modifiers in the MA algorithm, the idea is to study if a simple model can be used instead of a rigorous one. Accurate modelling of oil well networks can be very difficult and time consuming. Firstly, a few different simple steady-state models for the experimental rig were obtained. This was done by obtaining steady-state data from the rig. Then, statistical methods were used to obtain and evaluate the models. Secondly, standard MA and Output Modifier Adaptation were applied to the chosen obtained model and implemented in a simulation case study of the experimental rig. The difference in the two approaches lies in how they modify the optimization problem. They both use modifiers, but the modifiers are calculated from different plant measurements and gradients. The simulation results showed that the implementations with the simple model were able to drive the system to its optimum. However, this only yield when the measurement noise is low. Lastly, a trial run in the actual experimental rig was carried out. The experimental results showed that the output modifier adaptation approach has a potential. But, this requires significant better gradient estimation.

Table of Contents

List of Figures

List of Tables

Nomenclature

Acronyms

AIC	Akaike Information Criterion	
CFD	Central-finite-differencing	
DAE	Differential algebraic equation	
FDA	Finite-difference approximation	
FFD	Forward-finite-differencing	
KKT	Karush-Kuhn-Tucker	
LICQ	Linear Independence Constraint Qualification	
MA	Modifier Adaptation	
MAy	Ouput Modifier Adaptation	
MPC	Model Predictive Controller	
NLP	Nonlinear programming	
PI	Proportional-integral	
PID	Proportional-integral–derivative	
RTO	Real Time Optimization	

Symbols

α	Acceptable probability error	
ϵ	Experimental error	
λ	Gradient modifier	
$\mathcal{L}$	Lagrange function	
μ	Lagrange multiplier	
ν	Degrees of freedom	
ρ_g	Gas density	kg/m^3
ρ_{mix}	Mixture density	kg/m^3
θ	Model parameters	
ε	Modifier	
ξ	Model variables	
b	ε filter gain	
d	λ filter gain	
d	Disturbances/plant parameters	
f	Steady-state model	
G	Constraint	
J	Objective function	
K	Input filter gain	
m_g	Gas hold up	kg
m_l	Liquid hold up	kg
m_{gout}	Well outlet gas	kg/s
m_{lout}	Well outlet liquid	kg/s

n_g	Number of constraints	
n_u	Number of inputs	
n_y	Number of outputs	
p_{bi}	Pressure before injection point	bar
p_{pump}	Reservoir pressure	bar
p_{rh}	Riser head pressure	bar
q	λ filter gain	
Q_g	Gas flowrate	sL/min
Q_l	Liquid flowrate	L/min
R^2	R^2 statistic	
S	Sum of squares function	
s^2	Estimate of variance	
u	Input	
v_o	Valve opening	L/min
w_l	Water flowrate	kg/s
w_{total}	Total well production rate	kg/s
x	States	
y	Output	
Z	Function of experimental conditions	

1 Introduction

Optimal operation of continuously process plants is of great economic importance in the chemical industry. Optimal operation requires meeting goals in different time scales ranging from long-term planning to fast regulation for stable operation. Realizing all these goals as a whole is unrealistic and difficult. Operation is hence decomposed into various decision making layers. Figure 1 illustrates the plant decision hierarchy, where the third layer is of interest in this project. Real Time Optimization (RTO) aims to maximize the revenues and minimize the cost of hour-by-hour operation and typically provides setpoints to the lower layer [1]. The lower layer can be, among other things, an MPC controller or a PID controller. RTO incorporate measurements of the process plant in the optimization to drive the processes to optimal performance, without violating the constraints [2]. However, optimal operation can be difficult to achieve with inaccurate process models or in the presence of disturbances [3].

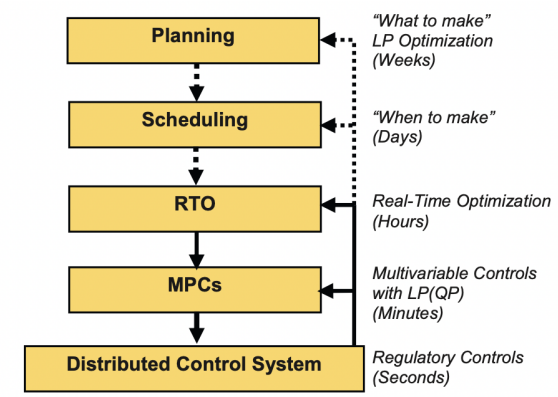


Figure 1: Plant decision hierarchy.

Source: Darby, Mark L., et al. "RTO: An overview and assessment of current practice." *Journal of Process Control* 21.6 (2011): 874-884.

Traditional steady-state RTO typically uses rigorous process models. These models are very costly to develop and obtaining them requires a very good understanding of the process. Incorrect model structures lead to plant-model mismatch, which can result in the RTO converging to an incorrect optimum. There are also computational issues associated to solving and simulating detailed and complex models, which can lead to convergence issues and numerical failure. An RTO variant called Modifier Adaptation (MA) can be used for addressing these issues. While traditional RTO uses measurements of the plant to improve the model, standard MA uses the measurements to modify the optimization problem and relies on a fixed model. The idea behind MA is to modify the optimization problem by adding correction terms to the cost and constraint functions and enforce plant optimality, despite the presence of parametric and structural plant-model mismatch.

Using MA can solve some of the traditional RTO challenges, but there are still some

challenges which need to be addressed:

- Steady-state wait time. MA is, as traditional RTO, a steady-state optimization method. The algorithm is only running at steady-state. The plant needs time to stabilize before the next RTO iteration, which can be very time consuming. For instance, in case of frequent disturbances.
- Accurate measurements are needed to detect steady-state and especially for gradient estimation. Noisy measurements are very challenging for gradient estimation.
- Dynamic limitations. Since MA is a steady-state optimization method, it does not take the dynamics of the process into account. The RTO layer does not care about how to actually reach the optimal value, because it does not consider the transients.

1.1 Specialization Project goals

The goal of this specialization project is to get a better understanding of the implementation of MA in real systems and the challenges that follows. The project covers:

- Steady-state model estimation of an experimental lab rig. This model is used in the implementation of MA.
- Implementation of MA and Output Modifier Adaptation (MAy), which is an alternative to the standard MA, in a simulation case study of the experimental rig. This simulation consists of a dynamic model representing the behaviour of the rig. The simulation case study is used to study the tuning parameters that affect the performance, as well as to study the performance of the estimated model.
- Implementation of MAy in the actual experimental rig. A trial run is carried out in the rig to study the performance of the algorithm. The performance of MAy in the rig is compared to a previously developed traditional RTO algorithm.

2 Theory and Background

2.1 Production Optimization

2.1.1 Steady-State Optimization

The steady-state production optimization problem for the plant can be formulated as:

$$\begin{aligned} \min_{\mathbf{u}} \quad & J_p(\mathbf{u}) := J(\mathbf{u}, \mathbf{y}_p(\mathbf{u}, \mathbf{d})) \\ \text{s.t.} \quad & \mathbf{G}_p(\mathbf{u}) := \mathbf{g}(\mathbf{u}, \mathbf{y}_p(\mathbf{u}, \mathbf{d})) \leq \mathbf{0}, \\ & \mathbf{u}^L \leq \mathbf{u} \leq \mathbf{u}^U \end{aligned} \tag{1}$$

The notation p is used for variables associated with the plant. $\mathbf{u} \in \mathbb{R}^{n_u}$ denotes the decision variables and $\mathbf{y}_p \in \mathbb{R}^{n_y}$ denotes the measured output variables of the plant. n_u and n_y are the number of inputs and outputs respectively. J_p: $\mathbb{R}^{n_u} \times \mathbb{R}^{n_y} \to \mathbb{R}$ is the operational costs of the plants which should be minimized. $\mathbf{g}$ is a set of n_g inequality constraint functions, where g_i: $\mathbb{R}^{n_u} \times \mathbb{R}^{n_y} \to \mathbb{R}$, $i = 1, \ldots, n_g$. The inequality constraints are often operational limitations. $\mathbf{y}_p(\mathbf{u},\mathbf{d})$ represents the steady-state input-output mapping of the plant, where $\mathbf{d} \in \mathbb{R}^{n_d}$ are representing plant parameters and disturbances. n_d is the number of plant parameters and disturbances. $\mathbf{u}^L$ and $\mathbf{u}^U$ are the lower and upper bounds of $\mathbf{u}$ respectively. These bounds are not dependent on $\mathbf{y}_p$ and are therefore not affected by uncertainty. The cost and constraint functions are assumed to be known directly from the measurements. Since $\mathbf{y}_p(\mathbf{u},\mathbf{d})$ is typically unknown, a steady-state process model is used for solving the problem in Eq. (1).

$$\begin{aligned} f(\mathbf{u}, \mathbf{x}, \mathbf{d}) &= \mathbf{0} \\ \mathbf{y} &= \mathbf{h}(\mathbf{u}, \mathbf{x}, \mathbf{d}) \end{aligned} \tag{2}$$

where f is the steady-state process model and $\mathbf{y} \in \mathbb{R}^{n_y}$ is the output variables predicted by the model. $\mathbf{x} \in \mathbb{R}^{n_x}$ are representing the state variables. For simplicity, $\mathbf{y}$ can be expressed as a function of only $\mathbf{u}$ and $\mathbf{d}$. The solution $\mathbf{u}^*$ to the original optimization problem can be obtained by solving to the following NLP problem:

$$\begin{aligned} \arg\min_{\mathbf{u}} \quad & J(\mathbf{u}) := J(\mathbf{u}, \mathbf{y}_p(\mathbf{u}, \mathbf{d})) \\ \text{s.t.} \quad & \mathbf{G}(\mathbf{u}) := \mathbf{g}(\mathbf{u}, \mathbf{d}) \leq \mathbf{0}, \\ & \mathbf{u}^L \leq \mathbf{u} \leq \mathbf{u}^U \end{aligned} \tag{3}$$

However, it should be mentioned that it is not guaranteed that the optimal solution to this problem, $\mathbf{u}^*$, coincide with the optimal plant value $\mathbf{u}_p^*$ [3]. In the presence of plant-model mismatch, it could be likely that $\mathbf{u}^*$ converge to an incorrect optimum.

2.1.2 Necessary Conditions of Optimality

The local minima of Problem (3) can be characterized by the Karush-Kuhn-Tucker (KKT) conditions. A solution $\mathbf{u}^*$ is a KKT point if there exist unique Langrange multipliers $\boldsymbol{\mu}^* \in \mathbb{R}_g^n$ such that the following holds:

$$\begin{aligned} \mathbf{G} \leq \mathbf{0}, \quad \boldsymbol{\mu}^{\boldsymbol{T}}\mathbf{G} = 0, \quad \boldsymbol{\mu} \geq \mathbf{0} \\ \frac{\partial \mathcal{L}}{\partial \mathbf{u}} = \frac{\partial J}{\partial \mathbf{u}} + \boldsymbol{\mu}^{\boldsymbol{T}} \frac{\partial \mathbf{G}}{\partial \mathbf{u}} = \mathbf{0} \end{aligned} \tag{4}$$

where $\mathcal{L}(\mathbf{u}, \boldsymbol{\mu}) := \mathrm{J}(\mathbf{u}) + \boldsymbol{\mu}^T\mathbf{G}(\mathbf{u})$ is the Lagrangian function. The KKT conditions alone do not sufficiently charaterize an optimum. Two additional conditions must hold; The Linear Independence Constraint Qualification (LICQ) and a second order necessary condition. LICQ requires that the gradients of the active constraints are linearly independent at $\mathbf{u}^*$. The second order necessary conditions require that the reduced Hessian of the Lagrangian is positive semi-definite at $\mathbf{u}^*$ [2].

2.1.3 Modifier Adaptation

The RTO method selected in this project is Modifier Adaptation. While traditional RTO is updating the model by estimating model parameters and disturbances, MA relies on a fixed process model. The advantage of MA lies in its proven ability to converge to the plant optimum despite the plant-model mismatch. The idea is to modify the optimization problem in such a way that a KKT point for the model coincide with the real optimum of the plant.

The standard MA scheme uses measurements of the plant constraints and estimates of plant gradients to modify the cost and constraint functions in the optimization problem. At the kth RTO iteration with input $\mathbf{u}_k$, the modified cost and constraint functions become:

$$J_{m,k}(\mathbf{u}) := J(\mathbf{u}) + \epsilon_k^J + (\boldsymbol{\lambda}_k^J)^T(\mathbf{u} - \mathbf{u}_k) \tag{5}$$

$$G_{m,i,k}(\mathbf{u}) := G_i(\mathbf{u}) + \varepsilon_k^{G_i} + (\boldsymbol{\lambda}_k^{G_i})^T(\mathbf{u} - \mathbf{u}_k) \leq 0, \quad i = 1, \ldots, n_g \tag{6}$$

with ε_k^J, $\varepsilon_k^{G_i}$, $\boldsymbol{\lambda}_k^J$ and $\boldsymbol{\lambda}_k^{G_i}$ given by:

$$\begin{aligned}
\epsilon_k^J &= J_p(\mathbf{u}_k) + J(\mathbf{u}_k) \\
\varepsilon_k^{G_i} &= G_{p,i}(\mathbf{u}) + G_i(\mathbf{u}), \quad i = 1, \ldots, n_g \\
(\boldsymbol{\lambda}_k^J)^T &= \frac{\partial J_p}{\partial \mathbf{u}}(\mathbf{u}_k) - \frac{\partial J}{\partial \mathbf{u}}(\mathbf{u}_k) \\
(\boldsymbol{\lambda}_k^{G_i})^T &= \frac{\partial G_{p,i}}{\partial \mathbf{u}}(\mathbf{u}_k) - \frac{\partial G_i}{\partial \mathbf{u}}(\mathbf{u}_k), \quad i = 1, \ldots, n_g
\end{aligned} \tag{7}$$

where n_g is the number of constraints. The modifiers ε_k^J and $\varepsilon_k^{G_i}$ represent the differences between the plant values and the predicted values at the kth RTO iteration. $\boldsymbol{\lambda}_k^J$ and $\boldsymbol{\lambda}_k^{G_i}$ represent the differences between the plant gradients and the model gradients at the kth RTO iteration. The next optimal input sequence $\mathbf{u}_{k+1}^*$ is found by solving the modified optimization problem:

$$\begin{aligned}
\arg\min_{\mathbf{u}} \quad & J_{m,k}(\mathbf{u}) := J(\mathbf{u}) + (\boldsymbol{\lambda}_k^J)^T \mathbf{u} \\
\text{s.t.} \quad & G_{m,i,k}(\mathbf{u}) := G_i(\mathbf{u}) + \varepsilon_k^{G_i} + (\boldsymbol{\lambda}_k^{G_i})^T (\mathbf{u} - \mathbf{u}_k) \leq \mathbf{0}, \quad i = 1, \ldots, n_g, \\
& \mathbf{u}^L \leq \mathbf{u} \leq \mathbf{u}^U
\end{aligned} \tag{8}$$

The constant term $\varepsilon_k^J - (\boldsymbol{\lambda}_k^J)^T \mathbf{u}_k$ does not affect the solution and, hence only the linear term in $\mathbf{u}$ is included in the objective function. The main implementation difficulty with this algorithm is the need to estimate cost and constraint gradients at each iteration. These gradients can not be measured directly and they have to be estimated based on noisy plant measurements. Since the modifiers and the optimal inputs are sensitive to measurement noise, first-order filters are often applied to both the modifiers and the optimal inputs, as shown in Eqs. (9).

$$\begin{aligned}
\varepsilon_{k+1}^{G_i} &= (1 - b_i)\varepsilon_k^{G_i} + b_i(G_{p,i}(\mathbf{u}) + G_i(\mathbf{u})), \quad i = 1, \ldots, n_g \\
(\boldsymbol{\lambda}_{k+1}^J)^T &= (1 - d)(\boldsymbol{\lambda}_k^J)^T + d(\frac{\partial J_p}{\partial \mathbf{u}}(\mathbf{u}_k) - \frac{\partial J}{\partial \mathbf{u}}(\mathbf{u}_k)) \\
(\boldsymbol{\lambda}_{k+1}^{G_i})^T &= (1 - q_i)(\boldsymbol{\lambda}_k^{G_i})^T + q_i(\frac{\partial G_{p,i}}{\partial \mathbf{u}}(\mathbf{u}_k) - \frac{\partial G_i}{\partial \mathbf{u}}(\mathbf{u}_k)), \quad i = 1, \ldots, n_g \\
\mathbf{u}_{k+1} &= \mathbf{u}_k + \mathbf{K}(\mathbf{u}_{k+1}^* - \mathbf{u}_k)
\end{aligned} \tag{9}$$

where b_i, d, q_i and $\mathbf{K}$ have values (0,1] [3].

2.1.4 Modification of Output Variables

Output Modifier Adaptation (MAy) is an alternative to the standard Modifier Adaptation method. Instead of modifying the cost and constraint functions, the outputs $\mathbf{y}$ are

modified. At the kth RTO iteration, the expression for the modified outputs are as follows:

$$\mathbf{y}_{m,k}(\mathbf{u}) := \mathbf{y}(\mathbf{u}) + \boldsymbol{\varepsilon^y}_k + (\boldsymbol{\lambda^y}_k)^T(\mathbf{u} - \mathbf{u}_k) \tag{10}$$

with $\boldsymbol{\varepsilon^y}_k$ and $\boldsymbol{\lambda^y}_k$ given by:

$$\begin{aligned} \boldsymbol{\varepsilon^y}_k &= \mathbf{y}_p(\mathbf{u}_k) - \mathbf{y}(\mathbf{u}_k) \\ (\boldsymbol{\lambda^y}_k)^T &= \frac{\partial \mathbf{y}_p}{\partial \mathbf{u}}(\mathbf{u}_k) - \frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k) \end{aligned} \tag{11}$$

The next optimal inputs $\mathbf{u}^*_{k+1}$ are found by solving the following modified optimization problem:

$$\begin{aligned} \arg\min_{\mathbf{u}} \quad & J(\mathbf{u}, \mathbf{y}_{m,k}(\mathbf{u})) \\ \text{s.t.} \quad & \mathbf{y}_{m,k}(\mathbf{u}) = \mathbf{y}(\mathbf{u}) + \boldsymbol{\varepsilon^y}_k + (\boldsymbol{\lambda^y}_k)^T(\mathbf{u} - \mathbf{u}_k), \\ & G_i(\mathbf{u}, \mathbf{y}_{m,k}(\mathbf{u})) \leq 0 \qquad i = 1, \ldots, n_g, \\ & \mathbf{u}^L \leq \mathbf{u} \leq \mathbf{u}^U \end{aligned} \tag{12}$$

First-order filters are also applied to these modifiers and to the optimal inputs, as shown in Eqs. (9).

2.1.5 Gradient Estimation

There are many methods available for estimating plant gradients. One very common approach is to use Finite-difference approximation (FDA). This method is a steady-state perturbation method, which relies on steady-state data. The FDA approach require usually $n_u + 1$ steady-state operating points to estimate the gradients. The Forward-finite-differencing (FFD) approach is perturbing each input individually around the current RTO point to get an estimate of the gradient [2]. As an example, the gradient of the output with respect to u_j at the kth RTO iteration is esimated as:

$$\widehat{\frac{\partial \mathbf{y}_p}{\partial u_j}}(\mathbf{u}_k) = [\mathbf{y}_p(\mathbf{u}_k + h\mathbf{e}_j) - \mathbf{y}_p(\mathbf{u}_k)]/h, \quad h > 0 \tag{13}$$

where h is the step size and $\mathbf{e}_j$ is the jth unit vector. The Central-finite-differencing (CDF) approach is an alternative to FFD. In this method, each input is perturbed twice. This yields more accurate approximations. The gradient of the output, with respect to u_j, at

the kth RTO iteration is esimated as:

$$\widehat{\frac{\partial \mathbf{y}_p}{\partial u_j}}(\mathbf{u}_k) = [\mathbf{y}_p(\mathbf{u}_k + h\mathbf{e}_j) - \mathbf{y}_p(\mathbf{u}_k - h\mathbf{e}_j)]/2h, \quad h > 0 \tag{14}$$

Both methods contain noisy measurements, which can lead to poor gradient estimates. Ideally, h should be as small as possible to obtain good gradients. However, h can not be too small in the presence of noisy measurements.

2.2 Model Identification

2.2.1 The Method of Least Squares

The method of least squares is used for fitting empirical functions to data. It can be useful to develop a model,

$$y = f(\boldsymbol{\xi}, \boldsymbol{\theta}) + \epsilon \tag{15}$$

where y is the output, $f(\boldsymbol{\xi}, \boldsymbol{\theta})$ is the mean level of the output and ϵ is the experimental error [4]. The experimental error is the error associated with the data, e.g. measurement error. The output is affected by the variables $\boldsymbol{\xi}$, which can include inputs and disturbances. In addition, the model contains parameters $\boldsymbol{\theta}$. Different experimental runs at n different sets of conditions are necessary to obtain such model.

The method of least squares selects the best estimate of $\boldsymbol{\theta}$ such that the sum of the errors $\{y_1 - f(\boldsymbol{\xi}_1, \boldsymbol{\theta})\}$, $\{y_2 - f(\boldsymbol{\xi}_2, \boldsymbol{\theta})\}$,..., $\{y_n - f(\boldsymbol{\xi}_n, \boldsymbol{\theta})\}$ become as small as possible. This is done by minimizing the sum of squares of the deviations, as follows:

$$\arg\min_{\boldsymbol{\theta}} \quad S(\boldsymbol{\theta}) = \sum_{j=1}^{n} \{y_j - f(\boldsymbol{\xi}_j, \boldsymbol{\theta})\}^2 \tag{16}$$

where $S(\boldsymbol{\theta})$ is the sum of squared errors. The solution to this problem will be the estimated parameters, $\hat{\boldsymbol{\theta}}$.

A great simplification in the computation of the parameter estimates can be done if the response function is linear in the parameters, as shown in Eq. (17). Adding the experimental error, we get the model shown in Eq. (15).

$$f(\boldsymbol{\xi}, \boldsymbol{\theta}) = \theta_1 z_1 + \theta_2 z_2 + \ldots \theta_p z_p \tag{17}$$

$$y = \theta_1 z_1 + \theta_2 z_2 + \ldots \theta_p z_p + \epsilon \tag{18}$$

p is the number of parameters and the z's are known functions of the experimental conditions, $\boldsymbol{\xi}$ e.g. $z_i = u^2 d$. These types of models are said to be linear because they are linear in the parameters.

If a model structure such as the one in Eq. (17) has been proposed and experimental data from n different sets of conditions have been obtained, then the model can relate the observations y_1, y_2,..., y_n to the known z_{ij}'s and the unknown θ_i's by n equations:

$$\begin{aligned} y_1 &= \theta_1 z_{11} + \theta_2 z_{21} + \ldots + \theta_p z_{p1} + \epsilon_1 \\ y_2 &= \theta_2 z_{12} + \theta_2 z_{22} + \ldots + \theta_p z_{p2} + \epsilon_2 \\ &\ldots \\ &\ldots \\ y_n &= \theta_1 z_{1n} + \theta_2 z_{2n} + \ldots + \theta_p z_{pn} + \epsilon_n \end{aligned} \tag{19}$$

Which in matrix form can be written as:

$$\mathbf{y} = \mathbf{Z}\boldsymbol{\theta} + \boldsymbol{\epsilon} \tag{20}$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{Z} = \begin{bmatrix} z_{11} & z_{21} & \ldots & z_{p1} \\ z_{12} & z_{22} & \ldots & z_{p2} \\ \vdots & \vdots & & \vdots \\ z_{1n} & z_{2n} & \ldots & z_{pn} \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_p \end{bmatrix}, \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix} \tag{21}$$

The sum of squared errors can now be written as:

$$S(\boldsymbol{\theta}) = \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} = (\mathbf{y} - \mathbf{Z}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{Z}\boldsymbol{\theta}) \tag{22}$$

The least squares estimates $\hat{\boldsymbol{\theta}}$, which minimizes $S(\boldsymbol{\theta})$ are found by solving the following equation:

$$\mathbf{Z}^T \mathbf{Z}\hat{\boldsymbol{\theta}} = \mathbf{Z}^T \mathbf{y} \tag{23}$$

which gives:

$$\hat{\boldsymbol{\theta}} = (\mathbf{Z}^T \mathbf{Z})^{-1} \mathbf{Z}^T \mathbf{Z} \tag{24}$$

The fitted equation $\hat{y}$ is therefore:

$$\hat{y} = \hat{\theta}_1 z_{11} + \hat{\theta}_2 z_{21} + \ldots + \hat{\theta}_p z_{p1} \tag{25}$$

2.2.2 The R-squared Statistic

The R^2 statistic often represents the overall fit for the model. It represents the proportion of variance about the mean that is explained by the fitted model [4]. R^2 can take values between 0 and 1. Generally, a higher value indicates a better fit for the model. The R^2 statistic is given by:

$$R^2 = \frac{\hat{\boldsymbol{\theta}}^T \mathbf{Z}^T \mathbf{y} - N\bar{y}^2}{\mathbf{y}^T \mathbf{y} - N\bar{y}^2} \tag{26}$$

where $\bar{y}$ is the overall mean value of y and N is the total number of observations.

2.2.3 The Akaike Information Criterion

The Akaike Information Criterion (AIC) is a widely used method for choosing between different competing models. AIC takes the model complexity into account, in addition to how well the model reproduces the data. The model complexity is determined by the number of independent variables used to build the model. AIC for ordinary least squares models is given by:

$$AIC = nln(\frac{\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}{n}) + 2(p+1) \tag{27}$$

Lower AIC scores are better, because AIC penalizes models that contain more parameters. If two models reproduce the data with the same level of precision, the one with fewer parameters will have the lowest AIC score [5].

2.2.4 Confidence Intervals

A confidence interval refers to the probability that a parameter will fall between a set of values for a certain proportion of times. It can be shown that a $1-\alpha$ confidence interval for $\hat{\theta}_i$ is given by:

$$\hat{\theta}_i \pm t_{\alpha/2}(\nu)(s^2 c^{ii})^{1/2} \tag{28}$$

where $t(\nu)$ represents a t distribution with $\nu = N - p$ degrees of freedom. c^{ii} denotes the ith diagonal term of $(\mathbf{Z}^T\mathbf{Z})^{-1}$ and s^2 denotes the estimate of the variance of y. Confidence

intervals are useful because they show the region that includes the true value of $\boldsymbol{\theta}$ with a given probability. However, they do not light the correlations between the different $\hat{\theta}_i$'s [4].

3 System Description

3.1 Subsea Oil Well Networks

The overall goal in subsea oil production is to maximize the production of oil from the reservoirs. Typically, the reservoir pressure drives the fluids from below the seafloor to the top facilities trough risers. A gaslift can be applied to the system if this pressure is not large enough. A simplified figure of the system is shown in Figure 2. The injected gas reduces the bulk density and decreases the hydrostatic pressure on the reservoir. However, if the gas injection rate becomes too large, the frictional pressure drop effect dominates and the injected gas will have a negative effect [6].

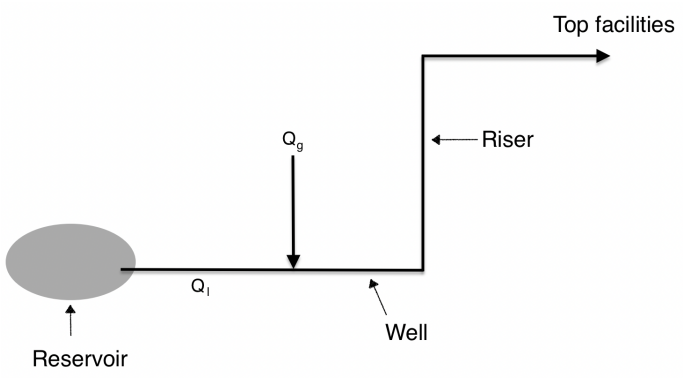


Figure 2: Simple model diagram of one gas lifted well. Q_g is the gas lift rate and Q_l is the oil production rate.

3.2 Experimental Lab Rig Setup

The experimental rig represents a subsea oil well network with three parallel gas lifted wells. The setup in the rig uses water and air as working fluids instead of oil and gas. A flowsheet of the rig is shown in Figure 3. The system measurements are the well top pressure (PI101, PI102 and PI103), the pump outlet pressure (PI104), the liquid flowrates (FI101, FI102, FI103) and the gas flowrates (FI104, FI105, FI106). The reservoir valve openings (CV101, CV102, CV103) are known disturbances in the experiments. The different valve openings represents different behaviours of the reservoir. The pump outlet pressure PI104 represents the reservoir pressure and is kept constant by a PI controller. Three PI controllers are used for controlling the gas flowrates, whose setpoints are the system inputs.

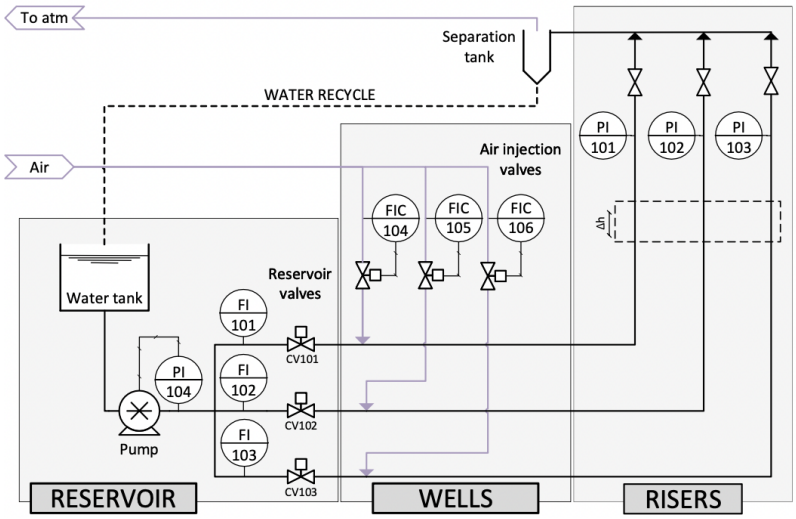


Figure 3: Flowsheet of the experimental rig.

3.3 The Optimization Problem

The objective is to maximize the economic profit J by determine the optimal injection rate (the setpoints of flow controllers FIC104, FIC105 and FIC106). J is a function of the liquid flowrates. The outputs $\mathbf{y}$ and the inputs $\mathbf{u}$ are:

$$\mathbf{y} = [Q_{l_1}\ Q_{l_2}\ Q_{l_3}]^T$$
$$\mathbf{u} = [Q_{g_1}\ Q_{g_2}\ Q_{g_3}]^T$$

The three parallel wells have different priority in the objective function due to economic reasons. This is indicated by using different weights. Gas availability constraints and gas injection bounds have to be respected. Maximum gas availability is 7.5 sL/min and the upper and lower bounds of each gas injection rate is 1 and 5 sL/min respectively. The optimization problem becomes:

$$\begin{aligned} \max_{Q_{g_1}, Q_{g_2}, Q_{g_3}} \quad & J = 20Q_{l_1} + 10Q_{l_2} + 30Q_{l_3} \\ \text{s.t.} \quad & Q_{g_1} + Q_{g_2} + Q_{g_3} \leq 7.5, \\ & 1 \leq Q_{g_1}, Q_{g_2}, Q_{g_3} \leq 5 \end{aligned} \tag{29}$$

where Q_{l_1}, Q_{l_2} and Q_{l_3} are the liquid flowrates FI101, FI102 and FI103 respectively. Q_{g_1}, Q_{g_2} and Q_{g_3} are the setpoints to the flow controllers FIC104, FIC105 and FIC106 respectively.

3.4 Experimental Rig Simulations

Before testing the optimization method in the actual experimental lab rig, some simulation was done using a rigorous first-principles model as the plant, while the simple model of Section 4 is used in the Modifier Adaptation layer for economic optimization purposes.

The model (plant in the simulation) of the three gas lifted wells is obtained from [7]. The model has been implemented in MATLAB to test and tune different optimization methods before they have been applied to the experimental rig. A model diagram is shown in Figure 4. The model uses mass balances of the different phases, density models, pressure models and flow models, which become the following differential algebraic equation (DAE):

$$\begin{aligned} \dot{\mathbf{x}}_i &= \mathbf{f}_i(\mathbf{x}_i, \mathbf{z}_i, \mathbf{u}_i, \mathbf{p}_i) \\ \mathbf{g}_i(\mathbf{x}_i, \mathbf{z}_i, \mathbf{u}_i, \mathbf{p}_i) &= \mathbf{0} \quad \forall i \in \mathcal{N} = \{1, \ldots, n_w\} \end{aligned} \tag{30}$$

where the subscript i is referring to a well in the set $\mathcal{N}$. x_i is the differential states, z_i is the algebraic states, u_i is the decision variables and p_i is the uncertain variables. n_w is the number of wells, i.e. n_w=3 in this case. $f_i(x_i, z_i, u_i, p_i)$ is the set of differential equations and $g_i(x_i, z_i, u_i, p_i)$ is the set of algebraic equations. The MATLAB code with these equations is shown in Appendix A.

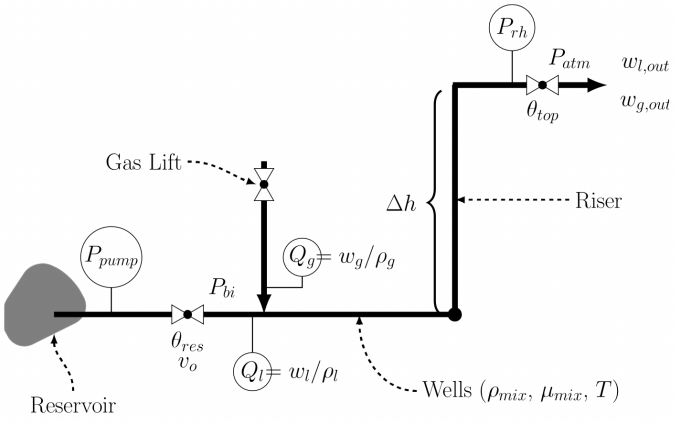


Figure 4: Model diagram of one single well.

The differential states, the algebraic states and the decision variables are given by:

$$\begin{aligned} \mathbf{x}_i &= [m_{g_i} \; m_{l_i}]^T \\ \mathbf{z}_i &= [w_{l_i} \; w_{total_i} \; p_{rh_i} \; p_{bi_i} \; \rho_{mix} \; \rho_{g_i} \; w_{gout_i} \; w_{lout_i}]^T \\ \mathbf{u}_i &= [Q_{g_i} \; v_{o_i} \; p_{pump}]^T \end{aligned} \tag{31}$$

where m_{g_i} is the gas hold up and m_{l_i} is the liquid hold up. w_{l_i} is the water rate from the reservoir and w_{total_i} is the total well production rate. p_{rh_i} is the riser head pressure

and p_{bi_i} is the pressure before injection point. ρ_{mix} is the mixture density in the system and ρ_{g_i} is the density of the gas. w_{gout_i} and w_{lout_i} is the well outlet gas and liquid outlet flowrate respectively. Q_{g_i} is the gas lift injection rate, v_{o_i} is the valve opening from the reservoir and p_{pump} is the reservoir pressure.

Note that the valve opening v_o is included as a decision variable in the model, but its seen as a (measured) disturbance in the simulation. p_{pump} is held at a constant value in the simulation.

4 Model Estimation

Four different steady-state models of the experimental rig were obtained from an experiment. All the obtained models are linear with respect to the parameters. The method of least squares was used to fit the models to the data. Then statistical methods were used for evaluation, and the fourth model turned out to be preferred over the other ones. The idea of obtaining simplified models of the process is to study if they can be used in the optimization instead of more complex ones, and still give good results.

4.1 Estimation of models

An experiment was carried out in the experimental rig to establish the steady-state relationship between the gas flowrates Q_{g_i} and the valve openings v_{o_i} and the liquid flowrates Q_{l_i}, shown in Eq. (32). The experimental grid is shown in Figure 5.

$$Q_{l_i} = f(Q_{g_i}, v_{o_i}, \boldsymbol{\theta}) + \epsilon \tag{32}$$

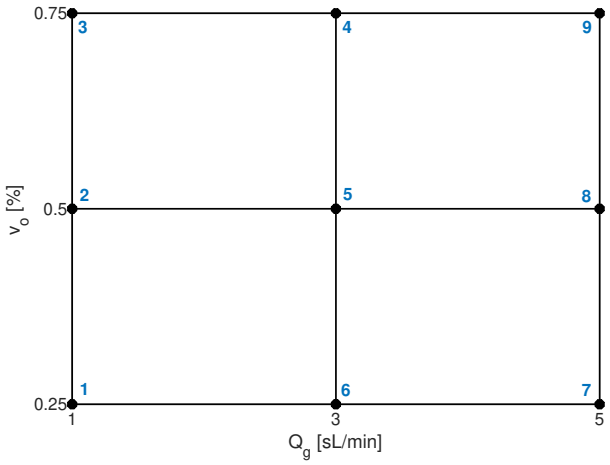


Figure 5: Experimental grid. The numbers 1-9 represents the different sets of conditions.

The experiment was carried out at 9 different sets of conditions. These are shown in the experimental grid. Four different steady-state model structures were suggested and since the models are linear w.r.t the parameters, ordinary least squares was used to fit the models to the data and obtain the different model parameters. The structures of model 1, 2, 3 and 4 are shown in Eqs. (33), (34), (35) and (36) respectively. The models are shown for one well. The structures were chosen arbitrarily, but based on a priori process knowledge. Model 1 is the simplest model of the four, i.e. it contains the smallest number of parameters. Model 2 is expanded with two terms and, hence has two more parameters. As the goal is to find the simplest model as possible, which also fits the dataset in a good way, two other models were obtained. Model 3 and 4 are equal to the second model with

one term removed and, therefore have one less parameter than the second model.

$$\text{Model 1} : \hat{Q}_l = \hat{\theta}_1 + \hat{\theta}_2 Q_g + \hat{\theta}_3 v_o + \hat{\theta}_4 Q_g v_o \tag{33}$$

$$\text{Model 2} : \hat{Q}_l = \hat{\theta}_1 + \hat{\theta}_2 Q_g + \hat{\theta}_3 v_o + \hat{\theta}_4 Q_g v_o + \hat{\theta}_5 Q_g^2 + \hat{\theta}_6 v_o^2 \tag{34}$$

$$\text{Model 3} : \hat{Q}_l = \hat{\theta}_1 + \hat{\theta}_2 Q_g + \hat{\theta}_3 v_o + \hat{\theta}_4 Q_g v_o + \hat{\theta}_5 Q_g^2 \tag{35}$$

$$\text{Model 4} : \hat{Q}_l = \hat{\theta}_1 + \hat{\theta}_2 Q_g + \hat{\theta}_3 v_o + \hat{\theta}_4 Q_g^2 + \hat{\theta}_5 v_o^2 \tag{36}$$

Instead of relying on physical knowledge to obtain the models, it is chosen to represent the process by using simple mathematical expressions for describing the input-output relationship. Instead of choosing more complex model structures, it is chosen to use only polynomials because they are linear in the parameters, and can represent the first and second order effects of the inputs on the outputs. Thus, the models in Eqs. (33)-(36) are referred to as simple. The estimated parameters are shown in Table 2, where the first number in each row is representing the first well, the second is representing second well and the third is representing third well.

Table 2: Estimated parameters in model 1, 2, 3 and 4.

Model 1				**Model 2**			
$\hat{\boldsymbol{\theta}}$	**Well 1**	**Well 2**	**Well 3**	$\hat{\boldsymbol{\theta}}$	**Well 1**	**Well 2**	**Well 3**
$\hat{\theta}_1$	3.657	2.924	2.892	$\hat{\theta}_1$	0.572	-0.361	-0.753
$\hat{\theta}_2$	0.182	0.265	0.210	$\hat{\theta}_2$	0.274	0.488	0.580
$\hat{\theta}_3$	4.813	5.642	5.298	$\hat{\theta}_3$	19.153	20.277	21.039
$\hat{\theta}_4$	0.054	-0.059	0.059	$\hat{\theta}_4$	0.053	-0.058	0.091
				$\hat{\theta}_5$	-0.015	-0.037	-0.065
				$\hat{\theta}_6$	-14.338	-14.639	-15.713
Model 3				**Model 4**			
$\hat{\boldsymbol{\theta}}$	**Well 1**	**Well 2**	**Well 3**	$\hat{\boldsymbol{\theta}}$	**Well 1**	**Well 2**	**Well 3**
$\hat{\theta}_1$	3.567	2.688	2.549	$\hat{\theta}_1$	0.493	-0.274	-0.883
$\hat{\theta}_2$	0.267	0.489	0.556	$\hat{\theta}_2$	0.300	0.459	0.620
$\hat{\theta}_3$	4.813	5.640	5.320	$\hat{\theta}_3$	19.312	20.102	21.313
$\hat{\theta}_4$	0.054	-0.059	0.093	$\hat{\theta}_4$	-0.015	-0.037	-0.064
$\hat{\theta}_5$	-0.014	-0.037	-0.061	$\hat{\theta}_5$	-14.338	-14.639	-15.714

4.2 Simulation of the models

A simulation of the four models was obtained to visualize how good they fit the dataset. Figure 6 shows the simulation of all the four models. The models for each individual well are plotted in the feasible region of operation, where Q_l is showed on the z-axis. The true value and the model estimated value are shown as well. Table 3 shows the absolute residuals in each datapoint.

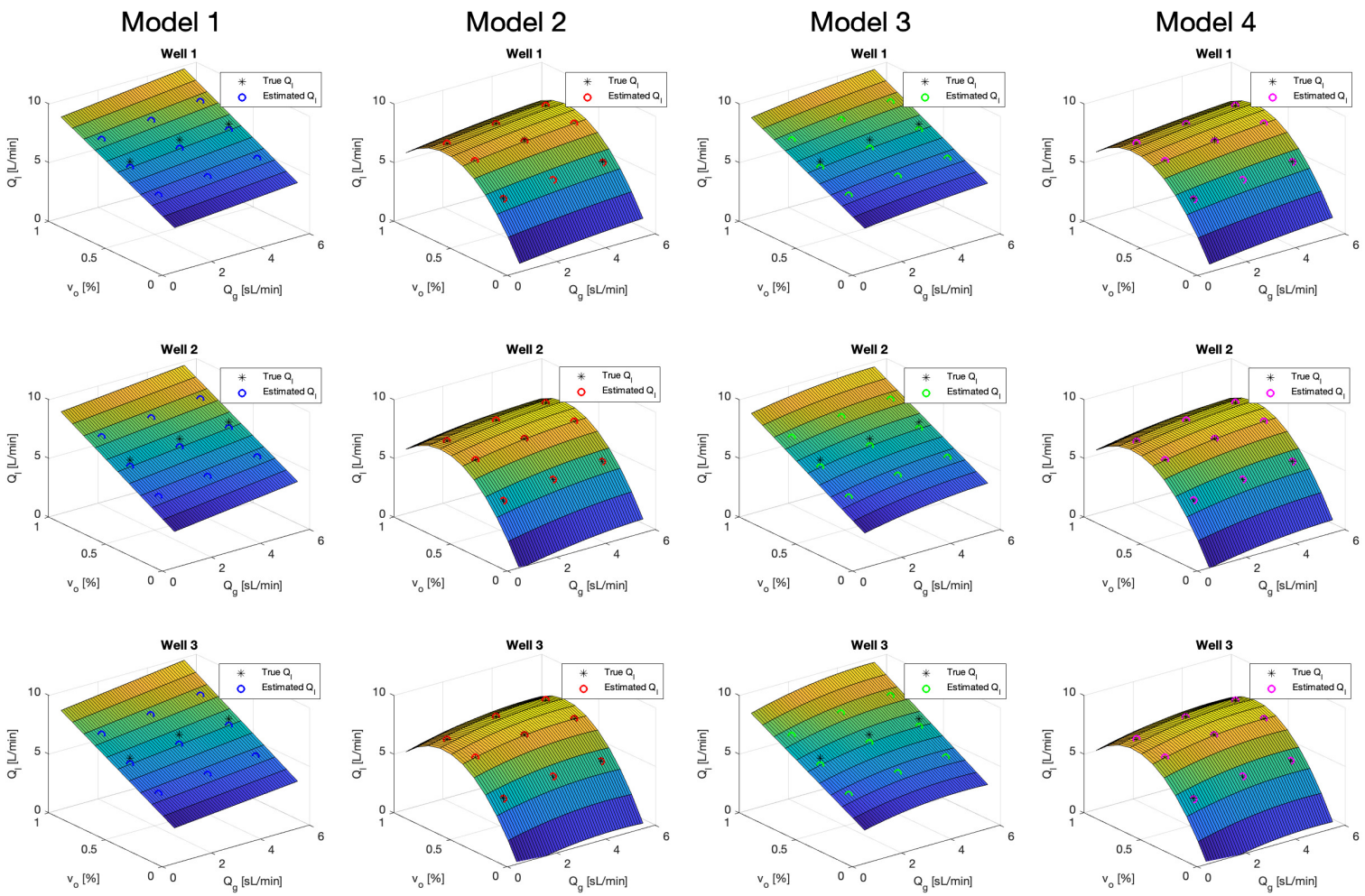


Figure 6: Surface plot of the four models and each well. True output value and model estimated output value marked on each plot. Each column is a different model while each row represents a different well.

Table 3: Abolute residuals.

	Model 1			Model 2		
Point	**Well 1**	**Well 2**	**Well 3**	**Well 1**	**Well 2**	**Well 3**
1	-0.244	-0.355	-0.382	0.0753	-0.0011	0.0207
2	0.510	0.560	0.522	-0.0680	-0.0004	-0.0475
3	-0.322	-0.354	-0.410	-0.0035	0.0016	0.0116
4	-0.250	-0.222	-0.136	0.0077	-0.0168	0.0167
5	0.771	0.737	0.869	0.133	0.0277	0.0457
6	-0.409	-0.218	-0.188	-0.152	-0.0122	-0.0225
7	-0.243	-0.342	-0.406	0.0764	0.0133	0.0286
8	0.511	0.533	0.561	-0.0646	-0.0273	0.0057
9	-0.324	-0.340	-0.430	-0.0042	0.0152	-0.0235
	Model 3			**Model 4**		
Point	**Well 1**	**Well 2**	**Well 3**	**Well 1**	**Well 2**	**Well 3**
1	-0.225	-0.306	-0.313	0.102	-0.0302	0.0650
2	0.529	0.610	0.602	-0.0680	-0.0004	-0.0488
3	-0.303	-0.304	-0.320	-0.0300	0.0308	-0.0351
4	-0.288	-0.322	-0.300	0.0077	-0.0168	0.0191
5	0.733	0.638	0.711	0.133	0.0276	0.0484
6	-0.447	-0.317	-0.340	-0.152	-0.0123	-0.0198
7	-0.224	-0.292	-0.303	0.0500	0.0425	-0.0165
8	0.530	0.583	0.655	-0.0646	-0.0272	0.0046
9	-0.306	-0.290	-0.357	0.0223	-0.0140	0.0210

4.3 R-squared statistic and AIC of the models

The R^2 statistic and AIC score were calculated for each of the four models of the three wells. Table 4 shows the mean R^2-value and the mean AIC score of the three wells for each model.

Table 4: Mean R^2 statistic and AIC score for the estimated models.

Model	**R^2**	**AIC**
1	0.8453	-4.44
2	0.9722	-47.2
3	0.8491	-2.73
4	0.9718	-44.8

4.4 Confidence intervals for parameters

A 95% confidence intervals for the model parameters in the second and fourth model are shown in Table 5.

Table 5: Confidence intervals for model parameters

Model 2			
$\hat{\boldsymbol{\theta}}$	**Well 1**	**Well 2**	**Well 3**
$\hat{\theta_1}$	0.572 ± 0.51	-0.361 ± 0.36	-0.753 ± 0.50
$\hat{\theta_2}$	0.274 ± 0.19	0.488 ± 0.13	0.580 ± 0.18
$\hat{\theta_3}$	19.153 ± 1.87	20.277 ± 1.33	21.039 ± 1.88
$\hat{\theta_4}$	0.053 ± 0.16	-0.058 ± 0.11	0.091 ± 0.16
$\hat{\theta_5}$	-0.015 ± 0.028	-0.037 ± 0.020	-0.065 ± 0.028
$\hat{\theta_6}$	-14.338 ± 1.79	-14.639 ± 1.28	-15.713 ± 1.80
Model 4			
$\hat{\boldsymbol{\theta}}$	**Well 1**	**Well 2**	**Well 3**
$\hat{\theta_1}$	0.493 ± 0.45	-0.274 ± 0.32	-0.883 ± 0.45
$\hat{\theta_2}$	0.300 ± 0.17	0.459 ± 0.12	0.620 ± 0.17
$\hat{\theta_3}$	19.312 ± 1.81	20.102 ± 1.29	21.313 ± 1.82
$\hat{\theta_4}$	-0.015 ± 0.028	-0.037 ± 0.020	-0.064 ± 0.027
$\hat{\theta_5}$	-14.338 ± 1.79	-14.639 ± 1.28	-15.714 ± 1.80

4.5 Discussion and conclusion

The simulation of the models clearly shows that model 1 and model 3 have the largest deviation from the true values. Table 3 shows that model 2 and model 4 have lower residuals all over the feasible region. Model 2 is expected to fit the dataset the best because it includes the first and second order input-output interaction terms, as well as a cross term of the two inputs.

The residuals of model 4 show that this model is almost as good as model 2 one, despite the fact that the second model contains an extra term. Table 2 shows that the parameters of the second and the fourth model are almost identical, except for $\hat{\theta_4}$ in model 2, which is from the cross term $Q_g v_o$. The cross term is the only difference between model 2 and 4. It could be likely that this term is not needed.

Table 4 shows the mean R^2 statistic of the models, as well as the mean AIC score. The R^2 statistic shows that the second and the fourth model are the best suited for the dataset, since they have values closer to 1 than the other two models. However, these values, as well as the residuals, are only showing how well the models suit the dataset from the experimental grid. These values may be different for a different experimental grid and

a different dataset. The mean AIC score of the models also show poor values for model 1 and 3, which is expected from the previous analysis. Model 2 and 4 have significantly smaller AIC scores, which is preferred.

Since the first and the third model showed such bad residuals compared to the other two, in addition to a lower R^2 value and a poor AIC score, confidence intervals were only computed for the second and the fourth model. These are shown in Table 5. The intervals are not showing the correlations between the different parameters, it only shows the uncertainty in the parameters when the others are held constant. The interval of $\hat{\theta_1}$ stands out as very large. It has a interval approximately 100% its value in both of the models. $\hat{\theta_3}$ in both models shows an acceptable value, as well as the $\hat{\theta_6}$ in the first model and $\hat{\theta_5}$ in the fourth model. The other $\hat{\theta}$'s show quite large intervals, which is expected since the models are relatively simple.

The R^2 value and the AIC score of model 2 and 4 can be considered as equivalent, due to the fact that that only 9 experimental points are used in the experiment. Since model 4 has fewer parameters than the second, and shows a similar accuracy, this model is chosen to be implemented in the optimization.

5 Modifier Adaptation Case Study

Modifier Adaptation is implemented in a simulation case study of the experimental rig, described in Section 3. The optimization problem is Problem (29). The *plant* is the dynamic model of the experimental rig and the *model* is the estimated steady state-model from Section 4,

$$Q_l = \theta_1 + \theta_2 Q_g + \theta_3 v_o + \theta_4 Q_g^2 + \theta_5 v_o^2$$

The model is shown for one well.

5.1 Method

Before optimizing the system using Modifier Adaptation, first-order modifiers $\boldsymbol{\lambda}^J$ need to be calculated. For computing them, the plant gradients need to be estimated. The Forward-finite differencing approach is used in this implementation, where the inputs are perturbed individually around the current operating point. Then, the plant gradient $\frac{\partial J_p}{\partial \mathbf{u}}(\mathbf{u}_k)$ is estimated as:

$$\frac{\partial J_p}{\partial u_j}(\mathbf{u}_k) = [J_p(\mathbf{u}_k + h\mathbf{e}_j) - J_p(\mathbf{u}_k)]/h \tag{37}$$

where j is representing the jth input and k is representing the kth MA iteration. J_p is calculated from plant measurements. Note that the plant must be perturbed n_u times at each MA iteration (here, $n_u = 3$). Additionally, it is required that the system reaches steady-state after every perturbation before computing the gradients. Based on previous knowledge, a sampling time of 60 seconds is enough to guarantee stationarity of the system after perturbations. Then the modifiers are computed as:

$$(\boldsymbol{\lambda}_{k+1}^J)^T = (1-d)(\boldsymbol{\lambda}_k^J)^T + d(\frac{\partial J_p}{\partial \mathbf{u}}(\mathbf{u}_k) - \frac{\partial J}{\partial \mathbf{u}}(\mathbf{u}_k)) \tag{38}$$

where d is the filter gain coefficient. The model gradient $\frac{\partial J}{\partial \mathbf{u}}(\mathbf{u}_k)$ is calculated from plant measurements of the gas flow rates and the valve openings. After obtaining $\boldsymbol{\lambda}^J$ and insert $\mathbf{y} = [Q_{l_1}\ Q_{l_2}\ Q_{l_3}]^T$ and $\mathbf{u} = [Q_{g_1}\ Q_{g_2}\ Q_{g_3}]^T$, the modified economic optimization problem

becomes:

$$\begin{aligned} \underset{Q_{g_1}, Q_{g_2}, Q_{g_3}}{\arg\max} \quad & J_{m,k} := 20Q_{l_1} + 10Q_{l_2} + 30Q_{l_3} + (\boldsymbol{\lambda}_k^J)^T [Q_{g_1}\ Q_{g_2}\ Q_{g_3}]^T \\ \text{s.t.} \quad & Q_{l_i} = \theta_{1_i} + \theta_{2_i} Q_{g_i} + \theta_{3_i} v_{o_{i,k}} + \theta_{4_i} Q_{g_i}^2 + \theta_{5_i} v_{o_{i,k}}^2, \quad i = 1, 2, 3, \\ & Q_{g_1} + Q_{g_2} + Q_{g_3} \leq 7.5 - h, \\ & 1 \leq Q_{g_1}, Q_{g_2}, Q_{g_3} \leq 5 \end{aligned} \tag{39}$$

$v_{o_{i,k}}$ is the measured valve opening of well i at the kth iteration. A backoff equal to the step size h in the perturbation is added to the gas availability constraint. Filters are applied to the optimal inputs. Algorithm 1 is lighting the most important steps of the algorithm. As described in Section 3, the inputs are the setpoints to the PI controllers used for controlling the gas flowrates. The "controller action" is implemented in the simulation by adding a 5 seconds delay to the inputs, in addition to input noise. The measurement noise in the simulation is drawn from a normal distribution with a variance σ^2. The variance of the noise was computed to represent the actual noise level in the experimental rig.

Algorithm 1 MA

for $k = 0 \rightarrow \infty$ **do**

1. Get steady-state measurements from the plant at $\mathbf{u}_k$. Calculate model gradient and evaluate the objective function.
2. Perturb each input individually $\rightarrow \mathbf{u}_j = \mathbf{u}_k + \mathrm{h}\mathbf{e}_j$
3. Get steady-state measurements from the plant at each of the perturbed points. Evaluate the objective function in each perturbed point.
4. Estimate gradients of the objective function w.r.t each input.
$\rightarrow \frac{\partial J_p}{\partial u_j}(\mathbf{u}_k) = [J_p(\mathbf{u}_k + h\mathbf{e}_j) - J_p(\mathbf{u}_k)]/h$
5. Update modifiers $\rightarrow (\boldsymbol{\lambda}_{k+1}^J)^T = (1-d)(\boldsymbol{\lambda}_k^J)^T + d(\frac{\partial J_p}{\partial \mathbf{u}}(\mathbf{u}_k) - \frac{\partial J}{\partial \mathbf{u}}(\mathbf{u}_k))$
6. Modify objective function $\rightarrow \mathrm{J}_{m,k}(\mathbf{u}) = \mathrm{J}(\mathbf{u}) + (\boldsymbol{\lambda}_k^J)^T \mathbf{u}$
7. Solve modified NLP, $\mathbf{u}_{k+1}^*$
8. Apply filter on optimal inputs $\rightarrow \mathbf{u}_{k+1} = \mathbf{u}_k + \mathbf{K}(\mathbf{u}_{k+1}^* - \mathbf{u}_k)$

end for

5.2 Results

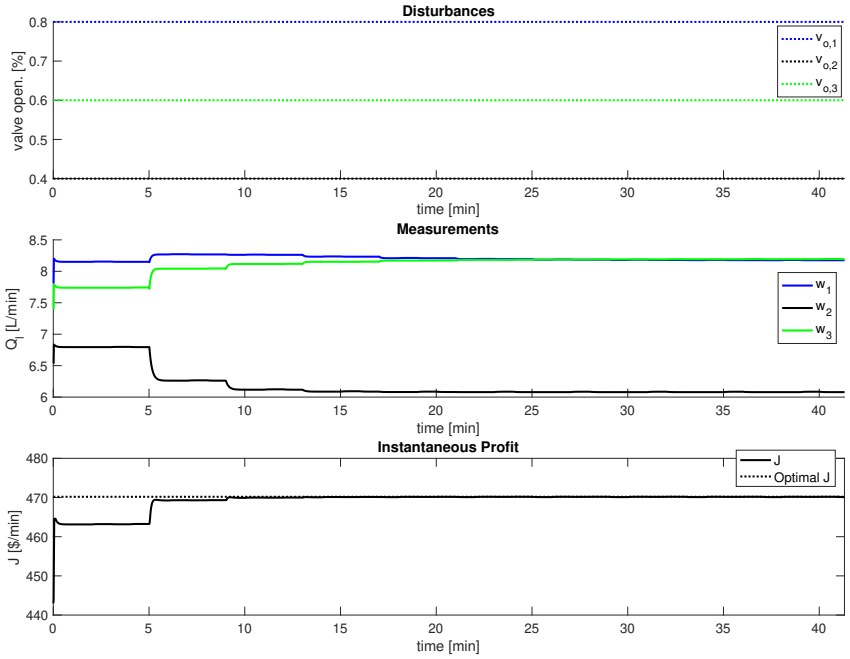
The results for the Modifier Adaptation implementation in the simulation case study are presented in this section. The algorithm is first applied to the case study without any noise. Then, it is applied to the system with both measurement noise and added controller action for the PI controllers. The Forward-finite-differencing approach is used to estimate the gradients. The tuning parameters for the simulations are shown in Table 6.

Table 6: Tuning parameters for MA.

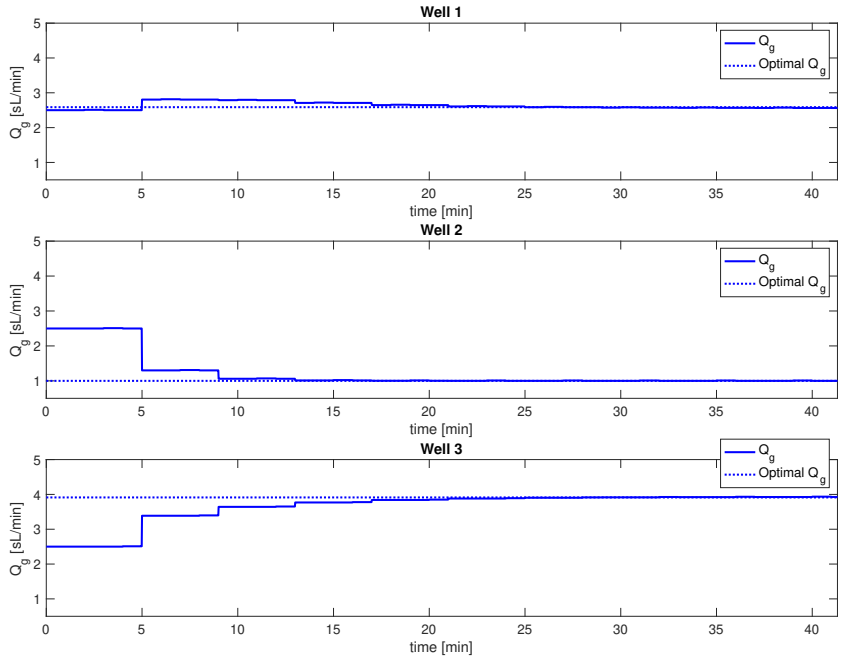
Parameter	**Description**	**Value**
K	$\mathbf{u}$ filter gain coefficient	0.8
d	$\boldsymbol{\lambda^J}$ filter gain coefficient	0.5

5.2.1 No noise

Figure 7a and 7b show the simulation results with no noise. This means that the measurements are perfect, as well as we do not have any controller delay, that is $Q_g = Q_{g,sp}$. Figure 7a shows the valve openings of each well, the liquid flow rates of each well and the objective function (instantaneous profit). The valve openings are held constant. Figure 7b shows the three inputs. The simulation shows that the method is able to drive the inputs to the optimal operating point, which was previously computed. The step size for the perturbation is set to $h = 0.01$ in this simulation because of no noise. Figure 8 shows the estimated gradients, as well as the true plant gradients.



(a) Disturbance profile, liquid flow rates and the objective function with MA and no noise.



(b) Implemented inputs with MA and no noise.

Figure 7: Simulation results with MA and no noise.

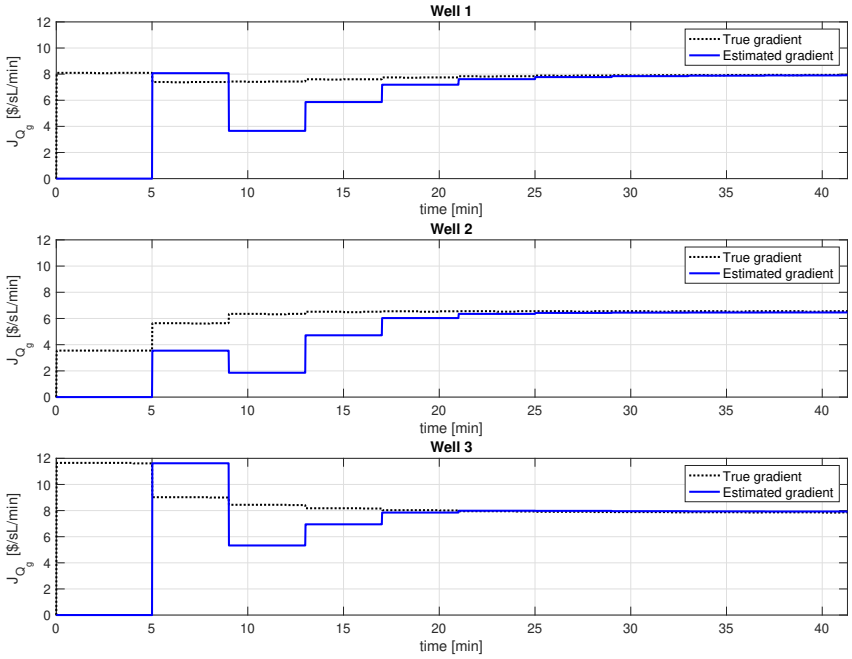
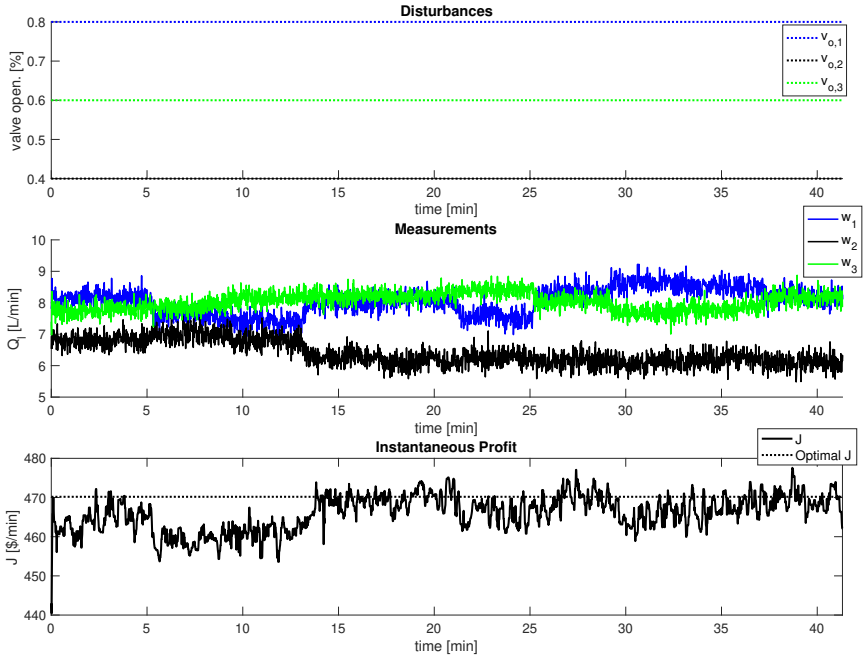


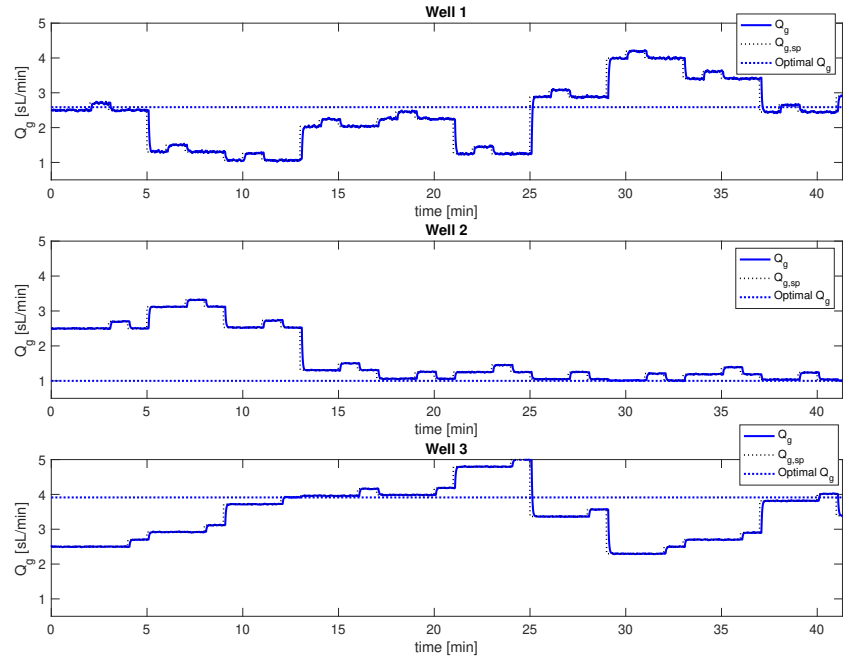
Figure 8: Estimated OF gradients with MA and no noise.

5.2.2 Noisy measurements

Figure 9a and 9b show the simulation results with added noise and controller action for the PI controller. A 10th-order one-dimensional median filter is applied to the measurement of the objective function in Figure 9a. The valve openings are still constant. Figure 10 shows the estimated gradients, as well as the true plant gradients. The step size for the perturbation is in this case increased to $h = 0.2$ because of significant measurement noise.



(a) Disturbance profile, liquid flow rates and the objective function with MA and added noise.



(b) Implemented inputs with MA and added noise.

Figure 9: Simulation results with MA. Measurement noise and controller delay is included.

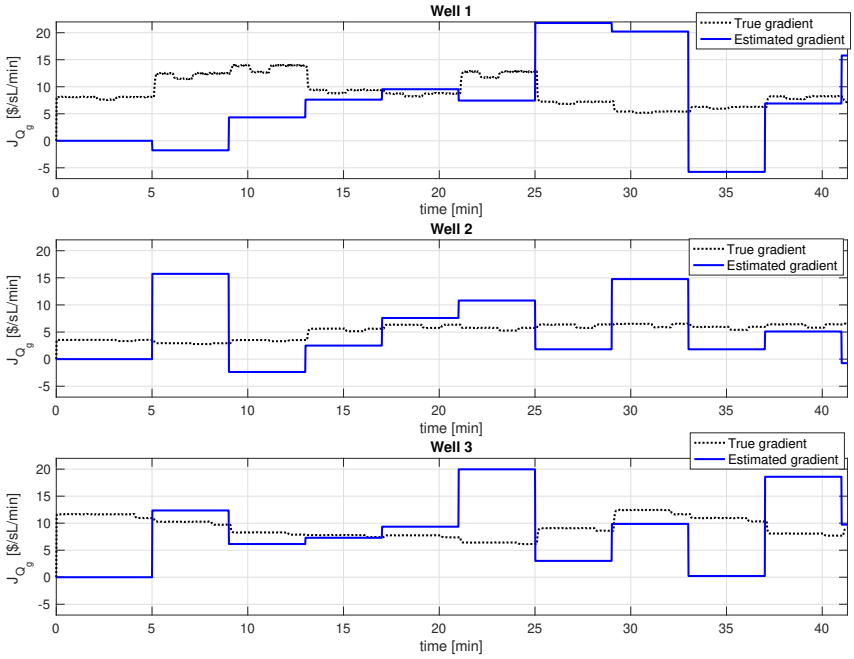


Figure 10: Estimated OF gradients with MA. Measurement noise and controller delay is included.

5.3 Discussion

The first simulation, without measurement noise, converged very nicely to plant optimum after some iterations. The third input has the highest optimal value because the liquid flow rate of well 3 is the most prioritized in the objective function. The liquid flow rate of well 2 is the least prioritized output in the objective function, which makes the second input the smallest and at its lower bound. The estimated gradients in this simulation almost overlapped with the true plant gradients in the end of the simulation. The perturbation of the inputs can not easily be seen from this simulation because of the small step size.

The perturbation is clearly shown in the second simulation with measurement noise and implemented controller delay. The inputs are perturbed individually, as can be seen in Figure 9b. For example at time 6, 7 and 8 minutes. The gradients and optimal inputs are then computed at time 9 minutes.

The second simulation showed that noise affects the performance significantly. Figure 9b shows that the inputs do not converge to plant optimum. Figure 10 shows the estimated objective function gradients. The gradients are always four minutes behind the true gradients, because of the chosen perturbation method. As an example, at time 17 minutes a new gradient is estimated. This gradient is representing the gradient of the plant four minutes before, at time 13 minutes. The estimated gradient at time 17 minutes for the first well is approximately equal to the true gradient at 13 minutes. However, it seems like

this is barely never the case in the rest of the plot. The gradient of the objective function w.r.t. the first and the second input are at some points negative, which does not make physical sense. There are a lot of uncertainty in these gradients. The objective function is a function of the measured outputs, which are affected by noise. The three outputs have different priority in the objective function due to economic reasons, which is indicated by using the weights 20, 10 and 30 respectively. Thus, accurate estimates of the objective function and its gradients are hard to obtain with a steady-state perturbation method like FFD.

The fact that the inputs converged to the optimal operating point in the simulation without noise shows that the MA method does not require an accurate model, since the true optimum is found by enforcing the plant KKT conditions. However, the challenge with MA lies in accurate gradient estimation. Instead of trying to improve the solution with standard MA, another case study with Output Modifier Adaptation was carried out, which is showed in the next section.

6 Output Modifier Adaptation Case Study

From the simulations with the standard Modifier Adaptation approach, it was seen that the simple model was enough to make the optimization method converge to plant optimum in the case of no measurement noise. The challenge lies in the gradient estimation. Output Modifier Adaptation is an alternative to the former standard approach, which does not calculate the gradient of the objective function. Instead, MAy uses gradients of the outputs. In addition, the Central-finite-differencing approach is used as perturbation method instead of FFD to improve accuracy.

6.1 Method

Before optimizing the system using MAy, zeroth-order modifiers $\boldsymbol{\varepsilon^y}$ and first-order modifiers $\boldsymbol{\lambda^y}$ need to be calculated. Ouput gradients of the plant need to be estimated for obtaining $\boldsymbol{\lambda^y}$. The CFD approach is used in this implementation, where each input is perturbed around the current operating point twice. Since the inputs are independent, they are perturbed at the same time. Then, the plant gradient $\frac{\partial y_{p,j}}{\partial u_j}(\mathbf{u}_k)$ is estimated as:

$$\frac{\partial y_{p,j}}{\partial u_j}(\mathbf{u}_k) = [y_{p,j}(\mathbf{u}_k + h\mathbf{e}_j) - y_{p,j}(\mathbf{u}_k - h\mathbf{e}_j)]/2h \tag{40}$$

where j is representing the jth input and k is representing the kth MAy iteration. The plant is perturbed twice in each MAy iteration. Steady-state has to be obtained after every perturbation before computing the gradients. A sampling time of 60 seconds is used in this implementation, which is enough to guarantee stationarity of the system. Then, the modifiers are computed as:

$$\begin{aligned} \boldsymbol{\varepsilon^y}_{k+1} &= (1-b)\boldsymbol{\varepsilon^y}_k + b(\mathbf{y}_p(\mathbf{u}_k) - \mathbf{y}(\mathbf{u}_k)) \\ (\boldsymbol{\lambda^y}_{k+1})^T &= (1-d)(\boldsymbol{\lambda^y}_k)^T + d(\frac{\partial \mathbf{y}_p}{\partial \mathbf{u}}(\mathbf{u}_k) - \frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)) \end{aligned} \tag{41}$$

where b is the $\boldsymbol{\varepsilon^y}$ filter gain coefficient and d is the $\boldsymbol{\lambda^y}$ filter gain coefficient. The ouput value of the model $\mathbf{y}(\mathbf{u}_k)$ and the model gradient $\frac{\partial y_j}{\partial u_j}(\mathbf{u}_k)$ is calculated from plant measurements of the gas flow rates and the valve openings. $\mathbf{y}_p(\mathbf{u}_k)$ is the measured output value. After obtaining the modifiers and insert $\mathbf{y} = [Q_{l_1}\ Q_{l_2}\ Q_{l_3}]^T$ and $\mathbf{u} = [Q_{g_1}\ Q_{g_2}\ Q_{g_3}]^T$, the modified

economic optimization problem becomes:

$$
\begin{aligned}
\max_{Q_{g_1}, Q_{g_2}, Q_{g_3}} \quad & J = 20Q_{l_{1,m,k}} + 10Q_{l_{2,m,k}} + 30Q_{l_{3,m,k}} \\
\text{s.t.} \quad & Q_{l_{i,m,k}} = \theta_{1_i} + \theta_{2_i} Q_{g_i} + \theta_{3_i} v_{o_{i,k}} + \theta_{4_i} Q_{g_i}^2 + \theta_{5_i} v_{o_{i,k}}^2 + \varepsilon_{i,k}^y + \lambda_{i,k}^y (Q_{g_i} - Q_{g_{i,k}}), \quad i = 1, 2, 3, \\
& Q_{g_1} + Q_{g_2} + Q_{g_3} \leq 7.5 - h, \\
& 1 \leq Q_{g_1}, Q_{g_2}, Q_{g_3} \leq 5
\end{aligned}
\tag{42}
$$

The step size of the perturbation is still $h = 0.2$. Therefore, a backoff equal to 0.2 is added to the gas availability constraint. $\varepsilon_{i,k}^y$ and $\lambda_{i,k}^y$ are the values of the modifiers of output i at the kth MAy iteration. $Q_{g_{i,k}}$ and $v_{o_{i,k}}$ are the measured gas rate and valve opening of well i at the kth iteration respectively. A filter is applied to the optimal inputs before they are implemented. Algorithm 2 summarizes the main steps in the MAy algorithm. The inputs are the setpoints to the PI controllers used for controlling the gas flowrates. The "controller action" is implemented in the simulation by adding a 5 seconds delay to the inputs, in addition to input noise. The measurement noise in the simulation is drawn from a normal distribution with a variance σ^2, which was computed to represent the actual noise level in the experimental rig.

Algorithm 2 MAy

for $k = 0 \rightarrow \infty$ **do**

1. Get steady-state measurements from the plant and model at $\mathbf{u}_k$. Calculate model gradient.
2. Perturb each input at the same time $\rightarrow \mathbf{u}_j = \mathbf{u}_k + h\mathbf{e}_j$ or $\mathbf{u}_j = \mathbf{u}_k - h\mathbf{e}_j$
3. Get steady-state measurements from the plant at each perturbed point.
4. Estimate output gradients of each well
$\rightarrow \frac{\partial y_{p,j}}{\partial u_j}(\mathbf{u}_k) = [y_{p,j}(\mathbf{u}_k + h\mathbf{e}_j) - y_{p,j}(\mathbf{u}_k - h\mathbf{e}_j)]/2h$
5. Update modifiers $\rightarrow \boldsymbol{\varepsilon}^{\boldsymbol{y}}{}_{k+1}$, $\boldsymbol{\lambda}^{\boldsymbol{y}}{}_{k+1}$
6. Modify outputs $\rightarrow \mathbf{y}_{m,k}(\mathbf{u}) = \mathbf{y}(\mathbf{u}) + \boldsymbol{\varepsilon}^{\boldsymbol{y}}{}_k + (\boldsymbol{\lambda}^{\boldsymbol{y}}{}_k)^T(\mathbf{u} - \mathbf{u}_k)$
7. Solve modified NLP, $\mathbf{u}_{k+1}^*$
8. Apply filter on optimal inputs $\rightarrow \mathbf{u}_{k+1} = \mathbf{u}_k + \mathbf{K}(\mathbf{u}_{k+1}^* - \mathbf{u}_k)$

end for

6.2 Results

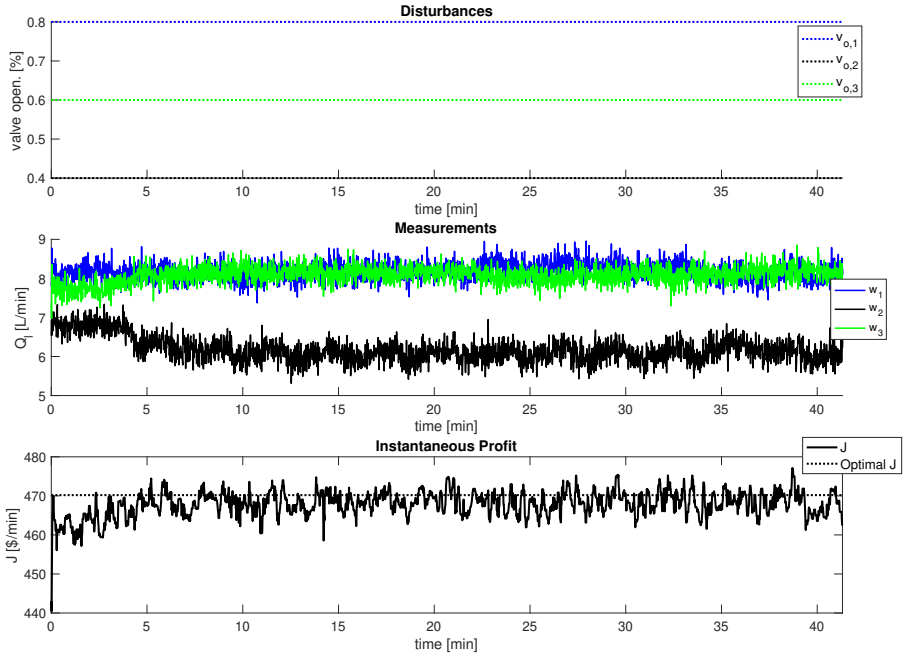
The simulation results for the MAy implementation are presented in this section. The algorithm is first applied to the case study with measurement noise and controller action for the PI controllers. The valve opening of each well is kept constant in this simulation to compare the results with the last simulation of MA. Then, step-wise disturbances are added to the system to see how well the algorithm is able to drive the system to the new optimum. Both simulations use the Central-finite-differencing approach to estimate plant gradients. The tuning parameters used in both simulations is shown in Table 7.

Table 7: Tuning parameters for MAy.

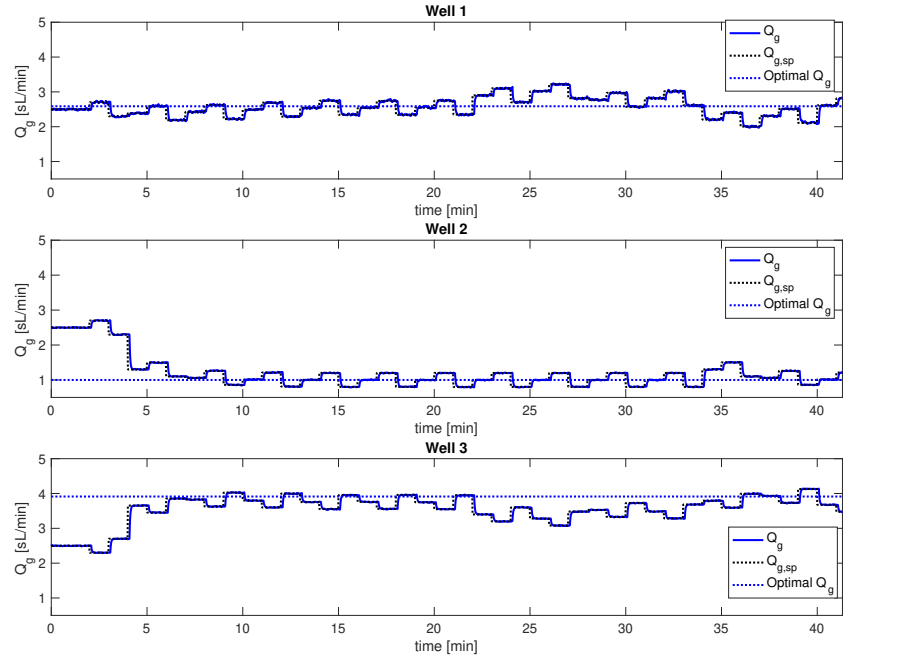
Parameter	**Description**	**Value**
K	$\mathbf{u}$ filter gain coefficient	0.8
b	$\boldsymbol{\varepsilon^y}$ filter gain coefficient	0.7
d	$\boldsymbol{\lambda^y}$ filter gain coefficient	0.3

6.2.1 Noisy measurements

Figure 11a shows the valve openings, the measured liquid flow rates and the objective function. The objective function is filtered with a 10-th-order one-dimensional median filter in the plot. The valve opening of each well is kept constant in this simulation. Figure 11b shows the implemented inputs and Figure 12 shows the estimated gradients, as well as the true plant gradients.



(a) Disturbance profile, liquid flowrated and the objective function with MAy.



(b) Implemented inputs with MAy.

Figure 11: Simulation results with MAy.

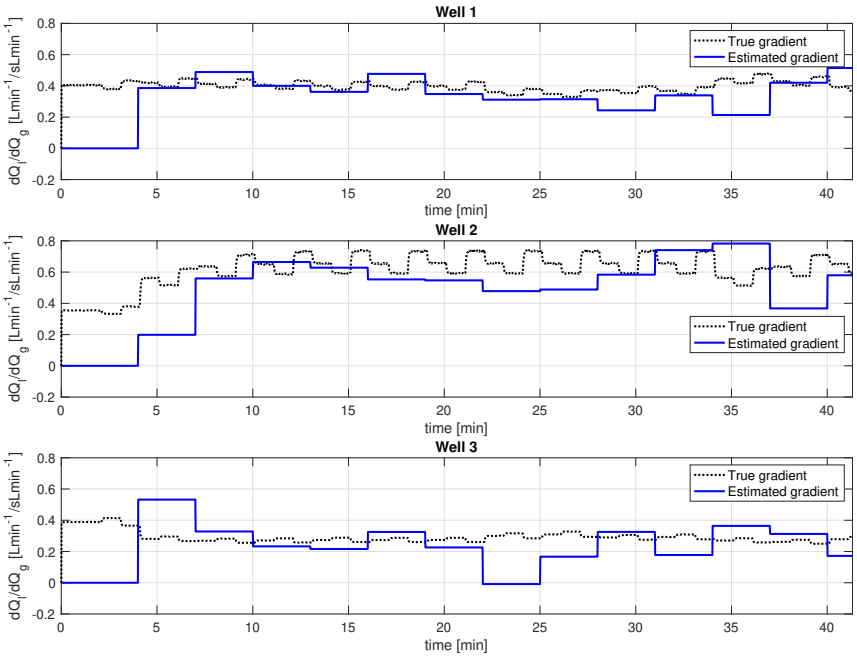
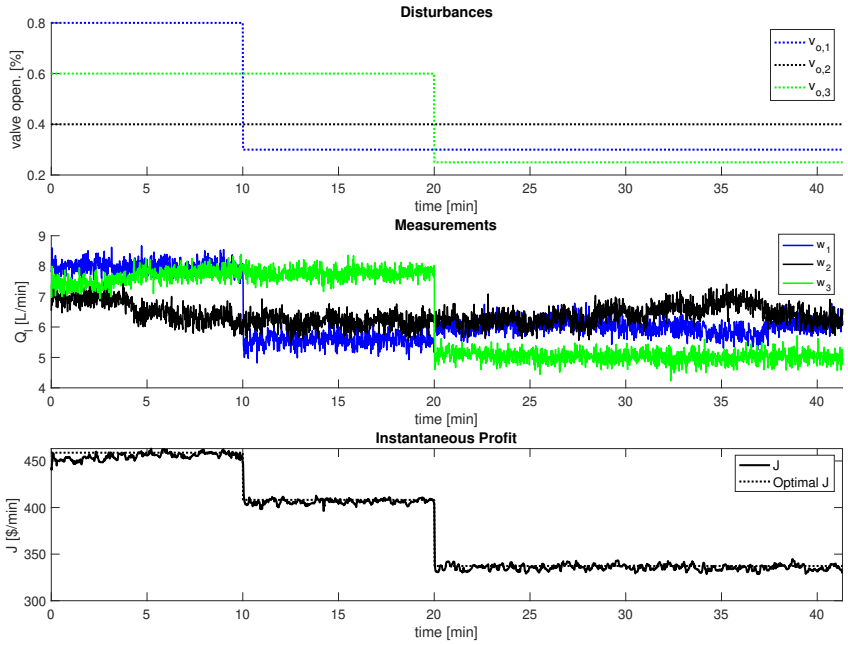


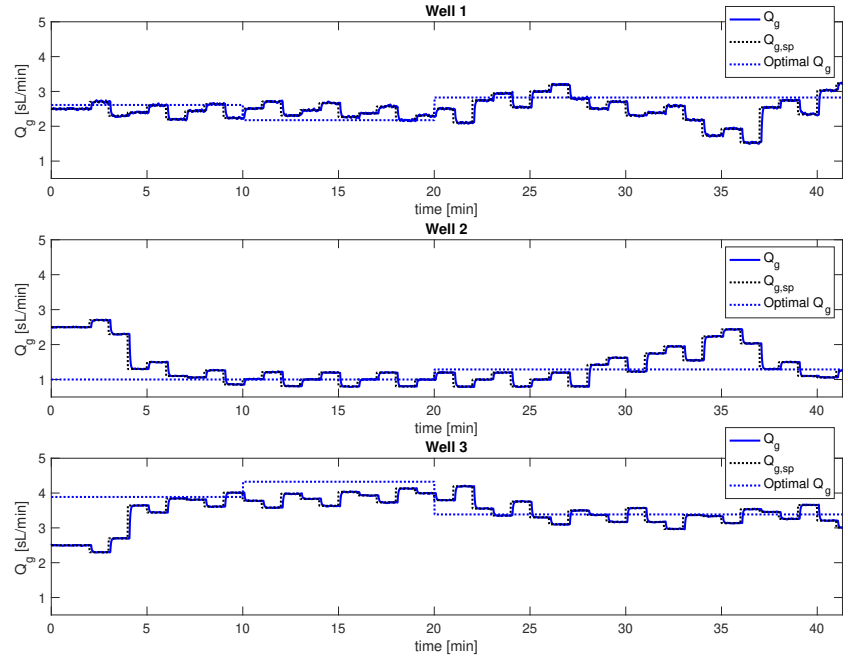
Figure 12: Estimated output gradients with MAy.

6.2.2 Noisy measurements + step-wise disturbances

In contrast to the previous simulations, the simulation in this section contains step-wise disturbances applied to the system. Figure 13 shows the simulation results for MAy implemented in the simulation case study with measurement noise and controller delay. Step-wise disturbances are applied to the system at time 10 min and time 20 min. The objective function is filtered with a 10th-order one-dimensional median filter in the plot. Figure 14a shows the estimated gradients, as well as the true gradient of the plant. Figure 14b shows the estimated output values by the algorithm, which is used for gradient estimation. The estimated output values are calculated as the mean values of the noisy measurements in the steady-state period, while the true output values are the actual values without noise.

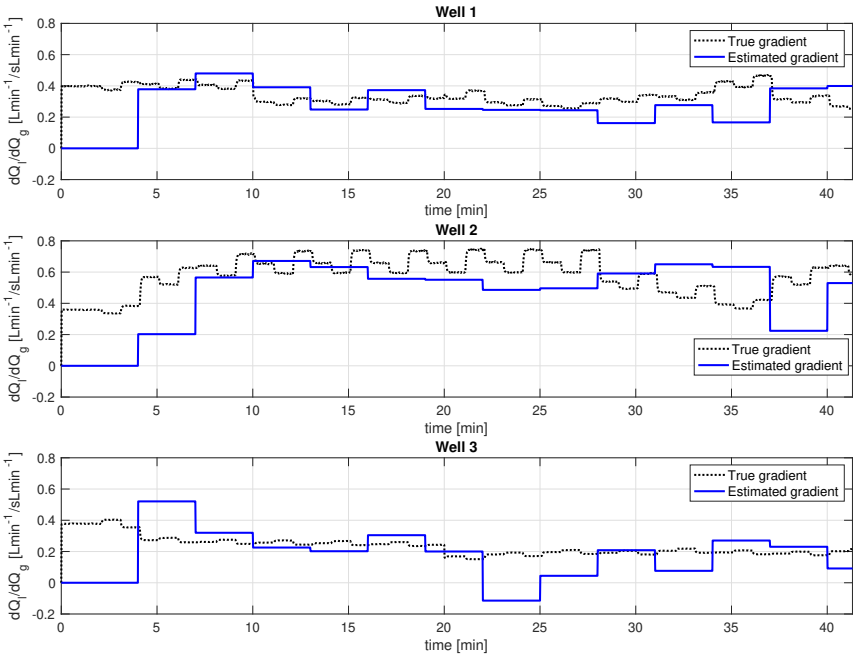


(a) Disturbance profile, liquid flowrates and the objective function with MAy. Step-wise disturbances are applied to the system.

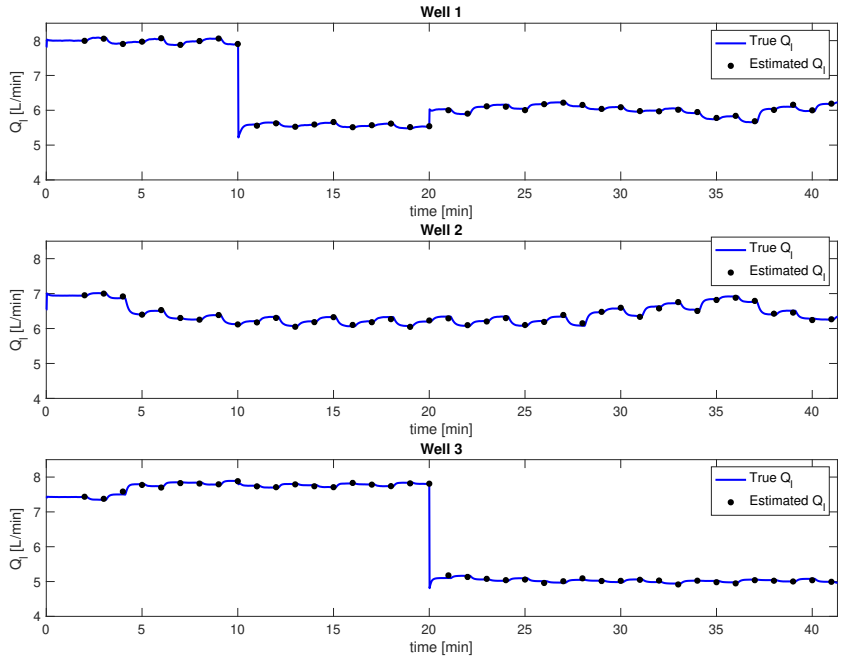


(b) Implemented inputs with MAy.

Figure 13: Simulation results with MAy. Step-wise disturbances are applied to the system.



(a) Estimated output gradients with MAy.



(b) Estimated output values used for gradient estimation with MAy.

Figure 14: Estimated outputs and gradients with MAy and step-wise disturbances.

6.3 Discussion

6.3.1 Simulation results

Both simulations starts from a suboptimal operating point where all the inputs are equal to 2.5 sL/min. After two perturbations, a new optimal operating point is calculated. The first optimal inputs are estimated at time 4 min. Then two other perturbations are carried out, and the next optimal inputs are hence estimated every 3 minutes. From the first MAy simulation with constant valve openings (no disturbances), Figure 11, it seems like the inputs converge more easily to the true optimum than the inputs with MA. The gradients in Figure 12 still have potential for improvement, but at least they are not negative as some of the gradients estimated in the MA algorithm.

Figure 13b shows the implemented inputs in the second MAy simulation with added disturbances. The inputs are converging to the first optimal operating point after some iterations. At time 10 min, Figure 13a shows that the valve opening of well 1 drops from 0.8% to 0.2%. As a result, the liquid flow of well 1 drops rapidly. The true optimal operating point changes, and it seems like the inputs are converging to the optimum after a while. At time 20 min, the valve opening of well 3 drops from 0.6% to 0.25%. As a consequence, the true optimum changes. The inputs seem to converge to the new optimum, but something unexpected is happening at approximately time 34 min. At time 34 min, the algorithm computes a new optimal input sequence which is far from the correct optimum. As seen from Figure 14a, the gradients of the wells are generally poor. The gradients are estimated from the operating point 3 minutes before, because of the chosen perturbation scheme. The gradients are especially poor around time 34 min, which leads to a sub optimal input sequence.

The gradient estimation is a severe challenge in the simulation. From Figure 14b it looks like the estimated outputs are almost identical to the true outputs. However, the gradients are very sensitive to measurement noise, and just a slight deviation from the true output gives poor gradients.

6.3.2 Tuning parameters

Since the greatest challenge in the MAy implementation is the gradient estimation, the tuning parameter d becomes the most decisive. This parameter is the $\boldsymbol{\lambda^y}$ filter gain coefficient. If this coefficient is high, less information of the previous calculated $\boldsymbol{\lambda^y}$ is used to calculate the new $\boldsymbol{\lambda^y}_k$. In the system of interest, d should be small, otherwise the inferred gradients can be inaccurate due to noise. A low value will avoid $\boldsymbol{\lambda^y}$ to overreact to measurement noise. However, a small value also leads to a slower response and a slower convergence to the optimum because less information of the current operating point is used in the modified optimization problem.

7 Trial Run in Experimental Lab Rig

A trial run in the experimental lab rig was carried out with MAy. In addition, a previously developed traditional steady-state RTO algorithm was run to compare the results. Both methods were run with a sampling time of 60 seconds. The MAy tuning and the disturbance profile is the same as in Section 6. The MATLAB code of the implemented MAy is shown in Appendix B.

7.1 Results

7.1.1 MAy trial run

Figure 15 shows the estimated output gradients of each well in the experimental run with MAy. Figure 16a shows the disturbance profile, the liquid measurements and the objective function. Figure 16b shows the implemented inputs and the estimated gradients.

7.1.2 MAy trial run and traditional RTO comparison

Figure 17a shows both the implemented inputs by the MAy algorithm and the implemented inputs by the traditional RTO algorithm. Figure 17b compares the total gas injection rate and the objective function of the two implementations.

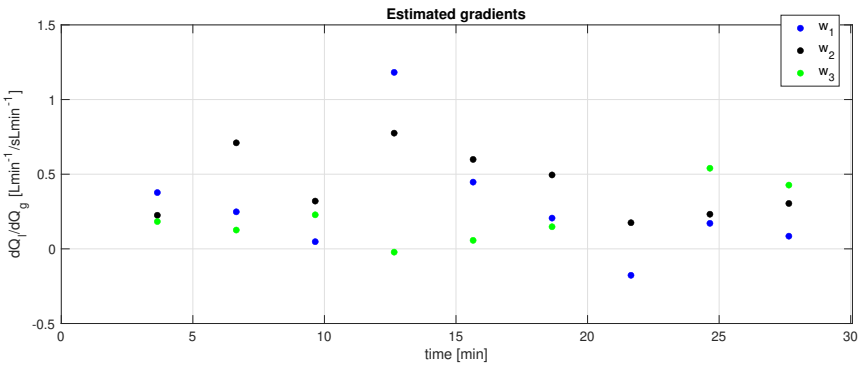
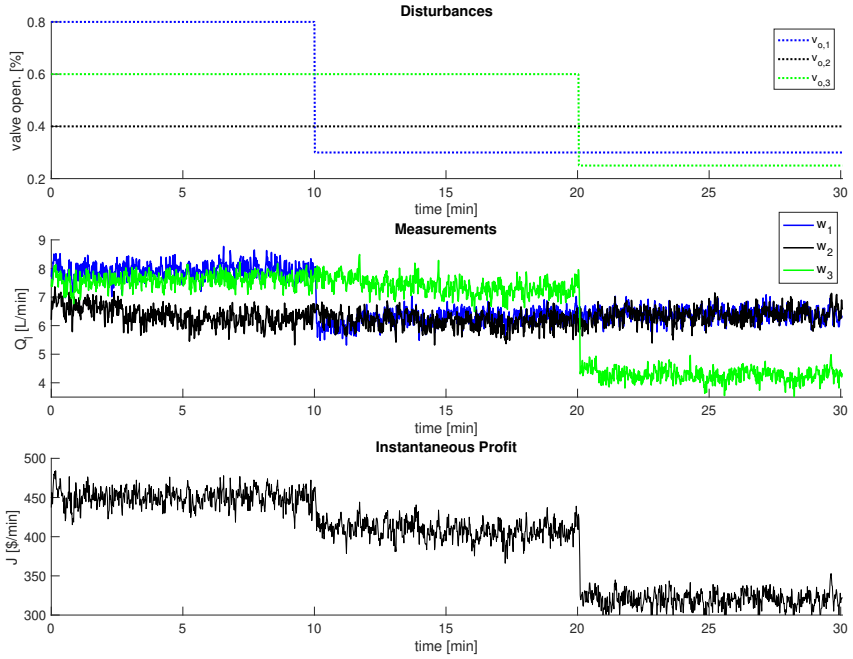
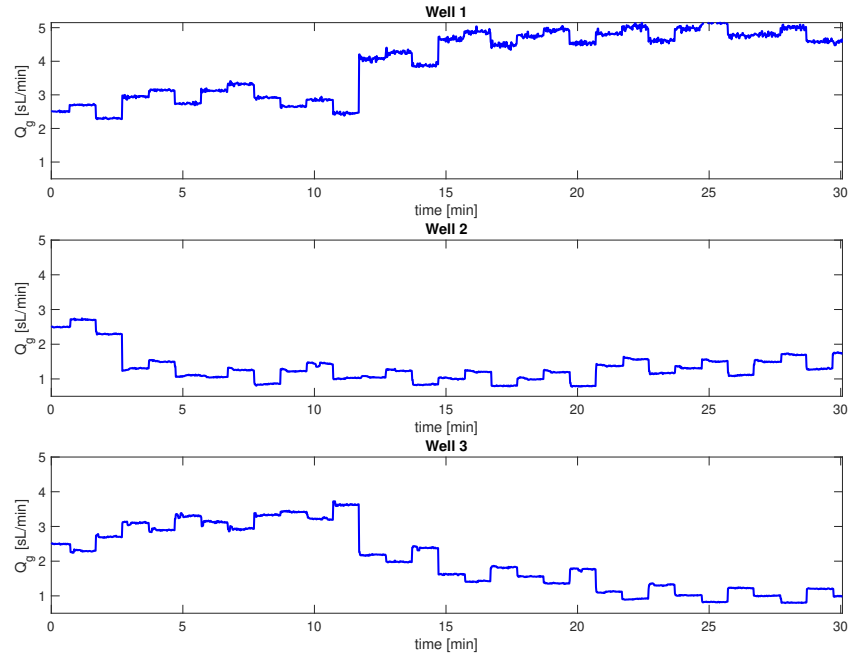


Figure 15: Estimated output gradients in the MAy experimental run.

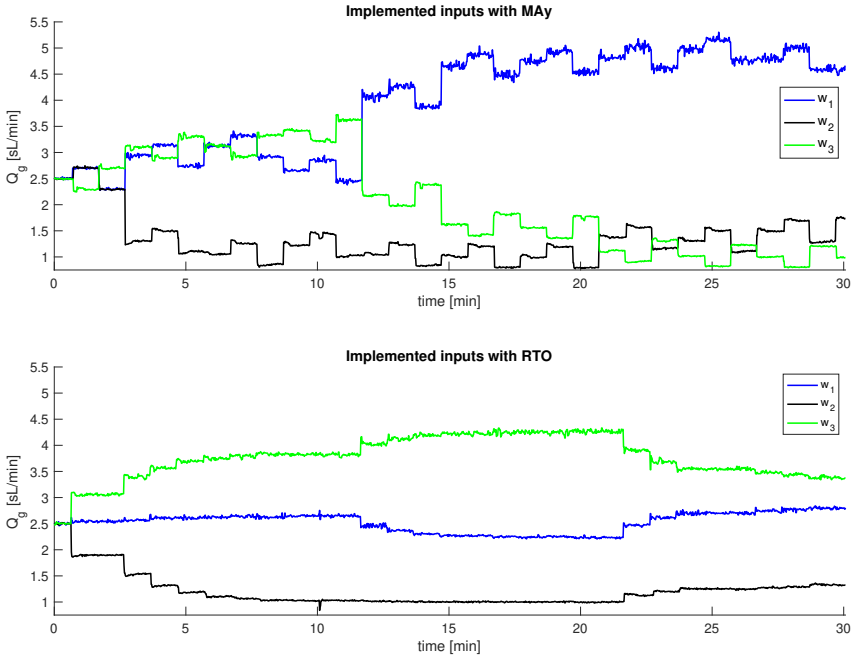


(a) Disturbance profile, liquid flowrates and the objective function from the experiment with MAy.

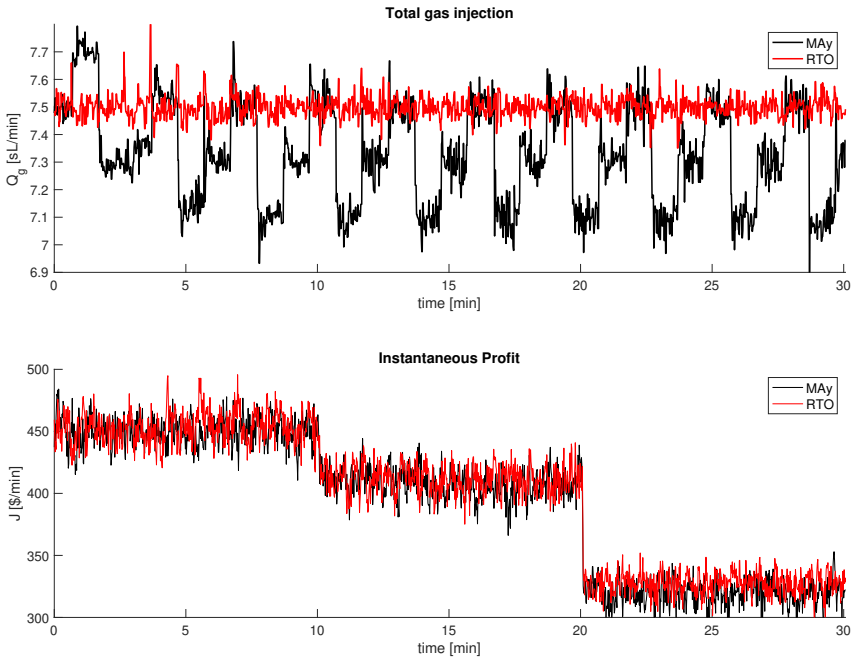


(b) Implemented input profiles in the experiment with MAy.

Figure 16: Experimental results with MAy.



(a) Comparison of the implemented inputs with MAy and traditional RTO.



(b) Comparison of the total gas injection rate and the instantaneous profit with MAy and traditional RTO.

Figure 17: Comparison of MAy and traditional RTO.

7.2 Discussion

7.2.1 MAy trial run

The disturbance profile is the same as in the last simulation with MAy. At time 10 min, the valve opening of the first well drops. Then at time 20 min, the third valve opening drops. The implemented input of well 3 seems to converge to a higher level than the input of well 1 in the first disturbance region, which was also observed in the second simulation with MAy in Section 6.2.2. This is expected since the liquid flow rate in well 3 is prioritized over the liquid flow rate in well 2 in the objective function. In the second region, the input of well 3 drops and converge to a very low level. This is not expected, but can be explained by the estimated gradients. In the second disturbance region, the estimated gradient of the third well is significantly smaller than the other gradients. In the last disturbance region, the input of well 3 drops to an even lower level. It is expected that the input drops when the valve opening of the respective well drops. However, the liquid flow rate of well 3 is still the most prioritized in the objective function, so the third input should not converge to its lower bound.

7.2.2 MAy trial run and traditional RTO comparison

The implemented input profiles with MAy and traditional RTO are very different. The inputs implemented by RTO are not perturbed and converge very nicely to a computed optimum. From Figure 17a the input of the third well is always larger than the first, and the second is always the smallest. This is expected from the weights in the objective function and were also observed in the second simulation of MAy in Section 6.2.2. When the valve openings drop, the corresponding input also drops.

Even though the input profiles of the two experimental runs are very different, the profit is not significantly affected. However, we can still see that the profit of the MAy run is lower than the RTO run. The optimization problem has a constraint on the total gas injection rate. The total injection rate from both experiments are shown in Figure 17b. The MAy algorithm violates the constraint in the first perturbation, because the constraint are not introduced until the first optimization. The other violations are due to measurement error in both experiments. Because of the perturbation in the MAy run, the total gas injection drops to 7.1 sL/min every 3 minutes.

8 Conclusion

In this specialization project a steady-state model of an experimental rig was estimated to be used in the implementation of Modifier Adaptation and Output Modifier Adaptation. The experimental rig represents a gas lifted well network with three parallel wells. The model of each well was obtained from steady-state data from the rig and statistical methods.

Both MA and MAy were applied to the obtained model and implemented in a simulation case study of the experimental rig. The simulation results showed that the obtained simple model was able to drive the system to its optimum. However, the simulations also showed that the core challenge in both methods lies in accurate gradient estimation in the presence of noisy measurements. A trial run in the actual rig was carried out in the end of the project to test the MAy implementation and compare the results with a previously developed traditional RTO algorithm. The results showed that the MAy implementation still has a huge potential for improvement.

As the greatest challenge in MA application is obtaining accurate gradient estimates, the next step should be to investigate other gradient estimation methods. Paper [2] describes several ways to estimate gradients. In addition to steady-state perturbation methods, it describes some dynamic perturbation methods such as dynamic model identification, in which the parameters of a simple dynamic input-output model are updated online based on transient plant data. Then, the steady-state gradients are obtained by application of the final-value theorem. To guarantee that the transient data is informative enough, the inputs need to be perturbed periodically (e.g. a sinusoidal dither) to excite the system such that the model parameters can be estimated properly. Dynamic perturbation methods will be considered and further investigated in the context of MA. Another alternative is to combine the simple model of the rig with data-based models, such as Gaussian process [8] to capture plant-model mismatch. As a consequence, the gradient estimation step is avoided and the MAy performance enhanced. This future work can be valuable to whether MA methods can be used for optimizing subsea oil well networks.

Bibliography

[1] Dinesh Krishnamoorthy. 'Novel approaches to online process optimization under uncertainty'. In: *Doktoravhandlinger ved NTNU.* (2019).

[2] Alejandro Marchetti et al. 'Modifier Adaptation for Real-Time Optimization – Methods and Applications'. In: *Processes* 4 (Dec. 2016). DOI: 10.3390/pr4040055.

[3] A. Marchetti, B. Chachuat and D. Bonvin. 'Modifier-Adaptation Methodology for Real-Time Optimization'. In: *Industrial & Engineering Chemistry Research* 48.13 (2009), pp. 6022–6033. DOI: 10.1021/ie801352x. eprint: https://doi.org/10.1021/ie801352x. URL: https://doi.org/10.1021/ie801352x.

[4] George E. P. Box and Norman R. Draper. *Response Surfaces, Mixtures, and Ridge Analyses*. Hoboken, New Jersey: Wiley, 2007.

[5] H.T. Banks and Michele L. Joyner. 'AIC under the framework of least squares estimation'. In: *Applied Mathematics Letters* 74 (2017), pp. 33–45. ISSN: 0893-9659. DOI: https://doi.org/10.1016/j.aml.2017.05.005.

[6] José Matias. 'Implementation of RTO using transient measurements on experimental rig'. In: (2020).

[7] Dinesh Krishnamoorthy, Bjarne Foss and Sigurd Skogestad. 'Real-Time Optimization under Uncertainty Applied to a Gas Lifted Well Network'. In: *Processes* 4.4 (2016). ISSN: 2227-9717. DOI: 10.3390/pr4040052. URL: https://www.mdpi.com/2227-9717/4/4/52.

[8] Ehecatl Antonio del Rio Chanona et al. 'Real-time optimization meets Bayesian optimization and derivative-free optimization: A tale of modifier adaptation'. In: *Computers & Chemical Engineering* 147 (2021), p. 107249.

A Experimental rig modelling

The model uses mass balances of the different phases, density models, pressure models and flow models, which become the following differential algebraic equation (DAE)

$$\begin{aligned} \dot{\mathbf{x}}_i &= \mathbf{f}_i(\mathbf{x}_i, \mathbf{z}_i, \mathbf{u}_i, \mathbf{p}_i) \\ \mathbf{g}_i(\mathbf{x}_i, \mathbf{z}_i, \mathbf{u}_i, \mathbf{p}_i) &= \mathbf{0} \quad \forall i \in \mathcal{N} = \{1, \ldots, n_w\} \end{aligned} \tag{43}$$

where $f_i(x_i, z_i, u_i, p_i)$ is the set of differential equations and $g_i(x_i, z_i, u_i, p_i)$ is the set of algebraic equations. n_w is the number of wells, i.e. n_w=3 in this case. The subscript i is referring to a well in the set $\mathcal{N}$. x_i is the differential states, z_i is the algebraic states, u_i is the decision variables.

The differential states, the algebraic states and the decision variables are given by

$$\begin{aligned} \mathbf{x}_i &= [m_{g_i} \; m_{l_i}]^T \\ \mathbf{z}_i &= [w_{l_i} \; w_{total_i} \; p_{rh_i} \; p_{bi_i} \; \rho_{mix} \; \rho_{g_i} \; w_{gout_i} \; w_{lout_i}]^T \\ \mathbf{u}_i &= [Q_{g_i} \; v_{o_i} \; p_{pump}]^T \end{aligned} \tag{44}$$

where m_{g_i} is the gas hold up and m_{l_i} is the liquid hold up. [w_{l_i} is the water rate from the reservoir and w_{total_i} is the total well production rate. p_{rh_i} is the riser head pressure and p_{bi_i} is the pressure before injection point. ρ_{mix} is the mixture density in the system and ρ_{g_i} is the density of the gas. w_{gout_i} and w_{lout_i} is the well outlet gas and liquid outlet flowrate respectively. Q_{g_i} is the gas lift injection rate, v_{o_i} is the valve opening from the reservoir and p_{pump} is the reservoir pressure.

In addition, the model contains some constant parameters and some uncertain parameters. These parameters are given by

$$\mathbf{p}_i = [p_{atm} \; T \; R \; M_w \; \rho_l \; \mu_{mix} \; L_w \; A_w \; L_r \; H_r \; D_{bh} \; \theta_{res} \; \theta_{top}]^T$$

where p_{atm} is the atmospheric pressure, T is the room temperature, R is the gas constant, M_w is the molecular weigth of air, ρ_l is the desity of water, μ_{mix} is the mixture viscosity, L_w is the well length, A_w is the well pipes cross section area, L_r is the riser length, A_r is the riser pipes cross section area, H_r is the riser height and D is well below injection. θ_{res} is the reservoir valve flow coefficient and θ_{top} is the top valve flow coefficient ([6]).

The code for the dynamic model is shown in the section below. The differential equations are $\mathbf{f} = [df_1 \; df_2]^T$, while the algebraic equations are $\mathbf{g} = [f_1 \; f_2 \; f_3 \; f_4 \; f_5 \; f_6 \; f_7 \; f_8]^T$.

A.1 ErosionRigDynModel.m

```
function [F,S_xx,S_zz,S_xz,S_xp,S_zp,x_var,z_var,u_var,p_var,diff,alg,L]
↪  = ErosionRigDynModel(par)
%    Creates a dynamic model of the rig and computes the sensitivity
%    matrices for EKF

% Inputs:
%    par = system parameters
%
% Outputs:
%   F: system integrator
%   S's: system sensitivities
%   x_var,z_var,u_var,p_var, diff,alg,L: Model in CasADi form

% Other m-files required: none
% MAT-files required: none

addpath('/Applications/casadi-osx-matlabR2014b-v3.5.5')
import casadi.*

%% Parameters
%number of wells
n_w = par.n_w; %[]
%gas constant
R = par.R; %[m3 Pa K^-1 mol^-1]
%air molecular weigth
Mg = par.Mw; %[kg/mol] -- Attention: this unit is not usual

%properties
%density of water - dim:  nwells x 1
rho_l = par.rho_o; %[kg/m3]
%mixture viscosity
mu_mix = par.mu_oil;% [Pa s]

%project - dim:  nwells x 1
% well length
L_w = par.L_w; %[m]
% well pipes cross section area
A_w = par.A_w;%[m2]
```

```
% riser length
L_r = par.L_r; %[m]
% riser pipes cross section area
A_r = par.A_r;%[m2]
%riser height
H_r = par.H_r; %[m]
%well below injection
D = par.D_bh; %[m]

%% System states
% differential
%gas holdup
m_g = MX.sym('m_g',n_w);            % 1:3 [1e-4 kg]
%water holdup
m_l = MX.sym('m_l',n_w);            % 4:6 [kg]

% algebraic
%water rate from reservoir
w_l = MX.sym('w_l',n_w);            % 1:3 [1e-2 kg/s]
%total well production rate
w_total = MX.sym('w_total',n_w);    % 4:6 [1e-2 kg/s]

%riser head pressure
p_rh = MX.sym('p_rh',n_w);          % 7:9 [bar]
%pressure - before injection point (bottom hole)
p_bi = MX.sym('p_bi',n_w);          % 10:12 [bar]

%mixture density in system
rho_mix =  MX.sym('rho_mix',n_w);   % 13:15 [1e2 kg/m3]
%density gas
rho_g= MX.sym('rho_g',n_w);         % 16:18 [kg/m3]

%well outlet flowrate (gas)
w_gout = MX.sym('w_gout',n_w);      % 19:21 [1e-5 kg/s]
%riser head gas production rate gas
w_lout = MX.sym('w_lout',n_w);      % 22:24 [1e-2 kg/s]

%% System input
%gas lift rate
Q_gl = MX.sym('Q_gl',n_w);        % 1:3 [sL/min]
%valve oppening
```

```
vo = MX.sym('vo',n_w);              % 4:6 [0-1]
%pump outlet pressure
Ppump = MX.sym('Ppump',1);          % 7 [bar]

%% parameters
%%%%%%%%
% fixed %
%%%%%%%%
%room temperature
T = MX.sym('T',1); %[K]
%atmospheric pressure
p_atm = MX.sym('p_atm',1); %[bar]

%time transformation: CASADI integrates always from 0 to 1 and the USER
↪  does the time
%scaling with T --> sampling time
t_samp = MX.sym('t_samp',1); %[s]

% estimable
%scaled reservoir valve parameters
res_theta = MX.sym('res_theta',n_w);
%scaled top valve parameters
top_theta = MX.sym('top_theta',n_w);

%% Modeling
% Algebraic
%conversion
CR = 60*10^3; % [L/min] -> [m3/s]
%reservoir outflow
f1 = -Ppump*ones(n_w,1)*1e5 +
↪  (w_l.*1e-2).^2.*(res_theta.*1e9)./(vo.^2.*rho_l) + p_bi.*1e5 ;
% total system production
f2 = - (w_total.*1e-2) + ((w_gout.*1e-5) + (w_lout.*1e-2));
%riser head pressure
f3 = -p_rh.*1e5 + (w_total.*1e-2).^2.*(top_theta.*1e8)./(rho_mix.*1e2) +
↪  p_atm.*1e5 ;
%before injection pressure
f4 = -p_bi.*1e5 + (p_rh.*1e5 + (rho_mix.*1e2).*9.81.*H_r +
↪  128.*mu_mix.*(L_w+L_r).*(w_l.*1e-2)./(3.14.*D.^4.*(rho_mix.*1e2)));
%mixture density
```

```
f5 = -(rho_mix.*1e2) + (((m_g.*1e-4) + m_l).*
↪  p_bi.*1e5.*Mg.*rho_l)./(m_l.*p_bi.*1e5.*Mg +
↪  rho_l.*R.*T.*(m_g.*1e-4));
%gas density (ideal gas law)
f6 = -rho_g + p_bi.*1e5.*Mg/(R*T);

% Simplifying assumption!
% liquid fraction in the mixture
xL = (m_l./((m_g.*1e-4) + m_l));
%Liquid outlet flowrate
f7 = -(w_lout.*1e-2) + xL.*(w_total.*1e-2);

% Total volume constraint
f8 = -(A_w.*L_w + A_r.*L_r) + (m_l./rho_l + (m_g.*1e-4)./rho_g);

% Differential
% gas mass balance
df1=  1e4*(-(w_gout.*1e-5) + Q_gl.*rho_g/CR);
% liquid mass balance
df2= -(w_lout.*1e-2) + (w_l.*1e-2);

% Form the DAE system
diff = vertcat(df1,df2);
alg = vertcat(f1,f2,f3,f4,f5,f6,f7,f8);

% give fixed parameter values
alg = substitute(alg,p_atm,par.p_s);
alg = substitute(alg,T,par.T_r);

% concatenate the differential and algebraic states
x_var = vertcat(m_g,m_l);
z_var = vertcat(w_l,w_total,p_rh,p_bi,rho_mix,rho_g,w_gout,w_lout);
u_var = vertcat(Q_gl,vo,Ppump);
p_var = vertcat(res_theta,top_theta,t_samp);

%objective function
L = 20*((w_lout(1)*1e-2)*CR/rho_l(1)) + 10*((w_lout(2)*1e-2)*CR/rho_l(2))
↪  + 30*((w_lout(3)*1e-2)*CR/rho_l(3));
%end modeling

%% Casadi commands
```

```
%declaring function in standard DAE form (scaled time)
dae =
↪  struct('x',x_var,'z',z_var,'p',vertcat(u_var,p_var),'ode',t_samp*diff,'alg',alg);

%calling the integrator, the necessary inputs are: label; integrator;
↪  function with IO scheme of a DAE (formalized); struct (options)
F = integrator('F','idas',dae);

% ================================================
%       Calculating sensitivity matrices
% ================================================

S_xx = F.factory('sensStaStates',{'x0','z0','p'},{'jac:xf:x0'});
S_zz = F.factory('sensStaStates',{'x0','z0','p'},{'jac:zf:z0'});
S_xz = F.factory('sensStaStates',{'x0','z0','p'},{'jac:xf:z0'});

S_xp = F.factory('sensParStates',{'x0','z0','p'},{'jac:xf:p'});
S_zp = F.factory('sensParStates',{'x0','z0','p'},{'jac:zf:p'});

end
```

B MATLAB Code MAy

InitializationLabViewMain.m sets the sampling time in the experiment. *ssModel.m* contains the model used for optimization. *LabViewMain.m* contains the MAy algorithm.

B.1 InitializationLabViewMain.m

```
%clear
%clc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% BE CAREFUL - it should match sampling time in LABVIEW interface %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Optimization sampling time
nExec = 60; %[s] 10
```

B.2 ssModel.m

```
function [y_model_ss,modelGrad] = ssModel(u, v0)
   % u = [FIC104; FIC105; FIC106]
    % v0 = [CV101; CV102; CV103]
    % J = 20*y1 + 10*y2 + 30*y3

    theta_well1 = [0.492764910641654; 0.300443533757929; 19.3122131197183;
                   -0.0153700130556438; -14.3380841612070];
    theta_well2 = [-0.273738047663130; 0.459291990662657;
    ↪  20.1022897064108;
                   -0.0373720166243209; -14.6387961216287];
    theta_well3 = [-0.883020415785438; 0.619837772512701;
    ↪  21.3127342608823;
                   -0.0640840666704814; -15.7140573084694];

    y_well1_ss = theta_well1(1) + theta_well1(2)*u(1) +
    ↪  theta_well1(3)*v0(1) ...
                + theta_well1(4)*u(1)^2 + theta_well1(5)*v0(1)^2;

    y_well2_ss = theta_well2(1) + theta_well2(2)*u(2) +
    ↪  theta_well2(3)*v0(2) ...
                + theta_well2(4)*u(2)^2 + theta_well2(5)*v0(2)^2;
```

```
    y_well3_ss = theta_well3(1) + theta_well3(2)*u(3) +
    ↪  theta_well3(3)*v0(3) ...
                + theta_well3(4)*u(3)^2 + theta_well3(5)*v0(3)^2;

    modelGrad_well1 = theta_well1(2) + 2*theta_well1(4)*u(1);
    modelGrad_well2 = theta_well2(2) + 2*theta_well2(4)*u(2);
    modelGrad_well3 = theta_well3(2) + 2*theta_well3(4)*u(3);

    y_model_ss = [y_well1_ss; y_well2_ss; y_well3_ss];
    % dydu
    modelGrad = [modelGrad_well1; modelGrad_well2; modelGrad_well3];

end
```

B.3 LabViewMain.m

```
% Main program
% Run Initialization file first
addpath ('C:\Users\lab\Documents\casadi-windows-matlabR2016a-v3.4.5')
import casadi.*

%%%%%%%%%%%%%%%%
% Get Variables
%%%%%%%%%%%%%%%%
% disturbances
%valve opening [-]
cv101 = P_vector(1);
cv102 = P_vector(2);
cv103 = P_vector(3);
% if value is [A] from 0.004 to 0.020
% if you want to convert to 0 (fully closed) to 1 (fully open)
% vo_n = (vo - 0.004)./(0.02 - 0.004);

%pump rotation [%]
pRate = P_vector(4);
% if value is [A] from 0.004 to 0.020
% if you want to convert to (min speed - max speed)
% goes from 12% of the max speed to 92% of the max speed
% pRate = 12 + (92 - 12)*(P_vector(4) - 0.004)./(0.02 - 0.004);
```

```
% 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 = 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 VALUE 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);

% number of measurements in the data window
dss = size(pi104,2);

%%%%%%%%%%%%%%%%%%
% CODE GOES HERE %
%%%%%%%%%%%%%%%%%%
% check for first iteration
if ~exist('k','var')
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % BE CAREFUL - it should match sampling time in LABVIEW interface %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Optimization sampling time
    nExec = 60; %[s] 10

    % Modifier Adaptation
    % Filters
    K = 0.8;
    d_lam = 0.3;
    b_eps = 0.7;

    % Modifiers
    lam_k = 0;
    eps_k = 0;

    % Other things

    % Perturbation 0, optimization 1
```

```
    flagOpt = 0;

    % For steady state detection
    k = 1;

    flagSS = 1;

    % Perturbation step
    h = 0.2;

end

% run SS identification
% relevant measurements are only the liquid flowrates
yPlant = [fi101;fi102;fi103];

if flagSS == 1 % we are at steady state at the current instant
    % excitation or optimization - flagOpt == 1 optimization, flagOpt ==
    ↪  0
    % excitation
    uPlant = [fic104;
              fic105;
              fic106;
              cv101*ones(1,dss); %workaround - just have the last
              ↪  measurement here. Since it is the disturbance, it
              ↪  doesn't really matter;
              cv102*ones(1,dss);
              cv103*ones(1,dss)];

    uEst = mean(uPlant(:,end - nExec/2:end),2);
    yEst = mean(yPlant(:,end - (nExec/2+10):end),2);

    if flagOpt == 0
        switch k
            case 1
                uEstOld = uEst;
                yEstOld = yEst;

                [ymodelssOld, modelGradOld] = ssModel(uEst(1:3),
                ↪  uEst(4:6));
                modelGradOld = diag(modelGradOld); % nu x ny
```

```
            % save old optimal values
            O_vector_old = uOold;

            O_vector = O_vector_old' + [h,h,-h];

            k = 2;

        case 2
            yEsth = yEst;

            O_vector = O_vector_old' + [-h,-h,h];

            flagOpt = 1;
            k = 1;
    end

    SS = 1;
    Estimation = 0;
    Optimization = 0;
    Result = 0;
    Parameter_Estimation = [0,0,0,0,0,0];
    State_Variables_Estimation = [0,0,0,0,0,0];
    State_Variables_Optimization = [0,0,0,0,0,0];
    Optimized_Air_Injection = [yEst(1),yEst(2),yEst(3)];

else
    flagOpt = 0;

    yEst2h = yEst;

    plantGrad_est = [(yEsth(1)-yEst2h(1))/(2*h);
    ↪  (yEsth(2)-yEst2h(2))/(2*h); (yEst2h(3)-yEsth(3))/(2*h)];
    % Estimate of output gradient
    plantGrad_estimated = diag([(yEsth(1)-yEst2h(1))/(2*h);
    ↪  (yEsth(2)-yEst2h(2))/(2*h); (yEst2h(3)-yEsth(3))/(2*h)]);

    % Filter on modifier
    lam_k = (1-d_lam)*lam_k + d_lam*(plantGrad_estimated -
    ↪  modelGradOld); % nu x ny
    eps_k = (1-b_eps)*eps_k + b_eps*(yEstOld - ymodelssOld);
```

```
% Symbols to the optimization problem
u = MX.sym('u',3); % FIC104SP, FIC105SP, FIC106SP
y = MX.sym('x',3); % FI101, FI102, FI103

% Ss model

ss_model_well1 = theta_well1(1) + theta_well1(2)*u(1) +
↪  theta_well1(3)*uEst(4) ...
        + theta_well1(4)*u(1)^2 + theta_well1(5)*uEst(4)^2 -
        ↪  y(1);

ss_model_well2 = theta_well2(1) + theta_well2(2)*u(2) +
↪  theta_well2(3)*uEst(5) ...
        + theta_well2(4)*u(2)^2 + theta_well2(5)*uEst(5)^2 -
        ↪  y(2);

ss_model_well3 = theta_well3(1) + theta_well3(2)*u(3) +
↪  theta_well3(3)*uEst(6) ...
        + theta_well3(4)*u(3)^2 + theta_well3(5)*uEst(6)^2 -
        ↪  y(3);

% Modify output
y_m_k = y + eps_k + lam_k*(u-O_vector_old);

J = 20*y_m_k(1) + 10*y_m_k(2) + 30*y_m_k(3);

gas_constraint = u(1) + u(2) + u(3);

nlp = struct('x', [u;y], 'f', -J, 'g',
↪  [ss_model_well1;ss_model_well2;ss_model_well3;gas_constraint]);
solver = nlpsol('solver','ipopt',nlp);

sol = solver('x0', [O_vector_old;yEst], 'lbx', [1;1;1;0;0;0],
↪  'ubx', [5;5;5;inf;inf;inf], 'lbg', [0;0;0;0], 'ubg',
↪  [0;0;0;7.5-h]);
opt = full(sol.x);
u_opt = opt(1:3);

J_opt = full(sol.f);
```

```
        %filter on u
        u_opt = O_vector_old + K*(u_opt-O_vector_old);

        O_vector = u_opt';

        SS = 1;
        Estimation = 0;
        Optimization = 0;
        Result = J_opt;
        Parameter_Estimation =
        ↪  [plantGrad_estimated(1,1),plantGrad_estimated(2,2),plantGrad_estimated(3,3),m
        State_Variables_Estimation =
        ↪  [yEstOld(1),yEstOld(2),yEstOld(3),ymodelssOld(1),ymodelssOld(2),ymodelssOld(3
        State_Variables_Optimization =
        ↪  [lam_k(1,1),lam_k(2,2),lam_k(3,3),eps_k(1),eps_k(2),eps_k(3)];
        Optimized_Air_Injection = [yEst(1),yEst(2),yEst(3)];
    end

else
    % compute new values for the gas flow rate setpoints
    O_vector = uOold'; % dummy

    SS = 0;
    Estimation = 0;
    Optimization = 0;
    Result = 0;
    Parameter_Estimation = [0,0,0,0,0,0];
    State_Variables_Estimation = [0,0,0,0,0,0];
    State_Variables_Optimization = [0,0,0,0,0,0];
    Optimized_Air_Injection = [yEst(1),yEst(2),yEst(3)];
end
```