Master's thesis

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

Vegard Aas

Distributed Feedback-optimizing Control Strategies: an Experimental Validation

Master's thesis in Chemical Engineering and Biotechnology
Supervisor: Sigurd Skogestad
Co-supervisor: Risvan Dirza

June 2023

NTNU
Norwegian University of
Science and Technology

Vegard Aas

Distributed Feedback-optimizing Control Strategies: an Experimental Validation

Master's thesis in Chemical Engineering and Biotechnology
Supervisor: Sigurd Skogestad
Co-supervisor: Risvan Dirza
June 2023

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

Abstract

In industrial settings, processes often consists of multiple subsystems that share a common constraint, such as a shared resource. This paper focuses on the steady-state real-time optimization (RTO) problem in a subsea gas-lifted oil production network comprising multiple wells with constrained access to shared gas-lift. Traditional approaches to solving such problems often involve computationally expensive centralized optimization methods, which are known to be slow. To overcome these challenges, the concept of feedback-optimizing control, as proposed by Morari et al.[21], is explored in this work. By shifting the optimization problem from numerical optimization techniques to the control layer, the need for complex optimization algorithms is eliminated. Instead, simpler control tools such as Proportional-Integral-Derivative (PID) controllers can be employed to achieve the desired optimization objectives. Importantly, this approach can accommodate diverse operation units characterized by different time scales, enhancing the flexibility and adaptability of the optimization method.

In previous work, there has been done a comparison between two recently developed methods of feedback-optimizing controlled which is referred to as primal-based and dual-based distributed feedback RTO[2]. The primal-based method is based on the work introduced by Dirza et al.[9], while the dual-based method is based on the work by Dirza et al.[12]. For the dual-based method the constraints are controlled in a slow time scale by the dual variables. As a result, it is necessary to employ a back-off strategy to ensure constraint satisfaction, which again would lead to profit loss. To reduce the back-off for the dual-based method, Dirza et al.[10] introduced an alternative dual-based approach with a constraint override controller.

In this paper, it is conducted a comparative experimental validation study of the primal-based, dual-based and dual-based with override approaches. This is executed in a lab-scale experimental rig which emulates a subsea gas lifted oil production network. The gas lift is a shared resource, and must therefore be distributed optimally between the wells. Based on the experimental results, it can be concluded that the primal-based method achieves the best performance for the given gas lift constrained scenario

Sammendrag

I industrielle settinger består prosesser ofte av flere undersystemer som deler en felles begrensning, for eksempel en delt ressurs. Denne oppgaven fokuserer på steady-state sanntidsoptimalisering (RTO) problemet i et subsea gassløftet oljeproduksjonsnettverk bestående av flere brønner med begrenset tilgang til delt gassløft. Tradisjonelle tilnærminger for å løse slike problemer involverer ofte beregningsintensive sentraliserte optimeringsmetoder som er kjent for å være trege. For å overkomme disse utfordringene, utforskes konseptet med tilbakekoblingsoptimerende kontroll, foreslått av Morari et al.[21], i dette arbeidet. Ved å flytte optimeringsproblemet fra numeriske optimeringsteknikker til kontrolllaget, elimineres behovet for komplekse optimeringsalgoritmer. I stedet kan enklere kontrollverktøy som Proporsjonal-Integral-Derivativ (PID) regulatorer brukes for å oppnå ønskede optimeringsmål. Viktigst av alt kan denne tilnærmingen tilpasse seg forskjellige driftsenheter med ulike tidsrammer, noe som forbedrer fleksibiliteten og tilpasningsevnen til optimeringsmetoden.

I tidligere arbeid er det gjort en sammenligning mellom to nylig utviklede metoder for tilbakekoblingsoptimerende kontroll, kalt primal-basert og dual-basert distribuert tilbakekoblingsoptimering RTO[2]. Den primal-baserte metoden er basert på arbeidet introdusert av Dirza et al.[9], mens den dual-baserte metoden er basert på arbeidet av Dirza et al.[12]. For den dual-baserte metoden styres begrensningene i en sakte tidsramme av de duale variablene. Derfor er det nødvendig å bruke en "back-off" strategi for å sikre begrensningstilfredshet, noe som igjen kan føre til tap av fortjeneste. For å redusere "back-off" for den dual-baserte metoden introduserte Dirza et al.[10] en alternativ dual-basert tilnærming med en begrensningsoverstyringskontroller.

I denne artikkelen gjennomføres det en eksperimentell valideringsstudie av primal-baserte, dual-baserte og dual-baserte med overstyring strukturene hvor disse blir sammenlignet. Dette blir utført i en lab-skala eksperimentell rigg som etterligner et subsea gassløftet oljeproduksjonsnettverk. Gassløftet er en delt ressurs og må derfor fordeles optimalt mellom brønnene. Ut ifra de eksperimentelle resultatene kan det konkluderes med at den primal-baserte metoden oppnår best ytelse i det gjeldene gassløft begrensede scenarioet.

Preface

I would like to express my gratitude to my supervisor Sigurd Skogestad and co-supervisor Risvan Dirza for their guidance and support throughout this master thesis. I would especially like to thank Risvan for assisting me throughout the entire process, both during as well as outside regular working hours, offering valuable input and assistance not only for my master thesis but also for my specialization project last semester.

Contents

List of Figures

List of Tables

1 Introduction

Industrial processes often involve the presence of multiple subsystems that share a common constraint, such as a shared resource. The optimal distribution of such shared resources among the subsystems becomes a critical objective. A common power plant where steam is delivered to different sub-processes[14] [29] serves as an example of such optimization problems. Challenges like this are commonly addressed within the context of real-time optimization (RTO). The conventional approach to RTO involves solving numerical optimization problems using detailed process models that are continuously updated based on process measurements. However, this traditional method often suffers from slow solutions and becoming computational expensive. Moriari et al.[21] proposed an approach to address this challenge by transferring the optimization problem from numerical optimization to the control layer. This approach, known as feedback-optimizing control, enables the utilization of simple tools like proportional-integral-derivative (PID) controllers. Importantly, this optimization method is able to handle different operation units with varying time scales. This paper considers a steady-state RTO problem in the context of a subsea gas-lifted oil production network which comprises multiple wells and considers a shared gas-lift supply as a constraint[8] [12] [17]. In such scenarios, it is often desirable to decompose the large scale problem into smaller subproblems and solving them locally. This approach offers several advantages over centralized optimization on a large scale. Distributed decision making tools are generally easier to maintain and implement, making them a practical choice for these problems. The decomposition strategies for such large scale problems can be broadly categorized as primal-based and dual-based decomposition methods.[5]

In recent work, Krishnamoorthy et al.[16] and Dirza et al.[12] have proposed a ditributed RTO framework based on dual decomposition, utilizing simple feedback control to ensure convergence to a stationary point of the system wide optimization problem. The suggested method employs Lagrangian relaxation to transform the constrained economic optimization problem into an unconstrained optimization problem. The optimal operation point is achieved through asymptotic convergence by employing constraint control via the Lagrange multipliers (dual variables) within a central constraint controller operating in the slow time scale. Simultaneously, in a faster time scale, the gradient of the Lagrangian is controlled to zero using the physical manipulated variables (primal variables) in a cascade manner, ensuring effective convergence towards the desired operating conditions.

Recently, Dirza et al.[10] proposed a dual-based decomposition framework implemented with override constraint control. The suggested method is largely based on the dual-based method mentioned above. However, the dual-based method could result in dynamic constraint violation. To accommodate this problem, override constraint control is implemented to directly control the active constraint. In the central constraint controller an auxiliary constraint is introduced in order to achieve consistency with the steady-state optimization, which considers the difference between controller output from the override constraint and local gradient controllers. As a result, the method achieves system wide optimal operation without loosing the flexibility of the dual-based method while minimizing the back-off necessary to achieve feasible operation condition.

Dirza et al.[9] recently proposed a distributed feedback-based real-time optimizing framework that uses online primal-based decomposition. In this method, the Lagrange multipliers corresponding to each individual subproblem are estimated and utilized within the central constraint controllers to find the optimal operating points. The utilization of the primal-based decomposition method aim to achieve optimal operation while minimizing dynamic constraint violations. This is achieved by employing a compensator in the central constraint control, which ensures constraint satisfaction.

This paper presents a comparative study and experimental validation of three feedback-optimizing control structures utilizing the decomposition strategies mentioned earlier. The three approaches is implemented in a lab-scale gas lifted oil well rig which consists of multiple wells and constrained access to a shared gas lift supply. Given the restricted access to a shared gas lift supply, it becomes crucial to efficiently allocate the gas lift among the various wells in the system. However, this work do not consider any limitations originating from a topside production facility. Hence, it is assumed that the input shared constraint remains consistently active throughout the analysis.

The work presented in this paper is a continuation of the work done in Aas et al.[2], where the primal-based and dual-based approaches was compared in a MATLAB model of the lab-rig utilized for experiments performed in this work.

The remainder of this paper is organized as follows. Section 2 describes some background theory. Section 3 presents the experimental lab-rig setup. Section 4 present the implemented control structures. Section 5 present and discuss the results from the experimental work before the paper is concluded in section 6.

2 Theory

2.1 Feedback Control

The primary goal of process control is to regulate the behavior of an industrial process. In numerous applications, it is crucial to regulate process variables such as flow rates, levels, pressures, or temperatures. Control objectives are commonly focused on setpoint tracking, disturbance rejection, and process safety. Setpoint tracking requires the process variable to follow a predetermined path over time. This is necessary since the optimal operating conditions of a process may change over time. Disturbance rejection aims to mitigate any disturbances that may affect the process output, ensuring consistency and predictability of the process output. In some industrial processes, there may be potential risks, such as thermal runaway or pressure buildup, that could pose a threat to the environment or personnel. In such cases, control strategies are essential to ensure process safety.

This paper employs feedback control, the structure of which is illustrated in figure 2.1. Feedback control is advantageous because the controller is driven by the error signal and therefore compensates for disturbances automatically. This feature eliminates the need for a detailed mathematical model to achieve good controller performance. However, the error-driven nature of the feedback controller means that it only is able to take corrective action after the error from the disturbance is detected in the output. Consequently, any significant delay between the measured output variable and the manipulated or input variable could significantly impact the controllers performance.[15]

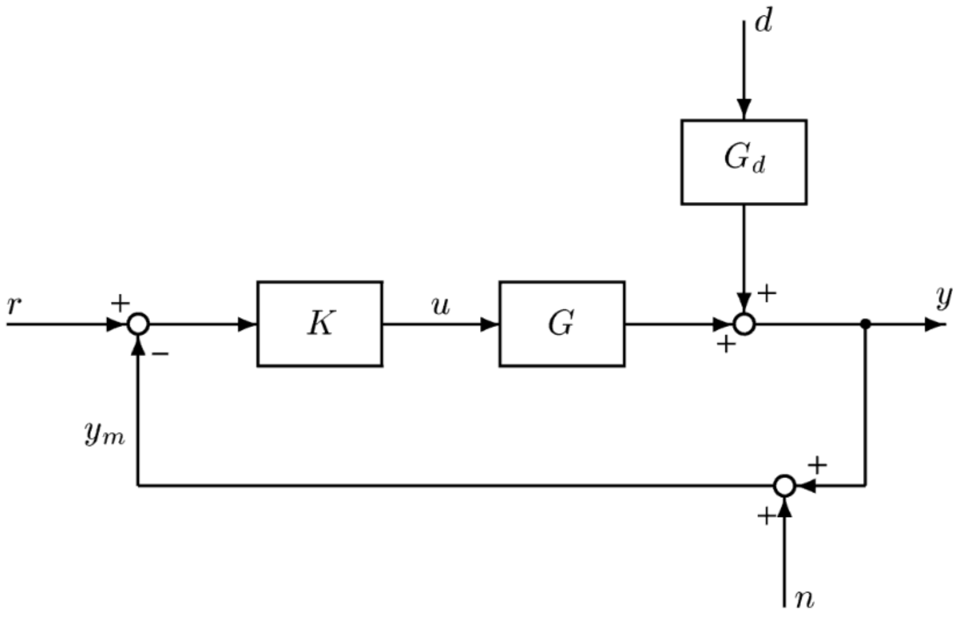

Figure 2.1: Block diagram of simple feedback control system[28]

Figure 2.1 depicts the input to the controller as $r - y_m$, where r represents the controller setpoint and $y_m = y + n$, where y is the actual value and n is the measurement noise. Thus, the plant input is expressed as

$$u = K(r - y - n) \tag{2.1}$$

As previously stated, the goal of control is to adjust the input such that the error remains small despite disturbances d. The control error e is defined as

$$e = y - r \tag{2.2}$$

This objective can be accomplished through the design of a controller K, which can be achieved by implementing a Proportional-Integral-Derivative (PID) controller and SIMC tuning rules.

2.1.1 PID Control

Apart from the on-off controller, the most straightforward feedback controller is the proportional controller. This controller operates on a linear basis, with the control signal being directly proportional to the control error. The output signal of a proportional controller is mathematically given by

$$u(t) = Ke(t) \tag{2.3}$$

where K is the proportional gain and e is the control error given by,

$$e(t) = y_s(t) - y(t) \tag{2.4}$$

where y is the controlled variable measurement and y_s is the setpoint. According to equation 2.4, it is evident that an error must be present for the controller to generate an output. Consequently, the proportional controller always exhibits an offset between the measured variable and the setpoint at steady-state. Therefore, the elimination of this offset requires the implementation of a more sophisticated controller.

The Proportional-Integral-Derivative (PID) controller is extensively employed as the primary control algorithm in the process industry. The PID algorithm, represented in the time domain, is described as follows

$$u(t) = K\left(e(t) + \frac{1}{\tau_I}\int_0^t e(\tau)d\tau + \tau_D\frac{de(t)}{dt}\right) \tag{2.5}$$

where e is the control error, τ_I and τ_D are the integral and derivative time respectively, t is the time and K is the proportional gain.

As observed in equation 2.5, the output of the controller, denoted as u, is determined by the summation of three components. The proportional term, P-term, is directly proportional to the error, the integral term, I-term, is proportional to the integral duration of the error, and the derivative term, D-term, is proportional to the derivative, or rate of change, of the error. Therefore, the tuning parameters for a PID controller include the proportional gain, K, the integral time, τ_I, and the derivative time, τ_D.

The integral action serves the purpose of correcting any steady-state offset from a constant reference signal value. Increasing the integral time, τ_I, reduces the integral action, resulting in slower control response.

The derivative action aims to predict the control action by utilizing the rate of change in the error to make the predictions and prevent significant errors in the future. However, the derivative action is sensitive to measurement noise. Increasing the derivative time, τ_D, amplifies the impact of the derivative action, resulting in faster control response.[15]

2.1.2 Digital Implementation of PID Control

When implementing PID control in computers it is not possible to use the continuous version presented in equation 2.5. In order to implement it in computers the PID controller can be discretized in the following manner. First a "bias" term is considered, which includes the integral action

$$\bar{u}(t) = \bar{u}^k \approx \bar{u}^{k-1} + K\frac{\Delta t}{\tau_I}e^k \tag{2.6}$$

where Δt is the time between measurements of the controlled variable, also called the time period. k denotes the current time t, while $k-1$ is the previous sampling time $t-\Delta t$. In order to avoid windup and get smooth transfer between manual and auto, the bias $\bar{u}^k$ is adjusted so u^k always is equal the actual input. As a result, the digital PID control is implemented as[25]

$$u^k = u^{k-1} + K\left(e^k + \frac{\Delta t}{\tau_I}e^k + \frac{\tau_D}{\Delta t}(e^k - e^{k-1})\right) \tag{2.7}$$

where the derivative action simply utilizes backward Euler method.

$$\frac{de(t)}{dt} = \frac{(e^k - e^{k-1})}{\Delta t} \tag{2.8}$$

2.1.3 SIMC Tuning Rules

In section 2.1.1, it is mentioned that a PID controller utilizes only three tuning parameters. However, determining appropriate tuning parameters can be challenging without a systematic approach. Various methods for tuning PID controllers have been proposed in literature, including Ziegler-Nichols[31] and Cohen-Coon[6] methods for example. In this paper, the Skogestad Internal Model Control (SIMC) method. The SIMC method was developed by professor Sigurd Skogestad with the objective of creating simple and easily memorizable rules while ensuring good controller performance. This method provides guidelines for tuning the PID controller based on the internal model control principle.

In the SIMC method, the tuning parameters for the controller are determined based on a first- or second-order plus delay model approximation of the process. The model information used for tuning includes the following key parameters:

- Plant gain, k
- Dominant time constant, τ_1

- Effective time delay, θ
- Second-order time constant, τ_2 (only used for dominant second-order process ($\tau_2 > \theta$))

The required information can be obtained through various means, including estimation from an open-loop step response, a closed-loop response using a P-controller, or utilizing a detailed model with an approximation of the effective dead time using the half rule. The process of finding these parameters from the open-loop step response is presented in figure 2.2

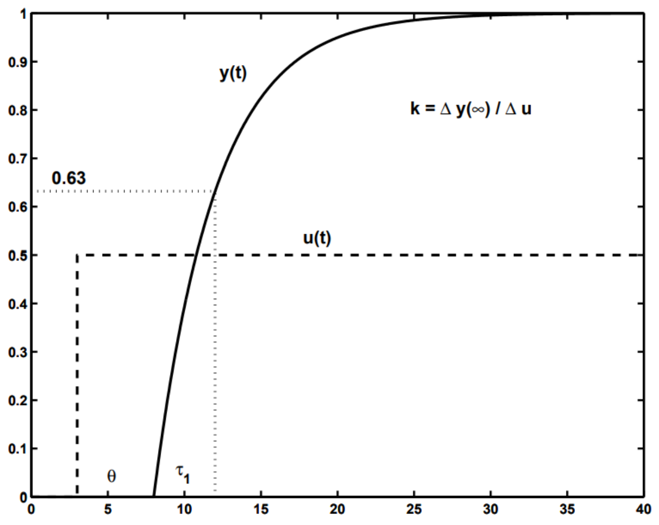

Figure 2.2: Step response of first-order system with delay, $g(s) = ke^{-\theta s}/(\tau_1 s + 1)$ [27]

By employing the open-loop step response method, the model can be approximated as a transfer function in the form of a first-order model with an additional time delay as follows

$$g(s) = ke^{-\theta s}/(\tau_1 s + 1) \tag{2.9}$$

The SIMC tuning rules are derived based on the principles of Internal Model Control, which involves specifying the desired first-order closed-loop response with a time constant of τ_c. Applying the SIMC tuning rules yields the following PI controller settings

$$K = \frac{1}{k}\frac{\tau_I}{\tau_c + \theta} \tag{2.10}$$

$$\tau_I = min\{\tau_1, 4(\tau_c + \theta)\} \tag{2.11}$$

where τ_1 represents the first-order time constant, while k represents the steady-state plant gain, as illustrated in figure 2.2. Consequently, only one tuning parameter, τ_c remains, which represents the desired closed-loop time constant. [27]

2.1.4 Anti-windup

When the output provided from a PID-controller deviate from the plant input, the integral action on the controller will accumulate error. This is the case when the controller output is manipulated before being fed into the plant. The manipulation could be a result of using selectors or saturation of the equipment that is controlled, for instance a valve that reaches its physical limits of fully closed or fully open.

Methods implemented to avoid such windup of error are denoted as anti-windup. Back-calculation is one method of implementing anti-windup. This method is presented in figure 2.3, where the τ_b is the back-calculation coefficient. This is usually selected as $\tau_b = 1/\tau_I$.[19] In the figure, the saturation block is causing the error windup. The saturation block could also be exchanged with a selector, the back-calculating principle will still be the identical, where the difference between the controller output u and the plant input u_{plant} is used to avoid windup of error in the integral controller.

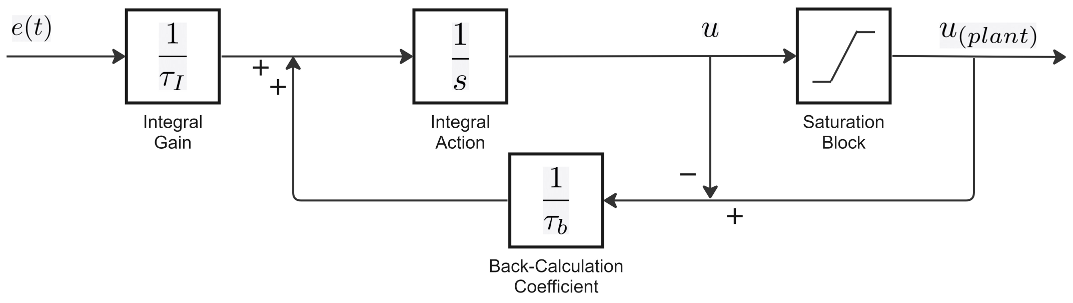

Figure 2.3: Illustration of the back-calculating anti-windup concept.

2.1.5 Cascade Control

The fundamental principle behind cascade control is the existence of an outer control loop that provides a setpoint for an inner control loop. A cascade loop consisting of only two controllers is commonly known as a single cascade. However, it is possible to incorporate additional controller loops on top of the outer control loop. The cascade control structure can be in series or parallel configuration.

The single series cascade control structure is illustrated in figure 2.4. The outer control loop, referred to as master controller, provides a setpoint y_2^{sp} for the inner control loop by controlling the output from process part g_1 in controller c_1 to the desired setpoint y_1^{sp}. The inner control loop, referred to as slave controller, consists of controller c_2 and the process part g_2. c_2 controls y_2 to y_2^{sp}. The output from the inner control loop y_2 is provided to the process part g_1. By employing the cascade principle, the system operating range is extended by using multiple inputs to control y_1.

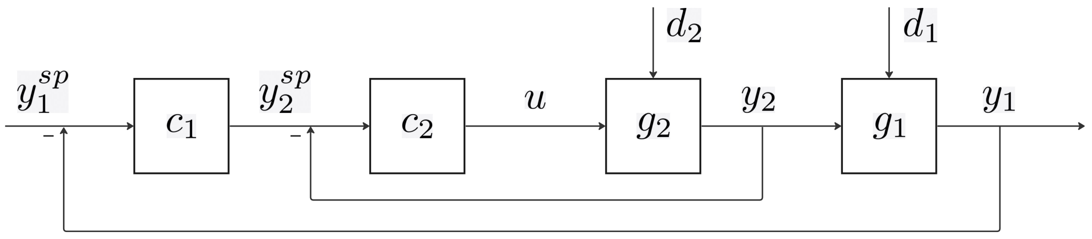

Figure 2.4: Block diagram of series cascade structure.

The parallel cascade principle is illustrated in figure 2.5. The parallel structure differs from the series structure by only having one process part or plant which provides both controlled variables y_1 and y_2. In this structure there is also a fast slave controller c_2 and a slower master controller c_1. In this configuration there is no individual disturbances, as a result any disturbances in the plant will affect both y_1 and y_2. In the parallel structure, one important aspect is that y_2 must be closely related to y_1[24]. This cascade structure is relevant for the control methods implemented later in this paper.

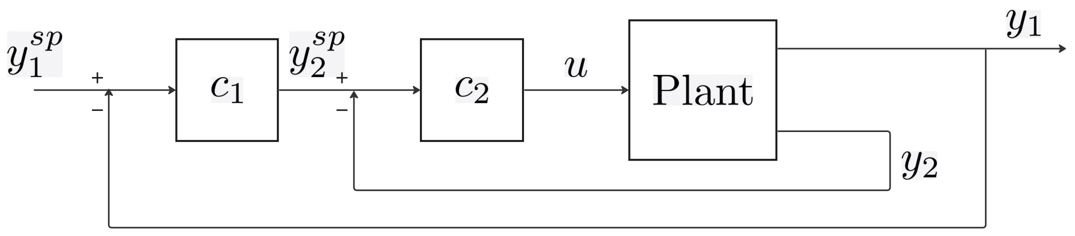

Figure 2.5: Block diagram of parallel cascade structure.

A challenge with the cascade structure is avoiding interactions between the master and slave controller, which could lead to instabilities and oscillations in the system. To avoid these interactions, it is important to consider time scale separation between the two control layers. Therefore, the master controller should be slower than the slave controller. A rule of thumb is time scale separation of at least 5 between control layers.[28] To ensure minimal interactions between the cascade layers, the time constant t_c should increase between every control loop increment.

2.1.6 Selector

Selectors serve as a logical component frequently employed in the implementation of advanced control structures. They are utilized in order to switch between controlled variables in a single-input system controlling the same process, this concept is called CV-CV switching. In this arrangement, a single manipulated variable controls multiple controlled variables. To achieve this, one controller is assigned to each controlled variable within the system. The selectors, typically designed as maximum, minimum, or mid-range selectors, ensure that the single manipulated variable only controls one controlled variable at any given time. In figure 2.6 a system with two controllers c_1 and c_2 controlling the same process part g_1 from outputs y_1 and y_2 is illustrated. Based on a logical statement, the selector block determines whether to choose the minimum

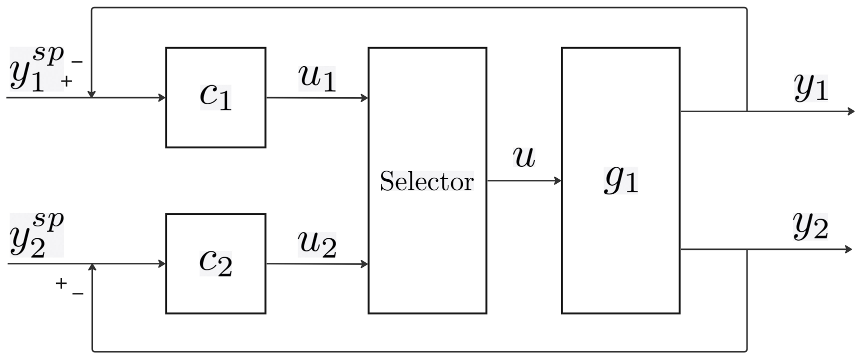

Figure 2.6: Block diagram of system with two controllers and a selector choosing input to the process part.

or maximum from the available manipulated variables u_1 or u_2 and provides u to the process part.[18] [19]

2.2 Process Control Hierarchy

The process control hierarchy of an industrial process can be presented as different decision levels as shown in figure 2.7. The lowest level of the hierarchy is the process layer, which is the process or plant itself that is controlled. The next level is the regulatory control layer, which is responsible for controlling individual process variables, such as temperature, pressure, and flow rate. The process layer is typically composed of simple control loops, each responsible for maintaining the desired setpoint of a particular process variable.

Above the regulatory layer is the supervisory layer, which manages the interaction between multiple regulatory systems. The supervisory layer is responsible for coordinating the operation of several regulatory loops to achieve a global process objective. The supervisory layer also uses more advanced control strategies, such as model predictive control (MPC) and adaptive control, to optimize process performance.

At the next level is the local optimization layer, which focuses on maximizing profits by adjusting process variables and setpoints based on economic criteria. The local optimization layer uses advanced optimization algorithms to find the optimal operation setpoints for the process variables, taking into account economic factors such as production costs, market demand, and energy consumption. The local optimization layer also considers process constraints such as equipment limitations, environmental regulations, and safety requirements. This layer operates in a relatively fast time scale, as a result, this layer is also called the real-time optimization layer.

At the next level is the system wide optimization, which makes sure the entire system is working optimally by considering inter-dependencies across multiple subsystems. the system wide optimization layer solves optimization problems that involve global objective, such as maximizing production or minimizing consumption.

In the highest level, the scheduling layer is found. This layer focuses on setting production goals in order to meet supply and logistical constraints. This layer is not relevant for the research in this paper, as the main focus lies within the local optimization and

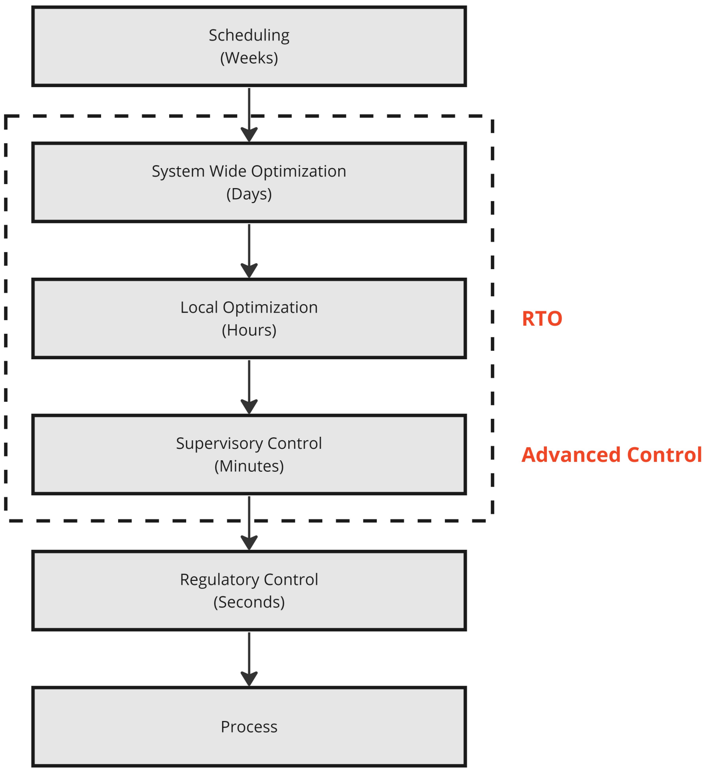

Figure 2.7: The levels of the process control hierarchy with respective time scales. The layers of interest in this research is highlighted in the dotted box.

supervisory control layers.

The process control hierarchy provides a systematic approach to control and optimize industrial processes across multiple levels, allowing for greater control, efficiency, and profitability. It also enables the integration of different control strategies and technologies, providing a comprehensive solution for managing industrial processes.[13]

2.3 Real-time Optimization

Traditionally the real-time optimization (RTO) framework solves a steady-state numerical optimization problem in order to compute the optimal steady-state setpoints. The numerical optimization problem consists of three components.

- Extensive nonlinear steady-state process models.
- Environmental, equipment and process constraints.
- Economic objective function that consists of the value of the products, cost of raw material, operational cost etc.

Conventional RTO implementations utilize steady-state nonlinear models. These models are updated by using data collected during time periods corresponding to steady-state process operation. The objective of this update is to match the plant measurements with the model predictions. When the model is updated, it provides a

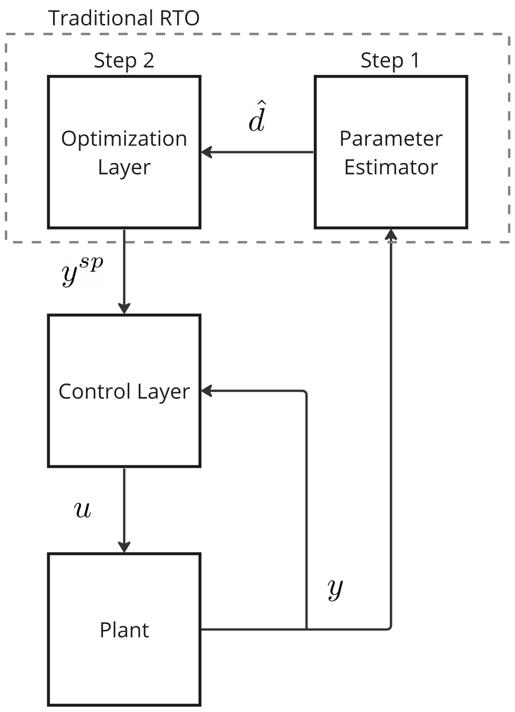

Figure 2.8: Conventional RTO structure

re-optimization for the process with new setpoints. The numerical solver-based RTO framework follows a two-step approach, which involves data reconciliation in step one, here the nonlinear model is updated by using steady-state measurements. In step two, new optimal setpoints are computed by performing numerical optimization utilizing the updated model[23]. The information flow of the conventional RTO implementation is presented in figure 2.8

Most of the RTO software packages that are available commercially employ a repeated identification and optimization method that is based on steady-state models. This method is a suitable approach for continuous processes, as these processes typically operate at a steady-state to achieve optimal economic operation. Processes with frequent grade changes as cyclic operations and batch processes are exceptions to this rule. Therefore, in most continuous processes the primary objective is to identify the optimal steady-state operating point that is economically efficient with the given operating conditions.

In the RTO layer, the decision variables are typically the setpoints for the controlled variables. These setpoints are subsequently transmitted to the setpoint tracking control layer situated below. The control layer regulates the manipulated variables to maintain the process measurements at the optimal setpoints that is determined by the RTO layer above.

2.4 Decomposition

In the case of process optimization, decomposition refers to breaking down complex optimization problems into smaller, more manageable subsystems that can be solved individually. In other words, decomposition simplify the problem solving process by reducing the complexity of the optimization problem with help of breaking it down into smaller problems that are easier to solve.

There are two types of decomposition which is vertical and horizontal decomposition. In vertical decomposition the system is split up in subsystems set up in a hierarchical structure. Each subsystem is dependent on the solution from the higher levels in the hierarchy. In horizontal decomposition the system is split up in smaller, independent subsystems at the same level of the hierarchy[26]. Horizontal decomposition promotes modularity and allows for distributed processing of the subsystems. Each subsystem can be solved independently, and the solutions can be combined or coordinated to obtain the overall solution[4]. In this work, decomposition is referring to horizontal decomposition.

Distributed control refers to control of a system where decision-making and control actions are distributed across multiple controllers within one system. However, distributed systems refer to systems that consists of multiple subsystems that work together to achieve a common goal. It is possible to have distributed control without the system being distributed. However, if the system is distributed there is automatically distributed control. As a result, the distributed feedback-optimizing control approaches discussed in this work can be categorized as distributed systems with distributed control.

2.5 Distributed Feedback-optimizing Control

Consider an integrated steady-state optimization problem of N separate subproblems, also called subsystems.

$$\min_{\mathbf{u}} \quad \mathbf{J}(\mathbf{u}, \mathbf{d}) = \sum_{i=1}^{N} J_i(u_i, d_i) \tag{2.12a}$$

$$s.t. \quad \mathbf{g}(\mathbf{u}, \mathbf{d}) = \sum_{i=1}^{N} \mathbf{g}_i(u_i, d_i) - \bar{\mathbf{g}} \leq 0 \tag{2.12b}$$

where $\mathbf{d} \in \mathbb{R}^{n_\mathbf{d}}$ are the set of disturbances, $\mathbf{u} \in \mathbb{R}^{n_\mathbf{u}}$ denotes the set of manipulated variables, $\mathbf{J} : \mathbb{R}^{n_\mathbf{u}} \times \mathbb{R}^{n_\mathbf{d}} \to \mathbb{R}$ denotes the cost function, $\mathbf{g} : \mathbb{R}^{n_\mathbf{u}} \times \mathbb{R}^{n_\mathbf{d}} \to \mathbb{R}^{n_\mathbf{g}}$ is the coupling constraints and the limit of the constraints is denoted by $\bar{\mathbf{g}} \in \mathbb{R}^{n_\mathbf{g}}$.

$\mathbf{d}$ and $\mathbf{u}$ is defined as $\mathbf{d} = \left[d_1, ..., d_N\right]^\top$ and $\mathbf{u} = \left[u_1, ..., u_N\right]^\top$, $d_i \in \mathbb{R}^{n_{d_i}}$ is the disturbances and n_{d_i} denotes the number of disturbances in subsystem i, and $u_i \in \mathbb{R}^{n_{u_i}}$ is the decision variables and n_{u_i} denotes the number of decision variables in subsystem i.

The local objective function and the (in-)equality constraints function in subsystem i is defined by $\mathbf{J}_i : \mathbb{R}^{n_{u_i}} \times \mathbb{R}^{n_{d_i}} \to \mathbb{R}$ and $g_i : \mathbb{R}^{n_{u_i}} \times \mathbb{R}^{n_{d_i}} \to \mathbb{R}^{n_g}$ respectively. n_g defines the number of coupling constraints.

In this paper, the aim is to convert a large-scale optimization problem 2.12 into smaller feedback control problems that can be solved with use simple tools as PID controllers, selector, or a configuration of them. In the following sections three structures of distributed feedback-optimizing control are described. These schemes aim to solve optimization problem 2.12 in a distributed manner using simple feedback control.

2.5.1 Distributed Feedback-Optimizing Control using Online Primal Decomposition

The primal-based distributed feedback optimizing control method is based on the method proposed in Dirza et al.[9]. In order to implement the primal-based method following assumption must be satisfied: *Each local system (subproblem) has enough decision variables/input, u_i, to control the active coupling constraints,* $\mathbf{g}$.

$$n_{u_i} \geq n_{\mathbf{g}}, \quad i = 1, ..., N \tag{2.13}$$

By introducing a virtual subsystem, called subsystem 0, and defining constraint 2.12b as a linear constraint,

$$\mathbf{g}(\mathbf{u}, \mathbf{d}) = g_0 + \sum_{i=1}^{N} g_i(u_i, d_i) - g^{max} \tag{2.14}$$

where g^{max} denotes the limit of the constraint, optimization problem 2.12 can be reformulated as an equality constraint problem:

$$\min_{\mathbf{u}} \quad J_0 + \sum_{i=1}^{N} J_i(u_i, d_i) \tag{2.15a}$$

$$s.t \quad \mathbf{g}_0 + \sum_{i=1}^{N} \mathbf{g}_i(u_i, d_i) - g^{max} = 0 \tag{2.15b}$$

J_0 will not influence any optimal solution because it is a constant, $J_0 = 0$, $\mathbf{g}_0$ functions as a storage for any unused values, slack variable. As a result, the inequality constraint 2.12b is transformed to an equality constraint. By introducing an initial value of local constraint, g_i^{sp}, which is defined by $g^{sp} = \sum_{i=1}^{N} g_i^{sp}$, for the coupling constraint variables the active coupling constraint satisfaction can be taken care of by a central problem. As a result constraint 2.15b can be rewritten as

$$g_i(u_i, d_i) - g_i^{sp} = 0, \quad i = 0, ..., N \tag{2.16a}$$

$$\sum_{i=0}^{N} g_i^{sp} = g^{max} \tag{2.16b}$$

while equation 2.16b is satisfied, the primal feasibility is guaranteed for the coupling constraint 2.12b.

By relaxing the local constraint 2.16a, problem 2.15 can be rewritten as a Lagrange function. This Lagrange function can be decomposed into subproblems, enabling the optimization problem to be solved for each individual subsystem i.

$$\mathcal{P}_i(g_i^{sp}) := \min_{u_i} \quad \mathcal{L}_i(u_i, g_i^{sp}, \lambda_i) \tag{2.17}$$

where

$$\mathcal{L}_i(u_i, g_i^{sp}, \lambda_i) = J_i + \lambda_i g_i(u_i, d_i). \tag{2.18}$$

λ_i is the local Lagrange multiplier, which is linked to the local constraint 2.16a. In steady-state optimal conditions the local constraint will converge to the same value.[9]

In each subproblem, local Lagrange multipliers are estimated and utilized by the central constraint controllers. In these controllers the setpoints are updated iteratively, where the aim is to provide a set of setpoints that satisfy the primal feasibility 2.16b.

$$\min_{g_0^{sp},\dots,g_N^{sp}} \quad \sum_{i=0}^{N} \mathcal{P}_i(g_i^{sp}) \tag{2.19a}$$

$$\text{s.t.} \quad \sum_{i=0}^{N} g_i^{sp} = g^{\max} \tag{2.19b}$$

$\mathcal{P}_i(g_i^{sp})$ is defined in 2.17, the constraint 2.19b is inherited from 2.16b.

The calculation for one of the local setpoints is performed in the following manner

$$g_N^{sp,k+1} = g^{\max} - (g_0^{sp,k+1} + ... + g_{N-1}^{sp,k+1}) \tag{2.20}$$

This is to ensure satisfaction of constraint 2.16b, in other words, ensuring primal feasibility. This subsystem, subsystem N as displayed in 2.20, is referred to as the compensator subsystem.

To determine the local setpoint g_i^{sp} at time step $k+1$ for the remaining subproblems, $j = \{0, ..., N-1\}$, the steepest descent direction of the central problem 2.19a can be utilized. This is determined by the subgradient,

$$\nabla_{g_j^{sp}} \left(\sum_{i=0}^{N-1} \mathcal{P}_i(g_i^{sp,k}) \right) = -\lambda_j^k + \lambda_N^k \tag{2.21}$$

The updated local setpoint at time step $k+1$ is found by,

$$g_i^{sp,k+1} = g_i^{sp,k} + K_{I,i} \nabla_{g_i^{sp}} \left(\sum_{i=0}^{N-1} \mathcal{P}_i(g_i^{sp,k}) \right) \tag{2.22}$$

Integrating controllers with integral gain $K_{I,i} = \frac{1}{K_i(\tau_{c,i})}$, where the step response gain is defined by K_i and the closed-loop time constant is given by $\tau_{c,i}$, may be utilized here.

Given that a slack variable, g_0^{sp}, has been introduced to store unutilized values, and the storage never is negative in a physically perspective, it is crucial to employ a max selector as follows.

$$g_0^{sp,k+1} = \max\left[0, \quad g_0^{sp,k} + K_{I,0}\nabla_{g_0^{sp}}\left(\sum_{i=0}^{N-1}\mathcal{P}_i(g_i^{sp,k})\right)\right] \tag{2.23}$$

The implementation of the compensator and virtual subsystem strategies ensures that the setpoints provided by these controllers guarantee the primal feasibility.

To determine the local setpoints, estimations of the local Lagrange multipliers in 2.21 is required. Normally, this is available when solving the numerical optimization problem in the traditional RTO framework. However, in this paper, feedback control is used, therefore are these not directly available. Hence, the multipliers needs to be estimated. For all subsystems, the stationary point is reached when

$$\nabla_{u_i}\mathcal{L}_i(u_i, g_i^{sp}, \lambda_i) = 0 \tag{2.24}$$

according to the Karush-Kuhn-Tucker (KKT) conditions. All multipliers, λ_i, will converge to the same optimal value. Rearranging 2.24 the local Lagrange multiplier can be computed in following manner,

$$\lambda_i = -\nabla_{u_i}J_i(u_i, d_i)(\nabla_{u_i}g_i(u_i, d_i))^{-1} \tag{2.25}$$

As mentioned earlier, the amount of local decision variables must be more than or equal to the amount of constraints, the solution must also be unique.

Note that in order to counterbalance any change in the different subsystems, it is assumed that each subsystem communicates its local Lagrange multiplier, λ_i^k, to the compensator subsystem, and receives the local Lagrange multiplier, λ_N^k from the compensator subsystem.

2.5.2 Distributed Feedback-Optimizing Control using Dual Decomposition

The dual-based distributed feedback optimizing control method is based on the method proposed in Krishnamoorthy[16] and Dirza et al.[12] [8]. In order to implement the dual-based method following assumption must be satisfied: *The entire system has enough decision variables/input,* $\mathbf{u}$*, to control the active coupling constraints,* $\mathbf{g}$.

$$n_{\mathbf{u}} \geq n_{\mathbf{g}} \tag{2.26}$$

Introducing the Lagrangian function for optimization problem 2.12,

$$\mathcal{L}(\mathbf{u}, \mathbf{d}, \lambda) = J(\mathbf{u}, \mathbf{d}) + \lambda^\top \mathbf{g}(\mathbf{u}, \mathbf{d}) \tag{2.27}$$

where the constraint and Lagrange multiplier are defined as $\mathbf{g} = \begin{bmatrix} g_1 & ... & g_{n_g} \end{bmatrix}^\top$ and $\lambda = \begin{bmatrix} \lambda_1 & ... & \lambda_{n_g} \end{bmatrix}^\top$. Then the necessary condition for optimality, KKT conditions, for optimization problem 2.12 can be written as

$$\nabla_{\mathbf{u}} \mathcal{L}(\mathbf{u}, \mathbf{d}, \lambda) = 0 \tag{2.28a}$$

$$\mathbf{g}(\mathbf{u}, \mathbf{d}) \leq 0 \tag{2.28b}$$

$$\lambda \geq 0 \tag{2.28c}$$

$$\lambda_i g_i(\mathbf{u}, \mathbf{d}) = 0, \quad \forall \quad i \in [1, ..., n_g] \tag{2.28d}$$

where $\mathbf{u}$ and λ are the unknown variables. To solve the equations in 2.28, dual ascent can be employed.[4] 2.28a is solved for $\mathbf{u}$ with a fixed value for λ. The remaining equations are satisfied by iteratively changing λ in an outer loop. When the constraints are active, which is the case in this paper, it is crucial to maintain $\mathbf{g} = 0$, this implies a non-zero λ. Since the constraint value $\mathbf{g}$ usually is measured, feedback control can be utilized to solve the equations. A benefit of using feedback control for solving equation 2.28a with respect to λ is that the constraints $\mathbf{g}$ in 2.28b and 2.28d does not need to rely on a model, eliminating the need for any model update. By implementing this approach, it is possible to achieve stationary steady-state conditions through control of

$$c(\lambda) := \nabla_{\mathbf{u}} \mathcal{L}(\mathbf{u}, \mathbf{d}, \lambda) = \nabla_{\mathbf{u}} J(\mathbf{u}, \mathbf{d}) + \lambda^\top \nabla_{\mathbf{u}} \mathbf{g}(\mathbf{u}, \mathbf{d}) \tag{2.29}$$

to a stationary setpoint $c^{sp} = 0$ for any λ. The function of achieving stationary steady-state conditions is carried out by the gradient controllers (fast time-scale), which use the Lagrange multipliers, λ_i, as manipulated variables in an outer loop to drive the corresponding constraints $g_i(\mathbf{u}, \mathbf{d})$ to 0 for $i \in [1, ..., n_g]$. λ_i and $g_i(\mathbf{u}, \mathbf{d})$ is paired together and a max selector is employed to ensure that the conditions $\lambda \geq 0$ 2.28c and the corresponding slackness condition 2.28d are met. The centralized controllers (slow time-scale) are responsible for satisfying these conditions. Together, the centralized constraint and gradient controllers fulfill the necessary optimal conditions 2.28 at steady-state.

The goal is to decompose the optimization problem and solve it in a distributed fashion. By defining constraint 2.12b as a linear constraint

$$\mathbf{g}(\mathbf{u}, \mathbf{d}) = \sum_{i=1}^{N} g_i(u_i, d_i) - g^{\max} \tag{2.30}$$

where g^{max} denotes the limit of the constraint. The structure of 2.12a remain unchanged, as the cost is assumed to be additively separable. As a result, the integrated optimization problem 2.12 can be rewritten as the separable problem below by introducing 2.30

$$\min_{\mathbf{u}} \quad \sum_{i=1}^{N} J_i(u_i, d_i) \tag{2.31a}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} g_i(u_i, d_i) - g^{max} \leq 0 \tag{2.31b}$$

Then the controlled variable becomes

$$c(\lambda) := \sum_{i=1}^{N} \nabla_{u_i} J_i(u_i, d_i) + \lambda^\top \sum_{i=1}^{N} \nabla_{u_i} g_i(u_i, d_i) \tag{2.32}$$

This expression is additively separable, thus, 2.32 can easily be decomposed and each subsystem control

$$c_i(\lambda) := \nabla_{u_i} J_i(u_i, d_i) + \lambda^\top \nabla_{u_i} g_i(u_i, d_i) \tag{2.33}$$

to a setpoint of $c_i^{sp} = 0$ by manipulation of u_i. By only considering integral action the controller action is given as

$$\Delta u_i = K_{I,i} c_i(\lambda) \tag{2.34}$$

where $K_{I,i}$ is the integral gain. For the next timestep, $k+1$, u_i is found by

$$u_i^{k+1} = u_i^k + K_{I,i} c_i^k(\lambda) \tag{2.35}$$

The Lagrange multiplier λ is found in the central controller in order to control the constraints, $\mathbf{g}(\mathbf{u}, \mathbf{d})$ to the setpoint 0,

$$\Delta\lambda = \alpha \left(\sum_{i=1}^{N} g_i(u_i, d_i) - g^{\max} \right) \tag{2.36}$$

where α is the integral gain. A max selector can be implemented with the controller action in order to satisfy 2.28c. The Lagrange multiplier at time $k+1$ is then given by

$$\lambda^{k+1} = \lambda^k + \alpha \left(\sum_{i=1}^{N} g_i^k(u_i^k, d_i^k) - g^{\max} \right) \tag{2.37a}$$

$$\lambda^{k+1} = \max(0, \lambda^{k+1}) \tag{2.37b}$$

$c_i(\lambda)$ is controlled in the fast time-scale for each subsystem, while the Lagrange multiplier λ is controlled in the slow time-scale in the outer layer to regulate the coupling constraint $\mathbf{g}(\mathbf{u}, \mathbf{d})$. By assuming that the stationary point represents the local minimum, all local subsystems achieve optimal operation for a specific Lagrange multiplier λ. Consequently, as the central constraint controller 2.37 updates λ, it enables optimal performance of the overall optimization problem 2.12.[8] [12]

2.5.3 Distributed Feedback-optimizing Control using Dual Decomposition with Override

The dual-based distributed feedback optimizing control with override method is based on the method proposed in Dirza et al.[10]. In order to implement the dual-based with override method following assumption must be satisfied: *The entire system has enough decision variables/input,* $\mathbf{u}$*, to control the active coupling constraints,* $\mathbf{g}$.

$$n_{\mathbf{u}} \geq n_{\mathbf{g}} \tag{2.38}$$

This method shares most of it structure with the dual-based method in section 2.5.2. The structure aim to solve problem 2.31, where each subsystem control

$$c_i(\lambda) := \nabla_{u_i} J_i(u_i, d_i) + \lambda^\top \nabla_{u_i} g_i(u_i, d_i) \tag{2.39}$$

to a setpoint of $c_i^{sp} = 0$, by manipulation of u_i. By considering integral action the controller action is the same as in 2.34.

For the dual-based with override method a override constraint controller is implemented in one subsystem where the aim is to control 2.31b to 0 by manipulation of u_i^{dir} as follows.

$$u_i^{dir,k+1} = u_i^{dir,k} + K_{I,i}^{dir}(\sum_{i=1}^{N} g_i^k(u_i^k, d_i^k) - g^{max}) \tag{2.40}$$

where $K_{I,i}^{dir}$ is the integral gain if integral action is considered. The implemented variable u_i, for the subsystem with override, must be selected between u_i^{dir} and u_i^{ind}, where u_i^{ind} is the output from the gradient controller in the given subsystem. If a small value is favorable for constraint satisfaction, then

$$u_i = \min(u_i^{dir}, u_i^{ind}) \tag{2.41}$$

Otherwise a max selector should be utilized to select u_i.

In order to insure consistency between the override constraint controller and the gradient controller, the constraint $\mathbf{g}(\mathbf{u}, \mathbf{d}) \leq 0$ is replaced by the constraint $u_i^{ind} - u_i^{dir} \leq 0$ in the central constraint controller for the outer control layer. As a result, the central constraint controller is responsible for ensuring that u_i^{ind} and u_i^{dir} are equal at steady state. Consequently, the Lagrange multiplier can by manipulated with use of I-controller as follows

$$\lambda^{k+1} = \lambda^k + \alpha(u_i^{ind} - u_i^{dir}) \tag{2.42}$$

where α is the integral gain. A max selector can be implemented with the Lagrange multiplier controller in order to satisfy 2.28c, giving

$$\lambda^{k+1} = \max(0, \lambda^{k+1}) \tag{2.43}$$

3 Experimental Setup

In subsea production systems, hydrocarbon reservoirs trapped in underground reservoirs are accessed through wells situated on the seafloor. The extracted oil and gas is then transferred to a topside processing facility via pipelines along the seafloor, where a riser pipeline brings it to the surface. In cases where the reservoir pressure is low, due to natural or depletion-related reasons, artificial lift techniques may be necessary to counteract the pressure losses and raise the hydrocarbons to the surface. A common way to overcome this problem is with use of gaslift, in which compressed gases are injected into the well tubing to reduce the density of the fluid mixture and, in turn, the hydrostatic pressure losses. However, excessive gas injection elevates the frictional pressure drop in the well tubing, which has a counteractive effect[1]. As a result, each well has a local optimum corresponding to the gaslift injection rate. Optimal allocation of the total available lift gas among the wells is crucial for maximizing the overall production from the production network since it is often a limited resource.

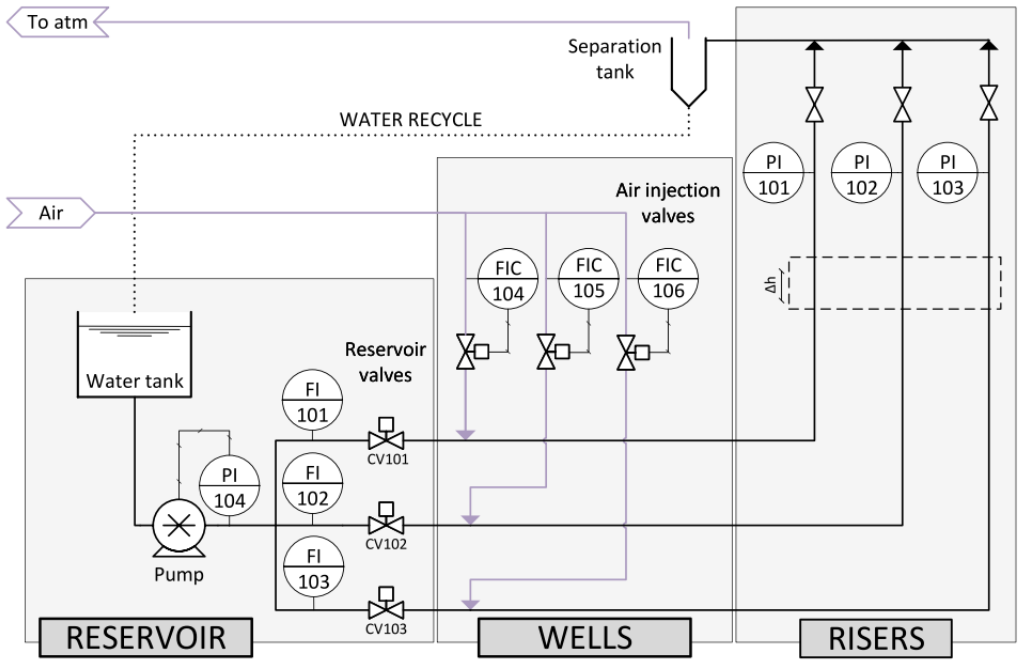

Figure 3.1: Experimental schematic, adapted from Matias et al.[20]. The system measurements y_p are the liquid flowrates (FI101, FI102 and FI103), the gas flowrates (FIC104, FIC105 and FIC106), the well top pressures (PI101, PI102 and PI103) and the pump outlet pressure (PI104). The gas flowrates, $u = [Q_{gl,1} \quad Q_{gl,2} \quad Q_{gl,3}]^T$ are controlled by three PI controllers to the calculated setpoints, $u^{sp} = [Q^{sp}_{gl,1} \quad Q^{sp}_{gl,2} \quad Q^{sp}_{gl,3}]^T$. The reservoir valve openings (CV101, CV102 and CV103) are functioning as the system disturbance. In order to represent different reservoir behaviors they are changing during the experiments. The pump outlet pressure is kept constant during the experiments by a PI controller.

3.1 Experimental Rig Emulating a Subsea Production System

For the work in this paper we use a lab-scale experimental rig that utilizes water and air instead of oil and gas as working fluids to emulate the subsea gas-lifted oil production system. The gas lift phenomenon remains unaffected by the choice of working fluids, which one can observe in the lab rig. Therefore, the rig is suitable for investigating production optimization methods that focus on the gas lift effect as the phenomenon of interest. The system, comprising of a reservoir, well, and riser section is shown in a simplified flowsheet depicted in figure 3.1.

The reservoir section contains a stainless steel tank, a centrifugal pump, and the three control valves (CV101, CV102 and CV103). These valves are used to portray disturbances from the reservoir, such as reservoir deplition, or emulate pressure oscillations. The reservoir can only produce liquid outflow rates ranging from 2 L/min to 15 L/min in this lab rig. Upstream of the reservoir valves the flow meters (FI101, FI102 and FI103) are located to measure outflow rates. The outlet pressure of the pump (PI104) is kept constant at 0.3 barg in this experiment. This is done by using a PI controller that adjusts the pump rotation.

The wells comprise of three flexible hoses arranged in parallel, each hose having an inner diameter of 2 cm and a length of 1.5 m. Pressurized air, at around 0.5 barg, is injected via three air flow controllers (FIC104, FIC105 and FIC106) approximately 10 cm after the reservoir valves. The injection rates of the air flow controllers are within a range of 1 sL/min to 5 sL/min.

The risers are three vertical pipelines arranged orthogonally to the well section, each riser having an inner diameter of 2 cm and a height of 2.2 m. At the top of the risers we measure the pressure using pressure sensors (PI101, PI102 and PI103). Downstream of these pressure sensors, three manual valves are kept open during the experiments. The air, used as gas lift, is vented out to the atmosphere, while the liquid is circulated back to the reservoir water tank.[20]

3.2 Optimization Problem Setup

The objective of the optimization problem in this paper is to maximize the profit from the network liquid throughput in the experimental lab-rig given a limited amount of gaslift injection available. Considering problem 2.12, the economic objection function can be written as

$$\begin{aligned}\mathbf{J}(\mathbf{u}, \mathbf{d}) :&= \sum_{i=1}^{3} f_i(u_i, d_i) \\ &= -20Q_{l,1}(u_1, d_1) - 25Q_{l,2}(u_2, d_2) - 30Q_{l,3}(u_3, d_3)\end{aligned} \tag{3.1}$$

where the produced liquid flowrates of well 1, 2 and 3 are depicted by $Q_{l,1}$, $Q_{l,2}$ and $Q_{l,3}$ respectively. In this work the values of the different hydrocarbon flows are assumed to vary as shown in equation 3.1 in order to illustrate how the individual subsystems are affected by the different values. The input vector is defined as

$$\mathbf{u} = [Q_{gl,1} \quad Q_{gl,2} \quad Q_{gl,3}]^T$$

where the injected gaslift flow rates of well 1, 2 and 3 are depicted by $Q_{gl,1}$, $Q_{gl,2}$ and $Q_{gl,3}$ respectively. These flowrates are the decision variables in context of the optimization. However, for the plant, these flowrates correspond to the setpoints that needs to be tracked. The experimental lab rig has three flow indicator and controllers (FIC104, FIC105 and FIC106) which regulate the air injection to their setpoints by manipulation of the valve openings, as shown in figure 3.1. As a result of this, we indicate the decision variables in the optimization problem as

$$\mathbf{u}^{sp} = [Q_{gl,1}^{sp} \quad Q_{gl,2}^{sp} \quad Q_{gl,3}^{sp}]^T$$

Considering the valve opening of the FICs as the decision variables could also be a possibility. Although, this alternative could lead to some practical issues due to the hysteresis behavior and non-linearity of the valves. In equation 3.1 the cost is also shown as a function of **d**. This is the case as the three elements of **d**, which are the reservoir valve openings (CV101, CV102 and CV103), are time-varying. The total availability of gas, which is a shared input constraint, can be expressed as below using the structure in 2.30.

$$\begin{aligned} g(\mathbf{u}, \mathbf{d}) &:= \sum_{i=1}^{3} g_i(u_i, d_i) - g^{\max} \\ &= Q_{gl,1} + Q_{gl,2} + Q_{gl,3} - Q_{gl}^{\max} \end{aligned} \tag{3.2}$$

where the constraint is measured directly, and the FICs is used to drive $Q_{gl,i}$ to $Q_{gl,i}^{sp}$.

In this paper we have the assumption of an always active constraint. The constraint can therefore be treated as an equality constraint. Consequently, the utilization of the max selectors 2.37b and 2.43 for updating λ in the constraint control of the dual decomposition structure with and without override, as discussed in section 2.5.2 and 2.5.3, is unnecessary. This is due to the fact that the value of the max selector will always be positive under these circumstances.

4 Distributed Feedback-optimizing Control Setup

In accordance with the problem formulation described in section 2.5, the distributed feedback-optimizing control structure for our experimental setup is implemented. The experimental setup have three wells, therefore the problem is decomposed into three subsystems. for each subsystem, we have employed a local gradient estimator that estimates the gradient of the local constraints $\nabla Q_{gl,i}$ and local cost gradients $\nabla Q_{l,i}$. As there is an available dynamic model of the system which is proven to be reliable[20], forward sensitivity analysis is implemented in order to estimate the gradients. It is also necessary to estimate the current states of the system, both differential and algebraic, to compute the local sensitivities, see section A.2. In this paper, an extended Kalman filter (EKF), see section A.1, is implemented in each subsystem, this relies only on local measurements to make these estimations. However, any suitable dynamic estimator may be utilized, provided it is capable of generating accurate estimates of the states while properly filtering out measurement noise.

When the control structure is defined the controllers must be tuned. The fastest possible sampling rate of the data acquisition software in the experimental rig is 1 second. In theory both the local controllers and the coordinator controllers (central constraint controllers) could be executed at the same rate. Despite that, the controllers could start competing against each other, depending on their tuning, this might cause instability in the system. As a result, timescale separation between the controllers are necessary. Prior to initial runs on the lab-rig, all controller tunings were implemented and evaluated in a MATLAB model of the lab-rig. This preliminary step ensured that the controller tunings provided relatively good performance before further testing was done in the lab-rig itself.

4.1 Primal-based Distributed Feedback-optimizing Control Setup

4.1.1 Implemented Control Structure

Based on the problem formulation in section 2.5.1, a distributed feedback-optimizing control scheme with online primal decomposition is implemented for the experimental rig. The central constraint controllers receives the estimated local Lagrange multipliers for each subsystem, these are derived from the local cost gradient and local constraint gradient as shown in equation 2.25. The Lagrange multipliers is then utilized to calculate the new setpoints, $Q^{sp}_{gl,i}$, which is sent to the individual subsystems. One subsystem needs to be chosen as the compensator in the central constraint control scheme, the new setpoint for this specific subsystem is provided by equation 2.20. The reasoning for choosing a specific subsystem as compensator is discussed in detail in section 4.1.2.

As the controller setup is defined, the central constraint controllers needs to be tuned. The SIMC tuning rules, described in section 2.1.3, are employed to tune the controllers. In this paper, integral controllers have been utilized, as proportional-integral controllers was found to aggressive in the MATLAB model. As a result the integral gain for the central constraint controllers are given by

$$K_{I,i} = \frac{1}{K_{\lambda_i}(\tau_{c,i} + \theta_i)} \tag{4.1}$$

where K_{λ_i} and θ_i are the step response and the time delay of the constraints by the local Lagrange multiplier, $\tau_{c,i}$ is the closed-loop time constants and $i = 1, 2, 3$ is the well index.

In order to determine the $K_{\lambda,i}$ and θ_i, an analysis of the systems step response is conducted. Ideally, the tuning parameter $\tau_{c,i}$ can be chosen as 1 to drive the system to steady-state as fast as possible. However, this aggressive approach may not be suitable, considering that the time-scale of the central constraint controller should be slower than that of the plant, where the three FICs has a time-constant of 5. Therefore, the tuning parameters are adjusted based on observation of the results and practical justifications. The resulting controller and tuning parameters for the primal-based method are presented in table 4.1.

4.1.2 Determine Compensator System

For the primal-based distributed feedback-optimizing control setup a compensator subsystem for the central constraint controller must be determined. In this case study, the compensator subsystem is determined by comparing the accumulated profit with the three different subsystems chosen as compensator. This analysis is done with the MATLAB model of the lab-rig. The simulation results is presented in figure 4.1, where the profit difference is calculated as

$$P^k_{diff} = \frac{P^k_3 - P^k_i}{P^k_i} \cdot 100 \tag{4.2}$$

where P^k_{diff} is the accumulated profit difference. The accumulated profit with well 3 chosen as the compensator is denoted P^k_3, while P^k_i is the accumulated profit with well i chosen as compensator, where $i = 1, 2$, at time-step k. The accumulated profit with the different wells chosen as compensator at time-step k is calculated as

$$P^k = \sum_{j=1}^{k-1} P^j \tag{4.3}$$

Based on the plot in 4.1, it is evident that the best performance is achieved when subsystem 3 is utilized as the compensator subsystem, although the difference between the different compensators are relatively small. The variation in performance can be attributed to the controller gains in each subsystem, which is presented in table 4.1. Notably, subsystem 3 has the highest gain magnitude. Thus, it can be concluded that in scenarios involving an input shared constraint, selecting the subsystem with the highest gain magnitude as compensator is recommended.

Given the selected compensator subsystem, and the description of the primal-based scheme in section 2.5.1 the implemented control structure of primal-based distributed feedback-optimizing control can be constructed. This is illustrated in figure 4.2, where well 1, well 2, well 3, FIC104, FIC105 and FIC106, in the experimental lab rig from figure 3.1, are inside the dashed green lines. The experimental lab rig has both input and output measurement noise, this is labeled by $\eta_{i,i}$ and $\eta_{o,i}$ respectively. In the flow controllers in the rig, the real manipulated variables are the valve openings,

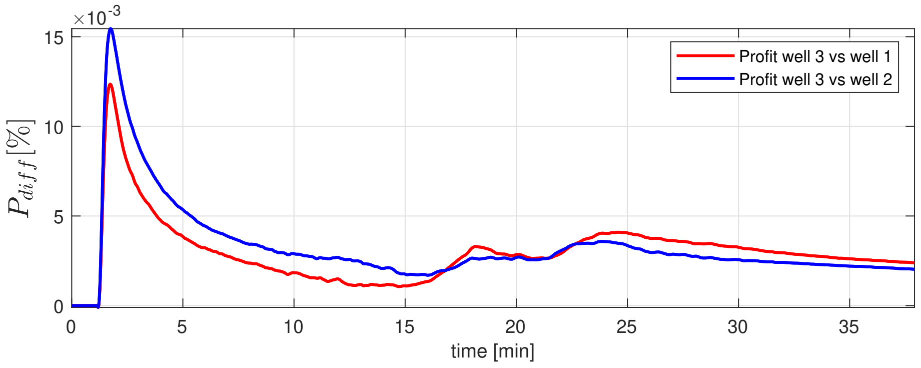

Figure 4.1: Comparison of accumulated profit with different compensator wells. Well 3 vs well 1 and well 3 vs well 2 denotes the comparison of accumulated profit with well 3 as compensator versus accumulated profit with well 1 and 2 as compensator respectively.

which is labeled by $v_{o,i}$. The differential states, $\hat{x}_i$, algebraic states, $\hat{z}_i$, and parameters/disturbances, $\hat{p}_i$, are estimated in the local dynamic model adaptation, which in this paper is an extended Kalman filter as discussed earlier.

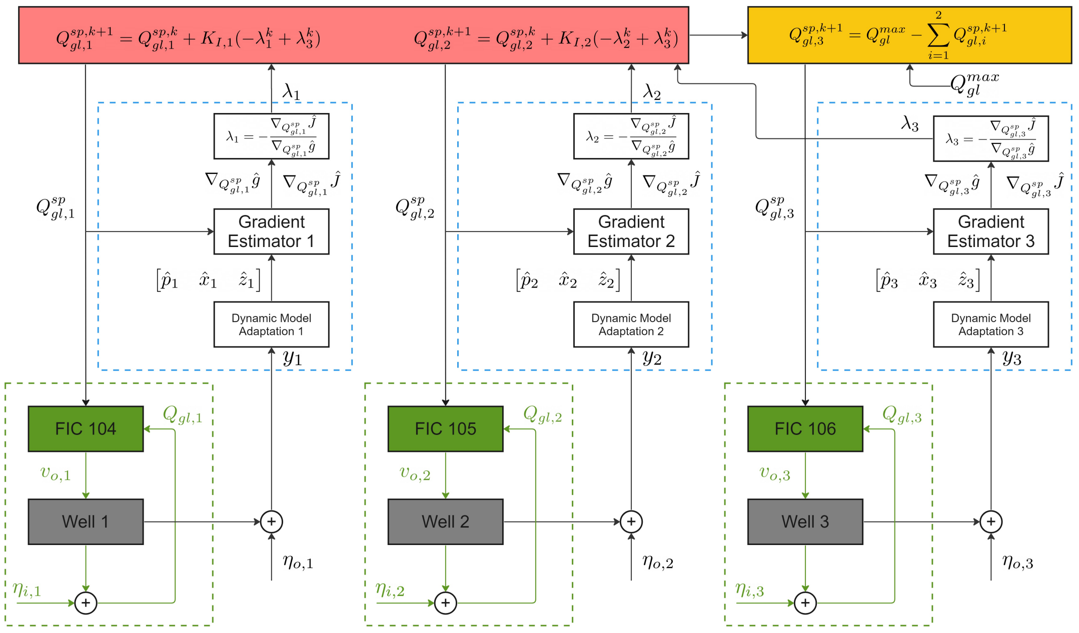
$Q_{gl,1}^{sp,k+1} = Q_{gl,1}^{sp,k} + K_{I,1}(-\lambda_1^k + \lambda_3^k)$
$Q_{gl,2}^{sp,k+1} = Q_{gl,2}^{sp,k} + K_{I,2}(-\lambda_2^k + \lambda_3^k)$
$Q_{gl,3}^{sp,k+1} = Q_{gl}^{max} - \sum_{i=1}^{2} Q_{gl,i}^{sp,k+1}$
Q_{gl}^{max}
λ_1
λ_2
λ_3
$\lambda_1 = -\frac{\nabla_{Q_{gl,1}^{sp}} \hat{J}}{\nabla_{Q_{gl,1}^{sp}} \hat{g}}$
$\lambda_2 = -\frac{\nabla_{Q_{gl,2}^{sp}} \hat{J}}{\nabla_{Q_{gl,2}^{sp}} \hat{g}}$
$\lambda_3 = -\frac{\nabla_{Q_{gl,3}^{sp}} \hat{J}}{\nabla_{Q_{gl,3}^{sp}} \hat{g}}$
$\nabla_{Q_{gl,1}^{sp}} \hat{g}$
$\nabla_{Q_{gl,1}^{sp}} \hat{J}$
$\nabla_{Q_{gl,2}^{sp}} \hat{g}$
$\nabla_{Q_{gl,2}^{sp}} \hat{J}$
$\nabla_{Q_{gl,3}^{sp}} \hat{g}$
$\nabla_{Q_{gl,3}^{sp}} \hat{J}$
$Q_{gl,1}^{sp}$
$Q_{gl,2}^{sp}$
$Q_{gl,3}^{sp}$
Gradient Estimator 1
Gradient Estimator 2
Gradient Estimator 3
$[\hat{p}_1 \quad \hat{x}_1 \quad \hat{z}_1]$
$[\hat{p}_2 \quad \hat{x}_2 \quad \hat{z}_2]$
$[\hat{p}_3 \quad \hat{x}_3 \quad \hat{z}_3]$
Dynamic Model Adaptation 1
Dynamic Model Adaptation 2
Dynamic Model Adaptation 3
y_1
y_2
y_3
FIC 104
FIC 105
FIC 106
$Q_{gl,1}$
$Q_{gl,2}$
$Q_{gl,3}$
$v_{o,1}$
$v_{o,2}$
$v_{o,3}$
Well 1
Well 2
Well 3
$\eta_{o,1}$
$\eta_{o,2}$
$\eta_{o,3}$
$\eta_{i,1}$
$\eta_{i,2}$
$\eta_{i,3}$

Figure 4.2: Primal-based distributed feedback-optimizing control scheme implemented in the experimental lab rig.[7]

4.2 Dual-based Distributed Feedback-optimizing Control setup

4.2.1 Implemented Control Structure

Based on the problem formulation in section 2.5.2, a distributed feedback-optimizing control scheme with dual decomposition is implemented for the experimental rig. For each subsystem there is an local gradient controller, which is implemented as a integral controller, that controls $c_i(\lambda)$ to a setpoint value of 0, from equation 2.33. The output calculated from the local gradient controllers, $Q_{gl,i}^{sp}$, are the gas-lift setpoints for each individual well. These are provided to the flow indicator controllers, which manipulates the air injection valve openings to achieve the mentioned setpoints in the experimental lab-rig for the respective subsystems. For the central constraint controller, the Lagrange multiplier is updated as shown in equation 2.37. As the constraint $\mathbf{g}$ can be directly measured from the lab-rig estimations are not necessary here. The dual-based scheme is set up in a cascade structure, where the local gradient controllers are the slaves and the central constraint controller act as the master.

As the controller setup is defined, the controllers needs to be tuned. The SIMC tuning rules, described in section 2.1.3, are utilized to tune the controllers. In this paper, integral controllers have been utilized for both the central constraint controller and the local gradient controllers, as proportional-integral controllers was found to aggressive from testing in the MATLAB model. As a result, the integral gain for the central constraint controller is given by

$$\alpha = \frac{1}{K_\lambda(\tau_{c,\lambda} + \theta_\lambda)} \tag{4.4}$$

where K_λ represents the step response and θ_λ corresponds to the time delay caused by the Lagrange multiplier in relation to the constraint. $\tau_{c,\lambda}$ denotes the closed-loop time constant, which dictates the progression of the constraints. The integral gain for the three local gradient controllers is expressed as

$$K_{I,i} = \frac{1}{K_{\mathbf{u}_i}(\tau_{c,\mathbf{u}_i} + \theta_{\mathbf{u}_i})} \tag{4.5}$$

where K_{u_i} and θ_{u_i} are the step response and the time delay of the gradients by the inputs, $Q^{sp}_{gl,i}$. τ_{c,u_i} is the closed-loop time constants that dictates the evolution of $c_i(\lambda)$ and $i = 1, 2, 3$ is the well index. To determine K_λ, θ_λ, K_{u_i} and θ_{u_i} the step response is analysed. The experimental lab-rig has a sampling rate of 1 second, hence, the central constraint controller and the local gradient controllers could be executed at the same rate. Nevertheless, depending on their respective tuning, these controllers may potentially compete with each other, which can result in system instability. Therefore, it is necessary to establish a time-scale separation between these controllers to mitigate such issues.

The convergence rate to the setpoint is determined by the closed-loop time constant $\tau_{c,i}$ of the control loop for each subsystem. In the case of a linear first-order system, the approach towards the steady-state can be characterized by $(1 - e^{-\tau_{c,\lambda}/\tau_{c,u_i}})$ where $\tau_{c,\lambda}$ represents the convergence time of the central constraint, and τ_{c,u_i} denotes the closed-loop time constant of the local gradient control in subsystem i as stated above. Hence, when $\tau_{c,\lambda}/\tau_{c,u_i} = 5$, approximately 99.3 % convergence is achieved, and is therefore considered reached. This serves as a rule of thumb for establishing a time scale separation of at least 5 between control layers.[28] Larger values of time scale separation offer increased robustness against process gain variations. However, excessively large values result in slower convergence of the controllers. For practical reasons, a recommended range for timescale separation is often between 5 to 10.

While tuning the time constant parameters $\tau_{c,\lambda}$ and $\tau_{c,\mathbf{u}_i}$ the concept of time scale separation, where $\epsilon_i = \frac{\tau_{c,\mathbf{u}_i}}{\tau_{c,\lambda}}$ should be around 0.2, should be considered. This implies

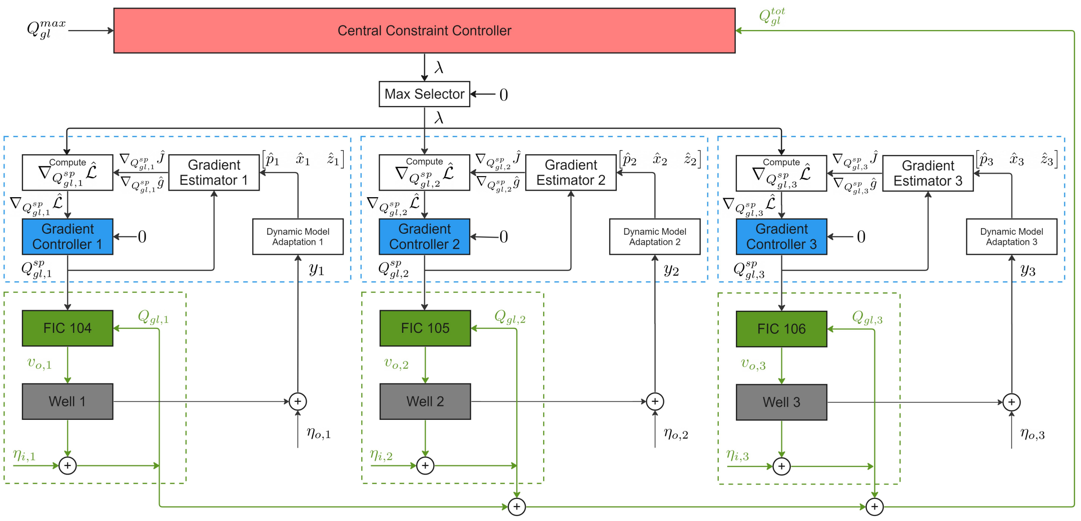

Figure 4.3: Dual-based distributed feedback-optimizing control scheme implemented in the experimental lab rig.[7]

that the outer control loop should be slower than the inner control loop. The inner time constants, $\tau_{c,\mathbf{u}_i}$ is adjusted from observation of results and practical justifications, based on the same arguments as for tuning the parameter in the primal-based method, section 4.1. The resulting controller and tuning parameters for the dual-based method are presented in table 4.2. The implemented control structure on the experimental lab-rig is shown in figure 4.3.

4.3 Dual-based Distributed Feedback-optimizing Control Setup with Override

4.3.1 Implemented Control Setup

Based on the problem formulation in section 2.5.3, a distributed feedback-optimizing control scheme with dual decomposition with override is implemented for the experimental rig. For each subsystem there is an local gradient controller, which is implemented as a integral controller, that controls $c_i(\lambda)$ to a setpoint value of 0, from equation 2.39. The output calculated from the local gradient controllers, $Q_{gl,i}^{sp}$, are the gas-lift setpoints for each individual well. These are provided to the flow indicator controllers, which manipulates the air injection valve openings to achieve the mentioned setpoints in the experimental lab-rig for the respective subsystems. For the central constraint controller, the Lagrange multiplier is updated as shown in equation 2.42. It controls the auxiliary constraint $u_{ind,i}^{sp} - u_{dir,i}^{sp}$. One selected subsystem involves a override constraint controller, including the local gradient controller, where the actual constraint $\mathbf{g}(\mathbf{u}, \mathbf{d})$ is controlled in a fast time scale. The gas-lift setpoint, $Q_{gl,i}^{sp}$, for this subsystem is decided with a selector, which is implemented as $\min(u_{dir,i}^{sp}, u_{ind,i}^{sp})$.

In addition to the local gradient controllers and central constraint controller mentioned in section 4.2, an override constraint controller is required for the dual-based with override method. Implemented as a integral controller, as proportional-integral controllers was found to aggressive from testing in the MATLAB model, the integral gain is given by

$$K_{I,i} = \frac{1}{K_{\mathbf{g},i}(\tau_{\mathbf{g},i} + \theta_{\mathbf{g},i})} \tag{4.6}$$

where $K_{\mathbf{g},i}$ and $\theta_{\mathbf{g},i}$ are the step response and the time delay of the constraint by the selected input $Q^{sp}_{gl,i}$. $\tau_{\mathbf{g},i}$ is the closed-loop time constant that dictates the evolution of the constraint $\mathbf{g}$. The implemented controller tuning are determined in the same manner as explained for the dual-method above.

As mentioned above, the input setpoint $Q^{sp}_{gl,i}$ is selected by a min selector. As a result it is necessary to include anti-windup in the override constraint controller. This must be included to avoid continuous integration of the deviation from the constraint when the output from the gradient controller is selected. Without anti-windup, the the override constraint controller could end up having slow response to constraint violation. Therefore, implementation of back off would be necessary. For this case, the anti-windup limit the integration of constraint deviation by controlling $(Q^{sp}_{gl,i} - Q^{sp}_{dir,i})$, with anti-windup gain given as[19]

$$K_{AW} = \frac{1}{\tau_I} \tag{4.7}$$

where τ_I is the integral time. As a result, the override constraint controller output is updated using I-control with anti-windup as follows.

$$Q^{sp,k+1}_{dir,i} = Q^{sp,k}_{dir,i} + K_I(7.5 - Q_{gl,tot}) + K_{AW}(Q^{sp,k}_{gl,i} - Q^{sp,k}_{dir,i}) \tag{4.8}$$

These controller and tuning parameters are presented in table 4.3. The implemented control structure on the experimental lab-rig is shown in figure 4.6.

4.3.2 Choosing Subsystem with Override Control

As there is only one constraint and three subsystems in the experimental lab-rig, there are three possible configurations of implementing override in this system, where the override scheme only should be implemented in one subsystem. In order to select a subsystem with the override scheme in this paper, two factors are taken into consideration.

To guarantee that the chosen configuration does not reach local input constraints, the local maximum or minimum gas lift flow rate mentioned in section 3.1, each is tested in the lab-rig model developed in MATLAB. It is important to prevent such situations because the current formulation does not incorporate these local input constraints. Including these constraints into the problem formulation would increase the complexity, potentially diverting from the main discussion in this paper. The simulation run results is shown in figure 4.4, where it is clear that applying the override scheme in subsystem 1 leads to saturation of the local input constraint in the respective system.

The constraint satisfaction of each configuration should also be taken into account. The metric of interest in this case is the maximum magnitude of the constraint violation. The results from the lab-rig model developed in MATLAB is presented in figure 4.5, where the total input constraint, Q^{max} at 7.5, is the constraint that should be satisfied.

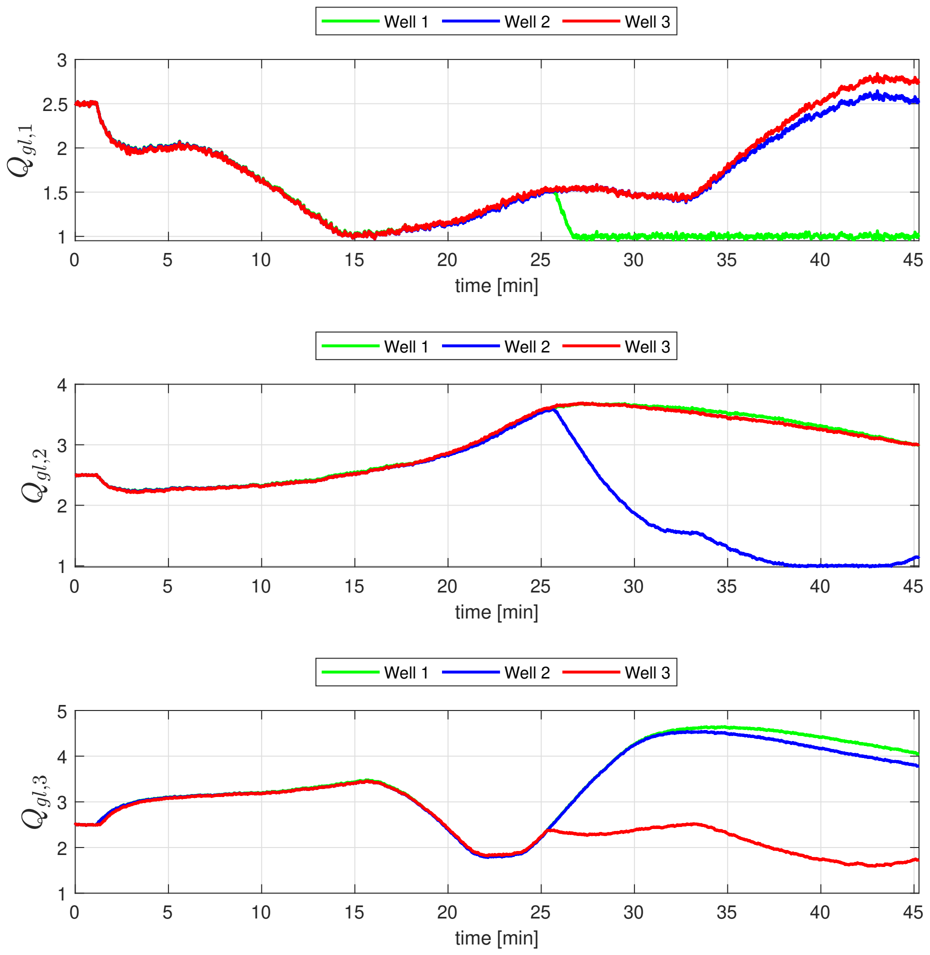

Figure 4.4: Comparison of the gas lift flow rates for every well with override constraint control implemented in the three different subsystems.

From this it is easy to conclude that it should be avoided to add the override scheme on subsystem 1, as it results in significant constraint violation. Override on subsystem 2 and 3 have more similar behavior. However, override on subsystem 3 has a better performance in terms of constraint violation. From figure 4.4, it is shown that these two methods have very similar behavior in therms of violating the local input constraints. As a result, the override scheme is implemented on subsystem 3 in this work. The resulting controller and tuning parameters for the dual-based with override method are presented in table 4.3, while the control structure which is implemented on the experimental lab-rig is shown in figure 4.6.

Another approach to choosing which subsystem the override scheme is implemented on is presented in Dirza et al.[11]. In this method, the decision is based on the manipulated variables sensitivities to its local disturbances.

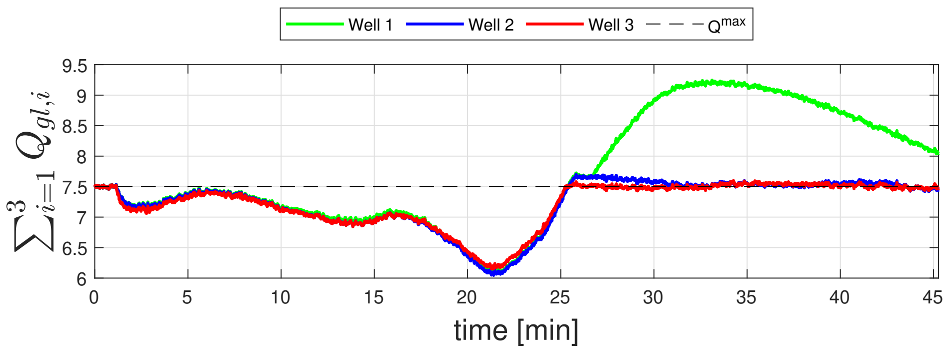

Figure 4.5: Comparison of the constraint satisfaction (total implemented gas lift) with override constraint control implemented in the three different subsystems.

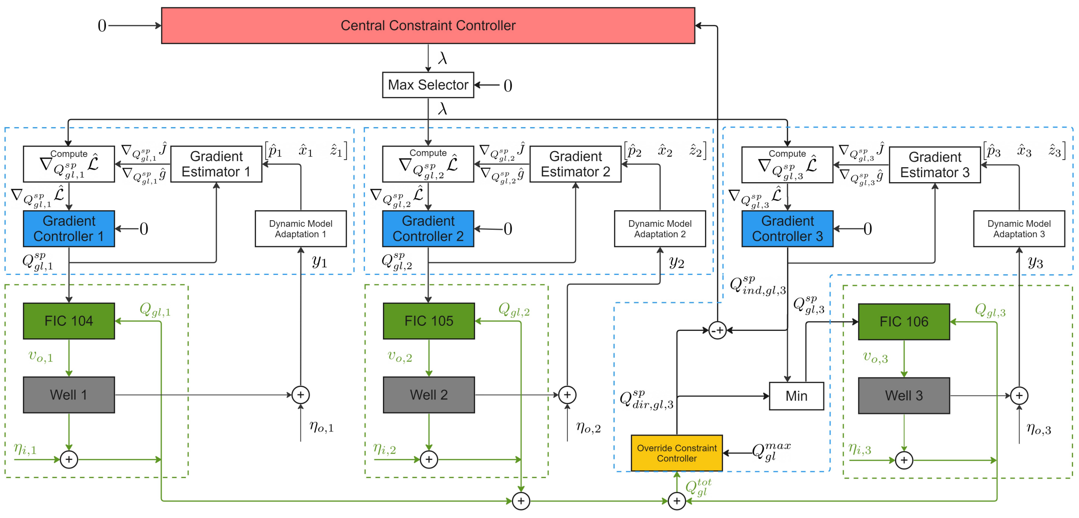

Figure 4.6: Dual-based with override distributed feedback-optimizing control scheme implemented in the experimental lab rig.[7]

Description	Variable	Value
Experimental rig sampling time	T_s	1 s
Time constant	$\tau_{c,1}$	25 s
Time constant	$\tau_{c,2}$	25 s
Time constant	$\tau_{c,3}$	25 s
Time delay	θ_{λ_1}	0 s
Time delay	θ_{λ_2}	0 s
Time delay	θ_{λ_3}	0 s
Lagrange multiplier gain	K_{λ_1}	-2.522
Lagrange multiplier gain	K_{λ_2}	-2.918
Lagrange multiplier gain	K_{λ_3}	-3.698
Integral gain	$K_{I,1}$	-0.016
Integral gain	$K_{I,2}$	-0.014
Integral gain	$K_{I,3}$	-0.011

Table 4.1: Controller and tuning parameters for primal decomposition

Description	Variable	Value
Experimental rig sampling time	T_s	1 s
Central constraint controller		
Time constant	$\tau_{c,\lambda}$	125 s
Time delay	θ_{λ}	0 s
Lagrange multiplier gain	K_{λ}	0.907
Integral gain	α	0.0088
Local gradient controllers		
Time constant	τ_{c,u_1}	25 s
Time constant	τ_{c,u_2}	25 s
Time constant	τ_{c,u_3}	25 s
Time delay	θ_{u_1}	0 s
Time delay	θ_{u_2}	0 s
Time delay	θ_{u_3}	0 s
Input gain	K_{u_1}	2.53
Input gain	K_{u_2}	2.93
Input gain	K_{u_3}	3.70
Integral gain	$K_{I,1}$	0.016
Integral gain	$K_{I,2}$	0.014
Integral gain	$K_{I,3}$	0.011

Table 4.2: Controller and tuning parameters for dual decomposition

Description	Variable	Value
Experimental rig sampling time	T_s	1 s
Central constraint controller		
Time constant	$\tau_{c,\lambda}$	125 s
Time delay	θ_λ	0 s
Lagrange multiplier gain	K_λ	0.2171
Integral gain	α	0.0368
Local gradient controllers		
Time constant	τ_{c,u_1}	25 s
Time constant	τ_{c,u_2}	25 s
Time constant	τ_{c,u_3}	25 s
Time delay	θ_{u_1}	0 s
Time delay	θ_{u_2}	0 s
Time delay	θ_{u_3}	0 s
Input gain	K_{u_1}	2.53
Input gain	K_{u_2}	2.93
Input gain	K_{u_3}	3.70
Integral gain	$K_{I,1}$	0.016
Integral gain	$K_{I,2}$	0.014
Integral gain	$K_{I,3}$	0.011
Override constraint controller		
Integral Time	τ_{I,u_3}	2.5 s
Time constant	τ_{c,u_3}	10 s
Time delay	θ_{u_3}	0 s
Input gain	K_{u_3}	1
Integral gain	$K_{I,3}$	0.1
Anti-windup gain	K_{aw}	0.4

Table 4.3: Controller and tuning parameters for dual decomposition with override

5 Experimental Results

5.1 Disturbances

Using the control and tuning parameters presented in section 4 above, the three approaches, primal-based, dual-based and dual-based with override, are implemented. In figure 5.1 the reservoir valve openings, CV101, CV102 and CV103, development throughout the experiment is presented. These are considered the disturbance in this work. The first disturbance occurs when CV101 gradually close from $t = 6.5$ to $t = 14$ minutes. During this time period, it is expected a decrease of gas lift injection for well 1, and a redistribution of the excess gas lift among the other wells. The second disturbance occurs when CV103 also gradually close from $t = 15.5$ to $t = 21$ minutes. During this time period, it is expected a decrease of gas lift to well 3. This is also expected to reduce with a higher rate because the "value" of this well is higher than the others. As a result, the other wells will receive more gas lift with a higher rate. The third disturbance occurs when CV103 is gradually opened from $t = 24$ to $t = 29.5$ minutes. The behavior of the gas lift distribution is expected to have the reverse reaction to that of the second disturbance. The fourth, and last, disturbance occurs when CV101 gradually open from $t = 33$ to $t = 42.5$ minutes. As before, it is expected an reverse reaction to what happens during the first disturbance. In this work, sudden disturbance change is avoided to ensure that the plant is adjusted smoothly by the implemented controllers.

In the experimental rig, a programming environment, LABVEIW[3], is used to automate the implementation of the disturbances. As a result, it is possible to use the same disturbance profile in independent experiments. The control is not activated during the first minute of the experiments. Therefore, all runs start with the same amount of gas lift injection in every well.

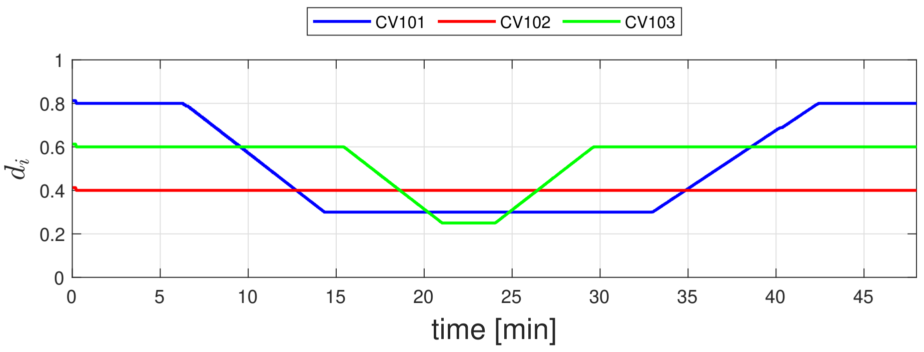

Figure 5.1: Disturbances implemented in the experiments

5.2 Comparison of Distributed Feedback-optimizing Control

In order to evaluate the effectiveness of the implemented control structure, a series of experiments were conducted. Multiple experiments were performed for each control method to ensure reliable results. The outcomes presented in this paper are the averages of the three best, or most average results obtained for each method. This approach was adopted to mitigate the influence of potential outliers and to provide a more robust assessment of the different control structures performances.

The averaged results presented in the following sections provide valuable insights into the performance of the different implemented control structures. These outcomes serve as a basis for understanding the effectiveness of the different control methods and their suitability for real-world applications.

5.2.1 Constraint Satisfaction

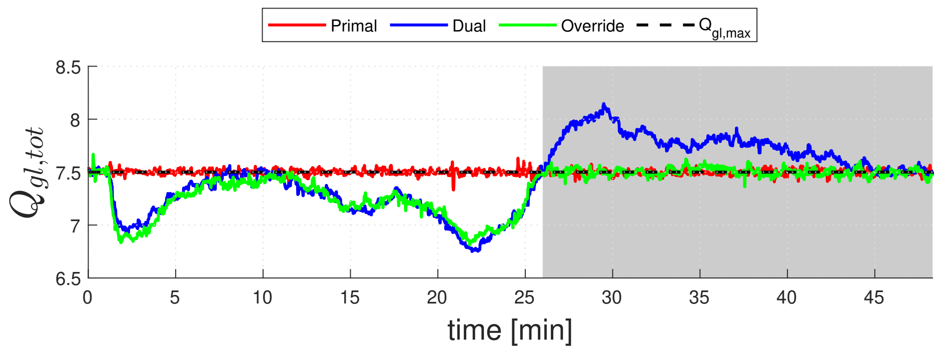

Figure 5.2: The constraint satisfaction of primal-based, dual-based and dual-based with override.

In figure 5.2 the constraint satisfaction of the three implemented approaches are displayed. It is clear from this result that the primal-based approach has a better constraint satisfaction than dual-based with and without override approaches, as it provides a total implemented gas lift of 7.5 sL/min during the whole time frame of the experiment. This behavior is desirable, as it is considered an always active input shared constraint in this case study. The reason why primal-based has this property, is because of the compensator subsystem, in this case subsystem 3, absorbs the deviations from the constraint. This absorbing ability stems from how the gas lift setpoint for the compensator subsystem is calculated, see equation 2.20. In relation to the operational implications, by implementing the primal-based method it is possible to operate the experimental lab-rig without any significant back off while maintaining feasibility consistently.

For the dual-based method however, it becomes evident that the implementation of back off is necessary, particularly from $t = 26$ minutes when the disturbances is increasing. In figure 5.2 this time period is highlighted in gray, in this area the dual-based method is violating the constraint and is therefore not a valid approach. The constraint violation is a result of the necessary time-scale separation between the control layer, from the plant to the local gradient controllers as well as from the local gradient controllers to the central constraint controller. Due to the comparatively slower nature of the central constraint control in the dual-based method, it is unable to effectively track the disturbances to the same extent as the primal-based central constraint controllers. In figure 5.3 the result of dual-based with back off is included as well. The back off is set to $\xi = 0.5$ because at $t = 29$, when the dual-based approach achieves the highest magnitude of constraint violation, the total implemented gas lift for the dual-based approach is around $Q_{gl,tot} = 8\ sL/min$. As a result, dual-based with back off implemented have no violation of the total implemented gas lift constraint and is therefore feasible during the entirety of the experimental runs.

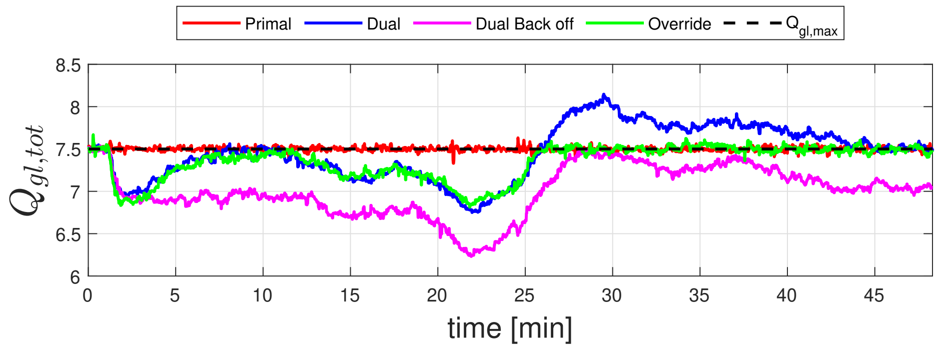

Figure 5.3: The constraint satisfaction of primal-based, dual-based, dual-based with back off and dual-based with override.

For the dual-based with override, the results in figure 5.2 and 5.3 show that the method remain valid throughout the whole experiment. The override constraint controller ensures that the total gas lift input is satisfying the maximum constraint, $Q_{gl,max}$. However, for this case, the dual-based with override method can be classified as worse than the primal-based method in therms of the constraint satisfaction. This is because it is considered an always active constraint, therefore, the dual-based with override is guaranteed not achieving optimum performance for the time period from $t = 0$ to $t = 26$. This is because the total gas lift implemented is not 7.5 sL/min during this time period.

For all implemented dual-based approaches the total implemented gas lift moves away from the constraint at around $t = 1$. As stated earlier, the control structures are activated around this time. This behavior is a result of both the initial value of the Lagrange multipliers and the initial implemented gas lift for each well, $Q_{gl,i}$, is deviating from the optimal operation point. As a result, the faster responding gradient controllers drive the system away from the constraint before the central constraint controller has time to reach the optimum operation point. From figure 5.7, it can be observed that the Lagrange multipliers for the dual approaches starts of at 9.5 before decreasing. In other words the Lagrange multipliers is larger than the optimum operation point initially.

5.2.2 Optimal Setpoints

In figure 5.4 the gas lift setpoints which is implemented in the FICs in the experimental lab-rig provided from the different control approaches is presented. As the dual-based and dual-based with override method do not satisfy the constraint during the entire experiment, as is the case for the primal-based method, one might expect that the dual-based methods have a slower response in the setpoints changes than the primal-based method. However, for the disturbance behavior in these experiments, this is not the case for the subsystems which is undergoing disturbance changes. During the time period from $t = 6.5$ to $t = 14$, when there is a disturbance in well 1, figure 5.4 shows that the dual-based approaches provide a slightly faster setpoint response for subsystem 1 compared to the primal-based approach. However, for the remaining subsystems, the primal-based approach demonstrates a faster response.

This behavior is rooted in the characteristics of the local gradient controllers within the dual-based methods framework. When a disturbance occurs in one of the subsystems, the associated local gradient estimators have a fast response to it, which leads to a fast response in the corresponding gradient controller. This can be observed for subsystem 1 in figure 5.4 from $t = 10$ to $t = 15$, because they are directly affected by the local disturbance, which is shown in equation 2.33 and 2.39. On the other hand, the remaining local gradient estimators are only directly influenced by their respective local disturbances. As a result, there is no response from these controllers until the central constraint control reacts to the constraint violation. The delay in response is therefore due to the necessary time scale separation between the central constraint controller and the local gradient controllers in the dual-based approaches framework, as explained in section 4.2 and 4.3. In comparison, for the primal-based structure, all subsystem have the same reaction time in regards of local disturbances. This is because the distribution of the gas lift is "negotiated" by the estimated local Lagrange multipliers in the central constraint controllers. As a result, all wells responds to any disturbances at the same time. Consequently, the dual-based methods achieve faster rejection to local disturbances in the respective subsystems. However, the primal-based method demonstrates better constraint control.

Figure 5.5 shows the actual gas lift flow rates in the experimental lab-rig. This plot differs slightly from the provided input setpoints presented in figure 5.4. This difference is a result of two factors: input measurement noise and the time required for the gas flowrate controllers, FIC104, FIC105 and FIC106 from figure 3.1, to adjust the actual gas lift flow rate, $Q_{gl,i}$ to match the setpoint gas lift flowrate, $Q_{gl,i}^{sp}$.

As the dual-based approach required back off, which is discussed in section 5.2.1, the gas lift flow rates from dual-based with back off approach is also included in figures 5.4 and 5.5. This approach results in the same behavior as the dual-based approach, the only difference being a downward shift in the injected gas lift flow rates, which is expected.

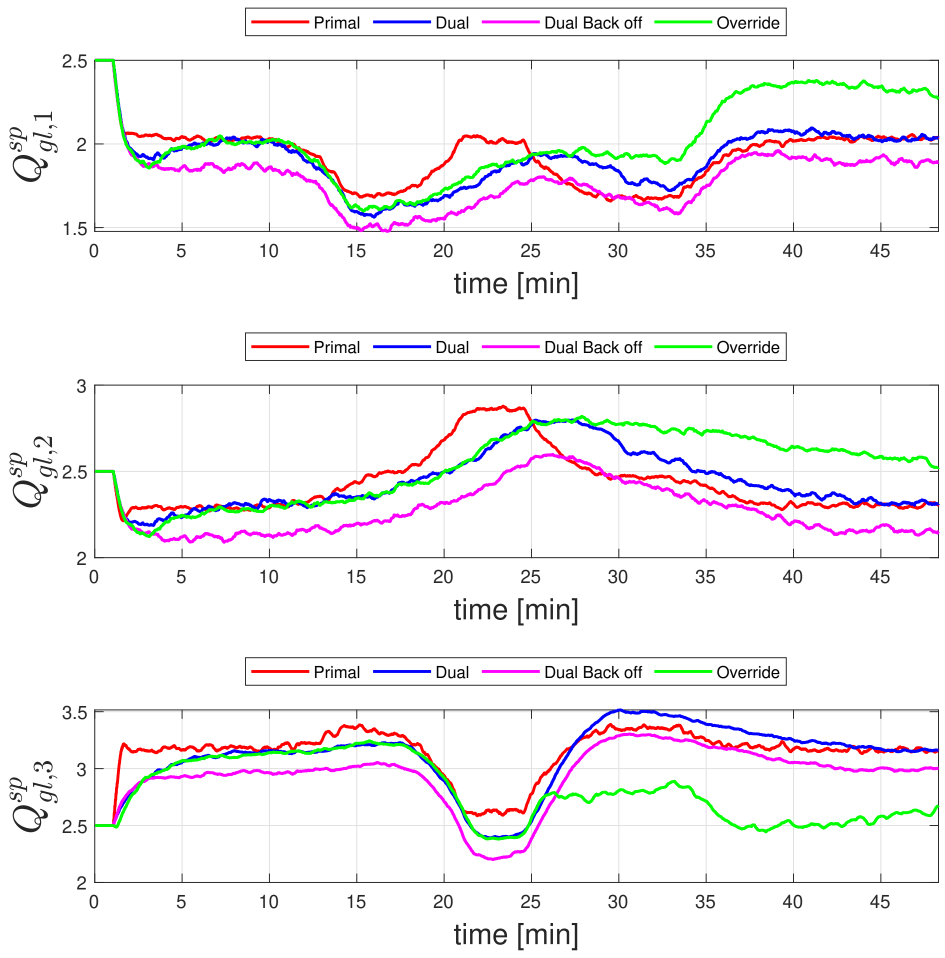

Figure 5.4: The gas lift flow rate setpoints, $Q_{gl,i}^{sp}$, for every well for primal-based, dual-based and dual-based with override control setup.

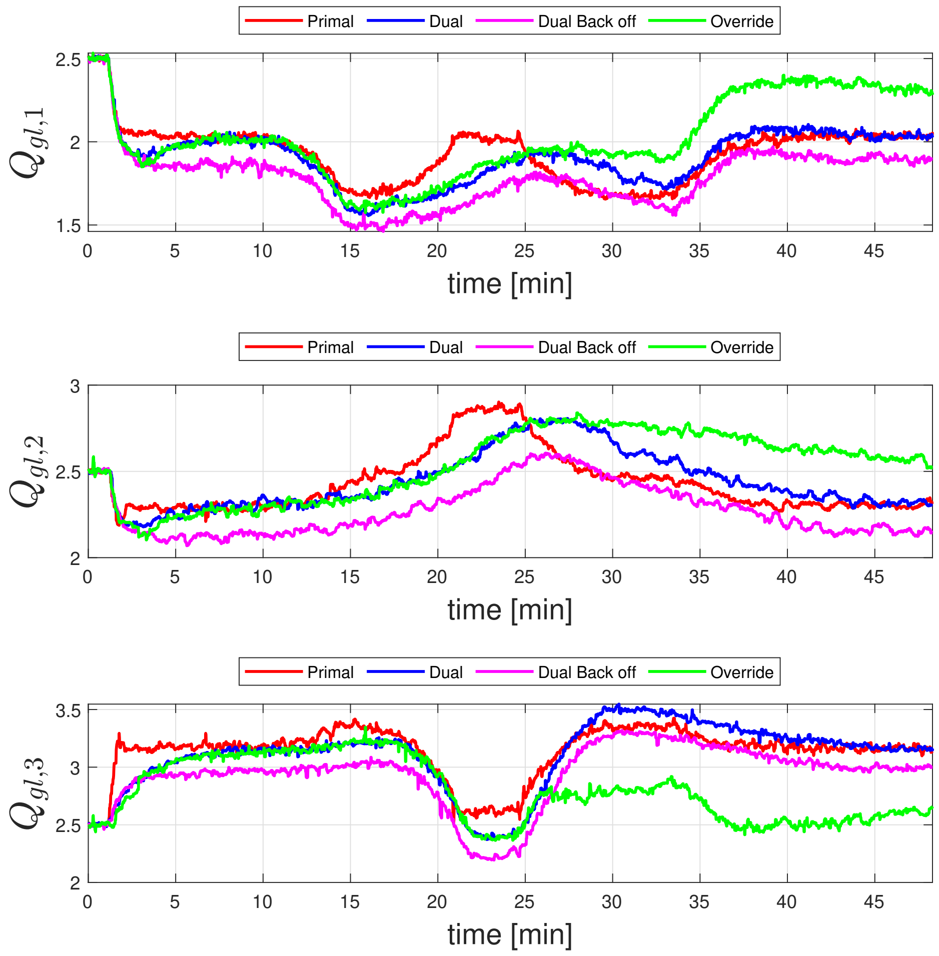

Figure 5.5: The measured gas lift flow rates, $Q_{gl,i}$, in every well for primal-based, dual-based and dual-based with override control setup.

5.2.3 Lagrange Multipliers

Figures 5.6 and 5.7 show the Lagrange multipliers from the experimental results. The Lagrange multipliers from the primal-based method is presented in figure 5.6, from the result it is clear that these are controlled fast, and they converge to the same values. As they respond fast, the result is presented with a moving mean of 10 seconds, this is done to get a better presentation of their development throughout the experiment. This fast behavior is due to the fast time scale the central constraint controller operates in for the primal-based method, which is presented in section 4.1.

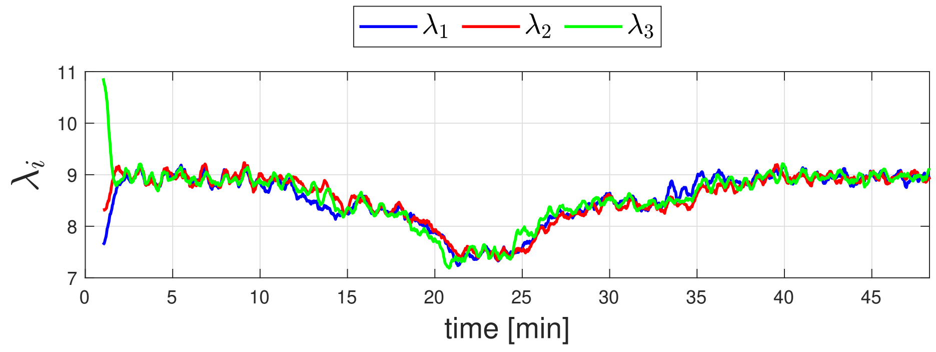

Figure 5.6: Local Lagrange multipliers from experimental runs with primal-based method with a moving mean of 10 second.

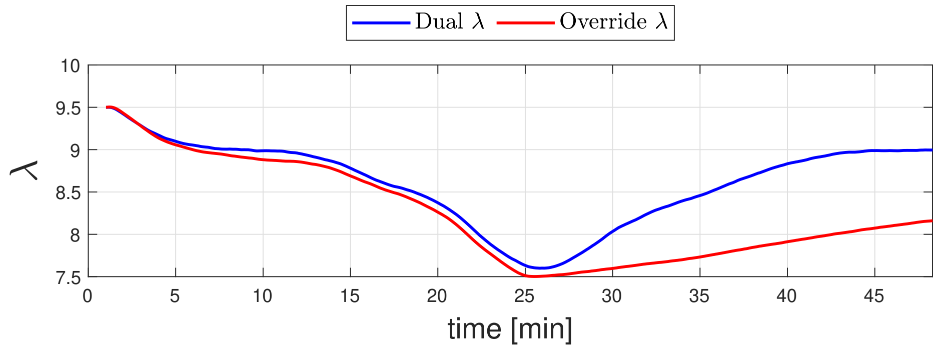

Figure 5.7: Local Lagrange multipliers from experimental runs with dual-based and dual-based with override methods.

Figure 5.7 show the Lagrange multiplier for the dual-based and dual-based with override methods. From comparison with the result from primal-based method, it is clear that these methods have a slower response in the control of the Lagrange multiplier. This behavior is due to the tuning of the controllers and the fact that the central constraint controllers for both dual-based methods operates on a slower time scale, which is explained in sections 4.2 and 4.3. At time $t = 10$ minutes the dual-based Lagrange multiplier is converged for a short period. This can be confirmed in figure 5.2 as the total input constraint is satisfied at this time for this method.

From $t = 26$ and on wards in figure 5.7 it can be observed that the evolution of the Lagrange multiplier in the dual-based with override approach is much slower than for the dual-based approach. During this time frame the override constraint controller is activated. As a result, the control of the auxiliary constraint in the central constraint controller becomes to slow and consequently the Lagrange multiplier does not converge during the time frame of the experiments. This problem could be accommodated by reducing the time scale separation between the control layers. However, this could lead to instabilities when the override constraint controller is deactivated.

5.2.4 Accumulated Profits

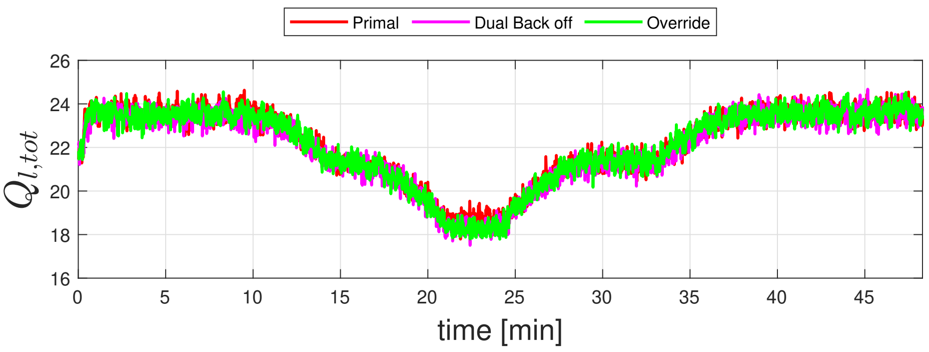

Figure 5.8: Comparison of the total liquid flow rate in the experimental runs for primal-based, dual-based and dual-based with override control setup

For the economic evaluation of the different implemented control approaches, the basic dual-based approach is not considered. This is because of the constraint violations shown in figures 5.2 and 5.3. Consequently, this approach is not feasible and the dual-based with back off is considered instead. In figure 5.8 the total liquid production of the three implemented control schemes is presented. However, it is not possible to gather any valuable information in terms of how the economic performances of the different methods compare to each other. To better display the economic performance of the implemented methods, they are compared to a naive approach in therms of gas lift distribution in figures 5.9 and 5.10. For the naive approach the inputs are considered fixed as

$$\mathbf{u} = \begin{bmatrix} \frac{Q_{gl}^{max}}{3} & \frac{Q_{gl}^{max}}{3} & \frac{Q_{gl}^{max}}{3} \end{bmatrix}^T \tag{5.1}$$

This is a representation of the case where no information is available. As a result, the best solution is to divide the available gas lift equally among the three wells in the system as the constraint is considered always active at the optimal operation point. The profit difference between the implemented control methods and the naive approach in figure 5.9 is calculated as

$$P_{diff} = \frac{P - P_{naive}}{P_{naive}} \cdot 100 \tag{5.2}$$

where P_{diff} is the percentage of profit difference, P is the profit of the method of interest, while P_{naive} is the profit of the naive approach. It is used a moving average

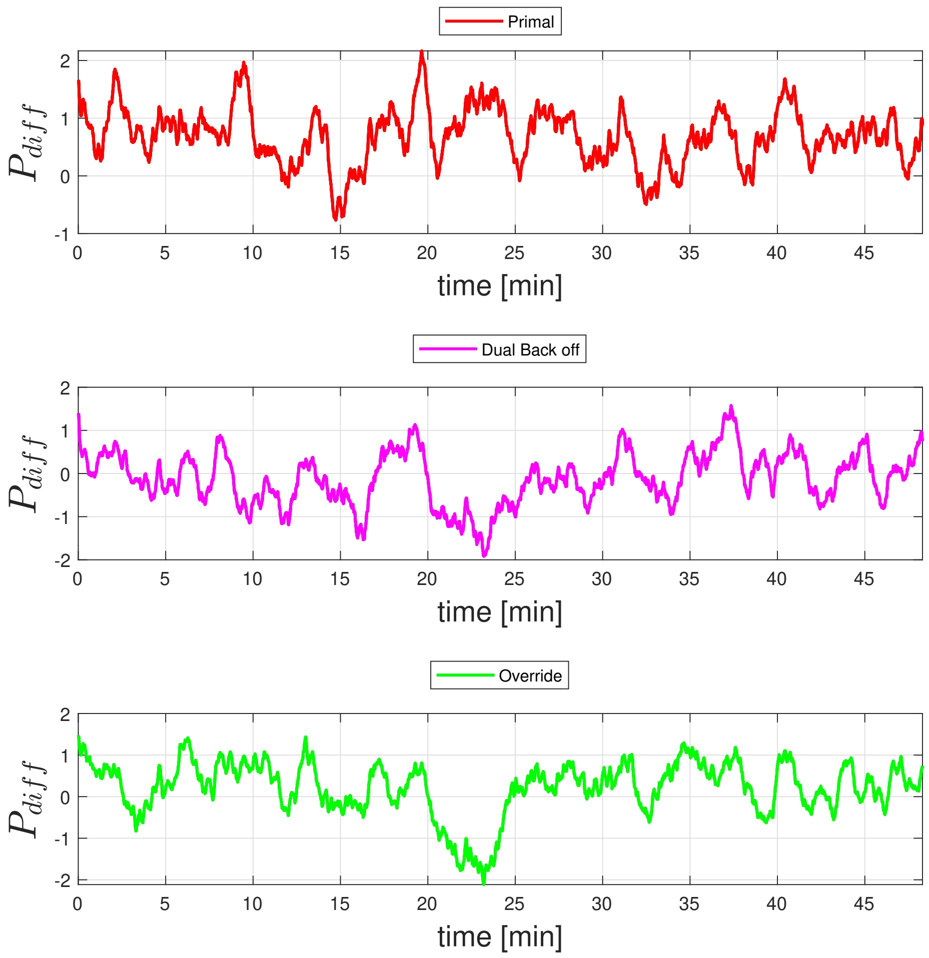

Figure 5.9: The instantaneous profit difference of the primal-based, dual-based and dual-based with override methods compared to the naive approach.

of 60 seconds in order to smoothing out the instantaneous profit profiles, because the measurements are noisy.

In figure 5.10, the cumulative profit $\sum P_{diff}$ of the implemented methods are compared. From this result it is clear that the primal-based method is more profitable than the other methods in this experimental case.

For the dual-based method and dual-based with override method it is expected similar economic performance during the first 26 minutes of the experiment. However, this is not the case because of the back-off implemented for dual-based method in order to satisfy the input shared constraint. As a result, the dual-based method has a smaller cumulative profit than the naive approach and is therefor worse than not controlling the gas lift distribution for this case.

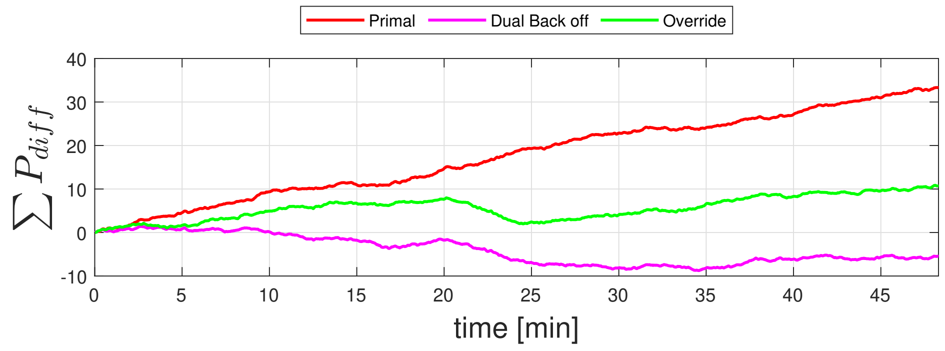

Figure 5.10: The accumulated profit difference between the primal-based, dual-based and dual-based with override methods and the naive approach.

5.3 Consistency of Results

Several measures have been taken to ensure the reliability of the experimental results presented in section 5. Firstly, there have been conducted at least five runs for each implemented method in order to account for random fluctuations, system variations and other factors that may influence the results. By having more runs, the robustness of the findings is enhanced, providing a more reliable assessment of the different implemented feedback-based optimization control structures performance.

Furthermore, the averaging of the three most normal runs is a sensible approach to mitigate the impact of outliers. Outliers can arise due to various factors, such as measurement errors or unexpected disturbances. By excluding abnormal results from the average calculation, the influence of such outliers is removed from the results. This filtering process aims to obtain a more better and stable representation of the system behavior during the experiments.

The averaging of the results also serves to reduce measurement noise in the result plots. Measurement noise can be a result of measurement error, environmental factors, or system noise. By averaging the results, the random noise tend to cancel out, leading to smoother and clearer result plots. As a result, the interpretation of the results becomes easier and allows for a more accurate assessment of the performance of each implemented control structure.

5.4 Performance of the Distributed Feedback-optimizing Approaches

The results in section 5 indicates that the primal-based feedback optimization method outperforms the dual-based and dual-based with override feedback optimization methods for the case of input shared constraint. However, it is important to consider the dual-based approaches, as these are more general in terms of number of constraints it is able to handle. The primal-based method has mathematical limitations when it comes to multiple constraint problems. As presented in section 2.5.1, there must be less than or equal number of active coupling constraints as decision variables in each subsystem for the primal-based method while this is not the case for the dual-based methods.

6 Conclusion

In this work, three distributed feedback-optimizing control schemes, primal-based, dual-based and dual-based with override, have been validated in a experimental lab-rig emulating a subsea oil production network. Based on the experimental results following conclusions can be made.

All three schemes yield are able to yield the same asymptotic optimal performance without the use of a numerical solver. However, there is differences is the transient behavior in each control scheme. This is a result of the implemented control structure and control tuning chosen for the different schemes.

In both dual-based approaches, considering timescale separation between the local gradient controllers and the central constraint controller is necessary. When the central constraint controller is tuned to operate on the same timescale as the local gradient controllers, instability and oscillatory behavior can arise. The timescale separation ensures that the control actions of the central constraint controller do not interfere with the dynamics of the local gradient controllers, preventing such behavior. On the other hand, if the central constraint controller is tuned excessively slow, it can lead to slow convergence to the optimal steady-state, which result in suboptimal performance. Therefore, finding the appropriate timescale separation between the control layers is essential to ensure acceptable system control.

By inclusion of an override mechanism, the dual-based scheme offers the advantage of better active constraint satisfaction while at the same time preserving the favorable characteristics of the dual-based scheme. As a result, the economic performance is enhanced as the requirement of back off is vastly reduced or eliminated. However, the convergence of the dual variable throughout the experimental runs is unsatisfactory. To address this issue, one solution could be to consider the implementation of gain switching in the central constraint controller in order to accommodate the nonlinearity in the auxiliary constraint.

In this specific case, with always active input shared constraints, the primal-based scheme has the best performance both in terms of economics and constraint satisfaction among the three schemes. The good constraint satisfaction is made possible by the compensator subsystem, which ensures that all available gas lift is utilized.

6.1 Further Work

6.1.1 Improvements for Dual-based with Override

In this work, the dual-based with override method is implemented with an integral action controller in the central constraint controller, as the other two methods. From the results presented in section 5.2.2 and 5.2.3 it is clear that this method did not reach steady-state during the experimental run time. This is a consequence of the central constraint controller becoming to slow when the override constraint controller is activated. However, this could be overcome by implementing the central constraint controller as a PI-controller, rather than solely an I-controller. This will drive the system to the optimal point faster compared to the current implemented central constraint controller. In spite of that, implementing a PI-central constraint controller could be to aggressive when the override constraint controller is inactive, where the control of the

Lagrange multiplier is relatively effective with the existing implementation.

This behavior is a consequence of that the auxiliary constraint is non-linear. As it controls the difference between the override constraint controller and local gradient controller, not the difference between a given setpoint and total input, which is the case for the normal dual-based method. In nonlinear cases, gain scheduling could be an option to effectively address the non-linearity and improve controller performance.[22] This could be implemented with a switch for when the output from the override constraint controller is chosen or not in the selector. For this case it would be sensible to apply proportional gain, K_p, equal to zero when the override constraint controller is not active, and $K_P = \frac{\tau_I}{k(\tau_c+\theta)}$ when it is active. By implementing this, it would be possible to overcome the slow convergence rate in the central constraint controller while maintaining stability for all system states. This idea is currently investigated further in Dirza et al.[7].

6.1.2 Local Response Time in Dual-based Methods

As mentioned in section 5.2.2, the dual-based and dual-based with override respond faster to the local disturbance changes in the respective subsystem. In this paper, only the case with one disturbance parameter is changing at a time is considered. If instead, the case was simultaneous disturbance changes in all subsystems, which would be a more realistic case, there would still be fast response to the local disturbance changes. However, since the central constraint controller operates in the slow timescale, this could lead to significant constraint violations as the local gradient controllers deal with the local disturbances before the central controller reacts. As a result, it would be required even more back off, which would result in less profitable operation.

6.1.3 Generalizing of Primal-based Approach

The primal-based approach suggested in this work exhibits a limitation when it comes to managing constraint switching scenarios. When multiple constraints are involved, it becomes necessary to compute the local Lagrange multipliers for each individual constraint. To address this challenge, a potential solution is to employ multiple primal-based feedback-optimizing control structures in parallel, where each structure is dedicated to handling a specific constraint. A logical decision making mechanism must then be implemented to determine which constraint should be controlled based on the magnitude of the Lagrange multipliers at any given time. To ensure fairness in selecting the constraint to be considered, normalization of the Lagrange multipliers would be required. This normalization process enables the structure to decide which set of central constraint controllers should be active. However, all of these premature suggestions should proceed through further investigation before making any conclusions or recommendations.

References

[1] O. M. Aamo, G. Eikrem, H. Siahaan, and B. A. Foss. Observer design for multiphase flow in vertical pipes with gas-lift—-theory and experiments. *Journal of process control*, 15(3):247–257, 2005.

[2] V. Aas, R. Dirza, D. Krishnamoorthy, and S. Skogestad. A comparative study of distributed feedback-optimizing control strategies. 2022.

[3] R. Bitter, T. Mohiuddin, and M. Nawrocki. *LabVIEW: Advanced programming techniques*. CRC press, 2017.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[5] S. Boyd, L. Xiao, A. Mutapcic, and J. Mattingley. Notes on decomposition methods. *Notes for EE364B, Stanford University*, 635:1–36, 2007.

[6] G. Cohen and G. Coon. Theoretical consideration of retarded control. *Transactions of the American Society of Mechanical Engineers*, 75(5):827–834, 1953.

[7] R. Dirza, V. Aas, S. Skogestad, and D. Krishnamoorthy. A comparative study of distributed feedback-optimizing control schemes: An experimental validation. 2023. To appear.

[8] R. Dirza, J. Matias, S. Skogestad, and D. Krishnamoorthy. Experimental validation of distributed feedback-based real-time optimization in a gas-lifted oil well rig. *Control Engineering Practice*, 126:105253, 2022.

[9] R. Dirza, M. Rizwan, S. Skogestad, and D. Krishnamoorthy. Real-time optimal resource allocation using online primal decomposition. *IFAC-PapersOnLine*, 55(21):31–36, 2022.

[10] R. Dirza and S. Skogestad. Online feedback-based optimization with multi-input direct constraint control. *IFAC-PapersOnLine*, 55(7):149–154, 2022.

[11] R. Dirza and S. Skogestad. Systematic pairing selection for economic-oriented constraint control. In *Computer Aided Chemical Engineering*, volume 51, pages 1249–1254. Elsevier, 2022.

[12] R. Dirza, S. Skogestad, and D. Krishnamoorthy. Optimal resource allocation using distributed feedback-based real-time optimization. *IFAC-PapersOnLine*, 54(3):706–711, 2021.

[13] J. Hahn and T. Edgar. *Process Control*. 04 2014.

[14] R. A. Jose and L. H. Ungar. Pricing interprocess streams using slack auctions. *AIChE Journal*, 46(3):575–587, 2000.

[15] M. King. *Process control: a practical approach*. John Wiley & Sons, 2016.

[16] D. Krishnamoorthy. A distributed feedback-based online process optimization framework for optimal resource sharing. *Journal of Process Control*, 97:72–83, 2021.

[17] D. Krishnamoorthy, M. A. Aguiar, B. Foss, and S. Skogestad. A distributed optimization strategy for large scale oil and gas production systems. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*, pages 521–526. IEEE, 2018.

[18] D. Krishnamoorthy and S. Skogestad. Online process optimization with active constraint set changes using simple control structures. *Industrial & Engineering Chemistry Research*, 58(30):13555–13567, 2019.

[19] D. Krishnamoorthy and S. Skogestad. Systematic design of active constraint switching using selectors. *Computers & Chemical Engineering*, 143:107106, 2020.

[20] J. Matias, J. P. Oliveira, G. A. Le Roux, and J. Jäschke. Steady-state real-time optimization using transient measurements on an experimental rig. *Journal of Process Control*, 115:181–196, 2022.

[21] M. Morari, Y. Arkun, and G. Stephanopoulos. Studies in the synthesis of control structures for chemical processes: Part i: Formulation of the problem. process decomposition and the classification of the control tasks. analysis of the optimizing control structures. *AIChE Journal*, 26(2):220–232, 1980.

[22] W. J. Rugh and J. S. Shamma. Research on gain scheduling. *Automatica*, 36(10):1401–1425, 2000.

[23] D. E. Seborg, T. F. Edgar, D. A. Mellichamp, and F. J. Doyle III. *Process dynamics and control*. John Wiley & Sons, 2016.

[24] S. Skogestad. course5 - advanced, ntnu: Tkp4555. `https://folk.ntnu.no/skoge/vgprosessregulering/lectures/`. Accessed 01.06.2023.

[25] S. Skogestad. Sis7pid controller, ntnu: Tkp4140. `https://folk.ntnu.no/skoge/prosessregulering/lectures/2021/`. Accessed 07.06.2023.

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

[27] S. Skogestad. Simple analytic rules for model reduction and pid controller tuning. *Journal of process control*, 13(4):291–309, 2003.

[28] S. Skogestad and I. Postlethwaite. *Multivariable feedback control: analysis and design*. john Wiley & sons, 2005.

[29] G. Stojanovski, L. Maxeiner, S. Krämer, and S. Engell. Real-time shared resource allocation by price coordination in an integrated petrochemical site. In *2015 European Control Conference (ECC)*, pages 1498–1503. IEEE, 2015.

[30] E. Walter, L. Pronzato, and J. Norton. *Identification of parametric models from experimental data*, volume 1. Springer, 1997.

[31] J. G. Ziegler, N. B. Nichols, et al. Optimum settings for automatic controllers. *trans. ASME*, 64(11), 1942.

A Steady-State Gradient Estimation

In this paper, the estimation of gradients is performed using forward sensitivity analysis. The gradient estimation involves two steps. Firstly, the current plant information is utilized to update the state and parameters of the model using a dynamic adaptation scheme (in this work, extended Kalman filter). Secondly, the updated model is employed to compute the steady-state gradients using forward sensitivity analysis.

Remark. The system has a dynamic model which is utilized for state and parameter estimation. This model is derived by Matias et al.[20], the reader is referred to this work for further details.

A.1 Extended Kalman Filter

In order to apply the Kalman filter equations, the model needs to be linearized. The model is an index-1 differential algebraic equation (DAE) system, which easily can be transformed into an ordinary differential equation (ODE) system. the unknown parameters in the model are assumed to vary with time, and their dynamics are modeled using a random walk model,

$$\mathbf{p}^{k+1} = \mathbf{p}^k + \mathbf{v}^k \tag{A.1}$$

where $\mathbf{v}^k$ follows a normal distribution with a mean zero and a covariance V_θ. Additionally, it is assumed that $\mathbf{v}^k$ is independent of $\mathbf{v}^{\neq k}$.

By combining the dynamics of the parameters and the system an extended model for parameter estimation is obtained. As the model has been linearized, the extended Kalman filter equations can be applied to simultaneously estimate $\mathbf{p}^k$, $\mathbf{x}^k$ and $\mathbf{z}^k$. For a comprehensive derivation of the EKF equations, the reader is referred to the work of Walter and Pronzato[30].

A.2 Forward Sensitivity Analysis

The original nonlinear DAE model is expressed in the following form

$$\begin{aligned} \mathbf{x}^{k+1} &= \check{F}(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k, \mathbf{p}^k) \\ 0 &= \check{G}(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k, \mathbf{p}^k) \end{aligned} \tag{A.2}$$

The steady-state gradients are estimated by utilizing the stationary values of the forward sensitivity equation.

$$\begin{aligned} 0 =& \frac{\delta \check{F}^\top}{\delta \mathbf{x}} \mathbf{S}_{SS} + \frac{\delta \check{F}^\top}{\delta \mathbf{z}} \mathbf{R}_{SS} + \frac{\delta \check{F}^\top}{\delta \mathbf{u}} \\ 0 =& \frac{\delta \check{G}^\top}{\delta \mathbf{x}} \mathbf{S}_{SS} + \frac{\delta \check{G}^\top}{\delta \mathbf{z}} \mathbf{R}_{SS} + \frac{\delta \check{G}^\top}{\delta \mathbf{u}} \end{aligned} \tag{A.3}$$

where the sensitivities $\mathbf{S}_{SS}$ and $\mathbf{R}_{SS}$ represent the differential states $\mathbf{x}$ and algebraic states $\mathbf{z}$ with right to the inputs $\mathbf{u}$. In this scenario, the objective $\mathbf{J}$ and constraint $\mathbf{g}$

can be expressed as the linear functions of the algebraic states, $\mathbf{J} = \check{\mathbf{H}}_{\mathbf{J}} z$ and $\mathbf{g} = \check{\mathbf{H}}_{\mathbf{g}} z$. Therefore, the chain rule can be applied to compute the gradients $\nabla_{\mathbf{u}}\mathbf{J}$ and $\nabla_{\mathbf{u}}\mathbf{g}$,

$$\begin{aligned} \mathbf{J} = \check{\mathbf{H}}_{\mathbf{J}} z \Rightarrow \nabla_{\mathbf{u}}\mathbf{J} = \check{\mathbf{H}}_{\mathbf{J}}\mathbf{R}_{SS} \\ \mathbf{g} = \check{\mathbf{H}}_{\mathbf{g}} z \Rightarrow \nabla_{\mathbf{u}}\mathbf{g} = \check{\mathbf{H}}_{\mathbf{g}}\mathbf{R}_{SS} \end{aligned} \tag{A.4}$$

B MATLAB Code

B.1 LabViewMain - Primal-based

```
% Main program
% Run Initialization file first

%%%%%%%%%%%%%%%%
% Get Variables
%%%%%%%%%%%%%%%%
% disturbances
%valve opening [%]
cv101 = P_vector(1);
cv102 = P_vector(2);
cv103 = P_vector(3);
% cv101 = 0.8; cv102 = 0.4; cv103 = 0.6;
% 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;

% liquid flowrates [L/min] %INCLUDE NOISE???
fi101 = I_vector(1,dataCrop);
fi102 = I_vector(2,dataCrop);
fi103 = I_vector(3,dataCrop);

% actual gas flowrates [sL/min] %INCLUDE NOISE???
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);

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

% we do no consider SS detection here
flagSS = 1;

% check for first iteration
if ~exist('dxHatkk','var')
    %initial condition
    [dx0,z0,u0,theta0] = InitialConditionGasLift(par);

    dxHatkk = dx0;
    zHatkk = z0;
    thetakk = theta0;

    load('EKFconf');
    qThres = ekf.qThresk;
    Pkk = ekf.Pkk;
    %ekf.R = noise.output; %added (different from Rig Implementation)

    lambda = 9.5;

    lambda_1 = 1;
    lambda_2 = 1;
    lambda_3 = 1;

    err0 = zeros(3,1);
end
```

```
if flagSS == 1
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Estimating SS model parameters (dynamic) %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    yPlant = [fi101;
              fi102;
              fi103;
              1.01325 + 10^-3*pi101; %[mbarg]-->[bar a];
              1.01325 + 10^-3*pi102;
              1.01325 + 10^-3*pi103];

    uPlant = [fic104; %conversion [L/min] --> [kg/s]
              fic105;
              fic106;
              cv101*ones(1,dss); %workaround - i just have the last measurement here. Since it i
              cv102*ones(1,dss);
              cv103*ones(1,dss);
              pi104 + 1.01325];

          try
              % running casadi - getting the last measurement and last
              % input that generated that measuremnt
              [dxHatkk,zHatkk,thetakk,Pkk,qThres] = RExtendedKalmanFilter(yPlant(:,end),uPlant(:

              % everything normal
              flagEst = 1;
          catch
              warning('Dynamic Estimation Problem!');
              beep
              % estimation problem
              flagEst = 0;

              % Note that in this case, we dont update
              % dxHatkk,zHatkk,thetakk,Pkk,qThres
          end


    if flagEst == 1
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Optimizing Systems %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % Gradient Estimation
        %conversion
        CR = 60*10^3; % [L/min] -> [m3/s]

        %%% NEED TO CHECK
        F_zp = full(S_zp(dxHatkk,zHatkk,[uPlant(:,end);thetakk;1e4]));

        %F_zp = full(S_zp(dxHatkk,zHatkk,[uSPPlantArray(1:3,end);uPlant(4:end,end);thetakk;1e4])

        J_qgl1 = - 20*((1*1e-2)*CR/par.rho_o(1))*F_zp(22,1);
        J_qgl2 = - 25*((1*1e-2)*CR/par.rho_o(2))*F_zp(23,2);
```

```
J_qgl3 = - 30*((1*1e-2)*CR/par.rho_o(3))*F_zp(24,3);

g_qgl1 = 1;
g_qgl2 = 1;
g_qgl3 = 1;

%------- Calculating lambda's -------
lambda_1 = -(J_qgl1)/g_qgl1;
lambda_2 = -(J_qgl2)/g_qgl2;
lambda_3 = -(J_qgl3)/g_qgl3;

lambda = [lambda_1 lambda_2 lambda_3];

%------ g_i --------
%choose compensator
well_comp = 3;
lambda_c = lambda(well_comp);

%------  Controller Tunings  --------
%---------------  1  ----------------

K_1 = -2.522;
tauC_1 = 25;
theta_1 = 0;

Ki1 = 1/(K_1*(tauC_1 + theta_1));

%---------------  2  ----------------

K_2 = -2.918;
tauC_2 = 25;
theta_2 = 0;

Ki2 = 1/(K_2*(tauC_2 + theta_2));

%---------------  3  ----------------

K_3 = -3.698;
tauC_3 = 25;
theta_3 = 0;

Ki3 = 1/(K_3*(tauC_3 + theta_3));

%----- Primal Decomposition Controller -----

if well_comp == 1
    %----- Subsys 1 Compensator -----
    g2new = u0old(2) + Ki2*(-lambda_2 + lambda_c);
    uOpt(2,1) = max(1.0,min(7.5, g2new));
    g3new = u0old(3) + Ki3*(-lambda_3 + lambda_c);
    uOpt(3,1) = max(1.0,min(7.5, g3new));
    g1new = 7.5 - (uOpt(2,1) + uOpt(3,1));
    uOpt(1,1) = max(1.0,min(7.5, g1new));
end
```

```
if well_comp == 2
    %----- Subsys 2 Compensator -----
    g1new = u0old(1) + Ki1*(-lambda_1 + lambda_c);
    uOpt(1,1) = max(1.0,min(7.5, g1new));
    g3new = u0old(3) + Ki3*(-lambda_3 + lambda_c);
    uOpt(3,1) = max(1.0,min(7.5, g3new));
    g2new = 7.5 - (uOpt(1,1) + uOpt(3,1));
    uOpt(2,1) = max(1.0,min(7.5, g2new));
end
if well_comp == 3
    %----- Subsys 3 Compensator -----
    g1new = u0old(1) + Ki1*(-lambda_1 + lambda_c);
    uOpt(1,1) = max(1.0,min(7.5, g1new));
    g2new = u0old(2) + Ki2*(-lambda_2 + lambda_c);
    uOpt(2,1) = max(1.0,min(7.5, g2new));
    g3new = 7.5 - (uOpt(1,1) + uOpt(2,1));
    uOpt(3,1) = max(1.0,min(7.5, g3new));
end


flagOpt = 1;

if flagOpt == 1
    %%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Send Variable
    %%%%%%%%%%%%%%%%%%%%%%%%%%%
    % compute new values for the gas flow rate setpoints
    %Filter for optimum
    fic104sp = uOpt(1);
    fic105sp = uOpt(2);
    fic106sp = uOpt(3);

    O_vector = vertcat(fic104sp,fic105sp,fic106sp)';
    SS = 1;
    Estimation = 1;
    Optimization = 1;
    Result = lambda;
    Parameter_Estimation = thetakk';
    State_Variables_Estimation = (par.H*zHatkk)';
    State_Variables_Optimization = ([J_qgl1;J_qgl2;J_qgl3;L_qgl1;L_qgl2;L_qgl3;])';
    Optimized_Air_Injection = uOpt';
else
    %%%%%%%%%
    %(dummy)%
    %%%%%%%%%
    % compute new values for the gas flow rate setpoints
    O_vector = vertcat(fic104sp,fic105sp,fic106sp)';

    SS = 1;
    Estimation = 1;
    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 = [0,0,0];
        end

    else
        %%%%%%%%%
        %(dummy)%
        %%%%%%%%%
        % compute new values for the gas flow rate setpoints
        O_vector = vertcat(fic104sp,fic105sp,fic106sp)';

        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 = [0,0,0];
    end


else
    %%%%%%%%%
    %(dummy)%
    %%%%%%%%%
    % compute new values for the gas flow rate setpoints
    O_vector = vertcat(fic104sp,fic105sp,fic106sp)';

    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 = [0,0,0];

end
```

B.2 LabViewMain - Dual-based

```
% Main program
% Run Initialization file first

%%%%%%%%%%%%%%%%
% 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;

% liquid flowrates [L/min] %INCLUDE NOISE???
fi101 = I_vector(1,dataCrop);
fi102 = I_vector(2,dataCrop);
fi103 = I_vector(3,dataCrop);

% actual gas flowrates [sL/min] %INCLUDE NOISE???
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);

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

% we do no consider SS detection here
flagSS = 1;

% check for first iteration
if ~exist('dxHatkk','var')
    %initial condition
    [dx0,z0,u0,theta0] = InitialConditionGasLift(par);

    dxHatkk = dx0;
    zHatkk = z0;
    thetakk = theta0;

    load('EKFconf');
    qThres = ekf.qThresk;
    Pkk = ekf.Pkk;
    %ekf.R = noise.output; %added (different from Rig Implementation)

    lambda = 9.5;
    err0 = zeros(3,1);
end

if flagSS == 1
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Estimating SS model parameters (dynamic) %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    yPlant = [fi101;
              fi102;
              fi103;
              1.01325 + 10^-3*pi101; %[mbarg]-->[bar a];
              1.01325 + 10^-3*pi102;
              1.01325 + 10^-3*pi103];

    uPlant = [fic104; %conversion [L/min] --> [kg/s]
              fic105;
              fic106;
              cv101*ones(1,dss); %workaround - i just have the last
              cv102*ones(1,dss); %measurement here. Since it is the
              cv103*ones(1,dss); %disturbance, it doesn't really matter;
              pi104 + 1.01325];

          try
```

```
            % running casadi - getting the last measurement and last
            % input that generated that measuremnt
            [dxHatkk,zHatkk,thetakk,Pkk,qThres] = RExtendedKalmanFilter(yPlant(:,end),uPlant(:

            % everything normal
            flagEst = 1;
        catch
            warning('Dynamic Estimation Problem!');
            beep
            % estimation problem
            flagEst = 0;

            % Note that in this case, we dont update
            % dxHatkk,zHatkk,thetakk,Pkk,qThres
        end


    if flagEst == 1
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Optimizing Systems %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        %---------------------------------
        %--- Central Constraint Control --
        %---------------------------------
        %---      Update Lambda        ---
        %---------------------------------

        Total_Qgl = fic104(end) + fic105(end) + fic106(end);

        K_ccc = 0.907;
        tau_ccc = 125;
        theta_ccc = 0;
        KI_ccc = 1/(K_ccc*(tau_ccc+theta_ccc));

        lambda_next =  max(0,lambda + KI_ccc*(Total_Qgl   - 7.5));

        lambda = lambda_next;

        %---------------------------------
        %---     Gradient Estimation    ---
        %---------------------------------
        %conversion
        CR = 60*10^3; % [L/min] -> [m3/s]

        F_zp = full(S_zp(dxHatkk,zHatkk,[uPlant(:,end);thetakk;1e4]));

        J_qgl1 = - 20*((1*1e-2)*CR/par.rho_o(1))*F_zp(22,1);
        J_qgl2 = - 25*((1*1e-2)*CR/par.rho_o(2))*F_zp(23,2);
        J_qgl3 = - 30*((1*1e-2)*CR/par.rho_o(3))*F_zp(24,3);

        g_qgl1 = 1;
        g_qgl2 = 1;
        g_qgl3 = 1;
```

```
L_qgl1 = J_qgl1 + lambda.*g_qgl1;
L_qgl2 = J_qgl2 + lambda.*g_qgl2;
L_qgl3 = J_qgl3 + lambda.*g_qgl3;


%---------------------------------
%---  Local Gradient Control   ---
%---------------------------------

% --------- Well 1 ---------
K_1 = 2.53;
tauC_1 = 25;
theta_1 = 0;

Ki1 = 1/(K_1*tauC_1 + theta_1);
err_1= -L_qgl1;

uOpt(1,1) = max(1,min(5.5, u0old(1) + (Ki1*err_1)));

err0(1) = err_1;

% --------- Well 2 ---------
K_2 = 2.93;
tauC_2 = 25;
theta_2 = 0;

Ki2 = 1/(K_2*tauC_2 + theta_2);
err_2 = -L_qgl2;

uOpt(2,1) = max(1,min(5.5, u0old(2) + (Ki2*err_2)));

err0(2) = err_2;

% --------- Well 3 ---------
K_3 = 3.70;
tauC_3 = 25;
theta_3 = 0;

Ki3 = 1/(K_3*tauC_3 + theta_3);
err_3= -L_qgl3;

uOpt(3,1) = max(1,min(5.5, u0old(3) + (Ki3*err_3)));

err0(3) = err_3;

flagOpt = 1;

if flagOpt == 1
    %%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Send Variable
    %%%%%%%%%%%%%%%%%%%%%%%%%%%
    % compute new values for the gas flow rate setpoints
    %Filter for optimum
```

```
            fic104sp = uOpt(1);
            fic105sp = uOpt(2);
            fic106sp = uOpt(3);

            O_vector = vertcat(fic104sp,fic105sp,fic106sp)';
            SS = 1;
            Estimation = 1;
            Optimization = 1;
            Result = lambda;
            Parameter_Estimation = thetakk';
            State_Variables_Estimation = (par.H*zHatkk)';
            State_Variables_Optimization = ([J_qgl1;J_qgl2;J_qgl3;L_qgl1;L_qgl2;L_qgl3;])';
            Optimized_Air_Injection = uOpt';
        else
            %%%%%%%%%
            %(dummy)%
            %%%%%%%%%
            % compute new values for the gas flow rate setpoints
            O_vector = vertcat(fic104sp,fic105sp,fic106sp)';

            SS = 1;
            Estimation = 1;
            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 = [0,0,0];
        end

    else
        %%%%%%%%%
        %(dummy)%
        %%%%%%%%%
        % compute new values for the gas flow rate setpoints
        O_vector = vertcat(fic104sp,fic105sp,fic106sp)';

        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 = [0,0,0];
    end


else
    %%%%%%%%%
    %(dummy)%
    %%%%%%%%%
    % compute new values for the gas flow rate setpoints
    O_vector = vertcat(fic104sp,fic105sp,fic106sp)';
    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 = [0,0,0];

end
```

B.3 LabViewMain - Dual-based with Override

```
% Main program
% Run Initialization file first

%%%%%%%%%%%%%%%%
% 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;

% liquid flowrates [L/min] %INCLUDE NOISE???
fi101 = I_vector(1,dataCrop);
fi102 = I_vector(2,dataCrop);
fi103 = I_vector(3,dataCrop);
```

```
% actual gas flowrates [sL/min] %INCLUDE NOISE???
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);

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

% we do no consider SS detection here
flagSS = 1;

% check for first iteration
if ~exist('dxHatkk','var')
    %initial condition
    [dx0,z0,u0,theta0] = InitialConditionGasLift(par);

    dxHatkk = dx0;
    zHatkk = z0;
    thetakk = theta0;

    load('EKFconf');
    qThres = ekf.qThresk;
    Pkk = ekf.Pkk;
    %ekf.R = noise.output; %added (different from Rig Implementation)

    lambda = 9.5;
    lambdaOrg = 9.5;
    lambda_hat = 9.5;
    Total_uOpt = 7.5;
    udirtot = 7.5;
    uOpt = u0old;
    uDir = u0old;

    err0 = zeros(3,1);
end

if flagSS == 1
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Estimating SS model parameters (dynamic) %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    yPlant = [fi101;
              fi102;
              fi103;
              1.01325 + 10^-3*pi101; %[mbarg]-->[bar a];
              1.01325 + 10^-3*pi102;
              1.01325 + 10^-3*pi103];

    uPlant = [fic104; %conversion [L/min] --> [kg/s]
              fic105;
              fic106;
              cv101*ones(1,dss); %workaround - i just have the last measurement here. Since it i
              cv102*ones(1,dss);
              cv103*ones(1,dss);
              pi104 + 1.01325];

          try
              % running casadi - getting the last measurement and last
              % input that generated that measuremnt
              [dxHatkk,zHatkk,thetakk,Pkk,qThres] = RExtendedKalmanFilter(yPlant(:,end),uPlant(:

              % everything normal
              flagEst = 1;
          catch
              warning('Dynamic Estimation Problem!');
              beep
              % estimation problem
              flagEst = 0;

              % Note that in this case, we dont update
              % dxHatkk,zHatkk,thetakk,Pkk,qThres
          end


    if flagEst == 1
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Optimizing Systems %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% MV Selector Matrix
Hc = [0; 0; 1]';

%---------------------------------
%--- Central Constraint Control --
%---------------------------------
%---      Update Lambda         ---
%---------------------------------

Total_Qgl = fic104(end) + fic105(end) + fic106(end);

K_ccc = 0.2171164;
tau_ccc = 125;
theta_ccc = 0;

KI_ccc = 1/(K_ccc*(tau_ccc+theta_ccc));

lambda_next =  lambda + KI_ccc*(Hc*uOpt - Hc*uDir);

lambda = max(0,lambda_next);


%---------------------------------
%---    Gradient Estimation    ---
%---------------------------------
%conversion
CR = 60*10^3; % [L/min] -> [m3/s]

%%% NEED TO CHECK
F_zp = full(S_zp(dxHatkk,zHatkk,[uPlant(:,end);thetakk;1e4]));

J_qgl1 = - 20*((1*1e-2)*CR/par.rho_o(1))*F_zp(22,1);
J_qgl2 = - 25*((1*1e-2)*CR/par.rho_o(2))*F_zp(23,2);
J_qgl3 = - 30*((1*1e-2)*CR/par.rho_o(3))*F_zp(24,3);

g_qgl1 = 1;
g_qgl2 = 1;
g_qgl3 = 1;

L_qgl1 = J_qgl1 + lambda.*g_qgl1;
L_qgl2 = J_qgl2 + lambda.*g_qgl2;
L_qgl3 = J_qgl3 + lambda.*g_qgl3;


% --------------------------------- %
% ----- Direct Const. Control ----- %
% --------------------------------- %
taui_dc = 2.5;
K_dir = 1;
tauc_dc = 10;

KI_dir = 1/(K_dir*tauc_dc);
KP_dir = taui_dc/((K_dir*tauc_dc));
```

```
K_aw = 1*(KI_dir)/KP_dir;


udir1 = max(1,min(5.5, uDir(1) + Hc(1)*KI_dir*(7.5 - Total_Qgl)+ K_aw*(u0old(1)-uDir(1))
udir2 = max(1,min(5.5, uDir(2) + Hc(2)*KI_dir*(7.5 - Total_Qgl)+ K_aw*(u0old(2)-uDir(2))
udir3 = max(1,min(5.5, uDir(3) + Hc(3)*KI_dir*(7.5 - Total_Qgl)+ K_aw*(u0old(3)-uDir(3))

uDir = [udir1; udir2; udir3];

% --------------------------------- %
% -- Local Gradient Controllers --- %
% --------------------------------- %

% ------------ Well 1 ------------- %

K_1 = 2.53;
tauC_1 = 25;
theta_1 = 0;

Ki1 = 1/(K_1*tauC_1 + theta_1);

err_1= -L_qgl1;

uOpt(1,1) = max(1,min(5.5, u0old(1) + (Ki1*err_1)));

err0(1) = err_1;

% ------------ Well 2 ------------- %

K_2 = 2.93;
tauC_2 = 25;
theta_2 = 0;

Ki2 = 1/(K_2*tauC_2 + theta_2);

err_2 = -L_qgl2;

uOpt(2,1) = max(1,min(5.5, u0old(2) + (Ki2*err_2)));

err0(2) = err_2;

% ------------ Well 3 ------------- %

K_3 = 3.70;
tauC_3 = 25;
theta_3 = 0;

Ki3 = 1/(K_3*tauC_3 + theta_3);

err_3= -L_qgl3;

uOpt(3,1) = max(1,min(5.5, u0old(3) + (Ki3*err_3)));

err0(3) = err_3;
```

```
        flagOpt = 1;

        Total_uOpt = uOpt(1) + uOpt(2) + uOpt(3);

        if flagOpt == 1
            %%%%%%%%%%%%%%%%%%%%%%%%%%%
            %Send Variable
            %%%%%%%%%%%%%%%%%%%%%%%%%%%
            % compute new values for the gas flow rate setpoints
            %Filter for optimum
            fic104sp = uOpt(1);%min(uOpt(1),uDir(1));%
            fic105sp = uOpt(2);%min(uOpt(2),uDir(2));%
            fic106sp = min(uOpt(3),uDir(3));%uOpt(3);%

            O_vector = vertcat(fic104sp,fic105sp,fic106sp)';
            SS = 1;
            Estimation = 1;
            Optimization = 1;
            Result = lambda;
            Parameter_Estimation = thetakk';
            State_Variables_Estimation = (par.H*zHatkk)';
            State_Variables_Optimization = ([J_qgl1;J_qgl2;J_qgl3;L_qgl1;L_qgl2;L_qgl3;])';
            Optimized_Air_Injection = uOpt';
        else
            %%%%%%%%%
            %(dummy)%
            %%%%%%%%%
            % compute new values for the gas flow rate setpoints
            O_vector = vertcat(fic104sp,fic105sp,fic106sp)';

            SS = 1;
            Estimation = 1;
            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 = [0,0,0];
        end

    else
        %%%%%%%%%
        %(dummy)%
        %%%%%%%%%
        % compute new values for the gas flow rate setpoints
        O_vector = vertcat(fic104sp,fic105sp,fic106sp)';

        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 = [0,0,0];
    end


else
    %%%%%%%%%
    %(dummy)%
    %%%%%%%%%
    % compute new values for the gas flow rate setpoints
    O_vector = vertcat(fic104sp,fic105sp,fic106sp)';

    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 = [0,0,0];

end
```