Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Simen Bergsvik

Enhancing Model Predictive Control for Non-Continuous Actuators in the Oil and Gas Industry: A Mixed Integer MPC Approach

Master's thesis in Cybernetics and Robotics
Supervisor: Lars Struen Imsland
June 2023

Simen Bergsvik

Enhancing Model Predictive Control for Non-Continuous Actuators in the Oil and Gas Industry: A Mixed Integer MPC Approach

Master's thesis in Cybernetics and Robotics
Supervisor: Lars Struen Imsland
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Department of Engineering Cybernetics

Master Thesis

Enhancing Model Predictive Control for Non-Continuous Actuators in the Oil and Gas Industry: A Mixed Integer MPC Approach

Author:
Simen Bergsvik

Supervisor NTNU:
Lars Struen Imsland

Supervisor Equinor:
John-Morten Godhavn

June, 2023

Abstract

Effective control systems play a critical role in achieving high productivity and increased earnings in the oil and gas industry. Model Predictive Control (MPC) has emerged as a widely used technique for control in this industry. Equinor has developed SEPTIC, an in-house software tool for model predictive control, which has demonstrated favorable business cases since its inception in 1997. However, traditional continuous MPC approaches, including SEPTIC, are limited to continuous decision variables in the optimization problem. As a result, these approaches encounter challenges when dealing with non-continuous actuators like step chokes, commonly found in the oil and gas sector.

Equinor currently tackles the discreteness of these actuators through external logic, heuristics, and special rules. Even though SEPTIC provides excellent performance, not accounting for the discrete actuators in the optimization problem leads to suboptimal performance. This thesis proposes Mixed Integer Model Predictive Control (MIMPC) to address the limitations of SEPTIC and similar systems in managing non-continuous actuators. Unlike continuous MPCs, which treat non-continuous actuators as continuous, MIMPC directly incorporates the discrete nature of the actuators into the optimization problem.

The primary objective of this thesis has been to address the discreteness of a production choke in a gas-lifted oil well by directly incorporating it into the optimization problem within the MPC framework. This has been accomplished by introducing integer decision variables into the MPC optimization problem and leveraging mixed integer programming techniques to obtain an optimal solution. By utilizing a solver that supports integer programming, the inclusion of integrality constraints has been seamlessly integrated.

When using continuous MPC for controlling the discrete choke, the actual choke position is determined by rounding the desired continuous choke position to the nearest integer using a deadband. However, this study demonstrates that utilizing rounding techniques in continuous optimization to obtain an integer solution does not necessarily provide an optimal outcome. Instead, the utilization of integer programming has been emphasized as the preferred approach to achieve an optimal solution in this context.

The continuous MPC approach gives rise to unpredictable responses when rounding the desired choke position to the nearest integer upon surpassing the deadband. The research findings highlight that these unanticipated movements result in significant overshoots in the controlled variables, compelling the MPC to readjust the choke back towards its previous position. This iterative process of rounding and readjustment leads to repeatedly large and unanticipated movements, causing oscillations in the system due to continuous control of a discrete actuator. It is crucial to acknowledge that these oscillations have a detrimental effect on the actuator, significantly reducing its lifespan.

The integration of integer decision variables into the optimization problem extends the continuous MPC framework to a MIMPC approach. This enables the controller to consider the discrete nature of the choke explicitly. The findings presented in this thesis provide compelling evidence that MIMPC successfully addresses the challenges associated with controlling a discrete actuator, eliminating the oscillations observed when using continuous control methods. Through the utilization of MIMPC, the controller gains knowledge of the discrete behavior of the choke, enabling more accurate predictions. As a result, the system exhibits enhanced stability and improved overall performance.

While the implementation of MIMPC effectively eliminates the oscillations resulting from continuous control of a discrete actuator, it is worth noting that some occasional oscillations could still be observed. These oscillations can be attributed to the highly nonlinear nature of the system being controlled. The MPC formulation employed in this study relies on a linear step response model for predictions, which introduces challenges associated with plant model mismatch. This linear approximation fails to capture the nonlinear dynamics of the system accurately. The investigation conducted in this thesis highlights that significant discrepancies between the linear model and the true nonlinear behavior can give rise to undesired oscillations in the system.

This study employs the Soft MPC method to mitigate the oscillatory behavior caused by plant model mismatch. By introducing a deadzone around the setpoints, the Soft MPC approach reduces the penalties for deviations from the desired values when the controlled variables are in close proximity to their setpoints. The findings of this research demonstrate the effectiveness of the Soft MPC method in mitigating oscillations resulting from plant model mismatch. By introducing a more flexible region around the setpoints, the Soft MPC method enhances the stability and robustness of the control system, leading to improved control performance. The use of Soft MPC provides a valuable approach for managing the challenges associated with plant model mismatch and contributes to more precise and reliable control of the system.

Solving integer optimization programs is computationally more complex than solving continuous optimization problems. To address this issue, input blocking is employed to reduce the dimensionality of the MPC problem. By selectively blocking the manipulated variables, the complexity of the optimization problem is reduced, enabling more efficient and faster computation.

In addition, bias filtering is incorporated to eliminate sudden spikes in feedback bias. These spikes can have a detrimental impact on control performance, leading to undesired oscillations. Moreover, mean choke move constraints are integrated into the MPC formulation. These constraints limit the average movement of the choke and mitigate the risks associated with sand production in the well.

To evaluate the efficacy of the proposed MIMPC approach in addressing the challenges posed by non-continuous actuators, comprehensive analysis and comparison with the traditional continuous MPC technique are conducted. The study presents the performance of the MIMPC, highlighting the benefits of incorporating integer decision variables and implementing the Soft MPC method.

The key findings of the study underscore that leveraging MIMPC with an understanding of the discreteness of the actuator yields superior performance compared to treating the discrete choke as continuous and utilizing rounding techniques to obtain an integer solution. The MIMPC approach effectively eliminates the observed oscillations and enhances control precision by accounting for the discrete nature of the actuator in the optimization problem. The findings also highlight the additional computational complexity of including integer variables in the optimization problem. This aspect should be carefully considered and weighed against the benefits of improved control performance. A thorough assessment of computational resources and time constraints is necessary to ensure practical feasibility. These results provide valuable insights into the potential application of MIMPC in the oil and gas industry, offering a promising solution for improving control strategies and maximizing operational efficiency.

Sammendrag

Effektive kontrollsystemer er avgjørende for å oppnå høy produktivitet og økte inntekter innen olje- og gassindustrien. Modellbasert prediktiv kontroll (*Model Predictive Control*, MPC) har blitt en mye brukt teknikk for kontroll i denne bransjen. Equinor har utviklet SEPTIC, et internt programvareverktøy for modellbasert prediktiv kontroll, som har vist gunstige resultater siden oppstarten i 1997. Imidlertid er tradisjonelle kontinuerlige MPC-tilnærminger, inkludert SEPTIC, begrenset til kontinuerlige beslutningsvariabler i optimeringsproblemet. Dette fører til utfordringer knyttet til ikke-kontinuerlige aktuatorer, som f.eks. ventiler som åpnes stegvis. Dette er mye brukt i olje- og gassektoren.

For øyeblikket håndterer Equinor diskretheten til disse aktuatorene gjennom ekstern logikk, heuristikker og spesielle regler. Selv om SEPTIC gir utmerket ytelse, fører manglende hensyn til de diskrete aktuatorene i optimeringsproblemet til suboptimal ytelse. Denne oppgaven foreslår modellbasert prediktiv kontroll med blandet heltall (*Mixed Integer Model Predictive Control*, MIMPC) for å håndtere begrensningene til SEPTIC og lignende systemer når det gjelder styring av ikke-kontinuerlige aktuatorer. I motsetning til kontinuerlige MPC-er, som behandler ikke-kontinuerlige aktuatorer som kontinuerlige, inkorporerer MIMPC den stegvise oppførselen til aktuatorene direkte i optimeringsproblemet.

Hovedmålet med denne oppgaven har vært å håndtere den stegvise oppførselen til en produksjonschoke i en gassløftet oljebrønn, direkte i optimeringsproblemet i en MPC. Dette er oppnådd ved å introdusere heltallsbeslutningsvariabler i MPC-optimeringsproblemet og bruke heltallsprogrammering for å finne en optimal løsning. Ved å bruke en løser som støtter heltallsprogrammering, blir inkluderingen av heltallrestriksjoner enkel.

Når kontinuerlig MPC brukes til å kontrollere den diskrete choken, bestemmes den faktiske chokeposisjonen ved å avrunde den ønskede kontinuerlige choke-posisjonen til nærmeste heltall ved hjelp av et dødbånd. Imidlertid viser denne studien at bruk av avrundingsmetoder i kontinuerlig optimering for å oppnå en heltallsløsning ikke nødvendigvis gir et optimalt resultat. I stedet bør heltallsprogrammering benyttes for å oppnå en optimal løsning.

Ettersom den kontinuerlig MPCen behandler den diskrete aktuatoren som kontinuerlig, oppstår uforutsigbare responser når ønsket chokeposisjon overstiger dødbåndet og avrundes til nærmeste heltall. Resultatene viser at disse uforutsette bevegelsene resulterer i mye større responser enn hva som er forventet av MPCen, noe som krever at MPCen må flytte choken tilbake mot forrige posisjon. Ved neste avrunding skjer det samme, en ny stor og uforutsett respons.

Denne iterative prosessen med avrunding og justering fører til gjentatte store og uforutsette bevegelser, noe som forårsaker oscillasjoner i systemet på grunn av kontinuerlig styring av en diskret aktuator. Det er viktig å merke seg at disse oscillasjonene er svært skadelige for aktuatoren og betydelig reduserer dens levetid.

Integrasjonen av heltallsbeslutningsvariabler i optimeringsproblemet utvider det kontinuerlige MPC-rammeverket til en MIMPC-tilnærming. Dette gjør det mulig for kontrolleren å eksplisitt ta hensyn til den diskrete naturen til choken. Resultatene viser at MIMPC med suksess håndterer utfordringene knyttet til styring av en diskret aktuator og eliminerer oscillasjonene som observeres ved bruk av kontinuerlige styringsmetoder. Ved å benytte MIMPC får kontrolleren kunnskap om den diskrete oppførselen til choken, noe som muliggjør mer nøyaktige prediksjoner. Som et resultat viser systemet forbedret stabilitet og forbedret totalytelse.

Selv om implementeringen av MIMPC effektivt eliminerer oscillasjoner som oppstår ved kontinuerlig styring av en diskret aktuator, er det verdt å merke seg at det fortsatt kan observeres noen oscillasjoner. Disse oscillasjonene kommer av den ikke-lineære oppførselen til det kontrollerte systemet. MPC-formuleringen som brukes i denne oppgaven, er basert på en lineær stegresponsmodell for prediksjoner, noe som fører til utfordringer knyttet til avvik mellom modellen og selve systemet. Denne lineære tilnærmingen klarer ikke å fange opp de ikke-lineære dynamikkene til systemet. Resultatene i oppgaven påpeker at betydelige avvik mellom den lineære modellen og den virkelige ikke-lineære atferden kan gi uønskede oscillasjoner i systemet.

For å redusere den oscillerende oppførselen til systemet som oppstår grunnet avvik mellom prediksjonsmodellen og det faktiske systemet, er Soft MPC metoden brukt i implementasjonen. Denne introduserer en dødsone rundt settpunktene, hvor straffen for avvik innenfor denne dødsonen reduseres. Ved å introdusere en mer fleksibel sone rundt settpunktene, forbedrer Soft MPC-metoden stabiliteten og robustheten til kontrollsystemet, noe som fører til forbedret kontrollprestasjon. Bruken av Soft MPC gir en verdifull tilnærming for å håndtere utfordringene forbundet med avvik mellom prediksjonsmodellen og det faktiske systemet, og bidrar til mer presis og pålitelig styring av systemet.

Løsning av heltallsoptimeringsproblemer er mye mer komplekst enn løsning av kontinuerlige optimeringsproblemer. For å håndtere dette problemet blir pådrags-blokking brukt for å redusere dimensjonaliteten til MPC-problemet. Ved å selektivt velge størrelsen på blokkene på de manipulerte variablene reduseres kompleksiteten i optimeringsproblemet, noe som muliggjør mer effektiv og raskere beregning.

I tillegg blir biasfiltrering implementert for å eliminere plutselige utslag i biasen. Disse utslagene kan ha en skadelig effekt på kontrollprestasjonen og føre til uønskede oscillasjoner. Videre er det integrert begrensninger for gjennomsnittlig choke-bevegelse i MPC-formuleringen. Disse begrensningene begrenser den gjennomsnittlig bevegelse av choken og demper risikoen forbundet med sandproduksjon i brønnen.

For å evaluere effektiviteten av den foreslåtte MIMPC-tilnærmingen for å håndtere utfordringene med ikke-kontinuerlige aktuatorer, blir grundige analyser og sammenligninger med den tradisjonelle kontinuerlige MPC-teknikken gjennomført. Studien presenterer ytelsen til MIMPC og fremhever fordelene med å inkorporere heltallsbeslutningsvariabler og implementere Soft MPC-metoden.

De viktigste funnene i studien understreker at bruk av MIMPC, med kunnskap om diskretheten til aktuatoren, gir bedre ytelse sammenlignet med å behandle den diskrete choken som kontinuerlig og bruke avrundingsmetoder for å oppnå en heltallsløsning. MIMPC-tilnærmingen eliminerer effektivt oscillasjonene som observeres og forbedrer kontrollpresisjonen ved å ta hensyn til den stegvise oppførselen til aktuatoren i optimeringsproblemet. Funnene fremhever også den økte beregningskompleksiteten ved inkludering av heltallsvariabler i optimeringsproblemet. Denne faktoren bør vurderes nøye og veies opp mot fordelene med forbedret kontrollprestasjon. En grundig vurdering av beregningsressurser og tidsbegrensninger er nødvendig for å sikre praktisk gjennomførbarhet. Disse resultatene gir verdifulle innsikter i den potensielle bruken av MIMPC i olje- og gassindustrien, og tilbyr en lovende løsning for å forbedre kontrollstrategier og maksimere effektivitet.

Preface

This thesis is written as the product of the finalization of my M.Sc degree in Cybernetics and Robotics at the Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU). It is with great pleasure and a sense of accomplishment that I present this work.

The past five months dedicated to this thesis have been immensely fulfilling. Diving deeper into the realm of Mixed Integer Model Predictive Control and delving into the intricacies of incorporating the inherent discreteness of a step choke of an oil well into an MPC problem has been an enlightening journey. This study builds upon the foundations established in my previous project thesis, [1], expanding upon the earlier work.

I want to express my sincere gratitude to my supervisor, Prof. Lars Struen Imsland, for his invaluable guidance and support throughout the course of this thesis. Without his help, this work would not have been possible. I would also like to thank my supervisor at Equinor, John-Morten Godhavn, for his direction and assistance.

[illegible]

Simen Bergsvik

Table of Contents

List of Figures

List of Tables

1 Introduction

Model Predictive Control (MPC) is a popular control strategy that can employ multivariable process models to forecast future system and control behavior. When the process model is nonlinear, the MPC scheme is known as nonlinear MPC (NMPC). Although NMPC is gaining more attention from both researchers and industrial practitioners, the most commonly used implementation of MPC relies on linear continuous models. Therefore, references to MPC usually imply linear continuous MPC, and the same applies to the use of MPC in this thesis.

The MPC framework allows for several control objectives, including constraints, to be explicitly specified and handled systematically. MPC aims to predict and optimize system behavior while considering constraints by repeatedly solving an optimization problem. This iterative optimization process allows for operations closer to constraints, often leading to potential long-term profitability gains, particularly in process industry applications [8]. MPC has been highly successful in various industries, including process control, robotics, and aerospace, due to its ability to handle complex control problems with multiple constraints.

1.1 Motivation & Background

Since 1996, Equinor has developed an in-house software tool for Model Predictive Control, known as SEPTIC - *Statoil Estimation and Prediction Tool for Identification and Control* [9]. The first installation of SEPTIC was done in 1997, and there were approximately 100 SEPTIC MPC applications in Equinor as of 2019. SEPTIC is used both upstream and downstream for a wide range of processes, ranging from production well control to gasoline blending. Business cases are generally very good – for example, in 2019 the Mongstad Refinery reported an incentive of 500 MNOK/year, and offshore activities show similar numbers.

Figure 1 illustrates a simplified gas-lifted production well that is commonly found on Equinor's offshore production sites. Oil and gas production is controlled through the position of the production choke and the gas-lift choke, with SEPTIC being used to achieve precise and robust control. However, while SEPTIC is an effective MPC software, it has some limitations. One of these limitations is its inability to handle discrete input variables such as non-continuous actuators within the optimization problem. Typically, actuators such as the production choke depicted in Figure 1 are non-continuous. To work around this limitation, SEPTIC optimizes these variables as if the actuator were continuous, and deals with the discrete limitations outside of the optimization problem through various methods such as hysteresis, deadband, and special rules. This approach allows SEPTIC to achieve great overall performance. However, even though the performance of SEPTIC is great, not accounting for the discrete actuators in the optimization problem leads to suboptimal performance. This limitation is the main motivation for this thesis.

To address the limitations of SEPTIC in handling non-continuous actuators, a possibly more effective solution can be implemented through the use of *Mixed Integer Model Predictive Control* (MIMPC). Unlike traditional continuous MPC, MIMPC can account for the discreteness of the actuators directly in the optimization problem, leading to optimal performance. By including discrete variables in the optimization problem, the MIMPC gains knowledge of the discrete nature of the actuator. With this knowledge in the control system, the system performance can be improved, potentially leading to gains in productivity and profitability. Overall, MIMPC offers a more advanced and potentially superior solution to handle the challenges posed by non-continuous actuators.

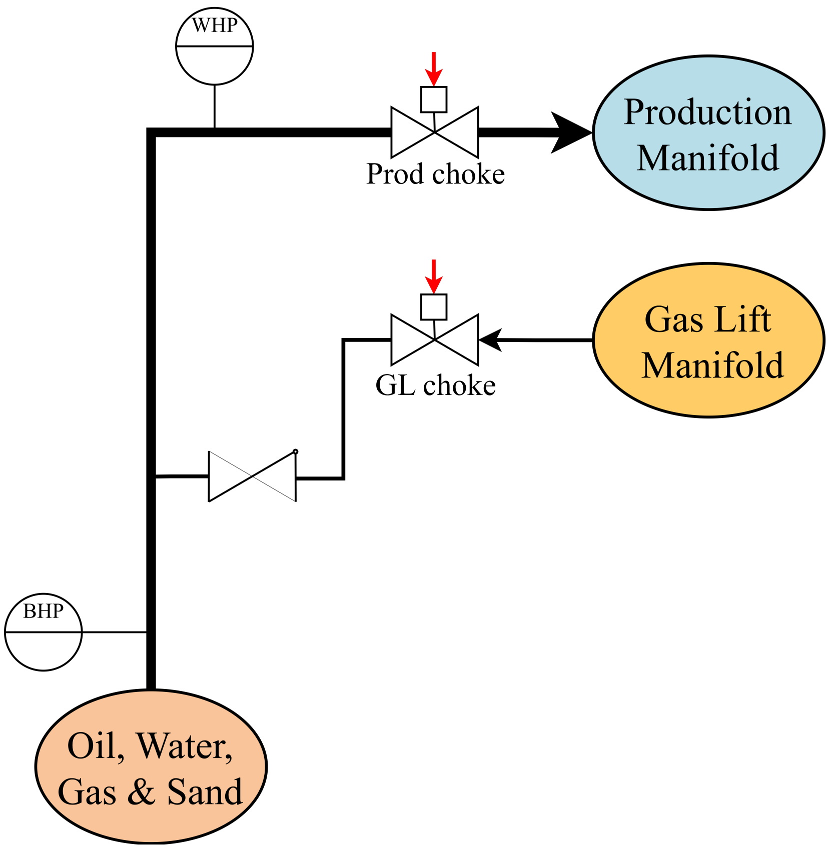

Figure 1: Simplified gas-lifted production well.

1.2 Objectives

The primary objective of this study is to explore how integer variables can be incorporated into an MPC optimization problem, and explore the potential advantages. The research question to be addressed is:

How can a discrete actuator be effectively integrated into an MPC problem to enable the controller to account for the inherent discreteness of the actuator? Furthermore, can this integration potentially result in enhanced performance compared to traditional continuous control?

The control performance will be evaluated in terms of accuracy, stability and robustness.

To accomplish this goal, the following tasks will be undertaken:

- Develop a mathematical model of the system to be controlled using step-response model representation.
- Formulate a continuous MPC problem and implement it using a Quadratic Programming (QP) solver in Python.
- Extend the continuous MPC problem to include integer decision variables and implement it using a Mixed Integer Programming (MIP) solver.
- Conduct simulation studies to compare the control performance of the two controller formulations.
- Analyze the simulation results and draw conclusions regarding the potential advantages of integrating integer variables into MPC optimization problems.

Previous work has been conducted and will be briefly outlined in this thesis. The first two tasks, developing a step-response model and implementing a continuous MPC problem to serve as a benchmark for analyzing the MIMPC, were completed by the author in the project thesis [1].

The results of this thesis will provide insight into the effectiveness of including integer variables in the MPC formulation and may help guide the design of control strategies for systems where integer control inputs are necessary.

1.3 Outline

The remainder of this report is structured as follows:
Chapter 2 provides a comprehensive overview of the fundamental theory relevant to the thesis. Chapter 3 offers a description of the system under control, while Chapter 4 delves into the specifics of the prediction model employed. The implementation of the continuous MPC, developed in the project thesis [1], is presented in Chapter 5, followed by the implementation of the Mixed Integer MPC in Chapter 6. In Chapter 7 the results are presented, followed by a discussion of the results in Chapter 8. Finally, Chapter 9 concludes the work conducted in this thesis.

2 Theory

In this chapter, the theory of mathematical optimization, mixed integer optimization, model predictive control, and mixed integer model predictive control will be delved into. Firstly, mathematical optimization is presented, which is crucial in various fields, and fundamental to understanding the subsequent sections.

Secondly, mixed integer optimization will be explored. This is a class of optimization problems involving both continuous and discrete decision variables. The challenges associated with solving these problems and the different approaches used to address them will be highlighted.

Next, model predictive control, a control strategy that uses optimization-based techniques to control a dynamic system, will be presented.

Finally, mixed integer model predictive control, a combination of mixed-integer optimization and model predictive control that has shown promise in various applications, such as chemical process control, power system control, and autonomous vehicles, will be examined.

Overall, this chapter aims to provide a comprehensive understanding of the theory of mathematical optimization and model predictive control, with a focus on mixed-integer optimization and mixed-integer model predictive control.

2.1 Optimization

This section presents a comprehensive overview of mathematical optimization, drawing primarily from the research conducted in the project thesis [1]. The objective is to establish a fundamental understanding of this crucial field, which serves as the basis for mixed integer optimization and model predictive control.

2.1.1 Mathematical Optimization

As one of the oldest branches of mathematics, optimization theory served as a catalyst for the development of geometry and differential calculus. Today it finds applications in a myriad of scientific and engineering disciplines [10].

Optimization is central to any problem involving decision-making in a broad spectrum of disciplines. Whether it is how to invest money to retrieve most profit of investments while minimizing the risk, or the optimal configuration to maximize oil production [11].

To make use of this tool, an objective, a quantifiable measure of the various decisions, often called objective function or performance index, must be defined. The objective function depends on certain characteristics of the system, called variables, and gives a measure of the various decisions, represented by a single number [2]. The goal of optimization is to choose the optimal value of the variables to either maximize or minimize the objective function. The variables are often constrained in some way, for instance, a choke can not open more than what is physically possible.

The process of identifying the objective, variables, and constraints is known as modeling. There are multiple ways to model a system, and the construction of an appropriate model might be the most important step in the optimization process. Once the model has been formulated, an optimization algorithm can be used to find its solution. There are a large collection of different optimization algorithms that can be used, and the choice of an appropriate optimization algorithm is usually done by the user. This choice may be critical, resulting in whether a solution might be found, and if so, if the problem is solved fast or slowly [2].

2.1.2 Mathematical Formulation

Mathematically speaking, there are multiple ways to formulate an optimization problem. Common for these formulations is that they all include the three main components, an objective function, decision variables, and constraints. In simple words, optimization is the minimization or maximization of the objective function subject to constraints on its variables [2]. Using this, a general optimization problem can be formulated as follows:

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & c_i(x) = 0,\ i \in \mathcal{E}, \\ & c_i(x) \geq 0,\ i \in \mathcal{I}. \end{aligned} \tag{2.1}$$

f is known as the objective function, and takes an n-dimensional vector and projects it onto the real axis. In this formulation, the decision variables are defined on a Euclidean space, as $x \in \mathbb{R}^n$, however the variables can also be integers or binary, and thus take place in other spaces. The constraints are divided into equality and inequality constraints, where $\mathcal{E}$ and $\mathcal{I}$ are disjunct index sets. The decision variables are constrained by c_i, and are thus limited to a subset of $\mathbb{R}^n$. This subset is known as the feasible set and is defined as follows:

$$\Omega = \Big\{ x \in \mathbb{R}^n \;\Big|\; \big(c_i(x) = 0,\ i \in \mathcal{E}\big) \wedge \big(c_i(x) \geq 0,\ i \in \mathcal{I}\big) \Big\}. \tag{2.2}$$

The solution to (2.1), is the point x that minimizes the objective function f, where x may only be selected from the feasible set Ω. Such a solution is denoted x^*, and is a solution if

$$f(x^*) \leq f(x), \text{ for all } x \in \Omega. \tag{2.3}$$

It is worth noting that any minimization problem can be translated into a maximization problem by observing that 'max $f(x)$' and 'min $-f(x)$' provide equal solutions apart from the opposite sign of objective function value.

If the condition (2.3) holds, x^* is a *global minimizer*. However, there may be several such minimizing points. In the case of a single global minimizer, the point x^* is a *strict global minimizer*. Such a minimizer is hard to find, as the optimization algorithms do not visit all points. Thus a picture of the overall shape of the objective function is not provided. As a result, most of the algorithms are only able to find *local minimizers*.

A local minimizer is the point x^* if there is a neighborhood $\mathcal{N}$ of x^* such that $f(x^*) \leq f(x)$ for all $x \in \mathcal{N} \cap \Omega$. However, as for a global minimizer, there may also be several local minimizers. If there only exists a single local minimizer, this is known as an *isolated local minimizer*. It can formally be said that a point x^* is an isolated local minimizer if there is a neighborhood $\mathcal{N}$ of x^* such that $f(x^*) < f(x)$ for all $x \in \mathcal{N} \cap \Omega$ with $x \neq x^*$. If x^* is an isolated local minimizer, x^* is also a *strict local minimizer*. A strict local minimizer is defined by replacing $\leq$ with $<$ in the definition of a local minimizer [2].

2.1.3 Optimality Conditions

A key question to optimization problems is how to identify a minimization point. One way to find out if a point x^* is a local minimum, would be to examine all points in the neighborhood, and verify that none of them will result in a smaller objective function. However, when f is smooth, there exists more efficient and practical ways to identify a local minimum.

For unconstrained optimization problems, where $\mathcal{E} \cup \mathcal{I} = \emptyset$, provided f is smooth and twice differentiable, it may be possible to examine the gradient $\nabla f(x^*)$ and the hessian $\nabla^2 f(x^*)$, to tell that x^* is a local minimizer. Nocedal and Wright derive a necessary and sufficient condition for optimality for unconstrained optimization in [2]. It is proven that for a point x^* to be a local optimum, x^* needs to be a stationary point. x^* is called a stationary point if $\nabla f(x^*) = 0$, which is the necessary condition for optimality. If $\nabla f(x^*) = 0$ and the hessian $\nabla^2 f(x^*) > 0$, the sufficient condition is fulfilled, which guarantees that x^* is a strict local minimizer. Note that the sufficient conditions are not necessary, thus x^* can be a local minimizer even though the hessian $\nabla^2 f(x^*)$ is not positive definite. The proof for the necessary and sufficient conditions is not further explained here, but can be seen in [2].

To characterize solutions of *constrained* optimization problems, the *Karush – Kuhn – Tucker* (KKT) conditions are to be introduced. These conditions are fundamental for many of the algorithms used to solve optimization problems. The KKT conditions include the *Lagrangian* (2.4), which is formulated as:

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(x), \tag{2.4}$$

where λ_i are the *Lagrange multipliers*. It is also necessary to define *active constraints* and *Active set* to formulate the KKT conditions.

"The active set $\mathcal{A}(x)$ at any feasible x consists of the equality constraint indices from $\mathcal{E}$ together with the indices of the inequality constraints i for which $c_i(x) = 0$; that is" [2]

$$\mathcal{A}(x) = \mathcal{E} \cup \{i \in \mathcal{I} \mid c_i(x) = 0\}. \tag{2.5}$$

This implies that all equality constraints are active at a feasible point. However, inequality constraints could be active or inactive.

With this background, the KKT conditions (2.6) can be formulated. These are also known as the *first order necessary condition* for constrained optimization problems. These conditions, derived by Nocedal and Wright in [2], are given below:

Theorem 1: First order necessary conditions:
"Assume that x^* is a local solution of (2.1) and that the function f and c_i are differentiable and their derivatives are continuous. Further, assume that all the active constraint gradients are linearly independent at x^*. Then there exists Lagrange multiplier λ^* for $i \in \mathcal{E} \cup \mathcal{I}$ such that the following conditions (called the KKT conditions) hold at (x^*, λ^*);" [2]

$$\begin{aligned}
\nabla_x \mathcal{L}(x^*, \lambda^*) &= 0 \\
c_i(x^*) &= 0, \quad \forall i \in \mathcal{E}, \\
c_i(x^*) &\geq 0, \quad \forall i \in \mathcal{I}, \\
\lambda_i^* &\geq 0, \quad \forall i \in \mathcal{I}, \\
\lambda_i^* c_i(x^*) &= 0, \quad \forall i \in \mathcal{E} \cup \mathcal{I}
\end{aligned} \tag{2.6}$$

The KKT conditions are necessary for a point to be a local solution to a constrained optimization problem. However, satisfying these conditions does not guarantee that the point is a minimum. To ensure that a point is indeed a minimum, the second-order sufficient condition has to be introduced.

Theorem 2: Second order sufficient conditions:
"Suppose that for some feasible point $x^* \in \mathbb{R}^n$ there exists Lagrange multipliers λ^* such that the KKT conditions (2.6) are satisfied, and that f and all c_i are twice differentiable and their derivatives are continuous. Assume also that

$$\nabla_{xx} \mathcal{L}(x^*, \lambda^*) \succ 0 \tag{2.7}$$

Then x^* is a strict local solution of problem (2.1)." [12]

The second order sufficient condition can be relaxed, as $\nabla_{xx} \mathcal{L}(x^*, \lambda^*)$ needs only to be positive in certain directions, where the directions are specified by the *critical cone*. Thus, the theorem above can be relaxed by $w^\top \nabla_{xx}\mathcal{L}(x^*, \lambda^*)w > 0$ where $w \in \mathcal{C}(x^*, \lambda^*)$. For further details on the critical cone, see [2].

2.1.4 Convexity

Theorem 1 provides the necessary conditions, and Theorem 2 provides sufficient conditions for a local solution. However, if the optimization problem (2.1) is a convex problem, the theorems provide conditions for a global solution. It is proven by Nocedal and Wright in [2], that a local minimizer of a convex problem also is a global minimizer for the convex problem. Thus the concept of convexity is fundamental in optimization. It is therefore essential to be able to identify convex problems.

An optimization problem is convex if the objective function f is a convex function, and the feasible set Ω is a convex set. If the objective function is *strictly convex* however, the optimization problem is *strictly convex*, and there only exists one solution, where the local solution is a global solution. For an optimization problem to be convex, it has to satisfy the following conditions [2]:

- *The objective function is a convex function.*
- *The equality constraint functions $c_i(\cdot), i \in \mathcal{E},$ are linear*
- *The inequality constraint functions $c_i(\cdot), i \in \mathcal{I},$ are concave*

For the definition of a convex function and a convex set, see [2].

2.1.5 Linear Programming

With the background from Section 2.1.2, a general optimization problem could be defined. The problem (2.1) is known as the *nonlinear program* (NLP), as the objective function and the constraints can be nonlinear. However, in the case where the objective function f and the constraints c_i are all linear, the problem lies within an optimization class called *Linear Programming* (LP). An LP problem written in the standard form can be seen in (2.8).

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & c^\top x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{2.8}$$

The vectors c and x are vectors in $\mathbb{R}^n$, b is a vector in $\mathbb{R}^m$, and A is a $m \times n$ matrix. The constraints and the objective function are all linear, hence the name *Linear Programming.* As all linear functions are convex, it is given that an LP problem is a convex optimization problem according to the conditions presented in Section 2.1.4. Thus any local solution is also a global solution for an LP problem. If the feasible set is empty, an LP problem does not provide any solution, which makes the problem infeasible. Additionally, suppose the objective function is unbounded below on the *feasible region*, which is the set of points satisfying all constraints. In that case, the minimization problem is unbounded, and no solution can be found. For an LP problem, the KKT conditions (2.6) is both necessary and sufficient.

The most common optimization algorithm for an LP problem is known as the *simplex method.* To be able to give a description of this method, the terms *basic feasible point* and *basis*, presented by Nocedal and Wright in [2], need to be introduced. The simplex method generates iterates, where each iterate is a basic feasible point. A vector x is a basic feasible point if it is feasible and if there exists a subset $\mathcal{B}$ of the index set $\{1,\ 2,\ ...,\ n\}$ such that:

- $\mathcal{B}$ contains exactly m indices;

- $i \notin \mathcal{B} \Rightarrow x_i = 0$ (that is, the bound $x_i \geq 0$ can be inactive only if $i \in \mathcal{B}$);
- The $m \times m$ matrix B defined by

$$B = [A_i]_{i \in \mathcal{B}} \tag{2.9}$$

  is nonsingular, where A_i is the ith column of A [2].

A set $\mathcal{B}$ satisfying these properties is called a *basis* for the LP problem (2.8). The corresponding B matrix is called the *basis matrix* [2]. By examining only basic feasible points, the strategy of the simplex method will lead to a solution of (2.8) only if

- the problem *has* basic feasible points; and
- at least one such point is a *basic optimal point*, that is, a solution of (2.8) that is also a basic feasible point.

Both conditions are true under reasonable assumptions, which are given by the theorem known as *Fundamental theorem of Linear Programming.* The proof of this theorem can be seen in [2].

Fundamental theorem for Linear Programming

- *If (2.8) has a nonempty feasible region, then there is at least one basic feasible point;*
- *If (2.8) has solutions, then at least one such solution is a basic optimal point.*
- *If (2.8) is feasible and bounded, then it has an optimal solution.*

The basic feasible points are vertices of the feasible polytope defined by the linear constraints of an LP problem. The proof for this can be seen in [2], Theorem 13.3. As each iterate of the simplex method is a basic feasible point, all iterates will be vertices of the feasible polytope. In Figure 2, a three-dimensional polytope can be seen, where the basic feasible points are indicated by $*$.

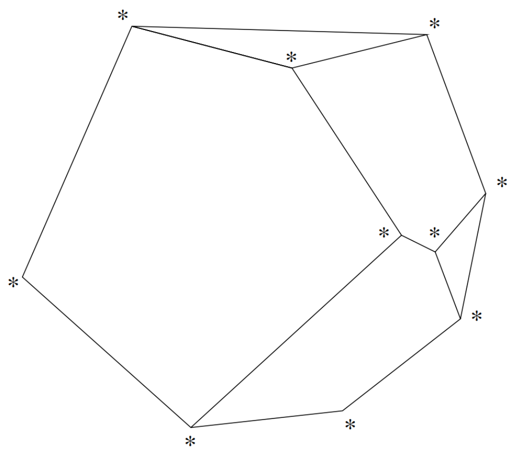

Figure 2: Polytope defined by the linear constraints of an LP problem, basic feasible points indicated by $(*)$ [2].

Most of the steps in the simplex method consist of moving from one vertex to an adjacent vertex for which the basis $\mathcal{B}$ differs with only one component. For most of these steps, the objective function is decreased, but not for all.

Simplex Method
The simplex method is based on the concept of a simplex, which is a geometric figure that consists of $n+1$ points in n-dimensional space. In the context of the simplex method, these points represent the feasible solutions to the linear programming problem (2.8).

The algorithm starts by finding a feasible solution to the problem, a basic feasible point. From there, the algorithm moves to a new basic feasible point by pivoting, which involves switching the values of one or more variables in the current solution.

At each iteration, the simplex method selects the variable to pivot on based on the current objective function and the current feasible solution. The pivot variable is chosen such that the value of the objective function will improve after the pivot. The algorithm continues to pivot until it reaches an optimal solution, which is a feasible solution that cannot be improved upon by pivoting. This is known as the basic optimal point.

One of the key challenges in implementing the simplex method is choosing the pivot variable at each iteration. There are several different pivot rules that can be used, each with their own advantages and disadvantages. The most commonly used pivot rule is the Bland's rule, which selects the pivot variable in a way that avoids cycling and ensures that the algorithm terminates in a finite number of steps [13]. Once the optimal solution is reached, the simplex method provides the values of the variables that achieve the optimal objective function value.

The simplex method is a powerful and widely-used tool for solving linear programming problems. It is an essential tool in the toolkit of anyone working in fields that involve optimization and decision-making. Additionally, the Simplex method is a building block for the Mixed Integer algorithm *Branch* & *Bound*, to be introduced in Section 2.2.6.

2.1.6 Quadratic Programming

Another class within optimization is *Quadratic Programming* (QP). Quadratic Programming differs from Linear Programming in that the objective function is quadratic and not linear. QPs are often used in control, economics and machine learning. The general QP can be stated as (2.10):

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & f(x) = \frac{1}{2}x^\top Gx + x^\top c \\ \text{subject to} \quad & a_i^\top x = b_i,\ i \in \mathcal{E}, \\ & a_i^\top x \geq b_i,\ i \in \mathcal{I}. \end{aligned} \tag{2.10}$$

where G is a symmetric $n \times n$ matrix, $\mathcal{E}$ and $\mathcal{I}$ are finite sets of indices, and c, x and $\{a_i\}, i \in \mathcal{E} \cup \mathcal{I}$, are vectors in $\mathbb{R}^n$ [2].

2.2 Mixed Integer Optimization

This section presents a comprehensive overview of Mixed Integer Optimization, a subfield of mathematical optimization that involves discrete decision variables. The theory presented in this section is mainly drawn from the research conducted in the project thesis [1], supplemented by some theoretical insights in Section 2.2.8 and Section 2.2.9, to enhance comprehension of this topic. Through this comprehensive overview, the reader will gain a deeper understanding of the complexities and challenges inherent in Mixed Integer Optimization.

2.2.1 Discrete Optimization

The optimization techniques formulated thus far belong to the general class of *Continuous optimization.* However, in many practical applications, the variables make sense only if they take on discrete values. An example of this can be in decision-making (yes or no), where the decision variables have to be binary. In other applications, e.g., production planning, the fact that the number of products to be produced must be an integer number needs to be considered. These kinds of optimization problems take place within the general class of *Discrete optimization*, more precisely, the class of *Integer Programming.*

Suppose a general linear program has been defined, but the decision variables are to be restricted only to take integer values. Such an optimization problem is known as *(Linear) Integer Programming* (IP) and can be formulated as:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^\top x \\ \text{subject to} \quad & Ax \geq b \\ & x \in \mathbb{Z}_+^n \end{aligned} \tag{2.11}$$

Notice that the variables must take place in the n-dimensional space of all *non-negative integers*, $\mathbb{Z}_+^n = \{x \in \mathbb{Z}^n \ : \ x \geq 0\}$. If all variables are restricted to binary values the problem is called a *Binary Integer Program*, which can be formulated as:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^\top x \\ \text{subject to} \quad & Ax \geq b \\ & x \in \{0,1\}^n \end{aligned} \tag{2.12}$$

For many applications, it is common to have both continuous and discrete decision variables. Linear optimizations problems where the decision variables are a combination of continuous and discrete variables are known as *(Linear) Mixed Integer Programming* (MILP):

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^\top x + d^\top y \\ \text{subject to} \quad & Ax + By \geq b \\ & (x, y) \in \mathbb{R}_+^n \times \mathbb{Z}_+^n \end{aligned} \tag{2.13}$$

A big advantage of mixed integer programming is the large flexibility to model and solve many real problems. Some examples of this are scheduling, e.g. pump scheduling optimization problem for an onshore oilfield [14], production planning, motion control, as well as the well-known *Knapsack problem* [15]. However, the inclusion of integer decision variables in integer programming constitutes a significant drawback as it transforms the original continuous problem from convex to non-convex. Consequently, traditional methods utilized for solving convex problems become ineffective and unsuitable, making the optimization process more complex.

Mixed Integer Programming poses a significant challenge as it belongs to the $\mathcal{NP}$-hard class [16]. In a mixed integer program, for each value every integer variable can take, a separate optimization problem can be constructed, where the variable is fixed. For a simple binary optimization problem with only five binary decision variables, a total of $2^5 = 32$ subproblems can be constructed. This illustrates the complexity of a mixed integer program, and how fast it grows. It should be noted that depending on the problem, many of these subproblems can be easily ruled out. Nevertheless, it is not guaranteed that there exists a simple way to reduce the search space of a problem. Further details on the different approaches to finding solutions for mixed integer problems are presented later in this section.

2.2.2 Mixed Integer Linear Programming

Mixed Integer Linear Programming (MILP), is a type of optimization problem that involves a linear objective function and linear constraints, in the same way as the LP problem (2.8). However, for MILP, some of the variables are restricted to integer values. The general form of a MILP problem is represented by (2.13).

MILP problems arise in a wide range of applications, such as production planning, resource allocation, scheduling, etc. [3]. However, the presence of the integer variables makes MILP problems computationally more challenging than their LP counterparts [17].

Much effort has been devoted to determining the computational complexity of a variety of MILP problems, and the NP-hardness. Despite extensive research, no polynomial-time algorithm has been found for solving MILP to optimality. It is currently not known whether MILP is NP-complete or not [18]. For the interested reader, an examination of a general integer programming problem and several of its variations can be seen in [18], which highlights the computational difficulties inherent in integer programming.

To solve MILP problems, various algorithms and techniques have been developed, including Branch & Bound and Cutting plane methods, presented later in this chapter. These methods aim to find feasible solutions efficiently and to narrow down the search space to find the optimal solution. The choice of the solution method often depends on the specific characteristics of the problem and the size of the problem instance.

By comparing the IP problem (2.11) with the LP problem (2.8), it is apparent that they are very familiar. It is no surprise that linear programming theory is fundamental in understanding and solving linear integer programs. A first thought could be to relax the integer program and solve it as an LP, and then use rounding to find the optimal solution to the integer program. This technique is often insufficient, which can be seen from the following example presented by Wolsey in [3].

Consider the linear integer optimization problem given by:

$$\begin{aligned} \max_{x} \quad & 1.00x_1 + 0.64x_2 \\ \text{s.t.} \quad & 50x_1 + 31x_2 \leq 250 \\ & 3x_1 - 2x_2 \geq -4 \\ & x \in \mathbb{Z}^2_+ \end{aligned} \tag{2.14}$$

The problem (2.14) is an integer problem in which the decision variables are restricted to be integers. Disregarding the integer restriction, considering it as an LP problem, and using the rounding technique to obtain an integer solution will lead to a suboptimal solution far away from the actual optimum. This is shown in Figure 3, where the optimal solution with and without the integer restrictions is illustrated. The LP solution is (376/193, 950/193), and using the insufficient method of rounding the solution to the nearest integer would then produce the solution (2, 4). However, the optimal solution to the IP problem (2.14) is (5, 0). This example shows the importance of accounting for the integer restriction in the optimization problem when solving IP problems to obtain the optimum.

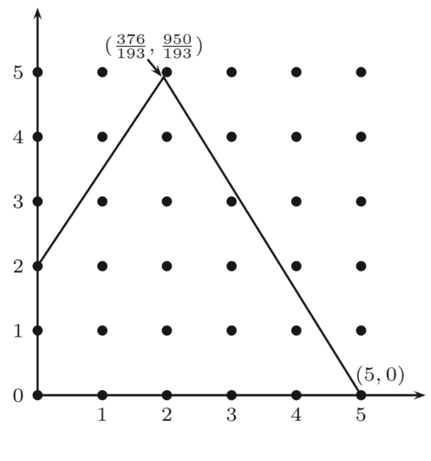

Figure 3: LP solution of a relaxed IP problem far away from the optimal solution to the IP problem

2.2.3 Mixed Integer Quadratic Programming

Mixed Integer Quadratic Programming (MIQP) extends the MILP problem by adding a quadratic objective function. MIQP is an optimization problem that involves finding the optimal value of a quadratic function over a polyhedral set, where some of the components are integers and others are continuous. The problem is similar to the QP formulation described in 2.1.6, but includes integrality constraints. When all variables are required to be integers, the problem is referred to as *Integer Quadratic Programming* (IQP). MIQP can be formally defined as an optimization problem in the following form:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^\top G x + c^\top x \\ \text{s.t.} \quad & Ax \leq b, \\ & x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}, \end{aligned} \tag{2.15}$$

where $G \in \mathbb{Q}^{n \times n}$ and is symmetric, $c \in \mathbb{Q}^n$, $A \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Q}^m$ [19].

2.2.4 Mixed Integer Nonlinear Programming

Another class of mixed integer optimization problems is *Mixed Integer Nonlinear Programming* (MINLP). A general formulation of an MINLP is a problem on the form:

$$\begin{aligned} \min \quad & f(x, y) \\ \text{subject to} \quad & c_i(x, y) = 0, \ i \in \mathcal{E}, \\ & c_i(x, y) \geq 0, \ i \in \mathcal{I} \\ & x \in \mathbb{R}^n, y \in \mathbb{Z}^q \end{aligned} \tag{2.16}$$

MINLP problems are typically more difficult to solve than linear and integer programming problems because the nonlinear objective and constraint functions make it difficult to find the optimal solution. The class of MINLP will not be further investigated in this thesis.

2.2.5 Optimality and Relaxation

In Section 2.1.3, the first order necessary conditions (2.6) for a local solution was formulated. For an LP, these KKT conditions are both necessary and sufficient and are used to find a solution. However, a fundamental difference for integer programming is that there are no such conditions similar to the KKT conditions to prove first-order optimality. It is a common question how to prove that a given solution x^* to an IP problem of the form (2.17) is optimal if there are no such optimality conditions such as the KKT conditions.

$$Z = \max\{c(x) \ : \ x \in X \subseteq \mathbb{Z}^n\} \tag{2.17}$$

As the KKT conditions cannot be applied directly to an IP problem, some optimality conditions need to be defined for an algorithm to be able to stop at the optimal point x^*. A simple and naive but important condition is finding a lower bound $\underline{Z} \leq Z$ and an upper bound $\overline{Z} \geq Z$ such that $Z = \underline{Z} = \overline{Z}$ which defines a stopping criteria for the algorithm [3].

Finding lower bounds for a maximization problem (or upper bounds for a minimization problem) is called *Primal Bounds*. Every feasible solution $x^* \in X$ provides a lower bound $\underline{Z} = c(x^*) \leq Z$.

Obtaining upper bounds for a maximization problem (or lower bounds for a minimization problem) is a different challenge. These bounds are known as dual bounds. One common approach to finding dual bounds is relaxation. This involves replacing the original integer programming problem with a simpler optimization problem whose optimal value is at least as large (for maximization) or small (for minimization) as the original problem. The relaxed problem can be solved more easily and obtain bounds on the optimal value of the original problem. Algorithms can use dual bounds to eliminate subproblems that cannot lead to an improved solution. One such algorithm that takes advantage of the dual bounds to speed up the solution process is the *Branch & Bound* algorithm.

2.2.6 Branch & Bound

A large number of discrete and combinatorial optimization problems share the same properties, that they are easy to state and have a finite but usually very large number of feasible solutions. Some of these problems, e.g., *Shortest Path* problem, have polynomial algorithms, however for the majority of these optimization problems, no such polynomial method for their solution is known. Such problems are called $\mathcal{NP}$-hard problems [16].

Solving $\mathcal{NP}$-hard problems is often an enormous job and very efficient algorithms are required to solve such problems to optimality, if possible. Branch & Bound (B&B) is an algorithm design paradigm for such optimization problems. The B&B algorithm is a kind of divide-and-conquer strategy for discrete and combinatorial programming. Essentially, the algorithm divides the problem into an equivalent set of subproblems, solves the subproblems, and obtains a solution for the original problem from the solutions of the subproblems. The divisions are performed iteratively such that the subproblems are easier to solve. An explicit enumeration is usually impossible due to the exponentially increasing number of potential solutions. However, the use of bounds on the function to be optimized, combined with the value of the current best solution, enables the algorithm to search parts of the solution space only implicitly [16].

More formally, problem (2.18) can be considered:

$$P \ : \ z = \max\{c^\top x \ : \ x \in S\} \tag{2.18}$$

The problem P (2.18) can be divided into a set of subproblems $\{SP_k\}$, forming a tree structure. The set S can be decomposed in K subsets by letting $S = S_1 \cup \ldots \cup S_k$, representing a node in the tree, where each node represents a possible solution. By letting $z^k = \max\{c^\top x \ : \ x \in S_k\}$ for $k = 1, ..., K$, then z can be formulated as:

$$z = \max\{z^k \ : \ k = 1, \ \dots, K\} \tag{2.19}$$

For small problems, an explicit enumeration is possible, however the number of feasible solutions increases exponentially with this strategy. Therefore complete enumeration is not viable for practical problems. By effectively bounding the subproblems z^k with upper and lower bounds, parts of the solutions space can be searched implicitly.

Considering the problem P (2.18), where (2.19) are optimal values for $k = 1, ..., K$. The upper and lower bounds of z^k can be defined as:

$$\overline{z}^k \text{ be an upper bound for } z^k \tag{2.20a}$$

$$\underline{z}^k \text{ be a lower bound for } z^k \tag{2.20b}$$

Thus, the upper and lower bounds of z can be defined as:

$$\overline{z} = \max\{\overline{z}^k \ : \ k = 1, \dots, K\} \text{ defines an upper bound for } z \tag{2.21a}$$

$$\underline{z} = \max\{\underline{z}^k \ : \ k = 1, \dots, K\} \text{ defines a lower bound for } z \tag{2.21b}$$

During the solution process, the status of the solution, with respect to the search of the solution space, is described by a pool of yet unexplored subsets, represented as nodes, and the best solution so far. At first, only one subset exists, which is the complete solution space, and the best solution so far is ∞. For each iteration, the B&B algorithm processes one of the yet unexplored nodes. Each iteration of the B&B algorithm consists of three main components; select what node to be processed, bound the subproblem, and branch out from this node [16].

One of the key advantages of B&B is that it is able to find the optimal solution even when the number of possible solutions is very large. This is because the algorithm is able to efficiently prune the tree, eliminating large numbers of nodes and focusing only on the most promising ones. There are three rules for cutting tree branches:

- By optimality: $z^k = \max\{c^\top x \ : \ x \in S_k\}$ has been solved
- By bounding: $\bar{z}_k < \underline{z}$.
- By infeasibility: $S_k = \emptyset$

Overall, Branch & Bound is a powerful algorithm design technique that can be used to efficiently find the optimal solution to a wide range of optimization problems. By systematically exploring all possible solutions and pruning the tree to focus on the most promising ones, Branch & Bound is able to efficiently and reliably find the best solution to even the most complex optimization problems. Another method that involves dividing the problem into smaller subproblems to make the solution process more efficient is called *Cutting Planes.*

2.2.7 Cutting Planes

In geometry, an equation with two variables is called a plane, and an equation with n variables is called a *hyperplane.* The term *cutting plane* is often used for an equality or inequality constraint that can cut off a fractional part of an LP feasible region of an IP problem, without excluding any integer feasible solution. Cutting planes is a different approach than Branch & Bound, but with the same target, to solve an integer program.

Repeated here, an integer problem in a general form can be written as:

$$\begin{aligned} \text{IP:} \quad & \max\{c^\top x \ : \ x \in X\}, \\ \text{where} \quad & X = \{x \ : \ Ax \leq b, \ x \in \mathbb{Z}^n_+\} \end{aligned} \tag{2.22}$$

The inequality $Ax \leq b$ defines the feasible set to the relaxed IP problem where $x \in \mathbb{R}^n_+$. However, the convex hull of the integer solutions is a polyhedron defined as:

$$\text{conv}(X) = \{x \ : \ \tilde{A}x \leq \tilde{b}, \ x \geq 0\} \tag{2.23}$$

Considering the same optimization problem as earlier (2.14), the convex hull of the IP problem can be seen in Figure 4.

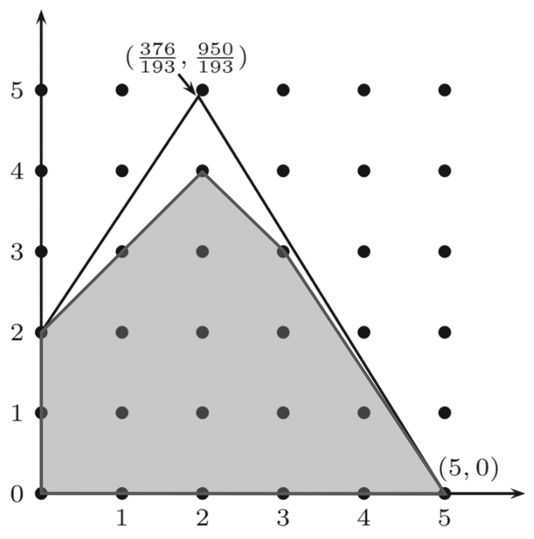

Figure 4: Convex hull of (2.14), marked in gray

The results above state that an IP problem can be reformulated as an LP problem defined by the convex hull (2.24). It should be noted that the optimal solution to an LP problem defined by the convex hull of the IP problem has the same optimal solution as the IP problem.

$$\text{LP:} \quad \max\{c^\top x \ : \tilde{A}x \leq \tilde{b}, \ x \geq 0\}, \tag{2.24}$$

Thus, if the convex hull of an IP problem is known, the solution can be obtained by solving the problem as an LP problem, and traditional methods for solving LP problems can be used, e.g. the Simplex method. In some applications, finding the convex hull is possible. However, in many applications, finding the convex hull can be very hard, and for $\mathcal{NP}$-hard problems, there is generally no hope of finding a complete description of the convex hull. Thus, the convex hull needs to be approximated to solve the IP problem as an LP problem, which is the main idea of the cutting plane approach. Such an approximation can be constructed by repeatedly adding *valid inequalities*, preferably that touches the convex hull.

Valid Inequalities

A linear inequality is *valid* for an IP problem if it is satisfied by the set of all feasible solutions of the IP. Given a polyhedron P and an optimal relaxed LP solution $x^* \in P$, a valid inequality is an inequality that cuts off the optimal solution x^* of the relaxed LP problem, but does not cut off any feasible solution of the IP problem [20]. More formally:

Theorem 3

"A linear inequality $\pi^\top x \leq \pi_0$ is valid for a nonempty polyhedron $P = \{x \ : \ Ax \leq b, \ x \geq 0\}$ if and only if there exists $u \geq 0$ such that $u^\top A \geq \pi$ and $u^\top b \leq \pi_0$." [20]

The proof of Theorem 3 can be seen in [20].

An example of a valid inequality can be seen in Figure 5. The inequality cuts of the fractional optimum to the relaxed LP problem, but do not cut away any of the feasible solutions to the IP problem.

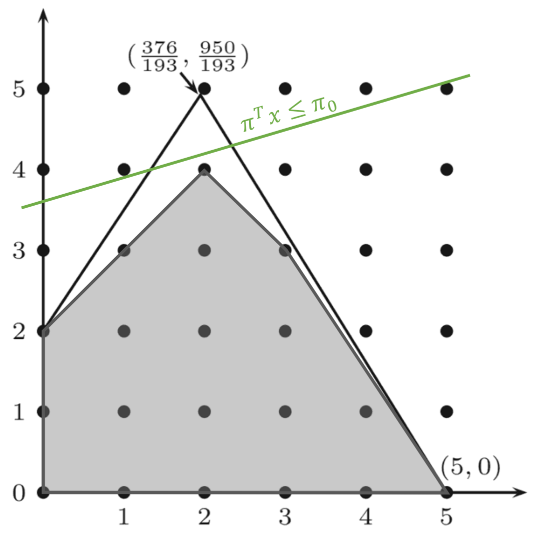

Figure 5: Example showing a valid inequality added to an IP problem

By repeatedly adding valid inequalities, the cutting plane approach will approximate the convex hull, and eventually, the IP problem can be solved as an LP problem. This approach of solving an IP problem, when used stand-alone, has the potential to solve IP programs of limited size, but may not work well in large-scale applications. Until the 1990s, the Branch & Bound had been the prevailing solution method, as this is far more effective than the cutting plane method. However, in the early 1990s, a new and more efficient approach was discovered, *Branch & Cut* [20].

2.2.8 Branch & Cut

The Branch & Cut method for solving IP problems combines the strengths of Branch & Bound and Cutting planes into a more effective approach. Conceptually, this can be viewed as a generalization of the Branch & Bound method, which involves doing more work to obtain a tight bound at each node before branching [3].

The Branch & Bound method applies simple bound cuts at each node, and takes advantage of the fast re-optimization of the LP at each node. On the other hand, Branch & Cut also divide each node, but do as much work as necessary to get a *tight bound* at the node, before branching. The work done at each node includes generating strong valid cuts, using the Cutting planes method, and improving the formulations and problem preprocessing before branching [21].

However, there is a trade-off involved. By adding many cuts at each node, the re-optimization may be much slower than before. In addition, keeping all the information in the tree becomes more challenging. In Branch & Bound, the problem at each node is obtained just by adding bounds. In Branch & Cut, a cut pool is used to store all the cuts. To keep track of which constraints are required to reconstruct the formulation at a given node, pointers to the appropriate constraints in the cut pool are kept, in addition to the bounds and a good basis in the node list [3].

A simplified flowchart of the Branch & Cut method is illustrated in Figure 6. This method is used in most of the mixed integer solvers today [20].

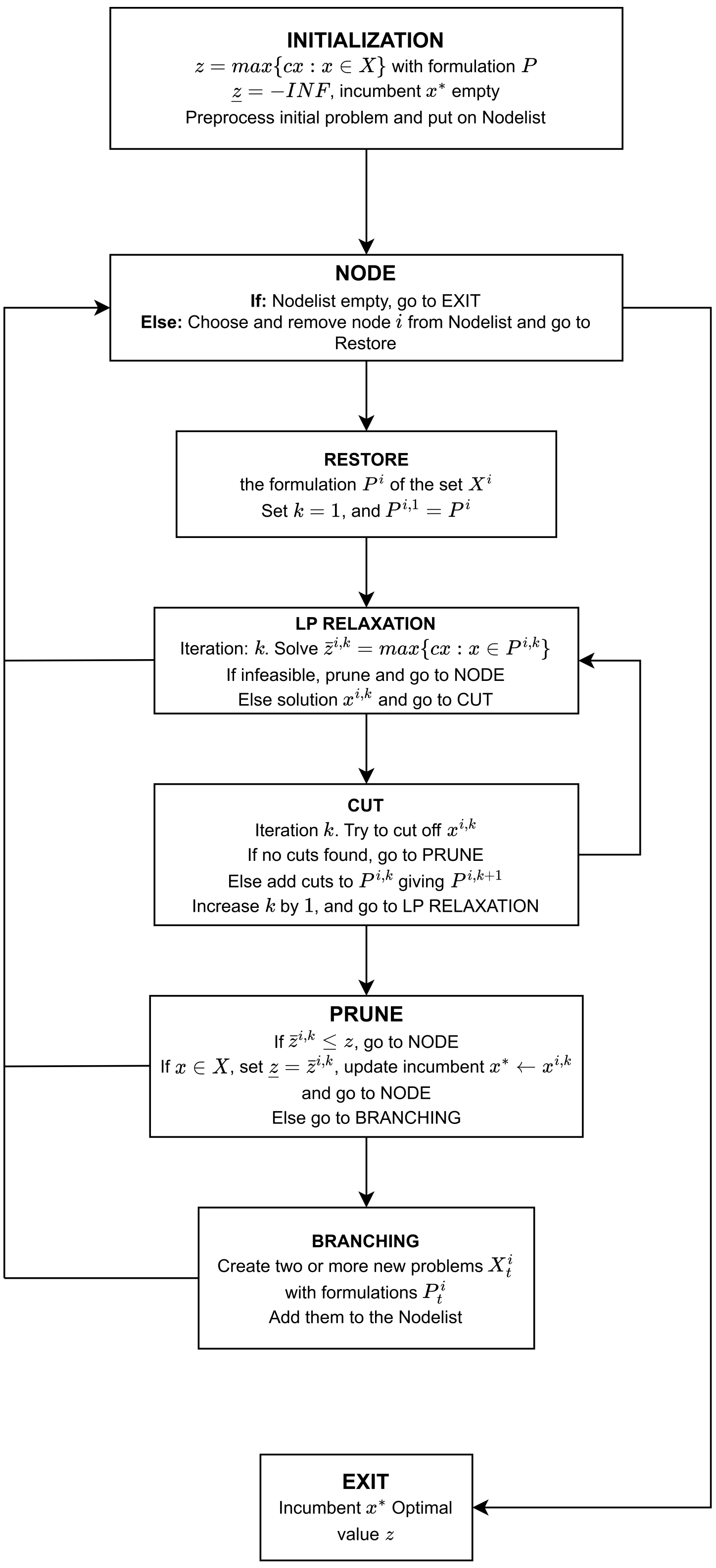
INITIALIZATION
$z = max\{cx : x \in X\}$ with formulation P
$\underline{z} = -INF$, incumbent x^* empty
Preprocess initial problem and put on Nodelist
NODE
If: Nodelist empty, go to EXIT
Else: Choose and remove node i from Nodelist and go to Restore
RESTORE
the formulation P^i of the set X^i
Set $k = 1$, and $P^{i,1} = P^i$
LP RELAXATION
Iteration: k. Solve $\bar{z}^{i,k} = max\{cx : x \in P^{i,k}\}$
If infeasible, prune and go to NODE
Else solution $x^{i,k}$ and go to CUT
CUT
Iteration k. Try to cut off $x^{i,k}$
If no cuts found, go to PRUNE
Else add cuts to $P^{i,k}$ giving $P^{i,k+1}$
Increase k by 1, and go to LP RELAXATION
PRUNE
If $\bar{z}^{i,k} \leq z$, go to NODE
If $x \in X$, set $\underline{z} = \bar{z}^{i,k}$, update incumbent $x^* \leftarrow x^{i,k}$
and go to NODE
Else go to BRANCHING
BRANCHING
Create two or more new problems X_t^i
with formulations P_t^i
Add them to the Nodelist
EXIT
Incumbent x^* Optimal value z

Figure 6: Flowchart Branch & Cut [3]

2.2.9 MIP Solvers

Mixed Integer Programming solvers are powerful tools for solving complex problems involving discrete and continuous variables. Unlike solvers focused solely on continuous optimization, MIP solvers take into account the combinatorial nature introduced by integer variables.

MIP solvers employ sophisticated algorithms and techniques, such as the algorithms presented in this section, to efficiently explore the search space and identify optimal solutions. Several MIP solvers are readily available, each offering distinct features and performance characteristics. Some popular solvers include Gurobi [22], CPLEX [23], and XPRESS [24]. These solvers have a long-standing reputation for their efficiency and reliability in solving large-scale, real-world optimization problems [25].

Gurobi, in particular, is widely recognized for its remarkable speed and state-of-the-art optimization algorithms. It is a high-performance solver that has been optimized to leverage modern computer architectures effectively. Gurobi provides a user-friendly interface, supports multiple programming languages, and offers various features that facilitate problem formulation and analysis. Additionally, Gurobi has extensive documentation, tutorials, and user communities, making it easily accessible to both new and experienced users [22].

For academic research and smaller-scale applications, open-source MIP solvers like CBC [26], GLPK [27], and IPOPT [28] present viable alternatives. These solvers continue to evolve and improve with ongoing contributions from the optimization community.

2.3 Model Predictive Control

This section provides the reader with an introduction to *Model Predictive Control* (MPC). A thorough understanding of MPC is essential to be able to expand a traditional continuous MPC to incorporate integrality constraints and extend it to a Mixed Integer MPC, which is one of the objectives of this study. By providing a foundational understanding of MPC, this section will enable the reader to grasp the fundamental principles and concepts that underpin this advanced control strategy.

MPC is a form of advanced control widely applied in petrochemical and related industries [29], and has become the most popular advanced control technology in the chemical processing industries [30]. In essence, MPC is a dynamic control strategy that involves solving a finite horizon open-loop optimal control problem at each sampling instant, using the current state of the system as the initial state. The optimization yields an optimal control sequence, where the first control input in this sequence is applied to the plant [29]. This differs significantly from conventional control strategies that rely on pre-computed control laws. By taking an on-line, real-time approach to control, MPC enables the system to adapt quickly to changes in operating conditions and disturbances, resulting in more efficient and effective control [29].

2.3.1 Constrained Model Predictive Control

There are many variants of MPC, both in academia and industry, but they all share the common feature of using an explicitly formulated process model to predict and optimize future process behavior [30]. While the initial focus of MPC was on controlling multivariable plants, it has become increasingly clear that the ability to handle control problems where off-line computation of a control law is difficult or impossible is one of its main strengths. In particular, MPC's ability to effectively handle constraints has made it a valuable tool in many practical applications [29].

In almost every application, constraints are imposed on the system. For instance, actuators are naturally limited by the force (or equivalent) they can apply, while safety limits states such as temperature, pressure, and velocity. Efficiency often requires steady-state operations near the boundary of such constraints. Despite the prevalence of hard constraints, there are few control methods capable of handling them efficiently. As a result, ad hoc methods have been frequently used in the industry in the absence of such control methods. MPC is one of the few methods capable of effectively handling constraints. By explicitly formulating a process model and using an optimization-based approach, MPC can account for constraints both on manipulated variables (often inputs) and controlled variables (often states/outputs). This ability makes MPC a suitable control strategy in numerous practical applications [29].

2.3.2 Optimal Control

The concept of MPC merges feedback control with dynamic optimization, to provide optimized responses, while accounting for state and input constraints of the system. By combining feedback control with dynamic optimization, MPC provides optimal performance, which can be a huge advantage when e.g. multiple control objectives are desired.

2.3.3 Linear MPC

One common feature of all MPC strategies is that they rely on three key elements: an objective function, constraints, and a prediction model. These key elements form the basis of the optimization problem in MPC, enabling the strategies to predict system behavior, optimize performance, and ensure safe and feasible control actions.

To achieve efficient solutions to the optimization problem in MPC, one must consider the limited computation time available between each sampling interval. Conventionally, the optimization problem is transformed into either LP or QP formulations. Although LP formulation offers a superior solution strategy for very large optimization problems, QP formulation typically generates smoother control actions and a more understandable impact of tuning parameter modifications [30]. Despite the computational efficiency of LP optimization, the majority of both industrial and academic MPCs employ QP optimization [31]. SEPTIC, the in-house MPC software at Equinor, also adopts QP formulation. Consequently, this section will focus on the description of QP MPC formulation, often called *Linear MPC*.

The linear MPC problem is a constrained convex optimization problem that seeks to minimize a cost function. This cost function is usually chosen as the l_2 type cost, representing the sum of the squared errors between the reference trajectory and the predicted trajectory, and the input moves [32].

$$\mathcal{J} = \sum_{j=H_w}^{H_p} \|y(k+j|k) - r_y(k+j)\|_Q^2 + \sum_{j=0}^{H_u-1} \|\Delta u(k+j)\|_P^2 \tag{2.25}$$

The selection of weighting matrices $Q \succeq 0$ and $P \succeq 0$, as well as prediction horizon H_p and control horizon H_u, play a crucial role in ensuring the performance and stability of the MPC system [33]. k is the current sampling instant and the range of j corresponds to the prediction horizon. The *controlled variables* are known as CVs, and the *manipulated variables* are known as MVs, where the number of the variables are denoted n_y and n_u, respectively. The Q matrix penalizes deviations of the predicted CVs, $y(k+j|k) \in \mathbb{R}^{n_y}$ from the reference trajectory vector $r_y(k+j)$, whereas the P matrix penalizes the changes in the MVs, $\Delta u(t) \in \mathbb{R}^{n_u}$. To avoid immediate penalization of deviations of y from r_y, the prediction horizon H_p may start from $H_w > 0$.

One of the benefits of transparent constraint handling in MPC is the ability to specify constraints for the system in a clear and straightforward manner. It is common to distinguish between *hard constraints* and *soft constraints*.

Hard constraints refer to the constraints that must be strictly adhered to, without exception. Physical limitations of actuators are an example of such constraints. Typically, input moves and inputs are hard constrained within a defined upper and lower bound [32]. These constraints can be formulated as follows:

$$\underline{\Delta u} \leq \Delta u(k+j) \leq \overline{\Delta u}, \quad \underline{u} \leq u(k+j) \leq \overline{u}, \quad j \in \{0, \, ... \, , H_u - 1\} \tag{2.26}$$

On the other hand, soft constraints refer to constraints that are desirable to be satisfied but not strictly necessary. In such cases, it may be permitted to violate the constraint to avoid infeasibility, but doing so incurs a cost. This involves modifying the constraints by introducing additional variables in a way that they are always feasible for sufficiently large values of these variables. A penalty function is then introduced into the objective function to penalize the magnitude of constraint violations. These additional variables, often known as *slack variables*, ensure the feasibility of the optimization problem and become free variables in the optimization problem [30]. A common way of incorporating soft constraints through the use of slack variables is:

$$\underline{y} - e_j \leq y(k+j|k) \leq \overline{y} + e_j, \quad e_j \geq 0, \quad j \in \{H_w, \, ... \, H_p\} \tag{2.27}$$

The slack variables e_j are then penalized in the objective function $\mathcal{J}$, (2.25), by the inclusion of a penalty function. Several penalty functions can be used, but the penalty function is desired to be exact. This means that no constraint violations occur if the original problem is feasible.

It is common to use either a quadratic penalty, $\sum_{j=H_w}^{H_p} \|e_j\|_{\mathcal{R}}^2$ or a linear penalty $\rho \sum_{j=H_w}^{H_p} \|e_j\|_1$. When ρ is chosen large enough, the constraint violation does not occur unless there is no feasible solution to the original 'hard' problem [32].

To enable the calculation of future outputs based on an optimal sequence of control inputs, a prediction model is required to capture the dynamics of the system. In the case of linear MPC, the plant is assumed to have a linear behavior governed by the following equation:

$$y(k+j|k) = F(\hat{p}, \Delta u(k+j)) \tag{2.28}$$

The variable $\hat{p}$ is dependent on the specific model representation employed. In a state-space representation, $\hat{p}$ corresponds to the state estimates, whereas in step-response models, it may include the output measurement and prior inputs [32]. Equinor employs the step-response representation. Consequently, the following subsection will concentrate on step-response models for the prediction model.

2.3.4 Step Response Model

Within this subsection, a brief yet comprehensive understanding of the fundamental components of step response models is presented. Drawing primarily upon research conducted in the project thesis [1], this overview provides the reader with a foundational understanding of step response modeling. These models play a crucial role in the Mixed Integer MPC implementation of this project, making this overview an essential component of this study.

Step response models are widely used in the industry. This is mainly because they are easy to build, understand and maintain. Even though, in some applications, a state-space representation would have improved the predictions, it may be challenging to model the dynamics of the system [9]. The step response models offer the advantage that they can represent stable processes with unusual dynamic behavior that cannot be described by simple transfer function models. Their main disadvantage is the large number of model parameters required to represent the dynamics of the system [34].

The step response model of a stable single-input single-output process (SISO) can be written as (2.29).

$$y(k) = \sum_{i=1}^{N-1} s(i)\Delta u(k-i) + s(N)u(k-N) \tag{2.29}$$

This model describes the dynamic and static interactions between the input u and the output y. The model parameters, also known as step response coefficients $s(i)$, are obtained from a step in the input, with the process initially at steady-state. Knowledge about the past control moves $\Delta u(k-i)$ is required to estimate the output, where $\Delta u(k)$ is defined as $\Delta u(k) = u(k) - u(k-1)$. The given step response model, (2.29), is only valid if the SISO system is asymptotically stable, which implies that the step response coefficients $s(i)$ reach constant values after N sampling periods. This implies that $s(N+1) \approx s(N)$ [35].

Model predictive control is based on predictions of future outputs over a prediction horizon. Based on the model (2.29), a prediction of the future output trajectory can be formulated by:

$$\hat{y}(k+j|k) = \sum_{i=1}^{j} s(i)\Delta u(k+j-i) + \sum_{i=j+1}^{N-1} s(i)\Delta\tilde{u}(k+j-i) + s(N)\tilde{u}(k+j-N) + v(k|k) \tag{2.30}$$

where $\hat{y}(k+j|k)$ represents the prediction of $\hat{y}(k+j)$ using available information at time k [36]. The terms on the right-hand side of (2.30) can be distinguished into *known* and *unknown* terms. The first term on the right-hand side consists of *unknown* terms, including the future and present input moves $\Delta u(\cdot)$. The other terms consists of *known* terms, computed using the past input $\tilde{u}(\cdot)$, past input moves $\Delta\tilde{u}(\cdot)$, and a disturbance model $v(k|k)$, that introduce feedback [32].

A challenge with step-response models is modeling error. Without feedback, the cumulative effect of model error and unmeasured disturbances will lead to inaccurate predictions. A disturbance model $v(k+j|k)$, also known as a *bias* model, is therefore used to correct the predictions:

$$\hat{y}(k+j|k) = \tilde{y}(k+j|k) + v(k+j|k) \tag{2.31a}$$

$$v(k+j|k) = v(k|k) = y_m(k) - \tilde{y}(k|k-1) \tag{2.31b}$$

where y_m is the measured process output at time k. By including the bias model (2.31b), integral action is provided in the MPC [32]. The bias term is used in correcting the predictions $\tilde{y}(k+j|k)$, where the notation $\tilde{\cdot}$ is used to indicate uncorrected predictions, and the notation $\hat{\cdot}$ is used for corrected predictions. The bias model (2.31b) assumes that the bias is constant throughout the whole prediction horizon. If this is not the case, the prediction will be incorrect, which may lead to poor control performance [32].

2.4 Mixed Integer Model Predictive Control

In Section 2.3, the traditional continuous MPC strategy was presented, which relies on the continuous optimization theory outlined in Section 2.1 to achieve optimal control. While this approach has been widely used as an effective control strategy, it is not without limitations. Many industrial processes involve discrete decisions, which the standard MPC methods are not well suited to handle [37]. In this case, “standard” MPC implies MPC without discrete-valued input decisions. Throughout this thesis, the standard MPC will be known as the ”*continuous MPC*”, implying that the optimization variables are continuous.

Despite the prevalence of discrete-valued actuators in industrial processes, the early literature on MPC focused entirely on continuous actuators, while leaving the discrete decisions to other automation methods such as heuristics. This was largely due to the computational complexity incurred when integrality constraints were introduced into the optimization problem. However, recent advancements in mixed integer optimization and increased computational power have enabled the direct inclusion of discrete decisions in many MPC optimization problems. This approach, known as *Mixed Integer Model Predictive Control* (MIMPC), has shown significant improvements in the operation of systems that require the use of discrete decisions [37].

To account for the additional class of discrete-valued decisions, the traditional MPC theory has been expanded to incorporate both continuous- and discrete-valued actuators. Despite this addition, the fundamental structure of MPC remains unchanged: at each sampling interval, an optimization problem is solved to determine an optimal trajectory of states and inputs, but only the first input is utilized in practice [38].

Incorporating discrete-valued actuators presents several stability and robustness challenges for closed-loop control. Specifically, discrete actuators have traditionally been assumed to exhibit local convexity and controllability. However, research has shown that results applicable to continuous systems can also apply to systems featuring discrete actuators (see e.g. [38, 37, 39]).

2.4.1 Stability

A stability analysis of MIMPC is not in the scope of this thesis. Still, it is worth mentioning some of the research conducted in this area, as there have been several notable findings regarding the stability of MIMPC.

Traditionally, stability analysis in MPC theory focused primarily on systems with continuous actuators. However, recent advancements in the field have led to more abstract theory of MPC stability, without explicit differentiation between continuous and discrete actuators. In a notable contribution, Risbeck formulates a general MPC problem in [38] and proves stability, which is also compatible for discrete-valued actuators. The stability proofs developed by Risbeck show that most of the stability already developed for traditional continuous MPCs can easily be extended to hold for MIMPC.

In 2017, James B. Rawlings and Michael J. Risbeck published a paper with the goal of seeing how far they could develop the following motivating stability idea:

Folk Theorem: ”*Any result that holds for standard MPC holds also for MPC with discrete actuators.*” [40]

Their research yielded remarkable findings, demonstrating that multiple classes of recent stability results established for continuous MPC with continuous actuators can be applied without modification to the case of discrete actuators. The results include closed-loop stability of an equilibrium point under both optimal and suboptimal MPC, as well as closed-loop stability of periodic solutions when using a tracking objective.

Other significant contributions regarding the stability of MPC with discrete actuators have also emerged in the literature. Bemporad and Morari, in their work [39], demonstrated convergence to the origin for systems involving both continuous and discrete decisions by employing a set of logical rules based on positive-definiteness restrictions for the stage cost. In another notable study by Di Cairano, Heemels, Lazar, and Bemporad, [41], a hybrid Lyapunov function incorporating both continuous and discrete variables was directly integrated into the optimal control problem, enforcing cost decrease as a hard constraint.

Among others, these works have significantly advanced the understanding of the stability of MPC with discrete actuators. By showcasing the compatibility and transferability of stability results from continuous MPC to the discrete actuator case, they provide valuable insights into the design and analysis of MPC controllers with discrete decision variables.

It should be noted that the extension from continuous MPC to mixed-integer MPC is not effortless. Instead, the key observation in [40] is that the challenges associated with incorporating integer constraints into MPC problems are manageable with the existing set of theoretical tools available.

2.4.2 Robustness

In addition to nominal stability, a control method must ensure some margin of robustness to disturbances for successful implementation [37]. The stability theory for MPC typically assumes nominal operation; that is, the system model is exact and the system evolves exactly as the prediction model. However, real-world systems often deviate from this ideal scenario due to unmodeled disturbances and inaccuracies in the prediction model. Therefore, it is desirable to ensure that nominal system properties do not catastrophically deteriorate when small disturbances are present [38].

In the literature, various techniques have been proposed to address disturbances during the design phase, see [42]. These robust-by-design approaches are particularly valuable when disturbances are prevalent. However, in [38], Risbeck demonstrates inherent robustness of MIMPC, i.e., robust to small disturbances without any modification to the nominal controller. The main consequence of the findings is essentially that a tracking MPC controller that is nominally stable cannot be destabilized by arbitrarily small disturbances. This property is important because even the most accurate model cannot perfectly capture the exact behavior of the true system.

While applications that require explicit guarantees may necessitate robust-by-design techniques, practical implementations often demonstrate that the system remains relatively close to the setpoint despite quite large disturbances. Therefore, the inherent robustness of tracking MIMPC is often sufficient in practice [38].

2.4.3 Computation

Computational complexity is a crucial aspect to consider when implementing MIMPC due to the additional complexities introduced by integer decision variables. Compared to continuous MPC, the inclusion of discrete decisions significantly impacts the computational requirements.

In MIMPC, solving the optimization problem involves solving mixed integer programs, such as described in Section 2.2. MIP is known to be computationally challenging, as it falls into the class of NP-hard problems. Finding the global optimum for large-scale MIP problems can be very hard. However, through available optimization packages, i.e. those described in Section 2.2.9, MILP and MIQP problems can often efficiently be solved at a significant scale [37]. In the context of MPC applications, where linear systems with quadratic objective functions are common, computational efficiency can be improved by exploiting the structure of the MPC problem and utilizing warm starts for the optimizer [43, 44].

A notable finding in [38] is that via liberal application of suboptimal MPC, the need to find globally optimal solutions to any MPC problem is eliminated. Thus, computational requirements are readily satisfiable using standard hardware.

The computational complexity of mixed-integer optimization problems is heavily impacted by the size of the problem instance. Instances with a high number of decision variables and constraints may exceed the time constraints imposed by real-time applications. Consequently, it is essential to carefully evaluate the problem size to ensure that the computational requirements can be feasibly met within the desired time constraints.

It is worth noting that the computational complexities associated with MIMPC should be carefully considered during the design phase. Balancing the trade-off between computational requirements and control performance is crucial to ensure that the MIMPC solution is feasible and can be implemented within the available computational resources.

3 System Description

3.1 Gas-lifted Oil Well System Description

This section offers an overview of the system being controlled. Given the use of experimental models to simulate the system, a comprehensive description is beyond the scope of this thesis. However, a brief physical description of the system is still required to provide context and clarity.

3.1.1 Oil And Gas Production

Equinor produces around 2 million barrels of oil equivalent every day, and is responsible for about 70 percent of overall Norwegian oil and gas production [4]. Their operations extend beyond the Norwegian coast, encompassing various production platforms that extract oil and gas from multiple wells within a single field. One of the platforms operated by Equinor, The Oseberg Field Center, can be seen in Figure 7.

Figure 7: The Oseberg Field Centre. One of the platforms operated by Equinor, with a total of 21 oil and gas wells tied into the Field Center [4]. Courtesy Harald Pettersen – Equinor [5].

To bring the oil and gas to the platforms, the wells are usually connected to a subsea pipeline that runs along the sea floor. The pipeline connects to a riser, which is a vertical pipe that extends from the sea floor up to the platform. The riser transports oil and gas from the well to the platform for processing and separation, while chokes are employed to regulate the flow [45]. As an oil reservoir matures, the declining internal pressure may hinder the natural lifting of fluids to the surface, necessitating the use of artificial lifting techniques. Various economic and technological factors influence the choice of technique, with gas-lift being a widely employed method due to its resemblance to natural flow and its reputation as a versatile artificial lift technique [46].

Gas-lift is a production technique that boosts well productivity by injecting high-pressure gas at the bottom of the well, reducing the hydrostatic pressure of the fluid column and increasing the production rate. The gas is introduced through a gas-lift valve, and as it ascends, it mixes with the fluids, decreasing the overall density of the fluid column. This density reduction creates a pressure gradient from the well's bottom to the surface, resulting in a more efficient flow as the well's natural pressure propels the fluids upward [45]. Figure 8 provides an illustration of a gas-lifted oil well.

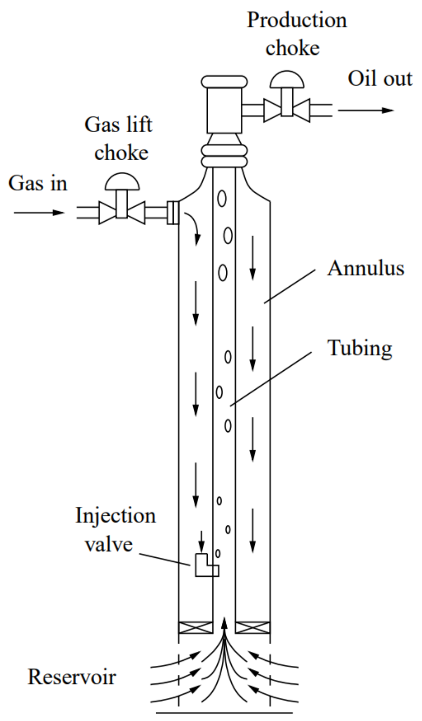

Figure 8: A gas-lifted oil well [6].

3.1.2 Gas-lifted Oil Well

This project aims to regulate the operation of a gas-lifted oil well. To conduct simulations and assess the control system's performance, a model provided by Equinor is employed. The model is constructed using JModelica, a comprehensive software platform that utilizes the Modelica language for developing, simulating, optimizing, and analyzing complex dynamic systems [47].

The gas-lifted oil well is modeled as a MIMO system with multiple inputs and multiple outputs. However, for this project, the system is simplified to a MIMO system with only two inputs, production choke and gas-lift rate, and two outputs, oil rate and gas rate. The choke position is expressed as a percentage between 0% and 100%, while the gas-lift rate is limited between $0 - 10000\frac{m^3}{hr}$. Low-level PI controllers are implemented in the model to acquire the desired choke position and gas-lift rate.

The production choke in this system is a step choke with a step size of 2%. Step-chokes are commonly used in production systems, where the choke position is adjusted in discrete steps rather than continuously. As a result, the choke does not open smoothly but rather in distinct increments of 2%.

The controlled variables in the system are the oil and gas rates, which are measured using an ideal sensor Modelica block to obtain accurate volume flow rates. To ensure smooth and stable measurements, the measured values are further processed through first-order filters.

3.2 System Implementation

Models are usually described by a combination of differential, algebraic, and discrete equations, with events associated with time, state, and step transitions. However, different modeling and simulation tools often have their own unique approaches to representing and storing model data, which can make it difficult to import and integrate models across multiple platforms [48]. This can pose a challenge when attempting to simulate a model in a different tool or when attempting to verify results obtained from different simulation tools. To address these challenges, a standardized interface known as the Functional Mockup Interface (FMI) has been developed through a European consortium, and is now managed as a Modelica Association Project (see [49]). To facilitate the use of different modeling and simulation tools, the oil well model is compiled according to the FMI standards.

3.2.1 Functional Mockup Interface

The Functional Mockup Interface (FMI) is a standardized approach that enables tool-independent exchange of dynamic models. This allows modeling environments to generate C-code of a dynamic system that can be utilized by other modeling and simulation environments, either in source or binary form. In particular, all Modelica models can be handled, and all Modelica variable attributes and description texts can be exchanged between different environments [49].

A model is distributed in a single zip-file with the ".fmu" (Functional Mockup Unit) extension. The zip-file contains several files, including an XML-file that contains the definition of all variables in the model and other model information. Additionally, a binary-file is included which provides all necessary model equations in the form of a small set of easy-to-use C-functions. These C-functions are typically provided in binary form, but can also be provided in source form for more advanced users. Other data such as maps and tables needed by the model can also be included in the zip-file. Overall, FMI provides a flexible and efficient way to distribute and integrate dynamic models across different modeling and simulation environments [48].

3.2.2 System Simulation

Once the model is compiled to comply with the FMI standard, it becomes possible to simulate it using Python. A Python interface for interacting with FMUs is available through the PyFMI package (see [50]), which enables loading FMUs, setting model parameters, and evaluating model equations. This package can be obtained as a stand-alone package or as a part of the JModelica.org distribution.

To import the dynamic model, the PyFMI object must be imported to Python. This object handles unzipping the model, loading the XML description, and connecting the binaries for use in Python. The instance returned depends on the FMI definition, which can either be Co-simulation or Model Exchange. If the FMUs are exported as Co-simulation, no additional packages are necessary since the solver is included inside the FMU. However, if the FMUs are exported as Model Exchange, the Assimulo package is required to solve the model [50].

Given that the FMU provided by Equinor is exported as Co-simulation, measurements can be directly extracted from the simulation without the need for additional packages.

To simulate the dynamic model, a step-wise implementation of inputs is required. This can be accomplished by calling the `model.do_step(t, delta_t)` function, where `t` represents the current time, and `delta_t` represents the time step. `t` is then incremented by `delta_t` for each time-step.

To set the input before executing a step, the following code can be used:

```
model.set_real(valRef_input , input_value)
```

The first argument is a unique identifier for each parameter in the model and can be found in the XML file stored in the .fmu file. The second argument represents the input value for the next time step. After the time step, the outputs can be extracted using the following code:

```
model.get_real(valRef_output)
```

The model also includes a boolean variable, `SepticControl`, which determines whether a pre-defined input sequence or user-specified inputs are used in the simulation. By default, `SepticControl` is set to 'False', indicating that the predefined input sequence will be used. To use user-specified inputs, the variable must be set to 'True' using the following Python code:

```
    model.set_boolean([ValueReference_SepticControl], [True])
```

It is important to note that this code must be executed before every simulation, after initializing the model.

4 Prediction Model

To enable a model predictive controller to accurately predict future states and optimize inputs, a prediction model that accurately represents the system under control is crucial. While Equinor has developed a theoretical model for a gas-lifted oil well, it is unsuitable as a prediction model for the MPC. Theoretical models that rely on the physics of complex processes with a significant number of process variables and unknown parameters are impractical as a prediction model due to their complexity. Moreover, an extension of the system would require new theoretical models to be developed, which is a very time-consuming and difficult task [34]. Instead, models based on experimental data, such as step response models, are widely used in the industry, including at Equinor.

In this project, the theoretical model represented by the FMU will serve as the plant in the control system, while a step response model will serve as the prediction model. Figure 9 shows an overview of the control system. Although it would be preferable to utilize a real oil well as the plant, the theoretical model developed by Equinor is a suitable alternative to be able to perform simulations of the control system. The use of the theoretical model as the plant will also serve as a foundation upon which the step response model can be constructed.

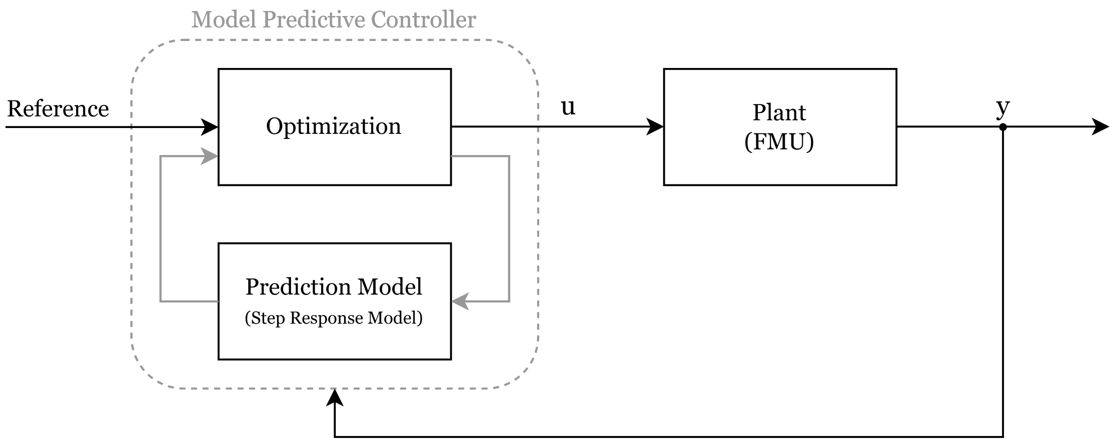

Figure 9: Overview of the control system with the FMU functioning as the plant, and a step response model used for predictions.

4.1 Step Response Modelling

In this work, a linear step response model has been employed as the prediction model. As mentioned, this modeling technique is widely used in the industry due to the challenges of obtaining an accurate state-space representation of complex systems [34]. This type of modeling is also done in Equinor and employed in SEPTIC.

The step response model employed in this study was originally developed in the project thesis [1]. A prediction model plays a crucial role in the successful implementation of an MPC. Given its significance, it is essential to provide a more detailed explanation and presentation of the prediction model in this work.

As described in Section 2.3.4, the step response model relates changes in the process output to a weighted sum of past input changes, referred to as input moves. The step response coefficients determine the weighting factors in this sum. For convenience, the step response model (2.29) is repeated here:

$$y(k) = \sum_{i=1}^{N-1} s(i)\Delta u(k-i) + s(N)u(k-N), \tag{4.1}$$

where the variable Δu denotes the change in input u, $s(i)$ is the corresponding step coefficient and N is the number of samples.

In this work, the process under control involves two inputs and two outputs, rendering a single-input-single-output (SISO) model, such as (4.1), inadequate for the plant representation. To overcome this limitation, the step response modeling utilizes the superposition principle for multivariable processes to represent MIMO systems. This technique involves applying a step to each manipulated variable individually and recording the corresponding responses of the controlled variables. These responses generate multiple SISO step responses of the process, which can be grouped together to obtain a MIMO model, as demonstrated in (4.2). It should be noted that the number of samples N may differ for each SISO model, although it is assumed to be equal in this particular case.

$$y(k) = \sum_{i=1}^{N-1} S(i)\Delta u(k-i) + S(N)u(k-N) \tag{4.2}$$

The step response matrix $S(i)$ contains the step coefficients of the SISO models and groups them together to represent the MIMO system. The step response matrix is defined in (4.3). $S(i)$ is an $n_{CV} \times n_{MV}$ matrix, where n_{CV} is the number of controlled variables and n_{MV} is the number of manipulated variables. The element $s_{j,k}(i)$ of the matrix corresponds to the step response coefficient of the j-th controlled variable due to a step in the k-th manipulated variable at time i.

$$S(i) = \begin{bmatrix} s_{1,1}(i) & s_{1,2}(i) & \cdots & s_{1,n_{MV}}(i) \\ s_{2,1}(i) & s_{2,2}(i) & \cdots & s_{2,n_{MV}}(i) \\ \vdots & \vdots & \ddots & \vdots \\ s_{n_{CV},1}(i) & s_{n_{CV},2}(i) & \cdots & s_{n_{CV},n_{MV}}(i) \end{bmatrix} \tag{4.3}$$

This model representation assumes that the step response coefficients are obtained from a process initially at steady-state. This is accomplished by simulating the plant for a period of time, with constant inputs, until a steady-state is reached. It should be noted that the model is only valid if the process is asymptotically stable, which is assumed in this study.

4.1.1 Step Response Model Generation

The step response model is constructed by independently applying a step to each manipulated variable and recording the corresponding response of the controlled variables. The resulting step coefficients are saved in NumPy arrays and loaded into the MPC algorithm's initialization file. It is worth noting that the step response model is based on an FMU, which is essentially a mathematical representation of the oil well rather than a real-world process. Since the FMU remains static, the step response model is also static and does not change over time.

Due to the presence of nonlinearities in the process, careful consideration is required when determining the working point for the step response model to ensure its accuracy. Nominal values of the manipulated variables can serve as a suitable reference point for the construction of the model. However, due to the considerable variation in the manipulated variables, it was found to be challenging to determine a nominal value. Figure 10 and Figure 11 illustrate the steady-state step coefficients obtained by applying a step to the choke while keeping the gas-lift rate constant. These figures clearly demonstrate the presence of nonlinearities in the system. Accuracy is important when generating a step response model to be used as a prediction model in the MPC. However, as revealed by the figures, the steady-state responses exhibit significant variability. Notably, the responses exhibit substantial differences depending on the extent of choke opening: responses are significantly larger when the choke is only slightly open while approaching nearly zero as the choke nears full opening. Selecting a representative working point for generating the step response model that encompasses the entire range of the system proves impossible, as proven by the figures.

Nevertheless, considering a 50% choke opening as a midpoint, wherein the response lies approximately in the middle range, offers a pragmatic approach to capturing the overall system dynamics as accurately as possible. Hence, this specific working point was chosen for the choke position, striving to encompass the system's behavior to a reasonable extent. The figures also display steady-state responses for three different gas-lift rates. A gas-lift rate of $7500\frac{m^3}{hr}$ was selected as the working point for the gas-lift rate. Consequently, the step response model was generated by applying a step to the choke position from 50% to 52% while keeping the gas-lift rate constant at $7500\frac{m^3}{hr}$, and applying a step to the gas-lift rate from $7500\frac{m^3}{hr}$ to $7600\frac{m^3}{hr}$ while holding the choke position constant at 50%. The recorded responses were then normalized by the magnitude of the step to obtain a unit-step response.

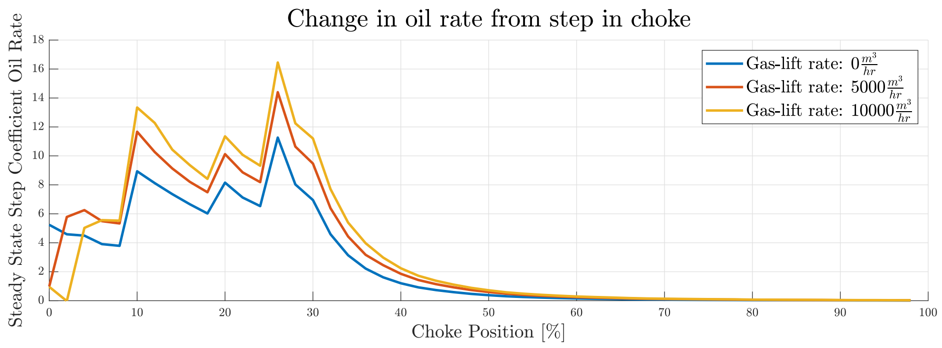

Figure 10: Oil rate steady-state step coefficients from steps in choke position.

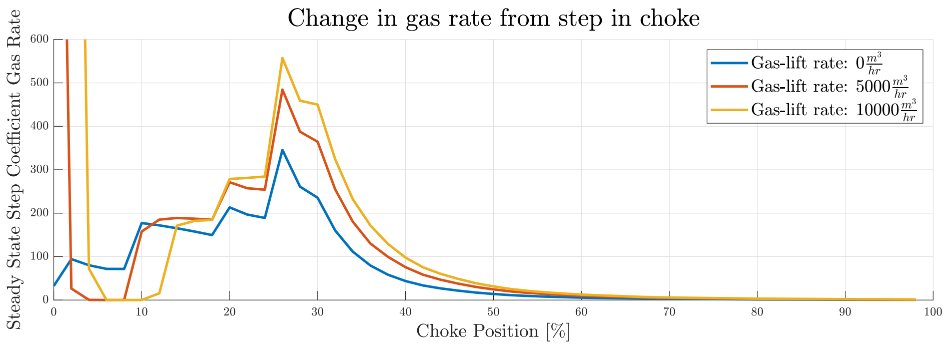

Figure 11: Gas rate steady-state step coefficients from steps in choke position.

5 MPC Implementation

This section provides a summary of the traditional continuous MPC implementation that serves as the foundation for the Mixed Integer MPC. The MPC was developed as part of the project thesis [1], with the MPC formulations mainly based on the work of Philosophia Doctor Dzordzoenyenye Kwame Minde Kufoalor [32].

5.1 SISO MPC Problem Formulation

A general MPC formulation was presented in Section 2.3.3. In this section, a more comprehensive MPC formulation is given. For a plant with a single output y and single input u, the MPC problem can be formulated as:

$$\min \sum_{j=H_w}^{H_p} \|y(k+j|k) - r_y(k+j)\|_{\bar{Q}}^2 + \sum_{j=0}^{H_u-1} \|\Delta u(k+j)\|_{\bar{P}}^2 + \bar{\rho}\bar{\epsilon} + \underline{\rho}\underline{\epsilon} \tag{5.1a}$$

subject to

$$\underline{y} - \underline{\epsilon} \leq y(k+j|k) \leq \bar{y} + \bar{\epsilon}, \quad \bar{\epsilon} \geq 0,\ \underline{\epsilon} \geq 0, \qquad j \in \{H_w, \dots, H_p\}, \tag{5.1b}$$

$$y(k+j|k) = \hat{y}(k+j|k), \qquad j \in \{H_w, \dots, H_p\}, \tag{5.1c}$$

$$\underline{\Delta u} \leq \Delta u(k+j) \leq \overline{\Delta u}, \qquad j \in \{0, \dots, H_u - 1\}, \tag{5.1d}$$

$$\underline{u} \leq u(k+j) \leq \bar{u}, \qquad j \in \{0, \dots, H_u - 1\}, \tag{5.1e}$$

$$u(k+j) = u(k+j-1) + \Delta u(k+j), \qquad j \in \{0, \dots, H_u - 1\}, \tag{5.1f}$$

where k denotes the current time sample, and $k + j$ denotes the future time along the prediction horizon H_p and control horizon H_u. For systems with a large time lag, the prediction horizon might start from H_w, where $H_w > 1$. This is done to avoid penalizing deviations of the output $y(k+j)$ from the reference trajectory r_y for the initial steps, as there is a time lag before the effect of the implemented control action is seen. The input rate limits $\overline{\Delta u}$ and $\underline{\Delta u}$ constrain the rate of input moves. The maximum and minimum allowed inputs are defined by $\bar{u}$ and $\underline{u}$. These bounds are normally defined by the system. Closed loop stability of the MPC problem can be achieved by the choice of $\bar{Q}$ and $\bar{P}$, and the length of the horizons H_p and H_u [32].

The prediction model $\hat{y}(k + j|k)$ can be formulated based on the step response model (2.30) as

$$\hat{y}(k+j|k) = \sum_{i=1}^{j} s(i)\Delta u(k+j-i) + \sum_{i=j+1}^{N-1} s(i)\Delta\tilde{u}(k+j-i) + s(N)\tilde{u}(k+j-N) + v(k+j|k), \tag{5.2}$$

$$v(k+j|k) = v(k|k) = y_m(k) - \hat{y}(k|k-1). \tag{5.3}$$

The past inputs $\tilde{u}(\cdot)$ and input moves $\Delta\tilde{u}(\cdot)$, and the predicted input moves $\Delta u(\cdot)$, are required for the prediction model. Through the constant disturbance model (5.3), output feedback is applied and integral action is introduced. The output measurements at time instant k are denoted $y_m(k)$.

For systems exposed to large disturbances, infeasibility can occur. Therefore, the slack variables $\bar{\epsilon}$, $\underline{\epsilon}$ and their weights $\bar{\rho}, \underline{\rho} > 0$ are added to the optimization problem to avoid infeasibility. The upper and lower bounds on the output, $\bar{y}$ and $\underline{y}$, are then relaxed if the optimization problem becomes infeasible.

5.2 MIMO MPC Problem Formulation

The MPC problem (5.1) can easily be extended to MIMO systems. Additionally, the MPC problem can be converted into a general QP problem compliant with most of the available QP solvers.

The decision variables in (5.1) can be stacked on top of each other in a vector by the following definitions:

$$\Delta U_j(k) = \begin{bmatrix} \Delta u_j(k) \\ \Delta u_j(k+1) \\ \vdots \\ \Delta u_j(k+H_u-1) \end{bmatrix}, \quad U_j(k) = \begin{bmatrix} u_j(k) \\ u_j(k+1) \\ \vdots \\ u_j(k+H_u-1) \end{bmatrix}, \quad Y_i(k) = \begin{bmatrix} y_i(k+H_w|k) \\ y_i(k+H_w+1|k) \\ \vdots \\ y_i(k+H_p|k) \end{bmatrix},$$

$$\mathcal{T}_i(k) = \begin{bmatrix} r_y(k+H_w) \\ r_y(k+H_w+1) \\ \vdots \\ r_y(k+H_p) \end{bmatrix}, \quad Q_i = \begin{bmatrix} \bar{Q} & 0 & \cdots & 0 \\ 0 & \bar{Q} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \bar{Q} \end{bmatrix}, \qquad P_j = \begin{bmatrix} \bar{P} & 0 & \cdots & 0 \\ 0 & \bar{P} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \bar{P} \end{bmatrix},$$

where the subscripts $i = 1, ..., n_{CV}$, and $j = 1, ..., n_{MV}$, where n_{CV} and n_{MV} denotes the number of CVs and MVs in the MIMO system respectively. The dimensions of Q_i and P_j are given by the prediction horizon H_p, the start of the prediction horizon H_w, and the control horizon H_u, where the dimensions are $(H_p - H_w + 1) \times (H_p - H_w + 1)$ and $H_u \times H_u$ respectively. These definitions together with the following, can be used to derive the general QP formulation for a MIMO system.

$$\begin{aligned} \mathcal{T}(k) &= \left[\mathcal{T}_1^\top(k), \quad \mathcal{T}_2^\top(k), \quad \ldots, \quad \mathcal{T}_{n_{CV}}^\top(k),\right]^\top, \\ Y(k) &= \left[Y_1^\top(k), \quad Y_2^\top(k), \quad \ldots, \quad Y_{n_{CV}}^\top(k),\right]^\top, \\ U(k) &= \left[U_1^\top(k), \quad U_2^\top(k), \quad \ldots, \quad U_{n_{MV}}^\top(k),\right]^\top, \\ \Delta U(k) &= \left[\Delta U_1^\top(k), \quad \Delta U_2^\top(k), \quad \ldots, \quad \Delta U_{n_{MV}}^\top(k),\right]^\top, \\ P &= \text{blkdiag}\left(P_1, \quad P_2, \quad ..., \quad P_{n_{MV}}\right), \\ Q &= \text{blkdiag}\left(Q_1, \quad Q_2, \quad ..., \quad Q_{n_{CV}}\right). \end{aligned}$$

A reference trajectory $\mathcal{T}(k)$ can be used to make a gradual transition to the desired setpoint. However, in this implementation, an internal setpoint trajectory is not used, as the plant should be driven to the setpoint as fast as possible. Using the definitions above, the MPC problem for MIMO systems can be formulated as:

$$\min \; Y(k)^\top Q Y(k) + \Delta U(k)^\top P \Delta U(k) - 2\mathcal{T}^\top Q Y(k) + \rho_h^\top \epsilon_h + \rho_l^\top \epsilon_l, \tag{5.4a}$$

subject to

$$GY(k) \leq g + M_h \epsilon_h + M_l \epsilon_l, \quad \epsilon_h \geq 0, \; \epsilon_l \geq 0, \tag{5.4b}$$

$$Y(k) = \Theta \Delta U(k) + \Psi \Delta \tilde{U}(k) + \Upsilon \tilde{U}(k-N) + V(k), \tag{5.4c}$$

$$E \Delta U(k) \leq e, \tag{5.4d}$$

$$FU(k) \leq f, \tag{5.4e}$$

$$KU(k) = \Gamma \tilde{U}(k-1) + \Delta U(k). \tag{5.4f}$$

The MPC problem has been rewritten in matrix form to take on a more general form applicable to most QP solvers. Constant terms not including decision variables are neglected in the cost function, as these do not affect the optimal solution. The output constraints on $Y(k)$ and input constraints $U(k)$ and $\Delta U(k)$ are now matrices and take a more general form. Additionally, the predictions are now computed explicitly by using (5.4c).

The slack variables are stacked in ϵ_h and ϵ_l, where the size corresponds to the number of CVs with high limits ($n_{\bar{y}}$) and low limits ($n_{\underline{y}}$). The corresponding weighting values are stacked similarly in ρ_h and ρ_l.

$$\epsilon_h = \begin{bmatrix}\bar{\epsilon}_1, & \bar{\epsilon}_2, & \dots, & \bar{\epsilon}_{n_{\bar{y}}}\end{bmatrix}^\top, \quad \epsilon_l = \begin{bmatrix}\underline{\epsilon}_1, & \underline{\epsilon}_2, & \dots, & \underline{\epsilon}_{n_{\underline{y}}}\end{bmatrix}^\top.$$

The block-diagonal matrices E, F, G, and the vectors e, f, g are defined as seen below:

$$\begin{aligned} E &= \text{blkdiag}\left(E_1, \quad E_2, \quad \dots, \quad E_{n_{MV}}\right), \quad e = \left[e_1^\top, \quad e_2^\top, \quad \dots, \quad e_{n_{MV}}^\top\right]^\top, \\ F &= \text{blkdiag}\left(F_1, \quad F_2, \quad \dots, \quad F_{n_{MV}}\right), \quad f = \left[f_1^\top, \quad f_2^\top, \quad \dots, \quad f_{n_{MV}}^\top\right]^\top, \\ G &= \text{blkdiag}\left(G_1, \quad G_2, \quad \dots, \quad G_{n_{CV}}\right), \quad g = \left[g_1^\top, \quad g_2^\top, \quad \dots, \quad g_{n_{CV}}^\top\right]^\top, \end{aligned}$$

These matrices are directly derived from the constraints of the problem formulation (5.1):

$$\underbrace{I_{H_u} \otimes \begin{bmatrix}1\\-1\end{bmatrix}}_{E_j} \Delta U_j(k) \leq \underbrace{\mathbf{1} \otimes \begin{bmatrix}\overline{\Delta u}\\-\underline{\Delta u}\end{bmatrix}}_{e_j}, \quad \underbrace{I_{H_u} \otimes \begin{bmatrix}1\\-1\end{bmatrix}}_{F_j} U_j(k) \leq \underbrace{\mathbf{1} \otimes \begin{bmatrix}\overline{u}\\-\underline{u}\end{bmatrix}}_{f_j}, \tag{5.5a}$$

$$\underbrace{I_{H_p-H_w+1} \otimes \begin{bmatrix}1\\-1\end{bmatrix}}_{G_i} Y_i(k) \leq \underbrace{\mathbf{1} \otimes \begin{bmatrix}\overline{y}\\-\underline{y}\end{bmatrix}}_{g_i} + \underbrace{\mathbf{1} \otimes \begin{bmatrix}1\\0\end{bmatrix}}_{\overline{m}_l} \bar{\epsilon}_l + \underbrace{\mathbf{1} \otimes \begin{bmatrix}0\\1\end{bmatrix}}_{\underline{m}_l} \underline{\epsilon}_l. \tag{5.5b}$$

The matrices involving the upper and lower bounds are computed using the Kronecker product $\otimes$. The length of the vector $\mathbf{1}$ is H_u in (5.5a) and $H_p - H_w + 1$ in (5.5b). The matrices M_h and M_l in (5.4b), are defined as follows, where $\bar{m}_l$ and $\underline{m}_l$ are provided from (5.5b):

$$M_h = \text{blkdiag}\big(\overline{m}_1, \quad \overline{m}_2, \quad \dots, \quad \overline{m}_{n_{\bar{y}}}\big), \quad M_l = \text{blkdiag}\big(\underline{m}_1, \quad \underline{m}_2, \quad \dots, \quad \underline{m}_{n_{\underline{y}}}\big).$$

The relationship between $\Delta U(k+j)$ and $U(k+j)$ in (5.4f) are defined by the matrices K and Γ. These matrices result from the transformation from the relationship in (5.1f), and are as follows:

$$K = \text{blkdiag}\big(K_1, \quad K_2, \quad \dots, \quad K_{n_{MV}}\big), \quad \Gamma = \text{blkdiag}\big(\Gamma_1, \quad \Gamma_2, \quad \dots, \quad \Gamma_{n_{MV}}\big),$$

where

$$K_j = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ -1 & 1 & \ddots & 0 & 0 \\ 0 & -1 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 & 1 \end{bmatrix}, \quad \Gamma_j = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \text{using} \quad \tilde{U}(k-1) = \begin{bmatrix} \tilde{u}_1(k-1) \\ \tilde{u}_2(k-1) \\ \vdots \\ \tilde{u}_{n_{MV}}(k-1) \end{bmatrix}.$$

The dimension of matrix K is $(H_u \cdot n_{MV}) \times (H_u \cdot n_{MV})$, and the dimension of Γ is $(H_u \cdot n_{MV}) \times n_{MV}$. The vectors and matrices used in the prediction model (5.4c) are derived from the prediction model (5.2). The vectors are defined as:

$$\Delta\tilde{U}(k) = \left[\Delta\tilde{U}_1^\top(k), \quad \Delta\tilde{U}_2^\top(k), \quad \dots, \quad \Delta\tilde{U}_{n_{MV}}^\top(k)\right]^\top,$$

$$\Delta\tilde{U}_j(k) = \begin{bmatrix} \Delta\tilde{u}(k-1) \\ \Delta\tilde{u}(k-2) \\ \vdots \\ \Delta\tilde{u}(k+H_w-N_j+1) \end{bmatrix}, \quad \tilde{U}(k-N) = \begin{bmatrix} \tilde{u}_1(k+H_w-N_j) \\ \tilde{u}_2(k+H_w-N_j) \\ \vdots \\ \tilde{u}_{n_{MV}}(k+H_w-N_j) \end{bmatrix},$$

$$V(k) = \left[\mathbf{1}^\top v_1(k), \quad \mathbf{1}^\top v_2(k), \quad \dots, \quad \mathbf{1}^\top v_{n_{CV}}(k),\right]^\top.$$

The length of the vector $\mathbf{1}$ is $H_p - H_w + 1$. In the implementation, N_j is assumed to be of the same length for all SISO models, such that $N_j = N$.

The matrices in the prediction model (5.4c) are defined as follows:

$$\Psi = \begin{bmatrix} \Psi_{1,1} & \Psi_{1,2} & \cdots & \Psi_{1,n_{MV}} \\ \Psi_{2,1} & \Psi_{2,2} & \cdots & \Psi_{2,n_{MV}} \\ \vdots & \vdots & \ddots & \vdots \\ \Psi_{n_{CV},1} & \Psi_{n_{CV},2} & \cdots & \Psi_{n_{CV},n_{MV}} \end{bmatrix}_{n_{CV}(H_p-H_w+1)\times\sum_{j=1}^{n_{MV}}(N_j-H_w-1)}$$

$$\Psi_{i,j} = \begin{bmatrix} s(H_w+1) & s(H_w+2) & \cdots & s(N_j-2) & s(N_j-1) \\ s(H_w+2) & s(H_w+3) & \cdots & s(N_j-1) & s(N_j) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ s(H_p+1) & s(H_p+2) & \cdots & s(N_j) & s(N_j) \end{bmatrix}_{(H_p-H_w+1)\times(N_j-H_w-1)}$$

$$\Theta = \begin{bmatrix} \Theta_{1,1} & \Theta_{1,2} & \cdots & \Theta_{1,n_{MV}} \\ \Theta_{2,1} & \Theta_{2,2} & \cdots & \Theta_{2,n_{MV}} \\ \vdots & \vdots & \ddots & \vdots \\ \Theta_{n_{CV},1} & \Theta_{n_{CV},2} & \cdots & \Theta_{n_{CV},n_{MV}} \end{bmatrix}_{n_{CV}(H_p-H_w+1)\times n_{MV}\cdot H_u}$$

$$\Theta_{i,j} = \begin{bmatrix} s(H_w) & s(H_w-1) & \cdots & 0 \\ s(H_w+1) & s(H_w) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ s(H_u) & s(H_u-1) & \cdots & s(1) \\ s(H_u+1) & s(H_u) & \cdots & s(2) \\ \vdots & \vdots & \ddots & \vdots \\ s(H_p) & s(H_p-1) & \cdots & s(H_p-H_u+1) \end{bmatrix}_{(H_p-H_w+1)\times H_u}$$

$$\Upsilon = \begin{bmatrix} \Upsilon_{1,1} & \Upsilon_{1,2} & \cdots & \Upsilon_{1,n_{MV}} \\ \Upsilon_{2,1} & \Upsilon_{2,2} & \cdots & \Upsilon_{2,n_{MV}} \\ \vdots & \vdots & \ddots & \vdots \\ \Upsilon_{n_{CV},1} & \Upsilon_{n_{CV},2} & \cdots & \Upsilon_{n_{CV},n_{MV}} \end{bmatrix}_{n_{CV}(H_p-H_w+1)\times n_{MV}}$$

$$\Upsilon_{i,j} = \begin{bmatrix} s(N_j) \\ s(N_j) \\ \vdots \\ s(N_j) \end{bmatrix}_{(H_p-H_w+1)\times 1}$$

As optimization problem (5.4) is a QP problem, many of the available QP solvers can be used to solve the optimization problem. Most of these solvers require the problem to be on the general QP form. Therefore, the optimization problem (5.4) has to be translated into a general QP problem, as seen in Section 2.1.6. By grouping the decision variables of (5.4), the general QP formulation can be given as:

$$\min \frac{1}{2} \begin{bmatrix} \Delta U(k) \\ U(k) \\ Y(k) \\ \epsilon_h \\ \epsilon_l \end{bmatrix}^\top \begin{bmatrix} 2P & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2Q & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta U(k) \\ U(k) \\ Y(k) \\ \epsilon_h \\ \epsilon_l \end{bmatrix} + \begin{bmatrix} 0 & 0 & -2\mathcal{T}^\top Q & \rho_h^\top & \rho_l^\top \end{bmatrix} \begin{bmatrix} \Delta U(k) \\ U(k) \\ Y(k) \\ \epsilon_h \\ \epsilon_l \end{bmatrix}, \tag{5.6a}$$

subject to

$$\begin{bmatrix} E & 0 & 0 & 0 & 0 \\ 0 & F & 0 & 0 & 0 \\ 0 & 0 & G & -M_h & -M_l \\ 0 & 0 & 0 & -I & 0 \\ 0 & 0 & 0 & 0 & -I \end{bmatrix} \begin{bmatrix} \Delta U(k) \\ U(k) \\ Y(k) \\ \epsilon_h \\ \epsilon_l \end{bmatrix} \leq \begin{bmatrix} e \\ f \\ g \\ 0 \\ 0 \end{bmatrix}, \tag{5.6b}$$

$$\begin{bmatrix} -I & K & 0 & 0 & 0 \\ -\Theta & 0 & I & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta U(k) \\ U(k) \\ Y(k) \\ \epsilon_h \\ \epsilon_l \end{bmatrix} = \begin{bmatrix} \Gamma \tilde{U}(k-1) \\ \Psi \Delta \tilde{U}(k) + \Upsilon \tilde{U}(k-N) + V(k) \end{bmatrix}. \tag{5.6c}$$

The QP problem (5.6) can be compactly represented as:

$$\min_z \frac{1}{2} z^T H_s z + g_s^T z \tag{5.7a}$$

subject to

$$\bar{A}_i z \leq \bar{b}_i, \quad A_e z = b_e, \tag{5.7b}$$

where $z \in \mathbb{R}^n$ is the decision vector $z = \begin{bmatrix} \Delta U(k), & U(k), & Y(k), & \epsilon_h, & \epsilon_l \end{bmatrix}^\top$, and $b_e \in \mathbb{R}^{m_e}$, $\bar{b}_i \in \mathbb{R}^{m_i}$, and $H_s \in \mathbb{R}^{n\times n}$ is a positive semi-definite Hessian. Where n, m_e and m_i is defined as:

$$\begin{aligned} n &= 2 \cdot n_{MV} \cdot H_u + n_{CV}(H_p - H_w + 1) + n_{\bar{y}} + n_{\underline{y}}, \\ m_e &= n_{MV} \cdot H_u + n_{CV}(H_p - H_w + 1), \\ m_i &= 4 \cdot n_{MV} \cdot H_u + (n_{\bar{y}} + n_{\underline{y}})(H_p - H_w + 1) + n_{\bar{y}} + n_{\underline{y}} \end{aligned}$$

The Hessian H_s, the vector g_s, and the vector b_e can be recalculated at each sampling time to update the problem with new information provided by the plant.

Condensed QP formulation
The QP problem implemented in this thesis is a variant of (5.7) where the decision variables Y and U from the QP formulation (5.7) are eliminated from the formulation. This leads to a more condensed QP formulation, where the decision variable $\bar{z} = \begin{bmatrix}\Delta U(k), & \epsilon_h, & \epsilon_l\end{bmatrix}^\top$:

$$\min \left\{ \tfrac{1}{2}\bar{z}^\top H_d \bar{z} + g_d^\top \bar{z} \mid A_d \bar{z} \le b_d,\ \bar{z}_{lb} \le \bar{z} \le \bar{z}_{ub} \right\} \tag{5.8}$$

One of the benefits of this condensed formulation is its lower-dimensional decision vector $\bar{z}$, which allows for more efficient optimization at the cost of less structured or dense QP matrices H_d and A_d.

Equation (5.8) can be derived directly from (5.4) by considering the following definitions:

$$Y(k) := \Lambda_d(k) + \Theta \Delta U(k), \tag{5.9}$$

$$U(k) := K^{-1}[\Gamma \tilde{U}(k-1) + \Delta U(k)], \quad K \succ 0, \tag{5.10}$$

$$\gamma := -2\Theta^\top Q, \tag{5.11}$$

$$\zeta(k) := \mathcal{T}(k) - \Lambda_d(k), \tag{5.12}$$

where

$$\Lambda_d = \Psi \Delta \tilde{U}(k-1) + \Upsilon \tilde{U}(k-1) + V(k). \tag{5.13}$$

This results in

$$\begin{aligned} H_d &= 2 \cdot \mathrm{blkdiag}(\bar{H}_d, 0, 0), \\ g_d(k) &= \begin{bmatrix}\zeta(k)^\top \gamma^\top & \rho_h^\top & \rho_l^\top\end{bmatrix}^\top, \\ A_d &= \begin{bmatrix} G\Theta & -M_h & -M_l \\ FK^{-1} & 0 & 0 \end{bmatrix}, \\ b_d &= \begin{bmatrix} -G\Lambda_d(k) + g \\ -FK^{-1}\Gamma \tilde{U}(k-1) + f \end{bmatrix}, \end{aligned}$$

where $\bar{H}_d = \Theta^\top Q \Theta + P$.

5.3 MPC Algorithm

The compact formulation of the MPC problem (5.8) allows for efficient optimization due to its lower-dimensional decision vector. As a result, this formulation was chosen for implementation in Python.

The process model provided by Equinor exhibits significant nonlinearity in the initial stages due to unnatural values assigned to certain parameters. To mitigate this issue, a simulation of the model is conducted for a brief duration, allowing the process to reach a steady state before commencing the control algorithm. This assumption underlies the implementation of the MPC algorithm, assuming that the process attains a steady state at the initiation of the algorithm. Furthermore, the initial input values are set to match those used to generate the step response models, thereby minimizing model errors during the initial stages. As the step response coefficients remain constant throughout the algorithm, any matrices dependent on these coefficients are constant and predefined before the optimization loop commences.

At each time step, the optimization problem defined by (5.8) is solved, and the first optimal input move is applied to the process. The constraints are also updated at each iteration, taking into account the bias model (5.3), which provides an integral effect in the MPC.

5.4 QP Solver

This thesis does not aim to explore the many different solvers that can be used in the MPC algorithm. Therefore, Gurobi was chosen as the solver for this work. Gurobi is a state-of-the-art optimization software package that is widely used by organizations to solve complex optimization problems.

One of the key advantages of Gurobi is its ability to solve mixed-integer optimization problems efficiently. This attribute renders it a favorable selection for use in a Mixed Integer MPC algorithm due to its compatibility with integer decision variables. Leveraging advanced Branch & Cut algorithms, Gurobi consistently identifies optimal solutions for mixed-integer problems, earning its popularity in applications where such complexities frequently arise.

6 Mixed Integer MPC Implementation

One of the main goals of this thesis is to explore how to integrate the discreteness of a step choke into an MPC problem. This integration aims to enable the controller to appropriately handle the inherent discreteness of the actuator. To achieve this, the thesis proposes the incorporation of integer decision variables into the MPC problem, thereby expanding the conventional continuous MPC into a Mixed Integer MPC. By developing a model predictive controller that directly incorporates integer constraints into the optimization problem, rather than relying on external logic, Equinor can gain valuable insights into the potential of this research field and identify key considerations.

In order to accomplish this objective, a foundational continuous MPC was initially developed as a basis in the project thesis [1], presented in Chapter 5. This chapter presents the extension to Mixed Integer MPC, and additional features added to the MPC formulation. The continuous MPC formulation presented in Chapter 5 is utilized as the fundamental framework for this extension.

6.1 Integer Decision Variables

The fundamental difference between the continuous MPC and the Mixed Integer MPC lies in the decision variables being either continuous or a combination of continuous and discrete. As outlined in Section 3.1, the system under control has two manipulated variables: production choke and gas-lift rate. While the gas-lift rate is continuous, the production choke operates in discrete steps of 2%. This section presents how the discreteness of the choke is incorporated into the MPC problem by introducing integer decision variables and thus extending the continuous MPC to a Mixed Integer MPC.

For convenience, the continuous MPC problem (5.8) is restated here:

$$\min \left\{ \tfrac{1}{2}\bar{z}^\top H_d \bar{z} + g_d^\top \bar{z} \mid A_d \bar{z} \le b_d,\ \bar{z}_{lb} \le \bar{z} \le \bar{z}_{ub} \right\} \tag{6.1}$$

where $\bar{z} = \begin{bmatrix} \Delta U(k), & \epsilon_h, & \epsilon_l \end{bmatrix}^\top$, and $\Delta U(k) = \begin{bmatrix} \Delta U_1(k), & \Delta U_2(k) \end{bmatrix}$. $\Delta U_1(k)$ represents future choke moves and $\Delta U_2(k)$ represents future gas-lift rate changes.

Incorporating integer decision variables into the MPC problem becomes straightforward by employing a solver that supports such integer variables. In this study, Gurobi is utilized as the solver (see [22]), which allows for explicit definition of variables as integers, enabling the enforcement of integer constraints on the decision variables. By including the argument `GRB.INTEGER` when defining the variables, the decision variables are constrained to take on only integer values. In the MIMPC, the choke moves, $\Delta U_1(k)$, are integer-restricted, while the gas-lift rate moves, $\Delta U_2(k)$, remain continuous.

The choke has a step size of $\pm 2\%$ in the system at hand. Thus, the MIMPC should only be allowed to make choke moves of $\pm 2\%$ or no move each timestep. However, defining the integer decision variables only to take on values of $\pm 2\%$ or 0 is not as straightforward. To deal with this, the lower and upper bounds of the choke move variables are set to $\pm 1\%$, as shown in (6.2). Subsequently, the desired choke move is scaled by a factor of 2, when applying the optimal choke move to the plant.

$$\begin{aligned} -1 \le{} & \Delta U_1(k+j) \le 1, \qquad j \in 0, ..., H_u - 1 \\ & \Delta U_1(k+j) \in \mathbb{Z} \end{aligned} \tag{6.2}$$

As the choke move is now restricted to integer values between $[-1, 1]$, the choke position $U_1(k+j)$ also requires scaling by the same factor. Thus, new lower and upper bounds for the choke position are defined as 0% and 50%, respectively, to compensate for the choke move scaling. Additionally, the step response coefficients associated with the choke moves must be scaled by the same scaling factor to ensure the appropriate model in the optimization problem. Therefore, the step coefficients $s_{1,2}(\cdot)$ and $s_{2,2}(\cdot)$, are also scaled by a factor of 2. Consequently, if the optimal choke move is determined to be 1%, it will result in a 2% move in practice, and the same principle applies to a move in the opposite direction.

6.2 Input Blocking

Input parameterization techniques are used in many practical online MPC systems to reduce computational complexity. The most classical technique is called *input blocking*, which introduces a class of input trajectories with a lower degree of freedom in order to reduce the dimensionality of the optimization problem [51]. The manipulated variables are fixed to be constant over a block of sampling instants, requiring fewer decision variables for the same control horizon. The most obvious potential advantage of this technique is that the dimensionality of the problem is greatly reduced, by the elimination of decision variables, while retaining a large control horizon [51].

Even though the dimensionality of the optimization is reduced and fewer decision variables are used in the problem, the performance is usually not affected. Integer programming is much more computationally complex than continuous optimization, and reducing the optimization problem can be crucial to be able to solve it efficiently. Therefore, input blocking has been implemented to solve the MPC problem more efficiently.

The blocking structure implemented can be seen in (6.3), wherein the length of the vector represents the total degrees of freedom, and the elements define the size of the individual blocks. To provide the controller with greater flexibility at the beginning of the control horizon, smaller blocks are employed initially.

$$\text{blocks} = \begin{bmatrix}1, & 1, & 1, & 1, & 2, & 2, & 4, & 4, & 8, & 8, & 8, & 16, & 16, & 32, & 32, & 64\end{bmatrix} \tag{6.3}$$

Implementing input blocking in the MPC, with the blocking structure (6.3), resulted in a significant reduction in the degrees of freedom, decreasing from 200 (H_u) to 16. This reduction in the problem dimensionality leads to a more efficient optimization problem and a faster solver. The objective of a well-designed blocking structure is to minimize computational time without compromising the performance of the MPC.

For a simulation of 3000 runs with the continuous MPC used for control, it was observed that the introduction of input blocking resulted in a remarkable reduction in simulation time. The computation time decreased from 29 minutes to just 4 minutes, translating to less than 0.1 seconds per iteration. Apart from the computational time, it was not possible to distinguish the performance of the MPC with and without input blocking. It is worth noting that the blocking structure employed in this study could have been further adjusted to include fewer blocks, potentially improving computational efficiency even more. For instance, in SEPTIC, only 6 blocks are utilized. Considering the satisfactory computational time achieved with the current blocking structure, there was no immediate focus placed on further reducing it in order to prevent any potential degradation in performance.

6.3 Bias Filtering

In the system at hand, the plant is a model and not a real process. Thus measurement noise is not an issue for this system. However, in a real-world process, measurement noise is typically present and needs to be addressed. To filter out high-frequency measurement noise, a simple lowpass filter is often sufficient [52].

Sudden spikes in the bias update can occur, which can cause oscillations in the system. Therefore, to ensure smooth control and avoid possible oscillations, the bias term (2.31b) is filtered using the simple lowpass filter (6.4). The filter employs an array that saves the last H_p biases, and for each sampling, the array is shifted, and the new current bias term is saved and lowpass filtered.

The lowpass filter is implemented using the following equation:

$$y_k = \left(\frac{2 - \Delta t w_c}{2 + \Delta t w_c}\right) y_{k-1} + \left(\frac{\Delta t w_c}{2 + \Delta t w_c}\right)(x_k + x_{k-1}), \tag{6.4}$$

where x_k is the actual bias at time $t = k\Delta t$, y_k is the filtered bias at time $t = k\Delta t$, and w_c is the cut-off frequency. In the MPC algorithm, w_c is set to $\frac{1}{20}$ [rad/s].

6.4 Soft MPC

Despite the widespread use of MPC in high-level control systems, there is limited guidance available on tuning methodologies of MPC controllers in the face of the inevitable plant model mismatch [7]. The closed-loop performance of linear MPC can be particularly poor in the presence of model uncertainty, especially when integrality constraints are imposed on the manipulated variables with a large step size. In such cases, oscillations can occur, fine regulation becomes challenging, and the impact of plant model mismatch becomes evident.

During the testing phase of the implemented MIMPC, it was noted that the robustness of the controller exhibited variations across different regions of the manipulated variables. Specifically, when the choke was nearly closed, the performance of the controller was notably influenced by plant model mismatch. The discrepancy between the predicted and actual responses became more pronounced, particularly considering the relatively large step size of the choke. This emphasized the impact of plant model mismatch on the overall performance of the controller.

While model mismatch can be mitigated to some extent through feedback mechanisms, the challenge becomes more significant when integrality constraints are imposed, and the actuator step size is large. In regions where plant model mismatch is prominent, a 2% step can result in a response that deviates greatly from the predictions and can, i.e., lead to a large and unpredicted overshoot. Consequently, the controller must compensate by reducing the manipulated variable by 2% back to the same position. This process can lead to oscillations in the inputs, ultimately manifesting as oscillations in the controlled variables.

To address the issue of oscillations in the inputs caused by plant model mismatch, the Soft MPC approach is employed in this study. This method aims to improve controller performance and mitigate oscillations by relaxing the penalization of the controller when the controlled variables are close to the reference. This is achieved by introducing soft output constraints that are relative to the set points. By creating a "deadzone" around the set point, oscillations in the controlled variables caused by plant model mismatch can be eliminated. Figure 12 illustrates the measured oil rate and choke position for both the nominal MIMPC and the MIMPC utilizing the Soft MPC method. The thin dashed lines represent the deadzone around the set point. It is evident that the nominal controller exhibits oscillations around the set point, corresponding to the oscillatory behavior of the choke position. In contrast, the Soft MPC approach avoids oscillations as the deviation from the set point is penalized to a lesser extent within the established deadzone. It should be noted that the gas rate and gas-lift rate are excluded from the figure as the figure serve solely to illustrate the Soft MPC method. The focus is on demonstrating the impact of the Soft MPC method, highlighting the effectiveness of the penalty deadzone in mitigating oscillations in the controlled variables caused by plant model mismatch.

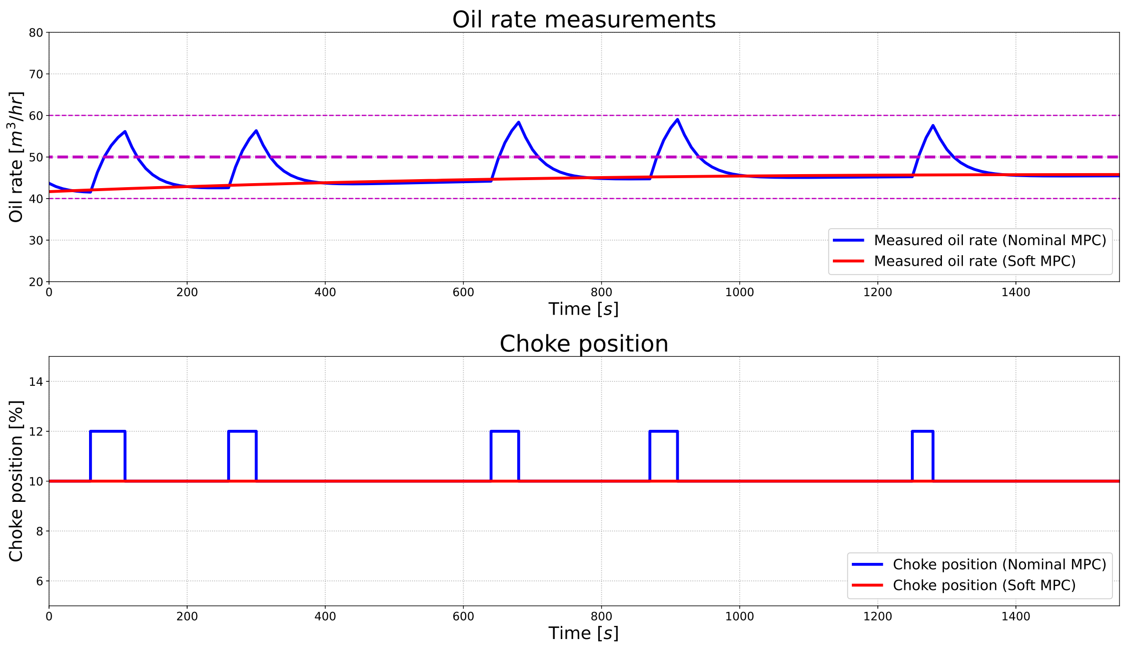

Figure 12: Comparison of nominal MIMPC (blue) and Soft MIMPC (red)

To provide an intuitive understanding of the Soft MPC formulation, it is first presented for a SISO system based on the MPC formulation given in (5.1). The Soft MPC formulations are based on the work of Guru Prasath and John Bagterp Jørgensen [7].

To incorporate the Soft MPC method, the optimization problem is augmented with the following soft constraints:

$$y(k+j) \leq (r_y(k+j)+c)+\eta(k+j), \quad j \in \{H_w,\dots,H_p\} \tag{6.5a}$$
$$y(k+j) \geq (r_y(k+j)-c)-\eta(k+j), \quad j \in \{H_w,\dots,H_p\} \tag{6.5b}$$
$$\eta(k+j) \geq 0, \quad j \in \{H_w,\dots,H_p\} \tag{6.5c}$$

$y(k+j)$ is the predicted output, $r_y(k+j)$ is the output reference, c is a positive constant defining the size of the deadzone, and $\eta(k+j)$ denotes the slack variables introduced to the optimization problem. Moreover, the objective function (5.1) is modified to incorporate the slack variables $\eta(k+j)$ as follows:

$$\min \sum_{j=H_w}^{H_p} \|y(k+j|k)-r_y(k+j)\|_Q^2+\sum_{j=0}^{H_u-1} \|\Delta u(k+j)\|_P^2+\bar{\rho}\bar{\epsilon}+\underline{\rho}\underline{\epsilon}+\sum_{j=H_w}^{H_p} \|\eta(k+j)\|_{S_\eta}^2+s_\eta'\eta(k+j) \tag{6.6}$$

The Soft MPC approach introduces a penalty function similar to the one depicted in Figure 13. This penalty function illustrates that when the controlled variable is close to the reference, it incurs a lower penalty compared to the nominal case. However, if it falls outside the deadzone region, it is penalized similarly to the nominal case. It is important to note that the penalty function shown in the figure is not exactly the same as the implemented Soft MPC penalty function since it relies on the tuning parameters. Nevertheless, it offers a visual representation of how the Soft MPC method works. Inside the deadzone, the penalty function is nonzero, indicating that the controller still aims to drive the controlled variable towards the reference but with a different weighting between the input move and the reference tracking.

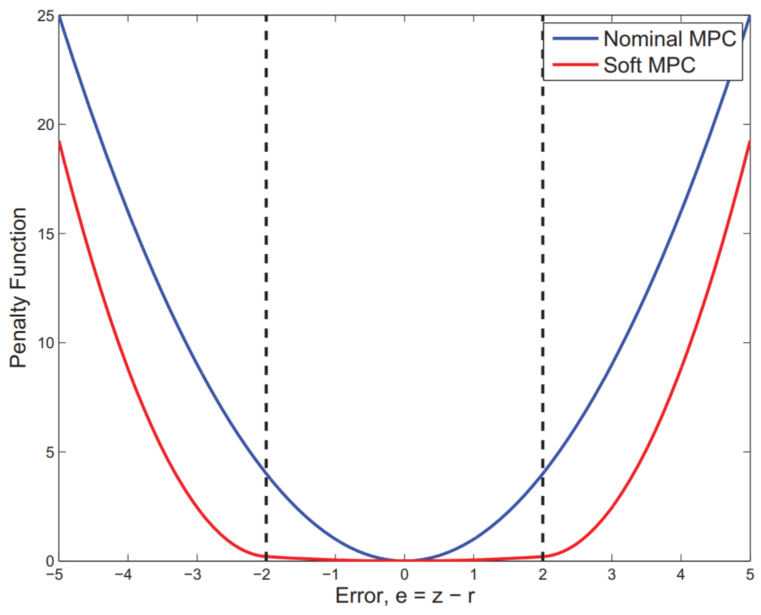

Figure 13: Penalty function for Soft MPC and nominal MPC [7]

The implementation of Soft MPC in the MIMPC system is based on the MPC formulations described in Section 5, and is repeated here for convenience:

$$\min \left\{ \tfrac{1}{2}\bar{z}^\top H_d \bar{z} + g_d^\top \bar{z} \mid A_d \bar{z} \le b_d,\ \bar{z}_{lb} \le \bar{z} \le \bar{z}_{ub} \right\}$$

$$\begin{aligned} H_d &= 2 \cdot \mathrm{blkdiag}(\bar{H}_d, 0, 0), \\ g_d(k) &= \begin{bmatrix} \zeta(k)^\top \gamma^\top & \rho_h^\top & \rho_l^\top \end{bmatrix}^\top, \\ A_d &= \begin{bmatrix} G\Theta & -M_h & -M_l \\ FK^{-1} & 0 & 0 \end{bmatrix}, \\ b_d &= \begin{bmatrix} -G\Lambda_d(k) + g \\ -FK^{-1}\Gamma\tilde{U}(k-1) + f \end{bmatrix}, \end{aligned}$$

To include the Soft MPC feature, the decision variable vector $\bar{z}$ is extended to include the slack variables $\eta(k)$:

$$\bar{z} = \begin{bmatrix} \Delta U(k), & \epsilon_h, & \epsilon_l, & \eta(k) \end{bmatrix}^\top, \tag{6.8}$$

where

$$\eta(k) = \begin{bmatrix} \eta_1(k+H_w) \\ \eta_1(k+H_w+1) \\ \vdots \\ \eta_1(k+H_p) \\ \eta_2(k+H_w) \\ \eta_2(k+H_w+1) \\ \vdots \\ \eta_2(k+H_p) \end{bmatrix}. \tag{6.9}$$

$\eta(k)$ is a vector of slack variables that represent the deviation from the deadzone region. The Hessian-matrix H_d, and the linear penalty vector g_d are enlarged to include the soft weighting parameters $\mathcal{S}_\eta(k)$ and $s_\eta(k)$.

$$H_{d,soft} = \begin{bmatrix} H_d & 0 \\ 0 & \mathcal{S}_\eta(k) \end{bmatrix}, \tag{6.10}$$

where

$$\mathcal{S}_\eta(k) = \begin{bmatrix} \mathcal{S}_{gas} & 0 \\ 0 & \mathcal{S}_{oil} \end{bmatrix}, \quad \mathcal{S}_{gas} = \begin{bmatrix} S_{gas} & 0 & \cdots & 0 \\ 0 & S_{gas} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & S_{gas} \end{bmatrix}, \quad \mathcal{S}_{oil} = \begin{bmatrix} S_{oil} & 0 & \cdots & 0 \\ 0 & S_{oil} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & S_{oil} \end{bmatrix},$$

$$g_d(k) = \begin{bmatrix} \zeta(k)^\top \gamma^\top & \rho_h^\top & \rho_l^\top & s_{\eta,gas}(k)^\top & s_{\eta,gas}(k)^\top \end{bmatrix}, \tag{6.11}$$

where

$$s_{\eta,gas}(k) = \begin{bmatrix} s_{gas}, & s_{gas}, & \dots & s_{gas} \end{bmatrix}^\top, \quad s_{\eta,oil}(k) = \begin{bmatrix} s_{oil}, & s_{oil}, & \dots & s_{oil} \end{bmatrix}^\top.$$

Here $\mathcal{S}_\eta(k)$ is a block diagonal matrix with the soft penalty weights S_{gas} and S_{oil} as its diagonal elements. $s_{\eta,gas}(k)$ and $s_{\eta,oil}(k)$ are vectors that define the linear soft penalty weights.

In the case of the MIMO system, the soft constraints can be written as:

$$\Theta \Delta U(k) + \Lambda_d(k) \leq T(k) + \boldsymbol{c} + \eta(k), \tag{6.12a}$$
$$\Theta \Delta U(k) + \Lambda_d(k) \geq T(k) - \boldsymbol{c} - \eta(k), \tag{6.12b}$$

where $\boldsymbol{c}$ is a vector defining the size of the gas and oil deadzone, with dimension $2(H_p - H_w + 1) \times 1$:

$$\boldsymbol{c} = \begin{bmatrix} c_{gas}, & c_{gas}, & \dots & c_{gas}, & c_{oil}, & c_{oil}, & \dots & c_{oil} \end{bmatrix}^\top. \tag{6.13}$$

Furthermore, the matrices A_d and b_d are extended to include the soft constraints. The extended matrices $A_{d,soft}$ and $b_{d,soft}$ become:

$$A_{d,soft} = \begin{bmatrix} G\Theta & -M_h & -M_l & 0 & 0 \\ FK^{-1} & 0 & 0 & 0 & 0 \\ \Theta & 0 & 0 & -I & -I \\ -\Theta & 0 & 0 & -I & -I \end{bmatrix}, \tag{6.14}$$

$$b_{d,soft} = \begin{bmatrix} -G\Lambda_d(k) + g \\ -FK^{-1}\Gamma\tilde{U}(k-1) + f \\ -\Lambda_d(k) + T(k) + \boldsymbol{c} \\ \Lambda_d(k) - T(k) + \boldsymbol{c} \end{bmatrix}. \tag{6.15}$$

In summary, the soft MPC implementation in the MIMPC system extends the decision variable vector, Hessian matrix, and linear penalty vector to include the slack variables and soft weighting parameters. The matrices A_d and b_d are also augmented to incorporate the soft constraints related to the deadzone region.

6.5 Oscilliation constraints

Oscillations can negatively affect control performance and may even cause damage to actuators. The introduction of integer decision variables in the MIMPC helps reduce the occurrence of oscillations in the manipulated variables by allowing the controller to predict the discrete behavior and adjust accordingly. Additionally, as seen in the previous section, the Soft MPC method helps mitigate oscillations. However, to further mitigate this risk of oscillations in the system, constraints can be implemented to prevent the MIMPC from rapidly oscillating the manipulated variables.

By imposing constraints that prohibit fast oscillations, the controller is not allowed to reverse the direction of the manipulated variable immediately after making a move. For instance, if the controller opens the choke in one time step, these constraints would prevent the controller from moving the choke toward a closed position in the following time step. This ensures that the actuator is given sufficient time to stabilize and settle into the new position, reducing the likelihood of oscillations and minimizing the potential for damage. Ultimately, this can extend the lifetime of the actuator.

It is worth noting that such constraints may introduce a one time step delay, but the impact on the overall performance is negligible in the context of the larger system. By including the constraints in (6.16) into the optimization problem, sudden changes in the choke move direction are restricted. As a result, the controller operates in a more predictable and controlled manner, leading to increased safety.

$$
\begin{aligned}
\Delta u_1(k) &\geq \frac{\Delta \tilde{u}_1(k-1)}{2} - 1, \\
\Delta u_1(k) &\leq \frac{\Delta \tilde{u}_1(k-1)}{2} + 1, \\
\Delta u_1(k+j) &\geq \Delta u_1(k+j-1) - 1, \quad j \in \{1, \dots, H_u - 1\} \\
\Delta u_1(k+j) &\leq \Delta u_1(k+j-1) + 1, \quad j \in \{1, \dots, H_u - 1\}
\end{aligned}
\tag{6.16}
$$

where $\Delta \tilde{u}_1(k-1)$ is the previous choke move, $\Delta u_1(\cdot)$ are the predicted choke moves, and the step size of the actuator is 2%. It should be noted that this formulation assumes that the predicted choke moves are scaled according to what was described in Section 6.1, meaning in short that $\Delta \tilde{u}_1(k-1) \in \{-2, 0, 2\}$ and $\Delta u_1 \in \{-1, 0, 1\}$.

In this study, Gurobi is used to solve the optimization problem. To solve integer programs, this solver utilizes among others the branch and cut algorithm, which relies on a series of linear programming relaxations. In Gurobi, an integer solution is deemed an integer if the variables are within a tolerance value (*IntFeasTol*) of an integer. The default value of *IntFeasTol* is 1^{-5}, meaning that e.g. 1.000000465 would be considered an integer. In some cases, such as when the constraints (6.16) are incorporated in the optimization problem, this tolerance could have a catastrophic outcome if not handled.

If looking at the case where the previous choke move $\Delta \tilde{u}_1(k-1) = -2.000000465$ which is deemed an integer by the solver, by the constraints (6.16), the next choke move would be constrained to:

$$
\begin{aligned}
\Delta u_1(k) \qquad &\geq \frac{-2.000000465}{2} - 1, \\
\Delta u_1(k) \qquad &\leq \frac{-2.000000465}{2} + 1,
\end{aligned}
$$

thus

$$
-2.000000233 \leq \Delta u_1(k) \leq -0.000000233.
$$

Given that $\Delta u_1(k)$ is already constrained to $\Delta u_1(k) \geq -1$, the range of feasible solutions become $-1 \leq \Delta u_1(k) \leq -0.000000233$. In this range, the only integer solution is $\Delta u_1(k) = -1$. However, if the current choke position $\tilde{u}_1(k-1) = 0\%$, this would result in an infeasible optimization problem as the choke position is constrained to be greater than or equal to zero, and a negative choke move would violate this constraint.

In such scenarios, if deemed necessary, the Gurobi developers recommend applying variable rounding due to the variables already being within the small default tolerance of Gurobi. Since Gurobi does not offer a built-in rounding feature, any necessary rounding should be done in post-processing after completing the optimization [53].

Applying post-processing rounding to the same case mentioned above, the next choke move would be constrained to:

$$\begin{aligned}\Delta u_1(k) &\geq -1\\ \Delta u_1(k) &\leq 0\end{aligned}$$

Thus, in this scenario, the solution $\Delta u_1(k) = 0$ is feasible, and the controller is constrained not to make any choke moves during this particular time step.

6.6 Mean Choke Move

To mitigate the risks associated with sand production and ensure the long-term sustainability of the oil well, it is important to control the rate at which the production choke is opened. Opening the choke too fast can lead to undesirable consequences such as reservoir compaction and flow path blockage caused by sand collapse [54]. In response to these concerns, Equinor has requested a specific feature in the MPC algorithm to enforce a mean choke move constraint.

Equinor has set a requirement to limit the average pace of opening the choke in the oil well to a rate not exceeding $0.55\%/\Delta t$, with $\Delta t = 10s$. To ensure compliance with this constraint, the implemented MPC controller includes specific measures to prevent excessively fast choke opening. The controller incorporates a constraint on the average choke move over a 250-second interval. If the average choke move surpasses the specified limit, the controller enforces a hard constraint, restricting further choke movement in the same direction, and allowing adjustments only to the gas-lift. This constraint remains in effect until the average choke move falls below the predefined limit.

It is important to note that the constraint limits the average speed of opening the choke to $0.55\%/\Delta t$, while the actual choke move per time step is subject to a separate constraint of the choke step size, set at 2%. This distinction ensures that the MPC algorithm maintains control over the choke opening within the defined limits while adhering to the specific choke step size.

By imposing this constraint, the MPC algorithm ensures that the choke opening changes gradually over time. This controlled rate of change enables better management of sand production and minimizes the associated risks. The mean choke move constraint strikes a balance between optimizing production rates and mitigating sand-related reservoir damage. By controlling the choke opening within specified limits, the MPC algorithm promotes the long-term sustainability and productivity of the oil well while safeguarding against the adverse effects of sand production [54].

6.7 Real Time Optimization

The initial articles on MPC introduced the concept of output error $\|y - y_{ref}\|^2$ and input move suppression $\|\Delta u - \Delta u_{ref}\|^2$ in the objective function [55]. Notably, the objective function did not incorporate any input error term $\|u - u_{ref}\|^2$, which is also the case for the implemented controller in this study, as seen in (2.25). Traditionally, optimal control strategies in linear systems have included a cost penalty for input variables in the objective function, leading to steady-state offsets. However, Cutler and Ramaker solved this problem by posing the optimization problem in terms of changes in control moves, incorporating integral action in the controller, and eliminating steady-state offsets [55].

In many applications where the number of inputs is equal or less than the number of outputs, achieving setpoints on both the inputs and the outputs can be advantageous. In such cases, Real Time Optimization (RTO) can be employed to calculate the target values for both output and input variables [56]. In most cases, an RTO application functions by optimizing the process operating conditions and updating set points for local MPCs. Typically a two-layered structure is employed in process plants such as oil and gas production systems, which facilitates the achievement of economically optimal operation. The upper layer focuses on optimizing the plant's steady-state operation, drawing from a stationary plant model to provide set points for the lower-layer MPC [57].

In this study, an RTO has been developed; however, not utilized in the final implementation due to a specific request from Equinor to exclude this from the implementation. Although it is not part of the final implementation, the RTO is presented here as it is considered beneficial in a broader context and enables the inclusion of several additional features in the system, such as the avoidance of defined regions in the input space.

6.7.1 Ideal MVs

By utilizing RTO it can provide setpoints for both the inputs and outputs. It can be seen as a steady-state optimization, not accounting for the dynamics of the system. There are multiple advantages to using RTO in a layer above the MPC. Firstly, it ensures that the set points are feasible for the system. This is especially useful in underactuated systems, where many setpoints are not attainable. In this project, the system under control has equally many MVs as CVs, and it is not always possible to reach the set point of both CVs. Then, RTO can make small changes to the desired set points and provide the best feasible setpoints.

Secondly, adding RTO to the control scheme facilitates for adding special features, i.e., if a region in the input should be avoided due to a higher risk of damage to the equipment in this region. The RTO can thus handle this in an effective way by providing setpoints for the inputs, avoiding the region where possible damage can occur.

In the system under control, it is advantageous not to use gas-lift to control when the gas-lift rate is between $0 - 2000\frac{m^3}{hr}$. This is due to the increased risk of damage to the well, as gas-lift rate in this region can lead to slugging in the well. Even though this region should be avoided, the controller has to go through this region to gain a gas-lift rate greater than $2000\frac{m^3}{hr}$. This can be accomplished by utilizing RTO, constraining the steady-state gas-lift rate to avoid this region, and using the optimal steady-state gas-lift rate as a reference in the MPC implementation. As gas is pumped down into the well when using the gas-lift method, this comes with a cost, as the gas has to be compressed and sent back into the well. Thus, the economic aspect can also be included in the RTO, such that it comes with a cost to use gas-lift. An example of an RTO problem is stated in (6.19).

$$\underset{z, Y_{gas}, Y_{oil}, u_{choke}, u_{gl}}{\text{minimize}} \quad -\alpha_{gas} Y_{gas} - \alpha_{oil} Y_{oil} + \alpha_{gl} u_{gl}, \tag{6.19a}$$

subject to

$$\begin{bmatrix} Y_{oil} \\ Y_{gas} \end{bmatrix} = f(u_{choke}, u_{gl}) \tag{6.19b}$$

$$\underline{y}_{gas} \leq Y_{gas} \leq \overline{y}_{gas} \tag{6.19c}$$

$$\underline{y}_{oil} \leq Y_{oil} \leq \overline{y}_{oil} \tag{6.19d}$$

$$z\underline{u}_{gl} \leq u_{gl} \leq z\overline{u}_{gl} \tag{6.19e}$$

The goal of this RTO is to maximize the total profit from gas and oil production, given by α_{gas} and α_{oil}. Additionally, the cost α_{gl} of using gas-lift as a lifting method should be taken into account. The RTO uses a steady-state model, $f(u_{choke}, u_{gl})$, to calculate the steady-state oil rate and gas rate, Y_{oil} and Y_{gas}. Additionally, the gas and oil rate is constrained to a minimum and maximum rate. The gas-lift rate, u_{gl}, is constrained to $\underline{u}_{gl} \leq u_{gl} \leq \overline{u}_{gl}$ or 0, by including the binary variable z. When $z = 1$, the gas-lift is constrained to $\underline{u}_{gl} \leq u_{gl} \leq \overline{u}_{gl}$, but when $z = 0$ the gas-lift is constrained to $u_{gl} = 0$. The optimal steady states of the gas rate, oil rate, choke position, and gas-lift rate can be provided to the MPC and used as setpoints in the controller, ensuring the setpoints of the MPC are feasible, economical, and that the gas-lift rate is not in the damaging region.

To use the input setpoints from the RTO, the MPC objective function needs to be changed to (6.20) to incorporate the setpoints of the inputs and penalize deviation.

$$J_{new} = J + J_u \tag{6.20}$$

where

$$\begin{aligned}
J_u &= \left(U(k) - U_{RTO}(k)\right)^\top Q \left(U(k) - U_{RTO}(k)\right) && (6.21a)\\
&= U(k)^\top QU(k) - 2U_{RTO}(k)QU(k) + U_{RTO}(k)^\top QU_{RTO}(k) && (6.21b)\\
&= \left(K^{-1}\left[\Gamma\tilde{U}(k-1) + \Delta U(k)\right]\right)^\top Q \left(K^{-1}\left[\Gamma\tilde{U}(k-1) + \Delta U(k)\right]\right) && (6.21c)\\
&\quad - 2U_{RTO}(k)Q\left(K^{-}1\left[\Gamma\tilde{U}(k-1) + \Delta U(k)\right]\right) + U_{RTO}(k)\top QU_{RTO}(k) &&\\
&= \Delta U(k)^\top \left(K^{-\top}QK^{-1}\right)\Delta U(k) - \left(U_{RTO}(k)QK^{-1} - 2K^{-1}\Gamma\tilde{U}(k-1)QK^{-1}\right)\Delta U(k) && (6.21d)
\end{aligned}$$

7 Results

This chapter presents the results obtained from simulating the gas-lifted oil production well, described in Section 3.1, utilizing the MPCs developed in this study. This study has aimed to explore how to incorporate the discreteness of a step choke into the MPC optimization problem and evaluate the performance of a Mixed Integer MPC compared to a continuous MPC. These findings show the advantages of incorporating discrete decision variables in the MPC and highlight the challenges of discrete actuators in a continuous control system. The term "continuous MPC" refers to the MPC detailed in Chapter 5, signifying that the optimization variables are continuous. The term "MIMPC" is used when utilizing the MIMPC described in Chapter 6. In this case, the decision variables corresponding to the step choke are integer decision variables, while the decision variables corresponding to the gas-lift rate remain continuous.

The results are presented in two parts. First, the performance of the continuous MPC is evaluated under two conditions: continuous and step choke. These findings highlight the complexities associated with continuous control of discrete actuators and demonstrate the limitations of the continuous MPC in this scenario.

Secondly, the performance of the MIMPC is presented, where integrality constraints are incorporated directly into the optimization problem. This integration provides the controller with knowledge of the discreteness of the choke, enabling it to predict the system's behavior while considering the choke's discrete nature. The simulations are conducted using identical reference trajectories for both controllers to facilitate a comprehensive comparison between the two methods.

To facilitate for comparison of the different control methods, the same tuning parameters are used for all simulations unless otherwise noted. The tuning parameters are given in Table 1. Additionally, the Soft MPC technique, input blocking, bias filtering, and mean choke move presented in Chapter 6 are used in both controller formulations.

Parameter	**Value**	**Description**
$\bar{P}_{choke}$	1000	Choke move penalty
$\bar{P}_{gas-lift}$	0.2	Gas-lift rate move penalty
$\bar{Q}_{gas}$	0.0001	Gas rate deviation penalty
$\bar{Q}_{oil}$	0.1	Oil rate deviation penalty
S_{gas}	0.0001	Soft MPC quadratic gas rate penalty
S_{oil}	0.1	Soft MPC quadratic oil rate penalty
s_{gas}	0.0001	Soft MPC linear gas rate penalty
s_{oil}	0.1	Soft MPC linear oil rate penalty
c_{gas}	$1000\frac{m^3}{hr}$	Deadzone gas rate
c_{oil}	$5\frac{m^3}{hr}$	Deadzone oil rate
$\bar{y}_{gas}$	$20000\frac{m^3}{hr}$	Soft upper bound gas rate
$\underline{y}_{gas}$	$0\frac{m^3}{hr}$	Soft lower bound gas rate
$\bar{y}_{oil}$	$400\frac{m^3}{hr}$	Soft upper bound oil rate
$\underline{y}_{oil}$	$0\frac{m^3}{hr}$	Soft lower bound oil rate
$\bar{\rho}, \underline{\rho}$	1000	Slack variable weight
H_p	250	Prediction horizon
H_u	200	Control horizon

Table 1: Tuning parameters.

Moreover, this chapter demonstrates how the MIMPC utilizes the prediction model to determine optimal inputs, incorporating re-optimization and feedback mechanisms to achieve precise control. Additionally, the findings highlight the controller's ability to handle disturbances and ensure accurate regulation of a single controlled variable.

7.1 Continuous MPC

This section presents the performance of the continuous MPC, presented in Chapter 5. The initial test is designed as a benchmark evaluation, with a continuous choke configuration employed to establish a baseline for further evaluations. This test is made feasible as the plant is a mathematical model instead of a physical system, making it possible to replace the step choke with a continuous choke. Additionally, the section presents the performance of the continuous MPC with the original step choke, highlighting the challenges of continuous control of discrete actuators. The simulations are run for $t = 78000s$, equivalent to approximately 22 hours, with $\Delta t = 10s$. The results obtained from these simulations serve as a basis for evaluating the performance of the MIMPC.

7.1.1 Continuous Choke

In this simulation, the continuous MPC is utilized to control the system where both the choke and the gas-lift rate are continuous. The controller is expected to produce optimal inputs that steer the controlled variables toward their respective reference trajectories. In Figure 14, the controlled variables are plotted against their reference trajectories. The figure shows that the controlled variables converge to the reference trajectories for most of the simulation period, exhibiting a fast, smooth, and stable response.

It is evident that, for some of the setpoints, both controlled variables exhibit a stationary deviation. Given that the system has an equal number of inputs and outputs, where both inputs affect both outputs, it may not always be feasible to achieve both setpoints simultaneously. Following discussions with Equinor, it was decided to prioritize the attainment of the oil rate setpoint over the gas rate setpoint. Consequently, a higher penalty is applied for the deviation of the oil rate than that of the gas rate in the objective function, as seen in Table 1. However, the gas rate should still converge to the setpoint whenever possible. In situations where it is not possible to attain both setpoints, the deviations from the setpoints are determined by the tuning parameters.

Figure 15 displays the choke position and gas-lift rate throughout the simulation. Despite a stationary deviation in the controlled variables for specific reference values, the controller converges to an optimal stationary point that minimizes the objective function in the MPC. Toward the end of the simulation, both inputs are saturated, and thus these setpoints are not feasible. Notably, the actual choke position tracks the optimal desired choke position from the MPC, as the choke is continuous in this scenario.

The mean choke move constraints, presented in section 6.6, impose limitations on the average speed of the choke movement. This effect is clearly depicted in Figure 15, where the choke position exhibits a staircase-like trajectory following from significant steps in the reference trajectory.

Figure 16 illustrates the bias, $v(k)$, through the simulation. It is evident that the bias exhibits a significant magnitude. However, an important observation is that the bias stabilizes at a steady-state following a reference step, which is effectively compensated for through feedback.

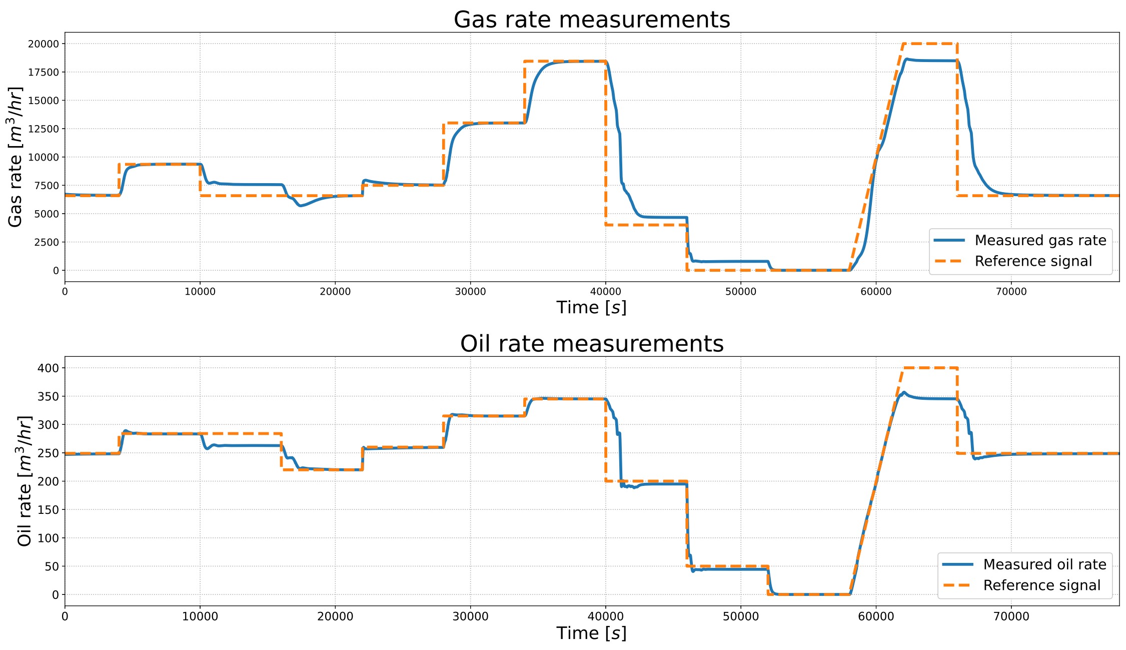

Figure 14: Oil and gas rate measurements when using the continuous MPC with continuous choke for control.

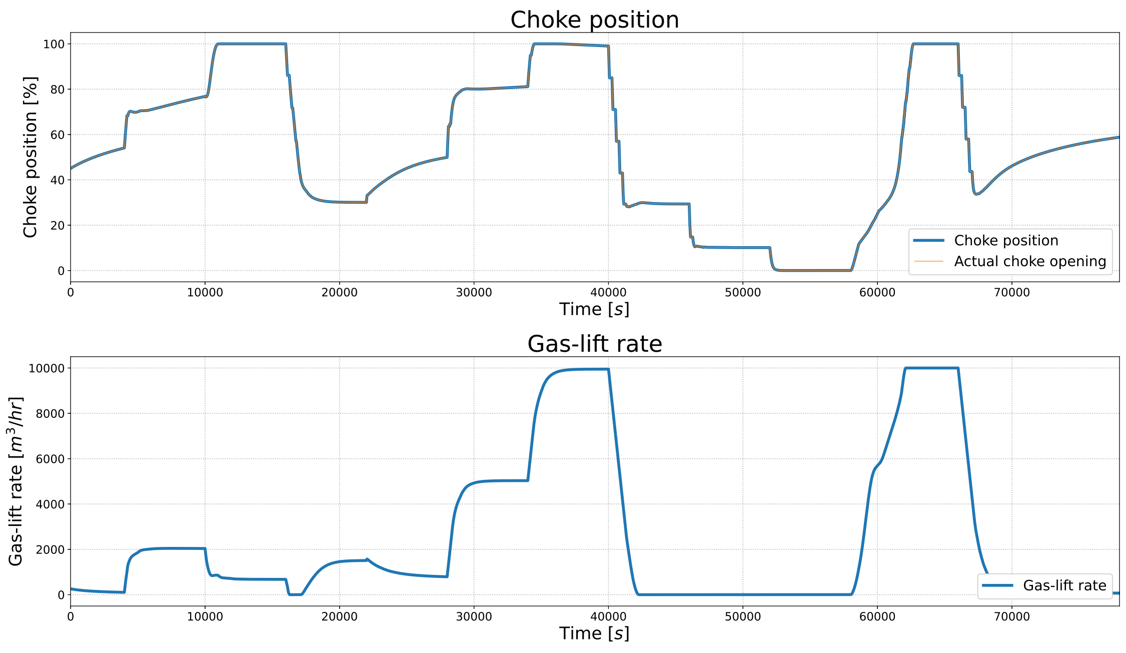

Figure 15: Choke position and gas-lift rate when using the continuous MPC with continuous choke for control.

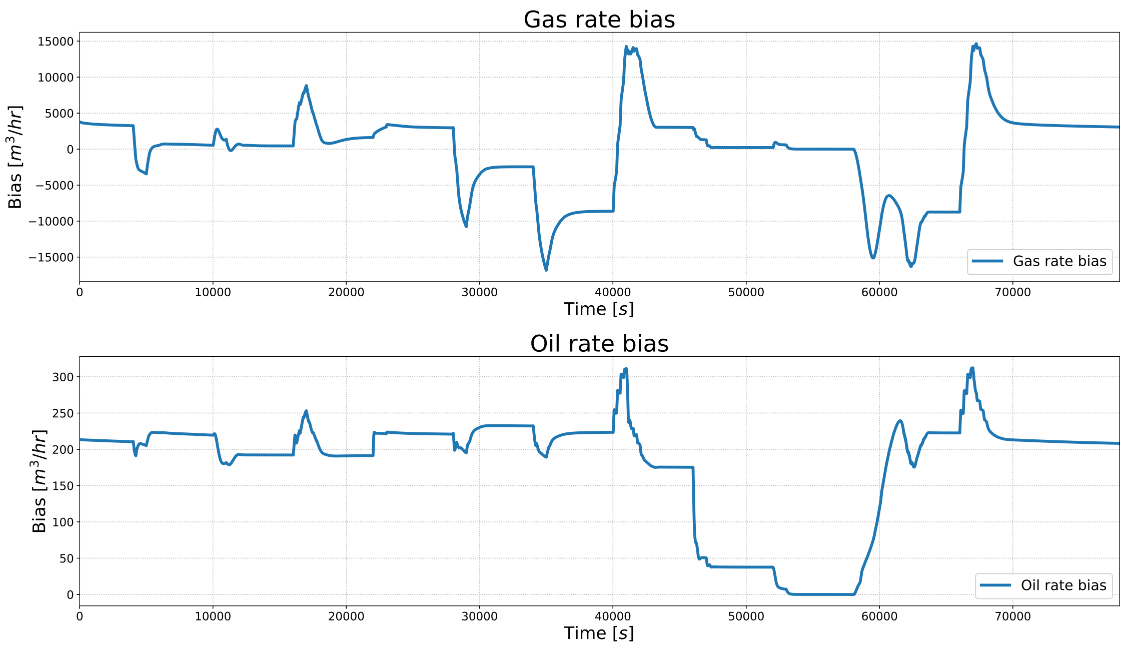

Figure 16: Bias when using the continuous MPC with continuous choke for control.

7.1.2 Step Choke

The previous subsection presented the performance of the continuous MPC when both inputs are continuous. However, in the real system, the production choke is a discrete actuator with a step size of 2%, as outlined in Section 3.1. This subsection presents a simulation of the system with the discrete choke treated as continuous by the controller.

As the continuous MPC considers the manipulated variables as continuous, the discrete decisions must be handled outside the optimization problem using additional logic. For this scenario, rounding with a deadband is used to determine the choke position. The desired choke position from the MPC is rounded to the nearest 2% when it is $\pm1.6\%$ from the actual choke position. This rounding logic is implemented in the plant, with the MPC having no knowledge of it.

The performance of the continuous MPC with the step choke is depicted in Figure 17. The figure shows that the controller can guide the controlled variables toward their set points, often with rapid convergence and high accuracy. However, large oscillations are observed in several parts of the simulation. Figure 18 illustrates the choke position and gas-lift rate throughout the simulation. Oscillations in the choke for several periods of the simulation are observed, translating to the large oscillations seen in the controlled variables. This is an expected outcome as the MPC is treating the discrete choke as continuous. From Figure 19, it can be observed that the bias is also oscillating in the same time periods.

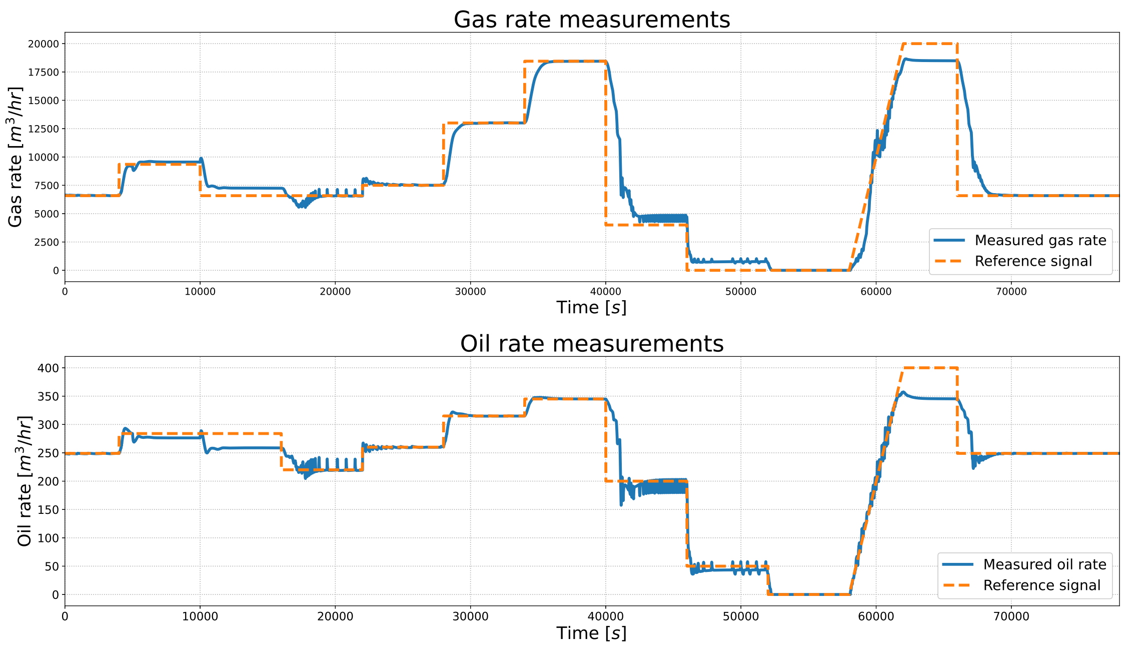

Figure 17: Oil and gas rate measurements when using the continuous MPC with step choke for control.

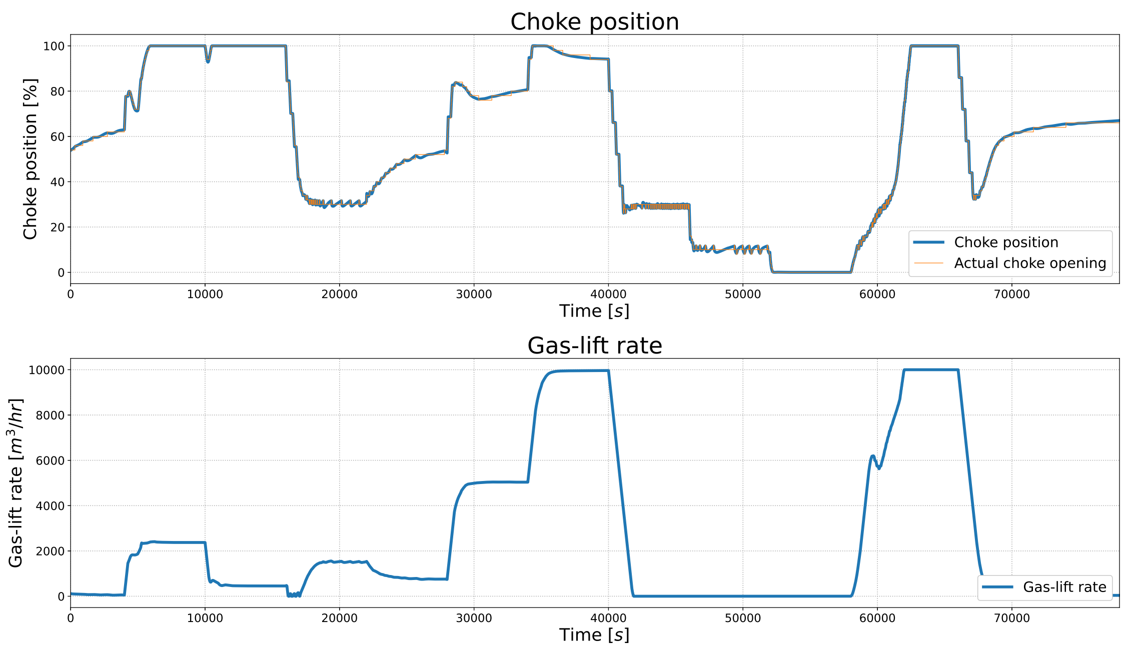

Figure 18: Choke position and gas-lift rate when using the continuous MPC with step choke for control.

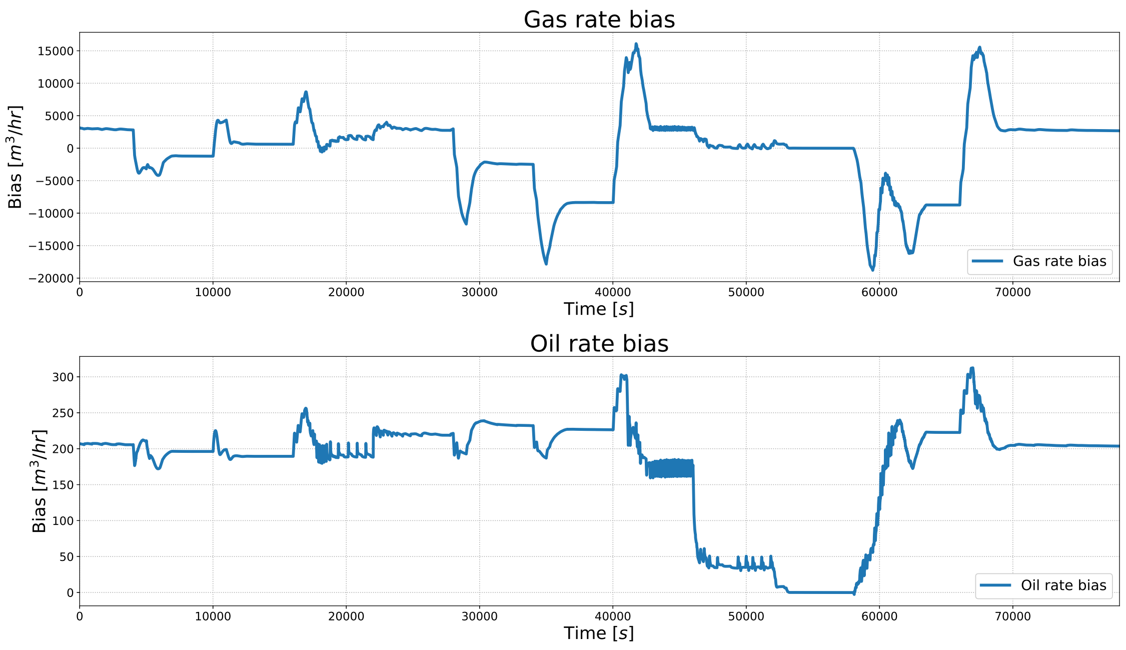

Figure 19: Bias when using the continuous MPC with step choke for control.

To better highlight the oscillating behavior, Figures 20 and 21 depicts a close-up during a period of oscillations. In Figure 20, it is seen that the controlled variables deviate only slightly from their setpoints. Despite employing the Soft MPC method, deviations within the deadzone are still penalized to some extent, as evident in Figure 21, where the controller makes small adjustments in the inputs to rectify these deviations.

However, due to the discrete nature of the choke and the implemented rounding logic with deadband, no movement is initiated in the choke until the desired choke position exceeds the deadband. Consequently, the controller tries to continuously adjust the inputs to mitigate the deviations further. The actual movement of the choke is thus delayed until the desired choke position shifts by more than 1.6% from its current position, at which point the choke position rapidly increases by 2%. This causes a significantly larger response in the controlled variables than what was anticipated by the MPC, resulting in substantial overshooting of the setpoints. To compensate for this unexpected response, the controller must decrease the inputs.

Likewise, when the desired choke position decreases by more than 1.6% from the actual choke position, the choke rapidly decreases by 2%, causing another large and unpredicted response, thereby reinstating the initial problem for the MPC. This pattern continues due to the MPC's lack of knowledge regarding the discreteness of the choke and the external logic employed. The resulting oscillations have a detrimental effect on the overall performance of the control system, and can dramatically reduce the lifespan of the actuator.

Upon examination, it becomes evident that the oscillations primarily occur when the choke position is below 50%. This observation can be further supported by referring to Figures 10 and 11, which depict the steady-state responses of the controlled variables when applying steps to the choke. Notably, the step response of the controlled variables is significantly more pronounced when the choke position is less than 50%. Consequently, a 2% step in the choke position within this region exerts a considerably greater impact on the controlled variables compared to when the choke is almost fully open.

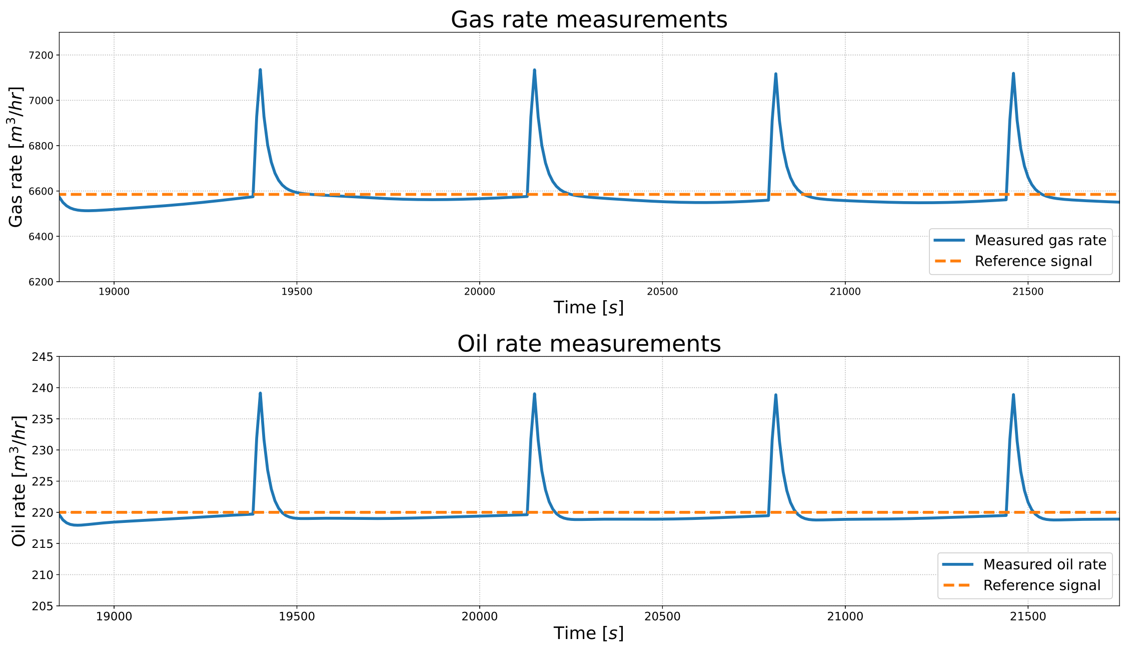

Figure 20: Oscilliations in oil and gas rates when the continuous MPC is used to control the system with step choke.

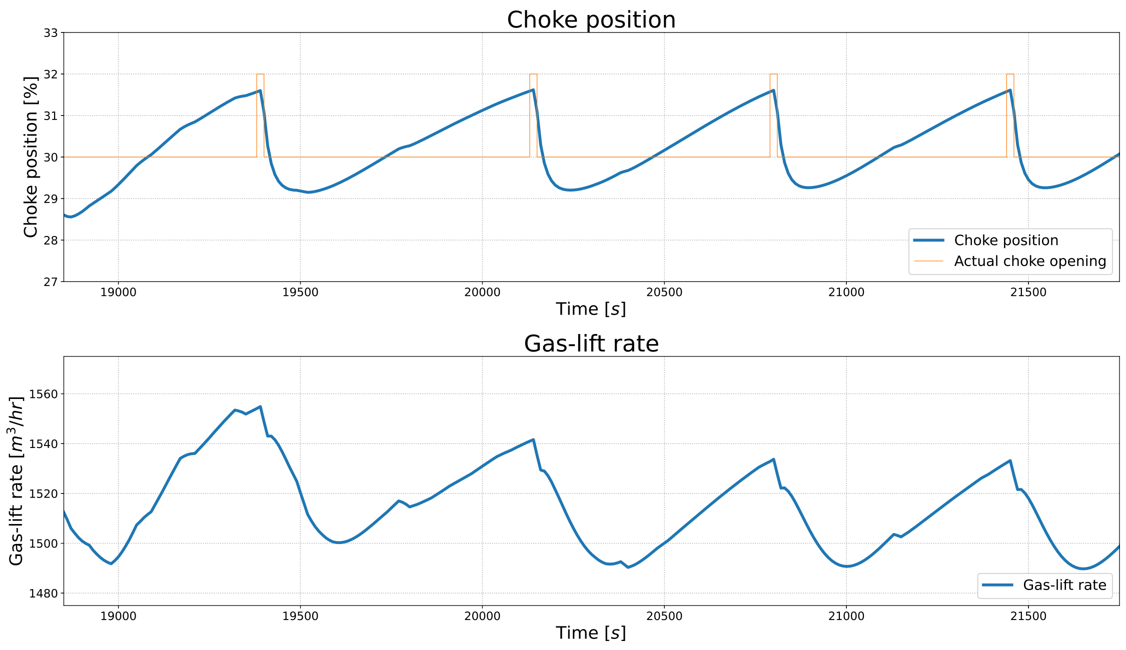

Figure 21: The continuous MPC treats the step choke as continuous, leading to oscillations in the choke position and gas-lift rate.

7.2 MIMPC

In the previous section, the results were obtained by employing the continuous MPC which treats the choke as continuous. This section presents the results obtained by utilizing the MIMPC for control, which recognizes the discreteness of the choke and incorporates it into the control system.

Initially, the MIMPC is utilized to regulate the system with identical reference trajectories, as seen in the previous scenarios, to allow for a direct comparison of the control approaches. Moreover, this section showcases how the MIMPC utilizes the prediction model and the integrality constraints to determine optimal inputs and takes advantage of re-optimization and plant feedback to achieve precise control. Subsequently, fine-regulation of a single controlled variable is exhibited, which is often conducted in practical applications. Lastly, the section demonstrates the MIMPC's ability to handle disturbances and the effect of the Soft MPC method.

7.2.1 MIMPC Performance

The performance of the MIMPC is illustrated in Figure 22. Remarkably, the MIMPC demonstrates a striking similarity in performance to the continuous MPC when the choke was continuous, exhibiting rapid convergence towards the setpoints in both scenarios. Stationary deviations are still observed for the same time periods as seen earlier, as convergence to both setpoints is not always feasible. It is worth noting that the deviations observed are slightly larger when employing the MIMPC for control, as achieving accurate control with a discrete actuator poses greater challenges.

In contrast to the utilization of continuous MPC for controlling the step choke, the MIMPC successfully eliminates oscillations. This achievement can be attributed to the MIMPC's knowledge of the discreteness of the choke. As a result, the MIMPC is able to make predictions with respect to the discreteness and anticipate the substantial response observed in the previous subsection.

Figure 23 illustrates the choke position and the gas lift rate through the simulation. It can be observed that the desired choke position from the MIMPC is integer, $\pm 2\%$, which is due to the incorporation of integrality constraints in the optimization problem. The actual choke position is thus able to track the desired choke position and the jumpiness observed in Figure 21, when the continuous MPC was used with the discrete choke, is not observed.

The bias, $v(k)$, throughout the simulation is illustrated in Figure 24. It is evident from the figure that the bias exhibits a considerable magnitude, due to modeling errors. As the step response model used for predictions is a linear representation of a nonlinear system, the model fails to capture the significant responses in the controlled variables that occur for instance when the choke is opened from a nearly closed position. Nevertheless, the bias is effectively compensated for through feedback, and the impact on performance is relatively minor.

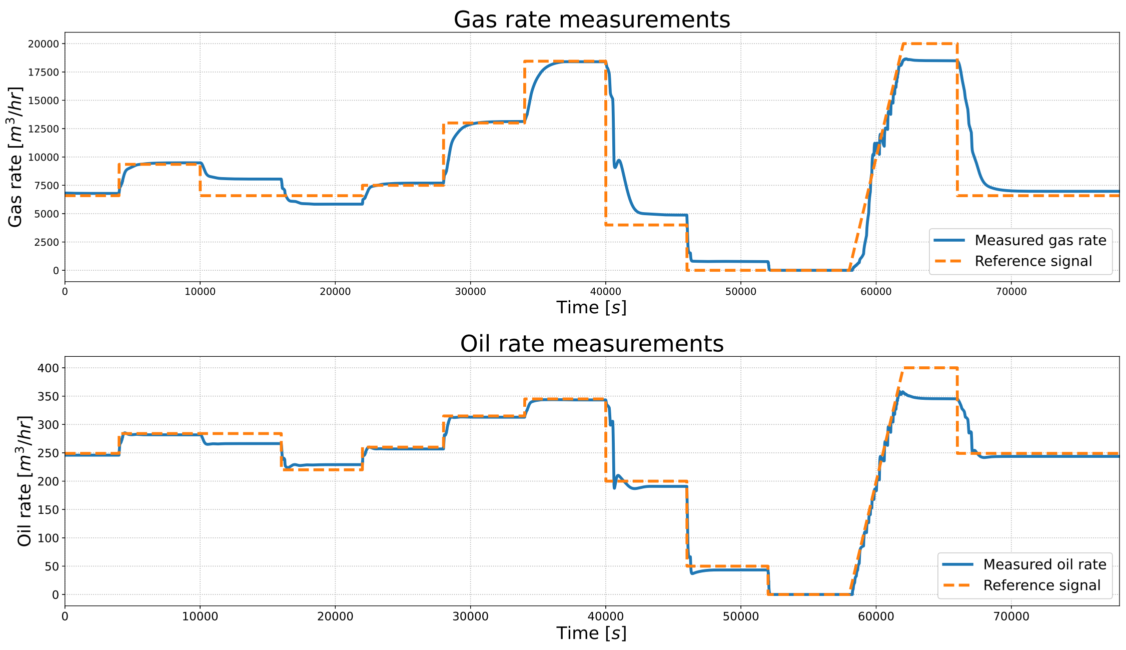

Figure 22: Oil and gas rate measurements when using the MIMPC for control.

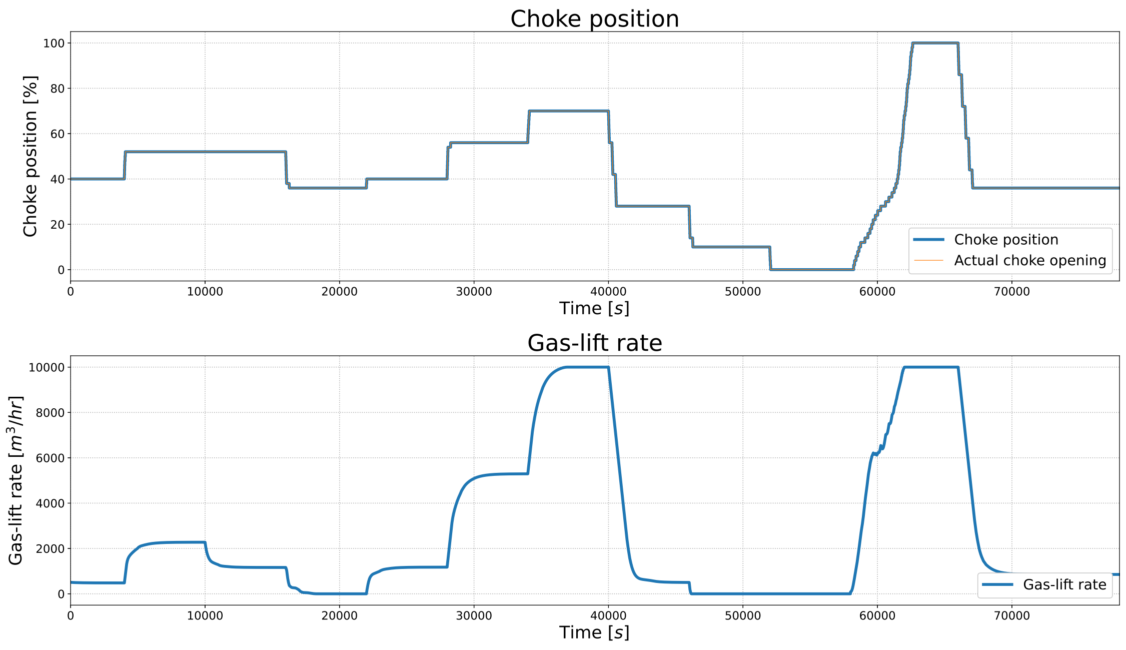

Figure 23: Choke position and gas-lift rate when using the MIMPC for control.

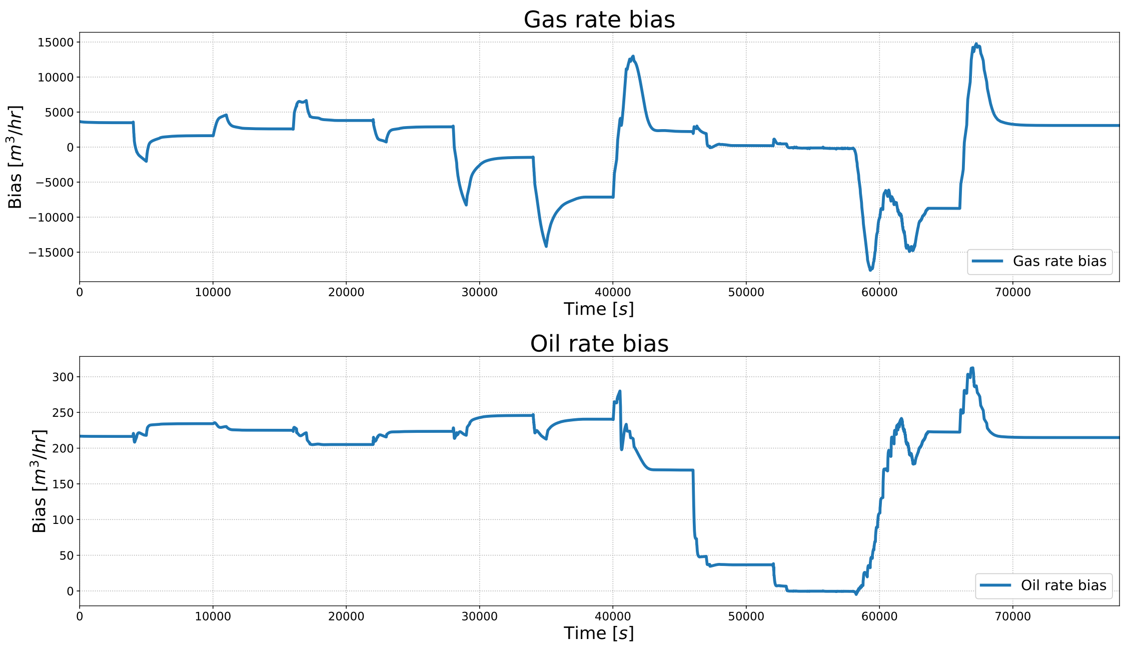

Figure 24: Bias when using the MIMPC for control.

7.2.2 MIMPC Predictions

The MIMPC employs the step-response model (5.2) to optimize future input moves and predict the corresponding responses of the controlled variables. Figure 25 shows the MIMPC's planned input moves and predicted responses following a step in the reference trajectories. The white area denotes the system's historical behavior, while the gray area represents the MIMPC's projected future behavior. The two plots at the bottom of the figure showcase the optimal open-loop future input moves, whereas the middle plots illustrate the future choke position and gas-lift rate. The upper two plots illustrate the predicted responses of the oil rate and gas rate.

As evidenced by the figure, the controller plans to gradually open the choke to 90%, with a 2% choke movement every timestep over a period of 25 timesteps. Additionally, the gas-lift rate is planned to be increased throughout the control horizon. The implemented input-blocking, described in Section 6.2, is evident in the planned gas-lift moves, with small blocks at the start of the horizon and larger blocks towards the end. The predicted responses of the controlled variables reveal that the oil rate is predicted to converge to a value slightly below the set point, while the gas rate exhibits a larger deviation from the set point due to the prioritization of the oil rate.

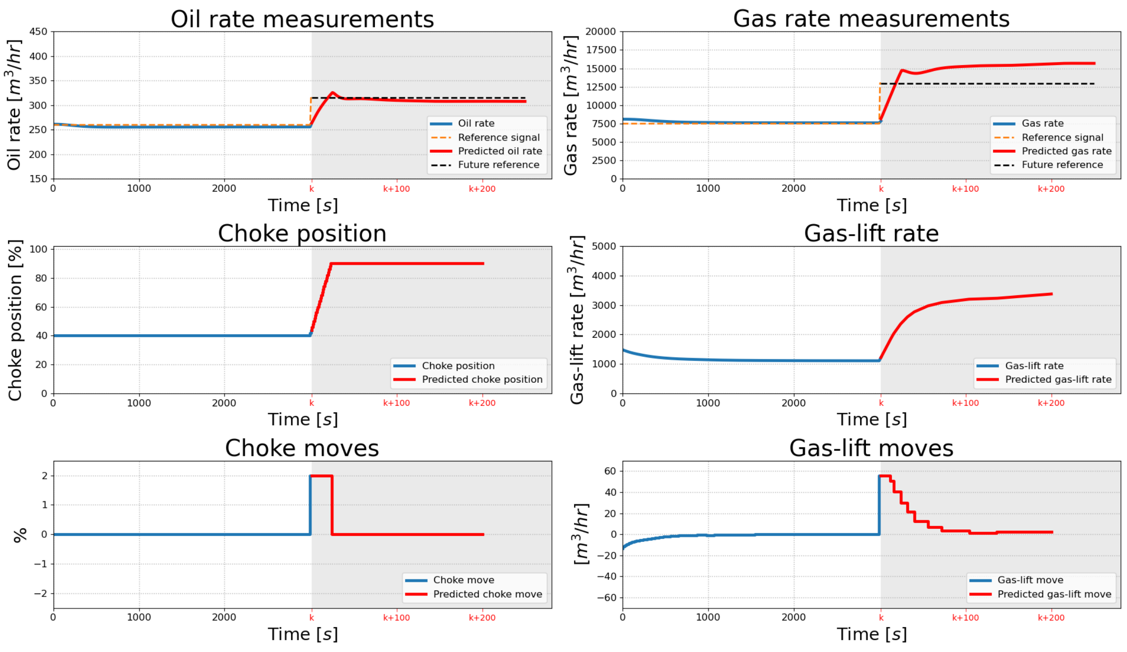

Figure 25: The planned input moves and predicted response of the MIMPC when a step in the reference trajectory is employed.

The MIMPC employs a receding horizon strategy. In this strategy, the first optimal input move is executed, followed by re-optimization based on the latest measurements fed back from the plant. This feedback mechanism enables the controller to adapt effectively to uncertainties, such as modeling errors and external disturbances. Consequently, the predicted optimal moves at one time step, may not necessarily remain optimal in subsequent time steps.

Figure 26 illustrates the actual evolution of the system following the step in the reference trajectories. A comparison between the actual evolution, seen in Figure 26, and the initially planned evolution, seen in Figure 25, reveals a difference between the planned and actual evolution of the system. It can be observed that both the choke position and the gas-lift rate exceed the initially planned trajectories. However, when examining the measured oil rate and gas rate, it becomes evident that both controlled variables were able to reach their setpoints, which is a much better performance than what was initially anticipated.

This observation highlights the crucial role played by the feedback and re-optimization mechanism in the MIMPC. By adapting to factors like modeling errors, the controller effectively ensures that both setpoints are ultimately reached. This demonstrates the efficacy of the control system in handling uncertainties and achieving the desired outcomes.

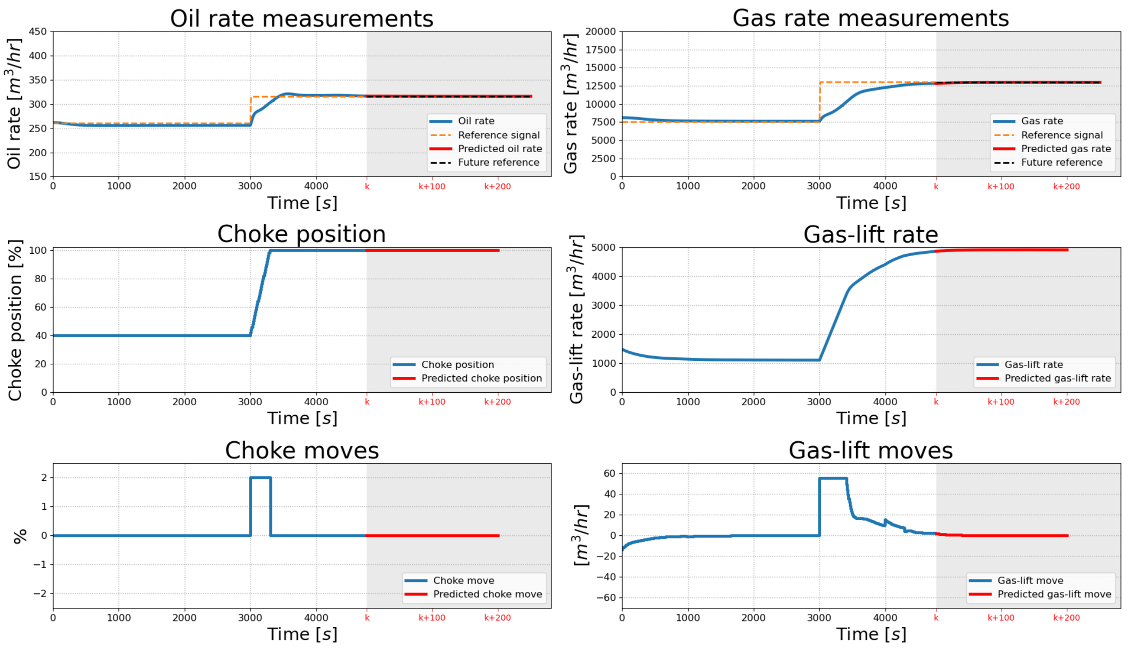

Figure 26: Evolution of the system and the actually implemented input moves by the MIMPC after a step in the reference trajectory was employed.

7.2.3 Open-Loop vs. Closed-Loop

In MPC applications, it is typically desired to achieve some similarity between the open-loop response and the closed-loop response. This similarity can indicate that the controller tuning and the model accuracy are sufficient. Consequently, feedback control and re-optimization are mainly focused on addressing minor discrepancies that may arise. Figure 27 and 28 provide a visual representation of how the system would have behaved if the initially open-loop input moves, seen in Figure 25, were implemented compared to the closed-loop performance. It is worth noting that the choke position is in the region where large nonlinearities are not present, as established earlier.

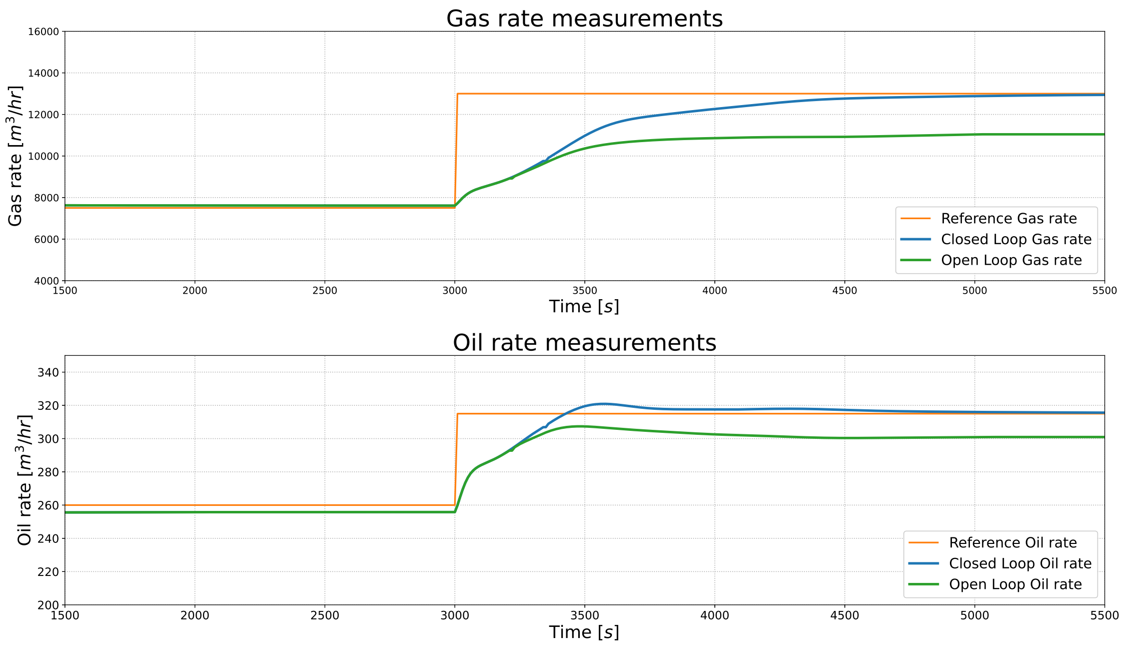

Figure 27: Oil and gas rate measurements: Open loop vs. closed loop.

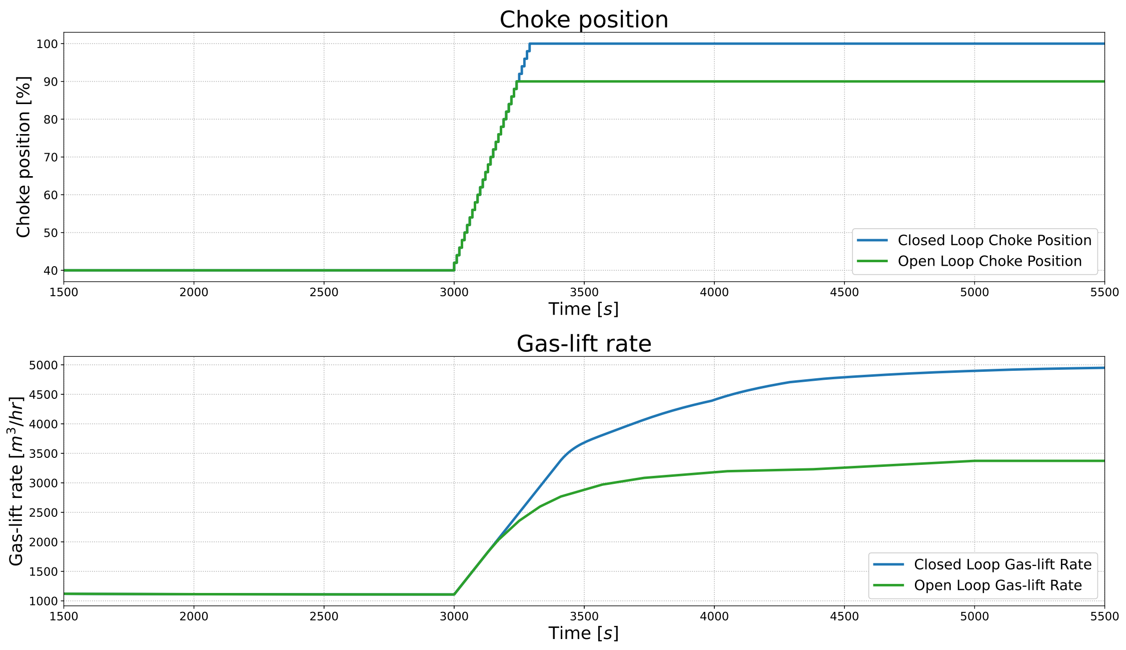

Figure 28: Choke position and gas-lift rate: Open loop vs. closed loop.

Figure 27 reveals that initially, the open-loop and closed-loop responses exhibit a similarity. However, as the simulation progresses, a deviation becomes apparent between the two. While the open-loop response settles with a noticeable deviation from the setpoints, the closed-loop control system recognizes that the initially planned input trajectories are insufficient for achieving precise convergence to the setpoints.

Figure 28 demonstrates that the controller adjusts the inputs beyond the initial plan to address deviations from the setpoints. The step response model utilized for predictions is a linear approximation of a nonlinear system. As a result, it is expected that the open-loop scenario fails to precisely reach the setpoints due to inherent modeling errors. It is worth noting that the step response model assumes a working point with a choke position of 50% open and a gas-lift rate of $7500\frac{m^3}{hr}$. Although the choke position initially remains relatively close to the working point, there is a significant deviation in the gas-lift rate. As the choke moves further away from the working point, the prediction model progressively diverges from the actual response, as depicted in Figure 11 and Figure 10. Consequently, the actual system response deviates from the MPC's predictions.

While the prediction model may not be perfect, it nonetheless offers valuable insights into the dynamics of the system. This empowers the closed-loop control system to effectively and accurately achieve the desired setpoints through feedback and re-optimization, leveraging the information obtained from the plant.

7.2.4 MIMPC Fine Regulation

The previous findings highlighted a deviation in both controlled variables for certain setpoint combinations. This occurs because the setpoints are not simultaneously attainable for both variables, as the manipulated variables impact both of them. Typically, predefined tuning parameters are used for the controller when optimizing the objective function. However, in practical applications, there may be a need for fine regulation of a single controlled variable.

In Equinor, there has been a historical focus on fine-regulating the oil rate. In this context, the gas rate setpoint is often set to the measured gas rate at that particular timestep. Alternatively, the penalty for the deviation of the gas rate can be turned off, resulting in the same outcome. It is important to note that the Gurobi solver, utilized in the MIMPC algorithm, is capable of handling zero weights in the objective function. Therefore, to turn off the penalty for gas rate deviation, the corresponding weighting parameters can be set to zero: $\bar{Q}_{gas} = S_{gas} = s_{gas} = 0$.

Figure 29 provides insight into the impact of turning off the penalty for gas rate deviation. Initially, both controlled variables exhibit a stationary deviation. The oil rate falls slightly below the setpoint, while the gas rate shows a larger deviation above its setpoint. Figure 30 demonstrates that both the choke position and the gas-lift rate remain constant, despite the deviation in the controlled variables. Notably, adjusting the choke position would lead to an increase in the oil rate, resulting in a smaller objective function value. However, it would also cause the gas rate to rise, thereby increasing the objective function value. The same trade-off applies to adjusting the gas-lift rate. As a result, the controller stabilizes at a stationary point that effectively minimizes the overall objective function.

At $3000s$, the penalty for gas rate deviation is turned off. Consequently, the gas rate deviation no longer affects the objective function value. As a result, the controller can solely focus on minimizing the oil rate deviation without considering the gas rate. This allows for precise regulation of the oil rate, leading to perfect convergence towards the setpoint. Figure 30 illustrates how the gas-lift rate is used for the fine regulation of the oil rate.

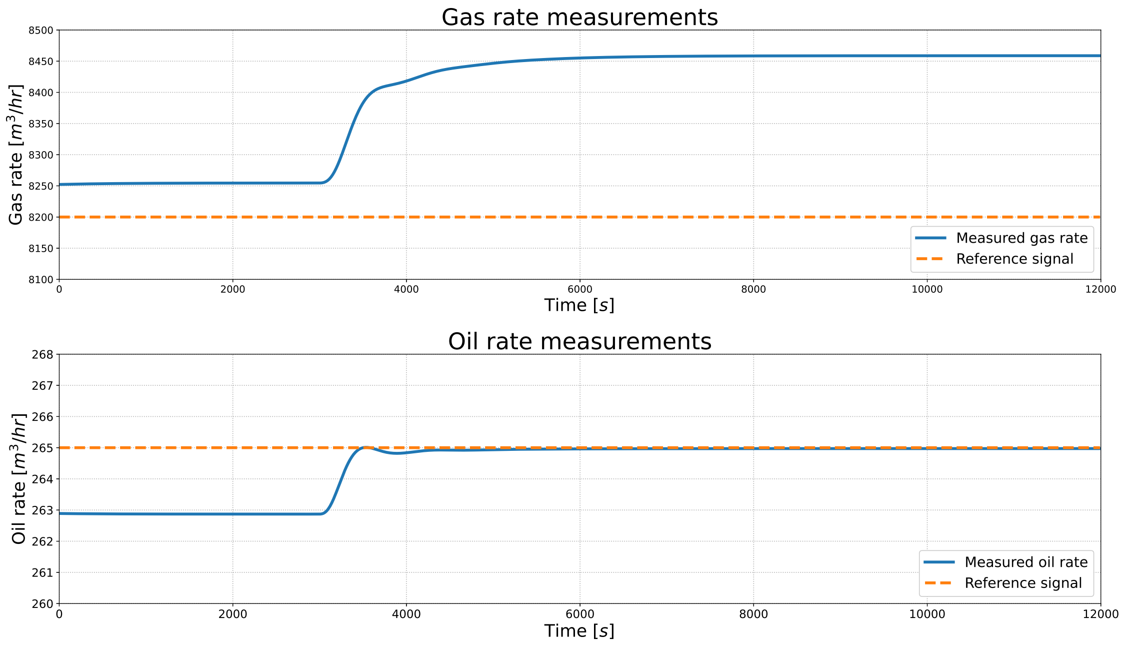

Figure 29: Fine regulation of the oil rate, turning off penalization of the gas rate deviation.

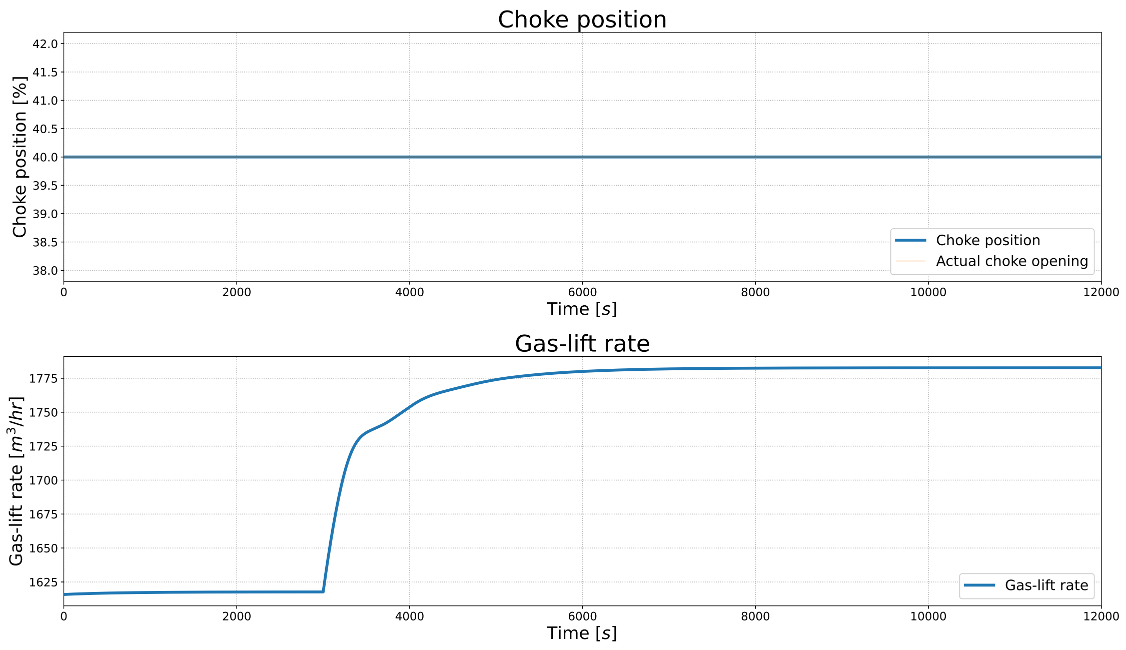

Figure 30: Gas-lift rate used to fine regulate the oil rate when the penalization of the gas rate deviation is turned off.

7.2.5 Disturbance

The robustness of the MIMPC to external disturbances is a crucial aspect of its practical applicability. In Figure 31, the controller's ability to handle a drop in the reservoir pressure of 3 Bar is presented. The MIMPC responds to the disturbance by increasing the gas-lift rate, as evident in Figure 32, and effectively compensates for the pressure drop, as the oil rate converges back to the setpoint. This illustrates the MIMPC's capability to maintain stability in the face of disturbances. Figure 33 depicts the drop in the pressure in the reservoir.

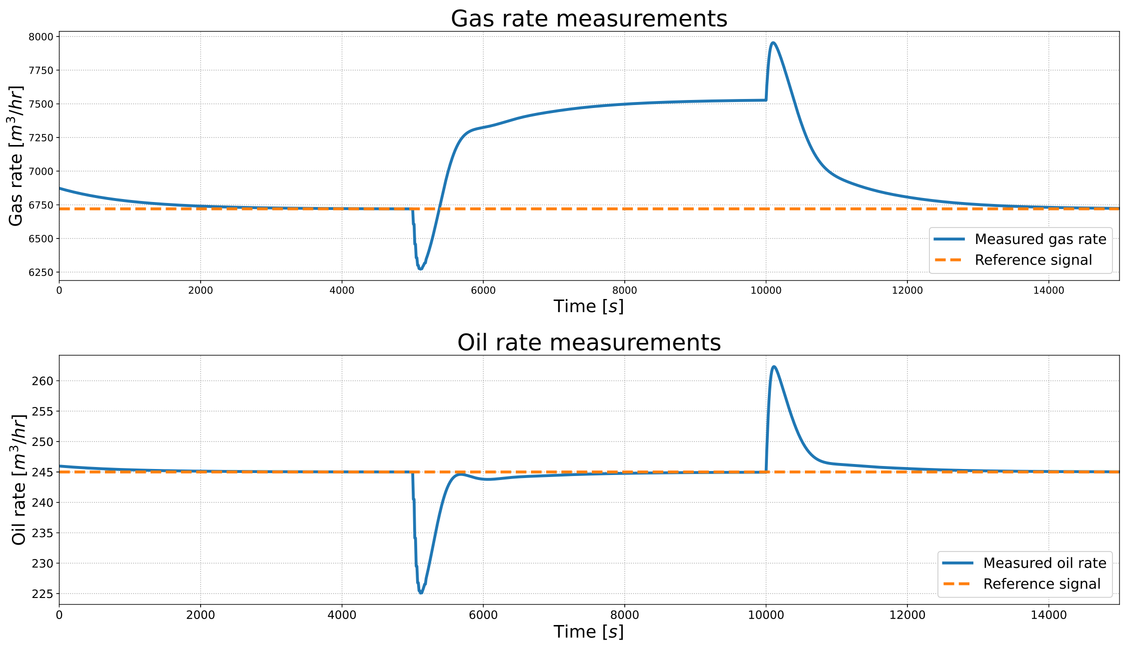

Figure 31: Oil and gas rate measurements when a pressure drop of 3 Bar is imposed on the reservoir.

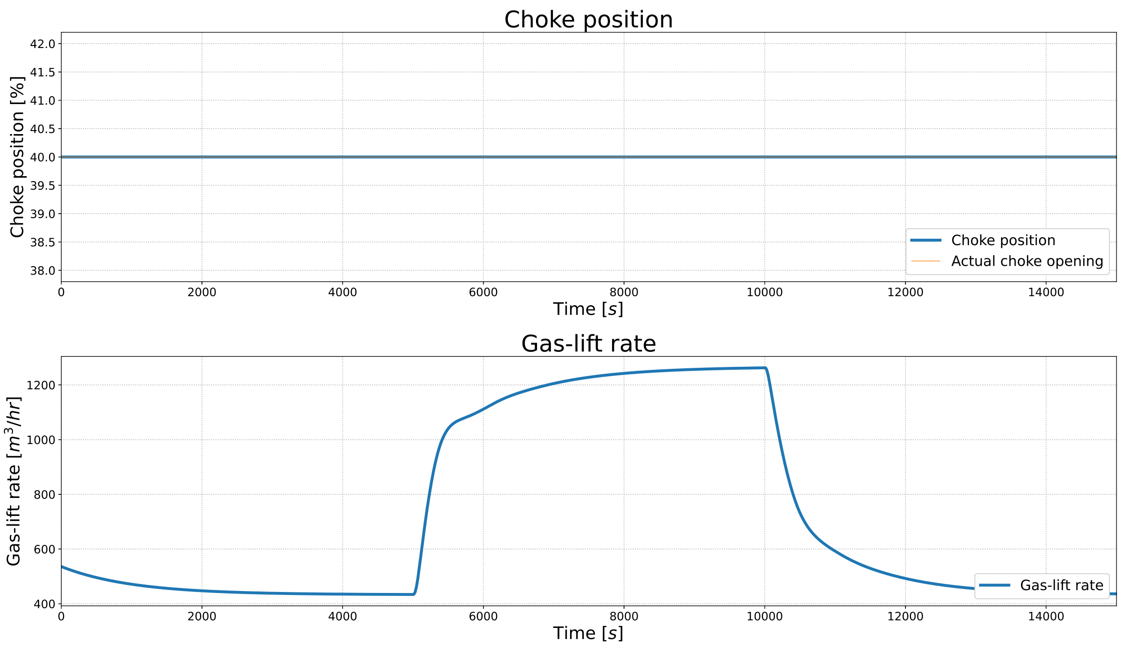

Figure 32: Gas-lift rate used to compensate for the 3 Bar pressure drop in the reservoir.

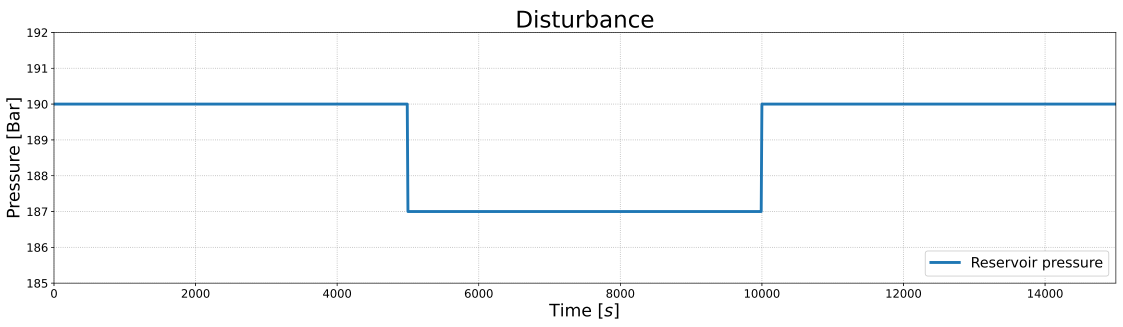

Figure 33: Disturbance imposed in the system. 3 Bar pressure drop in the reservoir.

7.2.6 MIMPC Performance Without Soft MPC

The Soft MPC method, introduced in Section 6.4, addresses the issue of input oscillations resulting from discrepancies between the plant model and the actual system. Figure 34 showcases the performance of the MIMPC when the Soft MPC method is not utilized. Overall, the controller performs well, even in the absence of Soft MPC. However, as observed in the figure, small oscillations are present during certain time periods. Figure 35 illustrates the behavior of the choke position and the gas-lift rate. It is clear that the oscillations in the controlled variables are caused by the oscillatory behavior of the choke. This effect is particularly pronounced when the choke is nearly closed, which aligns with the findings depicted in Figure 11 and Figure 10. As seen earlier, when the choke is nearly closed, the controlled variables exhibit significantly larger responses compared to what the prediction model is based on. This discrepancy indicates that the oscillations arise from a mismatch between the prediction model and the actual behavior of the system.

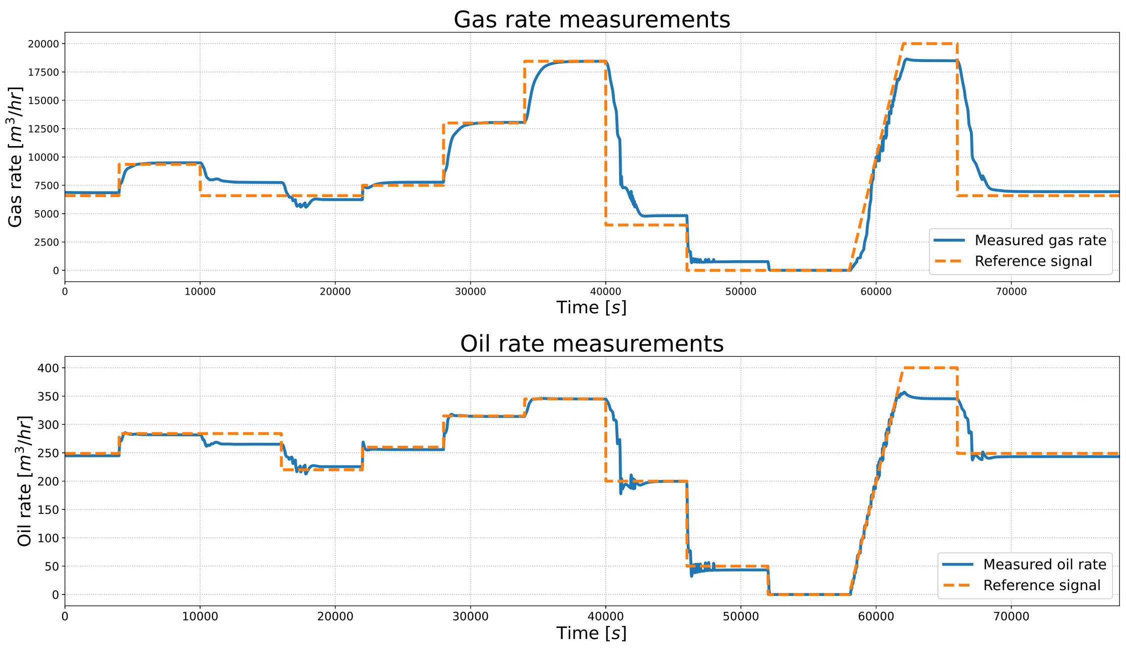

Figure 34: Oil and gas rate measurements when using the MIMPC without the Soft MPC method for control.

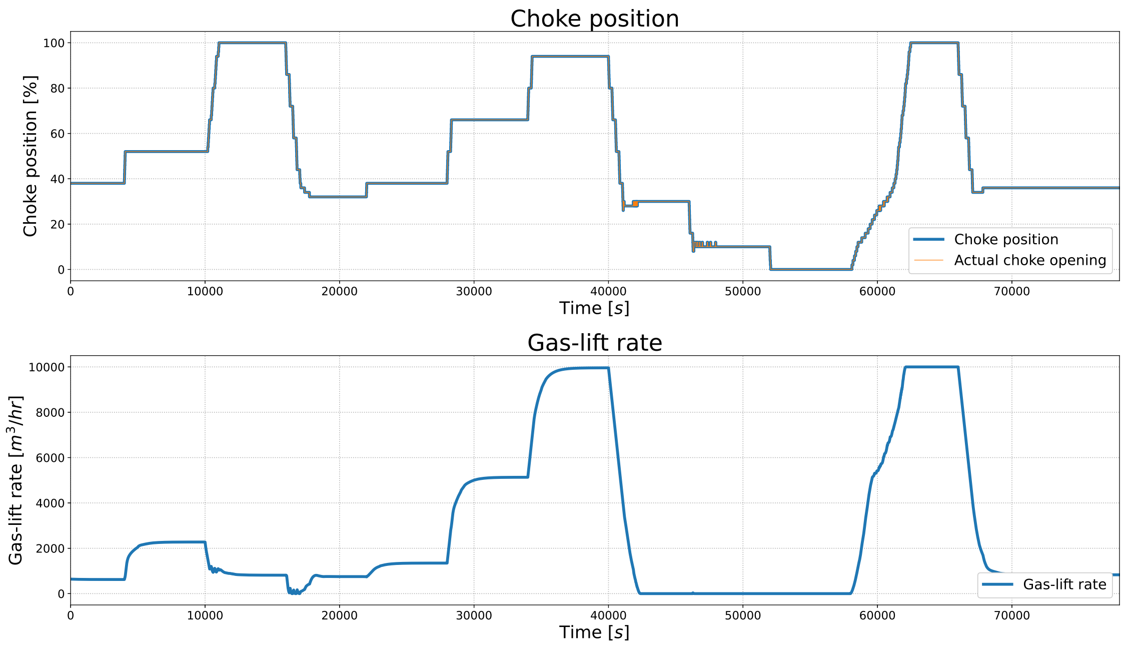

Figure 35: Choke position and gas-lift rate when using the MIMPC without the Soft MPC method for control.

To shed light on the causes of these oscillations, the following figures provide insight into the predictions made by the MIMPC during a period when oscillations occur (approximately at 48000 seconds). The figures display the same simulation as above, with a specific focus on the observed oscillations. Figure 36 clearly illustrates slight deviations of both the oil rate and gas rate from their setpoints. As a result, the MIMPC, based on the optimization problem, determines a 2% adjustment in the choke position to achieve a more optimal outcome. This adjustment is visible in the figure, and the predicted responses of the controlled variables are represented by the red line in the upper two plots. It can be observed that the predicted oil rate approaches the setpoint more closely while the steady-state gas rate remains relatively unchanged.

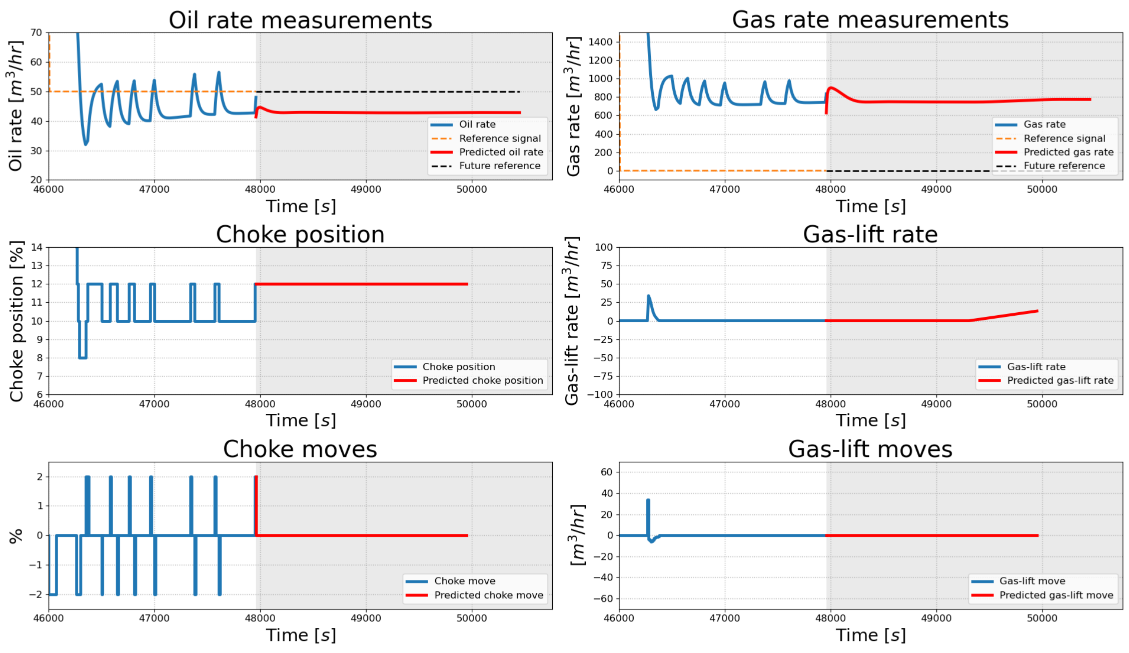

Figure 36: Planned input moves and predicted response of the MIMPC when the Soft MPC method is not used.

Moving to Figure 37, it is evident that the actual response, particularly in the oil rate, exceeds the MIMPC's prediction, resulting in an overshoot beyond the set point. Consequently, based on feedback from the plant, the MIMPC determines that a -2% adjustment back to the previous choke position would lead to a more optimal outcome. From the figure, it becomes apparent that the MIMPC predicts the oil rate will ultimately approach the setpoint closely, slightly below it.

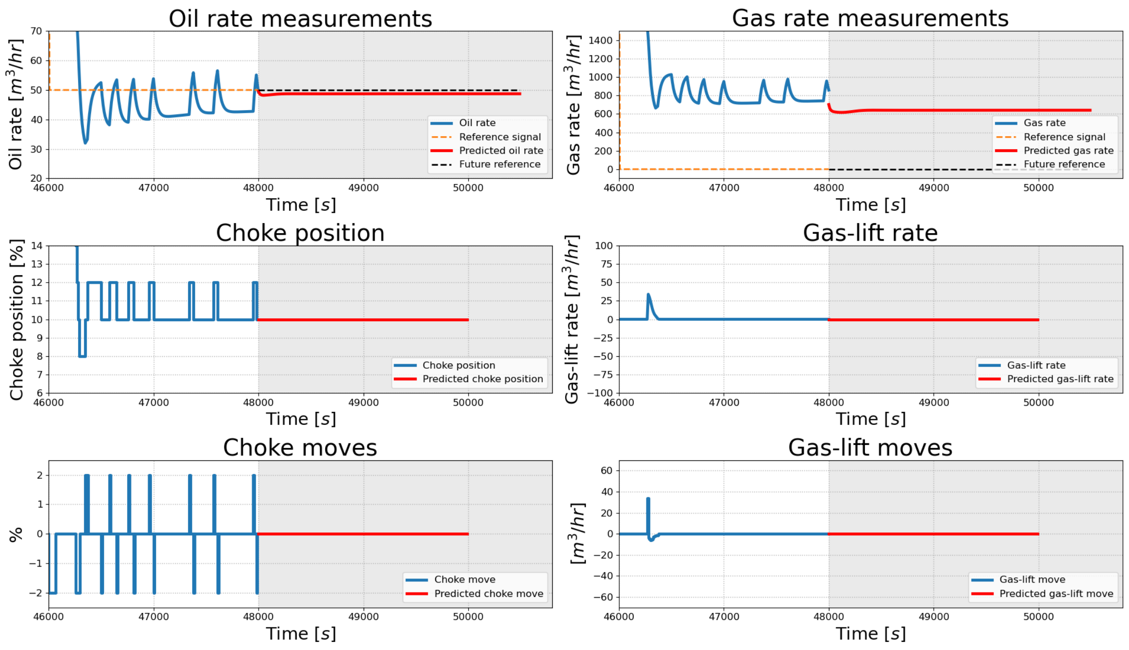

Figure 37: Mismatch between the predicted response and the actual response causes the controller to move the choke back to the previous position, causing oscillations in the controlled variables.

From Figure 38, it becomes evident once again that the actual response significantly deviates from the prediction done by the controller, and the controlled variables' actual responses do not align with the anticipated behavior.

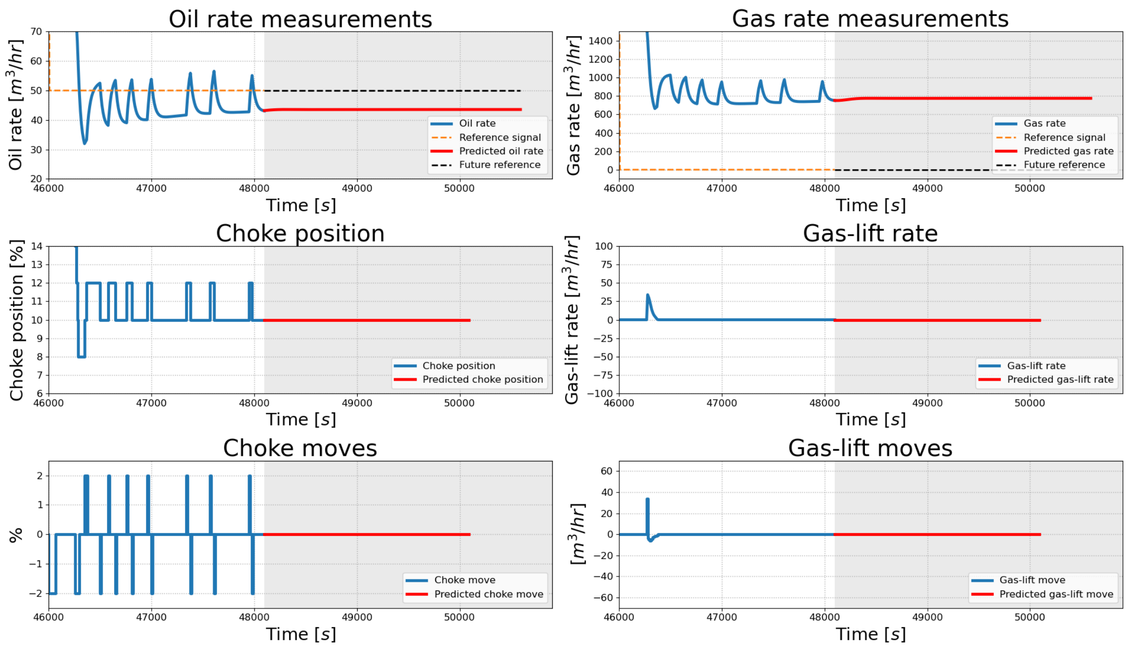

Figure 38: Actual response again deviates from the predicted response due to plant model mismatch.

This case underscores the detrimental impact of a significant plant model mismatch on controller performance. As observed in this scenario, when faced with a substantial plant model mismatch, the controller fails to predict the actual responses of the system accurately.

As previously demonstrated, the Soft MPC method proves highly effective in mitigating oscillations caused by plant model mismatch. By reducing the penalties associated with small deviations from the setpoint, the Soft MPC method enables the controller to achieve a small, stable, and stationary deviation instead of oscillating around the setpoint. This not only leads to smoother and more consistent control performance but also safeguards the actuator's lifespan by preventing excessive wear and tear. Ultimately, the Soft MPC method significantly enhances the overall system response.

8 Discussion

This chapter provides a discussion of the results obtained in this study. The research question posed in this thesis is repeated here:

"How can a discrete actuator be effectively integrated into an MPC problem to enable the controller to account for the inherent discreteness of the actuator? Furthermore, can this integration potentially result in enhanced performance compared to traditional continuous control?"

In Chapter 6, the implementation of the MIMPC was presented, highlighting how this control strategy incorporates integer decision variables into the MPC optimization problem. This provided the controller with knowledge of the discrete nature of the choke. In Chapter 7, the outcomes of applying the MIMPC for control were presented, along with the results obtained from using the continuous MPC to control the discrete choke. Additionally, a benchmark result was provided, where the step choke was replaced by a continuous choke. This benchmark result served to highlight the challenges of continuous control of a discrete actuator and to establish a baseline for evaluating the performance of the MIMPC.

The findings from applying continuous MPC to control the step choke were presented in Section 7.1.2. These results highlighted the challenges arising from the lack of knowledge about the discrete nature of the choke. In the presence of small deviations in the controlled variables, the controller attempted to make small adjustments to the choke in order to correct these deviations. However, due to the external rounding logic that was employed, the choke remained stationary until the desired choke position exceeded the deadband. This resulted in a sudden and unexpected response in the controlled variables once the deadband was surpassed, prompting the controller to readjust the choke back to its original position. Consequently, the oscillatory behavior in the system was observed as a consequence of the controller's lack of knowledge of the discrete nature of the choke.

Section 7.2.1 presented the results obtained from utilizing the MIMPC for control. A key distinction between the MIMPC and the continuous MPC is the incorporation of integer decision variables into the optimization problem, which grants the MIMPC knowledge of the discrete nature of the choke. This awareness enables the controller to recognize that the choke can only move in discrete steps. Consequently, the MIMPC can more accurately predict the significant response resulting from the step choke movement and effectively avoid large overshoots observed when using the continuous MPC. Instead, the MIMPC can utilize gas-lift rate adjustments to fine-regulate the controlled variables when possible. Notably, the presence of oscillations in the inputs and outputs is the primary performance difference between the continuous MPC and the MIMPC.

Comparing the average deviations of the controlled variables, it is interesting to note that the continuous MPC achieves smaller average deviations compared to the MIMPC. As illustrated in Figure 17, the continuous MPC demonstrates the ability to closely approach the setpoints, even with a discrete choke. However, the presence of oscillations significantly impacts its performance. Despite having smaller average deviations from the setpoints compared to the MIMPC, the continuous MPC's oscillatory behavior strongly affects its effectiveness.

In contrast, the MIMPC takes a different approach. By incorporating knowledge of the discreteness of the choke, the MIMPC recognizes that the choke can only move in discrete steps. As a result, the controller recognizes the large response from a step in the choke and thus settles for a small deviation rather than overshooting significantly due to the step size. Minimizing actuator oscillations is crucial in real-world applications as it places a substantial strain on the actuators and reduces their lifespan. Therefore, a small deviation from the setpoint is far preferable to oscillations around the setpoints.

Figure 14 displays the performance of the continuous MPC when a continuous choke is used. This serves as a benchmark to showcase the expected performance in the "best case" scenario. Comparing this benchmark with the results obtained from the MIMPC (Figure 22) and the continuous MPC with a discrete choke (Figure 17) allows for evaluating the effectiveness of the MIMPC as a control method for systems with discrete actuators.

Interestingly, the performance of the MIMPC closely resembles the benchmark simulation, indicating that the MIMPC is well-suited for systems with discrete actuators. The MIMPC exhibits similar characteristics to the benchmark by effectively utilizing the knowledge of the choke's discreteness and providing accurate control without excessive oscillations. This suggests that the MIMPC is a favorable control method when discrete actuators are present in the system.

In this study, the continuous MPC with a discrete choke employed a simple external logic of rounding the choke position with a deadband. It is worth discussing that alternative external logic or heuristics could potentially lead to improved performance. Equinor, for example, utilizes additional techniques to achieve excellent performance, even when treating the discrete actuator as continuous in the optimization problem. Fine-tuning and adjustments to the weighting parameters may also enhance performance.

However, the noteworthy aspect of the findings in this study is the remarkable performance of the MIMPC without relying on external logic or heuristics. The MIMPC incorporates the discreteness of the actuator directly into the optimization problem, enabling it to achieve optimal control. As highlighted in Section 2.2.2, rounding a continuous optimization solution to obtain an integer solution is generally not recommended. By including the discreteness of the actuator in the optimization problem, the MIMPC is able to deliver superior control performance compared to the continuous MPC.

Additionally, the MIMPC demonstrates a high level of robustness. Despite the presence of significant modeling errors, as a linear step response model is used to predict the behavior of a nonlinear system, the MIMPC demonstrates good performance. The compensation for these modeling errors is achieved through feedback control, allowing the MIMPC to handle the system dynamics and achieve satisfactory control effectively. Furthermore, the MIMPC exhibits robustness in the face of external disturbances. As illustrated in Figure 31, the MIMPC demonstrates its ability to effectively handle and mitigate the impact of external disturbances on the system. Overall, these findings emphasize the robustness of the MIMPC, showcasing its capability to handle significant modeling errors and external disturbances while delivering satisfactory control performance

From Figure 34, it is evident that the performance of the MIMPC without the Soft MPC method exhibits some oscillations in the system caused by plant model mismatch. It is important to differentiate these oscillations, caused by plant model mismatch, from those observed when using the continuous MPC, caused by continuous control of a discrete actuator. To address the issue of plant model mismatch while still utilizing step response models as prediction models, one potential solution is the implementation of *Gain-scheduling* [58]. By incorporating different step response models based on the operating points of the manipulated variables, the prediction model can be enhanced to be more accurate within the region surrounding the operating points. This approach has the potential to reduce plant model mismatch and improve system performance. By utilizing a more accurate prediction model, the MIMPC can be able to operate closer to the setpoints, possibly achieving improved control performance.

Although the MIMPC demonstrates good performance, it is important to acknowledge the additional computational complexity associated with mixed integer programming. Figure 39 provides insight into the computational time per iteration for the MIMPC and the continuous MPC. Notably, the computational time for the MIMPC is significantly longer compared to the continuous MPC. Furthermore, large spikes in computational time are observed in the case of the MIMPC, highlighting a critical aspect that requires careful consideration when utilizing integer programming in the optimization problem.

As established in Section 2.2, despite extensive research, no polynomial-time algorithm has been found for solving mixed integer programs to optimality. Figure 39 provides insight into the computational time per iteration for MIMPC and the continuous MPC. Notably, the computational time for the MIMPC is significantly longer compared to the continuous MPC. Furthermore, spikes in computational time are observed in the case of the MIMPC, highlighting a critical aspect that requires careful consideration when utilizing integer programming in the optimization problem.

The increased computational time of the MIMPC is primarily attributed to the need to solve mixed-integer optimization problems, which are inherently more computationally demanding compared to continuous optimization problems. These challenges emphasize the importance of assessing the trade-off between control performance and computational efficiency when implementing the MIMPC approach. Efforts should be directed towards optimizing the computational aspects of the MIMPC, such as exploring improvements in the MPC algorithm and considering problem-specific simplifications.

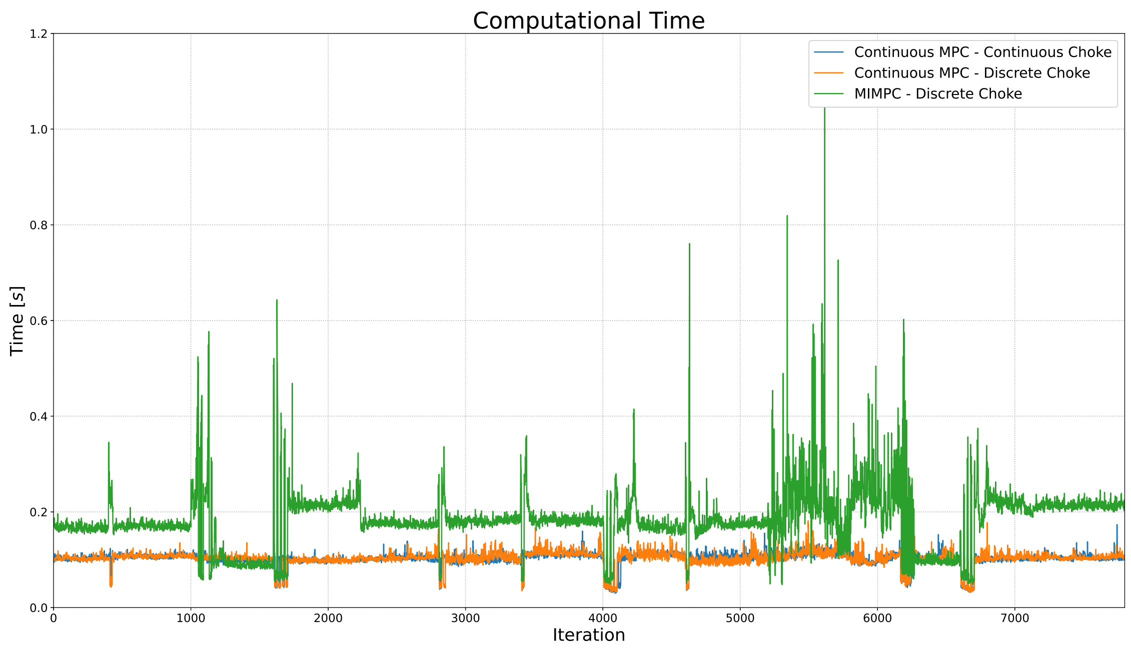

Figure 39: Computational time per iteration

It should be noted that the efficiency of the control algorithm has not been a focus in this study. There exist multiple techniques to improve the computational time, such as exploiting problem structure, reducing the number of blocks in the input blocking technique, or using incidence point techniques to evaluate constraints selectively. While these techniques have the potential to enhance efficiency, it is essential to consider the computational complexity associated with integer programming carefully. The current theory and algorithms in integer programming still present challenges in terms of computational requirements. On the other hand, as computer hardware becomes faster and mixed-integer optimization techniques become more advanced, it becomes increasingly possible to solve mathematical programming problems in real-time [38].

9 Conclusion And Further Work

In conclusion, this master thesis addresses the limitations of traditional continuous MPC approaches in managing non-continuous actuators, such as step chokes commonly found in the oil and gas industry. The proposed MIMPC method directly incorporates the discrete nature of the actuators into the optimization problem, resulting in improved control performance.

The thesis demonstrated that using rounding techniques in continuous optimization to handle discrete actuators does not provide optimal outcomes. Instead, the adoption of integer programming techniques in the MIMPC framework proves to be more effective. By introducing integer decision variables and leveraging mixed integer programming solvers, the MIMPC approach accurately accounts for the discrete steps in the choke and improves stability by eliminating the oscillations seen when using continuous control of discrete actuators.

Furthermore, the study explored the Soft MPC method as a solution to mitigate oscillations resulting from plant model mismatch. By establishing a deadzone around the setpoints, the Soft MPC method reduces penalties for small deviations, leading to improved stability and robustness of the control system.

The research findings underscored the superior performance of the MIMPC approach compared to traditional continuous MPC methods when handling non-continuous actuators. By accurately capturing the discrete behavior of the choke, MIMPC achieved improved control precision and stability. However, the inclusion of integer decision variables introduced additional computational complexity, which should be carefully evaluated in terms of practical feasibility.

Overall, this thesis provided valuable insights into the potential application of MIMPC in the oil and gas industry, offering a promising solution for improving control strategies and maximizing operational efficiency. The research contributes to advancing the field of control systems in the oil and gas sector, demonstrating the effectiveness of MIMPC and addressing plant model mismatch challenges.

9.1 Further Work

Although this study has demonstrated the benefits of utilizing MIMPC for managing non-continuous actuators in the oil and gas industry, there are several areas where the full potential of Mixed Integer Programming can be further explored. The following areas of further work present opportunities for utilizing MIP and MIMPC in various aspects of control and optimization in the oil and gas sector.

Control of Multiple Wells:
One potential avenue for future research is to extend the MIMPC approach to control multiple wells simultaneously. This would involve optimizing the operation of multiple wells, including turning them on or off based on real-time conditions. By considering the discrete nature of well operation, MIMPC could offer more efficient and coordinated control strategies, leading to improved production and resource management.

Power Usage Minimization:
Energy consumption is a significant concern in the oil and gas industry. MIMPC can be employed to minimize power usage by optimizing the operation of energy-consuming components, such as compressors, pumps, etc. By formulating the problem as a mixed integer program and integrating power consumption constraints, it becomes possible to find the optimal operating strategy that minimizes energy consumption while meeting production and process requirements.

Scheduling Optimization:
MIP and MIMPC can also be applied to scheduling optimization in the oil and gas industry. This includes optimizing the sequence and timing of various operations, such as maintenance activities, well interventions, and production schedules. By considering the discrete decisions involved in scheduling, MIP-based approaches can help minimize downtime, reduce costs, and maximize overall operational efficiency.

These are just a few examples of the potential applications of MIP and MIMPC in the oil and gas industry. The flexibility and capabilities offered by MIP techniques open up numerous opportunities for advanced control and optimization strategies. Further research in these areas can contribute to enhancing the efficiency, sustainability, and profitability of oil and gas operations, paving the way for improved decision-making and operational excellence.

Bibliography

[1] Simen Bergsvik. Mixed integer mpc. 2022.

[2] Jorge Nocedal and Stephen J Wright. *Numerical optimization.* Springer, 2nd edition, 2006.

[3] Laurence A Wolsey. *Integer programming.* John Wiley & Sons, 2020.

[4] Fields and platforms. https://www.equinor.com/energy/fields-and-platforms. Accessed: May 9, 2023.

[5] Oseberg. https://factpages.npd.no/pbl/field_jpgs/43625_Oseberg.jpg. Accessed: May 9, 2023.

[6] Lars Imsland, Bjarne A Foss, and Gisle Otto Eikrem. State feedback control of a class of positive systems: Application to gas-lift stabilization. In *2003 European Control Conference (ECC)*, pages 2499–2504. IEEE, 2003.

[7] Guru Prasath and John Bagterp Jørgensen. Soft constraints for robust mpc of uncertain systems. *IFAC Proceedings Volumes*, 42(11):225–230, 2009.

[8] Mohammad H Moradi. Predictive control with constraints, jm maciejowski; pearson education limited, prentice hall, london, 2002, pp. ix+ 331, price£ 35.99, isbn 0-201-39823-0, 2003.

[9] Stig Strand and Jan Richard Sagli. Mpc in statoil–advantages with in-house technology. *IFAC Proceedings Volumes*, 37(1):97–103, 2004.

[10] Kenneth Lange. *Optimization*, volume 95. Springer Science & Business Media, 2013.

[11] Edwin KP Chong and Stanislaw H Zak. *An introduction to optimization*, volume 75. John Wiley & Sons, 2013.

[12] Bjarne Foss and Tor Aksel N Heirung. Merging optimization and control. *Lecture Notes*, 2013.

[13] George Bernard Dantzig and Mukund N Thapa. *Linear programming: Theory and extensions*, volume 2. Springer, 2003.

[14] Marcelo Lopes de Lima, Eduardo Camponogara, Mario CMM de Campos, and Luis Kin Miyatake. Automatic control of flow gathering networks: A mixed-integer receding horizon control applied to an onshore oilfield. *Control Engineering Practice*, 86, 2019.

[15] Alain Billionnet and Éric Soutif. Using a mixed integer programming tool for solving the 0–1 quadratic knapsack problem. *INFORMS Journal on Computing*, 16(2), 2004.

[16] Jens Clausen. Branch and bound algorithms-principles and examples. *Department of Computer Science, University of Copenhagen*, pages 1–30, 1999.

[17] Sirikarn Chansombat, Pupong Pongcharoen, and Christian Hicks. A mixed-integer linear programming model for integrated production and preventive maintenance scheduling in the capital goods industry. *International Journal of Production Research*, 57(1):61–82, 2019.

[18] Ravindran Kannan and Clyde L Monma. On the computational complexity of integer programming problems. In *Optimization and Operations Research: Proceedings of a Workshop Held at the University of Bonn, October 2–8, 1977*, pages 161–172. Springer, 1978.

[19] Alberto Del Pia, Santanu S Dey, and Marco Molinaro. Mixed-integer quadratic programming is in np. *Mathematical Programming*, 162:225–240, 2017.

[20] Der-San Chen, Robert G Batson, and Yu Dang. *Applied integer programming: modeling and solution.* John Wiley & Sons, 2011.

[21] Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100, 1991.

[22] Gurobi optimization. https://www.gurobi.com/solutions/gurobi-optimizer/. Accessed: 2023-02-10.

[23] Cplex. https://www.ibm.com/products/ilog-cplex-optimization-studio?mhsrc=ibmsearch_a&amp;mhq=cplex. Accessed: 2023-04-21.

[24] Xpress. https://www.fico.com/en/products/fico-xpress-optimization. Accessed: 2023-04-21.

[25] Lars Magnus Hvattum, Arne Løkketangen, and Fred Glover. Comparisons of commercial mip solvers and an adaptive memory (tabu search) procedure for a class of 0–1 integer programming problems. *Algorithmic Operations Research*, 7(1):13–20, 2012.

[26] Cbc. https://coin-or.github.io/Cbc/intro.html. Accessed: 2023-04-21.

[27] Glpk. https://www.gnu.org/software/glpk/. Accessed: 2023-04-21.

[28] Ipopt. https://coin-or.github.io/Ipopt/. Accessed: 2023-04-21.

[29] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[30] Morten Hovd. Lecture notes for the course advanced control of industrial processes. *Department of Engineering Cybernetics*, 2009.

[31] Xingyu Zhou, Heran Shen, Zejiang Wang, and Junmin Wang. Individualizable vehicle lane keeping assistance system design: A linear-programming-based model predictive control approach. *IFAC-PapersOnLine*, 55(37):518–523, 2022.

[32] Dzordzoenyenye K Minde Kufoalor. High-performance industrial embedded model predictive control: Efficient implementation of step response models and fast solvers. 2016.

[33] Lars Grüne and Jürgen Pannek. *Nonlinear model predictive control*. Springer, 2017.

[34] Dale E Seborg, Thomas F Edgar, Duncan A Mellichamp, and Francis J Doyle III. *Process dynamics and control*. John Wiley & Sons, 2016.

[35] DKM Kufoalor, L Imsland, and TA Johansen. Efficient implementation of step response prediction models for embedded model predictive control. *IFAC-PapersOnLine*, 48(23):197–204, 2015.

[36] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[37] Robert D McAllister and James B Rawlings. Advances in mixed-integer model predictive control. In *2022 American Control Conference (ACC)*, pages 364–369. IEEE, 2022.

[38] Michael J Risbeck. *Mixed-integer model predictive control with applications to building energy systems*. The University of Wisconsin-Madison, 2018.

[39] Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[40] James B Rawlings and Michael J Risbeck. Model predictive control with discrete actuators: Theory and application. *Automatica*, 78:258–265, 2017.

[41] Stefano Di Cairano, WP Maurice H Heemels, Mircea Lazar, and Alberto Bemporad. Stabilizing dynamic controllers for hybrid systems: a hybrid control lyapunov function approach. *IEEE Transactions on Automatic Control*, 59(10):2629–2643, 2014.

[42] David Mayne. Robust and stochastic model predictive control: Are we going in the right direction? *Annual Reviews in Control*, 41:184–192, 2016.

[43] Pedro Hespanhol, Rien Quirynen, and Stefano Di Cairano. A structure exploiting branch-and-bound algorithm for mixed-integer model predictive control. In *2019 18th European Control Conference (ECC)*, pages 2763–2768. IEEE, 2019.

[44] Tobia Marcucci and Russ Tedrake. Warm start of mixed-integer programs for model predictive control of hybrid systems. *IEEE Transactions on Automatic Control*, 66(6):2433–2448, 2020.

[45] Håvard Devold. Oil and gas production handbook. *An introduction to oil and gas production, transport, refining and petrochemical industry*, page 162, 2013.

[46] J Lea, H Nickens, and M Wells. Gas lift. *Gas Well Deliquification (2 ed.), Gulf Professional Publishing*, page 333, 2008.

[47] Johan Åkesson, K-E Årzén, Magnus Gäfvert, Tove Bergdahl, and Hubertus Tummescheit. Modeling and optimization with optimica and jmodelica. org—languages and tools for solving large-scale dynamic optimization problems. *Computers & Chemical Engineering*, 34(11):1737–1749, 2010.

[48] Modelica association. https://modelica.org/publications/newsletters/2010-1/index_html#item8. Accessed: April 16, 2023.

[49] Functional mock-up interface. https://fmi-standard.org/. Accessed: April 16, 2023.

[50] Pyfmi. https://jmodelica.org/pyfmi/. Accessed: April 16, 2023.

[51] N Lawrence Ricker. Use of quadratic programming for constrained internal model control. *Industrial & Engineering Chemistry Process Design and Development*, 24(4):925–936, 1985.

[52] S Joe Qin and Thomas A Badgwell. A survey of industrial model predictive control technology. *Control engineering practice*, 11(7):733–764, 2003.

[53] Gurobi optimization integer tolerance. https://support.gurobi.com/hc/en-us/articles/360012237872-Why-does-Gurobi-sometimes-return-non-integral-values-for-integer-variables-. Accessed: 2023-05-10.

[54] Hisham Ben Mahmud, Van Hong Leong, and Yuli Lestariono. Sand production: A smart control framework for risk mitigation. *Petroleum*, 6(1):1–13, 2020.

[55] Charles R Cutler and Brian L Ramaker. Dynamic matrix control - a computer control algorithm. In *joint automatic control conference*, number 17, page 72, 1980.

[56] Chao-Ming Ying and Babu Joseph. Performance and stability analysis of lp-mpc and qp-mpc cascade control systems. *AIChE Journal*, 45(7):1521–1534, 1999.

[57] Glauce De Souza, Darci Odloak, and Antônio C Zanin. Real time optimization (rto) with model predictive control (mpc). *Computers & Chemical Engineering*, 34(12):1999–2006, 2010.

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

NTNU

Norwegian University of
Science and Technology