University of South-Eastern Norway

www.usn.no

FMH606 Master's Thesis 2021

Master of Science, Industrial IT and Automation

Optimal Operation of Processes Under Uncertainty Using Robust Model Predictive Control

Kushila Jayamanne

Faculty of Technology, Natural Sciences and Maritime Sciences
Campus Porsgrunn

University of South-Eastern Norway

www.usn.no

Course: FMH606 Master's Thesis, 2021

Title: Optimal Operation of Processes Under Uncertainty Using Robust Model Predictive Control

Number of pages: 42 pages (main report) + 30 pages (appendices)

Keywords: nonlinear model predictive control, robust, uncertainty, optimization, multi-stage, min-max, scenario tree

Student: Kushila Jayamanne

Supervisor: Associate Professor Roshan Sharma

External partners: Equinor, Skagerak Energi, SINTEF

Summary:

Model Predictive Control (MPC) is a well-established technology for advanced control of many industrial processes. One of its main benefits is the ability to handle process constraints. However, accuracy of the process model is critical to the output of standard/nominal MPC, and typically many sources of uncertainty exist. Even though it has some degree of inherent robustness—under quite strict assumptions—performance is usually insufficient for nonlinear systems.

The main goal of this thesis is to study existing robust Nonlinear Model Predictive Control (NMPC) approaches, specifically multi-stage NMPC and min-max NMPC, and design a robust controller for a real industrial process.

To this end, a literature review was conducted, covering the most prominent robust MPC techniques available today as well as the challenges involved. Then ways of addressing the issues of computational complexity and conservativeness were explored briefly. Finally, based on the acquired knowledge, robust NMPC was implemented on an oil production optimization case study.

Simulation results showed that, in contrast to standard NMPC, both multi-stage NMPC and its min-max variation can ensure constraint satisfaction for all possible values of the uncertainties—*if* properly designed. In terms of conservativeness, both performed equally well, but it was clear that multi-stage NMPC is the more computationally attractive choice.

It was also seen that there exist methods for successfully dealing with the inevitable loss of performance resulting from robust NMPC; results suggest that if an efficient implementation is carried out, for certain cases, even real-time implementation may be possible.

Overall, the thesis findings indicate that robust NMPC is an essential tool for advanced control of industrial processes and that the multi-stage NMPC approach in particular is rather promising.

Preface

This thesis was written during the spring of 2021 to fulfil the graduation requirements of the Master of Science degree in Industrial IT and Automation at the University of South-Eastern Norway (USN). The work that was involved allowed me to delve deeper into the field of model predictive control and expand my knowledge of industrial IT and automation. I am sure that the lessons learned, and the experiences gained will be beneficial to me in the future.

I would like to thank my supervisor, Associate Professor Roshan Sharma at the Department of Electrical Engineering, IT and Cybernetics at USN, Porsgrunn, for the invaluable guidance, insightful discussions, helpful critiques, and continued patience throughout these past few months. I am very grateful to have had the opportunity to write this thesis under his supervision. I also extend my gratitude to Beathe Furenes (Overingeniør Hydrologi, Skagerak Kraft) for introducing and providing information about a challenging, yet interesting case study for my work.

A very special word of thanks to my friends, the ones I knew and the ones I made here, for providing the right amount of support and distraction; for all the good times and the food shared.

And finally, I would like to thank my family for being an endless source of support and love. I owe you everything.

Porsgrunn, May 2021

Kushila Jayamanne

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Model Predictive Control (MPC) is a well-established technology used for advanced control of many industrial processes. One of its major advantages is the ability to handle process constraints. However, performance of standard/nominal MPC depends heavily on the accuracy of the process model and typically many sources of uncertainty exist, e.g., incomplete plant dynamics, uncertain model parameters, uncertain variable disturbances, etc. Even though standard MPC is known to possess a degree of inherent robustness, if state or terminal constraints are present in combination with a short prediction horizon, the ability to handle process constraints may be lost [1]. And in any case, for nonlinear systems, its performance—under the presence of uncertainties—is usually not sufficient.

The subject of robust Nonlinear Model Predictive Control (NMPC) has received a lot of attention in recent years. Unlike standard NMPC, robust NMPC is designed to be resilient in the face of uncertainty; the Optimal Control Problem (OCP) is formulated such that constraints are satisfied for all possible values of the uncertainty. Although a variety of robust schemes have been proposed, and all of them promise robust constraint satisfaction, they are not without their drawbacks.

1.2 Aim and Scope of the Thesis

The aim of this thesis is to investigate existing robust NMPC techniques and apply some of these to a real industrial case study. The work involves

- Reviewing the literature on robust NMPC techniques, particularly non-conservative ones,
- Implementing multi-stage and min-max NMPC on an industrial case study, and
- Analysing and comparing the performance of the two techniques.

Computational complexity and conservativeness are two of the biggest issues concerning the practical implementation of robust NMPC in an industrial setting. Therefore, additionally a brief review of potential methods for dealing with these issues is conducted and accordingly an effort is made to incorporate/implement them in the simulation studies.

1.3 Structure of the Thesis

After this introductory chapter the thesis is organized as follows.

Chapter 2 presents a literature review of the most prominent robust NMPC techniques available today. Chapter 3 introduces the theoretical framework of the considered robust NMPC techniques—multi-stage NMPC and min-max NMPC. It also presents several ways of improving the performance of robust NMPC and provides details of implementation. Chapter 4 presents the central results of this thesis. It details the application of robust NMPC to a challenging industrial case study, investigates the various aspects of the NMPC approaches, and compares their performance. And finally, chapter 5 presents the main conclusions and directions for future work.

2 Literature Review

Even though a wide range of robust NMPC methods exist, most of the literature is dominated by three specific ones: min-max NMPC, tube-based NMPC, and multi-stage NMPC; all three deal with cases where the uncertainty lies within a specified bounded set and all three try to guarantee robust constraint satisfaction. These techniques are the focus of this chapter.

2.1 Min-Max NMPC

The classical min-max MPC formulation [2] uses an open-loop decision variable for optimizing the open-loop performance. It aims to find a single sequence of control inputs, which minimizes the cost associated with the worst-case scenario of the uncertainty, and also satisfies the constraints for all possible scenarios; the fact that feedback is present in the sliding-horizon implementation is ignored. This results in extreme conservativeness in the optimal solution. Furthermore, in most cases the control strategy becomes inapplicable due to infeasibilities that occur during optimization: if the spread of the state trajectories is large, it may be impossible to satisfy the constraints for all possible scenarios at once [3].

To deal with these issues, closed-loop min-max NMPC formulations, which take into account the fact that new information will be available in the next timestep have also been proposed [3] [4]. They are sometimes referred to as “feedback MPC” methods. In these, the optimization is done over a sequence of feedback control laws—as opposed to a sequence of control inputs—that can limit the spread of the state trajectories. While the performance is improved in terms of conservativeness and feasibility, the complexity of the resulting OCP—which could also possibly be infinite-dimensional for nonlinear systems [5]—makes min-max feedback NMPC computationally intractable.

Another method of min-max feedback MPC for linear systems, which is based on the representation of the uncertainty using a scenario tree, is proposed in [6]. Instead of optimizing over a sequence of feedback policies, like [3] and [4] does, it optimizes over a set of control input sequences, each corresponding to a different scenario. It is shown that robust constraint satisfaction for all possible realizations of the uncertainty can be ensured by including only the extreme realizations, which occur at the vertices of the uncertainty set, in the scenario tree. But the same is not true when nonlinearity is present. Hence, this method cannot be applied to nonlinear systems. Moreover, the resulting problem is still computationally expensive when a larger prediction horizon is involved.

2.2 Tube-Based NMPC

An alternative to min-max NMPC, which guarantees recursive feasibility, is tube-based NMPC [7]. Set-theoretic methods are at the foundation of this technique. It tries to maintain the trajectory of the system within a tube-shaped region centred around the nominal trajectory. This is achieved with the use of two controllers: the nominal controller and the ancillary controller. The nominal controller generates the central trajectory by solving the nominal NMPC problem with tightened constraints—uncertainties, which disturb the nominal trajectory, are ignored. Then the ancillary controller ensures

that the disturbed trajectory of the actual system stays close to the computed nominal trajectory, within the said tube-shaped region.

There are several methods for determining the uncertainty region around the nominal trajectory, as well as various ancillary controllers; several variants of tube-based NMPC that mainly differ in these regards exist [8] [9]. However, for nonlinear systems, finding the necessary elements for the design of tube-based NMPC is typically very difficult. Furthermore, despite the fact that it guarantees robust constraint satisfaction, tube-based NMPC does not address the problem of achieving optimal operation *under* the presence of uncertainties [10].

2.3 Multi-Stage NMPC

In contrast to tube-based NMPC, multi-stage NMPC deals with achieving optimal operation under the presence of uncertainty. It can be viewed as an adaptation of the min-max feedback MPC method for linear systems presented in [6], for nonlinear systems, which also addresses the problems present in [6]; it, too, relies on the representation of the uncertainty by a scenario-tree and explicitly takes into account the fact that new information will be available in the next timestep and hence the corresponding control moves can be adapted accordingly in the future [11]. This approach is considered to lead to a non-conservative solution.

It is important to note, however, that multi-stage NMPC guarantees constraint satisfaction only for those values of the uncertainty considered in the generation of the scenario tree [12] [10]. In the case of continuous-valued uncertainty, representing all possible scenarios in the tree is impractical. But even though the values of the uncertainties that produce the worst-case scenario could lie anywhere in the specified intervals, they often lie on the boundaries [13]—much like the case with linear systems [6]. Hence, combinations of the extreme values of the uncertain parameters should be included in the scenario tree to achieve robust constraint satisfaction [12]; this has been shown to work well in practice [11] [14]. Additionally, intermediate scenarios should also be included to increase performance [11]. A good rule of thumb is to include the combinations of the maximum, minimum and nominal values of the uncertain parameters in the scenario tree [15]; this provides a good trade-off between performance and computational complexity.

One of the main challenges with multi-stage NMPC is the exponential growth of the scenario tree with the number of possible realizations of the uncertainty considered in its creation and the length of the prediction horizon—which is also the case with [6]. This is dealt with the use of the robust horizon assumption [11]: branching of the tree occurs only up to a certain point in time (i.e., the robust horizon) and after this point, for the remainder of the prediction horizon, the uncertainty remains constant. This has its basis on the fact that, because of the presence of feedback in the sliding-horizon implementation, the scenario tree need not represent the far future accurately [12].

It is also worthy to note that multi-stage NMPC includes the standard NMPC formulation as well as a closed-loop min-max NMPC formulation in its framework [11].

3 Background

This chapter introduces the theoretical concepts that form the basis for conducting the final case study. The first section provides a brief introduction to each of the two robust NMPC strategies under consideration in this thesis: multi-stage NMPC and min-Max NMPC. The problems of computational complexity and conservativeness associated with them are addressed in the second section; several methods for dealing with these issues are discussed. The last section provides details of implementation.

3.1 Introduction to Robust NMPC

This section details the mathematical formulations of multi-stage NMPC and min-max NMPC.

3.1.1 Multi-Stage NMPC

In multi-stage NMPC, the uncertainty is modelled using a scenario tree, as shown in Figure 3.1. The tree is branched at all the nodes present at each timestep/stage; the number of branches per node equals the number of possible discrete values of the uncertainty and each node represents a possible state of the system at the corresponding timestep—the initial node represents the current state of the system, which is known.

As explained in section 2.3, to limit the growth of the tree the robust horizon assumption is made: the tree stops branching, i.e., the uncertainty remains constant, after a certain point in time.

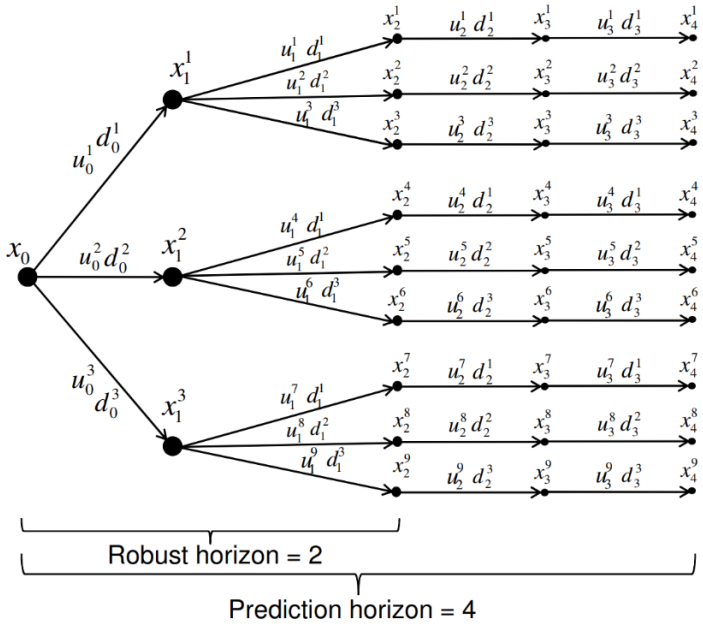

Figure 3.1: Example scenario tree. [12]

If the values of the uncertainties are not discrete, a good rule of thumb for creating the scenario tree is to include the combinations of the extreme and nominal values of the uncertainties.

The optimization problem resulting from the multi-stage NMPC approach is given in equation (3.1) - (3.4).

$$\min_{x_{k+1}^j, u_k^j, \forall (j,k) \in I} \sum_{i=1}^{N} \omega_i J_i(X_i, U_i) \tag{3.1}$$

subject to

$$x_{k+1}^j = f\left(x_k^{p(j,k)}, u_k^j, d_k^{r(j,k)}\right), \qquad \forall\, (j, k+1) \in I \tag{3.2}$$

$$0 \geq g\left(x_{k+1}^j, u_k^j, d_k^{r(j)}\right), \qquad \forall\, (j,k) \in I \tag{3.3}$$

$$u_k^j = u_k^l \text{ if } x_k^{p(j)} = x_k^{p(l)}, \qquad \forall\, (j,k), (l,k) \in I \tag{3.4}$$

where J_i and ω_i is the cost and probability of occurrence of the i^{th} scenario; the rest of the notation is as described in [12].

Equation (3.2) represents the state trajectory of the uncertain nonlinear system; equation (3.3) represents general, possibly nonlinear, constraints; and equation (3.4) represents the non-anticipativity constraints, which enforces the fact that, at each timestep, the uncertainty is realized only after the control inputs have been decided and hence its value cannot be anticipated, i.e., the control inputs branching at the same parent node must be the same.

3.1.2 Min-Max NMPC

As discussed in section 2.1, the classical min-max MPC formulation, which optimizes over a single sequence of control inputs, is known to be either extremely conservative or simply infeasible, and the closed-loop min-max NMPC formulations, which optimize over a sequence of feedback control laws, are likewise typically computationally intractable. A more realizable option is the closed-loop min-max NMPC formulation present within the multi-stage NMPC framework; this is chosen for the purposes of simulation studies in this thesis.

The min-max approach is obtained by simply replacing the summation in equation (3.1) with the maximization operator:

$$\min_{x_{k+1}^j, u_k^j, \forall (j,k) \in I} \max \omega_i J_i(X_i, U_i) \tag{3.5}$$

subject to

$$x_{k+1}^j = f\left(x_k^{p(j,k)}, u_k^j, d_k^{r(j,k)}\right), \qquad \forall\, (j, k+1) \in I \tag{3.6}$$

$$0 \geq g\left(x_{k+1}^j, u_k^j, d_k^{r(j)}\right), \qquad \forall\, (j,k) \in I \tag{3.7}$$

$$u_k^j = u_k^l \text{ if } x_k^{p(j)} = x_k^{p(l)}, \qquad \forall\, (j,k), (l,k) \in I \tag{3.8}$$

where the notation is the same as described in section 3.1.1.

3.2 Improving the Performance of Robust NMPC

This section looks at several methods that can be used to improve the performance of multi-stage and min-max NMPC in terms of computational complexity and conservativeness.

3.2.1 Computational Complexity

3.2.1.1 Direct Multiple-Shooting Method for Transcribing the OCP into an NLP

The dynamics of a system is typically described by a system of Ordinary Differential Equations (ODEs) or Differential Algebraic Equations (DAEs). Hence, the OCP associated with MPC will be infinite-dimensional. The type of methods that are state-of-the-art for solving this type of problems are the so-called “direct methods,” which are also referred to as “discretize-optimize” approaches. In these, the continuous control trajectory is replaced with a parameterization, e.g., piecewise constant trajectory. This allows the reformulation/transcription of the original infinite-dimensional OCP into a finite-dimensional NLP, which can be solved using available NLP solvers. There are several ways in which transcription can be performed and the chosen method affects the solvability and scalability of the problem.

Here, the focus is placed on two of the popular direct methods: direct single-shooting and direct multiple-shooting. Specifically, the advantages of using multiple-shooting over single-shooting, with multi-stage NMPC, is discussed.

The main difference between the two methods is that in single-shooting only the control trajectory is discretized and used as decision variables, whereas in multiple-shooting both the control trajectory and state trajectory are discretized and used as decision variables; the reader is referred to [16] for more detailed descriptions.

When solving NLPs produced using the single-shooting approach, most of the time is taken up by the sequential solution of initial value problems, which enforce the continuity of the state trajectory, at each iteration. While there are other disadvantages with single-shooting, in the context of multi-stage NMPC this constitute the main one: difficulty to parallelize; as much initial value problems as the number of scenarios must be solved at each iteration.

In contrast, in the multiple-shooting approach the continuity of the state trajectory is enforced using a set of equality constraints; essentially the solution of the initial value problems at each iteration is parallelized. The resulting NLP is much larger but is known to converge faster to the optimal solution [17].

3.2.1.2 Automatic Differentiation (AD) for Calculation of Derivatives

The NLP solver used in this work applies a gradient-based optimization algorithm, which requires derivative—first order and preferably also second order derivative—information of the objective function and the constraints. There exist three main methods for the computation of derivatives: numerical differentiation, symbolic differentiation, and automatic differentiation. The performance of the NLP solver is greatly affected by the efficiency and accuracy of the chosen method.

Numerical differentiation provides only an approximation of the derivatives. While reasonable approximations may be obtained for first order derivatives, the errors present in approximations of higher order derivatives tend to be quite significant.

On the other hand, symbolic differentiation provides exact derivatives but, even for moderately complex functions, the produced symbolic expressions of the derivatives tend to be rather unwieldy and hence expensive to evaluate.

Automatic differentiation too provides exact derivatives, however, unlike symbolic differentiation, generation of the expressions of derivatives is not the goal. Instead, the derivatives are computed through decomposition of the function into in a sequence of primitive operations (e.g., addition, subtraction, multiplication, etc.), which, when taken by themselves, are easily differentiable; the partial derivatives of the intermediate variables are combined using the chain rule to obtain the derivative of the original function. This method of derivative calculation is not only exact but is also very efficient.

3.2.1.3 Grouping of Control Inputs

Another potential method that can be used to tackle the problem of high computational cost associated with multi-stage NMPC is control input grouping [18]. In contrast to the direct multiple-shooting approach, in this approach, cost reduction is achieved by reducing the size of the OCP. It is quite similar to the concept of control horizon, in which the degrees of freedom are reduced by taking the control input trajectory to be constant after a certain point in time—until the end of the prediction horizon.

In this method the (discrete) control input, which maybe be a scalar or a vector, is grouped along the prediction horizon, and within each group the signal is kept constant. An example grouping scheme for a prediction horizon of 20 timesteps is illustrated in Figure 3.2; the case of standard NMPC is considered. As seen, the 20 discrete unknown values of the control signal are grouped into three blocks. This means that the number of decision variables in the OCP—corresponding to the control inputs—reduces from 20 to 3 (assuming the control input is a scalar), i.e., the size of the OCP reduces. It is worthy to highlight that even though grouping gives rise to additional equality constraints—which may seem to increase in the size of the problem—the computation time required for solving the optimization problem is more dependent on the number of decision variables [18].

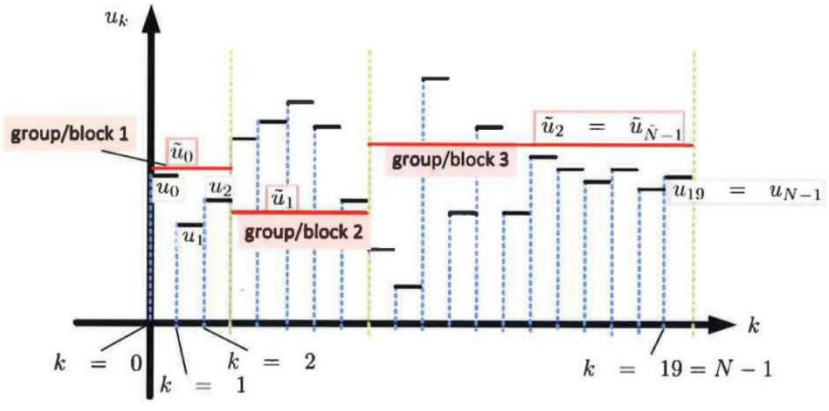

Figure 3.2: An example grouping of control inputs along a prediction horizon of 20 timesteps. [18]

Each block may be chosen to be longer than the preceding block or to be the same length as the preceding block [18]. However, it is advisable to try several different sensible grouping schemes and

compare the results against the original results (i.e., without grouping) to select the most effective one.

It is important to note that, in the case of multi-stage NMPC, grouping is done for each scenario, over the stages occurring after the robust horizon—not over the entire prediction horizon.

3.2.2 Conservativeness

It is known that all robust NMPC techniques result in an inevitable loss of optimality. So, even though multi-stage NMPC is known as a non-conservative method, it too will lead to some degree of conservativeness. Therefore, the best recourse is to remove as much uncertainty as possible through estimation, *if* the uncertainty does not vary over time [10]; whenever new information about the uncertainty is available it can be fed online to the multi-stage NMPC algorithm. The effectiveness of such an approach will be tested in this work.

The online estimation method proposed in [19] is considered. In broad terms, the approach can be summarized as follows. At each timestep the process measurements are compared to the predictions from each scenario, and based on this comparison, a recursive Bayesian weighting scheme is used to assign weights to each of the scenarios; scenarios with a higher probability of being the true realization are assigned larger weights, whereas those with a lower probability are assigned smaller weights. When the weight assigned to a scenario falls below a specified threshold, that scenario is updated in the direction of the scenario with the largest weight (i.e., the most likely scenario) at the time. As a result, the span of the uncertainty space is reduced. Consequently, the span of the scenario tree is also reduced, resulting in a decrease in conservativeness (note: the fact that the span of the uncertainty space affects the degree of conservativeness will be demonstrated through simulation studies in the following chapter). For further details about this estimation method and its algorithm, the reader is referred to [19].

3.3 Implementation Details

Based on the discussions in section 3.2, an effort is made to implement the NMPC schemes in a more efficient way. The details of the implementation are as follows.

For discretizing the dynamics of the system, the Runge-Kutta fourth order (RK4) integration scheme is used together with the direct multiple-shooting approach. Interior Point (IP) methods have been shown to perform better than Sequential Quadratic Programming (SQP) methods at solving the large-scale optimization problems resulting from multi-stage NMPC [11]. Hence, IPOPT [20] is used for solving the NLPs in this thesis and the CasADi framework [21] is used to automatically calculate and pass the first order and second order exact derivatives required by the solver.

The real plant is simulated, with the calculated optimal control inputs, using also RK4. For simplicity, it is assumed that the exact measures of all the states are available at each time step; state-feedback MPC is used.

All the implementations are carried out in MATLAB using a 4-core Intel i5 processor at 2.40 GHz, and 8 GB of RAM.

4 Case Study - Oil Production Optimization

This chapter presents the central case study in this thesis: optimization of oil production in a gas-lifted oil field. It constitutes a challenging real-world industrial case that involves highly nonlinear process dynamics and tight constraints; ensuring optimal operation of the plant through manual control is, in general, extremely difficult. The matters are further complicated by the presence of uncertainty in the system—which makes satisfaction of the tight constraints harder still. This chapter explores the possibility of employing NMPC for achieving the control objectives. Three different NMPC approaches—standard, multi-stage and min-max—are implemented on the oil production process and their performance, in the presence of the same uncertainty conditions, is analysed and compared.

The first section of this chapter provides a description of the gas-lifted oil production process; the considered mathematical model and the modelled uncertainties are also introduced. This is followed by a preliminary study of the process dynamics, through open-loop simulation, in the second section of the chapter. Here, the simulation results are used to establish the need for dynamic optimal control.

The third section describes the implementation of standard NMPC—whose formulation does not take into account the presence of uncertainties—on the oil production process. The resulting performance when uncertainty is absent vs. when it is present is evaluated. This leads to affirmation of the need for exploring the feasibility of robust NMPC techniques—which explicitly consider the uncertainties present in the system—for avoiding constraint violations.

The fourth and fifth sections of this chapter deal with the implementation of the two robust NMPC approaches—multi-stage NMPC and min-max NMPC—on the process. It is shown that both approaches have superior performance compared to standard NMPC in that they ensure satisfaction of all the constraints in the presence of uncertainties. Also, a comparison of the performance of the two, taken by themselves, is made and the issues of computational complexity and conservativeness are briefly looked at.

The simulation results are summarized in the final section of this chapter.

4.1 Process Description

Continuous extraction of oil from a reservoir causes a gradual reduction of reservoir pressure. Once the pressure drops below the point where it is no longer sufficient to lift the fluid column produced in an oil well, an artificial lift method must be used. The process investigated in this case study is the production of oil using the continuous-flow gas lift mechanism: a predominant artificial lift method used to keep up production when the natural flow of oil from a reservoir ceases (or drops below the desired rate).

The principle of operation of the continuous-flow gas lift mechanism is illustrated in Figure 4.1. High-pressure gas is injected into the hydrostatic column inside the tubing, from an appropriate distance down the well. The injected gas mixes with the produced oil and reduces the density of the hydrostatic column; consequently, the back pressure is reduced. The resulting differential pressure is what allows the continuation of fluid flow along the well.

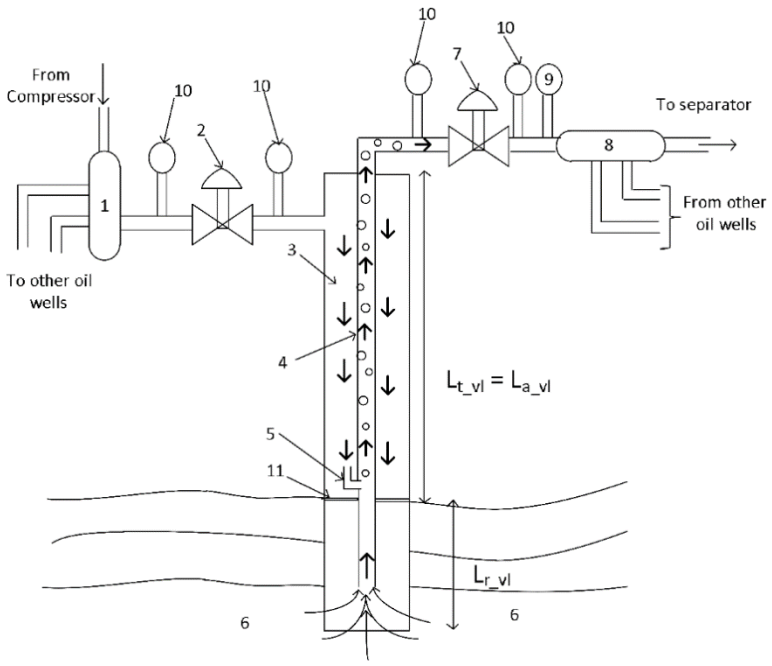

Figure 4.1: Principle of operation of continuous-flow gas lift. [22]

However, a higher gas injection rate does not always yield a higher oil production rate; Figure 4.2 shows the relationship between the two quantities. It can be seen that an optimum gas injection rate, which maximizes oil production, exists. This optimum will be different for different oil wells, i.e., two unidentical wells will produce different amounts of oil when injected the same amount of lift-gas.

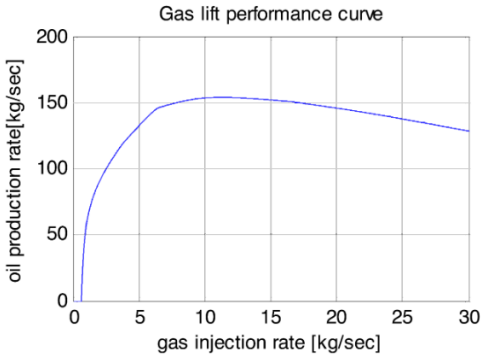

Figure 4.2: Gas-lift performance curve of a typical oil well. [22]

Given the fact that multiple *unidentical* wells are present in an oil field and that they all share the same source of lift-gas (at least in the case considered in this work), it is easily seen why proper control of the gas injection rate for each well is required. (Note: The need for dynamic control will be discussed in more details in section 4.2, in relation to the system considered in this work.)

4.1.1 Mathematical Model

A simple model of a gas lifted oil field with five oil wells, which has been appropriately validated against data from a real oil field, is proposed in [22]. For the purposes of this case study, an adaptation of the said model is used. The adapted version is presented here. However, it should be noted that only the relevant equations are listed, along with the definitions of the variables and the set of model parameters used in this work. The reader is referred to [22] for further details about the model.

A schematic of the oil field under consideration is given in Figure 4.3. The system consists of two oil wells, which share the same source of lift-gas. The fluid, i.e., the mixture of lift-gas, water and oil extracted from the reservoir, produced from both wells is sent to a separator, which separates it into its constituents; the lift-gas is recycled.

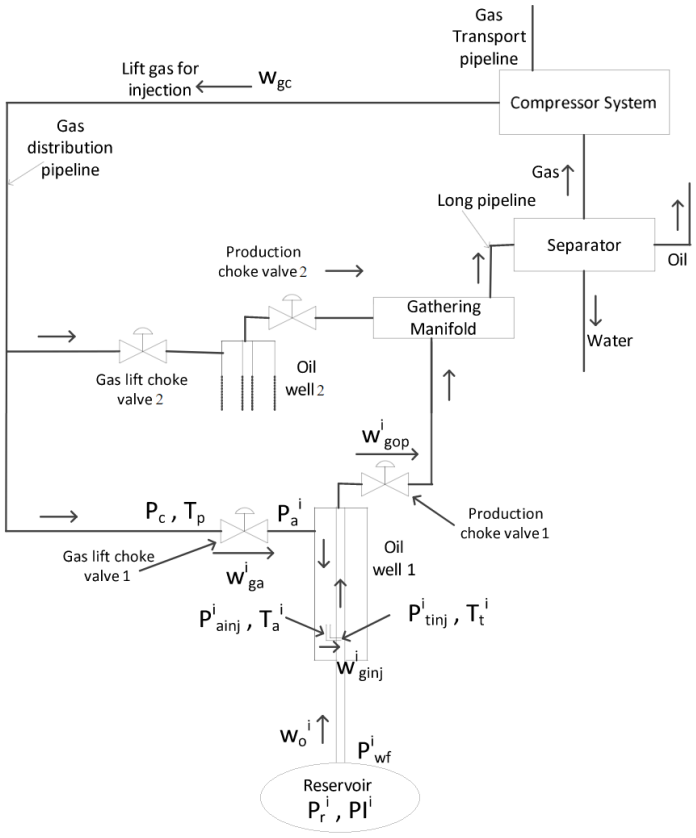

Figure 4.3: Schematic of the oil field. Adapted from [22]

The system is modelled by a set of differential-algebraic equations. It consists of six states m: mass of gas in the annulus m^i_{ga}; mass of gas in the tubing above the injection point m^i_{gt}; and mass of oil in the tubing above the injection point m^i_{ot} (superscript i denotes the i^{th} oil well). These are all obtained using mass balance and are given by equation (4.1) - (4.3).

$$\dot{m}^i_{ga} = w^i_{ga} - w^i_{ginj} \tag{4.1}$$

$$\dot{m}^i_{gt} = w^i_{ginj} - w^i_{gp} \tag{4.2}$$

$$\dot{m}^i_{ot} = w^i_{o} - w^i_{op} \tag{4.3}$$

where w_{ga}^i is the mass flow rate of gas leaving the distribution pipeline and entering the annulus via the gas-lift choke valve, and it is taken as the control input; w_{ginj}^i is the mass flow rate of gas injected into the tubing, from the annulus, via the gas injection valve; w_{gp}^i and w_{op}^i are the mass flow rates of gas and oil through the production choke valve; w_o^i is the mass flow rate of oil flowing into the tubing from the reservoir. The expressions required for the computation of these are given by equation (4.4) - (A.12).

$$w_{ginj}^i = K^i Y_2^i \sqrt{\rho_{ga}^i \max\left(P_{ainj}^i - P_{tinj}^i, 0\right)} \tag{4.4}$$

$$w_{gp}^i = \frac{m_{gt}^i}{m_{gt}^i + m_{ot}^i} w_{gop}^i \tag{4.5}$$

$$w_o^i = PI^i \max\left(P_r - P_{wf}^i\right) \tag{4.6}$$

$$w_{op}^i = \frac{m_{ot}^i}{m_{gt}^i + m_{ot}^i} w_{gop}^i \tag{4.7}$$

$$Y_2^i = 1 - \alpha_Y \left(\frac{P_{ainj}^i - P_{tinj}^i}{\max\left(P_{ainj}^i, P_{ainj}^{min}\right)} \right) \tag{4.8}$$

$$\rho_{ga}^i = \frac{M\left(P_a^i + P_{ainj}^i\right)}{2zRT_a^i} \tag{4.9}$$

$$P_{ainj}^i = P_a^i + \frac{m_{ga}^i}{A_a^i L_{a\_tl}^i} g L_{a\_vl}^i \tag{4.10}$$

$$P_{tinj}^i = \frac{z m_{gt}^i R T_t^i}{M V_G^i} + \frac{\rho_m^i g L_{t\_vl}^i}{2} \tag{4.11}$$

$$w_{gop}^i = 10 N_6 C_v\left(u_2^i\right) Y_3^i \sqrt{\rho_m^i \max\left(P_{wh}^i - P_s, 0\right)} \tag{4.12}$$

$$P_{wf}^i = P_{tinj}^i + \rho_o g L_{r\_vl}^i \tag{4.13}$$

$$P_a^i = \frac{z m_{ga}^i R T_a^i}{M A_a^i L_{a\_tl}^i} \tag{4.14}$$

$$V_G^i = A_a^i L_{t\_tl}^i - \frac{m_{ot}^i}{\rho_o} \tag{4.15}$$

$$\rho_m^i = \frac{m_{gt}^i + m_{ot}^i}{A_t^i L_{t\_tl}^i} \tag{4.16}$$

$$C_v\left(u_1^i\right) = \begin{cases} 0, & u_1^i < 5 \\ 0.111 u_1^i - 0.556, & 5 < u_1^i < 50 \\ 0.5 u_1^i - 20, & 50 < u_1^i \end{cases} \tag{4.17}$$

$$Y_3^i = 1 - \alpha_Y \left(\frac{P_{wh}^i - P_s}{\max\left(P_{wh}^i, P_{wh}^{min}\right)} \right) \tag{4.18}$$

$$P_{wh}^i = \frac{z m_{gt}^i R T_t^i}{M V_G^i} - \frac{\rho_m^i g L_{t\_vl}^i}{2} \tag{4.19}$$

where Y_2^i is the gas expansion factor across the gas injection valve; ρ_{ga}^i is the density of gas in the annulus; P_{ainj}^i is the pressure upstream of the gas injection valve—in the annulus; P_{tinj}^i is the pressure downstream of the gas injection valve—in the tubing; w_{gop}^i is the mass flow rate of the mixture of gas

and oil through the production choke valve; P_{wf}^{i} is the bottom hole pressure; P_{a}^{i} is the pressure downstream of the gas-lift choke valve; V_{G}^{i} is the volume of gas in the tubing above the injection point; ρ_{m}^{i} is the average density of the mixture of oil and gas in the tubing above the injection point; C_v is the production choke valve characteristic, which is a function of its opening, u_2^i—u_2^i is taken to be 100% for both valves, throughout this work and hence only the last equation of (A.12) is relevant; Y_3^i is the gas expansion factor across the production choke valve; P_{wh}^{i} is the pressure upstream of the production choke valve—in the tubing head.

The complete set of model parameters used in this work is given in Table 4.1.

Table 4.1: Parameters of the oil field model

Parameter	Description	Value		Unit
		well-1	well-2	
K	Gas injection valve constant	68.43	67.82	$\frac{\sqrt{\text{kgm}^3/\text{bar}}}{\text{hr}}$
PI	Productivity index	2.51×10^4	1.63×10^4	kg/hr/bar
P_r	Reservoir pressure	150		bar
α_Y	A constant	0.66	0.66	-
P_{ainj}^{min}	Minimum pressure of lift-gas in the annulus, at the point of injection into the tubing	0	0	bar
M	Molar mass of the lift-gas	0.020		kg/mol
z	Gas compressibility factor	1.3		-
R	Universal gas constant	8.3145		J/K/mol
T_a	Average temperature of lift gas in the annulus	280	280	K
A_a	Cross sectional area of the annulus	$=\frac{\pi}{4}(ID_a^2 - OD_t^2)$		m^2
L_{a_tl}	True/actual length of the annulus	2758	2559	m
g	Gravitational acceleration	9.8066		m^2/s
L_{a_vl}	Vertical length of the annulus from the well head to the point of injection	2271	2344	m
T_t	Average temperature of fluid in the tubing	280	280	K
L_{t_vl}	Vertical length of the tubing above the gas injection point	2271	2344	m
N_6	Gas lift choke valve constant	27.3	27.3	
P_s	Pressure of the common gathering manifold	30	30	bar
ρ_o	Density of crude oil	700		kg/m^3
L_{r_vl}	Vertical length of the tubing below the gas injection point up to the reservoir opening	114	67	m
A_t	Inner cross-sectional area of the tubing	$=\frac{\pi}{4}ID_t^2$		m^2
L_{t_tl}	Actual length of the tubing above the gas injection point	2758	2559	m
P_{wh}^{min}	Minimum pressure in the tubing at the well head	0	0	bar

The mass flow rate of gas entering the distribution pipeline from the compressor w_{gc} is considered a known input disturbance to the process. Since both wells share the lift-gas supplied by the compressor and given the fact that the total mass flow rate of gas leaving the distribution pipeline at any time could not be greater than the mass flow rate of gas entering the pipeline, the constraint given by equation (4.22) exists.

$$\sum_{i=1}^{2} w_{ga}^{i} \leq w_{gc} \tag{4.20}$$

Additionally, the presence of a processing capacity of 160 kg/s for the separator is assumed, i.e., the two oil wells should be operated such that their combined fluid production does not exceed the processing capacity of the separator. This gives rise to the constraint given by equation (4.23).

$$\sum_{i=1}^{2} w_{gop}^{i} \leq 160 \text{ kg/s} \tag{4.21}$$

The most significant differences in this adapted model in comparison to the original model can be listed as follows.

- *The presence of only two wells—Well 1 and Well 2 in* [1 Tab. 2]*—instead of five*. The reduction is made to lower the computational cost associated with testing the different NMPC strategies; as stated in section 2, especially min-max and multi-stage NMPC, involve the solution of computationally complex NLPs.
- *The omission of the dynamics associated with the gas distribution manifold and the gas-lift choke valves*. A control structure, such as the one used in [23], for maintaining the pressure inside the gas distribution manifold close to a given setpoint, in the events of fluctuations in the lift-gas supply, is assumed. Accordingly, the optimal lift-gas flow rates w_{ga}^{*} found by the NMPC algorithm are intended to serve as set points to the PID controllers operating the gas-lift choke valves.
- *The presence of a processing capacity constraint on the separator*. This assumption is made in order to have a constraint through which model uncertainty could possibly manifest during NMPC (note: this will be made clearer later in this chapter, in the sections corresponding to implementation of the different NMPC schemes).
- *The assumption of a constant gas compressibility factor* z, for simplicity.

4.1.2 Uncertainties

For the purposes of this work, only the uncertainty present in the Productivity Index (PI) value of the two wells is considered; everything else is assumed known/certain. It is also assumed that the PI is a time-invariant parameter.

As stated in [22] the PI values of the considered wells have been found by aggregating a year's worth of data from the real oil field. The values thus found are considered the nominal values. The two wells chosen for this work have the nominal values $2.51\text{x}10^4$ kg/hr/bar and $1.63\text{x}10^4$ kg/hr/bar.

The uncertainty is modelled as the error in the PI value, $d_{PI} = [d_{PI}^{1}, d_{PI}^{2}]$. For both wells, it is considered to lie within the range $d_{PI}^{i} \in [-0.25, 0.25] \times 10^4 \text{ kg/hr/bar}$ (note: the scaling factor

$\times 10^4$ is dropped from the notation for simplicity in the remainder of the report) and equal probability of occurrence is assumed for all possible values. The considered range corresponds to an uncertainty of ~±10% in the PI value of well-1 and to an uncertainty of ~±15% in the PI value of well-2. Figure 4.4 provides a simple illustration of the resulting box uncertainty set containing all possible realizations of the uncertainty.

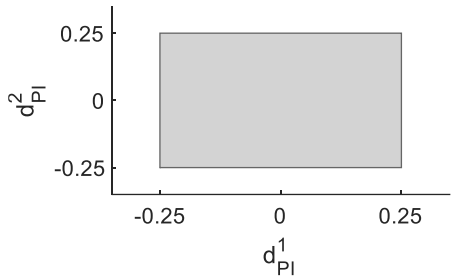

Figure 4.4: Uncertainty space

4.2 Openloop Simulation

For a preliminary study of the oil field dynamics the nominal model of the plant (i.e., with the error in the PI value of the two wells taken to be $d_{PI} = [0, 0]$) is simulated in openloop. The lift-gas supply w_{gc}, which is the (known) input disturbance to the system, is changed as a step from $33{,}000\ \mathrm{Sm^3/hr}$ to $36{,}000\ \mathrm{Sm^3/hr}$ at $time = 15\ \mathrm{hrs}$, and back to $33{,}000\ \mathrm{Sm^3/hr}$ at $time = 30\ \mathrm{hrs}$; and the total supply available at any time is distributed equally between the two wells. For solving the differential equations of the model, the RK4 integration scheme is used with a sampling time of $20\ \mathrm{sec}$. A simulation timespan of $45\ \mathrm{hrs}$ is considered.

The resulting dynamics of the oil field are shown in Figure 4.5. The principle of operation of the continuous-flow gas lift mechanism—detailed previously in section 4.1—can be clearly observed. When the supply of lift-gas to the annulus of each well is increased at $time = 15\ \mathrm{hrs}$ (first plot), the rate of injection into the tubing is, in turn, increased (second plot). More gas mixes with the produced oil causing further reduction of the density of the fluid columns (third plot). This means that the bottom hole pressure is also lowered further (fourth plot); the resulting increase in differential pressure causes more extraction of oil from the reservoir (last two plots).

It is also important to note that, since they are not identical, the two wells produce different amounts of oil when supplied with the same amount of lift-gas. According to equation (4.6), the amount of oil produced by each well corresponds to its PI value (note: this statement holds because the amount of oil produced by a well is known to be most sensitive to its PI value, as opposed to, e.g., its dimensions); since well-1 has the higher PI value it is observed to produce more oil in Figure 4.5.

At this point, it is readily seen that there may exist a possibility to achieve a higher production rate by allocating more of the available lift-gas to well-1 than to well-2. Figure 4.6 demonstrates that this indeed is a possibility: a higher production rate is observed when 55% of the available lift-gas is allocated to well-1 (the remaining 45% is allocated to well-2); conversely, a lower production rate is observed when more of the available lift-gas, 51%, is allocated to well-2. This establishes the fact that it is more economical to allocate more to well-1.

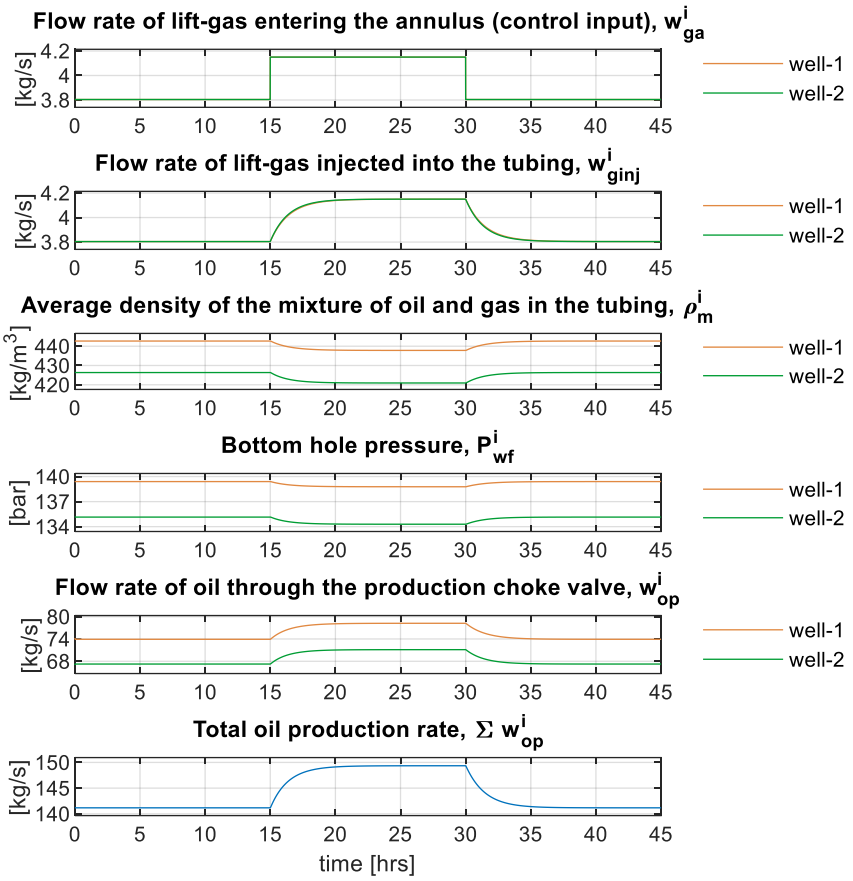

Figure 4.5: Openloop dynamics of the nominal oil field model under step changes in the input disturbance

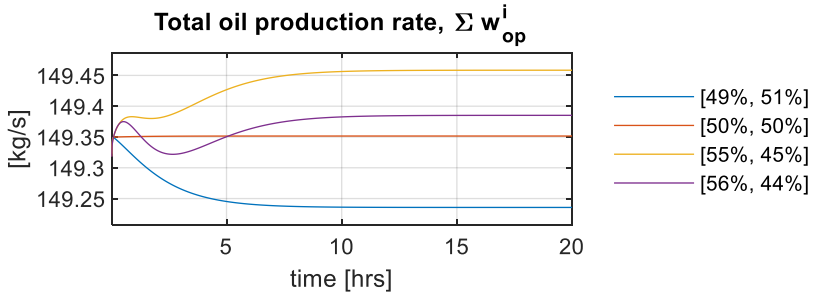

Figure 4.6: Total oil production corresponding to different allocations of the lift-gas supply $w_{gc} = 36{,}000\ \mathrm{Sm^3/hr}$

Naturally, there is a limit to which production can be maximized and this, too, is demonstrated in Figure 4.6: production corresponding to an allocation of 56% lift-gas to well-1 is lower in comparison to the case of 55%. That it is difficult to determine manually the point where the maximum lies is obvious.

The task of operating the oil field is made even more difficult by several other facts: the availability of lift-gas varies over time, and at any time there could be an excess supply of gas. The fluid production rate could easily exceed the processing capacity of the separator, i.e., the constraint in equation (4.21) could easily be violated, when there is an excess supply. So, in addition to working out the most

economical distribution of gas available at any time, care should also be taken to ensure that the process operates within the specified bounds. And finally, on top of all that, there is also the fact that uncertainty is present in the system.

It is glaringly obvious why manual operation of the plant is extremely challenging: production of oil should be optimized (i.e., maximum possible oil production should be achieved with the most economical usage of available lift-gas), under varying input disturbance conditions and under the presence of uncertainty, without violating any of the process constraints at any time. Hence, it is required to explore the possibility of employing NMPC for achieving optimal operation of the plant.

4.3 Standard NMPC

In this section the potential of employing standard NMPC for achieving optimal operation of the oil field is assessed.

4.3.1 Optimal Control Problem

The OCP for the oil field is formulated as given in equation (4.22) - (4.27).

$$\min_{m_1,\cdots,m_{N_p},w_{ga,0},\cdots,w_{ga,N_p-1}} \sum_{k=0}^{N_p-1} -w_{op,k+1}Qw_{op,k+1}^{\mathrm{T}} + w_{ga,k}Rw_{ga,k}^{\mathrm{T}} + \Delta w_{ga,k}S\Delta w_{ga,k}^{\mathrm{T}} \tag{4.22}$$

subject to

$$m_{k+1} = f\left(m_k, w_{ga,k}, d_{PI,k}\right), \qquad k = 0,\cdots,N_p-1 \tag{4.23}$$

$$\sum_{i=1}^{2} w_{ga,k}^{i} \le w_{gc,k}, \qquad k = 0,\cdots,N_p-1 \tag{4.24}$$

$$\sum_{i=1}^{2} w_{gop,k}^{i} \le 160, \qquad k = 0,\cdots,N_p-1 \tag{4.25}$$

$$-0.15 \le \Delta w_{ga,k}^{i} \le 0.15, \qquad k = 0,\cdots,N_p-1 \tag{4.26}$$

$$0.323 \le w_{ga,k}^{i} \le 11.66, \qquad k = 0,\cdots,N_p-1 \tag{4.27}$$

where N_p is the prediction horizon, and Q, R and S are the tuning parameters.

The main control objective is to maximize the total oil production of the oil field. Hence, $-w_{op}Qw_{op}^{T}$ is used as the cost function. Additionally, the penalty terms $w_{ga,k}Rw_{ga,k}^{T}$ and $\Delta w_{ga}S\Delta w_{ga}^{T}$ are introduced to limit excessive lift-gas utilization and to discourage large fluctuations in the control signals.

The constraints in (4.23) - (4.25) constitute the oil-field model equations presented in section 4.1.1. (4.23) represents the state trajectory, which, in addition to the model equations, involves also the integration scheme. The fact that the total fluid production rate should not exceed the processing capacity of the separator is enforced by the inequality constraint in (4.25). It is important to note that, since the fluid production rate of the two wells $w_{gop,k}^{i}$ is calculated using the process model, this

constraint is susceptible to model uncertainty. Hence, of all the constraints, it is the main one to look out for.

The last two constraints in (4.26) and (4.27) enforce assumed physical limitations of the gas-lift choke valves: (4.26) represents the fact that the valves cannot be opened/closed at a rate higher than a certain maximum, and (4.27) represent the fact that there are limits on the extent to which the valves can be opened/closed.

4.3.2 Simulation Results

In all the simulation runs discussed in the remainder of this chapter, the available lift-gas supply is taken to be $w_{gc} = 40{,}000\ \mathrm{Sm^3/hr}$. The process is first run in openloop until steady state is reached. During this period, only 80% of the lift-gas supply is utilized and this amount is distributed equally between the two wells.

With said settings, the process operates within the specified bounds, but there is a lot of room for improvement: the total fluid production is much lower than the allowed maximum, which also means that the total oil production is lower than the possible maximum. The profit can be increased substantially by bringing the plant to operate closer to—ideally, at—its production capacity.

Each of the different NMPC schemes explored in this work is tasked with figuring out how this may be done: by utilizing the remaining 20% of the lift-gas supply and/or by distributing the available gas in a more optimal manner between the two wells (as discussed in section 4.2, equal distribution is most likely not the optimal approach). Additionally, a step change in the gas supply is also introduced towards the end of each simulation run to further asses the effectiveness of each NMPC scheme. (Note: The point at which NMPC is switched on, after openloop operation, will be marked with a red dotted vertical line in all the plots.)

In the rest of this section, how successfully standard NMPC handles the said task when no uncertainty is present and when uncertainty *is* present is studied.

The tuning parameters Q, R and S in equation (4.22) are chosen to be I_2, $0.5I_2$ and $50I_2$, respectively. A sampling time of $20\ \mathrm{sec}$ and a prediction horizon of $N_p = 25\ \mathrm{timesteps}$ ($\sim 8.3\ \mathrm{min}$) is used. These values are maintained throughout the case study.

4.3.2.1 Performance When No Uncertainty is Present

Figure 4.7 shows the performance of standard NMPC when there is no uncertainty. When switched on, the controller immediately starts tapping into the remaining 20% of the lift-gas supply in order to increase the oil production rate as soon as possible. When plant operation nears the production capacity, the controller starts adjusting the control inputs such that the maximum allowed fluid production rate is reached without any overshoot. The attained rate is maintained thereafter. Since the model predictions are 100% accurate, no constraint violations occur.

It is important to note the influence of the penalty terms on performance. Action of the term $w_{ga,k} R w_{ga,k}^T$ in the objective function—which is used to enforce a more economical route—is evident

throughout the simulation run. It is more readily observable after the maximum production limit is reached: in order to maintain the production rate, only about 92% of the available lift-gas supply is utilized and more of this amount is allocated to well-1 than to well-2. Likewise, the term $\Delta w_{ga} S \Delta w_{ga}^T$ can be credited, in no small part, for the smooth variation of the control inputs observed throughout.

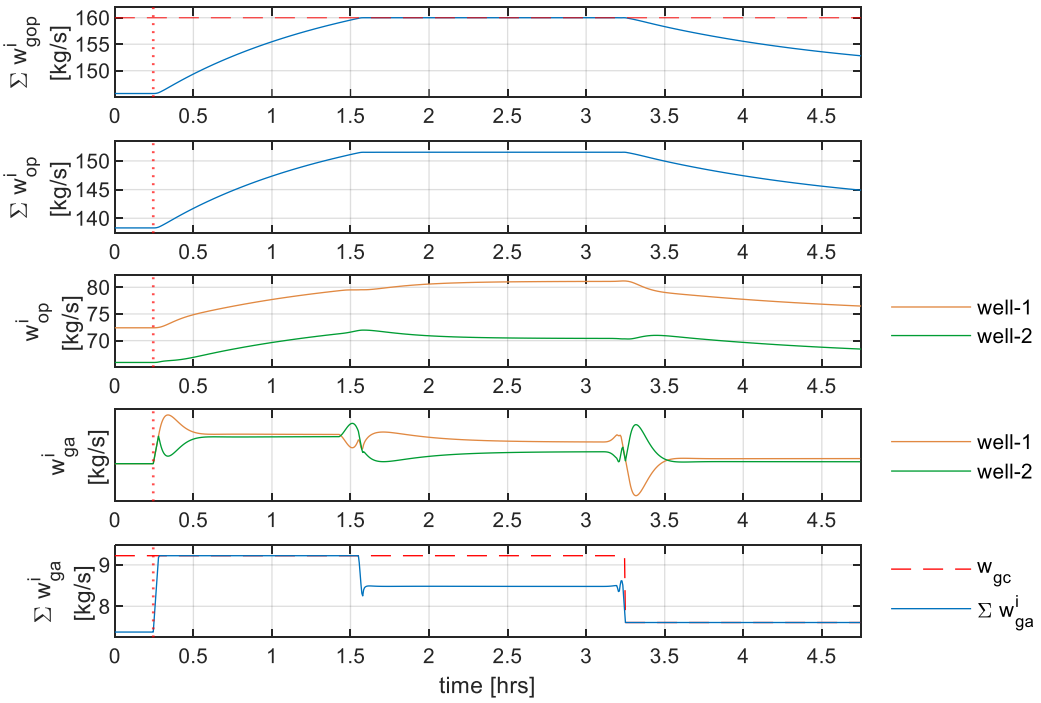

Figure 4.7: Performance of standard NMPC when no uncertainty is present, plant d_{PI} = controller d_{PI} = [0, 0]

The smooth response to the step change in the lift-gas supply w_{gc} (i.e., the known input disturbance), which is introduced at about the 3.25-hour mark, further demonstrates the effectiveness of standard NMPC.

In essence, standard NMPC effortlessly achieves optimal operation *when no uncertainty is present.*

4.3.2.2 Performance When Uncertainty Is Present

When plant-model mismatch exists, since the model predictions are not accurate, constraint violations may occur. To demonstrate this, the same case considered in section 4.3.2.1 is simulated, but now with plant $d_{PI} = [0.13, 0.13]$; the nominal values $d_{PI} = [0, 0]$ are retained in the model. The results are shown in Figure 4.8.

It can be seen that, in this case, the controller does not start to lower the control inputs until *some time after* the production limit is exceeded; for the most part, the controller doesn't realize that the plant is operating outside the specified bounds and so no action is taken to bring it back within limits (the exceptions being the sudden dips in the control inputs, observed after the 1.5-hour mark. However, these too are likely not actions resulting from *actual* knowledge that the plant is operating outside the specified boundaries). This occurs due to state feedback: the process model, which is now

inaccurate, is used to *compute* the current fluid production rate from the state information fed back from the plant; the nominal PI values used in the computation $PI = [2.51, 1.63] \times 10^4$ kg/hr/bar are lower than the actual PI values of the plant $PI = [2.64, 1.76] \times 10^4$ kg/hr/bar so consequently, the computed fluid production rate is also lower than the actual. The same holds for the prediction of the state trajectory, which is why the production rate exceeds the allowed maximum in the first place.

Since such constraint violations could be fatal in reality, performance of standard NMPC is not acceptable in this case. Instead, robust NMPC techniques that can account for the uncertainty present in the PI values, need to be explored.

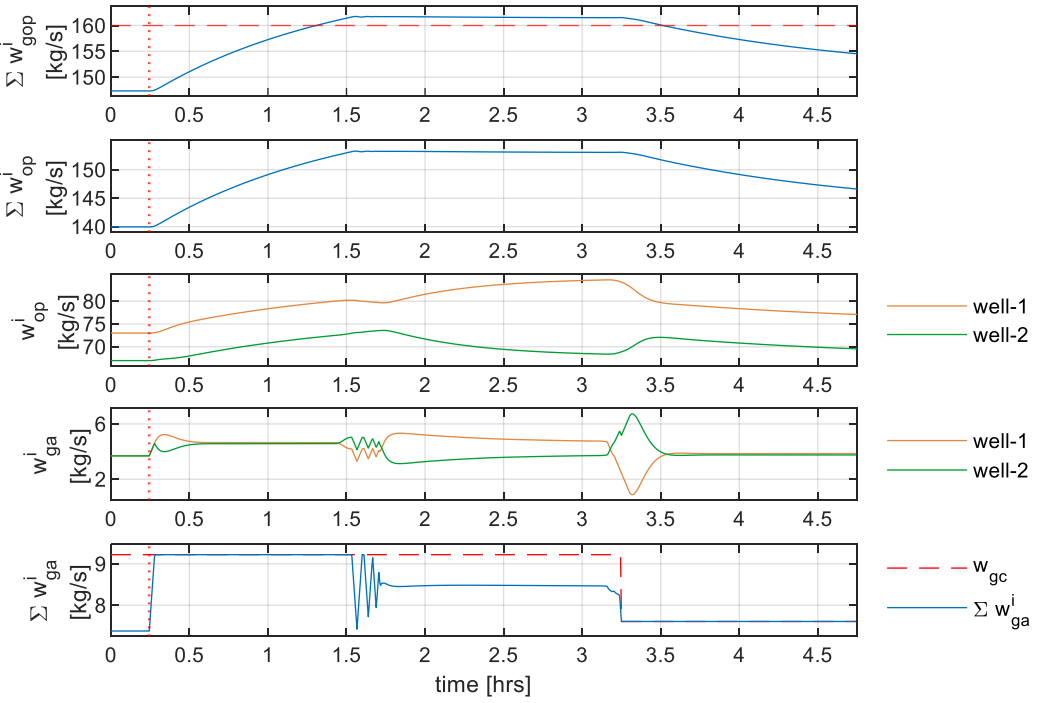

Figure 4.8: Performance of standard NMPC when uncertainty is present, plant $d_{PI} = [0.13, 0.13]$, controller $d_{PI} = [0, 0]$

4.4 Multi-Stage NMPC

In this section the potential of employing multi-stage NMPC for achieving optimal operation of the oil field, under the presence of uncertainty, is assessed.

4.4.1 Optimal Control Problem

The optimization problem resulting from the multi-stage approach is given in equation (4.28) - (A.12).

$$\min_{m^j_k, w^j_{ga,k} \forall (j,k) \in I} \sum_{i=1}^{N} \omega_i \sum_{k=0}^{N_p-1} -w_{op,k+1} Q w^{\mathrm{T}}_{op,k+1} + w_{ga,k} R w^{\mathrm{T}}_{ga,k} + \Delta w_{ga,k} S \Delta w^{\mathrm{T}}_{ga,k} \tag{4.28}$$

subject to

$$\mathrm{m}^j_{k+1} = f\left(m^{p(j)}_k, w^j_{ga,k}, d^{r(j)}_{PI,k}\right), \quad \forall\, (j, k+1) \in I \tag{4.29}$$

$$\sum_{i=1}^{2} w_{ga,k}^{i,j} \leq w_{gc,k}, \qquad \forall\, (j,k) \in I \tag{4.30}$$

$$\sum_{i=1}^{2} w_{gop,k}^{i,j} \leq 160, \qquad \forall\, (j,k) \in I \tag{4.31}$$

$$-0.15 \leq \Delta w_{ga,k}^{i,j} \leq 0.15, \qquad \forall\, (j,k) \in I \tag{4.32}$$

$$0.323 \leq w_{ga,k}^{i,j} \leq 11.66, \qquad \forall\, (j,k) \in I \tag{4.33}$$

$$w_{ga,k}^{i,j} = w_{ga,k}^{i,l} \text{ if } m_k^{p(j)} = m_k^{p(l)} \qquad \forall\, (j,k),(l,k) \in I \tag{4.34}$$

The stage cost and the constraints are the same as considered in section 4.3.1, for standard NMPC, except for the non-anticipativity constraint given in equation (A.12).

The weights of the scenarios ω_j are all chosen to be identical.

4.4.2 Simulation Results

Performance of multi-stage NMPC corresponding to several different cases is studied here:

1. realizations of the uncertainty included in the scenario tree,
2. realizations of the uncertainty not included in the scenario tree,
3. different scenario tree designs (i.e., different sets of values of the uncertainty considered in the tree),
4. different robust horizon lengths, and
5. different ranges of the uncertainty.

Furthermore, several ways of improving the performance, with respect to computational cost and conservativeness, are also explored:

6. direct multiple-shooting for reducing computational cost
7. grouping of control inputs for reducing computational cost
8. estimation of the uncertainties for reducing conservativeness.

4.4.2.1 Performance for Realizations of the Uncertainty Included in the Scenario Tree

The scenario tree considered in this section is generated by combining the minimum, maximum and nominal values of the uncertainties. These are marked with '×' in Figure 4.9, which provides a simple illustration of their relative positions in the uncertainty space. Since the PI values are assumed constant throughout the simulation timespan, the robust horizon is taken to be $N_r = 1$, i.e., the branching of the scenario tree is done only once. These choices lead to $N = 5$ discrete scenarios.

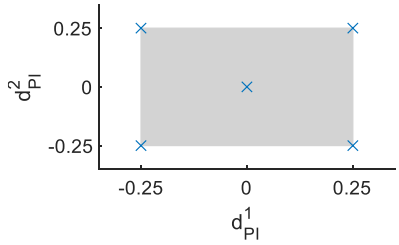

Figure 4.9: Uncertainty space and the values of the uncertainty considered in the scenario tree.

Figure 4.10 shows the performance of multi-stage NMPC corresponding to different realizations of uncertainty that are explicitly represented in the scenario tree. No constraint violations are observed for any of the scenarios. It is seen, however, that this comes at the price of conservativeness, i.e., the obtained solution is suboptimal in comparison to the solution provided by standard NMPC using a perfect model of the system. Nevertheless, the simulation results prove that constraint satisfaction is indeed guaranteed for values of the uncertainty that are present in the scenario tree.

Another important observation is that the degree of conservativeness is different for different realizations of the uncertainty. The highest is observed for $d_{PI} = [-0.25, -0.25]$. It is of interest to further investigate the conservativeness in the solution for this case; the results are compared with the corresponding results for the ideal case using standard NMPC, i.e., when no uncertainty is present, in Figure 4.11. It can be seen that optimality is severely affected by the need to account for the presence of uncertainty: in the ideal case, standard NMPC can make the plant produce oil at a rate of $\sum w_{op}^{i} \sim 151\ \mathrm{kg/s}$; but when uncertainty *is* present and accounted for using multi-stage NMPC, production drops down to $\sum w_{op}^{i} \sim 143\ \mathrm{kg/s}$. Hence, it can be concluded that, even though multi-stage NMPC is less conservative than other robust NMPC approaches, conservativeness could still be a concern depending on the actual value of the uncertainty.

4.4.2.2 Performance for Realizations of the Uncertainty Not Included in the Scenario Tree

In Figure 4.10, it is seen that for one of the realizations $d_{PI} = [0.25, 0.25]$ the plant is operated at the boundary. This indicates that, *of the five scenarios considered in the scenario tree*, the one corresponding to this value of the uncertainty is the worst-case scenario (the one corresponding to $d_{PI} = [-0.25, -0.25]$ is the best; the highest level of conservativeness is observed for it). However, it is not known whether it is the worst-case scenario out of *all* the possibilities. So, even though constraint satisfaction is guaranteed for values of the uncertainty explicitly included in the scenario tree, nothing can be said for the values that are not.

It is required to ensure constraint satisfaction also for values that are not explicitly included in the scenario tree. To do this, the same scenario tree—described in section 4.4.2.1—is used and multi-stage NMPC is tested for 32 different realizations that are evenly distributed in the uncertainty space. Figure 4.12 shows the corresponding results, together with the results for realizations that are included in the scenario tree.

It is seen that all the constraints are satisfied also for realizations that are not explicitly included in the scenario tree. Furthermore, the plant is not operated at the boundary for any of these—only for $d_{PI} = [0.25, 0.25]$, which *is* explicitly included. This *suggests* that the scenario corresponding to $d_{PI} = [0.25, 0.25]$ is indeed the worst possible scenario (the scenario corresponding to $d_{PI} = [-0.25, -0.25]$ also *appears* to be the best possible scenario; the highest level of conservativeness is still observed for it), which, in turn, *suggests* that robust constraint satisfaction is ensured for *all* possible values of the uncertainty. However, it is important to note that these simulation results do not provide conclusive evidence for this; only a small subset of the uncertainty space has been considered here.

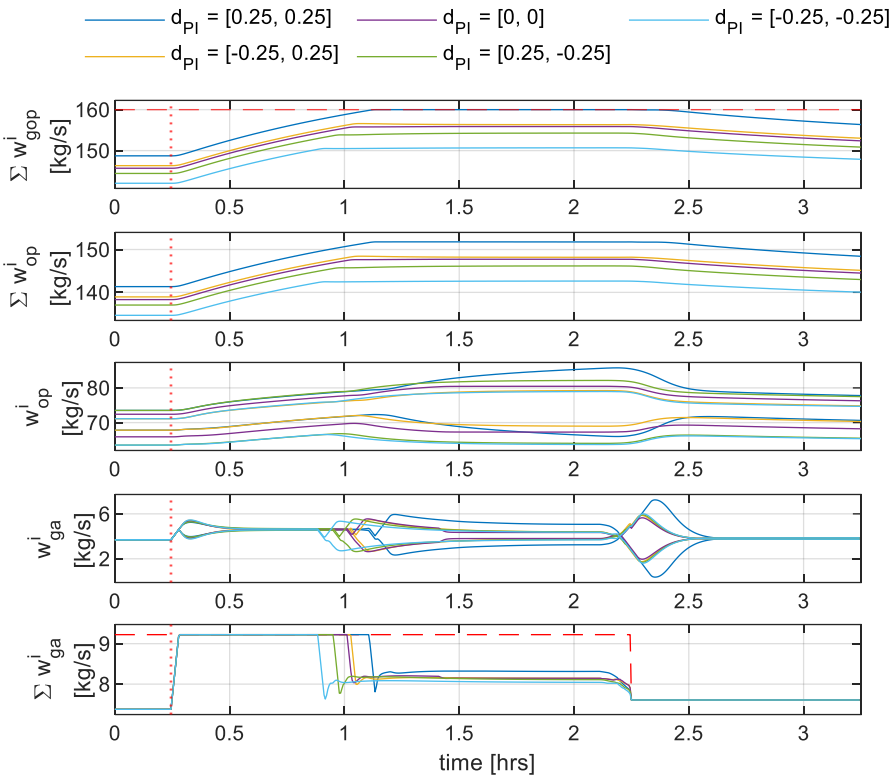

Figure 4.10: Performance of multi-stage NMPC for each realization of the uncertainty present in the scenario tree

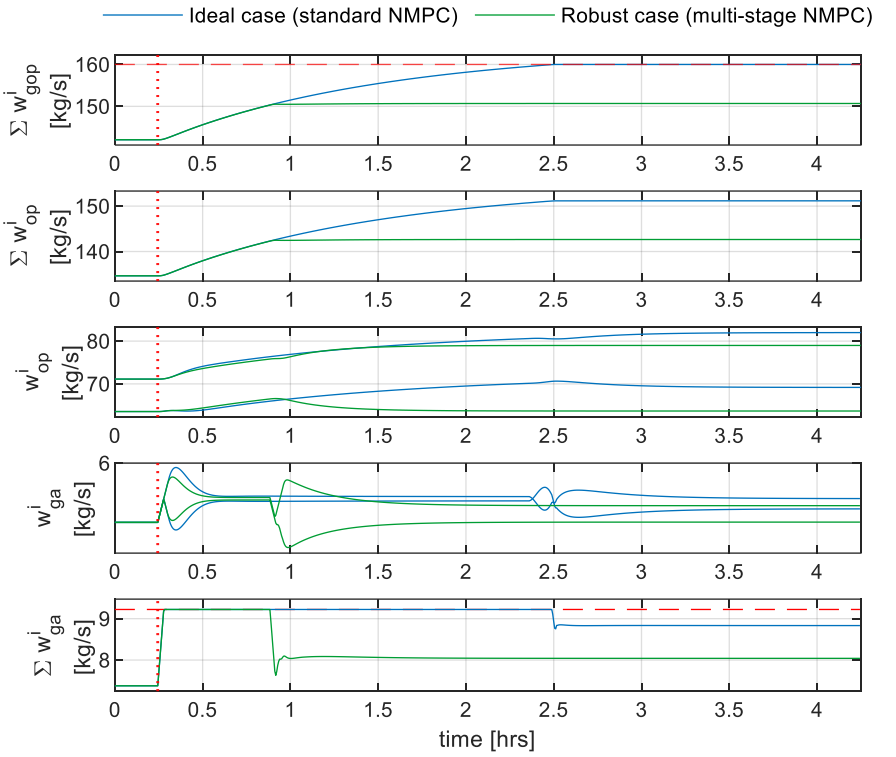

Figure 4.11: Performance of standard NMPC when no uncertainty is present, and performance when uncertainty is present and accounted for using multi-stage NMPC. Plant $d_{PI} = [-0.25, -0.25]$.

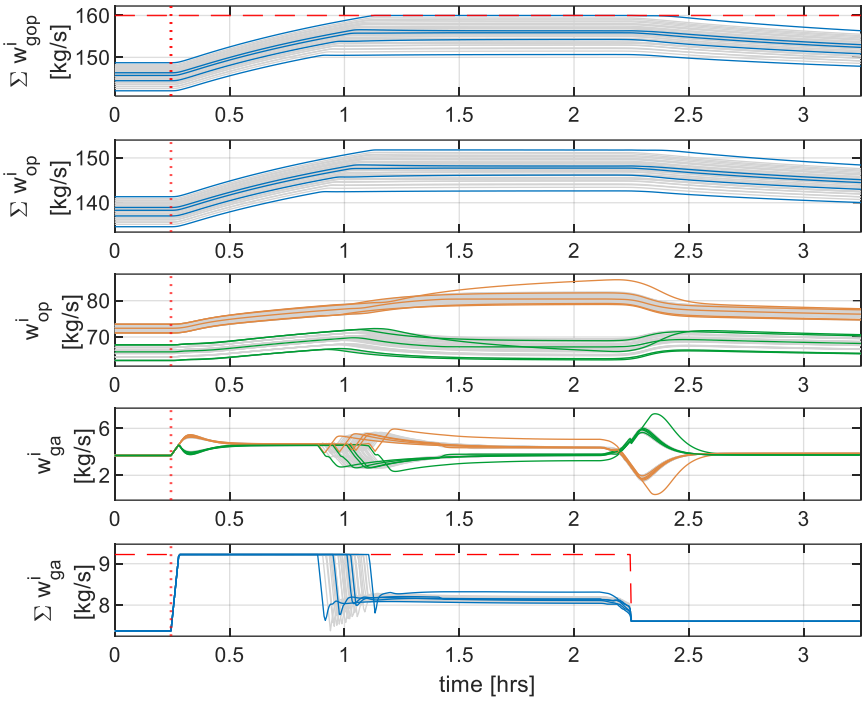

Figure 4.12: Performance of multi-stage NMPC for different realizations of the uncertainty. Grey lines correspond to realizations that are not present in the scenario tree, and coloured lines correspond to the ones that are.

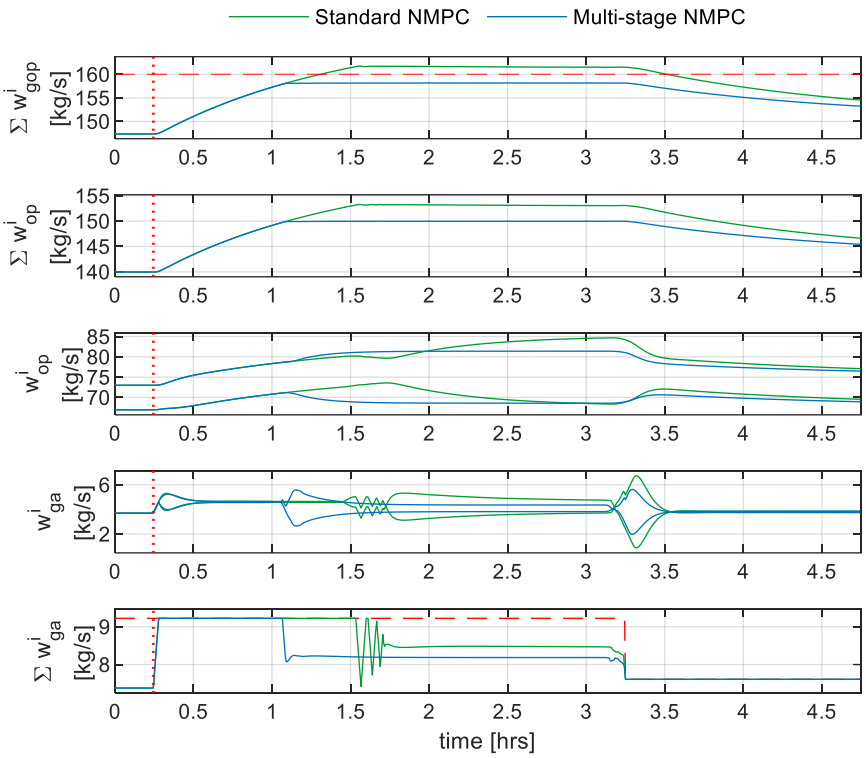

Figure 4.13: Performance of standard NMPC (controller $d_{PI} = [0, 0]$) and multi-stage NMPC corresponding to the same realization of the uncertainty $d_{PI} = [0.13, 0.13]$

Figure 4.13 provides a comparison of the performance of standard NMPC (controller $d_{PI} = [0, 0]$) and multi-stage NMPC corresponding to the same realization of the uncertainty $d_{PI} = [0.13, 0.13]$— which is the same value considered in section 4.3.2.2. Multi-stage NMPC successfully handles the uncertainty, albeit with a small level of conservativeness, whereas standard NMPC completely fails.

4.4.2.3 Influence of the Scenario Tree Design on Performance

To study the effect of scenario tree design on performance, four different trees are considered. Schematic representations of these are shown in Figure 4.14. The resulting performance of multi-stage NMPC with each of the trees, corresponding to the same realization $d_{PI} = [0.20, 0.20]$, are compared in Figure 4.15.

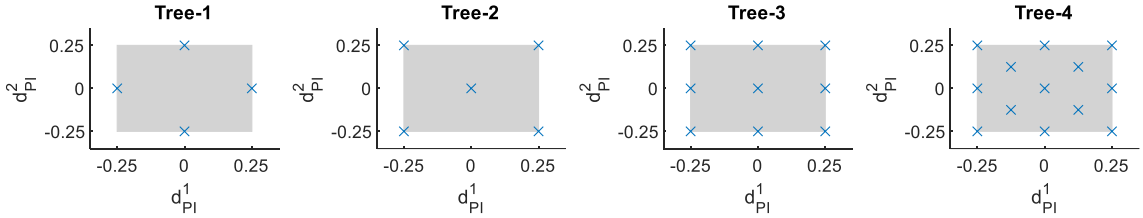

Figure 4.14: Several scenario trees with different values of the uncertainty taken into account and $N_r = 1$

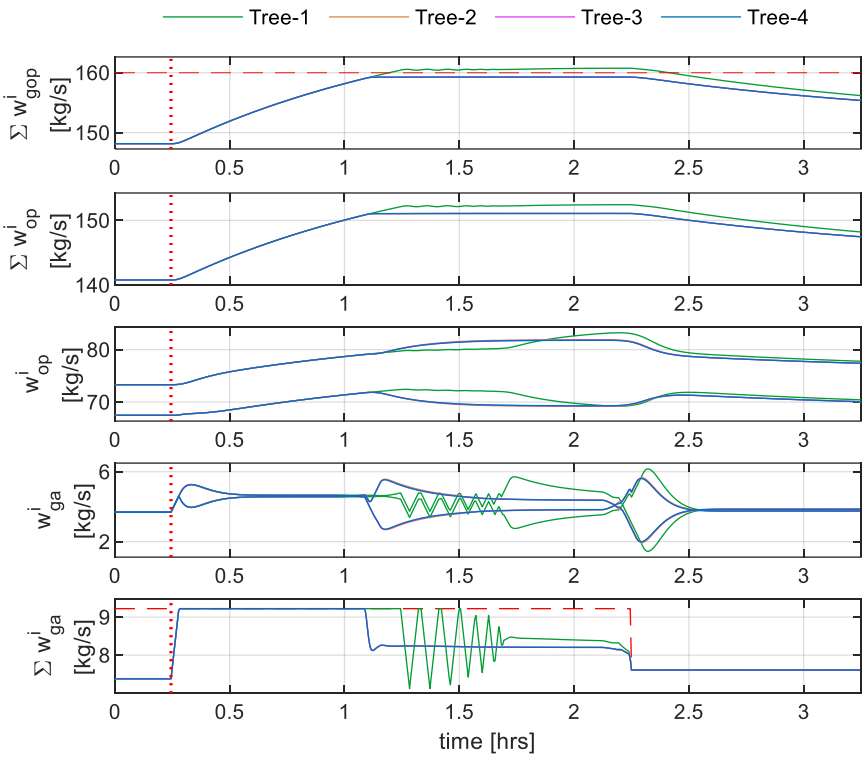

Figure 4.15: Performance of multi-stage NMPC with different scenario tree designs. Plant $d_{PI} = [0.20, 0.20]$.

In section 4.4.2.2 the worst-case scenario was established to be $d_{PI} = [0.25, 0.25]$. However, Tree-1 does not take this into consideration. Hence, multi-stage NMPC with Tree-1 could result in constraint

violations as demonstrated in Figure 4.15. This further emphasizes the importance of including the combinations of the extreme values of the uncertainties in the scenario tree.

As for the other three trees—all of which include the extreme values—the performance appears to be identical; inclusion of intermediate values does not improve performance. Tree-2 provides good performance at a lower computational cost, so it will be used in the remainder of the case study.

4.4.2.4 Influence of the Robust Horizon Length on Performance

Even though the PI is a constant—but uncertain—parameter, performance of multi-stage NMPC with a robust horizon $N_r = 1$ need not necessarily be the best: branching of the scenario tree does not solely represent (possible) time-variance of the uncertainties; it also represents the fact that a new tree is considered at the next timestep [12]. Taking the robust horizon as $N_r = 1$ is equivalent to making the *inaccurate* assumption that different control inputs can be taken at the next timestep, depending on the uncertainty, when in truth the new tree at the next timestep will enforce that all control inputs in the first stage be the *same* [10].

To study the effect of robust horizon on performance, three different robust horizons are tested for the same realization of the uncertainty $d_{PI} = [0.13, 0.13]$. Figure 4.16 provides a comparison of the results. Performance for all three cases seems identical. However, upon close inspection, it is seen that increasing the robust horizon leads to higher conservativeness. Given this, and the fact that it also leads to higher computational cost, it is easy to conclude that $N_r = 1$ is the best choice in this case; it will be used in the remainder of the case study.

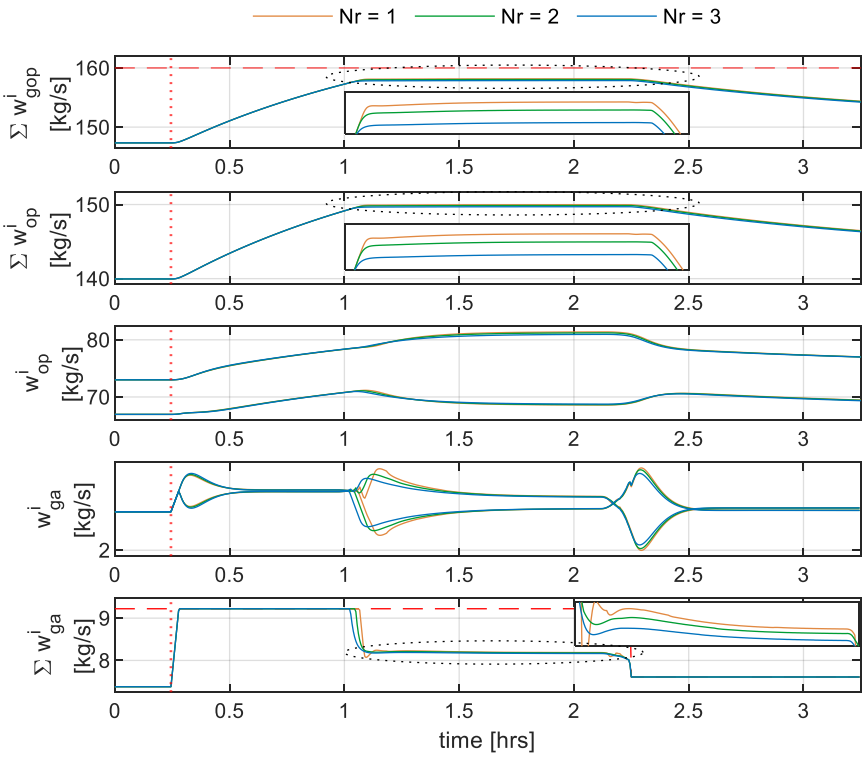

Figure 4.16: Performance of multi-stage NMPC with different robust horizons N_r. Plant $d_{PI} = [0.13, 0.13]$.

4.4.2.5 Influence of the Range of Uncertainty on Performance

In section 4.1.2 the uncertainty ranges were established to be $d_{PI}^{i} \in [-0.25, 0.25]$, which equates to an uncertainty of ~±10% in the PI value of well-1 and an uncertainty of ~±15% in the PI value of well-2. But here, a case where $d_{PI}^{i} \in [-0.375, 0.375]$ (i.e., a 150% increase in range) is also considered; the corresponding uncertainties in the PI value of well-1 and well-2 are ~±15% and ~±23%, respectively.

A comparison of the performance for each case is shown in Figure 4.17. Naturally, a higher degree of conservativeness is observed for the case with the higher uncertainty; for the considered realization $d_{PI} = [0.13, 0.13]$ the total oil production is seen to be lower by about 1% in comparison.

In section 4.4.2.1 it was remarked that multi-stage NMPC produces different degrees of conservativeness for different realizations of the uncertainty, and for some the degree of conservativeness is quite significant and hence could be a problem. The observation made here adds onto this fact: even though multi-stage NMPC is less conservative compared to other robust NMPC approaches, conservativeness could still be a problem depending on the actual value of the uncertainty *and* the range of the uncertainty.

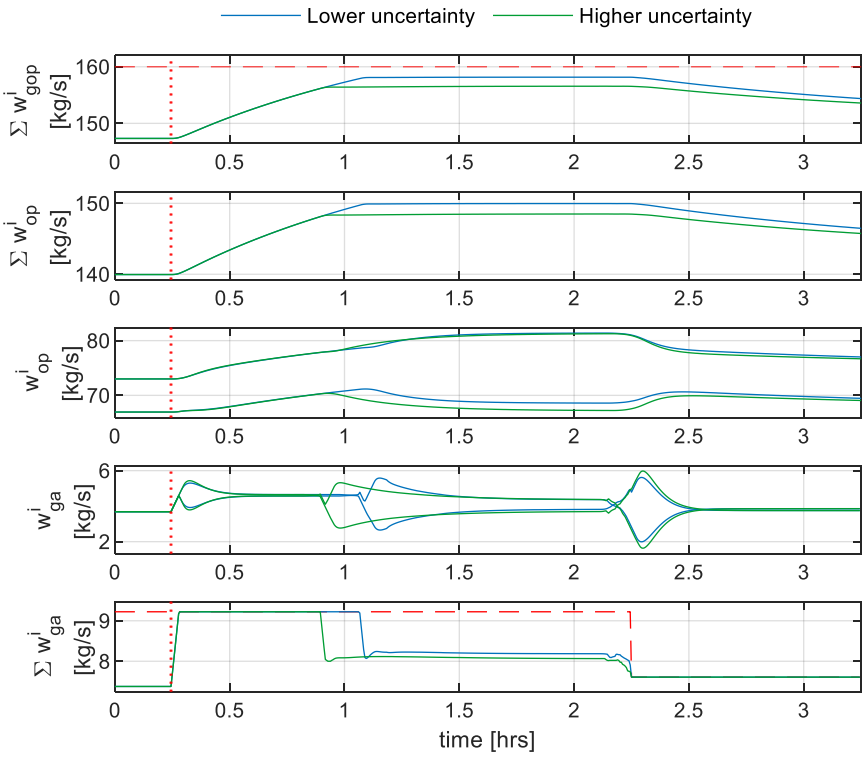

Figure 4.17: Performance of multi-stage NMPC for different ranges of the uncertainty and for plant $d_{PI} = [0.13, 0.13]$. (Note: The scenario tree for both cases is generated using the minimum, maximum and nominal values of the corresponding uncertainty range, and $N_r = 1$.)

4.4.2.6 Direct Multiple-Shooting for Reducing Computational Cost

The advantages of using the direct multiple-shooting strategy over the direct single-shooting strategy were discussed in detail in section 3.2.1.1, and up until this point it has been used throughout the case study. But here, for comparison purposes, multi-stage NMPC is implemented with direct single-shooting. Computation times for simulating the same scenario with each shooting strategy are compared in Table 4.2.

Table 4.2: Comparison of the average computation time per iteration of multi-stage NMPC with single-shooting and multiple-shooting

	Robust horizon N_r	**No. of decision variables**	**Average time taken to solve the NLP at each time step**	**Ratio between the average times $t_{multi-sh}$: $t_{single-sh}$**
Direct multiple-shooting	1	1022	0.2539	1:2
Direct single-shooting		242	0.5095	
Direct multiple-shooting	2	5062	0.8467	1:3
Direct single-shooting		1162	2.4086	

It is evident that, even though it results in a much larger NLP, the multiple-shooting strategy facilitates faster convergence to the optimal solution. The improvement in computation time makes real-time implementation possible even for a robust horizon $N_r = 2$.

4.4.2.7 Grouping of Control Inputs for Reducing Computational Cost

As discussed in section 3.2.1.3, grouping of control inputs is another method that can be used for cutting down the computational cost. For demonstrating its effectiveness with multi-stage NMPC, one case of grouping is presented here. The original case, without grouping, is compared with the considered case of grouping in Table 4.3. (Note: With multi-stage NMPC, only the control inputs corresponding to the stages occurring after the robust horizon, i.e., $w_{ga,k} = \left[w^1_{ga,k}, w^2_{ga,k}\right]$ for $k = 1, \cdots, 24$ in equation (4.28), are grouped.)

Table 4.3: Comparison of the number of decision variables in the original (non-grouped) OCP and the grouped OCP

	No. of groups	**No. of control inputs $w_{ga} = \left[w^1_{ga}, w^2_{ga}\right]$ per group (listed in the order they appear along the prediction horizon)**	**No. of decision variables corresponding to the control inputs**
Without grouping	24	[1, …, 1]	242
With grouping	8	[2, 2, 2, 2, 2, 4, 4, 6]	82

The resulting performance of multi-stage NMPC with control input grouping is compared with the original performance (i.e., without control input grouping) in Figure 4.18; the average time taken to solve the NLP at each time step in each case is provided in the legend. It can be seen that, although control input grouping introduces an extra assumption to the OCP, with a proper choice of grouping

scheme, more or less the same original performance can be achieved at an even lower computational cost than achieved with just direct multiple-shooting.

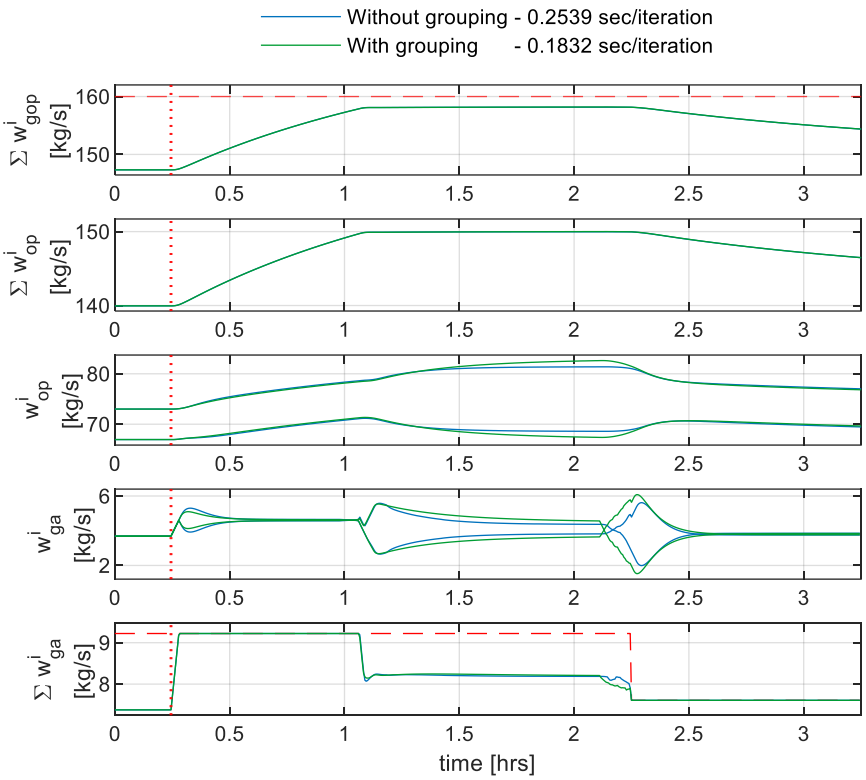

Figure 4.18: Performance of multi-stage NMPC with and without control input grouping. Plant $d_{PI} = [0.13, 0.13]$.

4.4.2.8 Estimation of the Uncertainties for Reducing Conservativeness

It was seen in section 4.4.2.1 and 4.4.2.5 that the conservativeness of multi-stage NMPC could still be a potential concern, even though it is known to be less conservative than other robust control approaches. Since the PI parameter is time-invariant, the performance could be improved by estimating the uncertainty online, as discussed in section 3.2.2. Here the estimation technique proposed in [19] is tested.

The estimation algorithm is implemented together with the multi-stage NMPC algorithm to update the scenario tree online; the results are compared to the case without online estimation in Figure 4.19. The parameters K, δ, and α in [19] are chosen to be 2, 10^{-10}, and 0.3, respectively and the estimation algorithm is turned on near the 1.25-hour mark (marked with a red dotted vertical line in all the plots in Figure 4.19).

After estimation is started, a gradual reduction in conservativeness can be seen; more and more of the available lift-gas is utilized to increase the total oil production. On the flip side, sudden fluctuations in the control signals result from updating of the scenario tree. However, this may be inconsequential in comparison to the production increase.

All in all, the simulation results suggest that significant improvements in performance could be achieved by employing estimation techniques together with multi-stage NMPC whenever possible.

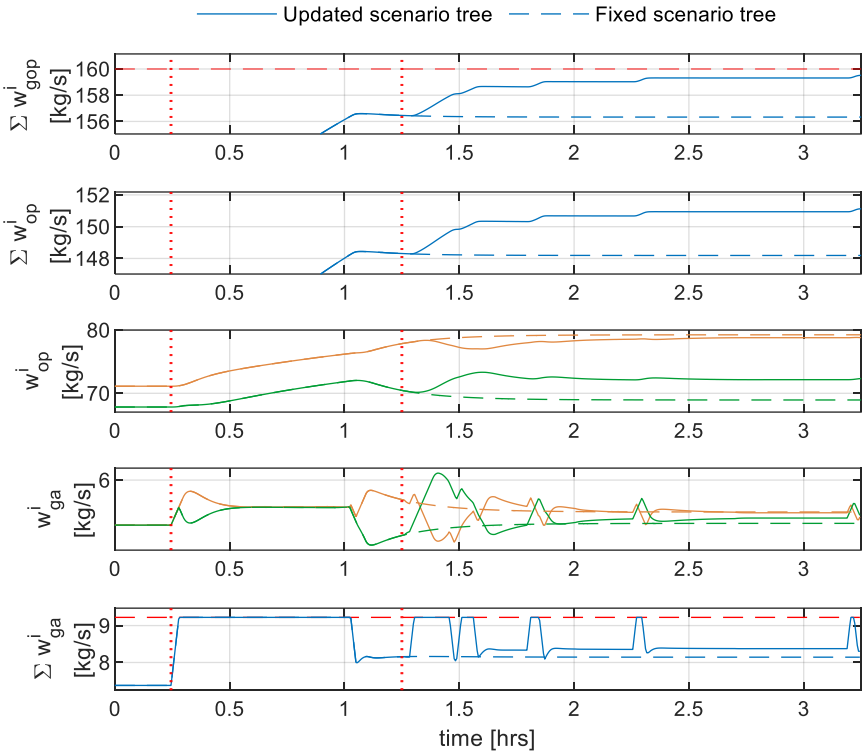

Figure 4.19: Performance of multi-stage NMPC with online update of scenario tree and with fixed scenario tree. Plant $d_{PI} = [-0.25, 0.25]$.

4.5 Min-max NMPC

In this section how the closed-loop min-max NMPC variation of multi-stage NMPC stacks up against the original formulation is evaluated.

4.5.1 Optimal Control Problem

The closed-loop min-max NMPC formulation is obtained by simply replacing the summation in (4.28) with the maximization operator:

$$\min_{m_k^j, w_{ga,k}^j \forall (j,k) \in I} \max \omega_i \sum_{k=0}^{N_p-1} -w_{op,k+1} Q w_{op,k+1}^{\mathrm{T}} + w_{ga,k} R w_{ga,k}^{\mathrm{T}} + \Delta w_{ga,k} S \Delta w_{ga,k}^{\mathrm{T}} \tag{4.35}$$

subject to

$$\mathrm{m}_{k+1}^j = f\left(m_k^{p(j)}, w_{ga,k}^j, d_{PI,k}^{r(j)}\right), \quad \forall\, (j, k+1) \in I \tag{4.36}$$

$$\sum_{i=1}^{2} w_{ga,k}^{i,j} \leq w_{gc,k}, \quad \forall\, (j,k) \in I \tag{4.37}$$

$$\sum_{i=1}^{2} w_{gop,k}^{i,j} \leq 160, \qquad \forall\, (j,k) \in I \tag{4.38}$$

$$-0.15 \leq \Delta w_{ga,k}^{i,j} \leq 0.15, \qquad \forall\, (j,k) \in I \tag{4.39}$$

$$0.323 \leq w_{ga,k}^{i,j} \leq 11.66, \qquad \forall\, (j,k) \in I \tag{4.40}$$

$$w_{ga,k}^{i,j} = w_{ga,k}^{i,l} \text{ if } m_k^{p(j)} = m_k^{p(l)} \qquad \forall\, (j,k),(l,k) \in I \tag{4.41}$$

4.5.2 Simulation Results

Performance of min-max NMPC and multi-stage NMPC, with the same scenario tree—generated using the combinations of the extreme values and the nominal values of the uncertainties and $N_r = 1$—corresponding to each of the two extreme realizations of the uncertainty, i.e., $d_{PI} = [0.25, 0.25]$ and $d_{PI} = [-0.25, -0.25]$, is compared. The results are shown in Figure 4.20 and Figure 4.21; the average time taken to solve the NLP at each time step in each case is provided in the legends.

As expected, min-max NMPC, too, ensures robust constraint satisfaction for all possible values of the uncertainty. While performance appears to be more or less the same in terms of conservativeness, the benefit of multi-stage NMPC can be seen in the form of reduced computational cost: multi-stage NMPC is approximately 4x faster.

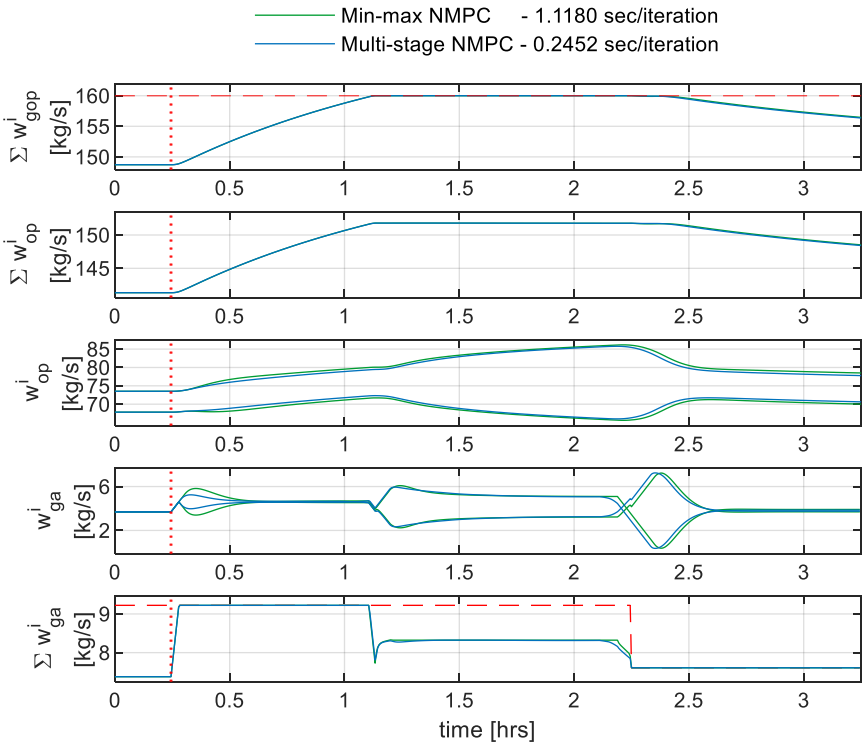

Figure 4.20: Performance of min-max and multi-stage NMPC for plant $d_{PI} = [0.25, 0.25]$

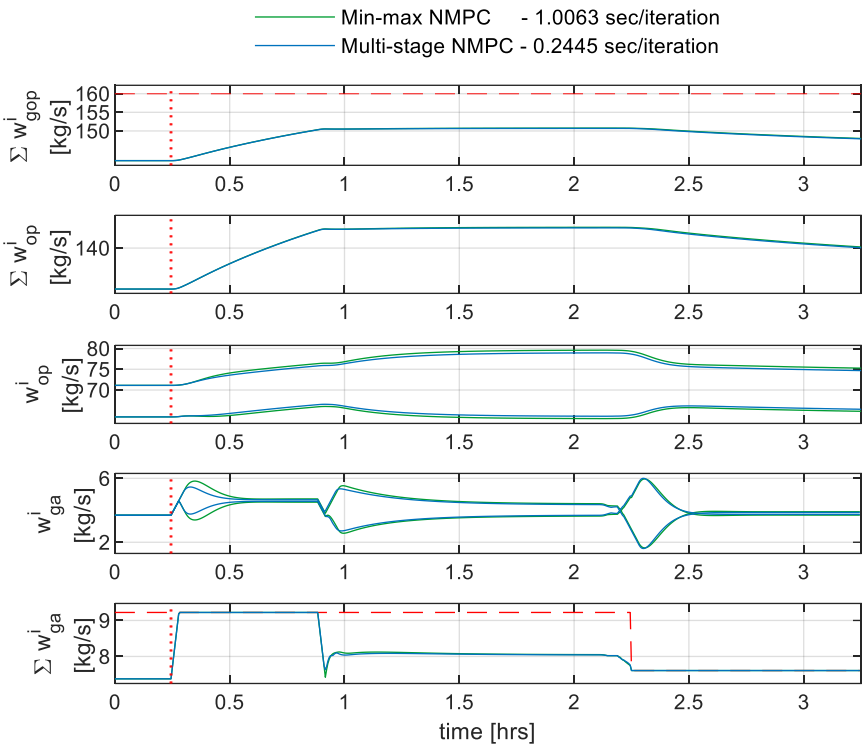

Figure 4.21: Performance of min-max and multi-stage NMPC for plant $d_{PI} = [-0.25, -0.25]$

4.6 Discussion

This chapter constituted the main results of this thesis; the implementation of two robust NMPC techniques—multi-stage and min-max—to an industrial case study was detailed and the resulting performance was evaluated.

Firstly, the ideal operation of the plant and the need for addressing the uncertainties was demonstrated through the implementation of standard NMPC: it was seen to work perfectly when knowledge about the plant was considered exact (i.e., no uncertainty), but constraint violations were observed when uncertainty was introduced.

Then multi-stage NMPC was implemented, and its performance was assessed. Simulation results proved that it leads to better performance than standard NMPC; it is capable of successfully handling the uncertainties and ensuring robust constraint satisfaction. More specifically, it was affirmed that inclusion of the combinations of the extreme values of the uncertainties in the scenario tree ensures robust constraint satisfaction for all possible realizations of the uncertainty—conversely, exclusion results in constraint violations (for some values of the uncertainty). Inclusion of intermediate values and extension of the robust horizon length was found to have little effect on the performance, and hence it was established that the scenario tree generated using the combinations of the extreme and nominal values, and a robust horizon of one timestep was the best choice—simulation results showed that it provides good performance at lower computational cost.

Despite the fact that multi-stage NMPC is considered a non-conservative robust control scheme it was clear that the solution will still have some degree of conservativeness in order to ensure robust constraint satisfaction. Comparison of the performance for the best-case scenario and the corresponding ideal performance—that could be achieved with standard NMPC—made it clear that conservativeness could be a potential problem, especially if the span of uncertainty is high—because the higher the uncertainty, the greater the degree of conservativeness.

This chapter also provided some insight into efficient implementation of robust NMPC. It was shown that the direct multiple-shooting approach and grouping of control inputs provides significant improvements in computational speed over simple implementations using the direct single-shooting approach. Efficient implementation becomes more important when problems with longer prediction horizons and more uncertainties are considered.

A simple online estimation technique was implemented together with multi-stage NMPC, to narrow the span of the uncertainties, and in turn the span of the scenario tree. The results suggest that, for problems involving time-invariant uncertain parameters, the performance could be improved through the use of such extensions.

Lastly, multi-stage NMPC was compared with its closed-loop min-max variation. Even though performance of both was seen to be more or less identical in terms of robust constraint satisfaction and conservativeness, results suggest that multi-stage should be preferred over min-max since comparable performance could be achieved at a substantially lower computational cost.

5 Conclusion

5.1 Conclusion

The main goal of this thesis was to investigate and implement robust NMPC, specifically multi-stage NMPC and min-max NMPC. To this end, a literature review on existing robust NMPC techniques was carried out and multi-stage and min-max NMPC methods were studied in-depth. Additionally, a brief review of methods available for dealing with the issues of computational complexity and conservativeness of the robust NMPC techniques was also conducted. Based on the acquired knowledge, robust NMPC was implemented on a challenging industrial case study.

Simulation results showed both multi-stage NMPC and min-max NMPC to be capable of ensuring robust constraint satisfaction for all possible values of the uncertainties—in contrast to standard NMPC. The performance of both was seen to be identical in terms of conservativeness, however it was evident that multi-stage NMPC is the more computationally attractive choice.

It was also demonstrated that there exist methods to successfully deal with the inevitable loss of performance resulting from robust NMPC; the results also suggest that if an efficient implementation is carried out, for certain cases, even real-time implementation may be possible.

All in all, it can be concluded that robust NMPC is an essential tool for advanced control of industrial processes and that the multi-stage NMPC approach, in particular, is rather promising— if designed properly it may be applied for real-time robust control of certain processes.

5.2 Future Work

In the oil production optimization case study, several simplifying assumptions were made in order to maintain focus on the main task—implementation of robust NMPC—and to make it achievable within the given timeframe. Further work on robust control of the oil production process could involve loosening these assumptions and dealing with a more complete representation of the real system. Specifically, the following ideas are proposed.

It was assumed that the optimal flow rates provided by NMPC serve as setpoints to PID controllers, which are coupled together to form a control structure for maintaining the pressure inside the gas distribution manifold near a given setpoint. Such a control structure would typically be used in reality and so it would be interesting to implement it along with the developed robust controller and test the performance of the entire control system.

State feedback was used in the design of the controller, i.e., it was considered that the exact states of the system are available at each timestep. In practice, however, this is not true. Since state information is required by the controller an estimation strategy must be used. Simulation studies have shown that multi-stage NMPC can be applied in combination with state estimation [24], however deterioration of performance due to propagation of measurement noise is reported and it is pointed out that in certain cases explicit consideration of measurement noise and estimation errors in the design of the scenario

tree may be required. Hence, the developed controller should be tested with output feedback (i.e., state estimation based on noisy measurements) to see if the performance is acceptable or modification of the scenario tree is necessitated.

Finally, in order to reduce the computational cost associated with testing, it was also assumed that the PI parameter was the only source of uncertainty and that there were only two wells present in the oil field—instead of five. Even though the methods considered in this thesis for reducing computational cost makes real-time implementation of multi-stage NMPC possible under the assumptions made, the problem may very well be intractable if more wells and more uncertainties were considered. Hence, it is important to look into more efficient ways of implementation—e.g., the performance resulting from full discretization of system dynamics using the orthogonal collocation approach is said to be superior to that achieved with direct multiple-shooting [10] [12].

References

[1] G. Grimm, M. J. Messina, S. E. Tuna, and A. R. Teel, 'Examples when nonlinear model predictive control is nonrobust', *Automatica*, vol. 40, no. 10, pp. 1729–1738, Oct. 2004, doi: 10.1016/j.automatica.2004.04.014.

[2] P.J. Campo and M. Morari, 'Robust model predictive control', *Proceedings of the American Control Conference*, pp. 1021–1026, 1987.

[3] D. Q. Mayne, 'Control of Constrained Dynamic Systems', *Eur. J. Control*, vol. 7, no. 2, pp. 87–99, Jan. 2001, doi: 10.3166/ejc.7.87-99.

[4] J. H. Lee and Z. Yu, 'Worst-case formulations of model predictive control for systems with bounded parameters', *Automatica*, vol. 33, no. 5, pp. 763–781, May 1997, doi: 10.1016/S0005-1098(96)00255-5.

[5] D. Q. Mayne, 'Robust and Stochastic MPC: Are We Going In The Right Direction?', *IFAC-Pap.*, vol. 48, no. 23, pp. 1–8, Jan. 2015, doi: 10.1016/j.ifacol.2015.11.255.

[6] P. O. M. Scokaert and D. Q. Mayne, 'Min-max feedback model predictive control for constrained linear systems', *IEEE Trans. Autom. Control*, vol. 43, no. 8, pp. 1136–1142, Aug. 1998, doi: 10.1109/9.704989.

[7] D. Q. Mayne and E. C. Kerrigan, 'Tube-based Robust Nonlinear Model Predictive Control', *IFAC Proc. Vol.*, vol. 40, no. 12, pp. 36–41, Jan. 2007, doi: 10.3182/20070822-3-ZA-2920.00006.

[8] S. Yu, C. Maier, H. Chen, and F. Allgöwer, 'Tube MPC scheme based on robust control invariant set with application to Lipschitz nonlinear systems', *Syst. Control Lett.*, vol. 62, no. 2, pp. 194–200, Feb. 2013, doi: 10.1016/j.sysconle.2012.11.004.

[9] S. V. Raković, B. Kouvaritakis, M. Cannon, C. Panos, and R. Findeisen, 'Fully Parameterized Tube MPC', *IFAC Proc. Vol.*, vol. 44, no. 1, pp. 197–202, Jan. 2011, doi: 10.3182/20110828-6-IT-1002.03110.

[10] S. Lucia and S. Engell, 'Potential and Limitations of Multi-stage Nonlinear Model Predictive Control', *IFAC-Pap.*, vol. 48, no. 8, pp. 1015–1020, Jan. 2015, doi: 10.1016/j.ifacol.2015.09.101.

[11] S. Lucia and S. Engell, 'Robust nonlinear model predictive control of a batch bioreactor using multi-stage stochastic programming', in *2013 European Control Conference (ECC)*, Jul. 2013, pp. 4124–4129. doi: 10.23919/ECC.2013.6669521.

[12] S. Lucia, 'Robust Multi-stage Nonlinear Model Predictive Control', Ph.D dissertation, Fac. of Biological and Chemical Eng., Technical University of Dortmund, Germany, 2015. [Online]. Available: https://www.iot.tu-berlin.de/fileadmin/fg336/PhD_Dissertation_no_CV.pdf

[13] B. Srinivasan, D. Bonvin, E. Visser, and S. Palanki, 'Dynamic optimization of batch processes: II. Role of measurements in handling uncertainty', *Comput. Chem. Eng.*, vol. 27, no. 1, pp. 27–44, Jan. 2003, doi: 10.1016/S0098-1354(02)00117-5.

[14] S. Lucia, J. A. E. Andersson, H. Brandt, A. Bouaswaig, M. Diehl, and S. Engell, 'Efficient Robust Economic Nonlinear Model Predictive Control of an Industrial Batch Reactor', *IFAC Proc. Vol.*, vol. 47, no. 3, pp. 11093–11098, Jan. 2014, doi: 10.3182/20140824-6-ZA-1003.01817.

[15] S. Thangavel, S. Lucia, R. Paulen, and S. Engell, 'Dual robust nonlinear model predictive control: A multi-stage approach', *J. Process Control*, vol. 72, pp. 39–51, Dec. 2018, doi: 10.1016/j.jprocont.2018.10.003.

[16] L. T. Biegler, *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. SIAM, 2010.

[17] J. Albersmeyer and M. Diehl, 'The Lifted Newton Method and Its Application in Optimization', *SIAM J. Optim.*, vol. 20, no. 3, pp. 1655–1684, Jan. 2010, doi: 10.1137/080724885.

[18] R. Sharma, 'Class Lecture, Topic: Reduction of the size of the LQ optimal control problem.', Department of Electrical engineering, Information Technology and Cybernetics, University of South-Eastern Norway, Porsgrunn, 2020.

[19] D. Krishnamoorthy, S. Skogestad, and J. Jaschke, 'Multistage Model Predictive Control with Online Scenario Tree Update using Recursive Bayesian Weighting', in *2019 18th European Control Conference (ECC)*, Jun. 2019, pp. 1443–1448. doi: 10.23919/ECC.2019.8795839.

[20] A. Wächter and L. T. Biegler, 'On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming', *Math. Program.*, vol. 106, no. 1, pp. 25–57, Mar. 2006, doi: 10.1007/s10107-004-0559-y.

[21] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, 'CasADi: a software framework for nonlinear optimization and optimal control', *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, Mar. 2019, doi: 10.1007/s12532-018-0139-4.

[22] Sharma, R., Fjalestad, K., and Glemmestad, B., 'Modeling and Control of Gas Lifted Oil Field With Five Oil Wells', in *52nd International Conference of Scandinavian Simulation Society, SIMS*, Västerås, Sweden, Sep. 2011, pp. 47–59.

[23] R. Sharma, K. Fjalestad, and B. Glemmestad, 'Optimization of lift gas allocation in a gas lifted oil field as non-linear optimization problem', vol. 33, no. 1, pp. 13–25, Jan. 2012, doi: 10.4173/mic.2012.1.2.

[24] S. Lucia, T. Finkler, D. Basak, and S. Engell, 'A new Robust NMPC Scheme and its Application to a Semi-batch Reactor Example*', *IFAC Proc. Vol.*, vol. 45, no. 15, pp. 69–74, Jan. 2012, doi: 10.3182/20120710-4-SG-2026.00035.

Appendix A
Case Study - Reservoir Control

The main case study presented in this thesis involved continuous-valued uncertainties. Even though the simulation results suggested that the developed multi-stage NMPC could ensure robust constraint satisfaction for all possible values of the uncertainty, there is no real guarantee for this.

The reservoir control case study presented here is particularly interesting because the involved uncertainty takes only a finite (and fairly manageable) number of discrete values. Hence, if implemented properly, multi-stage NMPC could guarantee robust constraint satisfaction for all possible values of the uncertainty and also ensure optimal operation under the presence of uncertainty.

Some preliminary results using deterministic NMPC, leading up to robust NMPC, are presented and discussed in the following sections.

A.1 Process Description

The case study presented here involves the control of the floodgate of a reservoir. A simple illustration of the considered system is given in Figure A.1. There are two inflows into the reservoir: the inflow from the Hjartsjå river and the inflow from the Hjartdøla hydropower station. The forecasts of these inflows are used for control purposes, and they constitute the main sources of uncertainty in the system.

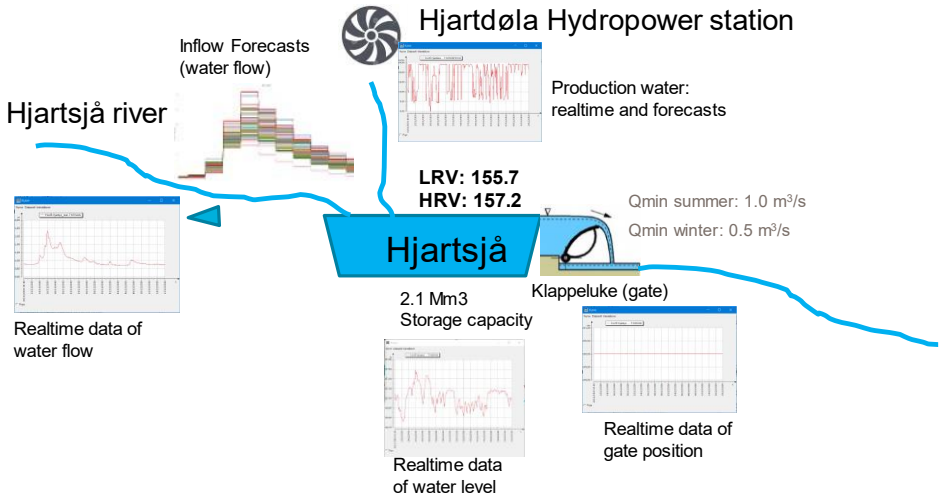

Figure A.1: Illustration of the reservoir
Source: Skagerak Energi

The control task here is to manipulate the floodgate such that the water level in the reservoir is maintained between the minimum and maximum prescribed levels, while also ensuring that the outflow from the reservoir is above a specified seasonal minimum and is stable. The main operational challenge is preventing abrupt drops in the reservoir outflow during low waterflow periods—which can lead to stranded fish and river mussels. This control task is made extremely difficult due to the presence of uncertainty.

The dynamics of the reservoir are described by the following mathematical model (source: Skagerak Energi).

$$\frac{dh}{dt} = \frac{1}{\tilde{A}}(Q_{inf} + Q_{prod} - Q_{flood} - Q_{gate}) \quad \text{(A.1)}$$

$$h_{in} = \min\left(\max(0, h - LRV), h_{in,max}\right) \quad \text{(A.2)}$$

$$A = \max\left(A_{min}, 10^6 \cdot a \cdot b \cdot h_{in}^{b-1}\right) \quad \text{(A.3)}$$

$$h_{out,g} = \max\left(0, h - \left(h_{g,top}^{msl} - h_g\right)\right) \quad \text{(A.4)}$$

$$Q_{gate} = 1.84 \cdot L_1 \cdot h_{out,g}^{1.5} \quad \text{(A.5)}$$

$$h_{out,OF1} = \max\left(0, h - OFT_1^{msl}\right) \quad \text{(A.6)}$$

$$h_{out,OF2} = \max\left(0, h - OFT_2^{msl}\right) \quad \text{(A.7)}$$

$$Q_{flood} = 1.8\left(L_1 \cdot h_{out,OF1}^{1.5} + L_2 \cdot h_{out,OF2}^{1.5}\right) \quad \text{(A.8)}$$

The variables that appear in the model equations are described in Table A.1 and the parameters are described in Table A.2.

Table A.1: Variables of the reservoir model

	Variable	Description	Unit
Differential state	h	Reservoir water level	m MSL
Algebraic states	h_{in}	Water level inside the reservoir	m
	A	Surface area of water in reservoir at the current level h	m^2
	$h_{out,g}$	Water level relevant to outflow through gate	m
	Q_{gate}	Outflow through gate	m^3/s
	$h_{out,OF1}$	Water level relevant to outflow through overflow channel 1	m
	$h_{out,OF2}$	Water level relevant to outflow through overflow channel 2	m
	Q_{flood}	Outflow through overflow channels	m^3/s
Control input	h_g	Gate opening	m
Disturbances	Q_{inf}	Inflow from Hjartsjå river	m^3/s
	Q_{prod}	Inflow from Hjartdøla hydropower station	m^3/s

Table A.2: Parameters of the reservoir model

Parameter	Description	Value	Unit
LRV	Lower regulated value	155.7	m MSL
HRV	Higher regulated value	157.5	m MSL
$h_{in,max}$	Maximum possible water level of the reservoir	$HRV + 3 - LRV = 4.8$	m
A_{min}	Minimum possible surface area of water in the reservoir	10^3	m^2
a	A constant	0.0474	-

b	A constant	1.6898	-
$h_{g,top}^{msl}$	Gate opening – top position	157.37	m MSL
L_1	Width of the gate	12	m
OFT_1^{msl}	Overflow threshold 1	157.5	m MSL
OFT_2^{msl}	Overflow threshold 2	158.5	m MSL
L_2	Width of the overflow channel	11	m

A.2 Standard NMPC

Performance of deterministic NMPC applied to the reservoir process is investigated here.

The cost function minimized at each timestep is chosen as

$$\min_{h_1,\cdots,h_{N_p},h_{g,0},\cdots,h_{g,N_p-1}} \sum_{k=0}^{N_p-1} (h_{k+1} - h_{set})Q(h_{k+1} - h_{set})^{\mathrm{T}} + \Delta h_{g,k} R \Delta h_{g,k}^{\mathrm{T}} \tag{A.9}$$

where h_{set} is the reservoir level setpoint.

The cost function includes a tracking term $(h_{k+1} - h_{set})Q(h_{k+1} - h_{set})^{\mathrm{T}}$ for the reservoir level and a regularization term $\Delta h_{g,k} R \Delta h_{g,k}^{\mathrm{T}}$ for the control signal. The tuning parameters Q and R are chosen to be 1 and 0.3, respectively.

The relevant process constraints are as follows.

$$0 \leq h_g \leq 1.5 \tag{A.10}$$

$$LRV \leq h \leq HRV \tag{A.11}$$

$$1 \leq Q_{out} \tag{A.12}$$

The simulation results obtained using actual recorded inflows are shown in Figure A.2; a timestep of one hour, a prediction horizon length of 13 days, and a simulation timespan of one month (Dec 2020) is considered.

It is important to first note that the inflow from the hydropower station Q_{prod} fluctuates quite rapidly between 10 – 30 m^3/s. The effect of these fluctuations on other variables is readily seen. It is more clearly reflected in the reservoir outflow Q_{out}: Q_{out} closely resembles Q_{prod}. Given the setpoint tracking term in the objective function, this behaviour is expected; when the inflow to the reservoir changes, to maintain the level near the setpoint, the outflow is adjusted by manipulating the floodgate. In comparison to Q_{prod}, the inflow from the river Q_{inf} is observed to have little impact on performance; this is because it is more stable and has a lower range of variability (at least during the period considered here).

During the period between Dec 08 and 14 there is a noticeable tracking error. The source of this error is obvious: Q_{prod} is more or less at its maximum level throughout. The two inflows combined together cause the control signal to saturate at its maximum limit; since the outflow cannot be increased further, the reservoir level is raised past the setpoint. In contrast, during the period between Dec 19 and 21, the setpoint is tracked perfectly; the input disturbance $Q_{out} = 0$.

All in all, given the rapid variations in the input disturbance, the setpoint tracking performance of standard NMPC could be deemed quite good. And even if the performance is not perfect with regard to setpoint tracking, the success in satisfying the process constraints should be appreciated; none of the constraints are violated throughout the simulation run.

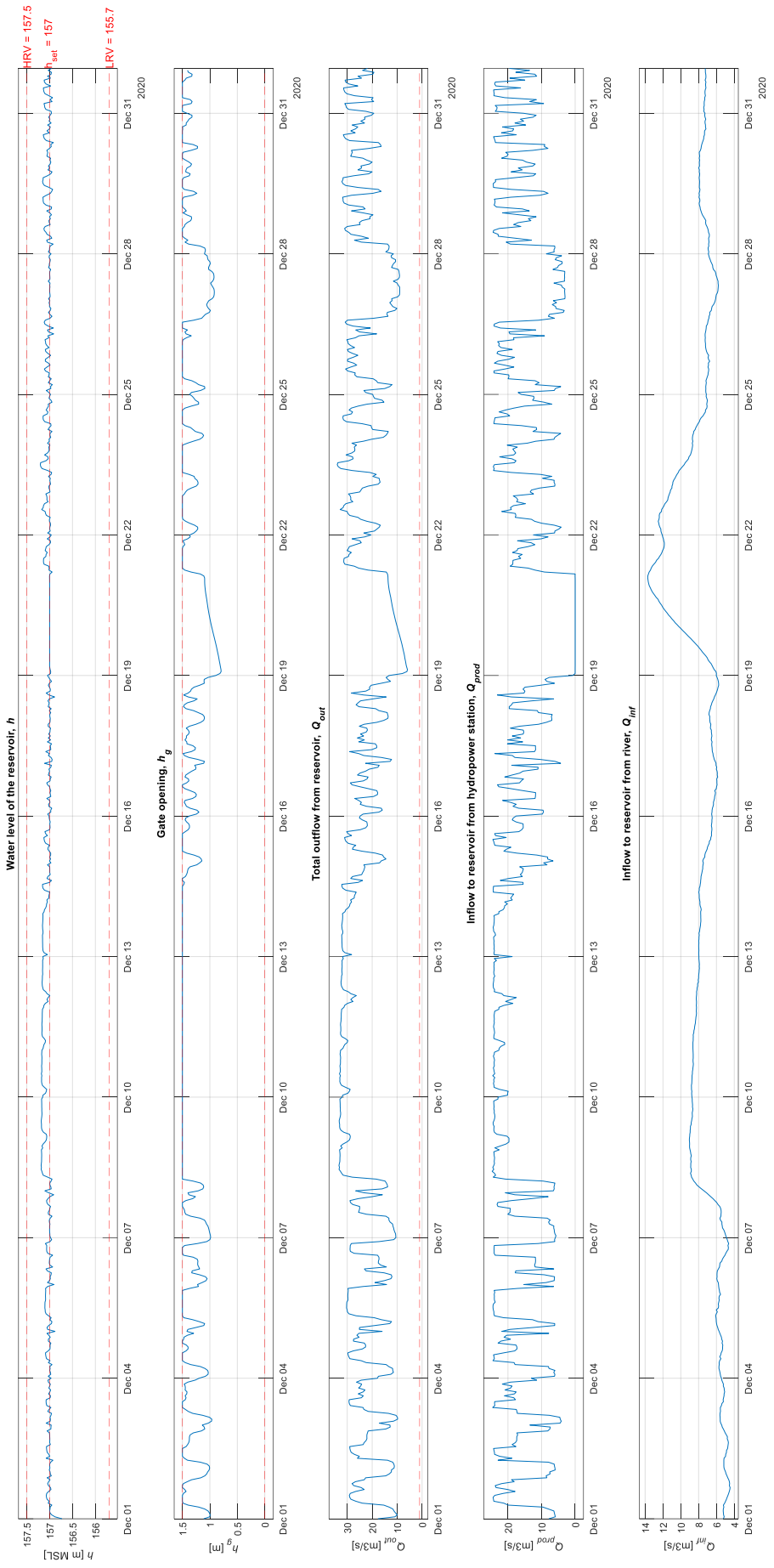

Figure A.2: Simulation results obtained using deterministic NMPC

A.3 Discussion

On the whole, the results presented here indicated that operation of the reservoir could benefit from implementation of standard NMPC, *if* there are no uncertainties associated with the system. But when uncertainty is introduced, the performance will surely deteriorate and constraint violations may occur; both the state and reservoir outflow constraints given by equation (A.11) and (A.12) are susceptible to uncertainty. Essentially, there might not be any advantage to using standard NMPC. Hence, it is of interest to investigate the feasibility robust NMPC.

Unlike the oil production case study, the uncertainties here take discrete values; the inflows to the reservoir are available as an ensemble forecast consisting of 50 members, over a timespan of about 13 days. When developing a multi-stage NMPC the scenario tree would branch only at the initial node and would ideally include 50 scenarios—each corresponding to a different ensemble member; the robust horizon assumption is not applicable here.

Even though there are only a finite number of possible scenarios, since the considered prediction horizon is quite long, the resulting optimization problem would still be quite large—in fact, it would be substantially larger than those encountered in the oil production case study—if all the scenarios are considered. While the strategies discussed in this thesis for reducing computation time may prove to be useful here—especially the control input grouping strategy—it is advisable to further investigate ways for dealing with this.

Finally, it should be noted that the simulation study conducted here is only a preliminary investigation intended to see how effective NMPC is at meeting the most basic control requirements. Indeed, the OCP formulation could—and should—be tweaked for better performance, particularly with regard to controlling the reservoir outflow, which was not addressed here. Furthermore, the study considered only the inflow data records for the month of December; it is important to also study the performance of standard NMPC for the flood seasons and adjust the optimization problem formulation accordingly.

Appendix B
MATLAB Files

A list of all the MATLAB script and function files used to produce the results of this thesis is given in Table B.1.

Table B.1: MATLAB files

	File name	Description
Common files	**oil_field_model.m**	Function used to implement the oil field model equations.
	update_states.m	Function used to implement the RK4 scheme.
	make_w_gc.m	Function used to create the input disturbance to the process.
	calculate_output.m	Function used to calculate the outputs of the process.
Files specific to standard NMPC	**settings_standard_NMPC.m**	Script used to specify the settings and generate data.
	create_NLP_prob_standard_NMPC.m	Function used to create the NLP problem.
	compute_both_standard_NMPC.m	Function used to evaluate the objective function and the constraints.
	standard_NMPC.m	Script used to implement the sliding horizon strategy.
Files specific to multi-stage NMPC	**settings_multi_stage_NMPC.m**	Script used to specify the settings and generate data.
	create_NLP_prob_multi_stage_NMPC.m	Function used to create the NLP problem.
	compute_both_multi_stage_NMPC.m	Function used to evaluate the objective function and the constraints.
	multi_stage_NMPC.m	Script used to implement the sliding horizon strategy.
Main files	**openloop_simulation.m**	Main script used to produce the openloop simulation results.
	openloop_plus_NMPC.m	Main script used to produce both standard NMPC and multi-stage NMPC results.

oil_field_model.m

```
function  [dm, w_op, w_gop] = oil_field_model(m, w_ga, d_PI)

% NOTE:
%     * This function can accept matrix inputs. But make sure rows of m,
%       w_ga & d_PI correspond to instances.
%
%     * Outputs are row vectors or, for matrix inputs, their instances are
%       arranged in rows.

%--------------------------------------------------------------------------
% Extracting info. from the inputs
%--------------------------------------------------------------------------
    % Finding the number of instances
    no_inst = size(m,1);

    % Extracting the 3 states separately from m
    no_wells = 2;
    m_ga = m(:, 1            :no_wells);
    m_gt = m(:, no_wells+1   :2*no_wells);
    m_ot = m(:, 2*no_wells+1 :end);

%--------------------------------------------------------------------------
% Parameters
%--------------------------------------------------------------------------
    % Well      = [Well1    Well2]
    PI_nominal        = [2.51     1.63]*1e4.*ones(no_inst,1); %[kg/hr/bar]
    PI          = PI_nominal + d_PI*1e4;
    L_a_tl      = [2758     2559].*ones(no_inst,1); %[m]
    L_t_tl      = L_a_tl; %[m]
    L_a_vl      = [2271     2344].*ones(no_inst,1); %[m]
    L_t_vl      = L_a_vl; %[m]
    ID_t        = [6.18     6.18]*0.0254.*ones(no_inst,1); %[m]
    A_t         = pi/4*ID_t.^2;
    ID_a        = [9.63     9.63]*0.0254.*ones(no_inst,1); %[m]
    OD_t        = [7.64     7.64]*0.0254.*ones(no_inst,1); %[m]
    A_a        = pi/4*(ID_a.^2 - OD_t.^2);
    L_r_vl      = [114      67].*ones(no_inst,1); %[m]
    K           = [68.43    67.82].*ones(no_inst,1); %[sqrt(kg*m3/bar)/hr]
    T_a               = [280      280].*ones(no_inst,1); %[K]
    T_t               = [280      280].*ones(no_inst,1); %[K]
    u_2         = [100      100].*ones(no_inst,1); %[%]

    P_r         = 150; %[bar]
    P_s         = 30; %[bar]
    N_6         = 27.3;
    alpha_Y     = 0.66;
    rho_o       = 700; %[kg/m3]
    M                 = 20*1e-3; %[kg/mol]
    z           = 1.3;
    P_ainj_min  = 0; %[bar]
    P_wh_min    = 0; %[bar]
    R           = 8.31446261815324; %[J/K/mol]
    g           = 9.80665; %[m/s2]

%--------------------------------------------------------------------------
```

```
% Algebraic equations
%-------------------------------------------------------------------------
                P_a = z*m_ga*R.*T_a./(M*A_a.*L_a_tl)*1e-5; %[bar] (6)

            P_ainj = P_a + m_ga*g.*L_a_vl./(A_a.*L_a_tl)*1e-5; %[bar] (11)

                V_G = A_t.*L_t_tl - m_ot/rho_o; %[m3] (eq. after 13)

                rho_m = (m_gt + m_ot)./(A_t.*L_t_tl); %[kg/m3] (eq. after 15)

            P_tinj = (z*m_gt*R.*T_t./(M*V_G) + rho_m*g.*L_t_vl/2)*1e-5; %[bar] (17)

        Y_2 = 1 - alpha_Y*(P_ainj - P_tinj)./max(P_ainj, P_ainj_min); %[const.] (eq. after
10)

        rho_ga = M*(P_a + P_ainj)*1e5./(2*z*R*T_a); %[kg/m3] (12)

    w_ginj = K.*Y_2.*sqrt(rho_ga.*max(P_ainj - P_tinj, 0))/3600; %[kg/s] (10)
    %-------------------------------------------------------------------------
        P_wf = P_tinj + rho_o*g*L_r_vl*1e-5; %[bar] (18)

    w_o = PI.*max(P_r - P_wf, 0)/3600; %[kg/s] (19)
    %-------------------------------------------------------------------------
            P_wh = (z*m_gt*R.*T_t./(M*V_G) - rho_m*g.*L_t_vl/2)*1e-5; %[bar] (16)

        Y_3 = 1 - alpha_Y*(P_wh - P_s)./max(P_wh, P_wh_min); %[const.] (eq. after 20)

    w_gop = 10*N_6*(0.5*u_2 - 20).*Y_3.*sqrt(rho_m.*max(P_wh - P_s, 0))/3600; %[kg/s] (20)
    %-------------------------------------------------------------------------
    w_gp = m_gt./(m_gt + m_ot).*w_gop; %[kg/s] (21)
    %-------------------------------------------------------------------------
    w_op = m_ot./(m_gt + m_ot).*w_gop; %[kg/s] (22)

%-------------------------------------------------------------------------
% Differential equations
%-------------------------------------------------------------------------
    dm_ga = w_ga - w_ginj; %[kg/s] (9)
    dm_gt = w_ginj - w_gp; %[kg/s] (23)
    dm_ot = w_o - w_op; %[kg/s] (24)

    dm = [dm_ga, dm_gt, dm_ot];
```

update_states.m

```
function m_next = update_states(m0, w_ga, d_PI, dt)

% RK4 algorithm

dm1 = oil_field_model(m0, w_ga, d_PI);
dm2 = oil_field_model(m0+0.5*dt*dm1, w_ga, d_PI);
dm3 = oil_field_model(m0+0.5*dt*dm2, w_ga, d_PI);
dm4 = oil_field_model(m0+dt*dm3, w_ga, d_PI);

m_next = m0 + dt/6*(dm1 + 2*dm2 + 2*dm3 + dm4);
```

make_w_gc.m

```
function w_gc = make_w_gc(type, w_gc_val_rat, t)

% Length of w_gc
len = length(t) - 1;

% Extract w_gc values and step ratio separately
w_gc_val    = w_gc_val_rat(:,1);
step_ratio  = w_gc_val_rat(:,2);

% Change the units of input disturbance values
w_gc_val = w_gc_val*0.83/60/60; %[kg/s]

switch type
    case 'constant'
        w_gc = w_gc_val*ones(len,1); %[kg/s]

    case '1step'
        % Lengths of the steps
        step_len1 = floor(len*step_ratio(1)/sum(step_ratio));
        step_len2 = len - step_len1;

        % Input disturbance
        w_gc = [w_gc_val(1)*ones(step_len1,1);
                w_gc_val(2)*ones(step_len2,1)]; %[kg/s]

    case '2step'
        % Lengths of the steps
        step_len1 = floor(len*step_ratio(1)/sum(step_ratio));
        step_len2 = floor(len*step_ratio(2)/sum(step_ratio));
        step_len3 = len - step_len1 - step_len2;

        % Input disturbance
        w_gc = [w_gc_val(1)*ones(step_len1,1);
                w_gc_val(2)*ones(step_len2,1);
                w_gc_val(3)*ones(step_len3,1)]; %[kg/s]
end
```

calculate_output.m

```
function [w_op, w_gop] = calculate_output(m)

% NOTE
%     * w_ga & d_PI is not required for calculating w_op & w_gop using the
%       oil_field_model; they are required only for calculating the
%       derivatives of the states. So, this function passes a dummy_w_ga &
%       a dummy_d_PI to oil_field_model.

dummy_w_ga = zeros(size(m,1),2);
dummy_d_PI = zeros(size(m,1),2);

[~, w_op, w_gop] = oil_field_model(m, dummy_w_ga, dummy_d_PI);
```

settings_standard_NMPC.m

```
%--------------------------------------------------------------------------
% Settings
%--------------------------------------------------------------------------
    timespan = 4.5*60*60; %[s]
    Np = 25; %[time steps]
    %----------------------------------------------------------------------
    % Control input grouping
    % Specify group lengths as a vector, groups. If there's no grouping,
    % set groups = 0.
    % NOTE: sum(groups) should be = Np
    groups = 0;
%     groups = [1 3 9 12]; %25
    %----------------------------------------------------------------------
    % Well          = [Well1    Well2]
    d_PI_controller    = [0        0]; %*1e4 [kg/hr/bar]
    %----------------------------------------------------------------------
    % Select the type of input disturbance, w_gc
    % NOTE: 1st column of w_gc_val correspond to value of w_gc in Sm3/hr
    %       2nd column of w_gc_val correspond to ratio of step lengths
%     w_gc_type = 'constant';    w_gc_val = [  40000, 1];
    w_gc_type = '1step';       w_gc_val = [  40000, 3;
                                           35000, 1.5];
%--------------------------------------------------------------------------
% Creating other necessary data
%--------------------------------------------------------------------------
    % No. of decision variables, Ndv
    if groups == 0 % When there is no grouping
        Ndv = Np;
    else % When there is grouping
        Ndv = length(groups);
    end
    %----------------------------------------------------------------------
    % NOTE: The same d_PI_plant used for openloop simulation is used
    % Well          = [Well1    Well2]
    d_PI_plant      = d_PI_plant_OL; %*1e4 [kg/hr/bar]
    %----------------------------------------------------------------------
    % NOTE: The same dt used for openloop simulation is used
    dt = dt_OL; %[s]

%--------------------------------------------------------------------------
% Confirming the settings
%--------------------------------------------------------------------------
    if groups ~= 0 % If there is grouping
        if sum(groups) ~= Np % and if the group lengths don't add up to the length of the
prediction horizon
            error('Group lengths do not add up to the length of the prediction horizon')
        end
    end

    % Displaying & writing the settings to a file
    fileID = fopen('saved_settings\my_settings_file.txt','w');
    fprintf(fileID, 'NMPC SETTINGS \n');
    fprintf(fileID, ' Np              : %d timesteps      \n',    Np);
    fprintf(fileID, ' I/P grouping    : %s                \n',    mat2str(groups));
    fprintf(fileID, ' Ndv             : %d                  \n',    Ndv*2);
```

```
    fprintf(fileID, ' dt             : %g s              \n',    dt);
    fprintf(fileID, ' PH length       : %g min            \n',    round(Np*dt/60,1));
    fprintf(fileID, ' timespan       : %g hrs            \n',    timespan/60/60);
    fprintf(fileID, ' d_PI_controller       :                    \n');
    fprintf(fileID, '                %7g %7g              \n',    d_PI_controller);
    fprintf(fileID, ' d_PI_plant     :                    \n');
    fprintf(fileID, '                %7g %7g             \n',      d_PI_plant);
    fprintf(fileID, ' w_gc            : %s, %s Sm3/hr      \n',    w_gc_type,
mat2str(w_gc_val(:,1)));
    type('saved_settings\my_settings_file.txt');

    % Prompting the user to confirm the settings
    disp(' ')
    disp('Please make sure the above settings are okay. Then press any key to start the
simulation OR press ctrl + c to stop execution.')
    pause
    disp(' ')
    disp('Formulating the NLP problem & setting up the optimizer...')

%-----------------------------------------------------------------------
% Creating other necessary data (cont.)
%-----------------------------------------------------------------------
    no_wells = 2;
    t = 0:dt:timespan; %[s]
    %-----------------------------------------------------------------------
    % NOTE: The final states from the openloop simulation are used as the
    % initial states for MPC
    m0 = m_OL(end,:);
    %-----------------------------------------------------------------------
    w_gc = make_w_gc(w_gc_type, w_gc_val, t);

    % + (Np-1) copies of the last w_gc value
    w_gc = [w_gc;
            w_gc(end)*ones(Np-1,1)];
    %-----------------------------------------------------------------------
    % Starting point for the optimizer
    % NOTE: The last w_ga and m from the openloop simulation are used as
    % the starting point for the optimizer
    w_ga_ini = w_ga_OL(end,:).*ones(Ndv,1);
    m_ini = m_OL(end,:).*ones(Np+1,1);
```

create_NLP_prob_standard_NMPC.m

```
function NLP_prob = create_NLP_prob_standard_NMPC(dt, Np, Ndv, groups, d_PI_controller,
no_wells)

% Importing CasADi
addpath('C:\Program Files\MATLAB\casadi-windows-matlabR2016a-v3.5.5')
import casadi.*

%-----------------------------------------------------------------------
% Defining symbols
%-----------------------------------------------------------------------
    % w_ga [Ndv, w_ga]
    w_ga = SX.sym('w_ga', Ndv, no_wells);
        % vect_w_ga
            % Transposing w_ga -- for easy extraction of the 1st optimal
```

```
            % control move, w_ga_ast(1,:), from the optimal solution,
            % x_ast, given by the NLP_solver)
            vect_w_ga = w_ga';
            vect_w_ga = vect_w_ga(:);

    % m [(Np+1) x m]
    m = SX.sym('m', Np+1, 3*no_wells);
        % vect_m
        vect_m = m(:);

    % Decision variables, x = [w_ga; m]
    x = [   vect_w_ga;
            vect_m];
    %--------------------------------------------------------------------------
    % m0 [1 x 3*no_wells]
    m0 = SX.sym('m0', 1, 3*no_wells);

    % w_ga_last [1 x no_wells]
    w_ga_last = SX.sym('w_ga_last', 1, no_wells);

    % Parameters, p = [m0, w_ga_last]
    p = [m0, w_ga_last];
%--------------------------------------------------------------------------

% Computing the objective function, F, and the constraints, G
[F, G] = compute_both_standard_NMPC(Np, dt, m0, m, w_ga, groups, d_PI_controller, w_ga_last);

% Creating the NLP problem
NLP_prob = struct('f', F, 'x', x, 'g', G, 'p', p);
```

compute_both_standard_NMPC.m

```
function [F, G] = compute_both_standard_NMPC(Np, dt, m0, m, w_ga, groups, d_PI_controller,
w_ga_last)

% Importing CasADi
addpath('C:\Program Files\MATLAB\casadi-windows-matlabR2016a-v3.5.5')
import casadi.*

%--------------------------------------------------------------------------
% Finding the information needed for evaluating the objective function &
% constraints
%--------------------------------------------------------------------------
    % w_ga_reconst [Np x w_ga]
    if groups ~= 0 % if there is grouping reconstruct w_ga
        w_ga_reconst = repelem(w_ga,groups,1);
    else
        w_ga_reconst = w_ga;
    end
    %--------------------------------------------------------------------------
    % Openloop simulation over the PH

        % Preallocation for continuity constraints (process dynamics), G1
        % [(Np+1) x (dv_m - pred_m)]
        G1 = SX.sym('G1', (Np+1), length(m0));
        G1(1,:) = m(1,:) - m0; % 1st continuity constraint (length(m0) number of them)
```

```
        % For each time step over the prediction horizon
        for k = 1:Np

            m_kp1 = update_states(m(k,:), w_ga_reconst(k,:), d_PI_controller, dt);

            % (k+1)th continuity constraint
            G1(k+1,:) = m(k+1,:) - m_kp1;

        end

        [w_op, w_gop] = calculate_output(m(2:end,:));
    %--------------------------------------------------------------------------
    % oil_prod [Np x oil_prod]
    oil_prod = sum(w_op,2);

    % fluid_prod [Np x fluid_prod]
    fluid_prod = sum(w_gop,2);

    % delta_w_ga [Np x delta_w_ga]
    delta_w_ga = w_ga_reconst(1,:) - w_ga_last;
    delta_w_ga = [  delta_w_ga;
                    w_ga_reconst(2:end,:) - w_ga_reconst(1:end-1,:)];

    % sum_w_ga [Np x sum_w_ga]
    sum_w_ga = sum(w_ga_reconst,2);
%--------------------------------------------------------------------------

% Objective function, F
Q = 1;
R = 0.5;
S = 50;
F = sum(-Q*oil_prod.^2 + R*sum(w_ga_reconst.^2,2) + S*sum(delta_w_ga.^2,2));

% Inequality constraints, G2
G2 = [  fluid_prod;
        delta_w_ga(:);
        sum_w_ga];

% Constraints, G
G = [   G1(:);
        G2];
```

standard_NMPC.m

```
% Importing CasADi
addpath('C:\Program Files\MATLAB\casadi-windows-matlabR2016a-v3.5.5')
import casadi.*

% Fetching the settings and other required data
settings_standard_NMPC

% Creating the NLP problem
NLP_prob = create_NLP_prob_standard_NMPC(dt, Np, Ndv, groups, d_PI_controller, no_wells);

% Specifying NLP solver options
options = struct;
options.ipopt.max_iter = 5000;
```

```
options.ipopt.print_level = 0; % 3, 5(default), upto 12
options.print_time = false;
options.ipopt.fixed_variable_treatment = 'make_constraint';
options.ipopt.warm_start_init_point = 'yes';

% Creating the NLP solver
NLP_solver = nlpsol('NLP_solver', 'ipopt', NLP_prob, options);

% Container for arguments passed to NLP_solver
% args {lbx, ubx, lbg, ubg, x0, p, lam_x0, lam_g0}
args = struct;

% Specifying the bounds of the NLP problem

    % Bounds on the decision variables
    %       LBX     <= x                <= UBX
    %       0.323   <= w_ga             <= 11.66     [Ndv*no_wells]
    %       0       <= m                <= inf      [(Np+1)*(3*no_wells)]
    args.lbx = repelem([0.323; 0],      [Ndv*no_wells, (Np+1)*(3*no_wells)]);
    args.ubx = repelem([11.66; inf],    [Ndv*no_wells, (Np+1)*(3*no_wells)]);

    % Bounds of the constraints
    %       LBG     <= G(x, p)          <= UBG
    %       0       <= dv_m - pred_m    <= 0        [(Np+1)*(3*no_wells)]
    %       0       <= fluid_prod            <= 160      [Np]
    %       -0.15   <= delta_w_ga            <= 0.15     [Np*no_wells]
    %       0       <= sum_w_ga              <= w_gc     [Np]
    args.lbg = repelem([0; 0; -0.15; 0],    [(Np+1)*(3*no_wells), Np, Np*no_wells, Np]);
    args.ubg = repelem([0; 160; 0.15],      [(Np+1)*(3*no_wells), Np, Np*no_wells]);

%-------------------------------------------------------------------------
% Running standard NMPC
%-------------------------------------------------------------------------
    loop_length = length(t)-1;

    % Preallocating
    m               = zeros(loop_length+1,length(m0));      m(1,:) = m0;
    w_ga            = zeros(loop_length,no_wells);
    loop_time       = zeros(loop_length,1);
    controller_time = zeros(loop_length,1);

    % Initializing the ODE solver used for simulating the real plant
    update_plant = @(m0_k, w_ga_k)update_states(m0_k, w_ga_k, d_PI_plant, dt);

    % Announcing the start of simulation
    disp(' ')
    disp('Starting standard NMPC...')
    disp(datetime('now'))

    elapsedTime = tic;
    % Sliding horizon strategy
    for k = 1:loop_length
        tic

        if k == 1 % lam_x0 & lam_g0 needed for warm-start not known
            args.x0 = [w_ga_ini(:); m_ini(:)];
            args.p = [m(k,:), w_ga_OL(end,:)];
```

```
            args.ubg(end+1:end+Np) = w_gc(k:k+Np-1);

            opt_time = tic;
            % Solving the NLP problem
            sol = NLP_solver(   'x0',       args.x0,...
                                'p',        args.p,...
                                'lbx',      args.lbx,...
                                'ubx',      args.ubx,...
                                'lbg',      args.lbg,...
                                'ubg',      args.ubg);
            opt_time = toc(opt_time);
        else
            args.x0 = sol.x;
            args.p = [m(k,:), w_ga(k-1,:)];
            args.ubg(end-Np+1:end) = w_gc(k:k+Np-1);
            args.lam_x0 = sol.lam_x;
            args.lam_g0 = sol.lam_g;

            opt_time = tic;
            % Solving the NLP problem
            sol = NLP_solver(   'x0',       args.x0,...
                                'p',        args.p,...
                                'lbx',      args.lbx,...
                                'ubx',      args.ubx,...
                                'lbg',      args.lbg,...
                                'ubg',      args.ubg,...
                                'lam_x0',   args.lam_x0,...
                                'lam_g0',   args.lam_g0);
            opt_time = toc(opt_time);
        end

        % Extracting the 1st optimal control move, w_ga_ast
        w_ga_ast = full(sol.x(1:no_wells))';

        % Applying the first (optimal) control move to the plant, i.e.,
        % sliding one timestep forward
        m(k+1,:) = update_plant(m(k,:), w_ga_ast);

        % Storing variables of interest
        w_ga(k,:) = w_ga_ast;
        controller_time(k) = opt_time;
        loop_time(k) = toc;
    end
    elapsedTime = toc(elapsedTime);

    % Announcing the end of simulation
    disp('Standard NMPC complete!')
    fprintf(['Elapsed time = \n\n',...
           '       %g min\n\n'], round(elapsedTime/60,1));
    disp('Please wait for the plots of the results...')
%--------------------------------------------------------------------------

% Writing elapsedTime to file
fprintf(fileID, '     Elapsed time    : %g min', round(elapsedTime/60,1));
fclose(fileID);

% Preparing to plot the results
```

```
    % Removing m0 from m (Reason: m0 is not required when combining with
    % openloop simulation results)
    m = m(2:end,:);

    % Calculating the outputs
    [w_op, w_gop] = calculate_output(m);

    % Removing the + (Np-1) copies of the last w_gc value
    w_gc = w_gc(1:end-(Np-1));
```

settings_multi_stage_NMPC.m

```
%-------------------------------------------------------------------------
% Settings
%-------------------------------------------------------------------------
    timespan = 3*60*60; %[s]
    Nr = 1; %[time steps]
    Np = 25; %[time steps]
    %---------------------------------------------------------------------
    % Control input grouping
    % Specify group lengths as a vector, groups. If there's no grouping,
    % set groups = 0.
    % NOTE: sum(groups) should be = Np-Nr
    groups = 0;
%     groups = [2 4 8 10]; %24
%     groups = [2 2 2 2 2 4 4 6]; %24
    %---------------------------------------------------------------------
    % Specify the values of the uncertainties included in the scenario tree
    % Well          = [Well1    Well2]
    d_PI_controller    = [ +0.25      +0.25;
                           -0.25        +0.25;
                           0     0;
                           +0.25        -0.25;
                           -0.25        -0.25]; %*1e4 [kg/hr/bar]
    %---------------------------------------------------------------------
    % Select the type of input disturbance, w_gc
    % NOTE: 1st column of w_gc_val correspond to value of w_gc in Sm3/hr
    %       2nd column of w_gc_val correspond to ratio of step lengths
%     w_gc_type = 'constant';    w_gc_val = [  40000, 1];
    w_gc_type = '1step';       w_gc_val = [  40000, 2;
                                           33000, 1];
%-------------------------------------------------------------------------
% Creating other necessary data
%-------------------------------------------------------------------------
    % No. of uncertainties included in the scenario tree, Nd
    Nd = size(d_PI_controller,1);
    %---------------------------------------------------------------------
    % No. of scenarios, Ns
    Ns = Nd^Nr;
    %---------------------------------------------------------------------
    % No. of decision variables, Ndv

        % No. of decision variables in the head-part, Nhead_dv
        Nhead_dv = (1-Nd^Nr)/(1-Nd);

        % No. of decision variables in the tail-part, Ntail_dv
```

```
        if groups == 0 % When there is no grouping
            Ntail_dv = Ns*(Np-Nr);
        else % When there is grouping
            Ntail_dv = Ns*length(groups);
        end

    Ndv = Nhead_dv + Ntail_dv;
    %-------------------------------------------------------------------------
    % NOTE: The same d_PI_plant used for openloop simulation is used
    % Well          = [Well1    Well2]
    d_PI_plant      = d_PI_plant_OL; %*1e4 [kg/hr/bar]
    %-------------------------------------------------------------------------
    % NOTE: The same dt used for openloop simulation is used
    dt = dt_OL; %[s]

%-------------------------------------------------------------------------
% Confirming the settings
%-------------------------------------------------------------------------
    if groups ~= 0 % If there is grouping
        if sum(groups) ~= (Np-Nr) % and if the group lengths don't add up to the tail length,
(Np-Nr)
            error('Group lengths do not add up to the tail length, (Np-Nr) = %d', Np-Nr)
        end
    end

    % Displaying & writing the settings to a file
    fileID = fopen('saved_settings\my_settings_file.txt','w');
    fprintf(fileID, 'MULTI-STAGE NMPC SETTINGS \n');
    fprintf(fileID, ' Nr              : %d                \n',     Nr);
    fprintf(fileID, ' Np              : %d %s             \n',     Np, mat2str([ones(1,Nr),
groups]));
    fprintf(fileID, ' Nd              : %d                 \n',    Nd);
    fprintf(fileID, ' Ns              : %d                 \n',    Ns);
    fprintf(fileID, ' Ndv             : %d                 \n',    Ndv*2);
    fprintf(fileID, ' dt              : %g s               \n',    dt);
    fprintf(fileID, ' PH length       : %g min             \n',    round(Np*dt/60,1));
    fprintf(fileID, ' timespan        : %g hrs             \n',    timespan/60/60);
    fprintf(fileID, ' d_PI_controller       :                    \n');
    fprintf(fileID, '                       %7g %7g            \n',      d_PI_controller');
    fprintf(fileID, ' d_PI_plant      :                    \n');
    fprintf(fileID, '                       %7g %7g            \n',      d_PI_plant);
    fprintf(fileID, ' w_gc            : %s, %s Sm3/hr     \n',    w_gc_type,
mat2str(w_gc_val(:,1)));
    type('saved_settings\my_settings_file.txt');

    % Prompting the user to confirm the settings
    disp(' ')
    disp('Please make sure the above settings are okay. Then press any key to start the
simulation OR press ctrl + c to stop execution.')
    pause
    disp(' ')
    disp('Formulating the NLP problem & setting up the optimizer...')

%-------------------------------------------------------------------------
% Creating other necessary data (cont.)
%-------------------------------------------------------------------------
    no_wells = 2;
```

```
t = 0:dt:timespan; %[s]
%-------------------------------------------------------------------------
% NOTE: The final states from the openloop simulation are used as the
% initial states for MPC
m0 = m_OL(end,:);
%-------------------------------------------------------------------------
w_gc = make_w_gc(w_gc_type, w_gc_val, t);

% + (Np-1) copies of the last w_gc value
w_gc = [w_gc;
        w_gc(end)*ones(Np-1,1)];
%-------------------------------------------------------------------------
% Starting point for the optimizer
% NOTE: The last w_ga and m from the openloop simulation are used as
% the starting point for the optimizer
w_ga_ini = w_ga_OL(end,:).*ones(Ndv,1);
m_ini = m_OL(end,:).*ones(Ns*(Np+1),1);
%-------------------------------------------------------------------------
% d_PI combinations (throughout the horizon) for all the scenarios

    % d_PI position combinations [scenarios x Np]

        % Position no.s over the robust horizon [scenarios x Nr]
        d_PI_posi_comb = unique(rem(nchoosek(0:Ns-1,Nr),Nd)+1,'rows');

        % Adding the position no.s over the remaining Np-Nr stages
        % (const. disturbance)
        d_PI_posi_comb = [d_PI_posi_comb, repmat(d_PI_posi_comb(:,end),1,Np-Nr)];

    % d_PI (value) combinations [scenarios x d_PI x Np]

        d_PI_comb = d_PI_controller(d_PI_posi_comb(:),:);
        d_PI_comb = reshape(d_PI_comb,Ns,Np,size(d_PI_controller,2));
        d_PI_comb = permute(d_PI_comb,[1 3 2]);
%-------------------------------------------------------------------------
% w_ga position combinations (throughout the horizon) for all the
% scenarios
% w_ga_posi_comb [scenarios x Np]
% This is required for reconstructing w_ga using w_ga values returned
% by the optimizer

if groups == 0 % When there's no grouping

    % w_ga position numbers
    dv = (1:Ndv)';

    % No. of decision variables per stage
    Ndv_stage = [Nd.^(0:Nr-1), Ns*ones(1,Np-Nr)];

    % No. of repetitions of one decision variable in each stage
    rep_stage = [Nd.^flip(1:Nr), ones(1,Np-Nr)];

    % No. of repetitons per decision variable
    rep_dv = repelem(rep_stage, Ndv_stage)';

    w_ga_posi_comb = repelem(dv, rep_dv);
    w_ga_posi_comb = reshape(w_ga_posi_comb, Ns, Np);
```

```
    else % When there is grouping

        % Head-part

            % w_ga position numbers
            dv_head = (1:Nhead_dv)';

            % No. of decision variables per stage
            Ndv_stage = Nd.^(0:Nr-1);

            % No. of repetitions of one decision variable in each stage
            rep_stage = Nd.^flip(1:Nr);

            % No. of repetitons per decision variable
            rep_dv = repelem(rep_stage, Ndv_stage)';

            w_ga_posi_comb_head = repelem(dv_head, rep_dv);
            w_ga_posi_comb_head = reshape(w_ga_posi_comb_head, Ns, Nr);

        % Tail-part

            % w_ga position numbers
            dv_tail = (Nhead_dv+1:Ndv)';

            w_ga_posi_comb_tail = reshape(dv_tail, Ns, length(groups));
            w_ga_posi_comb_tail = repelem(w_ga_posi_comb_tail,1,groups);

        w_ga_posi_comb = [w_ga_posi_comb_head, w_ga_posi_comb_tail];
    end
```

create_NLP_prob_multi_stage_NMPC.m

```
function NLP_prob = create_NLP_prob_multi_stage_NMPC(dt, Np, Ns, Ndv, w_ga_posi_comb,
d_PI_comb, no_wells)

% Importing CasADi
addpath('C:\Program Files\MATLAB\casadi-windows-matlabR2016a-v3.5.5')
import casadi.*

%-------------------------------------------------------------------------
% Defining symbols
%-------------------------------------------------------------------------
    % w_ga [Ndv, w_ga]
    w_ga = SX.sym('w_ga', Ndv, no_wells);
        % vect_w_ga
            % Transposing w_ga -- for easy extraction of the 1st optimal
            % control move, w_ga_ast(1,:), from the optimal solution,
            % x_ast, given by the NLP_solver)
            vect_w_ga = w_ga';
            vect_w_ga = vect_w_ga(:);

    % m {(Np+1) x [Ns x m]}
    m = SX.sym('m', Ns, 3*no_wells, (Np+1))';
        % vect_m
            % Converting m from {(Np+1) x [Ns x m]} to [Ns*(Np+1) x m]
            vect_m = vertcat(m{:});
```

```
            vect_m = vect_m(:);

    % Decision variables, x = [w_ga; m]
    x = [   vect_w_ga;
            vect_m];
    %-------------------------------------------------------------------------
    % m0 [1 x 3*no_wells]
    m0 = SX.sym('m0', 1, 3*no_wells);

    % w_ga_last [1 x no_wells]
    w_ga_last = SX.sym('w_ga_last', 1, no_wells);

    % Parameters, p = [m0, w_ga_last]
    p = [m0, w_ga_last];
%-------------------------------------------------------------------------

% Computing the objective function, F, and the constraints, G
[F, G] = compute_both_multi_stage_NMPC(Np, Ns, dt, m0, m, w_ga, w_ga_posi_comb, d_PI_comb,
w_ga_last);

% Creating the NLP problem
NLP_prob = struct('f', F, 'x', x, 'g', G, 'p', p);
```

compute_both_multi_stage_NMPC.m

```
function [F, G] = compute_both_multi_stage_NMPC(Np, Ns, dt, m0, m, w_ga, w_ga_posi_comb,
d_PI_comb, w_ga_last)

% Importing CasADi
addpath('C:\Program Files\MATLAB\casadi-windows-matlabR2016a-v3.5.5')
import casadi.*

%-------------------------------------------------------------------------
% Finding the information needed for evaluating the objective function &
% constraints
%-------------------------------------------------------------------------
    % w_ga_reconst {Np x [Ns x w_ga]}
    w_ga_reconst = cell(Np,1);
    for k = 1:Np
        w_ga_reconst{k} = w_ga(w_ga_posi_comb(:,k),:);
    end
    %---------------------------------------------------------------------
    % Openloop simulation over the PH

        % Preallocation

            % Continuity constraints (process dynamics), G1
            % {(Np+1) x [Ns x (dv_m - pred_m)]}
            G1 = SX.sym('G1', Ns, length(m0), (Np+1))';
            G1{1} = m{1} - repelem(m0,Ns,1); % 1st continuity constraint (Ns*length(m0)
number of them)

            % w_op {(Np+1) x [Ns x w_op]}
            w_op = SX.sym('w_op', Ns, size(w_ga,2), (Np+1))';

            % w_gop {(Np+1) x [Ns x w_gop]}
            w_gop = SX.sym('w_gop', Ns, size(w_ga,2), (Np+1))';   % [w_op{1}, w_gop{1}] =
```

```
calculate_output(m{1});

            % NOTE: 1st cell of w_op & w_gop is not used. They're used to
            % make the openloop simulation loop's notation look better (k+1
            % instead of k).

        % For each time step over the prediction horizon
        for k = 1:Np

            m_kp1 = update_states(m{k}, w_ga_reconst{k}, d_PI_comb(:,:,k), dt);

            % (k+1)th continuity constraint
            G1{k+1} = m{k+1} - m_kp1;

            [w_op{k+1}, w_gop{k+1}] = calculate_output(m{k+1});

        end
    %-------------------------------------------------------------------------
    % oil_prod [Ns*Np x oil_prod]
        % Converting w_op from {(Np+1) x [Ns x w_op]} to [Ns*Np x w_op]
        w_op = vertcat(w_op{2:end});
    oil_prod = sum(w_op,2);

    % fluid_prod [Ns*Np x fluid_prod]
        % Converting w_op from {(Np+1) x [Ns x w_gop]} to [Ns*Np x w_gop]
        w_gop = vertcat(w_gop{2:end});
    fluid_prod = sum(w_gop,2);

    % delta_w_ga [Ns*Np x delta_w_ga]
    delta_w_ga = w_ga_reconst{1} - repelem(w_ga_last,Ns,1);
    delta_w_ga = [  delta_w_ga;
                    vertcat(w_ga_reconst{2:end}) - vertcat(w_ga_reconst{1:end-1})];

    % w_ga_reconst [Ns*Np x w_ga]
    w_ga_reconst = vertcat(w_ga_reconst{:});

    % sum_w_ga [Ns*Np x sum_w_ga]
    sum_w_ga = sum(w_ga_reconst,2);
%-------------------------------------------------------------------------

% Objective function, F
    Q = 1;
    R = 0.5;
    S = 50;
    w_s = ones(1,Ns); % Weights assigned to the cost of each scenario

        % Cost at each timestep, F_k = [Ns x Np]
        F_k = -Q*oil_prod.^2 + R*sum(w_ga_reconst.^2,2) + S*sum(delta_w_ga.^2,2); %[Ns*Np x
F_k]
        F_k = reshape(F_k,Ns,Np);

        % Cost of each scenario, F_s [Ns x F_s]
        F_s = sum(F_k,2);

    F = w_s*F_s;

% Inequality constraints, G2
```

```
    G2 = [     fluid_prod;
            delta_w_ga(:);
            sum_w_ga];

% Constraints, G
    % Converting G1 from {(Np+1) x [Ns x (symb_m - pred_m)]} to
    % [Ns*(Np+1) x (symb_m - pred_m)]
    G1 = vertcat(G1{:});

    G = [   G1(:);
            G2];
```

multi_stage_NMPC.m

```
% Importing CasADi
addpath('C:\Program Files\MATLAB\casadi-windows-matlabR2016a-v3.5.5')
import casadi.*

% Fetching the settings and other required data
settings_multi_stage_NMPC

% Creating the NLP problem
NLP_prob = create_NLP_prob_multi_stage_NMPC(dt, Np, Ns, Ndv, w_ga_posi_comb, d_PI_comb,
no_wells);

% Specifying NLP solver options
options = struct;
options.ipopt.max_iter = 5000;
options.ipopt.print_level = 0; % 3, 5(default), upto 12
options.print_time = false;
options.ipopt.fixed_variable_treatment = 'make_constraint';
options.ipopt.warm_start_init_point = 'yes';

% Creating the NLP solver
NLP_solver = nlpsol('NLP_solver', 'ipopt', NLP_prob, options);

% Container for arguments passed to NLP_solver
% args {lbx, ubx, lbg, ubg, x0, p, lam_x0, lam_g0}
args = struct;

% Specifying the bounds of the NLP problem

    % Bounds on the decision variables
    %       LBX     <= x                <= UBX
    %       0.323   <= w_ga          <= 11.66       [Ndv*no_wells]
    %       0       <= m             <= inf      [Ns*(Np+1)*(3*no_wells)]
    args.lbx = repelem([0.323; 0],      [Ndv*no_wells, Ns*(Np+1)*(3*no_wells)]);
    args.ubx = repelem([11.66; inf],    [Ndv*no_wells, Ns*(Np+1)*(3*no_wells)]);

    % Bounds of the constraints
    %       LBG     <= G(x, p)          <= UBG
    %       0       <= dv_m - pred_m    <= 0        [Ns*(Np+1)*(3*no_wells)]
    %       0       <= fluid_prod       <= 160      [Ns*Np]
    %       -0.15   <= delta_w_ga            <= 0.15     [Ns*Np*no_wells]
    %       0       <= sum_w_ga       <= w_gc     [Ns*Np]
    args.lbg = repelem([0; 0; -0.15; 0],    [Ns*(Np+1)*(3*no_wells), Ns*Np, Ns*Np*no_wells,
Ns*Np]);
```

```
    args.ubg = repelem([0; 160; 0.15],      [Ns*(Np+1)*(3*no_wells), Ns*Np, Ns*Np*no_wells]);

%--------------------------------------------------------------------------
% Running multi-stage NMPC
%--------------------------------------------------------------------------
    loop_length = length(t)-1;

    % Preallocating
    m               = zeros(loop_length+1,length(m0));      m(1,:) = m0;
    w_ga            = zeros(loop_length,no_wells);
    loop_time       = zeros(loop_length,1);
    controller_time = zeros(loop_length,1);

    % Initializing the ODE solver used for simulating the real plant
    update_plant = @(m0_k, w_ga_k)update_states(m0_k, w_ga_k, d_PI_plant, dt);

    % Announcing the start of simulation
    disp(' ')
    disp('Starting multi-stage NMPC...')
    disp(datetime('now'))

    elapsedTime = tic;
    % Sliding horizon strategy
    for k = 1:loop_length
        tic

        if k == 1 % lam_x0 & lam_g0 needed for warm-start not known
            args.x0 = [w_ga_ini(:); m_ini(:)];
            args.p = [m(k,:), w_ga_OL(end,:)];
            args.ubg(end+1:end+Ns*Np) = repelem(w_gc(k:k+Np-1),Ns);

            opt_time = tic;
            % Solving the NLP problem
            sol = NLP_solver(   'x0',       args.x0,...
                                'p',        args.p,...
                                'lbx',      args.lbx,...
                                'ubx',      args.ubx,...
                                'lbg',      args.lbg,...
                                'ubg',      args.ubg);
            opt_time = toc(opt_time);
        else
            args.x0 = sol.x;
            args.p = [m(k,:), w_ga(k-1,:)];
            args.ubg(end-Ns*Np+1:end) = repelem(w_gc(k:k+Np-1),Ns);
            args.lam_x0 = sol.lam_x;
            args.lam_g0 = sol.lam_g;

            opt_time = tic;
            % Solving the NLP problem
            sol = NLP_solver(   'x0',       args.x0,...
                                'p',        args.p,...
                                'lbx',      args.lbx,...
                                'ubx',      args.ubx,...
                                'lbg',      args.lbg,...
                                'ubg',      args.ubg,...
                                'lam_x0',   args.lam_x0,...
                                'lam_g0',   args.lam_g0);
```

```
            opt_time = toc(opt_time);
        end

        % Extracting the 1st optimal control move, w_ga_ast
        w_ga_ast = full(sol.x(1:no_wells))';

        % Applying the first (optimal) control move to the plant, i.e.,
        % sliding one timestep forward
        m(k+1,:) = update_plant(m(k,:), w_ga_ast);

        % Storing variables of interest
        w_ga(k,:) = w_ga_ast;
        controller_time(k) = opt_time;
        loop_time(k) = toc;
    end
    elapsedTime = toc(elapsedTime);

    % Announcing the end of simulation
    disp('Multi-stage NMPC complete!')
    fprintf(['Elapsed time = \n\n',...
           '       %g min\n\n'], round(elapsedTime/60,1));
    disp('Please wait for the plots of the results...')
%-------------------------------------------------------------------------

% Writing elapsedTime to file
fprintf(fileID, '     Elapsed time    : %g min', round(elapsedTime/60,1));
fclose(fileID);

% Preparing to plot the results

    % Removing m0 from m (Reason: m0 is not required when combining with
    % openloop simulation results)
    m = m(2:end,:);

    % Calculating the outputs
    [w_op, w_gop] = calculate_output(m);

    % Removing the + (Np-1) copies of the last w_gc value
    w_gc = w_gc(1:end-(Np-1));
```

openloop_simulation.m

```
% clear
% clc

%-------------------------------------------------------------------------
% Settings
%-------------------------------------------------------------------------
    timespan = 100; %[hr]
    dt = 20; %[s]
    %---------------------------------------------------------------------
    % Well     = [Well1    Well2]
%     d_PI_plant  = [+0.25    +0.25]; %*1e4 [kg/hr/bar]
%     d_PI_plant  = [-0.25    +0.25]; %*1e4 [kg/hr/bar]
    d_PI_plant         = [0        0];     %*1e4 [kg/hr/bar]
%     d_PI_plant  = [+0.25    -0.25]; %*1e4 [kg/hr/bar]
%     d_PI_plant  = [-0.25    -0.25]; %*1e4 [kg/hr/bar]
```

```
%     d_PI_plant  = [+0.13    +0.13]; %*1e4 [kg/hr/bar]
    %-------------------------------------------------------------------------
    % Select the type of input disturbance, w_gc
    % NOTE: 1st column of w_gc correspond to value of w_gc in Sm3/hr
    %       2nd column of w_gc correspond to ratio of step lengths
%     type = 'constant';    w_gc_val = [  40000, 1];
    % type = '1step';       w_gc_val = [  40000, 1;
    %                                     36000, 2];
    type = '2step';       w_gc_val = [  40000, 1;
                                        36000, 1;
                                        40000, 1];
    %-------------------------------------------------------------------------
    % Specify the percentage distribution of w_gc to each well
    % NOTE: Sum should be <= 100%
    % Well     = [Well1    Well2]
    w_gc_dist  = [40          40]; %[%]

%-------------------------------------------------------------------------
% Confirming the settings
%-------------------------------------------------------------------------
    if sum(w_gc_dist) > 100 % if % distribution of w_gc to each well adds up to more than
100%
        error('Percentage distribution of w_gc to each well cannot add up to more than 100%')
    end

    % Displaying the settings
    fprintf('OPENLOOP SIMULATION SETTINGS \n')
    fprintf('  dt          : %g s                  \n', dt)
    fprintf('  timespan       : %g hrs               \n', timespan)
    fprintf('  w_gc        : %s, [%s] Sm3/hr       \n', type, join(string(w_gc_val(:,1)),"
"))
    fprintf('  w_ga        : [%s]%%*w_gc           \n', join(string(round(w_gc_dist,1))," "))
    fprintf('  d_PI_plant  : [%s]*1e4 [kg/hr/bar]  \n', join(string(d_PI_plant)," "))

    % Prompting the user to confirm the settings
    disp(' ')
    disp('Please make sure the above settings are okay. Then press any key to start the
simulation OR press ctrl + c to stop execution.')
    pause

%-------------------------------------------------------------------------
% Creating other necessary data
%-------------------------------------------------------------------------
    t = 0:dt:timespan*60*60; %[s]

    % Well             = [Well1    Well2]
    m_ga       = [20636       19331]; %[kg]
    m_gt       = [1205        1174]; %[kg]
    m_ot       = [21824       19325]; %[kg]
    m0 = [m_ga, m_gt, m_ot];

    w_gc = make_w_gc(type, w_gc_val, t); %[kg/s]

    w_ga = w_gc_dist/100.*w_gc; %[kg/s]

%-------------------------------------------------------------------------
```

```
% Running openloop simulation
%--------------------------------------------------------------------------
    % Initializing the ODE solver
    update_states = @(m0_k, w_ga_k)update_states(m0_k, w_ga_k, d_PI_plant, dt);

    % Preallocation
    m = zeros(length(t),length(m0));    m(1,:) = m0;

    % For each time step within the simulation timespan
    for k = 1:length(t)-1
        m(k+1,:) = update_states(m(k,:), w_ga(k,:));
    end

    [w_op, w_gop] = calculate_output(m);

%--------------------------------------------------------------------------
% Plotting the results
%--------------------------------------------------------------------------
    timestamps = t/60/60; %[hrs]
    x_label = 'Time [hr]';
    legend_labels = {'well 1','well 2'};

    figure
    tiledlayout(2,1,'Padding','compact', 'TileSpacing','compact')
    nexttile;
    plot(timestamps,sum(w_op,2))
    title('Total oil production rate, \Sigma w_{op}^i');
    ylim('padded'); xlim('tight')
    ylims = ylim; x = diff(ylim)*0.1; ylim([ylims(1)-x, ylims(2)+x])
    ylabel('[kg/s]')
    grid on
    xlabel('time [hrs]')

    nexttile
    plot(timestamps,sum(w_gop,2));
    title('Total fluid production rate \Sigma w_{gop}^i');
    xlim([timestamps(1), timestamps(end)]);
    yline(160,'--r')
    ylabel('[kg/s]');
    xlabel('time [hrs]')
    grid on
```

openloop_plus_NMPC.m

```
clear

% Choose
%     NMPC = 'standard';
    NMPC = 'multi_stage';

%--------------------------------------------------------------------------
% Running openloop simulation
%--------------------------------------------------------------------------
    openloop_simulation

%--------------------------------------------------------------------------
% Getting confirmation from the user to start NMPC
```

```
%--------------------------------------------------------------------------
    disp(' ')
    disp('Has the openloop simulation reached steady state? Press any key to initiate MPC OR
press ctrl + c to stop execution.')
    pause
    disp(' ')
    disp('Initiating NMPC...')

%--------------------------------------------------------------------------
% Clearing the workspace for running NMPC
%--------------------------------------------------------------------------
    % Close figure from openloop simulation
    close

    % Delete unnecessary variables in the workspace
    clearvars -except NMPC t m w_op w_gop w_gc w_ga d_PI_plant dt w_gc_dist

    % Renaming variables from openloop simulation
    % NOTE: Only the last x samples are kept
    x = 45;
    t_OL            = t(end-(x+1)+1:end) - t(end-(x+1)+1);  clear t;
    m_OL            = m(end-(x+1)+1:end,:);                 clear m;
    w_op_OL         = w_op(end-(x+1)+1:end,:);              clear w_op;
    w_gop_OL          = w_gop(end-(x+1)+1:end,:);           clear w_gop;
    w_gc_OL         = w_gc(end-x+1:end,:);                  clear w_gc;
    w_ga_OL         = w_ga(end-x+1:end,:);                  clear w_ga;
    d_PI_plant_OL     = d_PI_plant;                           clear d_PI_plant;
    dt_OL           = dt;                                   clear dt;
    w_gc_dist_OL      = w_gc_dist;                            clear w_gc_dist;

%--------------------------------------------------------------------------
% Running NMPC
%--------------------------------------------------------------------------
    if strcmp('standard',NMPC)
        standard_NMPC
    elseif strcmp('multi_stage',NMPC)
        multi_stage_NMPC
    end

%--------------------------------------------------------------------------
% Combining the results from openloop simulation and NMPC
%--------------------------------------------------------------------------
    % Renaming variables from NMPC
    t_MPC           = t;            clear t;
    m_MPC           = m;            clear m;
    w_op_MPC        = w_op;         clear w_op;
    w_gop_MPC         = w_gop;              clear w_gop;
    w_gc_MPC        = w_gc;         clear w_gc;
    w_ga_MPC        = w_ga;         clear w_ga;
    dt_MPC          = dt;           clear dt;
    elapsedTime_MPC = elapsedTime;  clear elapsedTime;

    % Shifting t_MPC to start at the point where openloop simulation ended
    t_MPC   = t_MPC + t_OL(end);

    % Combining the data
    t       = [ t_OL, t_MPC(2:end)];
```

```
    m       = [ m_OL;
                m_MPC];
    w_op    = [ w_op_OL;
                w_op_MPC];
    w_gop      = [ w_gop_OL;
                w_gop_MPC];
    w_gc    = [ w_gc_OL;
                w_gc_MPC];
    w_ga    = [ w_ga_OL;
                w_ga_MPC];

%-------------------------------------------------------------------------
% Plotting the combined results
%-------------------------------------------------------------------------
    timestamps = t/60/60; %[hrs]
    legend_labels = {'well-1','well-2'};

    figure;
    tiles = tiledlayout(5,1);

%       if strcmp('standard',NMPC)
%         title(tiles, sprintf([  'dt = %g s, (d_{PI})_{plant} = %s*1e4\n',...
%                                 'Openloop simulation: w_{ga} = w_{gc}*%s%%\n'...
%                                 'NMPC: Np = %d %s, PH length = %g min, (d_{PI})_{controller} =
%s*1e4, exec.t = %g min'],...
%                                 dt_OL, mat2str(d_PI_plant),...
%                                 mat2str(round(w_gc_dist_OL,1)),...
%                                 Np, mat2str(groups), round(Np*dt_MPC/60,1),
mat2str(d_PI_controller), round(elapsedTime_MPC/60,1)))
%     elseif strcmp('multi_stage',NMPC)
%         title(tiles, sprintf(['dt = %g s, (d_{PI})_{plant} = %s*1e4\n',...
%                                 'Openloop simulation: w_{ga} = w_{gc}*%s%%\n'...
%                                 'NMPC: Nr = %d, Np = %d %s, Nd = %d, Ns = %d, Ndv = %d, PH
length = %g min, timespan = %g min, exec.t = %g min'],...
%                                 dt_OL, mat2str(d_PI_plant),...
%                                 mat2str(round(w_gc_dist_OL,1)),...
%                                 Nr, Np, mat2str([ones(1,Nr), groups]), Nd, Ns, Ndv*2,
round(Np*dt_MPC/60,1), timespan/60, round(elapsedTime_MPC/60,1)))
%     end

    ax1 = nexttile;
    plot(timestamps, sum(w_gop,2))
    yline(160,'--r')
    xline(timestamps(length(t_OL)-1),'--r')
    ylim('padded'); xlim('tight')
    ylabel('\Sigma w_{gop}^i [kg/s]')
    grid on

    ax2 = nexttile;
    plot(timestamps, sum(w_op,2))
    xline(timestamps(length(t_OL)-1),'--r')
    ylim('padded'); xlim('tight')
    ylabel('\Sigma w_{op}^i [kg/s]')
    grid on

    ax3 = nexttile;
    plot(timestamps, w_op)
```

```
    xline(timestamps(length(t_OL)-1),'--r')
    legend(legend_labels, 'Location','eastoutside')
    ylim('padded'); xlim('tight')
    ylabel('w_{op}^i [kg/s]')
    grid on

    ax4 = nexttile;
    plot(timestamps(1:end-1), w_ga)
    xline(timestamps(length(t_OL)-1),'--r')
    legend(legend_labels, 'Location','eastoutside')
    ylim('padded'); xlim('tight')
    ylabel('w_{ga}^i [kg/s]')
    grid on

    ax5 = nexttile;
    plot(   timestamps(1:end-1), round(w_gc,4),...
            timestamps(1:end-1), round(sum(w_ga,2),4))
    xline(timestamps(length(t_OL)-1),'--r')
    legend({'w_{gc}', '\Sigma w_{ga}^i'}, 'Location','eastoutside')
    ylim('padded'); xlim('tight')
    ylabel('w_{gc}, \Sigma w_{ga}^i [kg/s]')
    grid on
    xlabel('time [hrs]')

    linkaxes([ax1 ax2 ax3 ax4 ax5],'x')

%-------------------------------------------------------------------------
% Saving the results
%-------------------------------------------------------------------------
%     if strcmp('standard',NMPC)
%         filename = sprintf('/dt = %g s, Np = %d %s, PH length = %g min, d_PI_plant = %s,
d_PI_controller = %s, exec.t = %g min',...
%                            dt_MPC, Np, mat2str(groups), round(Np*dt_MPC/60,1),
mat2str(d_PI_plant), mat2str(d_PI_controller), round(elapsedTime_MPC/60,1));
%     elseif strcmp('multi_stage',NMPC)
%         filename = sprintf('/dt = %g s, Nr = %d, Np = %d %s, Nd = %d, Ns = %d, Ndv = %d, PH
length = %g min, d_PI_plant = %s, exec.t = %g min',...
%                            dt_MPC, Nr, Np, mat2str([ones(1,Nr), groups]), Nd, Ns, Ndv*2,
round(Np*dt_MPC/60,1), mat2str(d_PI_plant_OL), round(elapsedTime_MPC/60,1));
%     end
%
%     save(append('saved_results',filename,'.mat'))
%     savefig(gcf, append('saved_plots',filename,'.fig'))
%     exportgraphics(gcf, append('saved_plots',filename,'.emf'))
%
movefile('saved_settings/my_settings_file.txt',append('saved_settings',filename,'.txt'),'f');
```

University of South-Eastern Norway

Faculty of Technology, Natural Sciences and Maritime Sciences, Campus Porsgrunn

FMH606 Master's Thesis

Title: Optimal operation of processes under uncertainty using stochastic model predictive control

USN supervisor: Associate Professor Roshan Sharma (USN)

External partner: Equinor, Skagerak Energi and SINTEF

Task background:
This thesis is related to two on-going projects at USN. The first project is the "Digiwell" project in collaboration with Equinor and SINTEF (and other partners like Imperial College London and MIT) where 5 (or possibly 6) PhD students will be involved. Some of these PhD positions will be announced in the near future at USN. The second project is the "stochastic MPC" project in collaboration with Skagerak Energi.

Traditional Model Predictive Controllers (MPC) are based on deterministic model with an assumption that the model is perfect, and that all the parameters of the model and disturbances are perfectly known. In such a case, i.e. in the absence of plant-model mismatch, the control action is always perfect and all the constraints are perfectly satisfied. In a traditional MPC, an open-loop optimization problem is formed and solved, and the solution of this open loop optimization problem is implemented in a closed loop fashion using the sliding horizon strategy.

However, in real world, the assumption that the model is perfect simply becomes too good to be true. Real world industrial processes are subjected to different kinds of uncertainty, from uncertain values of some parameters in the model, to uncertain disturbances acting on the real process. Under the presence of uncertainties, the solution of a deterministic MPC will not be enough, and some of the constraints might not be satisfied (or under satisfied) leading to non- optimal operation of the plant. For example: some of the equipment in the plant might not be used to its full capacity, or in contrary, some of the equipment might be operated above/below its prescribed range or limit. Under such cases, in the real world, the operators of the plant might simply choose not to use the control sequence generated by the deterministic MPC in the first place.

The aim of this thesis is to take into account the uncertainties present in a process/plant and develop an MPC that can handle it. An example is the robust MPC or stochastic MPC. In particular, it is of interest to look into non-conservative robust/stochastic MPC. Real world industrial case studies for applying robust/stochastic MPC can involve both oil production process (Equinor as partner) and/or the hydropower production process (Skagerak Energi as partner). The mathematical models of these two case studies (together with full process description) will be provided to the students by the supervisor. The students can also freely choose other case studies on their own.

Task description:
The following are the main tasks:

(i) Thorough literature review on different methods for Robust/Stochastic MPC.

(ii) Develop a min-max robust MPC for at least one of the real industrial case studies (either a Gas lift optimization problem during oil production, or, hydropower production). The mathematical models for these two case studies together with their detailed description will be provided to the student.

(iii) Develop a multi-stage MPC (or scenario based MPC) for the same case study as chosen in (ii).

(iv) Compare the simulation results between the control structures from points (ii) and (iii). State the pros and cons of these two methods with the help of the simulation results.

(v) If time permits, the student can also look into other methods for robust/stochastic MPC (like the ones based on multi-objective optimization methods) and apply them to at least one of the case studies.

(vi) Document the work in a report. The report should be technically sound. Presentation of the work.

Student category: IIA students

The task is suitable for online students (not present at the campus): Yes

Practical arrangements: N/A

Supervision:
As a general rule, the student is entitled to 15-20 hours of supervision. This includes necessary time for the supervisor to prepare for supervision meetings (reading material to be discussed, etc).

Signatures:

Supervisor (date and signature): 01.02.2021 

Student (write clearly in all capitalized letters): KUSHILA RASANDUNI JAYAMANNE

Student (date and signature): 18.01.2021