University of South-Eastern Norway

www.usn.no

FMH606 Master's Thesis 2022

Master of Science, Industrial IT and Automation

Robust MPC for optimal oil production under the presence of uncertainties

Ragnvald Leyser Gulløy

Faculty of Technology, Natural sciences and Maritime Sciences
Campus Porsgrunn

University of South-Eastern Norway

www.usn.no

Course: FMH606 Master's Thesis, 2022

Title: Robust MPC for optimal oil production under the presence of uncertainties

Number of pages: 58

Keywords: Deterministic linear MPC, Multi-stage linear MPC, Optimization, Linearization

Student: Ragnvald Leyser Gulløy

Supervisor: Roshan Sharma

External partner: Equinor

Summary:

Industries, business and private people often want to maximize income. The petroleum industry is no exception. Higher production leads to higher income. Model predictive control can help maximizing the production at oil field.

The objective is to design a robust multi-stage linear MPC for oil production optimization such that it is robust to uncertainties present in the system. The conservativeness and robustness of the developed robust MPC must be studied through detailed simulations.

To design a linear MPC it is necessary to have a linear model, which is derived of a nonlinear model of a gas lifted oil field. The robust multi-stage linear MPC is designed stepwise from a deterministic MPC to a robust MPC.

It is designed two variants of the robust multi-stage linear MPC. One with only hard constraint at w_s^{max} and one with a soft constraint "w_s^{max} – soft" below the w_s^{max}. The first MPC works, but under some circumstances it is infeasible. Bounds can be violated, and the inputs can oscillate. Therefore, the soft constraint is introduced. Then the simulation is feasible at all times, and constraints and bounds are satisfied. Introducing the soft constraint makes the MPC slightly more conservative. The robustness is taken care of since the hard constraint w_s^{max} is still operative.

The robust multi-stage linear MPC with an extra soft constraint below w_s^{max} using the IPOPT solver is a robust and efficient MPC.

Preface

This thesis is written to fulfill the Master of Science – Industrial IT and Automation online program at the University of South-Eastern Norway. It is written on part-time at spring and autumn semesters in 2022.

Special thanks to Associate Professor Roshan Sharma and Ph.D. Researcher Nima Janatianghadikolaei at the Department of Electrical engineering, IT and Cybernetics at USN for good guidance and support.

This thesis is related to an on-going project “Digiwell” in collaboration with Equinor (and other partners like SINTEF, Imperial College London and MIT).

Software used in this thesis:

- CasADi
- MATLAB R2022a
- Microsoft Excel
- Microsoft Visio
- Microsoft Word

Reader should have knowledge about programming in MATLAB.

Porsgrunn, 27.10.2022

Ragnvald Leyser Gulløy

Contents

Preface 3
Contents 4
Nomenclature 5
1 Introduction 6
1.1 Structure of the thesis 7
2 Literature review 8
2.1 Model predictive control 8
2.1.1 Robust multi-stage MPC *8*
2.2 Gas lifted oil field 10
2.3 CasADi framework 11
3 Developing linear nominal model 12
3.1 Nonlinear model of a gas lifted oil field 12
3.1.1 Open loop simulation of nonlinear model *16*
3.2 Linearization of nonlinear model 18
3.3 Comparing the nonlinear model and the linear model 19
3.3.1 Open loop simulation – Not updating linear model *19*
3.3.2 Open loop simulation – Updating linear model at every iteration *21*
4 Design of deterministic linear MPC 24
4.1 Formulation of deterministic linear MPC 24
4.2 Low-level interface 25
4.2.1 Closed loop simulation in low-level interface *28*
4.3 High-level interface 30
4.3.1 Closed loop simulation in high-level interface *30*
4.3.2 Grouping control input *31*
4.3.3 Comparing optimizing solvers *33*
4.4 Stochastic analysis of deterministic linear MPC 37
5 Design of robust linear MPC 40
5.1 Multi-stage MPC 40
5.1.1 Choosing scenarios *42*
5.2 Simulation results 43
5.2.1 Comparing robust MPC with deterministic MPC *44*
5.2.2 Simulation with short prediction horizon *45*
5.2.3 Robust MPC with soft constraint *49*
5.3 Robustness and conservativeness 51
6 Discussion 53
7 Conclusion 54
References 55
Appendices 57

Nomenclature

CPU	–	Central Processing Unit
GB	–	Gigabyte
HP	–	Hewlett-Packard
IT	–	Information Technology
LHS	–	Left Hand Side
MIMO	–	Multiple-Input and Multiple-Output
MIT	–	Massachusetts Institute of Technology
MPC	–	Model Predictive Control
NMPC	–	Nonlinear Model Predictive Control
QP	–	Quadratic Programming
RAM	–	Random Access Memory
RHS	–	Right Hand Side
USN	–	University of South-Eastern Norway

1 Introduction

Industries, business and private people often want to maximize income. The petroleum industry is no exception. Higher production leads to higher income. Model predictive control can help maximizing the production at oil field. Its advantage of MIMO systems and the ability to deal with constraints, makes it to a popular technique. Drawbacks of the traditional MPC is that it is very sensible for model uncertainty, and the computational time can be too high for real-time systems. To deal with model uncertainty, robust MPC is introduced, though it is normally even more computational time demanding.

The objective is to design a robust multi-stage linear MPC for oil production optimization such that it is robust to uncertainties present in the system. The conservativeness and robustness of the developed robust MPC must be studied through detailed simulations.

The project topic description can be read in Appendix A. The given model of a gas lifted oil field can be read in Appendix B.

The robust multi-stage linear MPC is designed stepwise. To design a linear MPC it is necessary to have a linear model, which is derived of the nonlinear model of the gas lifted oil field. It is designed a deterministic MPC, which is applied to a nominal model. This deterministic linear MPC is further developed to a robust multi-stage linear MPC. It is important to design the MPC as efficient as possible. If the MPC is implemented in real-time systems, it must be fast enough to run in real-time. Therefore, it is also focus on optimizing time.

It is not found literature that describe the method of building the scenarios in the multi-stage linear MPC, only for NMPC. Then the technique of building scenarios in NMPC is adopted and fitted to linear MPC.

Scope of the thesis is to develop the robust multi-stage linear MPC. The given model of the gas lifted oil field is used as it is. Constraints and bounds are fictive values chosen to test the MPC properly.

Assumptions taken in this thesis:

- Full state information of the model
- Uncertainties do not change in time

MATLAB codes are present during the report. The stepwise approach also gives stepwise progress in codes. Then some codes are custom made for some chapters and some codes are used throughout the report.

All simulations in this thesis are performed with a HP Spectre x360 laptop, 16 GB RAM and Intel Core i7 10510U processor.

1.1 Structure of the thesis

Chapter 2 presents a literature review of the most relevant subjects in this thesis.

Chapter 3 presents a nonlinear model of a gas lifted oil field and how it can be converted to a linear model. The models are compared in open-loop simulations.

Chapter 4 shows how to design deterministic linear MPC. It is designed in two different ways, using low-level interface and high-level interface in CasADi. The deterministic MPC is applied to uncertain models. Results of closed-loop simulations are present.

Chapter 5 shows how to design robust multi-stage linear MPC. It is shown detailed closed loop simulations of different operational scenarios. The conservativeness and robustness are studied.

Chapter 6 discuss the results.

Chapter 7 presents the conclusion and further work.

2 Literature review

This chapter shows literature review of the main, subjects which are model predictive control, gas lifted oil field and a short brief of CasADi.

2.1 Model predictive control

MPC is a popular control technique, because of its ability to deal with constraints and MIMO systems [1]. State feedback MPC, which is used in this thesis, assumes full state knowledge. The MPC solves an optimization problem at each time step, based on a mathematical model that represent the real process. The optimization problem contains a prediction horizon. This means that MPC optimize control inputs throughout a prediction horizon ahead of time. If the prediction horizon is 20 steps in future, the MPC calculates control inputs for 20 steps. Only the first control inputs are used and all other inputs in the prediction horizon are discarded. In this way the MPC can detect disturbance in future and take early action. When moving to next time step, the MPC solves the optimization problem again. This is sliding horizon strategy. An optimal controller with sliding horizon strategy is a MPC [2].

2.1.1 Robust multi-stage MPC

To ensure that the MPC do not violate constraints, the model used in the MPC must have no model uncertainties. But in the real world there are always model uncertainties [2]. A robust multi-stage MPC can deal with uncertainties. It is robust because it does not violate the constraint when having uncertainties. The multi-stage or also called multi-scenario MPC, branches the problem into several scenarios. The scenarios represent the effect of an unknown uncertain influence together with the optimal control input. All scenarios from the same node uses the same control input. The constraints in the MPC must be satisfied for all scenarios [3].

Literature recommends that scenarios have all possible combinations of the extreme values of the parameters [3]. Figure 2.1 shows a sketch of the branching of the multi-stage MPC of one uncertain parameter with three instances. The optimization problem grows exponentially with prediction horizon. The optimization problem can grow so much that it can be impossible to solve, or the computational cost can be too high. This is the drawback of the multi-stage MPC.

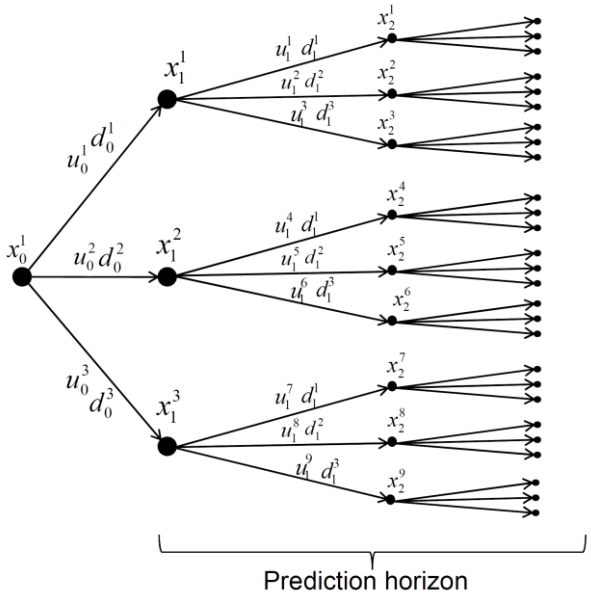

Figure 2.1: Robust multi-stage NMPC [3].

In general, the multi-stage MPC can be formulated as shown in equations (2.1) - (2.5) [3]. Where N_s is number of scenarios, ω is the probability for each scenario.

$$\min_{u_k^j} \quad \sum_{i=1}^{N_S} \omega^j J_i(X_i, U_i) \tag{2.1}$$

Subject to:

$$x_{k+1}^j = f\left(x_k^{p(j)}, u_k^j, d_k^{r(j)}\right) \tag{2.2}$$

$$g\left(x_{k+1}^j, u_k^j\right) \leq 0 \tag{2.3}$$

$$u_k^j = u_k^l \; if \; x_k^{p(j)} = x_k^{p(l)} \tag{2.4}$$

Cost of each scenario:

$$J_i(X_i, U_i) = \sum_{k=0}^{N-1} L\left(x_{k+1}^j, u_k^j\right) \tag{2.5}$$

Equation (2.2) represents the state trajectory. Equation (2.3) represents the inequality constraints. Equation (2.4) represents the non-anticipative constraint that ensure equal inputs when sharing parent node. Where l represents another branch and p represents parent node.

To deal with the growing optimization problem, robust horizon is introduced. The problem will only grow within the robust horizon, after the robust horizon the branching is maintain static. Figure 2.2 shows a sketch of the branching of robust multistage MPC with robust horizon. Then the exponential growing optimization problem is taken care of.

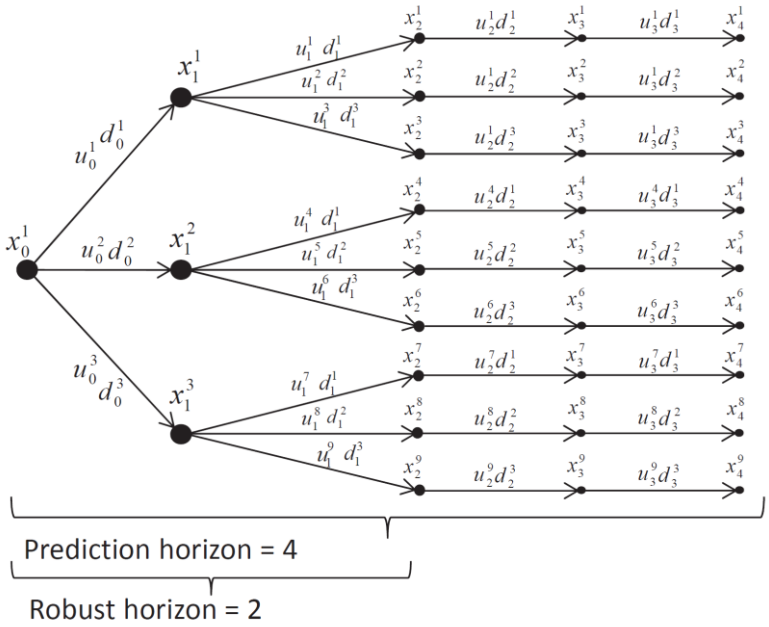

Figure 2.2: Sketch of robust multi-stage NMPC with robust horizon [3].

Number of scenarios are calculated with equation (2.6) [4]. Where N_s is number of scenarios, n_p is number of uncertain parameters, v_i is number of instances for each parameter and N_r is the robust horizon. Substituting numbers from Figure 2.2 into the equation, $N_r = 2$, $n_p = 1$, $v_i = 3$ gives $N_s = 9$.

$$N_s = \left(\prod_{i=1}^{n_p} v_i \right)^{N_r} \tag{2.6}$$

2.2 Gas lifted oil field

A gas lifted oil field contains of several gas lifted oil wells. The wells share a common source of injection gas and produce to an inlet separator. When a reservoir has produced for a while, the reservoir pressure decreases and the oil flow to the production platform will decrease or stop. To increase the production or to prevent the wells from dying, this technique is helpful. Gas lifted oil wells reduce the density in the tube to reduce the hydrostatic pressure. Gas is injected into the tube from annulus at a proper depth. The injection gas mixes with the liquid and gas from the reservoir in the tube. The density in the tube will decrease. Consequently, the mixture of liquid and gas will flow easier to the production platform [5].

The amount of injection gas is not unlimited. An injection compressor has its limitations. The wells must share the capacity of injection gas. None of the wells are equal and the perfect amount of injection gas will vary from well to well. Higher gas injection doesn't always mean higher oil production [6]. In addition, the capacity of the inlet separator must be considered. Overloading the separator can result in a disturbance in the process or shutdown. Figure 2.3 shows a sketch of a single gas lifted oil well. w_{ga} is the mass flow rate of injection gas at the gas lift choke valve, w_{ginj} is the mass flow rate of gas injected into tube, w_{gr} and w_{gl} is the mass

flow rate, from reservoir, of gas and liquid respectively. w_{glp} is the produced mass flow rate of gas and liquid.

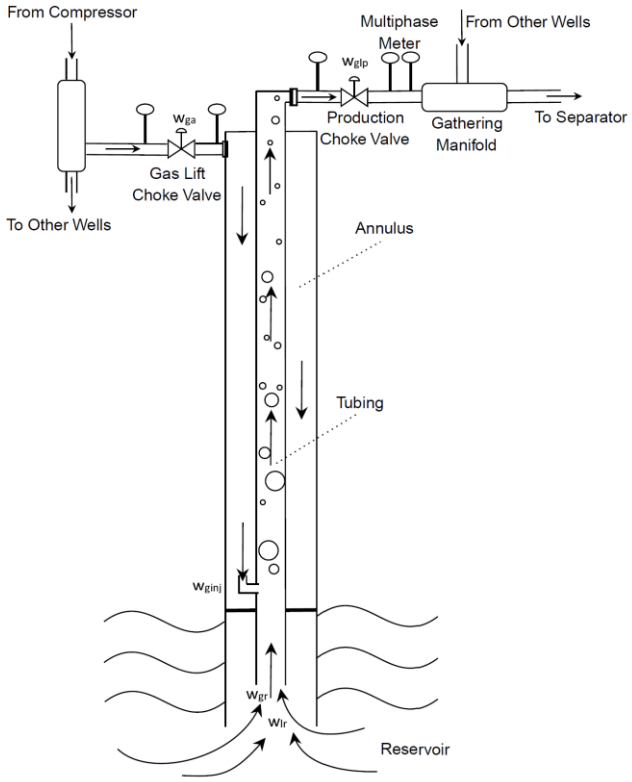

Figure 2.3: Schematic diagram of a single gas lifted oil well [5].

2.3 CasADi framework

CasADi is an open-source tool for nonlinear optimization and algorithmic differentiation. It is designed for building efficient optimal control software, with minimal effort. [7]. It can be used in nonlinear programming and in quadratic programming. CasADi can be used via full-featured interfaces to Python, MATLAB or Octave [8]. The CasADi framework is used as a part of the MPC to optimize the problem. CasADi is used instead of inbuilt optimization tools in MATLAB to improve the computation performance [3].

3 Developing linear nominal model

To design a linear MPC it is necessary to develop a linear model. In this chapter the nonlinear model is present, and it is shown how the linear model is derived of the nonlinear model. Both models are tested and compared in open loop simulations.

3.1 Nonlinear model of a gas lifted oil field

The model used in this thesis is a model of a gas lifted oil field with three oil wells [5]. The given model is attached as Appendix B. This model is made for five wells, but this thesis only evaluates three wells. The principle is the same but the problem to solve is smaller.

The model is described by three differential equations, equation (3.1) - (3.3), and 19 algebraic equations, equation (3.4) - (3.22). The different wells in the gas lifted oil field are marked as "i".

Differential equations:

i = 1, 2, 3

$$\dot{m}_{ga}^{i} = w_{ga}^{i} - w_{ginj}^{i} \tag{3.1}$$

$$\dot{m}_{gt}^{i} = w_{ginj}^{i} + w_{gr}^{i} - w_{gp}^{i} \tag{3.2}$$

$$\dot{m}_{lt}^{i} = w_{lr}^{i} - w_{lp}^{i} \tag{3.3}$$

Algebraic equations:

i = 1, 2, 3

$$P_{a}^{i} = \frac{Zm_{ga}^{i}RT_{a}^{i}}{MA_{a}^{i}L_{a.tl}^{i}} \tag{3.4}$$

$$P_{ainj}^{i} = P_{a}^{i} + \frac{m_{ga}^{i}}{A_{a}^{i}L_{a.tl}^{i}} + gL_{a.vl}^{i} \tag{3.5}$$

$$\rho_{l}^{i} = \rho_{w}^{i}WC^{i} + \rho_{o}^{i}(1 - WC^{i}) \tag{3.6}$$

$$V_{G}^{i} = A_{t}^{i}L_{t.tl}^{i} - \frac{m_{lt}^{i}}{\rho_{l}^{i}} \tag{3.7}$$

$$\rho_{m}^{i} = \frac{m_{gt}^{i} + m_{lt}^{i}}{A_{t}^{i}L_{t.tl}^{i}} \tag{3.8}$$

$$P_{tinj}^{i} = \frac{Zm_{gt}^{i}RT_{t}^{i}}{MV_{G}^{i}} + \frac{1}{2}\rho_{m}^{i}gL_{t.vl}^{i} \tag{3.9}$$

$$Y_2^i = 1 - \alpha_Y \left(\frac{P_{ainj}^i - P_{tinj}^i}{max\left(P_{ainj}^i, P_{ainj}^{min}\right)} \right) \tag{3.10}$$

$$\rho_{ga}^i = \frac{m_{ga}^i}{A_a^i L_{a.tl}^i} \tag{3.11}$$

$$w_{ginj}^i = k^i Y_2^i \sqrt{\rho_{ga}^i max\left(P_{ainj}^i - P_{tinj}^i, 0\right)} \tag{3.12}$$

$$P_{wf}^i = P_{tinj}^i + \rho_l^i g L_{r.vl}^i \tag{3.13}$$

$$w_{lr}^i = PI^i max\left(P_r^i - P_{wf}^i, 0\right) \tag{3.14}$$

$$w_{or}^i = \frac{\rho_o^i}{\rho_l^i}\left(1 - wc^i\right) w_{lr}^i \tag{3.15}$$

$$w_{gr}^i = GOR^i w_{or}^i \tag{3.16}$$

$$P_{wh}^i = \frac{Z m_{gt}^i R T_t^i}{M V_G^i} - \frac{1}{2}\rho_m^i g L_{t.vl}^i \tag{3.17}$$

$$Y_3^i = 1 - \alpha_Y \left(\frac{P_{wh}^i - P_s}{max\left(P_{wh}^i, P_{wh}^{min}\right)} \right) \tag{3.18}$$

$$w_{glp}^i = C_v Y_3^i \sqrt{\rho_m^i max\left(P_{wh}^i - P_s, 0\right)} \tag{3.19}$$

$$w_{gp}^i = \frac{m_{gt}^i}{m_{gt}^i + m_{lt}^i} w_{glp}^i \tag{3.20}$$

$$w_{lp}^i = \frac{m_{lt}^i}{m_{gt}^i + m_{lt}^i} w_{glp}^i \tag{3.21}$$

$$w_{op}^i = \frac{\rho_o^i}{\rho_l^i}\left(1 - wc^i\right) w_{lp}^i \tag{3.22}$$

The model is not described in detail, but the symbols and the constants values are present. For detail, see [5]. All symbols of the model of the gas lifted oil field are present in following tables:

- Table 3.1 show the states.
- Table 3.2 show the inputs.
- Table 3.3 show the variables.
- Table 3.4 show the constants.
- Table 3.5 show the values of the constants.

Table 3.1: States, i = 1, 2, 3.

Symbol	Description	Unit
m_{ga}^{i}	Mass of the gas in the annulus	kg
m_{gt}^{i}	Mass of the gas in the tubing above the injection point	kg
m_{lt}^{i}	Mass of the liquid in the tubing above the injection point	kg

Table 3.2: Inputs, i = 1, 2, 3.

Symbol	Description	Unit
w_{ga}^{i}	Mass flow rate of the injected lift gas into the annulus	kg/m^3

Table 3.3: Variables, i = 1, 2, 3.

Symbol	Description	Unit
P_{a}^{i}	Pressure of gas in the annulus downstream the lift gas choke	bar
P_{ainj}^{i}	Pressure of gas in the annulus upstream the gas injection valve	bar
P_{tinj}^{i}	Pressure in the tubing downstream the gas injection valve	bar
P_{wf}^{i}	The bottom hole pressure or well flow pressure	bar
P_{wh}^{i}	Pressure in the tubing upstream the production choke valve. Well head pressure	bar
V_{G}^{i}	The volume of gas present in the tubing above the gas injection point	m^3
w_{ginj}^{i}	The mass flow rate of the gas injected into the tubing from annulus	kg/s
w_{glp}^{i}	The mass flow rate of the mixture of gas and liquid from the production choke valve	kg/s
w_{gp}^{i}	Mass flow rate of gas through the production choke valve	kg/s
w_{gr}^{i}	Mass flow rate of gas from the reservoir	kg/s
w_{lp}^{i}	Mass flow rate of liquid through the production choke valve	kg/s
w_{lr}^{i}	Mass flow rate of liquid from the reservoir	kg/s
w_{op}^{i}	Oil compartment mass flow rate from liquid product considering water cut	kg/s
w_{or}^{i}	Oil compartment mass flow rate from liquid comes out from reservoir	kg/s
Y_{2}^{i}	Gas expansion factor in the gas injection valve	-
Y_{3}^{i}	Gas expansion factor in the production choke valve	-
ρ_{ga}^{i}	Density of gas in annulus	kg/m^3

ρ_l^i	Density of the liquid based on the water cut	kg/m^3
ρ_m^i	The average density of the mixture of liquid and gas in the tubing above the injection point	kg/m^3

Table 3.4: Constants, i = 1, 2, 3.

Symbol	Description	Unit
A_a^i	Annulus cross section area	m^2
A_t^i	Tubing cross section area	m^2
C_v	Valve characteristics as a function of its opening (this valve is always 100 % open)	-
g	Gravitational acceleration constant	m/s^2
GOR^i	Gas to oil ratio	-
k^i	Gas injection valve constant	$\sqrt{(kg \cdot m^3/bar)/h}$
$L_{a.tl}^i$	Total length of the annulus above the injection point	m
$L_{a.vl}^i$	Vertical length of the annulus above the injection point	m
$L_{r.vl}^i$	Vertical length of the tubing below the injection point	m
$L_{t.tl}^i$	Total length of the tubing above the injection point	m
$L_{t.vl}^i$	Vertical length of the tubing above the injection point	m
M	Molar mass	kg/mol
P_r^i	Pressure of the reservoir	bar
P_s	Separator pressure	bar
PI^i	Productivity index	kg/h/bar
R	Gas constant	J/K/mol
T_a^i	Temperature in the annulus	K
T_t^i	Temperature in the tubing	K
WC^i	Water cut	-
Z	Compressibility factor	-
α_Y	Constant	-
ρ_o^i	Density of oil	kg/m^3
ρ_w	Density of water	kg/m^3

Table 3.5: Values of constants.

Symbol	Well 1, i = 1	Well 2, i = 2	Well 3, i = 3
A_a^i	0,0174	0,0174	0,0174
A_t^i	0,0194	0,0194	0,0194
C_v	8190		
g	9,80665		
GOR^i	0,05	0,07	0,03
k^i	68,43	67,82	67,82
$L_{a.tl}^i$	2758	2559	2677
$L_{a.vl}^i$	2271	2344	1863
$L_{r.vl}^i$	114	67	61
$L_{t.tl}^i$	2758	2559	2677
$L_{t.vl}^i$	2271	2344	1863
M	$20 \cdot 10^{-3}$		
P_r^i	150	150	150
P_s	30		
PI^i	$2{,}51 \cdot 10^4$	$1{,}63 \cdot 10^4$	$1{,}62 \cdot 10^4$
R	8.31446261815324		
T_a^i	280	280	280
T_t^i	280	280	280
WC^i	0,20	0,10	0,25
Z	1,3		
α_Y	0,66		
ρ_o^i	800	800	800
ρ_w	1000		

3.1.1 Open loop simulation of nonlinear model

Figure 3.1 shows the open loop simulation of the nonlinear model. The chosen outputs are the bottom hole pressure P_{wf}, the well head pressure P_{wh} and the liquid produced w_{lp}. The simulation shows that when decreasing the inputs injected lift gas w_{ga}, the produced liquid is decreasing. The bottom hole pressure is increasing, and the well head pressure is decreasing while the injected gas is decreasing.

This makes sense because when injecting gas, the density of the mixture of gas and liquid will decrease. Then the weight of the liquid column will decrease, this results in lower bottom hole pressure and more liquid produced.

To perform the simulations the following MATLAB code are developed:

- Appendix C shows the MATLAB main code for the open loop simulation.
- Appendix D shows the MATLAB function of the nonlinear model.
- Appendix E shows the MATLAB function for generating inputs.
- Appendix F shows the MATLAB function for updating states using the Runge-Kutta 4th method.

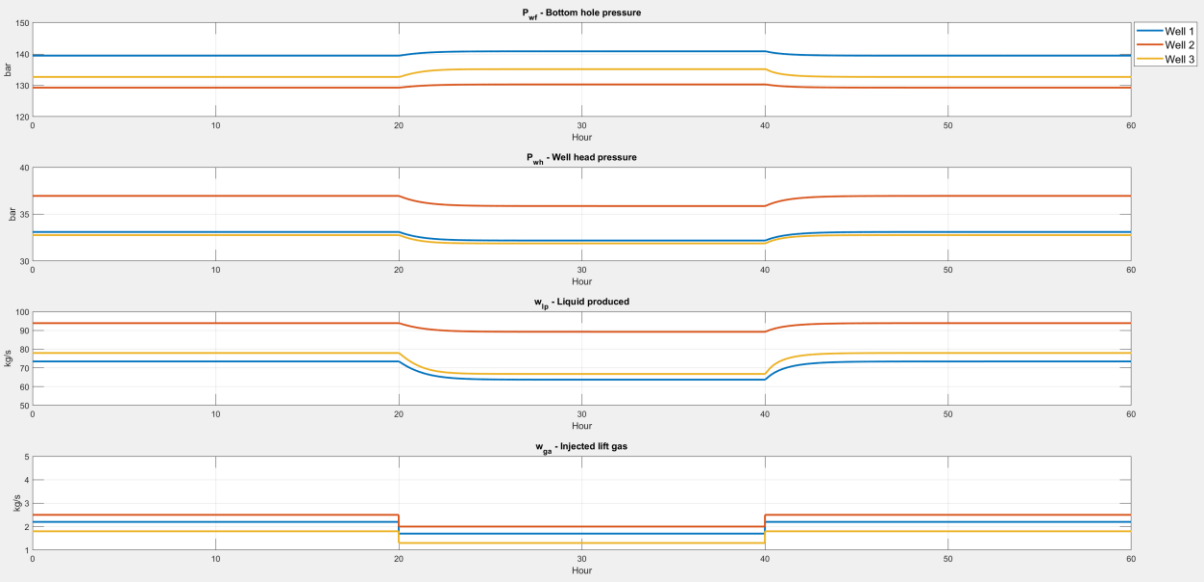

Figure 3.1: Open loop simulation of the nonlinear model.

The time constant of the system is relevant when designing the MPC. A rule of thumb is to have the prediction horizon at least equal to the time constant [2].

The time constant T is founded analytically after the time constant value is calculated according to equation (3.23). Here y_1 is before step change about 39 h and y_2 is after step change about 48 h and the time constant value T_{value} is about 41 h. The time constants are then founded analytically in Figure 3.2. The time constant for the three wells are T = [1.09, 1.16, 0.89] h.

$$T_{value} = (y_2 - y_1) * 0{,}63 + y_1 \tag{3.23}$$

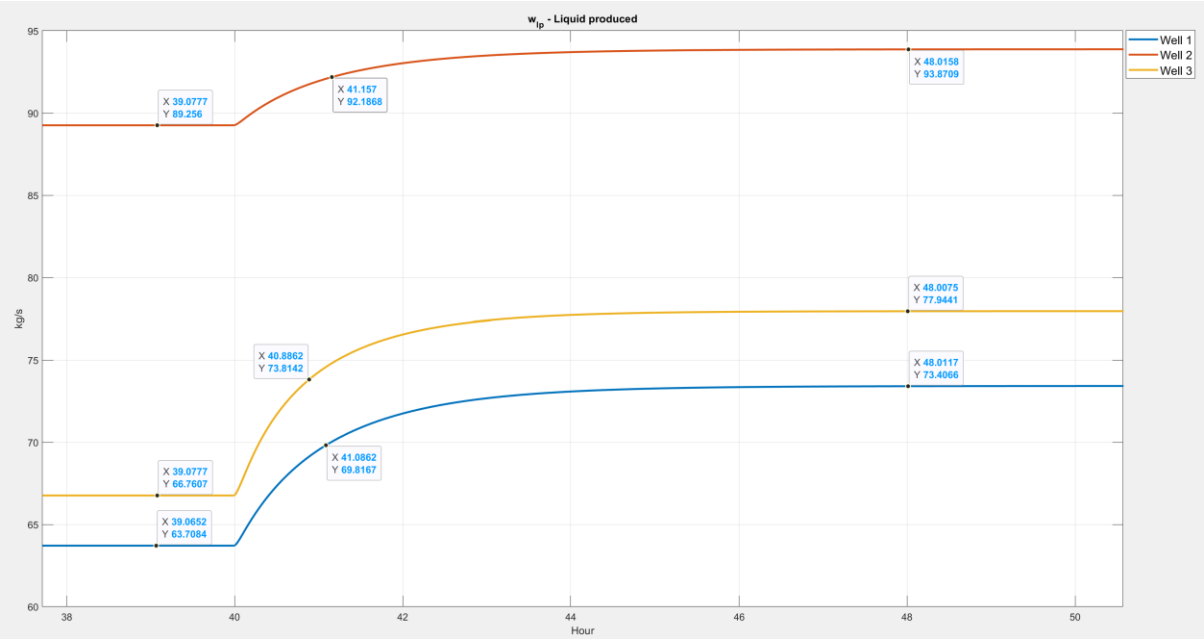

Figure 3.2: Finding time constant when w_{ga} stepping from [1.7, 2.0, 1.3] kg/s to [2.2, 2.5, 1.8] kg/s.

3.2 Linearization of nonlinear model

The nonlinear model is too complex to linearize by hand. Therefore, the Symbolic toolbox in MATLAB [9] is used to linearize the model. The wanted form of the linear model is a discrete time linear state space model in deviation form [2] shown in equation (3.24) and (3.25).

$$\delta x_{k+1} = A_d \delta x_k + B_d \delta u_k \tag{3.24}$$

$$\delta y_k = C_d \delta x_k \tag{3.25}$$

To find the real values of x_k, u_k and y_k the operating points x_{op}, u_{op} and y_{op}, where the model is linearized around, must be added as shown in equations (3.26) - (3.28).

$$x_k = \delta x_k + x_{op} \tag{3.26}$$

$$u_k = \delta u_k + u_{op} \tag{3.27}$$

$$y_k = \delta y_k + y_{op} \tag{3.28}$$

The nonlinear model is simplified to let the Symbolic toolbox in MATLAB linearize the model. The toolbox has trouble with the "max function" in the nonlinear model, e.g. equation (3.10). If the symbolic toolbox must consider negative constants, the Jacobian matrices will be unnecessarily complicated. Simulation performed later in section 3.3.1 and 3.3.2, shows that the simplifications don't affect the results.

Assumptions made to the nonlinear model before linearizing symbolically:

- All constants are positive. Which is realistic since all values are positive according to Table 3.5.

- The terms to prevent zero or negative values in equations (3.10), (3.12), (3.14), (3.18) and (3.19) are removed and instead assumed to be positive.

The MATLAB function in Appendix G, shows the code for linearizing the model symbolically. The linearization is done as following:

- First all variables and constants are defined as symbols.
- Assumptions are taken.
- The simplified nonlinear model is defined.
- Symbolic Jacobian matrices are made for system matrices A, B and C with the MATLAB function Jacobian [10].
- Constants are substituted to the system matrices.
- Vectors with outputs and inputs operating points are defined.
- Constants are substituted to the operating point vectors.
- System matrices and operating point vectors are converted to MATLAB functions before returning. The states are the inputs arguments to the functions. It is more efficient to use function handler than substitute into the symbolic expressions.

The system matrices A, B and C and the operating point vectors cannot be used as they are at this point. They are only functions, with states as input arguments. The system matrices are in continuous time and must be converted to discrete time. Appendix H shows an example of the system matrices in continues time. It shows the symbolic Jacobian matrices where the states are already substituted and the real value matrices.

Normally the control input is an input argument together with the states when linearizing, but in this case the system is only dependent on the states. It is because the only equation that contains the control input in the nonlinear model is the differential equation (3.1). When building the Jacobian matrices, the control input w_{ga} disappears since it is its own term.

The MATLAB function in Appendix I, updating the system matrices and operating points vectors with states and converting from continuous time to discrete time. Then system matrices and operating points can be updated whenever needed.

3.3 Comparing the nonlinear model and the linear model

First the system matrices and operating points of the linear model are updated ones to see how it acts against the nonlinear model. Then the same simulation is done again, but instead updating the system matrices and operating points at every iteration.

The time step in the simulations is chosen to be 15 s. Longer time step will lose information in both models. Shorter time step will not improve the result, but be more computation demanding. The time step, dt = 15 s, is used throughout the report.

3.3.1 Open loop simulation – Not updating linear model

The system matrices and the operating points are updated before the open loop simulation starts. The operating points are defined when the input w_{ga} to the nonlinear model is [2.2, 2.5, 1.8] kg/s. A, B and C matrices and operating points are static throughout the simulation. Figure 3.3 shows the open loop simulation where the linear model is compared to the nonlinear model. The unbroken lines represent the nonlinear model, and the dotted lines

represent the linear model. It is the same simulation done when simulating only nonlinear model in section 3.1.1. Appendix J shows the code for the simulation. Appendix K shows the function for updating the states of the linear model.

The linear model seems to act like the nonlinear model, but there is a deviation from the nonlinear model. The deviation is larger when far away from the operating point. Figure 3.4 shows the w_{op}, oil produced. In this figure the deviation between the models is more distinct.

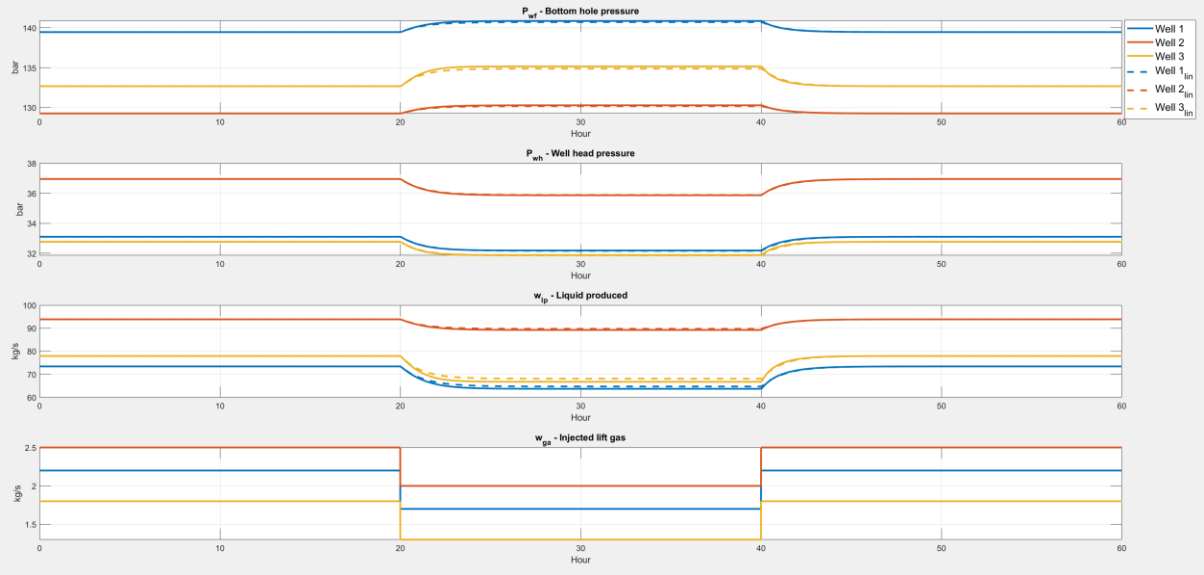

Figure 3.3: Comparing nonlinear and linear model, showing P_{wf}, P_{wh}, w_{lp} and w_{ga}.

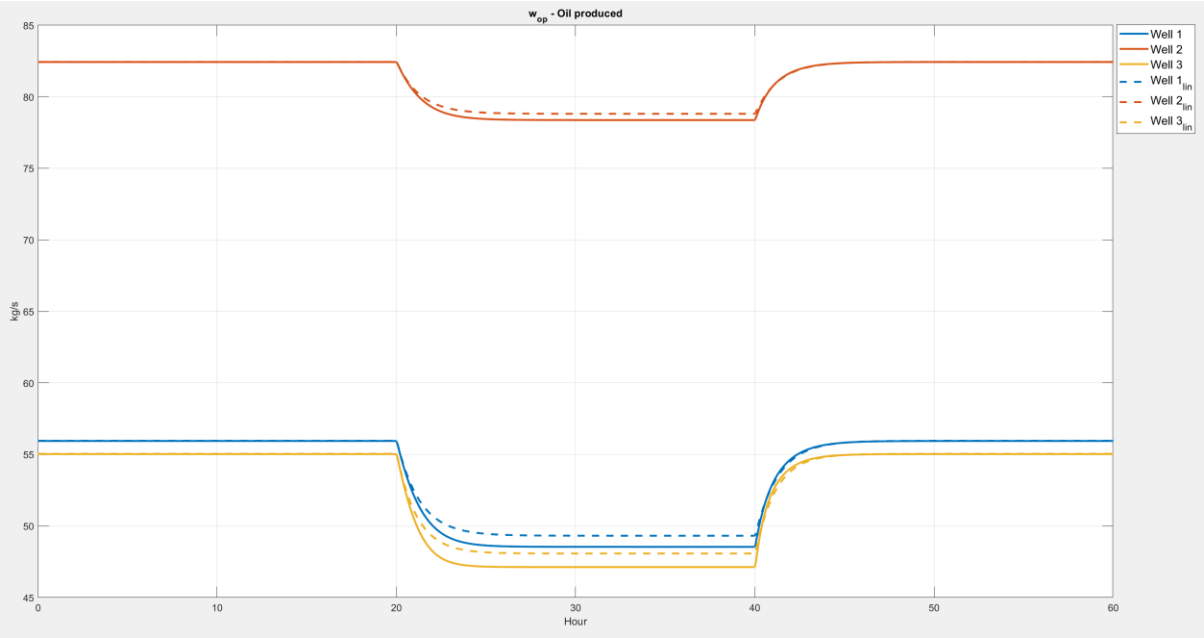

Figure 3.4: Comparing nonlinear and linear model, showing w_{op}.

To stress the linear model, w_{ga} is stepping from the operating points to [1, 1, 1] kg/s and then to [4.5, 4.5, 4.5] kg/s as shown in Figure 3.5. The deviation in w_{op} for well 1 when w_{ga} is 1 kg/s is 5,3 kg/s and 10,1 kg/s when w_{ga} is 4,5 kg/s.

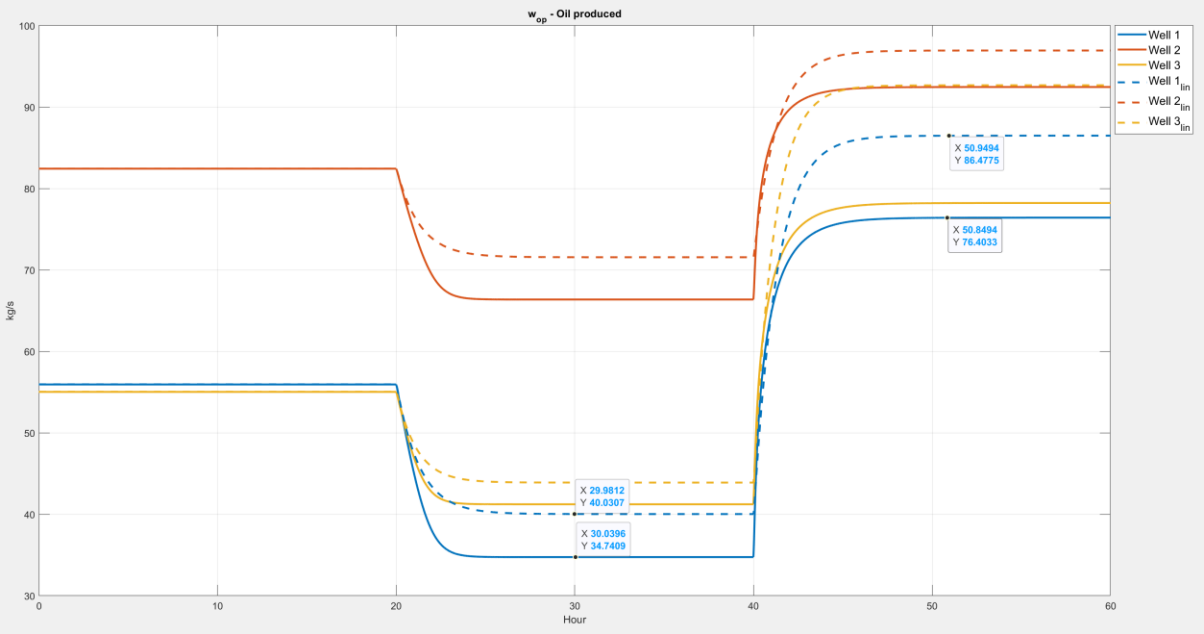

Figure 3.5: Stressing linear model, showing w_{op}.

3.3.2 Open loop simulation – Updating linear model at every iteration

The functions for the system matrices and operating points are created before open loop simulation starts. Then the linear model is updated at every time step. The deviation between models should be smaller than the previous simulation where the model is not updated through the simulation. Figure 3.6 shows the open loop simulation where the linear model is compared to the nonlinear model. This simulation is done with same simulation parameters as in section 3.1.1 where only the nonlinear model was simulated. Appendix L shows the code for the simulation.

The linear model seems to follow the nonlinear model and there is no deviation between the model in steady state. When the models are stepping there is a deviation between the models. Figure 3.7 shows the w_{op}. The deviation in the step change is clear. The time to update the system matrices and operating points is 2,1 ms for each iteration.

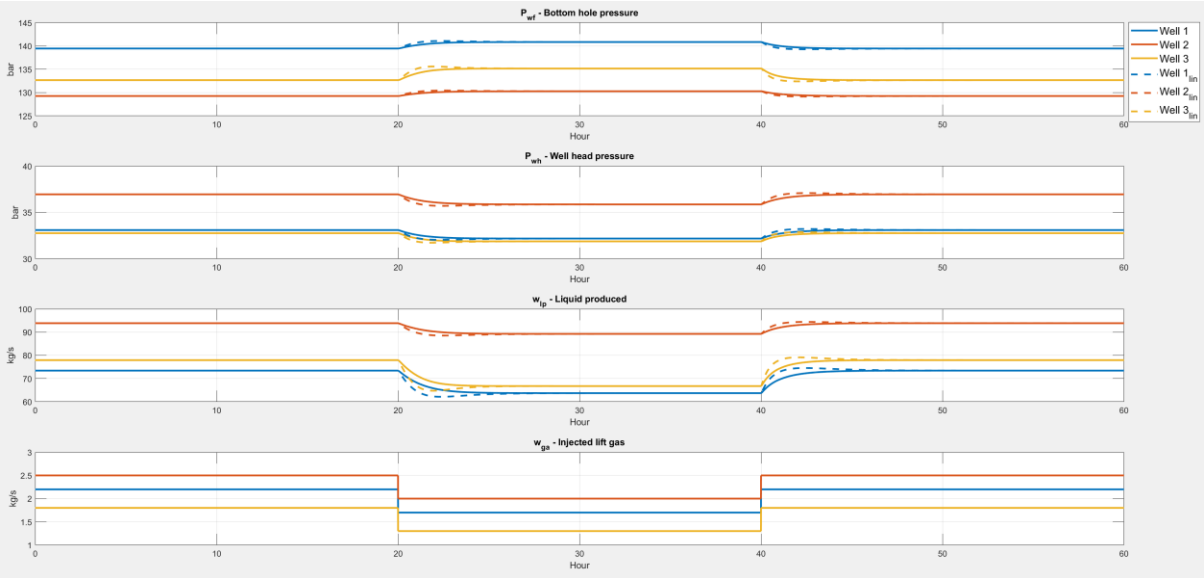

Figure 3.6: Comparing nonlinear and linear model, showing P_{wf}, P_{wh}, w_{lp} and w_{ga}.

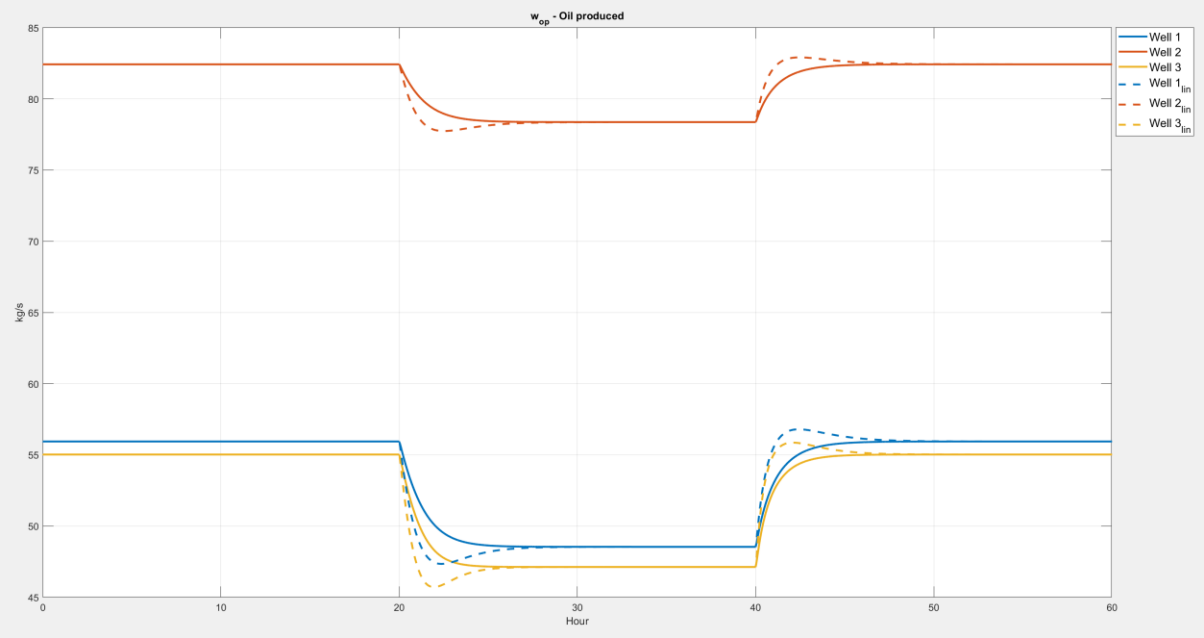

Figure 3.7: Comparing nonlinear and linear model, showing w_{op}.

To stress the linear model, w_{ga} is stepping from the operating points to [1, 1, 1] kg/s and then to [4.5, 4.5, 4.5] kg/s as shown in Figure 3.8. The deviation in the step change is clear, but in steady state the deviations between the models are going to zero.

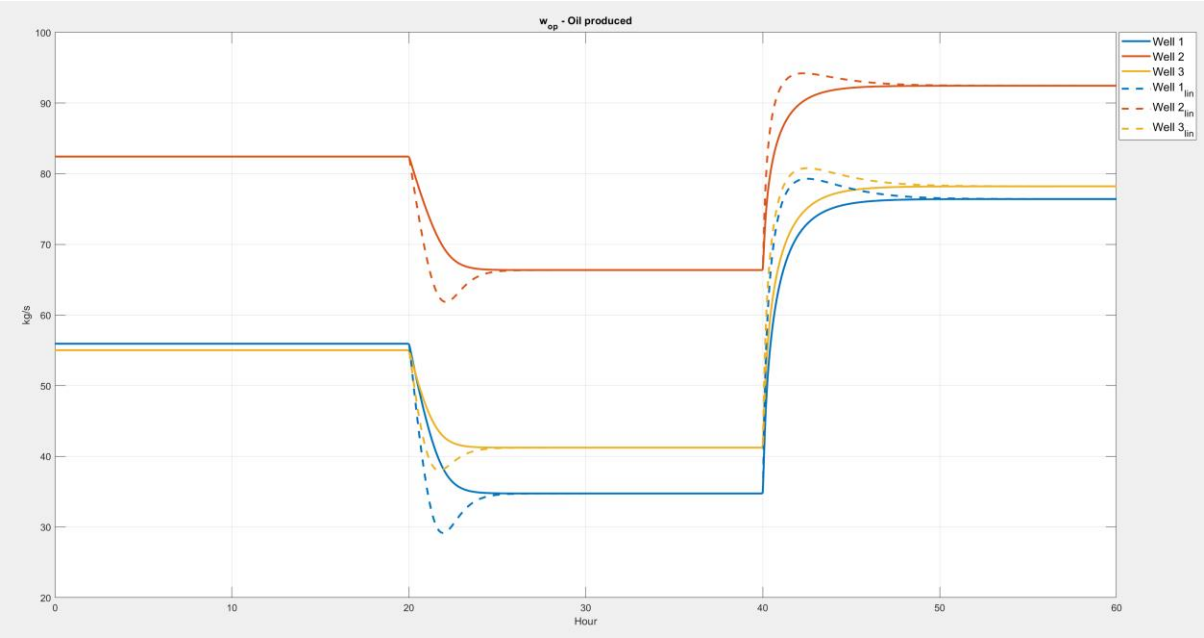

Figure 3.8: Stressing linear model, showing w_{op}.

4 Design of deterministic linear MPC

This chapter shows how the deterministic linear MPC is formulated. The MPC is implemented using two different methods using CasADi, a low-level interface and a high-level interface.

4.1 Formulation of deterministic linear MPC

The deterministic linear MPC calculates the optimized input w_{ga} to the nonlinear model which gives the highest total oil w_{op} produced with the lowest input w_{ga}. Figure 4.1 shows a sketch of the system. The nonlinear model simulates the real process. It is assumed full state knowledge of the system. The states are used as initial states for MPC and to update the system matrices and operating points used in the MPC. The MPC handles constraints such as max available injection gas from compressor w_{gc}^{max} and max separator capacity w_s^{max}. To avoid overloading the separator it is crucial that the model uncertainty is as small as possible.

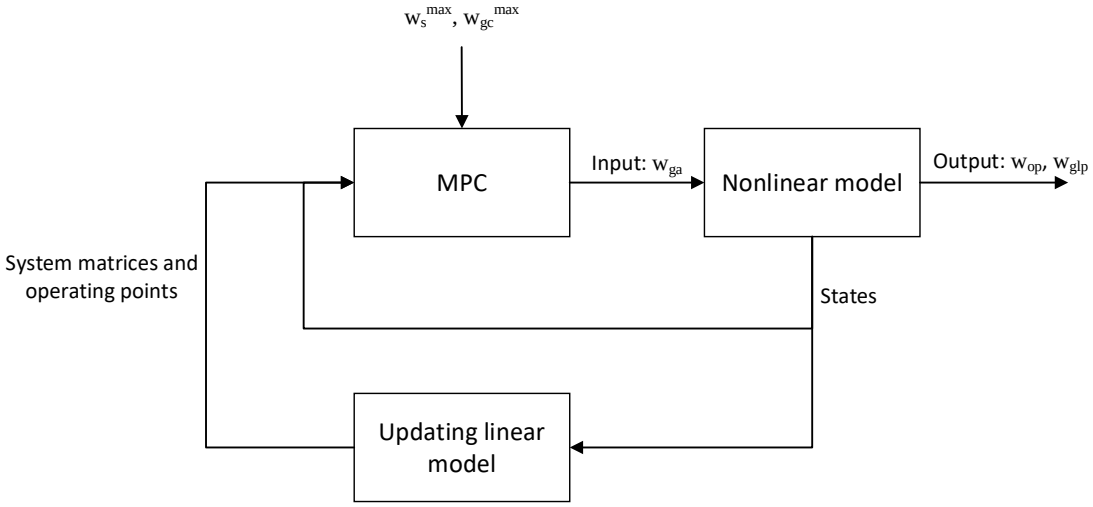

Figure 4.1: Sketch of closed loop simulation.

The linear model developed in chapter 3 is developed further to fit into the deterministic linear MPC. The equations (3.26) - (3.28) is used to convert the model to not be in deviation form. There is only needed two outputs, the w_{op} and w_{glp}. Those are used for maximizing and constraint respectively. For simplicity, in code, the C matrix is divided into C_1 for w_{op} and C_2 for w_{glp}. The linear model for the MPC is a discrete time linear state space model shown in equations (4.1) - (4.3) where the subscript *op* denotes the operating point. The subscript *d* that denotes discrete is removed for simplicity in this chapter. The subscript *k* denotes the time step.

$$x_{k+1} = Ax_k + Bw_{ga,k} - Ax_{op} - Bw_{ga_{op}} + x_{op} \tag{4.1}$$

$$w_{op} = C_1 x_k - C_1 x_{op} + w_{op_{op}} \tag{4.2}$$

$$w_{glp} = C_2 x_k - C_2 x_{op} + w_{glp_{op}} \tag{4.3}$$

Since the linear model deviates from the nonlinear model when moving from the operating points, the MPC is designed with updating the system matrices and the operating points at

every time step. Through the prediction horizon the system matrices and operating points are maintained static.

In CasADi there are two ways to solve a QP problem, using a low-level interface and a high-level interface [11]. The deterministic linear MPC is designed in both methods.

The problem is formulated as a QP optimal problem. The objective, constraints and bounds are formulated in (4.4) - (4.10). This is used in both methods, low-level interface and high-level interface.

$$\underset{w_{ga}}{min} \quad J = \frac{1}{2}\sum_{k=1}^{N}\left(-\left(w_{op,k}^{T} Q w_{op,k}\right) + w_{ga,k}^{T} P w_{ga,k}\right) \tag{4.4}$$

$Subject\ to$:

$$Equality\ constraints$$

$$x_{k+1} = Ax_k + Bw_{ga,k} - Ax_{op} - Bw_{ga_{op}} + x_{op} \tag{4.5}$$

$$w_{op} = C_1 x_k - C_1 x_{op} + w_{op_{op}} \tag{4.6}$$

$$w_{glp} = C_2 x_k - C_2 x_{op} + w_{glp_{op}} \tag{4.7}$$

$$Inequlity\ constraints$$

$$0 \leq \sum_{i=1}^{3} w_{ga,k}^{i} \leq w_{gc,k}^{max} \tag{4.8}$$

$$0 \leq \sum_{i=1}^{3} w_{glp,k}^{i} \leq w_{s}^{max} \tag{4.9}$$

$$Bounds$$

$$0{,}5 \leq w_{ga,k}^{i} \leq 5 \tag{4.10}$$

4.2 Low-level interface

The objective, constraints and bounds in equations (4.4) - (4.10) must be converted to standard structure of a QP problem shown in (4.11). Where z is the vector of unknowns and A_{ei} is the constraint matrix. The subscript *ei* in A_{ei} denotes equality and inequality. Subscript to avoid conflict with the system matrix A. The solver used by CasADi in low-level interface is qpOASES [12]. Appendix M shows the MATLAB function of the deterministic linear MPC in low-level interface.

This section show how it is converted. The conversion is based on lecture notes [2].

$$\begin{aligned} \underset{x}{minimize}: \quad & \frac{1}{2} z^T H z + g^T z \\ \text{Subject to:} \quad & z_l \leq z \leq z_u \\ & b_l \leq A_{ei} z \leq b_u \end{aligned} \tag{4.11}$$

Equation (4.12) shows how the objective looks like in standard structure of QP problems. The input w_{ga} is written as u in this section.

$$J = \frac{1}{2} \underbrace{\begin{bmatrix} u \\ w_{op} \\ w_{glp} \\ x \end{bmatrix}^T}_{z} \underbrace{\begin{bmatrix} H_{11} & 0 & 0 & 0 \\ 0 & -H_{22} & 0 & 0 \\ 0 & 0 & H_{33} & 0 \\ 0 & 0 & 0 & H_{44} \end{bmatrix}}_{H} \underbrace{\begin{bmatrix} u \\ w_{op} \\ w_{glp} \\ x \end{bmatrix}}_{z} + \underbrace{\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix}^T}_{g} \underbrace{\begin{bmatrix} u \\ w_{op} \\ w_{glp} \\ x \end{bmatrix}}_{z} \tag{4.12}$$

The optimization vector z is defined as equation (4.13). It contains inputs (u), outputs (w_{op}, w_{glp}) and states (x), shown in equations (4.14). All variables must be optimized the whole prediction horizon for every time step.

$$z = \begin{bmatrix} u \\ w_{op} \\ w_{glp} \\ x \end{bmatrix} \quad or \quad z^T = \begin{bmatrix} u^T & w_{op}^T & w_{glp}^T & x^T \end{bmatrix} = \begin{bmatrix} u \\ w_{op} \\ w_{glp} \\ x \end{bmatrix}^T \tag{4.13}$$

$$\begin{aligned} u^T &= [u_0^T \quad u_1^T \quad \cdots \quad u_{N-1}^T] && \epsilon\, \mathbb{R}^{(1 \times N.n_u)} \\ w_{op}^T &= \begin{bmatrix} w_{op_1}^T & w_{op_2}^T & \cdots & w_{op_N}^T \end{bmatrix} && \epsilon\, \mathbb{R}^{(1 \times N.n_y)} \\ w_{glp}^T &= \begin{bmatrix} w_{glp_1}^T & w_{glp_2}^T & \cdots & w_{glp_N}^T \end{bmatrix} && \epsilon\, \mathbb{R}^{(1 \times N.n_y)} \\ x^T &= [x_1^T \quad x_2^T \quad \cdots \quad x_N^T] && \epsilon\, \mathbb{R}^{(1 \times N.n_x)} \end{aligned} \tag{4.14}$$

Where:

N = prediction horizon
n_u = number of inputs
n_y = number of outputs for each output (w_{op} and w_{glp})
n_x = number of states
n_z = number of unknown parameters, calculated in equation (4.15).

$$n_z = N(n_u + 2n_y + n_x) \tag{4.15}$$

As shown in equation (4.15) the z vector can be large if using a long prediction horizon N.

Since there is no linear term in the original objective in equation (4.4) the g-vector is a zero-vector with n_z-number of elements.

The diagonal elements H_{11}, H_{22}, H_{33} and H_{44} in the matrix H is shown in equations (4.16) - (4.19).

$$u^T H_{11} u = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}^T \underbrace{\begin{bmatrix} Q_0 & 0 & \dots & 0 \\ 0 & Q_1 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & & Q_{N-1} \end{bmatrix}}_{H_{11}} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \tag{4.16}$$

$$Then:$$
$$H_{11} = I_N \otimes Q, When: Q_0 = Q_1 = \cdots = Q_{N-1}$$
$$Where:$$
$$Q = weighting\ matrix$$

$$w_{op}^T (-H_{22}) w_{op} = \begin{bmatrix} w_{op_1} \\ w_{op_2} \\ \vdots \\ w_{op_N} \end{bmatrix}^T \left(- \underbrace{\begin{bmatrix} P_0 & 0 & \dots & 0 \\ 0 & P_1 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & & P_{N-1} \end{bmatrix}}_{H_{22}} \right) \begin{bmatrix} w_{op_1} \\ w_{op_2} \\ \vdots \\ w_{op_N} \end{bmatrix} \tag{4.17}$$

$$Then:$$
$$H_{22} = I_N \otimes P, When: P_0 = P_1 = \cdots = P_{N-1}$$
$$Where:$$
$$P = weighting\ matrix$$

$$H_{33} = I_N \otimes 0_{ny} \tag{4.18}$$

$$H_{44} = I_N \otimes 0_{nx} \tag{4.19}$$

Since CasADi low-level interface does not support equality constraints all constraints must be formed as inequality constraints as shown in (4.20). Upper and lower bounds are equal in equality constraints to form those as inequality constraints.

$$\underbrace{\begin{bmatrix} b_{e,1} \\ b_{e,2} \\ b_{e,3} \\ b_{il,1} \\ b_{il,2} \end{bmatrix}}_{b_l} \leq \underbrace{\begin{bmatrix} A_{e1,u} & A_{e1,w_{op}} & A_{e1,w_{glp}} & A_{e1,x} \\ A_{e2,u} & A_{e2,w_{op}} & A_{e2,w_{glp}} & A_{e2,x} \\ A_{e3,u} & A_{e3,w_{op}} & A_{e3,w_{glp}} & A_{e3,x} \\ A_{i1,u} & A_{i1,w_{op}} & A_{i1,w_{glp}} & A_{i1,x} \\ A_{i2,u} & A_{i2,w_{op}} & A_{i2,w_{glp}} & A_{i2,x} \end{bmatrix}}_{A_{ei}} \underbrace{\begin{bmatrix} u \\ w_{op} \\ w_{glp} \\ x \end{bmatrix}}_{z} \leq \underbrace{\begin{bmatrix} b_{e,1} \\ b_{e,2} \\ b_{e,3} \\ b_{iu,1} \\ b_{iu,2} \end{bmatrix}}_{b_u} \tag{4.20}$$

Equality constraint 1 in equation (4.5) is converted into standard form in equation (4.21).

$$\underbrace{\begin{bmatrix} Ax_0 + x_{op} - Bu_{op} \\ Ax_0 + x_{op} \\ \vdots \\ Ax_0 + x_{op} \end{bmatrix}}_{b_{e,1}} \leq \left[\underbrace{\begin{bmatrix} -B & 0 & 0 & & 0 \\ 0 & -B & 0 & & 0 \\ 0 & 0 & -B & & 0 \\ & & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -B \end{bmatrix}}_{A_{e1u}} \middle| \underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}}_{A_{e1w_{op}}} \middle| \underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}}_{A_{e1w_{glp}}} \middle| \underbrace{\begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ -A & I & 0 & & 0 \\ 0 & -A & I & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & \cdots & -A & I \end{bmatrix}}_{A_{e1x}} \right] \cdot z \leq b_{e,1} \tag{4.21}$$

Equality constraint 2 in equation (4.6) is converted into standard form in equation (4.22).

$$\underbrace{\begin{bmatrix} W_{op_{op}} - C_1 x_{op} \\ W_{op_{op}} - C_1 x_{op} \\ \vdots \\ W_{op_{op}} - C_1 x_{op} \end{bmatrix}}_{b_{e,2}} \leq \left[\begin{array}{cccc|cccc|cccc|cccc} 0 & 0 & \cdots & 0 & I & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & -C_1 & 0 & \cdots & 0 \\ 0 & 0 & & 0 & 0 & I & & 0 & 0 & 0 & & 0 & 0 & -C_1 & & 0 \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ \underbrace{0}_{A_{e2u}} & 0 & \cdots & 0 & \underbrace{0}_{A_{e2w_{op}}} & 0 & \cdots & I & \underbrace{0}_{A_{e2w_{glp}}} & 0 & \cdots & 0 & \underbrace{0}_{A_{e2x}} & 0 & \cdots & -C_1 \end{array}\right] \cdot z \leq b_{e,2} \tag{4.22}$$

Equality constraint 3 in equation (4.7) is converted into standard form in equation (4.23).

$$\underbrace{\begin{bmatrix} w_{glp_{op}} - C_2 x_{op} \\ w_{glp_{op}} - C_2 x_{op} \\ \vdots \\ w_{glp_{op}} - C_2 x_{op} \end{bmatrix}}_{b_{e,3}} \leq \left[\begin{array}{cccc|cccc|cccc|cccc} 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & I & 0 & \cdots & 0 & -C_2 & 0 & \cdots & 0 \\ 0 & 0 & & 0 & 0 & 0 & & 0 & 0 & I & & 0 & 0 & -C_2 & & 0 \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ \underbrace{0}_{A_{e3u}} & 0 & \cdots & 0 & \underbrace{0}_{A_{e3w_{op}}} & 0 & \cdots & 0 & \underbrace{0}_{A_{e3w_{glp}}} & 0 & \cdots & I & \underbrace{0}_{A_{e3x}} & 0 & \cdots & -C_2 \end{array}\right] \cdot z \leq b_{e,3} \tag{4.23}$$

Inequality constraint 1 in equation (4.8) is converted into standard form in equation (4.24). The S-vector is used to sum the three inputs $w_{ga}^{1,2,3}$.

$$\underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{b_{il,1}} \leq \left[\begin{array}{cccc|cccc|cccc|cccc} S & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & S & & 0 & 0 & 0 & & 0 & 0 & 0 & & 0 & 0 & 0 & & 0 \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ \underbrace{0}_{A_{i1u}} & 0 & \cdots & S & \underbrace{0}_{A_{i1w_{op}}} & 0 & \cdots & 0 & \underbrace{0}_{A_{i1w_{glp}}} & 0 & \cdots & 0 & \underbrace{0}_{A_{i1x}} & 0 & \cdots & 0 \end{array}\right] \cdot z \leq \underbrace{\begin{bmatrix} w_{gc,1}^{max} \\ w_{gc,2}^{max} \\ \vdots \\ w_{gc,N}^{max} \end{bmatrix}}_{b_{iu,1}} \tag{4.24}$$

$$Where:$$
$$S = [1 \quad 1 \quad 1]$$

Inequality constraint 2 in equation (4.9) is converted into standard form in equation (4.25). The S-vector is used to sum the three outputs $w_{glp}^{1,2,3}$

$$\underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{b_{il,2}} \leq \left[\begin{array}{cccc|cccc|cccc|cccc} 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & S & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & & 0 & 0 & 0 & & 0 & 0 & S & & 0 & 0 & 0 & & 0 \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ \underbrace{0}_{A_{i2u}} & 0 & \cdots & 0 & \underbrace{0}_{A_{i2w_{op}}} & 0 & \cdots & 0 & \underbrace{0}_{A_{i2w_{glp}}} & 0 & \cdots & S & \underbrace{0}_{A_{i2x}} & 0 & \cdots & 0 \end{array}\right] \cdot z \leq \underbrace{\begin{bmatrix} w_s^{max} \\ w_s^{max} \\ \vdots \\ w_s^{max} \end{bmatrix}}_{b_{iu,2}} \tag{4.25}$$

$$Where:$$
$$S = [1 \quad 1 \quad 1]$$

4.2.1 Closed loop simulation in low-level interface

As mentioned in section 3.1.1 a rule of thumb is to have the prediction horizon at least equal to the time constant. Which give a prediction horizon N = 240 steps when time step dt = 15 s as calculated in (4.26). The time constant T is approximately 1 h.

$$N = \frac{T}{dt} = \frac{3600\ s}{15\ s} = 240\ steps \tag{4.26}$$

Although the prediction horizon should be 240 steps, the CasADi low-level interface does not manage to find a solution, because of the large optimization problem. The unknown z-vector contains 4320 elements, as calculated in equation (4.15), which must be optimized.

The prediction horizon must be lowered to N = 10 to ensure that the optimizer can solve the problem. Using this prediction horizon gives 180 elements in the z-vector. If using larger prediction horizon CasADi does not manage to find a solution. The constraints are not satisfied and after a while the simulation will stop, because the models are driven out of their working area, and it is not possible to update the system matrices.

Figure 4.2 shows the closed loop simulation. The deterministic MPC manage to maximize the total w_{op} and satisfy the constraint on w_{glp}. Appendix N shows the code for the closed loop simulation with deterministic MPC. Since the linear model is fitted to the linear MPC other MATLAB functions for system matrices and operating points are created. Appendix O show the function of creating system matrices and operating point vectors. Appendix P show the function of updating those. In addition, Appendix Q show a function for creating the vector w_{gc}^{max}.

Simulation parameters:

- N = 10 steps
- w_s^{max} = 305 kg/s
- w_{gc}^{max} = 10 kg/s

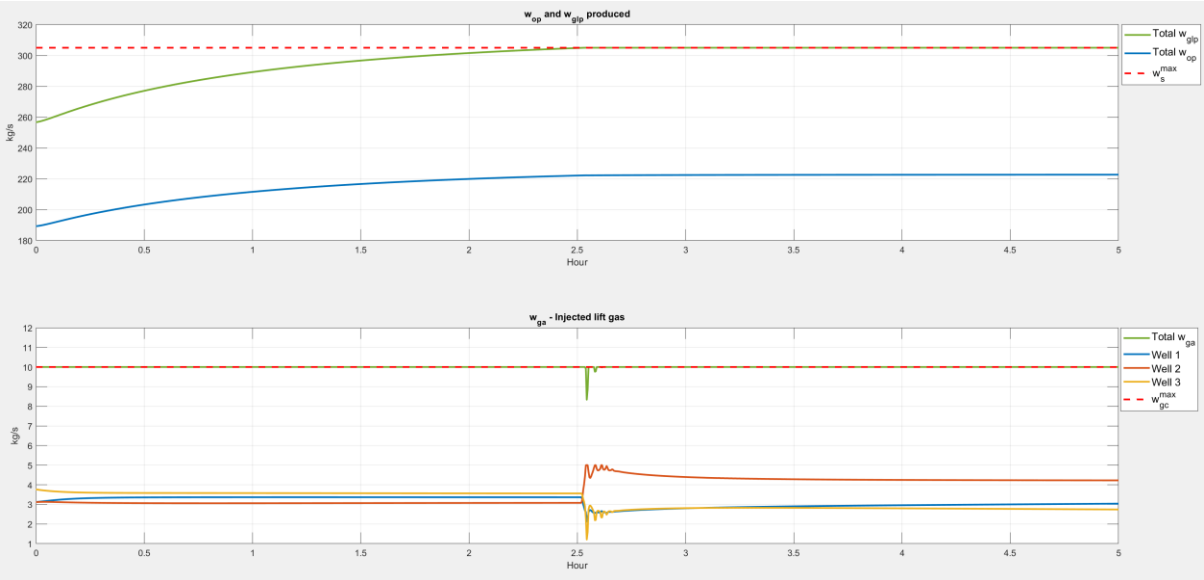

Figure 4.2: Closed loop simulation using CasADi low-level interface.

The mean optimizing time is 0,23 s. Figure 4.3 shows the optimizing time for the whole simulation. It shows clearly that the optimizing time increases when the constraint for w_s^{max} is active at about 2,5 h. Decreasing the w_s^{max} makes the problem infeasible.

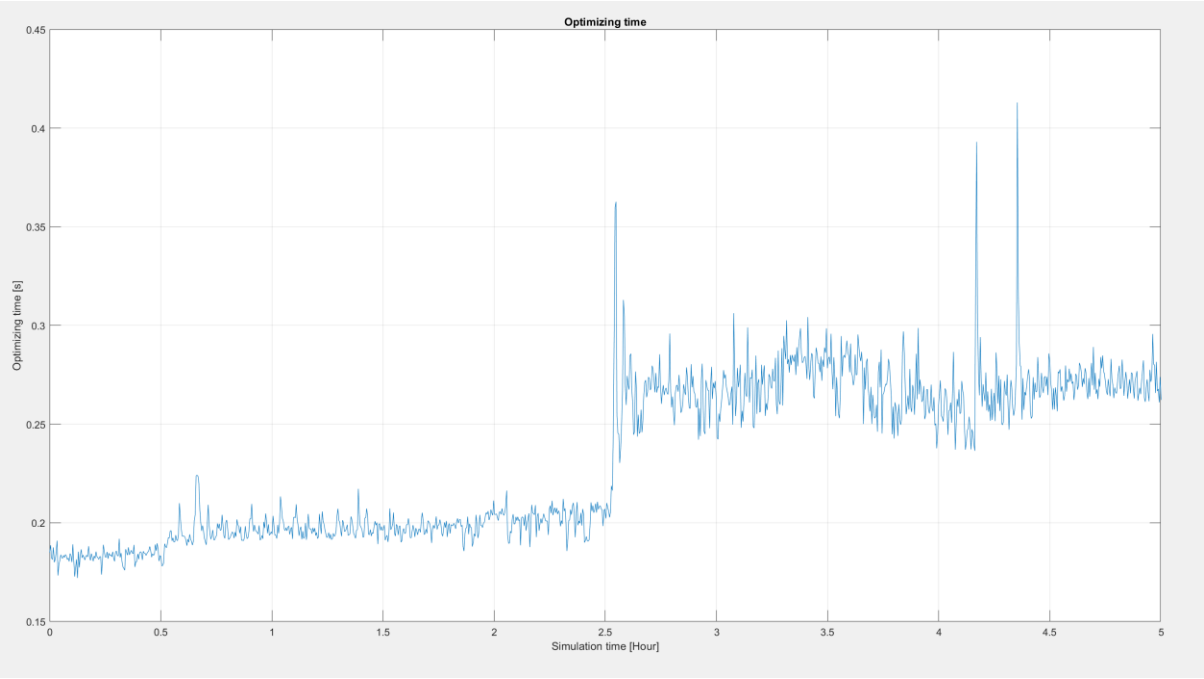

Figure 4.3: Optimizing time of closed loop simulation, mean time = 0,23 s.

4.3 High-level interface

CasADi high-level interface is more sophisticated than low-level interface. The objective, constraints and bounds in (4.4) - (4.10) are written in the code directly and there is no need to convert the problem into the standard form. The only unknowns in this problem are the inputs w_{ga}, and not the states and outputs in addition as in low-level interface. Then there are less variables to optimize, and the problem should be easier to solve. The MPC implementation is based on this instructional video [13]. The code is shown in Appendix R.

This section shows different methods of solving the problem. In high-level interface there are several choices. Since one of the drawbacks in MPC is the high computation time, there have been attempts to lower the computation time. To make the MPC more efficient it is evaluated two different solvers and grouping the input w_{ga}.

4.3.1 Closed loop simulation in high-level interface

Figure 4.4 shows the simulation performed with the high-level interface. The MPC manages to optimize the oil production and the constraints are not violated. The mean optimizing time shown in Figure 4.5 is 21,3 s. In some of the iterations the solver qpOASES gives error that the problem is infeasible, but it still manages to optimize and it seems to satisfy the constraints.

Simulation parameters:

- N = 240 steps
- w_s^{max} = 300 kg/s
- w_{gc}^{max} = 10 kg/s

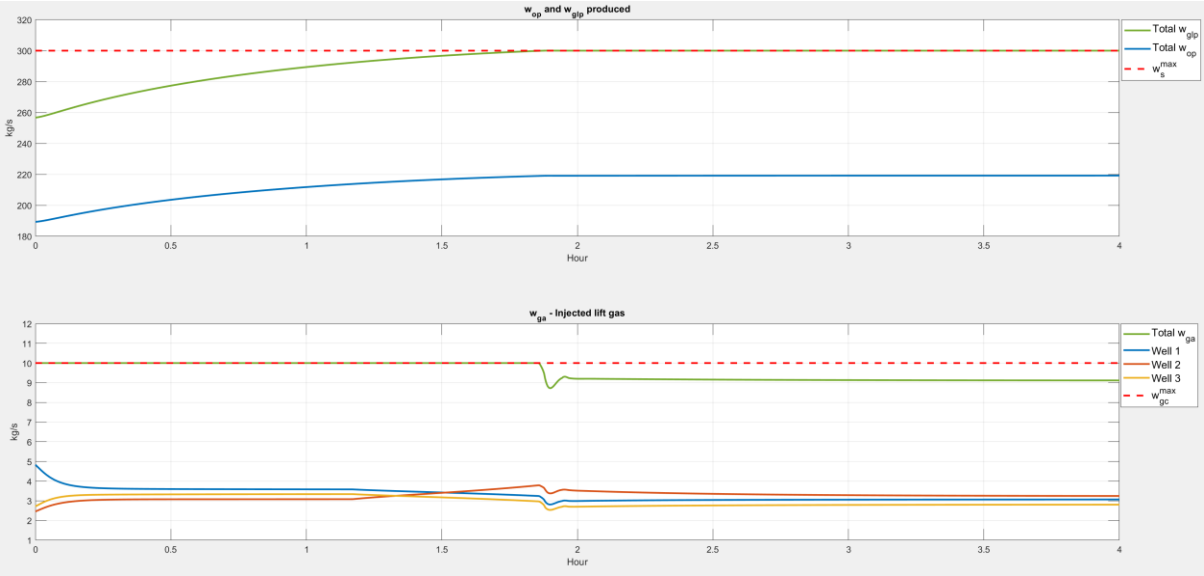

Figure 4.4: Closed loop simulation using CasADi high-level interface.

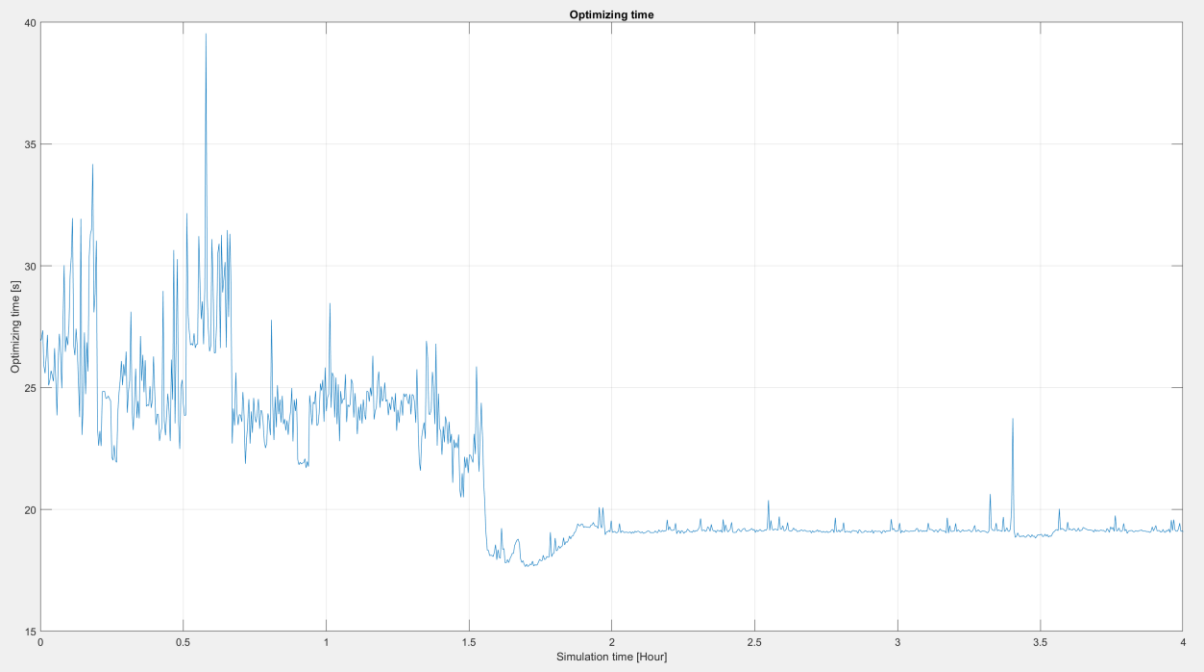

Figure 4.5: Optimizing time of closed loop simulation, mean time = 21,3 s.

4.3.2 Grouping control input

Grouping the optimal control input will let the MPC to optimize fewer inputs. Consequently, the optimizing time should be shorter. Instead of optimizing one input for each time step in the prediction horizon, the input can for example be grouped into three groups. Since the MPC only use the first input, the first group is the most important. Then the first group should be the smallest to obtain the best result.

Grouping the optimal control input u is generally described in Figure 4.6 and equation (4.27). The prediction horizon is 24 time steps. The blue dots represent the control input u and the red dots represent the grouped control input ũ. Equation (4.27) shows that the first three inputs are grouped into $\tilde{u}_0$, then the next nine inputs are grouped into $\tilde{u}_1$ and the last inputs are grouped into $\tilde{u}_2$ [2]. There are now only three inputs for the MPC to optimize, instead of 25.

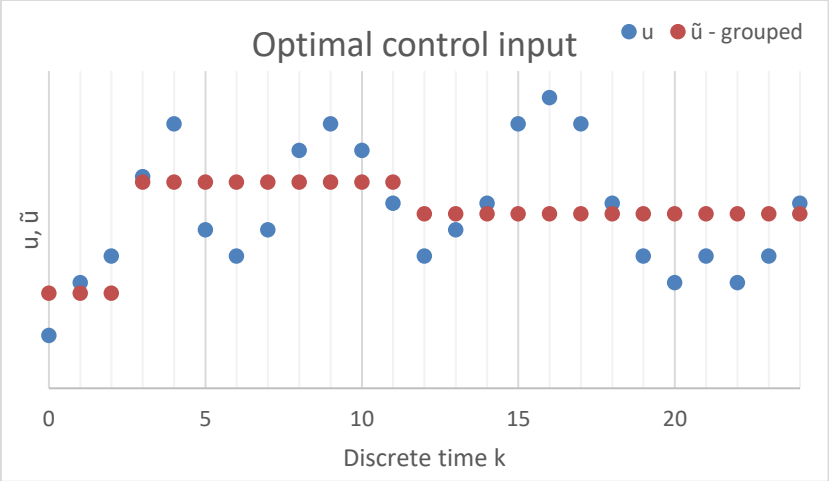

Figure 4.6: General sketch of grouping control input u.

$$\begin{aligned} u_0 = u_1 = u_2 = \tilde{u}_0 \\ u_3 = u_4 = \cdots = u_{11} = \tilde{u}_1 \\ u_{12} = u_{13} = \cdots = u_{24} = \tilde{u}_2 \end{aligned} \tag{4.27}$$

In this case the w_{ga} is grouped into three groups as defined in (4.28). Where N is prediction horizon and k is steps in N.

$$\begin{aligned} w^i_{ga,k} &= 1.group \quad when \quad k \leq 5 \\ w^i_{ga,k} &= 2.group \quad when \quad 5 < k \leq 0{,}5N \\ w^i_{ga,k} &= 3.group \quad when \quad 0{,}5N < k \leq N \end{aligned} \tag{4.28}$$

Figure 4.7 shows the optimizing time when doing the same simulation with and without grouping. The prediction horizon is 240 steps. There are 720 unknowns without grouping and nine with grouping. Grouping the inputs shows that the problem is less time consuming to solve:

- Mean optimizing time no grouping: 26,1 s
- Mean optimizing time with grouping: 1,8 s

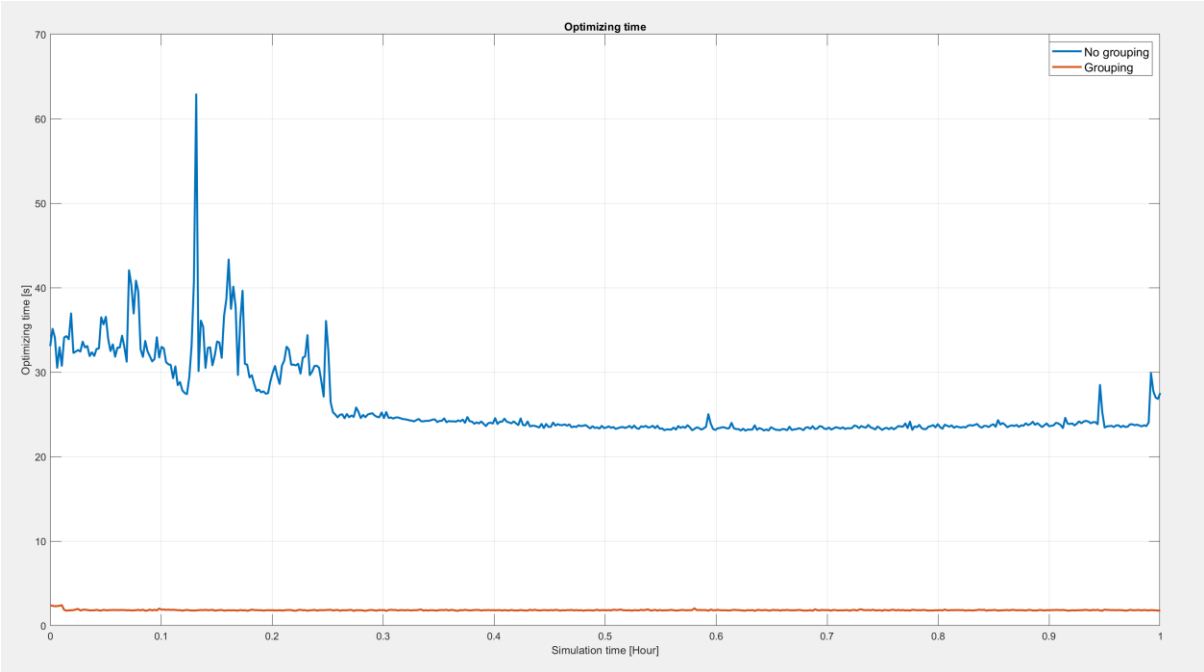

Figure 4.7: Optimizing time grouping vs. no grouping.

Figure 4.8 shows the closed loop simulation when grouping the inputs. The simulation parameters are the same as in section 4.3.1. The results from the simulation without grouping, Figure 4.4, is compared with the simulation with grouping. They are similar, but the optimizing time is lower. w_{op} which is maximized is 222,5 kg/s, when steady state in both simulations.

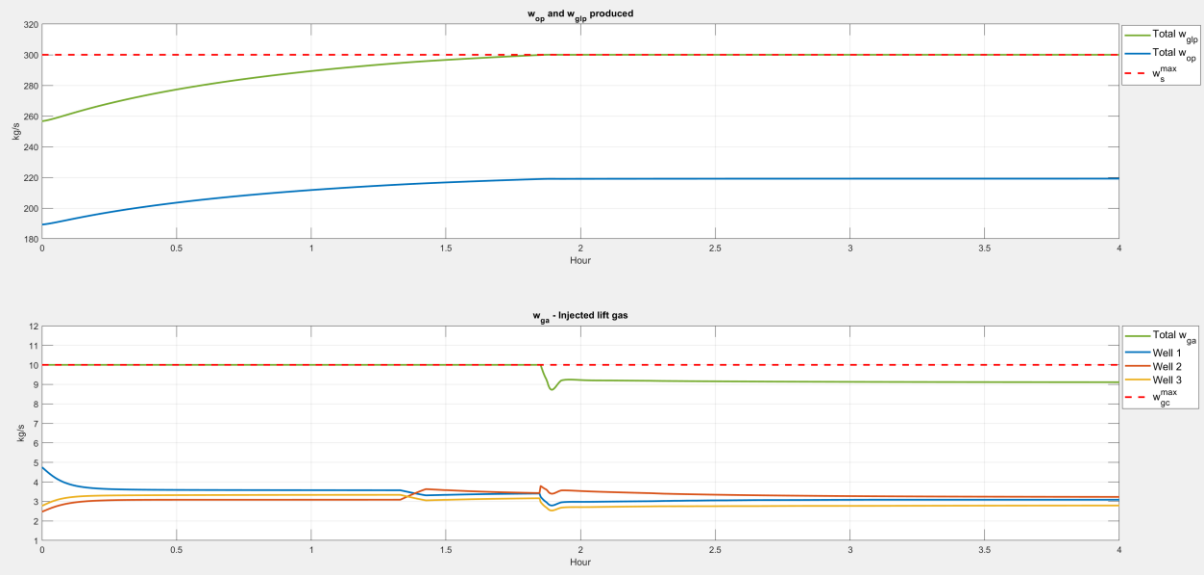

Figure 4.8: Closed loop simulation when grouping inputs w_{ga}.

4.3.3 Comparing optimizing solvers

Until this point it's only used the solver qpOASES. In high-level interface the solver used by CasADi can be changed. For nonlinear programming the IPOPT [14] solver is often used and is a part of the CasADi installation [11]. Since quadratic programming is a subset of nonlinear

programming the IPOPT solver should also work. The qpOASES is assumed to be fastest since it is developed for quadratic programming. The same simulation as in section 4.3.2 is done with the IPOPT solver instead of the qpOASES. Figure 4.9 shows the result and Figure 4.10 shows the optimizing time. The mean optimizing time is 2,3 s. The MPC manages to maximize the oil production and the constraints are satisfied. The inputs are oscillating when the constraint w_s^{max} is reached. Attempts have been made to adjust the weighting matrices, but the result is not better. The oscillations seem to be caused by the problem getting infeasible, and then adjusted into feasible region, repeatedly. The qpOASES seems to handle this better, since it does not oscillate.

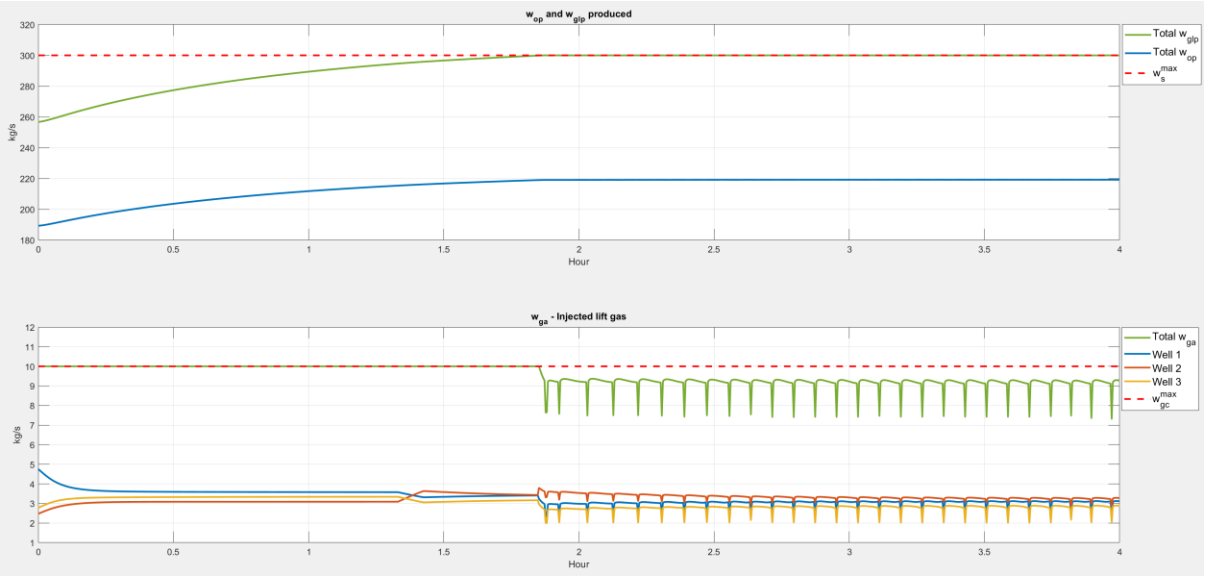

Figure 4.9: Closed loop simulation done with the IPOPT solver.

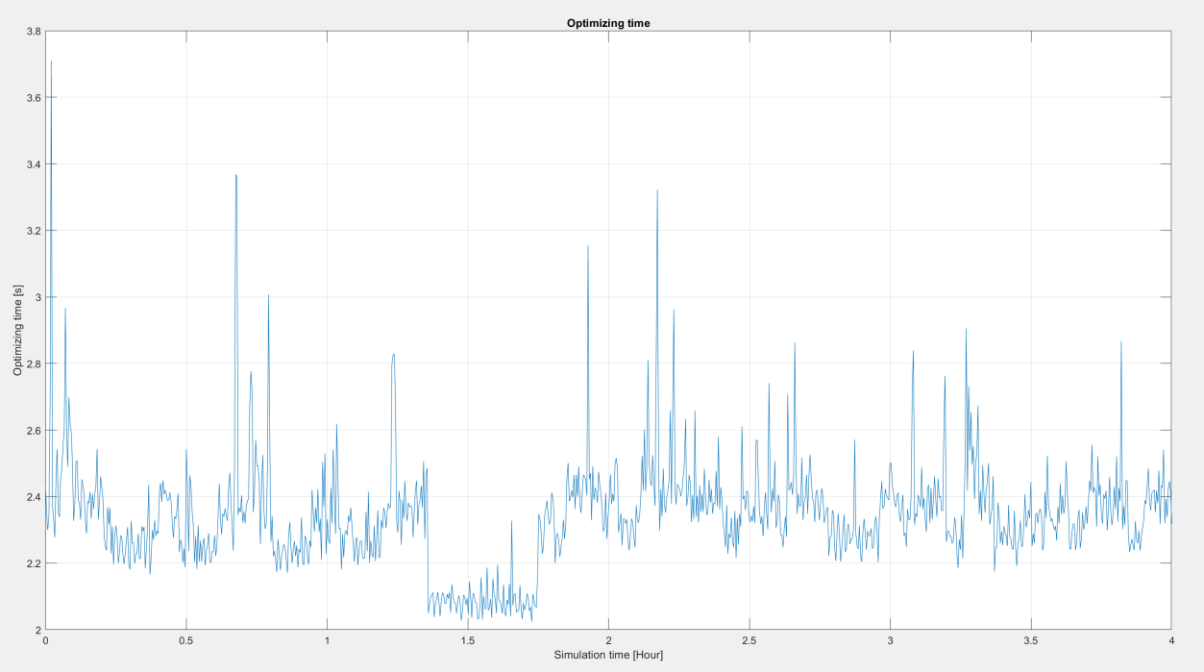

Figure 4.10: Optimizing time using IPOPT solver, mean optimizing time = 2,3 s.

To remove the oscillations and get the optimization problem feasible when using IPOPT, it is tested to add a soft constraint as a prewarning below the hard constraint w_s^{max}. The idea is an approach of the alarm strategy, high alarm and high-high alarm/trip [15], but in this case the w_{glp} should lie between the "high" as a soft constraint and the "high-high" as a hard constraint. The optimization problem is formulated as shown in equation (4.29) - (4.37). The difference from the optimization problem in section 4.3.1 are the objective in equation (4.29), a new inequality constraint in equation (4.35) and the bounds for the slack variable in equation (4.37). A slack variable s_1 and its weighting scalar S are added to the objective. In the new constraint the slack variable s_1 is added to LHS and an offset w_s^{offset} to the RHS. The value for the offset is typically 2 – 5 kg/s. The soft constraint will lie for example 2 kg/s below the w_s^{max}, but the MPC is allowed to violate it. The code of the modified MPC is shown in Appendix S.

$$\underset{w_{ga},\, s_1}{min} \quad J = \sum_{k=1}^{N} \left(-\left(w_{op,k}^T Q w_{op,k}\right) + w_{ga,k}^T P w_{ga,k} + s_{1,k}^2 S\right) \tag{4.29}$$

Subject to:

Equality constraints

$$x_{k+1} = Ax_k + Bw_{ga,k} - Ax_{op} - Bw_{ga_{op}} + x_{op} \tag{4.30}$$

$$w_{op} = C_1 x_k - C_1 x_{op} + w_{op_{op}} \tag{4.31}$$

$$w_{glp} = C_2 x_k - C_2 x_{op} + w_{glp_{op}} \tag{4.32}$$

Inequlity constraints

$$0 \leq \sum_{i=1}^{3} w_{ga,k}^i \leq w_{gc,k}^{max} \tag{4.33}$$

$$0 \leq \sum_{i=1}^{3} w_{glp,k}^i \leq w_s^{max} \tag{4.34}$$

$$0 \leq \sum_{i=1}^{3} w_{glp,k}^i - s_{1,k} \leq w_s^{max} - w_s^{offset} \tag{4.35}$$

Bounds

$$0{,}5 \leq w_{ga,k}^i \leq 5 \tag{4.36}$$

$$0 \leq s_{1,k} \leq \infty \tag{4.37}$$

Tuning the w_s^{offset} and the weighting scalar S properly gives no oscillations and w_{glp} close to w_s^{max}. Then the MPC uses the soft constraint in equation (4.35) to limit the w_{glp} and the hard constraint in equation (4.34) to ensure that the w_{glp} does not violate the w_s^{max}. This is shown in

Figure 4.11. The unknown variables are increasing by N-numbers. This results in longer optimizing time, shown in Figure 4.12. Comparing the result with the simulation performed in section 4.3.2 show higher optimizing time and slightly less w_{op}. Mean optimizing time is 2.8 s, which is 1,2 s higher than without soft constraint. w_{op} = 218,3 kg/s, which is 4,2 kg/s less than without soft constraint.

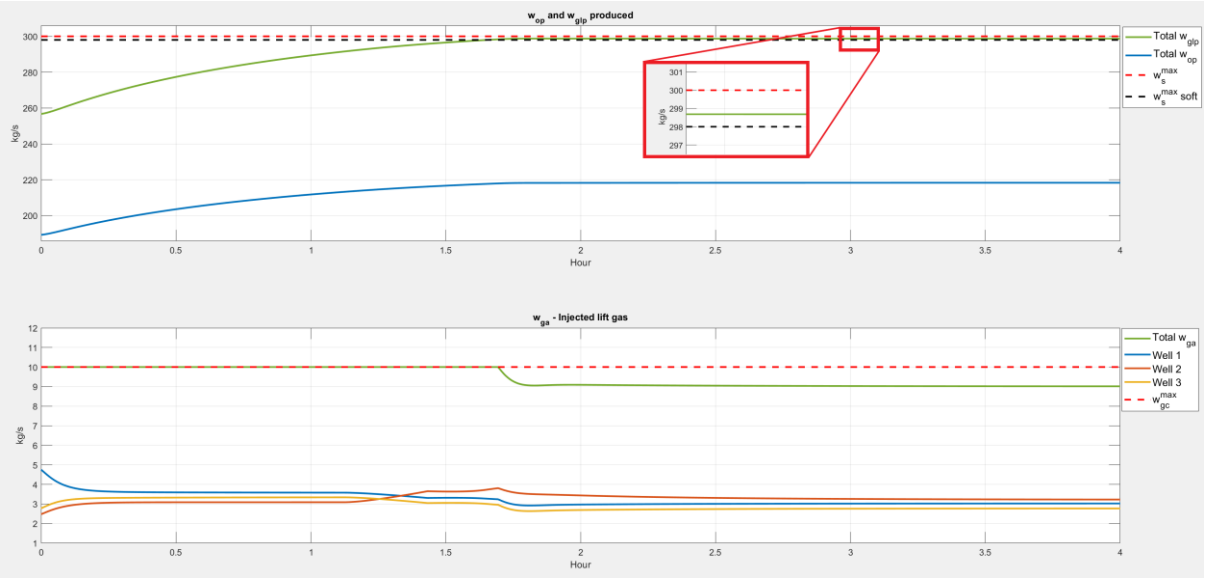

Figure 4.11: Closed loop simulation with an extra soft constraint[1].

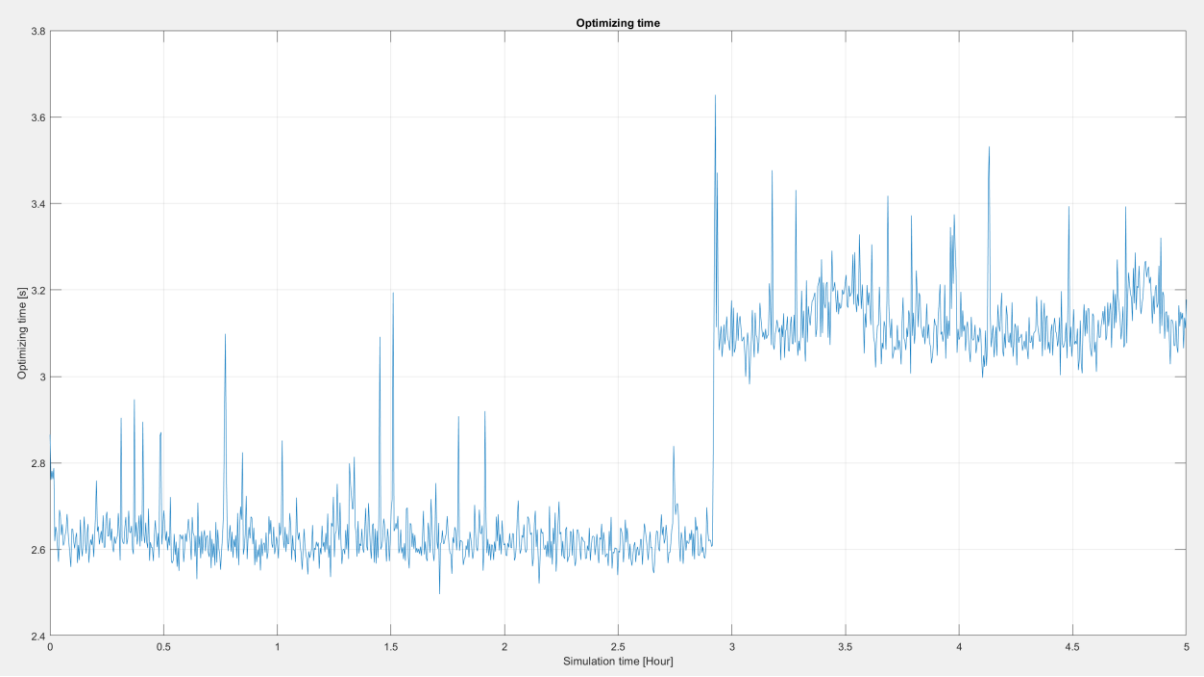

Figure 4.12: Optimizing time when adding slack variable. Mean time: 2,8 s.

[1] "w_s^{max} – soft" is equal to the term "w_s^{max} - w_s^{offset}" in equation (4.35)

4.4 Stochastic analysis of deterministic linear MPC

The deterministic MPC manages to optimize the nonlinear model where all parameters are known. Although, there are always model uncertainty [2]. This section shows how the deterministic MPC acts to the nonlinear model with uncertain parameters. To simulate uncertainty the constants PI, WC and GOR are defined as uncertain. Table 4.1 shows 15 different simulations. The first simulation is the nominal model. The 14 next simulations PI, WC and GOR deviate from their nominal values.

Table 4.1: Uncertain parameters PI, WC and GOR.

Deviation from nominal values in percent									
Nr.	PI^1	PI^2	PI^3	WC^1	WC^2	WC^3	GOR^1	GOR^2	GOR^3
1	0	0	0	0	0	0	0	0	0
2	-10	-10	10	20	20	-20	20	-20	-20
3	10	-10	10	-20	20	-20	-20	-20	-20
4	-10	10	10	20	-20	-20	20	20	20
5	10	10	10	-20	-20	-20	-20	20	20
6	-10	-10	-10	20	20	20	20	-20	20
7	10	-10	-10	-20	20	20	-20	-20	20
8	-10	10	-10	20	-20	20	20	20	-20
9	10	10	-10	-20	-20	20	-20	20	-20
10-15	Uniform random numbers in range [-10 10] for PI, and [-20 20] for WC and GOR								

Simulation parameters:

- N = 240 steps
- w_s^{max} = 300 kg/s
- w_{gc}^{max} = First $^1/_3$ 10 kg/s next $^1/_3$ ramping to 8 kg/s and the last $^1/_3$ ramping to 10 kg/s.

To test the deterministic MPC further, the w_{gc}^{max} is divided in three parts. All the simulations are done after each other and plotted in the same figures. Figure 4.13 shows how the total simulation is done. The method is like the method in Figure 4.1. The deterministic MPC is the same as designed in section 4.3, equation (4.4) - (4.10). The simulation is done to the simulation number one and then number two, listed in Table 4.1, and continues to number 15. In this simulation the code is divided into two parts. One part which handles the different simulations parameters, Appendix T and another part which run the different simulations, Appendix U.

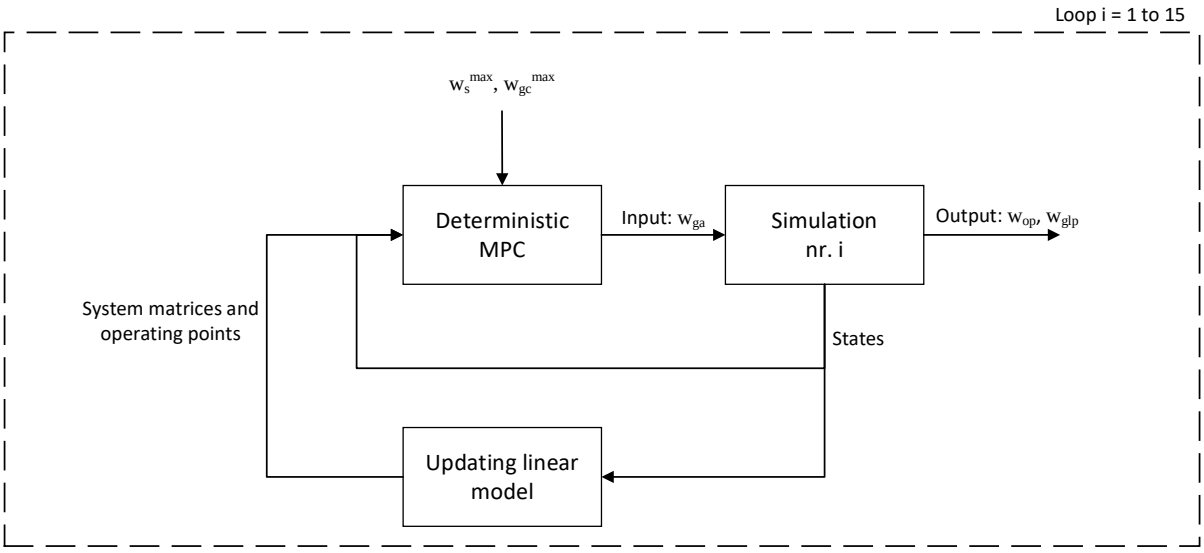

Figure 4.13: Sketch of how the simulation is done.

Figure 4.14 shows the oil produced and the total liquid and gas produced with its constraint. All the wells oil production are shown to show that none of the wells are dying when maximizing total oil production. The colored graphs represents the nominal model, and all the black graphs represent the uncertain models.

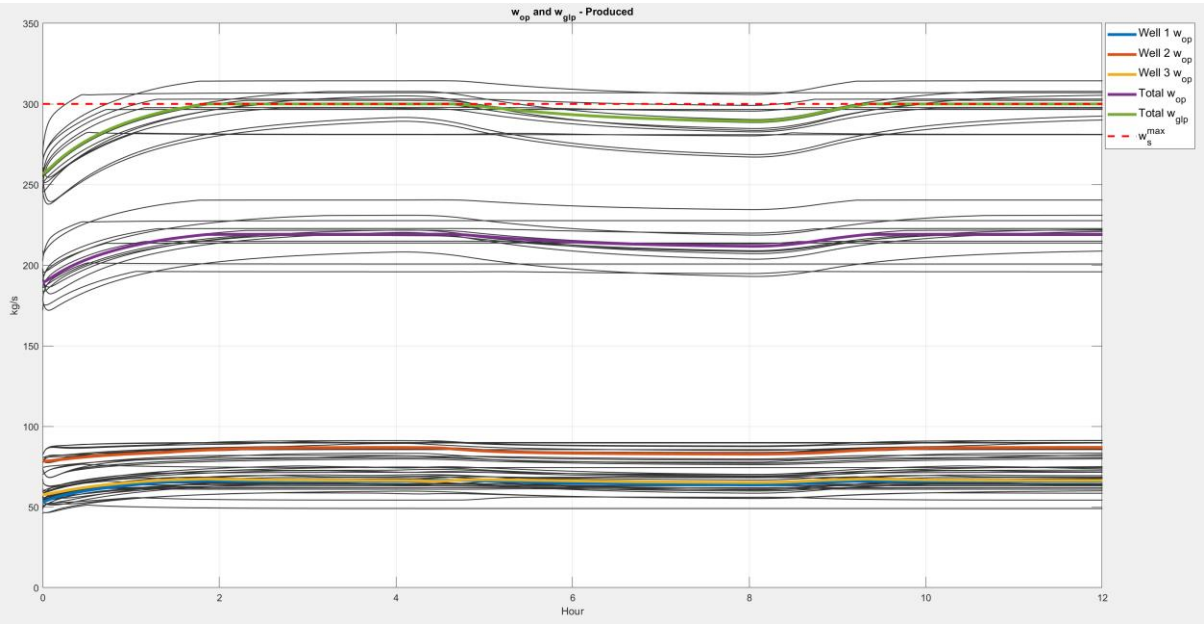

Figure 4.14: Closed loop simulation of deterministic MPC on uncertain models – Outputs.

Figure 4.15 shows the inputs. First the input for each well is present then in the last plot the total w_{ga} is present together with w_{gc}^{max}. Figure 4.16 shows the optimizing time for each iteration for each simulation. The mean optimizing time is 1,42 s, which is close to the result in section 4.3.2. This is not unexpected, since the simulation is similar.

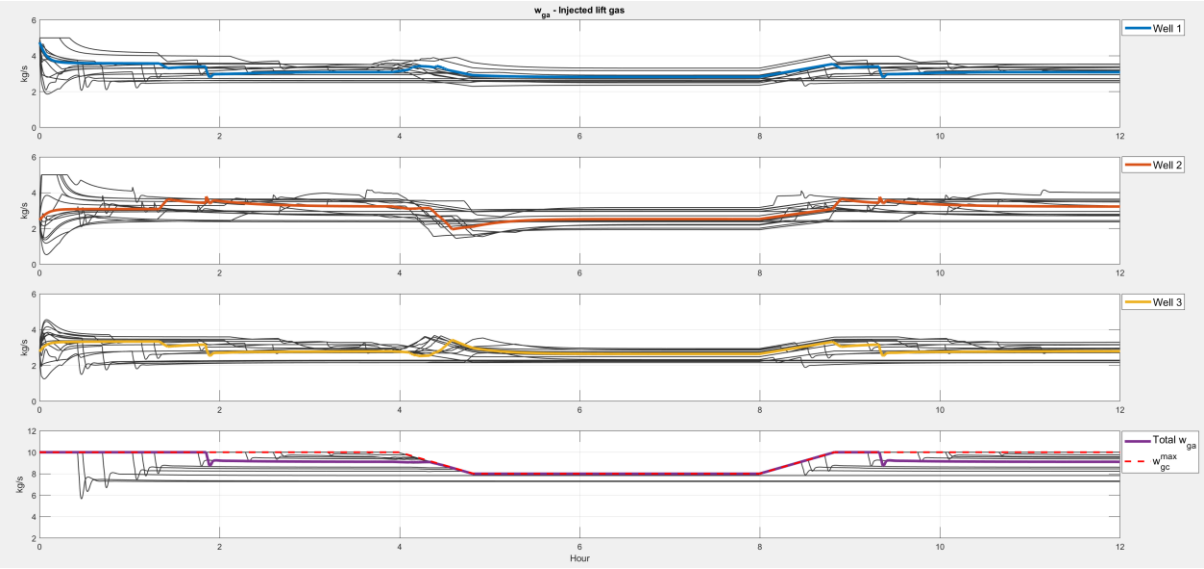

Figure 4.15: Closed loop simulation of deterministic MPC on uncertain models – Inputs.

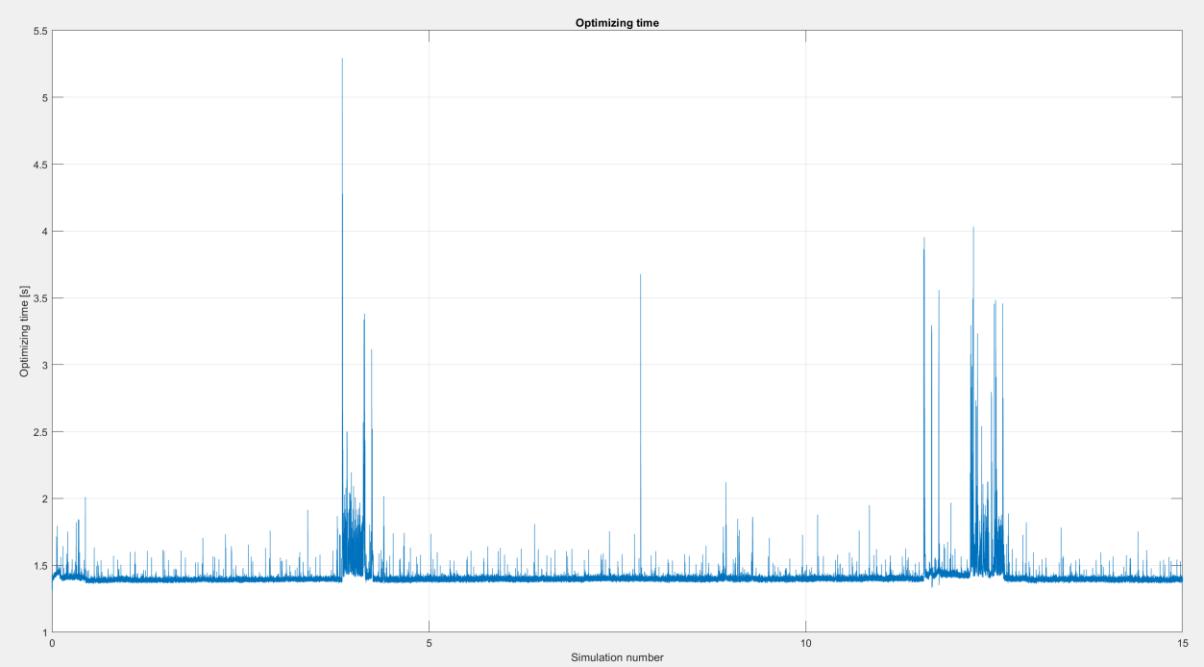

Figure 4.16: Closed loop simulation of deterministic MPC on uncertain models – Optimizing time.

The result shows that the deterministic MPC manages to maximize the uncertain models, but the constraint w_s^{max} is violated in several simulations. There is no guarantee that the deterministic MPC wont violate constraints and overload the separator.

5 Design of robust linear MPC

Since the deterministic MPC can overload the separator, because of model uncertainty, a robust MPC must be designed. Only the robust multi-stage MPC is evaluated. This chapter shows how the multi-stage MPC is designed and results of several simulations.

5.1 Multi-stage MPC

When building the scenarios, the uncertain parameters to the nonlinear model in the NMPC is normally changed. In the linear MPC there is no direct access to the parameters in the model used by the MPC. Then the system matrices and the operating points must be created for each branch according to the uncertain parameters. This means that when using three branches there are needed three instances of system matrices and operating points.

Figure 5.1 shows a sketch of how the branching of the multi-stage MPC is done with three branches. Where j is the branch and k is the step in time in A^j, B^j, C_1^j, C_2^j, op^j, $w_{ga,k}^j$ and x_k^j. Notice that the $w_{ga,1}^j = w_{ga,1}^1$ for all branches in the first time step. There are tested 3 and 10 branches. The uncertainties are assumed to not change in time. Then the robust horizon, Nr, is kept at 1 throughout this report [4].

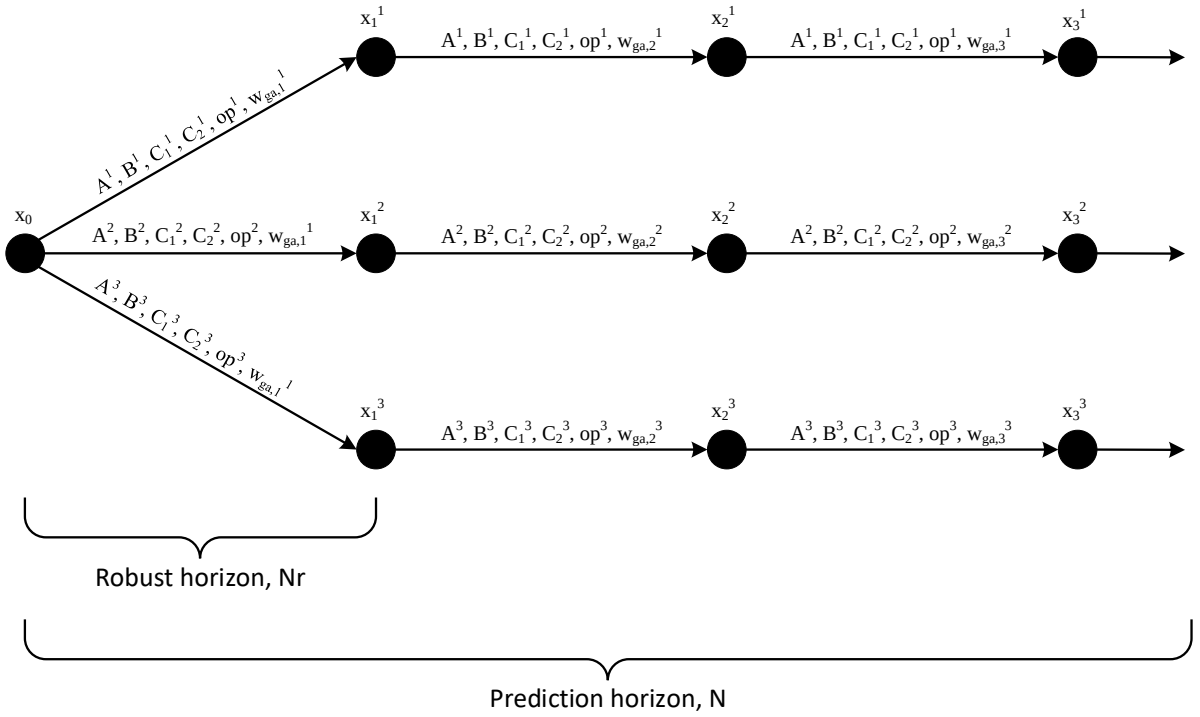

Figure 5.1: Branching in the robust multi-stage linear MPC.

The mathematical expression is shown in equation (5.1) - (5.8). The expression is close to the deterministic MPC shown in (4.4) - (4.10). All system matrices, operating points, states, inputs and outputs have several instances j according to the scenarios. A probability ω is added for each scenario in the objective in equation (5.1). Since no information about the probability

exists for the scenarios, the ω = 1 for all scenarios. It is one new non-anticipativity constraint, equation (5.5), that ensure equal inputs w_{ga} when scenario sharing parent node. If x_1^j shares parent node then all $w_{ga,1}^j$ are equal. In the code in Appendix V this is solved symbolically when building the problem and not as an extra constraint.

$$\min_{w_{ga}} \quad J = \sum_{j=1}^{N_S} \omega^j \sum_{k=1}^{N} \left(- \left({w_{op,k}^j}^T Q w_{op,k}^j \right) + {w_{ga,k}^j}^T P w_{ga,k}^j \right) \tag{5.1}$$

Subject to:

Equality constraints

$$x_{k+1}^j = A^j x_k^j + B^j w_{ga,k}^j - A^j x_{op}^j - B^j {w_{ga}}_{op}^j + x_{op}^j \tag{5.2}$$

$$w_{op}^j = C_1^j x_k^j - C_1^j x_{op}^j + {w_{op}}_{op}^j \tag{5.3}$$

$$w_{glp}^j = C_2^j x_k^j - C_2^j x_{op}^j + {w_{glp}}_{op}^j \tag{5.4}$$

$$w_{ga,k}^{j,i} = w_{ga,k}^{l,i} \; if \; x_k^{p(j)} = x_k^{p(l)} \tag{5.5}$$

Inequlity constraints

$$0 \leq \sum_{i=1}^{3} w_{ga,k}^{j,i} \leq w_{gc,k}^{max} \tag{5.6}$$

$$0 \leq \sum_{i=1}^{3} w_{glp,k}^{j,i} \leq w_s^{max} \tag{5.7}$$

Bounds

$$0{,}5 \leq w_{ga,k}^{j,i} \leq 5 \tag{5.8}$$

The multi-stage MPC can be formed with the slack variable introduced in section 4.3.3, shown in equation (5.9) - (5.18). Then it is possible to attempt the IPOPT solver in addition, shown in Appendix W.

$$\min_{w_{ga}, s_1} \quad J = \sum_{j=1}^{N_S} \omega^j \sum_{k=1}^{N} \left(- \left({w_{op,k}^j}^T Q w_{op,k}^j \right) + {w_{ga,k}^j}^T P w_{ga,k}^j + {s_{1,k}^j}^2 S \right) \tag{5.9}$$

Subject to:

Equality constraints

$$x_{k+1}^j = A^j x_k^j + B^j w_{ga,k}^j - A^j x_{op}^j - B^j {w_{ga}}_{op}^j + x_{op}^j \tag{5.10}$$

$$w_{op}^{j} = C_1^j x_k^j - C_1^j x_{op}^j + w_{op_{op}}^j \tag{5.11}$$

$$w_{glp}^{j} = C_2^j x_k^j - C_2^j x_{op}^j + w_{glp_{op}}^j \tag{5.12}$$

$$w_{ga,k}^{j,i} = w_{ga,k}^{l,i} \; if \; x_k^{p(j)} = x_k^{p(l)} \tag{5.13}$$

$$Inequlity\ constraints$$

$$0 \leq \sum_{i=1}^{3} w_{ga,k}^{j,i} \leq w_{gc,k}^{max} \tag{5.14}$$

$$0 \leq \sum_{i=1}^{3} w_{glp,k}^{j,i} \leq w_s^{max} \tag{5.15}$$

$$0 \leq \sum_{i=1}^{3} w_{glp,k}^{j,i} - s_{1,k}^{j} \leq w_s^{max} - w_s^{offset} \tag{5.16}$$

$$Bounds$$

$$0{,}5 \leq w_{ga,k}^{j,i} \leq 5 \tag{5.17}$$

$$0 \leq s_{1,k}^{j} \leq \infty \tag{5.18}$$

5.1.1 Choosing scenarios

The scenarios must be chosen properly. There are three uncertain parameters in three wells. This means nine uncertain parameters in total. Every uncertain parameter should have three instances, [min nominal max]. The number of scenarios is calculated according to equation (2.6). Using this approach will give unreasonably many scenarios. In this case 19683 scenarios. The computational cost will be too high.

Instead, the uncertain parameters are analyzed to find the combination of parameters that gives the worst-case scenarios. The worst-case scenarios are the combination of uncertain parameters which gives the highest and lowest total w_{glp}, since the w_s^{max} is an upper bound of system. In that way it is enough to use three scenarios to ensure that the constraints are not violated [16]. Then the combination of uncertain parameters that gives [min nominal max] total w_{glp} are used.

Table 5.1 shows the combination of uncertain parameters that are used in the multi-stage linear MPC with three scenarios. Number one gives nominal model, number two gives minimum and number three gives maximum. Appendix X shows the code for building the system matrices and operating point vectors for each scenario.

Table 5.1: Uncertain parameters PI, WC and GOR, [min nom max].

Deviation from nominal values in percent			
Nr.	$PI^{1,2,3}$	$WC^{1,2,3}$	$GOR^{1,2,3}$
1	0	0	0
2	-10	20	-20
3	10	-20	20

To test if it is better to use more than three scenarios it is performed simulations with 10 scenarios. In addition to Table 5.1 all combination of PI, WC and GOR are tested, but when all values inside each parameter are equal. Shown in Table 5.2. It is added one more scenario with individual values for each well, shown in Table 5.3. This table can be extended if require more scenarios.

Table 5.2: Uncertain parameters PI, WC and GOR.

Deviation from nominal values in percent			
Nr.	$PI^{1,2,3}$	$WC^{1,2,3}$	$GOR^{1,2,3}$
4	10	20	20
5	-10	20	20
6	-10	-20	20
7	10	20	-20
8	10	-20	-20
9	-10	-20	-20

Table 5.3: Uncertain parameters PI, WC and GOR. Individual values.

Deviation from nominal values in percent									
Nr.	PI^1	PI^2	PI^3	WC^1	WC^2	WC^3	GOR^1	GOR^2	GOR^3
10	-10	10	10	20	-20	-20	-20	20	20

5.2 Simulation results

This section contains several simulations of the robust multi-stage linear MPC. To do a precise evaluation of the MPC, it is necessary to test the MPC under different conditions. Appendix Y

and Appendix Z shows the codes for the closed loop simulation when using multi-stage MPC. Appendix AA shows the code for updating the system matrices and the operating point vectors.

5.2.1 Comparing robust MPC with deterministic MPC

To compare the robust MPC with the deterministic MPC, it is done a simulation which is the same as in section 4.4. Now the same simulation parameters are used as in the stochastic analysis of deterministic MPC, but instead with the robust MPC formulated in equation (5.1) - (5.8). Figure 5.2 shows the outputs of the simulation. The w_s^{max} constraint are satisfied and the oil production is maximized. None of the wells are dying.

The colored graphs represent the nominal model, and all the black graphs represent the uncertain models.

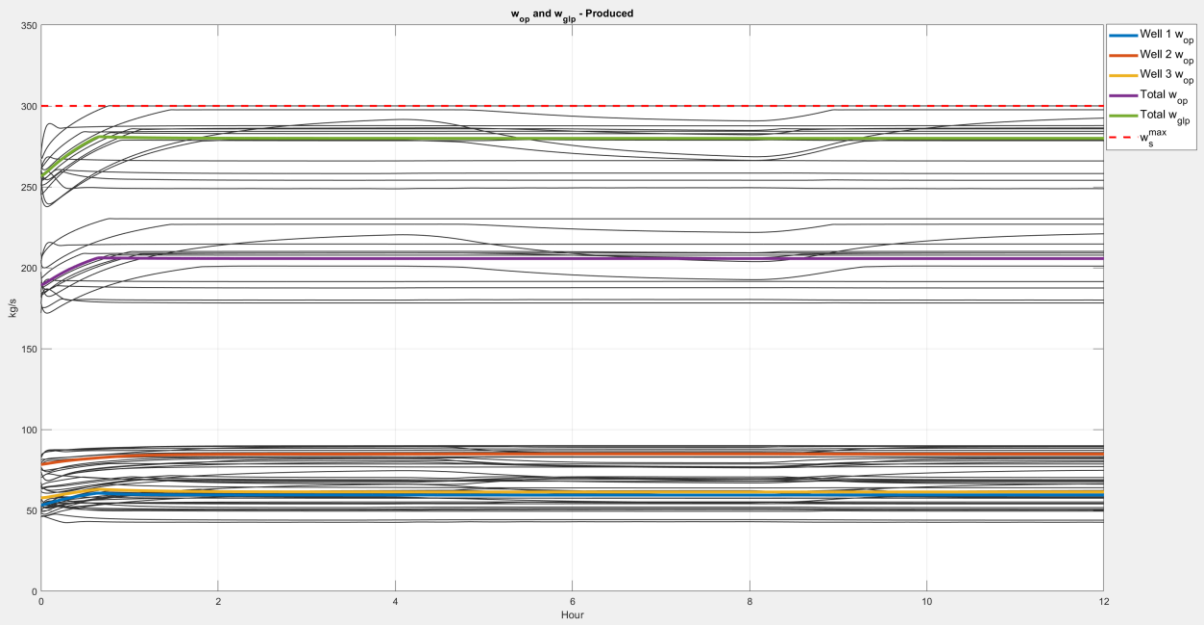

Figure 5.2: Simulation outputs.

The inputs are shown in Figure 5.3. Compared to section 4.4 it is less need for injection gas and the change in w_{gc}^{max} at 4 h are not affecting all the simulations. The bounds for w_{ga} are violated for some of the simulations. Figure 5.4 shows the optimizing time for each simulation.

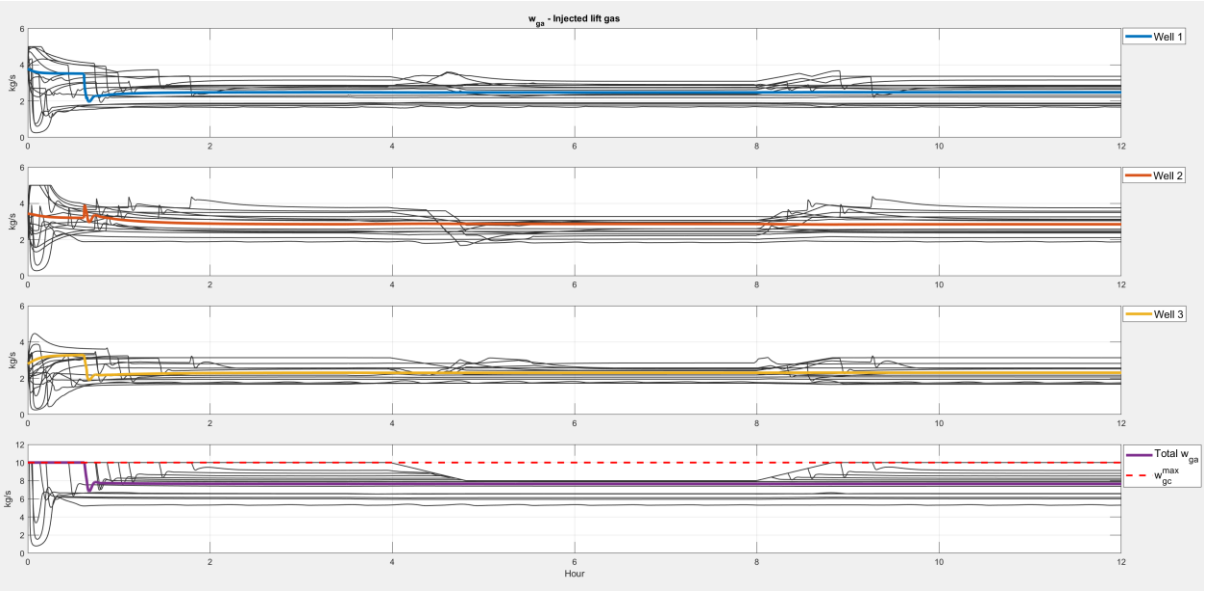

Figure 5.3: Simulation inputs.

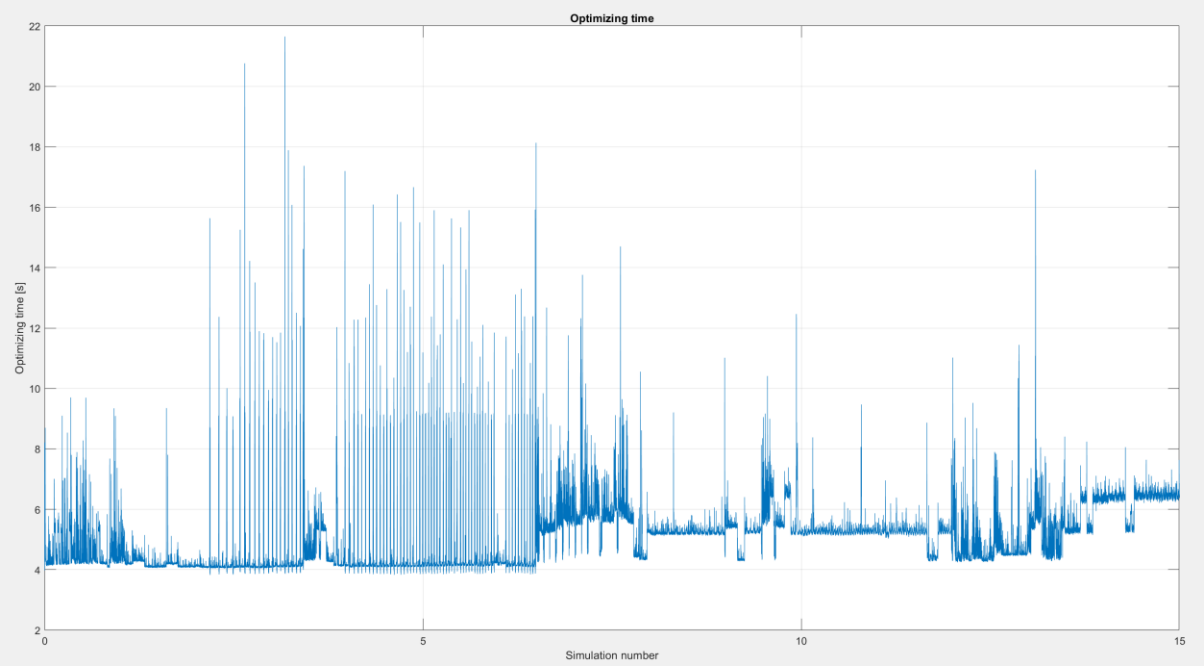

Figure 5.4: Simulation optimizing time.

5.2.2 Simulation with short prediction horizon

There is studied four different simulation cases with prediction horizon N = 25, which is short in this context. The MPC formulation in equation (5.1) - (5.8) is used.

1. Three scenarios without grouping. Figure 5.5 shows outputs and Figure 5.6 inputs.
2. Three scenarios with grouping. Figure 5.7 shows outputs and Figure 5.8 inputs.
3. 10 scenarios without grouping. Figure 5.9 shows outputs and Figure 5.10 inputs.

4. 10 scenarios with grouping. Figure 5.11 shows outputs and Figure 5.12 inputs.

Optimizing time for the different cases are shown in Table 5.4.

Table 5.4: Mean optimizing time.

Scenarios	Grouping	No grouping
3	0,55 s	0,91 s
10	1,82 s	7,6 s

The results shows that the MPC works with a short prediction horizon in this case. There is no significant different between the simulations. Increasing scenarios and don't group w_{ga} shows no significant improvement, but the optimizing time is increasing significant.

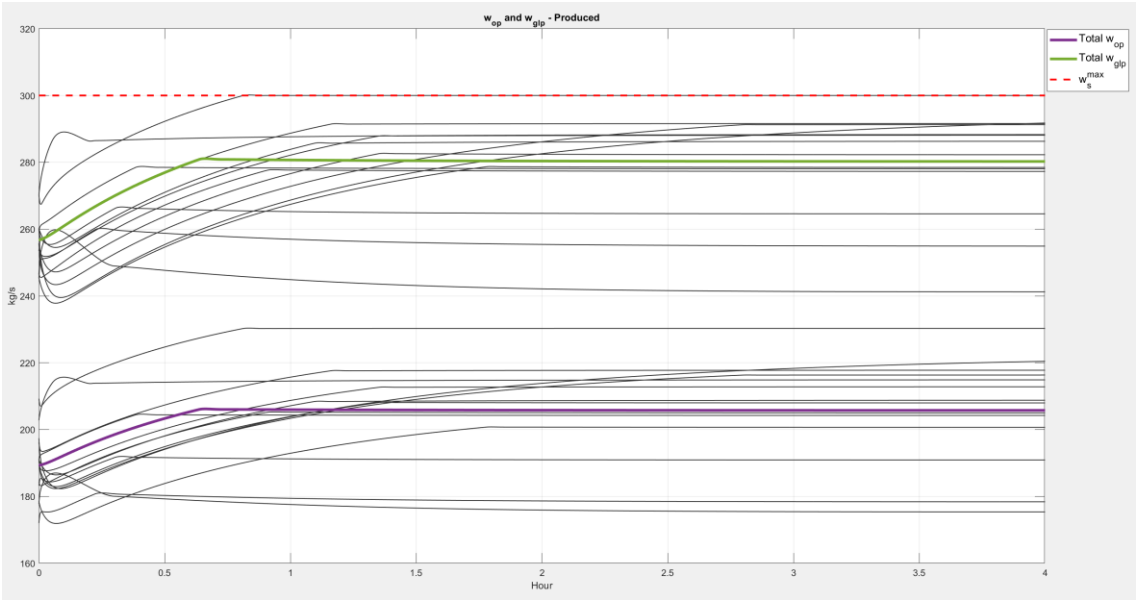

Figure 5.5: Three scenarios without grouping – outputs.

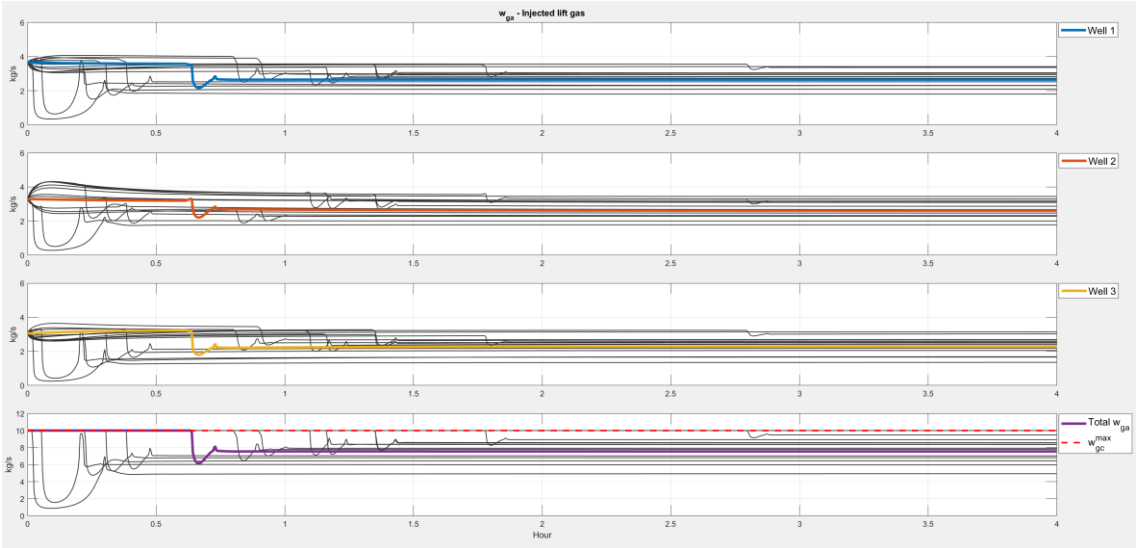

Figure 5.6: Three scenarios without grouping – inputs.

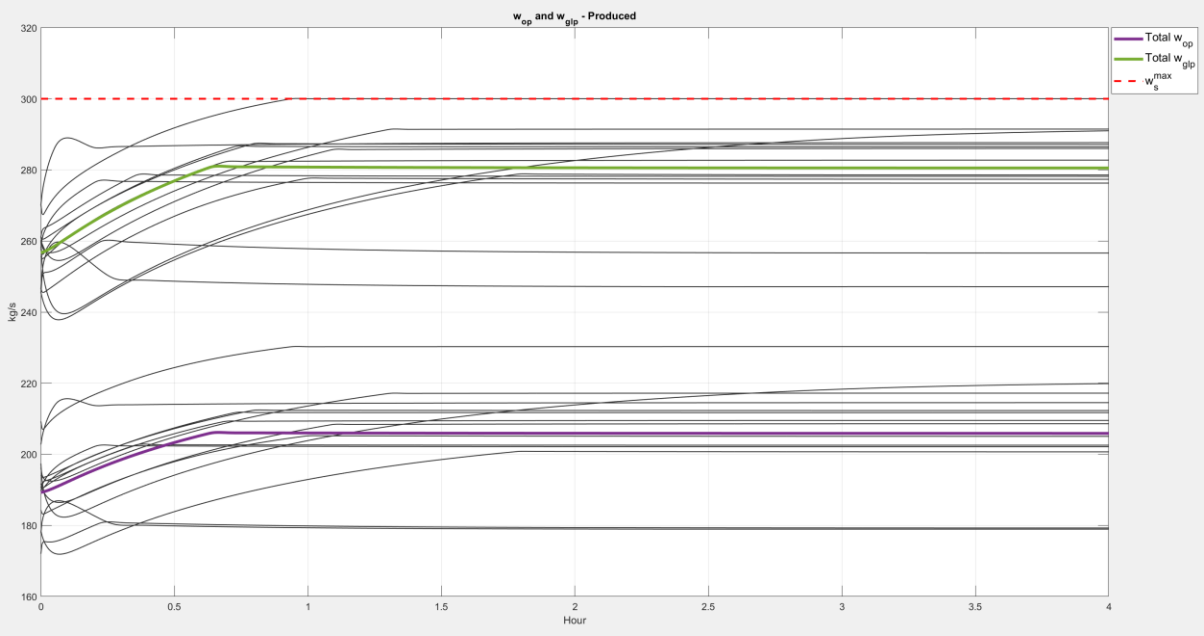

Figure 5.7: Three scenarios with grouping – outputs.

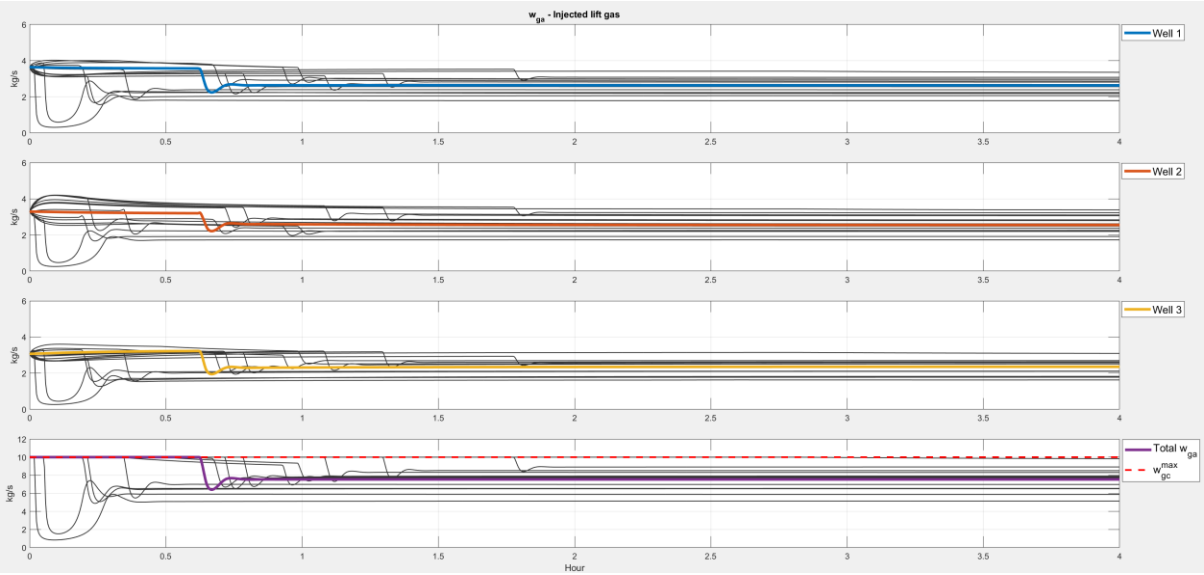

Figure 5.8: Three scenarios with grouping – inputs.

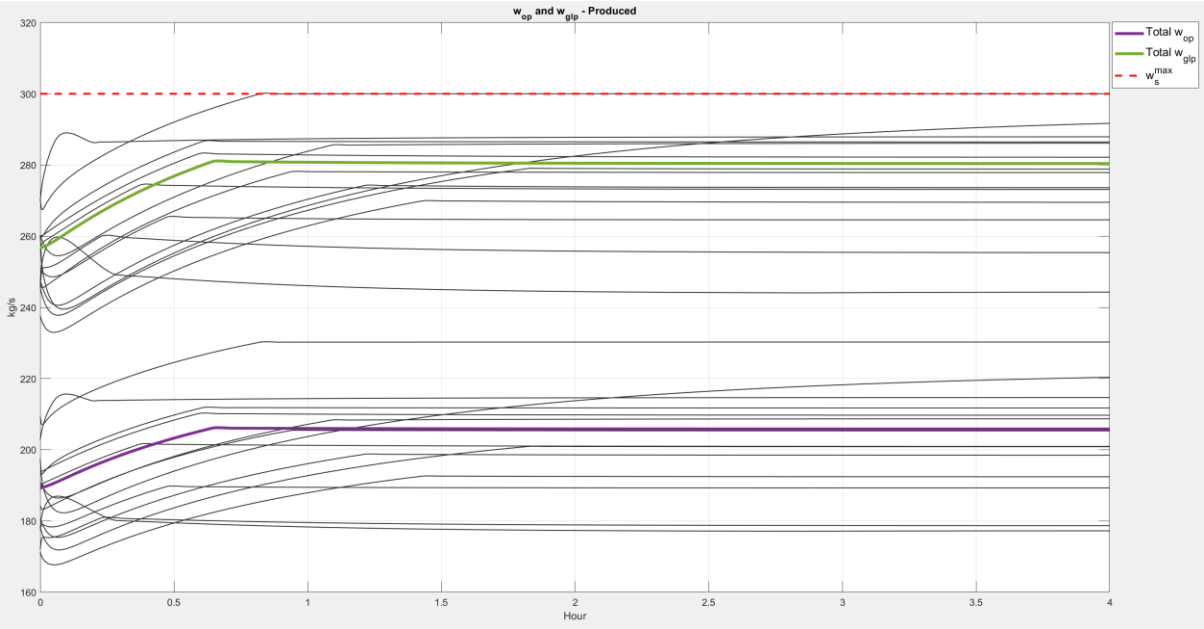

Figure 5.9: 10 scenarios without grouping – outputs.

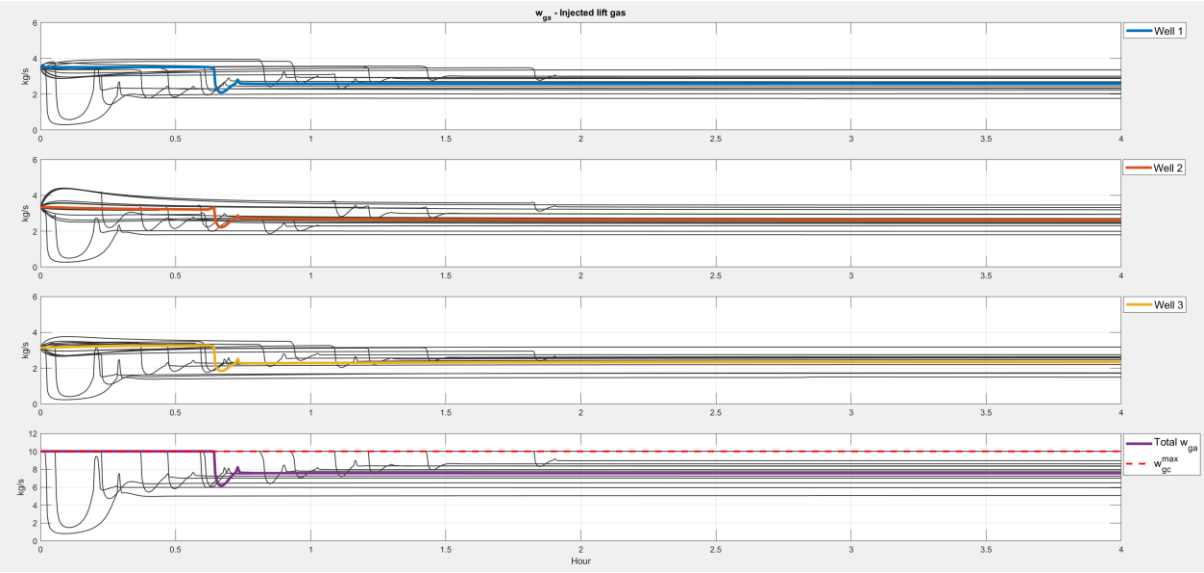

Figure 5.10: 10 scenarios without grouping – inputs.

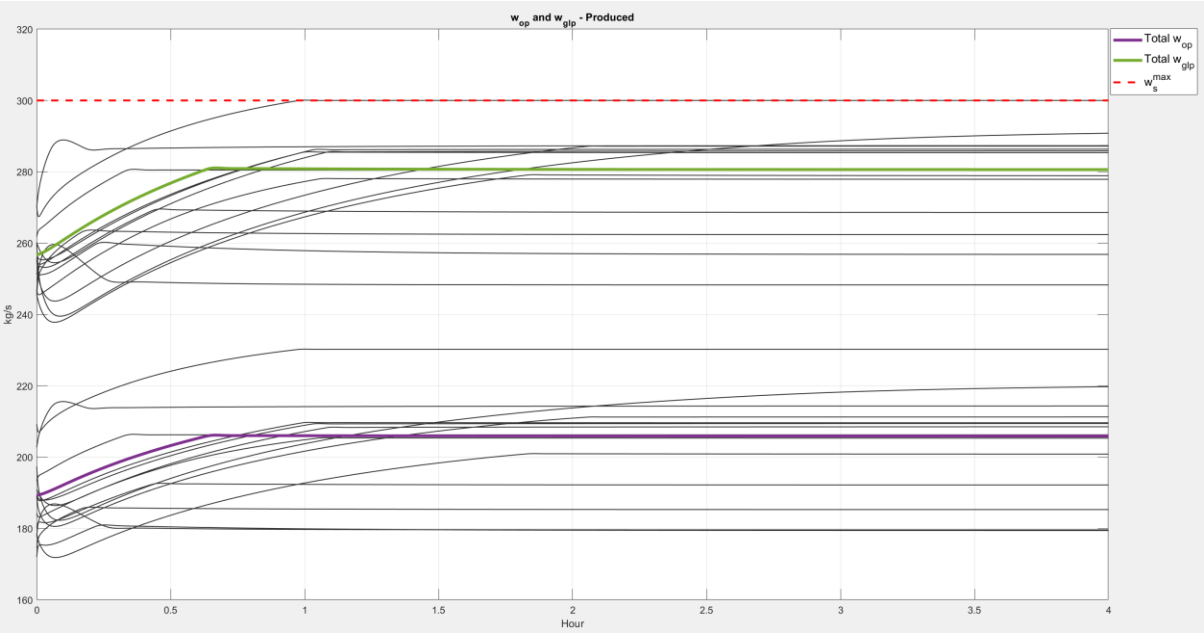

Figure 5.11: 10 scenarios with grouping – outputs.

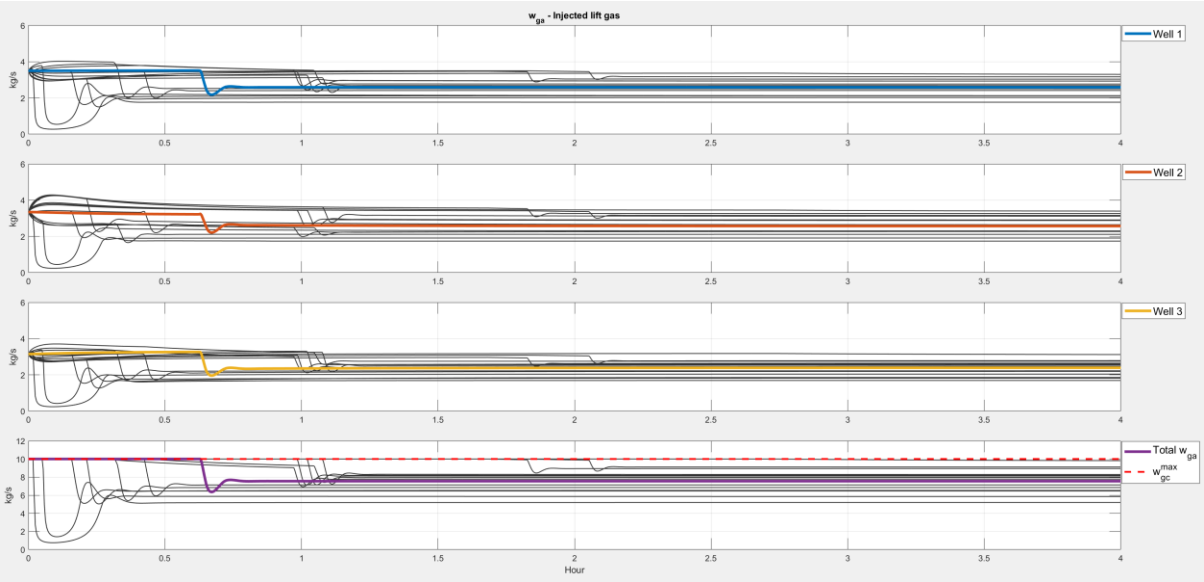

Figure 5.12: 10 scenarios with grouping – inputs.

5.2.3 Robust MPC with soft constraint

This simulation is performed with the IPOPT solver and the prediction horizon, N = 50 and 10 scenarios. The MPC formulation in equation (5.9) - (5.18) is used. Figure 5.13 shows the outputs and Figure 5.14 shows the inputs when grouping w_{ga}. The simulations are feasible at all times. The mean optimizing time is 4,1 s. Because of the uncertain parameters the w_s^{offset} is 10 kg/s. Lowering the w_s^{offset} cause that some of the uncertain models touch the w_s^{max} and start to oscillate.

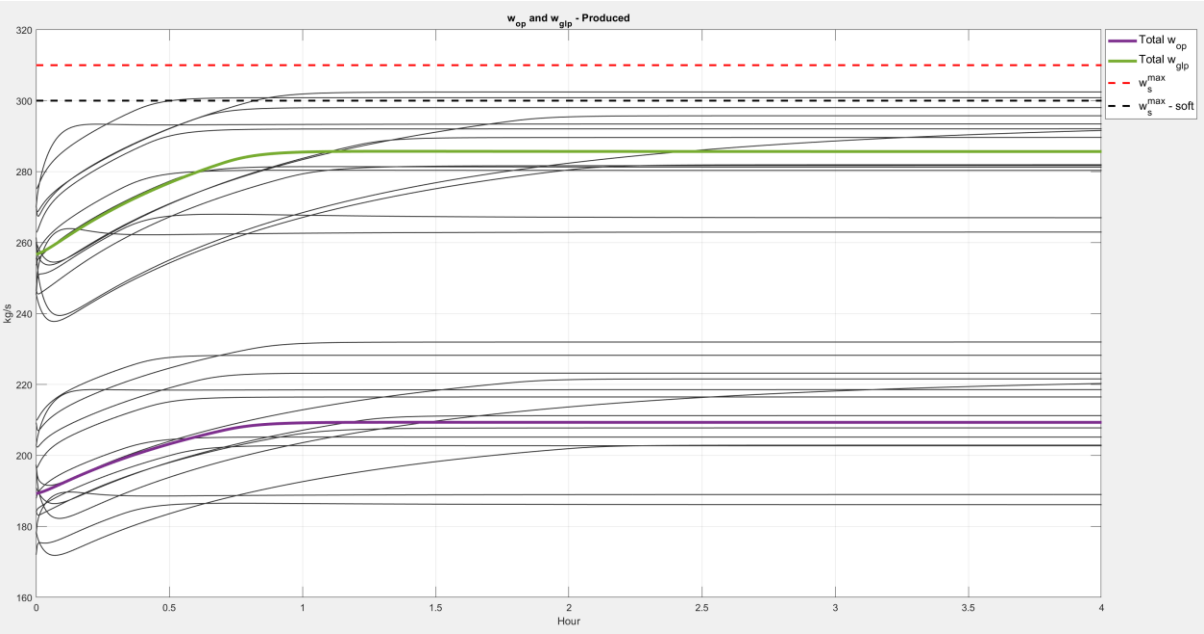

Figure 5.13: Simulation with grouping – outputs.

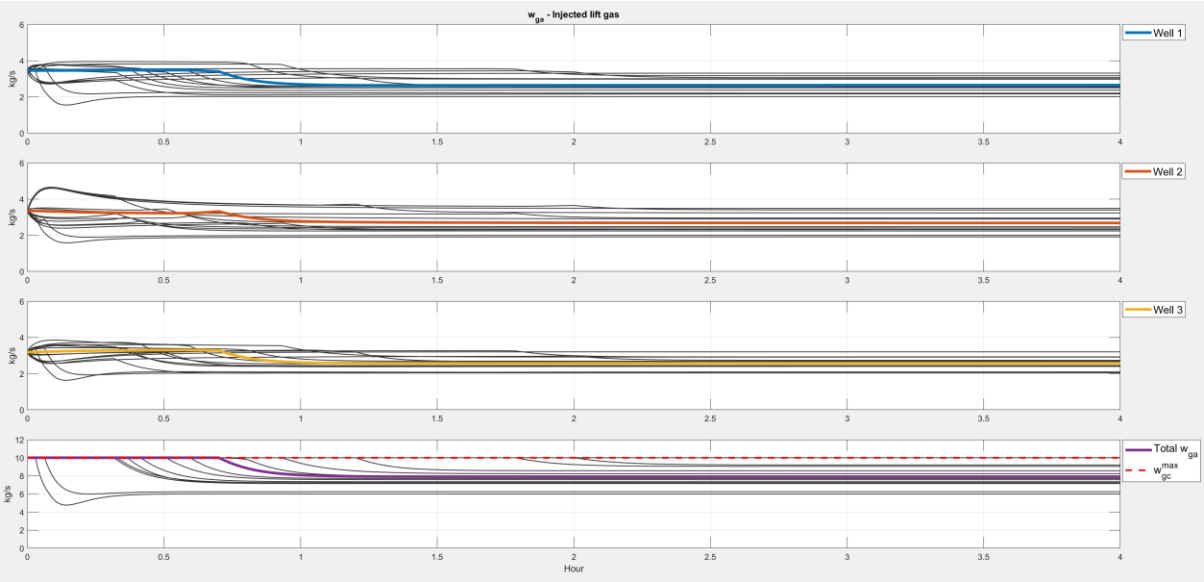

Figure 5.14: Simulation with grouping – inputs.

The same simulation is performed without grouping w_{ga}. Figure 5.15 shows the outputs and Figure 5.16 shows the inputs. The mean optimizing time is 7,6 s for each iteration. There is no significant improvement when not grouping w_{ga}.

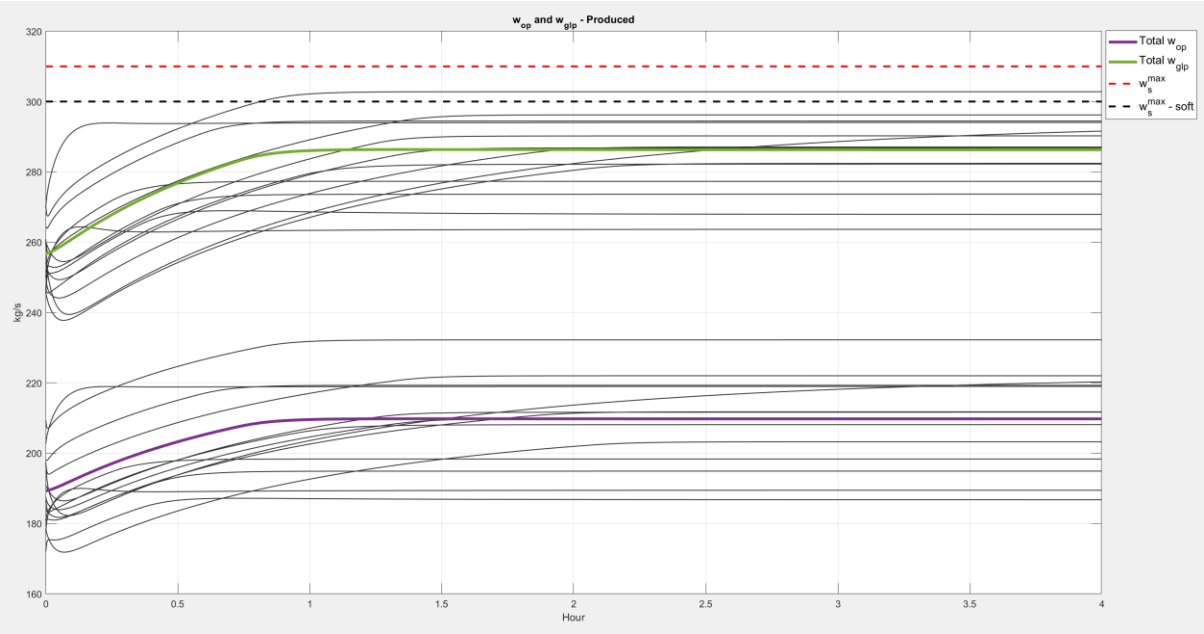

Figure 5.15: Simulation without grouping – outputs.

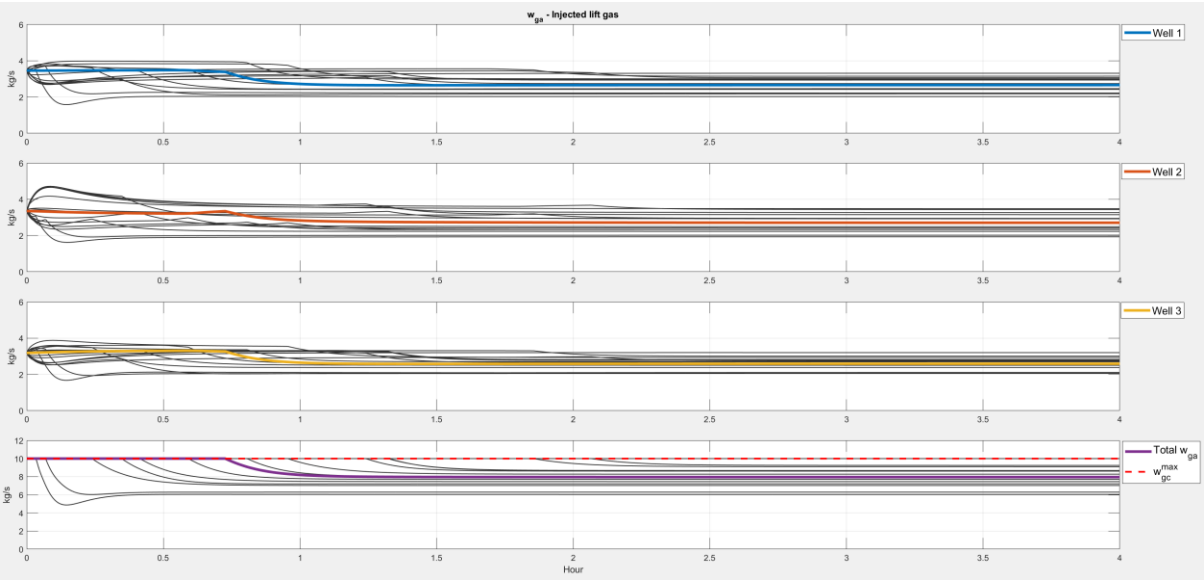

Figure 5.16: Simulation without grouping – inputs.

The same simulation is done with the qpOASES solver and with grouping the inputs to see the difference between the solvers. The result is very similar when using IPOPT shown in Figure 5.13 and Figure 5.14. The main different is the mean optimizing time. It is 21,0 s which is 16,9 s higher than using IPOPT.

5.3 Robustness and conservativeness

Simulations in section 5.2 shows that the designed robust MPC is robust. It does not violate the constraints. Although there is also a degree of conservativeness. The robust MPC applied to the different models gives less produced oil than deterministic MPC applied to nominal model.

Increasing scenarios, not grouping w_{ga}, or changing the prediction horizon slightly affect the conservativeness. Using IPOPT solver also require that a soft constraint is added. This makes the MPC even more conservative.

The weightings of the scenarios are assumed to be equal at this point since no other information exists. If sampling more data about the uncertainties the scenarios and weighting ω can be chosen carefully to maintain robustness and reduce conservativeness [6]. An approach is to use data-driven scenario selection [17]. This approach can also be used to provide a weighting ω for the scenarios.

6 Discussion

The thesis started with literature review, which is the basic of the thesis. Continued to present the nonlinear model of a gas lifted oil field. This model was used as a simulator throughout the thesis. The linear model, which is the base of the linear MPC was derived of the nonlinear model. Open loop simulation comparing the linear model and the nonlinear model showed deviation between the nonlinear model and the linear model when moving from the operating point. To minimize the deviation the linear model was updated at every iteration. Consequently, the deviation was removed in steady state, but the models where acting different to change in input. The objective is to develop a linear MPC for oil production optimization such that it is robust to uncertainties. In the light of that it was assumed that it is more important that the linear model do not deviate from the nonlinear model in steady state. Especially since the inlet separator must not be overloaded.

The MPC is designed stepwise. Started as a deterministic linear MPC and developed further to a robust multi-stage linear MPC. This also means that there is no general MATLAB-code that handles all design phases, but the different codes present in the thesis is based on previous codes. The stepwise approach has led to different design of the MPC. The deterministic MPC was first designed in CasADi low-level interface. This design worked but has its limitations. A result of the large z-vector to optimize, the possible prediction horizon was strictly limited. The CasADi high-level interface open more possibilities such as grouping the input and changing the solver. It showed to be much more efficient than the low-level interface. The way CasADi build the problem symbolically, makes it easy to change and extend the MPC.

It has been a focus to design the MPC to run as efficient as possible. If the MPC is implemented to a real process it must run in real-time. This focus has led to a more efficient MPC. Methods attempt to decrease the optimizing time are grouping inputs, testing different solvers and testing different number of scenarios. Early in the thesis the IPOPT showed to be slower than the qpOASES. When the problem increased significant when designing the multi-stage MPC the IPOPT showed to be much faster than qpOASES. Although the MPC with the IPOPT required a soft constraint in addition. This comes with a price in decreased w_{op}.

The model of the gas lifted oil field seems to be stiff. Simulation parameters are chosen carefully. When attempting to increase the time step more than 15 s it loses information and act bad. When lowering the input to near zero it also acts bad.

In some simulations the solver reports an error message that the problem is infeasible. This only happens in the MPC's without soft constraint. A hypothesis is that this is caused by the linearization of the nonlinear model where the linear model is updated every time step. Open loop simulation in section 3.3.2 shows overshooting when stepping the input. This might cause the error message.

All the simulations are performed on a laptop and not on a dedicated CPU. The optimizing time present in the thesis might be influenced by other software on the computer. The optimizing time present, still gives a good picture of how efficient the different MPC's are.

7 Conclusion

The nonlinear model of the gas lifted oil field is possible to linearize. Open loop simulations show that the linear model acting like the nonlinear model. The deviation between the models is taken care of by updating the linear model at every time step. Using this approach in linear MPC, helps to satisfy constraints.

Stochastic analysis of deterministic linear MPC shows that there is no guarantee that the deterministic MPC don't violate constraints and overload the separator. Therefore, the robust MPC is required under the presence of uncertainties.

It is designed two variants of the robust multi-stage linear MPC. One with only hard constraint at w_s^{max} and one with a soft constraint "w_s^{max} – soft" below the w_s^{max}. The first MPC works, but under some circumstances it is infeasible. Bounds can be violated, and the inputs can oscillate. Therefore, the soft constraint is introduced. Then the simulation is feasible at all times, and constraints and bounds are satisfied. Introducing the soft constraint makes the MPC slightly more conservative. The robustness is taken care of since the hard constraint w_s^{max} is still operative.

The robust multi-stage linear MPC with an extra soft constraint below w_s^{max} using the IPOPT solver is a robust and efficient MPC.

Attempt to make the MPC less conservative by increasing scenarios and not grouping inputs was unsuccessful. Three scenarios and grouping input into three groups are appropriate.

Using robust multi-stage linear MPC on the model of gas lifted oil field gives promising results. In further work the robust multi-stage linear MPC with the solver IPOPT should be looked more into. A state observer could be introduced instead of assuming full state information.

References

[1] S. Lucia, T. Finkler and S. Engell, "Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty," *Journal of Process Controll*, vol. 23, nr. 9, aug. 2013, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0959152413001686?via%3Dihub

[2] R. Sharma, *"Lecture notes for the course IIA 4117: Model Predictive Control"*, University of South-Eastern Norway, 2019.

[3] S. Lucia, *"Robust Multi-stage Nonlinear Model Predictive Control,"* Ph.D dissertation, Fakultät Bio- und Chemieingenieurwesen der Technischen Universität, Dortmund, Germany, 2015.

[4] Do-MPC, *"Robust horizon"*, 2021, https://www.do-mpc.com/en/latest/theory_mpc.html#robust-horizon Accessed on: 25.08.2022

[5] N. Janatian, K. R. Jayamanne and R. Sharma. "Model Based Control and Analysis of Gas Lifted Oil Field for Optimal Operation," Department of Electrical Engineering, IT and Cybernetics, USN, Norway, Oct. 11, 2021.

[6] K. Jayamanne, "Optimal Operation of Processes Under Uncertainty Using Robust Model Predictive Control", Master's Thesis, Master of Science, Industrial IT and Automation, University of South-Eastern Norway, Porsgrunn, 2021.

[7] CasADi, "*CasADi"*, 2018 https://web.casadi.org/ Accessed on: 28.01.2022

[8] J. A. E Andersson, J. Gillis, G. Horn, J. B. Rowlings and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol 11, nr. 1, 2019. doi: 10.1007/s12532-018-0139-4.

[9] MathWorks, *Symbolic Math Toolbox*, 2022 https://se.mathworks.com/products/symbolic.html Accessed on: 25.02.2022

[10] MathWorks, *"jacobian"*, 2022 https://se.mathworks.com/help/symbolic/sym.jacobian.html Accessed on: 03.03.2022

[11] CasADi, *"CasADi - Docs"*, 2018 https://web.casadi.org/docs/#quadratic-programming Accessed on: 28.01.2022

[12] H.J Ferreau, "qpOASES User's Manual", ABB Corporate Research, Switzerland, Version 3.2, April 2017. Accessed: 01.04.2022. [Online]. Available: https://www.coin-or.org/qpOASES/doc/3.2/manual.pdf

[13] YouTube, *"MPC and MHE implementation in Matlab using Casadi | Part 1"*, 2019 https://www.youtube.com/watch?v=RrnkPrcpyEA Accessed on: 29.05.2022

[14] A. Wächter and L. T. Biegler, "On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming," *Mathematical Programming*, 106, s. 25-27, Mar 2006. doi: 10.1007/s10107-004-0559-y

[15] P. Goel, A. Datta and M.S. Mannan, "Industrial alarm systems: Challenges and opportunities", *Journal of Loss Prevention in the Process Industries*, vol. 50, part A, Nov. 2017. doi: 10.1016/j.jlp.2017.09.001

[16] Do-MPC, *"Robust multi-stage NMPC",* 2021, https://www.do-mpc.com/en/latest/theory_mpc.html#robust-multi-stage-nmpc Accessed on: 25.08.2022

[17] D. Krishnamoorthy, M. Thombre, S. Skogstad and J. Jäschke, "Data-driven Scenario Selection for Multistage Robust Model Predictive Control," *IFAC-PapersOnLine,* vol. 51, issue 20, nov. 2018. doi: 10.1016/j.ifacol.2018.11.046

Appendices

Appendix A Project topic description.

Appendix B Modeling of Gas Lifted Oil field with five oil wells considering water cut.

Appendix C MATLAB code: Open loop simulation of nonlinear model.

Appendix D MATLAB function, *functionDAE_multi*: Nonlinear model.

Appendix E MATLAB function, *generateInput*: Generate inputs to Open loop simulation.

Appendix F MATLAB function, *updateState*: Update states and outputs of the nonlinear model using Runge-Kutta 4^{th} order method.

Appendix G MATLAB function, *CalcJac_y*: Create function of system matrices and operating point vectors for open loop simulation.

Appendix H Example of the system matrices in continues time.

Appendix I MATLAB function, *UpdateMatrices_op_Open*: Substitute states into system matrices and operating point vectors in open loop simulation.

Appendix J MATLAB code: Open loop simulation comparing nonlinear model and linear model with static system matrices.

Appendix K MATLAB function, *linearModel*: Updating states of linear model.

Appendix L MATLAB code: Open loop simulation comparing nonlinear model and linear model with updated system matrices.

Appendix M MATLAB function, *MPC_low*: Deterministic linear MPC low-level interface.

Appendix N MATLAB code: Closed loop simulation with deterministic MPC.

Appendix O MATLAB function, *Jac_func_op*: Create system matrices and operating point vectors for closed loop simulation.

Appendix P MATLAB function, *UpdateMatrices_op*: Substitute states into system matrices and operating point vectors in closed loop simulation.

Appendix Q MATLAB function, *injGas*: Generate w_{gc}^{max}.

Appendix R MATLAB function, *MPC_high*: Deterministic linear MPC high-level interface.

Appendix S MATLAB function, *MPC_high_slack*: Deterministic linear MPC high-level interface with soft constraint.

Appendix T MATLAB code: Closed loop, deterministic MPC and uncertain model.

Appendix U MATLAB function, *simulation_unc*: Simulation interface deterministic MPC and uncertain model.

Appendix V MATLAB function, *MS_MPC*: Robust Multi-stage linear MPC.

Appendix W MATLAB function, *MS_MPC_slack*: Robust Multi-stage linear MPC with soft constraint.

Appendix X MATLAB function, *MS_Scenarios*: Building system matrices and operating point vectors for each scenario used in the multistage MPC.

Appendix Y MATLAB code: Closed loop, multi-stage MPC and uncertain models.

Appendix Z MATLAB function, *MS_simulation_unc*: Simulation interface multi-stage MPC and uncertain model.

Appendix AA MATLAB function, *MS_UpdateMatrices_op*: Substitute states into system matrices and operating point vectors in multi-stage closed loop simulation.

University of South-Eastern Norway

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

FMH606 Master's Thesis

Title: Robust MPC for optimal oil production under the presence of uncertainties

USN supervisor: Roshan Sharma (USN)

External partner: Equinor

Task background:

Preface: This thesis is related to an on-going project "Digiwell" in collaboration with Equinor (and other partners like SINTEF, Imperial College London and MIT) where 4 PhD students are involved.

Background: 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. Oil production processes are subjected to different kinds of uncertainty, from uncertain values of some parameters in the model, uncertain measurements, 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 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.

Aim:

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. The developed robust MPC should be applied to an oil production case study (for e.g. gas lifted oil field or ESP lifted oil field). The mathematical model of the case study (together with full process description) will be provided to the students by the supervisor.

Task description:

The following are the main tasks:

(i) Detailed literature review on multi-stage MPC as the method for Robust/Stochastic MPC.

(ii) Develop linear MPC for oil production optimization such that it is robust to uncertainties present in the system. At least the parametric uncertainty should be taken into account. An oil production case study (gas lifted oil field or ESP lifted oil field) should be used.

(iii) Study about conservativeness and robustness of the developed MPC through detailed simulations for different operational scenarios.

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

Student category: Reserved

Reserved for online master student Ragnvald Leyser Gulløy

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

Practical arrangements:

N/A.

Signatures:

Supervisor (date and signature): 01.02.22 

Student (write clearly in all capitalized letters): RAGNVALD LEYSER GULLØY

Student (date and signature): 01.02.22 

Modeling of a Gas Lifted Oil field with five oil wells considering water cut:

1. Differential Equations

Mass balance in annulus:

$$\dot{m}_{ga}^{i} = w_{ga}^{i} - w_{ginj}^{i} \qquad i = 1,2,3,4,5 \qquad (1)$$

where

Variable	Description	Type
m_{ga}^{i}	Mass of the gas in the annulus	State
w_{ga}^{i}	Mass flow rate of the injected lift gas into the annulus	Input
w_{ginj}^{i}	Mass flow rate of the gas from the annulus into the tubing	Alg. Var.

Mass balance for the gas in tubing and above the injection point:

$$\dot{m}_{gt}^{i} = w_{ginj}^{i} + w_{gr}^{i} - w_{gp}^{i} \qquad i = 1,2,3,4,5 \qquad (2)$$

where

Variable	Description	Type
m_{gt}^{i}	Mass of the gas in the tubing above the injection point	State
w_{gr}^{i}	Mass flow rate of the gas from the reservoir into the tubing	Alg. Var.
w_{gp}^{i}	Mass flow rate of the gas through the production choke valve	Output

Mass balance for the liquid phase in tubing and above the injection point:

$$\dot{m}_{lt}^{i} = w_{lr}^{i} - w_{lp}^{i} \qquad i = 1,2,3,4,5 \qquad (3)$$

where

Variable	Description	Type

m_{lt}^i	Mass of the liquid in the tubing above the injection point	State
w_{lr}^i	Mass flow rate of the liquid phase from the reservoir into the tubing	Alg. Var.
w_{lp}^i	Mass flow rate of the liquid phase through the production choke valve	Output

2. Algebraic Equations

Pressure of gas in the annulus downstream the lift gas choke valve:

$$P_a^i = \frac{Z_{P_a^i} m_{ga}^i R T_a^i}{M A_a^i L_{a.tl}^i} \qquad i = 1,2,3,4,5 \qquad (4)$$

where

Variable	Description	Type
P_a^i	Pressure of gas in the annulus downstream the lift gas choke valve	Alg. Var.
$Z_{P_a^i}$	Compressibility factor	Constant
R	Gas constant	Constant
T_a^i	Temperature in the annulus	Constant
M	Molar mass	Constant
A_a^i	Annulus cross section area	Constant
$L_{a.tl}^i$	Total length of the annulus above the injection point	Constant

Pressure of gas in the annulus upstream the gas injection valve:

$$P_{ainj}^i = P_a^i + \frac{m_{ga}^i}{A_a^i L_{a.tl}^i} g L_{a.vl}^i \qquad i = 1,2,3,4,5 \qquad (5)$$

where

Variable	Description	Type

P^i_{ainj}	Pressure of gas in the annulus upstream the gas injection valve	Alg. Var.
g	Gravitational acceleration constant	Constant
$L^i_{a.vl}$	Vertical length of the annulus above the injection point	Constant

Density of the liquid based on the water cut:

$$\rho_l^i = \rho_w WC^i + \rho_o^i\left(1 - WC^i\right) \qquad i = 1,2,3,4,5 \qquad (6)$$

where

Variable	Description	Type
ρ_l^i	Density of the liquid based on the water cut	Constant
ρ_w	Density of water	Constant
WC^i	Water cut	Constant
ρ_o^i	Density of oil	Constant

The volume of gas present in the tubing above the gas injection point:

$$V_G^i = A_t^i L_{t.tl}^i - \frac{m_{lt}^i}{\rho_l^i} \qquad i = 1,2,3,4,5 \qquad (7)$$

where

Variable	Description	Type
V_G^i	The volume of gas present in the tubing above the gas injection point	Alg. Var.
A_t^i	Annulus cross section area	Constant
$L_{t.tl}^i$	Total length of the tubing above the injection point	Constant

The average density of the mixture of liquid and gas in the tubing above the injection point:

$$\rho_m^i = \frac{m_{gt}^i + m_{lt}^i}{A_t^i L_{t.tl}^i} \qquad i = 1,2,3,4,5 \qquad (8)$$

where

Variable	Description	Type
ρ_m^i	The average density of the mixture of liquid and gas in the tubing above the injection point	Alg. Var.

Pressure in the tubing downstream the gas injection valve:

$$P_{tinj}^i = \frac{Z_{P_G^i} m_{gt}^i R T_t^i}{M V_G^i} + \frac{1}{2}\rho_m^i g L_{t.vl}^i \qquad i = 1,2,3,4,5 \qquad (9)$$

where

Variable	Description	Type
P_{tinj}^i	Pressure in the tubing downstream the gas injection valve	Alg. Var.
T_t^i	Temperature in the tubing	Constant
$L_{t.vl}^i$	Vertical length of the tubing above the injection point	Constant

Gas expansion factor in the gas injection valve:

$$Y_2^i = 1 - \alpha_Y \left(\frac{P_{ainj}^i - P_{tinj}^i}{\max\left(P_{ainj}^i - P_{ainj}^{min}\right)} \right) \qquad i = 1,2,3,4,5 \qquad (10)$$

where

Variable	Description	Type
Y_2^i	Gas expansion factor in the gas injection valve	Alg. Var.
α_Y	Constant	Constant

Density of gas in the annulus:

$$\rho_{ga}^{i} = \frac{m_{ga}^{i}}{A_{a}^{i} L_{a,tl}^{i}} \qquad i = 1,2,3,4,5 \qquad (11)$$

where

Variable	Description	Type
ρ_{ga}^{i}	Density of gas in the annulus	Alg. Var.

The mass flow rate of the gas injected into the tubing from the annulus:

$$w_{ginj}^{i} = k^{i} Y_{2}^{i} \sqrt{\rho_{ga}^{i} \max\left(P_{ainj}^{i} - P_{tinj}^{i}, 0\right)} \qquad i = 1,2,3,4,5 \qquad (12)$$

where

Variable	Description	Type
w_{ginj}^{i}	The mass flow rate of the gas injected into the tubing from the annulus	Alg. Var.
k^{i}	Gas injection valve constant	Constant

The bottom hole pressure or well flow pressure:

$$P_{wf}^{i} = P_{tinj}^{i} + \rho_{l}^{i} g L_{r.vl}^{i} \qquad i = 1,2,3,4,5 \qquad (13)$$

where

Variable	Description	Type
P_{wf}^{i}	The bottom hole pressure or well flow pressure	Alg. Var.
$L_{r.vl}^{i}$	Vertical length of the tubing below the injection point	Constant

The mass flow rate of the liquid from the reservoir:

$$w_{lr}^{i} = PI^{i}\max\left(P_{r}^{i} - P_{wf}^{i}, 0\right) \qquad i = 1,2,3,4,5 \qquad (14)$$

where

Variable	Description	Type
w_{lr}^{i}	The mass flow rate of the liquid from the reservoir	Alg. Var.
PI^{i}	Productivity index	Constant
P_{r}^{i}	Pressure of the reservoir	Constant

The mass flow rate of oil from the reservoir:

$$w_{or}^{i} = \frac{\rho_{o}^{i}}{\rho_{l}^{i}}\left(1 - WC^{i}\right)w_{lr}^{i} \qquad i = 1,2,3,4,5 \qquad (15)$$

where

Variable	Description	Type
w_{or}^{i}	Oil compartment mass flow rate from liquid comes out from reservoir	Alg. Var.

The mass flow rate of gas from the reservoir:

$$w_{gr}^{i} = GOR^{i}w_{or}^{i} \qquad i = 1,2,3,4,5 \qquad (16)$$

where

Variable	Description	Type
w_{gr}^{i}	The mass flow rate of gas from the reservoir	Alg. Var.
GOR^{i}	Gas to oil ratio	Constant

Pressure in the tubing upstream the production choke valve:

$$P_{wh}^{i} = \frac{Z_{P_G^i} m_{gt}^{i} R T_t^{i}}{M V_G^{i}} - \frac{1}{2}\rho_m^{i} g L_{t.vl}^{i} \qquad i = 1,2,3,4,5 \qquad (17)$$

where

Variable	Description	Type
P_{wh}^{i}	Well head pressure	Alg. Var.

Gas expansion factor in the production choke valve:

$$Y_3^{i} = 1 - \alpha_Y \left(\frac{P_{wh}^{i} - P_s}{\max\left(P_{wh}^{i} - P_{wh}^{min}\right)} \right) \qquad i = 1,2,3,4,5 \qquad (18)$$

where

Variable	Description	Type
Y_3^{i}	Gas expansion factor in the production choke valve	Alg. Var.
P_s	Separator pressure	Constant

The mass flow rate of the mixture of gas and liquid from the production choke valve:

$$w_{glp}^{i} = C_v Y_3^{i} \sqrt{\rho_m^{i} \max\left(P_{wh}^{i} - P_s, 0\right)} \qquad i = 1,2,3,4,5 \qquad (19)$$

where

Variable	Description	Type
w_{glp}^{i}	The mass flow rate of the mixture of gas and liquid from the production choke valve	Alg. Var.
C_v	Valve characteristics as a function of its opening (this valve is always 100% open)	Constant

Mass flow rate of gas through the production choke valve:

$$w_{gp}^{i} = \frac{m_{gt}^{i}}{m_{gt}^{i} + m_{lt}^{i}} w_{glp}^{i} \qquad i = 1,2,3,4,5 \qquad (20)$$

where

Variable	Description	Type
w_{gp}^{i}	Mass flow rate of gas through the production choke valve	Alg. Var.

Mass flow rate of liquid through the production choke valve:

$$w_{lp}^{i} = \frac{m_{lt}^{i}}{m_{gt}^{i} + m_{lt}^{i}} w_{glp}^{i} \qquad i = 1,2,3,4,5 \qquad (21)$$

where

Variable	Description	Type
w_{lp}^{i}	Mass flow rate of liquid through the production choke valve	Alg. Var.

Oil compartment mass flow rate from liquid product considering water cut:

$$w_{op}^{i} = \frac{\rho_{o}^{i}}{\rho_{l}^{i}} \left(1 - WC^{i}\right) w_{lp}^{i} \qquad i = 1,2,3,4,5 \qquad (22)$$

where

Variable	Description	Type
w_{op}^{i}	Oil compartment mass flow rate from liquid product considering water cut	Alg. Var.

Appendix C


```
%Open loop simulation of nonlinear model

%-----------------------------------------------------------------------

clear
close all

% States initial values
% when w_ga = [2.2 2.5 1.8]
m_ga = [9.0127e3    9.8391e3    7.2142e3];
m_gt = [1.4847e3    1.52e3      1.1427e3];
m_lt = [2.1814e4    1.7254e4    2.5813e4];

state_ini = [m_ga m_gt m_lt];

% Simulation parameters
dt = 15 ; %[s]
hour = 60; %[h]        hours of simulationtime
Np = 3600*hour/dt;     %Number of datapoints
t=linspace(0,hour,Np); %Used for plotting
l = length(state_ini); %Number of states

%Input
%Generate input for whole horizon
w_ga = generateInput(Np);

Pc = zeros(Np,l);
P_out = zeros(Np,9);

for i = 1:Np

    %store the states
    Pc(i,:) = state_ini;

    %update the state
    %with RK method
    [x_next, out] = updateState(state_ini,dt,w_ga(i,:),1,1,1);
    state_ini = x_next;
    P_out(i,:) = out(:,1:9); %Storing outputs
end


%% Plotting

width = 2;

tiledlayout(4,1);
ax1 = nexttile;
plot(ax1,t,P_out(:,1),t,P_out(:,2),t,P_out(:,3), 'LineWidth',width);
title(ax1,'P_w_f - Bottom hole pressure');
ylabel(ax1,'bar');
xlabel(ax1,'Hour');
ylim([120 150]);
legend({'Well 1','Well 2','Well 3'}, 'FontSize',12);

grid('on');

ax2 = nexttile;
plot(ax2,t,P_out(:,4),t,P_out(:,5),t,P_out(:,6), 'LineWidth',width);
title(ax2,'P_w_h - Well head pressure');
ylabel(ax2,'bar');
xlabel(ax2,'Hour');
ylim([30 40]);
grid('on');

ax3 = nexttile;
plot(ax3,t,P_out(:,7),t,P_out(:,8),t,P_out(:,9), 'LineWidth',width);
title(ax3,'w_l_p - Liquid produced');
ylabel(ax3,'kg/s');
xlabel(ax3,'Hour');
ylim([50 100]);
grid('on');

ax4 = nexttile;
```

```
plot(ax4,t,w_ga, 'LineWidth',width);
title(ax4,'w_g_a - Injected lift gas');
ylabel(ax4,'kg/s');
xlabel(ax4,'Hour');
ylim([1 5]);
grid('on');

f2=figure;
plot(t,P_out(:,7),t,P_out(:,8),t,P_out(:,9), 'LineWidth',width);
title('w_l_p - Liquid produced');
ylabel('kg/s');
xlabel('Hour');
legend({'Well 1','Well 2','Well 3'}, 'FontSize',12);
grid('on');
```

Appendix D


```
function [dm, out] = functionDAE_multi(x,u,PI_dev, WC_dev, GOR_dev)

% This function contains the nonlinear model of 3 wells
%Choose all PI_dev, WC_dev, GOR_dev = 1 if using nominal model.

%states
m_ga = [x(1)   x(2)    x(3)];
m_gt = [x(4)   x(5)    x(6)];
m_lt = [x(7)   x(8)    x(9)];

%Input
w_ga = [u(1)   u(2)    u(3)];

%% Parameters

% Well    = [Well1  Well2   Well3]
PI        = [2.51   1.63    1.62]*1e4 .* PI_dev;
WC        = [0.20   0.10    0.25] .* WC_dev;
GOR       = [0.05   0.07    0.03] .* GOR_dev;
L_a_tl    = [2758   2559    2677];
L_a_vl    = [2271   2344    1863];
L_r_vl    = [114    67      61];
K         = [68.43  67.82   67.82];
T_a       = [280    280     280];
T_t       = T_a;
L_t_tl    = L_a_tl;
L_t_vl    = L_a_vl;


Z = 1.3;
R = 8.31446261815324;
M = 20*1e-3;    %Molar mass
A_a = 0.0174;   %Annulus cross section area
g = 9.80665;    %Gravity acceleration constant
rho_w = 1000;   %Density of water
rho_o = 800;    %Density of oil
A_t = 0.0194;   %Annulus cross section area
a_Y = 0.66;
P_r = 150;      %Reservoir pressure
P_s = 30;       %Separator pressure
Cv = 8190;      %Constant Cv

P_ainj_min = 10;
P_wh_min = 10;

%% Algebraic equation

% 4
P_a = ((Z.*m_ga*R.*T_a)./(M.*A_a.*L_a_tl))*1e-5; %[bar]

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

% 6
rho_l = rho_w.*WC+rho_o.*(1-WC);      %[kg/m3]

% 7
V_g = A_t.*L_t_tl - m_lt./rho_l;      %[m3]

% 8
rho_m = (m_gt+m_lt)./(A_t.*L_t_tl);   %[kg/m3]

% 9
P_tinj = ((Z.*m_gt.*R.*T_t)./(M.*V_g) + 0.5.*rho_m.*g.*L_t_vl)*1e-5; %[bar]

% 10
Y2 = 1 - a_Y.*((P_ainj - P_tinj)./(max(P_ainj,P_ainj_min)));

% 11
rho_ga = m_ga./(A_a.*L_a_tl);         %[kg/m3]

% 12
w_ginj = K.*Y2.*sqrt(max(rho_ga.*(P_ainj-P_tinj),0))/3600; %[kg/s]
```

```
% 13
P_wf = P_tinj + (rho_l.*g.*L_r_vl)*1e-5;  %[bar]

% 14
w_lr = PI.*max(P_r - P_wf,0)/3600;   %[kg/s]

% 15
w_or = (rho_o./rho_l).*(1-WC).*w_lr;  %[kg/s]

% 16
w_gr = GOR.*w_or;   %[kg/s]

% 17
P_wh = (((Z.*m_gt.*R.*T_t)./(M.*V_g)) - 0.5.*rho_m.*g.*L_t_vl)*1e-5; %[bar]

% 18
Y3 = 1 - a_Y.*((P_wh - P_s)./(max(P_wh,P_wh_min)));

% 19
w_glp = Cv.*Y3.*sqrt(max(rho_m.*(P_wh-P_s), 0))/3600;  %[kg/s]

% 20
w_gp = (m_gt./(m_gt+m_lt)).*w_glp;    %[kg/s]

% 21
w_lp = (m_lt./(m_gt+m_lt)).*w_glp;    %[kg/s]

% 22
w_op = (rho_o./rho_l).*(1-WC).*w_lp;   %[kg/s]


%% Differential equations
%1.ODE
dev_m_ga = w_ga - w_ginj;

%2.ODE
dev_m_gt = w_ginj + w_gr - w_gp;

%3.ODE
dev_m_lt = w_lr - w_lp;

dm = [dev_m_ga, dev_m_gt, dev_m_lt];

%Other variables as outputs

out=[P_wf P_wh w_lp w_op w_gp w_glp w_gr w_or w_lr w_ginj];
```

Appendix E


```
function w_ga = generateInput(Np)

% Generating inputs to open loop simulations

w_ga = zeros(Np,3);

section=Np/3;

% For stepping
for i = 1:Np
    if i < section
        w_ga(i,:)=[2.2 2.5 1.8];

    elseif i < 2*section
        w_ga(i,:)=[1.7 2.0 1.3];

    else
        w_ga(i,:)=[2.2 2.5 1.8];
    end
end


%----------------------

% For ramping
% for i = 1:Np
%     if i < section
%         w_ga(i,:)=[2.2 2.5 1.8];
%
%     elseif i < 2*section
%         if w_ga(i-1,1) > 1.701
%             w_ga(i,:) = w_ga(i-1,:) - 0.001;
%         else
%             w_ga(i,:) = w_ga(i-1,:);
%         end
%
%     else
%         if w_ga(i-1,1) < 2.2
%             w_ga(i,:) = w_ga(i-1,:) + 0.001;
%         else
%             w_ga(i,:) = w_ga(i-1,:);
%         end
%     end
% end
```

Appendix F


```
function [x_next, out] = updateState(states,dt,u,PI_dev, WC_dev, GOR_dev)

%Update states and output of the nonlinear model using Runge-Kutta 4th order method

%RK4
K1 = functionDAE_multi(states,u,PI_dev, WC_dev, GOR_dev);
K2 = functionDAE_multi(states+K1.*(dt/2),u,PI_dev, WC_dev, GOR_dev);
K3 = functionDAE_multi(states+K2.*(dt/2),u,PI_dev, WC_dev, GOR_dev);
K4 = functionDAE_multi(states+K3.*dt,u,PI_dev, WC_dev, GOR_dev);

x_next = states + (dt/6).*(K1+2.*K2+2.*K3 + K4);

%Gather other variables such as outputs
[~, out] = functionDAE_multi(x_next,u,PI_dev, WC_dev, GOR_dev);
```

Appendix G


```
function [Ac, Bc, Cc, y_op, u_op] = CalcJac_y()

% This function makes function of Ac, BC, Cc, y_op and u_op.
% Input arguments to the functions are states.

syms m_ga m_gt m_lt w_ga P_a P_ainj rho_l V_g rho_m P_tinj Y2 rho_ga...
    w_ginj P_wf w_lr w_or w_gr P_wh Y3 w_glp w_gp w_lp w_op...
    Z R T_a M A_a L_a_tl g L_a_vl rho_w WC rho_o A_t L_t_tl...
    T_t L_t_vl a_Y K L_r_vl PI P_r GOR P_s C_V

assume(Z > 0);
assume(R > 0);
assume(T_a > 0);
assume(M > 0);
assume(A_a > 0);
assume(L_a_tl > 0);
assume(g > 0);
assume(L_a_vl > 0);
assume(rho_w > 0);
assume(WC > 0);
assume(rho_o > 0);
assume(A_t > 0);
assume(L_t_tl > 0);
assume(T_t > 0);
assume(L_t_vl > 0);
assume(a_Y > 0);
assume(K > 0);
assume(L_r_vl > 0);
assume(PI > 0);
assume(P_r > 0);
assume(GOR > 0);
assume(P_s > 0);
assume(C_V > 0);

assume(P_ainj>0);
assume((P_ainj-P_tinj)>0);
assume(P_wh>0);
assume((P_wh-P_s)>0);
assume((P_r - P_wf)>0);

wells = 3;

%States
m_ga = sym('m_ga', [1 wells]);
m_gt = sym('m_gt', [1 wells]);
m_lt = sym('m_lt', [1 wells]);


%Input
w_ga = sym('w_ga', [1 wells]);

%% Parameters

% Well    = [Well1  Well2   Well3]
PI        = sym('PI', [1 wells]);
WC        = sym('WC', [1 wells]);
GOR       = sym('GOR', [1 wells]);
L_a_tl    = sym('L_a_tl', [1 wells]);
L_a_vl    = sym('L_a_vl', [1 wells]);
L_r_vl    = sym('L_r_vl', [1 wells]);
K         = sym('K', [1 wells]);
T_a       = sym('T_a', [1 wells]);
T_t       = sym('T_t', [1 wells]);
L_t_tl    = sym('L_t_tl', [1 wells]);
L_t_vl    = sym('L_t_vl', [1 wells]);



% 4
P_a = ((Z.*m_ga.*R.*T_a)./(M.*A_a.*L_a_tl))*1e-5; %[bar]

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

```
% 6
rho_l = rho_w.*WC+rho_o.*(1-WC);      %[kg/m3]

% 7
V_g = A_t.*L_t_tl - m_lt./rho_l;      %[m3]

% 8
rho_m = (m_gt+m_lt)./(A_t*L_t_tl);   %[kg/m3]

% 9
P_tinj = ((Z.*m_gt.*R.*T_t)./(M.*V_g) + 0.5.*rho_m.*g.*L_t_vl)*1e-5; %[bar]

% 10
Y2 = 1 - a_Y.*((P_ainj - P_tinj)./(P_ainj));

% 11
rho_ga = m_ga./(A_a.*L_a_tl);         %[kg/m3]

% 12
w_ginj = K.*Y2.*sqrt(rho_ga.*(P_ainj-P_tinj))/3600;

% 13
P_wf = P_tinj + (rho_l.*g.*L_r_vl)*1e-5;  %[bar]

% 14
w_lr = PI.*(P_r - P_wf)/3600;   %[kg/s]

% 15
w_or = (rho_o./rho_l).*(1-WC).*w_lr;  %[kg/s]

% 16
w_gr = GOR.*w_or;   %[kg/s]

% 17
P_wh = (((Z.*m_gt.*R.*T_t)./(M.*V_g)) - 0.5.*rho_m.*g.*L_t_vl)*1e-5; %[bar]

% 18
Y3 = 1 - a_Y.*(P_wh - P_s)./P_wh;

% 19
w_glp = C_V.*Y3.*sqrt(rho_m.*(P_wh-P_s))/3600;  %[kg/s]

% 20
w_gp = (m_gt./(m_gt+m_lt)).*w_glp;    %[kg/s]

% 21
w_lp = (m_lt./(m_gt+m_lt)).*w_glp;    %[kg/s]

% 22
w_op = (rho_o./rho_l).*(1-WC).*w_lp;   %[kg/s]

%1.ODE
dev_m_ga = w_ga - w_ginj;

%2.ODE
dev_m_gt = w_ginj + w_gr - w_gp;

%3.ODE
dev_m_lt = w_lr - w_lp;

%---------------------------------------------------------------

%Generate jacobians
dm = [dev_m_ga dev_m_gt dev_m_lt];
x = [m_ga m_gt m_lt];
y = [P_wf P_wh w_lp w_op];

J_A=jacobian(dm, x);
J_B=jacobian(dm, w_ga);
J_C=jacobian(y, x);


%Paramter values that will be substituted
```

```
% Well    = [Well1  Well2   Well3]
PI_        = [2.51   1.63    1.62]*1e4;
WC_        = [0.20   0.10    0.25];
GOR_       = [0.05   0.07    0.03];
L_a_tl_    = [2758   2559    2677];
L_a_vl_    = [2271   2344    1863];
L_r_vl_    = [114    67      61];
K_         = [68.43  67.82   67.82];
T_a_       = [280    280     280];
T_t_       = T_a_;
L_t_tl_    = L_a_tl_;
L_t_vl_    = L_a_vl_;


Z_ = 1.3;
R_ = 8.31446261815324;
M_ = 20*1e-3;    %Molar mass
A_a_ = 0.0174;   %Annulus cross section area
g_ = 9.80665;    %Gravity acceleration constant
rho_w_ = 1000;   %Density of water
rho_o_ = 800;    %Density of oil
A_t_ = 0.0194;   %Annulus cross section area
a_Y_ = 0.66;
P_r_ = 150;      %Reservoir pressure
P_s_ = 30;       %Separator pressure
Cv_ = 8190;      %Constant Cv

%Substituting
%Substitute the paramteres and the new states to the matrices.
A = subs(J_A,[A_a, A_t, a_Y, C_V, g, GOR, K, L_a_tl, L_a_vl, L_r_vl,...
    L_t_tl, L_t_vl, M, P_r, P_s, PI, R, rho_o, rho_w, T_a, T_t, WC, Z],...
    [A_a_, A_t_, a_Y_, Cv_, g_, GOR_, K_, L_a_tl_, L_a_vl_, L_r_vl_,...
    L_t_tl_, L_t_vl_, M_, P_r_, P_s_, PI_, R_, rho_o_, rho_w_, T_a_, T_t_, WC_, Z_]);


C = subs(J_C,[A_a, A_t, a_Y, C_V, g, GOR, K, L_a_tl, L_a_vl, L_r_vl,...
    L_t_tl, L_t_vl, M, P_r, P_s, PI, R, rho_o, rho_w, T_a, T_t, WC, Z],...
    [A_a_, A_t_, a_Y_, Cv_, g_, GOR_, K_, L_a_tl_, L_a_vl_, L_r_vl_,...
    L_t_tl_, L_t_vl_, M_, P_r_, P_s_, PI_, R_, rho_o_, rho_w_, T_a_, T_t_, WC_, Z_]);

Bc = double(J_B);

Ac = matlabFunction(A);
Cc = matlabFunction(C);

% If writing matrices to file
% Ac = matlabFunction(A,'File','Ac_1');
% Bc = matlabFunction(J_B, 'File','Bc_1');
% Cc = matlabFunction(C, 'File','Cc_1');


%Outputs operating point
Y = subs(y',[A_a, A_t, a_Y, C_V, g, GOR, K, L_a_tl, L_a_vl, L_r_vl,...
    L_t_tl, L_t_vl, M, P_r, P_s, PI, R, rho_o, rho_w, T_a, T_t, WC, Z],...
    [A_a_, A_t_, a_Y_, Cv_, g_, GOR_, K_, L_a_tl_, L_a_vl_, L_r_vl_,...
    L_t_tl_, L_t_vl_, M_, P_r_, P_s_, PI_, R_, rho_o_, rho_w_, T_a_, T_t_, WC_, Z_]);

y_op = matlabFunction(Y);

% If writing operating point to file
% y_op = matlabFunction(Y,'File','y_1');


%Input operating points
u = subs(w_ginj,[A_a, A_t, a_Y, C_V, g, GOR, K, L_a_tl, L_a_vl, L_r_vl,...
    L_t_tl, L_t_vl, M, P_r, P_s, PI, R, rho_o, rho_w, T_a, T_t, WC, Z],...
    [A_a_, A_t_, a_Y_, Cv_, g_, GOR_, K_, L_a_tl_, L_a_vl_, L_r_vl_,...
    L_t_tl_, L_t_vl_, M_, P_r_, P_s_, PI_, R_, rho_o_, rho_w_, T_a_, T_t_, WC_, Z_]);

u_op = matlabFunction(u);

% If writing operating point to file
% w_ga_op = matlabFunction(u,'File','u_1');
```

Example of the system matrices with real values

In this example it is chosen states which correspond to control input w_{ga} = [2.2, 2.5, 1.8]:

m_{ga} = [9012.7, 9839.1, 7214.2]

m_{gt} = [1484.7, 1520.0, 1142.7]

m_{lt} = [21814.0, 17254.0, 25813.0]

$x_{op} = [m_{ga}, m_{gt}, m_{lt}]^T$

Chosen outputs: w_{lp}, P_{wh}, P_{wf} and w_{op}

$$A_c = \begin{bmatrix} \frac{\partial \dot{m}_{ga}^1}{\partial m_{ga}^1} & \frac{\partial \dot{m}_{ga}^1}{\partial m_{ga}^2} & \frac{\partial \dot{m}_{ga}^1}{\partial m_{ga}^3} & \frac{\partial \dot{m}_{ga}^1}{\partial m_{gt}^1} & \frac{\partial \dot{m}_{ga}^1}{\partial m_{gt}^2} & \frac{\partial \dot{m}_{ga}^1}{\partial m_{gt}^3} & \frac{\partial \dot{m}_{ga}^1}{\partial m_{lt}^1} & \frac{\partial \dot{m}_{ga}^1}{\partial m_{lt}^2} & \frac{\partial \dot{m}_{ga}^1}{\partial m_{lt}^3} \\ \frac{\partial \dot{m}_{ga}^2}{\partial m_{ga}^1} & \frac{\partial \dot{m}_{ga}^2}{\partial m_{ga}^2} & \frac{\partial \dot{m}_{ga}^2}{\partial m_{ga}^3} & \frac{\partial \dot{m}_{ga}^2}{\partial m_{gt}^1} & \frac{\partial \dot{m}_{ga}^2}{\partial m_{gt}^2} & \frac{\partial \dot{m}_{ga}^2}{\partial m_{gt}^3} & \frac{\partial \dot{m}_{ga}^2}{\partial m_{lt}^1} & \frac{\partial \dot{m}_{ga}^2}{\partial m_{lt}^2} & \frac{\partial \dot{m}_{ga}^2}{\partial m_{lt}^3} \\ \frac{\partial \dot{m}_{ga}^3}{\partial m_{ga}^1} & \frac{\partial \dot{m}_{ga}^3}{\partial m_{ga}^2} & \frac{\partial \dot{m}_{ga}^3}{\partial m_{ga}^3} & \frac{\partial \dot{m}_{ga}^3}{\partial m_{gt}^1} & \frac{\partial \dot{m}_{ga}^3}{\partial m_{gt}^2} & \frac{\partial \dot{m}_{ga}^3}{\partial m_{gt}^3} & \frac{\partial \dot{m}_{ga}^3}{\partial m_{lt}^1} & \frac{\partial \dot{m}_{ga}^3}{\partial m_{lt}^2} & \frac{\partial \dot{m}_{ga}^3}{\partial m_{lt}^3} \\ \frac{\partial \dot{m}_{gt}^1}{\partial m_{ga}^1} & \frac{\partial \dot{m}_{gt}^1}{\partial m_{ga}^2} & \frac{\partial \dot{m}_{gt}^1}{\partial m_{ga}^3} & \frac{\partial \dot{m}_{gt}^1}{\partial m_{gt}^1} & \frac{\partial \dot{m}_{gt}^1}{\partial m_{gt}^2} & \frac{\partial \dot{m}_{gt}^1}{\partial m_{gt}^3} & \frac{\partial \dot{m}_{gt}^1}{\partial m_{lt}^1} & \frac{\partial \dot{m}_{gt}^1}{\partial m_{lt}^2} & \frac{\partial \dot{m}_{gt}^1}{\partial m_{lt}^3} \\ \frac{\partial \dot{m}_{gt}^2}{\partial m_{ga}^1} & \frac{\partial \dot{m}_{gt}^2}{\partial m_{ga}^2} & \frac{\partial \dot{m}_{gt}^2}{\partial m_{ga}^3} & \frac{\partial \dot{m}_{gt}^2}{\partial m_{gt}^1} & \frac{\partial \dot{m}_{gt}^2}{\partial m_{gt}^2} & \frac{\partial \dot{m}_{gt}^2}{\partial m_{gt}^3} & \frac{\partial \dot{m}_{gt}^2}{\partial m_{lt}^1} & \frac{\partial \dot{m}_{gt}^2}{\partial m_{lt}^2} & \frac{\partial \dot{m}_{gt}^2}{\partial m_{lt}^3} \\ \frac{\partial \dot{m}_{gt}^3}{\partial m_{ga}^1} & \frac{\partial \dot{m}_{gt}^3}{\partial m_{ga}^2} & \frac{\partial \dot{m}_{gt}^3}{\partial m_{ga}^3} & \frac{\partial \dot{m}_{gt}^3}{\partial m_{gt}^1} & \frac{\partial \dot{m}_{gt}^3}{\partial m_{gt}^2} & \frac{\partial \dot{m}_{gt}^3}{\partial m_{gt}^3} & \frac{\partial \dot{m}_{gt}^3}{\partial m_{lt}^1} & \frac{\partial \dot{m}_{gt}^3}{\partial m_{lt}^2} & \frac{\partial \dot{m}_{gt}^3}{\partial m_{lt}^3} \\ \frac{\partial \dot{m}_{lt}^1}{\partial m_{ga}^1} & \frac{\partial \dot{m}_{lt}^1}{\partial m_{ga}^2} & \frac{\partial \dot{m}_{lt}^1}{\partial m_{ga}^3} & \frac{\partial \dot{m}_{lt}^1}{\partial m_{gt}^1} & \frac{\partial \dot{m}_{lt}^1}{\partial m_{gt}^2} & \frac{\partial \dot{m}_{lt}^1}{\partial m_{gt}^3} & \frac{\partial \dot{m}_{lt}^1}{\partial m_{lt}^1} & \frac{\partial \dot{m}_{lt}^1}{\partial m_{lt}^2} & \frac{\partial \dot{m}_{lt}^1}{\partial m_{lt}^3} \\ \frac{\partial \dot{m}_{lt}^2}{\partial m_{ga}^1} & \frac{\partial \dot{m}_{lt}^2}{\partial m_{ga}^2} & \frac{\partial \dot{m}_{lt}^2}{\partial m_{ga}^3} & \frac{\partial \dot{m}_{lt}^2}{\partial m_{gt}^1} & \frac{\partial \dot{m}_{lt}^2}{\partial m_{gt}^2} & \frac{\partial \dot{m}_{lt}^2}{\partial m_{gt}^3} & \frac{\partial \dot{m}_{lt}^2}{\partial m_{lt}^1} & \frac{\partial \dot{m}_{lt}^2}{\partial m_{lt}^2} & \frac{\partial \dot{m}_{lt}^2}{\partial m_{lt}^3} \\ \frac{\partial \dot{m}_{lt}^3}{\partial m_{ga}^1} & \frac{\partial \dot{m}_{lt}^3}{\partial m_{ga}^2} & \frac{\partial \dot{m}_{lt}^3}{\partial m_{ga}^3} & \frac{\partial \dot{m}_{lt}^3}{\partial m_{gt}^1} & \frac{\partial \dot{m}_{lt}^3}{\partial m_{gt}^2} & \frac{\partial \dot{m}_{lt}^3}{\partial m_{gt}^3} & \frac{\partial \dot{m}_{lt}^3}{\partial m_{lt}^1} & \frac{\partial \dot{m}_{lt}^3}{\partial m_{lt}^2} & \frac{\partial \dot{m}_{lt}^3}{\partial m_{lt}^3} \end{bmatrix}_{x_{op}}$$

$$= \begin{bmatrix} -0.0002 & 0 & 0 & -0.0001 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.0002 & 0 & 0 & -0.0002 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.0002 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.0002 & 0 & 0 & -0.0558 & 0 & 0 & -0.0024 & 0 & 0 \\ 0 & 0.0002 & 0 & 0 & -0.0436 & 0 & 0 & -0.0019 & 0 \\ 0 & 0 & 0.0002 & 0 & 0 & -0.0478 & 0 & 0 & -0.0020 \\ 0 & 0 & 0 & -0.9472 & 0 & 0 & -0.0560 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.5106 & 0 & 0 & -0.0347 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1.1815 & 0 & 0 & -0.0620 \end{bmatrix}$$

$$B_c = \begin{bmatrix} \frac{\partial \dot{m}_{ga}^1}{\partial w_{ga}^1} & \frac{\partial \dot{m}_{ga}^1}{\partial w_{ga}^2} & \frac{\partial \dot{m}_{ga}^1}{\partial w_{ga}^3} \\ \frac{\partial \dot{m}_{ga}^2}{\partial w_{ga}^1} & \frac{\partial \dot{m}_{ga}^2}{\partial w_{ga}^2} & \frac{\partial \dot{m}_{ga}^2}{\partial w_{ga}^3} \\ \frac{\partial \dot{m}_{ga}^3}{\partial w_{ga}^1} & \frac{\partial \dot{m}_{ga}^3}{\partial w_{ga}^2} & \frac{\partial \dot{m}_{ga}^3}{\partial w_{ga}^3} \\ \frac{\partial \dot{m}_{gt}^1}{\partial w_{ga}^1} & \frac{\partial \dot{m}_{gt}^1}{\partial w_{ga}^2} & \frac{\partial \dot{m}_{gt}^1}{\partial w_{ga}^3} \\ \frac{\partial \dot{m}_{gt}^2}{\partial w_{ga}^1} & \frac{\partial \dot{m}_{gt}^2}{\partial w_{ga}^2} & \frac{\partial \dot{m}_{gt}^2}{\partial w_{ga}^3} \\ \frac{\partial \dot{m}_{gt}^3}{\partial w_{ga}^1} & \frac{\partial \dot{m}_{gt}^3}{\partial w_{ga}^2} & \frac{\partial \dot{m}_{gt}^3}{\partial w_{ga}^3} \\ \frac{\partial \dot{m}_{lt}^1}{\partial w_{ga}^1} & \frac{\partial \dot{m}_{lt}^1}{\partial w_{ga}^2} & \frac{\partial \dot{m}_{lt}^1}{\partial w_{ga}^3} \\ \frac{\partial \dot{m}_{lt}^2}{\partial w_{ga}^1} & \frac{\partial \dot{m}_{lt}^2}{\partial w_{ga}^2} & \frac{\partial \dot{m}_{lt}^2}{\partial w_{ga}^3} \\ \frac{\partial \dot{m}_{lt}^3}{\partial w_{ga}^1} & \frac{\partial \dot{m}_{lt}^3}{\partial w_{ga}^2} & \frac{\partial \dot{m}_{lt}^3}{\partial w_{ga}^3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$C_c = \begin{bmatrix}
\frac{\partial w_{lp}^1}{\partial m_{ga}^1} & \frac{\partial w_{lp}^1}{\partial m_{ga}^2} & \frac{\partial w_{lp}^1}{\partial m_{ga}^3} & \frac{\partial w_{lp}^1}{\partial m_{gt}^1} & \frac{\partial w_{lp}^1}{\partial m_{gt}^2} & \frac{\partial w_{lp}^1}{\partial m_{gt}^3} & \frac{\partial w_{lp}^1}{\partial m_{lt}^1} & \frac{\partial w_{lp}^1}{\partial m_{lt}^2} & \frac{\partial w_{lp}^1}{\partial m_{lt}^3} \\
\frac{\partial w_{lp}^2}{\partial m_{ga}^1} & \frac{\partial w_{lp}^2}{\partial m_{ga}^2} & \frac{\partial w_{lp}^2}{\partial m_{ga}^3} & \frac{\partial w_{lp}^2}{\partial m_{gt}^1} & \frac{\partial w_{lp}^2}{\partial m_{gt}^2} & \frac{\partial w_{lp}^2}{\partial m_{gt}^3} & \frac{\partial w_{lp}^2}{\partial m_{lt}^1} & \frac{\partial w_{lp}^2}{\partial m_{lt}^2} & \frac{\partial w_{lp}^2}{\partial m_{lt}^3} \\
\frac{\partial w_{lp}^3}{\partial m_{ga}^1} & \frac{\partial w_{lp}^3}{\partial m_{ga}^2} & \frac{\partial w_{lp}^3}{\partial m_{ga}^3} & \frac{\partial w_{lp}^3}{\partial m_{gt}^1} & \frac{\partial w_{lp}^3}{\partial m_{gt}^2} & \frac{\partial w_{lp}^3}{\partial m_{gt}^3} & \frac{\partial w_{lp}^3}{\partial m_{lt}^1} & \frac{\partial w_{lp}^3}{\partial m_{lt}^2} & \frac{\partial w_{lp}^3}{\partial m_{lt}^3} \\
\frac{\partial P_{wh}^1}{\partial m_{ga}^1} & \frac{\partial P_{wh}^1}{\partial m_{ga}^2} & \frac{\partial P_{wh}^1}{\partial m_{ga}^3} & \frac{\partial P_{wh}^1}{\partial m_{gt}^1} & \frac{\partial P_{wh}^1}{\partial m_{gt}^2} & \frac{\partial P_{wh}^1}{\partial m_{gt}^3} & \frac{\partial P_{wh}^1}{\partial m_{lt}^1} & \frac{\partial P_{wh}^1}{\partial m_{lt}^2} & \frac{\partial P_{wh}^1}{\partial m_{lt}^3} \\
\frac{\partial P_{wh}^2}{\partial m_{ga}^1} & \frac{\partial P_{wh}^2}{\partial m_{ga}^2} & \frac{\partial P_{wh}^2}{\partial m_{ga}^3} & \frac{\partial P_{wh}^2}{\partial m_{gt}^1} & \frac{\partial P_{wh}^2}{\partial m_{gt}^2} & \frac{\partial P_{wh}^2}{\partial m_{gt}^3} & \frac{\partial P_{wh}^2}{\partial m_{lt}^1} & \frac{\partial P_{wh}^2}{\partial m_{lt}^2} & \frac{\partial P_{wh}^2}{\partial m_{lt}^3} \\
\frac{\partial P_{wh}^3}{\partial m_{ga}^1} & \frac{\partial P_{wh}^3}{\partial m_{ga}^2} & \frac{\partial P_{wh}^3}{\partial m_{ga}^3} & \frac{\partial P_{wh}^3}{\partial m_{gt}^1} & \frac{\partial P_{wh}^3}{\partial m_{gt}^2} & \frac{\partial P_{wh}^3}{\partial m_{gt}^3} & \frac{\partial P_{wh}^3}{\partial m_{lt}^1} & \frac{\partial P_{wh}^3}{\partial m_{lt}^2} & \frac{\partial P_{wh}^3}{\partial m_{lt}^3} \\
\frac{\partial P_{wf}^1}{\partial m_{ga}^1} & \frac{\partial P_{wf}^1}{\partial m_{ga}^2} & \frac{\partial P_{wf}^1}{\partial m_{ga}^3} & \frac{\partial P_{wf}^1}{\partial m_{gt}^1} & \frac{\partial P_{wf}^1}{\partial m_{gt}^2} & \frac{\partial P_{wf}^1}{\partial m_{gt}^3} & \frac{\partial P_{wf}^1}{\partial m_{lt}^1} & \frac{\partial P_{wf}^1}{\partial m_{lt}^2} & \frac{\partial P_{wf}^1}{\partial m_{lt}^3} \\
\frac{\partial P_{wf}^2}{\partial m_{ga}^1} & \frac{\partial P_{wf}^2}{\partial m_{ga}^2} & \frac{\partial P_{wf}^2}{\partial m_{ga}^3} & \frac{\partial P_{wf}^2}{\partial m_{gt}^1} & \frac{\partial P_{wf}^2}{\partial m_{gt}^2} & \frac{\partial P_{wf}^2}{\partial m_{gt}^3} & \frac{\partial P_{wf}^2}{\partial m_{lt}^1} & \frac{\partial P_{wf}^2}{\partial m_{lt}^2} & \frac{\partial P_{wf}^2}{\partial m_{lt}^3} \\
\frac{\partial P_{wf}^3}{\partial m_{ga}^1} & \frac{\partial P_{wf}^3}{\partial m_{ga}^2} & \frac{\partial P_{wf}^3}{\partial m_{ga}^3} & \frac{\partial P_{wf}^3}{\partial m_{gt}^1} & \frac{\partial P_{wf}^3}{\partial m_{gt}^2} & \frac{\partial P_{wf}^3}{\partial m_{gt}^3} & \frac{\partial P_{wf}^3}{\partial m_{lt}^1} & \frac{\partial P_{wf}^3}{\partial m_{lt}^2} & \frac{\partial P_{wf}^3}{\partial m_{lt}^3} \\
\frac{\partial w_{op}^1}{\partial m_{ga}^1} & \frac{\partial w_{op}^1}{\partial m_{ga}^2} & \frac{\partial w_{op}^1}{\partial m_{ga}^3} & \frac{\partial w_{op}^1}{\partial m_{gt}^1} & \frac{\partial w_{op}^1}{\partial m_{gt}^2} & \frac{\partial w_{op}^1}{\partial m_{gt}^3} & \frac{\partial w_{op}^1}{\partial m_{lt}^1} & \frac{\partial w_{op}^1}{\partial m_{lt}^2} & \frac{\partial w_{op}^1}{\partial m_{lt}^3} \\
\frac{\partial w_{op}^2}{\partial m_{ga}^1} & \frac{\partial w_{op}^2}{\partial m_{ga}^2} & \frac{\partial w_{op}^2}{\partial m_{ga}^3} & \frac{\partial w_{op}^2}{\partial m_{gt}^1} & \frac{\partial w_{op}^2}{\partial m_{gt}^2} & \frac{\partial w_{op}^2}{\partial m_{gt}^3} & \frac{\partial w_{op}^2}{\partial m_{lt}^1} & \frac{\partial w_{op}^2}{\partial m_{lt}^2} & \frac{\partial w_{op}^2}{\partial m_{lt}^3} \\
\frac{\partial w_{op}^3}{\partial m_{ga}^1} & \frac{\partial w_{op}^3}{\partial m_{ga}^2} & \frac{\partial w_{op}^3}{\partial m_{ga}^3} & \frac{\partial w_{op}^3}{\partial m_{gt}^1} & \frac{\partial w_{op}^3}{\partial m_{gt}^2} & \frac{\partial w_{op}^3}{\partial m_{gt}^3} & \frac{\partial w_{op}^3}{\partial m_{lt}^1} & \frac{\partial w_{op}^3}{\partial m_{lt}^2} & \frac{\partial w_{op}^3}{\partial m_{lt}^3}
\end{bmatrix}_{x_{op}}$$

$$= \begin{bmatrix}
0 & 0 & 0 & 0.5496 & 0 & 0 & 0.0169 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.2605 & 0 & 0 & 0.0087 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.8578 & 0 & 0 & 0.0344 \\
0 & 0 & 0 & 0.0529 & 0 & 0 & 0.0014 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.0506 & 0 & 0 & 0.0011 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.0684 & 0 & 0 & 0.0026 \\
0 & 0 & 0 & 0.0570 & 0 & 0 & 0.0056 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.0552 & 0 & 0 & 0.0057 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.0719 & 0 & 0 & 0.0061 \\
0 & 0 & 0 & 0.4187 & 0 & 0 & 0.0129 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.2288 & 0 & 0 & 0.0077 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.6055 & 0 & 0 & 0.0243
\end{bmatrix}$$

Appendix I


```
function [A,B,C,y_op, u_op] = UpdateMatrices_op_Open(Ac, Bc, Cc , x, y, u, dt)

% Substitute states into system matrices and operating point vectors in open loop simulation
% Used in open loop simulation for linear model.

Ac_ = Ac(x(1), x(2), x(3), x(4), x(5), x(6), x(7), x(8), x(9));
Cc_ = Cc(x(4), x(5), x(6), x(7), x(8), x(9));


sys = ss(Ac_,Bc,Cc_,0);
d_sys = c2d(sys,dt);

A=d_sys.a;
B=d_sys.b;
C=d_sys.c;
%D=d_sys.d;

%Outout operationpoints
y_op = y(x(4),x(5),x(6),x(7),x(8),x(9));


%Input operation points
u_op_ = u(x(1), x(2), x(3), x(4), x(5), x(6), x(7), x(8), x(9));
u_op = u_op_';
```

Appendix J


```
%Open loop simulation comparing nonlinear model and linear model
%Static system matrices

clear
close all

%-----------------------------------------------------------
% Make symbolic matrices of A, B, C and operationg points
[J_A, J_B, J_C, y_sym, u_sym] = CalcJac_y();

% States initial values
% when w_ga = [2.2 2.5 1.8]
m_ga = [9.0127e3    9.8391e3    7.2142e3];
m_gt = [1.4847e3    1.52e3      1.1427e3];
m_lt = [2.1814e4    1.7254e4    2.5813e4];

% State operating point for linear model
m_op = [m_ga m_gt m_lt]';

%States initial values
state_ini_non = m_op';      % Non-linear
state_ini_lin = zeros(9,1); % Linear

%Simulation parameters
dt = 15; %[s]
hour = 60; %[h]        hours of simulationtime
Np = 3600*hour/dt;     %Number of datapoints
t=linspace(0,hour,Np); %Used for plotting
l = length(state_ini_non); %Number of states

% Updating matrices with states
[A,B,C,y_op,u_op] = UpdateMatrices_op_Open(J_A, J_B, J_C,state_ini_non, y_sym, u_sym, dt);


%Input
%Generate input for whole horizon
w_ga = generateInput(Np);

P_non = zeros(Np,l);       %Matrix for storing nonlinear states
P_lin = P_non;             %Matrix for storing linear states
P_out = zeros(Np,12);      %Matrix for storing nonlinear outputs
P_out_lin = P_out;         %Matrix for storing linear outputs

tic;
% Running simulations
for i = 1:Np

    %store the outputs
    P_non(i,:) = state_ini_non;
    P_lin(i,:) = (state_ini_lin + m_op)';

    %Run linear model in deviation form
    [x_next_lin, out_lin] = linearModel(state_ini_lin, (w_ga(i,:)'-u_op),A,B,C);
    state_ini_lin = x_next_lin;
    P_out_lin(i,:) = (out_lin+y_op)';

    % Run non-linear model with RK method
    [x_next_non, out_non] = updateState(state_ini_non,dt,w_ga(i,:),1,1,1);
    state_ini_non = x_next_non;
    P_out(i,:) = out_non(:,1:12);


end
toc;

%-------------------------------------------------------------------------

%% Plotting

%Defining colors
newcolors = {'#0072BD', '#D95319', '#EDB120', '#0072BD', '#D95319', '#EDB120'};
colororder(newcolors);
width = 2;
```

```
tiledlayout(4,1);
ax1 = nexttile;
plot(ax1,t,P_out(:,1), t,P_out(:,2), t,P_out(:,3), t,P_out_lin(:,1),'--', ...
    t,P_out_lin(:,2), '--', t,P_out_lin(:,3),'--', 'LineWidth',width);
title(ax1,'P_w_f - Bottom hole pressure');
ylabel(ax1,'bar');
xlabel(ax1,'Hour');
%ylim([125 145]);
legend({'Well 1','Well 2','Well 3','Well 1_l_i_n','Well 2_l_i_n','Well 3_l_i_n'}, 'FontSize',12);
grid('on');

ax2 = nexttile;
plot(ax2,t,P_out(:,4), t,P_out(:,5), t,P_out(:,6), t,P_out_lin(:,4),'--', ...
    t,P_out_lin(:,5), '--', t,P_out_lin(:,6),'--', 'LineWidth',width);
title(ax2,'P_w_h - Well head pressure');
ylabel(ax2,'bar');
xlabel(ax2,'Hour');
%ylim([30 40]);
grid('on');

ax3 = nexttile;
plot(ax3,t,P_out(:,7), t,P_out(:,8), t,P_out(:,9), t,P_out_lin(:,7),'--', ...
    t,P_out_lin(:,8), '--', t,P_out_lin(:,9),'--', 'LineWidth',width);
title(ax3,'w_l_p - Liquid produced');
ylabel(ax3,'kg/s');
xlabel(ax3,'Hour');
%ylim([60 100]);
grid('on');

ax4 = nexttile;
plot(ax4,t,w_ga, 'LineWidth',width);
title(ax4,'w_g_a - Injected lift gas');
ylabel(ax4,'kg/s');
xlabel(ax4,'Hour');
%ylim([1 3]);
grid('on');


f2=figure;
colororder(newcolors);
plot(t,P_out(:,10), t,P_out(:,11), t,P_out(:,12), t,P_out_lin(:,10),'--', ...
    t,P_out_lin(:,11), '--', t,P_out_lin(:,12),'--', 'LineWidth',width);
title('w_o_p - Oil produced')
ylabel('kg/s');
xlabel('Hour');
%ylim([45 85]);
legend({'Well 1','Well 2','Well 3','Well 1_l_i_n','Well 2_l_i_n','Well 3_l_i_n'}, 'FontSize',12);
grid('on');
```

Appendix K


```
function [delta_x_next, delta_y] = linearModel(delta_x,delta_u,A,B,C)
% Updating states of linear model

    delta_x_next = A*delta_x + B*delta_u;
    delta_y=C*delta_x;
end
```

Appendix L


```
% Openloop simulation comparing nonlinear model and linear model
% Updating system matrices and operating points at every iteration.

clear
close all

%--------------------------------------------------------------------
% Make symbolic matrices of A, B, C and operating points
[J_A, J_B, J_C, y_sym, u_sym] = CalcJac_y();

% States initial values
% when w_ga = [2.2 2.5 1.8]
m_ga = [9.0127e3    9.8391e3    7.2142e3];
m_gt = [1.4847e3    1.52e3      1.1427e3];
m_lt = [2.1814e4    1.7254e4    2.5813e4];

state_ini_non = [m_ga m_gt m_lt];
state_ini_lin = zeros(9,1);

%Simulation parameters
dt = 15; %[s]          15s is appropriate at "normal" condition
hour = 60; %[h]        hours of simulationtime
Np = 3600*hour/dt;     %Number of datapoints
t=linspace(0,hour,Np); %Used for plotting
l = length(state_ini_non); %Number of states

%Input
%Generate input for whole horizon
w_ga = generateInput(Np);

%Create matrices to store data
P_non = zeros(Np,l); %Matrix for storing nonlinear states
P_lin = P_non;       %Matrix for storing linear states
P_out = zeros(Np,12);%Matrix for storing nonlinear outputs
P_out_lin = P_out;   %Matrix for storing linear outputs


tic;
% Running simulations
for i = 1:Np

    %store the outputs
    P_non(i,:) = state_ini_non;
    P_lin(i,:) = state_ini_lin'+ state_ini_non; %state_ini_non is state operating point

    %Updating matrices and operating points for linear model
    [A,B,C,y_op,u_op] = UpdateMatrices_op_Open(J_A, J_B, J_C,state_ini_non, y_sym, u_sym, dt);

    %Run linear model in deviation form
    [x_next_lin, out_lin] = linearModel(state_ini_lin, w_ga(i,:)'-u_op, A,B,C);
    state_ini_lin = x_next_lin;
    P_out_lin(i,:) = (out_lin+y_op)';


    % Run non-linear model with RK method
    [x_next_non, out_non] = updateState(state_ini_non,dt,w_ga(i,:),1,1,1);
%     [x_next_non, out_non] = updateState_unc(state_ini_non,dt,w_ga(i,:),1,1,1);
    state_ini_non = x_next_non;
    P_out(i,:) = out_non(1,1:12);


end
toc;

%--------------------------------------------------------------------------

%% Plotting

%Defineing colors
newcolors = {'#0072BD', '#D95319', '#EDB120', '#0072BD', '#D95319', '#EDB120'};
colororder(newcolors);
width = 2;

tiledlayout(4,1);
```

```
ax1 = nexttile;
plot(ax1,t,P_out(:,1), t,P_out(:,2), t,P_out(:,3), t,P_out_lin(:,1),'--', ...
    t,P_out_lin(:,2), '--', t,P_out_lin(:,3),'--', 'LineWidth',width);
title(ax1,'P_w_f - Bottom hole pressure');
ylabel(ax1,'bar');
xlabel(ax1,'Hour');
ylim([125 145]);
legend({'Well 1','Well 2','Well 3','Well 1_l_i_n','Well 2_l_i_n','Well 3_l_i_n'}, 'FontSize',12);
grid('on');

ax2 = nexttile;
plot(ax2,t,P_out(:,4), t,P_out(:,5), t,P_out(:,6), t,P_out_lin(:,4),'--', ...
    t,P_out_lin(:,5), '--', t,P_out_lin(:,6),'--', 'LineWidth',width);
title(ax2,'P_w_h - Well head pressure');
ylabel(ax2,'bar');
xlabel(ax2,'Hour');
ylim([30 40]);
grid('on');

ax3 = nexttile;
plot(ax3,t,P_out(:,7), t,P_out(:,8), t,P_out(:,9), t,P_out_lin(:,7),'--', ...
    t,P_out_lin(:,8), '--', t,P_out_lin(:,9),'--', 'LineWidth',width);
title(ax3,'w_l_p - Liquid produced');
ylabel(ax3,'kg/s');
xlabel(ax3,'Hour');
ylim([60 100]);
grid('on');

ax4 = nexttile;
plot(ax4,t,w_ga, 'LineWidth',width);
title(ax4,'w_g_a - Injected lift gas');
ylabel(ax4,'kg/s');
xlabel(ax4,'Hour');
ylim([1 3]);
grid('on');


f2=figure;
colororder(newcolors);
plot(t,P_out(:,10), t,P_out(:,11), t,P_out(:,12), t,P_out_lin(:,10),'--', ...
    t,P_out_lin(:,11), '--', t,P_out_lin(:,12),'--', 'LineWidth',width);
title('w_o_p - Oil produced')
ylabel('kg/s');
xlabel('Hour');
ylim([45 85]);
legend({'Well 1','Well 2','Well 3','Well 1_l_i_n','Well 2_l_i_n','Well 3_l_i_n'}, 'FontSize',12);
grid('on');
```

Appendix M


```
function [w_ga_opt, z] = MPC_low(A,B,C1,C2,N,x0,x_op, w_op_op, w_glp_op, w_ga_op, w_ga_max, max_glp)

%-----------------------------------------------------------
% Deterministic linear MPC low-level interface

nx = size(A,1); nu = size(B,2); ny = size(C1,1); nz = N*(nx + nu + 2*ny);

Q = diag([3 3 3]);
P = diag([4 4 4]);

H11 = kron(eye(N),Q);
H22 = kron(eye(N),P);
H33 = zeros(N*ny,N*ny);
H44 = zeros(N*nx,N*nx);
H_mat = blkdiag(H11,(-H22),H33,H44);

c = zeros(nz,1);


%% Equality constraints

%Ae1
Ae1u = -kron(eye(N),B);
Ae1w_op = zeros(N*nx,N*ny);
Ae1w_glp = zeros(N*nx,N*ny);
Ae1x = eye(N*nx)-kron(diag(ones(N-abs(-1),1),-1),A);

Be1 = [A*x0+x_op-A*x_op-B*w_ga_op;kron(ones((N-1),1),x_op-A*x_op-B*w_ga_op)];

%Ae2
Ae2u = zeros(N*ny,N*nu);
Ae2w_op = eye(N*ny);
Ae2w_glp = zeros(N*ny,N*ny);
Ae2x = -kron(eye(N),C1);

Be2 = kron(ones(N,1),w_op_op-C1*x_op);

%Ae3
Ae3u = zeros(N*ny,N*nu);
Ae3w_op = zeros(N*ny,N*ny);
Ae3w_glp = eye(N*ny);
Ae3x = -kron(eye(N),C2);

Be3 = kron(ones(N,1),w_glp_op-C2*x_op);


% Inequality constraint

%Ai1
Ai1u = kron(eye(N),ones(1,3));
Ai1w_op = zeros(N,N*ny);
Ai1w_glp = zeros(N,N*ny);
Ai1x = zeros(N,N*nx);

%Biu1 = ones(N,1)*8;
Biu1 = w_ga_max;
Bil1 = ones(N,1)*(0);

%Ai2
Ai2u = zeros(N,N*nu);
Ai2w_op = zeros(N,N*ny);
Ai2w_glp = kron(eye(N),ones(1,3));
Ai2x = zeros(N,N*nx);

Biu2 = ones(N,1)*max_glp;
Bil2 = ones(N,1)*(0);


% Constraint matrices
Aei = [Ae1u Ae1w_op Ae1w_glp Ae1x;
    Ae2u Ae2w_op Ae2w_glp Ae2x;
    Ae3u Ae3w_op Ae3w_glp Ae3x;
    Ai1u Ai1w_op Ai1w_glp Ai1x;
    Ai2u Ai2w_op Ai2w_glp Ai2x];
```

```
BL = [Be1;Be2;Be3;Bil1;Bil2];
BU = [Be1;Be2;Be3;Biu1;Biu2];


% Bounds

zL = [ones(N*nu,1)*0.5;
    ones(N*(2*ny+nx),1)*(-inf)];

zU = [ones(N*nu,1)*5;
    ones(N*(2*ny+nx),1)*(inf)];


% Casadi optimizer
import casadi.*

H = DM.eye(nz)*H_mat;
A_ = DM(Aei);

g = DM(c);

lbx = zL;
ubx = zU;

lba = BL;
uba = BU;

qp = struct;
qp.h = H.sparsity();
qp.a = A_.sparsity();

opts = struct;
opts.printLevel = 'low'; % 'none' % Defines what information printed to command window
opts.nWSR = 40000; %Specify number of max iterations
opts.error_on_fail = false; %If false, the MPC continue when error

S = conic('S','qpoases',qp, opts);

%
r = S('h', H, 'g', g,...
      'a', A_, 'lba', lba, 'uba', uba,'lbx', lbx, 'ubx', ubx);
x_opt = r.x;

z = full(x_opt);
w_ga_opt = full(x_opt(1:3));
```

Appendix N


```
clear
close all

%Closed loop simulation with deterministic MPC and nominal model


% Initial states
m_ga = [8.1180e3    7.6192e3    8.0600e3];
m_gt = [1.4714e3    1.4879e3    1.1618e3];
m_lt = [2.2042e4    1.7740e4    2.5451e4];

x0 = [m_ga m_gt m_lt]';
x_op = x0;

%Simulation parameters
N = 10;                 %Prediction horizon
dt = 15; %[s]           15s is appropriate at "normal" condition
hour = 5; %[h]          hours of simulationtime
Np = 3600*hour/dt;      %Number of datapoints
t=linspace(0,hour,Np); %Used for plotting
l = length(x0);         %Number of states
grouping = false;       % Grouping inputs true/false
Ng = 3;                 % Number of groups, 3 or 5

% Create matrices for storing information
P_non = zeros(Np,l); %Matrix for storing states from nonlinear model
P_out = zeros(Np,14);%Matrix for storing outputs from nonlinear model
P_in = zeros(Np,3);  %Matrix for storing inputs given by MPC

w_gc_max = injGas(Np,N);

%Make function of matrices
[J_A, J_B, J_C, y1_op, y2_op, u_op] = Jac_func_op(1,1,1);


z_out = zeros(180,Np); % Number of unknowns
T = zeros(Np,2);

max_glp = 305;
tStart = tic;
for i = 1:Np
    tStart1 = tic;

    %store the states
    P_non(i,:) = x0;

    w_GA_max = w_gc_max(i:i+N-1); % Gas injection limit

    %Updating model matrices and operating points
    [A,B,C1,C2,w_op_op, w_glp_op, w_ga_op] = UpdateMatrices_op(J_A, J_B, J_C, ...
        x_op, y1_op, y2_op, u_op, dt);


    %MPC - choose one of the MPC's
    tStart2 = tic;
%     [w_ga, z] = MPC_high(A,B,C1,C2,N,x0, x_op, w_op_op, w_glp_op, w_ga_op, ...
%         w_GA_max, max_glp, grouping, Ng);
%     [w_ga, z] = MPC_high_slack(A,B,C1,C2,N,x0, x_op, w_op_op, w_glp_op, ...
%         w_ga_op, w_GA_max, max_glp, grouping, Ng);
    [w_ga, z] = MPC_low(A,B,C1,C2,N,x0, x_op, w_op_op, w_glp_op, w_ga_op, w_GA_max, max_glp);
    T(i,2) = toc(tStart2);

    %Running simulator with RK method
    [x_next_non, out_non] = updateState(x0',dt,w_ga',1,1,1);

    x0 = x_next_non';

    %Storing outputs values from non-linaer model.
    P_out(i,:) = [out_non(1,1:12),sum(out_non(1,10:12)), sum(out_non(1,16:18))];

    %Setting states operating point equal to next initial states.
    x_op=x0;

    %Storing the optimized z values
```

```
    z_out(:,i) = z;

    %Storing input
    P_in(i,:) = w_ga';

    T(i,1) = toc(tStart1);

    %Printing progress to command window
    prosent = i*100/(Np);
    text = ['Completed ', num2str(prosent), ' %'];
    disp(text)
end
tEnd = toc(tStart)


%% ---------------------------------------------------------------------
%Plotting

%Defineing colors
newcolors1 = {'#0072BD', '#D95319', '#EDB120', '#0072BD', '#D95319', '#EDB120'};
newcolors2 = {'#77AC30', '#0072BD', '#D95319', '#EDB120'};
colororder(newcolors1);
width = 2;

tiledlayout(4,1);
ax1 = nexttile;
plot(ax1,t,P_out(:,1), t,P_out(:,2), t,P_out(:,3), 'LineWidth',width);
title(ax1,'P_w_f - Bottom hole pressure');
ylabel(ax1,'bar');
xlabel(ax1,'Hour');
%ylim([120 150]);
legend({'Well 1','Well 2','Well 3'}, 'FontSize',12);
grid('on');

ax2 = nexttile;
plot(ax2,t,P_out(:,4), t,P_out(:,5), t,P_out(:,6), 'LineWidth',width);
title(ax2,'P_w_h - Well head pressure');
ylabel(ax2,'bar');
xlabel(ax2,'Hour');
%ylim([30 40]);
grid('on');

ax3 = nexttile;
plot(ax3,t,P_out(:,7), t,P_out(:,8), t,P_out(:,9), 'LineWidth',width);
title(ax3,'w_l_p - Liquid produced');
ylabel(ax3,'kg/s');
xlabel(ax3,'Hour');
%ylim([50 100]);
grid('on');

ax4 = nexttile;
plot(ax4,t,P_in,t,sum(P_in,2),t,w_gc_max(1:end-N), 'LineWidth',width);
title(ax4,'w_g_a - Injected lift gas');
ylabel(ax4,'kg/s');
xlabel(ax4,'Hour');
%ylim([1 3]);
grid('on');

max_glp_=linspace(max_glp,max_glp,Np);

f2=figure;
%colororder(newcolors);
plot(t,P_out(:,10), t,P_out(:,11), t,P_out(:,12), t,P_out(:,13),t,P_out(:,14),t,max_glp_,'--r',
'LineWidth',width);
title('Produced w_o_p and w_g_l_p')
ylabel('kg/s');
xlabel('Hour');
%ylim([45 85]);
legend({'w_o_p Well 1','w_o_p Well 2','w_o_p Well 3','Total w_o_p', 'Total w_g_l_p','w_s^m^a^x'},
'FontSize',12);
grid('on');

% Plot results
```

```
f3 = figure;
tiledlayout(2,1);

colororder(newcolors2);
ax1 = nexttile;
% plot(ax1,t,P_out(:,14),t,P_out(:,13),t,max_glp_,'--r',t,max_glp_-2, '--k', 'LineWidth',width);
plot(ax1,t,P_out(:,14),t,P_out(:,13),t,max_glp_,'--r', 'LineWidth',width);
title(ax1,'w_o_p and w_g_l_p produced')
ylabel(ax1,'kg/s');
xlabel(ax1,'Hour');
% legend({'Total w_g_l_p','Total w_o_p','w_s^m^a^x','w_s^m^a^x soft'}, 'FontSize',12);
legend({'Total w_g_l_p','Total w_o_p','w_s^m^a^x'}, 'FontSize',12);
grid('on');

ax2 = nexttile;
plot(ax2,t,sum(P_in,2),t,P_in, t,w_gc_max(1:end-N),'--r', 'LineWidth',width);
title(ax2,'w_g_a - Injected lift gas');
ylabel(ax2,'kg/s');
xlabel(ax2,'Hour');
ylim([1 12]);
legend({'Total w_g_a','Well 1','Well 2','Well 3','w_g_c^m^a^x'}, 'FontSize',12);
grid('on');


f4 = figure;
plot(t,T(:,2))
title('Optimizing time')
ylabel('Optimizing time [s]');
xlabel('Simulation time [Hour]');
%legend({'Optimizing time'}, 'FontSize',12);
grid('on');
```

Appendix O


```
function [Ac, Bc, Cc, w_op_op, w_glp_op, w_ga_op] = Jac_func_op(PI_dev,WC_dev,GOR_dev)

% This .m file generate function of Ac, Bc, Cc matrices and function
% of operating point w_op_op w_glp_op, w_ga_op. Used in linear MPC
% In deterministic MPC input arguments are (1,1,1).

syms m_ga m_gt m_lt w_ga P_a P_ainj rho_l V_g rho_m P_tinj Y2 rho_ga...
    w_ginj P_wf w_lr w_or w_gr P_wh Y3 w_glp w_gp w_lp w_op...
    Z R T_a M A_a L_a_tl g L_a_vl rho_w WC rho_o A_t L_t_tl...
    T_t L_t_vl a_Y K L_r_vl PI P_r GOR P_s C_V

assume(Z > 0);
assume(R > 0);
assume(T_a > 0);
assume(M > 0);
assume(A_a > 0);
assume(L_a_tl > 0);
assume(g > 0);
assume(L_a_vl > 0);
assume(rho_w > 0);
assume(WC > 0);
assume(rho_o > 0);
assume(A_t > 0);
assume(L_t_tl > 0);
assume(T_t > 0);
assume(L_t_vl > 0);
assume(a_Y > 0);
assume(K > 0);
assume(L_r_vl > 0);
assume(PI > 0);
assume(P_r > 0);
assume(GOR > 0);
assume(P_s > 0);
assume(C_V > 0);

assume(m_ga>0);
assume(m_gt>0);
assume(m_lt>0);

assume(P_ainj>0);
assume(P_ainj-P_tinj>0);

assume(P_wh>0);
assume(P_wh-P_s>0);

assume(P_r - P_wf>0);

wells = 3;

%States
m_ga = sym('m_ga', [1 wells]);
m_gt = sym('m_gt', [1 wells]);
m_lt = sym('m_lt', [1 wells]);


%Input
w_ga = sym('w_ga', [1 wells]);

%% Parameters

% Well    = [Well1  Well2   Well3]
PI        = sym('PI', [1 wells]);
WC        = sym('WC', [1 wells]);
GOR       = sym('GOR', [1 wells]);
L_a_tl    = sym('L_a_tl', [1 wells]);
L_a_vl    = sym('L_a_vl', [1 wells]);
L_r_vl    = sym('L_r_vl', [1 wells]);
K         = sym('K', [1 wells]);
T_a       = sym('T_a', [1 wells]);
T_t       = sym('T_t', [1 wells]);
L_t_tl    = sym('L_t_tl', [1 wells]);
L_t_vl    = sym('L_t_vl', [1 wells]);
```

```
% 4
P_a = ((Z.*m_ga.*R.*T_a)./(M.*A_a.*L_a_tl))*1e-5; %[bar]

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

% 6
rho_l = rho_w.*WC+rho_o.*(1-WC);      %[kg/m3]

% 7
V_g = A_t.*L_t_tl - m_lt./rho_l;      %[m3]

% 8
rho_m = (m_gt+m_lt)./(A_t*L_t_tl);   %[kg/m3]

% 9
P_tinj = ((Z.*m_gt.*R.*T_t)./(M.*V_g) + 0.5.*rho_m.*g.*L_t_vl)*1e-5; %[bar]

% 10
Y2 = 1 - a_Y.*((P_ainj - P_tinj)./(P_ainj));

% 11
rho_ga = m_ga./(A_a.*L_a_tl);         %[kg/m3]

% 12
w_ginj = K.*Y2.*sqrt(rho_ga.*(P_ainj-P_tinj))/3600;

% 13
P_wf = P_tinj + (rho_l.*g.*L_r_vl)*1e-5;  %[bar]

% 14
w_lr = PI.*(P_r - P_wf)/3600;   %[kg/s]

% 15
w_or = (rho_o./rho_l).*(1-WC).*w_lr;  %[kg/s]

% 16
w_gr = GOR.*w_or;   %[kg/s]

% 17
P_wh = (((Z.*m_gt.*R.*T_t)./(M.*V_g)) - 0.5.*rho_m.*g.*L_t_vl)*1e-5; %[bar]

% 18
Y3 = 1 - a_Y.*(P_wh - P_s)./P_wh;

% 19
w_glp = C_V.*Y3.*sqrt(rho_m.*(P_wh-P_s))/3600;  %[kg/s]

% 20
w_gp = (m_gt./(m_gt+m_lt)).*w_glp;    %[kg/s]

% 21
w_lp = (m_lt./(m_gt+m_lt)).*w_glp;    %[kg/s]

% 22
w_op = (rho_o./rho_l).*(1-WC).*w_lp;   %[kg/s]

%1.ODE
dev_m_ga = w_ga - w_ginj;

%2.ODE
dev_m_gt = w_ginj + w_gr - w_gp;

%3.ODE
dev_m_lt = w_lr - w_lp;

%-----------------------------------------------------------------
%Paramter values that will be substituted

% Well    = [Well1  Well2   Well3]
PI_        = [2.51   1.63    1.62]*1e4 .* PI_dev;
WC_        = [0.20   0.10    0.25] .* WC_dev;
GOR_       = [0.05   0.07    0.03] .* GOR_dev;
```

```
L_a_tl_    = [2758   2559    2677];
L_a_vl_    = [2271   2344    1863];
L_r_vl_    = [114    67      61];
K_         = [68.43  67.82   67.82];
T_a_       = [280    280     280];
T_t_       = T_a_;
L_t_tl_    = L_a_tl_;
L_t_vl_    = L_a_vl_;


Z_ = 1.3;
R_ = 8.31446261815324;
M_ = 20*1e-3;    %Molar mass
A_a_ = 0.0174;   %Annulus cross section area
g_ = 9.80665;    %Gravity acceleration constant
rho_w_ = 1000;   %Density of water
rho_o_ = 800;    %Density of oil
A_t_ = 0.0194;   %Annulus cross section area
a_Y_ = 0.66;
P_r_ = 150;      %Reservoir pressure
P_s_ = 30;       %Separator pressure
Cv_ = 8190;      %Constant Cv


%Generate jacobians
dm = [dev_m_ga dev_m_gt dev_m_lt];
x = [m_ga m_gt m_lt];
y = [w_op w_glp];

J_A=jacobian(dm, x);
J_B=jacobian(dm, w_ga);
J_C=jacobian(y, x);


%Substituting
%Substitute the paramteres and the new states to the matrices.
A = subs(J_A,[A_a, A_t, a_Y, C_V, g, GOR, K, L_a_tl, L_a_vl, L_r_vl,...
    L_t_tl, L_t_vl, M, P_r, P_s, PI, R, rho_o, rho_w, T_a, T_t, WC, Z],...
    [A_a_, A_t_, a_Y_, Cv_, g_, GOR_, K_, L_a_tl_, L_a_vl_, L_r_vl_,...
    L_t_tl_, L_t_vl_, M_, P_r_, P_s_, PI_, R_, rho_o_, rho_w_, T_a_, T_t_, WC_, Z_]);

C = subs(J_C,[A_a, A_t, a_Y, C_V, g, GOR, K, L_a_tl, L_a_vl, L_r_vl,...
    L_t_tl, L_t_vl, M, P_r, P_s, PI, R, rho_o, rho_w, T_a, T_t, WC, Z],...
    [A_a_, A_t_, a_Y_, Cv_, g_, GOR_, K_, L_a_tl_, L_a_vl_, L_r_vl_,...
    L_t_tl_, L_t_vl_, M_, P_r_, P_s_, PI_, R_, rho_o_, rho_w_, T_a_, T_t_, WC_, Z_]);


Bc = double(J_B);
Ac = matlabFunction(A);
Cc = matlabFunction(C);


%Outputs operating points
y1 = subs(w_op,[A_a, A_t, a_Y, C_V, g, GOR, K, L_a_tl, L_a_vl, L_r_vl,...
    L_t_tl, L_t_vl, M, P_r, P_s, PI, R, rho_o, rho_w, T_a, T_t, WC, Z],...
    [A_a_, A_t_, a_Y_, Cv_, g_, GOR_, K_, L_a_tl_, L_a_vl_, L_r_vl_,...
    L_t_tl_, L_t_vl_, M_, P_r_, P_s_, PI_, R_, rho_o_, rho_w_, T_a_, T_t_, WC_, Z_]);

y2 = subs(w_glp,[A_a, A_t, a_Y, C_V, g, GOR, K, L_a_tl, L_a_vl, L_r_vl,...
    L_t_tl, L_t_vl, M, P_r, P_s, PI, R, rho_o, rho_w, T_a, T_t, WC, Z],...
    [A_a_, A_t_, a_Y_, Cv_, g_, GOR_, K_, L_a_tl_, L_a_vl_, L_r_vl_,...
    L_t_tl_, L_t_vl_, M_, P_r_, P_s_, PI_, R_, rho_o_, rho_w_, T_a_, T_t_, WC_, Z_]);

w_op_op = matlabFunction(y1);
w_glp_op = matlabFunction(y2);

%Inputs operating points
% w_ga_op = w_ginj;
u = subs(w_ginj,[A_a, A_t, a_Y, C_V, g, GOR, K, L_a_tl, L_a_vl, L_r_vl,...
    L_t_tl, L_t_vl, M, P_r, P_s, PI, R, rho_o, rho_w, T_a, T_t, WC, Z],...
    [A_a_, A_t_, a_Y_, Cv_, g_, GOR_, K_, L_a_tl_, L_a_vl_, L_r_vl_,...
    L_t_tl_, L_t_vl_, M_, P_r_, P_s_, PI_, R_, rho_o_, rho_w_, T_a_, T_t_, WC_, Z_]);

w_ga_op = matlabFunction(u);
```

Appendix P


```
function [A,B,C1,C2,y1_op,y2_op, u_op] = UpdateMatrices_op(Ac, Bc, Cc , x, y1, y2, u, dt)

% Substitute states into system matrices and operating point vectors in closed loop simulation

Ac_ = Ac(x(1), x(2), x(3), x(4), x(5), x(6), x(7), x(8), x(9));
Cc_ = Cc(x(4), x(5), x(6), x(7), x(8), x(9));


sys = ss(Ac_,Bc,Cc_,0);
d_sys = c2d(sys,dt);

A=d_sys.a;
B=d_sys.b;
C=d_sys.c;

C1 = C(1:3,:);
C2 = C(4:6,:);


%Outout operation points
y1_op_ = y1(x(4),x(5),x(6),x(7),x(8),x(9));
y2_op_ = y2(x(4),x(5),x(6),x(7),x(8),x(9));

y1_op = y1_op_';
y2_op = y2_op_';

%Input operation points
u_op_ = u(x(1), x(2), x(3), x(4), x(5), x(6), x(7), x(8), x(9));
u_op = u_op_';
```

Appendix Q


```
function w_gc_max = injGas(Np,N)

% Generate w_gc_max

w_gc_max = zeros(Np+N,1);

section=Np/3;

%-------Ramping----------

for i = 1:Np+N
    if i < section
        w_gc_max(i) = 10;

    elseif i < 2*section
        if w_gc_max(i-1) > 8.01
            w_gc_max(i) = w_gc_max(i-1) - 0.01;
        else
            w_gc_max(i) = w_gc_max(i-1);
        end

    else
        if w_gc_max(i-1) < 10
            w_gc_max(i) = w_gc_max(i-1) + 0.01;
        else
            w_gc_max(i) = w_gc_max(i-1);
        end

    end

end

%-------Stepping----------
% for i = 1:Np+N
%     if i < section
%         w_gc_max(i) = 10;
%
%     elseif i < 2*section
%         w_gc_max(i) = 8;
%
%     else
%         w_gc_max(i) = 10;
%
%     end
%
% end
```

Appendix R


```
function [w_ga_opt, z] = MPC_high(A,B,C1,C2,N,x0,x_op, w_op_op, w_glp_op, w_ga_op, w_ga_max, max_glp,
grouping, Ng)

% Deterministic linear MPC high-level interface

import casadi.*

wells = 3; % Number of wells

% Symbolic states
m_ga = SX.sym('m_ga',wells);
m_gt = SX.sym('m_gt',wells);
m_lt = SX.sym('m_lt',wells);
states = [m_ga;m_gt;m_lt]; n_states = length(states);

% A matrix that represents the states over the optimization problem.
X = SX.sym('X',n_states,(N+1));

% Input
w_ga = SX.sym('w_ga',wells);
controls = w_ga; n_controls = length(controls);

Ni=N; %Number of inputs

U = SX.sym('U', n_controls,N); % Decision variables (controls)
Ug = SX.sym('Ug', n_controls,Ng); % Decision variables (controls) grouped
P = SX.sym('P',n_states);


% Grouping of control signal w_ga

if grouping == true && Ng == 5
    Ni=Ng;
    for i = 1:N
        if i<=5
            U(:,i)=Ug(:,1);
        elseif i<= 0.2*N
            U(:,i)=Ug(:,2);
        elseif i<= 0.4*N
            U(:,i)=Ug(:,3);
        elseif i<= 0.6*N
            U(:,i)=Ug(:,4);
        else
            U(:,i)=Ug(:,5);
        end
    end
end

if grouping == true && Ng == 3
    Ni=Ng;
    for i = 1:N
        if i<=5
            U(:,i)=Ug(:,1);
        elseif i<= 0.5*N
            U(:,i)=Ug(:,2);
        else
            U(:,i)=Ug(:,3);
        end
    end
end


obj = 0; % Objective function
g1 = SX.sym('g1',N,1);
g2 = SX.sym('g2',N,1);

%Weighting matrices when N=240
Q = diag([420 425 430]); %Weighting matrix for input
R = diag([10 10 10]); % Weighting matrix for output

%Weigting matrices when N=50
% Q = diag([15 20 25]);
% R = diag([1 1 1]);
```

```
% %Weigting matrices when N=25
% Q = diag([90 95 100]);
% R = diag([10 10 10]);

X(:,1) = P(1:9); % initial state
for k = 1:N

    % Compute solution symbolically
    con = U(:,k);
    st = X(:,k);
    st_next = A*st + B*con - A*x_op-B*w_ga_op + x_op;
    X(:,k+1) = st_next;

    % Compute objective
    w_op = C1*st - C1*x_op+w_op_op;
    obj = obj + -(w_op'*R*w_op) + con'*Q*con;

    % Compute constraints
    g1(k,1) = sum(U(:,k));
    g2(k,1) = sum(C2*st-C2*x_op+w_glp_op);


end

g = [g1;g2];

if grouping == false
    OPT_variables = reshape(U,3*N,1);
else
    OPT_variables = reshape(Ug,3*Ng,1);
end

prob = struct('f', obj, 'x', OPT_variables, 'g', g, 'p', P);

%---Choose solver---------


% opts = struct;
% opts.ipopt.max_iter = 10000;
% opts.ipopt.print_level = 0; %0,3
% opts.print_time = 0; %0,1
% opts.ipopt.acceptable_tol = 1e-8;
% % optimality convergence tolerance
% opts.ipopt.acceptable_obj_change_tol = 1e-6;
%
% solver = nlpsol('solver', 'ipopt', prob, opts);

optsQP = struct;
optsQP.printLevel = 'low';
optsQP.nWSR = 20000; %Specify number of max iterations
optsQP.error_on_fail = false; %If false, the MPC continue when error

solver = qpsol('solver', 'qpoases', prob, optsQP);

%%
args = struct;

% inequality constrains (state constraints)
args.lbg = zeros(2*N,1); % lower bound of the states x and y
args.ubg = [w_ga_max; ones(N,1)*max_glp];  % upper bound of the states x and y

% input constraints
args.lbx(1:3*Ni,1) = 0.5;
args.ubx(1:3*Ni,1) = 5;

args.p = x0;
args.x0 = kron(ones(Ni,1),w_ga_op);


sol = solver('x0', args.x0, 'lbx', args.lbx, 'ubx', args.ubx,...
        'lbg', args.lbg, 'ubg', args.ubg, 'p',args.p);

z=full(sol.x);
w_ga_opt = z(1:3);
```

Appendix S


```
function [w_ga_opt, z] = MPC_high_slack(A,B,C1,C2,N,x0,x_op, w_op_op, w_glp_op, w_ga_op, w_ga_max,
max_glp, grouping, Ng)

import casadi.*

% Deterministic linear MPC high-level interface with soft constraint

wells = 3; % Number of wells

m_ga = SX.sym('m_ga',wells);
m_gt = SX.sym('m_gt',wells);
m_lt = SX.sym('m_lt',wells);
states = [m_ga;m_gt;m_lt]; n_states = length(states);


%%
% A matrix that represents the states over the optimization problem.
X = SX.sym('X',n_states,(N+1));

% Input
w_ga = SX.sym('w_ga',wells);
controls = w_ga; n_controls = length(controls);


%%
Ni=N; %Number of inputs

U = SX.sym('U', n_controls,N); % Decision variables (controls)
Ug = SX.sym('Ug', n_controls,Ng); % Decision variables (controls) grouped
P = SX.sym('P',n_states); %Parameter input, used for initial states
S1 = SX.sym('S1', 1,N); %Slack varable


% Grouping of control signal w_ga

if grouping == true && Ng == 3
    Ni=Ng;
    for i = 1:N
        if i<=5
            U(:,i)=Ug(:,1);
        elseif i<= 0.5*N
            U(:,i)=Ug(:,2);
        else
            U(:,i)=Ug(:,3);
        end
    end
end

%%
obj = 0; % Objective function
g1 = SX.sym('g1',N,1);
g2 = SX.sym('g2',N,1);
g3 = SX.sym('g3',N,1);

% %Weighting matrices when N=240
Q = diag([420 425 430]); %Weighting matrix for input
R = diag([10 10 10]); % Weighting matrix for output
S = 400; % Weighting scalar for slack variable


%Weigting matrices when N=25
% Q = diag([15 20 25]); %8
% R = diag([1 1 1]);
% S = 1;

X(:,1) = P(1:9); % initial state
for k = 1:N

    % Compute solution symbolically
    con = U(:,k);
    st = X(:,k);
    s1 = S1(:,k);
    st_next = A*st + B*con - A*x_op-B*w_ga_op + x_op;
    X(:,k+1) = st_next;
```

```
    % Compute objective
    w_op = C1*st - C1*x_op+w_op_op;
    obj = obj + -(w_op'*R*w_op) + con'*Q*con + s1^2*S;

    % Compute constraints
    g1(k,1) = sum(U(:,k));
    g2(k,1) = sum(C2*st-C2*x_op+w_glp_op);
    g3(k,1) = sum(C2*st-C2*x_op+w_glp_op) - s1;


end

g = [g1;g2;g3];

if grouping == false
    OPT_variables = [reshape(U,3*N,1);reshape(S1,N,1)];
else
    OPT_variables = [reshape(Ug,3*Ng,1);reshape(S1,N,1)];
end

nlp_prob = struct('f', obj, 'x', OPT_variables, 'g', g, 'p', P);

opts = struct;
opts.ipopt.max_iter = 10000;
%opts.ipopt.print_level = 0; %0,3
opts.print_time = 0; %0,1
opts.ipopt.acceptable_tol = 1e-8;
% optimality convergence tolerance
opts.ipopt.acceptable_obj_change_tol = 1e-6;

solver = nlpsol('solver', 'ipopt', nlp_prob, opts);

% optsQP = struct;
% optsQP.printLevel = 'low';
% optsQP.nWSR = 20000; %Specify number of max iterations
% optsQP.error_on_fail = false; %If false, the MPC continue when error
%
% solver = qpsol('solver', 'qpoases', nlp_prob, optsQP);

%%
args = struct;

% inequality constrains
args.lbg = zeros(3*N,1); % lower bound of constraints
args.ubg = [w_ga_max; ones(N,1)*max_glp; ones(N,1)*(max_glp - 5)];  % upper bound of constraints

% input constraints
args.lbx = [ones(3*Ni,1)*0.5;zeros(N,1)];
args.ubx = [ones(3*Ni,1)*5;ones(N,1)*inf];

args.p = x0;
args.x0 = [kron(ones(Ni,1),w_ga_op);zeros(N,1)];


sol = solver('x0', args.x0, 'lbx', args.lbx, 'ubx', args.ubx,...
        'lbg', args.lbg, 'ubg', args.ubg, 'p',args.p);

z=full(sol.x);
w_ga_opt = z(1:3);
```

Appendix T


```
clear
close all

% Closed loop: deterministic MPC and uncertain model

%Simulation parameters
N = 240;                %Prediction horizon
dt = 15; %[s]           15s is appropriate at "normal" condition
hour = 12; %[h]         hours of simulationtime
Np = 3600*hour/dt;      %Number of datapoints
t=linspace(0,hour,Np); %Used for plotting

max_glp = 300;

% up to 10 % deviation from parameters
% Well         = [Well1  Well2   Well3]
PI_dev         = [0    0     0;
                  0.1   0.1   0.1;
                 -0.1   0.1   0.1;
                  0.1  -0.1   0.1;
                 -0.1  -0.1   0.1;
                  0.1   0.1  -0.1;
                 -0.1   0.1  -0.1;
                  0.1  -0.1  -0.1;
                 -0.1  -0.1  -0.1;
                 randi([-100 100],6,3)/1000] +1;

% WC_dev         = [0    0     0;
%                   0.1   0.1  -0.1;
%                  -0.1   0.1  -0.1;
%                   0.1  -0.1  -0.1;
%                  -0.1  -0.1  -0.1;
%                   0.1   0.1   0.1;
%                  -0.1   0.1   0.1;
%                   0.1  -0.1   0.1;
%                  -0.1  -0.1   0.1;
%                  randi([-100 100],6,3)/1000] +1;
%
% GOR_dev        = [0    0     0;
%                   0.1  -0.1  -0.1;
%                  -0.1  -0.1  -0.1;
%                   0.1   0.1   0.1;
%                  -0.1   0.1   0.1;
%                   0.1  -0.1   0.1;
%                  -0.1  -0.1   0.1;
%                   0.1   0.1  -0.1;
%                  -0.1   0.1  -0.1;
%                  randi([-100 100],6,3)/1000] +1;


% up to 20 % deviation, only WC and GOR

WC_dev         = [0    0     0;
                  0.2   0.2  -0.2;
                 -0.2   0.2  -0.2;
                  0.2  -0.2  -0.2;
                 -0.2  -0.2  -0.2;
                  0.2   0.2   0.2;
                 -0.2   0.2   0.2;
                  0.2  -0.2   0.2;
                 -0.2  -0.2   0.2;
                 randi([-200 200],6,3)/1000] +1;

GOR_dev        = [0    0     0;
                  0.2  -0.2  -0.2;
                 -0.2  -0.2  -0.2;
                  0.2   0.2   0.2;
                 -0.2   0.2   0.2;
                  0.2  -0.2   0.2;
                 -0.2  -0.2   0.2;
                  0.2   0.2  -0.2;
                 -0.2   0.2  -0.2;
                 randi([-200 200],6,3)/1000] +1;
```

```
PI_len = length(PI_dev);

P_out_all = [];
P_in_all = [];
T_all = [];

%% -------------------------------------------------------
tStart = tic;

for i=1:PI_len

[P_out, P_in, T] = simulation_unc(N, Np, dt, max_glp, true, 3, PI_dev(i,:), WC_dev(i,:),GOR_dev(i,:));

P_out_all = [P_out_all;P_out];
P_in_all = [P_in_all;P_in];
T_all = [T_all, T];

end
%-------------------------------------------------------
tEnd = toc(tStart)


%% Plotting

color1 = [0, 0.4470, 0.7410];
color2 = [0.8500, 0.3250, 0.0980];
color3 = [0.9290, 0.6940, 0.1250];
color4 = [0.4940, 0.1840, 0.5560];
color5 = [0.4660, 0.6740, 0.1880];
color6 = [0.2, 0.2, 0.2];
width = 1;
k=1+Np;

f1 = figure;
max_glp_=linspace(max_glp,max_glp,Np);

for i=2:PI_len
    plot(t,P_out_all(k:k+Np-1,10),'color',color6,'LineWidth',width,'HandleVisibility','off');
    hold on
    plot(t,P_out_all(k:k+Np-1,11),'color',color6,'LineWidth',width,'HandleVisibility','off');
    plot(t,P_out_all(k:k+Np-1,12),'color',color6,'LineWidth',width,'HandleVisibility','off');
    plot(t,P_out_all(k:k+Np-1,13),'color',color6,'LineWidth',width,'HandleVisibility','off');
    plot(t,P_out_all(k:k+Np-1,14),'color',color6,'LineWidth',width,'HandleVisibility','off');

    k=k+Np;
end

width = 3;
k=1;
plot(t,P_out_all(k:k+Np-1,10),'color',color1,'LineWidth',width,'DisplayName','Well 1 w_o_p ');
plot(t,P_out_all(k:k+Np-1,11),'color',color2,'LineWidth',width,'DisplayName','Well 2 w_o_p ');
plot(t,P_out_all(k:k+Np-1,12),'color',color3,'LineWidth',width,'DisplayName','Well 3 w_o_p ');
plot(t,P_out_all(k:k+Np-1,13),'color',color4,'LineWidth',width,'DisplayName','Total w_o_p ');
plot(t,P_out_all(k:k+Np-1,14),'color',color5,'LineWidth',width,'DisplayName','Total w_g_l_p');
plot(t,max_glp_,'--r','LineWidth',2, 'DisplayName','w_s^m^a^x');

title('w_o_p and w_g_l_p - Produced')
ylabel('kg/s');
xlabel('Hour');

legend('FontSize',12);
grid('on');
hold off


f2=figure;

width = 1;
k=1+Np;

for i=2:PI_len
    plot(t,P_in_all(k:k+Np-1,1),'color',color6,'LineWidth',width,'HandleVisibility','off');
    hold on
    plot(t,P_in_all(k:k+Np-1,2),'color',color6,'LineWidth',width,'HandleVisibility','off');
```

```
    plot(t,P_in_all(k:k+Np-1,3),'color',color6,'LineWidth',width,'HandleVisibility','off');

    k=k+Np;

end

width = 3;
k=1;
plot(t,P_in_all(k:k+Np-1,1),'color',color1,'LineWidth',width,'DisplayName','Well 1');
plot(t,P_in_all(k:k+Np-1,2),'color',color2,'LineWidth',width,'DisplayName','Well 2');
plot(t,P_in_all(k:k+Np-1,3),'color',color3,'LineWidth',width,'DisplayName','Well 3');

title('w_g_a - Injected lift gas')
ylabel('kg/s');
xlabel('Hour');
legend('FontSize',12);
grid('on');
hold off

% Plotting w_ga
%----1
f3 = figure;
tiledlayout(4,1);

ax1 = nexttile;
width = 1;
k=1+Np;

for i=2:PI_len
    plot(ax1,t,P_in_all(k:k+Np-1,1),'color',color6,'LineWidth',width,'HandleVisibility','off');
    hold on

    k=k+Np;

end

width = 3;
k=1;
plot(ax1, t,P_in_all(k:k+Np-1,1),'color',color1,'LineWidth',width,'DisplayName','Well 1');
title(ax1,'w_g_a - Injected lift gas')
ylabel(ax1, 'kg/s');
%xlabel(ax1, 'Hour');
ylim([0 6]);
legend('FontSize',12);
grid('on');
hold off

%-----2
ax2 = nexttile;
width = 1;
k=1+Np;

for i=2:PI_len
    plot(ax2,t,P_in_all(k:k+Np-1,2),'color',color6,'LineWidth',width,'HandleVisibility','off');
    hold on

    k=k+Np;

end

width = 3;
k=1;
plot(ax2, t,P_in_all(k:k+Np-1,2),'color',color2,'LineWidth',width,'DisplayName','Well 2');
%title(ax2,'w_g_a - Injected lift gas well 2')
ylabel(ax2, 'kg/s');
%xlabel(ax2, 'Hour');
ylim([0 6]);
legend('FontSize',12);
grid('on');
hold off

%-----3
ax3 = nexttile;
width = 1;
```

```
k=1+Np;

for i=2:PI_len
    plot(ax3,t,P_in_all(k:k+Np-1,3),'color',color6,'LineWidth',width,'HandleVisibility','off');
    hold on

    k=k+Np;

end

width = 3;
k=1;
plot(ax3, t,P_in_all(k:k+Np-1,3),'color',color3,'LineWidth',width,'DisplayName','Well 3');
%title(ax3,'w_g_a - Injected lift gas well 3')
ylabel(ax3, 'kg/s');
%xlabel(ax3, 'Hour');
ylim([0 6]);
legend('FontSize',12);
grid('on');
hold off

%---4
%sum(P_in_all,2)

w_gc_max = injGas(Np,N);

ax4 = nexttile;
width = 1;
k=1+Np;

for i=2:PI_len
    plot(ax4,t,sum(P_in_all(k:k+Np-1,:),2),'color',color6,'LineWidth',width,'HandleVisibility','off');
    hold on

    k=k+Np;
end

width = 3;
k=1;
plot(ax4, t, sum(P_in_all(k:k+Np-1,:),2),'color',color4,'LineWidth',width,'DisplayName','Total w_g_a');
plot(ax4, t, w_gc_max(1:end-N),'--r','LineWidth',2 , 'DisplayName','w_g_c^m^a^x');
ylabel(ax4, 'kg/s');
xlabel(ax4, 'Hour');
ylim([2 12]);
legend('FontSize',12);
grid('on');
hold off

% Plotting optimizing time
t2 = linspace(0,PI_len,Np*PI_len);

f4=figure;

plot(t2,T_all)
title('Optimizing time')
ylabel('Optimizing time [s]');
xlabel('Simulation number');
grid('on');
```

Appendix U


```
function [P_out, P_in, T] = simulation_unc(N, Np, dt, max_glp, grouping, Ng, PI_dev, WC_dev, GOR_dev)

% Simulation interface deterministic MPC and uncertain model

% Initial states
m_ga = [8.1180e3    7.6192e3    8.0600e3];
m_gt = [1.4714e3    1.4879e3    1.1618e3];
m_lt = [2.2042e4    1.7740e4    2.5451e4];

x0 = [m_ga m_gt m_lt]';
x_op = x0;

w_ga_max = injGas(Np,N);

%Make function of matrices
[J_A, J_B, J_C, y1_op, y2_op, u_op] = Jac_func_op(1,1,1);

% For no grouping
P_out = zeros(Np,14);%Matrix for storing outputs from nonlinear model
P_in = zeros(Np,3);  %Matrix for storing inputs given by MPC
T = zeros(1,Np); % Storing iteration time no grouping

for i = 1:Np
    %tic
    w_GA_max = w_ga_max(i:i+N-1);
    %Updating model matrices and operating points
    [A,B,C1,C2,w_op_op, w_glp_op, w_ga_op] = UpdateMatrices_op(J_A, J_B, J_C, ...
        x_op, y1_op, y2_op, u_op, dt);

    tic

    %MPC
    w_ga = MPC_high(A,B,C1,C2,N,x0, x_op, w_op_op, w_glp_op, w_ga_op, w_GA_max, ...
        max_glp, grouping, Ng);

    T(i) = toc;

    %Running simulator with RK method
    [x_next_non, out_non] = updateState(x0',dt,w_ga', PI_dev, WC_dev, GOR_dev);
    x0 = x_next_non';

    % summing all w_op and w_glp
    P_out(i,:) = [out_non(1,1:12),sum(out_non(1,10:12)), sum(out_non(1,16:18))];

    x_op=x0;

    P_in(i,:) = w_ga';

    prosent = i*100/(Np);
    text = ['Completed ', num2str(prosent), ' %'];
    disp(text)
end
```

Appendix V


```
function [w_ga_opt, z] = MS_MPC(A,B,C1,C2,N,x0,x_op, w_op_op, w_glp_op, w_ga_op, w_ga_max, max_glp,
grouping)

% Robust Multi-stage linear MPC

import casadi.*

Ns = length(A); %Number of scenarios
Ng = 3; %Number of input groups
wells = 3; % Number of wells

m_ga = SX.sym('m_ga',wells);
m_gt = SX.sym('m_gt',wells);
m_lt = SX.sym('m_lt',wells);
states = [m_ga;m_gt;m_lt]; n_states = length(states);

% A matrix that represents the states over the optimization problem.
X = SX.sym('X',n_states,(N+1),Ns);

% Input
w_ga = SX.sym('w_ga',wells);
controls = w_ga; n_controls = length(controls);

U = SX.sym('U', n_controls,N,Ns); % Decision variables (controls)
Ug = SX.sym('Ug', n_controls,Ng,Ns); % Decision variables (controls) grouped
Us = SX.sym('Us', n_controls,1); % First scenario
P = SX.sym('P',n_states); %Parameter input, used for initial states

%Setting input u equal in first branch.
for i = 1:Ns
    Ug{i}(:,1) = Us;
end

%Building U if grouping or not
if grouping == true
    OPT_variables = SX.sym('OPT_variables',(wells*Ng)+(wells*Ng-3)*(Ns-1),1);
    for j = 1:Ns
        for i = 1:N
            if i<=5
                U{j}(:,i)=Ug{j}(:,1);
            elseif i<= 0.5*N
                U{j}(:,i)=Ug{j}(:,2);
            else
                U{j}(:,i)=Ug{j}(:,3);
            end
        end
    end
else
    OPT_variables = SX.sym('OPT_variables',(wells*N)+(wells*N-3)*(Ns-1),1);
    for j = 1:Ns
        U{j}(:,1)=Us;
    end
end

obj = 0; % Objective function
g1 = SX.sym('g1',N,Ns);
g2 = SX.sym('g2',N,Ns);


%Weighting matrices when N=240
Q = diag([420 425 430]); %Weighting matrix for input
R = diag([10 10 10]); % Weighting matrix for output

%Weigting matrices when N=50
% Q = diag([15 20 25]);
% R = diag([1 1 1]);

% %Weigting matrices when N=25
% Q = diag([90 95 100]);
% R = diag([10 10 10]);

% Weigting matrices when N=10 low
% Q = diag([3 3 3]);
% R = diag([4 4 4]);
```

```
for i=1:Ns
    X{i}(:,1) = P(1:9); % initial state
end

for j = 1:Ns
    for k = 1:N

        % Compute solution symbolically
        con = U{j}(:,k);
        st = X{j}(:,k);
        st_next = A{j}*st + B{j}*con - A{j}*x_op-B{j}*w_ga_op{j} + x_op;
        X{j}(:,k+1) = st_next;

        % Compute objective
        w_op = C1{j}*st - C1{j}*x_op+w_op_op{j};
        obj = obj + -(w_op'*R*w_op) + con'*Q*con;

        % Compute constraints
        g1(k,j) = sum(U{j}(:,k));
        g2(k,j) = sum(C2{j}*st-C2{j}*x_op+w_glp_op{j});
    end

end

g = [reshape(g1,[N*Ns,1]);
    reshape(g2,[N*Ns,1])];

if grouping == false
     OPT_variables(1:numel(U{1})) = reshape(U{1},3*N,1);
     U_elem = numel(U{1});
     j=U_elem+1;
     for i = 2:Ns
         opt = reshape(U{i}(wells+1:end),3*N-wells,1);
         OPT_variables(j:U_elem+j-wells-1) = opt;
         j=j+U_elem-wells;

     end

else
     OPT_variables(1:numel(Ug{1})) = reshape(Ug{1},3*Ng,1);
     U_elem = numel(Ug{1});
     j=U_elem+1;
     for i = 2:Ns
         opt = reshape(Ug{i}(wells+1:end),3*Ng-wells,1);
         OPT_variables(j:U_elem+j-wells-1) = opt;
         j=j+U_elem-wells;

     end
end

Nu=length(OPT_variables)/wells;

nlp_prob = struct('f', obj, 'x', OPT_variables, 'g', g, 'p', P);

% opts = struct;
% opts.ipopt.max_iter = 10000;
% %opts.ipopt.print_level = 0; %0,3
% opts.print_time = 0; %0,1
% opts.ipopt.acceptable_tol = 1e-8;
% % optimality convergence tolerance
% opts.ipopt.acceptable_obj_change_tol = 1e-6;
% solver = nlpsol('solver', 'ipopt', nlp_prob, opts);

%
optsQP = struct;
optsQP.printLevel = 'low';
optsQP.nWSR = 20000; %Specify number of max iterations
optsQP.error_on_fail = false; %If false, the MPC continue when error

solver = qpsol('solver', 'qpoases', nlp_prob, optsQP);

args = struct;
```

Appendix V


```
% inequality constrains (state constraints)
args.lbg = zeros(2*N*Ns,1); % lower bound
args.ubg = [kron(ones(Ns,1),w_ga_max); ones(N*Ns,1)*max_glp];  % upper bound

% input constraints
args.lbx(1:3*Nu,1) = 0.5;
args.ubx(1:3*Nu,1) = 5;

% Initial states
args.p = x0;

%Initial guess
args.x0 = kron(ones(Nu,1),w_ga_op{1});

sol = solver('x0', args.x0, 'lbx', args.lbx, 'ubx', args.ubx,...
        'lbg', args.lbg, 'ubg', args.ubg, 'p',args.p);

z=full(sol.x);
w_ga_opt = z(1:3);
```

Appendix W


```
function [w_ga_opt, z] = MS_MPC_slack(A,B,C1,C2,N,x0,x_op, w_op_op, w_glp_op, w_ga_op, w_ga_max,
max_glp, grouping)

% Robust Multi-stage linear MPC with soft constraint

import casadi.*

Ns = length(A); %Number of scenarios
Ng = 3; %Number of input groups
wells = 3; % Number of wells

m_ga = SX.sym('m_ga',wells);
m_gt = SX.sym('m_gt',wells);
m_lt = SX.sym('m_lt',wells);
states = [m_ga;m_gt;m_lt]; n_states = length(states);

% A matrix that represents the states over the optimization problem.
X = SX.sym('X',n_states,(N+1),Ns);

% Input
w_ga = SX.sym('w_ga',wells);
controls = w_ga; n_controls = length(controls);

U = SX.sym('U', n_controls,N,Ns); % Decision variables (controls)
Ug = SX.sym('Ug', n_controls,Ng,Ns); % Decision variables (controls) grouped
Us = SX.sym('Us', n_controls,1); % First scenario
P = SX.sym('P',n_states); %Parameter input, used for initial states
S1 = SX.sym('S1', 1,N,Ns); %Slack varable

%Setting input u equal in first branch.
for i = 1:Ns
    Ug{i}(:,1) = Us;
end

%Building U if grouping or not
if grouping == true
    OPT_variables = SX.sym('OPT_variables',(wells*Ng)+(wells*Ng-3)*(Ns-1)+N*Ns,1);
    for j = 1:Ns
        for i = 1:N
            if i<=5
                U{j}(:,i)=Ug{j}(:,1);
            elseif i<= 0.5*N
                U{j}(:,i)=Ug{j}(:,2);
            else
                U{j}(:,i)=Ug{j}(:,3);
            end
        end
    end
else
    OPT_variables = SX.sym('OPT_variables',(wells*N)+(wells*N-3)*(Ns-1)+N*Ns,1);
    for j = 1:Ns
        U{j}(:,1)=Us;
    end
end

obj = 0; % Objective function
g1 = SX.sym('g1',N,Ns);
g2 = SX.sym('g2',N,Ns);
g3 = SX.sym('g3',N,Ns);


%Weighting matrices when N=240
Q = diag([420 425 430]); %Weighting matrix for input
R = diag([10 10 10]); % Weighting matrix for output
S = 300; % Weighting scalar for slack variable

% %Weigting matrices when N=50
% Q = diag([200 205 210]);
% R = diag([10 10 10]);
% S = 200; %80

for i=1:Ns
    X{i}(:,1) = P(1:9); % initial state
end
```

```
for j = 1:Ns
    for k = 1:N

        % Compute solution symbolically
        con = U{j}(:,k);
        st = X{j}(:,k);
        s1 = S1{j}(:,k);

        st_next = A{j}*st + B{j}*con - A{j}*x_op-B{j}*w_ga_op{j} + x_op;
        X{j}(:,k+1) = st_next;

        % Compute objective
        w_op = C1{j}*st - C1{j}*x_op+w_op_op{j};
        obj = obj + -(w_op'*R*w_op) + con'*Q*con + (s1^2)*S;

        % Compute constraints
        g1(k,j) = sum(U{j}(:,k));
        g2(k,j) = sum(C2{j}*st-C2{j}*x_op+w_glp_op{j});
        g3(k,j) = sum(C2{j}*st-C2{j}*x_op+w_glp_op{j}) - s1;

    end

end

g = [reshape(g1,[N*Ns,1]);
    reshape(g2,[N*Ns,1]);
    reshape(g3,[N*Ns,1])];

if grouping == false
     OPT_variables(1:numel(U{1})) = reshape(U{1},3*N,1);
     U_elem = numel(U{1});
     j=U_elem+1;
     for i = 2:Ns
         opt = reshape(U{i}(wells+1:end),3*N-wells,1);
         OPT_variables(j:U_elem+j-wells-1) = opt;
         j=j+U_elem-wells;

     end

     % Adding slack variables to unknown variables
     for i = 1:Ns
         OPT_variables(j:j+N-1)=S1{i};
         j=j+N;
     end

else
     OPT_variables(1:numel(Ug{1})) = reshape(Ug{1},3*Ng,1);
     U_elem = numel(Ug{1});
     j=U_elem+1;
     for i = 2:Ns
         opt = reshape(Ug{i}(wells+1:end),3*Ng-wells,1);
         OPT_variables(j:U_elem+j-wells-1) = opt;
         j=j+U_elem-wells;

     end

     % Adding slack variables to unknown variables
     for i = 1:Ns
         OPT_variables(j:j+N-1)=S1{i};
         j=j+N;
     end
end

Nu = (length(OPT_variables) - N*Ns)/wells;

nlp_prob = struct('f', obj, 'x', OPT_variables, 'g', g, 'p', P);

opts = struct;
opts.ipopt.max_iter = 10000;
opts.ipopt.print_level = 0; %0,3
opts.print_time = 0; %0,1
opts.ipopt.acceptable_tol = 1e-8;
% optimality convergence tolerance
```

```
opts.ipopt.acceptable_obj_change_tol = 1e-6;
solver = nlpsol('solver', 'ipopt', nlp_prob, opts);

%
% optsQP = struct;
% optsQP.printLevel = 'low';
% optsQP.nWSR = 20000; %Specify number of max iterations
% optsQP.error_on_fail = false; %If false, the MPC continue when error
%
% solver = qpsol('solver', 'qpoases', nlp_prob, optsQP);

args = struct;

% inequality constrains (state constraints)
args.lbg = zeros(3*N*Ns,1); % lower bound
args.ubg = [kron(ones(Ns,1),w_ga_max); % Upper bound w_gc
    ones(N*Ns,1)*max_glp; % upper bound max_glp
    ones(N*Ns,1)*max_glp - 10];  % upper bound soft constraint

% input constraints
args.lbx = [ones(3*Nu,1)*0.5;zeros(N*Ns,1)];
args.ubx = [ones(3*Nu,1)*5;ones(N*Ns,1)*inf];

% Initial states
args.p = x0;

%Initial guess
args.x0 = [kron(ones(Nu,1),w_ga_op{1});zeros(N*Ns,1)];

sol = solver('x0', args.x0, 'lbx', args.lbx, 'ubx', args.ubx,...
        'lbg', args.lbg, 'ubg', args.ubg, 'p',args.p);

z=full(sol.x);
w_ga_opt = z(1:3);
```

Appendix X


```
function [Ac, Bc, Cc, w_op_op, w_glp_op, w_ga_op] = MS_Scenarios(Ns)

% Building system matrices and operating point vectors for each scenario used in the multistage MPC

Ac = {Ns};
Bc = {Ns};
Cc = {Ns};
w_op_op = {Ns};
w_glp_op = {Ns};
w_ga_op = {Ns};

% 10 % deviation from parameters
% Well        = [Well1  Well2   Well3]
PI_dev        = [0    0     0;
                -0.1 -0.1  -0.1;
                 0.1  0.1   0.1;
                 0.1  0.1   0.1;
                -0.1 -0.1  -0.1;
                -0.1 -0.1  -0.1;
                 0.1  0.1   0.1;
                 0.1  0.1   0.1;
                -0.1 -0.1  -0.1;
                -0.1  0.1   0.1] +1;


% 20 % deviation from parameters
% Well        = [Well1  Well2   Well3]
WC_dev        = [0    0     0;
                 0.2   0.2   0.2;
                -0.2  -0.2  -0.2;
                 0.2   0.2   0.2;
                 0.2   0.2   0.2;
                -0.2  -0.2  -0.2;
                 0.2   0.2   0.2;
                -0.2  -0.2  -0.2;
                -0.2  -0.2  -0.2;
                 0.2  -0.2  -0.2] +1;


GOR_dev       = [0    0     0;
                -0.2  -0.2  -0.2;
                 0.2   0.2   0.2;
                 0.2   0.2   0.2;
                 0.2   0.2   0.2;
                 0.2   0.2   0.2;
                -0.2  -0.2  -0.2;
                -0.2  -0.2  -0.2;
                -0.2  -0.2  -0.2;
                -0.2   0.2   0.2] +1;



for i=1:Ns
    [Ac{i}, Bc{i}, Cc{i}, w_op_op{i}, w_glp_op{i}, w_ga_op{i}] =
Jac_func_op(PI_dev(i,:),WC_dev(i,:),GOR_dev(i,:));

    prosent = i*100/(Ns);
    text = ['Completed building scenarios ', num2str(prosent), ' %'];
    disp(text)

end
```

Appendix Y


```
clear
close all

% Closed loop simulation using robust MPC applied to uncertain models

%Simulation parameters
N = 25;                %Prediction horizon
dt = 15; %[s]           15s is appropriate at "normal" condition
hour = 4; %[h]         hours of simulationtime
Np = 3600*hour/dt;      %Number of datapoints
t=linspace(0,hour,Np); %Used for plotting
grouping = false;

max_glp = 310;

% up to 10 % deviation from parameters
% Well         = [Well1  Well2   Well3]
PI_dev         = [0    0     0;
                  0.1   0.1   0.1;
                 -0.1   0.1   0.1;
                  0.1  -0.1   0.1;
                 -0.1  -0.1   0.1;
                  0.1   0.1  -0.1;
                 -0.1   0.1  -0.1;
                  0.1  -0.1  -0.1;
                 -0.1  -0.1  -0.1;
                 randi([-100 100],6,3)/1000] +1;


% up to 20 % deviation, only WC and GOR

WC_dev         = [0    0     0;
                  0.2   0.2  -0.2;
                 -0.2   0.2  -0.2;
                  0.2  -0.2  -0.2;
                 -0.2  -0.2  -0.2;
                  0.2   0.2   0.2;
                 -0.2   0.2   0.2;
                  0.2  -0.2   0.2;
                 -0.2  -0.2   0.2;
                 randi([-200 200],6,3)/1000] +1;

GOR_dev        = [0    0     0;
                  0.2  -0.2  -0.2;
                 -0.2  -0.2  -0.2;
                  0.2   0.2   0.2;
                 -0.2   0.2   0.2;
                  0.2  -0.2   0.2;
                 -0.2  -0.2   0.2;
                  0.2   0.2  -0.2;
                 -0.2   0.2  -0.2;
                 randi([-200 200],6,3)/1000] +1;

PI_len = length(PI_dev);

P_out_all = [];
P_in_all = [];
T_all = [];

%% -------------------------------------------------------
tStart = tic;

for i=1:PI_len

[P_out, P_in, T] = MS_simulation_unc(N, Np, dt, max_glp, grouping, PI_dev(i,:),
WC_dev(i,:),GOR_dev(i,:));

P_out_all = [P_out_all;P_out];
P_in_all = [P_in_all;P_in];
T_all = [T_all; T];

end
%-------------------------------------------------------
tEnd = toc(tStart)
```

```
%% Plotting

color1 = [0, 0.4470, 0.7410];
color2 = [0.8500, 0.3250, 0.0980];
color3 = [0.9290, 0.6940, 0.1250];
color4 = [0.4940, 0.1840, 0.5560];
color5 = [0.4660, 0.6740, 0.1880];
color6 = [0.2, 0.2, 0.2];
width = 1;
k=1+Np;

f1 = figure;
max_glp_=linspace(max_glp,max_glp,Np);

for i=2:PI_len
    plot(t,P_out_all(k:k+Np-1,10),'color',color6,'LineWidth',width,'HandleVisibility','off');
    hold on
    plot(t,P_out_all(k:k+Np-1,11),'color',color6,'LineWidth',width,'HandleVisibility','off');
    plot(t,P_out_all(k:k+Np-1,12),'color',color6,'LineWidth',width,'HandleVisibility','off');
    plot(t,P_out_all(k:k+Np-1,13),'color',color6,'LineWidth',width,'HandleVisibility','off');
    plot(t,P_out_all(k:k+Np-1,14),'color',color6,'LineWidth',width,'HandleVisibility','off');

    k=k+Np;
end

width = 3;
k=1;
plot(t,P_out_all(k:k+Np-1,10),'color',color1,'LineWidth',width,'DisplayName','Well 1 w_o_p');
plot(t,P_out_all(k:k+Np-1,11),'color',color2,'LineWidth',width,'DisplayName','Well 2 w_o_p');
plot(t,P_out_all(k:k+Np-1,12),'color',color3,'LineWidth',width,'DisplayName','Well 3 w_o_p');
plot(t,P_out_all(k:k+Np-1,13),'color',color4,'LineWidth',width,'DisplayName','Total w_o_p');
plot(t,P_out_all(k:k+Np-1,14),'color',color5,'LineWidth',width,'DisplayName','Total w_g_l_p');
plot(t,max_glp_,'--r','LineWidth',2, 'DisplayName','w_s^m^a^x');
% plot(t,max_glp_-10,'--k','LineWidth',2, 'DisplayName','w_s^m^a^x - soft'); %Activate on MS_MPC_slack

title('w_o_p and w_g_l_p - Produced')
ylabel('kg/s');
xlabel('Hour');

legend('FontSize',12);
grid('on');
hold off


f2=figure;

width = 1;
k=1+Np;

for i=2:PI_len
    plot(t,P_in_all(k:k+Np-1,1),'color',color6,'LineWidth',width,'HandleVisibility','off');
    hold on
    plot(t,P_in_all(k:k+Np-1,2),'color',color6,'LineWidth',width,'HandleVisibility','off');
    plot(t,P_in_all(k:k+Np-1,3),'color',color6,'LineWidth',width,'HandleVisibility','off');

    k=k+Np;

end

width = 3;
k=1;
plot(t,P_in_all(k:k+Np-1,1),'color',color1,'LineWidth',width,'DisplayName','Well 1');
plot(t,P_in_all(k:k+Np-1,2),'color',color2,'LineWidth',width,'DisplayName','Well 2');
plot(t,P_in_all(k:k+Np-1,3),'color',color3,'LineWidth',width,'DisplayName','Well 3');

title('w_g_a - Injected lift gas')
ylabel('kg/s');
xlabel('Hour');
legend('FontSize',12);
grid('on');
hold off
```

```
% Plotting w_ga
%----1
f3 = figure;
tiledlayout(4,1);

ax1 = nexttile;
width = 1;
k=1+Np;

for i=2:PI_len
    plot(ax1,t,P_in_all(k:k+Np-1,1),'color',color6,'LineWidth',width,'HandleVisibility','off');
    hold on

    k=k+Np;

end

width = 3;
k=1;
plot(ax1, t,P_in_all(k:k+Np-1,1),'color',color1,'LineWidth',width,'DisplayName','Well 1');
title(ax1,'w_g_a - Injected lift gas')
ylabel(ax1, 'kg/s');
ylim([0 6]);
legend('FontSize',12);
grid('on');
hold off

%-----2
ax2 = nexttile;
width = 1;
k=1+Np;

for i=2:PI_len
    plot(ax2,t,P_in_all(k:k+Np-1,2),'color',color6,'LineWidth',width,'HandleVisibility','off');
    hold on

    k=k+Np;

end

width = 3;
k=1;
plot(ax2, t,P_in_all(k:k+Np-1,2),'color',color2,'LineWidth',width,'DisplayName','Well 2');
ylabel(ax2, 'kg/s');
ylim([0 6]);
legend('FontSize',12);
grid('on');
hold off

%-----3
ax3 = nexttile;
width = 1;
k=1+Np;

for i=2:PI_len
    plot(ax3,t,P_in_all(k:k+Np-1,3),'color',color6,'LineWidth',width,'HandleVisibility','off');
    hold on

    k=k+Np;

end

width = 3;
k=1;
plot(ax3, t,P_in_all(k:k+Np-1,3),'color',color3,'LineWidth',width,'DisplayName','Well 3');
ylabel(ax3, 'kg/s');
ylim([0 6]);
legend('FontSize',12);
grid('on');
hold off

%---4
w_gc_max = injGas(Np,N);
```

```
ax4 = nexttile;
width = 1;
k=1+Np;

for i=2:PI_len
    plot(ax4,t,sum(P_in_all(k:k+Np-1,:),2),'color',color6,'LineWidth',width,'HandleVisibility','off');
    hold on

    k=k+Np;

end

width = 3;
k=1;
plot(ax4, t, sum(P_in_all(k:k+Np-1,:),2),'color',color4,'LineWidth',width,'DisplayName','Total w_g_a');
plot(ax4, t, w_gc_max(1:end-N),'--r','LineWidth',2 , 'DisplayName','w_g_c^m^a^x');
ylabel(ax4, 'kg/s');
xlabel(ax4, 'Hour');
ylim([0 12]);
legend('FontSize',12);
grid('on');
hold off

% Plotting iteration time
t2 = linspace(0,PI_len,Np*PI_len);

f4=figure;

plot(t2,T_all)
title('Optimizing time')
ylabel('Optimizing time [s]');
xlabel('Simulation number');
grid('on');

% Plotting only w_glp and w_op

width = 1;
k=1+Np;

f5 = figure;
max_glp_=linspace(max_glp,max_glp,Np);

for i=2:PI_len

    plot(t,P_out_all(k:k+Np-1,13),'color',color6,'LineWidth',width,'HandleVisibility','off');
    hold on
    plot(t,P_out_all(k:k+Np-1,14),'color',color6,'LineWidth',width,'HandleVisibility','off');

    k=k+Np;
end

width = 3;
k=1;
plot(t,P_out_all(k:k+Np-1,13),'color',color4,'LineWidth',width,'DisplayName','Total w_o_p');
plot(t,P_out_all(k:k+Np-1,14),'color',color5,'LineWidth',width,'DisplayName','Total w_g_l_p');
plot(t,max_glp_,'--r','LineWidth',2, 'DisplayName','w_s^m^a^x');
% plot(t,max_glp_-10,'--k','LineWidth',2, 'DisplayName','w_s^m^a^x - soft'); %Activate on MS_MPC_slack

title('w_o_p and w_g_l_p - Produced')
ylabel('kg/s');
xlabel('Hour');

legend('FontSize',12);
grid('on');
hold off
```

Appendix Z


```
function [P_out, P_in, T2] = MS_simulation_unc(N, Np, dt, max_glp, grouping, PI_dev, WC_dev, GOR_dev)

% Simulation interface multi-stage MPC and uncertain model

% Initial states
m_ga = [8.1180e3    7.6192e3    8.0600e3];
m_gt = [1.4714e3    1.4879e3    1.1618e3];
m_lt = [2.2042e4    1.7740e4    2.5451e4];

x0 = [m_ga m_gt m_lt]';
x_op = x0;

w_gc_max = injGas(Np,N);

% Storing
P_out = zeros(Np,14);%Matrix for storing outputs from nonlinear model
P_in = zeros(Np,3);  %Matrix for storing inputs given by MPC
T = zeros(Np,2); % Storing iteration time

% Building scenarios
[Ac, Bc, Cc, w_op_op, w_glp_op, w_ga_op] = MS_Scenarios(10);

for i = 1:Np
    tStart1 = tic;

    w_GC_max = w_gc_max(i:i+N-1);
    %Updating model matrices and operating points
    [A,B,C1,C2,y1_op,y2_op, u_op] = MS_UpdateMatrices_op(Ac, Bc, Cc , x0, w_op_op, w_glp_op, w_ga_op,
dt);

    tStart2 = tic;
    %MPC
    [w_ga, z] = MS_MPC(A,B,C1,C2,N,x0, x_op, y1_op, y2_op, u_op, w_GC_max, max_glp, grouping);
%     [w_ga, z] = MS_MPC_slack(A,B,C1,C2,N,x0, x_op, y1_op, y2_op, u_op, w_GC_max, max_glp, grouping);
    T(i,2) = toc(tStart2);


    %Running simulator with RK method
    [x_next_non, out_non] = updateState(x0',dt,w_ga', PI_dev, WC_dev, GOR_dev);
    x0 = x_next_non';

    P_out(i,:) = [out_non(1,1:12),sum(out_non(1,10:12)), sum(out_non(1,16:18))]; % summing all w_op and
w_glp


    x_op=x0;


    P_in(i,:) = w_ga';

    prosent = i*100/(Np);
    text = ['Completed ', num2str(prosent), ' %'];
    disp(text)

    T(i,1) = toc(tStart1);
end

T2=T(:,2);
```

Appendix AA


```
function [A,B,C1,C2,y1_op,y2_op, u_op] = MS_UpdateMatrices_op(Ac, Bc, Cc , x, y1, y2, u, dt)

% Substitute states into system matrices and operating point vectors
% in multi-stage closed loop simulation


l = length(Ac);
A = {l};
B = {l};
C1 = {l};
C2 = {l};
y1_op = {l};
y2_op = {l};
u_op = {l};

for i = 1:l
    Ac_ = Ac{i}(x(1), x(2), x(3), x(4), x(5), x(6), x(7), x(8), x(9));
    Cc_ = Cc{i}(x(4), x(5), x(6), x(7), x(8), x(9));


    sys = ss(Ac_,Bc{i} ,Cc_,0);
    d_sys = c2d(sys,dt);

    A{i}=d_sys.a;
    B{i}=d_sys.b;
    C=d_sys.c;

    C1{i} = C(1:3,:);
    C2{i} = C(4:6,:);


    %Outout operation points
    y1_op_ = y1{i}(x(4),x(5),x(6),x(7),x(8),x(9));
    y2_op_ = y2{i}(x(4),x(5),x(6),x(7),x(8),x(9));

    y1_op{i} = y1_op_';
    y2_op{i} = y2_op_';

    %Input operation points
    u_op_ = u{i}(x(1), x(2), x(3), x(4), x(5), x(6), x(7), x(8), x(9));
    u_op{i} = u_op_';

end
```