NTNU – Trondheim
Norwegian University of
Science and Technology

Non-linear model predictive control for an oil production network based on gas-lift

Thomas Lund

Master of Science in Cybernetics and Robotics
Submission date: June 2014
Supervisor: Bjarne Anton Foss, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU
Norwegian University of Science and Technology

Faculty of Informaton Technology, Mathematics and Electrical Engineerig
Department of Engineering Cybernetics

Master project

Name of candidate:	Thomas Lund
Subject:	Engineering Cybernetics
Title:	Non-linear model predictive control for an oil production network based on gas-lift
Title (in Norwegian):	Regulering av oljeproduksjon ved bruk av ulineær MPC

Control and optimization of oil production based on gas-lift is a difficult task due to well fluid flow oscillations, interaction between wells and process uncertainty. Model based control and optimization requires a predictive model capable to describe main process characteristics such as frequency of oscillations, process equilibria and control input-output dependencies (predictions). This model should be as simple as possible to enable fast computation of control laws. In addition, the model is adjusted during the control process to conform with measurements and thereby reduce the prediction error.

This project shall evaluate non-linear model predictive control (NMPC) techniques applied to a gas-lifted oil production network. To this end, an oil production network represented in a detailed multiphase dynamic fluid flow simulator (OLGA) will play the role of the process being controlled. For control purposes, a simple dynamic model inspired by [1][2][3] must be fit to the detailed model. Further based on the experience acquired in [4], a state estimator must be proposed to enable control feedback.

Task description:

1. Perform a literature review on methods for NMPC. This is an extensive area so the review should be limited to methods of particular interest to the current study.
2. Propose a simple production network consisting of wells and a single manifold-riser and instantiate it in OLGA.
3. Propose a simple model for control purposes based on [1][2][3], and fit it to the detailed model components.
4. Develop, implement and test methods for NMPC (using simulation).

5. Compare static and dynamic optimization solutions on a set of operating sceanrios, either for the complete system or parts thereof.

6. Assess the benefits and drawbacks of applying NMPC. Further, compare NMPC with alternative optimization-based strategies, in particular steady state optimization.

[1] K. Nalum. Modeling and Dynamic Optimization in Oil Production. Master Thesis. Norwegian University of Science and Technology. 2013.

[2] M.A. Aguiar. Optimal oil production network control using Modelica. Engineering Thesis. Federal University of Santa Catarina. 2013.

[3] E. Jahanshahi. Control Solutions for Multiphase Flow Linear: Linear and nonlinear approaches to anti-slug control. Ph.D Thesis. Norwegian University of Science and Technology, 2013.

[4] T. Lund. State estimation for an oil production network based on gas-lift . Project report 2013.

Start date: 10.01.2014

End date: 17.06.2014

Co-supervisor: Andres Codas, NTNU and Esmaeil Jahashahi, NTNU

Trondheim, 10.01.2014

Bjarne Foss
Professor/supervisor

Abstract

Control and optimization of an oil production network based on gas-lift is a difficult task with challenges including well fluid flow oscillations and pipeline slugging which can lead to instabilities in the system. A possible control solution to this is Model predictive control, which requires a model capable of capturing the main characteristics of the controlled process, such as oscillations and input dependencies.

In this thesis models for wells and flowlines are instantiated in the detailed multiphase dynamic fluid flow simulator OLGA. Further, simpler dynamic models are attempted fitted to these models, and the comparison between the resulting models is discussed. In addition state estimation using an Extended Kalman Filter is performed on the flowline model, based on measurements proposed in [1].

Based on the fitting of models and implementation of state estimation for the flowline system, methods for NMPC are attempted implemented using the JModelica.org environment, and limitations and advantages associated with this environment are assessed.

Sammendrag

Kontroll og optimalisering av et olje-produksjonsnettverk basert på gass-løft er en vanskelig oppgave med utfordinger som osillerende strømninger i brønner og rørledninger. En aktuell kontrolløsning i slike produksjonsnettverk er Model Prediktiv Kontroll. Denne kontrolløsningen er avhengig av en god modell av systemet som skal kontrolleres, da den må være i stand til å fange opp de viktigste egenskapene til produksjonsnettverket, som osilleringer og pådragsavhengigheter.

I denne oppgaven blir modeller for brønner og rørledninger instansiert i den detaljerte multifasestrømningssimulatoren OLGA. Deretter blir enklere dynamiske modeller forsøkt tilpasset til disse komplekse modellene, og forskjellen mellom de resulterende modellene blir diskutert. Videre blir det utført tilstandsestimering med et utvidet Kalmanfilter i rørledningsmodellen basert på foreslåtte målinger fra [1].

Med utgangspunkt i den utførte modelltilpasningen og tilstandsestimeringen, blir det forsøkt implementert en ulineær modelprediktiv kontroller for rørledningsmodellen i optimalisering- og simuleringsverktøyet JModelica.org. Deretter blir fordeler og begrensninger ved dette verktøyet vurdert.

Preface

This master thesis was written during the spring semester of 2014 and is the result of the subject Engineering Cybernetics Master Thesis *TTK4900*, which is obligatory in the tenth semester of the M.Sc. degree in Engineering Cybernetics at the Norwegian University of Scinece and Technology (NTNU).

I would like to thank my supervisor Prof. Bjarne Foss and co-supervisor Andres Codas for valuable advice and guidance in the work with this thesis.

Trondheim
June 17, 2014.

Thomas Lund

Table of Contents

List of Tables

List of Figures

Acronyms

AD	=	Automatic Differentiation
CasADi	=	Computer Algebra System for Automatic Differentiation
DAE	=	Differential Algebraic Equation
DC	=	Direct Collocation
DOP	=	Dynamic Optimization Problem
FMI	=	Functional Mock-up Interface
FMU	=	Functional Mock-up Unit
IP	=	Interior Point
IPOPT	=	Interior Point Optimizer
IVP	=	Initial Value Problem
JMI	=	JModelica Model Interface
MPC	=	Model Predictive Control
NMPC	=	Non Linear Model Predictive Control
MS	=	Multiple Shooting
NLP	=	Non Linear Problem
OC	=	Optimal Control
OCP	=	Optimal Control Problem
ODE	=	Ordinary Differential Equation
RTPO	=	Real Time Production Optimization
SQP	=	Sequential Quadratic Programming
SS	=	Single Shooting

Chapter 1

Introduction

In this chapter the background for the work described in this master thesis will be presented. This is followed by the scope for the thesis, and an outline of the report.

1.1 Production optimization in oil gathering systems

Most oil reservoirs consist of relatively thin slabs of porous rock buried at depths of hundreds to thousands of meters. In the early stages of oil production from a reservoir, it is important to plan and drill wells in order to reach the predefined plateau production stage as fast as possible. In this stage, the oil usually flows to the surface naturally, and no further methods for enhanced oil recovery are needed. However, after some years, when the primary recovery phase ends, it is necessary to apply additional production components in order to maintain a satisfactory oil recovery.

The system has then entered the secondary recovery phase, where water or gas is injected into the reservoir in order to maintain the pressure and to displace the oil from the injection wells towards the production wells. Even with the use of injection wells, most of the oil will remain trapped in the pores of the rock, and the recovery factor often stays somewhere between 10% and 50% [5].

A possible tertiary recovery phase includes the use of enhanced oil recovery techniques such as injection of surfactants, polymers, or steam. This is done to change the properties in the reservoirs in a favourable way. This can for example be to reduce the surface tension or to increase the permeability of the oil. However, the feasibility of these techniques is strongly dependent of a high oil price.

Another emerging method for increasing the oil recovery factor from a reservoir is to use model-based control concepts combined with sensors and remotely controllable valves in wells and at the surface. These methods often rely on large-scale subsurface flow models. To effectively work with the mentioned models, there will be a need for strong and effective computers and computer algorithms.

Decisions regarding when to enter a new recovery phase, and which strategies to use are parts of the many important decisions that has to be made before, and under production

of an oil reservoir. These decisions require planning on several time horizons, which is illustrated in Figure 1.1 that is taken from [6].

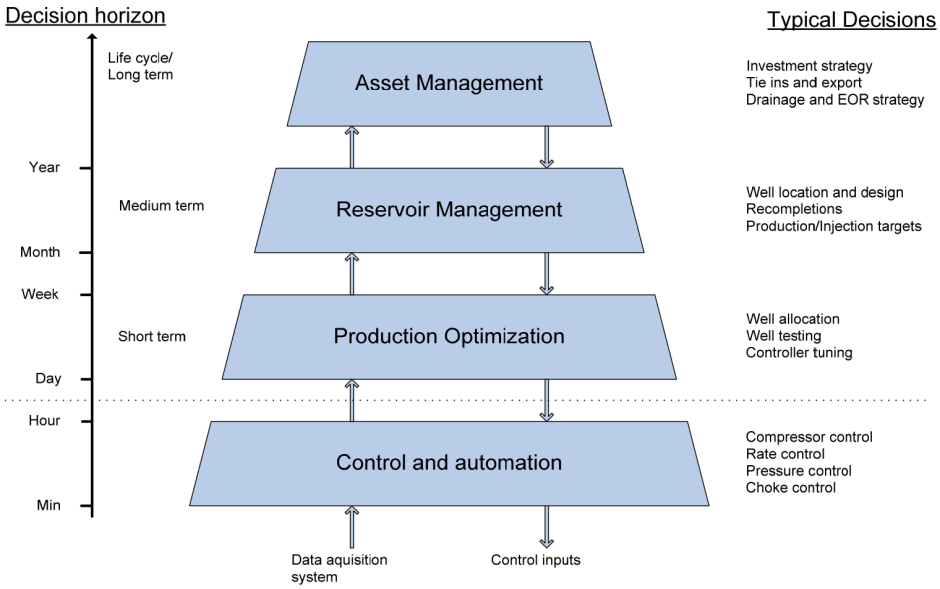

Figure 1.1: Multi-level control hierarchy.

As shown in Figure 1.1, the two upper boxes represent long term decisions, as mentioned above, with a decision horizon that can span from months to several years, while the two lower boxes represents decisions with decision horizons spanning from weeks to only a minute. It is in these boxes one can find real-time production optimization (RTPO), which can be utilised to optimize the production with respect to the constraints in the system. Such optimization may require models of both the sub-surface components, such as the reservoir and the wells, and also the surface components such as pipelines and other process facilities.

In the bottom of the hierarchy, closed-loop controllers are widely used to control flow rates, pressures and other controllable aspects relevant to production. The performance of these closed-loop controllers is limited however, and there is a growing interest in applying advanced process control concepts to these controllers. In particular Model Predictive Control (MPC) is considered, as an MPC can give better coordination to the production, and help avoiding instabilities such as slugging.

1.2 The principle of Model Predictive Control

In [4] it is given that Model predictive control (MPC), also referred to as moving horizon control or receding horizon control, is the only advanced control method widely used in the industry. Today, it is mainly applied in petrochemical industries, but its use in other industries is growing. The reason for such popularity is the ability of MPC designs to

yield high performance control systems capable of operating without expert intervention for long periods of time [7].

The MPC consists of three main parts; the cost function, the constraints, and the prediction model. The cost function is a scalar criterion measuring the performance of the system, for example with regard to offset from a defined set point, or with regard to economical performance, as for example produced oil in a oil-gas network. The cost function is measured over a predefined prediction horizon, and minimized with respect to future control inputs. When the the optimal solution to the cost function is found, the first control input is applied to the real system. This procedure is repeated at the next time step $t' = t' + 1$.

The constraints are one of the main motivations behind the MPC, as they enable precise specification of the operational area of the process variables. This can be very valuable, as it can allow production systems to operate close to their system constraints, with low risk of violating them.

One of the main challenges when developing an MPC is the prediction model. The performance of the MPC is dependant of a prediction model which captures the main behaviour of the system, in order to find a good control solution and ensure preservation of the constraints. When supplying the prediction model to the MPC, one has to consider the trade off between a complex model, which gives a very accurate description of the system but requires much computation power and time, or a simpler model which may only be valid for a smaller operation area, but which can be solved faster.

The iterative steps of the MPC is illustrated in Figure1.3, where one can see how the first input of the whole control solution is applied at each step.

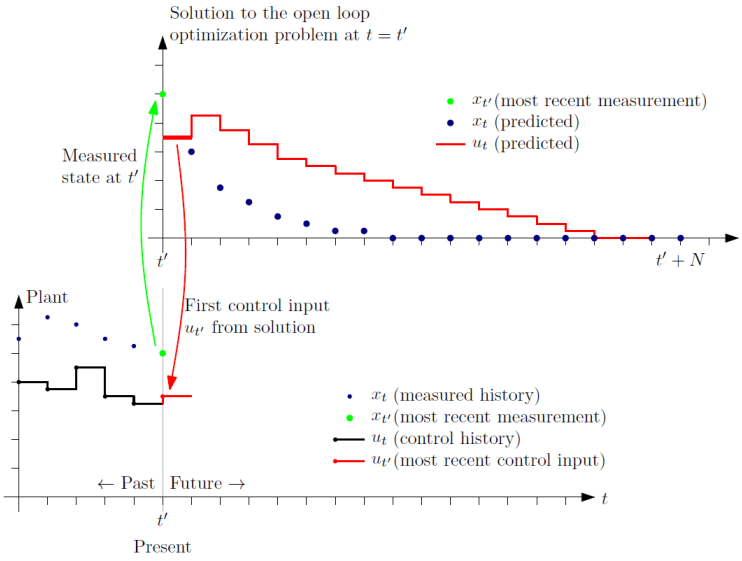

Figure 1.2: Principle of model predictive control.

As mentioned, the figure above showns how the MPC uses a measured state of the system at the current time t', to solve a finite horizon open-loop optimal control problem for the time horizon from t' to $t' + N$. One can then see how the first input of the finite horizon solution is applied to the real system, before another finite horizon open-loop optimal control problem is solved to find the next input.

1.3 Modeling and optimization in Optimica and JModelica.org

In order to fully utilize the potential of the MPC, the need for an efficient and structured optimization platform is evident. The modeling language Modelica is targeted at modeling of complex physical systems, and is about to establish itself as one of the main tools in this area [8]. Optimica is an extension to Modelica, enabling optimization formulations based on developed models. Modelica does not support optimization in it self however, but a framework incorporating this feature is JModelica.org, which is an open source platform that integrates state of the art algorithms for simulation, optimization and automatic differentiation. Further, it provides an interface for dynamic optimization of Modelica models through the Optimica extension. Some of the main components included in JModelica.org is shown in the figure below.

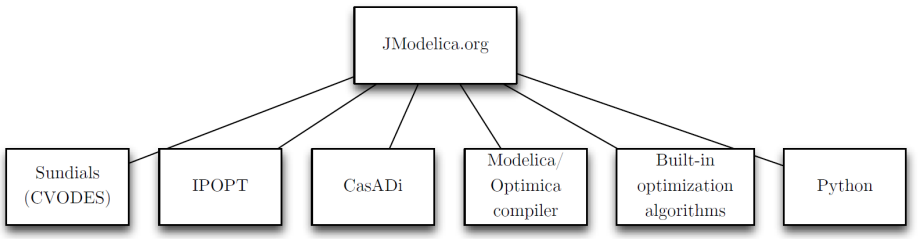

Figure 1.3: JModelica.org components.

The linking of these sophisticated packages for simulation and automatic differentiation forms a promising platform for optimization and control for dynamic systems.

1.4 Scope and emphasis

The work in this master project will investigate how the previous work [3], [9] and [1] can be utilised when developing a nonlinear model predictive controller in the JModelica.org platform. In [3], dynamic optimization were applied to an oil gathering system consisting of a well-flowline network where the wells and flowlines were described by low order dynamic models. In [9], an alternative and more complex low order model were proposed for the flowlines.

In this thesis, the validity of the mentioned low-order models will be assesed through comparison with more complex and realistic models developed in the commercial mul-

tiphase simulator OLGA, which is the industry standard tool for transient simulation of multiphase petroleum production. Further, the application of the low-order models will be tested through the implementation of a nonlinear model predictive controller, using the low-order models to control the OLGA models.

1.5 Report outline

In order to provide the necessary theoretical background for working with nonlinear model predictive control, a literature study of methods for NMPC in particular interest to this thesis is presented the next chapter. Thereafter, in Chapter 3, the various implementation tools used in the implementation are described, before the models and optimal control problem formulations used in this thesis are presented in Chapter 4.

As the basis for the theoretical and practical tools now is described, Chapter 5 presents how this is utilized in implementation and the results of a series of simulations describing the resulting systems are shown. Thereafter, in Chapter 6, the results are discussed and proposals for improvements and further work are given, before a conclusion is made in Chapter 7.

Chapter 2

Literature study

As the goal of this project is to control a production system containing gas-lift wells and a flowline using non-linear model predictive control (NMPC), there will in this chapter be given a review on previously published work and studies in this area.

When studying model predictive control, it is natural to start with the linear MPC, since it is far less complex than the non linear MPC. Therefore the basis of the method will first be presented, before methods for non linear MPC are more thoroughly described.

2.1 Linear MPC

Theory revolving linear MPC has previously been studied extensively in various literature. In this section, the work of Foss and Heirung[2], and Garcia, Prett and Morari [7] is considered in particular.

In the mentioned work, it is described how MPC controllers are divided into two main classes, linear and non-linear MPC. They are separated by the classification of the optimization problem solved within the MPC. In linear MPC, the optimization problem is a convex QP problem, which means that a global solution always can be found within a finite time frame. From [2] it is given that a linear MPC problem can be formulated as follows.

$$\min_{z\in\mathbb{R}^n} f(z) = \sum_{t=0}^{N-1} \frac{1}{2} x_{t+1}^T Q_{t+1} x_{t+1} + d_{xt+1} x_{t+1} + \frac{1}{2} u_t^T R_t u_t + d_{ut} u_t + \frac{1}{2} \Delta u_t^T R_{\Delta t} u_t \tag{2.1a}$$

subject to

$$x_{t+1} = A_t x_t + B_t u_t \tag{2.1b}$$
$$x_0 = \text{given} \tag{2.1c}$$
$$x^{low} \leq x_t \leq x^{high} \tag{2.1d}$$
$$u^{low} \leq u_t \leq u^{high} \tag{2.1e}$$
$$-\Delta u^{high} \leq \Delta u_t \leq \Delta_u^{high} \tag{2.1f}$$
$$Q_t \succeq 0 \tag{2.1g}$$
$$R_t \succeq 0 \tag{2.1h}$$
$$R_{\Delta t} \succeq 0, \tag{2.1i}$$

where

$$\Delta u_t = u_t - u_{t-1} \tag{2.1j}$$
$$z^T = (x_1^T, ..., x_N^T, u_0^T, ..., u_{N-1}^T). \tag{2.1k}$$

In the equation set above, x is defined as the system state vector, and u as the system input. It should be noted that the objective function given in equation (2.1a) is quadratic and positive semi-definite, and that the constraints (2.1b) - (2.1i) are linear. This is an absolute requirement for a linear MPC. When state feedback is added, the algorithm given in table 2.1 describes the steps of the controller.

Table 2.1: The table contains an algorithm describing the steps of a linear MPC with state feedback, as given in [2].

Algorithm	Linear MPC with state feedback

	for $t = 0, 1, 2, ...$ **do**
	Get the current state x_t
	Solve the optimization problem (2.1) on the prediction horizon from t to $t + N$ with x_t as the initial condition.
	Apply for the first control move u_t from the solution above.
	end for

Linear MPC is successfully used in many applications, especially in the process industry and this is despite the fact that the dynamics of the closed loop systems it controls are non-linear [4]. However, there are many systems which are inherently non-linear, and for these systems linear models are often inadequate. This motivates the use of non-linear model predictive control (NMPC).

2.2 Non-linear MPC

As mentioned in the previous section, some systems are too non-linear to be adequately controlled by a linear MPC and a non-linear approach is therefore needed. The difference between linear MPC, as described in the previous section, and the non-linear MPC, lies in the classification of the dynamic optimization problem. While the linear MPC contains a linear, convex, and relatively simple dynamic optimization problem, the NMPC needs to solve a considerably much more complex problem with non-linear constraints.

2.2.1 Optimal control problem formulation

The systems studied in this thesis are initially described by Differential Algebraic Equations (DAE) and the equations describing the system can therefore be formulated in the following way

$$F(x(t), \dot{x}(t), u(t), t) = 0 \tag{2.2a}$$
$$h(x(t_0)) = 0, \tag{2.2b}$$

where $x : [t_0, t_f] \to \mathbb{R}^{n_x}$ are states, $u : [t_0, t_f] \to \mathbb{R}^{n_u}$ are control inputs, $\mathbf{F} : \mathbb{R}^{n_x \times n_x \times n_u \times 1} \to \mathbb{R}^{n_x}$ are system functions and $x(t_0)$ are initial conditions. Start and end time are given by t_0 and t_f respectively [10]. An alternative representation of this is given by the semi-explicit form

$$\dot{z}(t) = f(z(t), y(t), u(t)) \tag{2.3a}$$
$$z(t_0) = z_0 \tag{2.3b}$$
$$0 = g(z(t), y(t), u(t)), \tag{2.3c}$$
$$\tag{2.3d}$$

where $z : [t_0, t_f] \to \mathbb{R}^{n_z}$ and $y : [t_0, t_f] \to \mathbb{R}^{n_y}$. The differential and algebraic equations are given by $f : \mathbb{R}^{n_z \times n_y \times n_u} \to \mathbb{R}^{n_z}$ and $g : \mathbb{R}^{n_z \times n_y \times n_u} \to \mathbb{R}^{n_y}$, while $z(t_0)$ are the initial conditions. It should be noted that in this system formulation, the system is time-invariant as the time variable t does not appear explicitly.

This is further reformulated to

$$\dot{z}(t) = f(z(t), y[z(t), u(t)], u(t)) = \bar{f}(z(t), u(t)) \tag{2.4a}$$
$$z(t_0) = z_0 \tag{2.4b}$$

by using theorem 8.1:

For $u(t)$ and p specified, let $\bar{f}(z(t), u(t), p)$ be Lipschitz continuous for all $z(t)$ in a bounded region with $t \in [0, t_f]$. Then the solution of the initial value problem (2.4) *exists and is unique, $z(t)$ for $t \in [0, t_f]$,*

as given in [10]. This is the system formulation that will be used when applying state estimation to the systems in this thesis. Based on (4.14), an optimal control formulation (OCP) for an nonlinear MPC can be given as

$$\min_{x,y,u} \Phi(z,y,u) = \int_{t_0}^{t_f} L(t,z,y,u)dt + E(t_f, z(t_f), y(t_f), u(t_f)) \tag{2.5a}$$

subject to

$$\dot{z} = f(x,y,u), \tag{2.5b}$$
$$z(t_0) = z_0, \tag{2.5c}$$
$$g(z,y,u) = 0, \tag{2.5d}$$
$$g_I(z,y,u) \leq 0, \tag{2.5e}$$
$$z_L \leq z \leq z_U, \tag{2.5f}$$
$$y_L \leq y \leq y_U, \tag{2.5g}$$
$$u_L \leq u \leq u_U, \tag{2.5h}$$
$$t \in [t_0, t_f]. \tag{2.5i}$$

In this OCP formulation, the cost function Φ is given by the sum of the integral over L, which accounts for the cost over the time horizon $[t_0, t_f]$, and the terminal cost denoted by E. The constraints (4.16c)-(4.16d) are the system equation constraints, (4.16e)-(4.16f) are the path constraints, while (4.16g)-(4.16i) define the upper and lower bounds for z, y and u. When state feedback is added, the algorithm given in table 2.2 describes the steps of the controller.

Table 2.2: The table contains an algorithm describing the steps of a non-linear MPC with state feedback, as given in [2].

Algorithm	Nonlinear MPC with state feedback
	for $t = 0, 1, 2, ...$ **do**
	Get the current state z_t
	Solve the optimization problem (4.16) on the prediction horizon from t to $t + N$ with z_t as the initial condition.
	Apply for the first control move u_t from the solution above.
	end for

2.2.2 Single shooting

As the formulation of NMPC is presented in the previous section, it is now natural to look into methods for solving the non-linear problem (NLP) which arises within the non-linear MPC. The solution method chosen in this thesis is single shooting. This choice is made based on the results from [3], where single shooting was found to be the most reliable method in comparison to methods such as Multiple shooting and Direct collocation. This was shown for a similar production system and for the same optimization software as used in this thesis.

Single shooting is a optimization method where states and algebraic variables are treated as implicit functions of control inputs. This means that they can be eliminated from the optimization problem, reducing the decision variables to only the control inputs.

The single shooting method can be efficient due to the fact that it is simple, and that it reduces the size of the NLP. It is however reliant on a strong differential algebraic equation (DAE) solver to function optimally. The interaction between the NLP solver and the DAE solver is described in the figure below.

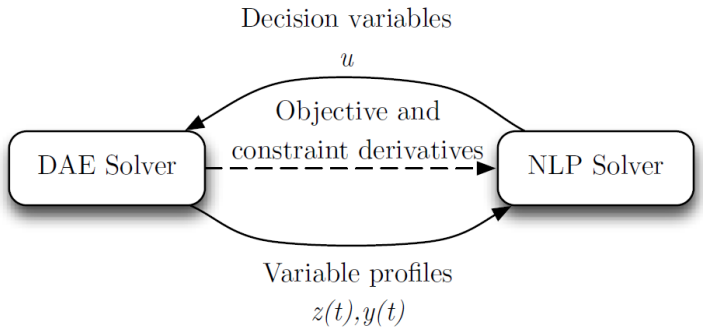

Figure 2.1: The figure illustrates the interaction between the NLP solver and the DAE solver. The DAEs are solved in an outer loop providing variable profiles and derivative information to the NLP solver, which in turn uses these to compute a new sequence of control inputs u. The figure is taken from [3].

2.2.3 Robustness

Even though the application of a NMPC to a system allows the user to provide more correct system equations compared to the linear MPC, there will always be some deviation between the model provided to the NMPC and the real system. This has been considered in [7] and [4], where the preserved quality of the performance for the feedback system when the dynamic behaviour of the real system differs from the behaviour of the assumed system in the model, is referred to as robustness.

To include the deviation in the model equations with regard to the real system a new formulation of the model equations can be defined as follows

$$\dot{z} = f(z(t), u(t), d(t)), \tag{2.6}$$

where d represents the uncertainty. Because of this uncertainty, there will be a difference between the predicted open-loop and the actual closed-loop trajectory. This represents a problem when conducting stability analysis of the system, as there is not a single future trajectory, but a whole three of possible trajectories which must be analyzed. As a result of this, analysis of robustness properties in nonlinear NMPC is still considered as an unsolved problem in general. Depite this however, the controller may work very well in practice, even with considerable differences between the model and the real system.

2.2.4 Output feedback NMPC

So far it has been assumed that all the necessary system states are perfectly accessible through measurements. In general, this is not the case, and therefore it is necessary to include a state observer in the control loop. This is discussed in [4], where the "certainty equivalence principle" is presented as the most often used approach for output feedback NMPC. This means that the estimated state $\hat{z}$ first is calculated by a state observer, before the estimated state is used in the model predictive controller. However, this is no guarantee for stability, as often only local stability of the closed loop is achieved, even if the observer error is exponentially stable. The "certainty equivalence principle" is still applied successfully in many applications, and it is applied in this thesis. A schematic presentation of this principle is given in figure 2.2.

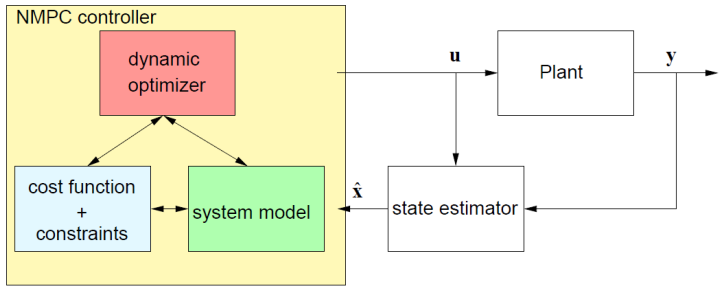

Figure 2.2: The figure is taken from [4] and shows how the state estimator is coupled with the plant and the NMPC controller.

Chapter 3

Implementation tools

The implementation in this thesis is performed on multiple platforms, which are interfaced together in order to benefit from the strengths of each individual platform. In this chapter these platforms are presented before the interface between them is described.

3.1 Optimica

High level modelling frameworks such as Modelica are becoming increasingly used in industrial applications, as they allow for rapid development of complex large-scale models. Traditionally, the target of such models has been simulation, and therefore several tools supporting simulation of Modelica models exist. As dynamic optimization of large scale systems has become increasingly used during the last decades, the motivation for interfacing these high level modelling frameworks with NLP solvers has grown.

Optimica is a extension to Modelica and enables compact and intuitive optimization problem formulations based on Modelica models. Figure 3.1 and 3.2 demonstrates how an optimization problem can be defined in Optimica based on a Modelica model.

The optimization problem formulations in Optimica make it possible to interface the Modelica and Optimica code with an NLP solver. JModelica.org is a Modelica-based open source platform specifically targeted at dynamic optimization that also supports the Optimica extension. By importing the Modelica and Optimica models into the JModelica.org platform, they can be handled by all the different NLP solvers which are available through JModelica.org. This platform will be further described in the following section.

```
model tank
  import SI = Modelica.SIunits;
  parameter SI.Area a1=0.03;
  parameter SI.Area a2=0.02;

  parameter SI.Area A1=8.4;
  parameter SI.Area A2=7.1;
  parameter SI.Area A_u=1;

  parameter SI.Acceleration g = 9.81;
  parameter Real k = 0.4;    // [.]

  Real z1(start=0.4,fixed=true); // State z1 with initial value  [m]
  Real z2(start=3,fixed=true);  // State z2 with initial value [m]

  parameter Real u_initGuess = 0.1;
  input Real u(start=u_initGuess,fixed=true); // Control input with initial guess
       [m^3/s]

equation
  der(z1) = -(a1/A1)*sqrt(2*g*z1)+(a2/A2)*sqrt(2*g*z2);
  der(z2) = -(a2/A2)*sqrt(2*g*z2)+(k/A_u)*u;
end tank;
```

Figure 3.1: Example of Modelica code for a system of two coupled tanks taken from [3].

```
optimization ocp(objective=zc(finalTime),startTime=0,finalTime=600)
   extends tank;            // Include tank model equations
   Real zc(start=0,fixed=true);
   constant Real z1r = 2;     // Reference value
equation
   // Cost function state
   der(zc) = (z1-z1r)^2+u^2;
constraint
   // State bounds z_min <= z <= z_max
   z1<=5;
   z1>=0.1;

   z2<=5;
   z2>=0.1;

   // Control input bounds u_min <= u <= u_max
   u<=0.3;
   u>=0;
end ocp;
```

Figure 3.2: Example of Optimica code for a system of two coupled tanks taken from [3].

3.2 The JModelica.org platform

As mentioned in the previous section, JModelica.org is an open source platform targeted at dynamic optimization. The JModelica.org platform architecture is shown in Figure 3.3 where one can see that it consists of two main parts, the compiler and the JModelica.org Model Interface (JMI) run-time library. In the compiler, the Modelica and Optimica code is first translated into a flat model description, which is further transformed into a hybrid DAE before a C or XML code representation of the model is generated. The C code representation contains the model equations on a form which is suitable for efficient evaluation. The XML code on the other hand, contains all the model meta data, such as variable names and parameter values.

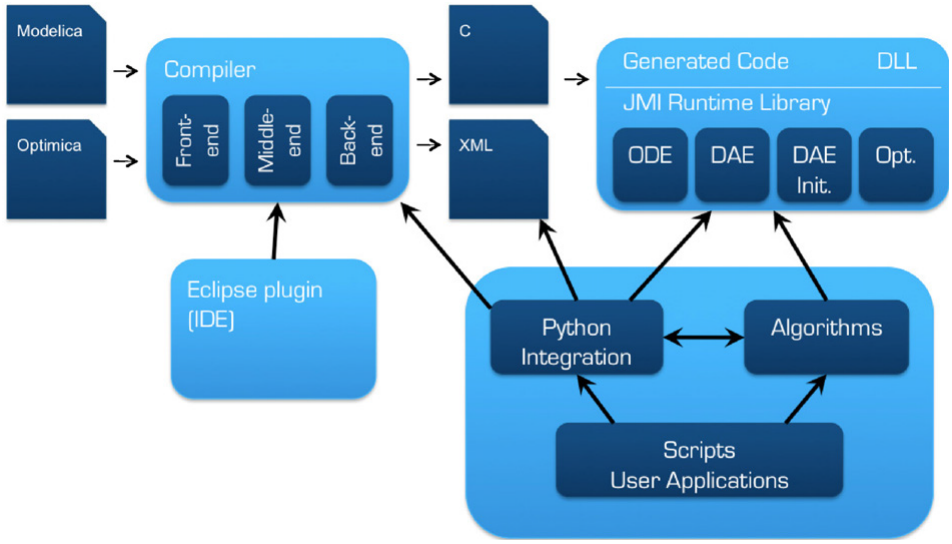

Figure 3.3: JModelica.org platform architecture.

Another important thing to notice from Figure 3.3 is that JModelica.org is integrated with Python. This allows the user to import models from Modelica/Optimica into Python, where it can be interfaced with other frameworks. One such framework is the symbolic framework CasADi, which is a free open source framework for automatic differentiation and optimal control. CasADi will be discussed further in section 3.3.

3.2.1 Advantages and Disadvantages using JModelica.org

JModelica.org is a platform that mainly provides a lot of opportunities, but there are also a few drawbacks. Among the advantages we have that the model can be declared independent of casuality, which means that the model will be compiled into an index-1 model regardless of how it originally was defined. The platform also makes it easy to interface NLP solvers and Integrators with the model implemented in Modelica, providing effective simulations and NLP solutions. The fact that everything can be written in python, is also clearly an advantage as python is an easy to read and widely used programming language.

The drawbacks are currently that JModelica.org is in continuous development, and as a result, some features can be unstable and may contain bugs. In addition the interface between Modelica and CasADi is not very robust, and this has caused the work with these models to take up at lot more time than expected.

3.3 CasADi

CasADi is a symbolic framework for automatic differentiation and numeric optimization that uses the syntax of computer algebra systems, and therefore allows the user to construct symbolic expressions of scalar or matrix-valued operations. These symbolic expressions can be efficiently differentiated using state of the art algorithms for automatic differentiation. As JModelica.org, CasADi has a full-featured Python front end, and it also contains back ends to code for optimization and simulation, for example Sundials which is an efficient tool for solving initial value problems for ODE and DAE systems. Another back end code is IPOPT, which is a interior point optimizer for large scale non-linear optimization. The platform is built as a large class hierarchy, where automatic differentiation forms the foundation of all other classes, as illustrated in Figure 3.4.

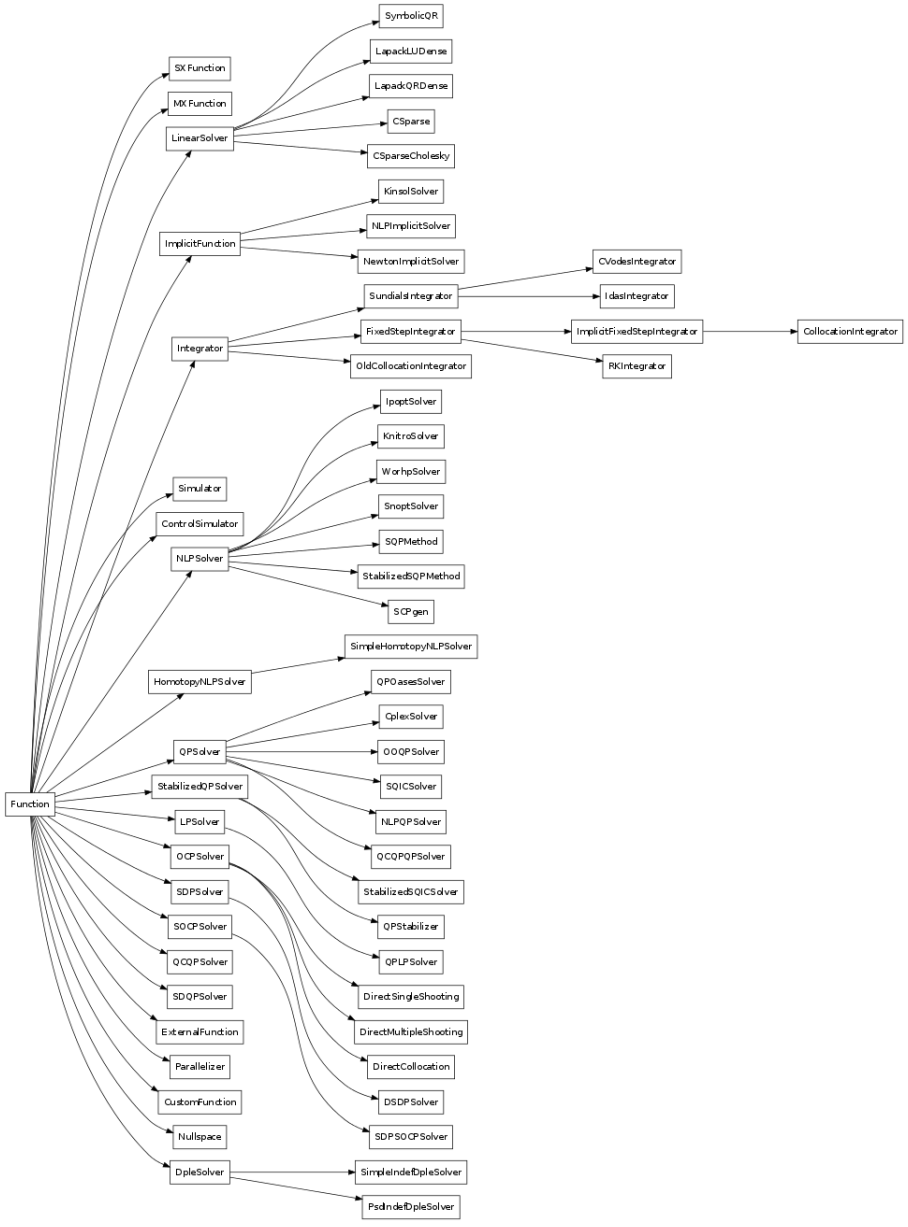

Figure 3.4: The CasADi class hierachy.

3.4 The OLGA simulation software

OLGA is the industry standard tool for transient simulation of multiphase petroleum production. Currently 7.3 is the latest version, but the tool is in continuous development. The software originated at the Institute for Energy Research (IFE) in 1980, but the oil industry did not start using it before 1984 after Statoil had supported its development for 3 years. Data from the large scale flow loop at SINTEF, and later from the medium scale loop at IFE was essential for the development of the multiphase flow correlations and also for the validation of OLGA. Oil companies have since then supported the development and provided field data to help manage uncertainty, predominantly within the OLGA Verification and Improvement Project (OVIP). OLGA has been commercially available since the SPT Group started marketing it in 1990 and is used for networks of wells, flowlines and pipelines and process equipment, covering the production system from bottom hole into the production system. Further, OLGA comes with a steady state pre-processor included, which is intended for calculating initial values to the transient simulations, but which also is useful for traditional steady state parameter variations. However, the transient capabilities of OLGA dramatically increase the range of applicability compared with steady state simulators. In OLGA it is also possible to set up an OPC-server, in order to interface OLGA with other modules. OPC stands for Object Linking and Embedding(OLE) for Process Control, and is a standard for communication of real-time data between control devices from different manufacturers. In this thesis, the open source OPC toolkit OpenOPC is used to communicate between the JModelica.org framework and OLGA through Python.

3.5 How the different JModelica.org components have been used in this project

In this section, an explanation of how the various simulation and optimization tools presented in the earlier sections are coupled together, and how they are represented in implementation of an MPC. This is illustrated in Figure 3.5 and from this one can see that the MPC is designed using interfaces between Modelica/Optimica, JModelica.org and OLGA. The process plant, which imitates a real production system, is simulated in OLGA. Further, OLGA communicates with Python using an OPC server, which allows the the state estimator to receive measurements from the plant. The state estimation problem is then solved in JModelica.org using CasADi, before the estimated state is passed to the NMPC-controller where an optimization problem defined by a cost function and constraints from Optimica, and a system model defined in Modelica is solved by a dynamic optimizer in JModelica.org with CasADi.

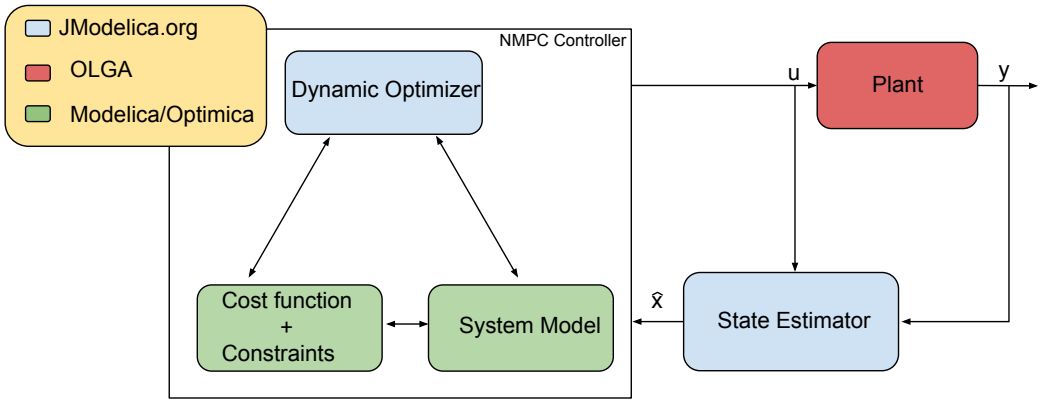

Figure 3.5: Overview of how the software is coupled together in the NMPC design.

Chapter 4

Modeling and optimal control formulation

This chapter first gives a presentation of the simplified models in Modelica inspired by [9] and [3]. Thereafter, the corresponding more complex models instantiated in OLGA are presented. At last, the optimal control problem formulations used in this thesis are presented.

4.1 Modelica Well model

The Modelica well model is based on the well model proposed in [3], which is based on the Eikrem model [11]. The model was constructed with the aim to capture the dynamic behavior phenomenon known as *casing-heading*, which leads to oscillating production for low gas injection rates. Figure 4.1 shows the main components of the gas-lift which the model is based on.

The model contains three state variables; mass of gas in the annulus volume m_{ga}, mass of gas in the tubing volume m_{gt} and mass of liquid in the tubing volume m_{lt}. The change of these states with respect to time is given by the difference between the mass flow into the volume and the mass flow out, as given in equation (4.1)

$$\dot{m}_{ga} = \omega_{gl} - \omega_{gi} \tag{4.1a}$$
$$\dot{m}_{gt} = \omega_{gr} + \omega_{gi} - \omega_{gp} \tag{4.1b}$$
$$\dot{m}_{lt} = \omega_{lr} - \omega_{lp}. \tag{4.1c}$$

The different mass flows are denoted by ω and two subscripts, where the first specifies if it is a gas g or a liquid l, and the second defines the position on Figure 4.1, where l refers to the Gas lift choke, i to the injection valve, r to the reservoir and p to the production choke. This notation is used throughout this thesis.

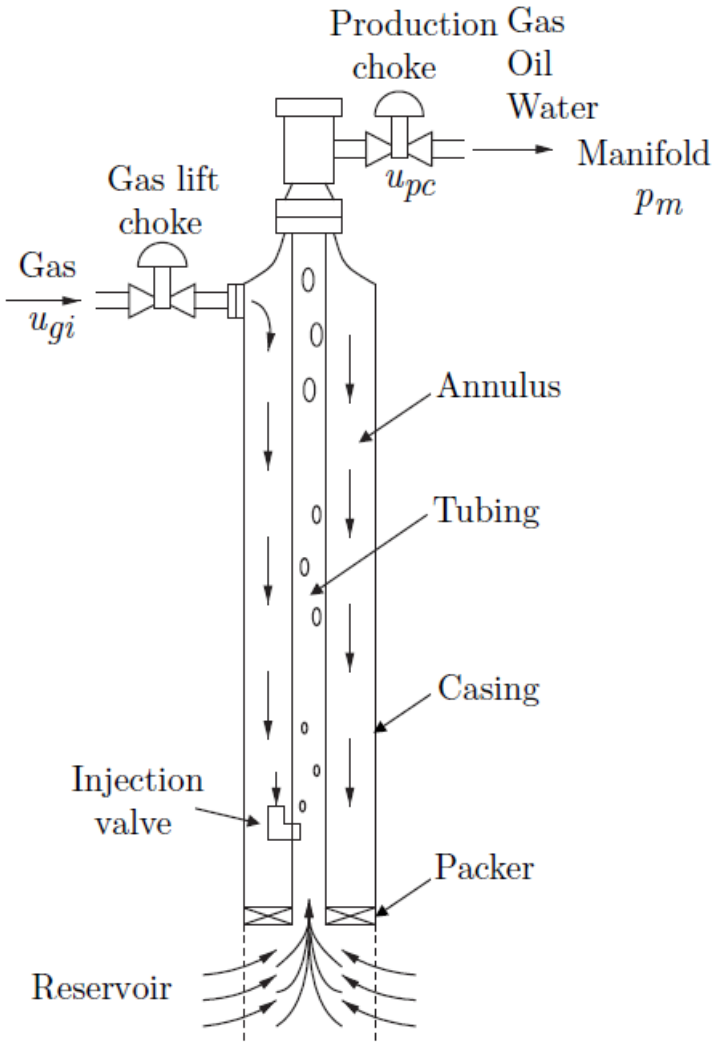

Figure 4.1: The figure is figure 4.1 in [3] and shows how the artificially gas-lifted well is modelled.

The algebraic equations describing the different mass flows in the model are given in equation (4.2). It should be noted that ω_{gl} is defined as an input, ω_{gi} and ω_p are defined as a function of the square root of the pressure difference across the valve, while ω_{lr} is calculated using Vogel's equation [12].

$$m_t = m_{gt} + m_{lt} \tag{4.2a}$$
$$\omega_{gl} = u_{gi} \tag{4.2b}$$
$$\omega_{gi} = C_{iv}\sqrt{\rho_{gi} max\{0, p_{ai} - p_{ti}\}} \tag{4.2c}$$
$$\omega_p = C_{pc}\sqrt{\rho_p max\{0, p_r - p_m\} u_{pc}} \tag{4.2d}$$
$$\omega_{gp} = \frac{m_{gt}}{m_t}\omega_p \tag{4.2e}$$
$$\omega_{lp} = \frac{m_{lt}}{m_t}\omega_p \tag{4.2f}$$
$$\omega_{\omega p} = r_{\omega c}\omega_{lp} \tag{4.2g}$$
$$\omega_{op} = (1 - r_{\omega c})\omega_{lp} \tag{4.2h}$$
$$\omega_{lr} = \rho_l Q_{max}\left(1 - (1 - C)\left(\frac{p_{bh}}{p_r}\right) - C\left(\frac{p_{bh}}{p_r}\right)^2\right) \tag{4.2i}$$
$$\omega_{gr} = r_{glr}\omega_{lr} \tag{4.2j}$$
$$r_{glr} = (1 - r_{\omega c})r_{gor} \tag{4.2k}$$
$$\tag{4.2l}$$

Based on the assumption of ideal gas and on the assumption that the liquid in the tubing is perfectly mixed at all times, the equations below are provided for the densities used in the model

$$\rho_{gi} = \frac{M_g}{RT_a}p_{ai} \tag{4.3a}$$
$$\rho_p = \frac{\rho_l M_g p_p m_t}{\rho_l R T_t m_{lt} + M_g p_p m_{gt}} \tag{4.3b}$$
$$\rho_l = r_{\omega c}\rho_\omega + (1 - r_{\omega c})\rho_o. \tag{4.3c}$$

The last parameters that must be defined before the model description is complete, are definitions of the pressures involved. These are given based on assumptions of ideal gas and uniform density in each volume. In addition the model does not consider friction in

the estimation of the pressure drop. The result is the following pressure equations

$$p_{ai} = \left(\frac{RT_a}{V_a M_g} + \frac{g}{2A_a} \right) m_{ga} \tag{4.4a}$$

$$p_p = \frac{RT_t m_{gt}}{M_g V_t - M_g \nu_l m_{lt}} - \frac{g m_t}{2A_t} \tag{4.4b}$$

$$p_{ti} = p_p + \frac{g m_t}{A_t} \tag{4.4c}$$

$$p_{bh} = \frac{\left(1 + r_{glr} + \frac{r_{glr} M_g g L_\omega}{2RT_t}\right) p_{ti} + \rho_l g L_\omega}{1 + r_{glr} - \frac{r_{glr} M_g g L_\omega}{2RT_t}}. \tag{4.4d}$$

4.2 Modelica Flowline model

The Modelica flowline model, as illustrated in Figure 4.2, is based on the low-order model proposed in [9]. The flowline model is divided in two parts; a horizontal slightly decreasing pipeline and a vertical riser. This model is designed to capture the phenomenon known as riser slugging, which is characterised by severe flow and pressure oscillations. This phenomenon can often occur in pipeline-riser systems which transport an oil and gas mixture from the seabed to the surface facilities [13], and can occur if the mass flow in the flowline is too large. The result is an accumulation of mass at the low point, as illustrated in Figure 4.3, which causes the gas portion of the transported fluid to be blocked from entering the riser. Eventually this will cause the pressure to build up until it gets high enough to remove the accumulated mass at the low point. This behaviour will continue as long as the mass flow at the inlet is too large, resulting in a oscillating system.

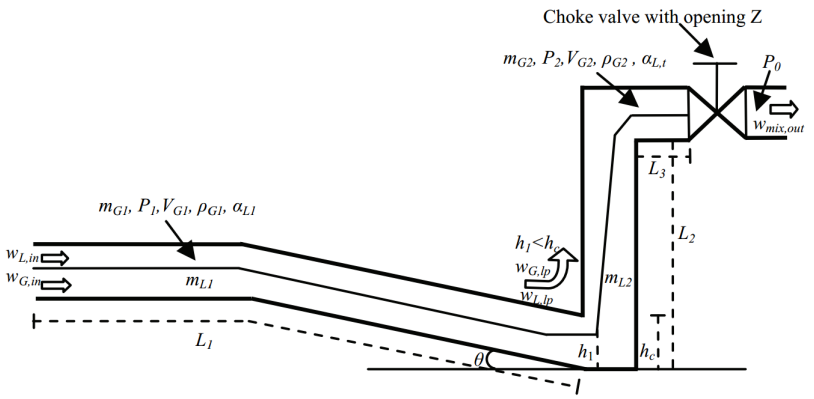

Figure 4.2: The figure shows a illustration of the flowline when there is no slugging.

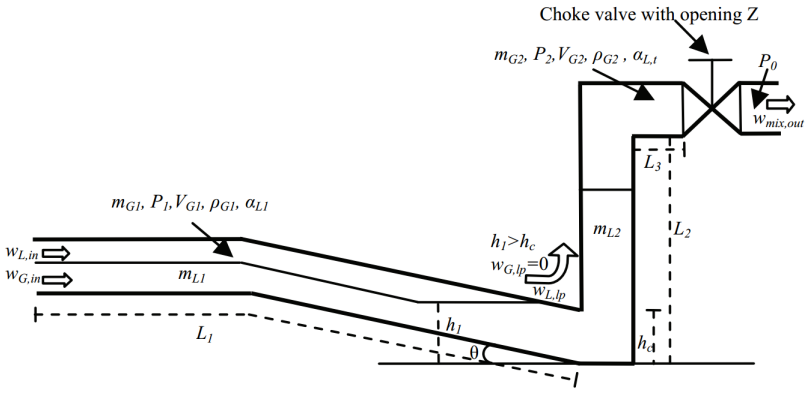

Figure 4.3: The figure shows a illustration of the flowline when there is slugging.

As for the well model, the governing state equations are written as mass balances, which are given in equation (4.5). The main concept of the notation is the same as the one described for the well model in the previous section, but with a few additional notes. These are the subscripts in and out, which refer to the inlet and the outlet of the flowline, and the subscript lp which refer to the low point. In addition p and r now denotes the pipeline and the riser

$$\dot{m}_{gp} = \omega_{g,in} - \omega_{g,lp} \tag{4.5a}$$
$$\dot{m}_{lp} = \omega_{l,in} - \omega_{l,lp} \tag{4.5b}$$
$$\dot{m}_{gr} = \omega_{g,lp} - \omega_{g,out} \tag{4.5c}$$
$$\dot{m}_{lr} = \omega_{l,lp} - \omega_{l,out}. \tag{4.5d}$$

The outflow conditions for the flowline are given as follows

$$\omega_{l,out} = \alpha_{lm,t}\omega_{mix,out} \tag{4.6}$$

and

$$\omega_{g,out} = (1 - \alpha_{lm,t})\omega_{mix,out}, \tag{4.7}$$

where the parameter $\omega_{mix,out}$ is defined by

$$\omega_{mix,out} = K_{pc} f(z) \sqrt{\rho_t (p_2 - p_0)}, \tag{4.8}$$

and $\alpha_{lm,t}$ is the approximated liquid-gas volume fraction at the top of the riser. Further, $(p_2 - p_0)$ describes the pressure drop over the choke valve, ρ_t the estimated fluid density at the top of the riser, and K_{pc} a tuning parameter.

In addition the level of liquid at the flowline low-point is approximated by

$$\bar{h}_1 = K_h h_c \bar{\alpha}_{lp}, \tag{4.9}$$

where h_c is the diameter of the flowline at the low-point, $\bar{\alpha}_{lp}$ the approximated liquid-gas volume fraction and K_h a tuning parameter.

Further, the mass flow of gas and liquid at the low-point are given by

$$\omega_{l,lp} = K_l A_l \sqrt{\rho_l \Delta p_l}, \tag{4.10}$$

$$\omega_{g,lp} = 0, \;\; h_1 \geq h_c, \tag{4.11}$$

and

$$\omega_{g,lp} = K_g A_g \sqrt{\rho_{gp} \Delta p_g}, \;\; h_1 < h_c, \tag{4.12}$$

where A_l and A_g are the free areas for liquid and gas flow and ρ_l and ρ_{gp} are the approximated densities of liquid and gas at the low-point. Further h_{lp} is the height of the liquid in the pipeline, while K_l and K_g are tuning parameters.

The pipeline-riser model is rather large and the complete list of parameter equations is therefore not presented here, but can be found in Appendix A.1. It is important to mention that the model is based on several assumptions, where the most important are the assumption of uniform liquid volume fractions in the pipeline and in the riser. In addition, opposed to the well model, friction is considered, but only for the liquid.

4.3 OLGA Well model

The well model in OLGA is in this thesis created with the intention that it should be comparable to the simplified well model presented in section 4.1. The OLGA model is based on a test case "WELL-GLV", which is found under the sample cases in OLGA 7.1.

As shown in Figure 4.4, the OLGA well model consists of two separate flow paths,one representing the annulus of the well, and the other representing the tubing. Both flowpaths are vertical, and have their start point at the sea bed. The annulus is 1500 meter long with a diameter equal to 0.15957 m, while the tubing is 1900 meter long with a diameter equal to 0.12361 m. At the bottom of the annulus, there is modelled a leak, which functions as a check-valve leading gas from the annulus into the tubing. At the bottom of the tubing, the flowpath is connected to a reservoir. A mass flow source is placed at the top of the annulus, which functions as an actuator to the system, similar to what is modelled in the simplified model Modelica model illustrated in Figure 4.1.

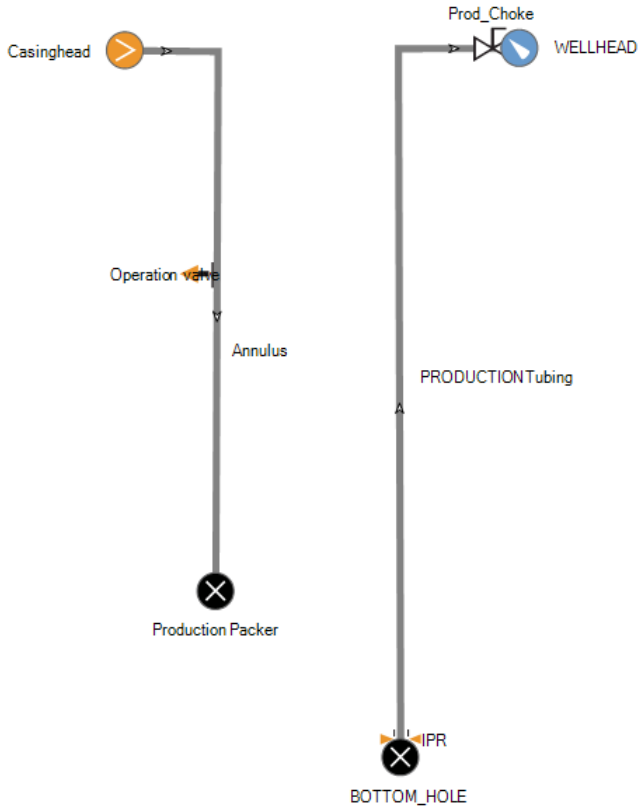

Figure 4.4: Overview of the OLGA well model

4.4 OLGA Flowline model

In section 4.2, the simplified flowline model implemented in Modelica was described. The flowline model in OLGA created in this thesis was based on a test case for severe slugging available in OLGA 5.0 and was adapted so that it easily could be compared to the Modelica model. The geometry of the flowline is shown in Figure 4.5, where the red pipe represents the pipeline, and the green pipe represents the riser. At the top of the riser, there is a small horizontal pipeline where a choke valve is inserted. The boundary condition at the end of this pipeline is given by a constant pressure, representing a separator.

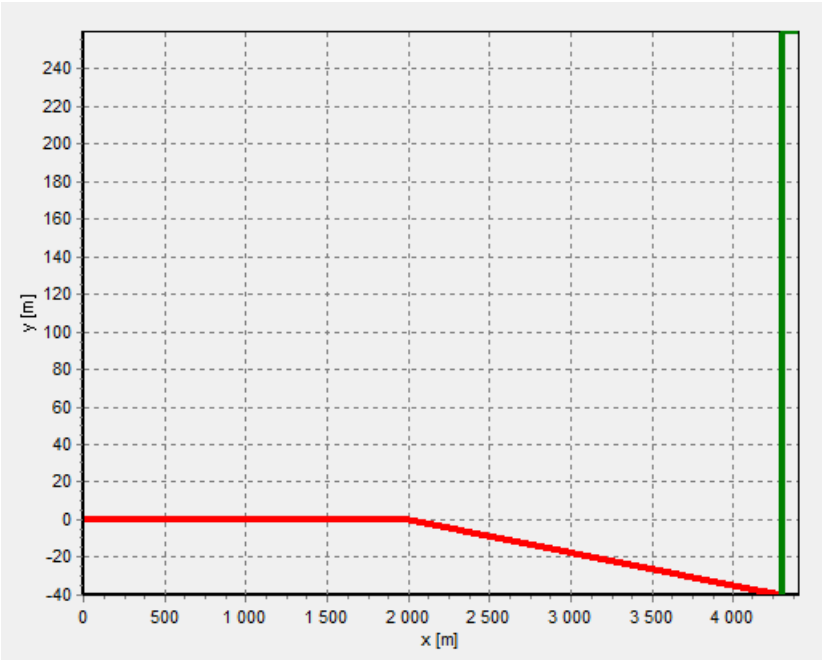

Figure 4.5: Overview of the OLGA flowline model

4.5 Approximating discontinuities

When importing the model equations from Modelica to JModelica.org, maximum and minimum functions are not supported. In order to handle this, the following smooth approximation is used

$$max(x, y) \approx f(x, y) = \frac{\sqrt{(x-y)^2 + \epsilon^2}}{2} + \frac{x+y}{2}, \tag{4.13}$$

where the parameter ϵ determines the approximation error and the steepness of the continuous function approximation.

4.6 Optimal control problem formulation

In this section, the optimal control problems used in this thesis are described. First a static optimization problem formulation is presented. This is used to find optimal set points, which are further utilised in the dynamic optimization problem formulation. Thereafter, the dynamic optimization problem which is used for both closed and open loop control of the system is defined.

4.6.1 Steady state optimization

The problem to be solved using steady state optimization is to find a optimal operating point for the pipeline-riser system.

Let the flowline system described in section 4.2, be written on the form

$$\dot{z}(t) = f(z(t), y(t), u(t)) \tag{4.14a}$$
$$z(t_0) = z_0 \tag{4.14b}$$
$$0 = g(z(t), y(t), u(t)), \tag{4.14c}$$

where z are the state variables, y the algebraic variables, and u the input given by the valve opening in the pipeline-riser system. The steady state optimization problem is then defined as

$$\min_{z,y,u} \Phi(z, u) = -w_{l,out} \tag{4.15a}$$

subject to

$$\dot{z} = 0, \tag{4.15b}$$
$$g(z, y, u) = 0, \tag{4.15c}$$
$$0 \leq u \leq 1, \tag{4.15d}$$

4.6.2 Dynamic optimization

Let the flowline system be written on the form (4.14). Then the dynamic optimization problem formulation is then defined as follows.

$$\min_{x,y,u} \Phi(z, y, u) = \int_{t_0}^{t_f} q_1(m_{gp} - m_{gp,ref})^2 + q_2(m_{lp} - m_{lp,ref})^2) \tag{4.16a}$$
$$+ q_3(m_{gr} - m_{gr,ref})^2 + q_4(m_{lr} - m_{lr,ref})^2 + q_5 u \tag{4.16b}$$

subject to

$$\dot{z} = f(x, y, u), \tag{4.16c}$$
$$z(t_0) = z_0, \tag{4.16d}$$
$$g(z, y, u) = 0, \tag{4.16e}$$
$$g_I(z, y, u) \leq 0, \tag{4.16f}$$
$$z_L \leq z \leq z_U, \tag{4.16g}$$
$$y_L \leq y \leq y_U, \tag{4.16h}$$
$$u_L \leq u \leq u_U, \tag{4.16i}$$
$$t \in [t_0, t_f]. \tag{4.16j}$$

Chapter 5

Simulations

The basis of the models used in both OLGA and Modelica was explained in the previous chapter. In this chapter, the process of fitting the Modelica models to the OLGA models is described and the results are presented. Further the implementation of state estimation based on these findings will be tested.

5.1 Well

In section 4.1, the theoretical background for the Modelica well model was described. When implementing the corresponding model in OLGA, parameters from earlier implementations of the Modelica well model were used as a basis for realistic well specifications in OLGA. In Table 5.1, the main parameters of the OLGA well model can be found.

Based on these values the well model was simulated, and the plots on Figure 5.1-5.4 illustrate its behaviour. From these it can be seen, that with $u_{gl} = 0.5$, the well reaches a stable operating point. In Figure 5.2 and 5.4, u_{gl} is set to $0.1kg/s$. It is clear that the system now reaches an unstable operating point.

Table 5.1: Simulation parameters for the OLGA Well model

Component	Parameter description	Value	Unit
Annulus			
	Start point	$(x, y) = (0, 0)$	[m]
	End point	$(x, y) = (0, -1500)$	[m]
	Diameter	0.15957	[m]
	Roughness	$4.572 \cdot 10^{-5}$	[m]
	Number of sections	150	[-]
	Section length	10	[m]
Tubing			
	Start point	$(x, y) = (50, -1900)$	[m]
	End point	$(x, y) = (50, 0)$	[m]
	Diameter	0.12361	[m]
	Roughness	$4.572 \cdot 10^{-5}$	[m]
	Number of sections	190	[-]
	Section length	10	[m]
Leak from Annulus to Tubing			
	Diameter	0.0105	[m]
	Valve type	OLGAVALVE	[-]
Production Choke			
	Diameter	0.0185	[m]
	Model type	HYDROVALVE	[-]
Well			
	Reservoir pressure	$250 \cdot 10^5$	[Pa]
	Production Option	VOGELS	[-]
	Q_{max}	0.0031	$[\mathrm{m}^3/\mathrm{s}]$
	Reservoire Temperature	348.15	[K]
	Water cut	0.8	[-]

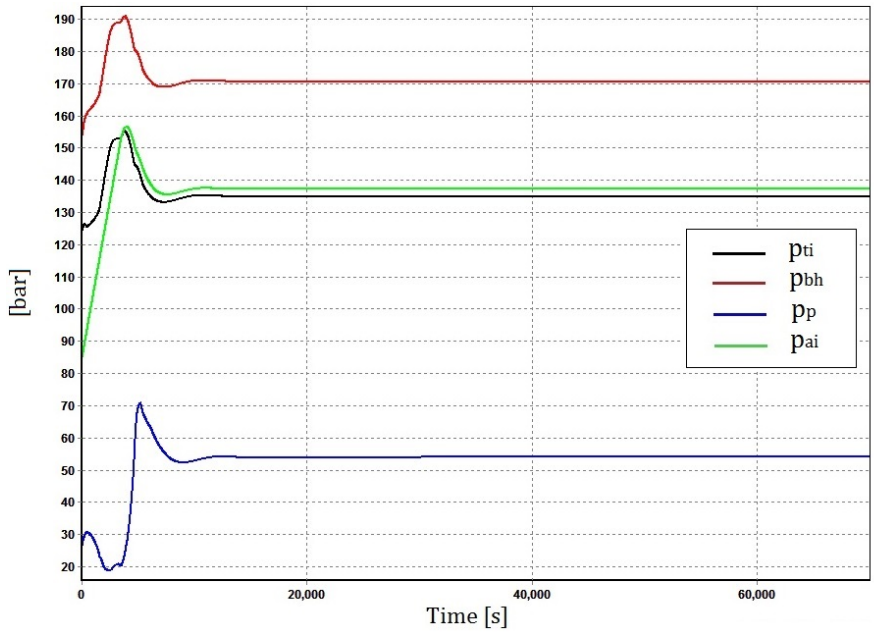

Figure 5.1: Simulated well pressures in OLGA with $u_{gl} = 0.5\text{kg/s}$

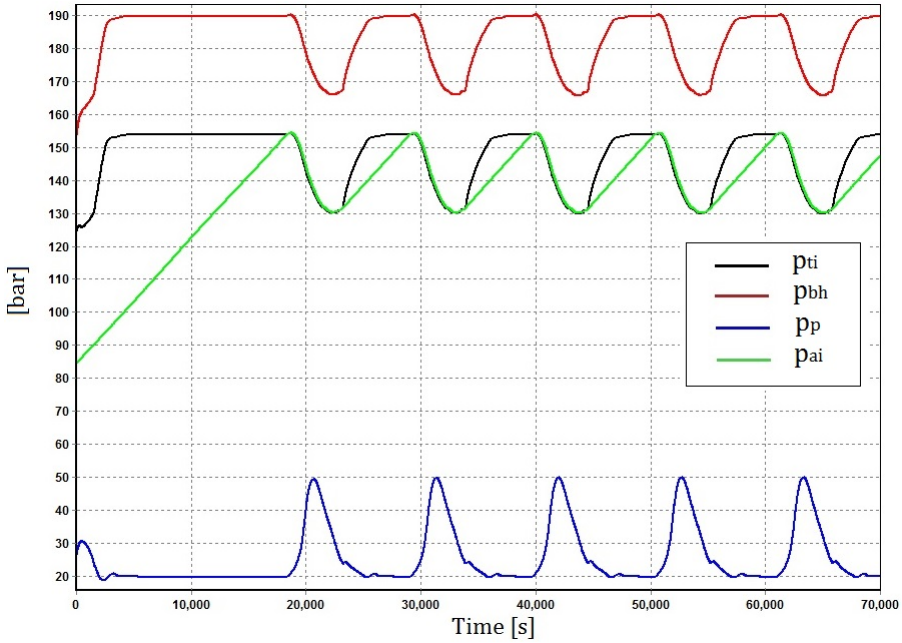

Figure 5.2: Simulated well pressures in OLGA with $u_{gl} = 0.1\text{kg/s}$

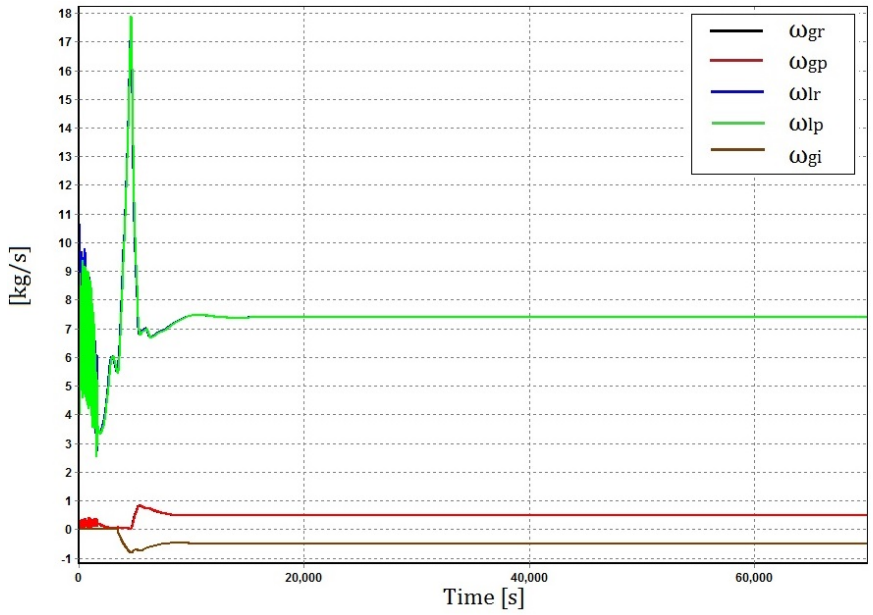

Figure 5.3: Simulated well flows in OLGA with $u_{gl} = 0.5\text{kg/s}$

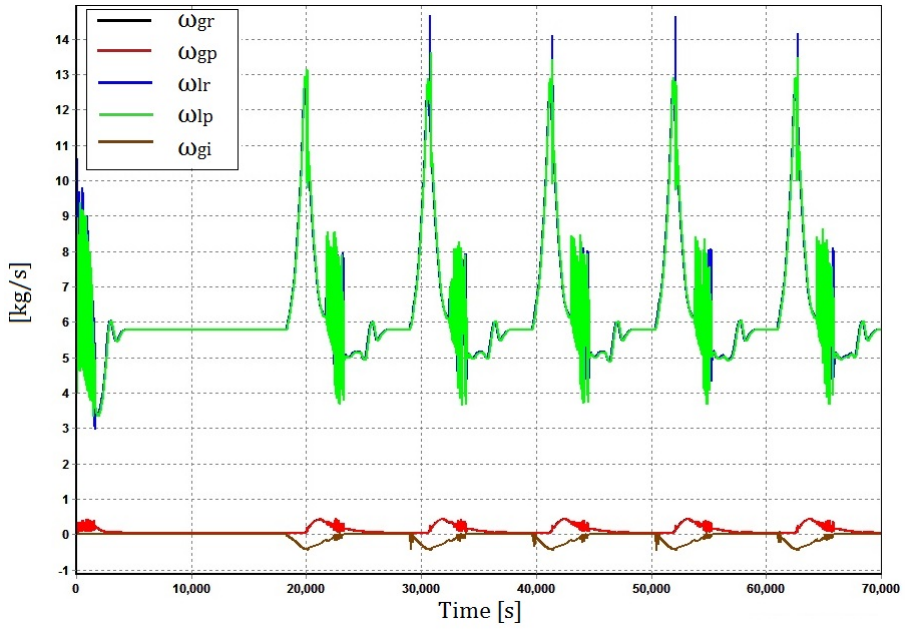

Figure 5.4: Simulated well flows in OLGA with $u_{gl} = 0.1\text{kg/s}$

5.1.1 Fitting the OLGA well model with the Modelica well model

When fitting the well models, several stategies were tried. First, the well parameters where set to best fit the OLGA model parameters shown in Table 5.1, and further the tuning parameters C_{iv} and C_{pc} were estimated. This approach did not prove to be successful, as the behaviour of the resulting Modelica model greatly differed from the OLGA model. Next, a more methodical approach was tried. First, the pressure in the annulus was compared for both models, as a function of mass and gas in the annulus m_{ga}. From this it was found that the Modelica model gave a good approximation of the pressure in the annulus. Thereafter, the pressure in the tubing was compared for the two models. It was found that the estimated pressure in the mass center of the well, given by

$$p_{cm} = \frac{p_{ti} + p_p}{2}, \tag{5.1}$$

gave a good approximation of the pressure in the mass center of the OLGA model tubing. Despite of this, the pressures at the top and bottom of the tubing volume, p_{ti} and p_p, were both poorly estimated.

In order to get a better approximation of the pressures in the tubing, the Modelica model was slightly changed. As the error seemed to be connected with the difference between the pressure at the center of mass in the tubing p_{cm}, and the pressures at the top and bottom of the tubing p_{ti} and p_p, the model equations directly related to these parameters were modified. In the original model equation given in (4.4b), the first term represents the pressure at the mass center. As this was found to give an accurate value, the second term was modified by adding a tuning parameter θ resulting in the following modified equations for p_p and p_{ti}

$$p_p = \frac{RT_t m_{gt}}{M_g V_t - M_g \nu_l m_{lt}} - \theta \frac{g m_t}{2A_t} \tag{5.2a}$$

$$p_{ti} = p_p + \theta \frac{g m_t}{A_t}. \tag{5.2b}$$

From comparisons between the models θ was tuned to $\theta = 0.87$.

When this was done, the model was compared to OLGA by simulations where the initial conditions were set to the steady state values of the OLGA model. Figure 5.5 and 5.6 shows the pressures and the mass flows in Modelica, compared to the corresponding steady state values in OLGA. The parameters used in the comparison of the models can be found in Table 5.2. The analysis of the results will be presented in the discussion chapter.

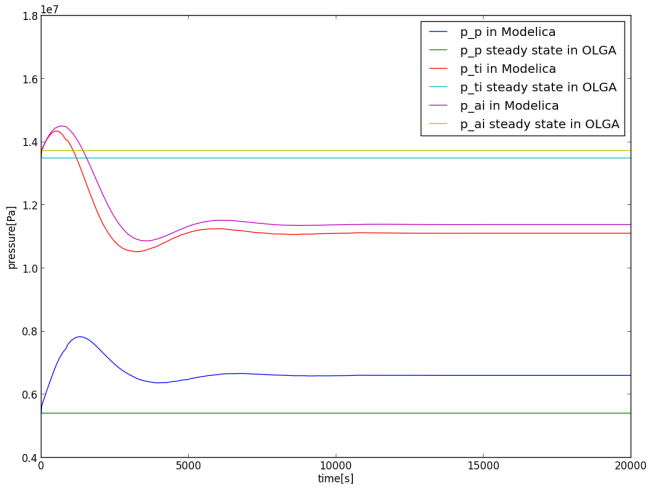

Figure 5.5: Simulation of the Modelica model pressures compared to the steady state values of the OLGA model.

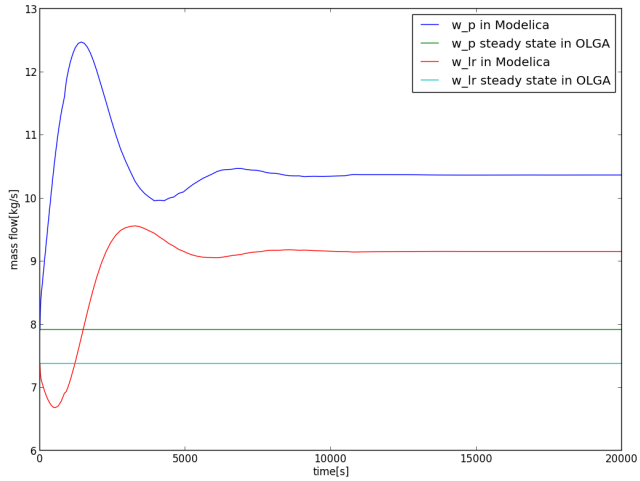

Figure 5.6: Simulation of the Modelica model mas flows compared to the steady state values of the OLGA model.

Table 5.2: Simulation parameters Modelica Well model

Parameter description	Symbol	Value	Unit
Temperature in the annulus	T_a	300	[K]
Temperature in the tubing	T_t	350	[K]
Molar mass of gas	M_g	0.0195	[kg/kmol]
Annular cross sectional area	A_a	0.02	[m^2]
Tubing cross sectional area	A_t	0.012	[m^2]
Reservoir pressure	p_r	$25{\cdot}10^6$	[Pa]
Annular volume	V_a	30	[m^3]
Tubing volume	V_t	18	[m^3]
Tubing length from the injection point to the reservoir	L_w	400	[m]
Well water cut	r_{wc}	0.8	[−]
Well gas oil ratio	r_{gor}	0.38783	[−]
Theoretical absolute open flow from reservoir	Q_{max}	0.0151	[m^3/s]
Water density	ρ_w	1030	[kg/m^3]
Oil density	ρ_o	930	[kg/m^3]
Vogels equation parameter	C	0.8	[−]
Tuning parameter	C_{iv}	$1.0068{\cdot}10^{-4}$	[m^2]
Tuning parameter	C_{pc}	$0.4062{\cdot}10^{-4}$	[m^2]

5.1.2 State Estimation og MPC

As can be seen from Figure 5.5 and 5.6, the Modelica well model was not fitted to the OLGA model in a satisfactory manner. In the next section it will be shown that the fitting of the Modelica flowline model was much more successful. Based on these findings, it was decided to focus the efforts on state estimation and MPC to the flowline model.

5.2 Flowline

When implementing the flowline model in Modelica, the equations presented in section 4.2 where used. As mentioned in section 4.4, the OLGA model was created based on an example case for flowline slugging in OLGA 5.0, which was found to be a good foundation for simulating flowline slugging. The main parameters used in the OLGA model are described in Table 5.3.

Table 5.3: Simulation parameters OLGA Well model

Component	Parameter description	Value	Unit
Pipeline			
	Start point	$(x, y) = (0, 0)$	[m]
	End point	$(x, y) = (4300, -40)$	[m]
	Diameter	0.12	[m]
	Roughness	$2.8 \cdot 10^{-5}$	[m]
	Number of sections	401	[-]
	Section length	10	[m]
Riser			
	Start point	$(x, y) = (4300, -40)$	[m]
	End point	$(x, y) = (4300, 260)$	[m]
	Diameter	0.10	[m]
	Roughness	$2.8 \cdot 10^{-5}$	[m]
	Number of sections	60	[-]
	Section length	5	[m]
Outlet Choke			
	Diameter	0.10	[m]
	Model type	HYDROVALVE	[-]

For these parameters a series of simulations were run to study the model behaviour. From these simulations it was found that the critical value of the valve opening for the transition between stable and oscillatory flow is around $z = 0.05$. Therefore the model fitting is performed around this value.

5.2.1 Fitting the OLGA Flowline Model with the Modelica Flowline Model

When fitting the flowline model this was done mainly using trial and error. As mentioned in the previous section a valve opening of $z = 0.05$ were used. The first step when fitting the models, was to change the Modelica model parameters, in order to get the same critical value opening as the OLGA model.

The first three figures compare the behaviour of the OLGA model and the Modelica model around the critical valve opening $z = 0.05$, where the first figure shows the mass flow of liquid at the outlet w_l, out, the second the pressure at the low point in the riser p_{lp} and the third the pressure at the top of the riser p_{tp}.

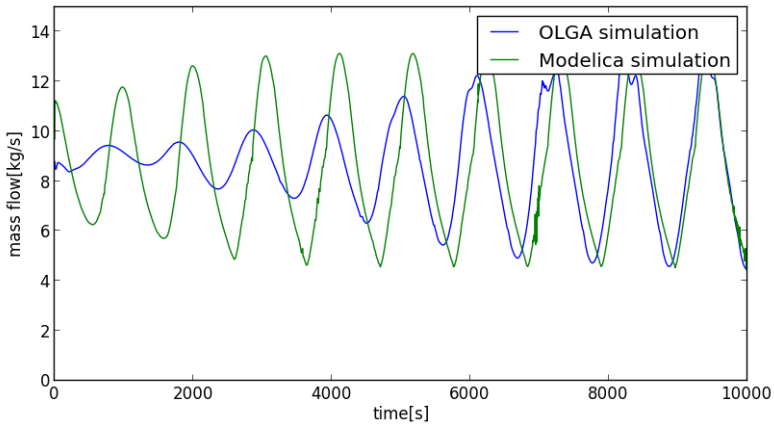

Figure 5.7: Mass flow of liquid at the outlet for 5% choke opening.

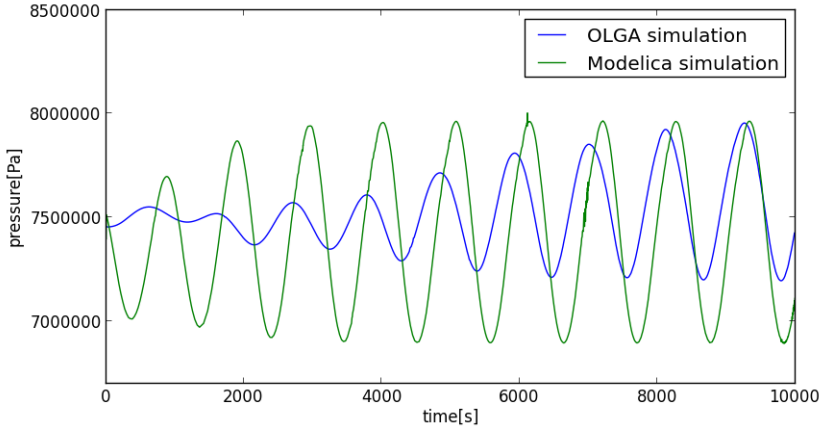

Figure 5.8: Pressure at the low point for 5% choke opening.

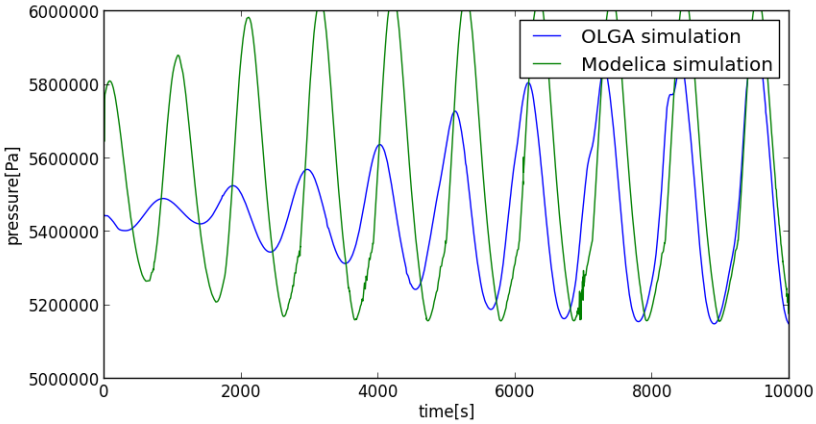

Figure 5.9: Pressure at the top of the riser for 5% choke opening.

The previous figures compared the models at the critical valve opening where the tuning were performed. The next three figures show the behaviour just below the critical valve opening, where both systems are at a stable operating point for the same parameters as in the previous figures.

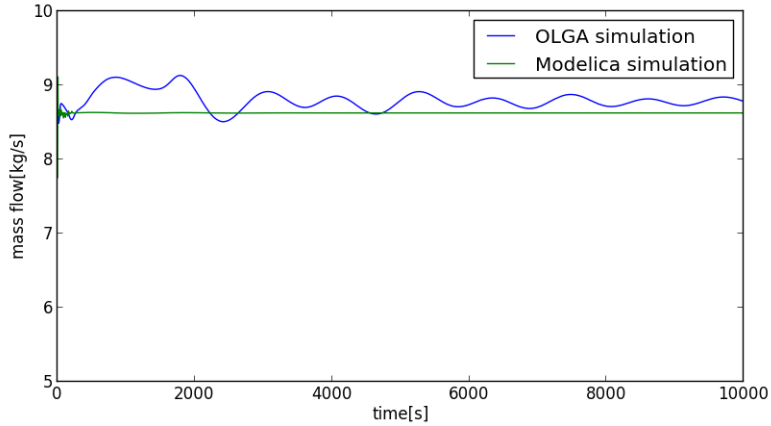

Figure 5.10: Mass flow of liquid at the outlet for 4% choke opening.

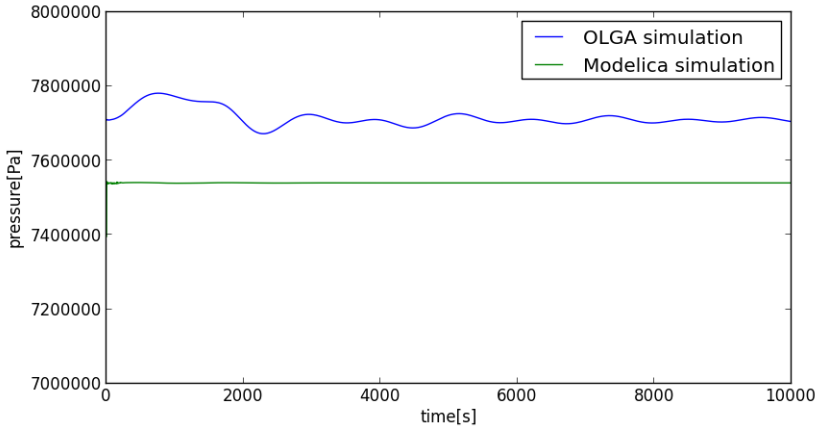

Figure 5.11: Pressure at the low point for 4% choke opening.

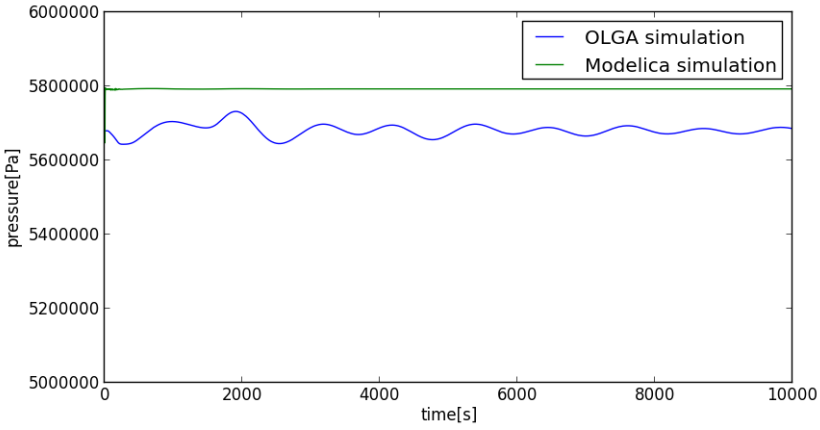

Figure 5.12: Pressure at the top of the riser for 4% choke opening.

It is also of interest to compare the behaviour of the systems, right above the critical valve value. In the next three figures, the same parameters as previously described are shown for the valve opening $z = 0.06$.

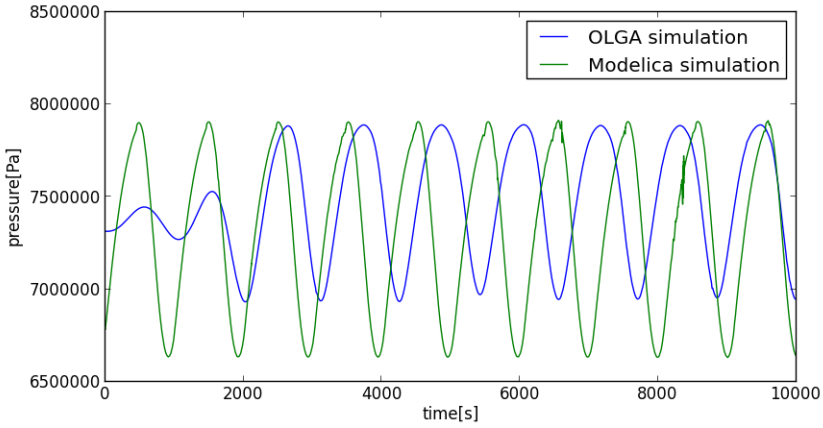

Figure 5.13: Pressure at the low point for 6% choke opening.

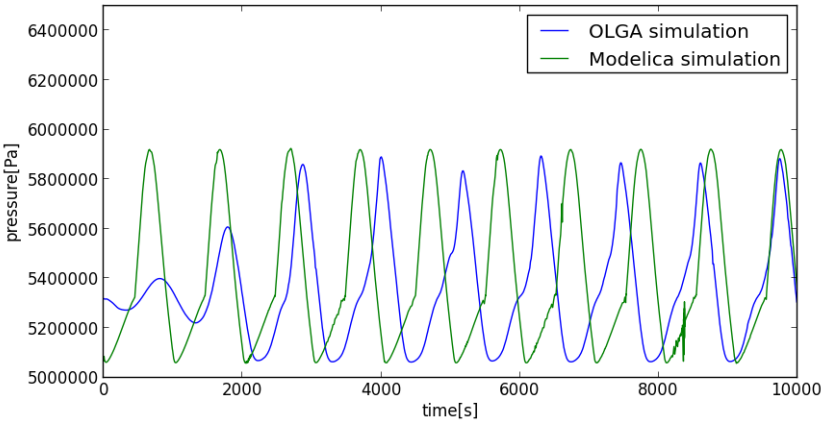

Figure 5.14: Pressure at the top of the riser for 6% choke opening.

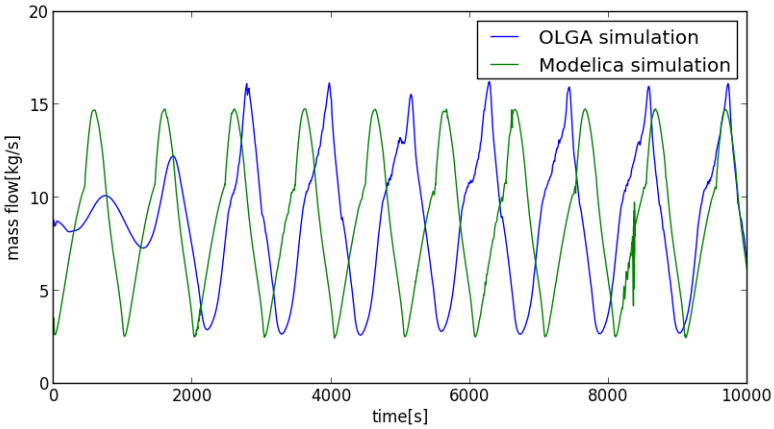

Figure 5.15: Mass flow of liquid at the outlet for 6% choke opening.

After observing that the models are fitted reasonably well around the critical valve opening, the value of z is set far away from this value at $z = 0.3$, and the systems are compared once again.

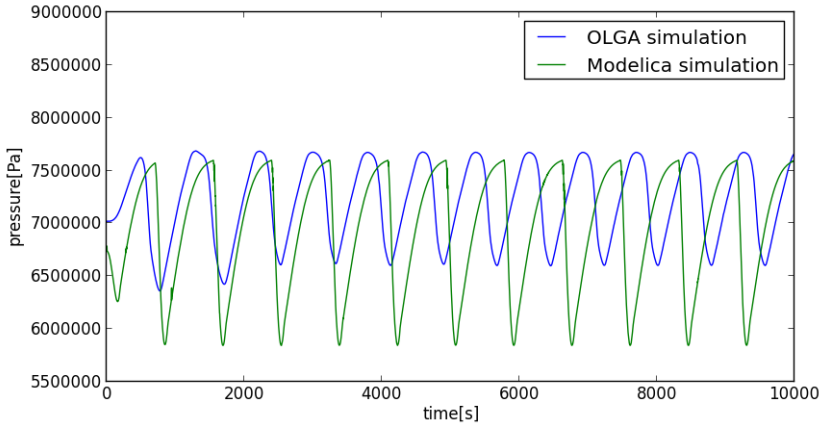

Figure 5.16: Pressure at the low point for 30% choke opening.

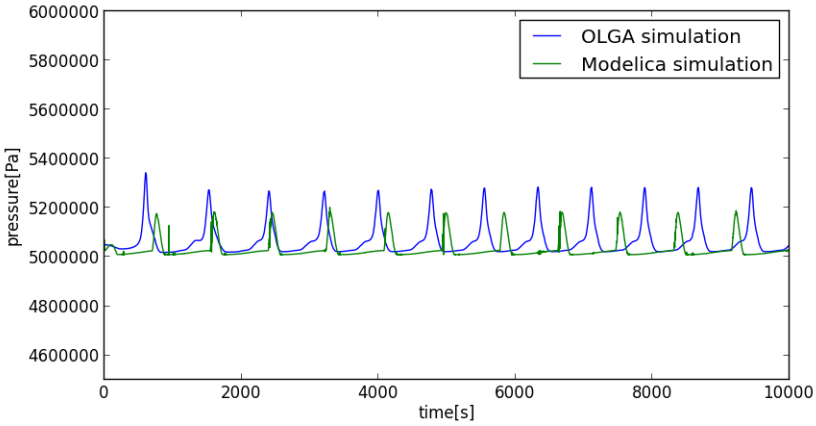

Figure 5.17: Pressure at the top of the riser for 30% choke opening.

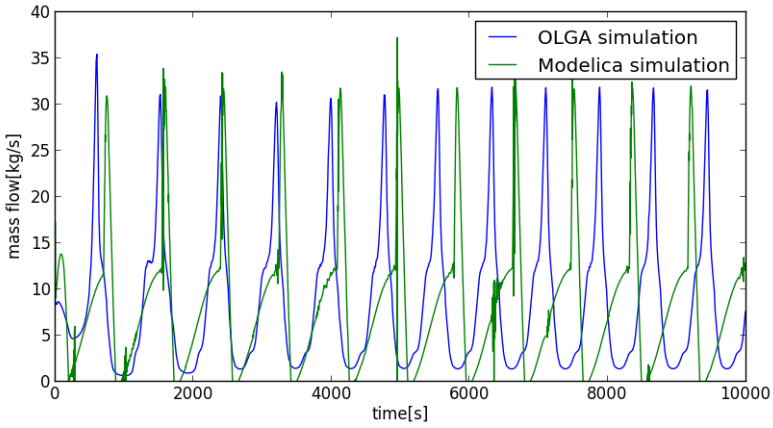

Figure 5.18: Mass flow of liquid at the outlet for 30% choke opening.

In the previous figures in this section, comparisons around various constant valve openings have been conducted. In order to say more about the similarity between the two models, the step response is compared. For each of the three plots below, the models have been simulated for 10000 seconds with input $z = 0.04$. After 10000 seconds, the valve opening is changed to $z = 0.042$.

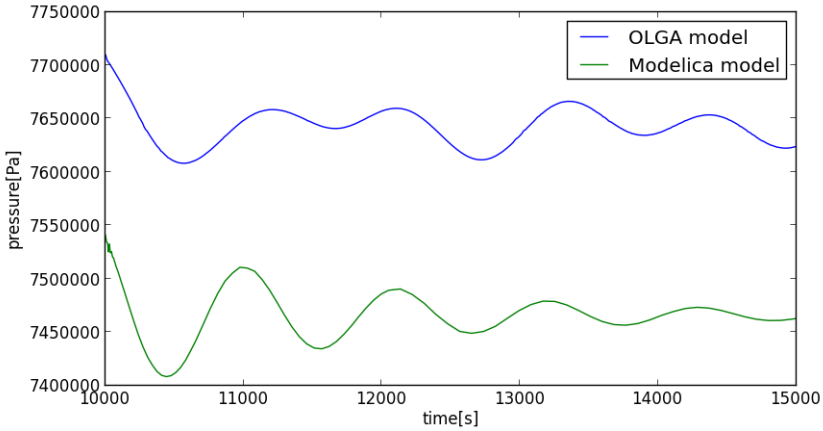

Figure 5.19: Step response of the pressure at the top of the riser.

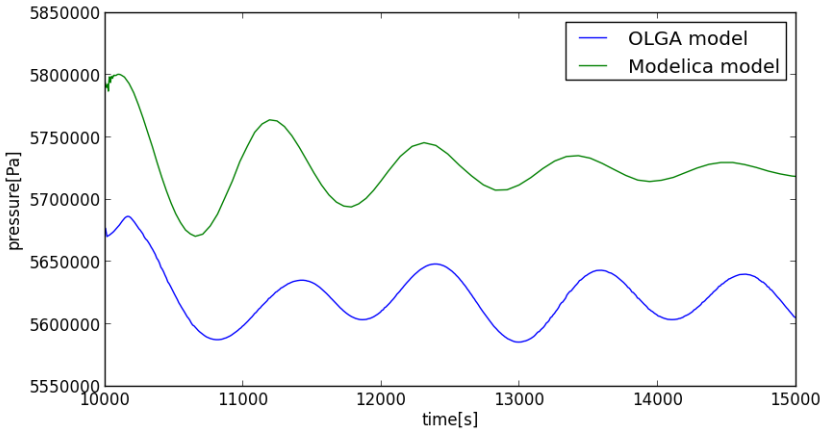

Figure 5.20: Step response of the pressure at the bottom of the riser.

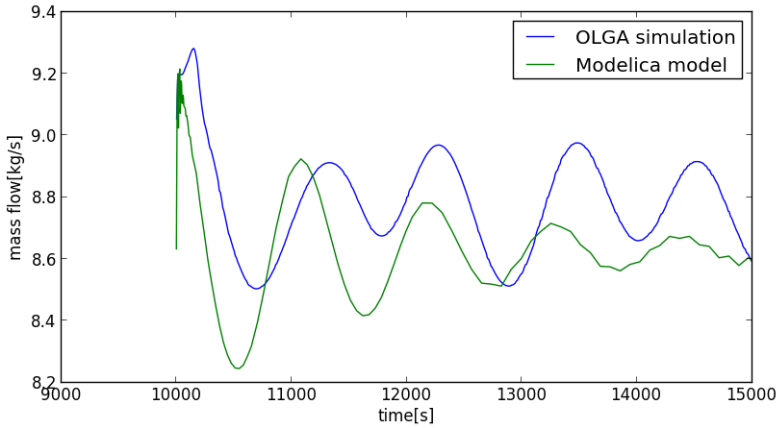

Figure 5.21: Step response of the mass flow out of the outlet.

The parameters used in the Modelica flowline model when performing these simulations can be found in the following table.

Table 5.4: Simulation parameters Modelica flowline model

Parameter description	Symbol	Value	Unit
Liquid density	ρ_l	832.2	$[\mathrm{kg/m^3}]$
Water density	ρ_w	1000	$[\mathrm{kg/m^3}]$
Feed pipe inclination	θ	1	$[^\circ]$
Radius of pipeline	r_1	0.06	[m]
Radius of riser	r_2	0.05	[m]
Length of upstream pipe	L_1	4300	[m]
Length of riser	L_2	300	[m]
Length of horizontal top section	L_3	100	[m]
Pipeline temperature	T_p	335	[K]
Riser temperature	T_r	298.3	[K]
Molar mass of gas	M_g	0.023	[kg/kmol]
Separator pressure	p_0	$50.1{\cdot}10^{5}$	[Pa]
Dynamic viscosity	μ	$1.426{\cdot}10^{-4}$	[Pa s]
Roughness of pipe	ε	$2.8{\cdot}10^{-5}$	[m]
Mass flow gas in	$\omega_{g,in}$	0.36	[kg/s]
Mass flow liquid in	$\omega_{l,in}$	8.64	[kg/s]
Tuning parameter	K_h	0.7400	[-]
Tuning parameter	K_g	0.0547	[-]
Tuning parameter	K_o	0.1232	[-]
Tuning parameter	K_{pc}	0.0112	[-]

5.2.2 State Estimation

After fitting the flowline models, the next step in order to create feedback control is to implement state estimation. This is done based on the work in [1], where the following measurements were assumed available in the flowline; the pressure at the inlet of the flowline p_{in}, the pressure at the low point of the flowline p_{lp}, the pressure at the top of the riser $p_{r,t}$ and the mass flow of oil, gas, and water out of the flowline. This is illustrated in Figure 5.22. It should be noted that the pipeline is not strictly horizontal in this case, as explained in section 4.2 and 4.4.

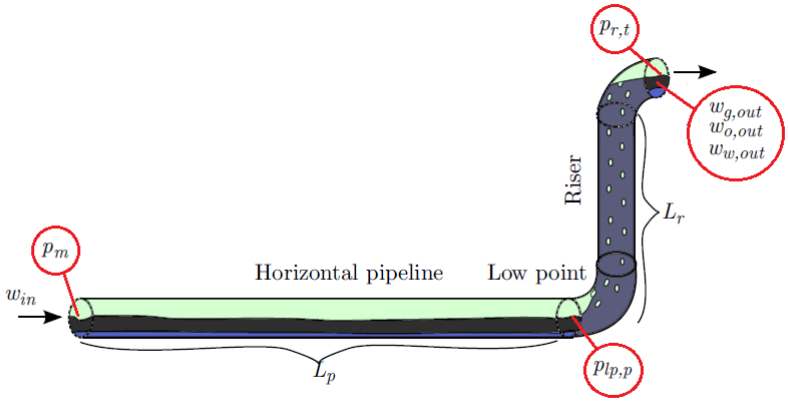

Figure 5.22: Illustration of the available measurements in the flowline.

Based on the available measurements shown in the figure above, both a Unscented Kalman Filter and an Extended Kalman Filter were implemented. The Extended Kalman Filter proved to be more robust, at the Unscented Kalman Filter tended to enter infeasible regions of the flowline models for state values close to zero.

In the following figures the Extended Kalman Filter has been applied to the OLGA model using the Modelica model for predictions. The comparisons between the measured value in OLGA and the estimated value from the Extended Kalman Filter are shown in the figures below. This is done for the pressure at the low point in the riser p_{lp}, the pressure at the top of the riser p_{tp} and the mass flow of liquid at the outlet of the flowline $\omega_{l,out}$.

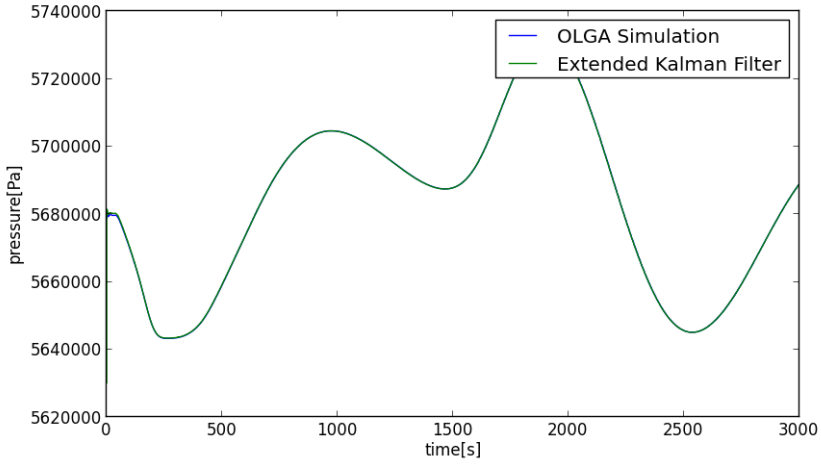

Figure 5.23: Comparison between the pressure at the top of the riser in the OLGA simulation and the estimated value from the EKF.

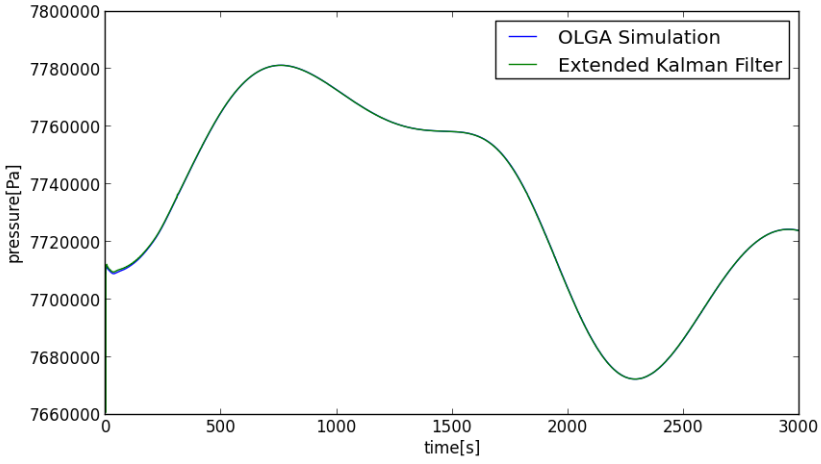

Figure 5.24: Comparison between the pressure at the bottom of the riser in the OLGA simulation and the estimated value from the EKF

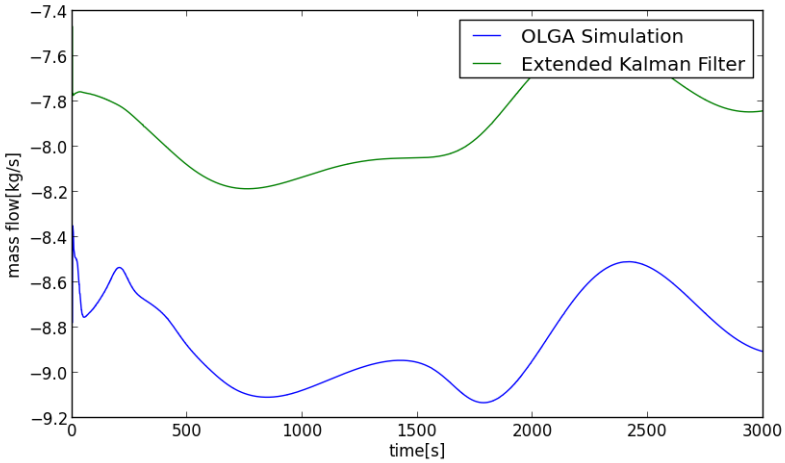

Figure 5.25: Comparison between the mass flow of liquid at the outlet of the flowline in the OLGA simulation and the estimated value from the EKF.

5.2.3 Development of NMPC for the flowline model

In the previous sections in this chapter, the results from model fitting and state estimation are shown. These components form an important part of the Output feedback NMPC-controller, as can be seen in Chapter 2, from Figure 2.2. In this section, the results achieved using this as a basis for optimization is presented. First, the results from steady state optimization on the system will be presented, before open loop dynamic optimization is tested. At last the findings from implementing and testing the NMPC are shown.

The steady state optimization problem given in (4.15) was solved using the values from Table 5.4 as system parameters. This gave the solution $u = 0.397$, $m_{gp} = 1172.32$ kg, $m_{gr} = 54.3748$ kg and $m_{lr} = 1770.59$ kg, which is used as a reference point in the dynamic optimization problem.

As mentioned above, the steady state solution was used as a reference point when the dynamic optimization problem (4.16) was solved. In order to assess the performance of the open loop solution, the Model was run with valve opening $z = 0.02$ until it reached steady state. Then an optimal control solution of (4.16) was found, and applied to the model. The following plots show the resulting behaviour when the optimal control solution is applied to the Modelica model, with a time horizon of 20000 seconds.

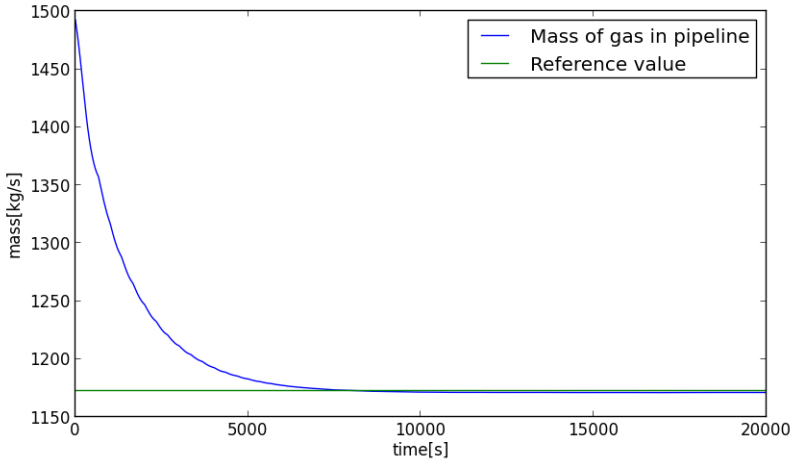

Figure 5.26: The figure shows how the mass of gas in the pipeline compared to the set reference value.

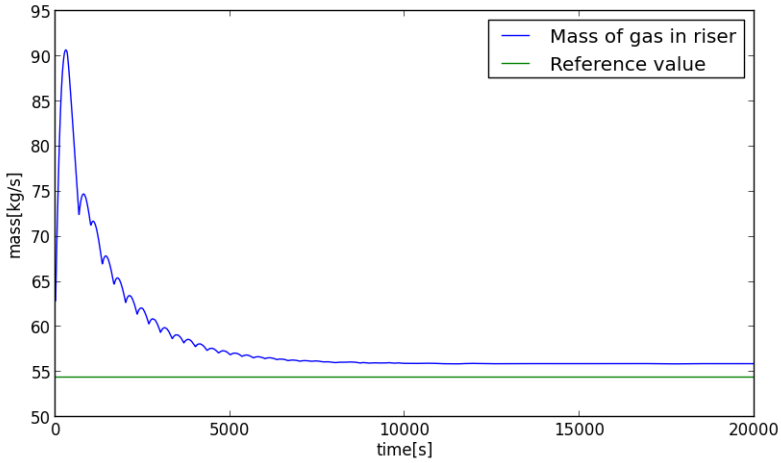

Figure 5.27: The figure shows how the mass of gas in the riser compared to the set reference value.

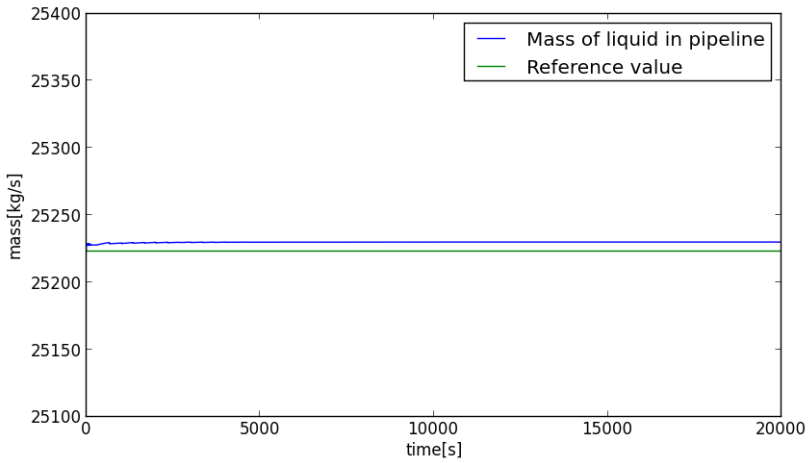

Figure 5.28: The figure shows how the mass of liquid in the pipeline compared to the set reference value.

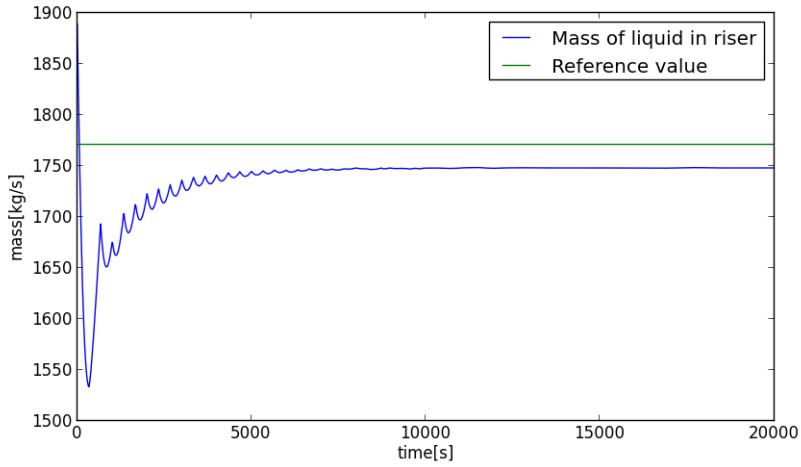

Figure 5.29: The figure shows how the mass of liquid in the riser compared to the set reference value.

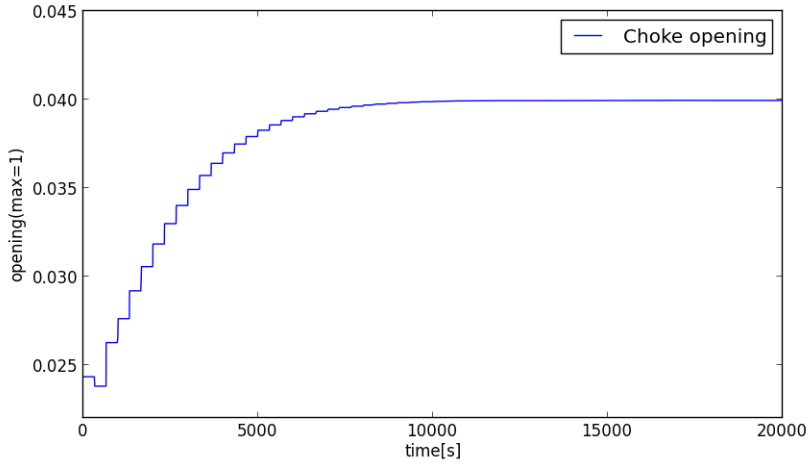

Figure 5.30: The figure shows the solution of the dynamic optimization problem that was applied to the system.

Further, the OLGA flowline model is simulated with the same input $z = 0.02$ until steady state, before the same control is applied to this system. The resulting system behaviour is shown in the figures below.

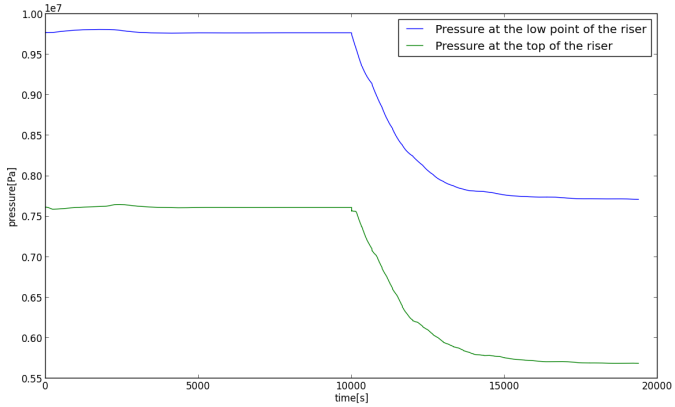

Figure 5.31: The figure shows the pressures in the OLGA model, and how they change when the open loop optimal control solution is applied after 10000 seconds.

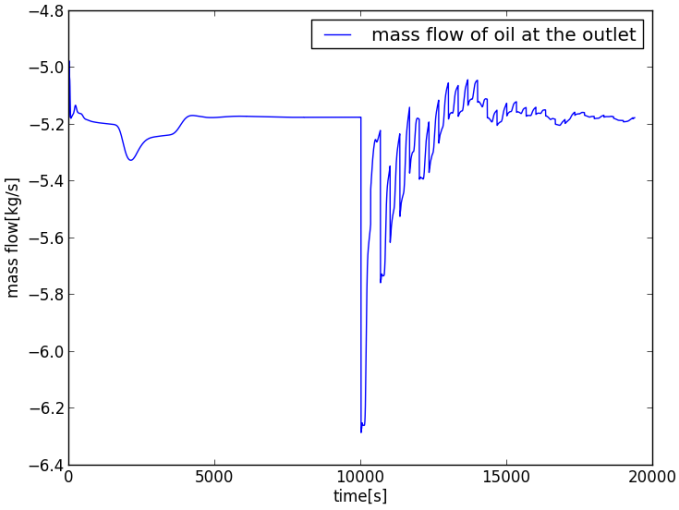

Figure 5.32: The figure shows the flow of oil out of the OLGA model, and how it changes when the open loop optimal control solution is applied after 10000 seconds.

After testing the open loop control solutions for the system, the next step is to close the loop, using the state estimator in order to implement an Output Feedback NMPC as illustrated in Section 2.2.4. After many attempts, this was found to be infeasible during the time horizon of this master thesis, due to large difficulties when solving dynamic optimization problems in JModelica.org. The circumstances revolving this issue will be discussed in the next chapter.

Chapter 6

Discussion of results and implementation

6.1 Well

6.1.1 Behaviour of the OLGA well model

The first graphical presentation in the previous chapter was plots of the OLGA well model behaviour at two different operating points, with $u_{gl} = 0.5$ kg/s and $u_{gl} = 0.1$ kg/s, for for both pressure and flow. In Figure 5.1 and 5.3, the pressures and the flows are shown for $u_{gl} = 0.5$ kg/s. From these figures it is clear that the system reaches a stable operating point.

Further, Figure 5.2 and 5.4, show the same parameters for another operating point where $u_{gl} = 0.1$ kg/s, which means that there is significantly less lift-gas injected into the well. This value was chosen for u_{gl}, as it clearly shows how the system reach an unstable operating point, where the casing-heading phenomenon described in Section 4.1 is present. The presence of this effect is common in gas-lift wells, and supports the validity of the OLGA model.

6.1.2 Fitting the OLGA well model with the Modelica well model

As the results in the previous section gave reason to assume validity of the OLGA model, the Modelica model presented in 4.1, was compared and adapted to the OLGA model. A natural starting point when fitting the models, was to look at a stable operating point, which was found for $u_{gl} = 0.5$ kg/s in the previous section. The comparison of the two models after attempted fitting is illustrated for pressures and flows in Figure 5.5 and 5.6. The comparison was done by simulating the OLGA well model to steady state for u_{gl} providing the steady state parameters from OLGA as a starting point for the Modelica model, and comparing the difference between the resulting steady state parameters for the two models.

As the Modelica model was fed with the steady state parameters from OLGA, one would expect the Modelica parameters to stay constant if the models were tuned perfectly, but from the figures it can then be seen that the Modelica well model variables move away from the starting points, before they reach a significantly different steady state value than the OLGA well model.

In Section 5.1.1 it was described how the model equations were changed, in order to fit the Modelica well model better to the OLGA well model. However, these changes did not affect the assumptions the model was based on, and the pressures are still calculated with no regard to friction. As model changes had to be done in order to reach similar pressures in the tubing for both models, it seems clear that the pressure calculations in the Modelica model are too inaccurate. This is not unexpected, as the Modelica model does not consider friction when modelling the pressure and flow in the well and introduction of friction in the model could very well cause the same pressure difference, as the one constructed by altering the model in this thesis.

6.2 Flowline

6.2.1 Behaviour and fitting of the OLGA flowline model with the Modelica model

In the simulation chapter under section 5.2.1, the results from fitting the flowline model in Modelica to the flowline model in OLGA was presented. The plots that were given in this section gives information about the models behaviour.

The first three figures shows three key parameters for the Modelica model compared to the OLGA model for a choke opening of 5 %. From these we see that both the amplitudes and the frequencies seems to correspond very good. The next three figures shows the same three parameters, but for a choke opening of 4 %. From these figures it becomes clear that there is a small deviation between the steady state values from the OLGA flowline model and the Modelica flowline model, but that they still are quite similar. Thereafter the choke opening is altered to 6 % and again the same parameters for the models are compared. From these figures the amplitudes still seem to correspond well, but the deviation in frequency has increased. While the previous figures show the behaviour of the system for choke opening values in the same area as where the model was tuned, the three plots in Figure 5.16-5.18 shows the behaviour when the choke opening is set to 30%. From these figures one can see that the amplitudes still match very well, but that the deviation in frequency now is significant. Based on these observations it becomes clear that the Modelica model is very well fitted to the OLGA model, and that it captures all the relevant dynamics of the system, but that the frequency dependence becomes poorer when the choke opening is far away from the value at which it was tuned.

As mentioned in the simulation chapter the system step response where also studied. This is shown in Figure 5.19-5.21. From these figures one can see that step responses are very similar, which supports the claim that the models are very well fitted.

6.2.2 State estimation

In Section 5.2.2 in the Simulations chapter, the output of the Extended Kalman Filter and the values of the simulated system in OLGA are compared. It can be seen from the figures 5.23-5.25 that the EKF tracks the measured pressures perfectly, but that there is a deviation between the estimated mass flow out of the system and the actual mass flow out of the system. This indicates either that the tuning of the EKF is not optimal, or that the number of measurements are insufficient in order to get good estimations from the system. Based on the results in the previous section, the models seems to be fitted well enough to allow for successful state estimation, especially around the operating point $z = 0.05$.

6.2.3 Development of NMPC for the flowline model

When developing an NMPC for the flowline model, many problems arose in association with different parts of the JModelica.org platform. Locating and handling these problems proved to be very time consuming. One of the reasons for this, is that JModelica.org still is in an early state in the development process both when it comes to robustness and when it comes to interaction with the user in form of error messages.

The first problem was encountered when importing the models from Modelica and Optimica into the JModelica.org framework, as this lead to a series of errors which proved hard to track. The main reason for this, is that the JModelica.org platform consist of many different tools which earlier was mentioned to be an advantage, but that also makes it very hard to locate the error. After spending a substantial amount of time debugging this, the cause of the errors where found to be that JModelica.org does not support importing user defined functions from Modelica, and also that it does not support certain mathematical functions such as log_{10}. This was solved by hard coding all user defined functions used in the Modelica model, replacing $log_{10}(x)$ with $\frac{log_2(x)}{log_2(10)}$ and inserting approximations for all maximization functions in the model, using the approximation presented in Section 4.5. In order to reproduce the behaviour of the models when simulating them from JModelica.org and from Modelica, the parameter ϵ defined in (4.13) had to be set no larger than $\epsilon = 0.01$.

Based on the conclusions made in [3], single shooting was chosen as the optimal control method, and this was implemented in JModelica.org. This optimal control method proved not to be very robust at all. Numerous error messages concerning problems evaluating the gradients of the objective function and the constraints were received. Some of these problems were traced back to the approximation of the max functions, as the integrators used in JModelica.org were sometimes unable to evaluate the values of part of the gradient, where these expressions were present.

In search of a solution to the problem explained above, a thorough review of the implementation of dynamic optimization using JModelica.org in [3] was preformed.

From this review it was found that the same approximation to the maximization functions, as used in this thesis was applied, but with $\epsilon = \sqrt{0.1} \approx 0.316$. As explained earlier, ϵ could not be set higher than 0.01 in order to simulate the behaviour of the flowline model in this thesis, but it was possible in [3], as a simpler flowline model was used.

As a result of the problems explained above, the testing of methods for NMPC and the comparison of these with alternative optimization based strategies has been severely limited.

When testing the developed dynamic optimization algorithm for various objective functions and constraints, it was found that the errors from evaluating the gradients were less frequent when the objective function only consisted of differential states and inputs. Therefore, the algebraic variables of the flowline system were not included in the objective function. As the goal of the optimization often is to maximize the oil flow, and as ω_{out} is a algebraic variable, this is disadvantageous. In order to work around this, a static optimization problem maximizing liquid flow out of the flowline was solved as explained in the beginning of Section 5.2.3.

Figure 5.26 - 5.30 show the behaviour of the Modelica flowline system when applying open-loop optimization in order to get to the optimal steady state value. From these figures we can see that the state variables converge to the reference value, without any notable oscillations or instabilities. In Figure 5.31 and 5.32, it can be seen that this is also the case when the same open-loop optimization is applied to the OLGA flowline system. Based on these figures, it seems that the flowline Modelica model is applicable for controlling the OLGA flowline model, which was expected as the behaviour of the two models was shown to be very similar in Section 5.2.1.

6.3 Further Work

With the implementation problems described in the previous section fresh in mind, it becomes obvious that a thorough analysis of the errors in JModelica.org must be conducted. This would be the natural starting point when continuing the work presented in this thesis. Due to the severity of the implementation issues found in this thesis, there will also be a need to make a further assessment of whether JModelica.org is a suitable platform for implementation of NMPC.

In this thesis it is also found that it might be valuable to develop the well model in Modelica so that it takes regard to friction.

Another point for further work will be to connect and analyze the well and flowline models in OLGA in order to gain a better understanding of how these models interacts.

If the problems in JModelica.org are assessed and handled, a complete implementation of NMPC can be tested on the complete system. Further, this can be compared to other optimization based strategies such as steady state optimization.

Chapter 7

Conclusion

In this thesis models for wells and flowlines has been instantiated in OLGA, and simple dynamic models inspired by [3] and [9] have been fit to these OLGA models. It was found that the simple dynamic well model did not give a good representation of the more complex model in OLGA, and the lack of regard to friction in the tubing was pointed out at the main reason behind this. For the simple dynamic flowline model it was found that is can be tuned to fit the more complex OLGA model very well for a set operating point, but that it will deviate when moving far away from this point.

Further state estimation using an Extended Kalman Filter was performed on the flowline model, based on measurements proposed in [1]. Here it was found that the state estimates are fairly accurate, but some deviations are present, therefore the addition of more measurements to the system is recommended.

Based on the fitting of models and implementation of state estimation for the flowline system, methods for NMPC was attempted implemented using the JModelica.org environment. This implementation was troubled with many bugs, and lack off robustness in the interface between the various implementation tools used is assumed to play a large role here.

Bibliography

[1] T. Lund. State estimation for an oil production nework based on gas-lift, Project report. 2013.

[2] B. Foss and T. A. N. Heirung. Merging Optimization and Control, Norwegian University of Science and Technology. 2013.

[3] K. Nalum. Modeling and Dynamic Optimization in Oil Production, Master Thesis, Norwegian University of Science and Technology. 2013.

[4] R. Findeisen and F. Allgöwer. An introduction to Nonlinear Model Predictive Control. *21st Benelux Meeting on Systems and Control*, pages 1–23, 2002.

[5] J-D. Jansen, O.H. Bosgra, and P.M.J. van den Hof. *Model-based control of multiphase flow in subsurface oil reservoirs*. Journal of Process Control, 18(9), 2008.

[6] B. Foss. *Process control in the upstream petroleum industries*. 2011.

[7] C. E. Garca, D. M. Pretti, and M. Morari. Model Predictive Control: Theory and Practice - a Survey. *Automatica*, 25(3):335–348, 1989.

[8] J. Åkesson, K.-E. Årzn, T. Bergdahl, and H. Tummesheit. Modeling and optimization with Optimica and JModelica.org - Languages and tools for solving large-scale dynamic optimzation problems. *Computers & Chemical Engineering*, 34(11):1737–1749, 2010.

[9] E. Jahanshahi. Control Solutions for Multiphase Flow Linear: Linear and nonlinear approches to anti-slug control, Ph.d Thesis. Norwegian University of Science and Technology. 2013.

[10] L. T. Biegler. *Nonlinear Programming, Concepts, Algorithms and Applications to Chemical Processes*. Mathematical Optimization Society and the Society for Industrial and Applied Mathematics, 2010.

[11] G. O Eikrem, O. M Aamo, and B. A. Foss. On Instability in Gas Lift Wells and Schemes for Stabilization by Automatic Control. *Society of Petroleum Enigneers*, 23(2):268–279, 2008.

[12] J. V. Vogel. Inflow Performance Relationships for Solution-Gas Drive Wells. *Society of Petroleum Engineers*, 20(1):83–92, 1968.

[13] J. Jahanshahi and S. Skogestad. Simplified Dynamical Models for Control of Severe Slugging in Multiphase Risers. *Prepints of the 18th IFAC World Congress, Milano (Italy) August 28. -September 2.*, 2011.

Appendix A

Flowline

A.1 Pipeline-Riser model

The average liquid volume fraction in the pipeline is estimated based on the the approximation that the liquid volume fraction at the inlet is equal to the liquid volume distribution in the pipeline, and is given in the following equation

$$\bar{\alpha}_{lp} \cong \frac{\bar{\rho}_{gp}\omega_{l,in}}{\bar{\rho}_{g1}\omega_{l,in} + \rho_l\omega_{g,in}}. \tag{A.1}$$

The density of the gas in the pipeline $\bar{\rho}_{gp}$ is given by the mass of gas in the pipeline m_{gp} and the volume available for the gas $V_p - m_{lp}/\rho_l$.

$$\bar{\rho}_{gp} = \frac{m_{gp}}{V_p - m_{lp}/\rho_l} \tag{A.2}$$

$$\dot{m}_{gp} = \omega_{g,in} - \omega_{g,lp} \tag{A.3a}$$
$$\dot{m}_{lp} = \omega_{l,in} - \omega_{l,lp} \tag{A.3b}$$
$$\dot{m}_{gr} = \omega_{g,lp} - \omega_{g,out} \tag{A.3c}$$
$$\dot{m}_{lr} = \omega_{l,lp} - \omega_{l,out} \tag{A.3d}$$

$$\bar{\alpha}_{l1} \cong \frac{\bar{\rho}_{g1}\omega_{l,in}}{\bar{\rho}_{g1}\omega_{l,in} + \rho_l\omega_{g,in}} \tag{A.4}$$

$$\bar{\rho}_{g1} = \frac{m_{gp}}{V_1 - m_{lp}/\rho_l} \tag{A.5}$$

$$\omega_{mix,out} = K_{pc} f(z) \sqrt{\rho_t (p_2 - p_0)} \tag{A.6}$$

$$\omega_{l,out} = \alpha_{lm,t} \omega_{mix,out} \tag{A.7}$$

$$\omega_{g,out} = (1 - \alpha_{lm,t}) \omega_{mix,out} \tag{A.8}$$

$$\bar{h}_1 = K_h h_c \bar{\alpha}_{l1} \tag{A.9}$$

$$h_1 = \bar{h}_1 + \left(\frac{m_{l1} - \rho_l V_1 \bar{\alpha}_{l1}}{\pi r_1^2 (1 - \bar{\alpha}_{l1}) \rho_l} \right) \sin(\theta) \tag{A.10}$$

$$V_{g1} = V_1 - m_{l1}/rho_l \tag{A.11}$$

$$\rho_{g1} = \frac{m_{g1}}{V_{g1}} \tag{A.12}$$

$$p_1 = \frac{\rho_{g1} R T_1}{Mg} \tag{A.13}$$

$$\Delta p_{fp} = \frac{\bar{\alpha}_{l1} \lambda_p \rho_l \bar{U}_{sl,in}^2 l_1}{4 r_1} \tag{A.14}$$

$$\lambda_p = 0.0056 + 0.5 Re_p^{-0.32} \tag{A.15}$$

$$\bar{U}_{sl,in} = \frac{\omega_{l,in}}{\pi r_1^2 \rho_l} \tag{A.16}$$

A.1.1 Riser Model

$$V_2 = \pi r_2^2 (l_2 + l_3) \tag{A.17}$$

$$V_{g2} = V_2 - m_{l2}/\rho_l \tag{A.18}$$

$$\rho_{g2} = \frac{m_{g2}}{V_{g2}} \tag{A.19}$$

$$p_2 = \frac{\rho_{g2} R T_2}{M_g} \tag{A.20}$$

$$\bar{\alpha}_{l2} = \frac{m_{l2}}{V_2 \rho_l} \tag{A.21}$$

$$\bar{\rho}_m = \frac{m_{g2} + m_{l2}}{V_2} \tag{A.22}$$

$$\Delta P_{fr} = \frac{\bar{\alpha}_{l2}\lambda_r \bar{\rho}_m \bar{U}_m^2 (l_2 + l_3)}{4r_2} \tag{A.23}$$

$$\lambda_r = 0.0056 + 0.5 Re_r^{-0.032} \tag{A.24}$$

$$Re_r = \frac{2\bar{\rho}_m \bar{U}_m r_2}{\mu} \tag{A.25}$$

$$\bar{U}_m = \bar{U}_{sl2} + \bar{U}_{sg2} \tag{A.26}$$

$$\bar{U}_{sl2} = \frac{\omega_{l,in}}{\rho_l \pi r_2^2} \tag{A.27}$$

$$\bar{U}_{sg2} = \frac{\omega_{g,in}}{\rho_{g2} \pi r_2^2} \tag{A.28}$$

A.1.2 Gas Flow model at the low-point

$$\omega_{g,lp} = 0, \;\; h_1 \geq h_c \tag{A.29}$$

$$\omega_{g,lp} = K_g A_g \sqrt{\rho_{g1} \Delta p_g}, hspace5pth_1 < h_c \tag{A.30}$$

$$\Delta p_g = p_1 - \Delta p_{fp} - p_2 - \bar{\rho}_m g l_2 - \Delta p_{fr}. \tag{A.31}$$

$$\omega_{l,lp} = K_l A_l \sqrt{\rho_l \Delta p_l} \tag{A.32}$$

$$\Delta p_l = p_1 - \Delta pfp + \rho_l g h_1 - p_2 - \bar{\rho}_m g l_2 - \Delta p_{fr} \tag{A.33}$$

$$A_g \cong \pi r_1^2 \left(\frac{h_c - h_1}{h_c} \right)^2, \;\; h_1 < h_c \tag{A.34}$$

$$A_l \cong \pi r_1^2 - A_g \tag{A.35}$$

$$\alpha_{lm,t} = \frac{\alpha_{l,t}\rho_l}{\alpha_{l,t}\rho_l + (1 - \alpha_{l,t}\rho_{g2})} \tag{A.36}$$

$$\rho_t = \alpha_{l,t}\rho_l + (1 - \alpha_{l,t})\rho_{g2} \tag{A.37}$$

$$\bar{\alpha}_{l2} = \frac{\alpha_{l,lp} + \alpha_{l,t}}{2} \tag{A.38}$$

$$\alpha_{l,lp} = \frac{A_l}{\pi r_1^2} \tag{A.39}$$

$$\alpha_{l,t} = 2\bar{\alpha}_{l2} - \alpha_{l,lp} = \frac{2m_{l2}}{V_2 \rho_l} - \frac{A_l}{\pi r_1^2} \tag{A.40}$$