NTNU – Trondheim
Norwegian University of
Science and Technology

Anti-Slug Control with Non-Linear State Estimation

Terese Vardenær Syre

Master of Science in Engineering Cybernetics

Supervisor: Morten Hovd, ITK
Co-supervisor: Esmaeil Jahanshahi, IKP
Sigurd Skogestad, IKP

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Abstract

In offshore production, the two-phase mixture of oil and gas is transported from the seabed oil wells to the surface facilities by pipelines and risers. The two-phase flow can have different flow regimes, where severe slugging is one undesirable flow regime and an effective solution is needed to prevent it. The recommended solution is active control of the top-side choke valve.

Previously, controllability analysis is done of two-phase flow in a 4-state pipeline-riser model. This analyze concludes that the best way to control the choke valve is by using the subsea pressure measurement combined with topside flow measurement. However, the subsea pressure might be difficult to measure correctly because the pipeline is placed under tough conditions, hundredth or even thousands of meters under sea level. One possibility is to combine topside pressure with topside flow measurement and use for estimation of states or other sub-sea measurements that are normally not available.

Simulation studies are done in MATLAB of different anti-slug control solutions. Linear Kalman filter, extended Kalman filter (EKF) and unscented Kalman filter (UKF) are used for state estimation and combined with controllers such as PI, LQR and MPC. The input to the system is flow rate of gas and liquid, and the nominal choke opening. The input disturbance to the process is change in the flow rate of gas and liquid imitating slug flow.

As expected, when only topside measurements are used, because of the highly nonlinear system dynamics the linear Kalman filter fails in stabilizing the system. The EKF works good when the system has low input disturbance, while the UKF is the best nonlinear filter when the system has high input disturbance. However, when the nominal choke opening is increased, the UKF combined with a controller fails. The LQR controller combined with UKF shows slightly better results than the PI and MPC controller combined with the same filter for state estimation. There is also potential in using the high-gain observer in control strategies.

Sammendrag

I offshore produksjon blir miks av olje og gass transportert på havbunnen til produksjonsenheter på havoverflaten gjennom rørledninger. Denne miksen av olje og gass kan ha forskjellige strømningsmåter, hvor "slugging" er en uønsket strømningsmåte. Mye tid og krefter er brukt for å forhindre slik strømning siden den kan gjøre store skader på produksjonsenhetene. Den anbefalte løsningen er kontroll av ventilen inn til produksjonsenheten.

Tidligere har det blitt ut ført kontrollerbarhetsanalyse av en modell basert på 4 tilstander av tofase strømning i en L-formet rørledning fra havbunnen til vannoverflaten. Denne analysen konkluderer med at den enkleste måten å unngå slugging er å måle trykket i rørledningen på havbunnen kombinert med strømningen ut av ventilen. Men det kan være vanskelig å få gode målinger ved havbunnen, rørledningene kan være plassert hundre- eller kanskje tusenvis av meter under havoverflaten. En løsning er å kombinere målinger i rørledningen ved havoverflaten til å estimere trykket ved bunn av rørledningen å bruke denne estimerte verdien i reguleringsløsningen.

Simuleringer av forskjellige reguleringsløsninger er gjort i MATLAB. Lineært Kalman filter, "extended Kalman filter" (EKF) og "unscented Kalman filter" (UKF) er brukt til tilstandsestimering og kombinert med kontrollere som PI, LQR og MPC. Inputen til systemet er strømningsrate for gass og væske, og den nominelle ventilåpningen. Inputforstyrrelsen til systemet er variasjon i strømningsraten for gass og væske som etterligner slugging.

Som forventet, på grunn av ulinearitetene i systemet, blir ikke lineært Kalman filter tilstrekkelig når målinger fra havoverflaten blir brukt. EKF virker best lokalt, altså når det er liten inputforstyrrelse, mens UKF er det beste ulineære filteret når systemet får økt inputforstyrrelse. Men, når den nominelle ventilåpningen blir økt klarer ikke UKF kombinert med en regulator å stabilisere systemet. LQR kombinert med UKF viser litt bedre resultater enn LQR kombinert med PI eller MPC. Videre er det også potensiale i å bruke "high-gain observer" i reguleringsløsningen for anti-slug kontroll.

Problem Formulation

Master Project Proposals on Anti-Slug Multiphase Flow Control.

"In offshore production, the two-phase mixture of oil and gas is transported from the seabed oil wells to the surface facilities by pipelines and risers. Two-phase flow can be in different flow regimes; severe slugging is one flow regime which is undesirable in offshore production and an effective solution is needed to prevent it. Active control of the top-side choke valve is the recommended solution. Different control strategies can be used for stabilization of this system. Finding a simple and robust solution is motivation for this research."

Anti-slug Control With Non-Linear State Estimation

"Based on the results from the controllability analysis it is known that using top-side measurements is very difficult. This is because of RHP-zeros in dynamic of the top pressure. However, one possibility is to combine top pressure with one flow measurement and use them for estimation of states or other sub-sea measurements that are not normally available. Because of highly nonlinear nature of the system, linear estimation techniques such as Kalman Filter will fail in more real conditions. Nonlinear estimation such as Unscented Kalman Filter can be tested in conditions that Kalman Filter fails. Having the states estimated, the optimal stated feedback is one immediate control solution. However, it is known that if choke valve saturates on fully-open or fully-closed, the control system fails. It can be taken care of by considering a constraint on valve opening which leads to use of MPC.

- Studying about Kalman filter (KF), extended Kalman filter (EKF) and unscented Kalman filter (UKF).
- Becoming familiar with simplified models for severe-slugging.
- Using KF, UKF or EKF with PI or LQR for control of nonlinear model with only topside pressure measurements for the estimator.
- Using KF, UKF or EKF with PI or LQR for control of nonlinear model with subsea pressure measurements for the estimator.
- Using EKF+MPC and UKF+MPC (constraint on valve opening ($0<U<1$)
- Evaluating performance of different approaches."

Forord

Denne masteroppgaven er et resultat i forbindelse med avslutningen av mastergraden min i teknisk kybernetikk ved Norges teknisk-naturvitenskaplige universitet (NTNU), skrevet våren 2012.

Jeg vil takke Professor Morten Hovd for å akseptere oppgaven min. Videre vil jeg gi en takk til Professor Sigurd Skogestad for å ta meg godt i mot i prosesskontrollgruppen, og for gode diskusjoner rundt oppgaven. Ph. D student Esmaeil Jahanshahi har vært en veldig god veileder gjennom hele oppgaven og jeg vil takke for godt samarbeid. Han har tatt seg god tid, vist engasjement om mitt arbeid og kommet med nyttige innvendinger underveis.

Jeg vil takke Marie og Martin for korrekturlesing av oppgaven. En ekstra takk vil jeg også rette til Martin som brukte flere fine sommerdager til å hjelpe meg gjennom en vanskelig eksamen i første klasse. Det satt jeg stor pris på!

Min snille samboer, Vegard fortjener også en takk. Vi har begge jobbet godt med hver vår masteroppgave helt fra januar og sammen har vi holdt arbeidsmoralen på topp. Han har stilt spørsmål som har fått meg til å tenke i riktige baner når jeg har støtt på utfordringer i oppgaven, og tålmodig hørt på mine tanker og spørsmål.

Sist, men ikke minst, vil jeg takke mine studievenner for hyggelige lunchpauser under masterskrivingen og generelt 5 flotte studieår!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In an offshore oilfield, pipelines and risers transfer multiphase mixture of oil, gas and water from oil wells at seabed to the surface processing facilities. A more and more upcoming trend is connecting subsea wells to either remote wellhead platforms or directly to on-shore processing plants. As a result, more and longer multiphase pipelines are used. The development of slugs of liquid in multiphase pipelines is a major, and expensive, challenge for oil producers. Slugging is varying or irregular flows of gas and liquids in pipelines. A way to prevent slugging is by reducing the opening of the top-side choke valve. However, this conventional solution increases the back pressure of the valve and reduces the production rate from the oil wells.

The goal of this master thesis is to make a control system to stabilize flow and avoid slugs in the pipeline by adjusting the position of the choke valve. Here, the 4 state model developed by Esmaeil Jahanshahi and Sigurd Skogestad is used [2]. This model is for an L-shaped pipeline-riser, where the 4 states correspond to the mass of the gas and the liquid, in the pipeline and the riser. The best way to decide the choke valve position is by knowing the pressure in the pipeline. But, the pipeline might be hundreds or even thousands of meters under sea level where there are tough conditions for pressure measurements. Therefore, an observer will be used to estimate the pressure in the pipeline, and this estimate will be used in a controller to find the optimal choke position.

Previously, controllability analysis of the pipeline-riser model is performed[3][4]. The conclusion of the work is that a filter, based on the pressure at the top of the riser combined with either the volume flow rate or the mass flow, gives the best estimate value of the pressure in the pipeline.

Chapter 2

Background

2.1 Background

Multiphase flow and its behavior has been a concern in offshore oil and gas industry, since the beginning. One common undesired flow variation is *slug flow*, in which the liquid flows intermittently along the pipes in concentrated mass, called a *slug*. Slugging has been recognized as a serious problem and much effort has been put into prevention of this problem. The unstable behavior of slug flow has a negative impact on the production facilities. The worst case scenario is when severe slugging causes platform trips and plant shutdowns.

This section is an introduction to two-phase flow and riser slugging. As there are several different types of slug flows, these will be explained. The goal of this project is to avoid slugging by the use of a controller that stabilizes the flow. The controllability analysis [3] concludes that measurement of the pressure in the pipeline (P_1) give good results in anti-slug control. However, the pressure deep below sea level is difficult to measure, so this is estimated used an observer.

2.1.1 Multiphase Flow

Multiphase flow refers to any fluid flow consisting of more than one component, and is an important topic in oil and gas industry. Multiphase pipelines connecting remote wellhead platforms to subsea wells are a common feature. In the future, long-distance tie-back pipelines connecting subsea processing

units to on-shore processing plants will be more common. Development of such pipelines are turning the spotlight on one of the biggest challenges involved in operating offshore processing facilities and subsea separation units: control of the disturbances in the feed to the separation process [5]. Since a relatively small change in operating conditions changes the behavior of the flow in pipelines drastically, a lot of time and effort has been spent studying multiphase flow.

2.1.2 Slug Flow

Multiphase flow in pipelines frequently involves the formation of slugs. Slugs are plugs of liquid or gas that travel through the pipeline. They are unwanted because they can create significant pressure fluctuations. The slugs can be formed due to transient effects related to pigging, start-up, blow-down and changes in pressure or flow rates. Slug flow can occur on different time- and length scales depending on the slug formation. An example of slug can be liquid plugs that are accumulated at the bottom of the riser until sufficient pressure is generated behind it to push the liquids over the top of the riser.

The unstable behavior of slug flow in multiphase pipelines has a negative impact on the operation on production facilities. Normal problems are flaring, reduced operating capacity and stress on valves and bends. Separator and compressor trains will also face problems because of the uneven pressure in the riser during slug flow. The worst case scenario is plant shutdown, which is extremely expensive. The different slug flows can be put into different categories:

- *Hydrodynamic slugging* develops in horizontal parts of the pipeline. They are relatively short, typically less than 500 pipe diameters [1].
- *Terrain slugging* is caused by low-points in the pipeline topography. The blockage in the low-point initiates the slug until the pressure in the compressed gas is large enough to overcome the hydrostatic head of the liquid. Figure 2.1 shows how terrain slug is generated.
- *Transient slugging* is caused by increased liquid flow rates at pipeline exit to processing facilities in response to changes in operating conditions.

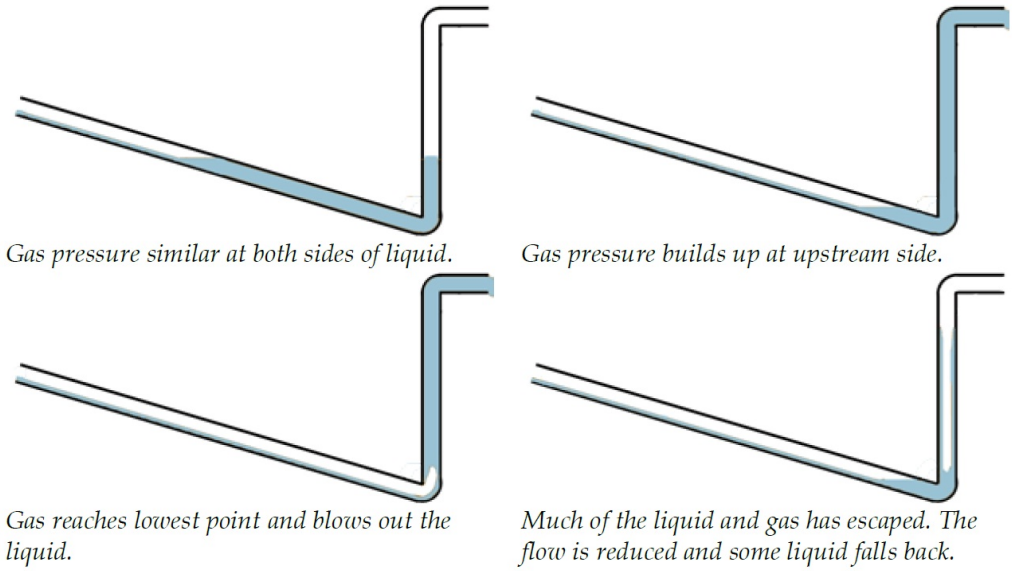

Figure 2.1: Terrain slug generation[1]

2.1.3 Anti-Slug Control

The role of anti-slug control systems is to completely remove slugs by stabilizing an unstable flow regime. Anti-slug control has the following definition [6]:

Definition 1.1 *An anti-slug controller is a controller that stabilizes a desired, non-oscillatory flow regime that exists at the same boundary conditions as riser slugging and thereby avoids the formation of riser slugging in the system.*

Some of the advantages of anti-slug control with the choke valve as actuator are:

- cheaper than implementing new equipment,
- completely removes slug flow, which results in less strain on the system,
- reduces maintenance expenses.

2.2 Previous Work

There are several techniques applied to avoid slugs. One approach is to design the system to avoid slugs altogether. This can be done by changing the pipeline topology, increase the size of the separator, adding a slug catcher or installing gas lift. Another technique is to let the pipeline gas be transferred to the riser at a point above the riser-base, a technique known as self-gas lifting[7]. However, since thise methods involves changing the design, it usually costly.

There have been many approaches to feedback control since the end of 1970's, see chapter on previous work in [6](and its references). One of the first key concepts was to avoid riser slugging by automatically adjusting the topside choke valve position, based on an algorithm with a pressure and flow measurement in the riser as input. About 10 years later, the PI-controller based on upstream pressure measurement, to avoid riser slugging, was used. Both approaches were based on experimental work, and did not result in any reported industrial application.

In 1996, another technique was introduced. This technique was to implement a control system that used the topside choke valve to keep the pressure at the riser base at or above the peak pressure in the riser slug cycle, thus preventing liquid accumulation in the bottom of the riser. This introduces an extra pressure drop in the system due to the high setpoint for the controller. This removes the riser slugging, but it did so by automating the old choking strategy rather than affecting the stability of the flow regimes in the pipeline. So, there is still unstable flow, but the choke is adjusted to avoid slug.

In the last two decades, solutions which have resulted in industrial application have been proposed. The first industrial implementation of anti-slug control is reported by ABB[5], where the new control system is applied on the Hod-Valhall pipeline, and it manages to stabilize an unstable operating point. This operating point, where the flow in the pipeline is steady, exists at the same boundary condition as would normally result in riser slugging. Further, in 2005 engineers from Statoil [8] successfully applied advanced control for the inlet facilities, where active slug control for two long multiphase flow lines is combined with model predictive control (MPC) to handle slugs entering the inlet separators.

Chapter 3

Modelling Pipeline Riser Flow

3.1 The Pipeline Riser Model

This model is based on the mass conservation law for individual phases in the pipeline and the riser[2]. The mass conservation of gas and liquid in the pipeline and riser results in this model:

$$\begin{aligned} \dot{m}_{G1} &= w_{G,in} - w_{G,lp} \\ \dot{m}_{L1} &= w_{L,in} - w_{L,lp} \\ \dot{m}_{G2} &= w_{G,lp} - w_{G,out} \\ \dot{m}_{L2} &= w_{L,lp} - w_{L,out} \end{aligned} \tag{3.1}$$

where lp indicates low-point.

The state variables of the system are as following

- m_{G1}, mass of gas in the pipeline
- m_{L1}, mass of liquid in the pipeline
- m_{G2}, mass of gas in the riser
- m_{L1}, mass of liquid in the riser

The mass flow in the low point $w_{G,lp}$ and $w_{L,lp}$ are described by the rate of flow of liquid through an orifice, called an "orifice equation" [9],

$$\begin{aligned} w_{G,in} &= K_G A_G \sqrt{\rho_{G1} \Delta P_G} \\ w_{L,in} &= K_L A_L \sqrt{\rho_{L1} \Delta P_L} \end{aligned} \tag{3.2}$$

The variables in the model are; pressure drop over the low-point ΔP, the opening area A, a tuning parameter K and the density for the gas and liquid in the pipeline ρ_1 with index G and L for hence gas and liquid.

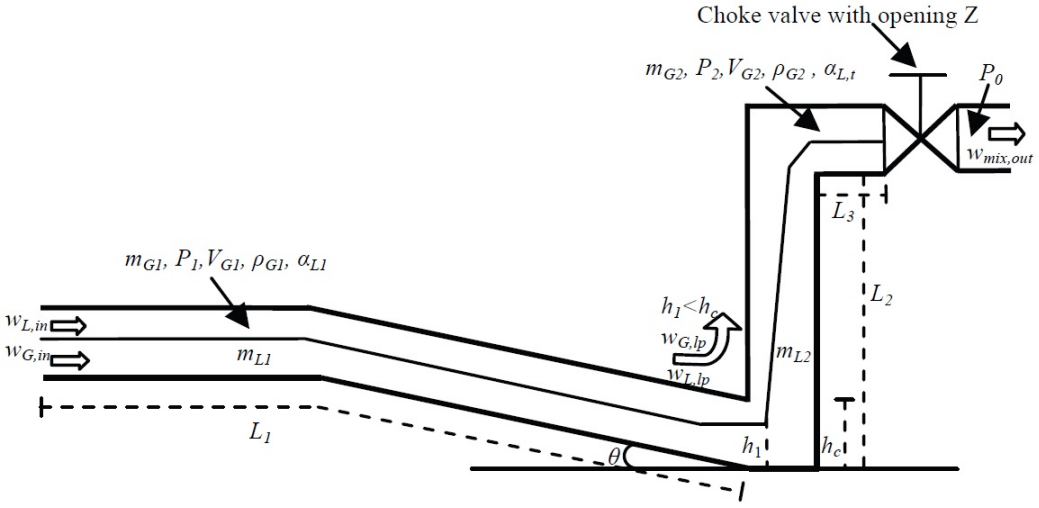

Figure 3.1: Simplified representation of desired flow regime

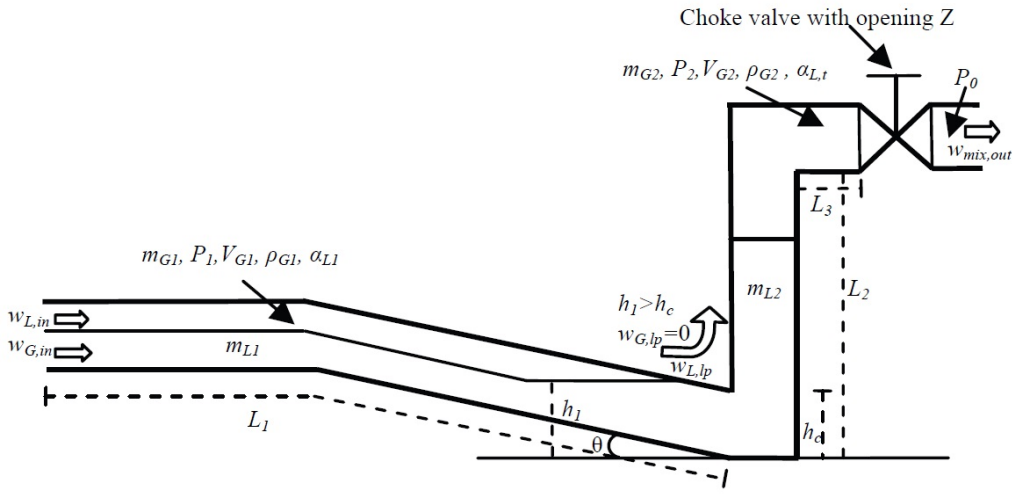

Figure 3.2: Simplified representation of liquid blocking leading to riser slugging

In Figure 3.1, when the liquid is not blocking at the low-point (h_1<h_c), the gas will flow from volume in the pipeline (V_{G1}) to the volume in the riser (V_{G2}) with a mass rate $w_{G,lp}[kg/s]$. Moreover, when the liquid level in the

pipeline section is above the critical level ($h_1 > h_c$), liquid blocks the low-point and the gas flow rate at the low-point is zero, $w_{G,lp} = 0 \quad h_1 \geq h_c$, as seen in Figure 3.2. This leads to riser slugging.

All the model equations can be found in[2].

3.1.1 Controllability Analysis

A controllability analysis of this system has been performed [3]. The controllability analysis concludes that for multiple input multiple output (MIMO) control with one pressure measurement from the pipeline (P_1), combined with choke flow rates(Q) gives the best ability to control the system. In many cases the subsea measurement is not available. Then combining the top side measurements gives satisfactory result. The ability to control the system is dependent on the disturbance in the system. For instance the input rate of liquid ($w_{L,in}$) and gas ($w_{G,in}$) into the system, or measurement noise.

Chapter 4

Theory

4.1 State Space Representation

A state space model is a mathematical model of a process, where the process state x is represented by a vector of finite dimensionality. The state space usually consists of two models, a process model and a measurement model. The first model is a representation of the dynamics that describes how the state propagates in time as a function of external inputs, the second model describes how a vector of numerical measurements y are related to the process states.

The most general discrete form of state space representation is the nonlinear model

$$x_{k+1} = f(x_k, u_k, w_k), \tag{4.1}$$

$$y_k = h(x_k, u_k, v_k), \tag{4.2}$$

where x is the state vector, u is the input, k is the discrete time variable,w and v are the stochastic noise for the process and measurement vectors respectively.

The discrete linear process can be modeled as

$$x_{k+1} = F_k x_k + B_k u_k + h(w_k), \tag{4.3}$$

where $h(w_k)$ is the input disturbance. The measurements of the process have the relationship

$$y_k = H_k x_k + D_k u_k + v_k, \tag{4.4}$$

where F_k, B_k, H_k and D_k are matrices describing the discrete system dynamics. v_k is the measurement noise. If the system has nonlinear dynamics, the matrices can be found by linearization[10].

4.2 Observers

The goal of an observer is to estimate the state x_k based on our knowledge of the noisy system dynamics and the available noisy measurements. For a linear system with noisy measurements, an optimal observer (with respect to expected observer error) is the Kalman filter.

4.2.1 Kalman Filter

In 1960 R.E Kalman presented a new way of formulating the minimum mean square error (MMSE) filtering problem [11]. The Kalman filter uses state space representation that makes it efficient to handle multiple input, multiple output (MIMO) systems. The Kalman filter estimates the internal states of linear dynamics using series of noisy input measurements. It is a two-step process, where the first step is a prediction step and the second step is an update step. The model is linear and the input is Gaussian, and the output will also be Gaussian. As a result, the mean vector $\hat{x}$ and $\hat{P}$ will be sufficient to describe the system. $\hat{x}$ is the state estimate, k is the discrete time variable and $\hat{P}$ is the covariance matrix which contains information about the accuracy of the estimate,

$$P_k = E[e_k e_k^T] = E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T]. \tag{4.5}$$

First, we predict the a priori state vector x and covariance matrix P:

$$\hat{x}_{k+1}^- = F_k \hat{x}_k + B_k u_k, \tag{4.6}$$

$$P_{k+1}^- = F_k P_k F_k^T + Q_k,$$

By using the measurement y, we update the Kalman gain K, a posteriori state vector x, and the covariance matrix P:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}, \tag{4.7}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - (H_k \hat{x}_k^- + D_k u_k)),$$

$$P_k = (I - K_k H_k) P_k^-.$$

Q_k and R_k are the covariance for the process and measurement noise respectively. The system has initial conditions, $\hat{x}_0^-$ and P_0^-. K_k is the Kalman gain which minimize the mean-square estimation error.

4.2.1.1 Properties of the Linear Kalman Filter

If w_k and v_k are zero-mean, uncorrelated, and white, then the Kalman filter is the best linear solution to the above problem. This means, the Kalman filter is the best filter that is a linear combination of the measurements. However, there may be a nonlinear filter that gives a better solution.

4.2.2 Extended Kalman Filter

Unfortunately, most processes are not linear and they need to be linearized before they can be estimated with the linear Kalman filter. The extended Kalman filter tries to solve this. The idea is to linearize the nonlinear system around a trajectory that is updated with the state estimates resulting from the measurements.

1. The system and measurement equations are given as follows:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k, w_k), \\ y_k &= h(x_k, u_k, v_k), \\ w_k &\sim (0, Q_k) \\ v_k &\sim (0, R_k) \end{aligned} \tag{4.8}$$

   where x is the state, y is the measurement, u is the system and measurement input, w and v are the process and measurement noise with zero expected value and covariance Q and R.

2. Initialize the filter as follows:

$$\begin{aligned} \hat{x}_0^+ &= E(x_0) \\ P_0^+ &= E[(x_0 - \hat{x}_0^+)((x_0 - \hat{x}_0^+)^T] \end{aligned} \tag{4.9}$$

3. For $k = 1, 2, \cdots$, perform the following

(a) Compute the Jacobian, which is the first order term of its Taylor expansion evaluated around the point of interest[12]:

$$F_{k-1} = \frac{\partial f_{k-1}}{\partial x}|_{\hat{x}^+_{k-1}} \tag{4.10}$$

$$L_{k-1} = \frac{\partial f_{k-1}}{\partial w}|_{\hat{x}^+_{k-1}} \tag{4.11}$$

(b) Perform the time update of the state estimate and estimation-error covariance as follows:

$$P^-_k = F_{k-1}P^+_{k-1}F^T_{k-1} + Q_{k-1}L^T \tag{4.12}$$

$$\hat{x}^-_k = f_{k-1}(\hat{x}^+_{k-1}, u_{k-1}, 0) \tag{4.13}$$

(c) Compute the following partial derivative matrices

$$H_k = \frac{\partial h_k}{\partial x}|_{\hat{x}^-_k} \tag{4.14}$$

$$M_k = \frac{\partial h_k}{\partial v}|_{\hat{x}^-_k} \tag{4.15}$$

(d) Perform the measurement update of the state estimate and estimation error covariance as follows:

$$K_k = P^-_k H^T_k (H_k P^-_k H^T_k + R_k)^{-1} \tag{4.16}$$

$$\hat{x}^+_k = \hat{x}^-_k + K_k[y_k - h_k(\hat{x}^-_k, u_{k-1}, 0)] \tag{4.17}$$

$$P^+_k = (I - K_k H_k)P^-_k \tag{4.18}$$

4.2.2.1 Properties of the Extended Kalman Filter

The EKF is probably the most widely used estimation algorithm for nonlinear systems. However, many decades of experience has shown that it is difficult to tune, and only reliable for systems that are almost linear on the time scale of the updates. Further, it is also difficult to implement, much because of the Jacobian. The Jacobian needs to exist, and calculating the Jacobian produces many pages of dense algebra that must be converted to code. This might introduce human coding errors that are hard to debug since you do not know which performance to expect [13]. Because the Jacobian is based on the first order Taylor expansion, the EKF is accurate to the first order[11].

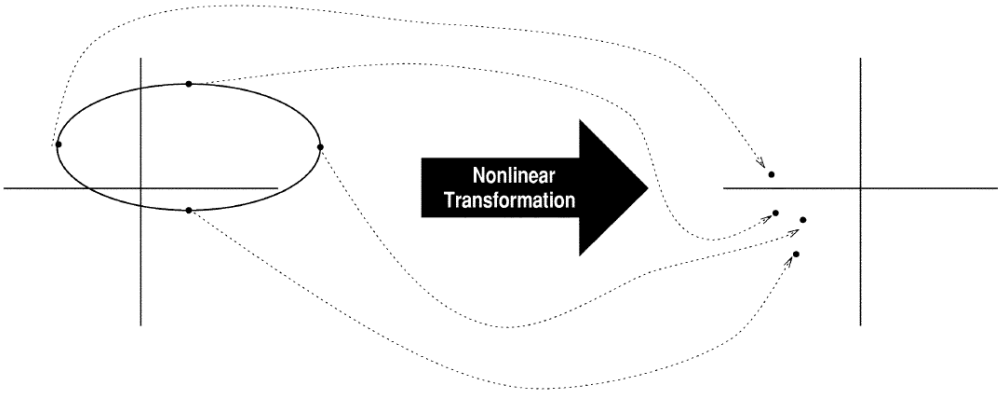

Figure 4.1: The principle of Unscented Kalman Filter[13].

4.2.3 Unscented Kalman Filter

The Unscented Kalman filter is developed to address the deficiencies of algorithms with linearization. It is known for derivative-free optimization and therefore better performance in systems that are categorized as "highly nonlinear". To overcome the difficulties with linearization the unscented transformation was developed as a method to propagate mean and covariance information through nonlinear transformations. The parameterization of the unscented transformation is designed to satisfy the following principle [14].

The information contained in the mean vector and covariance matrix of the random vector x is captured accurately by the unscented transformation in a way that permits the propagation of this information through the nonlinear equation $y = f(x)$ in a computationally efficient manner.

This requirement is satisfied by generating a set of sigma points. Sigma points are a set of points chosen according to a specific, deterministic algorithm, so that their mean and covariance are $\bar{x}$ and $\sum_x$. Further, the nonlinear function is applied to each point, in turn, to yield a cloud of transformed points. The principle of the transformation can be seen in Figure 4.1.

- L: dimension of the state,
- α: small positive constant determining the spread of the sigma points around the mean value of the state: $0.001 < \alpha < 1$,
- β: parameter incorporating prior knowledge on the distribution of the state,

- κ: scaling parameter: [1 ∞),
- index: c is for covariance and m is for mean.

$$\begin{aligned} \gamma &= L + \lambda \\ \lambda &= \alpha^2(L + \kappa) - L \\ W_0^{(m)} &= \frac{\lambda}{L + \lambda} \\ W_0^{(c)} &= \frac{\lambda}{L + \lambda} + (1 - \alpha^2 + \beta) \end{aligned}$$

$$W_i^{(m)} = W_i^{(c)} = \frac{\lambda}{2(L + \lambda)} \quad for \quad i = 1, 2, \ldots, 2L. \tag{4.19}$$

The algorithm is described in steps 1-4 below:

1. We have an L-state discrete-time nonlinear system given by

$$\begin{aligned} x_{k+1} &= f(x_k, u_k, w_k) \\ y_k &= h(x_k, u_k, v_k) \\ w_k &\sim (0, Q_k) \\ v_k &\sim (0, R_k) \end{aligned} \tag{4.20}$$

2. The UKF is initialized as follows:

$$\begin{aligned} \hat{x}_0^+ &= E(x_0) \\ P_0^+ &= E[(x_0 - \hat{x}_0^+)((x_0 - \hat{x}_0^+)^T] \end{aligned} \tag{4.21}$$

3. The following time update equations are used to propagate the state estimate and covariance from one measurement to the next.

   (a) To propagate from time step $(k-1)$ to k, first choose sigma points $x_{k-1}^{(i)}$. The sigma points should be picked using the current best guess for the mean x_{k-1}^+ and covariance P_{k-1}^+:

$$\hat{x}_{k-1}^{(i)} = \begin{bmatrix} \hat{x}_{k-1} \\ \hat{x}_{k-1}^+ + \tilde{x}^{(i)} \end{bmatrix} \qquad i = 1, \ldots 2n \tag{4.22}$$

   where

$$\begin{aligned} \tilde{x}^{(i)} &= (\sqrt{\gamma P_{k-1}^+})_i^T \qquad i = 1, \ldots n \\ \tilde{x}^{n+i} &= -(\sqrt{\gamma P_{k-1}^+})_i^T \qquad i = 1, \ldots n \end{aligned} \tag{4.23}$$

(b) Use the known $f(\cdot)$ to transform the sigma points into $\hat{x}_k^{(i)}$ vectors.

$$\hat{x}_k^{(i)} = f(\hat{x}_{k-1}^{(i)}, u_k, t_k) \tag{4.24}$$

(c) Combine the $\hat{x}_k^{(i)}$ vectors to obtain the a priori state estimate at time k.

$$\hat{x}_k^- = \sum_{i=1}^{2n} W_i^{(m)} \hat{x}_k^{(i)} \tag{4.25}$$

(d) Estimate of the a priori error covariance, with the process Q_{k-1}.

$$P_k^- = \sum_{i=1}^{2n} W_i^{(c)} (\hat{x}_k^{(i)} - \hat{x}_k^-)(\hat{x}_k^{(i)} - \hat{x}_k^-)^T + Q_{k-1} \tag{4.26}$$

4. Next, implement the measurement-update equations

(a) Chose sigma points $\hat{x}_k^{(i)}$ with appropriate changes since the current best guess for the mean and covariance of x_k are $\hat{x}_k^-$ and P_k^-:

$$x_k^{(i)} = \begin{bmatrix} \hat{x}_k \\ \hat{x}_k^+ + \tilde{x}^{(i)} \end{bmatrix} \qquad i = 1, \ldots 2n \tag{4.27}$$

where

$$\begin{aligned} \tilde{x}^{(i)} &= (\sqrt{\gamma P_k^-})_i^T \qquad i = 1, \ldots n \\ \tilde{x}^{n+i} &= -(\sqrt{\gamma P_k^-})_i^T \qquad i = 1, \ldots n \end{aligned} \tag{4.28}$$

(b) Now, use the know nonlinear measurement equation $h(\cdot)$ to transform the sigma points into $\hat{y}_k^{(i)}$ vectors (predicted measurements).

$$\hat{y}_k^{(i)} = h(\hat{x}_k^{(i)}, u_k, t_k) \tag{4.29}$$

(c) Combine the $\hat{y}_k^{(i)}$ vectors to obtain the predicted measurement at time k.

$$\hat{y}_k = \sum_{i=1}^{2n} W_k^{(m)} \hat{y}_k^{(i)} \tag{4.30}$$

(d) Estimate the covariance of the predicted measurement. To take the measurement noise into account, R_k should be added to the expression.

$$P_{y,k} = \sum_{i=1}^{2n} W_k^{(c)} (\hat{y}_k^{(i)} - \hat{y}_k)(\hat{y}_k^{(i)} - \hat{y}_k)^T + R_k \tag{4.31}$$

(e) Further, the cross covariance between $\hat{x}_k^-$ and $\hat{y}_k$ should be estimated

$$P_{xy,k} = \sum_{i=1}^{2n} W_k^{(c)} (\hat{x}_k^{(i)} - \hat{x}_k^-)(\hat{y}_k^{(i)} - \hat{y}_k)^T \tag{4.32}$$

(f) Finally, the measurement update of the state estimate can be performed using the Kalman filter equations

$$\begin{aligned} K_k &= P_{xy} P_y^{-1} \\ \hat{x}_k^+ &= \hat{x}_k^- + K_k(y_k - \hat{y}_k) \\ P_k^+ &= P_k^- - K_k P_y K_k^T \end{aligned} \tag{4.33}$$

4.2.3.1 Properties of the Unscented Kalman Filter

For processes and measurements that have Gaussian error distribution and the prior state is Gaussian, the UKF is accurate to the third order. For non-Gaussian distributions, the UKF is accurate to at least the second order [14]. Hence, the UKF is more accurate than the EKF.

4.2.3.2 Unscented Kalman Filter with Forgetting Factor

Another tuning parameter for the UKF is forgetting factor. The forgetting factor is used to scale the a posteriori covariance $\hat{P}^+$.

$$P_k^+ = \frac{P_k^+}{ff} \tag{4.34}$$

The a posteriori covariance is used in eq. 4.23 to make sigma points. When the forgetting factor is higher than one ($ff > 1$) the covariance reduces and it "forgets" some of its covariance, hence the observer gain is decreased. Further, if the forgetting factor is less than one ($ff < 1$), the covariance is artificially increased and the observer gain is increased. A disadvantage of increased observer gain is that it becomes sensitive to noise.

4.3 High-Gain Observer

In the end of my thesis work the high-gain observer was investigated in cooperation with my supervisor Esmaeil Jahanshahi, and it is included for completeness. The high-gain observer guarantees that the output feedback controller recovers the performance of the state feedback controller when the observer gain is sufficiently high. The observer gain is designed so that the observer is robust to uncertainties in modeling of the nonlinear functions. The structure of the high-gain observer is:

$$\begin{aligned} \dot{\hat{z}}_1 &= f_1(\hat{z}) \\ \dot{\hat{z}}_2 &= f_2(\hat{z}) \\ \dot{\hat{z}}_3 &= f_3(\hat{z}) + \frac{1}{\epsilon}(y - \hat{y}) \\ \dot{\hat{z}}_4 &= f_4(\hat{z}) \end{aligned} \tag{4.35}$$

where

- z_1, mass of gas in the pipeline(x_1)
- z_2, mass of liquid in the pipeline(x_2)
- z_3, Pressure at top of riser (P_2)
- z_4, mass of liquid in the riser(x_4)

and $\frac{1}{\epsilon}$ is the high-gain. The high-gain observer is essentially the model that we have transformed in to the observability form[15]. For this, we used top-pressure P_2, which is the measured output, in place of x_3(mass of gas inside the riser) as observer state.

$$P_2 = \frac{ax_3}{b - x_4} \tag{4.36}$$

We need only to derive the time derivative of the top-pressure by using partial derivatives:

$$\begin{aligned} f_3(\hat{z}) &= \frac{dP_2}{dt} \\ f_3(\hat{z}) &= \frac{\partial P_2}{\partial x_3}\dot{x}_3 + \frac{\partial P_2}{\partial x_4}\dot{x}_4 \end{aligned} \tag{4.37}$$

where

$$\frac{\partial P_2}{\partial x_3} = \frac{a}{b - x_4} \tag{4.38}$$

$$\frac{\partial P_2}{\partial x_4} = \frac{a x_3}{(b - x_4)^2} \tag{4.39}$$

and a and b are model constants. The MATLAB code can be found in Appendix A.2. More theory about the high-gain nonlinear observer can be found in [15] chapter 12.5 and in [16].

4.4 Controllers

4.4.1 PI-controller

The PI controller is widely used in the industry, but it can only be used for single input, single output (SISO) cases.

- P, proportional (K_p): Increased K_p makes the rise time decrease, but the overshoot increase and the stability margin decrease.
- I, integrator (K_i): Increased K_i decrease the rise time, but increase the overshoot, increase settling time and the stability margin decrease. The steady state error is eliminated.

The PI-controller integrates the input error to the controller and multiplies it with an integration gain K_i. This is added to the input error and multiplied with the proportional gain K_p.

$$\begin{aligned} e(k) &= y - reference \\ I &= I + e(k)Ts \\ u(k) &= K_p(e(k) + K_i I) \end{aligned} \tag{4.40}$$

where Ts is the time step. The desired closed loop dynamics is obtained by adjusting the tuning parameters K_p and K_i. This controller is fast and eliminates the steady state error. Tuning can be done by trial and error, or by simple analytical tuning rules[17].

4.4.2 Linear Quadratic Regulator

In LQR control, it is assumed that the plant dynamics are linear and known,

$$x_{k+1} = F_k x_k + B_k u_k \tag{4.41}$$

and that you have a deterministic initial value problem: given the system in eq. (4.41) with a non-zero initial state $x(0)$, find the optimal input signal u_k which takes the system to the zero state $x = 0$ in an optimal manner, by minimizing the quadratic cost function

$$J = \sum_{k=1}^{i} x_k^T Q x_k + u_k^T R u_k. \tag{4.42}$$

This requires that the pair (F_k, B_k) is stabilizable (its state can be transferred to the origin from any initial state in infinite time).

Q and R are symmetric positive-definite weight matrices for the states x and the input u respectively. The task is to minimize the cost function, so if R is high, using the input u will be expensive, and you will get slow response. By having a high value on the first element on Q compared to the other diagonal elements, the controller will prioritize the first state. Further, if Q is much greater than R, you will get a fast response. The optimal Q and R are in many cases found by trial and error.

The optimal solution is $u_k = -K_{lqr}x_k$, where

$$K_{lqr} = R^{-1}B^T X \tag{4.43}$$

and X is the symmetric positive definite solution of the Riccati equation [18]

$$F^T X + XF - XBR^{-1}B^T X + Q = 0 \tag{4.44}$$

4.4.3 Linear Quadratic Gaussian Controller

The principle is much the same for the LQG controller as for the LQR. The name LQG comes from a linear model, an integral (or sum) quadratic cost function, and Gaussian white noise processes to model disturbance signals and noise.

$$\hat{x}(k+1) = F_k\hat{x}(k) + B_k u(k) + w_k \tag{4.45}$$
$$y(k) = H_k\hat{x}(k) + D_k u(k) + v_k \tag{4.46}$$

and the process and measurement noise, w_k and v_k, are uncorrelated zero-mean Gaussian stochastic processes. Usually, a linear Kalman filter is used to model the process state $\hat{x}$. The solution to the LQG problem is known as the Separation Theorem[18], and it consists of first determining the K_lqr in eq. (4.43), then finding the optimal state estimate $\hat{x}$ given by a Kalman filter. The principal of the controller is shown in Figure 4.2.

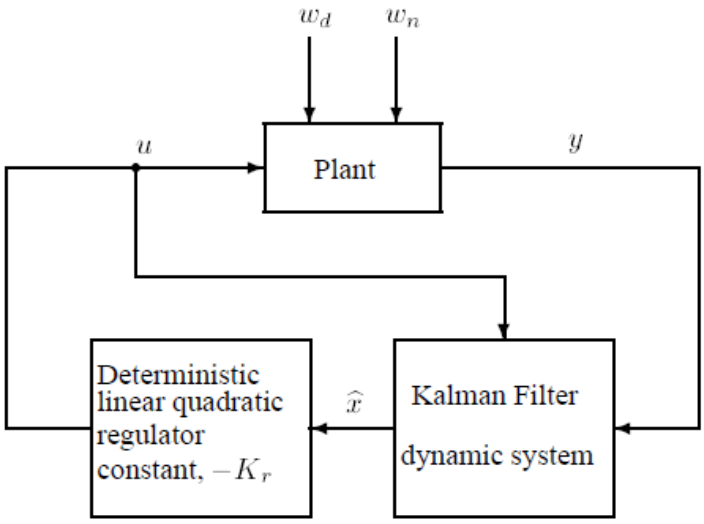

Figure 4.2: LQG-controller, which consists of LQR and Kalman Filter[18]

4.4.4 Model Predictive Control

The more advanced Model Predictive Controller (MPC) will now be addressed. The MPC is one of the controllers which has made significant impact on the process industry. The MPC is a controller which

- uses a multivariable process model to predict future behavior,
- solves mathematical optimization problems of the predicted future performance, and
- handles multiple constraints on inputs, states and outputs.

One of the main reasons for its success in the process industry is the constraint handling. For instance, a choke valve and flow rate have saturation characteristics, because the valve can only be 100% open and the flow rate might have maximum values due to fixed pipe diameters. When the process operates at its constraints it is often running at its most profitable condition. In addition, the MPC is a controller which is easy to understand, and has few tuning parameters. However, since the MPC controller is based on a model of the plant, it is important to always have a model of the system which are good enough, but not to complex. If the model is to complex, you will use too much computational time on the model when optimizing. To have a robust system, the plant should not be operated exactly at the real limits of

its capability, due to unexpected disturbances. But, the better the control system is, the closer to the constraints you can operate.

In the unconstrained case the MPC controller reduces to an LQR controller which optimizes over a horizon. Therefore, the main reason for widespread use of MPC in place of the LQR is that it offers a straightforward and transparent approach to handling multiple constrains.

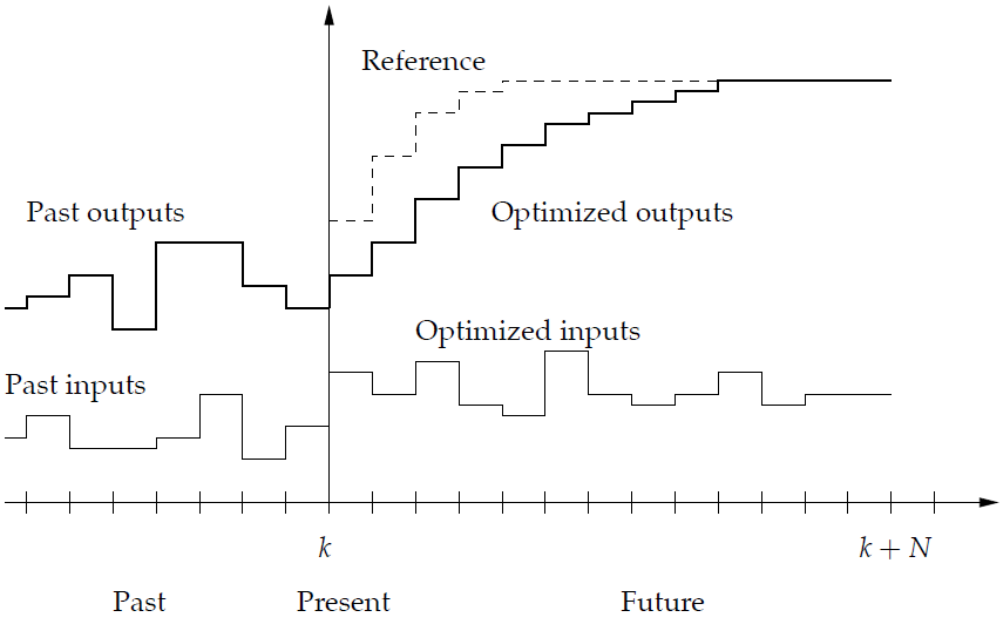

Figure 4.3: MPC principle[19]

The basic idea for the optimization problem is illustrated in Figure 4.3.

1. At time instant k, solve the *quadratic programming* (QP) (see Section 4.4.4.8) problem to obtain an optimal, feasible input sequence.
2. Apply the first input to the process.
3. Set $k = k + 1$, and repeat the previous steps.

4.4.4.1 Linear MPC

This MPC implementation is taken from *Predictive Control: with Constraints* By J.M. Maciejowski[20]. The process to be controlled is described by a discrete state space model

$$\begin{aligned} x_{k+1} &= F_k x_k + B_k u_k \\ y_k &= H_k x_k + D_k u_k \end{aligned} \tag{4.47}$$

where F_k and B_k are discrete time process dynamics. F_k can be either stable, or unstable, but we assume that the pair (F_k, B_k) is stabilizable.

The objective we want to obtain by control is to optimize performance by minimizing:

$$V(k) = \sum_{i=H_w}^{H_p} (x_k - x_0)^T Q (x_k - x_0) + \sum_{i=0}^{H_u - 1} (u_k - u_{k-1})^T R (u_k - u_{k-1}) \tag{4.48}$$

subject to

$$x_{min} \leq \quad x \quad \leq x_{max} \tag{4.49}$$

$$\Delta u_{min} \leq \quad \Delta u \quad \leq \Delta u_{max} \tag{4.50}$$

$$u_{min} \leq \quad u \quad \leq u_{max} \tag{4.51}$$

where Δu is the input change, u is the input and k is the discrete time variable. x_k is given by eq. (4.47) which is based on x_0 and the previous calculated u_{k-1}. As in Section 4.4.2, Q and R are positive definite tuning parameters. Further, the state must be detectable trough Q (all unobservable states are stable). In all cases we want $(x_k - x_0) \to 0$. H_u and H_p are tuning parameters called the control- and prediction horizon. For simplicity, we will assume that the control and prediction horizon will be equal. Another tuning parameter is H_w, and this is when you start predicting. Further, the state space dynamics is represented without index k to save space.

4.4.4.2 Prediction

Assume that we know nothing about any disturbance or measurement noise. In order to solve the predictive control problem, we must have a way of computing the predicted values of the controlled variables, $x(k+i|k)$. By writing out eq.(4.47)

$$\begin{aligned} x(k+1|k) &= Fx(k) + Bu(k|k) \\ x(k+2|k) &= Fx(k+1|k) + Bu(k+1|k) \\ &= F^2x(k) + FBu(k|k) + Bu(k+1|k) \\ &\vdots \\ x(k+H_p|k) &= Fx(k+H_p-1|k) + Bu(k+H_p-1|k) \\ &= F^{H_p}x(k) + F^{H_p-1}Bu(k|k) + \ldots + u(k+H_p-1|k) \end{aligned} \tag{4.52}$$

and

$$\begin{aligned} u(k|k) &= \Delta u(k|k) - u(k-1) \\ u(k+1|k) &= \Delta u(k+1|k) + \Delta u(k|k) - u(k-1) \\ &\vdots \\ u(k+H_u-1|k) &= \Delta u(k+H_u-1|k) + \ldots + \Delta u(k|k) + u(k-1) \end{aligned} \tag{4.53}$$

Finally, we can write this as matrix form:

$$\begin{bmatrix} x(k+1|k) \\ \vdots \\ x(k+H_u+1) \end{bmatrix} = \Psi x(k) + \Upsilon u(k-1) + \Theta \begin{bmatrix} \Delta u(k|k) \\ \vdots \\ \Delta u(k+H_u-1|k) \end{bmatrix} \tag{4.54}$$

where

$$\Psi = \begin{bmatrix} F \\ \vdots \\ F^{H_u} \end{bmatrix}, \Upsilon = \begin{bmatrix} B \\ \vdots \\ \sum_{i=0}^{H_u-1} F^i B \end{bmatrix}$$

and

$$\Theta = \begin{bmatrix} B & 0 & \ldots & 0 \\ F + F^0 B & B & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=0}^{H_u-1} F^i B & \sum_{i=0}^{H_u-2} F^i B & \ldots & B \end{bmatrix} \tag{4.55}$$

4.4.4.3 Using an Observer

If we cannot measure the full state vector, or if the output consists of some linear combinations of the states, then an observer can be used to estimate the state vector. If an observer is used in the MPC, then the x needs to be replaced with $\hat{x}$ since you use the estimated x in the MPC formulation. More about observers in Section 4.2. Your MPC formulation will now be

$$V(k) = \sum_{i=H_w}^{H_p} (\hat{x}_k - x_0)^T Q(\hat{x}_k - x_0) + \sum_{i=0}^{H_u-1} (u_k - u_{k-1})^T R(u_k - u_{k-1}) \quad (4.56)$$

subject to

$$x_{min} \leq \quad \hat{x} \quad \leq x_{max} \quad (4.57)$$
$$\Delta u_{min} \leq \quad \Delta u \quad \leq \Delta u_{max} \quad (4.58)$$
$$u_{min} \leq \quad u \quad \leq u_{max} \quad (4.59)$$

4.4.4.4 Solving Predictive Control Problems

We can rewrite the cost function which we want to minimize

$$V(k) = \|X(k) - T(k)\|_Q^2 + \|\Delta U(k)\|_R^2 \quad (4.60)$$

where

$$X(k) = \begin{bmatrix} x(k+H_w|k) \\ \vdots \\ x(k+H_p|k) \end{bmatrix} \quad T(k) = \begin{bmatrix} x_0(k+H_w|k) \\ \vdots \\ x_0(k+H_p|k) \end{bmatrix}$$

$$\Delta U(k) = \begin{bmatrix} \Delta\hat{u}(k|k) \\ \vdots \\ \Delta\hat{u}(k+H_u-1|k) \end{bmatrix} \quad (4.61)$$

The weighting matrices Q and R are given by

$$Q = \begin{bmatrix} Q(H_w) & 0 & \dots & 0 \\ 0 & Q(H_w+1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Q(H_p) \end{bmatrix} \quad (4.62)$$

and

$$R = \begin{bmatrix} R(0) & 0 & \dots & 0 \\ 0 & R(1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & R(H_u - 1) \end{bmatrix} \tag{4.63}$$

They are tuned the same way as described in Section 4.4.2.

Recall from eq. (4.54) that X has the form

$$X(k) = \Psi x(k) + \Upsilon u(k-1) + \Theta \Delta U(k) \tag{4.64}$$

Define

$$\varepsilon(k) = T(k) - \Psi x(k) - \Upsilon u(k-1) \tag{4.65}$$

This can be thought of as "tracking error" in the sense of the difference between $T(k)$ which is the future target trajectory, and the "free response" of the system. The response that would occur over the prediction horizon if no input changes were made, $\Delta u = 0$, is the "free response". If the tracking error is zero, $\varepsilon = 0$, then it will be correct to set $\Delta u = 0$.

Thus we have,

$$\begin{aligned} V(k) &= \|X(k) - T(k)\|_Q^2 + \|\Delta U(k)\|_R^2 \\ &= \|\Theta \Delta U(k) - \varepsilon(k)\|_Q^2 + \|\Delta U(k)\|_R^2 \\ &= [\Delta U(k)^T \Theta^T - \varepsilon(k)^T] Q [\Theta \Delta U(k) - \varepsilon(k)] \\ &\qquad + \Delta U(k) T R \Delta U(k) \\ &= \varepsilon(k)^T Q \varepsilon(k) - \Delta U(k)^T G + \Delta U(k)^T H \Delta U(k) \end{aligned} \tag{4.66}$$

where

$$G = 2\Theta^T Q \varepsilon(k) \qquad and \qquad H = \Theta^T Q \Theta + R \tag{4.67}$$

and neither G nor H depends on $\Delta U(k)$

This has the form

$$V(k) = const - \Delta U(k)^T G + \Delta U(k)^T H \Delta U(k). \tag{4.68}$$

To clarify the dimensions are summarized in Table 4.1.

Matrix	Dimensions
Q	$m(H_p - H_w + 1) \times m(H_p - H_w + 1)$
R	$lH_u \times lH_u$
Ψ	$m(H_p - H_w + 1) \times n$
Υ	$m(H_p - H_w + 1) \times l$
Θ	$m(H_p - H_w + 1) \times lH_u$
ε	$m(H_p - H_w + 1) \times 1$
G	$lH_u \times 1$
H	$lH_u \times lH_u$

Table 4.1: This table shows the dimensions on the matrices and vectors involved in computing the optimal input vector. The plant has l inputs, n states, and m controlled outputs.

4.4.4.5 Unconstrained Optimization

The solution to the unconstrained optimization problem is straight forward. To find the optimal input change, $\Delta U(k)$, find the gradient of the cost function $V(k)$ and set it to zero. From eq. (4.68) we find

$$\nabla_{\Delta U(k)} V = -G + 2H\Delta U(k) \tag{4.69}$$

which gives the optimal set of future input moves

$$\Delta U(k)_{opt} = \frac{1}{2} H^{-1} G \tag{4.70}$$

For each calculation of $\Delta U(k)$ only the first elements corresponding to the number of plant inputs of $\Delta U(k)$ are applied, then $\Delta U(k)$ is recalculated. This has the form

$$\Delta u(k)_{opt} = [I_l, 0_l \ldots, 0_l] \Delta U(k)_{opt} \tag{4.71}$$

where I_l is the $l \times l$ identity matrix, and 0_l is the $l \times l$ zero matrix.

4.4.4.6 Comments on the MPC

- In eq. (4.54), the calculation of Ψ involves computing F^i, and i can be quite large. If the plant is stable or unstable, some elements in F^i might become extremely high or small compared to others. Since computers work with finite-precision arithmetic, the results might become wrong. One way to handle this problem is "pre-stabilizing" the plant[20].

- As described earlier, Q and R are important tuning parameters for the MPC. The value of the prediction horizon H_p, control horizon H_u and working horizon H_w needs to be considered. H_p and H_u affect the close loop performance and the computational complexity. Generally, shorter horizons gives lower complexity optimization problem and hence lower computational load for the online optimization problem, while longer horizons gives better performance, at the cost of higher computational load.

4.4.4.7 Constrained Optimization

Now we consider the case when constraints are present. They are written in the form

$$E\begin{bmatrix}\Delta U(k)\\ 1\end{bmatrix} \leq 0 \tag{4.72}$$

$$F\begin{bmatrix}U(k)\\ 1\end{bmatrix} \leq 0 \tag{4.73}$$

$$G\begin{bmatrix}Z(k)\\ 1\end{bmatrix} \leq 0 \tag{4.74}$$

where $U(k) = [\tilde{u}(k|k)^T, \ldots, \tilde{u}(k+H_u-1|k)^T]^T$. The $\tilde{u}$ is used to indicate that it is not yet the optimal u. We want to end up with constraints for the optimization problem expressed in terms of $\Delta U(k)$

$$\Omega \Delta U(k) = w. \tag{4.75}$$

Suppose F has the form

$$F = [F_1, F_2, \ldots, F_{H_u}, f], \tag{4.76}$$

where q is the number of constraints on u, and each F_i is of the size $q \times m$, and f has the size $q \times 1$. To illustrate $0 \leq u_1 \leq 1$ gives $q = 2$. Now, eq. (4.73) can be written as:

$$\sum_{i=1}^{H_u} F_i \tilde{u}(k+i-1|k) + f \leq 0. \tag{4.77}$$

Since

$$\tilde{u}(k+i-1|k) = u(k-1) + \sum_{j=0}^{i-1} \Delta\tilde{u}(k+j|k) \tag{4.78}$$

we can write eq. (4.73) as

$$\begin{aligned} \sum_{j=1}^{H_u} F_j \Delta\tilde{u}(k|k) \quad &+ \quad \sum_{j=2}^{H_u} F_j \Delta\tilde{u}(k+1|k) \\ + \ldots + F_{H_u} \Delta\tilde{u}(k+H_u-1|k) \quad &+ \quad \sum_{j=1}^{H_u} F_j u(k-1) + f \leq 0 \end{aligned} \tag{4.79}$$

By defining $\mathcal{F}_i = \sum_{j=i}^{H_u} F_j$ and $\mathcal{F} = [\mathcal{F}_1, \ldots, \mathcal{F}_{H_u}]$ we only have to summarize to see that we have converted eq. (4.73) into a linear inequality constraint on $\Delta U(k)$.

$$\mathcal{F}\Delta U(k) \leq -\mathcal{F}_1 U(k-1) - f \tag{4.80}$$

Equation (4.74) is treated similarly:

$$G \begin{bmatrix} \Psi x(k) + \Upsilon u(k-1) + \Theta \Delta U(k) \\ 1 \end{bmatrix} \leq 0 \tag{4.81}$$

Now, letting $G = [\Gamma, g]$ where g is the last column of G, this can be written in the desired form as

$$\Gamma\Theta\Delta U(k) \leq -\Gamma[\Psi x(k) + \Upsilon u(k-1)] - g \tag{4.82}$$

The only remaining constraint to handle is the simplest one, eq. (4.72), we write

$$W\Delta u(k) \leq w \tag{4.83}$$

Finally, we can summarize

$$\begin{bmatrix} \mathcal{F} \\ \Gamma\Theta \\ W \end{bmatrix} \Delta U(k) \leq \begin{bmatrix} -\mathcal{F}_1 u(k-1) - f \\ -\Gamma[\Psi x(k) + \Upsilon u(k-1)] - g \\ w \end{bmatrix} \tag{4.84}$$

If an observer is used, replace x by $\hat{x}$.

Now, we have obtained the following constrained optimization problem:

$$minimize \quad \Delta U(k)^T H \Delta U(k) - G^T \Delta U(k) \tag{4.85}$$

subject to the inequality constraints in eq. (4.84). This is a standard optimization problem know as *quadratic programming (QP)*, which can be solved with standard algorithms.

4.4.4.8 Solving Quadratic Programming Problems with Constraints

There are several software programs that can be used to solve QP problems, for instance *MATLAB* and *Microsoft Excel*. *MATLAB*[21] has a build-in function *quadprog* which solves the problem and gives you the optimal input. In *quadprog* you can decide which algorithm that should be used, for example *Active-Set* or *interior-point* method. These are briefly described in [20], and more carefully derived in [22].

Even though the QP problem has a global solution that exists, hence it is convex[22], it might be infeasible because of the constraints. There are several approaches to solve this problem:

- Implement soft constraints
- Actively manage the constraint definition at each k
- Actively manage the horizons at each k
- Use non-standard solution algorithms

Recommended literature on MPC and examinations of these solutions can be found in [20] and [19].

Chapter 5

Implementation

As mentioned in Section 4.2 about Kalman filters, there is an assumption that the process and measurement noise are white with zero mean. However, to simulate a more realistic case, and to be able to generate slug flow into the anti-slug controller the process noise is not set to be like this. Figure 5.1 presents an input that generates slug flow.

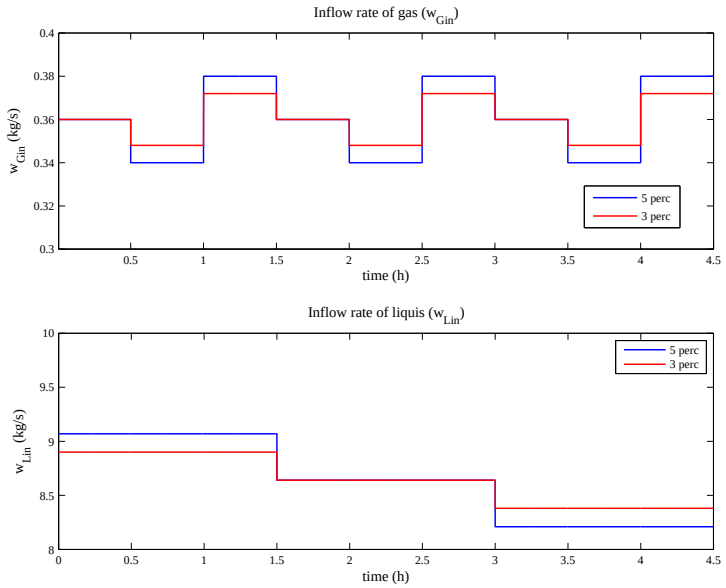

Figure 5.1: Gas and liquid flow into the system

The files for simulating the system are found on the CD.

For simulation of the nonlinear model the *MATLAB* function *ode15s*[21] is used. The state for the nonlinear model includes one steady-state part x_0 plus one deviation part dx. Further, linear controllers are used, they work with the deviation part dx, therefor the steady-state part x_0 is removed when states are used in the controller.

5.1 Tuning Parameters

Tuning of the observers and controllers are not the main focus of this thesis. Because of the nonlinearity in the system, the time steps need to be small and the simulation is very time consuming. In compromise between time and performance simulations are done to find the best values for the tuning parameters.

5.2 Matlab Code

5.2.1 Controllers

PI Controller:

```
%% Initialization
Kp = 0.1;                    %% Kp=1 is also used.
Ki = 0.0001;

%% inside loop

% ================ PI Controller ======================
    P1_hat(k-1)=y_hat(1,k-1);
    e(k-1) = P1_hat(k-1) - y0(1);
    I = I + e(k-1)*Ts;
    u_c(k-1) = Kp*(e(k-1)+Ki*I);

    \% Saturates the controller betwEen [0 1].
    if(u_c(k-1)<-u_pc(k-1))
        u_c(k-1)=-u_pc(k-1);
    end
    if(u_c(k-1)>(1-u_pc(k-1)))
        u_c(k-1)=1-u_pc(k-1);
    end
```

LQR Controller

```
% ================ LQR Controller ====================
Q_lqr=1e-3*diag([1 0 1 0]);
R_lqr=1e9;
N_lqr=zeros(4,1);
[K_lqr,S,e] =dlqr(F,B,Q_lqr,R_lqr,N_lqr);

u_c(k-1)=-K_lqr*x_hat(:,k-1);

\% Saturates the controller between [0 1].
if(u_c(k-1)<-u_pc(k-1));
        u_c(k-1)=-u_pc(k-1);
end
if(u_c(k-1)>(1-u_pc(k-1)));
        u_c(k-1)=1-u_pc(k-1);
end
```

MPC Controller

Code for initialization of the MPC can be found in appendix A.4.

```
% ================ MPC Controller ====================
%% Formulate H and G.
  x_prev=x_hat(:,k-1);
  u_control_prev=u_MPC(k-1);

  T=repmat([0 0 0 0]',Hu,1);
  EPS=T-Psi*x_prev-Gamma*u_control_prev; %eq. 4.62
  H=Theta'*Q_MPC*Theta+R_MPC;      %eq. 4.64
  H=(H+H')/2;                      %To ensure symmetry/compensate
                          for numerical inaccuracies
  G=2*Theta'*Q_MPC*EPS;            %eq. 4.64



  %% Formulate constraints as eq. 4.81

%  Ff                %1st line, left side
   GcFi=Gcons*Theta; %2nd line, left side
%  W                 %3rd line, left side

  %Eps=-Psi*x_prev-Gamma*u_control_prev; %2nd line right side,
                                         midle part

  ohm1=-f1*u_control_prev-f;      %1st line, right side
  g1=Gcons*EPS-g;                %2nd line, right side
% w                               %3rd line, right side

  OhmL=[Ff;GcFi;W];       % Total vector, left side of constraints
```

```
OhmR = [ohm1; g1; w];  % Total vector, right side of constraints
                       % OhmL*deltaU<=OhmR, evt. Ax<=b

%% Quadratic program, constrained case:
[delU(:,k),fval,exFlag,output,lambda] = quadprog(H, -0.5*G,
                        OhmL,OhmR, [],[],[],[],[],optionsQP);

%% Input to simulation of plant
u_MPC(1,1)=0;
u_MPC(1,k) = delU(1,k)+u_MPC(1,k-1); % Add up to total control
                                     (deltaU computed)
if (u_MPC(k)>1-u(1,k))
    u_MPC(k)=1-u(1,k);
elseif(u_MPC(k)<-u(1,k))
    u_MPC(k)=-u(1,k);
end
```

Simulation

```
% ========== Simulation of System ==========
    u_in(:,k)=[u(1,k)+u_c(k-1); u_n(2,k); u_n(3,k)];
    u_no_noise(:,k)=[u(1,k)+u_c(k-1); u(2,k); u(3,k)];

     [tt,xt]=ode15s(@v4_new_4d_model,[k-1;k],x(:,k-1)+x0,
                options,u_in(:,k),'derivatives',par);
    x(:,k)= xt(end,:)'-x0;
    yt=v4_new_4d_model(k,xt(end,:)',u_in(:,k),'measurements',par);
    y(:,k) = yt+n_m(:,k);
```

5.2.2 Observers

The process noise is w_k and the output noise is $vt = Ew + Mv$

$$\begin{aligned} vt &= Ew + Mv \\ var(vt) &= var(Ew + Mv) \\ &= Exp((Ew + Mv)(Ew + Mv)') \\ &= Exp(Ew + Mv)(w'E' + v'M') \\ &= Exp(Eww'E' + Ewv'M' + Mvw'E' + Mvv'M') \\ &= EQnE + ENnM' + MNn'E' + MRnM' \\ R &= EQn + Hn + Hn' + MRnM' \end{aligned} \tag{5.1}$$

where

- E is the matrix corresponding to the noisy process input $w_{G,in}$ and $w_{L,in}$.
- M is the matrix corresponding to the noisy stochastic measurement input.

Kalman Filter
Simulink was used to implement the linear KF. To simulate with the nonlinear function as for the EKF and UKF, an S-Function was used. The Simulink diagram is attach in Appendix A.2.

Extended Kalman Filter

```
% ========== Extended Kalman Filter ==========
% p.409 in Optimal State Estimation. By Dan J. Simon
%3a)
%F =expm(A*Ts)
%BL=(F-I)inv(A)B
%L =BL(:,sensors)

%b)
P_pri=F*P*F'+Q;

[tt,xt]=ode15s(@v4_new_4d_model,[k-1;k],x_hat(:,k-1)+x0,
    options,u_no_noise(:,k),'derivatives',par);
x_hat_pri(:,k)= xt(end,:)'-x0;
yt=v4_new_4d_model(k,xt(end,:)',u_no_noise(:,k),
    'measurements',par);
y_hat_pri(:,k) = yt(1:3);
% y_hat_pri is calculated based on x_hat_pri.

%c)
%H=C(sensors,:);
%M=eye(length(sensors));

%d)
K=P_pri*H'/(H*P_pri*H'+R);
x_hat(:,k)= x_hat_pri(:,k) + K*(y(sensors,k)
    - y_hat_pri(sensors,k));
y_hat(:,k) = v4_new_4d_model(t(k),x_hat(:,k)+x0,
    u_no_noise(:,k),'measurements',par);
P=(eye(size(P_pri))-K*H)*P_pri;
```

Unscented Kalman Filter

```
%%%% Initialization of UKF %%%%
ff = .98;
L = 4;
```

```
alpha=1;
beta=0;
kapa=1;
lambda=alpha^2*(L+kapa)-L;
gama=sqrt(L+landa);
Wc=[lambda/(L+lambda)+(1-alpha^2+beta) , ones(1,2*L)*lambda/
                    (2*(L+lambda))];
Wm=[lambda/(L+lambda) , ones(1,2*L)*lambda/(2*(L+lambda))];
Wc1=ones(L,1)*Wc;
Wc2=ones(2,1)*Wc;
Wm1=ones(L,1)*Wm;
Wm2=ones(2,1)*Wm;
```

```
%%%% Inside loop%%%%
  % ============ Choosing Sigma Points ================
    xhatk_1 = x_e(:,k-1)+x0;
    [RSK,p]=chol( (gamma^2) * Pk_1 );
    if p==0,
        SK = RSK';
    else
            SK=sqrt(abs( (gamma^2) * Pk_1 ));
    end

    Xk_1=[xhatk_1 , xhatk_1*ones(1,L)+SK , xhatk_1*ones(1,L)-SK ];
                                              %eq.4.25

    %================== Propagation of Sigma Points ============
    Xk = zeros(L,2*L+1);
    for j=1:2*L+1,
        [tt,xt]=ode15s(@v4_new_4d_model,[k-1;k],Xk_1(:,j),
                    options,u_in_noNoise(:,k),'derivatives',par);
        Xk(:,j) = xt(end,:)';    %eq.4.27
    end

    xhatk_=sum((Xk).*Wm1,2); %eq.4.28
    temp1=Xk-xhatk_*ones(1,2*L+1);
    Pk_=Wc1.*temp1*temp1'+Q_KF;% eq.4.29
    %================ Measurement Update ====================

    SK=chol( (gamma^2) * Pk_ )';
    Xk=[xhatk_ , xhatk_*ones(1,L)+SK , xhatk_*ones(1,L)-SK];
                  %eq.4.30
    Yk = zeros(2,2*L+1);
    for j=1:2*L+1,
        y_k = v4_new_4d_model(t(k),Xk(:,j),u_in_noNoise(:,k),
                    'measurements',par); %eq.4.32
        Yk(:,j)= y_k(sensors);
    end
```

```
yhatk_=sum(Wm2.*Yk,2);% eq. 4.33


Pxkyk=Wc1.*(Xk-xhatk_*ones(1,2*L+1))*(Yk-
    yhatk_*ones(1,2*L+1))';% eq.4.35

temp1=Yk-yhatk_*ones(1,2*L+1);
Pyk_yk_=Wc2.*temp1*temp1'+R_KF; % eq.4.34


Kk=Pxkyk/Pyk_yk_; % eq. 4.36

xhatk=xhatk_+Kk*(y(sensors,k)-yhatk_);% eq. 4.36

x_e(:,k)=xhatk-x0;
y_hat(:,k)=v4_new_4d_model(t(k),xhatk,u_in_noNoise(:,k),
    'measurements',par);

Pk=Pk_-Kk*Pyk_yk_*Kk';%eq 4.36

p11(k,:)=diag(Pk)';
Pk_1=Pk/ff;
```

Chapter 6

Results

All simulations are done with the nonlinear model in Chapter 3. The measurement noise is generated with the *randn()* function in MATLAB for each time step, consequently the output have a lot of noise. To improve the plots, only the 10'th sample is shown for the subsea pressure P_1, topside pressure P_2 and mass rate W_{out}. However, the noise characteristic of the choke valve is important, so every sample is plotted for u. All simulations have nominal choke valve opening $u = 0.10$ if nothing else is mentioned.

For simplicity some notation;

- "$PI + Observer,\ P_2\ and\ W_{out},\ 3\%$" means a PI controller based on $\hat{P}_1$ from an observer with 3% input disturbance. Further, P_2 and W are measured.
- "$Controller + Observer,\ P_1\ and\ W_{out},\ 5\%$" or $MPC + Observer$ means a controller based on $\hat{x}$ from an observer with 5% input disturbance. Further, P_1 and W are measured.

		Nominal u=0.10			
		Using subsea measurement and mass rate(P_1 and W_{out})		Using topside measurements(P_2 and W_{out})	
		3% disturbance	5% disturbance	3% disturbance	5% disturbance
Direct measure-ment	PI	Yes			
	LQR	Yes			
	MPC	Yes			
Linear KF Estimate	PI	Yes	Yes	No	No
	LQR	Yes	Yes	No	No
	MPC	--	--	--	--
EKF Estimate	PI	Yes	Yes	Yes, fig 6.2	No, fig 6.1
	LQR	Yes	Yes	No, fig 6.4	No, fig A.3
	MPC	Yes	Yes	No	No, fig A.4
UKF Estimate	PI	Yes	Yes	Yes, fig 6.3	Yes fig 6.5
	LQR	Yes	Yes	Yes	Yes, fig 6.6 & 6.8
	MPC	Yes	Yes	Yes	Yes, fig 6.7
High-Gain Observer**	LQR	--	--	Yes	Yes, fig 6.11 &6.12
	PI	--	--	Yes	Yes

Table 6.1: Summary of the results, Yes means it works, No means it does not work, and −− means it is not investigated. **See section 4.3.

		Nominal u=0.12			
		Using subsea measurement and mass rate(P_2 and W_{out})		Using topside measurements(P_2 and W_{out})	
		3% disturbance	5% disturbance	3% disturbance	5% distrubance
EKF Estimate	PI	Yes	--	No	--
	LQR	Yes	No	--	--
	MPC	No	--	--	--
UKF Estimate	PI	Yes	Yes	Yes, fig A.5	No
	LQR	Yes	No	Yes	No, fig 6.9
	MPC	Yes	No	Yes, fig A.6	No

Table 6.2: Summary of the results, Yes means it works, No means it does not work, and −− means it is not investigated.

6.1 Simulation Results

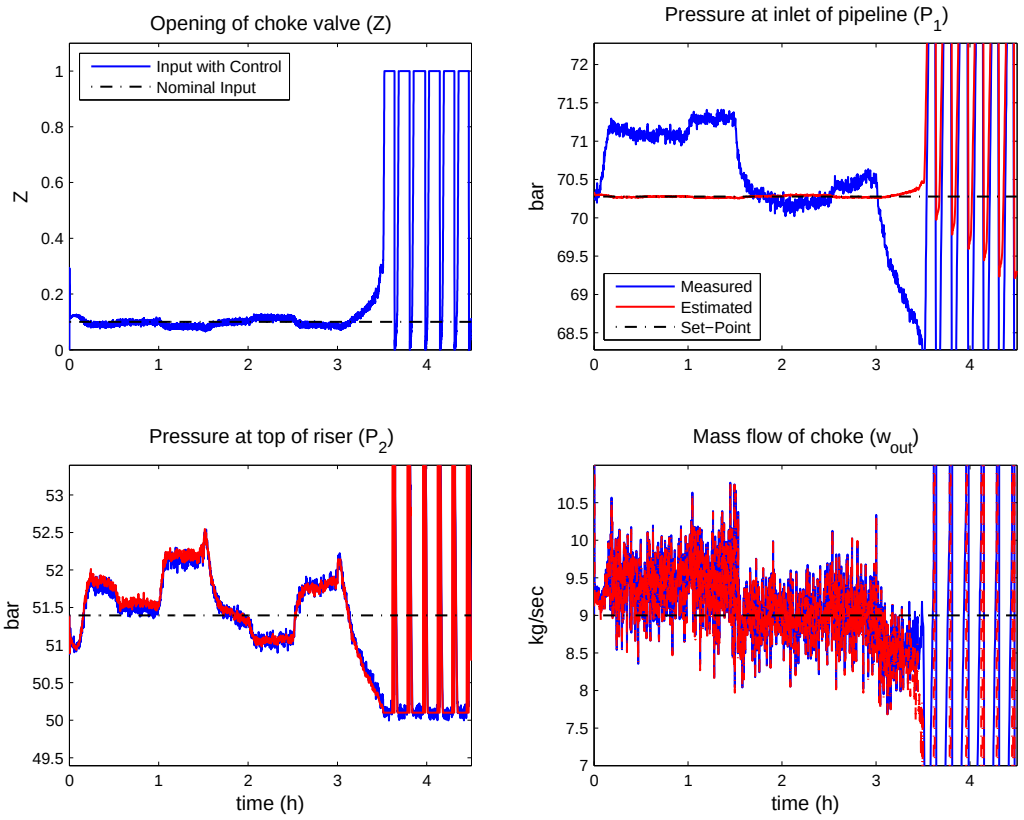

Figure 6.1: PI+EKF, P_2 and W_{out},5%

In Figure 6.1, the system is controlled with a PI controller based on the EKF estimate of $\hat{P}_1$. This system is stable for about 3 hours, but the input is noisy. Further, after about 3.5h, the combination of low $W_{L,in}$ and low $W_{G,in}$ makes the system unstable (Figure 5.1).

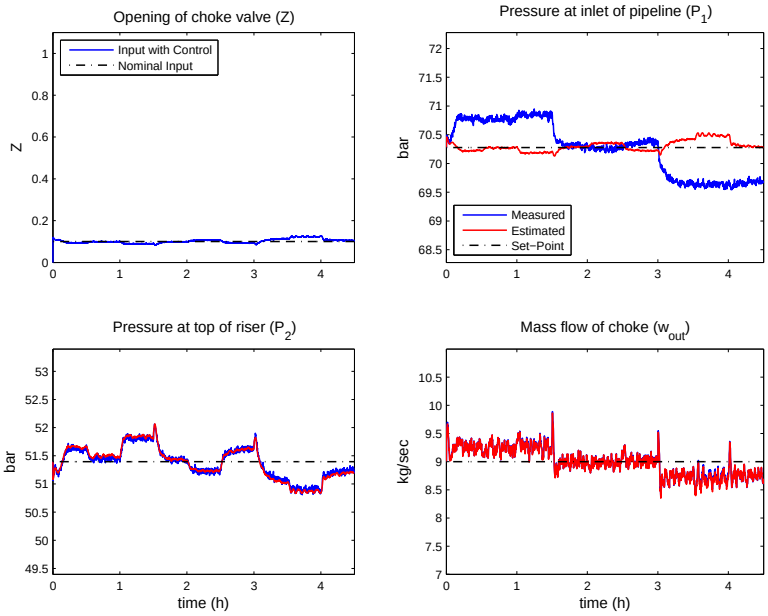

Figure 6.2: PI+EKF, P_2 and W_{out}, 3%

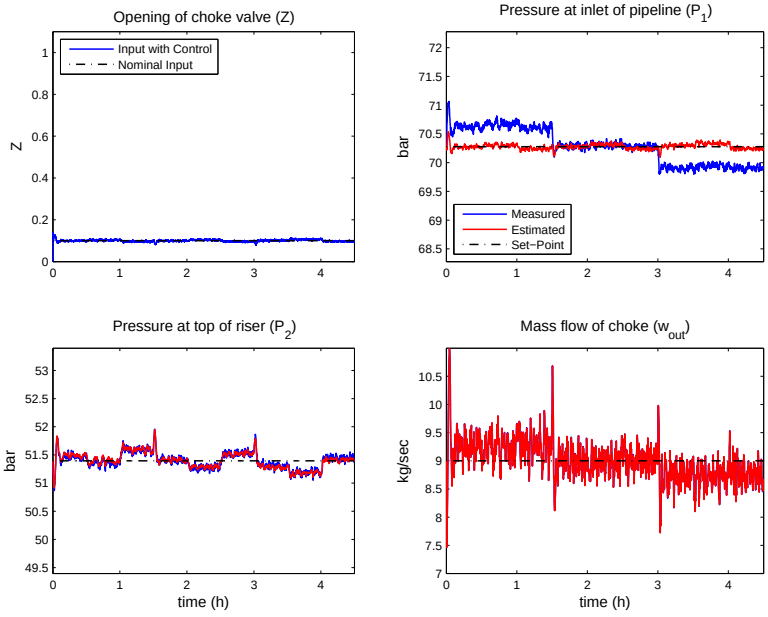

Figure 6.3: PI+UKF, P_2 and W_{out}, 3%

As seen in Table 6.1, the PI controller is the only controller which can stabilize the system with EKF or UKF, with topside measurements and 3% input disturbance. In Figure 6.2 and 6.3 the simulations are shown. Both control solutions give good results.

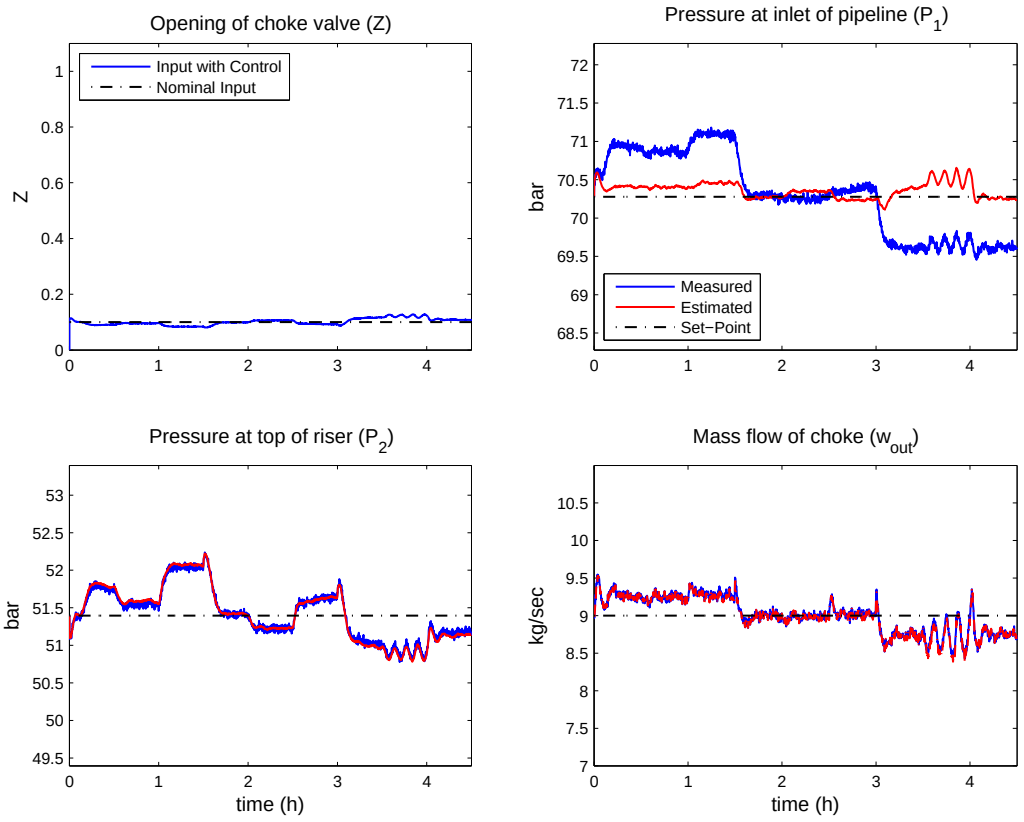

Figure 6.4: LQR+EKF, P_2 and W_{out}, 3%

Small oscillations can be seen after about 3.5h in Figure 6.4. This means that although the input disturbance is reduced to 3%, LQR+EKF gets unstable for a short period. However, the system recover when the combination of low input of $W_{L,in}$ and low $W_{G,in}$ changes to low input of $W_{L,in}$ and high $W_{G,in}$.

Another observation from Table 6.1 is the performance of the UKF. The UKF using only topside measurements, combined with PI, LQR or MPC (Figure 6.5-6.7) stabilizes the system, even with 5% input disturbance.

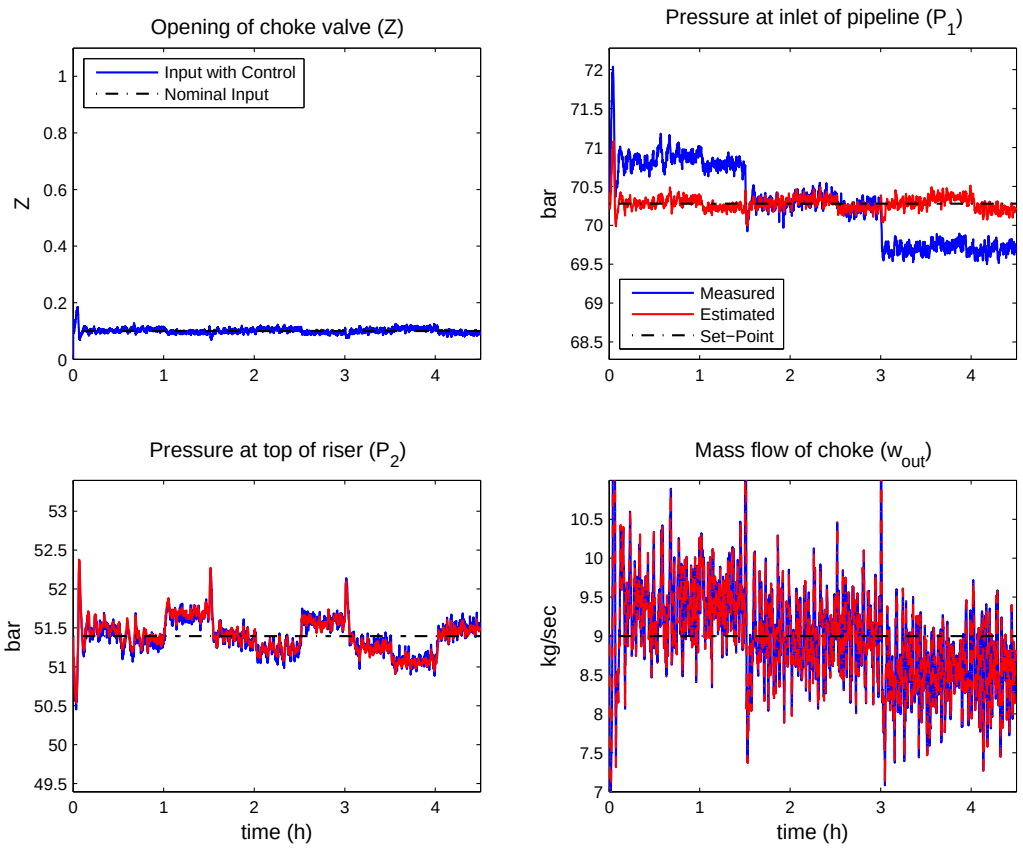

Figure 6.5: PI+UKF, P_2 and W_{out}, 5%

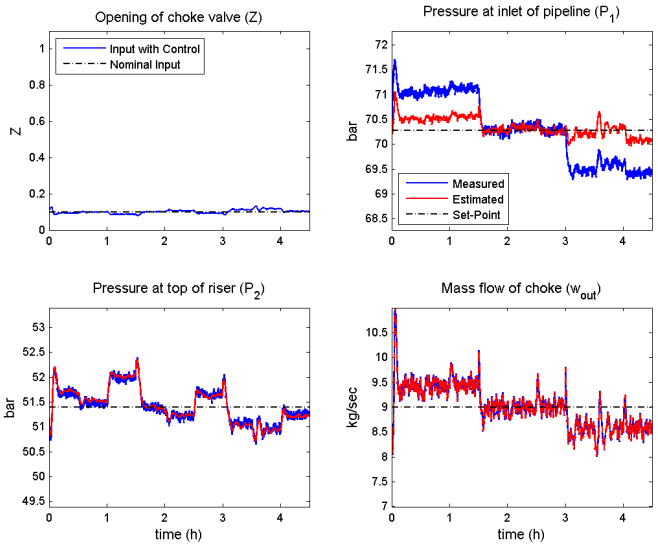

Figure 6.6: LQR+UKF, P_2 and W_{out}, 5%

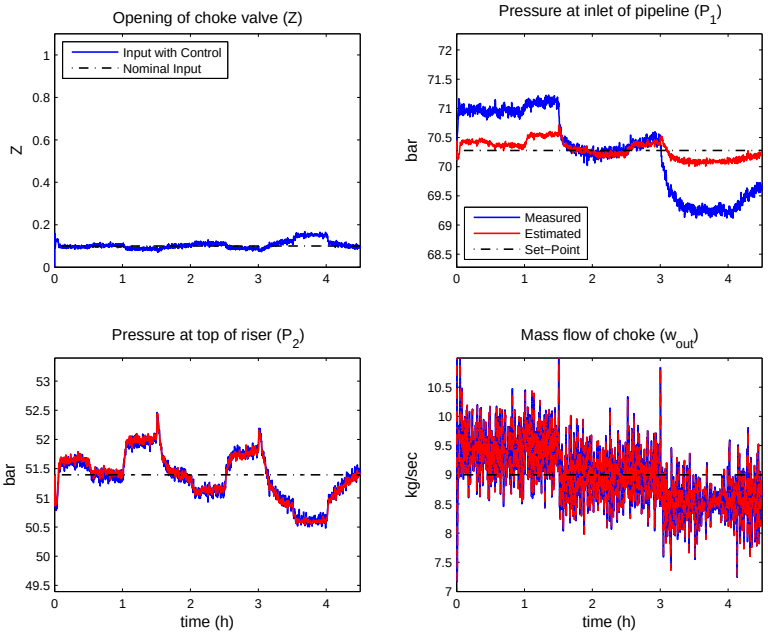

Figure 6.7: MPC+UKF, P_2 and W_{out}, 5%

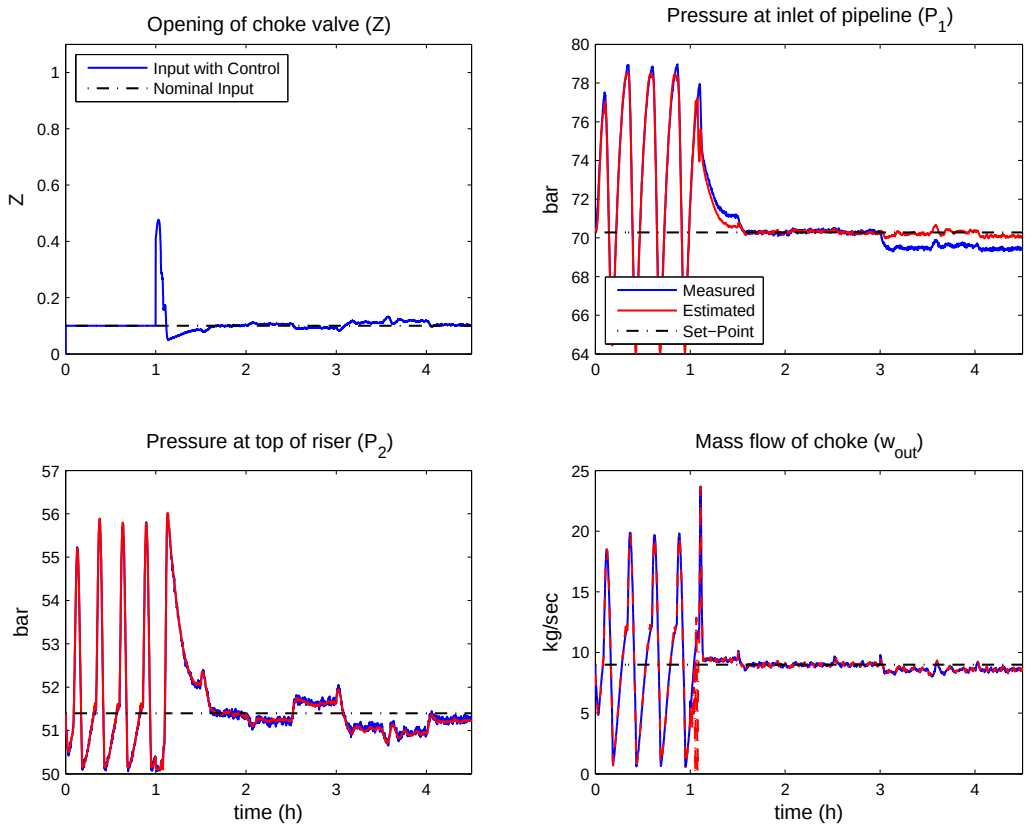

Figure 6.8: LQR+UKF, P_2 and W_{out}, 5%, the control system is turned on after 1 hour[1].

As seen in Figure 6.8, the LQR+UKF using topside measurements, is able to stabilize the unstable slug-flow. The control system is turned on after 1 hour.

[1]Make notice on the different scales compared to figure 6.1-6.7

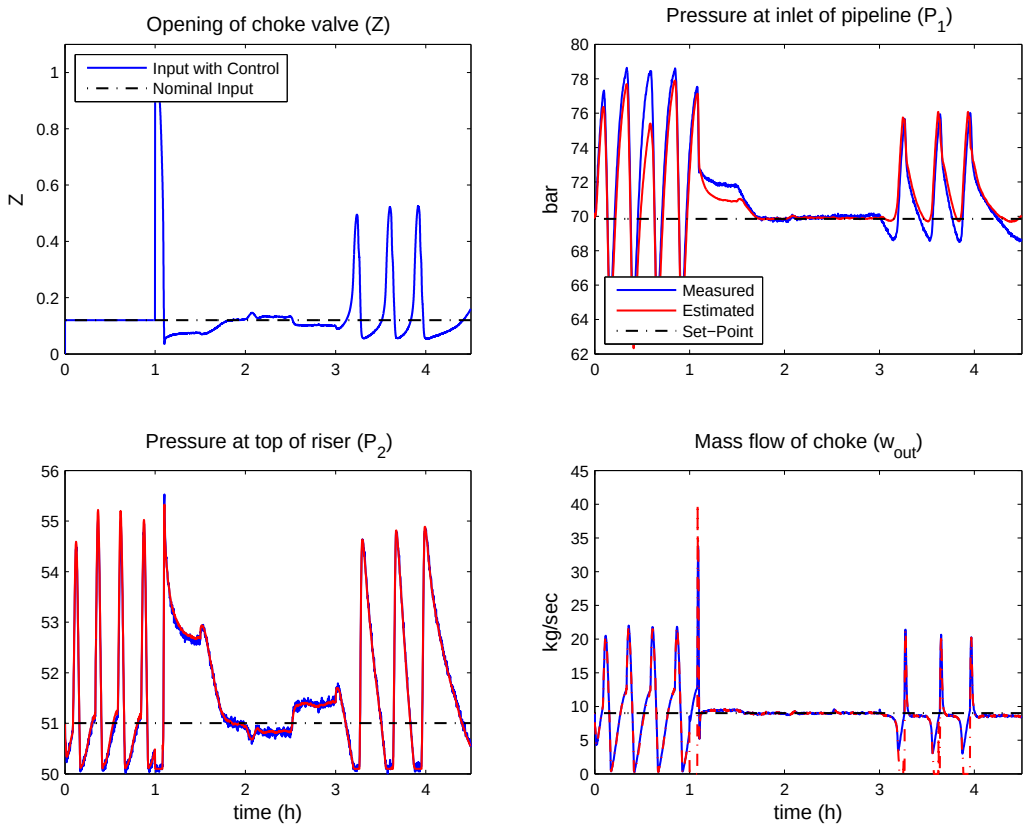

Figure 6.9: LQR+UKF, P_2 and W_{out}, 5%, nominal choke valve $u = 0.12$, the control system is turned on after 1 hour[1].

Further, the system input is changed, and the nominal choke valve position is increased from $u = 0.10$ to $u = 0.12$. In Figure 6.9, it can be seen that the system cannot handle the input combination at 3.5 hours.

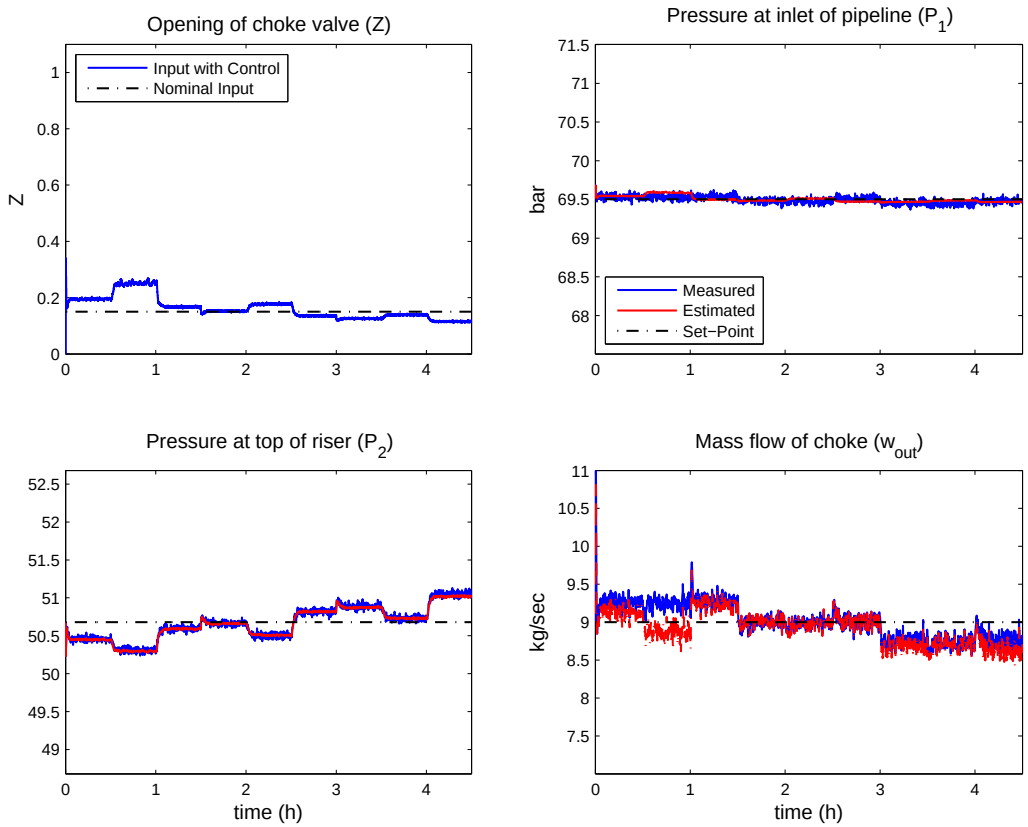

Figure 6.10: PI+EKF, P_1 and W_{out}, 3%, nominal choke valve $u = 0.15$

Moreover, if the subsea pressure and topside flow are used as measurements, the PI+EKF is able to stabilize the the system with nominal choke valve increased to $u = 0.15$.

LQR and High-Gain[2] Observer with 5% Disturbance

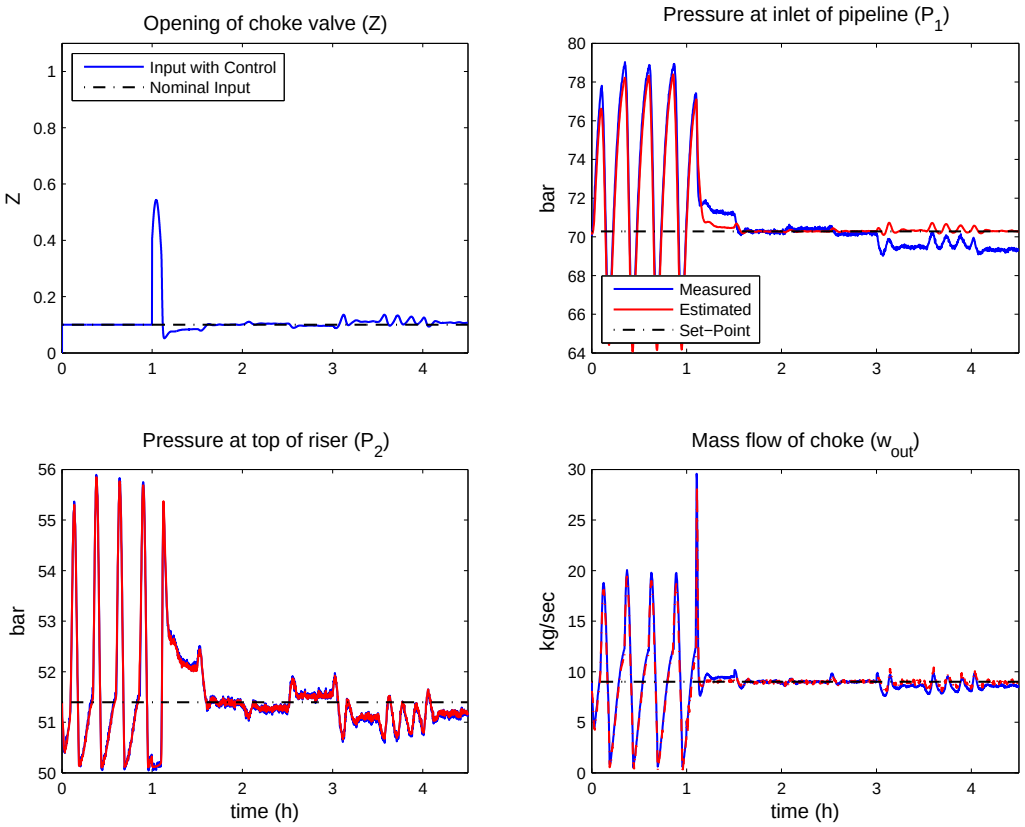

Figure 6.11: High-gain+LQR, P_2, 5% The control system is stared after simulating 1 hour[1].

Promising results are shown in Figure 6.11. The unstable slug flow is stabilized when the control system is started after 1 hour.

[2]In cooperation with supervisor Esmaeil Jahanshahi

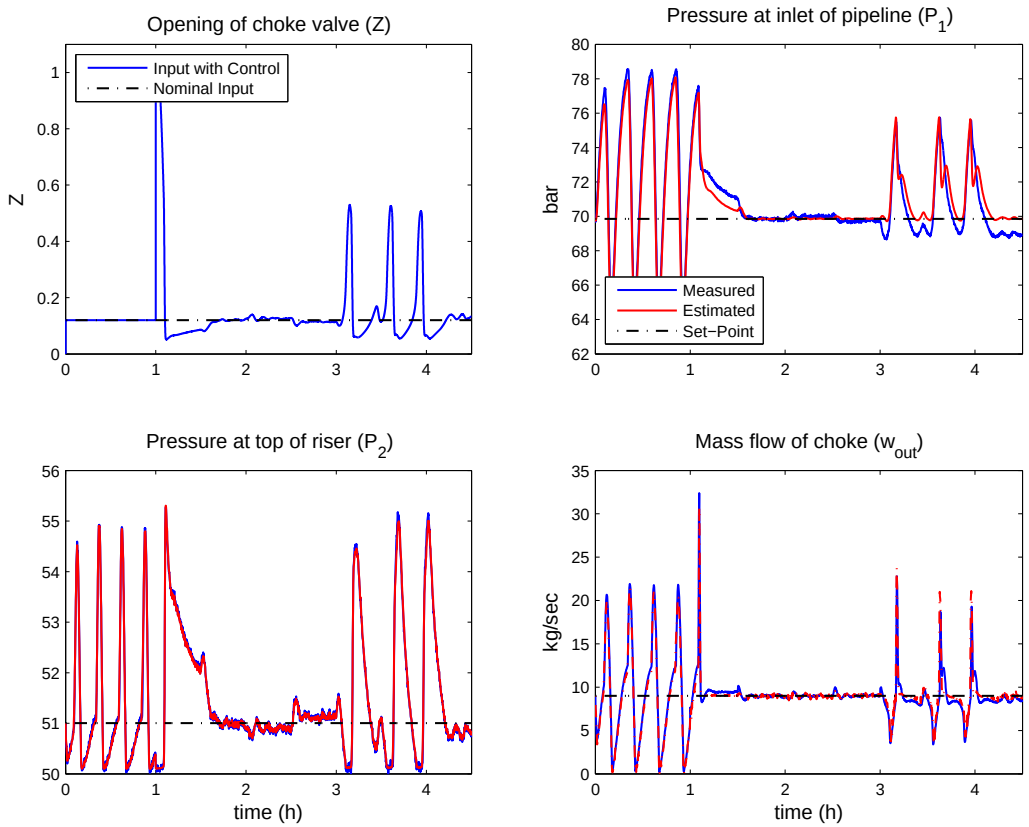

Figure 6.12: High-gain+LQR, P_2 , 5%, nominal choke valve $u = 0.12$. The control system is stared after simulating 1 hour[1].

In Figure 6.12, the high-gain is again simulated with LQR. For this simulation the operating point is changed from $u = 0.10$ to $u = 0.12$. This change makes the system unstable.

6.2 Comparison of the controllers

As seen from Table 6.1, the best nonlinear estimator is the UKF. Comparison is done of the accuracy of the estimators by calculating root mean square error between the estimated state and the state. Then, the root mean square error is calculated to between the output and the steady-state value to check the controller performance. The comparison is done with 5% input disturbance, and topside measurements.

Root Mean Square Estimation Error

$$RMSE = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(x(i) - \hat{x}(i))^2} \tag{6.1}$$

	x_1	x_2	x_3	x_4
UKF+PI	7.59	0.8	2.89	50.13
UKF+LQR	7.09	0.65	2.67	46.25
UKF+MPC	8.16	0.71	3.1	53.96
High-Gain+LQR	10.61	2.15	3.51	61.34

Table 6.3: RMSE for the state estimation

Root Mean Square Output Error

$$Error = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(y(i) - y_{ss})^2} \tag{6.2}$$

	P_1	P_2	W_{out}
UKF+PI	0.509	0.283	1.271
UKF+LQR	0.624	0.323	0.435
UKF+MPC	0.675	0.409	0.544
High-Gain+LQR	0.884	0.479	0.995

Table 6.4: RMSE output error

None of the controllers are significant better than others. However, we can see slightly better results for the UKF+LQR.

Chapter 7

Discussion

Active control of the choke valve is the recommended solution for anti-slug control. In this thesis work state estimation is performed and used in the control solution. The linear Kalman filter, extended Kalman filter (EKF) and unscented Kalman filter (UKF) were used for state estimation, using top-side measurements, with a controller to stabilize the system.

As expected, the linear Kalman filter fails in stabilizing the system when only topside measurements are used, even when the input disturbance is reduced from 5% to 3%. Therefore, we move to nonlinear state estimation.

In Figure 6.1, the PI+EKF is not able to stabilize the system when the input disturbance is 5% and the nominal values of $W_{L,in}$ and $W_{G,in}$ are low. However, when the input disturbance is reduced to 3%, the system is stable. This can be seen in Figure 6.2. The same unstable behavior, with 5% input disturbance, can be seen for LQR+EKF and MPC+EKF, Figure A.3 and A.4 in the appendix. Further, LQR+EKF has less noisy input compared to PI+EKF and MPC+EKF. In contrast to PI+EKF, the PI+UKF can stabilize the system (both with 5% input disturbance). This can be seen in Figure 6.5.

As a conclusion, the EKF is working fine with 3% input disturbance because it is based on first order Taylor series, which gives a good approximation locally (i.e. small disturbance), but for large disturbance it is not working.

The only controller which stabilizes the system with EKF for 3% input disturbance is the PI controller. The two other controllers (LQR and MPC) where not able to stabilize the system in this case. For comparison, PI+EKF and PI+UKF are shown in Figure 6.2 and 6.3. They have much the same perfor-

mance, but the estimated P_1 and P_2 are closer to the set-point for the UKF. Further, W_{out} has less noise for EKF. In Figure 6.4, the LQR+EKF controller has small oscillations after about 3.5 hours which recovers when the amount of gas increases in the input to the pipeline. Because of the oscillations, the controller is not very robust, and is marked as *No* in Table 6.1.

The UKF is the filter which works in most cases, as seen in Table 6.1. The performance of the UKF is improved with the implemented forgetting factor less than one. By having the forgetting factor less than one, the covariance is artificially increased and the observer gain K is increased. But with large observer gain, the system becomes sensitive to noise. In Figure 6.5-6.7, the different controllers can be compared. The PI and MPC controller have some noise on choke valve opening, however, a solution to this can be to implement a low-pass filter. Moreover, the UKF combined with LQR is able to stabilize the unstable system when the controller is turned on after one hour, as seen in Figure 6.8. But, when the nominal choke valve opening is increased from $u = 0.10$ to $u = 0.12$ the system gets unstable because of the input disturbances, as seen in Figure 6.9. The Q and R for LQR tuning is set so the R is much higher than Q, which gives a slow controller response.

As a result, we can conclude that since the UKF is accurate to the third order, it can cover more nonlinearity when large disturbances occur. With $ff < 1$, we increased observer gain and much faster estimation makes the separation principle better, since the observer is much faster than the controller. However, the system is sensitive to noise and it is not very robust, since an increase in the nominal choke valve opening makes the system unstable.

Moreover, to check the robustness of the different control systems, simulations are done with $u = 0.12$, as seen in Table 6.2. In Figure A.5 and A.6, the PI+UKF and MPC+UKF, with 3% input disturbance, nominal choke valve opening $u = 0.12$ and topside measurements are simulated. The PI+UKF is only marginally stable, while MPC+UKF is stable. Both the PI+UKF and MPC+UKF fails if the input disturbance is increased to 5% while the choke opening is remained on $u = 0.12$.

As discussed earlier, the idea for implementing MPC was to have efficient constraint handling. One important characteristic of the choke valve is that it cannot be more than fully open. This means that there is a constraint in the system $0 < Z < 1$. However, the system is not close to have the choke fully open when the system is stable. Moreover, we tried tightening the bounds on ΔZ, the change in the choke valve, but without improved results. Further, for MPC tuning the R was tuned much higher than Q. This means that it is "costly" to use the input, and we will get a slow controller. This is the same

strategy as for the LQR tuning. Further, for simplicity, the control horizon and prediction horizon, were set to be equal each other hence, $H_u = 15$ and $H_p = 15$. The value was found as a compromise between system performance and simulation time.

As seen in Table 6.1 and 6.2, there is no trouble with controlling the system when the subsea pressure is measured. This is as expected from the controllability analysis done of the model [3]. Moreover, in Figure 6.10, it can be seen that the PI+EKF is able to stabilize the system with $u = 0.15$. This results in a higher choke valve opening, which might result in a more profitable production. Higher values than $u = 0.15$ is tested for PI+EKF, but without stable results.

The high-gain observer in Figure 6.11, stabilizes the unstable flow when topside pressure is measured. The estimated output is close to the set-point, and shows good performance. There is room for further research on this observer in anti-slug control solutions. Moreover, as for the LQR+UKF, when the nominal value of the choke opening is increased to $u = 0.12$, the high-gain+LQR with 5% input disturbance gets unstable, as seen in Figure 6.12.

In Table 6.3 the accuracy of the estimator is calculated, and in Table 6.4 the performance of the controller is calculated. Both are calculated by the root mean square error method. The LQR+UKF is slightly better than the other solutions. Further, high-gain + LQR shows the highest values, which is because of its sensitivity to noise.

7.1 Challenges

There have been quite a few challenges throughout this master work. First, the choice of solver and its operation point. In the beginning, the "fixed step solver" (FSS) implemented by my supervisor was used. This is a faster solver compared to *ode15s*. The reason that we switched to the *ode15s* solver was that when the simulations did not work in the beginning, we were not sure if something was wrong with the control strategy, or with the solver. The linear Kalman filter implementation in MATLAB was not working either, so Simulink implementation was tested with success.

Further, the tuning was a challenge. Simulations took up to 40 minutes each, so tuning estimators and controllers was very time consuming. Time was also spent on deciding which MPC formulation that will perform the

best in this case and of course the implementation. There are a lot of tuning parameters for the UKF. Without luck, a lot of effort was made in tuning the UKF parameters, when topside measurements were used, to get good UKF performance with $ff = 1$.

Chapter 8

Conclusion

There has been done research on anti-slug control with nonlinear state estimation in this thesis. From the controllability analysis of the L-shaped pipeline-riser model it is concluded that the subsea pressure measurement and flow measurement are the best measurements for active control of the choke valve. However, since the subsea measurement is not always available, the topside pressure and flow measurements are used in nonlinear state estimation to estimate states and other control variables.

Simulation studies of different control strategies have shown that the EKF works good locally, while the UKF with forgetting factor works best for high input disturbance, hence it is more robust. The LQR controller has slightly better performance compared to the MPC and PI. However, this might be a result of different tuning. Further, to use the high-gain observer in control has a great potential, and needs further research. The UKF+LQR and high-gain+LQR are stable, with measurements of topside pressure and mass rate with 5% input disturbance, for $u = 0.10$ but for $u = 0.12$ the control solution is unstable, which gives us the conclusion that they are not robust.

Further work for anti-slug control with nonlinear state estimation is more investigation on control strategies with high-gain observer. Another possibility can be to implement an MPC which tries to maximize the choke valve opening, while still make sure that the choke valve is not more than fully open. This can be a profitable solution because the larger choke opening, the more is produced. Further, for noisy choke valve opening, a low-pass filter can be tried out.

Bibliography

[1] Ove Bratland. *Multi-phase Flow Assurance.* 2010.

[2] Esmaeil Jahanshahi and Skogestad Sigurd. Simplified Dynamical Models for Control of Severe Slugging in Multiphase Risers. 2011.

[3] Anette H.Helgesen, Jahanshahi Esmaeil, and Sigurd Skogestad. Controllability Analysis of Severe Slugging in Well-Pipeline-Riser Systems. 2012.

[4] Anette Hoel Helgesen. Anti-slug Control of Two-Phase Flow in Risers with:Controllability Analysis Using Alternative Measurements. Master's thesis, NTNU, 2010.

[5] K. Havre, K. Stornes and Stray H. Taming Slug Flow in Pipelines. *ABB review*, pages 55–63, 2000.

[6] Espen Storkaas. *Stabilizing Control and Controllability: Control Solutions to Avoid Slug Flow in Pipeline-Riser Systems.* 2005.

[7] Cem Sarica and Jarl Ø. Tengesdal. A New Technique to Eliminate Severe Slugging in Pipeline/Riser Systems. *SPE Annual Technical Conference and Exhibition, 1-4 October 2000, Dallas, Texas*, 2000.

[8] John-Morten Godhavn, Stig Strand, and Gunleiv Skofteland. Increased Oil Production by Advanced Control of Receiving Facilities. *Statoil R&D Process Control*, 2005.

[9] Sigurd Skogestad. *Chemical and Energy Process Engineering.* CRC Press, 2009.

[10] João Hespanha. *Linear Systems Theory.* Princeton University Press, 2009.

[11] Dan J. Simon. *Optimal State Estimation.* John Wiley & Sons, Inc, 2006.

[12] Chi-Tsong Chen. *Linear Systems Theory and Design.* Oxford University Press, 1999.

[13] S.J. Julier and J.K. Uhlmann. Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE*, 92(3):401 – 422, mar 2004.

[14] S.S. Haykin. *Adaptive Filter Theory.* Prentice-Hall information and system sciences series. Prentice Hall, 2002.

[15] H.K. Khalil. *Nonlinear Systems.* Macmillan Pub. Co., 1992.

[16] F. Di Meglio, G.-O. Kaasa, N. Petit, and V. Alstad. Reproducing Slugging Oscillations of a Real Oil Well. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 4473 –4479, dec. 2010.

[17] Sigurd Skogestad. Simple Analytic Rules for Model Reduction and PID Controller Tuning. *Modeling, Identification and Control*, 25(2):85–120, 2004.

[18] S. Skogestad and I. Postlethwaite. *Multivariable Feedback Control: Analysis and Design.* John Wiley, 2005.

[19] Lars Imsland. Introduction to Model Predictive Control.

[20] J.M. Maciejowski. *Predictive Control: with Constraints.* Pearson Education. Prentice Hall, 2002.

[21] Matlab, www.mathworks.se, 2012.

[22] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization.* Springer, 2000.

Appendix A

Appendix

A.1 Attached CD

The attached CD contains the files to run the simulations. The folder "P1 and W" means subsea pressure and mass rate as sensors, while "P2 and W" means topside pressure and mass rate as sensors. To simulate with 5% input disturbance load wG_wL_nm, while simulating with 3% input disturbance requires loading $wG\_wL\_nm3percent$.

A.2 Simulink Diagram

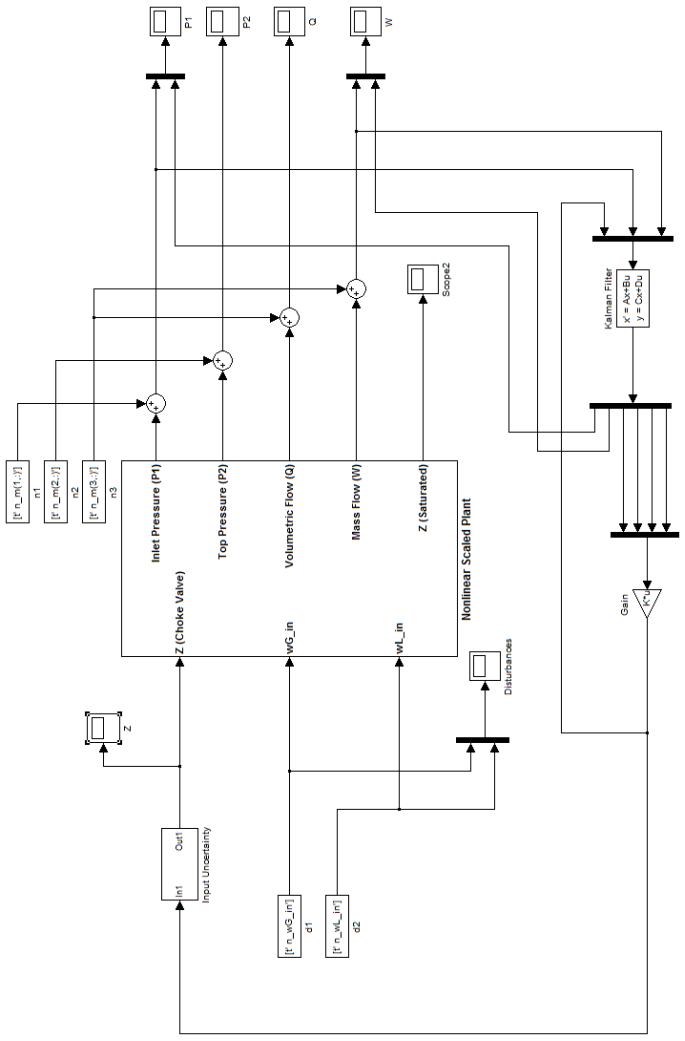

Figure A.1: Simulink Diagram for LQG

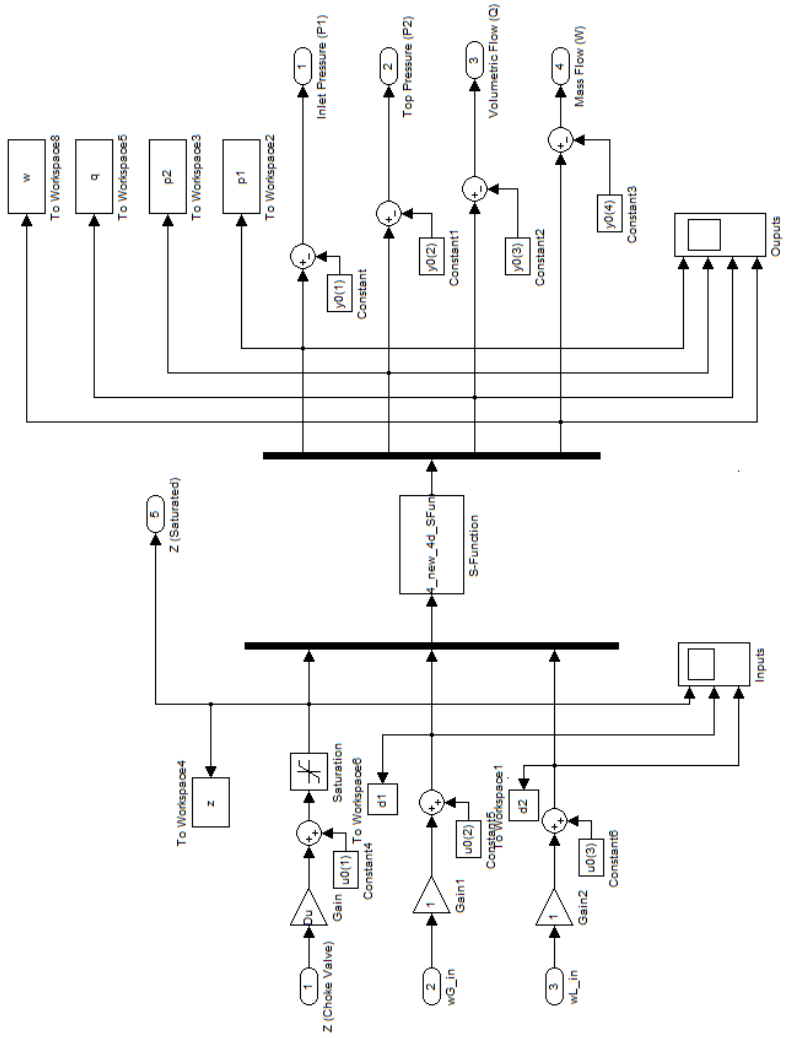

Figure A.2: Nonlinear Plant

A.3 Figures from the Result Section

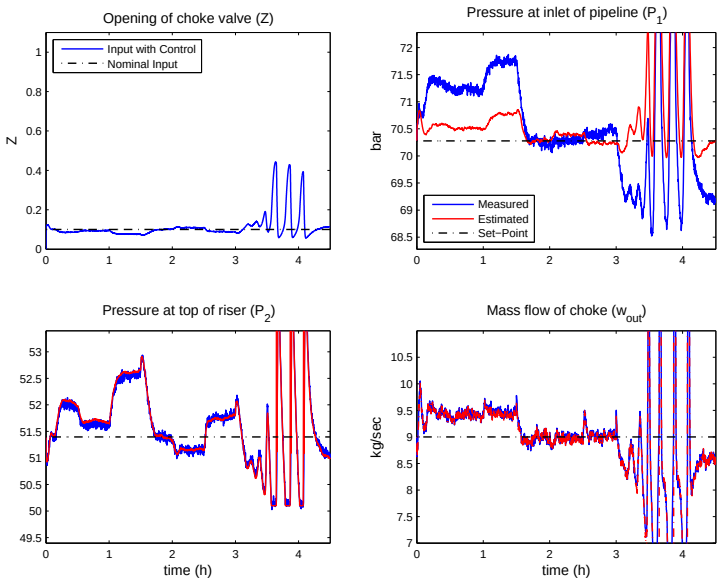

Figure A.3: LQR+EKF, P_2 and W_{out}, 5%

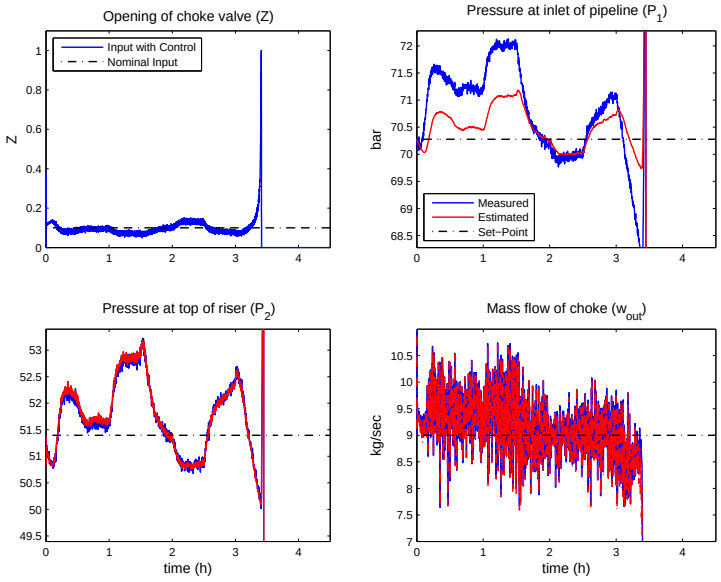

Figure A.4: MPC+EKF, P_2 and W_{out}, 5%

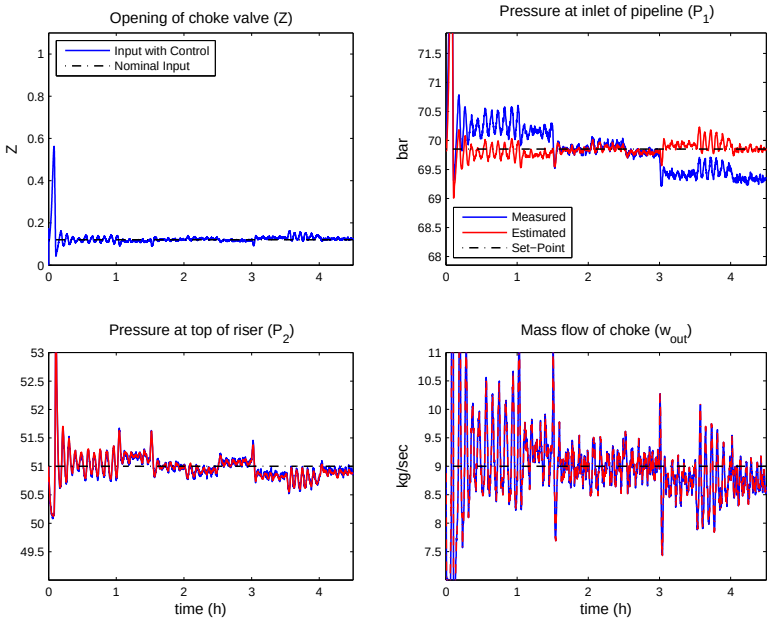

Figure A.5: PI+UKF, P_2 and W_{out}, 3%, $u = 0.12$

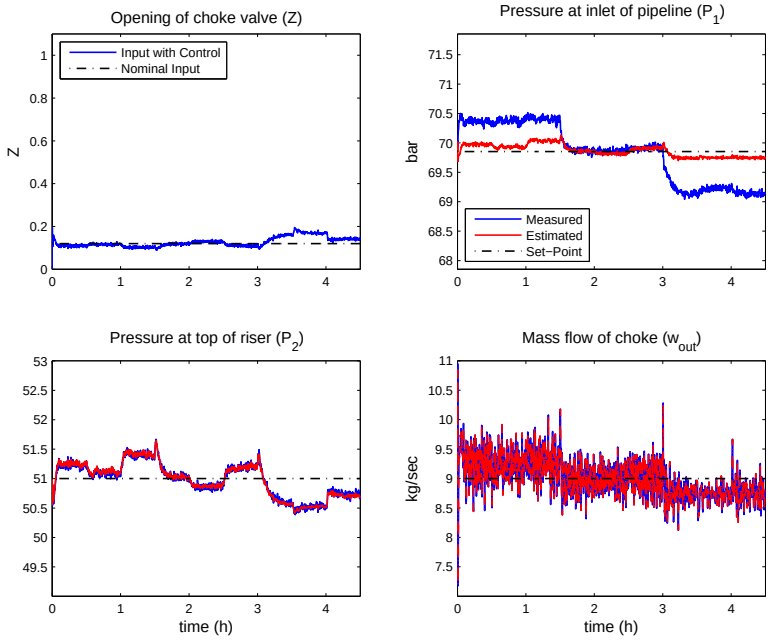

Figure A.6: MPC+UKF, P_2 and W_{out}, 3% , $u = 0.12$

A.4 MATLAB Code

MPC initialization

```
%% Initialize MPC

%%Dimensions

Hu=15;  %Control horizon
Hp=Hu;  %Prediction horizon
Hw=1;   %When you start controlling


%Table 3.1 in Predictive Control with constraints by Maciejowski
l=1; %inputs
n=4; %States
m=4; %Controlled outputs ,z=cx, c=eye(4);
Q_MPC=zeros(m*(Hp-Hw+1),m*(Hp-Hw+1));
R_MPC=zeros(l*Hu,l*Hu);
Psi=zeros(m*(Hp-Hw+1),n);
Gamma=zeros(m*(Hp-Hw+1),l);
Theta=zeros(m*(Hp-Hw+1),l*Hu);


%Initialize constraints matrix
Ff    = zeros(2*Hu,Hu);
%f1    = zeros(2*Hu,1);
f     = zeros(2*Hu,1);
Gcons = zeros(2*Hu,Hu);
g1    = zeros(2*Hu,1);
%g     = zeros(2*Hu,1);
W     = zeros(2*Hu,Hu);
w     = zeros(2*Hu,1);

%Weighting on the different states and input.
Q_MPCd=1*diag([100 1 1 1]);
R_MPCd=1e3;
Q_MPC=blkdiag2(Q_MPCd,Hu);
R_MPC=blkdiag2(R_MPCd,Hu);

%% Formulation of constraints

bias=u(1,1);
uMax=1;  %uMin<u<uMax
uMin=0; %

uMax=uMax-bias;
uMin=uMin-bias;
```

```
violation=0.5;
zMax1=violation*x0(1);
zMin1=-violation*x0(1);
zMax2=violation*x0(2);
zMin2=-violation*x0(2);
zMax3=violation*x0(3);
zMin3=-violation*x0(3);
zMax4=violation*x0(4);
zMin4=-violation*x0(4);
uRateMax=1e-1;
uRateMin=-1e-1;


%% Ff is 1. line on left side of constraint (P)
for j = 1:Hu,
    f(2*(j-1)+1,1) = -uMax;
    f(2*j,1)       = uMin;
    for i = j:Hu,
        Ff(2*i-1,j)       = 1;
        Ff(2*i,(j-1)+1:j) = -1;
    end
end

f1    = Ff(1:2*Hu,1);

constZ=[-zMax1; zMin1;-zMax2;zMin2;-zMax3;zMin3;-zMax4;zMin4];
g=repmat(constZ,Hu,1);

for i = 1:Hu*m,
    Gcons(2*i-1,i) = 1;
    Gcons(2*i,i)   = -1;
end

%% W and w matrices in constraints
%% W is 3. line on left side of constraint
for i = 1:Hu,
    w(2*i-1,1) = uRateMax;
    w(2*i,1)   = -uRateMin;
    W(2*i-1,i) = 1;
    W(2*i,i)   = -1;
end

x = zeros(4,nf+1);
x(:,1) = 0.001*x0;
y = zeros(size(y0,1),nf);
y(:,1) = y0;
u_in=zeros(3,nf);
```

Implementation of High-Gain Observer

```
u_in(:,k) = u(:,k-1)+[u_c(k-1);0;0];
[~,zt]=ode15s(@v1_new_4d_Observer,[k-1;k],z_e(:,k-1),options,
        u_in(:,k),y(sensors,k),'derivatives',par,ep);
z_e(:,k) = zt(end,:)';
x_e(:,k) = v1_new_4d_Observer(t,z_e(:,k),u_in(:,k),
        y(sensors,k),'measurements',par,ep);
y_k = v4_new_4d_model(t(k),x_e(:,k),u_in(:,k),'measurements',par);
y_e(:,k)=y_k(1:3);
```

function *v1_new_4d_Observer*

```
function [sys] = v1_new_4d_Observer(t,x_hat,u,y_m,output,par,ep)

% Observer based on the 4-state pipeline-riser model
% By: Esmaeil Jahanshahi
% May 2012, NTNU, Norway


% x1_hat: Mass of gas in the pipeline (m_G1)
% x2_hat: Mass of liquid in the pipeline (m_L1)
% x3_hat: Mass of gas in the riser (m_G2)
% x4_hat: Mass of liquid in the riser (m_L2)


[zdot,xhat] = v1_new_4d_Observer0(x_hat,u,y_m,par,ep);

if isequal(output,'derivatives')
    sys =zdot;
    if ~isreal(sys)
        disp('Complex')
        sys=0*sys;
    end
elseif isequal(output,'measurements')
    sys = xhat;
end

end
```

function *v1_new_4d_Observer0*

```
function [zdot,xhat] = v1_new_4d_Observer0(z,u,y_m,par,ep)

% Observer based on the 4-state pipeline-riser model
% By: Esmaeil Jahanshahi
% May 2012, NTNU, Norway
```

```
% z1: Mass of gas in the pipeline (m_G1)
% z2: Mass of liquid in the pipeline (m_L1)
% z3: Pressure at top of riser (P_rt)
% z4: Mass of liquid in the riser (m_L2)

x1 = z(1);
x2 = z(2);
P2_t = z(3);
x4 = z(4);

u1 = u(1);
w_G_in = u(2);
w_L_in = u(3);

a = par.R*par.T2*par.rho_L/par.M_G;
b = par.rho_L*par.V2;


% rho_G1_norm = par.P1_norm*par.M_G/(par.R*par.T1);
% Alpha_L1_av = w_L_in*rho_G1_norm/(w_L_in*rho_G1_norm +
          w_G_in*par.rho_L);
h1ss = par.k_h*par.Alpha_L1_av*par.hc;
x2ss = par.V1*par.rho_L*par.Alpha_L1_av;
h1 = h1ss + sin(par.theta)*(x2 - x2ss)/(par.A1*
        (1-par.Alpha_L1_av)*par.rho_L);
h1 = max(h1,0);

V_G2 = par.V2 - x4/par.rho_L;


P1 = x1*par.R*par.T1/(par.M_G*(par.V1 - x2/par.rho_L));
rho_G1 = x1/(par.V1 - x2/par.rho_L);

Uslin = w_L_in/(par.A1*par.rho_L);
Re1=par.rho_L*Uslin*(2*par.r1)/par.visl;
Lambda1=0.0056 + 0.5*Re1^(-0.32);
Fric_pipe=0.5*par.Alpha_L1_av*Lambda1*par.rho_L*Uslin^2
        *par.L1/(2*par.r1);


%P2_t = x3*par.R*par.T2/(par.M_G*V_G2);
%P2_t = max(P2_t,par.P0);
x3 = P2_t*(b - x4)/a;

rho_G2 = x3/V_G2;
Alpha_L2_av = x4/(par.rho_L*par.V2);
rho_mix_av = (x3+x4)/par.V2;
```

```
Usl2 = w_L_in/(par.rho_L*par.A2);
Usg2 = w_G_in/(rho_G2*par.A2);
Um = Usl2+Usg2;

Re2=rho_mix_av*Um*(2*par.r2)/par.visl;
Lambda2=0.0056 + 0.5*Re2^(-0.32);
Fric_riser=0.5*Alpha_L2_av*Lambda2*rho_mix_av*Um^2*
        (par.L2+par.L3)/(2*par.r2);


A_g = (h1<par.hc)*(par.A1/par.hc^2)*(par.hc - h1)^2;

A_l = par.A1 - A_g;

P2_b = P2_t + rho_mix_av*par.g*par.L2 + Fric_riser;

w_G1 = par.K_g*A_g*sqrt(rho_G1*max(0,P1-Fric_pipe-P2_b));
w_L1 = par.K_o*A_l*sqrt(par.rho_L*max(0,P1-Fric_pipe+
        par.rho_L*par.g*h1-P2_b));

Alpha_Lb = 1 -  A_g/par.A1;

if(Alpha_Lb<=Alpha_L2_av)
    Alpha_Lb=Alpha_L2_av;
end

Alpha_Lt =  2*Alpha_L2_av - Alpha_Lb;


if(Alpha_Lt>Alpha_L2_av)
    Alpha_Lt = Alpha_L2_av;
elseif(Alpha_Lt<0)
    Alpha_Lt = 0;
end


Alpha_Lmt = Alpha_Lt*par.rho_L/(Alpha_Lt*par.rho_L +
        (1-Alpha_Lt)*rho_G2);
rho_t = Alpha_Lt*par.rho_L + (1-Alpha_Lt)*rho_G2;

% if(z>0.95)
%     par.Cd = 0.95;
% end

%ORF = abs(1/(z^2*par.Cd^2) -1);
%w_mix_out = par.A2*sqrt(2*rho_t*max(0,P2_t-par.P0)/ORF);
w_mix_out = par.K_pc*u1*sqrt(rho_t*max(0,P2_t-par.P0));
%Q_out = w_mix_out/rho_t;
```

```
w_L_out = Alpha_Lmt*w_mix_out;
w_G_out = (1 - Alpha_Lmt)*w_mix_out;


dx1 = w_G_in - w_G1;
dx2 = w_L_in - w_L1;
dx3 = w_G1 - w_G_out;
dx4 = w_L1 - w_L_out;

y_hat = P2_t/1e5; % Unit conversion from Pa to Bar

dz1 = dx1;
dz2 = dx2;
dz4 = dx4;
dz3 = (a*(b-x4)*dx3 + a*x3*dz4)/((b-x4)^2)+ (1/ep)
        *(y_m-y_hat);

zdot = [dz1;dz2;dz3;dz4];

xhat = [x1; x2; x3; x4];
end
```