NTNU
Norwegian University of
Science and Technology

Inculding Prices in Control Structure Design

Applied on a Subsea Separation System

Åge Johansen

Chemical Engineering and Biotechnology
Submission date: July 2016
Supervisor: Johannes Jäschke, IKP

Norwegian University of Science and Technology
Department of Chemical Engineering

Summary

The selection of controlled variables and measurements is an important aspect of the control structure design. Using the self-optimizing control framework makes it possible to select controlled variables as combinations of measurements which minimizes the steady state loss. However, in a broader perspective, one is not only interested in the steady state loss, but also in the chemical plant's economy. This work has combined the steady state loss with prices of measurements using a Mixed Integer Quadratic Programming (MIQP)-formulation, so that the optimal subset of measurements also results in the overall best economics for the process plant. In this way the control structure design could be implemented as part of the process design phase, to make the plant as profitable as possible. This work also utilized the power of the MIQP-formulations to include wider selection criteria, which made it possible to select different measurement devices with a variety of prices and uncertainty. From this it was possible to calculate the best trade-off between prices and losses due to measurement uncertainty when different measuring devices are available to a project. Normal process plants also handles constraints, which also need to be controlled. The constraints also have a corresponding loss - also called back-off (due to measurement uncertainty) related them. This has also been included in the total cost calculation, and evaluated both within an active constraint region and an unconstrained region. The developed methods have been tested and evaluated on a Dummy problem and a Subsea separation system.

Sammendrag

Det å velge hvilke tilstander man skal regulere, og hvilket måleutstyr man skal installere er en viktig del av i utviklingen av et prosessanleggs reguleringsystem. Bestemmelsen av disse tilstandene er utgangspunktet for rammeverket «Self Optimizing Control» som kan brukes (blant annet) til å bestemme hvilket tap et bestemt utvalg av kontrollerte variable har på et system i stasjonær tilstand. Som ingeniører er vi ikke bare interessert i å regne på tapet et reguleringsystem har, men også hvilken kostnad det medbringer. I denne oppgaven kombineres tapet i stasjonær tilstand, med priser på måleutstyr for å finne den optimale sammensetningen av hvilke og hvor mange målinger et system burde ha for å minimere anleggets totale kostnader. For å få til dette har systemet blitt satt opp i et «Mixed Integer Quadratic Programming» (MIQP) optimaliserings program. Gjennom dette har det også vært mulig å inkludere andre utvelgingskriterer slik at det er mulig å velge den beste kombinasjonen av nøyaktige (men dyre) og billige (men unøyaktige) målinger ut fra hvilken kombinasjon som gir den laveste totale kostnaden for reguleringsystemet. Et annet viktig aspekt i regulering av prosessanlegg er hvordan man skal regulere begrensninger. Slike begrensninger vil også komme med et assosiert tap (grunnet måleusikkerhet). Slike tap har også blitt inkludert i beregningen av den totale kostnaden til reguleringsystemet, og blitt evaluert både i et begrenset og ubegrenset område. For å teste metodene har både et enkelt «Dummy»-problem og et undervanns separasjonsystem blitt benyttet.

Preface

This master thesis was completed during the spring semester of 2016, and is the final part of my integrated Master of Science degree in Chemical Engineering at the Norwegian University of Science and Technology.
I would sincerely thank my supervisor Associate Professor Johannes Jäschke for coming up with the ideas which have led to this thesis. He has been very patient through numerous hours trying to learn a poor chemical engineering student optimization, which actually turns out to be NP-hard. However, even these problems may be solvable, this semester has therefore been an eyeopener into this interesting field of study, and one could easily say Johannes has expanded the feasible region of my knowledge.
I would also thank my class mates in the study hall who have worked as a maximum productivity constraint, with a high shadow price. However, this constraint has been needed in order to avoid the infeasible (and possibly non-convex) region called Social anxiety.
Finally I would like thank my girlfriend Mari for her constant support. Her ability to handle my nagging and complaints about non-converging MATLAB script has a impressively threshold limit, and still maximizes my well-being.

Declaration of Compliance

I hereby declare this thesis as an independent work in agreement with the exam rules, and regulations of the Norwegian University of Science and Technology.

Trondheim, 31. July 2016

[illegible]

Åge Johansen

Table of Contents

List of Tables

List of Figures

List of Symbols

Abbreviations

CV	Controlled variables
DO	Deoiler
DW	Dewaterer
HPO	Heavy phase outlet
KKT	Karush–Kuhn–Tucker conditions
LPO	Light phase outlet
MIQP	Mixed interger quadratic programming
SOS	Structured Ordered Set

Number Sets

$\mathbb{R}$	Real Numbers

Latin Letters

$\bar{p}_{j,k}$	Price of measurement $_j$ in device set $_k$	\$/s
$\mathbb{L}$	Lagrangian function	
A	In MIQP: Linear constraints matrix	
b	In MIQP: Linear constraint vector	
C	Number of possible combinations	
c	Selected controlled variables	
c_s	Set point to controller	

d	Disturbance	
d'	Scaling factor for disturbances	
E_{oil}	Earnings of oil	\$/barrel
F	Sensitivity matrix, derivative of y^{opt} with respect to d	
$f(x)$	Cost/Objective function in a general optimization problem	
$g(x)$	Inequality constraints in a general optimization problem	
G^y	Gain matrix, derivative of y with respect to u	
G^y_d	Gain matrix of disturbances, derivative of y with respect to d	
G^y_δ	In MIQP: Gain matrix restructured	
H	Selection matrix	
$h(x)$	Equality constraints in a general optimization problem	
H_δ	Selection matrix H restructured as a vector	
J	Cost/Objective function	\$/s
J^{opt}	Optimum point for cost function	\$/s
J_i	Derivative of the cost function with respect to i	
j_δ	Hessian matrix J_{uu} restructured as a vector	
J_{ij}	Double derivative of cost function with respect to i and j	
L	Loss	\$/s
l	In MIQP: Lower bounds of x	
L^{cons}	Constrained loss	\$/s
L^{total}	Total loss, constrained and unconstrained	\$/s
L^{uncons}	Unconstrained loss	\$/s
L_{av}	Average loss	\$/s
L_{worst}	Worst case loss	\$/s
M	Local loss matrix	
n'	Scaling factor for measurement noise	
n^y	Measurement noise/uncertainty	
n_c	Number of controllers	

n_d	Number of disturbances	
n_m	Number of measurements in the subset	
n_u	Number of inputs	
n_w	Number of measurement device sets	
n_y	Number of measurements	
n_{cons}	Number of constraints	
P	Vector with length n_y	
p_{water}	Price of residual water processing	\$/barrel
Q	In MIQP: Symmetric objective matrix	
s	In MIQP: Number of measurements in the subset	
t	Time	s
u	In MIQP: Upper bounds of x	
u	Input	
W_d	Scaled magnitude of disturbances	
W_n	Scaled magnitude of measurement noise	
x	In MIQP: objective vector	
x	State	
Y	Argumented sensitivity matrix	
y	Measurement signal including noise	
y^{cons}	Constrained measurements	
y^{opt}	Nominal values of y at the optimal point	
y^{uncons}	Unconstrained measurements	
y_0	Measurement signal excluding noise	
Y_Δ	In MIQP: Y_δ extended with zeros to fit dimensions	
Y_δ	In MIQP: Quadratic form of Y_{blk}	
Y_{blk}	Symmetric block diagonal matrix of Y for each input u	
z	In dummy problems: Outputs	
z	In self-optimizing control: Loss matrix	

$\mathbf{M}$	size of big-$\mathbf{M}$	

Greek Letters

$\bar{\sigma}$	Largest Singular Value	
ϵ	Small distance	
λ	Lagrangian multiplier of inequality constraints	
ν	Lagrangian multiplier of equality constraints	
$\sigma_{j,k}$	Binary selection variable for measurement $_j$ and measurement set $_k$	

Subsea Case Symbols

α_i	Oil fraction at point $_i$	
$\mu(\alpha)$	Viscosity at given oil fraction	Pa s
Ω	Empricial swirl constant	rad s^{-1}
τ	Residence time	s
A_i	Cross section are at point $_i$	m^2
C_{decay}	Decay factor for lost momentum	
d	Height of emulsion phase at weir	m
FS_i	Flow split in unit $_i$	
g	Gravity constant	$\mathrm{m\,s}^{-2}$
h	Vertical distance traveled	m
H_w	Height of weir	m
k_{re}	empirical fitted re-entrainment factor	
L	Horizontal distance traveled from inlet to weir	m
q_i	Flow at point $_i$	$\mathrm{m}^3\,\mathrm{h}^{-1}$
q_{re}	Re-entrainment flow	$\mathrm{m}^3\,\mathrm{s}^{-1}$
R	Total radius	m
r	Radial coordinate	m
R_c	Characteristic radius	m
r_d	Droplet radius	m
R_i	Radius of the inner pipe	m

v_h	Horizontal velocity	$\mathrm{m\,s^{-1}}$
v_r	Radial velocity	$\mathrm{m\,s^{-1}}$
v_v	Vertical velocity	$\mathrm{m\,s^{-1}}$
v_z	Axial velocity	$\mathrm{m\,s^{-1}}$
v_θ	Tangential velovity	$\mathrm{m\,s^{-1}}$
z	Axial coordinate	m

Chapter 1

Introduction

In chemical engineering plants, it is important to control and operate the system as close to its optimum in order to maximize profit. Plantwide control looks at the decision on how to make the control system design for the whole chemical plant[3], instead of focusing on each unit operation individually. Self-optimizing control is a framework within plantwide control which helps us selecting (among others) controlled variables. The idea is to select a control structure so that it is possible to control the system with an acceptable economic loss minimizing the impact of uncertainties in disturbances. This economic loss links the system profitability, through a cost function with the control structure design. The goal is to minimize the economical loss which is equivalent to control the system as close to optimum as possible. One area of interest is the selection of measured variables. In general, more measurements give the controller a better understanding on how the system behaves and therefore provide a closer to optimum control of the system. However, as seen in Figure 1.1 the loss decreases rapidly for the first few measurements, but after that installing new measurements have little effect on the system's performance. By not installing measurement devices we save both investment and maintenance costs. It is therefore of interest to choose only a subset of the measurement candidates. Although a method for choosing a subset of measurements has been presented in literature [4], so far no method of coupling the prices of installing these measurements within the self-optimizing control framework has been introduced.

An area of interest within control is subsea processing of crude oil. As oil fields are getting increasingly more difficult to access, subsea processing has gained attention over the last years[5]. There are many potential benefits of using subsea processing: By removing water and sand from the oil stream at the sea bed, these components do not need to be transported to the top side facility reducing the costs of artificial lift dramatically, reduce the risk of hydrate formation,and the tube diameter of oil pipelines can be shrank. The need an offshore topside facility can be eliminated or reduced, which is a huge potential benefit, especially in harsh condi-

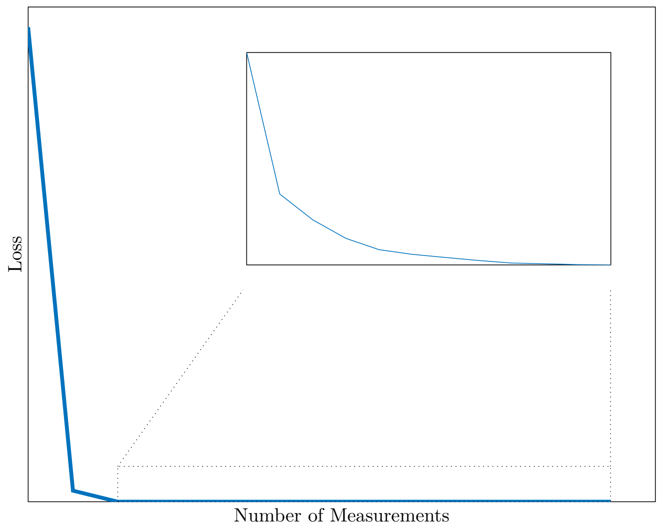

Figure 1.1: Illustration of optimal average loss of a system for the best subset of measurements.

tions as the arctic or at very deep water where topside processing are considered difficult or not economical feasible. Over the last years, the oil price has dropped, making the need for cost savings even higher. The field of subsea processing is therefore an interesting candidate when linking up the selection of measurements for control with the price of measurements for control structure design.

1.1 Objective

This work aims to develop a method to include the prices of measurements into a optimization problem that finds the optimal subset of measurements for a control structure in a process plant. This should be done within the Self-optimizing control framework. The work could be seen as a continuation of Yelchuru and Skogestad [4] which utilize Mixed Integer Quadratic Programming (MIQP) to optimize the selected control variables (CV).
The prices of measurements are included into the same optimization problem as the loss calculation determined by the control structure, where the goal is to minimize the total cost. By doing so, the control structure could be implemented as part of the process design phase, which is beneficial, since today's process plants is generally tested in process models, and to include operational aspects (control) into the plant design phase may lead to savings during operation[6]. It is thereafter of interest to investigate how a second measurement candidate set with different prices and uncertainties (sometimes referred to as noise) influences the selection of measurements as well as the total project cost. In the end, the framework is extended to also consider a process with constraints. The goal is to design a control

structure which takes into account the back-off due to measurement noise, the unconstrained loss and the prices of each selected measurement in a constrained system. The control structure giving the lowest total cost should be selected. The methods are presented and applied to both a dummy problem and a subsea oil/water processing system with three separators.

Chapter 2

Literature Review

This chapter presents some theoretical background for the work that is going to be conducted.

2.1 Self-Optimizing Control

This section gives a introduction to self-optimizing control, and methods within this field of study. It is based on the paper [6]. The term self-optimizing control was first used by Skogestad [3] and is regarded a framework from which *"the goal is to find controlled variables (CV) which, when kept at constant setpoints, indirectly lead to near-optimal operation with acceptable economic loss"*. To understand what loss is, a cost function J is defined so that it creates a optimization problem which minimizes the cost (or maximizes profit). The cost function could be given in \$/s:

$$\begin{aligned} \min \quad & J(u,x,d) \\ \text{subject to:} \quad & g(u,x,d) \le 0 \\ & h(u,x,d) = 0 \end{aligned} \tag{2.1}$$

where u are inputs to values, x are states, d are disturbances, g is inequality constraints and h is equality constraints. The loss is defined as the distance between the optimum point found in the optimization problem above and the actual operating value, hence the loss can be defined as in (2.2)

$$L = J(u,x,d) - J^{opt}(d) \tag{2.2}$$

where $J(u,x,d)$ is the cost function at given inputs (u), states (x) and disturbances (d) compared to an optimal point (J^{opt}).

The idea of self-optimizing control is that by carefully selecting the controlled variables, the work done by a online real-time optimizer can be minimized. The real-time optimizer can be a computer or trained human operators. Consider a general feedback system as given in Figure 2.1: In this system the different timescales

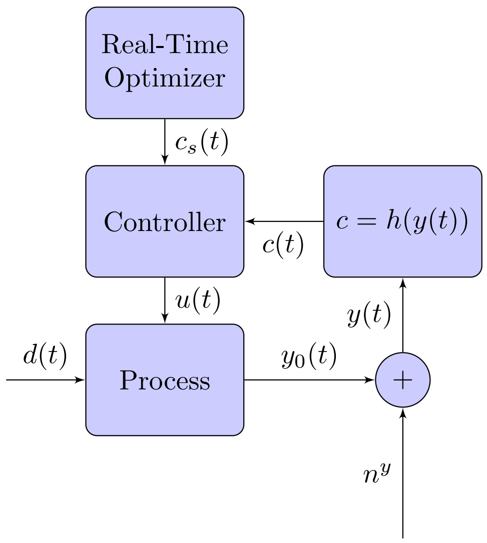

Figure 2.1: A general feedback control structure used in self-optimizing control

can be read from top to bottom. The Real-Time optimizer works on a timescale of hours and computes set points (c_s) to the normal controllers which reject disturbances (d) on timescales from seconds up to minutes and hence giving inputs (u) to valves. The selection of controlled variables (c) could be described as in (2.3). y is all possible measurements which consist of a reading/signal from the process (y_0) and a related measurement noise (n^y).

$$c(t) = h(y(t)) \tag{2.3}$$

where $c(t)$ is the controlled variable, h is a selection matrix based on the measurements $y(t)$. This is where the self-optimizing control theory comes in hand. Normally this is denoted $c = Hy$, where $H \in \mathbb{R}^{n_c \times n_y}$. Values in H can simply be a single 1 in every row which symbols an one-to-one relationship between the measurement and controller, but it could also be a linear combination of multiple measurements into one controller. Finding the optimal measurement combination is an important aspect in self optimizing control. An important aspect of using self-optimizing control (instead of only real-time optimizer) is that the control structure selection is solved off-line in steady state. This saves time for the real-time optimizer, but may lead to a loss L as defined in (2.2). Self optimizing control has to be seen as a complement, not instead of Model Predictive Control (MPC).

2.1.1 Brute-Force Approaches

Over the years many approaches have been made within the framework to find the best controlled variables. The earliest methods were based on brute-force approaches, to evaluate all possible combinations [3]. However, the brute-force approaches has limitations due to the complexity of large chemical plants, which requires finding solutions to a huge set of optimization problems. For instance, consider the classical Tennessee-Eastman challenge by Downs and Vogel [7] which have 42 measurements (n_y) and 12 controllers (n_c). This leads to almost 8 billion possibilities as seen in (2.4).

$$C_{n_y}^{n_c} = \binom{n_y}{n_c} = 7.98 \times 10^9 \tag{2.4}$$

This leads to unfavorable calculation time, so that one need to make assumptions to reduce the number of problems. A common simplification is to neglect measurement noise. However, this can lead to poor CV choices and can be numerically difficult to solve.

2.1.2 Local Methods

Whereas brute-force methods tries to evaluate all possible measurement-controller possibilities, the local method approach is trying to reduce this number by pre-screening. This pre-screening of CVs is done by looking at which measurements perform well close to the nominal points, by making small perturbations. This is done by Taylor expansions so that the cost function can be approximated by the quadratic function:

$$\begin{aligned} J(\Delta u, \Delta d) \approx J^{opt} + [J_u + J_d] \begin{bmatrix} \Delta u \\ \Delta d \end{bmatrix} \\ + \frac{1}{2}[\Delta u^T \Delta d^T] \begin{bmatrix} J_{uu} & J_{ud} \\ J_{du} & J_{dd} \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta d \end{bmatrix} \end{aligned} \tag{2.5}$$

where $J_i = \partial J / \partial i$ and $J_{ij} = \partial^2 J / \partial i \partial j$ are the first and second derivative of the cost function with respect to general variables i and j[8]. Around a optimal nominal point, $J_u = 0$ which is then used in (2.5) while differantiating with respect to Δu, which gives:

$$\frac{\partial J}{\partial u} \approx \underbrace{J_u}_{0} + J_{uu}\Delta u + J_{ud}\Delta d = [J_{uu} J_{ud}] \begin{bmatrix} \Delta u \\ \Delta d \end{bmatrix} = 0 \tag{2.6}$$

This equation is then solved for the optimal input ($\Delta u^{opt}(d)$) which yields:

$$\Delta u^{opt}(d) = -J_{uu}^{-}1 J_{ud} \Delta d \tag{2.7}$$

Combining the quadratic approximation, (2.5) with (2.7) and the expression for the loss, (2.2) gives[9]:

$$L = \frac{1}{2}(\Delta u - \Delta u^{opt}(d)) J_{uu} (\Delta u - \Delta u^{opt}(d)) \tag{2.8}$$

This can be also written as:

$$L = \frac{1}{2}\|z\|_2^2 \tag{2.9}$$

where $\| \cdot \|_2$ is the two-norm and z is defined as

$$z = J_{uu}^{1/2}(\Delta u - \Delta u^{opt}) \tag{2.10}$$

Finding Local Optimal Control Structures

It can be shown that the measurement $y = y_0 + n^y$, when linearized around the optimum nominal point can be rewritten as:

$$\Delta y = G^y \Delta u + G_d^y \Delta d + n^y \tag{2.11}$$

where $\Delta y = y - y^{opt}$ is the distance away from the nominal point, $G^y = \partial y / \partial u$ and $G_d^y = \partial y / \partial d$ and are the gain from inputs and gain from disturbances respectively. The controlled variables c are then chosen to be linear combinations of the measurements:

$$\Delta c = H \Delta y \tag{2.12}$$

Inserting (2.11) into (2.12) gives (2.13) which denotes the local approximation of c around the nominal point.

$$\Delta c = HG^y \Delta u + HG_d^y \Delta d + Hn^y \tag{2.13}$$

Exact Local Method

To evaluate the loss as well as the best measurement combination for a given control structure, it is possible to use the exact local method[8]. The loss is an important indicator because it shows how the system behaves compared to the optimum. Rearranging (2.13) and assuming that the optimal steady state point does not require active control ($\Delta c = 0$) gives:

$$\Delta u = -(HG^y)^{-1} H (G_d^y \Delta d + n^y) \tag{2.14}$$

This is inserted into the expression for the loss (2.10) together with the expression of optimal input ($\Delta U^{opt} = -J_{uu}^{-1} J_{ud} \Delta D$) becomes:

$$z = -J_{uu}^{1/2} (HG^y)^{-1} H[(G_d^y - G^y J_{uu}^{-1} J_{ud}) \Delta d + n^y] \tag{2.15}$$

A matrix F is then introduced as seen in (2.16):

$$F = G_d^y - G^y J_{uu}^{-1} J_{ud} \tag{2.16}$$

so that (2.15) can be rewritten as

$$z = -J_{uu}^{1/2} (HG^y)^{-1} H[F \Delta d \quad n^y] \tag{2.17}$$

The matrix F is not easy to obtain in the current form, but since it is the sensitivities of the optimal measurement values with respect to the disturbances it could also be defined as:

$$F = \partial y^{opt} / \partial d \tag{2.18}$$

which can be obtained by for instance re-optimization. The exact local method also takes measurement or disturbance noise into account. These are defined as

$$\Delta d = W_d d' \tag{2.19}$$

$$n^y = W_n n' \tag{2.20}$$

where W_d and W_n are diagonal matrices which denotes the magnitude of the noise for disturbance and measurement noise respectively whereas d' and n' are the scaling vectors. Define matrices Y and M:

$$Y = [FW_d \quad W_n] \tag{2.21}$$

$$M = -J_{uu}^{1/2}(HG^y)^{-1}HY \tag{2.22}$$

so that when using (2.17), (2.9) becomes

$$L = \|M \begin{bmatrix} d' \\ n' \end{bmatrix} \|_2^2 \tag{2.23}$$

The loss can then be calculated depending on the disturbance and noise. The two-norm is used for the worst case loss(2.24) [8].

$$L_{worst} = \frac{1}{2}\bar{\sigma}^2(M) \tag{2.24}$$

where $\bar{\sigma}$ is the largest singular value. The infinity norm is used when disturbance are assumed independent and uniformly distributed resulting the average loss [10].

$$L_{av} = \frac{1}{6}\|M\|_F{}^2 \tag{2.25}$$

where $\|\cdot\|_F$ is the Frobenius norm. When d' and n' are normally distributed with zero mean and unit variance, the worst case loss goes to infinity whereas the average loss becomes:

$$L_{av} = \frac{1}{2}\|M\|_F^2 \tag{2.26}$$

Minimum Loss Method (Explicit Solution)

To find the optimal selection of variables several methods have been suggested, for instance the null-space method[11] and the extended null space method [12]. However, in this work the minimum loss method with explicit solution has been chosen [12] and [4]. Starting out with matrix M (2.22), which is the basis for the loss calculations, one can introduce a invertible matrix Q that does not affect the loss, so that $\Delta c = QH\Delta y$. Inserting Q into (2.22) becomes:

$$\begin{aligned} M &= -J_{uu}^{1/2}(QHG^y)^{-1}QHY \\ &= -J_{uu}^{1/2}(HG^y)^{-1}Q^{-1}QHY \\ &= -J_{uu}^{1/2}(HG^y)^{-1}HY \end{aligned} \tag{2.27}$$

In the last part of (2.27) Q was chosen so that $HG^y = J_{uu}^{1/2}$ to cancel the non-linearity so that (2.27) becomes:

$$\begin{aligned} &\min_H \|HY\|_F \\ \text{subject to} \quad &HG^y = J_{uu}^{1/2} \end{aligned} \tag{2.28}$$

The optimization problem in (2.28) has the explicit solution:

$$H = (G^y)^T (YY^T)^{-1} \tag{2.29}$$

which is the best locally measurement combination for a given set of all measurements, based on the assumption that YY^T is invertible. It is not given that this H is the same in the entire operating range, it could change in different operating areas. Note that (2.28) is the same as (2.26) scaled with two. This means that the optimization problem (2.28) actually minimizes the average loss.

2.1.3 Selecting subsets of measurements

In the local methods explained above, the selection matrix H is based on all available measurements. However, in subsea applications or other processes in general, it may not be feasible or necessary to use all the possible measurement candidates. Is it possible to control the system just as good (or nearly as good) by choosing only a few measurements? A measurement has always an investment cost related to it, so by not installing the measurement equipment this could be economically beneficial. There could also be other reasons for not installing measurement equipment, for instance space considerations. When one starts to limit the number of measurements the previous formulations need extra attention due to the combinatorial problem that arises from all the possible control structures. There are currently two main approaches that can solve subset of measurement problems. Firstly, it is the (Bidirectional) Branch and Bound approach, Cao and Kariwala [13]. Alternatively, it is possible to formulate the selection of subset as a mixed integer quadratic optimization problem (MIQP) which is solved by mathematical programming solvers (which utilize among others Branch and Bound algorithms),Yelchuru and Skogestad [4]. MIQP-formulations is chosen in this work since they have the benefit over Branch and Bound algorithms beacuse modern MIQP solvers can handle multiple constraints directly without further customization.

Mixed Integer Quadratic Programming (MIQP)

A mixed integer quadratic optimization problem takes the general form:

$$\begin{aligned} &\min_x x^T Q x + q^T x \\ &\text{s.t.} Ax \le b \\ &\quad l \le x \le u \end{aligned} \tag{2.30}$$

where x is the objective vector, Q is a symmetric objective matrix, q is linear objective vector, A is a linear constraint matrix, b is a linear constraints vector,

l is the lower bounds for x and u is the upper bounds for x. Some or all of the values in x, must take integer values. It is also possible to limit some or all values to binary values.

MIQP formulation for selection of subsets

The following paragraph explains the restructuring needed to express the selection of a subset of measurements as an MIQP. It differs slightly from the purely mathematical form.
When adapting the general formulation in the previous section into the selection of a subset of measurements small modifications has to be made, and letters are changed to follow the previous notation. Starting out with (2.28), the previous matrix H, which still is the objective, is transformed into a vector of length $n_u \cdot n_y$ and called h_δ. The selection of measurements are taken care of by a vector of binary variables, σ_j, which is of length n_y. These are called selection variables. If σ_j has a value of 1 this means that the measurement was chosen in the subset, a value of 0 means that the measurement was not chosen in the subset. σ is appended to h_δ. A single integer is also appended to take care of the number of measurements that is going to be selected. The matrix Y from (2.28) is transformed into symmetric block matrix (Y_{blk}) repeating Y one time for each degree of freedom (normally n_u). Y_{blk} is then multiplied with itself transposed due to use of Frobenius norm (giving the system its quadratic properties as given in (2.28)). Y_Δ is calculated from (2.31) and (2.32). Extra rows and columns are added for the selection criteria (these are all set to zero, to not having any impact on the quadratic term). In the mathematical it is not needed to include these changes in Y_Δ. However, this is done to express it the way it is implemented in solver software.

$$Y_{blk} = \begin{bmatrix} [Y] & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & [Y] \end{bmatrix} \tag{2.31}$$

$$Y_\delta = Y_{blk} \cdot Y_{blk}^T \tag{2.32}$$

$$Y_\Delta = \begin{bmatrix} \overbrace{\begin{bmatrix} Y_\delta \end{bmatrix}}^{n_y \cdot n_u} & \cdots & \mathbf{0} \\ \vdots & \overbrace{\begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}}^{n_y} & \vdots \\ \mathbf{0} & \cdots & \overbrace{\begin{bmatrix} 0 \end{bmatrix}}^{1} \end{bmatrix} \tag{2.33}$$

To avoid influence on loss and cost by measurements that is not chosen, linear constraints make sure that the corresponding elements in the selection matrix H are set to zero. This is done by using the big-**M** approach [14]. This constraints also limits the values of H. An increase in big-**M** (not to be confused with matrix

M) gives longer calculation time, but may be needed if the numerical values in H is within a large range. The final formulation of the problem becomes:

$$
\begin{aligned}
\min_{h_\delta,\sigma_j} \quad & h_\delta^T Y_\Delta h_\delta \\
\text{s.t} \quad & {G^y}_\delta^T h_\delta = j_\delta \quad \text{linear constraints} \\
& l \leq h_\delta \leq u \quad \text{bound constraints} \\
& P\sigma = s \\
& -\mathbf{M}\sigma_j \leq H_{i,j} \leq \mathbf{M}\sigma_j \quad \text{j=1...}n_y\text{,i=1...}n_u
\end{aligned}
\tag{2.34}
$$

where j_δ is a reformulated J_{uu}, ${G^y}_\delta$ is reformulated G^y, P is a vector of ones ($P = [1...1] \in \mathbb{R}^{1\times n_y}$), s is an integer equal the number of measurements in the subset (n_m), and l and u is the lower and upper bounds on h_δ. Another benefit of using MIQP-formulations is that it is easy to incorporate structural constraints on which measurements to choose, for instance, if one wants two temperature measurements and three flow measurements. This can be selected by extending the $P\sigma = s$ term with more rows of s. This was also introduced by Yelchuru and Skogestad [4] and will not be covered in this work.

2.2 Constrained problems

So far, only unconstrained measurement selections have been investigated, that is, the original objective function for the process (not the MIQP) has no active constraints. However, in many chemical engineering problems there will be constraints, reducing the degrees of freedom (DoF) in the system. Such constraints can for instance be maximum temperature in a reactor, or maximum molar fraction of a bi-product in a product stream. These constraints will in most cases restrict (or constrain) the optimum operation point. So what is a constrained optimization problem? A general optimization problem is defined as:

$$
\begin{aligned}
\min_{x} \quad & f(x) \\
\text{s.t} \quad & g(x) \leq 0 \\
& h(x) = 0
\end{aligned}
\tag{2.35}
$$

where $f(x)$ is the cost function, $g(x)$ are inequality constraints and $h(x)$ are equality constraints. An illustration of a constrained problem is given in Figure 2.2, where the system has an inequality constraint $g(x) \leq 0$. In addition a system can also have equality constraints $h(x) = 0$ (not drawn in Figure 2.2). A physical illustration of this is a ball rolling down a hill ($f(x)$) towards the bottom (blue dot). If the ball rolls into a fence, the fence is equal to inequality constraints ($g(x)$). The ball is free to move away from the fence if the optimum lies on the same side of the fence as the ball (this will however make the constraint unnecessary). Equality conditions ($h(x)$) on the other hand can be taught of as rails. The ball can hence only roll on the railings (as a train).

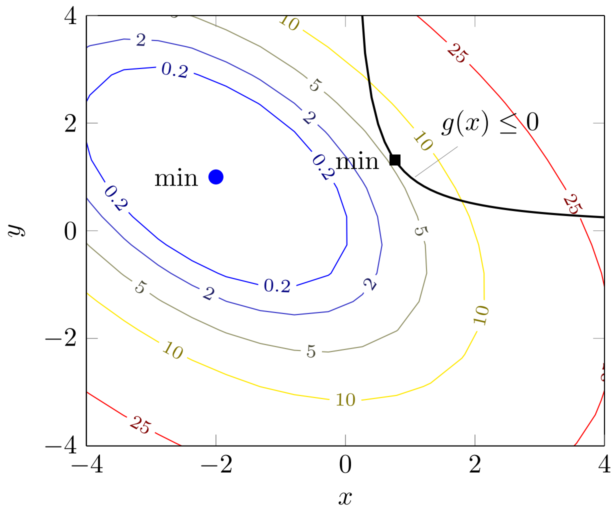

Figure 2.2: A constrained problem. The problem is constrained by the black line $g(x)$ which limits possible solution to points in the top right corner. The optimal solution is hence moved from the blue circle to the black square.

In general the Lagrange function is given as:

$$\mathbb{L}(x, \lambda, \nu) = f(x) + g(x)^T \lambda + h(x)\nu \tag{2.36}$$

where x is the system variables, λ is the Lagrange multiplier for the inequality constraint $g(x)$ and ν is the Lagrange multiplier for the equality constraint $h(x)$. The first KKT condition[15] can be physically explained as a force balance and is easily obtained by differentiating (2.36) with respect to x around the optimal point (x^*,λ^*,ν^*) becomes:

$$\nabla_x \mathbb{L} = \nabla f(x^*) + \nabla g(x^*)^T \lambda^* + \nabla h(x^*)\nu^* = 0 \tag{2.37}$$

How hard the system pushes towards the constraint is therefore equal to the Lagrange multiplier. Now, consider a specific inequality constraint $g_1(x)$ with a corresponding Lagrange multiplier λ_1, which is perturbed by a small amount ϵ so that $g_1(x) + \epsilon \leq 0$. As long as ϵ is small it can be shown that (2.37) can be written asBiegler [15]:

$$\begin{aligned} 0 &= h(x^\epsilon) - h(x^*) \approx \nabla h(x^*)^T (x^\epsilon - x^*) \\ 0 &= g_i(x^\epsilon) - g_i(x^*) \approx \nabla g_i(x^*)^T (x^\epsilon - x^*), i \neq 1 \\ -\epsilon &= g_i(x^\epsilon) - g_i(x^*) \approx \nabla g_i(x^*)^T (x^\epsilon - x^*), i = 1 \end{aligned} \tag{2.38}$$

combining the results found in (2.38) with (2.37) gives that the differnce in cost function $f(x)$ a small perturbation away from the optimal point becomes:

$$\begin{aligned} f(x^\epsilon) - f(x^*) &\approx \nabla f(x^*)^T f(x^\epsilon) - f(x^*) \\ &= -\sum_i \nu_i^* \nabla h_j(x^*)^T (x^\epsilon - x^*) - \sum_i \lambda_i^* \nabla g_i(x^*)^T (x^\epsilon - x^*) \qquad (2.39) \\ &\approx \sum_i \nu_j^* \overbrace{(h_j(x^\epsilon) - h_j(x^*))}^{=0} - \sum_i \lambda_i^* \overbrace{(g_i(x^\epsilon) - g_i(x^*))}^{=-\epsilon} \\ &= \lambda_1^* \epsilon \qquad (2.40) \end{aligned}$$

Chapter 3

Including Prices in Control Structure Selections

This chapter explains the new methods developed in this thesis. First, a possible method to include measurement prices is shown and then a method for selection between two different measurement sets is presented. The last method of measurement selection that is demonstrated, is when one of the states in the process is constrained.

3.1 Including Prices for Unconstrained Systems

The following section aims to find new methods to give new selection criteria based on price and accuracy considerations.

3.1.1 Motivation

If the control structure is created alongside the design phase, it is possible to optimize which measurement devices is going to be (bought and) installed in order to operate the plant at its optimal capacity. A measurement installed in a plant will always come at a price, with both a fixed purchase cost and a maintenance cost. The motivation is therefore clear from an economic perspective; reduce the number of measurements to a minimum, without affecting the plant performance. Including prices of measurements and alongside, calculating the performance of a plant (loss) is therefore of interest. From this, a number of questions arises. For instance: To what degree do you loose performance if you choose an inexpensive temperature measurement over an expensive pressure measurement? The price of a measurement device is often related to its accuracy, so what if you can choose a cheap inaccurate measurement device over an expensive accurate measurement device? This chapter aims to find a method, which when implemented can answer questions like these.

3.1.2 Including Prices in Measurement Selection

In Section 2.1.3, which is based on [4], a method for selecting a subset of measurements using an MIQP-formulation was explained. The reason for not including all possible measurements could be for instance cost considerations, but a method to select measurements also based on the prices of measurements is yet to be presented in literature. To include a price of a measurement, a price term is added to the linear term for each of the measurements in the objective function. If the measurement is not selected, the selection variable (σ_j) will be zero and hence not affect the final cost. If the measurement is selected, the corresponding selection variable will be one and the measurement price is added to the objective function. The MIQP formulation with prices as linear costs in the objective function is then given by:

$$\begin{aligned}
&\min_{h_\delta} \quad h_\delta^T Y_\Delta h_\delta + \bar{p}^T \sigma && (3.1)\\
\text{s.t} \quad & G^{yT}_{\delta} h_\delta = j_\delta \quad \text{linear constraints}\\
& l \leq h_\delta \leq u \quad \text{bound constraints}\\
& P\sigma = s\\
-\mathbf{M}\sigma_j \leq\ & H_{i,j} \leq \mathbf{M}\sigma_j \quad \text{j=1...}n_y\text{,i=1...}n_u\\
& \sigma_j \in [0,1]
\end{aligned}$$

where the linear term $\bar{p}^T \sigma_j$ is added to (2.34) to give (3.1). Here $\bar{p}^T$ is a vector of prices with length n_y.

3.1.3 Select a Subset From Two Different Measurement Sets

The previous section explains how a price of a measurement can be inserted into a MIQP-problem. But choosing measurements is not just about choosing where in the process measurement equipment should be located, but also what kind of measurement device is going to be purchased. This can be illustrated by having the choice between the measurement device from producer A which is accurate, but expensive, and a measurement device from producer B which is less accurate, but cheaper. In order to cope with this, the MIQP-formulation needs to be extended. Recall that the measurement noise (W_n) is the measurement uncertainty which is given in (2.21). A second set of measurement noise W_{n2} is defined so that:

$$Y_2 = [FW_d \quad W_{n2}] \tag{3.2}$$

Y, from now denoted Y_1, where the goal is to define a new Y which consists of both Y_1 and Y_2. In order to build the final matrices for the MIQP formulation, Y

needs to be restructured to fit the number of inputs u as shown in Section 2.1.3:

$$Y_{1,blk} = \begin{bmatrix} [Y_1] & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & [Y_1] \end{bmatrix} \tag{3.3}$$

$$Y_{2,blk} = \begin{bmatrix} [Y_2] & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & [Y_2] \end{bmatrix}$$

The quadratic term in the MIQP-formulation arises due to the Frobenius norm in (2.28). The quadratic term becomes:

$$Y_{1,\delta} = Y_{1,blk} Y_{1,blk}^T \tag{3.4}$$

$$Y_{2,\delta} = Y_{2,blk} Y_{2,blk}^T \tag{3.5}$$

By merging the two sets of measuring devices (3.4) and (3.5), the final restructured Y_δ becomes:

$$Y_\delta = \begin{bmatrix} Y_{1,\delta} & 0 \\ 0 & Y_{2,\delta} \end{bmatrix} \tag{3.6}$$

where $Y_\delta \in \mathbb{R}^{(2n_u n_y \times 2n_u n_y)}$. The number of selection variables σ needs to be increased with n_y. One particular selection variable is from now on denoted $\sigma_{j,k}$, which is the selection variable for y_i in measurement set w_k. The final quadratic objective matrix Y_Δ now becomes:

$$Y_\Delta = \begin{bmatrix} \overbrace{\begin{bmatrix} Y_\delta \end{bmatrix}}^{2\cdot n_y \cdot n_u} & \cdots & \cdots & \mathbf{0} \\ \vdots & \overbrace{\begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}}^{n_y} & \cdots & \vdots \\ \vdots & \cdots & \overbrace{\begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}}^{n_y} & \vdots \\ \mathbf{0} & \cdots & \cdots & \overbrace{\begin{bmatrix} 0 \end{bmatrix}}^{1} \end{bmatrix} \tag{3.7}$$

where $Y_\Delta \in \mathbb{R}^{2n_u n_y + 2n_y + 1 \times 2n_u n_y + 2n_y + 1}$. A new set of linear constraint is added, to force values in the selection matrix to zero, if the selected measurement belongs to a different measurement set w_k. Example: Measurement $y_1 5 w_2$ ($\sigma_{15,2} = 1$) is selected, the constraint has to make sure that both $\sigma_{15,1} = 0$ and $H_{15,i,1} = 0$. The constraints are added using the selection variables and the big-**M** constraints[16].

In a case where two measurement devices sets are available, the constraints are formulated as:

$$|H_{y_j,u_i,w_1}| + \mathbf{M}\sigma_{j,2} \leq \mathbf{M} \tag{3.8}$$
$$|H_{y_j,u_i,w_2}| + \mathbf{M}\sigma_{j,1} \leq \mathbf{M} \tag{3.9}$$

where y_j is the measurement, corresponding to an input u_i and a measurement device set w_k. These constraints are created for each $H_{j,i,k}$. The absolute sign is included to take care of negative values of H. The constraints work so that if a measurement device from the second set ($w = 2$) is chosen, then $|H_{y_j,u_i,w_1}| + M \leq M$ and the only value for $|H_{y_j,u_i,w_k}|$ that makes this true is $|H_{y_j,u_i,w_1}| = 0$. To force the solver to only select either $y_j w_1$ or $y_j w_2$ SOS constraints was added [16].

Final MIQP-formulation

The final MIQP-formulation is given in (3.10)

$$\begin{aligned}
\min_{h_\delta,\sigma_{j,k}} \quad & h_\delta^T Y_\Delta h_\delta + \bar{p}^T \sigma_{j,k} && && (3.10)\\
\text{s.t} \quad & G^{y}{}_\delta^T h_\delta = j_\delta && \text{linear constraints}\\
& l \leq h_\delta \leq u && \text{bound constraints}\\
& P\sigma_{j,k} = s\\
& -\mathbf{M}\sigma_{j,k} \leq H_{i,j,k} \leq \mathbf{M}\sigma_{j,k} && \text{j=1...}n_y\text{,i=1...}n_u\text{,k=1...}n_w\\
& |H_{i,j,w=k}| + \mathbf{M}\sigma_{j,w\neq k} = \mathbf{M}\\
& \sigma_{j,k} \in [0,1]
\end{aligned}$$

3.1.4 Select the Optimal Number of Measurements

Previously the MIQP-formulation has only been used to minimize the loss as given in (2.26). However, since prices of measurements are also included, the optimum number of measurements is the smallest total cost (the loss plus the prices of measurements). In an engineering perspective we are interested in the total cost of a project. By minimizing the total cost, this should give an important insight since the total cost already include all the process losses (distance away from the optimal operating point) and the costs related (in this case the prices of measurements). As opposed to only looking at the loss, as in Figure 1.1, which will reach its minimum at the maximum number of available measurements. The total cost could provide a function which has a minimum at a number of measurements smaller than the maximum number of measurements n_y.

3.2 Including Prices for Constrained Systems

3.2.1 Back-off Loss From Constrained Input

As seen in Section 2.1.2 the loss L, which was defined as $L = J^{actual} - J^{opt}$, arises due to measurement noise (W_n) or disturbance noise (W_d). If a constraint is added, it is needed to back-off away from the new constrained optimum to avoid constraint violation due to measurement uncertainty. This is true for systems where the constrained optimum lies on the constraint (active constraints). Example: One needs to control a temperature so that it is maximum 100 ° °C, the measurement has 1 ° °C uncertainty. This means that the temperature actually need to be controlled at 99 ° °C to avoid constraint violation.
Back-off is defined here as the loss due to uncertainty of measurement of the constrained, or in mathematical terms: L^{cons}. When backing off, the least the size of the movement possible is equal to the measurement uncertainty. This can be seen as a perpendicular perturbation. As seen in (2.40), moving away from the optimal point by a perturbation ϵ, moves the value of the cost function $\lambda\epsilon$ away from the optimum point. Combining (2.40) with (2.2), and introducing the measurement noise (W_n) as ϵ gives:

$$L^{\text{cons}} = \lambda W_n \tag{3.11}$$

Note that the more strongly active a constraint is, the larger the Lagrange multiplier will be and hence the loss. A weakly active constraint will produce a very small Lagrange multiplier and hence the loss will be small. The loss given in (3.11) is only valid for one particular constraint at one specific point. However, so is also the loss calculated using the exact local method for the unconstrained case. It is now of interest to combine the unconstrained loss and the constrained loss into a single problem to find the total loss. Also note that there are ways to reduce this back-off, for instance by using squeeze and shift methods [17]. However, this has not been considered in this work.

3.2.2 Assumptions

When introducing a constraint it is assumed that we know which input (u) and which measurement (y) is used to control the constraint. It is also assumed that only one measurement is needed to control the constraint using only one input. This is will reduce the degrees of freedom and hence affect which inputs and measurements are available for selection the unconstrained case. In addition it is assumed that a change in disturbance does not change the active constraint around the local point. This is due to the local methods used for the unconstrained case. Further it is assumed that the unconstrained loss stays the same while backing off. It is then possible to split up the optimization problem into two smaller independent optimization problems.

3.2.3 Split Into Two Optimization Problems

Consider a process with $j = 1, 2...n_y$ measurements, $i = 1, 2...n_u$ inputs, and $d = 1, 2...n_d$ disturbances, which is linearized around a optimal nominal point so that (2.11) applies. A constraint is added and one measurement y_j and one input u_i are used to control the constraint. Split up (2.11) into a constrained and an unconstrained case:

$$\Delta y^{\text{cons}} = G^y(y_j, :)\Delta u + G_d^y(y_j, :)\Delta d + n^y \tag{3.12}$$

$$\Delta y^{\text{uncons}} = G^y(y_{\neq j}, :)\Delta u + G_d^y(y_{\neq j}, :)\Delta d + n^y \tag{3.13}$$

where $G^y(y_j, :)$ means row y_j and all columns of G^y and $G^y(y_{\neq j}, :)$ means all rows except row y_j and all columns of G^y. As seen in (3.12) and (3.13) one row in G^y and G_d^y is moved from the unconstrained case to the constrained case. Noise is neglected (for now) and perfect control of the constraint is assumed so that $\Delta c = H\Delta y = 0 \implies \Delta y^{\text{cons}} = 0$. Next solve (3.12) for the input used to control the constraint, u_i:

$$\Delta u_i = \frac{G_d^y(y_j, :)}{-G^y(y_j, u_i)}\Delta d + \frac{G^y(y_j, u_{\neq i})}{-G^y(y_j, u_i)}\Delta u_{\neq i} \tag{3.14}$$

In practice, the second term in (3.14) could be very small if paring the y_j with u_i is done properly. However, in the general case inserting (3.14) into (3.13) gives:

$$\Delta y^{\text{uncons}} = G^y(y_{\neq j}, :)\begin{bmatrix} \frac{G_d^y(y_j, :)}{-G^y(y_j, u_i)}\Delta d + \frac{G^y(y_j, u_{\neq i})}{-G^y(y_j, u_i)}\Delta u_{\neq i} \\ \Delta u_{\neq i} \end{bmatrix} + G_d^y \Delta d$$

$$\begin{aligned} \Delta y^{\text{uncons}} = &\overbrace{\Big(\frac{-G^y(y_{\neq j}, u_i)G^y(y_j, u_{\neq i})}{G^y(y_j, u_i)} + G^y(y_{\neq j}, u_{\neq i})\Big)}^{G^{y,\text{uncons}}} \overbrace{\Delta u_{\neq i}}^{\Delta u^{\text{uncons}}} \\ &+ \underbrace{\Big(- \frac{G^y(y_{\neq j}, u_i)G_d^y(y_j, :)}{G^y(y_j, u_i)} + G_d^y(y_{\neq j}, :)\Big)\Delta d}_{G^y_{d,\text{uncons}}} \end{aligned} \tag{3.15}$$

In general $G^y_{d,\text{uncons}}$ is only used in (2.16) to calculate F, and is therefore not needed in cases where F is found by re-optimization as in (2.18). The system described in (3.15) represents the reduced version of (2.11), which then is implemented in the self optimizing control framework described earlier.

3.2.4 Calculating the Total Loss

If only one set of measurement devices is available ($w = 1$), the constrained loss given in (3.11) can only be added directly to the unconstrained loss found using the self optimizing control framework. However, in general the constrained case becomes an optimization problem since it is possible to choose different devices with different prices ($w > 1$). It is still assumed that a known measurement is

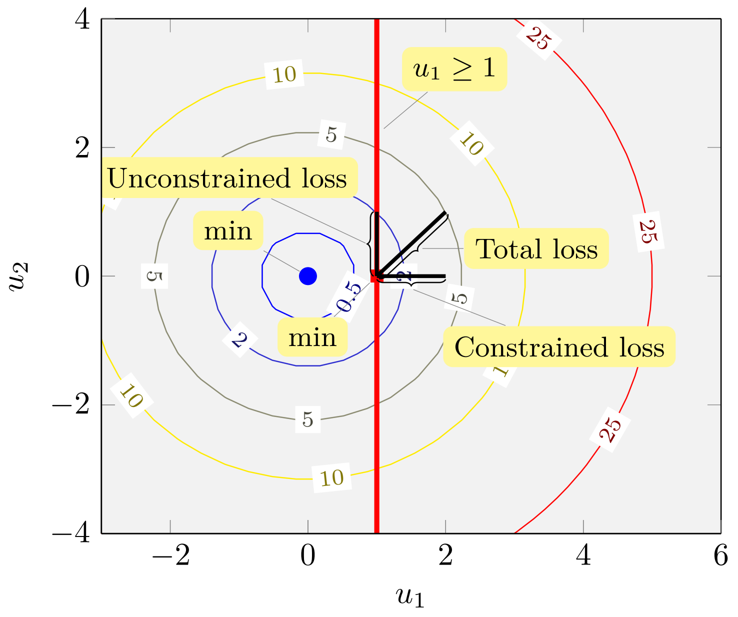

Figure 3.1: Illustration of constrained loss (back-off) and unconstrained loss (process loss). A constraint is added so that $u_1 \geq 1$. The illustration is not to scale.

selected to control the constraint so that the only choice left is to choose from the different measurement device sets (w). It has previously be shown that the unconstrained is a MIQP-problem. However, the constrained case becomes an integer linear optimization problem (ILP) as seen (3.16):

$$\begin{aligned} \min \quad & (\lambda W_{n_k})\sigma_k + \bar{p}^T\sigma_k \\ s.t \quad & P\sigma_k = 1 \\ & \sigma_k \in [0,1] \end{aligned} \tag{3.16}$$

where λ is the value of the constraint's Lagrange multiplier, W_{nk} is the measurement noise of measurement device set w_k, σ_k is the selection variable for measurement device set w_k. $\bar{p}^T$ is the prices and P is a vector of ones with length equal the number of measurement sets (k). Note that this problem actually is very simple to calculate, since all σ_k is binary. Note that the unconstrained and constrained optimization problem can be solved separately given the mentioned assumptions. An illustration of the constrained and unconstrained loss is given in Figure 3.1. The figure illustrates that the losses are orthogonal (not a linear combination of each other) and could therefore be added to find the total loss so that:

$$L^{\text{total}} = L^{\text{uncons}} + L^{\text{cons}} \tag{3.17}$$

The total loss vector in Figure 3.1 is for illustration only and is not to scale.

3.2.5 Implementation

A steady state optimization is ran at a given nominal point and the solver provides a λ for the given constraint. After a nominal point with a corresponding λ is found, the same self optimizing control sequence as described above is ran in order to calculate the gain matrix G^y, the sensitivty matrix F and the Hessian matrix J_{uu}. The corresponding input u_i and measurement y_j for the constraint need to be removed from each matrix before entering the MIQP-formulation. This means for instance that the size of the gain matrix is reduced from $G^y \in \mathbb{R}^{n_y \times n_u}$ to $G^{y,uncons} \in \mathbb{R}^{n_y - n_{cons} \times n_u - n_{cons}}$ where n_{cons} is the number of constraints.

3.3 Software Implementation

The models described in the previous sections has been implemented in MATLAB (version 2015b). For source code of the described methods and models see Appendix A. The MIQP-formulation has been implemented in the commercial mathematical programming solver GUROBI (version 6.5.1) with MATLAB API. Gurobi provides user readable optimization files, these are included in Appendix B. The general software implementation steps are given as follows:

1. Perform steady state simulations using MATLAB with `fmincon`-solver.
2. Choose a nominal point for local study, using an active constraint map
3. Perform small feed perturbations around nominal point to calculate sensitivity matrix F.
4. Perform small input perturbations around nominal point to calculate gain matrix and Hessian matrix from finite differences (Appendix C)
5. Define magnitude for disturbances and noise
6. Build all matrices needed for MIQP formulations
7. Include a second measurement set, if applicable
8. Define prices for measurements
9. Define constraints for the MIQP formulations
10. Perform MIQP calculation using GUROBI

In addition to what has been described earlier the following assumptions apply.

- Assume if d changes, active constraint does not change

Chapter 4

Case Study: Dummy Problems

This chapter applies the methods for measurement selection as described in the previous chapter, on two dummy problems. The first dummy problem presents an unconstrained problem. The second dummy problem is created in order to present a constrained problem. The results are regarded as proof of concepts. First the following evaluation criteria are defined:

- *Loss* is the unconstrained loss, process loss or steady state loss. It is defined in (2.26)
- *Price* is defined as the prices of measurements. The price could be of one single measurement device or total price, which is the sum of prices for the selected measurements.
- *Back-off* is the constrained loss. Due to measurement uncertainty, it is unknown how close to the constraint the measurement actually is, hence a back-off away from the constrained value is required. However, when backing-off we also travel away from the optimum, and the distance away from the optimum is the constrained loss or back-off.
- *Cost* or total cost, is the sum of all the terms mentioned above. In a unconstrained case the cost (or total cost) will be the sum of the unconstrained loss and the prices of measurements. Whereas in a constrained case the total cost will be the sum of the unconstrained loss, prices of measurements and back-off.

4.1 Unconstrained Dummy Problem Model Description

A simple dummy problem was created as a test case. The same problem is defined in [8] and [4]. Imagine a case with two inputs u, one disturbance d and two outputs z. The four measurements y are given as:

$$y = \begin{bmatrix} z_1 & z_2 & u_1 & u_2 \end{bmatrix}^T \tag{4.1}$$

The objective function J defined as:

$$J = (z_1 + z_2)^2 + (z_1 - d)^2 \tag{4.2}$$

The outputs depend linearly on inputs through (4.3)

$$z = G^z u + G^z_d d \tag{4.3}$$

with

$$G^z = \begin{bmatrix} 11 & 10 \\ 10 & 9 \end{bmatrix} \tag{4.4}$$

$$G^z_d = \begin{bmatrix} 10 \\ 9 \end{bmatrix} \tag{4.5}$$

The Hessian matrix evaluated around a nominal point for the inputs u was found to be:

$$J_{uu} = \begin{bmatrix} 244 & 222 \\ 222 & 202 \end{bmatrix} \tag{4.6}$$

$$J_{ud} = \begin{bmatrix} 198 \\ 180 \end{bmatrix} \tag{4.7}$$

Then the gain matrix becomes:

$$G^y = \begin{bmatrix} 11 & 10 \\ 10 & 9 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.8}$$

whereas G^y_d becomes:

$$G^y_d = \begin{bmatrix} 10 & 9 & 0 & 0 \end{bmatrix}^T \tag{4.9}$$

The sensitivity vector is then calculated from (2.16) with the matrices found in (4.6) to (4.9).

$$F = \begin{bmatrix} -1 \\ -1 \\ 9 \\ -9 \end{bmatrix} \tag{4.10}$$

In the base case the disturbance noise is $W_d = 1$ and the measurement noise is:

$$W_n = \operatorname{diag}(\begin{bmatrix}0.01 & 0.01 & 0.01 & 0.01\end{bmatrix}) \tag{4.11}$$

Y is calculated from (2.21) and becomes:

$$Y = \begin{bmatrix} -1 & 0.01 & 0 & 0 & 0 \\ -1 & 0 & 0.01 & 0 & 0 \\ 9 & 0 & 0 & 0.01 & 0 \\ -9 & 0 & 0 & 0 & 0.01 \end{bmatrix} \tag{4.12}$$

4.2 Model Verification

In order to verify the model it was tested to produce the same results as Yelchuru and Skogestad [4]. This does not include prices. Big-**M** was set to 120. A plot showing the optimal average loss is given in Figure 4.1 shows good correspondence with the results from [4]. Figure 4.1 and shows that it may be unnecessary to choose all four measurements as the losses are 0.000502 vs. 0.000366 for three and four measurements respectively. Although this represent a percentage big increase, the absolute number is so small that it is unlikely that it will be economical beneficial to choose all four measurements in this case.

In addition, the corresponding selection matrix H for a set of three measurements ($n_m = 3$), was calculated and is given in (4.13) and shows that the model produces the same results as Yelchuru and Skogestad [4, Section 5.1]. Recall that the selection matrix $H \in \mathbb{R}^{n_u \times n_y}$. Also recalls that $c = Hy$ so that the columns with 0's means that the measurement was not chosen. In a cost perspective this is equal to not installing the measurement device in the actual plant, saving both investment and operating cost of that measurement.

$$H(n_m = 3) = \begin{bmatrix} 1.0182 & 0 & 0.3959 & 0.2828 \\ 0.7637 & 0 & 2.0643 & 1.9795 \end{bmatrix} \tag{4.13}$$

4.3 Including Price of Measurements

With a working model, it is now of interest to test the influence of prices of measurements. So far the cost function from the MIQP-formulation has been equal to the loss. When prices are introduced this means that the project both has a loss related to the control structure, but also a price related to the measuring devices. The objective is still to be the same both from an engineering and economic perspective, to minimize the project cost.

4.3.1 Identical Prices of Measurements

The dummy case was simulated with constant prices of measurements, meaning that all measurements have the same price. The simulation was then repeated by changing the price for each iteration in order to produce a figure similar to Figure

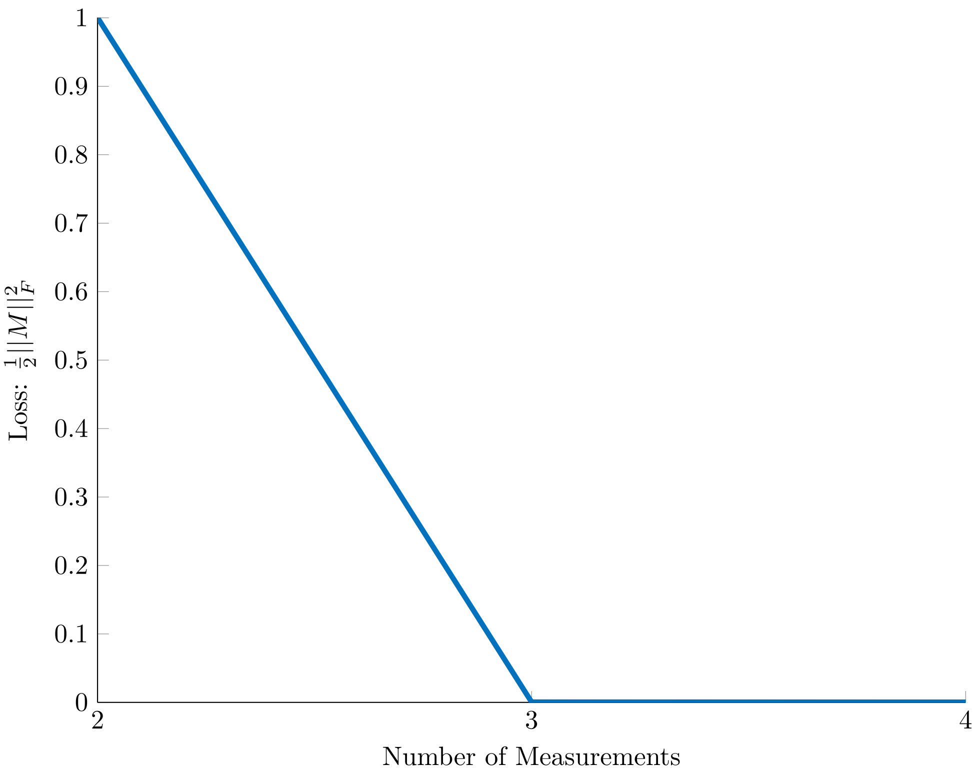

Figure 4.1: The loss plotted against the number of measurement. All prices of measurement are set to 0.

4.1. The prices were set to 0.05, 0.5 and 1. The case with prices equal to zero was added for comparison. The result is given in Figure 4.2.

The figure illustrates how price influence the total cost. At low prices, choosing between three or four measurements has only a small effect. When the prices increase to 0.5, choosing four measurements has a significant higher total cost. In these cases choosing three measurements will be the best solution. However at a price of 1 per measurement the price of measurements are much higher than the loss, meaning that only two measurements gives the lowest total cost.

4.3.2 Individual Prices of Each Measurement

In the previous section, identical prices for each measurement were assumed. However, in reality each (type of) measurement could have an unique price. For instance, a composition measurement is more expensive than a flow measurement or a temperature measurement. To show this effect, new prices were given to the different measurements, one at at time, as seen in the first column of 4.1.

The most interesting result is found when y_2 is given the price of 0.01: y_2 was

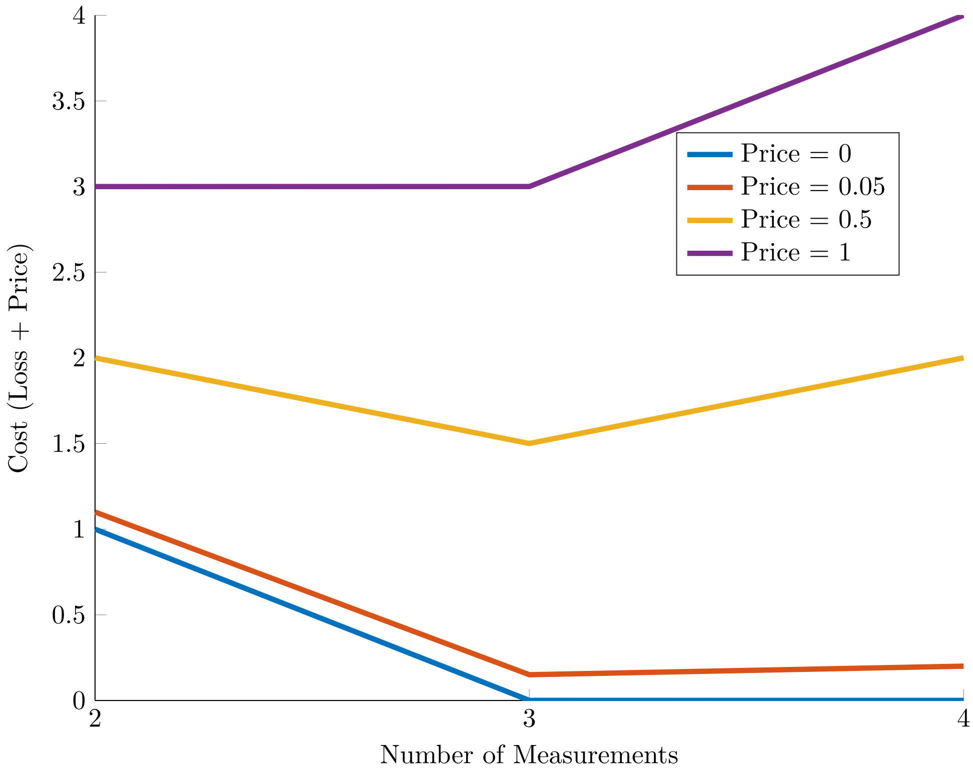

Figure 4.2: The total cost plotted against the number of measurements. All measurements have the same cost in each line. A cost as low as possible is wanted.

namely not chosen when $n_m = 3$ in any of the other cases, as seen in Table 4.1. However, if the price of $y_2 = 0.01$, this is changed. In this case both the price of measurements and the loss play a role when selecting the measurement set. Also note that choosing three measurements always gives the lowest total cost. An extension of this case is when both prices and measurement noise are unique for each device.

4.4 Include a Second Measurement Device Set

When constructing a plant the engineer may be faced with a choice of either buying inexpensive measurement devices with bad accuracy (high noise level) or expensive measurement devices with good accuracy (low noise level). The engineer is always looking for the best project economy and hence choosing the device giving the lowest total cost. By using the procedure from Section 3.1.3, the trade-off between a cheap investment and poor performance, and expensive investment and better performance is optimized.

Table 4.1: Measurement selection with individual prices. The prices for each measurement is given in the first column. The next column denotes the number of measurement in the simulation. The third column is the selected subset of measurements and last the related total cost.

Measurement price	No. y	Selected Measurements	Cost
y_1 , y_2 , y_3 , y_4			
0.01, 0.10, 0.10, 0.10			
	2	y_1,y_2	1.110300
	3	y_1,y_3,y_4	0.210502
	4	y_1,y_2,y_3,y_4	0.310366
0.10, 0.01, 0.10, 0.10			
	2	y_1,y_2	1.110300
	3	y_1,y_2,y_3	0.210662
	4	y_1,y_2,y_3,y_4	0.310366
0.10, 0.10, 0.01, 0.10			
	2	y_1,y_2	1.200300
	3	y_1,y_3,y_4	0.210502
	4	y_1,y_2,y_3,y_4	0.310366
0.10, 0.10, 0.10, 0.01			
	2	y_1,y_2	1.200300
	3	y_1,y_3,y_4	0.210502
	4	y_1,y_2,y_3,y_4	0.310366

4.4.1 No Prices of Measurements

The simplest test for this optimization problem is to use two measurement device sets (w_1 and w_2) with different measurement noise (W_n) and then set the price to zero. The program should choose only measurements with a small measurement noise, because choosing «bad» measurements when «good» measurements are available at no cost will only lead to higher losses. Define the two device sets as:

$$\begin{aligned} W_{n1} &= \text{diag}(\begin{bmatrix}0.1 & 0.1 & 0.1 & 0.1\end{bmatrix}) \\ W_{n2} &= \text{diag}(\begin{bmatrix}0.01 & 0.01 & 0.01 & 0.01\end{bmatrix}) \end{aligned} \tag{4.14}$$

The results from the simulation is given in Table 4.2. The table shows that if the prices are zero, the optimizer selects the «good» measurements.

4.4.2 Good and Expensive vs. Bad and Cheap

To show the trade off between «good» and expensive vs. «bad» and inexpensive measurements, the measurement noise was defined as in (4.14) and the prices were set to 0.02 and 0.2 for w_1 and w_2 respectively. The results when using these parameters for the noise and prices is given in Table 4.3. The table shows that with the prices given in this problem, the best trade-off is to use inaccurate measurements (w_1). The optimal number of measurements are three, with a slightly lower cost than using four measurements. The final selection matrix (H) for the optimal number of measurements ($n_m = 3$) is given in (4.15). Empty columns (non chosen measurements) are not included, so that each column represents the measurements given in Table 4.3.

$$H = \begin{bmatrix} 1.0188 & 0.3893 & 0.2768 \\ 0.7678 & 2.0198 & 1.9391 \end{bmatrix} \tag{4.15}$$

4.5 Constrained Dummy Problem Model Description

To test the method described in Section 3.2 for a constrained problem, a new dummy problem that better illustrates a constrained case, was created. Consider a process with the following objective/cost function:

$$J = \frac{1}{2}u_1^2 + \frac{1}{2}(u_2 - d_2)^2 \tag{4.16}$$

where u_1 and u_2 are the two inputs, and d_2 is the second of two disturbances. A constraint is given on the value of u_1 so that:

$$u_1 - d_1 - 1 \geq 0 \tag{4.17}$$

The disturbances varies so that: $-0.5 < d_1 < 0.5$ and $-1 < d_2 < 1$. The nominal value for both disturbances are therefore 0. This results in the following optimization problem, which is minimized around the nominal point:

$$\begin{aligned} &\min \frac{1}{2}u_1^2 + \frac{1}{2}u_2^2 \\ &s.t. u_1 \geq 1 \end{aligned} \tag{4.18}$$

which gives a value Lagrange multiplier for the constraint at $\lambda = 1$. Next define the measurements y as:

$$y = \begin{bmatrix} u_1 & u_2 & d_1 & d_2 & (u_1 - d_1) \end{bmatrix} \tag{4.19}$$

Table 4.2: Selected subset of measurements with two measurement device sets (w_1 and w_2) with all prices set to zero. Only measurements from the «good» device set (w_2) are chosen.

No. y	Selected Measurements	Loss
2	y_1w_2,y_2w_2	1.000300
3	y_1w_2,y_3w_2,y_4w_2	0.000502
4	y_1w_2,y_2w_2,y_3w_2,y_4w_2	0.000366

Table 4.3: Selected subset of measurement with two device sets (w_{n1} and w_{n2}) at different number of chosen measurement and the subset's corresponding cost (loss + prices).

No. y	Selected Measurements	Cost
2	y_1w_1,y_2w_1	1.070000
3	y_1w_1,y_3w_1,y_4w_1	0.109329
4	y_1w_1,y_2w_1,y_3w_1,y_4w_1	0.116290

Remember that:

$$y = y_{nom} + \frac{\partial y}{\partial u}(u - u_{nom}) + \frac{\partial y}{\partial d}(d - d_{nom}) \tag{4.20}$$

which is equivalent to (2.11) with the gain matrices defined in (4.21) and (4.22).

$$G^y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0.1 & 2 \\ 2 & 3 \\ 1 & 0 \end{bmatrix} \tag{4.21}$$

$$G^y_d = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0.2 \\ 0.5 & 3 \\ -1 & 0 \end{bmatrix} \tag{4.22}$$

Let's assume that the measurement $y_5 = u_1 - d_1$ is used to control the constraint on u_1 and we can perfectly control this constraint, then the system can be split into a constrained and unconstrained problem. The constrained problem becomes:

$$\Delta y^{\text{cons}} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta u_1 \\ \Delta u_2 \end{bmatrix} + \begin{bmatrix} -1 & 0 \end{bmatrix} \begin{bmatrix} \Delta d_1 \\ \Delta d_2 \end{bmatrix} + n_y \tag{4.23}$$

Whereas the unconstrained problem becomes:

$$\Delta y^{\text{uncons}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0.1 & 2 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} \Delta u_1 \\ \Delta u_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0.2 \\ 0.5 & 3 \end{bmatrix} \begin{bmatrix} \Delta d_1 \\ \Delta d_2 \end{bmatrix} + n_y \tag{4.24}$$

If the noise is neglected(very small compared to the inputs and disturbances) and perfect control ($\Delta c = H\Delta y = 0 \implies \Delta y = 0$) of this constraint is assumed, (4.23) becomes:

$$-\Delta u_1 = \Delta d_1 \tag{4.25}$$

Table 4.4: The noise and price for each of the measurement sets

	w_1	w_2
Noise	0.1	1
Price	0.2	0.02

Inserting (4.25) into (4.24) gives:

$$\Delta y^{\text{uncons}} = \underbrace{\begin{bmatrix}0\\1\\2\\3\end{bmatrix}}_{G^y_{\text{uncons}}} \begin{bmatrix}\Delta u_2\end{bmatrix} + \underbrace{\begin{bmatrix}-1+0 & 0\\ -0+0 & 0\\ -0.1+1 & 0.2\\ -2+0.5 & 3\end{bmatrix}}_{G^y_{d,\text{uncons}}} \begin{bmatrix}\Delta d_1\\ \Delta d_2\end{bmatrix} + n_y \tag{4.26}$$

The expression for the unconstrained measurements, (4.26) is then used in the self optimizing control framework as presented earlier (Section 2.1 and Chapter 3). The unconstrained cost function becomes:

$$J^{\text{uncons}} = \frac{1}{2}(u_2 - d_2)^2 \tag{4.27}$$

The Hessian matrices are found analytically: $J_{uu} = 1$, $J_{ud} = \begin{bmatrix}0 & -1\end{bmatrix}$. From this the sensitivity matrix is calculated from (2.16). The maginitude of the disturbances was previously defined so that:

$$W_d = \begin{bmatrix}0.5 & 1\end{bmatrix}^T \tag{4.28}$$

Two measurement device sets (w_1 and w_2) is defined with measuring noise and prices given in Table 4.4. Note that y_5, which is used on the constraint, still belongs in w_1 and w_2. So that choosing between these still applies for y_5.

4.6 Measurement Selection for a Constrained Dummy Problem

The price and uncertainty for this model is given in Table 4.4. The unconstrained part of the problem has four measurements ($n_y = 4$), one input ($n_u = 1$) and two measurement device sets ($n_w = 2$). The constrained part of the system has one measurement, one input and two measurement device sets. The system was implemented with a big-**M** of 500. The results from the simulation is given in Table 4.5. The table shows that the unconstrained loss quickly decreases, the constrained loss stays the same, which is due to the fact that the same measurement device ($y_5 w_1$) is chosen for all cases. The optimum number of (unconstrained) measurements is two since this gives the lowest total cost. Note that if three or four (unconstrained) measurements is chosen, these come from the second device set (w_2), probably because of the low price, and the decrease in loss is very small anyway. The selection

Table 4.5: Selected Measurements including different losses and total cost for the constrained dummy problem. The chosen measurement device for the constraint is included in selected measurements, and counted in the first column through (+1).

No. y	Selected Measurements	L^{uncons}	L^{cons}	L^{total}	Total Cost
1(+1)	y_2w_1,y_5w_1	0.505000	0.100000	0.605000	1.005000
2(+1)	y_1w_1,y_2w_1,y_5w_1	0.024231	0.100000	0.124231	0.724231
3(+1)	y_1w_1,y_2w_1,y_3w_2,y_5w_1	0.023659	0.100000	0.123659	0.743659
4(+1)	y_1w_1,y_2w_1,y_3w_2,y_4w_2,y_5w_1	0.023604	0.100000	0.123604	0.763604

matrix, H for the best case ($n_y = 2$) is given in (4.29). Empty columns (non chosen measurements) are removed, and the constrained measurement does not contribute to H as it is a 1-to-1 relationship between y_5 and u_1.

$$H = \begin{bmatrix} 1.9231 & 1.0000 \end{bmatrix} \tag{4.29}$$

Chapter 5

Case Study: Subsea separation system

This chapter applies the methods for measurement selection on a subsea separation system, where the final goal is a control structure design using the selection matrix (H).

5.1 Subsea Separation Model Description

The model has been adapted from Tyvold [1]. The model has only been modified in order to increase numerical robustness. The cost function and constraint have also been modified. This section only briefly describes the model, as it has not been the focus in this thesis, for details; consult [1]. Only a two-phase (oil and water) system is considered.

5.1.1 Gravity Separator

In order to do an initial bulk separation, a model of a gravity separator is used. It is created as a horizontal tank with one inlet and two outlets. The liquid at the inlet is considered an emulsion and the only separation force is the gravitational buoyancy forces acting on the droplets due to the density difference between oil and water. Hence, a pure water phase and a pure oil phase is formed at the bottom and top of the separator respectively. A weir separates the two outlet, so that all liquid hitting the weir goes in the bottom stream, and all liquid above the weir goes in the top stream, notated q_b and q_t respectively. It is assumed that a movement of oil in one direction is counteracted by an equal movement of water in the opposite direction. A figure showing the phase separation is given in Figure 5.1.

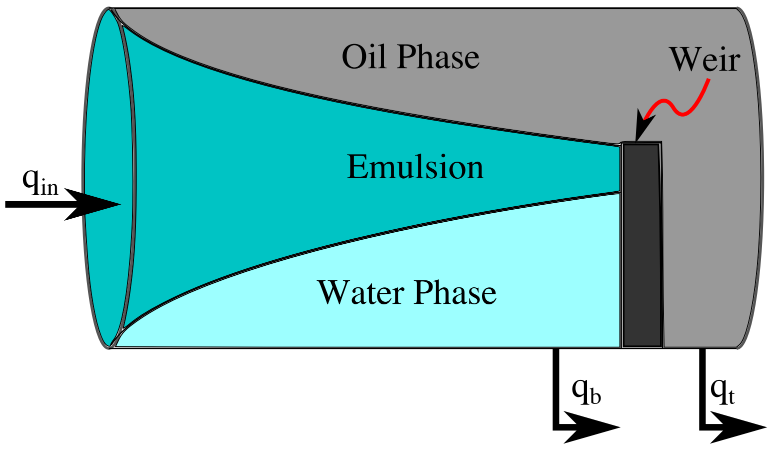

Figure 5.1: Horizontal gravity separator. Figure taken from [1]. Liquid enters as an emulsion, the gravitational buoyancy forces acting on the density differences separates the two phases so that an oil layer and a water layer is created. A weir separates the outlet streams.

Horizontal Velocity

The horizontal velocity is modelled as two separate plug flows, which are separated by the weir height. The horizontal velocity in the bottom part is given by:

$$v_h = \frac{q_b}{A_b} \tag{5.1}$$

where A_b is the cross sectional area of the lower part of the separator derived by trigonometry:

$$A_b = \frac{R^2}{2}\left[2\cos^{-1}\left(\frac{R - H_w}{R}\right) - \sin\left(2\cos^{-1}\left(\frac{R - H_w}{R}\right)\right)\right] \tag{5.2}$$

where R is the vessel radius and H_W is the height of the weir.

Vertical Velocity

The droplets experience a vertical velocity due to the gravitational forces, assuming Stoke's law, so that the velocity expression becomes:

$$v_v = \frac{2r_d^2(\rho_d - \rho)g}{9\mu(\alpha)} \tag{5.3}$$

where r_d is the radius of the droplet, ρ_d and ρ are the densities of the droplet and continuous (emulsion) phase respectively, g is the gravitational constant and $\mu(\alpha)$ is the viscosity as a function of oil fraction (sometimes referred to as oil-cut). It is assumed that the emulsion phase does not have any vertical velocity itself. This includes also the neglection of all turbulence in the vertical direction.

Droplet size

The model assumes an average droplet diameter instead of a droplet size distribution. It is also assumed that the droplet size is independent of the flow. It is further assumed that neither droplet break-up or coalescence take place in the emulsion phase.

Concentration profile

It is assumed uniform concentration profiles within the three different phases (oil phase, emulsion phase and water phase). This means that the concentration is constant in each of the three phases. In the oil phase, it is assumed pure oil $\alpha = 1$, the emulsion phase has the inlet oil fraction $\alpha = \alpha_{in}$ and the water phase consists of pure water, $\alpha = 0$. This assumption becomes less accurate when there is an increase in the standard deviation of the droplet size distribution.

Viscosity

The viscosity is regarded, as a function of oil fraction and is calculated for the emulsion phase from[18]. The viscosity is constant inside the emulsion phase due to the assumption of constant concentration in each phase.

Oil Fraction in Product Streams

The volumetric volume fractions of the top and bottom streams are estimated from the vertical distance covered as a function of the residence time. This means that a droplet located below the weir height (H_w) will travel a vertical distance:

$$\Delta h = \frac{v_v}{v_h} L \tag{5.4}$$

where L is the horizontal distance covered from the inlet to the weir, v_v is the vertical droplet velocity and v_h is the horizontal droplet velocity. The amount of oil left in the bottom stream is limited by

$$d = H_w - \Delta h \tag{5.5}$$

By geometry the area with remaining oil (emulsion phase going into bottom stream) given by d becomes:

$$A_e = \frac{R^2}{2}\left[2\cos^{-1}\left(\frac{R-d}{R}\right) - \sin\left(2\cos^{-1}\left(\frac{R-d}{R}\right)\right)\right] \tag{5.6}$$

Due to the concentration assumptions given earlier the oil fraction in the bottom is

$$\alpha_b = \alpha_{in}\frac{A_e}{A_b} \tag{5.7}$$

The oil fraction in the top stream is then calculated by a component volume balance (in practice a mass balance due to constant density).

$$\alpha_t = \frac{1}{q_t}[\alpha_{in}q_{in} - \alpha_b q_b] \tag{5.8}$$

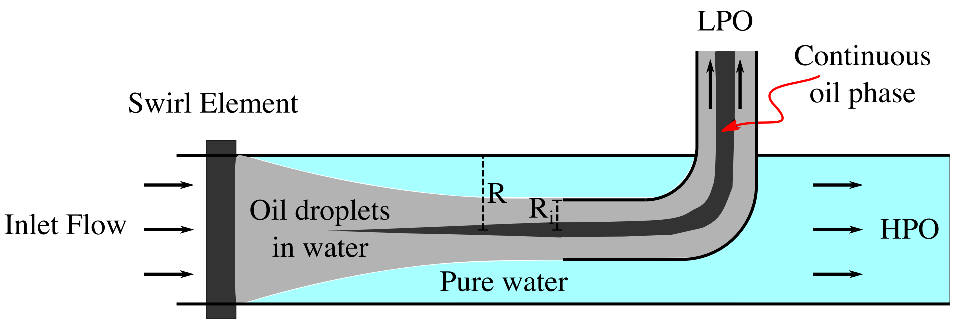

Figure 5.2: Principle sketch of the swirl separator. Figure adapted from [1]

where q_t, q_{in} and q_b are the volumetric flows in the top-, in- and bottom stream respectively.

5.1.2 Swirl Separator

The system also consists of two Swirl/In-Line separators. This is a relative new separator technology developed by CDS Technology (now: FMC Technology) and Statoil [19]. Figure 5.2 shows the working principle of the swirl separator. The following subsection describes how the swirl separators are modeled. Define first that an input to the separator model is the flow split between the light phase outlet (LPO) flow and the inlet flow:

$$FS = \frac{q_{LPO}}{q_{in}} \tag{5.9}$$

Axial velocity

The axial velocity profile is assumed as an annular plug flow without friction at the wall. It is assumed two discrete constant velocities based on radius of the droplet so that there is one velocity in the area which goes into the light phase output and one velocity for the area which goes into the heavy phase output. The axial velocity profile is given in (5.10).

$$v_z(r) = \begin{cases} \frac{q_{LPO}}{\pi R_i^2} & , 0 \leq r \leq R_i \\ \frac{q_{HPO}}{\pi(R^2 - R_i^2)} & , R_i \leq r \leq R \end{cases} \tag{5.10}$$

In (5.10) R_i is the tube radius for the light phase outlet, R is the total radius (radius of heavy phase outlet), q_{HPO} is the volumetric flow in the heavy phase outlet. Although this velocity profile is unrealistic from a fluid flow perspective, it could be an OK assumption for finding an estimation of separation performance.

Tangential Velocity

The point of using a swirl separator is that tangential/centrifugal forces will separate heavier droplets from lighter droplets. Experimental work has shown that the tangential velocity can be described as a Rankine vortex [20]. A Rankine vortex is a velocity profile that can be divided into an inner region with a solid body rotation and an outer region with free vortex movement. These regions are separated at a characteristic radius R_c. In this model the tangential velocity is assumed constant above this characteristic radius, so that the final expression for the tangential velocity downstream of the swirl element is given as:

$$v_\theta^0(r) = \begin{cases} v_\theta^{max} \frac{r}{R_c} & , 0 \leq r \leq R_c \\ v_\theta^{max} & , R_c \leq r \leq R \end{cases} \tag{5.11}$$

where v_θ^{max} is the maximum tangential velocity described by:

$$v_\theta^{max} = \Omega v_{z,b} \tag{5.12}$$

where Ω is an empirical proportionality constant based on the swirl element's geometry and $v_{z,b}$ is the axial bulk velocity (in this case:$v_z = v_{z,b}$). Equation (5.11) is given immediately downstream the swirl element where the swirl has maximum momentum. However, when the droplets travel in the axial direction, they loose momentum due to friction from the pipe wall. A new expression is needed so that it may be possible to calculate the tangential velocity at a distance away from the swirl element. An experimentally determined decaying factor, C_{decay} is a added to the expression [21], so that:

$$v_\theta(r,z) = v_\theta^0(r) \exp\left(-C_{\text{decay}} z / 2R\right) \tag{5.13}$$

where z is the axial coordinate A sketch showing how the swirl element gives tangential velocity to the droplets is given in Figure 5.3.

Radial Velocity

The radial velocity is determined by Stoke's law so that the radial velocity of droplet is:

$$v_r(r,z) = \frac{2r_d^2(\rho_d - \rho)v_\theta^2(r,z)}{9\mu(r,z)r} \tag{5.14}$$

where r_d is the droplet radius, μ is the viscosity, ρ_d is the dispersed phase density and ρ is the continuous phase density.

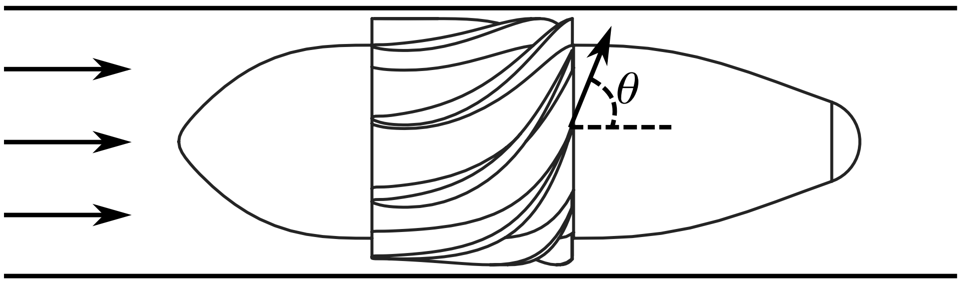

Figure 5.3: Sketch showing how the swirl element gives tangential forces to the droplets. Figure gathered from [2].

Viscosity

The viscosity is calculated the same way as in the gravity separator, Section 5.1.1.

Concentration

The concentration at the center of the separator is calculated by:

$$\alpha_c(r,z) = \alpha_{in}\frac{FS(R^2 - R_i^2) + (1-FS)(r_{in}^2 - R_i^2)}{(1-FS)(r^2 - R_i^2) + FS(R^2 - R_i^2)} \tag{5.15}$$

where α_{in} is the inlet oil fraction, FS is the flow split, R_i is the radius of the inner pipe, r is the radial coordinate and r_{in} is the radial coordinate at the inlet

Solving the Separator Model

The aim of the swirl model is to find a position r_{in} which is the last droplet position entering the light phase output stream $r = R_i$. To find this, the residence time is needed. The residence time, τ, is calculated from the tangential velocity and is written as:

$$\tau = \frac{\pi(R^2 - R_i^2)L}{(1-FS)q_{in}} \tag{5.16}$$

where q_{in} is the inlet volumetric flow. Then (5.14) is integrated over time $t = 0$ to $t = \tau$ using a constant time step, Runge-Kutta integrator. Then r_{in} for the droplet leaving at $r = R_i$ is found. The oil fraction of the light phase outlet stream can be calculated from:

$$\alpha'_{\mathrm{LPO}} = \min\left[1, \alpha_{in}\left(\frac{FS(R^2 - R_i^2) + (1-FS)(r_{in}^2 - R_i^2)}{FS(R^2 - R_i^2)}\right)\right] \tag{5.17}$$

The minimum expression is only to make sure that the model only calculates oil fractions below 1. The other outlet stream is then calculated using component mass balance as seen in (5.18)

$$\alpha'_{\mathrm{HPO}} = \frac{\alpha_{in} - \alpha'_{\mathrm{LPO}}FS}{1-FS} \tag{5.18}$$

To compensate for the simplified velocity profiles, a constant re-entrainment factor is added. The re-entrainment q_{re} is calculated as:

$$q_{re} = k_{re}\Delta v \tag{5.19}$$

where k_{re} is an empirical fitting parameter and Δv is the velocity difference between the two phases. The equation for calculating oil fraction including re-entrainment becomes:

$$\alpha_{\mathrm{LPO}} = \frac{1}{q_{\mathrm{LPO}}}[\alpha'_{\mathrm{LPO}}(q_{\mathrm{LPO}} - q_{re} + \alpha'_{\mathrm{HPO}}q_{re}] \tag{5.20}$$

The oil fraction in the heavy phase outlet is still calculated by (5.18), but α'_{LPO} is replaced with α_{LPO}.

Switching Continuous Phase in the Swirl Separator

The swirl separator explained above is created for a continuous water phase. However, in the system a swirl separator for a continuous oil phase is also required. The main difference between these two are that the droplets in a continuous oil phase consists of water which have a higher density than oil droplets. The consequence is that the droplets in this case are pushed outwards instead of inwards. Due to the higher oil content, this requires a larger inner tube radius, R_i than before. Other small changes include smoothing of the tangential velocity function.

5.1.3 Final System

A flow sheet of the final system is given in Figure 5.4. A flow (q_{in}) with a given oil fraction (α_{in}) is inserted into a gravity separator, which sends oil (and some water) in the top stream (q_t) and water (and some oil) through the bottom stream (q_b). The top stream enters a swirl separator (Dewaterer) which aims to remove the residual water. The residue water stream ($q_{\mathrm{HPO,DW}}$) is sent down for mixing with the bottom stream from the gravity separator (q_b) and enters the second swirl separator (Deoiler), where residual oil ($q_{\mathrm{LPO,DO}}$) is removed and sent to mix with the oil stream in the Dewaterer ($q_{\mathrm{LPO,DW}}$), to form the oil product stream ($q_{\mathrm{L,prod}}$). The water product stream out of the deoiler is denoted as $q_{\mathrm{H,prod}}$.

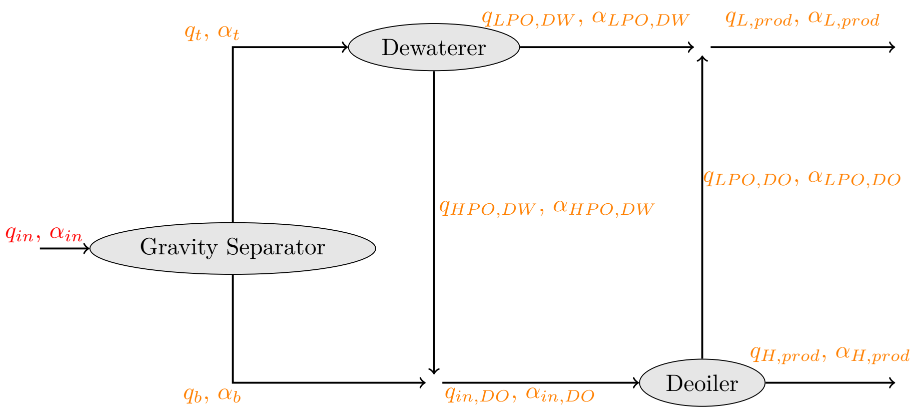
q_t, α_t
Dewaterer
$q_{LPO,DW}$, $\alpha_{LPO,DW}$
$q_{L,prod}$, $\alpha_{L,prod}$
$q_{LPO,DO}$, $\alpha_{LPO,DO}$
$q_{HPO,DW}$, $\alpha_{HPO,DW}$
q_{in}, α_{in}
Gravity Separator
$q_{H,prod}$, $\alpha_{H,prod}$
Deoiler
q_b, α_b
$q_{in,DO}$, $\alpha_{in,DO}$

Figure 5.4: Flow diagram of a subsea separation system. The dewaterer and deoiler are swirl separators for a continuous oil phase and continuous water phase respectively. Disturbances are marked with red, whereas the possible measurements are marked in orange. q_j denotes a flow measurement whereas α_j denotes a oil fraction measurement. The figure is based on a figure from [1].

5.2 System Analysis

The following sections present own work, but using the subsea model from [1] as a background. There are two disturbances in the system, q_{in} and α_{in} (drawn in red) and 16 measurement signals (drawn in orange) in the system as seen in Figure 5.4. There are three inputs, which is the flow splits in the separators. An illustration of the inputs is given in Figure 5.5. Since the feeds are given, the three inputs equals the degrees of freedom in the system. A flow split in separator $_i$ is defined as:

$$FS_i = q_{LPO,i}/q_{in,i} \tag{5.21}$$

where LPO stands for the light phase outlet. In practice the flow split will be a controller setting a ratio between the opening of two valves which controls the flow.

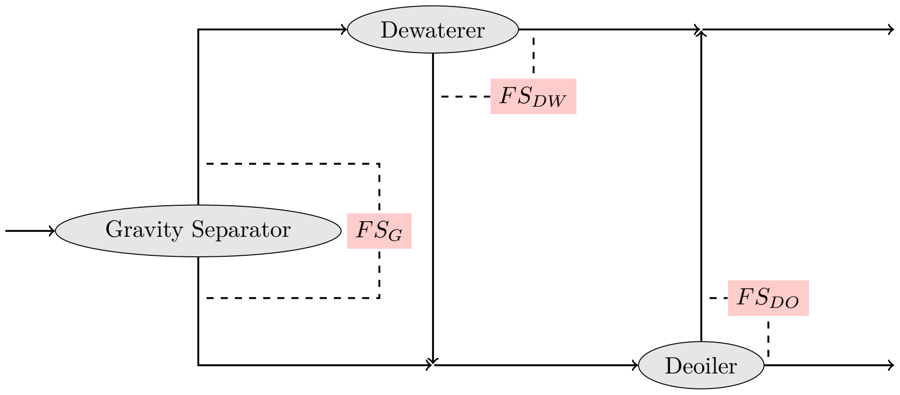

Figure 5.5: Flow diagram where the inputs FS_G, FS_{DW} and FS_{DO} are drawn. FS is an abbreviation for flow split, and the subscripts are abbreviations for gravity separator, dewaterer and deoiler, respectively.

5.2.1 Constraint

The subsea processing plant could have multiple constraints, for instance due to regulations. One of these regulations is the consternation of oil in produced water. The Norwegian Petroleum Directorate states that the maximum allowance of oil in discharged water is 30 mg L^{-1}[22], which depending on the oil density is $\approx$ 30 ppm. Other restrictions could apply for well re-injection. However, due to the numerical robustness of the model, the constraint of oil in produced water is set to 1% ($\alpha_{H,prod} \leq 0.01$). From an economic perspective, having oil in the water stream means that valuable product is spilled, and should therefore be avoided. However, there is also a trade off, since a low concentration of oil in water, could give a low yield in the oil product stream. The constraint removes a degree of freedom since it must be controlled. The natural choice is to choose the input FS_{DO} and the measurement $\alpha_{H,prod}$ to control this constraint. This reduces the steady state degrees of freedom to two (FS_G and FS_{DO}) and the unconstrained measurements to 15.

5.3 Steady State Optimization

To find the best control structure design, a steady state optimization needs to be conducted. Firstly a cost function is defined, next a steady state optimization is ran to find the active constraint region. The steady state optimization was conducted using a script from Tyvold [1] as a basis.

5.3.1 Cost Function

A cost function that could represent a subsea separation system was created and given in (5.22).

$$\min_{J} J = -\alpha_{L,prod} q_{L,prod} E_{oil} + (1 - \alpha_{L,prod}) q_{L,prod} p_{water} \tag{5.22}$$

where the first term is the earnings from the oil in the product stream. E_{oil} is the earnings from oil. A price of 40 \$/barrel was assumed. The second term is the costs for processing the residual water in the oil production stream. p_{water} is the price of topside water processing, a cost of 20 \$/barrel was assumed. In addition a regulation term was added, this does not influence the final cost, but could increase numerical robustness.

5.3.2 Active Constraint Region

To map out when the constraint given in Section 5.2.1 is active, a simulation was ran with different values for the feed($18\,\mathrm{m^3\,h^{-1}} \leq q_{in} \leq 23\,\mathrm{m^3\,h^{-1}}$ and $0.35 \leq \alpha_{in} \leq 0.6$). That an constraint is active means that the optimum value of the cost function would change if the constraint is removed. To decide whether the constraint is active, the Lagrange multiplier of the constraint is plotted against the two different feed changes, since a small value of the Lagrange multiplier will symbolize an unconstrained region, whereas a high value will consider an active constraint. The active constraint map is given in Figure 5.6, a whiter pixel indicates a higher Lagrange multiplier. The whiter dots in the middle could suggest some numerical noise, as this region moved to another part of the plot when the case was simulated again. The figure shows that the constraint is mostly inactive except for cases with a high flow rate (q_{in}) and high oil fraction (α_{in}).

5.3.3 Simulation results

A simulation was ran in order to see how the inputs (flow splits) change with different feed changes. The results from the steady state simulations are given as contour plots in Figure 5.7. The figure shows relatively large changes in the flow splits at different in flows and oil fractions, especially for the gravity separator. In addition plots showing the oil fractions in the product streams was created. The results are plotted with different feed concentration and a constant feed flow of $23\,\mathrm{m^3\,h^{-1}}$. The results are given in Figure 5.8 and 5.9. Note that a value of $\alpha_{H,prod} = 0.01$ in Figure 5.9, means that the constraint is active. Be aware that the non-smooth behaviour of the graphs could suggest some numerical difficulties.

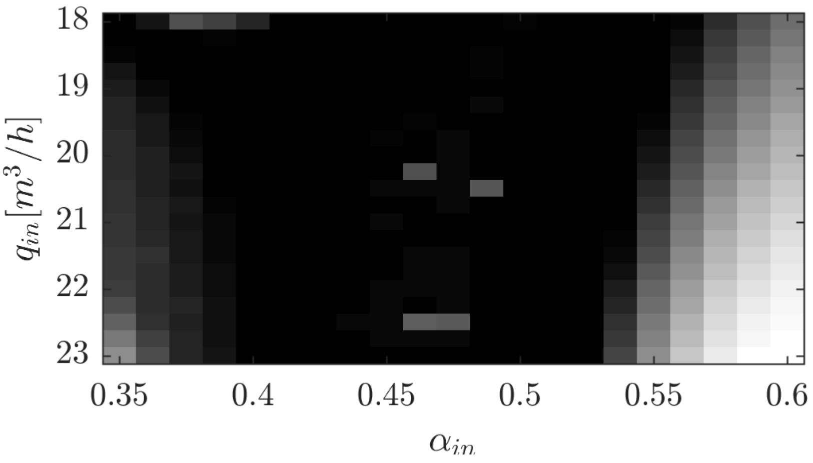

Figure 5.6: Active constraint map. A white pixel indicates an active constraint, whereas black indicates an unconstrained region. Figure created by plotting the value of the Lagrange multiplier.

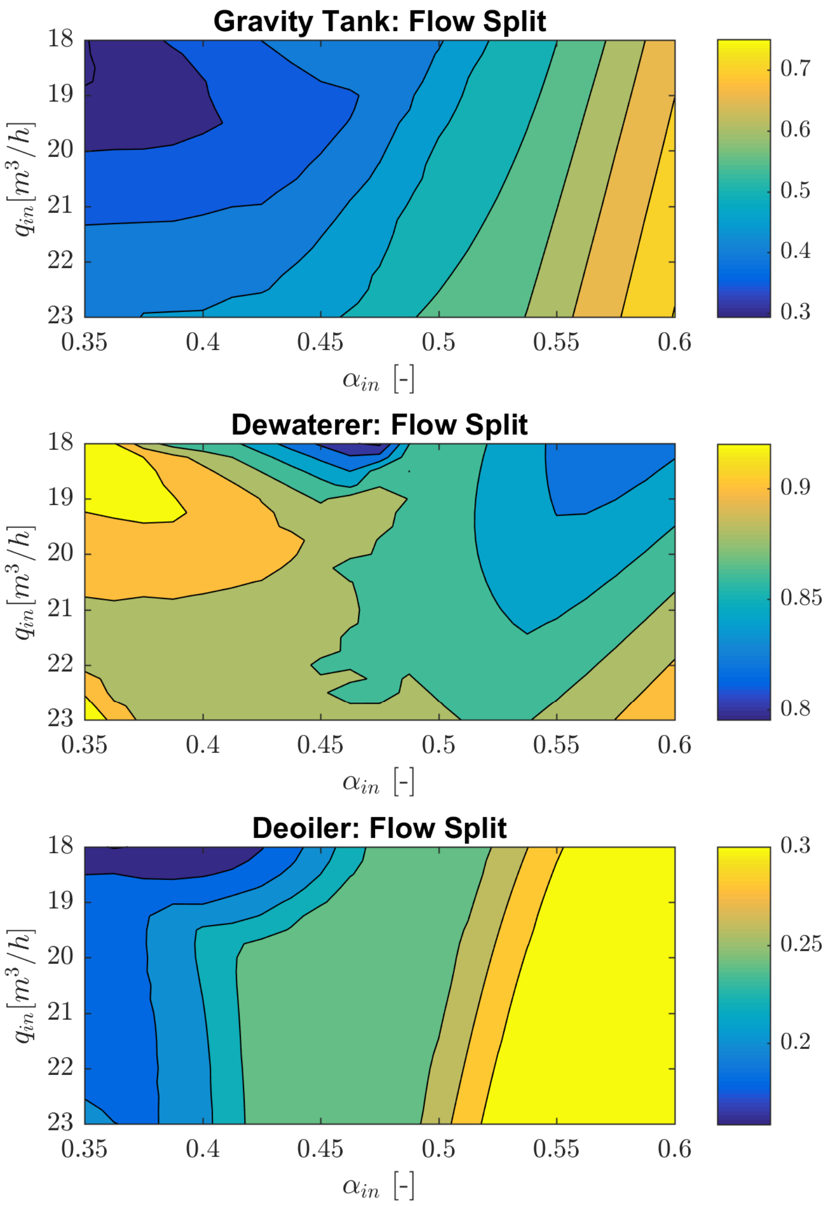

Figure 5.7: Figure shows a contour plot for each of the three flow splits.

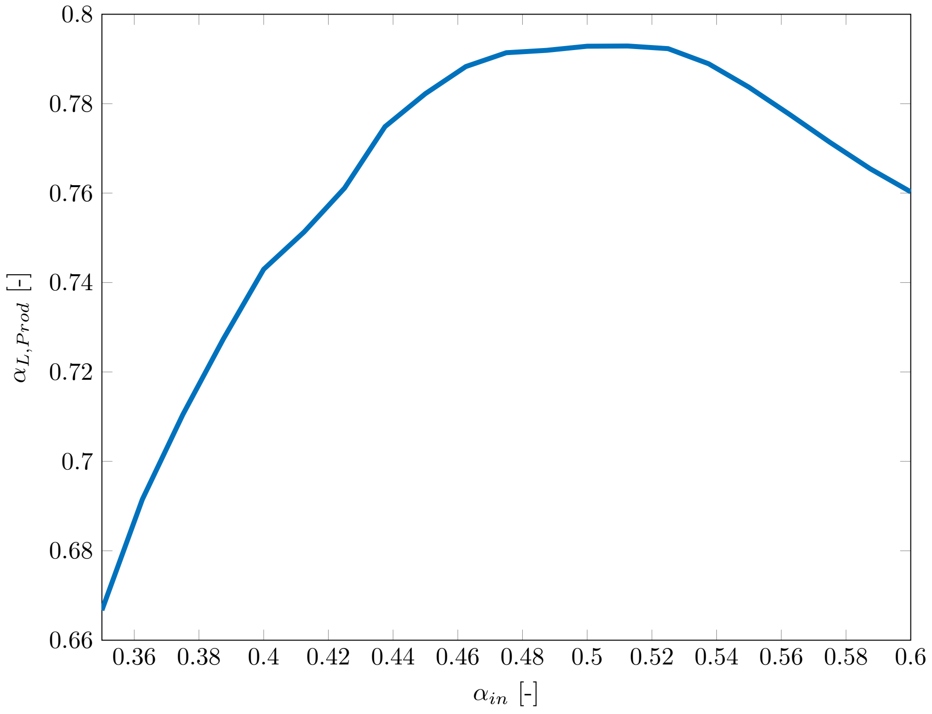

Figure 5.8: Oil fraction in production stream plotted against inlet oil fraction with a constant in flow of $23\,\mathrm{m^3\,h^{-1}}$

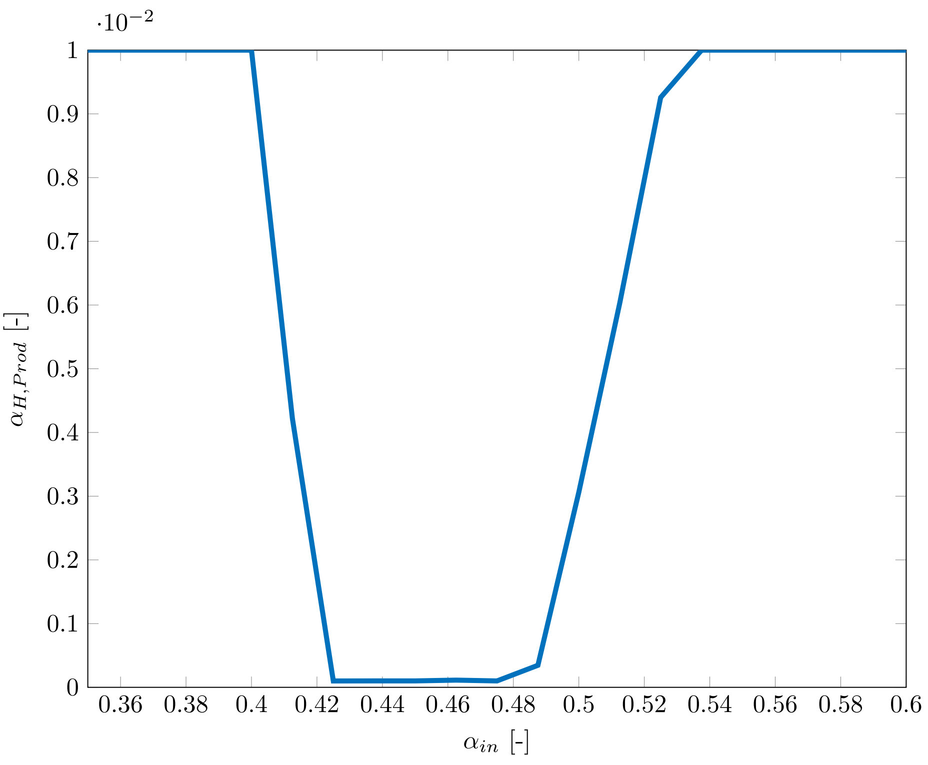

Figure 5.9: Oil fraction in water production stream plotted against inlet oil fraction with a constant in flow of $23\,\mathrm{m^3\,h^{-1}}$. The figure clearly shows that there is an unconstrained region between 0.42 and 0.49.

5.4 Control Structure Design

The main purpose of this work has been to develop methods for measurement selection for control structure design. Two nominal points were chosen and the methods described in Chapter 3 was applied. One nominal point has an active constraint, while the other lies in the unconstrained region. In addition budget constraints were added to see how this influence the choice of measurements. The final goal of this section is to find a selection matrix (H). Recall the evaluation criteria from the previous chapter:

- *Loss* is the unconstrained loss, process loss or steady state loss. It is defined in (2.26)
- *Price* is in this work defined as the prices of measurements. The price could be of one single measurement device or total price, which is the sum of prices for the selected measurements.
- *Back-off* is the constrained loss. Due to measurement uncertainty, it is unknown how close to the constraint the measurement actually is, hence a back-off away from the constrained value is required. However, when backing-off we also travel away from the optimum, and the distance away from the optimum is the constrained loss or back-off.
- *Cost* or total cost, is the sum of all the terms mentioned above. In a unconstrained case the cost (or total cost) will be the sum of the unconstrained loss and the prices of measurements. Whereas in a constrained case the total cost will be the sum of the unconstrained loss, prices of measurements and back-off. From a economic perspective, it is the total cost of a project that is of interest, and is therefore minimized.

5.4.1 Operation in Unconstrained Region

Based on the active constraint map in Figure 5.6, the point where $\alpha_{in} = 0.4$ and $q_{in} = 20\,\mathrm{m^3\,h^{-1}}$, was chosen to study a nominal point in the unconstrained region. The steady state results are given in Table 5.1 and 5.2 for inputs and measurements respectively.

Table 5.1: Nominal values for all inputs with feed at $\alpha_{in} = 0.4$ and $q_{in} = 20\,\mathrm{m^3\,h^{-1}}$.

Input	Nominal Value
FS_G	0,3612
FS_{DW}	0,9089
FS_{DO}	0,2239

As seen in the Table 5.2, the oil fraction in the water stream ($\alpha_{H,prod}$) is less than 0.01 which was the limit for the constraint.

Table 5.2: Nominal values for all measurements with feed at $\alpha_{in} = 0.4$ and $q_{in} = 20\,\mathrm{m}^3\,\mathrm{h}^{-1}$.

Measurement	Flow [$\mathrm{m}^3\,\mathrm{s}^{-1}$]	Oil fraction [-]
t	0,0020	0,8953
b	0,0035	0,1199
LPO,DW	0,0018	0,9411
HPO,DW	0,0002	0,4389
in,DO	0,0037	0,1355
LPO,DO	0,0008	0,5945
$H,prod$	0,0029	0,0031
$L,prod$	0,0027	0,8322

Sensitivity Matrix

The sensitivity matrix (F) was found using re-optimization around the nominal point. The re-optimization was done by running the simulation with a 1% increase in the disturbances, one disturbance at the time. The sensitivity matrix was then calculated by using (2.18) and assuming that:

$$F = \frac{\partial y}{\partial d} \approx \frac{\Delta y}{\Delta d} \tag{5.23}$$

Hessian Matrix

The Hessian matrix of the cost function was found using finite differences. There are three unconstrained inputs (the nominal point is not in the active constraint region), finite differences in several variables and second derivatives using the central step method were applied. The step size for both FS_{DW} and FS_{DO} was set to 0.1% of the nominal values. The equations used are given in Appendix C. The Hessian Matrix was found to be:

$$J_{uu} = \begin{bmatrix} 0.5153 & 0.5564 & 0.0869 \\ 0.5564 & 1.4047 & 0.0930 \\ 0.0869 & 0.0930 & 0.1687 \end{bmatrix} \tag{5.24}$$

To make sure that the cost function is convex, the eigenvalues were checked. Positive eigenvalues of the Hessian matrix implies that the system is positive semi-definite, and means that the system is convex. Be aware that this analysis is done locally, so it is not known if the system is globally convex. In this case all three eigenvalues were positive, and the function is therefore convex around this nominal point.

Gain Matrix

The gain matrix is the Jacobian matrix of the system, and was found using finite differences, as was the case for the Hessian matrix. The gain matrix was found to be:

$$G^y = \begin{bmatrix} 0.0020 & 0 & 0 \\ -0.5166 & 0 & 0 \\ -0.0020 & 0 & 0 \\ -0.1510 & 0 & 0 \\ 0.0027 & 0.0018 & 0 \\ -0.7346 & -0.4773 & 0 \\ -0.0007 & -0.0018 & 0 \\ -0.7506 & -0.0835 & 0 \\ -0.0027 & -0.0018 & 0 \\ -0.2323 & -0.1578 & 0 \\ -0.0006 & -0.0004 & 0.0008 \\ -0.1832 & -0.1330 & -0.2612 \\ -0.0021 & -0.0014 & -0.0008 \\ -0.2465 & -0.1650 & -0.0965 \\ 0.0021 & 0.0014 & 0.0008 \\ -0.3957 & -0.2595 & -0.1564 \end{bmatrix} \tag{5.25}$$

In (5.25) $G^y \in \mathbb{R}^{n_y \times n_u}$ each row corresponds to a measurement and each column to a input. The first four rows in the second column, represent the flow and oil fraction measurements out of the gravity separator. It is no surprise that an input change in the second input (FS_{DW}) has no effect on these measurements, since these are located upstream of the Dewaterer. The same argument can be used regarding the ten first measurements in the third column, all these measurements are located upstream of the Deoiler.

Disturbance and Measurement Noise

Define that the disturbances for the system are:

$$d_1 = q_{in} \pm 5\,\mathrm{m}^3\,\mathrm{h}^{-1} \tag{5.26}$$

$$d_2 = \alpha_{in} \pm 0.2 \tag{5.27}$$

This gives a disturbance W_d of:

$$W_d = diag(\begin{bmatrix} 5 & 0.2 \end{bmatrix}) \tag{5.28}$$

In order to set the prices to each measurement it is assumed that the flow measurements are cheaper than the oil fraction measurements, since the latter are much more complicated. More importantly, the prices of measurements must be given as annualized costs (or more precise \$/s), as oppose to fixed cost. This is due to how the cost function is defined. In a subsea case, it is possible to think that the operator company outsources the subsea operation and maintenance of the measurements for a given number of years to a service company. Two set of measurement devices (w_1 and w_2) with a given measurement noise (W_n) and prices are defined in Table 5.3. Note that the values given in the table are not based on actual data, but are set in what is considered a realistic order of magnitude, relative to the actual model and cost function formulation.

Table 5.3: Measurement noise and price, for different measurement equipment with two different measurement device sets.

	w_1	w_2
Oil fraction noise [-]	± 0.001	± 0.02
Flow noise [$\mathrm{m^3\,h^{-1}}$]	± 0.01	± 0.4
Price oil fraction measurements [$/s]	2×10^{-3}	1×10^{-3}
Price flow measurements [$/s]	1×10^{-3}	5×10^{-4}

Unconstrained Loss

A big-**M** of 1000 was chosen, in order for the system not to reach the big-**M** constraint, as this may influence the calculated loss for the system. The system was implemented as described in Section 3.3. The unconstrained loss for each number of measurement was calculated, and is shown in Figure 5.10. The figure

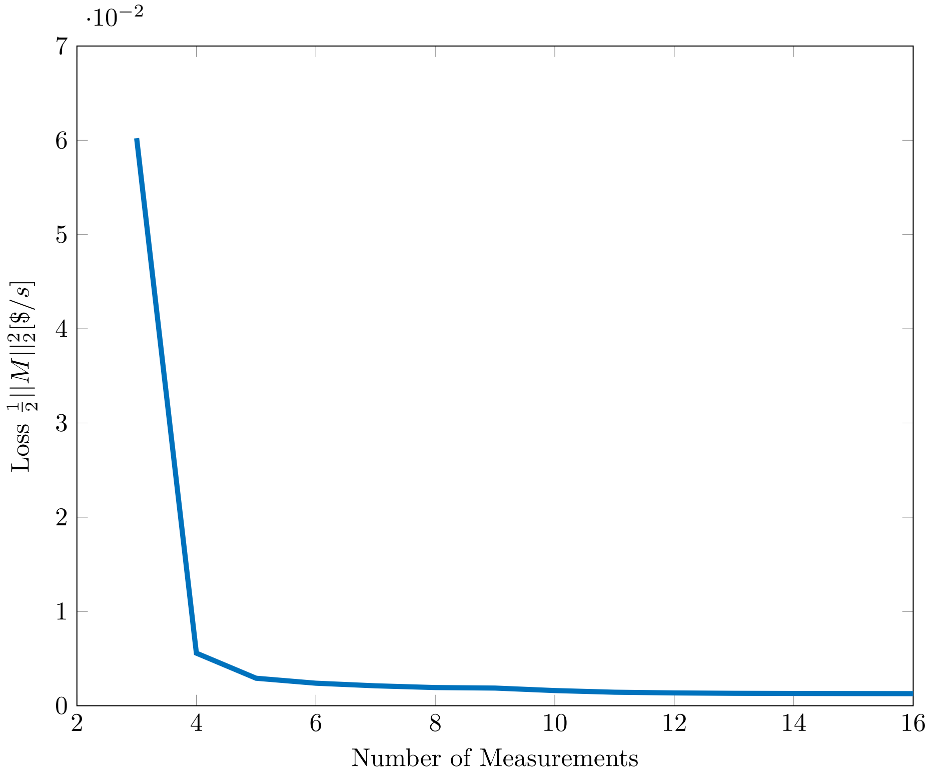

Figure 5.10: The unconstrained loss for the system operating in the unconstrained region.

shows that the unconstrained loss decreases rapidly for the first few measurements before flattening out at a very small loss when more measurements are added.

Total Cost

The total cost of the system (unconstrained loss + prices of measurements) is given in Figure 5.11. As seen in the figure, the optimal number of measurements

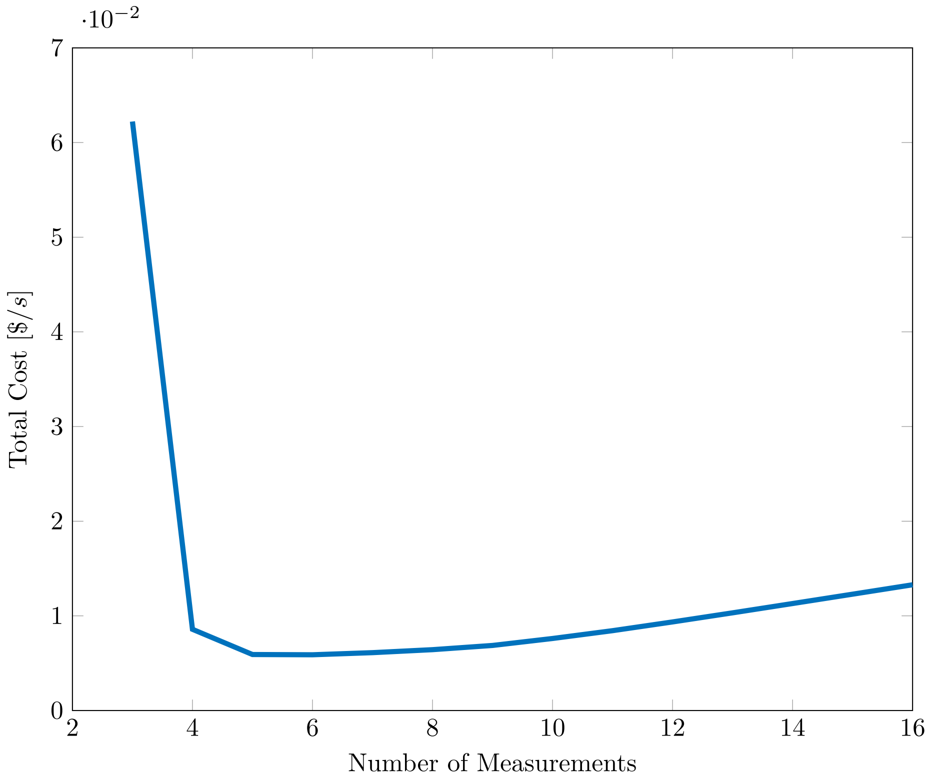

Figure 5.11: The total cost for the system operating in the unconstrained region. Total cost includes prices of measurements and unconstrained loss.

(number that gives the lowest total cost) is six. After six measurements, the prices of the additional measurements dominates over the decrease in unconstrained loss. Note that the difference between five and six measurements is very small, and the engineer may therefore decide to select only five in a more realistic case.

Analysis of the Selected Measurements

The costs between four and seven measurements are investigated to see which part of the total cost contributes so that the optimal number of measurements are six. There are, in this unconstrained case, two components that makes up the total cost, the unconstrained loss, the prices of measurement. A bar plot showing how each of these components make up the total cost for four to seven measurements, are given in Figure 5.12. A table showing the selected measurements, when a sub-

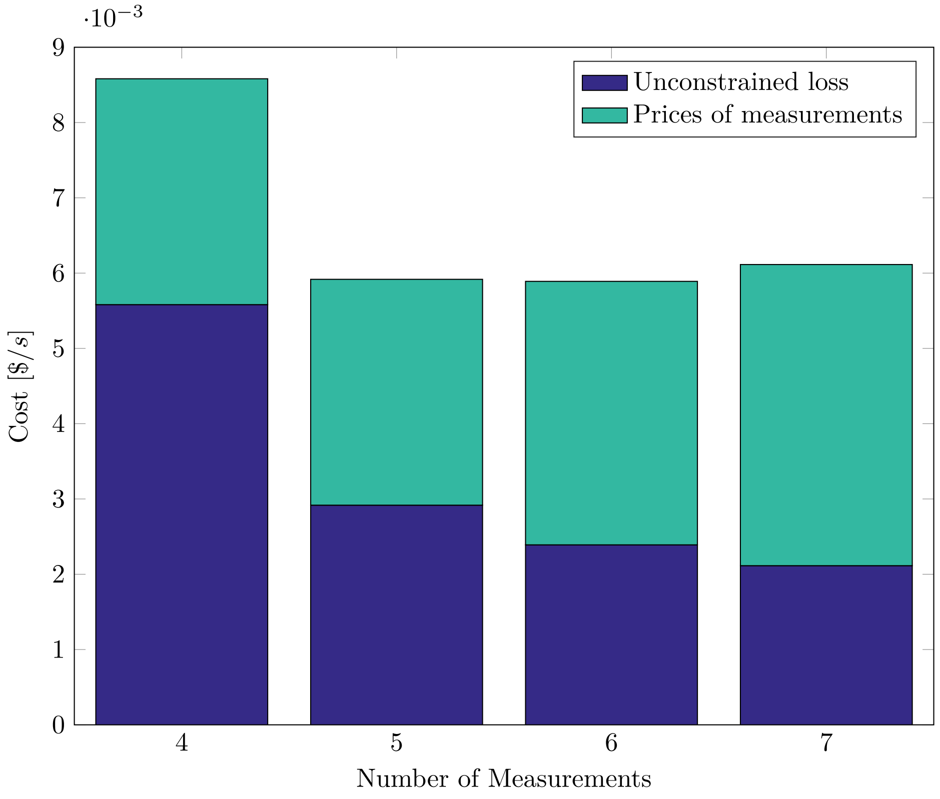

Figure 5.12: The total cost for the system separated in each of the components making up the total cost.

set of four to seven measurements is chosen is given in Table 5.4. As seen in Table 5.4 all measurements comes from the cheap measurement set (w_2). When more measurements are selected the loss decreases whereas the price of measurements increases. Note that the optimal combination always include one oil fraction measurement. It has been established that choosing five flow measurements and one oil fraction measurement from the inaccurate measurement set (w_2) gives the best trade-off between measurement costs and loss. The final selection matrix (H) is now presented and could be sent to a controller. The selection matrix is given in (5.29). Only measurements included in the subset are given in the selection matrix.

$$H = \begin{array}{c c} & \begin{array}{cccccc} q_b w_2 & q_{LPO,DW} w_2 & q_{HPO,DW} w_2 & q_{in,DO} w_2 & q_{LPO,DO} w_2 & \alpha_{LPO,DO} w_2 \end{array} \\ \begin{array}{c} FS_G \\ FS_{DW} \\ FS_{DO} \end{array} & \begin{pmatrix} -110.52 & 12.42 & -22.56 & -133.08 & 37.18 & -0.15 \\ 295.31 & 55.00 & -399.51 & -104.20 & -125.59 & -0.58 \\ -10.13 & -9.54 & -2.55 & -12.67 & 202.37 & -0.89 \end{pmatrix} \end{array} \tag{5.29}$$

Table 5.4: Selected measurements including the unconstrained loss and the total prices of measurements.

No. y	Selected Measurements	L^{uncons}	Price
4	$q_b w_2$, $q_{HPO,DW} w_2$, $\alpha_{HPO,DW} w_2$, $\alpha_{LPO,DO} w_2$	0.0056	0.003
5	$q_b w_2$, $q_{LPO,DW} w_2$, $q_{HPO,DW} w_2$, $q_{LPO,DO} w_2$, $\alpha_{LPO,DO} w_2$	0.0029	0.003
6	$q_b w_2$, $q_{LPO,DW} w_2$, $q_{HPO,DW} w_2$, $q_{in,DO} w_2$, $q_{LPO,DO} w_2$, $\alpha_{LPO,DO} w_2$	0.0024	0.0035
7	$q_t w_2$, $q_b w_2$, $q_{LPO,DW} w_2$, $q_{HPO,DW} w_2$, $q_{in,DO} w_2$, $q_{LPO,DO} w_2$, $\alpha_{LPO,DO} w_2$	0.0021	0.004

A flowsheet with the chosen measurement is presented in Figure 5.13 to give the viewer a better systematic look. The chosen control structure have a very small loss, it combines both flow and oil fraction measurements. Since the inputs u are flow splits, it makes sense to use more flow measurements than oil fraction measurements, as these are directly a consequence of the given flow split. Based on the chosen cost function, which aims to maximize the total amount of oil (not just maximize flow or maximize oil fraction) it also make sense that one oil fraction measurement is needed and this is located close to the final production stream. The five flow measurements is located around the whole flow sheet, but not directly on the oil product stream. The reason for this may be that it is better to choose to pair the measurement and input as close as possible, which is suggested by Larsson and Skogestad [23]. Of the chosen measurements, $q_{in,DO}$ may seem a little unnecessary since both q_b and $q_{HPO,DW}$ are chosen. $q_{in,DO}$ is also the last measurement to be added as seen in Table 5.4, and does not contribute a great deal to the loss reduction. However, by knowing both $q_{in,DO}$ and $q_{LPO,DO}$ it should be possible to give a very good controller values to FS_{DO} which actually is defined as: $q_{LPO,DO}/q_{in,DO}$. This is confirmed by the values in the selection matrix H where

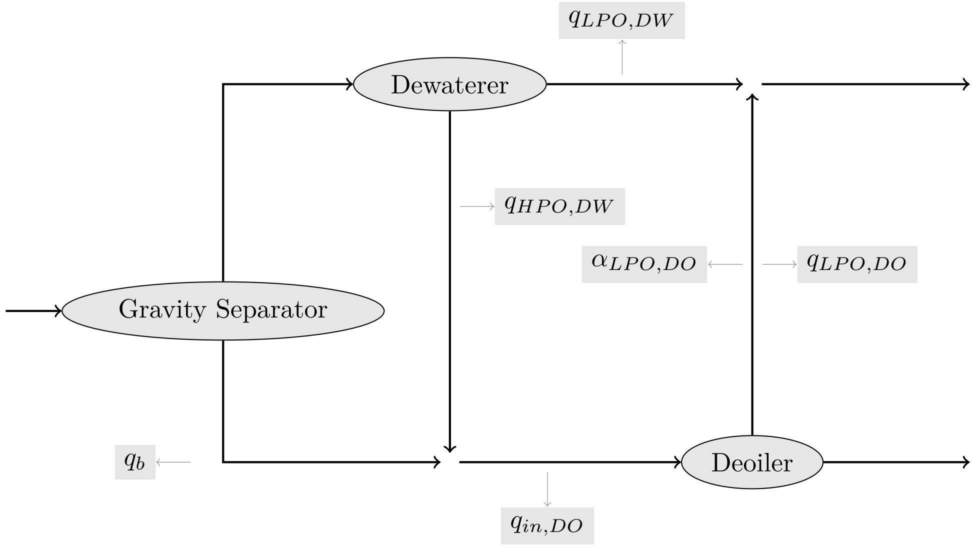

Figure 5.13: Flowsheet of the subsea processing system where only the selected measurements are shown.

$q_{in,DO}$ and $q_{LPO,DO}$ has the highest weight for the input FS_{DO}.

Budget constraint

Given the measurement noise and the prices, the best measurement combination was found above. However, it is possible that the project runs into financial difficulties, in that case the engineer may be asked to cut the investment cost (installation of measurements) in the project, for instance by 20%. The previous price of measurements was 0.0035, as seen in Table 5.4. This is then reduced. How much will the loss increase, and what is the new selected subset of measurements? This budget constraint is implemented through a new linear constraint defined as:

$$\sum_{k}^{n_w}\sum_{j}^{n_y} p_{\bar{j},k}\sigma_{j,k} \leq 0.0028 \tag{5.30}$$

where $p_{\bar{j},k}$ is the price of measurement $_j$ in set $_k$. This will naturally limit the results to a large extend. The selected measurements with corresponding unconstrained loss and prices of measurements are given in Table 5.5. As seen in the table, the optimal number of measurements was four for the case with budget constraint. Note that four measurements is the absolute best case, this is because that it has the crucial oil fraction measurement, which is removed when choosing five measurements. The corresponding selection matrix, H is given in (5.31). Each component correspond to the measurements given in Table 5.5, in the same order

Table 5.5: Selected measurements including unconstrained loss and the total prices of measurement with the budget constraint active.

No. y	Selected Measurements	L^{uncons}	Price
3	$q_b w_2$, $q_{HPO,DW} w_2$, $\alpha_{H,prod} w_2$	0.06	0.002
4	$q_b w_2$, $q_{HPO,DW} w_2$, $q_{LPO,DO} w_2$, $\alpha_{LPO,DO} w_2$	0.0063	0.0025
5	$q_b w_2$, $q_{LPO,DW} w_2$, $q_{HPO,DW} w_2$, $q_{LPO,DO} w_2$, $q_{H,prod} w_2$	0.17	0.0025

as in the table.

$$H = \begin{array}{c} \\ FS_G \\ FS_{DW} \\ FS_{DO} \end{array} \begin{array}{c} \begin{array}{cccc} q_b w_2 & q_{HPO,DW} w_2 & q_{LPO,DO} w_2 & \alpha_{LPO,DO} w_2 \end{array} \\ \begin{pmatrix} -255.66 & -167.67 & 36.33 & -0.15 \\ 136.98 & -557.78 & -127.63 & -0.59 \\ -13.37 & -5.80 & 202.62 & -0.89 \end{pmatrix} \end{array} \tag{5.31}$$

A comparison of the best subset with and without budget constraint is given in Figure 5.14. As illustrated in the figure, the prices of measurements has dropped, but the controller is not able to get a good view of the system and hence, the unconstrained loss increases. This is as expected.

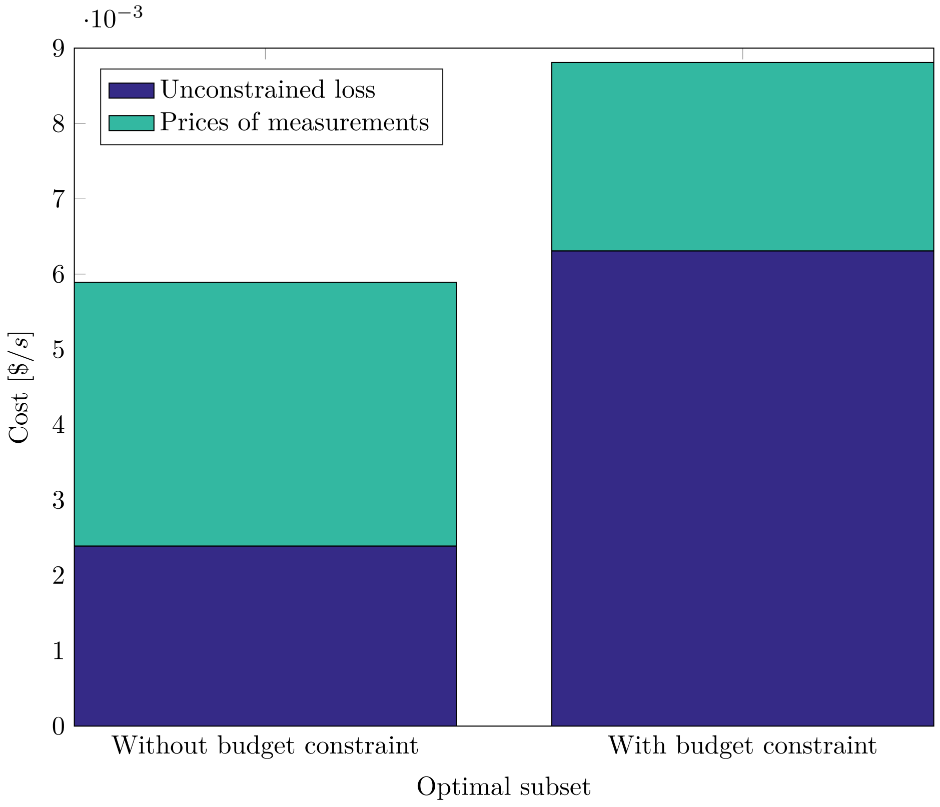

Figure 5.14: Comparison of best measurement selection for cases with and without budget constraint. With budget constraint, the selected measurement set consists of four measurements, whereas it consists of six for the case without the budget constraint.

5.4.2 Operation in Active Constraint Region

Based on the active constraint map in Figure 5.6, the point where $\alpha_{in} = 0.6$ and $q_{in} = 23\,\mathrm{m^3\,h^{-1}}$ was chosen for the study of a nominal point with an active constraint. The nominal values are given in Table 5.6 and 5.7, for the inputs and the measurements respectively.

Table 5.6: Nominal valus for the input with feed at $\alpha_{in} = 0.6$ and $q_{in} = 23\,\mathrm{m^3\,h^{-1}}$.

Input	Nominal Value
FS_g	0.7522
FS_{DW}	0.9172
FS_{DO}	0.3110

Table 5.7: Nominal values for all measurements with feed at $\alpha_{in} = 0.6$ and $q_{in} = 23\,\mathrm{m^3\,h^{-1}}$.

Measurement	Flow $[\mathrm{m^3\,s^{-1}}]$	Oil fraction [-]
t	0.0048	0.74343
b	0.0016	0.1923
$LPO{,}DW$	0.0044	0.7535
$HPO{,}DW$	0.0004	0.5221
$in{,}DO$	0.0020	0.2585
$LPO{,}DO$	0.0006	0.8091
$H{,}prod$	0.0014	0.01
$L{,}prod$	0.0050	0.7603

Note that $\alpha_{H,prod} = 0.01$ which indicates that the constraint is active. The Lagrange multiplier at this nominal point was found to be 0.8817. In order to calculate the gain matrix and the Hessian, each row and column corresponding to either the input FS_{DO} or the measurement $\alpha_{H,prod}$ is removed, as seen in Section3.2, so that $J_{uu} \in \mathbb{R}^{2\times 2}$ and $G^y \in \mathbb{R}^{n_y-1\times n_u-1}$. The Hessian matrix was found to positive semi-definite, as all eigenvalues were positive, so that the system is convex around this nominal point. The noise and prices for each measurements are assumed to be the same as in the unconstrained case.

Unconstrained Loss

A big-**M** of 1000 was chosen. A selection criteria for the constraint was created (the optimizer can still choose from $\alpha_{H,prod}w_1$ and $\alpha_{H,prod}w_2$). A term calculating the back-off due to the constraint was also added to the simulation. As shown in Figure 5.15, the unconstrained loss decreases to a minimum at four and five measurements, before increasing again from five to six. This is because the optimizer shifts the measurement set from choosing only accurate measurements (w_1) to more inaccurate (w_2). This is shown in Table 5.8. Note that as oppose to cases where

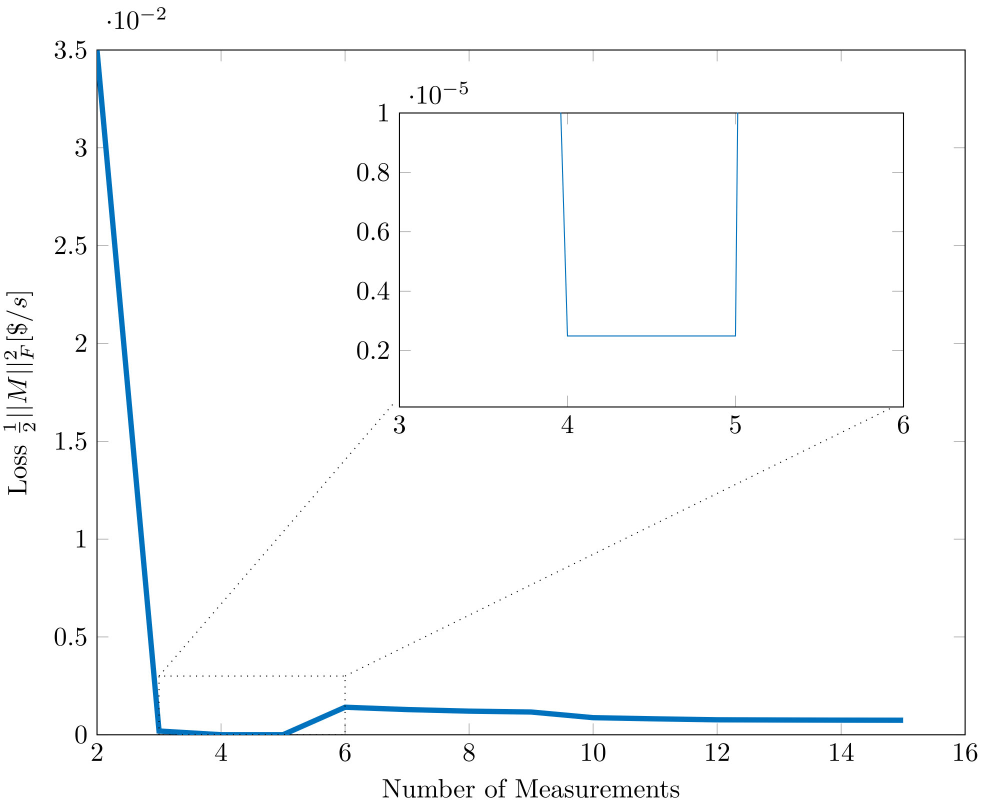

Figure 5.15: The unconstrained loss for the system operating in the constrained region. A zoom-in of the plot is included in the area between three and six selected measurements, however, the values for four and five are several order of magnitudes smaller than three and six. The measurement chosen for the constraint is not included in the counting of measurements.

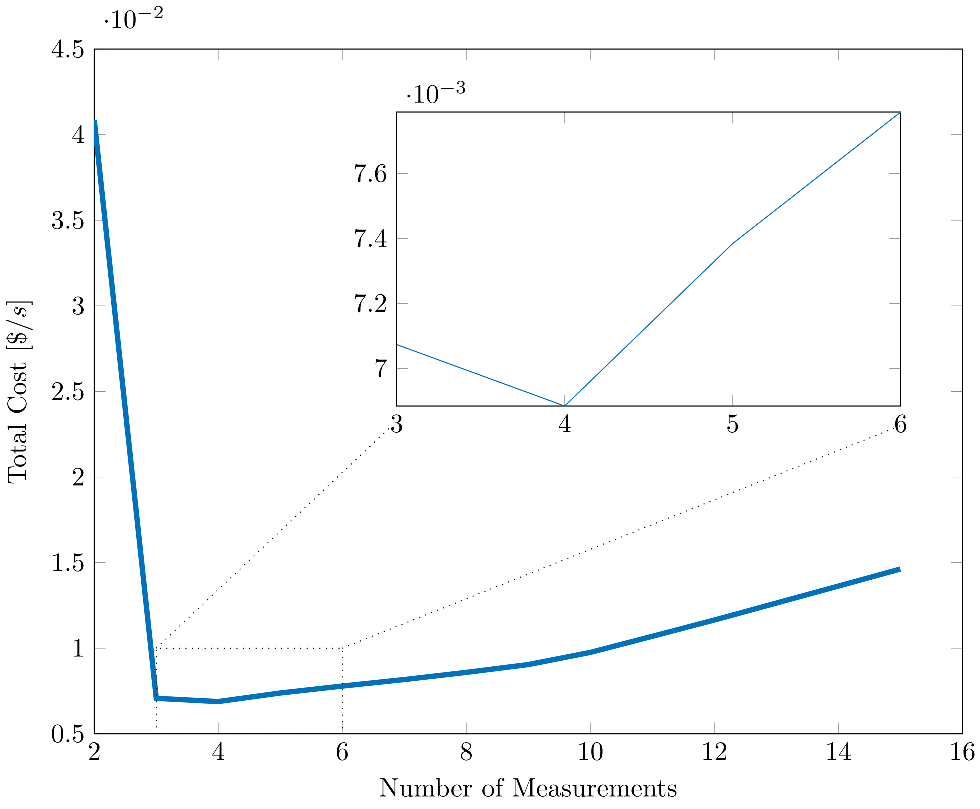

Figure 5.16: The total cost of the system operating in the constrained region, including prices of measurements, unconstrained loss and constrained back-off. A zoom-in of the plot is included in the area between three and six measurements to better see the details in this part of the graph.

the prices are set to zero, it is possible that the minimum value of the loss comes at fewer measurements than the maximal number of measurements. This is because the optimizer minimizes the total cost for each number of measurements and not just the unconstrained loss. When many measurements are chosen the optimizer often selects the «bad» measurements as the loss will be relatively small anyway.

Total Cost

The total cost of a constrained system consist of unconstrained loss, prices of measurements and constrained back-off. The total cost of the system with an active constraint is given in Figure 5.16. As seen in the figure, the lowest total cost is at four (unconstrained) measurements. However, the difference is very small between four and three measurements.

In order to study which part makes up the total cost, a bar plot separating each part of the cost component was created. The plot is given in Figure 5.17. The

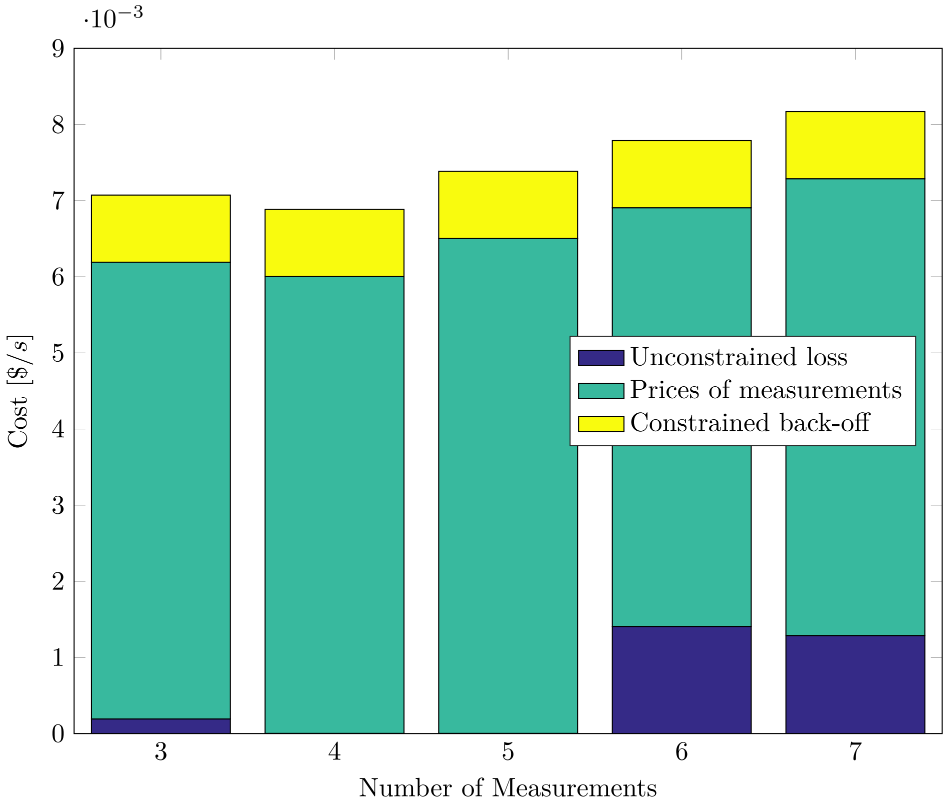

Figure 5.17: The total cost for three to seven unconstrained measurements separated into each of the components, making up the total cost.

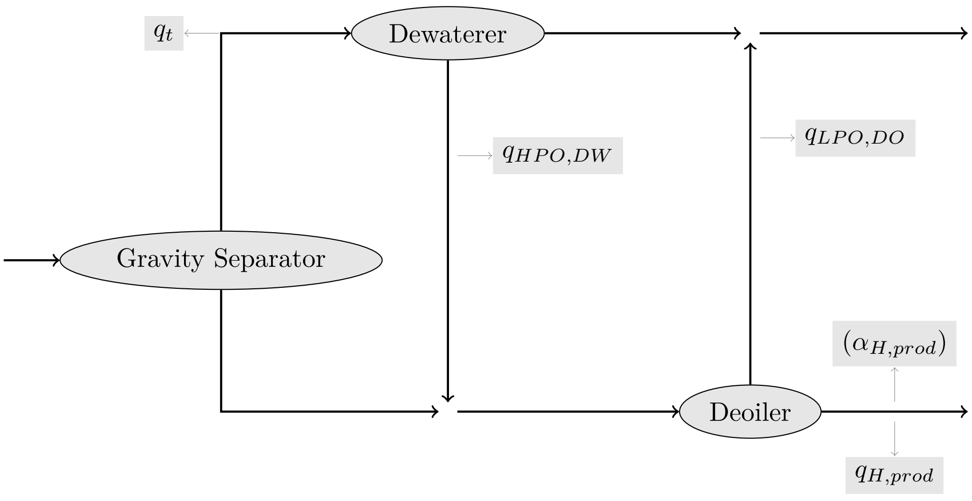

Figure 5.18: Flowsheet of the subsea processing system where only the selected measurements are shown. $\alpha_{H,prod}$ is given in parenthesis since this is a measurement for a constrained input.

figure illustrates that most of the total cost is the prices of measurements. The constrained back-off make up a considerable part of the total cost. The unconstrained loss is as has been shown in Figure 5.15 almost negligible for four and five selected measurements. Be aware that this nominal point is actually the point where the Lagrangian multiplier of the constraint has its maximum value of the whole investigated region (as seen in Figure 5.6).

Analysis of the Selected Measurements

The selected subset for three to seven (unconstrained) measurements are investigated. The selected subsets are given in Table 5.8. As seen earlier, the optimal number of unconstrained measurements is four. The selected measurements are all from the accurate and expensive set (w_1). Only flow measurements are chosen. However, when increasing the number of measurements to five, the additional measurements are flow measurements from the inaccurate and inexpensive measurement set (w_2). If more than five measurements are selected, all measurements comes from w_2. Note that the expensive measurement is chosen for the constraint, this is because the Lagrangian is so large that it makes up a considerable part of the total cost. It was therefore important to minimize the noise for the constrained back-off. The chosen measurements are drawn on the flow sheet in Figure 5.18. The final selection matrix for the optimum number of measurements is given in

Table 5.8: Selected measurements including the unconstrained loss and the total prices of measurements. The chosen measurement device for the constraint is included in the selected measurements and counted through (+1).

No. y	Selected Measurements	L^{uncons}	Price
3(+1)	$q_t w_1$, $q_{HPO,DW} w_1$, $\alpha_{in,DO} w_1$, $(\alpha_{H,prod} w_1)$	0.00019	0.006
4(+1)	$q_t w_1$, $q_{HPO,DW} w_1$, $q_{LPO,DO} w_1$, $q_{H,prod} w_1$, $(\alpha_{H,prod} w_1)$	2.5e-06	0.006
5(+1)	$q_t w_1$, $q_{HPO,DW} w_1$, $q_{LPO,DO} w_1$, $q_{H,prod} w_1$, $q_{L,prod} w_2$, $(\alpha_{H,prod} w_1)$	2.5e-06	0.0065
6(+1)	$q_t w_2$, $\alpha_b w_2$, $q_{LPO,DW} w_2$, $q_{HPO,DW} w_2$, $q_{LPO,DO} w_2$, $q_{H,prod} w_2$, $(\alpha_{H,prod} w_1)$	0.0014	0.0055
7(+1)	$q_t w_2$, $q_b w_2$, $\alpha_b w_2$, $q_{HPO,DW} w_2$, $q_{in,DO} w_2$, $q_{LPO,DO} w_2$, $q_{H,prod} w_2$, $(\alpha_{H,prod} w_1)$	0.0013	0.006

(5.32)

$$H = \begin{array}{c} \\ FS_g \\ FS_{DW} \end{array} \begin{array}{c} \begin{array}{cccc} q_t w_1 & q_{HPO,DW} w_1 & q_{LPO,DO} w_1 & q_{H,prod} w_1 \end{array} \\ \begin{pmatrix} 52.15 & 293.78 & -560.73 & 256.00 \\ -66.98 & -370.36 & -174.47 & 68.93 \end{pmatrix} \end{array} \tag{5.32}$$

where each column in H corresponds to the measurements listed in Table 5.8. The selection matrix now only has two rows since FS_{DO} is controlled by $\alpha_{H,prod}$. When comparing the constrained case to the unconstrained case, the unconstrained loss is much smaller, since the measurements are selected from the accurate and expensive set (w_1). However, the prices of measurements makes the total cost higher for the constrained case (6×10^{-3} \$/s vs. 7×10^{-3} \$/s) which is due to the expensive oil fraction measurement used to reduce the back-off and control the constraint. When it comes to which measurement that was selected one could argue that both $q_{HPO,DW}$ and $q_{LPO,DO}$ plays an important role as fluid can be cross-fed along these points. In addition $q_{HPO,DW}$ has the highest weighting in the selection matrix for the input FS_{DW} which make sense as this is the measurement mostly affected by a change in the Dewaterer. q_t could be needed to pair close to the gravity separator.

Budget Constraint

A budget constraint was added so that the prices of measurements are reduced by at least 20% from the previous optimum. The sum of all prices has to be smaller than 0.0048. As opposed to the unconstrained case in the previous section, where the optimal selected measurements without budget constraint were mostly flow measurements from the inexpensive measurement set (w_2), this nominal point has a better price saving possibility as the selected subset consisted of flow measurements from the expensive device set (w_1), as seen in Table 5.8. The results from the simulation with an active budget constraint is given in Figure 5.19, which shows the total cost for two to seven (unconstrained) measurements. The optimal number of measurements with the budget constraint is still four. However, the difference between four and three is very small. The selected measurements for three to seven (unconstrained) measurements are investigated, and the measurements as well as the corresponding unconstrained loss and price is shown in Table 5.9. As shown in the table, the optimizer has, due to the budget constraint, a tendency to choose more measurements from the second device set (w_2) than previously.

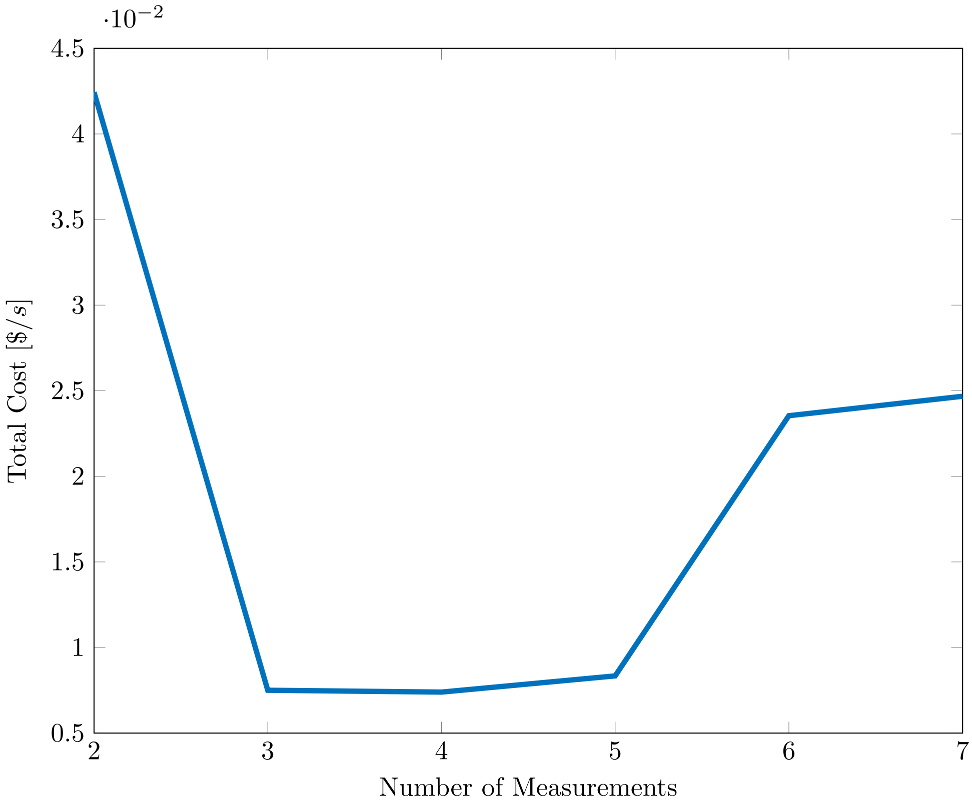

Figure 5.19: Total cost of the system operating in the constrained region, with budget constraint. Only measurements two to seven are possible.

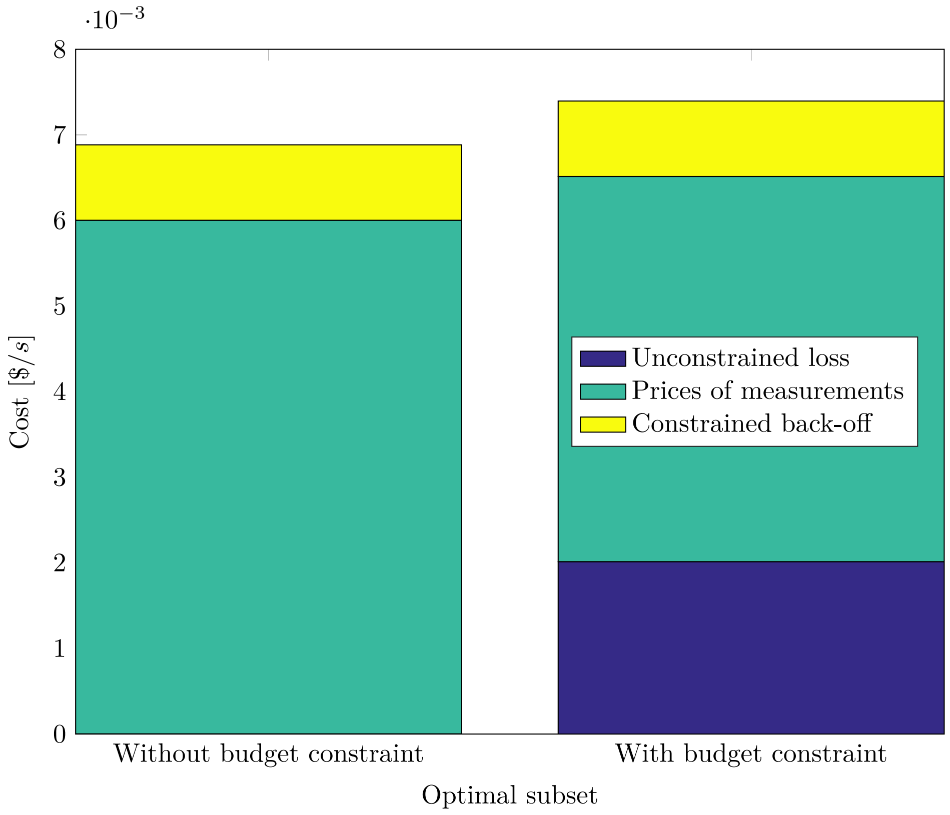

Figure 5.20: Comparison of the loss and prices of measurements with and without the budget constraint.

The final selection matrix H for the case with active budget constraint is given in (5.33)

$$H = \begin{array}{c} \\ FS_g \\ FS_{DW} \end{array} \begin{array}{cccc} q_t w_2 & \alpha_b w_2 & q_{HPO,DW} w_2 & q_{H,prod} w_2 \\ \multicolumn{4}{c}{\left(\begin{array}{cccc} 5.22 & -1.50 & -27.83 & 130.12 \\ -82.25 & -0.47 & -469.96 & 28.40 \end{array} \right)} \end{array} \tag{5.33}$$

where each column of H corresponds to the measurements listed in Table 5.9. The optimum subset of measurements is compared to the case with no budget constraint. The result is shown in Figure 5.20. As can be seen in the figure, adding a budget constraint naturally increases the unconstrained loss and the prices of measurements have decreased, but not so much that it compensates for the increase in unconstrained loss.

Table 5.9: Selected measurements including the unconstrained loss and the total prices, with active budget constraint. The chosen measurement device for the constraint is included in the selected measurements and counted through (+1).

No. y	Selected Measurements	L^{uncons}	Price
3(+1)	$q_t w_2$, $q_{HPO,DW} w_2$, $\alpha_{in,DO} w_2$, $(\alpha_{H,prod} w_1)$	0.0026	0.004
4(+1)	$q_t w_2$, $\alpha_b w_2$, $q_{HPO,DW} w_2$, $q_{H,prod} w_2$, $(\alpha_{H,prod} w_1)$	0.002	0.0045
5(+1)	$q_t w_2$, $q_{HPO,DW} w_2$, $q_{LPO,DO} w_2$, $q_{H,prod} w_2$, $q_{L,prod} w_2$, $(\alpha_{H,prod} w_1)$	0.003	0.0045
6(+1)	$q_t w_2$, $\alpha_b w_2$, $q_{LPO,DW} w_2$, $q_{HPO,DW} w_2$, $q_{LPO,DO} w_2$, $q_{H,prod} w_2$, $(\alpha_{H,prod} w_2)$	0.0014	0.0045
7(+1)	$q_t w_2$, $q_b w_2$, $q_{LPO,DW} w_2$, $q_{HPO,DW} w_2$, $q_{LPO,DO} w_2$, $q_{H,prod} w_2$, $q_{L,prod} w_2$, $(\alpha_{H,prod} w_2)$	0.0025	0.0045

Chapter 6

Discussion and Conclusion

A method for combining the cost of measurements with the corresponding process losses due to measurement uncertainty, has been presented within the self-optimizing control framework. The goal in self-optimizing control is to identify an optimal control structure design. This has been done by using MIQP optimization tools. By using the MIQP-formulation it become possible to extend the framework to also include different sets of measurements and one was able to find the best trade-off between accurate (and expensive) and inaccurate (and inexpensive) measurements. A method of including a process constraint and calculate the necessary back-off loss has also been implemented. It was shown that the back-off makes up a considerable part of the loss when the constraint is active, and that the constrained back-off can be larger than the unconstrained loss. The MIQP-formulations also handles budget constraints. It was shown that the total cost of a project increases due to higher unconstrained loss when a budget constraint is added. The methods were applied to both a Dummy case, for proof-of-concept purposes, and a more advanced Subsea separation process where all the developed methods where implemented simultaneously. The Dummy problem presented was proven to produce results similar to already published literature[4], before the new methods were applied. All in all, this work has shown that it is possible to link economical considerations with the design of the control structure, and this could be a step to introduce process control ideas directly into the process design phase.

6.1 Limitations of the Presented Methods

The methods presented in this work have, however, some limitations:
In this work it has been assumed that the disturbance does not change the active constraints. However, the plant may be operated in different ways depending on which of the constraints are active. Some approaches to this problems has been proposed in literature, for instance by the multi-parametric programming approach [24]. When comparing the selected subset in the active constraint and the uncon-

strained region, the selected measurements are not necessarily the same, this may lead to losses.
Another limitation is that the results are only valid locally around the nominal point, this is of course a simplification, and may be OK for a stable system running in steady state. However, if large disturbances occur, this may lead to large losses because of the selected control structure design. The self-optimizing control framework is therefore not applicable to highly dynamic systems (for instance batch processes). The resulting measurement combination for controlled variables must therefore be seen as good candidate for control structure rather than necessarily be the final one.

6.2 Uncertainty and Price of Measurements

The main focus in this work has been to present methods. The selection of the actual prices and measurement uncertainty has therefore not been investigated in detail. For instance, flow measurements has been given a constant uncertainty independent of the actual flow. This full scale accuracy means that the error in percent is much higher at smaller volumetric flows, this could have had an impact on the selection of measurements. The magnitude of the noise is also questionable. For instance, it was assumed that inaccurate flow measurement has an full scale uncertainty of $5 \times 10^{-4}\,\mathrm{m}^3\,\mathrm{h}^{-1}$, which is 2% given that the maximum is $20\,\mathrm{m}^3\,\mathrm{h}^{-1}$. This is the maximum uncertainty listed for volumetric flow meters according to Perry and Green [25, Chapter 10-15]. Since most of the flows in the system is less than $20\,\mathrm{m}^3\,\mathrm{h}^{-1}$, the error in percent will be greater, due to the full scale accuracy. The prices set to each type of measurement is given with the following assumptions: 1) Flow meters are cheaper than oil fraction measurements. 2) Inaccurate measurements are cheaper than more accurate devices. Besides that, prices are only set so that they are in what could be excepted to be a reasonable range. For instance 0.006 \$/s is equivalent to 189 000 \$/year

6.3 Some Words About Numeric Issues

Although the MIQP-formulations has a lot of benefits, the reader should be aware of some issues that may occur: It is important, when working with MIQP-formulations, to be aware that finding the optimal solution is not guaranteed [26]. This is because mixed integer solvers does not normally evaluate all possible combinations (picking eight measurements out of 30 candidates has for instance almost 6 million combinations). Instead, the solvers really on pre-solving techniques to reduce the number of possibilities and hence reduce the calculation time[27]. However, this may lead to inaccurate results, as seen in Figure 6.1.

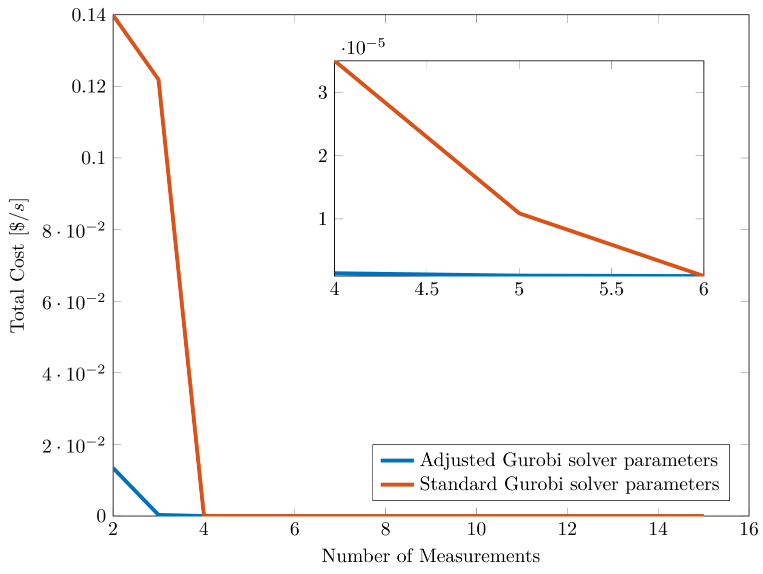

Figure 6.1: Comparing solver parameters

The following Gurobi solver parameters were found to have an effect in order to improve the results (taken from scripts in Appendix A):

```
params.Presolve = 2;
params.MIPFocus = 3;
params.NumericFocus = 3;
params.IntFeasTol = 1e-9;
```

`Presolve` is set to aggressive (2 of 2) which means it takes more time, but leads to a tighter model. `MIPfocus` is set to 2 (of 3) to improve optimality. `NumericFocus` is set to 3 (of 3) so that the solver is much more careful in numerical calculations. `IntFeasTol` is an abbreviation for Integer Feasibility Tolerance. When solving integer problems, a solution value may be 1×10^{-7} when it should be 0. This will affect the loss calculations (making smaller loss than what is realistic, since the value of 1×10^{-7} is included in the selection matrix, but not counted as a measurement), the tolerance is therefore set to its tightest level.
It should also be noted that using a large enough value for big-**M** is important. The values in the selection matrix H, should never be close to this value in order to not affect the loss calculations.
In general, numerical uncertainties regarding the MIQP-formulations provides a drawback to the methods presented in this work. Numerical methods (Finite difference method) were also applied to calculate Hessian and Jacobian matrices. These matrices are very important to have correct, since they make up the crucial $G^y H = J_{uu}$ constraint that is needed to find the minimum loss. Hence, using (simple) numerical methods to calculate these matrices rises the uncertainty of the results.

6.4 Further Work

Although the methods presented here has been tested for two cases, these have been relatively simple compared to many chemical engineering plants. Especially the number of inputs have been very limited. The methods should therefore be tested on more complex plants in order to verify their validity.
It is also of interest to verify the performance by implementing the control structure with a controller in a dynamic model, where dead time is an issue. One could for instance also include some time delay in the oil fraction measurements. As these measurements may be slower than a simpler flow measurement.
There has been a number of assumptions, especially regarding the implementation of the loss from the constraint. A natural extension of this work is therefore to; 1) increase the number of constraints, 2) Let the selection of which measurement and input that is needed to control the constraint be selected by the control structure, instead of letting this be predetermined.
There are also open issues within the self-optimizing control framework, which this work is a part of [6], that could be of interest. Specific issues related to this work are: 1) Further integration of the control structure into the design phase of the project and 2) Active set changes due to multiple active constraint regions. The latter is important while using the present methods will most likely produce different measurement subsets, as already seen when comparing Table 5.4 and 5.8.
As explained in Section 6.2, measurement noise has only been implemented with full scale accuracy. However, this may lead to large errors in percent, since a lot of the flows in the subsea case is very small. A natural continuation could be to investigate noise given in rate accuracy.

Bibliography

[1] P. Tyvold. Modeling and optimization of a subsea oil-water separation system. Master's thesis, 2015. URL `http://www.nt.ntnu.no/users/jaschke/Masters/MasterTheses/2015/Tyvold/thesis.pdf`. Accessed 1.3.2016.

[2] Laurens Joseph Arnold Marie Van Campen. *Bulk dynamics of droplets in liquid-liquid axial cyclones.* TU Delft, Delft University of Technology, 2014.

[3] Sigurd Skogestad. Plantwide control: the search for the self-optimizing control structure. *Journal of Process Control*, 10(5):487 – 507, 2000. doi: http://dx.doi.org/10.1016/S0959-1524(00)00023-8.

[4] Ramprasad Yelchuru and Sigurd Skogestad. Convex formulations for optimal selection of controlled variables and measurements using mixed integer quadratic programming. *Journal of Process Control*, 22(6):995–1007, 2012.

[5] R. Fantoft O.T. McClimans. Status and new development in subsea processing. Technical report, Offshore Technology Conference, 2006. URL `https://www.onepetro.org/download/conference-paper/OTC-17984-MS?id=conference-paper%2FOTC-17984-MS`. Accessed 21.4.2016.

[6] Johannes Jäschke, Yi Cao, and Vinay Kariwala. Self-optimizing control - a framework for controlled variables selection. Manuscript in preparation, contact: `johannes.jaschke@ntnu.no`.

[7] J.J. Downs and E.F. Vogel. Industrial challenge problems in process control a plant-wide industrial process control problem. *Computers & Chemical Engineering*, 17(3):245 – 255, 1993. doi: http://dx.doi.org/10.1016/0098-1354(93)80018-I.

[8] Ivar J Halvorsen, Sigurd Skogestad, John C Morud, and Vidar Alstad. Optimal selection of controlled variables. *Industrial & Engineering Chemistry Research*, 42(14):3273–3284, 2003.

[9] Vidar Alstad. Studies on selection of controlled variables. 2005.

[10] Vinay Kariwala, Yi Cao, and S Janardhanan. Local self-optimizing control with average loss minimization. *Industrial & Engineering Chemistry Research*, 47(4):1150–1158, 2008.

[11] Vidar Alstad and Sigurd Skogestad. Null space method for selecting optimal measurement combinations as controlled variables. *Industrial & engineering chemistry research*, 46(3):846–853, 2007.

[12] Vidar Alstad, Sigurd Skogestad, and Eduardo S Hori. Optimal measurement combinations as controlled variables. *Journal of Process Control*, 19(1):138–148, 2009.

[13] Yi Cao and Vinay Kariwala. Bidirectional branch and bound for controlled variable selection: Part i. principles and minimum singular value criterion. *Computers & Chemical Engineering*, 32(10):2306–2319, 2008.

[14] John N Hooker and Maria A Osorio. Mixed logical-linear programming. *Discrete Applied Mathematics*, 96:395–442, 1999.

[15] Lorenz T Biegler. *Nonlinear programming: concepts, algorithms, and applications to chemical processes*, volume 10. SIAM, 2010.

[16] H Paul Williams. *Model building in mathematical programming.* John Wiley & Sons, 2013.

[17] Jacques Richalet, A Rault, JL Testud, and J Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14(5):413–428, 1978.

[18] William Woelflin et al. The viscosity of crude-oil emulsions. In *Drilling and Production Practice.* American Petroleum Institute, 1942.

[19] Olav Kristiansen. Compact, inline separation technology – what and why?, 2014. URL `http://www.ipt.ntnu.no/~jsg/undervisning/prosessering/gjester/2014Kristiansen.pdf`. Accecced 14 June 2016.

[20] Laurens van Campen, Robert Frank Mudde, Jesse Slot, and Harry Hoeijmakers. A numerical and experimental survey of a liquid-liquid axial cyclone. *International journal of chemical reactor engineering*, 10(1), 2012.

[21] Maarten Dirkzwager. *A new acial cyclone design for fluid-fluid separation.* TU Delft, Delft University of Technology, 1996.

[22] Oljedirektoratet. Utslipp av produsert vann, 2012. URL `http://www.npd.no/publikasjoner/rapporter/havet-og-kysten/havet-og-kysten/2-utslipp-av-produsert-vann/`. Accecced 12. July 2016.

[23] Truls Larsson and Sigurd Skogestad. Plantwide control-a review and a new design procedure. *Modeling, Identification and control*, 21(4):209, 2000.

[24] Henrik Manum and Sigurd Skogestad. Self-optimizing control with active set changes. *Journal of Process Control*, 22(5):873–883, 2012.

[25] Robert H Perry and Don W Green. *Perry's chemical engineers' handbook*. McGraw-Hill Professional, 2008.

[26] Ed Klotz and Alexandra M Newman. Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science*, 18(1):18–32, 2013.

[27] Gurobi Optimization Inc. Gurobi optimizer reference manual, 2016. Available on www.gurobi.com , accessed 14. June 2016.

[28] The MathWorks Inc. Matlab documentation, 2016. Available on www.mathworks.com , accessed 14. June 2016.

[29] Stanislaw Rosloniec. *Fundamental numerical methods for electrical engineering*, volume 18. Springer Science & Business Media, 2008.

Appendix A

MATLAB-scripts

This appendix contains all the MATLAB code used in this work. For space considerations, only scripts used to produce results are listed here. That means, for instance scripts used for plotting is not included here.

A.1 Dummy Case

A.1.1 Dummy With One Measurement Set

```
close all;clear all;clc
% Dummy case, only one measurement set available

%% Define the matrix
Juu = [244 222;222 202];
Jud = [198 180];

Gy = [ 11 10 1 0 ; 10 9 0 1]';
Gd = [10 9 0 0]';

Wd =1;
Wn = diag([0.01 0.01 0.01 0.01]);
F = [-1 -1 9 -9 ]';
Juu12=Juu^(1/2);
nu=2;
nd=1;
ny=4;

Y = [F*Wd Wn];
%RandomCase
%% Building the restructured matrices
Ydel=Y;

for i=1:nu-1
    Ydel=blkdiag(Ydel,Y);
```

```
end
Fdel=Ydel*Ydel'

Gydel=[];
for i=1:nu
    Gydel=blkdiag(Gydel,Gy);
end
Gydel';

jdel=[];
for i=1:nu
    jdel=[jdel ; Juu12(:,i)];
end


%% Find the optimal number of measurements
M=120; %Big-M
% price for measurements
mprices=0.05*ones(1,ny);
%Define empty results matrices
cost = zeros(length(nu:ny),1);
loss = zeros(length(nu:ny),1);
Hres=zeros(nu,ny,length(nu:ny));
Meas=zeros(1,ny,length(nu:ny));
% measurements required
for nm=nu:ny % Run through all possible number of measurements

    Q=blkdiag(Fdel,zeros(ny,ny),0);

    A= [Gydel'            zeros(nu*nu,ny)              ...
        zeros(nu*nu,1)   ;
        zeros(1,nu*ny)   ones(1,ny)                   -1               ;
        -eye(nu*ny)      -M*repmat(eye(ny),nu,1)      zeros(nu*ny,1)   ;
        eye(nu*ny)       -M*repmat(eye(ny),nu,1)       zeros(nu*ny,1) ...
             ;
        zeros(1,nu*ny)   zeros(1,ny)                   1     ;
        ];

    c = [zeros(1,nu*ny) mprices                0                ];

    lb=[-inf(nu*ny,1);  zeros(ny,1)    ;0  ];
    ub=[inf(nu*ny,1);   ones(ny,1)     ;ny ];

    b= [jdel;
        0
        zeros(nu*ny,1);
        zeros(nu*ny,1)
        nm
        ];


        clear model;
        names = {'y1u1', 'y2u1', 'y3u1','y4u1','y1u2','y2u2','y3u2'...
                 'y4u2','sigma1','sigma2','sigma3','sigma4','nm'};
        model.varnames = names;
        model.Q = sparse(Q);
        model.A = sparse(A);
        model.obj = c;
        model.rhs = b;
        model.lb = lb;
        model.ub = ub;
        model.sense = ...
            [repmat('=',nu*nu+1,1);repmat('<',2*nu*ny,1);'='];
        model.modelsense='min'
        model.vtype = [repmat('C',nu*ny,1);repmat('B',ny,1);'I']
        gurobi_write(model, 'MeasSelectPrice0.lp');

        %params.ResultFile = 'test.mps'
        params.Presolve = -1;
        params.FeasibilityTol=1e-9;
        params.OptimalityTol=1e-9;
        %params.FeasibilityTol=1e-4
        results = gurobi(model, params)
        %results = gurobi(model)
        % for v=1:length(names)
        %     fprintf('%s %e\n', names{v}, results.x(v));
        % end
        fprintf('Obj: %e\n', results.objval);
        % Fill results matrices
        % Selected measurements
        Meas(:,:,nm-nu+1) = results.x(2*ny+1:3*ny)';
        % Selection matrix
        Hres(:,:,nm-nu+1) = [results.x(1:ny)';  results.x(ny+1:2*ny)'];
        H = (Hres(:,:,nm-nu+1));
        % Total cost
        cost(nm-nu+1)= (results.x'*Q*results.x)/2+model.obj*results.x;
        % Calculate loss with expression from Exact Local Method
        loss(nm-nu+1)= 0.5*norm(Juu12/(H*Gy)*H*Y,'fro')^2;

end
```

A.1.2 Dummy With Two Measurement Sets

```
close all;clear all;clc
% This is the dummy case for two measurement sets, without constraints
% Cost function:
% J = (z1 - z2)^2 + (z1- d)^2
Juu = [244 222;222 202];
Jud = [198 180];

Gy = [ 11 10 1 0 ; 10 9 0 1]';
Gd = [10 9 0 0]';

Wd =1;
Wn1 = diag([0.1 0.1 0.1 0.1]);
Wn2 = diag([0.01 0.01 0.01 0.01]);
F = [-1 -1 9 -9 ]';
Juu12=Juu^(1/2);
nu=2;
nd=1;
ny=4;
```

```

Y1 = [F*Wd Wn1];
Y2 = [F*Wd Wn2];
Y = [Y1;Y2];
% Building the restructured matrices
Ydel1=Y1;
Ydel2=Y2;
for i=1:nu-1
    Ydel1=blkdiag(Ydel1,Y1);
    Ydel2=blkdiag(Ydel2,Y2);
    size(Ydel1);
    size(Ydel2);
end
Fdel1=Ydel1*Ydel1';
Fdel2=Ydel2*Ydel2';
Fdel = blkdiag(Fdel1,Fdel2);


%%
Gydel=[];
for i=1:nu
    Gydel=blkdiag(Gydel,Gy);
end
Gydel = [Gydel;Gydel];
Gy1 = [Gy;Gy];
%%
jdel=[];
for i=1:nu
    jdel=[jdel ; Juu12(:,i)];
end
%% Solve for a measurements combination with all measurements
% Create gurobi model
M=100; % Big-M

mprices=[0.02*ones(1,ny) 0.2*ones(1,ny)]; % Prices of measurements

%Define empty result vectors
cost = zeros(length(nu:ny),1);
loss = zeros(length(nu:ny),1);
Hres=zeros(nu,2*ny,length(nu:ny));
Meas=zeros(1,2*ny,length(nu:ny));
Hadj = zeros(nu,2*ny,length(nu:ny));
% If statement so that one only chooses the correct set for ...
    corresponding
% to the measurement selection
subM = zeros(ny*nu*2,ny*2);
for i=1:ny
        subM(i,i+ny) = 1; % u1w1 <-> w2
        subM(i+ny,i+ny)=1; % u2w1 <-> w2
        subM(i+2*ny,i) = 1; % u1w2 <-> w1
        subM(i+3*ny,i) = 1; % u2w2 <-> w1
end

for nm = nu:ny
        %       H  sigma_1      sigma_2      nm
    Q=blkdiag(Fdel,zeros(ny,ny),zeros(ny,ny),0);
        %H                   sigma_1              sigma_2  %nm
    A= [Gydel'             zeros(nu*nu,ny)     zeros(nu*nu,ny) ...
                 zeros(nu*nu,1)   ;
        zeros(1,nu*ny*2)   ones(1,ny)           ones(1,ny)        -1 ...
                         ;
         -eye(nu*ny*2)      ...
             -M*([repmat(eye(ny),nu,1);zeros(2*ny,ny)]) ...
             -M*([zeros(2*ny,ny);repmat(eye(ny),nu,1)]) ...
             zeros(nu*ny*2,1);
        eye(nu*ny*2)      -M*([repmat(eye(ny),nu,1);zeros(2*ny,ny)]) ...
            -M*([zeros(2*ny,ny);repmat(eye(ny),nu,1)]) ...
            zeros(nu*ny*2,1);
        zeros(1,nu*ny*2)   zeros(1,ny)    zeros(1,ny)                 1 ...
              ;
        eye(nu*ny*2)       M*subM          zeros(ny*nu*2,1);
       -eye(nu*ny*2)       M*subM          zeros(ny*nu*2,1);
        ];

    c = [zeros(1,nu*ny*2) mprices                0                ];
    lb=[-inf(nu*ny*2,1);  zeros(ny,1); zeros(ny,1)   ;0  ];
    ub=[inf(nu*ny*2,1);   ones(ny,1) ; ones(ny,1)   ;ny ];
    b= [jdel;
        0
        zeros(2*nu*ny,1);
        zeros(2*nu*ny,1)
        nm
        M*ones(2*nu*ny,1)
        M*ones(2*nu*ny,1)
        ];


        clear model;
        model.Q = sparse(Q);
        model.A = sparse(A);
        model.obj = c;
        model.rhs = b;
        model.lb = lb;
        model.ub = ub;
        model.sense = [repmat('=',nu*nu+1,1);repmat('<',2*nu*ny,1);...
                       repmat('<',2*nu*ny,1);'=';repmat('<',2*nu*ny,1)...
                       ;repmat('<',2*nu*ny,1)]
        model.modelsense='min'
        model.vtype = [repmat('C',2*nu*ny,1);repmat('B',2*ny,1);'I']
        gurobi_write(model, 'DummyTwoSets.lp');

        params.Presolve = -1;
        params.FeasibilityTol=1e-9;
        params.OptimalityTol=1e-9;
        results = gurobi(model, params)
        % Selected measurements
        Meas(:,:,nm-nu+1) = [results.x(17:20)' results.x(21:24)'];
        Measdiag = diag(Meas(:,:,nm-nu+1));
        % prices of measurements
        price(nm-nu+1) = c*results.x;
        % Selection matrix
        Hres(:,:,nm-nu+1) = [results.x(1:4)'  results.x(9:12)'; ...
                             results.x(5:8)' results.x(13:16)'];
        Hadj(:,:,nm-nu+1) = Hres(:,:,nm-nu+1)*Measdiag;
        H = (Hres(:,:,nm-nu+1));
        %Total cost
        cost(nm-nu+1)= (results.x'*Q*results.x)/2+model.obj*results.x;
        % Unconstrained loss
        loss(nm-nu+1)= 0.5*norm(Juu12/(H*Gy1)*H*Y,'fro')^2;

end
```

A.1.3 Constrained Dummy

```
clear all
clc
% This script calculates the loss when one of the inputs are ...
    constrained
% The constrained input must be specified by user-promt.
% Objective
%min(J)  = (1/2)*u1^2 + (1/2)(u2-d2)^2
%s.t. u1-d1-1 >= 0
dnom = [0 0];

Q = [0.5 0;
     0 0.5];
A = [1 0]; % Define constraint
c = [0 -dnom(2)];
b = (1+dnom(1)); % Define constraint
clear model;
names = {'u1', 'u2'};
model.varnames = names;
model.Q = sparse(Q);
model.A = sparse(A);
model.obj = c;
model.rhs = b;
model.sense = ['>'];
model.objcon = dnom(2)^2;
gurobi_write(model, 'qp.lp');

results = gurobi(model); % Calculate nominal point
lambda = results.pi; % Calculate lagrange multiplier of constraint
%%
nu = 2;
ny = 4;% y = [u1 u2 d1 d2]
nd = 2;
nc = 1; % number of constraints
%fprintf('Which u is constrained, enter value less or equal to ...
    %d:',nu);
%consu = input('');
consu = 1; % Which u is constrained?
% Reduced J= 0.5*(u2^2-2*u2*d2+d2^2) + 1

Gy = [1 0;
      0 1;
      0.1 2;
      2 3];
Gyd = [0 0;
```

```
          0 0;
          1 0.2;
          0.5 3];
%Juu = [0 0; 0 1];
Juu = 1;
Juu12 = Juu^(1/2);
Jud = [0 -1]; %d1 d2

Gyd(:,1) = (-Gy(:,consu)+Gyd(:,1));
Gy(:,consu) = [];
F = (-Gy/Juu)*Jud+Gyd;

Wd =  [0.5 1]';
Wn1vec = [0.1 0.1 0.1 0.1];
Wn2vec = [1 1 1 1];
Wn1 = diag(Wn1vec);
Wn2 = diag(Wn2vec);
Y1 = [F*Wd Wn1];
Y2 = [F*Wd Wn2];
Y = [Y1;Y2];
Ydel1=Y1;
Ydel2=Y2;
Fdel1=Ydel1*Ydel1';
Fdel2=Ydel2*Ydel2';
Fdel = blkdiag(Fdel1,Fdel2);

Gydel=[];
for i=1:nu-nc %unconstrained u???
    Gydel=blkdiag(Gydel,Gy);
end
size(Gydel)
Gydel = [Gydel;Gydel];% Wn1 Wn2
Gy1 = [Gy;Gy];
%
jdel=[];
for i=1:nu-nc % Unconstrained
    jdel=[jdel ; Juu12];
end


M=500; % Big-M
%Define empty matrices
price = zeros(length(nu-nc:ny),1); % price of measurements
loss = zeros(length(nu-nc:ny),1); % Unconstrained loss
closs = zeros(length(nu-nc:ny),1); % Constrained loss
cost = zeros(length(nu-nc:ny),1); % price+loss+closs
ltot =zeros(length(nu-nc:ny),1); % Total loss
Hres=zeros((nu-nc),2*ny,length(nu-nc:ny));
Meas=zeros(1,2*ny,length(nu-nc:ny));
cmeas=zeros(1,2,length(nu-nc:ny)); % Selected constrained measurement
%% Unconstrained Two Set of Measurements
mprices = [0.2*ones(1,ny) 0.02*ones(1,ny)]; % Prices of each ...
    measurement
cprices = [0.2 0.02]; % Prices of constrained measurement
% Link up so that if one measurement set is selected, the corresponding
% value in Selection matrix for the same measurement in the other set
% Is set to zero
```

```
subM = zeros(ny*(nu-nc)*2,ny*2);
for i=1:ny
        subM(i,i+ny) = 1; % u2w1 <-> w2
        subM(i+ny,i)=1; % u2w2 <-> w1
       % subM(i+2*ny,i) = 1;
        %subM(i+3*ny,i) = 1;
end
% Constrained big-M
consumat = eye(ny*(nu-nc));
sigma1Mconstr = -M*([consumat*repmat(eye(ny),nu-1,1);zeros(ny,ny)]);
sigma2Mconstr = -M*([zeros(ny,ny);consumat*repmat(eye(ny),nu-1,1)]);

for nm = (nu-nc):ny

    Q=blkdiag(Fdel,zeros(ny,ny),zeros(ny,ny),0,zeros(4,4)); %Last ...
        two is for u1
        %H                   sigma_w1                sigma_w2 ...
                                 %nm
    A= [Gydel'              zeros((nu-nc)*(nu-nc),ny)      ...
        zeros((nu-nc)*(nu-nc),ny)       zeros((nu-nc)*(nu-nc),1) ...
        zeros(1,4) ;
        zeros(1,(nu-nc)*ny*2)  ones(1,ny)            ones(1,ny) ...
                  -1              zeros(1,4);
       -blkdiag(eye((nu-nc)*ny),eye((nu-nc)*ny))      sigma1Mconstr ...
                sigma2Mconstr  zeros((nu-nc)*ny*2,1)  ...
           zeros((nu-nc)*ny*2,4) ;
        blkdiag(eye((nu-nc)*ny),eye((nu-nc)*ny))      sigma1Mconstr ...
                 sigma2Mconstr  zeros((nu-nc)*ny*2,1)  ...
            zeros((nu-nc)*ny*2,4);
        zeros(1,(nu-nc)*ny*2)  zeros(1,ny)   zeros(1,ny) ...
                          1   zeros(1,4) ;
        eye((nu-nc)*ny*2)       M*subM              ...
            zeros(ny*(nu-nc)*2,1) zeros((nu-nc)*ny*2,4);
       -eye((nu-nc)*ny*2)       M*subM              ...
           zeros(ny*(nu-nc)*2,1) zeros((nu-nc)*ny*2,4);
       zeros(1,(nu-nc)*ny*2+2*ny+1) 0 0 1 1; % Force to select only ...
           one of these
       zeros(1,(nu-nc)*ny*2+2*ny+1) 1 1 0 0; % Force to select only ...
           one of these
       ];
                                                  % sigma_cons
    c = [zeros(1,(nu-nc)*ny*2) mprices       0     cprices(1) ...
        cprices(2) 0 0];
    c(end-1) = lambda*Wn1vec(1);
    c(end) = lambda*Wn2vec(1);
    lb=[-inf((nu-nc)*ny*2,1);  zeros(ny,1); zeros(ny,1)  ;0;0;0  ; ...
        -inf ;-inf];
    ub=[inf((nu-nc)*ny*2,1);   ones(ny,1) ; ones(ny,1)  ;ny;1;1 ...
        ;inf;inf];

    b= [jdel;
        0
        zeros(2*(nu-nc)*ny,1);
        zeros(2*(nu-nc)*ny,1)
        nm
        M*ones(2*(nu-nc)*ny,1)
        M*ones(2*(nu-nc)*ny,1)
        1
        1
      ];

    %try
        clear model;
        names = {...%'H_y1u1w1', 'H_y2u1w1', 'H_y3u1w1','H_y4u1w1', ...
                 'H_y1u2w1', 'H_y2u2w1', 'H_y3u2w1','H_y4u2w1',...
                 ...%'H_y1u1w2', 'H_y2u1w2', 'H_y3u1w2','H_y4u1w2',...
                 'H_y1u2w2', 'H_y2u2w2', 'H_y3u2w2','H_y4u2w2',...
                 '$y_1w_1$', '$y_2w_1$', '$y_3w_1$', '$y_4w_1$',...
                 '$y_1w_2$', '$y_2w_2$', '$y_3w_2$', '$y_4w_2$',...
                 'nm'...
                 '$y_5w_1$','$y_5w_2$','H_y5u1w1','H_y5u1w2'};
        model.varnames = names;


        model.sos(1).type =1;
        model.sos(1).index=[(4*ny+3) (4*ny+4)];
        model.sos(1).weight=[1 1];
        model.Q = sparse(Q);
        model.A = sparse(A);
        model.obj = c;
        model.rhs = b;
        model.lb = lb;
        model.ub = ub;
        model.sense = ...
            [repmat('=',(nu-nc)*(nu-nc)+1,1);repmat('<',2*(nu-nc)*ny,1);...
            repmat('<',2*(nu-nc)*ny,1);'=';repmat('<',2*(nu-nc)*ny,1);...
            repmat('<',2*(nu-nc)*ny,1);'=';'='];
        model.modelsense='min'
        model.vtype = ...
            [repmat('C',2*(nu-nc)*ny,1);repmat('B',2*ny,1);'I';...
                        repmat('B',2,1);repmat('B',2,1)];
        gurobi_write(model, 'qp2.lp');
        params.Presolve = -1
        params.FeasibilityTol=1e-9
        params.OptimalityTol=1e-9
        results = gurobi(model, params)
 % Fill in results
    Hres(:,:,nm-(nu-nc)+1) = [results.x(1:2*ny)'];
    H = Hres(:,:,nm-(nu-nc)+1);
    Meas(:,:,nm-(nu-nc)+1) = [results.x(2*ny+1:4*ny)'];
    cmeas(:,:,nm-(nu-nc)+1) = [results.x(end-3:end-2)'];
    loss(nm-(nu-nc)+1)= ...
        (results.x'*Q*results.x)/2;%+model.obj*results.x;
    loss1= 0.5*norm(Juu12/(H*Gy1)*H*Y,'fro')^2; %unconstrained loss
    % Make sure that loss is calculated correctly in optimizer vs.
    % Exact local method
    if loss(nm-(nu-nc)+1)-loss1 > 1e-3
        disp('Error in unconstrained loss calculation')
        break
    end
    % Constrained Loss, H(1) is constrained
    closs(nm-(nu-nc)+1) = c(end-1:end)*results.x(end-1:end);
    price(nm-(nu-nc)+1) = c(1:end-2)*results.x(1:end-2);
    ltot(nm-(nu-nc)+1) = loss(nm-(nu-nc)+1) + closs(nm-(nu-nc)+1);
    cost(nm-(nu-nc)+1) = loss(nm-(nu-nc)+1)+closs(nm-(nu-nc)+1)...
                        +price(nm-(nu-nc)+1);
end
```

A.2 Subsea Case

A.2.1 Subsea Model Scripts

All scripts in this subsection was created by Tyvold [1].

Gravity separator - Initial Guess

```
function [Vo_t,Vo_b]=gavity_sep2(FS,qin,Vo_in)
Lsep=7;
R=1.7;
Hw=0.75*2*R;
g=-9.81;

qt=FS*qin;
qb=qin-qt;

rhoo=881;rhow=1064;%kg/m^3
% Viscosity (eq:2.7)
% mum=(0.47*Vo_in^3-0.4*Vo_in^2+0.11*Vo_in+0.001); %Pa*s
mum=(0.6*Vo_in^3-0.506*Vo_in^2+0.137*Vo_in+0.001); %Pa*s
% Droplet size
rd=60*10^-6; %m^3

%Cross section area of weir (eq:3.27)
AHw=R^2/2*((2*acos((R-Hw)/R))-sin(2*acos((R-Hw)/R)));
% Horinzontal velocity of bottom plug flow (eq:3.26)
vh=qb/AHw;
%Vertical velocity of droplet (eq:3.28)
vv=2*g*(rhoo-rhow)*rd^2/(9*mum);
%Droplet Vertical distance travel (eq:3.29)
h=Lsep*vv/vh;
% If the "slowest" droplet of oil travel h>Hw -> No oil in waterphase
d=max(Hw-h,0); %if h>Hw -> Vo_b=0
% if Hw>h there will be oil in bottom:
%Cross section area of bottom section
%that still contains emulsion (Fig:3.11b, eq:3.30)
Ad=R^2/2*((2*acos((R-d)/R))-sin(2*acos((R-d)/R)));

Vw_b=((AHw-Ad)/AHw+Ad/AHw*(1-Vo_in));
Vo_b=1-Vw_b;

Vo_t=(Vo_in-Vo_b*(1-FS))/FS;
if Vo_t>1
    Vo_t=1;
    Vo_b=(Vo_in-Vo_t*FS)/(1-FS);
end

```

```
end
```

Dewaterer - Intitial Guess

```
function [Vo_LPO,Vo_HPO,qen]=DeWaterer(qin,Vo_in,FS,rin,p3)

Ro=p3(2);Ri=p3(3);

qi=FS*qin;       %Light phase out flow
qo=qin-qi;       %Heavy phase out flow

% Volume fractio of water in outlets before
% the re-entrainment is accounted for
% eq:3.21
Vw_HPO=(1-Vo_in)*(((1-FS)*Ri^2+FS*(Ri^2-rin^2))/((1-FS)*Ri^2));
% eq:3:22
Vw_LPO=((1-Vo_in)*qin-Vw_HPO*qo)/qi;
% Re-entrainment definitions
u_LPO=qi/(pi*Ri^2); % Area averaged velocity
u_HPO=qo/(pi*(Ro^2-Ri^2)); % Area averaged velocity
du=u_LPO-u_HPO;
k=2*10^-4;
qen=k*du;

%Re-Entrainment
if du>=0 % Velocity in oil > Velocity in water
    Vw_LPO=(Vw_LPO*(qi-qen)+Vw_HPO*qen)/qi;
    Vw_HPO=((1-Vo_in)*qin-Vw_LPO*qi)/qo;
else
    Vw_HPO=(Vw_HPO*(qi+qen)-Vw_LPO*qen)/qi;
end

% Restrict water volume fraction in HPO
% to range Vw_in->1
if Vw_HPO>1
    Vw_HPO=1;
elseif Vw_HPO<(1-Vo_in)
    Vw_HPO=(1-Vo_in);
end

Vw_LPO=((1-Vo_in)*qin-Vw_HPO*qo)/qi;

Vo_LPO=1-Vw_LPO;
Vo_HPO=1-Vw_HPO;
end
```

Deoiler - Initial guess

```
function [Vo_LPO,Vo_HPO,qen]=DeOiler(qin,Vo_in,FS,rin,p2)
Ro=p2(2);Ri=p2(3);

```

```matlab
qi=FS*qin; qo=qin-qi;

Vo_LPO=Vo_in*(FS*(Ro^2-Ri^2)+(rin^2-Ri^2)*(1-FS))...
    /(FS*(Ro^2-Ri^2));
Vo_HPO=(Vo_in*qin-Vo_LPO*qi)/qo;
u_HPO=qo/(pi*(Ro^2-Ri^2));u_LPO=qi/(pi*Ri^2);

du=(u_LPO-u_HPO);

k=2*10^-4;
%k=0;
qen=k*du;
if du>=0
    Vo_LPO=(Vo_LPO*(qi-qen)+Vo_HPO*qen)/qi;
else
    Vo_HPO=(Vo_HPO*(qo+qen)-Vo_LPO*qen)/qo;
    Vo_LPO=(Vo_in*qin-Vo_HPO*qo)/qi;
end

% k=5*10^-4;
% if du>=0
%     qen=k*du^2;
%     Vo_LPO=(Vo_LPO*(qi-qen)+Vo_HPO*qen)/qi;
% else
%     qen=-k*du^2;
%     Vo_HPO=(Vo_HPO*(qo+qen)-Vo_LPO*qen)/qo;
%     Vo_LPO=(Vo_in*qin-Vo_HPO*qo)/qi;
% end

if Vo_LPO>1
    Vo_LPO=1;
elseif Vo_LPO<Vo_in
    Vo_LPO=Vo_in;
end

Vo_HPO=(Vo_in*qin-Vo_LPO*qi)/qo;
end
```

Dewaterer - Used in Integrator

```matlab
function [vr]=swirl_sep2_o(t,x,in,element)
% Dewaterer
qin=in(1);
Ro=in(2);
Ri=in(3);
Vo_in=in(4);
ta=in(5);
rin=in(6);
FS=in(7);

Vw_in=(1-Vo_in);

rhoo=881; %kg/m^3
rhow=1064;
```

```

r=x(1);

Rc=0.25*Ro;
va=qin/(pi*Ro^2);
if element=='w' %weak
    k=3.5;
elseif element=='s' %strong
    k=5;
elseif element=='l' %large
    k=7;
end
vt0=k*va; %eq:3.5
%Experimental correlation between rd and vt
 %if vt0>4.45%28
    % rd=(-8*vt0+160)/2*10^-6;

 %else
   % rd=(-107*vt0+600)/2*10^-6;

 %end
rd= (-107*vt0+600)/2*10^-6+0.5*(1+tanh(10000*(vt0-4.45)))...
    *(((-8*vt0+160)/2*10^-6)-(-107*vt0+600)/2*10^-6);
%rd = 6e-6*vt0^2-0.0001*vt0+0.0007;
%Volume fraction of water on the inside
%of the droplet
Vw=Vw_in*(((1-FS)*Ri^2+FS*(Ri^2-rin^2))...
    /((1-FS)*Ri^2+FS*(Ri^2-r^2)));

%Emprical viscosity
mum=(0.203*Vw^3+0.237*Vw^2-0.014*Vw+0.0088);

%Smooth centrifugal acceleration
f2=(vt0*exp(-0.04*va*t/(2*Ro)))^2/r;
f1=(vt0*exp(-0.04*va*t/(2*Ro))/Rc)^2*r;
f=f2-f1; beta=30;
ac=f2-0.5*((f^2+beta^2)^.5+f); % eq:3.18
%vt=sqrt(ac*r);

%Radial velocity of droplet
vr=2/9*(rhow-rhoo)*rd^2/mum*ac; % vr(r,z)eq:3.7
end
```

Deoiler - Used in Integrator

```
function DXDT=swirl_sep2(t,x,in,element)
%deoiler
qin=in(1);
Ro=in(2);
Ri=in(3);
Vo_in=in(4);
ta=in(5);
rin=in(6);
FS=in(7);
```

```

rhoo=881;%872; %kg/m^3
rhow=1064;

r=x(1);

Rc=0.25*Ro;
va=qin/(pi*Ro^2);
if element=='w' %weak
    k=3.5;
elseif element=='s' %strong
    k=5;
elseif element=='l' %large
    k=7;
end

vt=k*va;

%Experimental correlation between rd and vt
% if vt>4.45%28
%     rd=(-8*vt+160)/2*10^-6;
% else
%     rd=(-107*vt+600)/2*10^-6;
% end
rd= (-107*vt+600)/2*10^-6+0.5*(1+tanh(10000*(vt-4.45)))...
    *(((-8*vt+160)/2*10^-6)-(-107*vt+600)/2*10^-6);
%   if vt>4%28
%      rd=10^-6*(80-4*vt);
%
%  else
%     rd=10^-6*(300-59*vt);
%
%  end
%   if vt>0.2%28
%      rd=0.0014*vt^(-1.244);
%
%  else
%     rd=-0.0645*vt+ 0.0233;
%
%  end

va=qin*(1-FS)/(pi*(Ro^2-Ri^2));
%rd=(-8*vt+160)/2*10^-6;

% if qin>50/3600
%     rd=30*10^-6;
% else
%     rd=50*10^-6;
% end

%diameter=2*rd*10^6
Vo_c=Vo_in*(FS*(Ro^2-Ri^2)*(Ro^2-r^2)+(rin^2-Ri^2)*(Ro^2-r^2)*(1-FS))...
    /((r^2-Ri^2)*(Ro^2-r^2)*(1-FS)+FS*(Ro^2-Ri^2)*(Ro^2-r^2));

mum=(0.6*Vo_c^3-0.506*Vo_c^2+0.137*Vo_c+0.001);
%mum=(0.47*Vo_c^3-0.4*Vo_c^2+0.11*Vo_c+0.001);

```

```
vr=((2/9*(rhoo-rhow)*rd^2/(mum)...
    *(vt*exp(-0.04*va*t/(2*Ro)))^2/r)*heaviside(r-Rc)... %r>Rc
    +(2/9*(rhoo-rhow)*rd^2/mum...
    *r*(vt*exp(-0.04*va*t/(2*Ro))/Rc)^2)*heaviside(Rc-r))...; %r<Rc
    *heaviside(r-0.9*Ri); %To make sure r>0

drdt=vr;

DXDT=[drdt];
end
```

A.2.2 Optimization Scripts

All scripts in this subsection was created by Tyvold [1].

Steady State optimizer script

Main script. Everything run from this script.

```

clearvars -except y
clear all
clc
tic
%xnom = load('xvar.mat');
%xnom = load('xnom2.mat');
%load('xnom20-0.35.mat');
load('xnomadj2.mat')
x1 = x;
qin=[18:0.25:23]/3600;
%qin =[30:-0.25:15]/3600;
%qin= (20/3600);
%Vo_in=0.40;%
Vo_in=[0.35:0.0125:0.6];

%Vo_in = [0.3:0.0125:0.7];
Lsw=1.7;Ro=0.05;Ri=0.025;p2=[Lsw;Ro;Ri];
Lsw_DW=1.7;Ro_DW=0.05;Ri_DW=0.043;p3=[Lsw_DW;Ro_DW;Ri_DW];

element='l';
dotvec = zeros(length(qin),length(Vo_in));
x=zeros(length(qin),length(Vo_in),21);
comp=zeros(length(qin),length(Vo_in),6);
exitflag=zeros(length(qin),length(Vo_in));
Cost=zeros(length(qin),length(Vo_in));
options =...
    optimset('Algorithm','interior-point','Display','Off','MaxIter'...
    ,1e5,'TolX',1e-13,'MaxFunEvals',1e6,'TolCon',1e-5);
A=[];b=[];Aeq=[];beq=[];
% xopt = zeros(21,2);
% xopt(:,1) = [0.33;0.91;Ri_DW;0.15;Ri;6.7/3600;0.94;13.3/3600;0.13;...
%              6.1/3600;0.98;0.6/3600;0.5;13.9/3600;0.15;2/3600;0.83;...
%              11.9/3600;0.03;8.1/3600;0.94];
% xopt(:,2) = ...
    [0.4;0.86;Ri_DW;0.27;Ri;8/3600;0.91;12/3600;0.22;7/3600;...
%             ...
    0.97;1/3600;0.48;13/3600;0.24;3.5/3600;0.82;9.5/3600;0.03;...
%             10.5/3600;0.92];
% grad = zeros(length(qin),length(Vo_in),21);
%Optimization
for j=1:length(Vo_in)
    for i=1:length(qin)
        p=[qin(i);Vo_in(j)];
        % Bounds for inputs
        %The bounadries on the FSs are just to help the solver
        %and should not be active!!
        lb=[0.2;0.5;0;0.01;p2(3);zeros(16,1)];
        %lb=[0;0.5;0;0;p2(3);zeros(16,1)];
        ub=zeros(21,1);
        %ub(1:5,1)=[0.9;0.95;p3(3);0.6;p2(2)];
        ub(1:5,1)=[0.999;0.999;p3(3);0.6;p2(2)];
        for k =6:length(ub)
            a=factor(k);
            if a(1)==2; %flow
                ub(k,1)=qin(i);
            else %volume fraction
                ub(k,1)=1;
            end
        end
        % Turn off initial guess and replace with old nominal values
%         if j==1 && i==1
%            x0 = squeeze(xnom);
%
%        %Initial guess
%        if j==1 && i==1
%             %  FS_g,FS_DW,rin_DW,FS_DO,rin_DO
%             x01=[Vo_in(i);0.8;0.4*Ri_DW;0.3;0.65*Ro];
%             x0=InitialGuess(x01,p,p2,p3);
            toc
%         elseif i==1 && j~=1
%           x0 = squeeze(xnom);
%
           %x0=x02;
%         else
%             x01=[Vo_in(i);x0(2);x0(3);x0(4);x0(5)];
%             x0=InitialGuess(x01,p,p2,p3);
  %       end
        % Forces guesses to be close to "true" values
%         if Vo_in(j) < 0.45
%             for k = 1:length(x0)
%                 if (x0(k)/xopt(k,1)) > 1.1 || (x0(k)/xopt(k,1)) < 0.9
%                     x0(k) = xopt(k,1);
%                     teller = teller +1;
%                 end
%             end
%
%         elseif Vo_in(j) >= 0.45
%             for k = 1:length(x0)
%                 if (x0(k)/xopt(k,2)) > 1.1 || (x0(k)/xopt(k,2)) < 0.9
%                     x0(k) = xopt(k,2);
%                   end
%               end
%           end
        x0 = squeeze(x1(i,j,:));
        [x(i,j,:),~,exitflag(i,j),output(i,j),lambda(i,j)] = ...
            fmincon(...
            @(x)CostFunc2(x,p,p2,p3),x0,A,b,Aeq,beq,lb,ub,...
            @(x)constraints2(x,p,p2,p3,element,Vo_in(j)),options);
        [Cost(i,j)]=CostFunc2(x(i,j,:),p,p2,p3);
        if exitflag(i,j)==1 ||exitflag(i,j)==2
            x0=squeeze(x(i,j,:));
        elseif exitflag(i,j) == -2 && j>2
                xold = squeeze(x(i,j-2,:));
                x(i,j-1,:) = globaloptim(xold,Vo_in(j-1),qin(i));
                x(i,j,:) = globaloptim(x0,Vo_in(j),qin(i));
                x0=squeeze(x(i,j,:));
         elseif exitflag(i,j) == -2 && j<=2
                x(i,j,:) = globaloptim(x0,Vo_in(j),qin(i));
                x0=squeeze(x(i,j,:));
        end
        if j==1
            x02=x0;
        end
        %Define a plot tolerance 1%
        plottol = 0.01*0.99;
        if x0(19)<plottol
            dotvec(i,j) = 1;
        else
            dotvec(i,j) = 0;
        end
    end
end
Cost
y=x(:,:,15);
exitflag
toc
%%
for i=1:length(qin)
for j=1:length(Vo_in)
lambdaplot(i,j)=  lambda(i,j).ineqnonlin(3)
end
end
```

Initial Guess

This is ran if no initial solution is available.

```
function [y]=InitialGuess(x,p,p2,p3)
%InitialGuess(x,p,p2,p3) Finds the values of the state
%vector that are consistent with the 5 first elements
FS_g=x(1);FS_DW=x(2);rin_DW=x(3);FS=x(4);rin=x(5);
qin=p(1);Vo_in=p(2);

%Gravity
[Vo_t,Vo_b]=gavity_sep2(FS_g,qin,Vo_in);
```

```
qb=(1-FS_g)*qin;
qt=qin-qb;

%DeWaterer
qi_DW=FS_DW*qt; qo_DW=(1-FS_DW)*qt;
[Vo_LPO_DW,Vo_HPO_DW]=DeWaterer(qt,Vo_t,FS_DW,rin_DW,p3);

%DeOiler
q3=qb+qo_DW;
Vo_3=(Vo_b*qb+Vo_HPO_DW*qo_DW)/q3;
[Vo_LPO_DO,Vo_HPO_DO]=DeOiler(q3,Vo_3,FS,rin,p2);
qi=FS*q3; qo=q3-qi;
%Oil Product
qoil=qi_DW+qi;
Vo_oil=(Vo_LPO_DW*qi_DW+Vo_LPO_DO*qi)/qoil;

%Output
x2=[qt;Vo_t;qb;Vo_b;qi_DW;Vo_LPO_DW;qo_DW;Vo_HPO_DW;...
    q3;Vo_3; qi;Vo_LPO_DO; qo; Vo_HPO_DO; qoil; Vo_oil]; % age; ...
        Added ; between q3 and Vo_3, otherwise: error
y=[x;x2];

end
```

Cost Function

```
function [c]=CostFunc2(x,p,p2,p3)
qin=p(1);Vo_in=p(2);

%OilInWater=x(18)*x(19);
OilInWater=x(19)*x(18);
WaterInOil=(1-x(21))*x(20);
%WaterInOil=(1-x(17))*x(16);
%c=10^4*(WaterInOil+OilInWater);
%c=(1-x(21))*10;
priceoilinoil = 40;
pricewaterinoil = 20;
priceoilinwater= 10;
c=(-x(21)*x(20))*6.289*priceoilinoil ...
    + x(20)*(1-x(21))*6.289*pricewaterinoil ...
        +0.00001*(x(16)*x(17))^2;...
   % + x(19)*x(18)*3600*6.289*priceoilinwater/1000 ;%+0.0001*x(4)^2;
end
```

Constraint function

```
function [c,ceq]=constraints2(x,p,p2,p3,element,Vo_in)
FS_g=x(1);FS_DW=x(2);rin_DW=x(3);FS=x(4);rin=x(5);
qin=p(1);Vo_in=p(2);
Lsw=p2(1);Ro=p2(2);Ri=p2(3);
Lsw_DW=p3(1);Ro_DW=p3(2);Ri_DW=p3(3);
```

```

%Gravity
[Vo_t,Vo_b]=gavity_sep2(FS_g,qin,Vo_in);
qb=(1-FS_g)*qin;
qt=qin-qb;
%DeWaterer
qi_DW=FS_DW*qt; %Light phase out flow
qo_DW=qt-qi_DW;    %Heavy phase out flow
ta_DW=pi*Ri_DW^2*Lsw_DW/qi_DW; % eq:3.20

in_DW=[qt,Ro_DW,Ri_DW,Vo_t,ta_DW,rin_DW,FS_DW];
h_DW=ta_DW/10;
[T,X_DW]=RK2(@swirl_sep2_o,[0 ta_DW],rin_DW,h_DW,in_DW,element);
%options = odeset;
%[T,X_DW]= ode45(@swirl_sep2_o,[0 ta_DW],rin_DW,options,in_DW,element);

[Vo_LPO_DW,Vo_HPO_DW]=DeWaterer(qt,Vo_t,FS_DW,rin_DW,p3);

rout_DW=X_DW(end,1);

%DeOiler
q3=qb+qo_DW;
Vo_3=(Vo_b*qb+Vo_HPO_DW*qo_DW)/q3;
qi=FS*q3; %Light phase out flow
qo=q3-qi;    %Heavy phase out flow
ta=pi*(Ro^2-Ri^2)*Lsw/qo;

in=[q3,Ro,Ri,Vo_3,ta,rin,FS];
h=ta/10;
[T,X]=RK2(@swirl_sep2,[0 ta],rin,h,in,element);
%options = odeset;
%[T,X]=ode45(@swirl_sep2,[0 ta],rin,options,in,element);
rout=X(end,1);

[Vo_LPO_DO,Vo_HPO_DO]=DeOiler(q3,Vo_3,FS,rin,p2);

%Oil Product
qoil=qi_DW+qi;
Vo_oil=(Vo_LPO_DW*qi_DW+Vo_LPO_DO*qi)/qoil;

%constraints

%In case of fixed flow splits:
FSzero=[FS_g-0.33;FS_DW-0.91;FS-0.15];

ExplEq=[qt;Vo_t;qb;Vo_b;qi_DW;Vo_LPO_DW;...
    qo_DW;Vo_HPO_DW;q3;Vo_3;...
    qi;Vo_LPO_DO; qo; Vo_HPO_DO; qoil; Vo_oil]-x(6:21);
ceq=[(Ri_DW-rout_DW)/Ri_DW;(rout-Ri)/Ri;ExplEq];
c=[0.7-Vo_t;Vo_3-0.6;x(19)-0.01;0.0001-x(19)];%

%c=[0.7-Vo_t;Vo_3-0.6];%
end
```

Integrator

```
function [t,y]=RK2(ODEfile,tspan,yi,h,varargin)
%2nd-order Runge-Kutta
t=tspan(1):h:tspan(2);
if t(end)~=tspan(2)
    t(end+1)=tspan(2);
end
d=diff(t);
yi=(yi(:).')';

y(:,1)=yi;
for i=1:length(t)-1
    k1=d(i)*feval(ODEfile,t(i),y(:,i),varargin{:});
    k2=d(i)*feval(ODEfile,t(i+1),y(:,i)+k1,varargin{:});
    y(:,i+1)=y(:,i)+(k1+k2)/2;
end
y=y';
t=t';
end
```

A.2.3 Self optimizing Control Scripts

This section includes the files for self-optimizing control for the subsea case. Note that there are two scripts called `optimizer` and `optimizeru` inside this main script, these are the same script as the steady state optimizer script given in Appendix A.2.2.

```
% Script for selfoptimizing control
% Calculates measurements through exact local method
clear all
clc
qin   = 23/3600; % Define a nominal point, inflow
Vo_in = 0.6;     % Define a nominal point, oil fraction
% Nominal point
%[xnom, cost2] = optimizer(Vo_in,qin);
load('xq23vo06.mat');% Using results from simulations as start points
%lambda = load('lambda1.mat');% Using lambda from simulation
xnom = x1;
lambda = 0.8817;
nu = 3; ny =16; nd = 2; nc=1;% Number of inputs, measurements, ...
    disturbances
yc = 14; % Which y is linked to constrained u3?
nyc = ny-nc; %Total available measurements
nuc = nu-nc; %Total available inputs
%% Add 1% Disturbance to calculate F
di =1.01;
d1 = qin*di;
d2 = Vo_in*di;
deltad1 = d1-qin;
deltad2 = d2-Vo_in;
for i=1:2
    if i == 1
        [xd1, costd1] = optimizer(Vo_in,d1,xnom);
    else
```

```
        [xd2, costd2]= optimizer(d2,qin,xnom);
    end
end
xd = squeeze([xd1 xd2])';
d = [deltad1 deltad2];
%% Calculates sensitivity matrix
F = zeros(ny,nd);
for j = 6:length(xnom)
    for i = 1:length(d)
        F(j-5,i) = (xd(j,i)-xnom(j))/d(i);
    end
end
F(yc,:) = [];
%% Calculate Juu (Hessian)

costu = zeros(nu,nu);
%xud = zeros(nu,nu,nu);
udiff = [0.999 1 1.001];
unom = [xnom(1) xnom(2) xnom(4)];
x0 = squeeze(xnom);
teller = 0;
xud = zeros(21,9);
% Using small pertubations (+- 0.001) in three dimensions
for i=1:length(unom) %u1
    for j =1:length(unom) %u2
        %for k = 1:length(unom) u3 already constrained
            %if i==2 || j==2 %|| k==2
            ceq= ['ceq=[(Ri_DW-rout_DW)/Ri_DW;(rout-Ri)/Ri;ExplEq; ...
                x(1)-' ...
                ,num2str(unom(1)*udiff(i))...
                ';x(2)-', num2str(unom(2)*udiff(j)),'];'];
            filehandle(ceq); % Rewrite the constraint file
            %if teller == 9
            %    x0 = squeeze(xud(:,8));
            %end
            x0(1) = ...
                x0(1)*udiff(i);x0(2)=x0(2)*udiff(j);%x0(4)=x0(4)*udiff(k);
            teller = teller +1 ;
            %[xud(i,j,k), costu(i,j,k)] = optimizeru(Vo_in,qin,x0);
            %[xud(:,teller), costu(i,j,k)] = ...
                optimizeru(Vo_in,qin,x0) <-this one;
            [xud(:,teller), costu(i,j)] = optimizeru(Vo_in,qin,x0);
            %else
            %    teller = teller + 1;
            %end
        %end
    end
    ceq = [''];
    filehandle(ceq); % Reset the constraint file
    x0 = squeeze(xnom);
end
% Calcualte the finite differences using central step method
Ju1u1 = (costu(3,2)-2*costu(2,2)+costu(1,2))/(0.001^2);
Ju1u2 = (costu(3,3)-costu(3,1)-costu(1,3)+costu(1,1))/(4*0.001*0.001);
Ju2u2 = (costu(2,3)-2*costu(2,2)+costu(2,1))/(0.001^2);
Ju2u1 = Ju1u2;
% Build Hessian matrix for the cost function
```

```
Juu = [Ju1u1 Ju1u2;Ju2u1 Ju2u2];
%% Calculate Gain Matrix (Gy) = Jacobian of measurements/inputs

deltau = 0.001;
% Calculating differentials using finite differences
Gy1 = (xud(6:21,7)-xud(6:21,1))/(2*deltau);  %deltay/deltau1
Gy2 = (xud(6:21,6)-xud(6:21,4))/(2*deltau); %deltay/deltau2
Gy = [Gy1 Gy2];
% Remove measurement correlated with u3 => x19 = y14
Gy(yc,:) = [];
%% Scaling Matrices, Y, M and H
noisevec1 = zeros(ny,1);%zeros(ny,1);
noisevec2 = zeros(ny,1);%zeros(ny,1);
% Insert measurement noise
% High accuracy: set w_1
for i=1:ny
    if mod(i,2) == 1
        noisevec1(i) = 0.01/3600; % Odd for q  (UNIT: /3600 or NOT?)
    else
        noisevec1(i) = 0.001; % even for oilcut
    end
end
% % Low accuracy: set w_2
for i=1:ny
    if mod(i,2) == 1
        noisevec2(i) = 0.4/3600 ;%0.1/3600; % Odd for q (UNIT: ...
            /3600 or NOT?)
    else
        noisevec2(i) = 0.02;%0.01; % Even for oil-cut
    end
end
% Keep u3 => y14
Wn14(1) = noisevec1(yc);
Wn14(2) = noisevec2(yc);
% Remove u3 => y14
noisevec2(yc) = [];
noisevec1(yc) = [];

% Insert disturbance noise here!!
Wd = diag([5/3600 0.2]); %[q alfa]
Wn1 = diag(noisevec1);
Wn2 = diag(noisevec2);
% Extend length of F and Gy
Y1 = [F*Wd Wn1];
Y2 = [F*Wd Wn2];

% H = Gy'*inv(Y*Y');
% M = ((Juu)^(1/2))*inv(H*Gy)*H*Y;
% Lwc = (0.5*max(svd(M)))^2;
% Lavg = 0.5*norm((Juu^(1/2)/(H*Gy))*H*Y,'fro')^2;

% Test for convexity (positive semi definite)
posdef = find(eig(Juu) < 0);
if isempty(posdef) == false
    disp('Hessian is not positive definite')
else
    disp('Hessian is positive definite')
```

```
end


%% Mesurement selection
Ydel1=Y1;
Ydel2=Y2;
for i=1:nu-1-nc
    Ydel1=blkdiag(Ydel1,Y1);
    Ydel2=blkdiag(Ydel2,Y2);
    size(Ydel1);
    size(Ydel2);
end
Fdel1=Ydel1*Ydel1';
Fdel2=Ydel2*Ydel2';
Fdel = blkdiag(Fdel1,Fdel2);

Juu12 = Juu^(1/2);

Gydel=[];
for i=1:nu-nc
    Gydel=blkdiag(Gydel,Gy);
end
Gydel';
Gydel = [Gydel;Gydel];
Gyres = [Gy;Gy]; % For loss calculation only
Yres = [Y1;Y2]; % For loss calculation only
jdel=[];
for i=1:nu-nc
    jdel=[jdel ; Juu12(:,i)];
end
Y = [Y1;Y2]; % For loss Calculation only
Gy1 = [Gy;Gy]; % For loss calculation only
% Create GUROBI model
%% Find the optimal number of measurements
M = 1e3; %Big-M
% Prices of measurements
Loss = zeros(1,ny-nc);
mprices = zeros(1,ny);
mprices2= zeros(1,ny);
for i = 1:ny
    if mod(i,2) == 1
        mprices(i) = 0.001; % Flow w1
        mprices2(i) = 0.0005; %Flow w2
    else
        mprices(i) = 0.002; %Oil fraction w1
        mprices2(i) = 0.001;% Oil fraction, w2
    end
end
mprices(yc) = [];%removed constrained measurement
mprices2(yc) = [];
% Prices for the constraint
cprices = [mprices(2) mprices2(2)]; % w1 w2

%Define empty matrices
Hres=zeros(nuc,2*nyc,length(nuc:nyc));
Meas=zeros(2,nyc,length(nuc:nyc));
cmeas=zeros(1,2,length(nu-nc:ny));
```

```
Hcons = zeros(2,2,length(nuc:nyc));
Hadj = zeros(nuc,2*nyc,length(nuc:nyc));
price = zeros(length(nuc:nyc),1); % price of measurements
loss = zeros(length(nuc:nyc),1); % Unconstrained loss
closs = zeros(length(nuc:nyc),1); % Constrained loss
cost = zeros(length(nuc:nyc),1); % price+loss+closs
ltot =zeros(length(nuc:nyc),1); % Total loss

subM = zeros(nyc*nuc*2,nyc*2);
% Selection criteria between each measurement set.
for i=1:nyc
        subM(i,i+nyc) = 1;
        subM(i+nyc,i+nyc)=1;
        subM(i+2*nyc,i) = 1;
        subM(i+3*nyc,i) = 1;
end
consumat = eye(nyc*nuc); %diag(helpvec)
sigma1Mconstr = -M*([repmat(eye(nyc),nu-1,1);zeros(nyc*nuc,nyc)]);
sigma2Mconstr = -M*([zeros(nyc*nuc,nyc);repmat(eye(nyc),nu-1,1)]);
tic
% MIQP Model for Gurobi starts here
for nm = nuc:nyc
    Q=blkdiag(Fdel,zeros(nyc,nyc),zeros(nyc,nyc),0,zeros(4,4));
        %H            sigma_w1             sigma_w2      nm  ...    sigma_c H_c

    A= [Gydel'            zeros(nuc*nuc,nyc)    zeros(nuc*nuc,nyc) ...    zeros(nuc*nuc,1) zeros(4,4) ;
        zeros(1,nuc*nyc*2)  ones(1,nyc)           ones(1,nyc)        ...    -1               zeros(1,4);
       -blkdiag(eye(nuc*nyc),eye(nuc*nyc))      sigma1Mconstr      ...    sigma2Mconstr  zeros(nuc*nyc*2,1) zeros(nuc*nyc*2,4) ;
        blkdiag(eye(nuc*nyc),eye(nuc*nyc))      sigma1Mconstr      ...    sigma2Mconstr  zeros(nuc*nyc*2,1) zeros(nuc*nyc*2,4);
        zeros(1,nuc*nyc*2)  zeros(1,nyc)   zeros(1,nyc) ...    1   zeros(1,4) ;
        eye(nuc*nyc*2)      M*subM          zeros(nyc*nuc*2,1) ...    zeros(nuc*nyc*2,4);
       -eye(nuc*nyc*2)      M*subM          zeros(nyc*nuc*2,1) ...    zeros(nuc*nyc*2,4);
       zeros(1,nuc*nyc*2+2*nyc+1) 1 1 0 0;
       zeros(1,nuc*nyc*2+2*nyc+1) 1 0 0 1;
       zeros(1,nuc*nyc*2+2*nyc+1) 0 1 1 0;
        zeros(1,nuc*nyc*2)   mprices         mprices2          0 ...    cprices(1) cprices(2)  0 0;
       ];

    c = [zeros(1,nuc*nyc*2) mprices mprices2       0     cprices(1) ...    cprices(2) 0 0];
    c(end-1) = lambda*Wn14(1); % Back-off loss
    c(end) = lambda*Wn14(2); % Back-off loss
    lb=[-inf(nuc*nyc*2,1);  zeros(nyc,1); zeros(nyc,1)  ;0;0;0  ; ...    -inf ;-inf];
    ub=[inf(nuc*nyc*2,1);   ones(nyc,1) ; ones(nyc,1)  ;nyc;1;1 ...    ;inf;inf];

    b= [jdel;
```

```
            0
            zeros(2*nuc*nyc,1);
            zeros(2*nuc*nyc,1)
            nm
            M*ones(2*nuc*nyc,1)
            M*ones(2*nuc*nyc,1)
            1
            1
            1
            M;%0.0048 % Insert budget constraint here!
          ];

        %try
            clear model;

            model.varnames = namescript(nuc,ny,yc);
            for n = 1:nyc
                 model.sos(n).type =1;
                 model.sos(n).index=[(4*nyc+n) (5*nyc+n)];
                 model.sos(n).weight=[1 1];
             end

    %
            model.Q = 0.5*sparse(Q);
            model.A = sparse(A);
            model.obj = c;
            model.rhs = b;
            model.lb = lb;
            model.ub = ub;
            model.sense = ...
                [repmat('=',nuc*nuc+1,1);repmat('<',2*nuc*nyc,1);...
                repmat('<',2*nuc*nyc,1);'=';repmat('<',2*nuc*nyc,1);...
                repmat('<',2*nuc*nyc,1);'=';'=';'=';'<'];
            model.modelsense='min';
            model.vtype = ...
                [repmat('C',2*nuc*nyc,1);repmat('B',2*nyc,1);'I';...
                            repmat('B',2,1);repmat('B',2,1)];
            gurobi_write(model, 'qp2.lp');
            % Parameters for better solving of the MIQP
            params.Presolve = 2;
            params.MIPFocus = 3;
            params.NumericFocus = 3;
            params.FeasibilityTol=1e-9;
            params.OptimalityTol=1e-9;
            params.IntFeasTol = 1e-9;
            results = gurobi(model, params)
            Hres(:,:,nm-nuc+1) = [results.x(1:nyc)'   ...
                results.x(2*nyc+1:3*nyc)';...
                              results.x(nyc+1:2*nyc)' ...
                                  results.x(3*nyc+1:4*nyc)';...
                              ];
            H = Hres(:,:,nm-nuc+1);
            %Selection matrix
            Meas(:,:,nm-nuc+1) = [results.x(4*nyc+1:5*nyc)';...
                                  results.x(5*nyc+1:6*nyc)'];

            cmeas(:,:,nm-(nuc)+1) = [results.x(end-3:end-2)'];
            Hcons(:,:,nm-nuc+1)= ...
                [results.x(end-3:end-2)';results.x(end-1:end)'];
            %cost(nm-nuc+1)= results.x'*Q*results.x+model.obj*results.x;
            loss1= 0.5*norm(Juu12/(H*Gyres)*H*Yres,'fro')^2; % ...
                Unconstrained loss
            loss(nm-(nu-nc)+1)= ...
                (results.x'*Q*results.x)/2;%+model.obj*results.x;
            % Test for error in the loss calculation: Optimizer vs.
            % Exact local method
            if abs(loss(nm-(nu-nc)+1)-loss1) > 1e-3 && nm~=2
                disp('Error in unconstrained loss calculation')
                break
            end


            closs(nm-(nuc)+1) = c(end-1:end)*results.x(end-1:end);
            price(nm-(nuc)+1) = c(1:end-2)*results.x(1:end-2);
            ltot(nm-(nuc)+1) = loss(nm-(nuc)+1) + closs(nm-(nuc)+1);
            cost(nm-(nuc)+1) = loss(nm-(nuc)+1)+closs(nm-(nuc)+1)...
                            +price(nm-(nuc)+1);
    end
    toc
```

Re-write the constraint file

```
function filehandle(ceq)
clear fin
fin = fopen('constraintsu.m','r+');
replaceline = 54;
newline =  ceq;
%endfile = 'c=[0.7-Vo_t;Vo_3-0.6;x(19)-0.03;0.001-x(19)];%';
endfile = 'c=[0.7-Vo_t;Vo_3-0.6];%';% <- use this if unconstrained
%endfile = 'c=[0.7-Vo_t;Vo_3-0.6;x(19)-0.01];'; %Use this if ...
    constrained
endfile2 = 'end';
for k=1:(replaceline-1)
    fgetl(fin);
end
fseek(fin,0,'cof');
fprintf(fin,'%s\n',newline);
fprintf(fin,'%s\n',endfile);
fprintf(fin,'%s\n',endfile2);
fclose(fin);
clear fin constraintsu endfile endfile2
%pause(2);
```

Appendix B

GUROBI Optimization Problem Model Files

This appendix shows the optimization problems in a human readable format provided from the Gurobi solver. The file includes the cost functions with all bounds and constraints.

B.1 Unconstrained Dummy Problem

```
Minimize
  0.02 C16 + 0.02 C17 + 0.02 C18 + 0.02 C19 + 0.2 C20 + 0.2 C21 + ...
      0.2 C22
   + 0.2 C23 + [ 2.02 C0 ^2 + 4 C0 * C1 - 36 C0 * C2 + 36 C0 * C3
   + 2.02 C1 ^2 - 36 C1 * C2 + 36 C1 * C3 + 162.02 C2 ^2 - 324 C2 * C3
   + 162.02 C3 ^2 + 2.02 C4 ^2 + 4 C4 * C5 - 36 C4 * C6 + 36 C4 * C7
   + 2.02 C5 ^2 - 36 C5 * C6 + 36 C5 * C7 + 162.02 C6 ^2 - 324 C6 * C7
   + 162.02 C7 ^2 + 2.0002 C8 ^2 + 4 C8 * C9 - 36 C8 * C10 + 36 C8 ...
      * C11
   + 2.0002 C9 ^2 - 36 C9 * C10 + 36 C9 * C11 + 162.0002 C10 ^2
   - 324 C10 * C11 + 162.0002 C11 ^2 + 2.0002 C12 ^2 + 4 C12 * C13
   - 36 C12 * C14 + 36 C12 * C15 + 2.0002 C13 ^2 - 36 C13 * C14
   + 36 C13 * C15 + 162.0002 C14 ^2 - 324 C14 * C15 + 162.0002 C15 ^2
   ] / 2
Subject To
 R0: 11 C0 + 10 C1 + C2 + 11 C8 + 10 C9 + C10 = 11.5965512114594
 R1: 10 C0 + 9 C1 + C3 + 10 C8 + 9 C9 + C11 = 10.46518036156088
 R2: 11 C4 + 10 C5 + C6 + 11 C12 + 10 C13 + C14 = 10.46518036156088
 R3: 10 C4 + 9 C5 + C7 + 10 C12 + 9 C13 + C15 = 9.61665222413707
 R4: C16 + C17 + C18 + C19 + C20 + C21 + C22 + C23 - C24 = 0
 R5: - C0 - 100 C16 <= 0
 R6: - C1 - 100 C17 <= 0
 R7: - C2 - 100 C18 <= 0
 R8: - C3 - 100 C19 <= 0
```

```
R9: - C4 - 100 C16 <= 0
R10: - C5 - 100 C17 <= 0
R11: - C6 - 100 C18 <= 0
R12: - C7 - 100 C19 <= 0
R13: - C8 - 100 C20 <= 0
R14: - C9 - 100 C21 <= 0
R15: - C10 - 100 C22 <= 0
R16: - C11 - 100 C23 <= 0
R17: - C12 - 100 C20 <= 0
R18: - C13 - 100 C21 <= 0
R19: - C14 - 100 C22 <= 0
R20: - C15 - 100 C23 <= 0
R21: C0 - 100 C16 <= 0
R22: C1 - 100 C17 <= 0
R23: C2 - 100 C18 <= 0
R24: C3 - 100 C19 <= 0
R25: C4 - 100 C16 <= 0
R26: C5 - 100 C17 <= 0
R27: C6 - 100 C18 <= 0
R28: C7 - 100 C19 <= 0
R29: C8 - 100 C20 <= 0
R30: C9 - 100 C21 <= 0
R31: C10 - 100 C22 <= 0
R32: C11 - 100 C23 <= 0
R33: C12 - 100 C20 <= 0
R34: C13 - 100 C21 <= 0
R35: C14 - 100 C22 <= 0
R36: C15 - 100 C23 <= 0
R37: C24 = 4
R38: C0 + 100 C20 <= 100
R39: C1 + 100 C21 <= 100
R40: C2 + 100 C22 <= 100
R41: C3 + 100 C23 <= 100
R42: C4 + 100 C20 <= 100
R43: C5 + 100 C21 <= 100
R44: C6 + 100 C22 <= 100
R45: C7 + 100 C23 <= 100
R46: C8 + 100 C16 <= 100
R47: C9 + 100 C17 <= 100
R48: C10 + 100 C18 <= 100
R49: C11 + 100 C19 <= 100
R50: C12 + 100 C16 <= 100
R51: C13 + 100 C17 <= 100
R52: C14 + 100 C18 <= 100
R53: C15 + 100 C19 <= 100
R54: - C0 + 100 C20 <= 100
R55: - C1 + 100 C21 <= 100
R56: - C2 + 100 C22 <= 100
R57: - C3 + 100 C23 <= 100
R58: - C4 + 100 C20 <= 100
R59: - C5 + 100 C21 <= 100
R60: - C6 + 100 C22 <= 100
R61: - C7 + 100 C23 <= 100
R62: - C8 + 100 C16 <= 100
R63: - C9 + 100 C17 <= 100
R64: - C10 + 100 C18 <= 100
R65: - C11 + 100 C19 <= 100
```

```
 R66: - C12 + 100 C16 <= 100
 R67: - C13 + 100 C17 <= 100
 R68: - C14 + 100 C18 <= 100
 R69: - C15 + 100 C19 <= 100
Bounds
 C0 free
 C1 free
 C2 free
 C3 free
 C4 free
 C5 free
 C6 free
 C7 free
 C8 free
 C9 free
 C10 free
 C11 free
 C12 free
 C13 free
 C14 free
 C15 free
 C24 <= 4
Binaries
 C16 C17 C18 C19 C20 C21 C22 C23
Generals
 C24
End
```

B.2 Constrained Dummy Problem

```
Minimize
   0.2 $y_1w_1$ + 0.2 $y_2w_1$ + 0.2 $y_3w_1$ + 0.2 $y_4w_1$ + 0.02 ...
       $y_1w_2$
   + 0.02 $y_2w_2$ + 0.02 $y_3w_2$ + 0.02 $y_4w_2$ + 0.2 $y_5w_1$
   + 0.02 $y_5w_2$ + 0.1 H_y5u1w1 + H_y5u1w2 + [ 0.52 H_y1u2w1 ^2
   - 2 H_y1u2w1 * H_y2u2w1 - 5.3 H_y1u2w1 * H_y3u2w1
   - 10.5 H_y1u2w1 * H_y4u2w1 + 2.02 H_y2u2w1 ^2 + 10.6 H_y2u2w1 * ...
       H_y3u2w1
   + 21 H_y2u2w1 * H_y4u2w1 + 14.065 H_y3u2w1 ^2
   + 55.65 H_y3u2w1 * H_y4u2w1 + 55.145 H_y4u2w1 ^2 + 2.5 H_y1u2w2 ^2
   - 2 H_y1u2w2 * H_y2u2w2 - 5.3 H_y1u2w2 * H_y3u2w2
   - 10.5 H_y1u2w2 * H_y4u2w2 + 4 H_y2u2w2 ^2 + 10.6 H_y2u2w2 * ...
       H_y3u2w2
   + 21 H_y2u2w2 * H_y4u2w2 + 16.045 H_y3u2w2 ^2
   + 55.65 H_y3u2w2 * H_y4u2w2 + 57.125 H_y4u2w2 ^2 ] / 2
Subject To
 R0: H_y2u2w1 + 2 H_y3u2w1 + 3 H_y4u2w1 + H_y2u2w2 + 2 H_y3u2w2
   + 3 H_y4u2w2 = 1
 R1: $y_1w_1$ + $y_2w_1$ + $y_3w_1$ + $y_4w_1$ + $y_1w_2$ + $y_2w_2$
   + $y_3w_2$ + $y_4w_2$ - nm = 0
 R2: - H_y1u2w1 - 500 $y_1w_1$ <= 0
 R3: - H_y2u2w1 - 500 $y_2w_1$ <= 0
 R4: - H_y3u2w1 - 500 $y_3w_1$ <= 0
 R5: - H_y4u2w1 - 500 $y_4w_1$ <= 0
 R6: - H_y1u2w2 - 500 $y_1w_2$ <= 0
 R7: - H_y2u2w2 - 500 $y_2w_2$ <= 0
 R8: - H_y3u2w2 - 500 $y_3w_2$ <= 0
 R9: - H_y4u2w2 - 500 $y_4w_2$ <= 0
 R10: H_y1u2w1 - 500 $y_1w_1$ <= 0
 R11: H_y2u2w1 - 500 $y_2w_1$ <= 0
 R12: H_y3u2w1 - 500 $y_3w_1$ <= 0
 R13: H_y4u2w1 - 500 $y_4w_1$ <= 0
 R14: H_y1u2w2 - 500 $y_1w_2$ <= 0
 R15: H_y2u2w2 - 500 $y_2w_2$ <= 0
 R16: H_y3u2w2 - 500 $y_3w_2$ <= 0
 R17: H_y4u2w2 - 500 $y_4w_2$ <= 0
 R18: nm = 4
 R19: H_y1u2w1 + 500 $y_1w_2$ <= 500
 R20: H_y2u2w1 + 500 $y_2w_2$ <= 500
 R21: H_y3u2w1 + 500 $y_3w_2$ <= 500
 R22: H_y4u2w1 + 500 $y_4w_2$ <= 500
 R23: H_y1u2w2 + 500 $y_1w_1$ <= 500
 R24: H_y2u2w2 + 500 $y_2w_1$ <= 500
 R25: H_y3u2w2 + 500 $y_3w_1$ <= 500
 R26: H_y4u2w2 + 500 $y_4w_1$ <= 500
 R27: - H_y1u2w1 + 500 $y_1w_2$ <= 500
 R28: - H_y2u2w1 + 500 $y_2w_2$ <= 500
 R29: - H_y3u2w1 + 500 $y_3w_2$ <= 500
 R30: - H_y4u2w1 + 500 $y_4w_2$ <= 500
 R31: - H_y1u2w2 + 500 $y_1w_1$ <= 500
 R32: - H_y2u2w2 + 500 $y_2w_1$ <= 500
 R33: - H_y3u2w2 + 500 $y_3w_1$ <= 500
 R34: - H_y4u2w2 + 500 $y_4w_1$ <= 500
 R35: H_y5u1w1 + H_y5u1w2 = 1
 R36: $y_5w_1$ + $y_5w_2$ = 1
Bounds
 H_y1u2w1 free
 H_y2u2w1 free
 H_y3u2w1 free
 H_y4u2w1 free
 H_y1u2w2 free
 H_y2u2w2 free
 H_y3u2w2 free
 H_y4u2w2 free
 nm <= 4
Binaries
 $y_1w_1$ $y_2w_1$ $y_3w_1$ $y_4w_1$ $y_1w_2$ $y_2w_2$ $y_3w_2$ ...
     $y_4w_2$
 $y_5w_1$ $y_5w_2$ H_y5u1w1 H_y5u1w2
Generals
 nm
SOS
 s0:  S1 :: $y_5w_2$:1 H_y5u1w1:1
End
```

B.3 Subsea Case: Inactive Constraint and Budget Constraint

```
Minimize
  0.1 sigma_y1w1 + sigma_y2w1 + 0.1 sigma_y3w1 + sigma_y4w1 + 0.1 ...
      sigma_y5w1
   + sigma_y6w1 + 0.1 sigma_y7w1 + sigma_y8w1 + 0.1 sigma_y9w1
   + sigma_y10w1 + 0.1 sigma_y11w1 + sigma_y12w1 + 0.1 sigma_y13w1
   + 0.1 sigma_y15w1 + sigma_y16w1 + 0.05 sigma_y1w2 + 0.1 sigma_y2w2
   + 0.05 sigma_y3w2 + 0.1 sigma_y4w2 + 0.05 sigma_y5w2 + 0.1 ...
       sigma_y6w2
   + 0.05 sigma_y7w2 + 0.1 sigma_y8w2 + 0.05 sigma_y9w2 + 0.1 ...
       sigma_y10w2
   + 0.05 sigma_y11w2 + 0.1 sigma_y12w2 + 0.05 sigma_y13w2
   + 0.05 sigma_y15w2 + 0.1 sigma_y16w2 + 0.5 sigma_y14w1
   + 0.05 sigma_y14w2 + 4.6876e-08 H_y14u3w1 + 4.6876e-07 H_y14u3w2 ...
       + [
   2.05106e-06 H_y1u1w1 ^2 - 0.0011758790300657 H_y1u1w1 * H_y2u1w1
   - 2.2176e-07 H_y1u1w1 * H_y3u1w1
   - 1.0784676742445517e-04 H_y1u1w1 * H_y4u1w1
   + 3.4765e-06 H_y1u1w1 * H_y5u1w1
   - 0.00112207174454022 H_y1u1w1 * H_y6u1w1
   + 6.25246e-07 H_y1u1w1 * H_y7u1w1
   - 0.00104639528555011 H_y1u1w1 * H_y8u1w1
   + 4.03486e-07 H_y1u1w1 * H_y9u1w1 - 0.0001013 H_y1u1w1 * H_y10u1w1
   - 5.46843e-07 H_y1u1w1 * H_y11u1w1
   - 1.6862640249887278e-04 H_y1u1w1 * H_y12u1w1
   + 9.50329e-07 H_y1u1w1 * H_y13u1w1 + 2.92965e-06 H_y1u1w1 * ...
       H_y15u1w1
   - 6.4005539531045608e-04 H_y1u1w1 * H_y16u1w1
   + 0.2560617341559827 H_y2u1w1 ^2
   + 2.4487874710108203e-04 H_y2u1w1 * H_y3u1w1
   + 0.1278197511216987 H_y2u1w1 * H_y4u1w1
   - 0.000786141 H_y2u1w1 * H_y5u1w1
   + 0.3936303332784352 H_y2u1w1 * H_y6u1w1
   - 0.000389738 H_y2u1w1 * H_y7u1w1
   + 0.9451428507673202 H_y2u1w1 * H_y8u1w1
   - 1.4485927075530913e-04 H_y2u1w1 * H_y9u1w1
   + 0.1337454565160713 H_y2u1w1 * H_y10u1w1
   + 7.1829771912962576e-04 H_y2u1w1 * H_y11u1w1
   + 0.1813111469841493 H_y2u1w1 * H_y12u1w1
   - 0.000863157 H_y2u1w1 * H_y13u1w1 - 6.78433e-05 H_y2u1w1 * ...
       H_y15u1w1
   + 0.2331254765029068 H_y2u1w1 * H_y16u1w1 + 1.00094e-07 H_y3u1w1 ^2
   + 1.062113515606881e-04 H_y3u1w1 * H_y4u1w1
   + 3.00931e-08 H_y3u1w1 * H_y5u1w1
   + 1.3520449449874706e-04 H_y3u1w1 * H_y6u1w1
   - 2.51853e-07 H_y3u1w1 * H_y7u1w1
   + 7.2489481218670148e-04 H_y3u1w1 * H_y8u1w1
   - 5.20508e-08 H_y3u1w1 * H_y9u1w1 + 0.00011394 H_y3u1w1 * H_y10u1w1
   + 6.1125e-07 H_y3u1w1 * H_y11u1w1
   + 1.468591868055178e-04 H_y3u1w1 * H_y12u1w1
   - 6.63301e-07 H_y3u1w1 * H_y13u1w1 + 6.41343e-07 H_y3u1w1 * ...
       H_y15u1w1
```

```
    + 8.60218e-05 H_y3u1w1 * H_y16u1w1 + 0.0282441149930041 H_y4u1w1 ^2
    + 2.51339e-05 H_y4u1w1 * H_y5u1w1
    + 0.0693420227334188 H_y4u1w1 * H_y6u1w1
    - 1.3298069215577417e-04 H_y4u1w1 * H_y7u1w1
    + 0.3847118906703809 H_y4u1w1 * H_y8u1w1
    - 2.67693e-05 H_y4u1w1 * H_y9u1w1
    + 0.0606339992451147 H_y4u1w1 * H_y10u1w1
    + 3.2527239935002455e-04 H_y4u1w1 * H_y11u1w1
    + 0.078053131902459 H_y4u1w1 * H_y12u1w1
    - 3.5204173994488654e-04 H_y4u1w1 * H_y13u1w1
    + 3.5040632408146104e-04 H_y4u1w1 * H_y15u1w1
    + 0.0443100304573928 H_y4u1w1 * H_y16u1w1 + 1.60005e-06 H_y5u1w1 ^2
    - 8.6448934890581382e-04 H_y5u1w1 * H_y6u1w1
    + 2.76784e-07 H_y5u1w1 * H_y7u1w1
    - 1.1097248953901862e-04 H_y5u1w1 * H_y8u1w1
    + 3.06877e-07 H_y5u1w1 * H_y9u1w1 + 4.00665e-05 H_y5u1w1 * H_y10u1w1
    + 2.11846e-07 H_y5u1w1 * H_y11u1w1 + 1.69959e-05 H_y5u1w1 * ...
        H_y12u1w1
    + 9.50305e-08 H_y5u1w1 * H_y13u1w1 + 3.41156e-06 H_y5u1w1 * ...
        H_y15u1w1
    - 0.000482794 H_y5u1w1 * H_y16u1w1 + 0.1682697096011319 H_y6u1w1 ^2
    - 2.5758239563441348e-04 H_y6u1w1 * H_y7u1w1
    + 0.5514979461831324 H_y6u1w1 * H_y8u1w1
    - 0.000122378 H_y6u1w1 * H_y9u1w1
    + 0.0707589478282818 H_y6u1w1 * H_y10u1w1
    + 0.000380456 H_y6u1w1 * H_y11u1w1
    + 0.1007971916201678 H_y6u1w1 * H_y12u1w1
    - 5.0283386692081644e-04 H_y6u1w1 * H_y13u1w1
    - 4.8403338312081766e-04 H_y6u1w1 * H_y15u1w1
    + 0.1955001364558541 H_y6u1w1 * H_y16u1w1 + 1.74424e-07 H_y7u1w1 ^2
    - 9.3542279601113699e-04 H_y7u1w1 * H_y8u1w1
    + 9.66093e-08 H_y7u1w1 * H_y9u1w1
    - 1.4136654198423814e-04 H_y7u1w1 * H_y10u1w1
    - 7.5869e-07 H_y7u1w1 * H_y11u1w1
    - 1.8562234891509341e-04 H_y7u1w1 * H_y12u1w1
    + 8.55299e-07 H_y7u1w1 * H_y13u1w1 - 4.81906e-07 H_y7u1w1 * ...
        H_y15u1w1
    - 1.572614535664045e-04 H_y7u1w1 * H_y16u1w1
    + 1.322567291666435 H_y8u1w1 ^2 - 0.000210528 H_y8u1w1 * H_y9u1w1
    + 0.4118031201287415 H_y8u1w1 * H_y10u1w1
    + 0.0022094002926212 H_y8u1w1 * H_y11u1w1
    + 0.5331673263470015 H_y8u1w1 * H_y12u1w1
    - 0.00241992827644428 H_y8u1w1 * H_y13u1w1
    + 0.00209842780308302 H_y8u1w1 * H_y15u1w1
    + 0.3462502596962378 H_y8u1w1 * H_y16u1w1 + 2.24722e-08 H_y9u1w1 ^2
    - 2.74266e-05 H_y9u1w1 * H_y10u1w1 - 1.4744e-07 H_y9u1w1 * H_y11u1w1
    - 3.87632e-05 H_y9u1w1 * H_y12u1w1 + 1.91998e-07 H_y9u1w1 * ...
        H_y13u1w1
    + 1.59437e-07 H_y9u1w1 * H_y15u1w1 - 7.12397e-05 H_y9u1w1 * ...
        H_y16u1w1
    + 0.032571026112372 H_y10u1w1 ^2
    + 3.4943252974706013e-04 H_y10u1w1 * H_y11u1w1
    + 0.0837117645940744 H_y10u1w1 * H_y12u1w1
    - 3.7685910544789462e-04 H_y10u1w1 * H_y13u1w1
    + 0.000389499 H_y10u1w1 * H_y15u1w1
    + 0.0455012061103151 H_y10u1w1 * H_y16u1w1 + 9.3743e-07 ...
        H_y11u1w1 ^2
    + 4.4909076423386112e-04 H_y11u1w1 * H_y12u1w1
    - 2.02191e-06 H_y11u1w1 * H_y13u1w1 + 2.08632e-06 H_y11u1w1 * ...
        H_y15u1w1
    + 2.4457940120790401e-04 H_y11u1w1 * H_y16u1w1
    + 0.0539775445068836 H_y12u1w1 ^2 - 0.000487854 H_y12u1w1 * ...
        H_y13u1w1
    + 4.6608671065024329e-04 H_y12u1w1 * H_y15u1w1
    + 0.064022876007293 H_y12u1w1 * H_y16u1w1 + 1.10715e-06 ...
        H_y13u1w1 ^2
    - 1.92688e-06 H_y13u1w1 * H_y15u1w1 - 0.000315819 H_y13u1w1 * ...
        H_y16u1w1
    + 2.74913e-06 H_y15u1w1 ^2
    - 2.3821454053607547e-04 H_y15u1w1 * H_y16u1w1
    + 0.0569778797619619 H_y16u1w1 ^2 + 2.05106e-06 H_y1u2w1 ^2
    - 0.0011758790300657 H_y1u2w1 * H_y2u2w1
    - 2.2176e-07 H_y1u2w1 * H_y3u2w1
    - 1.0784676742445517e-04 H_y1u2w1 * H_y4u2w1
    + 3.4765e-06 H_y1u2w1 * H_y5u2w1
    - 0.00112207174454022 H_y1u2w1 * H_y6u2w1
    + 6.25246e-07 H_y1u2w1 * H_y7u2w1
    - 0.00104639528555011 H_y1u2w1 * H_y8u2w1
    + 4.03486e-07 H_y1u2w1 * H_y9u2w1 - 0.0001013 H_y1u2w1 * H_y10u2w1
    - 5.46843e-07 H_y1u2w1 * H_y11u2w1
    - 1.6862640249887278e-04 H_y1u2w1 * H_y12u2w1
    + 9.50329e-07 H_y1u2w1 * H_y13u2w1 + 2.92965e-06 H_y1u2w1 * ...
        H_y15u2w1
    - 6.4005539531045608e-04 H_y1u2w1 * H_y16u2w1
    + 0.2560617341559827 H_y2u2w1 ^2
    + 2.4487874710108203e-04 H_y2u2w1 * H_y3u2w1
    + 0.1278197511216987 H_y2u2w1 * H_y4u2w1
    - 0.000786141 H_y2u2w1 * H_y5u2w1
    + 0.3936303332784352 H_y2u2w1 * H_y6u2w1
    - 0.000389738 H_y2u2w1 * H_y7u2w1
    + 0.9451428507673202 H_y2u2w1 * H_y8u2w1
    - 1.4485927075530913e-04 H_y2u2w1 * H_y9u2w1
    + 0.1337454565160713 H_y2u2w1 * H_y10u2w1
    + 7.1829771912962576e-04 H_y2u2w1 * H_y11u2w1
    + 0.1813111469841493 H_y2u2w1 * H_y12u2w1
    - 0.000863157 H_y2u2w1 * H_y13u2w1 - 6.78433e-05 H_y2u2w1 * ...
        H_y15u2w1
    + 0.2331254765029068 H_y2u2w1 * H_y16u2w1 + 1.00094e-07 H_y3u2w1 ^2
    + 1.062113515606881e-04 H_y3u2w1 * H_y4u2w1
    + 3.00931e-08 H_y3u2w1 * H_y5u2w1
    + 1.3520449449874706e-04 H_y3u2w1 * H_y6u2w1
    - 2.51853e-07 H_y3u2w1 * H_y7u2w1
    + 7.2489481218670148e-04 H_y3u2w1 * H_y8u2w1
    - 5.20508e-08 H_y3u2w1 * H_y9u2w1 + 0.00011394 H_y3u2w1 * H_y10u2w1
    + 6.1125e-07 H_y3u2w1 * H_y11u2w1
    + 1.468591868055178e-04 H_y3u2w1 * H_y12u2w1
    - 6.63301e-07 H_y3u2w1 * H_y13u2w1 + 6.41343e-07 H_y3u2w1 * ...
        H_y15u2w1
    + 8.60218e-05 H_y3u2w1 * H_y16u2w1 + 0.0282441149930041 H_y4u2w1 ^2
    + 2.51339e-05 H_y4u2w1 * H_y5u2w1
    + 0.0693420227334188 H_y4u2w1 * H_y6u2w1
    - 1.3298069215577417e-04 H_y4u2w1 * H_y7u2w1
    + 0.3847118906703809 H_y4u2w1 * H_y8u2w1
    - 2.67693e-05 H_y4u2w1 * H_y9u2w1
    + 0.0606339992451147 H_y4u2w1 * H_y10u2w1
    + 3.2527239935002455e-04 H_y4u2w1 * H_y11u2w1
    + 0.078053131902459 H_y4u2w1 * H_y12u2w1
    - 3.5204173994488654e-04 H_y4u2w1 * H_y13u2w1
    + 3.5040632408146104e-04 H_y4u2w1 * H_y15u2w1
    + 0.0443100304573928 H_y4u2w1 * H_y16u2w1 + 1.60005e-06 H_y5u2w1 ^2
    - 8.6448934890581382e-04 H_y5u2w1 * H_y6u2w1
    + 2.76784e-07 H_y5u2w1 * H_y7u2w1
    - 1.1097248953901862e-04 H_y5u2w1 * H_y8u2w1
    + 3.06877e-07 H_y5u2w1 * H_y9u2w1 + 4.00665e-05 H_y5u2w1 * H_y10u2w1
    + 2.11846e-07 H_y5u2w1 * H_y11u2w1 + 1.69959e-05 H_y5u2w1 * ...
        H_y12u2w1
    + 9.50305e-08 H_y5u2w1 * H_y13u2w1 + 3.41156e-06 H_y5u2w1 * ...
        H_y15u2w1
    - 0.000482794 H_y5u2w1 * H_y16u2w1 + 0.168269709601131 9 H_y6u2w1 ^2
    - 2.5758239563441348e-04 H_y6u2w1 * H_y7u2w1
    + 0.5514979461831324 H_y6u2w1 * H_y8u2w1
    - 0.000122378 H_y6u2w1 * H_y9u2w1
    + 0.0707589478282818 H_y6u2w1 * H_y10u2w1
    + 0.000380456 H_y6u2w1 * H_y11u2w1
    + 0.1007971916201678 H_y6u2w1 * H_y12u2w1
    - 5.0283386692081644e-04 H_y6u2w1 * H_y13u2w1
    - 4.8403338312081766e-04 H_y6u2w1 * H_y15u2w1
    + 0.1955001364558541 H_y6u2w1 * H_y16u2w1 + 1.74424e-07 H_y7u2w1 ^2
    - 9.3542279601113699e-04 H_y7u2w1 * H_y8u2w1
    + 9.66093e-08 H_y7u2w1 * H_y9u2w1
    - 1.4136654198423814e-04 H_y7u2w1 * H_y10u2w1
    - 7.5869e-07 H_y7u2w1 * H_y11u2w1
    - 1.8562234891509341e-04 H_y7u2w1 * H_y12u2w1
    + 8.55299e-07 H_y7u2w1 * H_y13u2w1 - 4.81906e-07 H_y7u2w1 * ...
        H_y15u2w1
    - 1.572614535664045e-04 H_y7u2w1 * H_y16u2w1
    + 1.322567291666435 H_y8u2w1 ^2 - 0.000210528 H_y8u2w1 * H_y9u2w1
    + 0.4118031201287415 H_y8u2w1 * H_y10u2w1
    + 0.0022094002926212 H_y8u2w1 * H_y11u2w1
    + 0.5331673263470015 H_y8u2w1 * H_y12u2w1
    - 0.00241992827644428 H_y8u2w1 * H_y13u2w1
    + 0.00209842780308302 H_y8u2w1 * H_y15u2w1
    + 0.3462502596962378 H_y8u2w1 * H_y16u2w1 + 2.24722e-08 H_y9u2w1 ^2
    - 2.74266e-05 H_y9u2w1 * H_y10u2w1 - 1.4744e-07 H_y9u2w1 * H_y11u2w1
    - 3.87632e-05 H_y9u2w1 * H_y12u2w1 + 1.91998e-07 H_y9u2w1 * ...
        H_y13u2w1
    + 1.59437e-07 H_y9u2w1 * H_y15u2w1 - 7.12397e-05 H_y9u2w1 * ...
        H_y16u2w1
    + 0.032571026112372 H_y10u2w1 ^2
    + 3.4943252974706013e-04 H_y10u2w1 * H_y11u2w1
    + 0.0837117645940744 H_y10u2w1 * H_y12u2w1
    - 3.7685910544789462e-04 H_y10u2w1 * H_y13u2w1
    + 0.000389499 H_y10u2w1 * H_y15u2w1
    + 0.0455012061103151 H_y10u2w1 * H_y16u2w1 + 9.3743e-07 ...
        H_y11u2w1 ^2
    + 4.4909076423386112e-04 H_y11u2w1 * H_y12u2w1
    - 2.02191e-06 H_y11u2w1 * H_y13u2w1 + 2.08632e-06 H_y11u2w1 * ...
        H_y15u2w1
    + 2.4457940120790401e-04 H_y11u2w1 * H_y16u2w1
    + 0.0539775445068836 H_y12u2w1 ^2 - 0.000487854 H_y12u2w1 * ...
        H_y13u2w1
```

```
+ 4.6608671065024329e-04 H_y12u2w1 * H_y15u2w1
+ 0.064022876007293 H_y12u2w1 * H_y16u2w1 + 1.10715e-06 ...
    H_y13u2w1 ^2
- 1.92688e-06 H_y13u2w1 * H_y15u2w1 - 0.000315819 H_y13u2w1 * ...
    H_y16u2w1
+ 2.74913e-06 H_y15u2w1 ^2
- 2.3821454053607547e-04 H_y15u2w1 * H_y16u2w1
+ 0.0569778797619619 H_y16u2w1 ^2 + 2.05164e-06 H_y1u1w2 ^2
- 0.0011758790300657 H_y1u1w2 * H_y2u1w2
- 2.2176e-07 H_y1u1w2 * H_y3u1w2
- 1.0784676742445517e-04 H_y1u1w2 * H_y4u1w2
+ 3.4765e-06 H_y1u1w2 * H_y5u1w2
- 0.00112207174454022 H_y1u1w2 * H_y6u1w2
+ 6.25246e-07 H_y1u1w2 * H_y7u1w2
- 0.00104639528555011 H_y1u1w2 * H_y8u1w2
+ 4.03486e-07 H_y1u1w2 * H_y9u1w2 - 0.0001013 H_y1u1w2 * H_y10u1w2
- 5.46843e-07 H_y1u1w2 * H_y11u1w2
- 1.6862640249887278e-04 H_y1u1w2 * H_y12u1w2
+ 9.50329e-07 H_y1u1w2 * H_y13u1w2 + 2.92965e-06 H_y1u1w2 * ...
    H_y15u1w2
- 6.4005539531045608e-04 H_y1u1w2 * H_y16u1w2
+ 0.2561607341559827 H_y2u1w2 ^2
+ 2.4487874710108203e-04 H_y2u1w2 * H_y3u1w2
+ 0.1278197511216987 H_y2u1w2 * H_y4u1w2
- 0.000786141 H_y2u1w2 * H_y5u1w2
+ 0.3936303332784352 H_y2u1w2 * H_y6u1w2
- 0.000389738 H_y2u1w2 * H_y7u1w2
+ 0.9451428507673202 H_y2u1w2 * H_y8u1w2
- 1.4485927075530913e-04 H_y2u1w2 * H_y9u1w2
+ 0.1337454565160713 H_y2u1w2 * H_y10u1w2
+ 7.1829771912962576e-04 H_y2u1w2 * H_y11u1w2
+ 0.1813111469841493 H_y2u1w2 * H_y12u1w2
- 0.000863157 H_y2u1w2 * H_y13u1w2 - 6.78433e-05 H_y2u1w2 * ...
    H_y15u1w2
+ 0.2331254765029068 H_y2u1w2 * H_y16u1w2 + 1.00673e-07 H_y3u1w2 ^2
+ 1.062113515606881e-04 H_y3u1w2 * H_y4u1w2
+ 3.00931e-08 H_y3u1w2 * H_y5u1w2
+ 1.3520449449874706e-04 H_y3u1w2 * H_y6u1w2
- 2.51853e-07 H_y3u1w2 * H_y7u1w2
+ 7.2489481218670148e-04 H_y3u1w2 * H_y8u1w2
- 5.20508e-08 H_y3u1w2 * H_y9u1w2 + 0.00011394 H_y3u1w2 * H_y10u1w2
+ 6.1125e-07 H_y3u1w2 * H_y11u1w2
+ 1.468591868055178e-04 H_y3u1w2 * H_y12u1w2
- 6.63301e-07 H_y3u1w2 * H_y13u1w2 + 6.41343e-07 H_y3u1w2 * ...
    H_y15u1w2
+ 8.60218e-05 H_y3u1w2 * H_y16u1w2 + 0.0283431149930041 H_y4u1w2 ^2
+ 2.51339e-05 H_y4u1w2 * H_y5u1w2
+ 0.0693420227334188 H_y4u1w2 * H_y6u1w2
- 1.3298069215577417e-04 H_y4u1w2 * H_y7u1w2
+ 0.3847118906703809 H_y4u1w2 * H_y8u1w2
- 2.67693e-05 H_y4u1w2 * H_y9u1w2
+ 0.0606339992451147 H_y4u1w2 * H_y10u1w2
+ 3.2527239935002455e-04 H_y4u1w2 * H_y11u1w2
+ 0.078053131902459 H_y4u1w2 * H_y12u1w2
- 3.5204173994488654e-04 H_y4u1w2 * H_y13u1w2
+ 3.5040632408146104e-04 H_y4u1w2 * H_y15u1w2
+ 0.0443100304573928 H_y4u1w2 * H_y16u1w2 + 1.60063e-06 H_y5u1w2 ^2
```

```
   - 8.6448934890581382e-04 H_y5u1w2 * H_y6u1w2
   + 2.76784e-07 H_y5u1w2 * H_y7u1w2
   - 1.1097248953901862e-04 H_y5u1w2 * H_y8u1w2
   + 3.06877e-07 H_y5u1w2 * H_y9u1w2 + 4.00665e-05 H_y5u1w2 * H_y10u1w2
   + 2.11846e-07 H_y5u1w2 * H_y11u1w2 + 1.69959e-05 H_y5u1w2 * ...
       H_y12u1w2
   + 9.50305e-08 H_y5u1w2 * H_y13u1w2 + 3.41156e-06 H_y5u1w2 * ...
       H_y15u1w2
   - 0.000482794 H_y5u1w2 * H_y16u1w2 + 0.168368709601319 H_y6u1w2 ^2
   - 2.5758239563441348e-04 H_y6u1w2 * H_y7u1w2
   + 0.5514979461831324 H_y6u1w2 * H_y8u1w2
   - 0.000122378 H_y6u1w2 * H_y9u1w2
   + 0.0707589478282818 H_y6u1w2 * H_y10u1w2
   + 0.000380456 H_y6u1w2 * H_y11u1w2
   + 0.1007971916201678 H_y6u1w2 * H_y12u1w2
   - 5.0283386692081644e-04 H_y6u1w2 * H_y13u1w2
   - 4.8403338312081766e-04 H_y6u1w2 * H_y15u1w2
   + 0.1955001364558541 H_y6u1w2 * H_y16u1w2 + 1.75003e-07 H_y7u1w2 ^2
   - 9.3542279601113699e-04 H_y7u1w2 * H_y8u1w2
   + 9.66093e-08 H_y7u1w2 * H_y9u1w2
   - 1.4136654198423814e-04 H_y7u1w2 * H_y10u1w2
   - 7.5869e-07 H_y7u1w2 * H_y11u1w2
   - 1.8562234891509341e-04 H_y7u1w2 * H_y12u1w2
   + 8.55299e-07 H_y7u1w2 * H_y13u1w2 - 4.81906e-07 H_y7u1w2 * ...
       H_y15u1w2
   - 1.572614535664045e-04 H_y7u1w2 * H_y16u1w2
   + 1.322666291666435 H_y8u1w2 ^2 - 0.000210528 H_y8u1w2 * H_y9u1w2
   + 0.4118031201287415 H_y8u1w2 * H_y10u1w2
   + 0.0022094002926212 H_y8u1w2 * H_y11u1w2
   + 0.5331673263470015 H_y8u1w2 * H_y12u1w2
   - 0.00241992827644428 H_y8u1w2 * H_y13u1w2
   + 0.00209842780308302 H_y8u1w2 * H_y15u1w2
   + 0.3462502596962378 H_y8u1w2 * H_y16u1w2 + 2.30509e-08 H_y9u1w2 ^2
   - 2.74266e-05 H_y9u1w2 * H_y10u1w2 - 1.4744e-07 H_y9u1w2 * H_y11u1w2
   - 3.87632e-05 H_y9u1w2 * H_y12u1w2 + 1.91998e-07 H_y9u1w2 * ...
       H_y13u1w2
   + 1.59437e-07 H_y9u1w2 * H_y15u1w2 - 7.12397e-05 H_y9u1w2 * ...
       H_y16u1w2
   + 0.032670026112372 H_y10u1w2 ^2
   + 3.4943252974706013e-04 H_y10u1w2 * H_y11u1w2
   + 0.0837117645940744 H_y10u1w2 * H_y12u1w2
   - 3.7685910544789462e-04 H_y10u1w2 * H_y13u1w2
   + 0.000389499 H_y10u1w2 * H_y15u1w2
   + 0.0455012061103151 H_y10u1w2 * H_y16u1w2 + 9.38008e-07 ...
       H_y11u1w2 ^2
   + 4.4909076423386112e-04 H_y11u1w2 * H_y12u1w2
   - 2.02191e-06 H_y11u1w2 * H_y13u1w2 + 2.08632e-06 H_y11u1w2 * ...
       H_y15u1w2
   + 2.4457940120790401e-04 H_y11u1w2 * H_y16u1w2
   + 0.0540765445068836 H_y12u1w2 ^2 - 0.000487854 H_y12u1w2 * ...
       H_y13u1w2
   + 4.6608671065024329e-04 H_y12u1w2 * H_y15u1w2
   + 0.064022876007293 H_y12u1w2 * H_y16u1w2 + 1.10773e-06 ...
       H_y13u1w2 ^2
   - 1.92688e-06 H_y13u1w2 * H_y15u1w2 - 0.000315819 H_y13u1w2 * ...
       H_y16u1w2
   + 2.74971e-06 H_y15u1w2 ^2
```

```
    - 2.3821454053607547e-04 H_y15u1w2 * H_y16u1w2
    + 0.0570768797619619 H_y16u1w2 ^2 + 2.05164e-06 H_y1u2w2 ^2
    - 0.0011758790300657 H_y1u2w2 * H_y2u2w2
    - 2.2176e-07 H_y1u2w2 * H_y3u2w2
    - 1.0784676742445517e-04 H_y1u2w2 * H_y4u2w2
    + 3.4765e-06 H_y1u2w2 * H_y5u2w2
    - 0.00112207174454022 H_y1u2w2 * H_y6u2w2
    + 6.25246e-07 H_y1u2w2 * H_y7u2w2
    - 0.00104639528555011 H_y1u2w2 * H_y8u2w2
    + 4.03486e-07 H_y1u2w2 * H_y9u2w2 - 0.0001013 H_y1u2w2 * H_y10u2w2
    - 5.46843e-07 H_y1u2w2 * H_y11u2w2
    - 1.6862640249887278e-04 H_y1u2w2 * H_y12u2w2
    + 9.50329e-07 H_y1u2w2 * H_y13u2w2 + 2.92965e-06 H_y1u2w2 * ...
        H_y15u2w2
    - 6.4005539531045608e-04 H_y1u2w2 * H_y16u2w2
    + 0.2561607341559827 H_y2u2w2 ^2
    + 2.4487874710108203e-04 H_y2u2w2 * H_y3u2w2
    + 0.1278197511216987 H_y2u2w2 * H_y4u2w2
    - 0.000786141 H_y2u2w2 * H_y5u2w2
    + 0.3936303332784352 H_y2u2w2 * H_y6u2w2
    - 0.000389738 H_y2u2w2 * H_y7u2w2
    + 0.9451428507673202 H_y2u2w2 * H_y8u2w2
    - 1.4485927075530913e-04 H_y2u2w2 * H_y9u2w2
    + 0.1337454565160713 H_y2u2w2 * H_y10u2w2
    + 7.1829771912962576e-04 H_y2u2w2 * H_y11u2w2
    + 0.1813111469841493 H_y2u2w2 * H_y12u2w2
    - 0.000863157 H_y2u2w2 * H_y13u2w2 - 6.78433e-05 H_y2u2w2 * ...
        H_y15u2w2
    + 0.2331254765029068 H_y2u2w2 * H_y16u2w2 + 1.00673e-07 H_y3u2w2 ^2
    + 1.062113515606881e-04 H_y3u2w2 * H_y4u2w2
    + 3.00931e-08 H_y3u2w2 * H_y5u2w2
    + 1.3520449449874706e-04 H_y3u2w2 * H_y6u2w2
    - 2.51853e-07 H_y3u2w2 * H_y7u2w2
    + 7.2489481218670148e-04 H_y3u2w2 * H_y8u2w2
    - 5.20508e-08 H_y3u2w2 * H_y9u2w2 + 0.00011394 H_y3u2w2 * H_y10u2w2
    + 6.1125e-07 H_y3u2w2 * H_y11u2w2
    + 1.468591868055178e-04 H_y3u2w2 * H_y12u2w2
    - 6.63301e-07 H_y3u2w2 * H_y13u2w2 + 6.41343e-07 H_y3u2w2 * ...
        H_y15u2w2
    + 8.60218e-05 H_y3u2w2 * H_y16u2w2 + 0.0283431149930041 H_y4u2w2 ^2
    + 2.51339e-05 H_y4u2w2 * H_y5u2w2
    + 0.0693420227334188 H_y4u2w2 * H_y6u2w2
    - 1.3298069215577417e-04 H_y4u2w2 * H_y7u2w2
    + 0.3847118906703809 H_y4u2w2 * H_y8u2w2
    - 2.67693e-05 H_y4u2w2 * H_y9u2w2
    + 0.0606339992451147 H_y4u2w2 * H_y10u2w2
    + 3.2527239935002455e-04 H_y4u2w2 * H_y11u2w2
    + 0.078053131902459 H_y4u2w2 * H_y12u2w2
    - 3.5204173994488654e-04 H_y4u2w2 * H_y13u2w2
    + 3.5040632408146104e-04 H_y4u2w2 * H_y15u2w2
    + 0.0443100304573928 H_y4u2w2 * H_y16u2w2 + 1.60063e-06 H_y5u2w2 ^2
    - 8.6448934890581382e-04 H_y5u2w2 * H_y6u2w2
    + 2.76784e-07 H_y5u2w2 * H_y7u2w2
    - 1.1097248953901862e-04 H_y5u2w2 * H_y8u2w2
    + 3.06877e-07 H_y5u2w2 * H_y9u2w2 + 4.00665e-05 H_y5u2w2 * H_y10u2w2
    + 2.11846e-07 H_y5u2w2 * H_y11u2w2 + 1.69959e-05 H_y5u2w2 * ...
        H_y12u2w2
    + 9.50305e-08 H_y5u2w2 * H_y13u2w2 + 3.41156e-06 H_y5u2w2 * ...
        H_y15u2w2
    - 0.000482794 H_y5u2w2 * H_y16u2w2 + 0.168368709601l319 H_y6u2w2 ^2
    - 2.5758239563441348e-04 H_y6u2w2 * H_y7u2w2
    + 0.5514979461831324 H_y6u2w2 * H_y8u2w2
    - 0.000122378 H_y6u2w2 * H_y9u2w2
    + 0.0707589478282818 H_y6u2w2 * H_y10u2w2
    + 0.000380456 H_y6u2w2 * H_y11u2w2
    + 0.1007971916201678 H_y6u2w2 * H_y12u2w2
    - 5.0283386692081644e-04 H_y6u2w2 * H_y13u2w2
    - 4.8403338312081766e-04 H_y6u2w2 * H_y15u2w2
    + 0.1955001364558541 H_y6u2w2 * H_y16u2w2 + 1.75003e-07 H_y7u2w2 ^2
    - 9.3542279601113699e-04 H_y7u2w2 * H_y8u2w2
    + 9.66093e-08 H_y7u2w2 * H_y9u2w2
    - 1.4136654198423814e-04 H_y7u2w2 * H_y10u2w2
    - 7.5869e-07 H_y7u2w2 * H_y11u2w2
    - 1.8562234891509341e-04 H_y7u2w2 * H_y12u2w2
    + 8.55299e-07 H_y7u2w2 * H_y13u2w2 - 4.81906e-07 H_y7u2w2 * ...
        H_y15u2w2
    - 1.572614535664045e-04 H_y7u2w2 * H_y16u2w2
    + 1.322666291666435 H_y8u2w2 ^2 - 0.000210528 H_y8u2w2 * H_y9u2w2
    + 0.4118031201287415 H_y8u2w2 * H_y10u2w2
    + 0.0022094002926212 H_y8u2w2 * H_y11u2w2
    + 0.5331673263470015 H_y8u2w2 * H_y12u2w2
    - 0.00241992827644428 H_y8u2w2 * H_y13u2w2
    + 0.00209842780308302 H_y8u2w2 * H_y15u2w2
    + 0.3462502596962378 H_y8u2w2 * H_y16u2w2 + 2.30509e-08 H_y9u2w2 ^2
    - 2.74266e-05 H_y9u2w2 * H_y10u2w2 - 1.4744e-07 H_y9u2w2 * H_y11u2w2
    - 3.87632e-05 H_y9u2w2 * H_y12u2w2 + 1.91998e-07 H_y9u2w2 * ...
        H_y13u2w2
    + 1.59437e-07 H_y9u2w2 * H_y15u2w2 - 7.12397e-05 H_y9u2w2 * ...
        H_y16u2w2
    + 0.032670026112372 H_y10u2w2 ^2
    + 3.4943252974706013e-04 H_y10u2w2 * H_y11u2w2
    + 0.0837117645940744 H_y10u2w2 * H_y12u2w2
    - 3.7685910544789462e-04 H_y10u2w2 * H_y13u2w2
    + 0.000389499 H_y10u2w2 * H_y15u2w2
    + 0.0455012061103151 H_y10u2w2 * H_y16u2w2 + 9.38008e-07 ...
        H_y11u2w2 ^2
    + 4.4909076423386112e-04 H_y11u2w2 * H_y12u2w2
    - 2.02191e-06 H_y11u2w2 * H_y13u2w2 + 2.08632e-06 H_y11u2w2 * ...
        H_y15u2w2
    + 2.4457940120790401e-04 H_y11u2w2 * H_y16u2w2
    + 0.0540765445068836 H_y12u2w2 ^2 - 0.000487854 H_y12u2w2 * ...
        H_y13u2w2
    + 4.6608671065024329e-04 H_y12u2w2 * H_y15u2w2
    + 0.064022876007293 H_y12u2w2 * H_y16u2w2 + 1.10773e-06 ...
        H_y13u2w2 ^2
    - 1.92688e-06 H_y13u2w2 * H_y15u2w2 - 0.000315819 H_y13u2w2 * ...
        H_y16u2w2
    + 2.74971e-06 H_y15u2w2 ^2
    - 2.3821454053607547e-04 H_y15u2w2 * H_y16u2w2
    + 0.0570768797619619 H_y16u2w2 ^2 ] / 2
Subject To
 R0: 0.002 H_y1u1w1 - 0.5090991037076109 H_y2u1w1 - 0.002 H_y3u1w1
    - 0.1489493216895732 H_y4u1w1 + 0.00181598 H_y5u1w1
    - 0.4888618616206708 H_y6u1w1 + 0.00018402 H_y7u1w1
```

```
  - 0.7088079771181977 H_y8u1w1 - 0.00181598 H_y9u1w1
  - 0.1523008720973329 H_y10u1w1 - 0.00100428881865002 H_y11u1w1
  + 0.0663214671335344 H_y12u1w1 - 8.11691181473468e-04 H_y13u1w1
  + 8.1169118131409028e-04 H_y15u1w1 - 0.1498406426914123 H_y16u1w1
  + 0.002 H_y1u1w2 - 0.5090991037076109 H_y2u1w2 - 0.002 H_y3u1w2
  - 0.1489493216895732 H_y4u1w2 + 0.00181598 H_y5u1w2
  - 0.4888618616206708 H_y6u1w2 + 0.00018402 H_y7u1w2
  - 0.7088079771181977 H_y8u1w2 - 0.00181598 H_y9u1w2
  - 0.1523008720973329 H_y10u1w2 - 0.00100428881865002 H_y11u1w2
  + 0.0663214671335344 H_y12u1w2 - 8.11691181473468e-04 H_y13u1w2
  + 8.1169118131409028e-04 H_y15u1w2 - 0.1498406426914123 H_y16u1w2
  = 37.87087872548276
R1: 0.00181708916666677 H_y5u1w1 - 0.4773620917672417 H_y6u1w1
  - 0.00181708916666677 H_y7u1w1 - 0.0843350702927625 H_y8u1w1
  - 0.00181708916666677 H_y9u1w1 - 0.1578454846379168 H_y10u1w1
  - 0.00165995414535771 H_y11u1w1 + 0.2536437430674665 H_y12u1w1
  - 0.000157135 H_y13u1w1 + 0.000157135 H_y15u1w1
  - 0.0255172204707832 H_y16u1w1 + 0.00181708916666677 H_y5u1w2
  - 0.4773620917672417 H_y6u1w2 - 0.00181708916666677 H_y7u1w2
  - 0.0843350702927625 H_y8u1w2 - 0.00181708916666677 H_y9u1w2
  - 0.1578454846379168 H_y10u1w2 - 0.00165995414535771 H_y11u1w2
  + 0.2536437430674665 H_y12u1w2 - 0.000157135 H_y13u1w2
  + 0.000157135 H_y15u1w2 - 0.0255172204707832 H_y16u1w2
  = 21.69049292750346
R2: 0.002 H_y1u2w1 - 0.5090991037076109 H_y2u2w1 - 0.002 H_y3u2w1
  - 0.1489493216895732 H_y4u2w1 + 0.00181598 H_y5u2w1
  - 0.4888618616206708 H_y6u2w1 + 0.00018402 H_y7u2w1
  - 0.7088079771181977 H_y8u2w1 - 0.00181598 H_y9u2w1
  - 0.1523008720973329 H_y10u2w1 - 0.00100428881865002 H_y11u2w1
  + 0.0663214671335344 H_y12u2w1 - 8.11691181473468e-04 H_y13u2w1
  + 8.1169118131409028e-04 H_y15u2w1 - 0.1498406426914123 H_y16u2w1
  + 0.002 H_y1u2w2 - 0.5090991037076109 H_y2u2w2 - 0.002 H_y3u2w2
  - 0.1489493216895732 H_y4u2w2 + 0.00181598 H_y5u2w2
  - 0.4888618616206708 H_y6u2w2 + 0.00018402 H_y7u2w2
  - 0.7088079771181977 H_y8u2w2 - 0.00181598 H_y9u2w2
  - 0.1523008720973329 H_y10u2w2 - 0.00100428881865002 H_y11u2w2
  + 0.0663214671335344 H_y12u2w2 - 8.11691181473468e-04 H_y13u2w2
  + 8.1169118131409028e-04 H_y15u2w2 - 0.1498406426914123 H_y16u2w2
  = 21.69049292750347
R3: 0.00181708916666677 H_y5u2w1 - 0.4773620917672417 H_y6u2w1
  - 0.00181708916666677 H_y7u2w1 - 0.0843350702927625 H_y8u2w1
  - 0.00181708916666677 H_y9u2w1 - 0.1578454846379168 H_y10u2w1
  - 0.00165995414535771 H_y11u2w1 + 0.2536437430674665 H_y12u2w1
  - 0.000157135 H_y13u2w1 + 0.000157135 H_y15u2w1
  - 0.0255172204707832 H_y16u2w1 + 0.00181708916666677 H_y5u2w2
  - 0.4773620917672417 H_y6u2w2 - 0.00181708916666677 H_y7u2w2
  - 0.0843350702927625 H_y8u2w2 - 0.00181708916666677 H_y9u2w2
  - 0.1578454846379168 H_y10u2w2 - 0.00165995414535771 H_y11u2w2
  + 0.2536437430674665 H_y12u2w2 - 0.000157135 H_y13u2w2
  + 0.000157135 H_y15u2w2 - 0.0255172204707832 H_y16u2w2
  = 66.50518021113163
R4: sigma_y1w1 + sigma_y2w1 + sigma_y3w1 + sigma_y4w1 + sigma_y5w1
  + sigma_y6w1 + sigma_y7w1 + sigma_y8w1 + sigma_y9w1 + sigma_y10w1
  + sigma_y11w1 + sigma_y12w1 + sigma_y13w1 + sigma_y15w1 + ...
      sigma_y16w1
  + sigma_y1w2 + sigma_y2w2 + sigma_y3w2 + sigma_y4w2 + sigma_y5w2
  + sigma_y6w2 + sigma_y7w2 + sigma_y8w2 + sigma_y9w2 + sigma_y10w2
  + sigma_y11w2 + sigma_y12w2 + sigma_y13w2 + sigma_y15w2 + ...
      sigma_y16w2
  - nm = 0
R5: - H_y1u1w1 - 1e+06 sigma_y1w1 <= 0
R6: - H_y2u1w1 - 1e+06 sigma_y2w1 <= 0
R7: - H_y3u1w1 - 1e+06 sigma_y3w1 <= 0
R8: - H_y4u1w1 - 1e+06 sigma_y4w1 <= 0
R9: - H_y5u1w1 - 1e+06 sigma_y5w1 <= 0
R10: - H_y6u1w1 - 1e+06 sigma_y6w1 <= 0
R11: - H_y7u1w1 - 1e+06 sigma_y7w1 <= 0
R12: - H_y8u1w1 - 1e+06 sigma_y8w1 <= 0
R13: - H_y9u1w1 - 1e+06 sigma_y9w1 <= 0
R14: - H_y10u1w1 - 1e+06 sigma_y10w1 <= 0
R15: - H_y11u1w1 - 1e+06 sigma_y11w1 <= 0
R16: - H_y12u1w1 - 1e+06 sigma_y12w1 <= 0
R17: - H_y13u1w1 - 1e+06 sigma_y13w1 <= 0
R18: - H_y15u1w1 - 1e+06 sigma_y15w1 <= 0
R19: - H_y16u1w1 - 1e+06 sigma_y16w1 <= 0
R20: - H_y1u2w1 - 1e+06 sigma_y1w1 <= 0
R21: - H_y2u2w1 - 1e+06 sigma_y2w1 <= 0
R22: - H_y3u2w1 - 1e+06 sigma_y3w1 <= 0
R23: - H_y4u2w1 - 1e+06 sigma_y4w1 <= 0
R24: - H_y5u2w1 - 1e+06 sigma_y5w1 <= 0
R25: - H_y6u2w1 - 1e+06 sigma_y6w1 <= 0
R26: - H_y7u2w1 - 1e+06 sigma_y7w1 <= 0
R27: - H_y8u2w1 - 1e+06 sigma_y8w1 <= 0
R28: - H_y9u2w1 - 1e+06 sigma_y9w1 <= 0
R29: - H_y10u2w1 - 1e+06 sigma_y10w1 <= 0
R30: - H_y11u2w1 - 1e+06 sigma_y11w1 <= 0
R31: - H_y12u2w1 - 1e+06 sigma_y12w1 <= 0
R32: - H_y13u2w1 - 1e+06 sigma_y13w1 <= 0
R33: - H_y15u2w1 - 1e+06 sigma_y15w1 <= 0
R34: - H_y16u2w1 - 1e+06 sigma_y16w1 <= 0
R35: - H_y1u1w2 - 1e+06 sigma_y1w2 <= 0
R36: - H_y2u1w2 - 1e+06 sigma_y2w2 <= 0
R37: - H_y3u1w2 - 1e+06 sigma_y3w2 <= 0
R38: - H_y4u1w2 - 1e+06 sigma_y4w2 <= 0
R39: - H_y5u1w2 - 1e+06 sigma_y5w2 <= 0
R40: - H_y6u1w2 - 1e+06 sigma_y6w2 <= 0
R41: - H_y7u1w2 - 1e+06 sigma_y7w2 <= 0
R42: - H_y8u1w2 - 1e+06 sigma_y8w2 <= 0
R43: - H_y9u1w2 - 1e+06 sigma_y9w2 <= 0
R44: - H_y10u1w2 - 1e+06 sigma_y10w2 <= 0
R45: - H_y11u1w2 - 1e+06 sigma_y11w2 <= 0
R46: - H_y12u1w2 - 1e+06 sigma_y12w2 <= 0
R47: - H_y13u1w2 - 1e+06 sigma_y13w2 <= 0
R48: - H_y15u1w2 - 1e+06 sigma_y15w2 <= 0
R49: - H_y16u1w2 - 1e+06 sigma_y16w2 <= 0
R50: - H_y1u2w2 - 1e+06 sigma_y1w2 <= 0
R51: - H_y2u2w2 - 1e+06 sigma_y2w2 <= 0
R52: - H_y3u2w2 - 1e+06 sigma_y3w2 <= 0
R53: - H_y4u2w2 - 1e+06 sigma_y4w2 <= 0
R54: - H_y5u2w2 - 1e+06 sigma_y5w2 <= 0
R55: - H_y6u2w2 - 1e+06 sigma_y6w2 <= 0
R56: - H_y7u2w2 - 1e+06 sigma_y7w2 <= 0
R57: - H_y8u2w2 - 1e+06 sigma_y8w2 <= 0
R58: - H_y9u2w2 - 1e+06 sigma_y9w2 <= 0
```

```
R59: - H_y10u2w2 - 1e+06 sigma_y10w2 <= 0
R60: - H_y11u2w2 - 1e+06 sigma_y11w2 <= 0
R61: - H_y12u2w2 - 1e+06 sigma_y12w2 <= 0
R62: - H_y13u2w2 - 1e+06 sigma_y13w2 <= 0
R63: - H_y15u2w2 - 1e+06 sigma_y15w2 <= 0
R64: - H_y16u2w2 - 1e+06 sigma_y16w2 <= 0
R65: H_y1u1w1 - 1e+06 sigma_y1w1 <= 0
R66: H_y2u1w1 - 1e+06 sigma_y2w1 <= 0
R67: H_y3u1w1 - 1e+06 sigma_y3w1 <= 0
R68: H_y4u1w1 - 1e+06 sigma_y4w1 <= 0
R69: H_y5u1w1 - 1e+06 sigma_y5w1 <= 0
R70: H_y6u1w1 - 1e+06 sigma_y6w1 <= 0
R71: H_y7u1w1 - 1e+06 sigma_y7w1 <= 0
R72: H_y8u1w1 - 1e+06 sigma_y8w1 <= 0
R73: H_y9u1w1 - 1e+06 sigma_y9w1 <= 0
R74: H_y10u1w1 - 1e+06 sigma_y10w1 <= 0
R75: H_y11u1w1 - 1e+06 sigma_y11w1 <= 0
R76: H_y12u1w1 - 1e+06 sigma_y12w1 <= 0
R77: H_y13u1w1 - 1e+06 sigma_y13w1 <= 0
R78: H_y15u1w1 - 1e+06 sigma_y15w1 <= 0
R79: H_y16u1w1 - 1e+06 sigma_y16w1 <= 0
R80: H_y1u2w1 - 1e+06 sigma_y1w1 <= 0
R81: H_y2u2w1 - 1e+06 sigma_y2w1 <= 0
R82: H_y3u2w1 - 1e+06 sigma_y3w1 <= 0
R83: H_y4u2w1 - 1e+06 sigma_y4w1 <= 0
R84: H_y5u2w1 - 1e+06 sigma_y5w1 <= 0
R85: H_y6u2w1 - 1e+06 sigma_y6w1 <= 0
R86: H_y7u2w1 - 1e+06 sigma_y7w1 <= 0
R87: H_y8u2w1 - 1e+06 sigma_y8w1 <= 0
R88: H_y9u2w1 - 1e+06 sigma_y9w1 <= 0
R89: H_y10u2w1 - 1e+06 sigma_y10w1 <= 0
R90: H_y11u2w1 - 1e+06 sigma_y11w1 <= 0
R91: H_y12u2w1 - 1e+06 sigma_y12w1 <= 0
R92: H_y13u2w1 - 1e+06 sigma_y13w1 <= 0
R93: H_y15u2w1 - 1e+06 sigma_y15w1 <= 0
R94: H_y16u2w1 - 1e+06 sigma_y16w1 <= 0
R95: H_y1u1w2 - 1e+06 sigma_y1w2 <= 0
R96: H_y2u1w2 - 1e+06 sigma_y2w2 <= 0
R97: H_y3u1w2 - 1e+06 sigma_y3w2 <= 0
R98: H_y4u1w2 - 1e+06 sigma_y4w2 <= 0
R99: H_y5u1w2 - 1e+06 sigma_y5w2 <= 0
R100: H_y6u1w2 - 1e+06 sigma_y6w2 <= 0
R101: H_y7u1w2 - 1e+06 sigma_y7w2 <= 0
R102: H_y8u1w2 - 1e+06 sigma_y8w2 <= 0
R103: H_y9u1w2 - 1e+06 sigma_y9w2 <= 0
R104: H_y10u1w2 - 1e+06 sigma_y10w2 <= 0
R105: H_y11u1w2 - 1e+06 sigma_y11w2 <= 0
R106: H_y12u1w2 - 1e+06 sigma_y12w2 <= 0
R107: H_y13u1w2 - 1e+06 sigma_y13w2 <= 0
R108: H_y15u1w2 - 1e+06 sigma_y15w2 <= 0
R109: H_y16u1w2 - 1e+06 sigma_y16w2 <= 0
R110: H_y1u2w2 - 1e+06 sigma_y1w2 <= 0
R111: H_y2u2w2 - 1e+06 sigma_y2w2 <= 0
R112: H_y3u2w2 - 1e+06 sigma_y3w2 <= 0
R113: H_y4u2w2 - 1e+06 sigma_y4w2 <= 0
R114: H_y5u2w2 - 1e+06 sigma_y5w2 <= 0
R115: H_y6u2w2 - 1e+06 sigma_y6w2 <= 0
```

```
R116: H_y7u2w2 - 1e+06 sigma_y7w2 <= 0
R117: H_y8u2w2 - 1e+06 sigma_y8w2 <= 0
R118: H_y9u2w2 - 1e+06 sigma_y9w2 <= 0
R119: H_y10u2w2 - 1e+06 sigma_y10w2 <= 0
R120: H_y11u2w2 - 1e+06 sigma_y11w2 <= 0
R121: H_y12u2w2 - 1e+06 sigma_y12w2 <= 0
R122: H_y13u2w2 - 1e+06 sigma_y13w2 <= 0
R123: H_y15u2w2 - 1e+06 sigma_y15w2 <= 0
R124: H_y16u2w2 - 1e+06 sigma_y16w2 <= 0
R125: nm = 15
R126: H_y1u1w1 + 1e+06 sigma_y1w2 <= 1e+06
R127: H_y2u1w1 + 1e+06 sigma_y2w2 <= 1e+06
R128: H_y3u1w1 + 1e+06 sigma_y3w2 <= 1e+06
R129: H_y4u1w1 + 1e+06 sigma_y4w2 <= 1e+06
R130: H_y5u1w1 + 1e+06 sigma_y5w2 <= 1e+06
R131: H_y6u1w1 + 1e+06 sigma_y6w2 <= 1e+06
R132: H_y7u1w1 + 1e+06 sigma_y7w2 <= 1e+06
R133: H_y8u1w1 + 1e+06 sigma_y8w2 <= 1e+06
R134: H_y9u1w1 + 1e+06 sigma_y9w2 <= 1e+06
R135: H_y10u1w1 + 1e+06 sigma_y10w2 <= 1e+06
R136: H_y11u1w1 + 1e+06 sigma_y11w2 <= 1e+06
R137: H_y12u1w1 + 1e+06 sigma_y12w2 <= 1e+06
R138: H_y13u1w1 + 1e+06 sigma_y13w2 <= 1e+06
R139: H_y15u1w1 + 1e+06 sigma_y15w2 <= 1e+06
R140: H_y16u1w1 + 1e+06 sigma_y16w2 <= 1e+06
R141: H_y1u2w1 + 1e+06 sigma_y1w2 <= 1e+06
R142: H_y2u2w1 + 1e+06 sigma_y2w2 <= 1e+06
R143: H_y3u2w1 + 1e+06 sigma_y3w2 <= 1e+06
R144: H_y4u2w1 + 1e+06 sigma_y4w2 <= 1e+06
R145: H_y5u2w1 + 1e+06 sigma_y5w2 <= 1e+06
R146: H_y6u2w1 + 1e+06 sigma_y6w2 <= 1e+06
R147: H_y7u2w1 + 1e+06 sigma_y7w2 <= 1e+06
R148: H_y8u2w1 + 1e+06 sigma_y8w2 <= 1e+06
R149: H_y9u2w1 + 1e+06 sigma_y9w2 <= 1e+06
R150: H_y10u2w1 + 1e+06 sigma_y10w2 <= 1e+06
R151: H_y11u2w1 + 1e+06 sigma_y11w2 <= 1e+06
R152: H_y12u2w1 + 1e+06 sigma_y12w2 <= 1e+06
R153: H_y13u2w1 + 1e+06 sigma_y13w2 <= 1e+06
R154: H_y15u2w1 + 1e+06 sigma_y15w2 <= 1e+06
R155: H_y16u2w1 + 1e+06 sigma_y16w2 <= 1e+06
R156: H_y1u1w2 + 1e+06 sigma_y1w1 <= 1e+06
R157: H_y2u1w2 + 1e+06 sigma_y2w1 <= 1e+06
R158: H_y3u1w2 + 1e+06 sigma_y3w1 <= 1e+06
R159: H_y4u1w2 + 1e+06 sigma_y4w1 <= 1e+06
R160: H_y5u1w2 + 1e+06 sigma_y5w1 <= 1e+06
R161: H_y6u1w2 + 1e+06 sigma_y6w1 <= 1e+06
R162: H_y7u1w2 + 1e+06 sigma_y7w1 <= 1e+06
R163: H_y8u1w2 + 1e+06 sigma_y8w1 <= 1e+06
R164: H_y9u1w2 + 1e+06 sigma_y9w1 <= 1e+06
R165: H_y10u1w2 + 1e+06 sigma_y10w1 <= 1e+06
R166: H_y11u1w2 + 1e+06 sigma_y11w1 <= 1e+06
R167: H_y12u1w2 + 1e+06 sigma_y12w1 <= 1e+06
R168: H_y13u1w2 + 1e+06 sigma_y13w1 <= 1e+06
R169: H_y15u1w2 + 1e+06 sigma_y15w1 <= 1e+06
R170: H_y16u1w2 + 1e+06 sigma_y16w1 <= 1e+06
R171: H_y1u2w2 + 1e+06 sigma_y1w1 <= 1e+06
R172: H_y2u2w2 + 1e+06 sigma_y2w1 <= 1e+06
```

```
R173: H_y3u2w2 + 1e+06 sigma_y3w1 <= 1e+06
R174: H_y4u2w2 + 1e+06 sigma_y4w1 <= 1e+06
R175: H_y5u2w2 + 1e+06 sigma_y5w1 <= 1e+06
R176: H_y6u2w2 + 1e+06 sigma_y6w1 <= 1e+06
R177: H_y7u2w2 + 1e+06 sigma_y7w1 <= 1e+06
R178: H_y8u2w2 + 1e+06 sigma_y8w1 <= 1e+06
R179: H_y9u2w2 + 1e+06 sigma_y9w1 <= 1e+06
R180: H_y10u2w2 + 1e+06 sigma_y10w1 <= 1e+06
R181: H_y11u2w2 + 1e+06 sigma_y11w1 <= 1e+06
R182: H_y12u2w2 + 1e+06 sigma_y12w1 <= 1e+06
R183: H_y13u2w2 + 1e+06 sigma_y13w1 <= 1e+06
R184: H_y15u2w2 + 1e+06 sigma_y15w1 <= 1e+06
R185: H_y16u2w2 + 1e+06 sigma_y16w1 <= 1e+06
R186: - H_y1u1w1 + 1e+06 sigma_y1w2 <= 1e+06
R187: - H_y2u1w1 + 1e+06 sigma_y2w2 <= 1e+06
R188: - H_y3u1w1 + 1e+06 sigma_y3w2 <= 1e+06
R189: - H_y4u1w1 + 1e+06 sigma_y4w2 <= 1e+06
R190: - H_y5u1w1 + 1e+06 sigma_y5w2 <= 1e+06
R191: - H_y6u1w1 + 1e+06 sigma_y6w2 <= 1e+06
R192: - H_y7u1w1 + 1e+06 sigma_y7w2 <= 1e+06
R193: - H_y8u1w1 + 1e+06 sigma_y8w2 <= 1e+06
R194: - H_y9u1w1 + 1e+06 sigma_y9w2 <= 1e+06
R195: - H_y10u1w1 + 1e+06 sigma_y10w2 <= 1e+06
R196: - H_y11u1w1 + 1e+06 sigma_y11w2 <= 1e+06
R197: - H_y12u1w1 + 1e+06 sigma_y12w2 <= 1e+06
R198: - H_y13u1w1 + 1e+06 sigma_y13w2 <= 1e+06
R199: - H_y15u1w1 + 1e+06 sigma_y15w2 <= 1e+06
R200: - H_y16u1w1 + 1e+06 sigma_y16w2 <= 1e+06
R201: - H_y1u2w1 + 1e+06 sigma_y1w2 <= 1e+06
R202: - H_y2u2w1 + 1e+06 sigma_y2w2 <= 1e+06
R203: - H_y3u2w1 + 1e+06 sigma_y3w2 <= 1e+06
R204: - H_y4u2w1 + 1e+06 sigma_y4w2 <= 1e+06
R205: - H_y5u2w1 + 1e+06 sigma_y5w2 <= 1e+06
R206: - H_y6u2w1 + 1e+06 sigma_y6w2 <= 1e+06
R207: - H_y7u2w1 + 1e+06 sigma_y7w2 <= 1e+06
R208: - H_y8u2w1 + 1e+06 sigma_y8w2 <= 1e+06
R209: - H_y9u2w1 + 1e+06 sigma_y9w2 <= 1e+06
R210: - H_y10u2w1 + 1e+06 sigma_y10w2 <= 1e+06
R211: - H_y11u2w1 + 1e+06 sigma_y11w2 <= 1e+06
R212: - H_y12u2w1 + 1e+06 sigma_y12w2 <= 1e+06
R213: - H_y13u2w1 + 1e+06 sigma_y13w2 <= 1e+06
R214: - H_y15u2w1 + 1e+06 sigma_y15w2 <= 1e+06
R215: - H_y16u2w1 + 1e+06 sigma_y16w2 <= 1e+06
R216: - H_y1u1w2 + 1e+06 sigma_y1w1 <= 1e+06
R217: - H_y2u1w2 + 1e+06 sigma_y2w1 <= 1e+06
R218: - H_y3u1w2 + 1e+06 sigma_y3w1 <= 1e+06
R219: - H_y4u1w2 + 1e+06 sigma_y4w1 <= 1e+06
R220: - H_y5u1w2 + 1e+06 sigma_y5w1 <= 1e+06
R221: - H_y6u1w2 + 1e+06 sigma_y6w1 <= 1e+06
R222: - H_y7u1w2 + 1e+06 sigma_y7w1 <= 1e+06
R223: - H_y8u1w2 + 1e+06 sigma_y8w1 <= 1e+06
R224: - H_y9u1w2 + 1e+06 sigma_y9w1 <= 1e+06
R225: - H_y10u1w2 + 1e+06 sigma_y10w1 <= 1e+06
R226: - H_y11u1w2 + 1e+06 sigma_y11w1 <= 1e+06
R227: - H_y12u1w2 + 1e+06 sigma_y12w1 <= 1e+06
R228: - H_y13u1w2 + 1e+06 sigma_y13w1 <= 1e+06
R229: - H_y15u1w2 + 1e+06 sigma_y15w1 <= 1e+06
```

```
 R230: - H_y16u1w2 + 1e+06 sigma_y16w1 <= 1e+06
 R231: - H_y1u2w2 + 1e+06 sigma_y1w1 <= 1e+06
 R232: - H_y2u2w2 + 1e+06 sigma_y2w1 <= 1e+06
 R233: - H_y3u2w2 + 1e+06 sigma_y3w1 <= 1e+06
 R234: - H_y4u2w2 + 1e+06 sigma_y4w1 <= 1e+06
 R235: - H_y5u2w2 + 1e+06 sigma_y5w1 <= 1e+06
 R236: - H_y6u2w2 + 1e+06 sigma_y6w1 <= 1e+06
 R237: - H_y7u2w2 + 1e+06 sigma_y7w1 <= 1e+06
 R238: - H_y8u2w2 + 1e+06 sigma_y8w1 <= 1e+06
 R239: - H_y9u2w2 + 1e+06 sigma_y9w1 <= 1e+06
 R240: - H_y10u2w2 + 1e+06 sigma_y10w1 <= 1e+06
 R241: - H_y11u2w2 + 1e+06 sigma_y11w1 <= 1e+06
 R242: - H_y12u2w2 + 1e+06 sigma_y12w1 <= 1e+06
 R243: - H_y13u2w2 + 1e+06 sigma_y13w1 <= 1e+06
 R244: - H_y15u2w2 + 1e+06 sigma_y15w1 <= 1e+06
 R245: - H_y16u2w2 + 1e+06 sigma_y16w1 <= 1e+06
 R246: H_y14u3w1 + H_y14u3w2 = 1
 R247: sigma_y14w1 + sigma_y14w2 = 1
 R248: 0.1 sigma_y1w1 + sigma_y2w1 + 0.1 sigma_y3w1 + sigma_y4w1
   + 0.1 sigma_y5w1 + sigma_y6w1 + 0.1 sigma_y7w1 + sigma_y8w1
   + 0.1 sigma_y9w1 + sigma_y10w1 + 0.1 sigma_y11w1 + sigma_y12w1
   + 0.1 sigma_y13w1 + 0.1 sigma_y15w1 + sigma_y16w1 + 0.05 sigma_y1w2
   + 0.1 sigma_y2w2 + 0.05 sigma_y3w2 + 0.1 sigma_y4w2 + 0.05 ...
       sigma_y5w2
   + 0.1 sigma_y6w2 + 0.05 sigma_y7w2 + 0.1 sigma_y8w2 + 0.05 ...
       sigma_y9w2
   + 0.1 sigma_y10w2 + 0.05 sigma_y11w2 + 0.1 sigma_y12w2
   + 0.05 sigma_y13w2 + 0.05 sigma_y15w2 + 0.1 sigma_y16w2
   + 0.5 sigma_y14w1 + 0.05 sigma_y14w2 <= 1e+06
Bounds
 H_y1u1w1 free
 H_y2u1w1 free
 H_y3u1w1 free
 H_y4u1w1 free
 H_y5u1w1 free
 H_y6u1w1 free
 H_y7u1w1 free
 H_y8u1w1 free
 H_y9u1w1 free
 H_y10u1w1 free
 H_y11u1w1 free
 H_y12u1w1 free
 H_y13u1w1 free
 H_y15u1w1 free
 H_y16u1w1 free
 H_y1u2w1 free
 H_y2u2w1 free
 H_y3u2w1 free
 H_y4u2w1 free
 H_y5u2w1 free
 H_y6u2w1 free
 H_y7u2w1 free
 H_y8u2w1 free
 H_y9u2w1 free
 H_y10u2w1 free
 H_y11u2w1 free
 H_y12u2w1 free
 H_y13u2w1 free
 H_y15u2w1 free
 H_y16u2w1 free
 H_y1u1w2 free
 H_y2u1w2 free
 H_y3u1w2 free
 H_y4u1w2 free
 H_y5u1w2 free
 H_y6u1w2 free
 H_y7u1w2 free
 H_y8u1w2 free
 H_y9u1w2 free
 H_y10u1w2 free
 H_y11u1w2 free
 H_y12u1w2 free
 H_y13u1w2 free
 H_y15u1w2 free
 H_y16u1w2 free
 H_y1u2w2 free
 H_y2u2w2 free
 H_y3u2w2 free
 H_y4u2w2 free
 H_y5u2w2 free
 H_y6u2w2 free
 H_y7u2w2 free
 H_y8u2w2 free
 H_y9u2w2 free
 H_y10u2w2 free
 H_y11u2w2 free
 H_y12u2w2 free
 H_y13u2w2 free
 H_y15u2w2 free
 H_y16u2w2 free
 nm <= 15
Binaries
 sigma_y1w1 sigma_y2w1 sigma_y3w1 sigma_y4w1 sigma_y5w1 sigma_y6w1
 sigma_y7w1 sigma_y8w1 sigma_y9w1 sigma_y10w1 sigma_y11w1 sigma_y12w1
 sigma_y13w1 sigma_y15w1 sigma_y16w1 sigma_y1w2 sigma_y2w2 sigma_y3w2
 sigma_y4w2 sigma_y5w2 sigma_y6w2 sigma_y7w2 sigma_y8w2 sigma_y9w2
 sigma_y10w2 sigma_y11w2 sigma_y12w2 sigma_y13w2 sigma_y15w2 ...
     sigma_y16w2
 sigma_y14w1 sigma_y14w2 H_y14u3w1 H_y14u3w2
Generals
 nm
SOS
 s0:  S1 :: sigma_y1w1:1 sigma_y1w2:1
 s1:  S1 :: sigma_y2w1:1 sigma_y2w2:1
 s2:  S1 :: sigma_y3w1:1 sigma_y3w2:1
 s3:  S1 :: sigma_y4w1:1 sigma_y4w2:1
 s4:  S1 :: sigma_y5w1:1 sigma_y5w2:1
 s5:  S1 :: sigma_y6w1:1 sigma_y6w2:1
 s6:  S1 :: sigma_y7w1:1 sigma_y7w2:1
 s7:  S1 :: sigma_y8w1:1 sigma_y8w2:1
 s8:  S1 :: sigma_y9w1:1 sigma_y9w2:1
 s9:  S1 :: sigma_y10w1:1 sigma_y10w2:1
 s10:  S1 :: sigma_y11w1:1 sigma_y11w2:1
 s11:  S1 :: sigma_y12w1:1 sigma_y12w2:1
 s12:  S1 :: sigma_y13w1:1 sigma_y13w2:1
 s13:  S1 :: sigma_y15w1:1 sigma_y15w2:1
 s14:  S1 :: sigma_y16w1:1 sigma_y16w2:1
End
```

Appendix C

Finite Difference Method

MATLAB's `fmincon` solver could in theory provide Hessian matrices. However, these are based on Hessian of the Lagrangian [28] and is therefore not a good representation of the Hessian of the cost function based on inputs. Due to this, finite differences are used to calculate the Hessian. The finite difference method is a method for approximation of derivatives. Finite differences can be given as forward differences, backward differences and central differences. In this work, only central differences was used[29]. Equations are given in (C.1) to (C.5). An illustration showing how each step relate to each other are presented in Figure C.1.

$$f_{u1}(u_1,u_2) \approx \frac{f(u_1+h,u_2)-f(u_1-h,u_2)}{2h} \tag{C.1}$$

$$f_{u2}(u_1,u_2) \approx \frac{f(u_1,u_2+k)-f(u_1,u_2-k)}{2h} \tag{C.2}$$

$$f_{u2u2}(u_1,u_2) \approx \frac{f(u_1,u_2+k)-2f(u_1,u_2)+f(u_1,u_2-k)}{k^2} \tag{C.3}$$

$$f_{u1u1}(u_1,u_2) \approx \frac{f(u_1+h,u_2)-2f(u_1,u_2)+f(u_1-h,u_2)}{h^2} \tag{C.4}$$

$$f_{u1u2}(u_1,u_2) \approx \frac{f(u_1+h,u_2+k)-f(u_1+h,u_2-k)-f(u_1-h,u_2+k)+f(u_1-h,u_2-k)}{4hk} \tag{C.5}$$

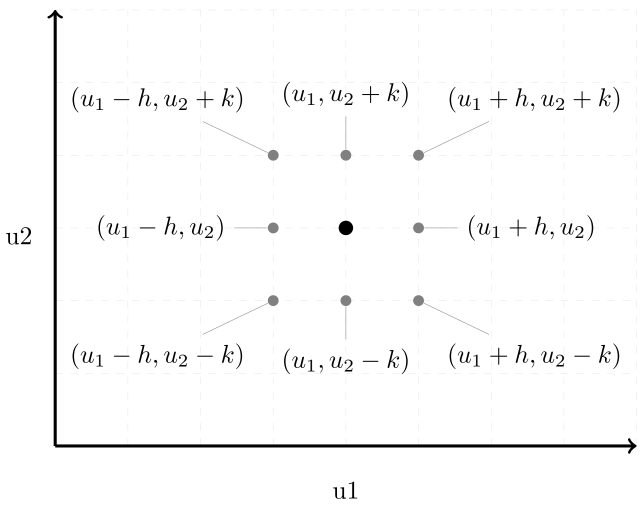

Figure C.1: Illustriation of a grid for finite differences