NTNU – Trondheim
Norwegian University of
Science and Technology

Finding Self-optimizing Control Variables Using Process Data

Rannei Solbak Simonsen

Chemical Engineering and Biotechnology
Submission date: June 2014
Supervisor: Sigurd Skogestad, IKP
Co-supervisor: Johannes Jäschke, IKP

Norwegian University of Science and Technology
Department of Chemical Engineering

Preface

I would like to thank Sigurd Skogestad for the project assignment and guidance throughout the project. I would also like to thank Johannes Jäschke for the excellent help and guidance in this project, for always being available and patient.

It is a strange feeling to be writing the first, yet final words of a master thesis. Throughout this project I have learned incredible much, and in the end I feel proud to finish my academic carrier in this way. This project has been challenging in many ways. For me it involved learning a new simulation program called ChemCad, and even though I did not use this program in the end I am thankful for the possibility to learn how to use it. In this context I would like to thank the eminent support team in Nor Par (license holders of ChemCad), and especially Stefan Mikulski for patiently answering my e-mails regarding ChemCad and the excellent introduction course on how to use the program.

The task of studying a rather straight forward, step-by-step method might seem simple enough. This was also my first thought when I chose this project as my master thesis. However, it turned out to be a rather complex assignment, involving evaluation of a complicated mathematical tool, modeling, and evaluating model accuracy. As well as testing different aspects of a newly developed method, where little research had been done before me. All the work has been done in MATLAB, which before this semester was a program I was only vaguely familiar with. For me it has been especially exiting to see how much my programming skills have improved during this project. For this, I would like to thank Vladimiros L. Minasidis, Chriss Gimholt and again Johannes Jäschke for outstanding help and support in MATLAB, without their help this thesis would never be done.

My good friends Silje, Ingrid, Evaldo, Stian, Ambari and Lisbet deserve all the thanks in the world. For always, in their own way, helping me in my work and offering comforting words when needed so that I would feel happy and motivated. I am truly lucky to have such great friends.

And last but not least, I would like to thank my parents for all their love, motivating words and never ending support.

Declaration of Compliance

I declare that this is an independent work according to the exam regulations of the Norwegian University of Science and Technology (NTNU).

Place and date :

Signature:

.....
Rannei Solbak Simonsen

Abstract

In this project we have studied a newly developed way to find self-optimizing variables. The method studied in this thesis uses data measurements, y, to fit a quadratic cost function. By using parameters found in the cost function modeling we can identify a combination matrix H. The H-matrix gives a linear combination of measurement, c=Hy, which can be used in a feedback control structure. The control variable, c, is said to be self-optimizing when, kept at a constant set-point, the operation of the process is close to optimal operation even when it is exposed to disturbances. Self-optimizing control structures are beneficial because they remove the need for re-optimization of a process after disturbances occur. The data-based method uses only historical measurements and the easily obtainable measurement gain. It does not require extensive knowledge found from complicated experiments or a well defined description of the process such as a process model. This is a great advantage with this method compared to more established methods such as the exact local and null space method.

The data-based method for finding self-optimizing variables is a rather newly developed method, and little was therefore known before. This thesis present preliminary research on different aspects of the data-based method. In order to investigate the method we applied it to three different test cases: a dummy case, an evaporator process and a CSTR and distillation column connected with a recycle process. Through research of the method using these test cases, we found some indicative trends regarding factors affecting the modeling of the cost function and estimation of the H-matrix. Additionally, we found that this method can in some cases be a good alternative to for example the exact local method. However more research is needed to be able to understand the factors affecting the outcome from the data-based method.

We hope that the work presented here will inspire to future research on this promising method to find self-optimizing variables.

Sammendrag

I denne avhandlingen har vi studert en nyutviklet måte å finne selvoptimaliserende variabler. Metoden bruker historiske data målinger, y, til å modelere en kvadratisk kostfunksjon. Ved å bruke parametre funnet fra modeleringen av kostfunsjonen, kan vi identifisere en optimal komibnasjon av tilgjengelige variabler. Denne optimale lineære kombinasjonen blir gitt av matrisen H, og resulterer i en kontrol variabel c=Hy, som kan brukes i regulering med tilbakekobling. Kontrollvaraibelen c, sies å være selvoptimaliserende dersom den ved å holdes konstant sikter nærmest optimal drift av prosessen selv når det skjer forstyrrelser. Selvoptimaliserende kontrollstrukturer er fordelaktige fordi man med det fjerner behovet for reoptimalisering av en prosess når forstyrrelser oppstår.

Fordelen med den databaserte metoden er at den ikke krever omfattende kunnskap funnet fra kompliserte eksperimenter, eller en veldefinert beskrivelse av prosessen, som for eksempel en prosessmodell. For å bruke data-metoden behøvs bare historiske målinger og prosessforsterkningen. Dette er en stor fordel med denne metoden i forhold til etablerte metoder som exact local- og null space metoden.

Fremgangsmåten presentert i avhandlingen, for å finne selvoptimaliserende variabler ved å bruke historiske målinger av prosessdata, er ganske nylig utviklet. Det er dermed utført relativt lite forskning på metoden fra før. Denne oppgaven presenterer innledende forsking på ulike aspekter ved den databasert metoden. For å forstå data metoden bedre i praksis anvendte vi den på tre ulike prosesser. En "dummy" -case" for å helt enkelt bare begynne innlednde forsøksrunder, deretter benyttet vi en fordamperprosess for å belyse bruk av metoden og utfordringer knyttet til dette. Til slutt studerte vi data-metoden ved å bruke en CSTR-reaktor knyttet til en destillasjonskolonne med resirkulering. Dette ble den mest omfattende prosessen med de mest omfattende undersøkelsene.

Vi fant at denne fremgangsmåten i noen tilfeller kan være et godt alternativ til exact local metoden. Men mere forskning er nødvendig for å være i stand til å forstå hvilke og hvordan ulike faktorer påvirker utfallet for den databasert metoden.

Vi håper at arbeidet som presenteres her vil inspirere til framtidig forskning på denne lovende metoden for å finne selvoptimaliserende variabler.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

While reading articles written on the subject of process control a frequent topic is how to achieve optimal operation of process plants. It is necessary to ensure optimal operation both from an economical and environmental point of view. Competition is growing largely due to a growing global market. This constraints the requirements for the products both in terms of price and quality. Moreover, a rising focus on establishing an environmentally friendly industry forces many industrial plants to improve their operation of the processes. For example to decrease the emissions of hazardous gases or reduce energy usage, as new government regulation on the amount of toxic wasted from a process. This introduces new and/or stricter constraints on an already existing plant. We are therefore looking at ways to systematically improve the operation of already existing plants, as well as how to ensure that new process plants are financially efficient and environmentally friendly.

When discussing optimization in this thesis we only consider steady-state operation. Most of the plant operation is carried out at steady-state which therefore has the greatest impact on the economical aspect of operation. Dynamic operations such as shut-down or start-ups will have a different solution than the daily basis steady-state operation. It is, however, usually sufficient to consider optimal operation at steady state only. This thesis is primarily inspired by Skogestad's pant-wide control approach to ensure optimal operation of a plant at steady-state operation [1].

The first step in optimization is to quantify what we want to optimize. This means defining a scalar cost function together with the limitations or constraints in the process. Subsequently, we identify the available degrees of freedom (manipulative variables) in the plant. The process is then optimized by minimizing the defined cost function while ensuring that the constraints are not violated. The cost function can for example be steam used in the re-boiler in a distillation column, which from an economical point of view is optimal to keep at minimum. However, enough steam must be used to ensure that the product from distillation has the desired composition.

Once the optimal solution is found it must be implemented in the process. This is achieved through a control structure. This includes deciding what to control and to which set-point. Additionally, we know that the conditions for a process are under constant change. For example changes such as temperature or feed flow and variation in composition. The challenge is to ensure optimal operation also when the operational conditions change. A recurring question is: when we know the nominal optimal solution, how can we use control systems to keep the process at optimal operation when disturbances occur?

The true optimal solution to this question would be to use a centralized online optimizer that would continuously monitor and update the model parameters [2]. However, this is practically impossible to implement and the control system is usually decomposed into several layers, operating in different timescales. Typically, the optimizing takes from a few hours to a day, while the control of the system works within seconds or minutes. Choosing the right control variables and set-points can be considered as the link between these two layers. Based on the present operation conditions the optimizer provides the set-point to the (lower layer) controllers. The result is optimal operation for exactly the present operation conditions. The problem arises when disturbances occur in between optimization, causing the process operation to deviate from optimal operation until the process is re-optimized again.

A possible solution is using self-optimizing control variables. These are variables that when kept at a constant set-point restrain the deviation from optimal operation [5]. A control variable is said to be self-optimizing if keeping it constant leads to an acceptable loss also when the process is influenced by disturbances. Consequently eliminating the need for re-optimization when disturbances occur.

The (self-optimizing) control variables can either be single variables kept at a constant value, or a (linear) combination of variables kept at a constant set-point. The control variable can be expressed as:

$$\mathbf{c = Hy}$$

where H is the selection or combination matrix, selecting or weighting the variables. There are already well developed methods to find the optimal H such as the maximum gain rule, null space method and the exact local method. The issue is that all these methods rely on a process model, which not always is well known. In the cases without a process model methods like surface response methods[22] or extreme seeking [21] can be used. However, as pointed out by Jäsche and Skogestad [7] surface response methods rely on disturbance measurements which are often not available, and extreme seeking requires excitation of the process, in many cases not possible to perform. Therefore, there has been research on ways to find an optimal combination of measurements using only empirical process data.

Process data is collected continuously at practically all process plants, and this data contains huge amounts of information about the processes. Today this data is used mainly for supervisory purposes, and in some cases to update process model in order to do re-optimization. Recently, Jäschke and Skogestad have researched the possibilities to use the historical plant data to find self-optimizing control variables. The main work on the subject can be found in the articles "*Controlled Variables from Optimal Operation Data*" [4], "*Optimal Use of Measurements for Control, Optimization and Estimation using the Loss Method: Summary of Existing Results and Some New*" [6] and the most recent "*Using Process Data for finding Self-optimizing Controlled Variables* [7]". This new idea or method uses plant data to estimate a quadratic cost function, and find the best way to combine measurement variables as a self-optimizing variable. Two of the main advantages are that this approach needs neither a process model nor disturbance measurements. The control variables are found as an estimate of the cost function gradient.

Scope of the thesis

The aim of this thesis is to introduce this new method and the ideas it is built upon. It is of interest to know more about which parameters and what preconditions affect the method. In this

thesis, we have therefore looked further into the data-based method by applying it on different processes. Some of the questions raised (in this thesis) are: (1) how does it perform in comparison to other model based methods, (2) will changing the number of measured variables have any effect on the outcome, (3) can we improve the performance by including the disturbances as measured variables and (4) how well does the data method handle measurement noise,.

Originally the plan was to test this method on a biodiesel plant. We wanted to use an already developed model of a biodiesel plant in ChemCad to generate data, and treat it as if it were a real process plant. We wanted to use this data to estimate the combination matrix, H, with the data-based method, and then use H to control the process in ChemCad. This turned out to be somewhat more complicated and time consuming than first expected. We had problems getting a license to the program, as well as a computer to run on ChemCad. Getting this ready took several weeks and the progress in the start-up face of the project was therefore slow. We also wanted to compare the data method to other model-based methods such as the exact local and null space method. And for this we would have to be able to optimize the process. Furthermore, to be able to use the data based method the data must be collected when the process is operated close to optimum. This is namely one of the conditions for using the data-based method. The process must therefore be optimized before collecting the data. In addition, optimization is necessary to be able to compare loss between controlling the process with the estimated H-matrix and re-optimizing it for a given disturbances. Which is a way to compare how well a control variable works as a self-optimizing variable. After working with ChemCad for a few weeks, we learned that the program itself has no optimizing features. Optimization could therefore only be done using a different programming language to control ChemCad. MATLAB could not be use for this purpose. We decided that learning a new language and then connect it with ChemCad would be too time consuming, especially since we already had spend long time on setting up the license for ChemCad already.

The importance for this thesis was not the biodiesel plant itself, which was only meant to work as a test case. The aim of this project was rather to research different aspects of the data-based method. We decided that this could be done using any type of process, and for simplicity we chose some smaller test cases. So, instead of the biodiesel plant, we used an evaporator process and more complicated CSTR-distillation column with recycle process as test cases. We already had a model ready in MATLAB for these two cases. We were able to manipulate the already existing codes to generate data, and optimization the processes was done by using the built-in MATLAB function *fmincon*.

The next chapter (Chapter 2) will present the most central theory needed to understand the data based method. It tires to place the data-based method in the whole optimization and control problem, together with a short presentation of the other alternative methods. The theory part is limited to cover only relevant topics about the concept of self-optimizing variables and the approaches used to identify such variables in this thesis. Hence, the reader is expected to have some basic knowledge about process control.

One of the main tools used in finding the combination matrix is Partial Least Square (PLS) regression. The second chapter (Chapter 3) will put forward some background information and basic theory about PLS-regression. However, no more detailed explanation about the mathematical technicalities will be given. For this, the reader is referred to the literature, the procedure is well explained in both [9] and [10] .

In Chapter 4 a more detailed introduction to the data based method is given. It explains and elaborates the theory presented in Chapter 2 and Chapter 3 with a “dummy” test case. Here

the practical usage of the method is presented, and some main aspects of the PLS-regression are pointed out. In the two subsequent chapters, Chapter 5 and 6, the method is used to identify an optimal control structure, i.e calculating the combination matrix. These chapters research the effects of changing different preconditions for the H-matrix calculation. Each chapter starts by presenting the process used as a test case, what parameters are changed and the procedure used in order to test the data method. This is followed by the results from the testing and a discussion of the results found for the test-case.

In Chapter 7 a final discussion is given, where the analysis is based on perceptions from all three test cases together, this discussion also includes recommendation for future work. The thesis is rounded with a general conclusion summing up the most significant findings.

Chapter 2

Theory
Process control and optimization

This theory chapter will present relevant background information and theory for the newly developed data based method to find self optimizing control variables. The aim is to understand how the method works, what alternatives exists, how are the self-optimizing variables found today, and last, but not least, where it fits in, in the whole plant wide control procedure.

2.1 Skogestads Plant-wide Control Procedure

This thesis is mainly inspired by the systematic plant wide control procedure developed by Skogestad [1]. The procedure is divided into two parts, the top down and the bottom up part. Where the top down part focuses on achieving a favorable economic performance. Whereas the bottom up part focuses more directly in the actual control structure and layout. The procedure is further divided into seven steps (4 + 3), briefly summarized here.

Top down

Step 1 Define the operational objectives; the cost function J and the process constraint.

Step 2 Identify steady-state degree of freedom, optimal steady state condition (optimize without disturbance) and find the active constraints.

Step 3 Select primary economic controlled variables by using the degrees of freedom. After the active constraints are controlled, find self-optimizing control variables.

Step 4 Locate the throughput manipulator

Bottom up

Step 5 Select the control structure of the regulatory control layer.

Step 6 Select the control structure of the supervisory control layer.

Step 7 Select the control structure of the optimization layer.

The data based method studied in this thesis is linked to *Step 3*. In many processes we find that after controlling the active constraint there are still some remaining degrees of freedom left. They can be used to optimize the process. However, despite the fact that the issue of self-optimizing variables does not come into account before *Step 3*, both *Step 1* and *Step 2* are important for the data method. These two steps decides the pre-conditions under which the data method is based upon.

Selecting a good control structure for the process to actually execute the findings from the top down part is important as well. This will, however, not be a topic in this thesis. Nevertheless, it is important to keep in mind that a self-optimizing control structure is not fully tested before it has been implemented as a control structure in the research process, and found feasible in terms of control valves, product quality (and similar). It is a long way from theoretically testing and finding an adequate procedure, to the actual implementation of the control structure.

Cost function and degrees of freedom analysis

Following Skogestads plant-wide control procedure, the first step is defining the operational objectives, meaning the cost and operational constraints. It is assumed that these objectives can be quantified in terms of a scalar cost (or profit) function. If it is expressed as a cost function, we want to minimize it. On the other hand, if it is expressed in terms of profit we want to maximize the profit by minimizing the negative profit function. The cost function is a function of the system states (x), the inputs (u) and disturbances (d). It is also subject to both equality constraints ($h(x)$) given by the system model such as mass flows. And inequality constraints ($g(x)$) limiting the operation, such as temperature and pressure limitations.

A general optimization problem can be formulated:

$$\max_x \ f(x) \quad \text{or} \quad \min_x \ -f(x) \tag{2.1a}$$

Subject to:

$$h(x) = 0 \tag{2.1b}$$
$$g(x) \leq 0 \tag{2.1c}$$

When operating a process there will (almost) always be some constraints that are active, for example because of safety reasons. They must therefore be controlled, and according to Skogestad, it will always be optimal to control the active constraints [1]. If there still are some degrees of freedom left after ensuring that all the active constraints are controlled, they can be adjusted to optimize the process. Which constraints are active or not will change depending on the disturbances, this means that there are different operation modes. When optimizing a process by using

self-optimizing control, it is important to ensure that the set of active constraints remains the same for the encounter disturbances. A change in the active constraints will cause the optimal operation settings to change as well.

The second step in Skoegestad's plantwide control procedure is to decide the number of degrees of freedom in the process. The simplest way to decide the number of degrees of freedom is counting the valves. Each valve serves as a degree of freedom, in addition some process equipment also represent one degrees of freedom. A way to understand the degrees of freedom is as things that can be adjusted in the process and will affect the operation of the plant. When counting all the degrees of freedom we are usually only interested in those with a steady-state effect. Therefore we subtract those with merely a dynamic effect, which are for example valves controlling the level in tanks. After subtracting the degrees of freedom used in control for safety reasons and those with only a dynamic effect, we are left with the degrees of freedom we can use to optimize the process.

Once the degrees of freedom available iare decided, important disturbances and their range are identified. Common disturbances are feed rate or composition, or changes in temperature or pressure, as well as prices on both feed and product. The process is optimized for given disturbances with the degrees of freedom available, mainly to establish if the active constraints change.

2.2 Control variables and self-optimizing variables

Once an optimal operation policy is found for a process, the next step will be how to implement it. This is done by choosing control structures. The next question is then basically to choose what to control. For each degree of freedom we can choose one control variable. As explained earlier the active constraints must be controlled and therefore use one degree of freedom. They are referred to as constrained degrees of freedom. Skogestad suggests to use the remaining *unconstrained* degrees of freedom to find self-optimizing variables [1]. The layout for self-optimizing control is given in Figure 6.2.

The process is controlled such that certain variables are kept at given values called *set-points*. The set-points are given from an optimizer. The optimizing layer computes the set-points for the process at a given state, and the control layer tries to keep the variables at the set-point value using for example PI- or PID-controllers. The problem is that no process stays the same over a longer period of time, a process will be exposed to changes. These changes are both foreseen and unforeseen disturbances. The problem is that optimization of a process is typically carried out once a day. Therefore when a disturbance occur in-between optimizations, the operation is no longer optimal, This leads to a loss expressed as $L = J - J_{opt}$, where J is the cost function [16].

Definition: **Self-optimizing Control** [5]
Self-optimizing control is when we can achieve an acceptable loss with constant set-point values for the controlled variables

The control structure is self-optimizing if the operation stays near optimal even when disturbances occur. If the subsequent loss is acceptable small, there is no need to re-optimize the operation of the plant. The operation of the plant is merely done by keeping certain variables at constant set-points. These key variables are referred to as self-optimizing variables. In a process

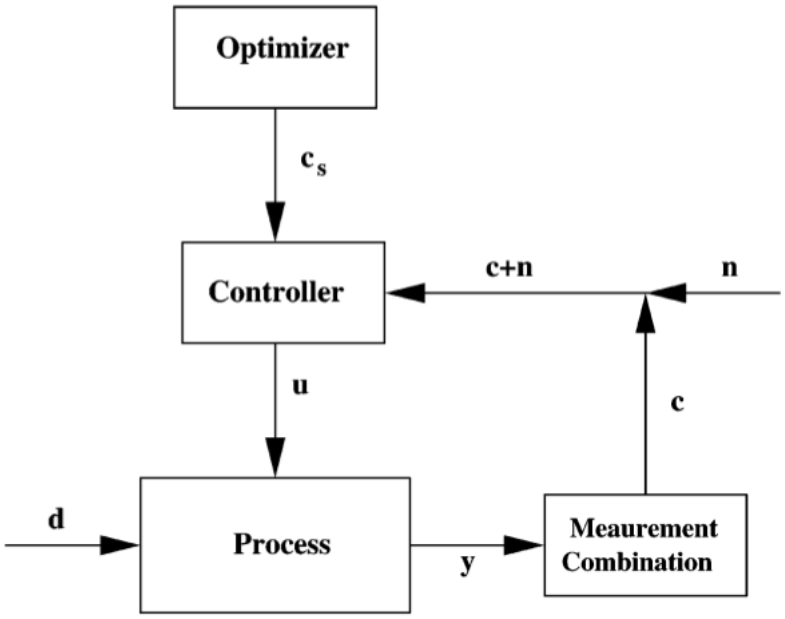


Figure 2.1: Dividing optimization and control into two layers can be illustrated in this way. Where the optimizer provides the set-points, and the controller tries to keep c=Hy at the constant set-point value when disturbances occur.[16]

there are as many self-optimizing variables as unconstrained degrees of freedom. How well a variable works as a self-optimizing variable can be determined by considering the loss between re-optimizing the process and using self-optimizing control to deal with the disturbance ($J_{opt}(d)$)-$J_{soc}(d)$). Some variables are better than others (leads to a smaller loss) as self-optimizing variables (Figure 2.2).

There are four requirements for a good self-optimizing variable, which all must be satisfied [1]. The requirements are: (1) To avoid disturbances to have a too large effect on the self-optimizing variable, the optimal value should be insensitive to disturbances; (2) it should be easily measured and controlled; (3) the gain from the input variable to the self-optimizing variable should be large to ensures that a small change in the input will have a large effect on the output. Large control actions will therefore not be necessary to keep the process at a desired operation level; And last, (4) in cases where there are more than one self-optimizing variable, the variables should not be closely connected.

The issue is how to find a good self-optimizing structure. Normally the self-optimizing control structure is a combination of several measurements, and the goal is to find a good linear combination of the available measurements and keep the combination at a constant value. This thesis addresses this topic and will test a new method to find a good self-optimizing structure.

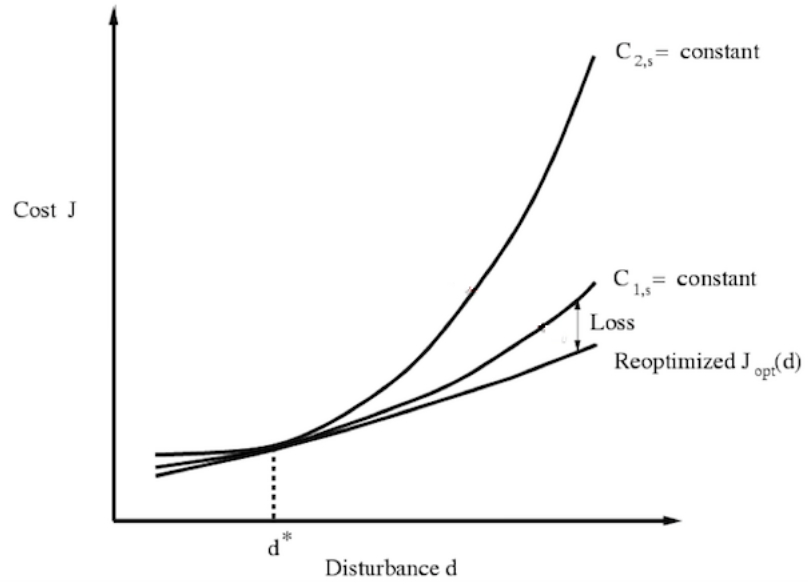


Figure 2.2: The different outcomes in terms of the cost function when the a process is re-optimized or controlled with self-optimizing control when a disturbance occur. [3]

2.3 The optimal solution

Before finding a self-optimizing control structure, we must redefine the optimization problem. So that it only concerns the remaining degrees of freedom, since these are the degrees of freedom we are trying to find the best combination of. When all the active constraints are controlled, the optimization problem is reduced to an unconstrained lower-dimensional problem. From this point, the cost function is only a function of the inputs and disturbances (not the system states x).

$$J = f(u, d) \tag{2.2}$$

This function can be expanded around its nominal point (labeled with a star), using second-order Taylor-expansion. The nominal point is usually found by optimizing the process with no disturbances.

$$J \approx J^* + \begin{bmatrix} J_u^* & J_d^* \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta d \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \Delta u^T & \Delta d^T \end{bmatrix} \begin{bmatrix} J_{uu}^* & J_{ud}^* \\ J_{du}^* & J_{dd}^* \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta d \end{bmatrix} \tag{2.3}$$

where $\Delta u = u - u^*$ and $\Delta d = d - d^*$, J_u^* and J_d^* are the first derivatives and J_{uu}^*, J_{ud}^*, J_{du}^* and J_{dd}^* are the second derivatives, all evaluated at the nominal point.

Using the same approach as in Equation 2.3 the gradient can be approximated as;

$$J_u = J_u^* + \begin{bmatrix} J_{uu}^* & J_{ud}^* \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta d \end{bmatrix} \tag{2.4}$$

The optimal solution would be to use the gradient as a self-optimizing variable and control it to zero (which would ensure either a maximum or minimum value). Unfortunately, this cannot be done since it would be very difficult to measure the gradient, besides the cost function depends on disturbances which cannot be manipulated. Instead we express the gradient as an approximation in terms of variables we can measure, and utilize this as a self-optimizing variable.

To do this, we need to linearize the relation between the measurements, the inputs and the disturbances **??**. The linearized model can be expressed as:

$$\Delta y = G^y \Delta u + G_d^y \Delta d = \tilde{G}^y \begin{bmatrix} \Delta u \\ \Delta d \end{bmatrix} \tag{2.5}$$

Where the gains are $G^y = (\frac{\delta f_y}{\delta u})^{*T}$ and $G_d^y = (\frac{\delta f_y}{\delta d})^{*T}$.

Solving Equation 2.5 with respect to the Δu and Δd matrices and inserting this into Equation 2.4, gives us the gradient expressed in terms of the measurements [1].

$$J_u = \begin{bmatrix} J_{uu}^* & J_{ud}^* \end{bmatrix} \left[\tilde{G}^y\right]^\dagger \Delta y \tag{2.6}$$

The self-optimizing controlled variable (Δc) will be a selected combination of the available measurements (y), and can be written

$$\Delta c = H \Delta y \tag{2.7a}$$

Where H is a constant selection or combination matrix:

$$H \equiv \begin{bmatrix} J_{uu}^* & J_{ud}^* \end{bmatrix} \left[\tilde{G}^y\right]^\dagger \tag{2.7b}$$

The H-matrix gives the *locally* optimal combination of the available control variables as shown in Equation 2.7b, and is therefore a key parameter in finding a self-optimizing control structure. The H can in reality be chosen freely and the simplest choice is using one single measurement to control c. However, this is not always possible to achieve good self-optimizing control using only one measurement. The challenge is how to select H, or in other words how to combine the available measurements, to achieve the best control structure for the process. In order to calculate H, the parameters J_{uu}^*, J_{ud}^* and $\tilde{G}^y$ must be know (Equation 2.7b). However, these parameters can be difficult to obtain.

For that reason other methods have been developed in order to find good self-optimizing control structure, where less information about the process is needed. There are different approaches in order to decide H and some well-known examples are: the "brute force" approach, maximum gain rule, the null space method and the exact local method [1]. In this thesis we will use a

[1] This requires the number of measurements to be equal or greater than the number of inputs plus disturbances

newly developed method using historical plant data in order to find H. The data-based method is described in the article "*Using Process Data for Finding Self-optimizing Controlled Variables*" by Jäschke and Skogestad [7].

In the test cases given in this thesis, the data-based method will be compared to the exact local and null space method. A short explanation of these two methods are therefore given next. Followed by a more through derivation of the data-based-method.

2.3.1 The null space method

The null space method is used to select the H-matrix in cases where there is no implementation error, i.e no noise. The basic idea behind this method is that H is found such that $\mathbf{HF} = 0$, where $\mathbf{F}$ is the optimal sensitivity matrix [14].

$$F = \frac{\delta y^{opt}}{\delta d} \tag{2.8}$$

We already explained that controlling $\Delta c = H\Delta y$ to zero yields optimal operation. If we rearrange Equation 2.8 and insert it to this expression it gives:

$$\Delta c^{opt} = H\Delta y^{opt} = HF\Delta d = 0 \tag{2.9}$$

We know that neither Δd nor F are zero, therefore to ensure optimal control HF must in this case be zero.

2.3.2 The exact local method

The exact local method can be used also when noise is taken into consideration. In the case with measurement noise (y^n) we want to control

$$c_m = H(y + y^n) = Hy_m$$

The disturbances are expressed in the matrix W_d. Where the element in the diagonal in the matrix W_d represents the magnitudes of each disturbance. The measurement noise is expressed in the matrix W_n, where the diagonal elements in W_n are the magnitude of the noise for each measurement. In order to use the exact local method these two matrices must be known.

The magnitudes can be expressed as

$$n = HW_n^y n^{y'} = W_n n^{y'} \tag{2.10a}$$

$$d - d^* = W_d d^{'} \tag{2.10b}$$

where $n^{y'}$ and d' are normalized such that their magnitudes are less than one. It is common to speak of the worst case loss, which is the loss when the combined normalization vectors for the errors have 2-norm less than one.

$$\left\| \begin{matrix} d' \\ n^{y'} \end{matrix} \right\| \leq 1 \tag{2.11}$$

Or an average loss when the errors are normal distributed.

$$\left\| \begin{matrix} d' \\ n^{y'} \end{matrix} \right\| \in N(0,1) \tag{2.12}$$

The two losses can be expressed as

$$L_{wc} = \frac{1}{2}\bar{\sigma}(M)^2 \tag{2.13a}$$

$$L_{avg} = \frac{1}{2}\|M\|_F^2 \tag{2.13b}$$

Where:

$$M = J_{uu}^{\frac{1}{2}}(HG^y)^{-1}HY \tag{2.13c}$$

with

$$Y = [FW_d \quad W_{n^y}] \tag{2.13d}$$

Basically, the main idea of the exact local method is that you want to pick the H that minimizes M, and thereby also minimizes the loss.

The H-matrix from the exact local method is calculated from the formula:

$$H_{exl} = G_y^T(YY^T)^{-1} \tag{2.14}$$

Where G_y^T is the measurement gain matrix.

2.4 Using operational data

In this section the idea of using plant data to find self-optimizing variables will be explained based on the description given in [7] Jäschke and Skogestad.

At a process plant data is collected (almost) continuously to supervise the process. It is of interest to be able to use the already available data to control and optimize a process. Until recently, the data is mainly used to estimate unmeasured variables, and over the last years most of the publications on the area has been suggestions on how to use the plant data for online

process optimization, such as empirical data based modeling. Recently, Jäschke and Skogestad [4] and Skogestad et. al [6] have developed data-based methods to optimize a process. In the paper "*Using Process Data for Finding Self-Optimizing Controlled Variables*" [7] Jäschke and Skogestad suggest to use historical plant data to obtain a quadratic model of the cost function and obtain the H-matrix to find self-optimizing control variables. The advantage with this method is that no model is needed. It is therefore a cheap and easy way to find an optimal control structure. The idea is to detect how the cost function changes with certain measurements by using regression to predict the relationship between the cost function and measured variables. A requirement is that enough data around the optimal point of operation is available.

2.4.1 Expressing the relationship between the cost function and output variables

The first step in the data-based method is expressing the cost function in terms of measurements. The cost function can be approximated around the nominal point by a Taylor expansion as shown in Equation 2.3. The linearized model of the measurement model is given in Equation 2.5, which can be rewritten as:

$$\begin{bmatrix}\Delta u\\ \Delta d\end{bmatrix} = \left[\tilde{G}^y\right]^\dagger \Delta y \tag{2.15}$$

Inserting this into the expression for the approximated cost function in Equation 2.3 yields the cost function expressed in terms of measurements.

$$J = J^* + \begin{bmatrix}J_u^* & J_d^*\end{bmatrix}\left[\tilde{G}^y\right]^\dagger \Delta y + \frac{1}{2}\Delta y^T \left[\tilde{G}^y\right]^{\dagger T}\begin{bmatrix}J_{uu}^* & J_{ud}^*\\ J_{du}^* & J_{dd}^*\end{bmatrix}\left[\tilde{G}^y\right]^\dagger \Delta y \tag{2.16}$$

Grouping the first derivatives in one term and the second derivatives in another term makes the expression somewhat neater.

$$J = J^* + J_y^* \Delta y + \frac{1}{2}\Delta y^T J_{yy}^* \Delta y \tag{2.17a}$$

Where:

$$J_y^* = \begin{bmatrix}J_u^* & J_d^*\end{bmatrix}\left[\tilde{G}^y\right]^\dagger$$
$$J_{yy}^* = \left[\tilde{G}^y\right]^{\dagger T}\begin{bmatrix}J_{uu}^* & J_{ud}^*\\ J_{du}^* & J_{dd}^*\end{bmatrix}\left[\tilde{G}^y\right]^\dagger$$

The second derivative can also be expressed as:

$$J_{yy}^* = \left[\tilde{G}^y\right]^{\dagger T}\begin{bmatrix}[J_{uu}^* & J_{ud}^*]\tilde{G}^{y\dagger}\\ [J_{du}^* & J_{dd}^*]\tilde{G}^{y\dagger}\end{bmatrix}$$

Remembering the definition for the H-matix given in Equation 2.7b, we recognize the H-matrix in the expression above.

$$J_{yy}^{*} = \left[\tilde{G}^{y}\right]^{\dagger T} \begin{bmatrix} H \\ [J_{du}^{*} \quad J_{dd}^{*}]\tilde{G}^{y\dagger} \end{bmatrix}$$

The H-matrix that we are looking for will therefore be the n_u (the number of inputs) rows of $J_{yy}^{*}\left[\tilde{G}^{y}\right]^{T}$. Since H is the main interest for control purposes, we only need the first part of the $\tilde{G}^{y}$-matrix. Pre-mulitplying J_{yy}^{*} with with $\begin{bmatrix} G^{y} & 0_{n_y x n_d} \end{bmatrix}^{T}$ yields an expression which contains only J_{yy}^{*}, G^{y} and H.

$$\boxed{\begin{bmatrix} G^{y} & 0_{n_y x n_d} \end{bmatrix}^{T} J_{yy}^{*} = \begin{bmatrix} H \\ 0_{n_y x n_d} \end{bmatrix}} \tag{2.18}$$

The gain matrix

Obtaining the measurement gain matrix, G^{y}, is easy. The simplest way will be to preform a step change in the inputs and measure the change in the outputs. The i^{th} element of the gain matrix can be expressed as:

$$g^{(i)} = \frac{y - y^{*}}{u_i - u_i^{*}}$$

Where the star (*) indicates the nominal value and, the new value after the step change are without the star.

The second derivative approximation $\mathbf{J}_{yy}$

Obtaining J_{yy}^{*} is slightly more complicated than finding the gain matrix. It is found by using a mathematical tool: Partial Least Square (PLS) regression.

Before the measurements can be used in PLS-regression we need to make some assumptions: (1) The data is measured when the process is operated in open loop; (2) The number of independent measurements are greater than or equal to the number of independent inputs plus disturbances, $n_y \geq n_u + n_d$; (3) Active constraints are kept constant by control; (4) The data contains all relevant disturbances; (5) As the data is collected the plant is at steady state; and finally (6) the process is operated close to optimum such that the cost can be approximated by a quadratic cost function.

When the assumptions are valid the data can be used to find J_{yy}^{*}, however before the data can be used some preparations are needed.

Scaling and Centering

To be able to use all the different measurements together they need to be centered and scaled. The measurements are centered by subtracting the mean value of the measurements of the same variable. The values are scaled by dividing all the measurements of the same variable by the largest value of the measurements.

If they are not scaled, high value measurements such as temperature or pressure will be weighted as more important that low value measurements such as mole concentrations, which is not desired. The scaling and centering are therefore crucial in order to obtain a realistic and useful result.

In the next chapter we will present some basic theory for the PLS-regression. And the scaling and centering procedures are more thoroughly explained there.

Estimating the quadratic model of the cost function

To obtain a quadratic model the product of the measurements are also taken into consideration, This is done by augmenting the data by all the second order terms. For n different measured variables the raw data is given below in the matrix Y_{raw} and the augmented data is given in the matrix Y_{aug}.

$$Y_{raw} = \begin{bmatrix} y_1 & y_2 & ... & y_i & ... & y_n \end{bmatrix} \tag{2.19a}$$

$$Y_{aug} = \begin{bmatrix} y_1 & y_2 & ... \, y_n & y_1^2 & y_1 y_2 \, ... \, y_1 y_n & y_2^2 & y_2 y_3 \, ... \, y_{n-1} y_n & y_n^2 \end{bmatrix}^T \tag{2.19b}$$

The cost function that we wish to model must also be measured. Here it is given in matrix J.

$$J_m = \begin{bmatrix} J_1 & J_2 & ... & J_i & ... & J_n \end{bmatrix}^T \tag{2.20}$$

$$\tag{2.21}$$

Because the measurements are usually not independent variables we cannot use normal regression to fit the data to the quadratic cost function, hence Partial Least Square regression is used. It is especially suitable since it handles both co-linearity and linear dependence of the data. Essentially, the PLS algorithm projects the Y and J_m data onto a lower dimensional space, simplifying the problem, while still calculating the most significant correlations. After running PLS-regression in for example MATLAB with Y and J as input variables, the regression method calculates a regression factor β which predicts J as a function of Y_{aug}. The prediction of the cost function is then modeled as:

$$J = \begin{bmatrix} 1 & y_{aug}^T \end{bmatrix} \beta \tag{2.22a}$$

Or written out, where m is the dimension of β:

$$J = \beta_1 + y_1\beta_2 + y_2\beta_3 + ... + y_n\beta_{n+1} + y_1^2\beta_{n+2} + y_1y_2\beta_{n+3} + ...y_{1-n}y_n\beta_{m-1} + y_n^2\beta_m \tag{2.22b}$$

Equation 2.22b gives the cost function in terms of measurements, just like in Equation 2.17a. The expression in Equation 2.22b can be rearranged to fit the quadratic expression from the cost function given in 2.17a.

$$J = J^* + J_y^* \Delta y + \frac{1}{2} \Delta y^T J_{yy}^* \Delta y$$

$$J = \beta_1 + \begin{bmatrix} \beta_2 & \beta_3 & \cdots & \beta_{n+1} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} + \frac{1}{2} \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix} \begin{bmatrix} 2\beta_{n+2} & \beta_{n+3} & \beta_{n+4} & \cdots \\ \beta_{n+3} & \ddots & & \cdots \\ \beta_{n+4} & & & \\ \vdots & \vdots & \ddots & \vdots \\ & & & 2\beta_m \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{2.23}$$

Remembering the coefficients from Equation 2.17a, we see that J_{yy} is the largest β-matrix. We are now able to calculate the H-matrix as given in Equation 2.18 by only utilizing historical plant data.

Chapter 3

Theory
Partial least square regression

The main mathematical tools used in the data-based method is Partial least square (PLS) regression. This tool, which is an inbuilt command in MATLAB, allows us to find a linear model describing the relationship between the plant measurements y and the cost function J. The main idea behind PLS is to find directions in a data set X with the greatest covariance with another data set Y. From this the relationship between X and Y is modeled. X and Y will in four case be plant measurements y and the cost function J, respectively. This leads back to the idea that if we have enough measurements around the optimal operation point, we can determine the optimal control policy. The optimal structure (in this case in terms of a combination matrix H) is found by identifying the relations between the measurement values and the corresponding cost function value. The parameters used to model the cost function, can also be used to estimate the H-matrix.

PLS-regression is especially useful in this case because it can handle and analyze data which is highly correlated, co-linear and noisy [10]. This property makes PLS-regression a better analyzing tool than normal linear regression.

3.0.2 Number of components

When using data to fit a model one important decision is; how complex should the model be? The model complexity will in our case be determined by the number of components used in the PLS-regression, or in other words how many directions in the data should be explained by the model. Too few components will lead to an inaccurate model where important information, such as certain relations between variables, most likely is lost. Including too many components on the other hand, can result in an "over-fitted" model. This leads to a perfectly fitted model for the data at hand, however when the model is used on a new data-set the fit will be poor. We risk to model relations that do not exist in reality, but is still found due to a too high complexity specification.

Different approaches on how to decide the ideal number of components will be explained in the next chapter using a “dummy” example.

3.0.3 Preparing the data

Plant measurement will typically have different units, and the value of the different variables differs a lot with respect to size. Pressure measurements will have a much higher value than for example composition measurements. However, the pressure is not necessarily more important than composition, when it comes to affecting the cost function. To avoid that measurements of levels and pressure are weighted as more important, than for example composition measurements, the measurement data is scaled and centered before used in the PLS-regression. This way, the modeling is focused evenly on all the variables.

Centering and scaling do not change the overall interpretation of the data. If two variables were strongly correlated before centering and scaling, they will still be strongly correlated after as well.

Centering

Centering is done by finding the average of all the samples, and subtracting it from all the samples.

In the test-cases described later in this thesis the data is fabricated by creating random differences in the disturbances and inputs, or directly in the measurements, which means that the data is already centered. In "real plant cases", the data is not created in this almost systematic way, and should be centered. In the dummy case the measurement data is calculated from randomly created parameters, and are for example not centered.

Scaling

To handle the issue with process data having different units, the data is scaled. This way, the data becomes unit-less. The scaling is done by dividing all the samples of a variable by the maximum absolute value measured of the variable.

The centering and scaling scheme is summarized in Figure 3.1.

Centering and scaling a data-set of n variables and i samples:

$$Y_{raw} = \begin{bmatrix} y_1^1 & y_1^2 & y_1^3 & \dots & y_1^i \\ y_2^1 & y_2^2 & y_2^3 & \dots & y_2^i \\ \vdots & & & & \\ y_n^1 & y_n^2 & y_n^3 & \dots & y_n^i \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$$

$$Y_{mean} = \begin{bmatrix} |Y_1| \\ |Y_2| \\ \vdots \\ |Y_n| \end{bmatrix} \qquad Y_{max} = \begin{bmatrix} y_{max}^1 \\ y_{max}^2 \\ y_{max}^3 \\ \vdots \\ y_{max}^n \end{bmatrix}$$

Centering the data:

$$Y_{centered} = Y/Y_{mean} = \begin{bmatrix} Y_1/|Y_1| \\ Y_2/|Y_2| \\ \vdots \\ Y_n/|Y_n| \end{bmatrix} = \begin{bmatrix} Y_c^1 \\ Y_c^2 \\ \vdots \\ Y_c^n \end{bmatrix}$$

Scaling the centered data:

$$Y_{scaled} = Y_{centered}/Y_{max} = \begin{bmatrix} Y_c^1/y_{max}^1 \\ Y_c^2/y_{max}^2 \\ \vdots \\ Y_c^n/y_{max}^n \end{bmatrix}$$

Figure 3.1: The procedure for scaling and centering the raw measurement data.

Chapter 4

The data based method
A "dummy" test case

This chapter does not include any comparisons or evaluations of this method compared to other methods. Nor does it use an actual process as an example. It will simply explain the usage of the data based method and point out some of the most important aspects.

4.1 Building a test case

The first step in building a test case is to define the problem. To use the data based method we need the gain matrix, measurements of process data and the cost function. Since this is only a dummy case we set the gain and the second derivatives of the cost function randomly. The input and disturbance values are also created randomly. To produce random values the *randn* function in MATLAB was used. The "measured" outputs were generated from the inputs and disturbances together with the gain matrix, by using Equation 4.1. The cost function was calculated by using the second derivatives and the input and disturbance matrix, ud.

$$Y_m = G_p \times ud \tag{4.1a}$$

where G_p is the gain matrix, build up by the maesurement gain and disturbance gain matrix

$$G_p = \begin{bmatrix} G_u & G_d \end{bmatrix} \tag{4.1b}$$

Table 4.1: The symbol explanation for the dummy case, with explanation of the meaning and the procedure to calculate it. *randn* refers to an inbuilt function for choosing random matrices in MATLAB, and just a line means it is freely chosen.

Symbol	Meaning	Calculation	Size
n_y	Number of measured variables	-	-
n_u	Number of inputs (DoF)	-	-
n_d	Number of disturbances	-	-
ns	Number of samples	-	-
G_u	The input gain	randn(nu,ny)	$n_u \times n_y$
G_d	The disturbance gain	randn(nd,ny)	$n_d \times n_y$
G_y	The process gain	$[G_u\ G_d]$	$n_y \times (n_u + n_d)$
J_{uu}	Second derivative of the cost function	randn(nu,nu)	$n_u \times n_u$
J_{ud}	Second derivative of the cost function	randn(nd,nu)	$n_d \times n_u$
J_{du}	Second derivative of the cost function	randn(nd,nu)	$n_d \times n_u$
J_{dd}	Second derivative of the cost function	randn(nd,nd)	$n_d \times n_d$
J_{sd}	Second derivative combination matrix	$[J'_{uu}\ J_{ud}\ ]$	
ud	Input and disturbance matrix	randn(ns,nu+nd)	$ns \times (nu + nd)$
Y_m	Data measurements	$G_y \times ud$	$n_y \times ns$
J	cost function	$\text{diag}(ud \times J_{sd} \times ud)$	$1 \times ns$

The measurement matrix for n measured variables and ns samples is in general:

$$Y_m = \begin{bmatrix} y_1^{ns=1} & y_1^{ns=2} & \cdots & y_1^{ns=ns} \\ y_2^{ns=1} & y_2^{ns=2} & \cdots & y_2^{ns=ns} \\ \vdots & & \ddots & \\ y_n^{ns=1} & y_n^{ns=2} & \cdots & y_n^{ns=ns} \end{bmatrix} \tag{4.2}$$

The cost function can in general be given as:

$$J_m = \begin{bmatrix} J_m^{ns=1} \\ J_m^{ns=2} \\ \vdots \\ J_m^{ns=n} \end{bmatrix} \tag{4.3}$$

The number of measured variables, disturbances and inputs (degrees of freedom) can easily be changed. However one must always make sure that the number of measured variables, n_y is greater than equal to the number of disturbances plus inputs $n_u + n_d$. An overview of how the case is build up is given in Table 4.1

With all these parameters chosen or determined the test case is well enough defined to use as a

test case for the data method. However, before using the data both the measurements and the cost function must be scaled and centered, as explained in the previous chapter.

The test cases all use the same data set as a basis for the calculations. The randomly chosen parameter and the data set calculated from them are created only once for test case A and once for test case B. This way, only the number of components used to calculate H and estimate J is changed. This makes it possible to compare the results for different number of components, since it is the only parameter changed between the test rounds.

4.2 Using PLS-regression to find the H-matrix

After the data is scaled and centered PLS-regression can be used to model the relationship between the measurements and the cost function. I.e. to identify how the cost function changes for different sets of measurements. The idea behind the method can be understood from this: We use the data and the cost function measurements to model the cost function as a linear function of the measured data. The cost function is modeled as:

$$J = \begin{bmatrix} 1 & Y'_{aug} \end{bmatrix} \beta$$

The β matrix is found by the PLS-regression. The linear approximation of the cost function can now be used in for example model validation, comparing the actually measured value to the modeled value. As explained in the theory chapter (Equation 2.23) the β matrix is used to estimate the first and second derivatives in the Taylor approximated cost function, J_y and J_{yy}. And as explained in the theory chapter, when J_{yy} is known together with the measurement gain we can find the combination matrix H from the formula:

$$\begin{bmatrix} G^y & 0_{n_y x n_d} \end{bmatrix}^T J^*_{yy} = \begin{bmatrix} H \\ 0_{n_y x n_d} \end{bmatrix}$$

In the estimation of the H-matrix we use only the top part of J_{yy}. We use as many rows of J_{yy} as we have extra degrees of freedom (u) in the process.

The gain matrix is known (randomly created) and the second derivative matrix, J_{yy}, is modeled from the data. The procedure seems simple enough, but there are some aspects that must be taken into account. One aspect of interest is how many number of significant components should be used. It is also interesting to know if the amount of measured samples makes a difference. If there is a huge process plant, with hundreds of measurement available, should only a few of them be used as a basis for the H-matrix calculations, or is it better to use all of the available measurements?

Last but not least, it is important that the modeling of the cost function is correct. If the estimated cost function differs a lot from the measured cost function, the model itself is poor. The calculation on the H-matrix is in this case not based on reality, and practically useless as a control variable.

4.3 Number of components

It is expected that the number of components used in the PLS-regression will affect both the model accuracy and control performance of the H-matrix. Next we will examine the influence the number of components (ncomp) has on the outcome from the data method. We decided to run two cases with different number of measured variables, disturbances and inputs. The test cases are labeled case A and B, and for both cases $n_y \geq n_u + n_d$:

Case A	Case B
$\mathrm{n}_y = 3$	$\mathrm{n}_y = 6$
$\mathrm{n}_u = 2$	$\mathrm{n}_u = 4$
$\mathrm{n}_d = 1$	$\mathrm{n}_d = 2$

The maximum number of components that can be used in a case depends on the size of the augmented measurements matrix, Y_{aug}. The maximum number of components is equal to the number of elements in the Y_{aug} matrix.

For example, for Case A where the measurements matrix consists of three measured variables:

$$Y_m = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix}$$

The augmented measurement matrix will be:

$$Y_{aug} = \begin{bmatrix} y_1 & y_2 & y_3 & y_1y_1 & y_1y_2 & y_1y_3 & y_2y_2 & y_2y_3 & y_3y_3 \end{bmatrix}$$

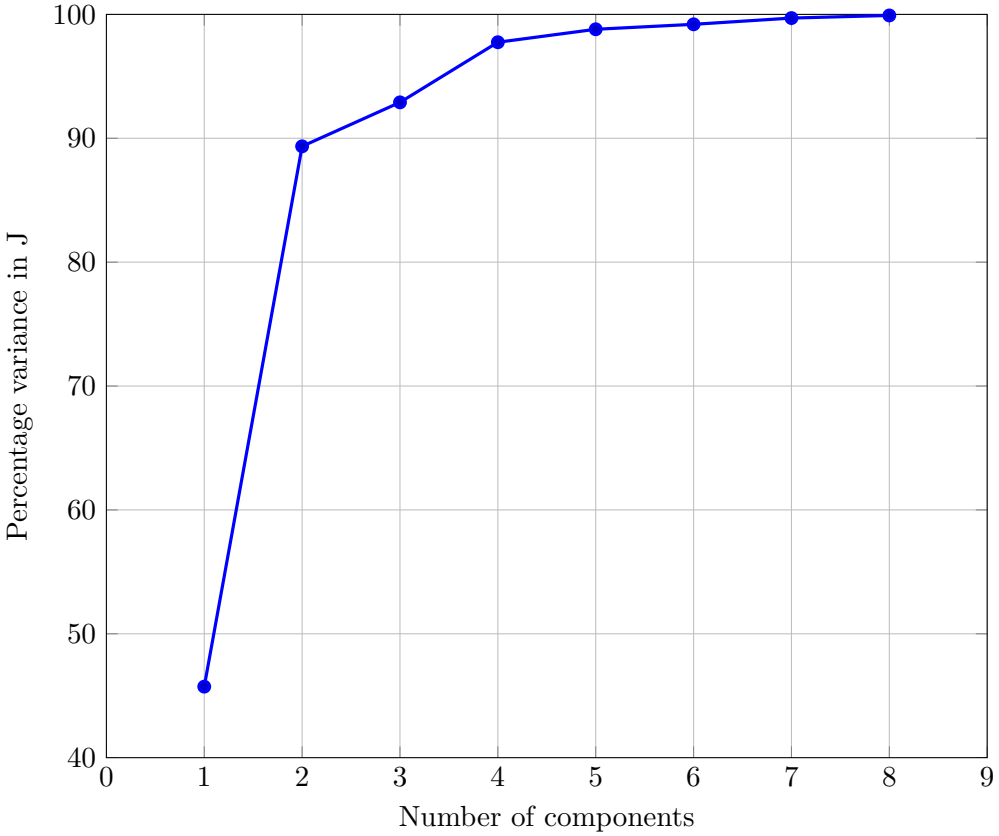
Since the augmented measurement matrix has 9 element, the maximum number of components will be 9. For the case where the measurements matrix consists of six measured variables, the maximum number of components will be 27. If a higher number of components is defined in the MATLAB-code, the PLS-regression does not work.

Still, even if we have 9 (or 27) number of components to use, it is not necessary or even desirable to use all of them. Usually, there is a point where adding extra number of component to use in the PLS-regression will not improve the model. In fact, if too many ncomp are used we risk modeling noise and not the general relationship between the measurements and the cost function.

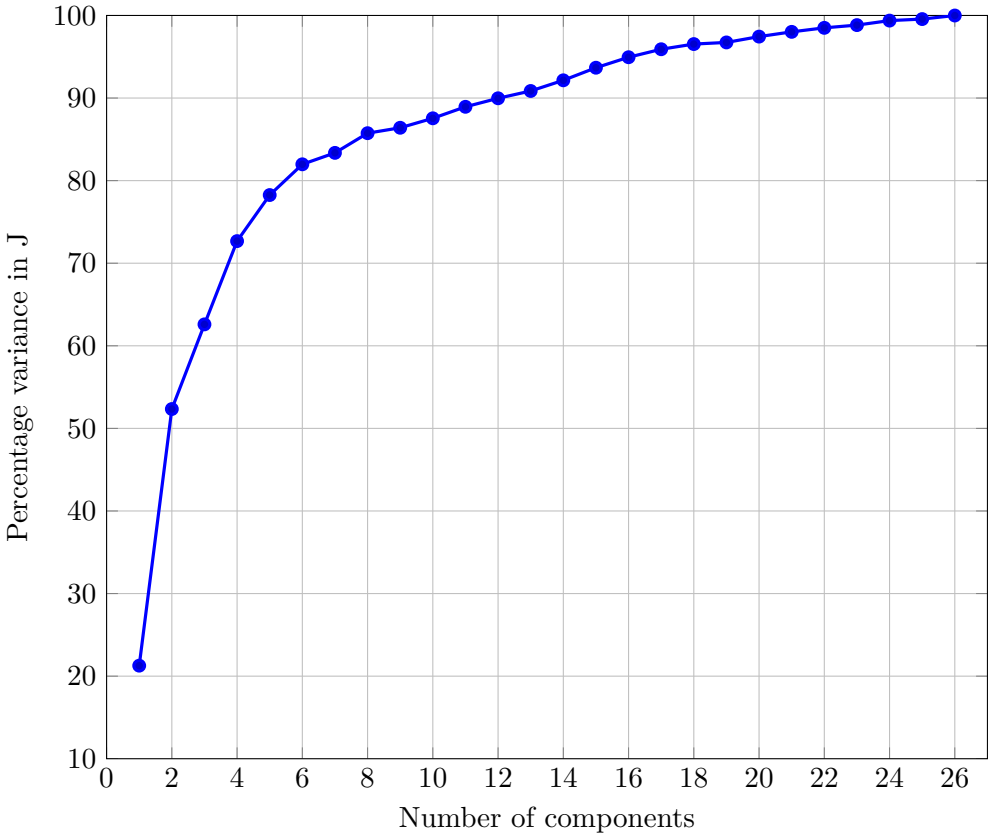
4.3.1 Using the percentage variance explained in J

There are different approaches to decide how many number of components to use. One example is a line plot. It shows the percentage variance explained in the cost function, or in other words, how much of the cost function is explained by the model. For example, when this value is 100% everything is explained by the model. The more number of components you use, more directions in the data is explained. When you apply the maximum number of components, all the directions are taken into account and the model explains the cost function 100 % . The percentage variance is given by the PLS-regression in MATLAB.

The percentage variance for our two cases are given in Figure 4.1.



a) Case A



b) Case B

Figure 4.1: Percentage variance in J, the cost function, for a) Test-case A with 3 measured variables, and b) Test-case B with 6 measured variables

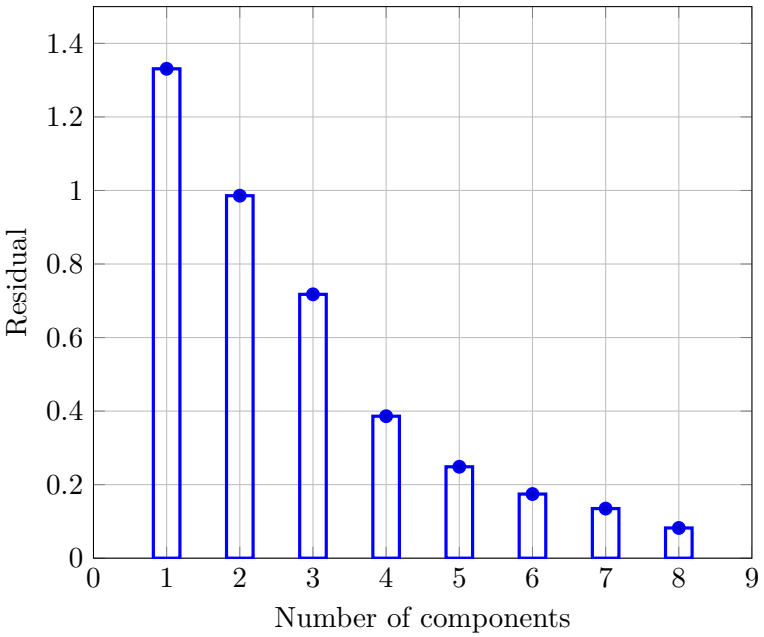
4.3.2 Using beta to estimate the cost function

A second way to evaluate how many number of components to use, is by utilizing the beta parameter from the PLS-regression to estimate the cost function. It is estimated for all the samples except for one data set which is left out of the modeling. The cost function measurement not used in the modeling is then compared to all the estimated values, and the norm of this residual is used as comparison. This procedure is carried out for all number of components possible to use for the case.

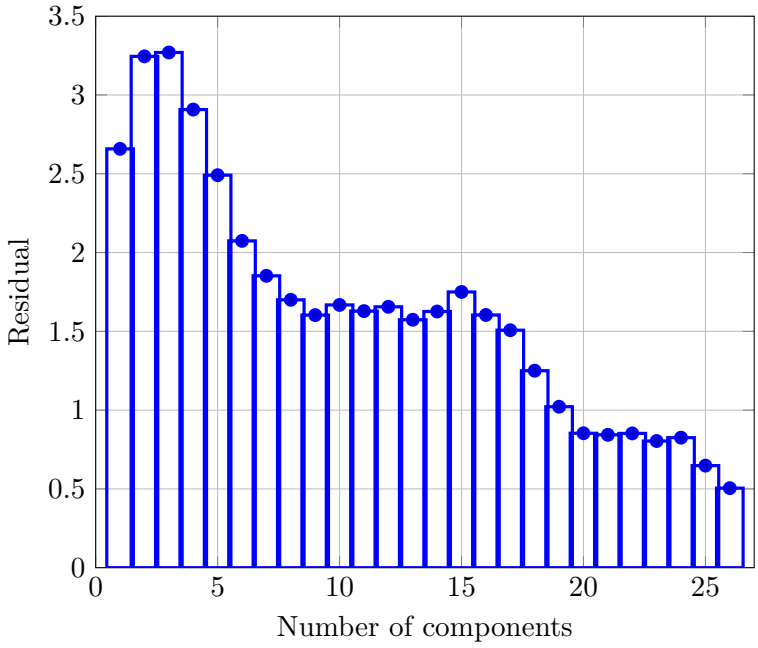
For example in a case with 100 samples. The cost function model, $J = \begin{bmatrix} 1 & Y_{aug}^T \end{bmatrix} \beta$, is estimated using all of the samples but one, f.ex 99 of 100 samples. The sample of the cost function not included in the modeling is used to calculate a residual between the estimated and measured cost function value. If for example sample-series 4 is not included in the modeling, the residual become $J_4^{measured} - J_4^{est}$. The residual is calculated, for all possible number of components, where one sample series is excluded from the modeling. This is run until all the sample-series have been excluded once. For each number of component value, the norm of all the residuals are calculated and used to compare the different number of components. The normal trend is that if a low number of components is applied, the residuals are relatively high. The residuals are decreasing when the number of components increases, but only up to a certain point. At some point the residuals stay close to constant, even if the number of components increases. After this there is no point in increasing the number of components further because it will not add additional information to the model. The PLS-regression scheme will simply start to include noise in the modeling.

This procedure can be used both to check how many number of components to use, and how accurate the cost function is modeled for a specific number of components.

For our two cases it seems that for $n_y = 3$ the ideal will be to use the maximum number of components. Adding one extra direction to model (increasing ncomp) always reduces the residual. The same is found for case B, even if the decrease in the residual is not as even as for case A (Figure 4.2).



a) For $\mathbf{n}_y = 3$



b)For $\mathbf{n}_y = 6$

Figure 4.2: Comparing the measured cost function to the estimated cost functions from the same data set. Using the beta-matrix to estimate the cost function *not* modeled to the measured value of the cost function in the data set not used in the modeling.

The measured and estimated cost function

A third way to check how well the cost function is estimated is to compare the actual measured value J_m with the estimated value J_{est} (Equation 2.23). This is an efficient way to check how well the cost function is modeled. Perfect modeling should yield $J_m = J_{est}$, and a model is considered good if this residual is low.

$$J_{est} = [1\ Yaug']\beta$$

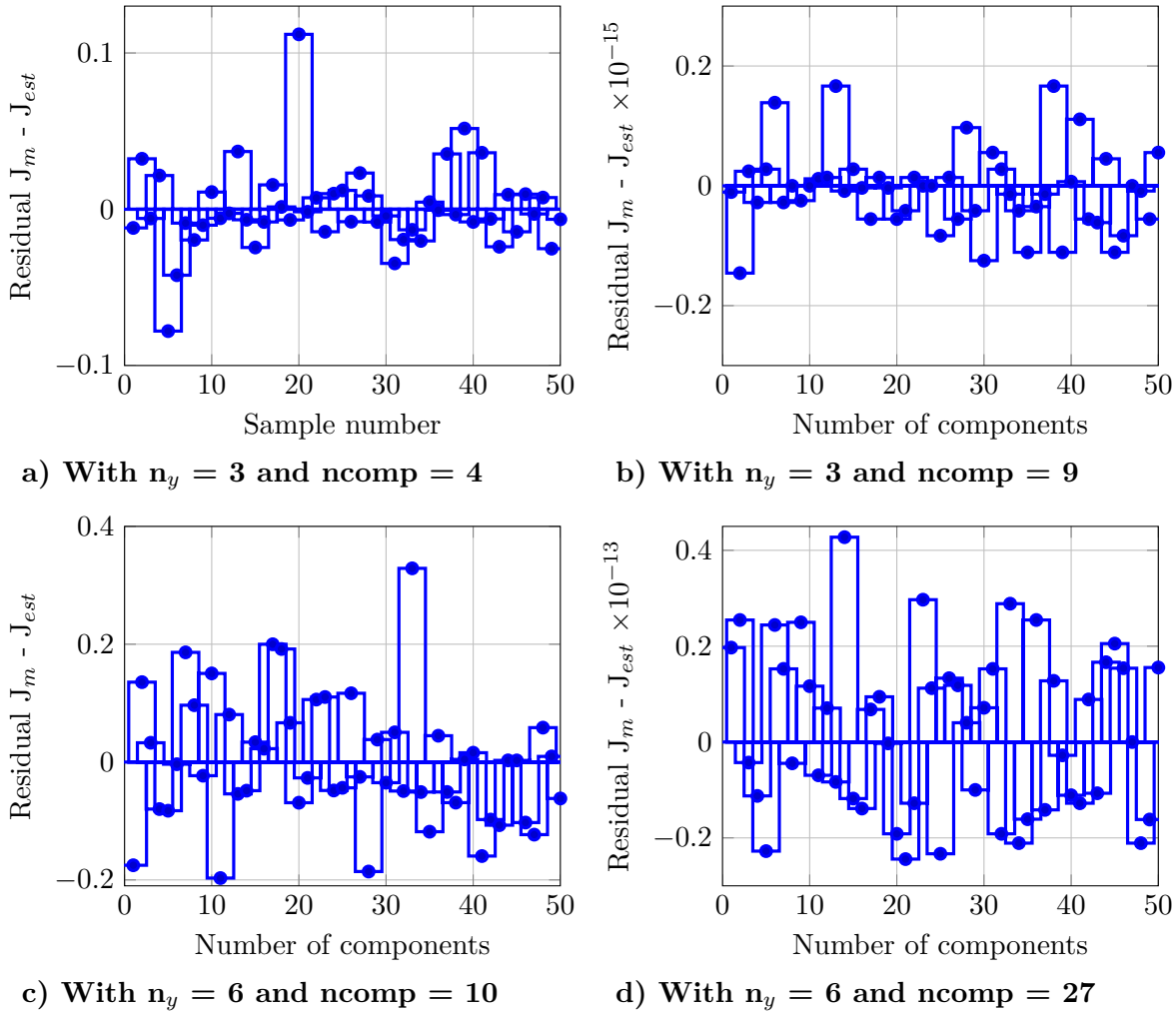
$$\text{residual} = J_m - J_{est}$$

We wanted to research the effect the number of components have on the model accuracy. We therefore calculated the residual by estimating the cost function using different number of components, but for the same data set. For the case with three measured variables (Case A) we used 4, 7, 8 and 9 ncomp. And in the case with six measured variables (Case B) we estimated the cost function using 10, 23, 26 and 27 number of components.

The residuals for all the four ncomp values are given in Table 4.2. In both cases the model accuracy is significantly improved when the number of components is increased. The residuals for case A with ncomp=4 and 9 and for case B with ncomp=10 and 27 are shown in Figure 4.2. The total residual is the sum of all the bars.

Table 4.2: The total residual; J_m - J_{test} of 50 data samples, for test case A with ncomp = 4 and 9, and case B with ncomp = 10 and 27.

Test case	ncomp	Total residual	Reason for choosing this ncomp
Case A:	4	0.018	Testing a low number
	7	0.0071	Best choice according to the percentage variance
	8	0.0041	No change in the residual when adding an extra ncomp
	9	3.86×10^{-17}	Maximum number of component allowed
Case B:	10	0.084	Testing a low number
	23	0.027	Best choice according to the percentage variance
	26	0.003	No change in the residual when adding an extra ncomp
	27	1.49×10^{-15}	Maximum number of component allowed



a) With $n_y = 3$ and ncomp $= 4$ **b) With $n_y = 3$ and ncomp $= 9$**

c) With $n_y = 6$ and ncomp $= 10$ **d) With $n_y = 6$ and ncomp $= 27$**

Figure 4.3: The residual between the measured value of the cost function and the estimated value from modeling the relationship between the cost function and the measurements. To the left, a smaller number of components are used, while to the right the maximum number of components are used.

4.3.3 Changing the data set

The results above are so far only valid for this specific test case. In order to say something more general, the same research tests must be made several times with different randomly chosen parameter and data sets.

After running the same research scheme three times, where each time new parameters and data sets were calculated, we found that using the maximum number of components always gave the best results in terms on model accuracy (Table 4.3). The percentage variance explained in J was 100% for the maximum number of components, and the residual J_m - J_{test} practically zero.

Likewise for all the three test runs, using the second highest number of components turned out to be a poor choice compared to using the maximum number of components.

Table 4.3: The total residual; J_m - J_{test} of 50 data samples, for test case A with ncomp = 4 and 9, and case B with ncomp = 10 and 27. The total residual is found for three different test-dummy cases (1, 2, 3), where the randomly chosen parameters and the data set are changed each time.

Test case	ncomp	Total residual (1)	Total residual (2)	Total residual (3)
Case A:	4	0.018	0.0036	0.24
	8	0.0041	0.040	0.024
	9	3.86×10^{-17}	9.16×10^{-17}	1.33×10^{-14}
Case B:	10	0.084	0.061	0.079
	26	0.003	0.0054	0.024
	27	1.49×10^{-15}	8.54×10^{-16}	3.77×10^{-15}

4.4 Discussion of results from the "dummy" test case

The aim with this chapter was to illustrate the idea and theory behind the data-based method, as well as give a brief introduction to some of the complexity of the regression tool use in the calculation. Hence, the discussion for this chapter will be short, only summarizing the observations made.

The total residuals given in Table 4.2 indicates that the number of components suggested by using the percentage variance (Figure 4.1) gives a relatively high loss compared to using for example the maximum number allowed for the case. Furthermore, the improvement by using the maximum number of components compared to using the second highest value is extensive. When the model validation was repeated for three different data set, the same trend was found all three times; using the maximum number of components gave a significantly better model than using the second highest value or less. In this case we therefore conclude that it is ideal to use the maximum number of components. However, in this case the measurements did not include measurement noise.

In this dummy case, using 6 measured variables gave in general a higher residual between the measured and estimated cost function compared to the case with $n_y = 3$ (Figure 4.3). This indicates that using few measured variables in the PLS-regression might give a better model than the case where many variables are used in the estimation. A reason for this could be that with more variables, there will also be more relations to estimate. For the case with $n_y = 3$ there are only (size($J_{yy} = 3 \times 3 = 9$)) nine relations that must be modeled, which are the elements in the J_{yy} matrix. Whereas the case with $n_y = 6$ there are (size($J_{yy} = 6 \times 6 = 36$)) 36 relations to estimate. The possibility of making a mistake is therefore higher in the case with many measured variables, and the modeling is more difficult.

This chapter did not include any loss calculations for the different H-matrices. When trying to calculate loss by using the formula given in Equation 2.13, the loss values varied extensively depending on the data set and random variables used in the calculation. And in most of the cases the we could not find the loss value for the exact local method due to calculation fails. The loss calculations will therefore be presented only for the more realistic test-cases in the next chapters.

The data generated in the dummy case did not contain measurement noise. As data never can be collected completely without measurement noise, a more realistic test cases should include measurement noise in the data. This is therefore done in the two upcoming test cases. First we used an evaporator process to generate data to find the H-matrix. Thereafter we use a slightly more complex case-study with a CSTR-reactor and distillation column with recycle. In the next two test cases we will use the procedures described in Section 4.3.2 to identify the ideal number of components. To investigate the model accuracy we will use the approach described in Section 4.3.2.

Chapter 5

Test-case one: An evaporator process

In order to test the data-based method we will use a case study. The process used as the first case study is an evaporator model described in "*Applied Process Control - A case study*" by R.B Newell and P.L Lee [20], and modified by Kariwala et al. [15]. This evaporator process has been studied by several authors, such as Govatsmark and Skogestad (2001) [19], Cao (2003) [18] and Alstad (2005) [17]. The focus of the studies was mainly to find the best possible self-optimizing structure. We will use the modified version described in the article by Kariwala et. al [15]. Key parameters and values are taken from this article and reprinted here.

The process flow sheet is shown in Figure 5.1. The feed stream is a dilute liquid stream in which we want to increase the concentration. This is done by evaporating the solvent in the feed stream in a vertical heat exchanger. We want to maximize the operational profit, and the the objective function is defined as the negative profit function for the process given in Equation 5.1 [15]. This is therefore a minimization problem. The three first terms are expenses related to steam, water and pumps, the fourth term is the cost for raw materials (F_1) and the last term is the income from the product (F_2).

$$J = 600F_{100} + 0.6F_{200} + 1.009(F_2 + F_3) + 0.2F_1 - 4800F_2 \tag{5.1}$$

After optimizing the process Kariwala found that the minimum negative profit at the nominal point was \$-582.23/h. This value will be the reference value when evaluating an acceptable loss.

The inputs to the process are the cooling water flow rate (F_{200}) and the feed flow rate (F_1). The disturbances are feed composition (X_1), feed temperature (T_1) and the inlet temperature of the cooling water (T_{200}).

$$u = [F_1 \quad F_{200}]$$

$$d = [X_1 \quad T_1 \quad T_{200}]$$

The model equations are given in the article by Kariwala et al. [15]. The initial values for the different parameters in the process are given in R.B Newell and P.L Lee [20].

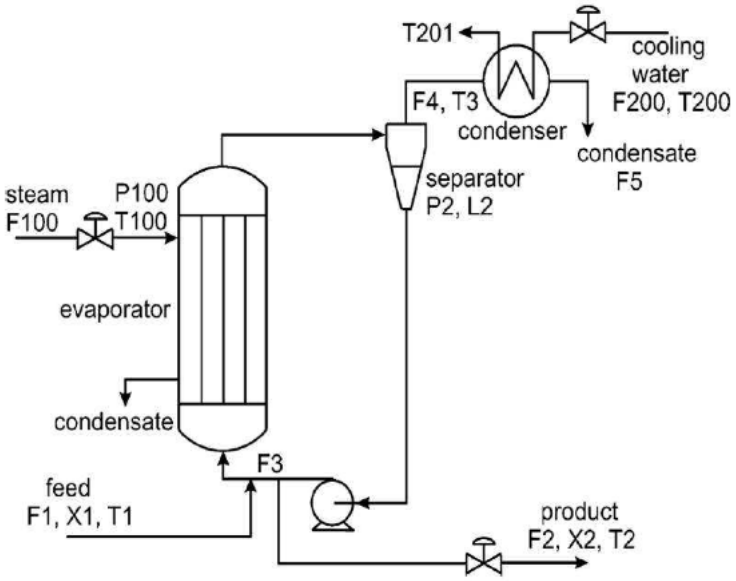


Figure 5.1: Flowsheet of the evaporator process as given in "*Applied Process Control - A case study*" by R.B Newell and P.L Lee [20],

Degrees of freedom

The degree of freedom analysis carried out by Kariwala et al. [15] concludes that in this process there are in total 8 degrees of freedom. Where three are disturbances, two are used to control active constraints and one is without a steady state effect. This leaves two degrees of freedom that can be used as self-optimizing control variables.

There are in total 10 available measured variables, which the two self-optimizing control variables can be chosen from. We can either choose two single variables or construct a linear combination of measurements, and keep the combination constant at an optimal value. The 10 measurements are given in matrix y below, as well as the gain matrices, G^y and G^y_d, and the second derivative matrices, J_{uu} and J_{ud} required to evaluate the process. All the matrices are taken from the process description given by Kariwala et al. in his article "*Local Self-Optimizing control with Average Loss Minimization*" [15].

$$y = \begin{bmatrix} P_2 & T_2 & T_3 & F_2 & F_{100} & T_{201} & F_3 & F_5 & F_{200} & F_1 \end{bmatrix}^T$$

$$G^y = \begin{bmatrix} -0.0930 & 11.678 \\ -0.0520 & 6.5590 \\ -0.0470 & 5.9210 \\ 0.0000 & 0.1410 \\ -0.0010 & 1.1150 \\ -0.0940 & 2.1700 \\ -0.0320 & 6.5940 \\ 0.0000 & 0.8590 \\ 1.0000 & 0.0000 \\ 0.0000 & 1.0000 \end{bmatrix} \qquad G_d^y = \begin{bmatrix} -3.6260 & 0 & 1.9720 \\ -2.0360 & 0 & 1.1080 \\ -1.8380 & 0 & 1.0000 \\ 0.2670 & 0 & 0.0000 \\ -0.3170 & -0.0180 & 0.0200 \\ -0.6740 & 0 & 1.0000 \\ -2.2530 & -0.0660 & 0.6730 \\ -0.2670 & 0 & 0.0000 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.2}$$

$$J_{uu} = \begin{bmatrix} 0.0060 & -0.1330 \\ -0.1330 & 16.7370 \end{bmatrix} \qquad J_{ud} = \begin{bmatrix} 0.0230 & 0.0000 & -0.0010 \\ -158.373 & -1.1610 & 1.4830 \end{bmatrix} \tag{5.3}$$

The noise-matrix and the disturbance noise matrix are both taken from the article by Kariwala et.al and reprinted here. The disturbance in the composition varies ± 5%, and the temperatures varies ± 20% from the initial value.

$$W_n = \text{diag}[1.285 \quad 1 \quad 1 \quad 0.027 \quad 0.189 \quad 1 \quad 0.494 \quad 0.163 \quad 4.355 \quad 0.189]$$
$$W_d = \text{diag}[0.25 \quad 8 \quad 5]$$

5.0.1 Generating data samples

Data, for the inputs and disturbances, is created by adding or subtracting small values from the nominal value of the variable. The sizes of the fluctuations in the variables are given in Table 5.1. The percentage change is the upper and lower limit of how much the variable can change. The variation is therefore somewhere in between these percentages. For example, as seen in Table 5.1, the variation in the feed composition is given as ± 5%. This means, that the variation in the feed composition is *maximum* 5% of its nominal value. Data generated for 1000 samples is shown in Figure 5.2. The values in the plots are given as deviation variables, i.e the variation from the nominal values. The plots show that the data is randomly spread out and centered. For all cases, 5000 data samples were generated to use in the model estimations.

Table 5.1: The variations in the input and disturbance variables.

Variable	Symbol	Variation %	Nominal Value	Maximum value	Minimum value	Unit
F_{200}	u_1	± 10%	208	228.8	187.2	$\frac{kg}{min}$
F_1	u_2	± 10 %	10	11	9	$\frac{kg}{min}$
X_1	d_1	± 5%	5	5.25	4.75	-
T_1	d_2	± 20%	40	48	32	^{o}C
T_{200}	d_3	± 20%	25	30	20	^{o}C

The input and disturbance matrix will then be given as :

$$ud = \begin{bmatrix} u \\ d \end{bmatrix} = \begin{array}{c} \begin{matrix} ns=1 & ns=2 & \cdots & ns=n \end{matrix} \\ \begin{bmatrix} u_1^1 & u_1^2 & \cdots & u_1^n \\ u_2^1 & u_2^2 & \cdots & u_2^n \\ d_1^1 & d_1^2 & \cdots & d_1^n \\ d_2^1 & d_2^2 & \cdots & d_2^n \\ d_3^1 & d_3^2 & \cdots & d_3^n \end{bmatrix} \end{array}$$

The measurements are calculated using the the gain matrix given in Equation A.7, according to the following equation:

$$Y_m = G_p \times ud = \begin{bmatrix} y_1 & \cdots & y_1^n \\ y_2 & \cdots & y_2^n \\ \vdots & \ddots & \vdots \\ y_{10} & \cdots & y_{10}^n \end{bmatrix}$$

Where G_p is the process gain matrix. Each column in Y_m is a new data sample set, in a case with *ns* samples T_m will therefore have *ns* columns.

The cost function is calculated using the second derivative matrices, and it is given as:

$$J_m = \begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_{ns} \end{bmatrix}$$

5.0.2 Evaluation of the data based method

We now have all the parameters and variables needed in order to test the data based method against other methods such as the null space method and the exact local method. In this test case we calculate the H-matrix using the three different approaches, and compare the loss (Equation 2.13b). The average loss was in the theory chapter given as:

$$L_{avg} = \frac{1}{2}\|M\|_F^2$$

Where:

$$M = J_{uu}^{\frac{1}{2}}(HG^y)^{-1}H[FW_d \quad W_{n^y}]$$

To test the data based method we will calculate H using both ten and five of the available measurements. Five is the lowest number of measured variables that can be used in this case, since $n_y \geq n_u + n_d$ which in this process is five. The ten measurement are:

$$y = \begin{bmatrix} P_2 & T_2 & T_3 & F_2 & F_{100} & T_{201} & F_3 & F_5 & F_{200} & F_1 \end{bmatrix}^T$$

and the five measurements we chose to use are:

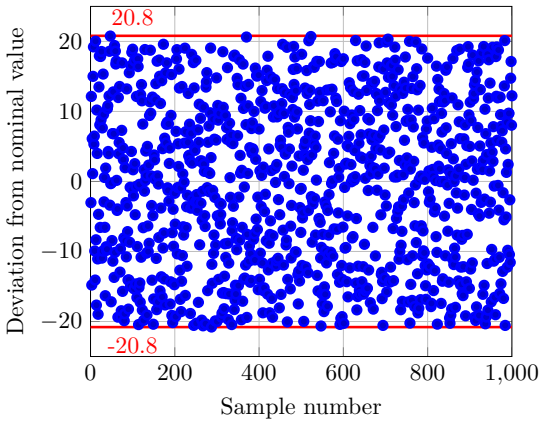
$$y = \begin{bmatrix} F_2 & F_3 & P_2 & F_{200} & T_{201} \end{bmatrix}^T \tag{5.6}$$

These are the five measurements which Alstad (2005) found to be the most promising set of measurements when he studied this example [17].

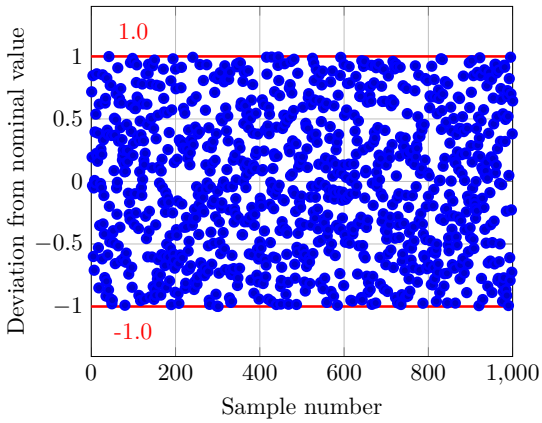
It should be noted that the disturbances are not measured in this process.

When measuring variables in a real process, measurement noise must always be taken into account. To test how well the data based method handles measurement noise we generated data both with and without measurement noise. The performance of the data method was then compared to the exact local and null space method.

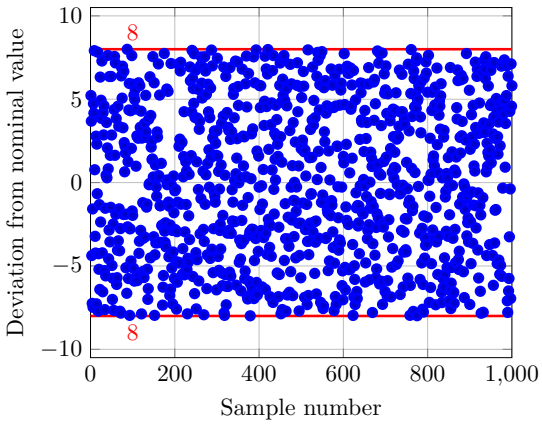
As explained earlier, one of the key parameters in the calculation of the H-matrix is the number of components used in the PLS-regression scheme. Therefore, before calculating H we identify the ideal number of components for this test case, both for 10 and 5 measurements.



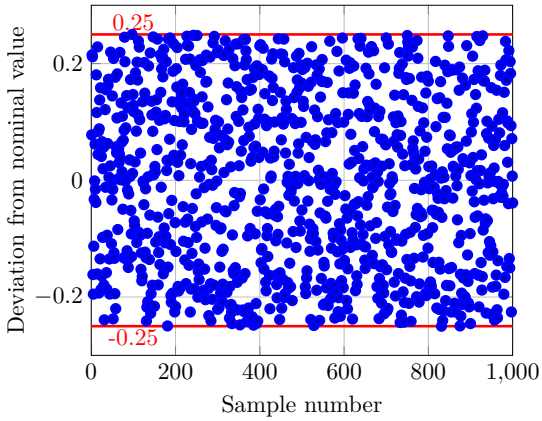
(a) Data in deviation values F_{200}



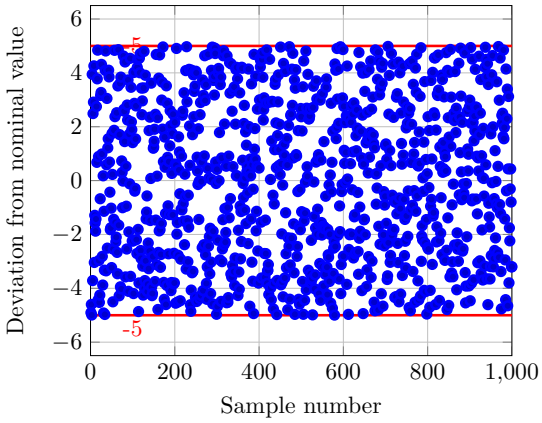
(b) Data in deviation values F_1



(c) Data in deviation values T_1



(d) Data in deviation values X_1



(e) Data in deviation values T_{200}

Figure 5.2: Data in deviation values for the inputs (a and b), and for the disturbances (c,d and e). The true value of the variable is found by adding these values to the nominal value of the variable.

Summary of test criteria for evaluating the data based method:

- Number of measurement: $n_y = 10$ or $n_y = 5$
- Measurement noise: with (y_n), and without (y)
- Number of components: Identifying the ideal number of components

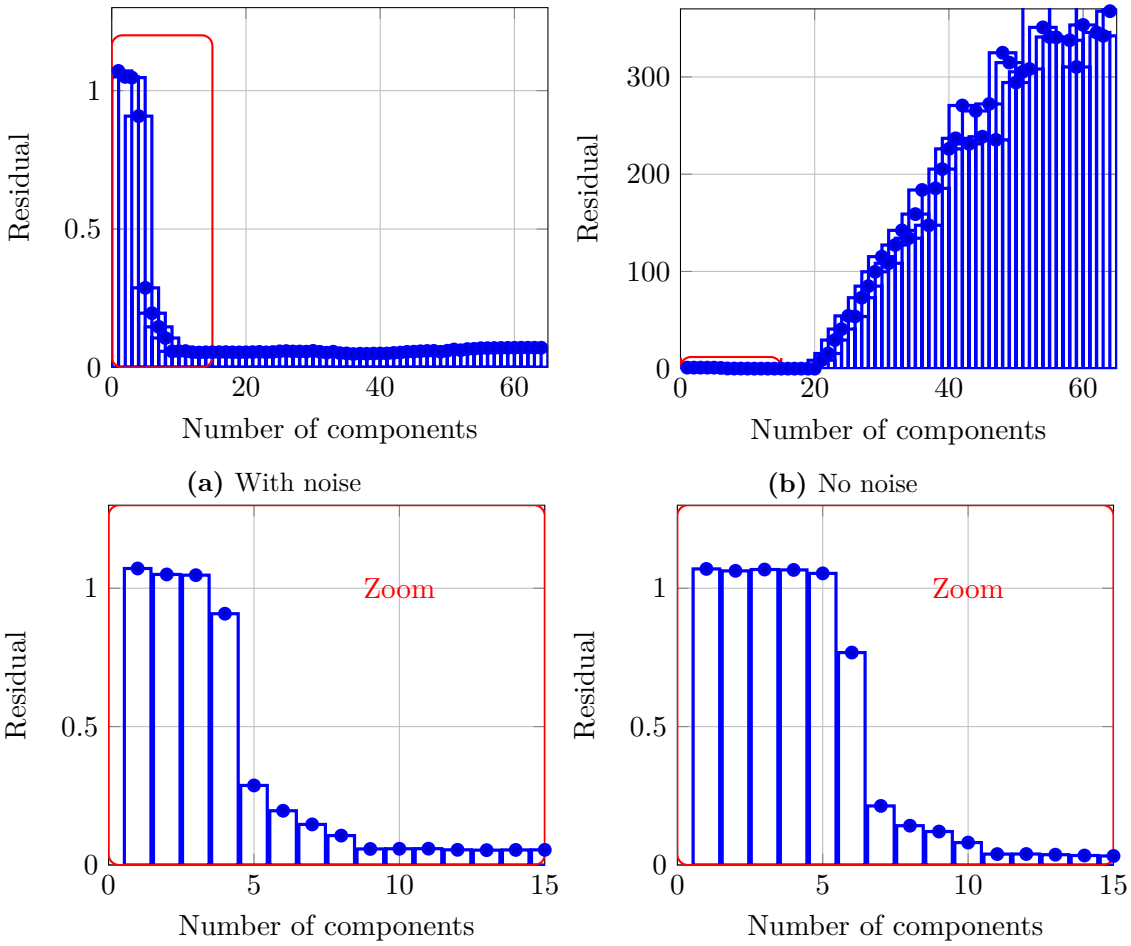
5.1 Results

Deciding the ideal number of components for 10 measurements

When we use 10 measurements Y_{aug} has 65 elements, this means that the maximum number of components will be 65. The computed norm of the residuals for a all number of components is plotted from 1 to 65 components, both in the case with and without noise (Figure 5.3a and 5.3b). To get a better impression of the behavior when using a low number of components we limited the plot to show only the 15 first number of components (Figure 5.3c and 5.3d).

The rapid increase in the residual in the case with 10 measurements and no noise (Figure 5.3b) is assumed to be caused by a numerical error in the calculation. When the plots are limited to cover only the first 15 components, we observe the same trend both in the case with and without noise; that there is little or no change after 10 components.

The simulations for this case were therefore run with 10 components in the PLS-regression.



(a) With noise

(b) No noise

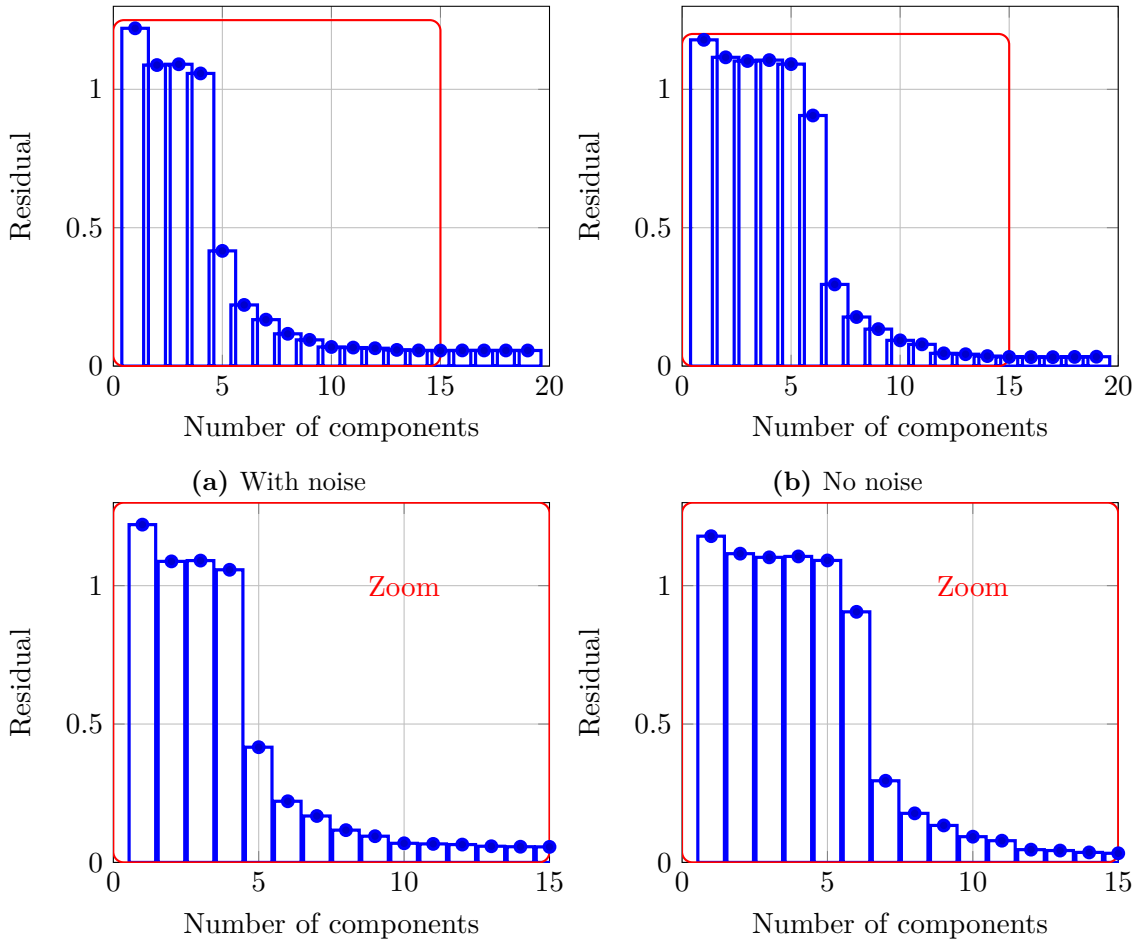
(c) With noise: Showing 15 first components

(d) No noise: Showing 15 first components

Figure 5.3: Deciding number of components for the case with 10 measurements, with and without noise

5.1.1 Deciding number of component with 5 measurements

When we use 5 measurements Y_{aug} has 20 elements, this means that the maximum number of components will be 20. The computed norm of the residuals for a certain number of components is plotted from 1 to 20 components (Figure 5.4), both in the case with and without noise. Also in this case we observe little or no change occurring after 10 number of components. The simulations for this case were therefore run with 10 components in the PLS-regression.



(a) With noise

(b) No noise

(c) With noise: Showing 15 first components

(d) No noise: Showing 15 first components

Figure 5.4: Deciding number of components for the case with 5 measurements, with and without noise

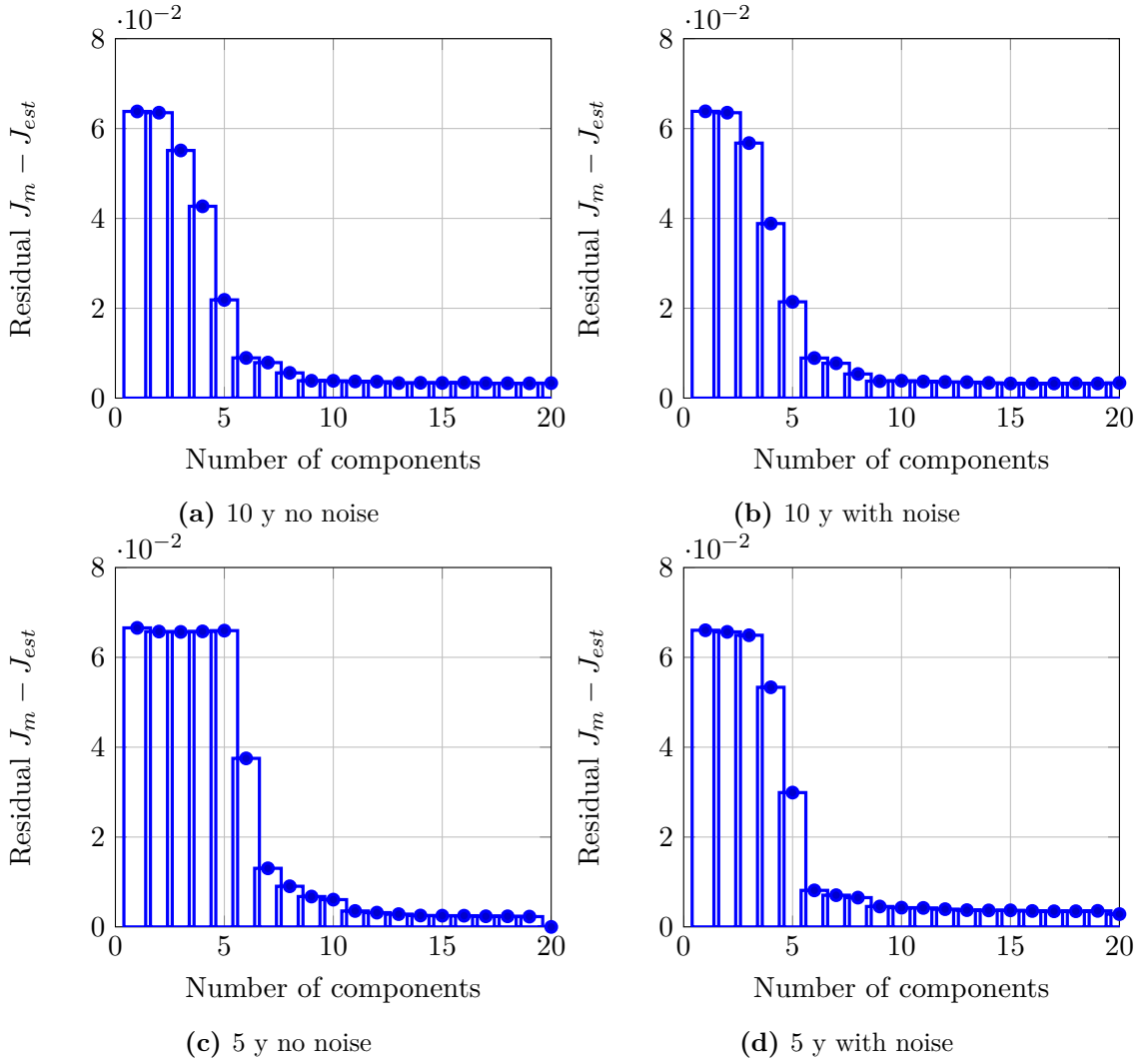
5.1.2 Model validation

The model accuracy was found by calculating the residual between the measured cost function J_m and the estimated cost function using PLS-regression, J_{est}. For each number of component this was done for 1000 different data samples, resulting in 1000 different residuals. The average residual is plotted for all number of components in Figure 5.5. From this we see that the residual is relatively high when we use a few number of components, but the model accuracy increases when we increase the number of components. After a certain point however, the effect of adding an additional component wears off. This means that adding one extra direction in the data does not give any additional information to use in the modeling.

There was no difference in the residuals after 20 components for the case with 10 measured variables. Therefore the plot shows only the 20 first number of components in this case even though there are 65 in total (Figure5.5a and 5.5b).

We see that even for few number of components (1 to 5) the residual is relatively small, and the cost function model rather accurate. Surprisingly, this is also the case when measurement noise is included. The residual decreases faster in the case with 10 measured variables, indicating that a lower number of components is needed in this case to get an accurate model.

In all the cases it seems sufficient to use 6 number of components to get a rather accurate model, and after ncomp=10 there is no noticeable change in the residual when ncomp is increased.



(a) 10 y no noise

(b) 10 y with noise

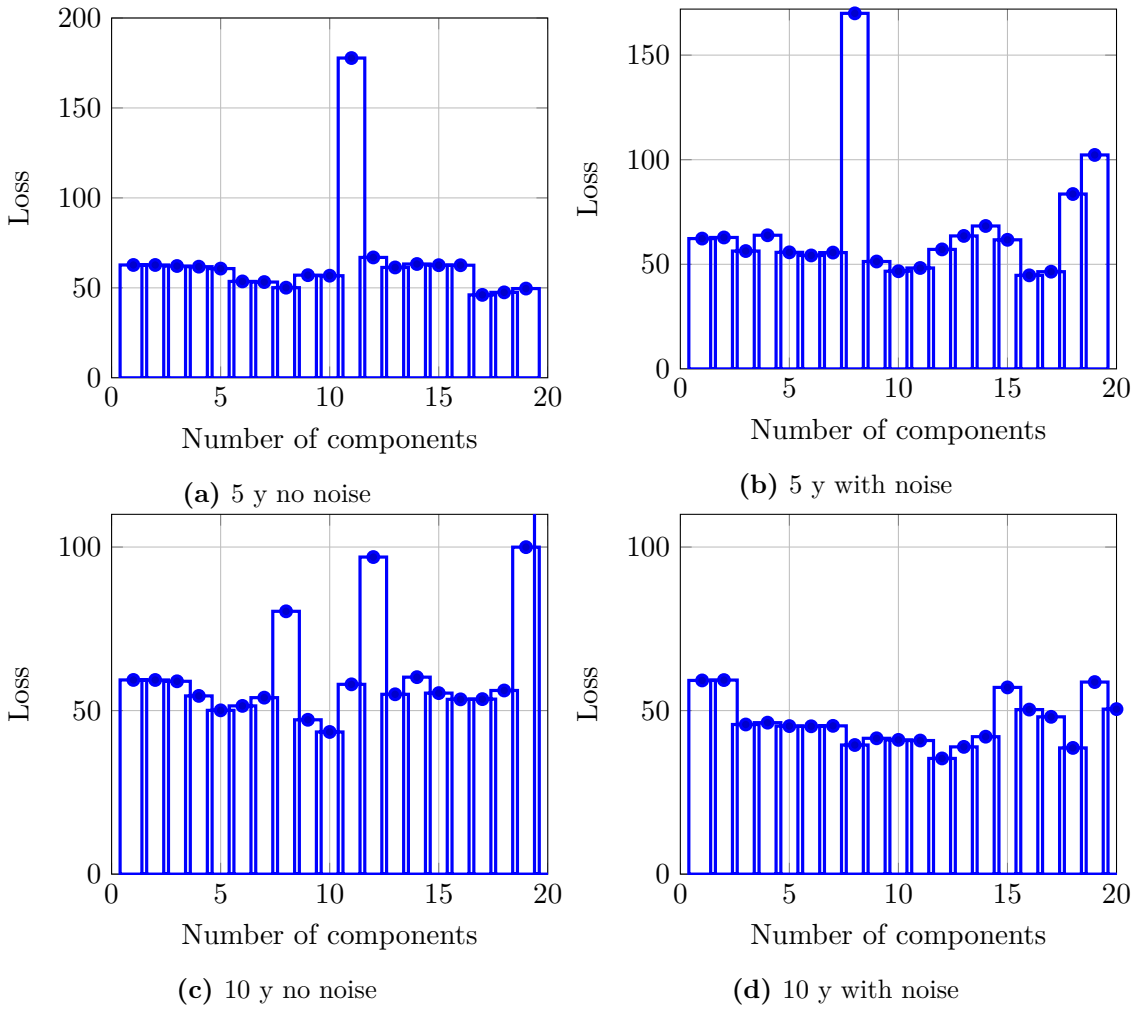
(c) 5 y no noise

(d) 5 y with noise

Figure 5.5: Model validation using the residual $J_m - J_{est}$ to evaluate the model accuracy for the case with 10 and 5 measured variables, with and without measurement noise in the data set used in the modeling

5.1.3 Loss calculation with different number of components

To further investigate how the result changes when the number of components is changed we calculated the H-matrix using all available number of components in turn. The result is presented for the 20 first components in Figure 5.6. From the results shown in these plots it seems like the loss is generally somewhere between 50 and 100, regardless of the number of components used in the PLS-regression. The overall impression is that using 10 measurement with measurement noise in the data samples, gives the lowest loss (Figure 5.6d)



(a) 5 y no noise

(b) 5 y with noise

(c) 10 y no noise

(d) 10 y with noise

Figure 5.6: The loss (Equation 2.13) when 1 to 20 number of components are used in the PLS-regression to calculate the H-matrix.

Loss calculations for ncomp=10 and ncomp=6

The average loss (Equation 2.13) for each of the H-matrices calculated are all given in Table 5.2. The number of components used in these calculations are 10 and 6 for all cases.

Table 5.2: Loss calulated for all the different H-matrices with ncomp=10 and ncomp=6

	Data, ncomp = 6		Data, ncomp = 10		Exact local		Null space	
	with noise	no noise	with noise	no noise	with noise	no noise	with noise	no noise
n_y=10	51.4	54.9	41.05	43.46	7.55	1.03	226.7	0
n_y=5	55.01	55.2	46.72	47.38	9.25	1.55	11.2	0

The worst and the best outcome of the data-based method is marked with green and red in Table 5.2. From the figures we see that using 10 measured variables results in a lower loss then the case with 5 measured variables. Additionally, the loss is lower for the cases where the data includes measurement noise.

5.1.4 The H-matrices:

The H-matrices found for the different cases are presented below. They are presented to give the reader a feeling of how they look. In the next test cases the H-matrix itself will no longer be given. This is because we are primarily interested in the result by using the H-matrix, i.e loss calculations, rather than the matrix itself.

10 measurements

$$H_{data}^{w/n} = \begin{bmatrix} -0.0002 & -0.0002 & -0.0003 & 0.071 & -0.0019 & 0.0004 & -0.0004 & -0.0021 & 0.012 & -0.0002 \\ 0.041 & 0.066 & 0.071 & -9.79 & 0.66 & -0.048 & 0.10 & 0.78 & 0.15 & 0.37 \end{bmatrix}$$

$$H_{data} = \begin{bmatrix} -0.0001 & -0.0001 & -0.0001 & 0.0055 & -0.0008 & 0.0001 & -0.0001 & -0.0010 & -0.13 & -0.0006 \\ 0.0084 & 0.013 & 0.015 & -0.56 & 0.097 & 0.015 & 0.017 & 0.12 & 0.11 & 0.078 \end{bmatrix}$$

$$H_{exl} = \begin{bmatrix} 0.0013 & 0.0014 & 0.0012 & -0.92 & -0.12 & 0.0023 & -0.022 & -0.33 & 0.042 & -0.27 \\ -0.070 & -0.070 & -0.062 & -8.64 & 3.91 & -0.20 & 1.05 & 8.21 & -0.91 & 5.93 \end{bmatrix}$$

$$H_{ns} = \begin{bmatrix} -0.0080 & -0.0045 & -0.0041 & -0.0253 & -0.0075 & 0.0226 & 0.0020 & -0.0061 & 0.0070 & -0.0315 \\ 13.33 & 7.01 & 6.38 & -241.08 & -54.05 & -59.64 & 32.33 & 15.49 & -2.85 & -225.59 \end{bmatrix}$$

5 measurements

$$H_{data}^{w/n} = \begin{bmatrix} -0.0016 & 0.042 & 0.0032 & -0.0028 & -0.0280 \\ 0.27 & -4.76 & -0.012 & 0.56 & 0.29 \end{bmatrix}$$

$$H_{data} = \begin{bmatrix} -0.0002 & -0.0057 & -0.0004 & -0.0003 & -0.10 \\ 0.024 & 0.74 & 0.053 & 0.041 & 0.070 \end{bmatrix}$$

$$H_{exl} = \begin{bmatrix} -0.0008 & -2.09 & 0.040 & -0.066 & 0.033 \\ -0.053 & 21.20 & -1.13 & 2.18 & -0.69 \end{bmatrix}$$

$$H_{ns} = \begin{bmatrix} -0.017 & -0.059 & 0.032 & -0.0000 & 0.0074 \\ -1.21- & 481.25 & -7.98 & 17.59 & -0.43 \end{bmatrix}$$

5.2 Discussion - evaporator process

5.2.1 Model validation

The ability to handle correlated measurements and measurement noise is one of the reasons to use PLS-regression [9]. The ability to identify a good model even with measurement noise in the data is shown in Figure 5.5. The residual $J_m - J_{est}$, where J_m is the measured value of the cost function and J_{est} is the estimated value, is of the same order of magnitude for the data set with and without noise. This indicates that the model is well estimated also in the cases where we add measurement noise to the data.

We see that the model is slightly more accurate in the case with 10 measured variables, in addition the residual decreases somewhat faster in that case. The changes in the residuals after 10 number of components, in both the cases with 10 and 5 measured variables, are however very small. Due to this, it seems that the number of variables used in the modeling has a minor effect. We recon more parallel test cases must be run to be able to say anything final, and these findings are merely indicative.

5.2.2 Number of components

To decide the ideal number of components to use in the PLS-regression in this test case, we used the approach described in the theory section 4.3.2. Where we use all data samples except for one in the modeling, and then use the beta matrix to estimate the cost function from the data set not included in the modeling. This is repeated until all data samples have been excluded once, and the norm of the residuals is found for all number of components. In Figure 5.3 and 5.4 we see that in most cases here there is little or no change in the residual after 10 number of components.

An interesting observation is made by comparing these two figures (Figure 5.3 and 5.4) to the model validation plot in Figure 5.5. In the latter figure we see that the model improves significantly in terms of the residual defined $J_m - J_{est}$, after 6 number of components. And shows basically no change in the residual after 10 number of components, the same point where the bars begin to level out in Figure 5.3 and 5.4.

In terms of model accuracy, including 10 directions in the modeling, i.e using ncomp=10, seems like a good choice.

5.2.3 Loss calculations

As discussed, we saw an effect on the model accuracy when we used different number of components, and found ncomp=10 to be a good choice. The effect is, however, not as clear in terms of loss calculations. In this case-study we estimated the H-matrix using 1 to 20 number of components for the same data set. Thus, the only factor that changed between each of the estimations was the number of components. This was done for both the case with 10 and 5 measurements, with and without noise.

The result is shown in Figure 5.6, and from this it seems that the loss calculated using Equation 2.13b is generally the same regardless of how many number of components we use in the estimation of H. We observe however, some spikes in the loss for certain number of components. We reran the generation of data, the H-matrix estimation and the loss calculation a few times, and each time these spikes seemed to occur at different number of components. With the knowledge we posses at this moment we are not able to explain why these spikes occur. It is not likely to be because of the measurement noise, since these spikes also occur in the case without measurement noise. Of course, it could be due to a numerical error or calculation fail.

From the four plots in Figure 5.6 we see a trend that using 10 measurements with measurement noise, gives the best result in terms of loss. However, as explained, we reran the simulations a few times, and did observe a slight change in the loss when the data set used as a basis was changed. This points to that the outcome from the method depends on the actual data set used in the estimation, and not only the preconditions and parameters used in the modeling. This theory is back up by the fact that the location of the spikes in the loss keeps changing when we change the data set. If there were more time, it would be interesting to run this simulation enough times to get a representative average. In addition we would like to study the data set itself and the estimated J_{yy}-values more closely, to see if the spikes and the loss variations could be explained from the data.

The spikes of higher loss for some number of components are not found in the model validation. The residual $J_m - J_{est}$ is decreasing evenly as the number of components is increased (Figure 5.6). This could indicate that even with a good model of the cost function, the H-matrix estimated from the same modeling scheme will not perform well as a combination matrix for control purposes. Model accuracy is important, however it seems that a good model does not necessarily assure a well working linear combination of the measured variables in control purposes.

Loss calculations with a fixed the number of components

The loss from using the exact local method, the null space method and the data-based method with ncomp=6 and ncomp=10 were all given in Table 5.2. The same data set was used as a basis for the estimation in the data-based method so that only the number of components were changed. We chose 6 because the model accuracy improved significantly after ncomp=6 in most of the cases. And as explained above, we chose 10 number of components because the model accuracy seemed to be unaffected by the number of components used after 10, in addition there was little or no change in the norm of the residual after 10 (Figure 5.3 and 5.4).

From the loss values presented in Table 5.2 we see that both in the case with ncomp=6 and ncomp=10, using 10 measured variables with measurement noise gives the lowest loss. And the case with 5 measured variables without measurement noise gives the highest loss. This indicates that using many measured variables gives a better result both in terms of model accuracy and loss. Moreover it seems like including measurement noise improves the result.

As expected the null space method gives zero loss in the case with no measurement noise, but performs poorer when measurement noise is included. However, despise of measurement noise the null space method is actually better than the data-based method for 5 measured variables. The result shows that the exact local method is superior to the data method in all the cases. However, this approach is model based, and depending on a well described problem like this evaporator process. All the parameters needed in order to calculate the H-matrix are in this case known. The question is, if we did not have a model describing this process and using the exact

local method would not be an alternative, is the loss found by the data-based method acceptable small? We see that the loss from the data-based method is roughly 10 times worse than for the exact local method. In addition the loss is about one tenth of the nominal optimal value, which by Kariwala was given to be \$582/h. This can be taken into account when evaluating the combinations of variables as a self-optimizing variable. In the end, it will be up to the plant operators to decide.

Chapter 6

Test-case two: CSTR and distillation column with recycle

The second test case is a plant consisting of a CSTR-reactor where A $\rightarrow$ B, and the product stream from the reactor is the feed to a distillation column. In the column un-reacted A is separated from B and sent back to the reactor through a recycle stream consisting mostly of A. The desired product is B and is the bottom product from the distillation column. The column has 22 stages in total, and the stream from the reactor is fed into the column at stage 13. The process has been studied by Skogestad et. al (2003) [12], however we use the process as described in an exercise given for in an Advanced Process Control course [11].

6.1 Process description

Degree of freedom analysis

To find the number of degrees of freedom we use the valve counting technique (Table 6.1). As seen in Figure 6.1 there are 7 valves in the plant. There are two constraints which must be satisfied at all times, the product quality (x_b) and the reactor hold-up (M_r). After optimizing the process with no disturbances, using *fmincon* in MATLAB, we find that the two constraints are active.

$$x_b = 0.0105$$
$$M_r = 2800 \text{ mol}$$

Even though the levels in the condenser and the reboiler are not constrained they must be controlled, and means each level control uses one valve. For safety reasons we must have pressure control in the top of the column. In addition, the feed is the throughput manipulator and given in this case.

This leaves us with one degree of freedom which can be used to optimize the process.

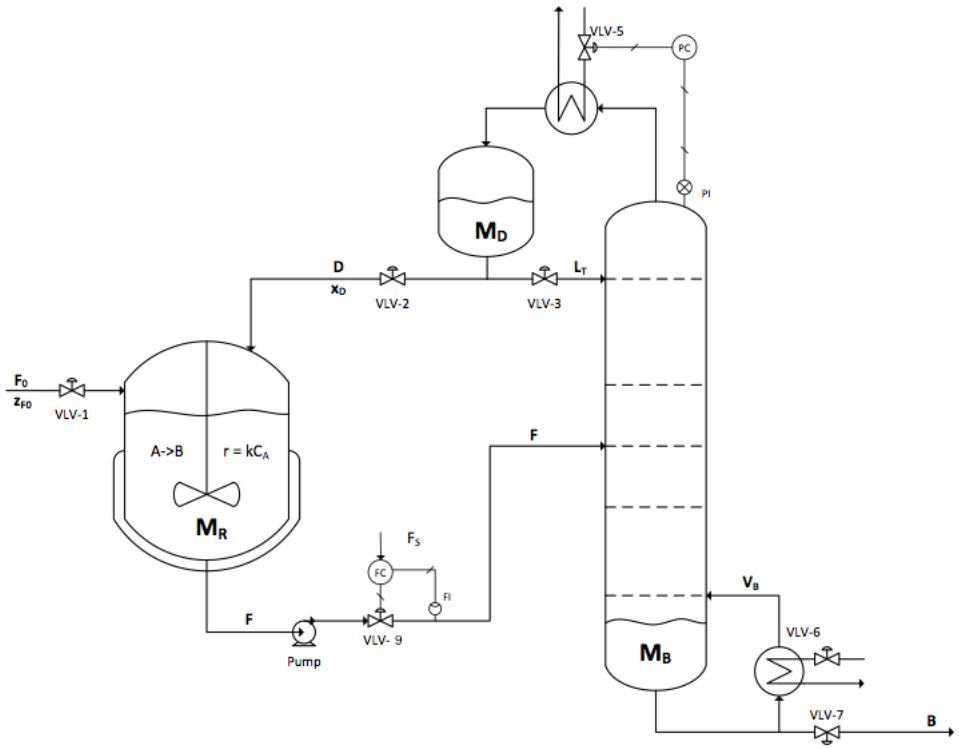


Figure 6.1: Process plant with a CSTR-reactor and a 22 stage distillation column with recycle [11].

Table 6.1: Degree of Freedom analysis

	Number of valves	7
-	Active constraints	2
-	Level control	2
-	Given feed	1
-	Pressure control	1
=	Degrees of freedom	1

Table 6.2: Suggested control pairing for the CSTR-distillation plant with recycle. Variable symbol and valve index is referenced to Figure 6

Valve number	Controlled variable		Manipulated variable	
VLV-5	Pressure in column	P_c	Amount of cooling	
VLV-7	Level in column	M_B	Flow rate bottom product	B
VLV-2	Level of condensate	M_D	Flow rate recycle stream	D
VLV-6	Bottom product composition	x_B	Re-boiler duty	V_B
VLV-9	Reactor level	M_R	Flow rate from reactor to column	
VLV-1	Feed flow rate	F_0	Throughput manipulator	

The cost function

The objective function for this process is the amount of steam used in the re-boiler, the vapor boilup. When the feed changes the amount of steam needed to achieve the desired product composition changes as well. The goal is to minimize the energy consumption without violating the product quality constraint.

The nominal value of the cost function is found by optimization of the problem without any disturbance occurring. The nominal value of the cost function is found, through the optimization, to be:

$$J^* = V^* = 1275.7\text{kmol/h}$$

Inputs, disturbances, measurements and noise

To decide what variable to use as the extra degree of freedom, we first decide on the pairing of the 7 valves. Once a valve is used to control a variable it is no longer free, after controlling all the levels the valve left without a paring will serve as our degree of freedom. A suggested control structure is given in Table 6.2 with reference to Figure 6.1. This leaves the flow rate of condensate back to the column (L) as the degree of freedom. We can therefore manipulate L freely and thereby optimize the process.

There is only one disturbance in the system and that is the feed flow rate, which can vary between ± 10 % of its nominal value. We assume that we can measure 30 variables in the system, the variables are given in matrix x.

$$x = \begin{bmatrix} x_B^{tray\ 1-22} & L_T & V_B & \text{D} & \text{B} & \text{F} & z_F & M_R & F_0 \end{bmatrix}$$

The first 22 measurements in x are composition measurements at each tray in the column. To save time and computational power we choose only to use three or four of these measurements. To investigate how the result changes when we use many or few measurements we calculate the H-matrix using 10 and 5 measured variables. In addition, we both include and exclude the disturbance (in this case F_0) in the measurements. The method is therefore tested with four

different schemes, depending on which of the measured variables are used as a basis for the H-matrix calculation. The cases are presented next.

Measurement matrix for 10 measured variables:

CASE A: the disturbance is not included as a measured variable:

$$y_{10} = \begin{bmatrix} T_8 & T_{13} & T_{18} & T_{22} & L_T & V_B & \text{D} & \text{B} & \text{F} & z_F \end{bmatrix} \tag{6.2a}$$

CASE B: the disturbance is included in as a measurement:

$$y_{10}^d = \begin{bmatrix} T_8 & T_{13} & T_{22} & L_T & V_B & \text{D} & \text{B} & \text{F} & z_F & F_0 \end{bmatrix} \tag{6.2b}$$

Measurement matrix for 5 measured variables:

CASE C: the disturbance is not included as a measured variable:

$$y_5 = \begin{bmatrix} T_8 & T_{22} & L_T & \text{B} & F \end{bmatrix} \tag{6.2c}$$

CASE D: the disturbance is included in as a measurement:

$$y_5^d = \begin{bmatrix} T_8 & T_{22} & L_T & \text{B} & F_0 \end{bmatrix} \tag{6.2d}$$

Each measurement combination is tested with and without measurement noise for the data-based method. The combination matrix is also found by using the exact local method (with noise included in the calculation) and the null space method for these four cases.

The original measurement matrix (x) is given with composition measurements. But instead of composition we use a simple formula to convert the composition measurements into temperatures (Equation 6.3).

$$T(x) = Tb_L x + (1 - x)Tb_H \tag{6.3}$$

Where Tb_L is the boiling point of the light component and Tb_H is the boiling point of the heavy component, in this case component A and B respectively.

This is done because the change in temperature is larger than the same change expressed by composition, and this decreases the possibility for numerical mistakes due to very small numbers.

Gain matrices

The measurement gain matrix is found by doing a small step change in the input value of L, and recording the change in the outputs y. The disturbance gain matrix is found in a similar manner, by doing a small step change in the feed. The gain matrices can be found in Appendix A.

6.2 Evaluating the data method

As explained earlier, the purpose with this thesis is to get a better understanding of the newly developed data-based method, where we use data to find self-optimizing variables. We were interested in investigating how different parameters and conditions affect the method's behavior. In order to understand the data method better the following parameters and conditions were changed:

- The number of data samples 50, 100 and 1000 samples were used.
- The PLS-regression was run with both 5 and 10 number of components.
- Both 5 and 10 measured variables were used in the calculations.
- Including and not including the disturbance as one of the measured variables.
- Using data samples with and without measurement noise.
- Calculating H with other methods and comparing the losses.

To be able to compare the different combinations, the calculated H-matrix was used to control the process. The loss, L, between re-optimizing the process and controlling it for a given disturbance is the measure used to evaluate the performance of the H-matrix.

$$\boxed{L = J_{ctrl} - J_{opt}} \tag{6.4}$$

Where J_{opt} is the cost function value when the process is re-optimized and J_{ctrl} is the cost function value when the H-matrix is used as a self-optimizing parameter. We assume perfect control and implementation error in the control structure was not taken into account in the control simulations. This improves the performance of the control structures found by the exact local and null space method, compared to how they would be in reality where implementation errors are inevitable.

The process was both re-optimized and run using self-optimizing control (the H-matrix), when influenced by a disturbance in the feed. The disturbances were s $\pm 1\%$ and $\pm 5\%$. The loss between re-optimizing and controlling the process was found for all four disturbances. This was done after the cost function was modeled and the H-matrix estimated (or found using exact local and null space method). These disturbances must therefore not be confused with the disturbance measurement used as a basis in the cost function modeling.

The evaluation of the method is summarized below. It was done systematically according to this procedure.

Evaluating the data method by systematically changing different parameters:

1. Set the number of samples wanted as a data basis.
2. Set the number of components

3. Define the data base combination, and in this case we have four combinations to choose from.
   - **Case A:** 10 measured variables, not including the disturbance (F_0) in the measurements.
   - **Case B:** 10 measured variables including the disturbance (F_0) in the measurements.
   - **Case C:** 5 measured variables, not including the disturbance (F_0) in the measurements.
   - **Case D:** 5 measured variables including the disturbance (F_0) in the measurements.
4. Choose if the data should be collected with or without measurement noise.
5. Simulate the process with disturbance ± 1 or ± 5 using the H-matrix calculated from

   First the data method

   Second the exact local method

   Third the null space method
6. Save the loss between controlling the process with the calculated H and re-optimizing it for the given disturbance.

Using an average-loss for the data method

For the data method, we ran the data generation five times, and for each generated data set we estimated the H-matrix and used it to control the process. The loss value$J_{opt} - J_{ctrl}$ was found for all the five runs. The loss given for the data-based method is therefore an average from these five simulations.

For example, for five measurements (not including the disturbance and using data with measurement noise) five different data sample sets with 100 samples in each set were used to calculate five different H-matrices presented below. And depending on the data set used we see that there is a rather clear difference between the H-matrices.

To get a more general and statistical valid result the test should have been run more than 5 times. However, generating data , estimating H and then using it as a control variable were quite time consuming, for this reason the simulations were run only 5 times.

$$
\begin{aligned}
H_{dataset\ 1} &= \begin{bmatrix} 0.0034 & -0.0213 & 0.8310 & -0.0286 & -0.0439 \end{bmatrix} \\
H_{dataset\ 2} &= \begin{bmatrix} 0.0553 & 0.1111 & 1.3381 & -0.5231 & 0.0355 \end{bmatrix} \\
H_{dataset\ 3} &= \begin{bmatrix} 0.0638 & 0.1454 & 1.3772 & -0.2049 & 0.0604 \end{bmatrix} \\
H_{dataset\ 4} &= \begin{bmatrix} 0.0154 & -0.0207 & 0.2317 & -0.3002 & -0.0187 \end{bmatrix} \\
H_{dataset\ 5} &= \begin{bmatrix} 0.0261 & 0.1441 & 3.2118 & 0.2848 & 0.0279 \end{bmatrix}
\end{aligned}
$$

The H-matrices were tested for a 1% disturbance. The optimal cost function in the case of 1

% disturbance was found by re-optimization to be $J_{opt} = 21.6885$. The cost function values achieved by controlling the process with the H-matrices are given in Table 6.3

Table 6.3: The cost function when the process is controlled with different H-matrices calculated under the same conditions but with different data samples as a basis. The process is disturbed with a +1% increase in the feed rate, and by re-optimizing the process for this disturbance the cost function is found to be: $J_{opt} = 21.6885$.

H-matrix	J_{ctrl} [kmol/min]	Loss [kmol/h]
$H_{dataset\ 1}$	21.6903	0.1061
$H_{dataset\ 2}$	21.6905	0.1182
$H_{dataset\ 3}$	21.6911	0.1545
$H_{dataset\ 4}$	21.6888	0.0202
$H_{dataset\ 5}$	21.6911	0.1551

Even if the conditions for creating the data set are the same for all the five cases over, both the H-matrix itself and then also the loss when using the calculated H-matrix change. For this reason, we decided to use an average value when testing the data method. The loss itself is more interesting than the actual matrix or the cost function values, therefore only the average loss from the five simulations will be presented in the results (Section 6.3).

6.2.1 Comparing the control structure found by the data method to other possible control structures.

In order to test how well the H-matrix found by the data method works, we compared it by simulating the process with other control structures. The two other control structures were found by using the exact local method and the null space method. We included measurement noise in the exact local method calculations. When the noise matrix, W_n was set to zero, the calculation of H failed in the case with the exact local method.

The loss for exact local method and null space method

For each of the test cases, when the exact local method or the null space method are used there is only one loss. This is because these two methods are unaffected by which data is used or the different parameters changing the collected data. The only parameter affecting the results from these methods is the number of variables used in the calculations (5 or 10) and if the disturbance is included or not.

6.3 Results

In order to follow the presentation of the result more easily, the notation for the different cases is repeated below. The different test cases are noted **case A, B, C** and **D**, and as explained earlier the case name corresponds to whether there are 10 or 5 measured variables, and whether the disturbance is included as a measured variable or not. In addition, the cases are run both with and without measurement noise in the data. The variables used as measurements are given earlier in this chapter (Equation 6.2a-d).

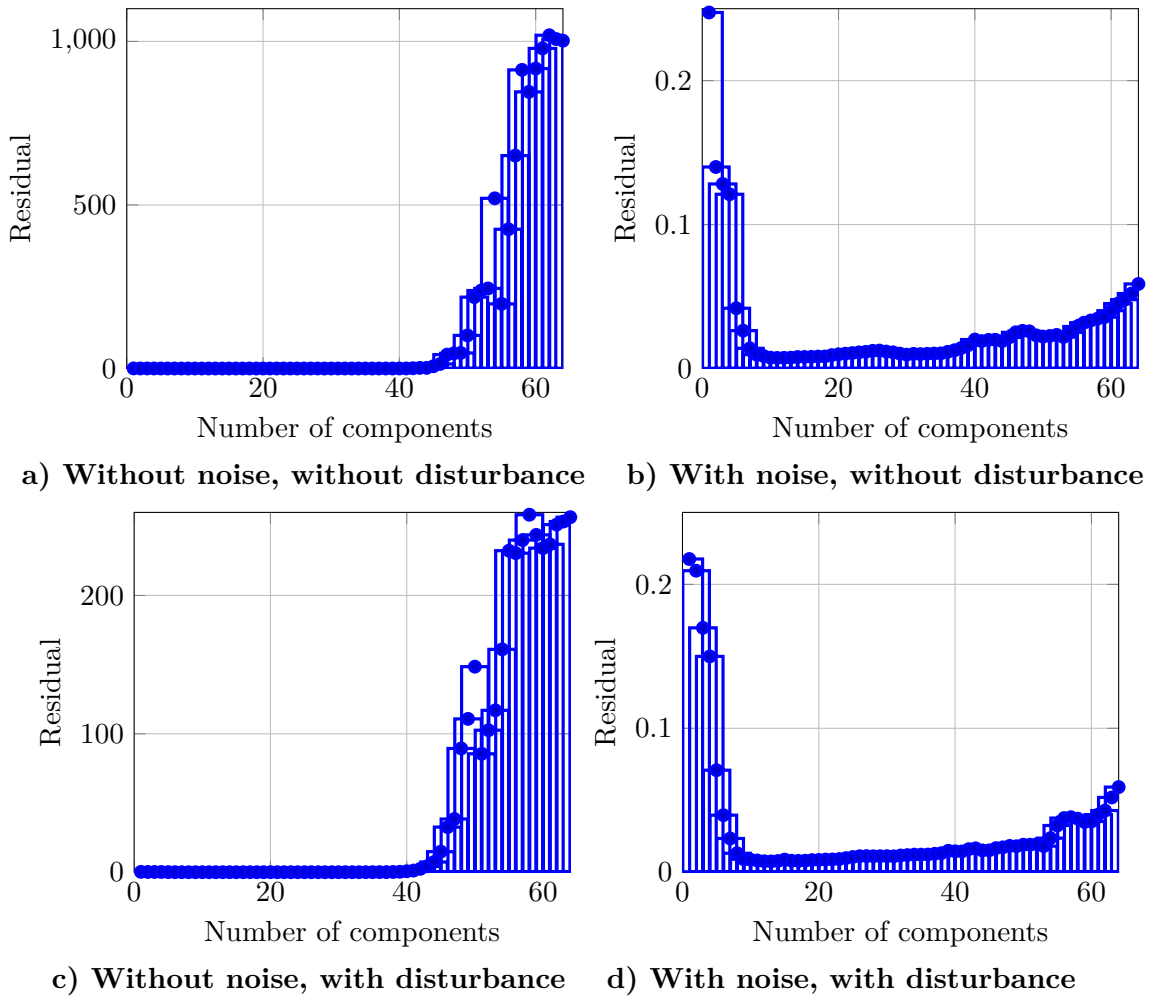
- **Case A:** 10 measured variables, not including the disturbance (F_0) as a measurements.
- **Case B:** 10 measured variables including the disturbance (F_0) as a measurements.
- **Case C** 5 measured variables, not including the disturbance (F_0)as a measurements.
- **Case D:** 5 measured variables including the disturbance (F_0) as a measurements.

6.3.1 Number of components

Before using the data method one of the key parameter must be decided, namely the number of components to be used in the PLS-regression. The scheme to find the optimal number of components, as explained in Section 4.3.2, was also run for this process for all the four test cases, with and without measurement noise.

The optimal number of components for the cases with 10 measurements is 10. After 10 components there is practically no change in the residual. The trend that the residual increases rapidly towards the maximum number of component for the case with no measurement noise is believed to be caused by numerical mistakes. When we zoom in on the 20 first components (Figure 6.3) we discover that both cases (with and without noise) behave similarly. As stated in the Evaporator case study, the rapid increase in the residual for high number of components for the case without measurement noise is most likely cause by numerical error (Figure 6.2).

For the case with 5 measurements without including the disturbance as a measured variable, the ideal number of components identified with this method is 8 (Figure 6.4). However, in the end we decided to use 10 number of components after all. Because after testing how well the problem converged we found that ncomp=10 had a much higher convergence rate than ncomp=8. If the problem did not converge, it means that the control structure set up by the H-matrix turned out to be unfeasible. This will be more thoroughly explained in the discussion section. We found that for ncomp=8 the control problem converged only 52 out of 100 times, while for ncomp=10 the convergence was 100%. Therefore, 10 number of components were used as the ideal number of components also in the case with 5 measurements. Moreover, in the case where the disturbance is included in the 5 measured variables, 10 number of components seems to be the ideal number to use (Figure 6.4b and d).



a) Without noise, without disturbance **b) With noise, without disturbance**

c) Without noise, with disturbance **d) With noise, with disturbance**

Figure 6.2: In the case with 10 measured variables. Deciding the number of components for the case with 50 data samples, with and without noise, including and not including the disturbance in the measurements.

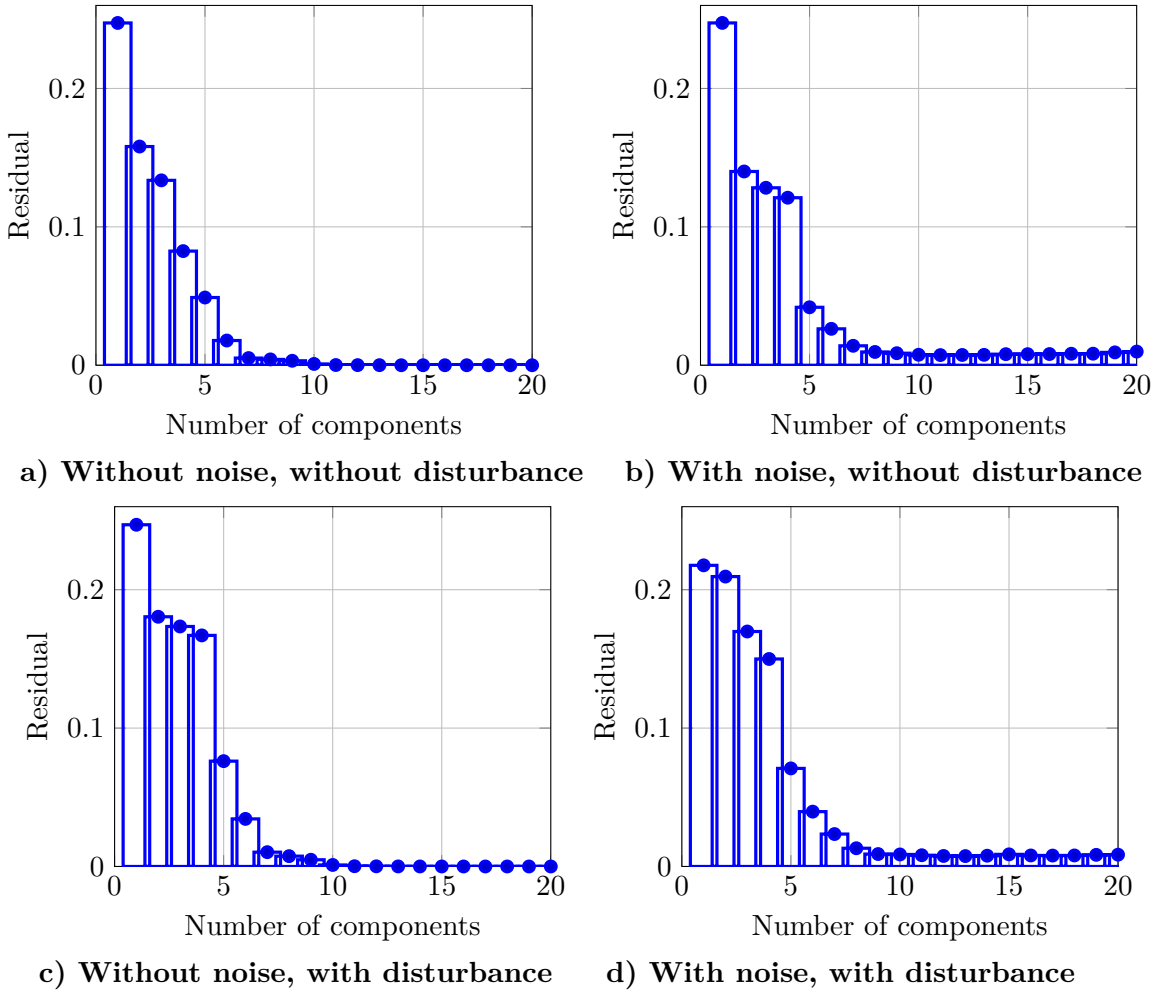
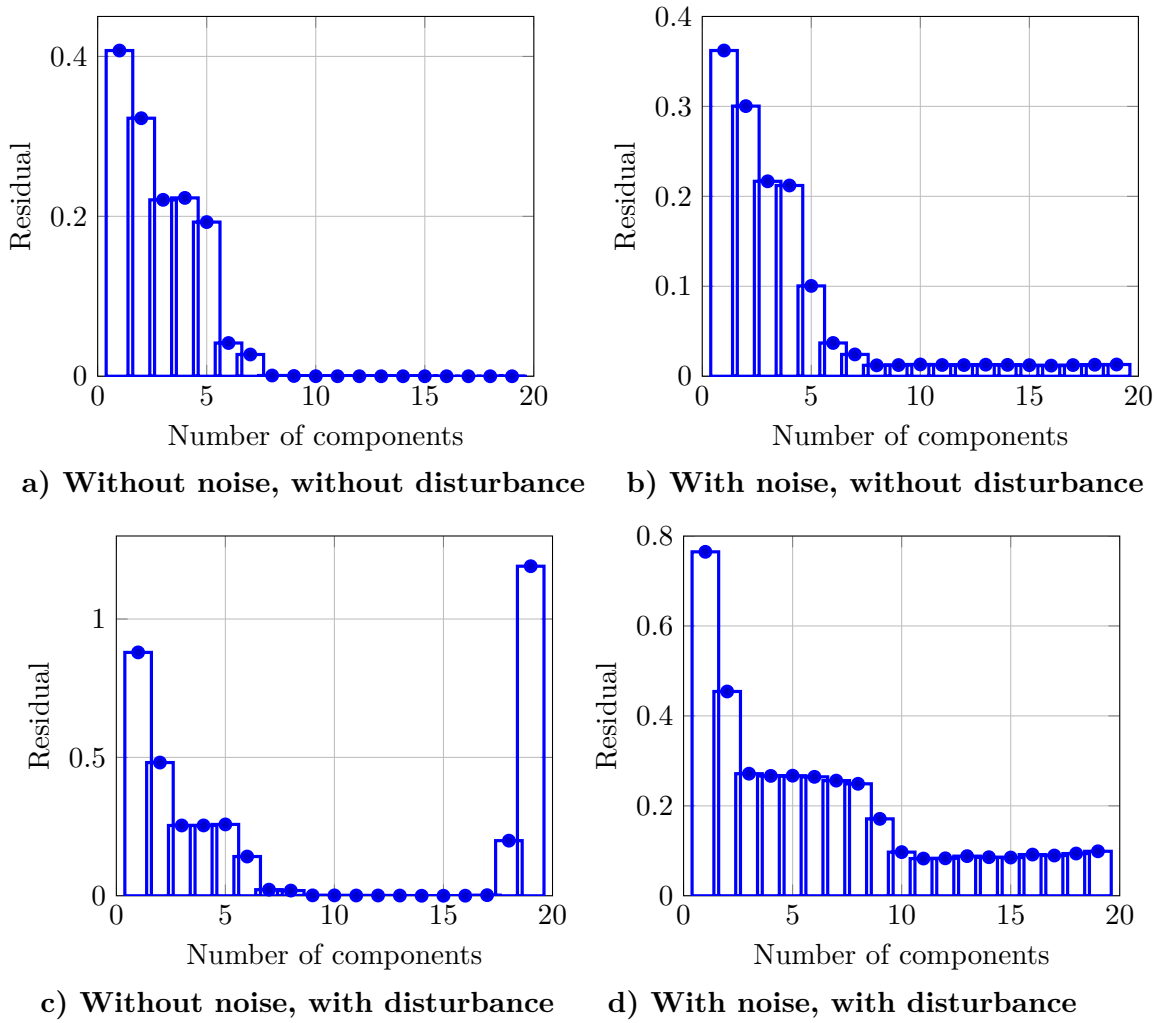


Figure 6.3: In the case with 10 measured variables. Deciding the number of components for the case with 50 data samples, with and without noise, including and not including the disturbance in the measurements. Showing only 1 to 20 number of components.



a) Without noise, without disturbance **b) With noise, without disturbance**

c) Without noise, with disturbance **d) With noise, with disturbance**

Figure 6.4: In the case with 5 measured variables. Deciding number of components for the case with 50 data samples, with and without noise, including and not including the disturbance in the measurements.

Using 5 and 10 number of components in the PLS-regression

The PLS-regression was run with both 10 and 5 number of components. This was done to investigate the effect the different number of components have on the outcome of the regression.

The findings from these tests are somewhat complicated. It is expected that the ideal number of components should give the best result in terms of the magnitude of the loss. However, this was not always the outcome for the cases with 10 measured variables. In Figure 6.5 we see that using 5 number of components actually gives the best result in terms of loss for all cases except for Case B, the case without the disturbance, with measurement noise.

However, controlling the process with the H-matrices found with 5 number of components proved to be complicated. Very often the control process did not converge. And bear in mind that for each test case five H-matrices were found. The loss is therefore an average of the losses found for each of the five control structures. For the cases where 5 number of components were used, the magnitude of the loss for the five different simulations varied a great deal. Whilst the variation was not nearly as great when 10 components were used.

In the case with 5 measured variables (Case C and D) using 10 number of components gave without doubt the best result in terms of loss (Figure 6.6). Reducing the number of components to 5 in this case had a clear negative effect on the outcome.

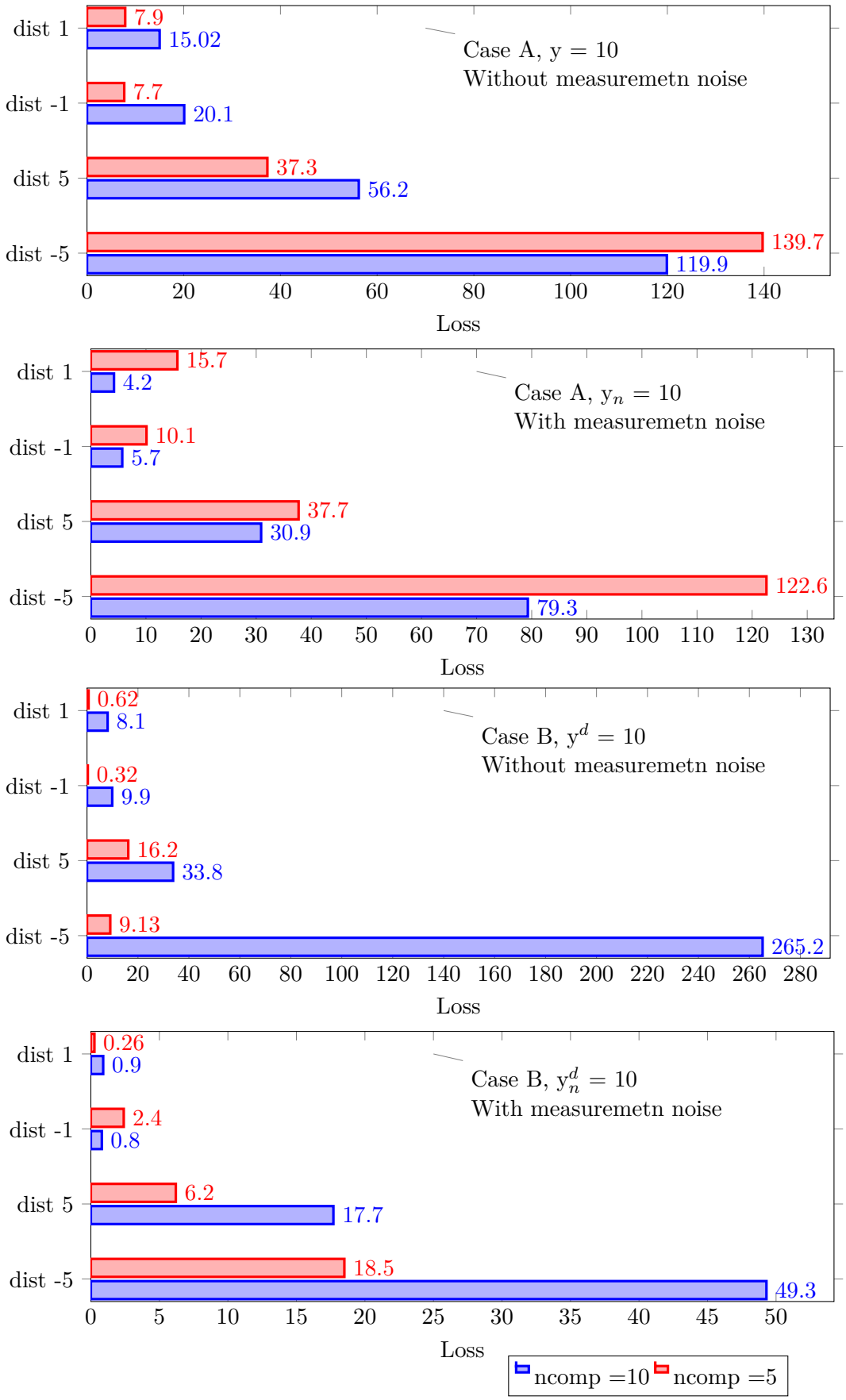


Figure 6.5: The loss between optimizing the process and controlling it with the calculated H-matrix, found when using 5 ncomp and 10 ncomp, for y=10.

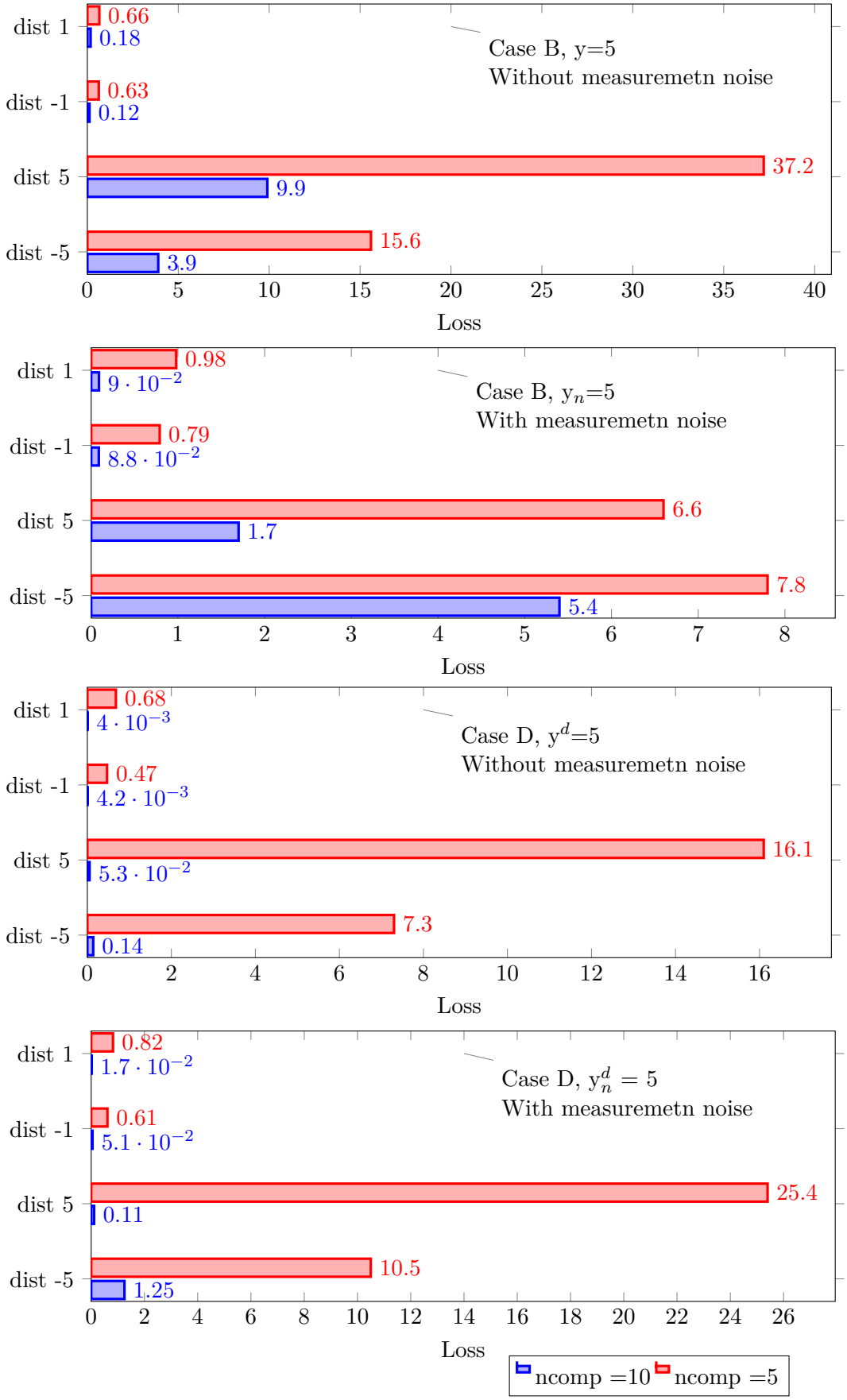


Figure 6.6: The loss between optimizing the process and controlling it with the calculated H-matrix, found when using 5 ncomp and 10 ncomp, for y=5.

6.3.2 Model validation

Before calculating the combination matrix the model quality should be considered. This is done by comparing the modeled cost function to the measured value. If there is a large difference the model is inaccurate, and the calculation of the matrix is based on a false picture of reality. If this is the case, we expect that the H-matrix will most likely result in poor control of the process. Testing the model is done as described in Section 4.3.2, by calculating the residual between the modeled and the measured cost function $J_m - J_{mod}$.

To make sure the matrix is based on a realistic model, the model is validated by checking the residual. This is done every time the combination matrix is calculated. The result is presented in Table 6.4. It is an average residual since each of the five H-matrices calculated for each case study gave a residual.

For the case with 10 number of components we found that the case with 10 measurements in general gave a more accurate model (smaller residual) than the case with 5 measurements. In addition, the model accuracy is significantly improved when the data does not contain measurement noise. Further more, the model accuracy became extremely worse when the number of components was reduced from 10 to 5.

Due to the unstable result in terms of loss magnitude and convergence problems for the case with 5 number of components, the rest of the test cases are run with ncmop = 10 only. Overall it seems that the difference between the estimated value of the cost function and the actual measured value is then acceptably small. We therefore assume that the modeling works well enough to give an useful H-matrix.

Table 6.4: The residual $J_m - J_{mod}$ for all the combination cases, for 50, 100 and 1000 samples in a data set for the CSTR-distialltion case-study. It also shows the residual for the case with 50 samples when ncomp=5 in the PLS regression.

Number of component = 10				
Without measurement noise:				
	Description	ns=50	ns = 100	ns = 1000
Case A	10 y	6.1×10^{-5}	1.3×10^{-4}	0.0013
Case B	10 y^d	8.8×10^{-5}	1.5×10^{-4}	0.0017
Case C	5 y	3.5×10^{-4}	7.0×10^{-4}	6.7×10^{-3}
Case D	5 y^d	1.1×10^{-4}	2.5×10^{-4}	0.0024
With measurement noise:				
	Description	ns=50	ns = 100	ns = 1000
Case A	10 y_n	7.5×10^{-4}	0.0014	0.013
Case B	10 y_n^d	7.7×10^{-4}	0.0014	0.014
Case C	5 y_n	0.001	0.002	0.025
Case D	5 y_n^d	0.008	0.016	0.16
Number of component = 5				
Without measurement noise:				
	Description	ns=50		
Case A	10 y	0.004		
Case B	10 y^d	0.006		
Case C	5 y	0.008		
Case D	5 y^d	0.02		
With measurement noise:				
	Description	ns=50		
Case A	10 y_n	0.004		
Case B	10 y_n^d	0.007		
Case C	5 y_n	0.007		
Case D	5 y_n^d	0.02		

6.3.3 Main findings

In this section the main findings are presented. These are findings for calculating the H-matrix using the data-based method, using different pre-conditions and parameters in the calculations and analyzing the effect of changing them. As well as finding the H-matrix using the exact local and null space method. A summary of all the main finding are given in Table 6.5 and in Table 6.8 all the loss values for all the different test cases are presented. All the plots given in this result section is based on figures from Table 6.8.

The loss is the difference in the cost function when the process is re-optimized for a given disturbance, contra controlled it with the calculated combination matrix, H. If the loss is small it indicates that the H-matrix used to control the process works well as a self-optimizing variable. To clearly show the findings, some of the losses are visualized with bar plots. The plot compares the outcome of changing the conditions and/or the parameters for the different cases.

First, a short summary of which parameters and what conditions were changed in order to test the data method:

- Using 50, 100 and 1000 data samples as a basis for the calculations.
- Including measurement noise for the data set.
- Using 5 measured variables and 10 measured variables.
- Using a data set with and without the disturbance included in the measured variables.
- Running the PLS-regression with 10 and 5 number of components.
- Comparing the combination of parameters which gave the minimum loss for the data method with other methods to calculate the H-matrix.

The main findings are summarized below together with references to which figures supports the result. A more thorough review of the results is given together with the plots in the next section. The results are discussed in the last part of this chapter.

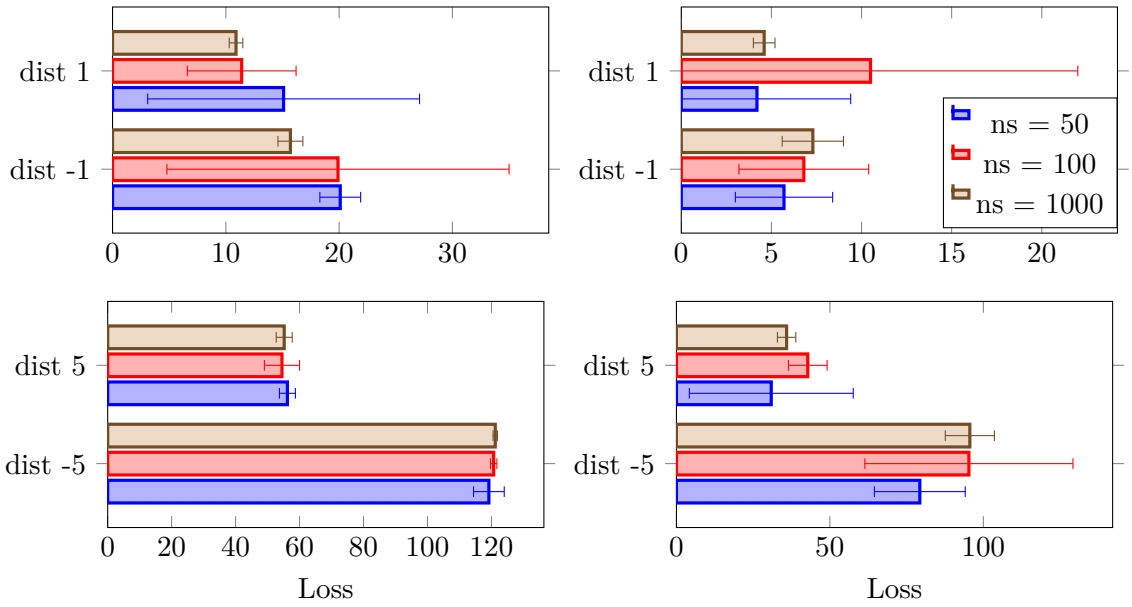
Table 6.5: The main results from testing the data method on the CSTR distillation column test case, as well as references to the figures which corroborate the result.

Finding	Figure reference
5 number of components did in some cases give the best result but was unstable in terms of convergence	Figure 6.6 - 6.6
10 number of components gave in general the best result both in therm of loss and convergence	Figure6.6 - 6.6
There is not an extensive difference in the loss when 50, 100 or 1000 samples are used	Figure 6.7-6.10
Using 5 measured variables gives a smaller loss compared to using 10 measured variables	Figure 6.11
Including measurement noise in the data sampling improves the performance	Figure 6.12-6.13
Including the disturbance as a measured variable will in general improve the performance	Figure 6.12-6.13
The null space method always gives the best result	Figure 6.12-6.13
For some cases the data method gives a better result than the exact local method	Figure 6.12-6.13

Using a different number of samples to generate data

To address the question of how many data points are required to obtain a good result in terms of loss calculations we ran three parallel cases. One where the number of sample point were 50 samples, a second with 100 sample points, and last with 1000 samples in one data set.

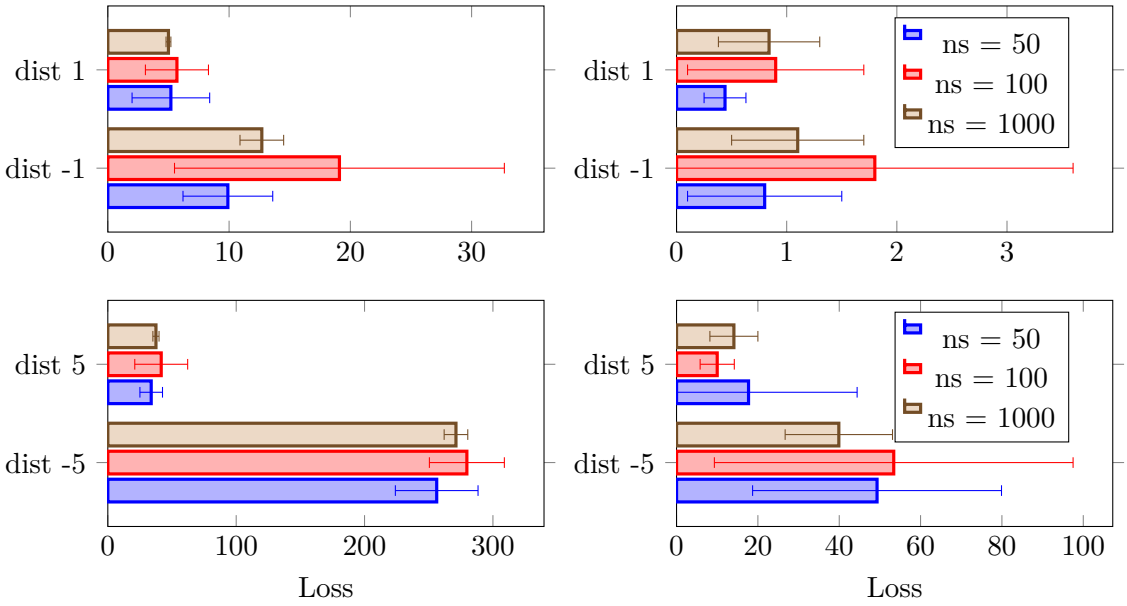
We ran the data generation five times, and for each generated data set we estimated the H-matrix and used it to control the process. The loss value$J_{opt} - J_{ctrl}$ was found for all the five runs. The bars in Figure 6.7-6.10 are the average loss values of these five runs. The thinner lines give the maximum and minimum loss value from the simulation.



(a) Without meareuement noise

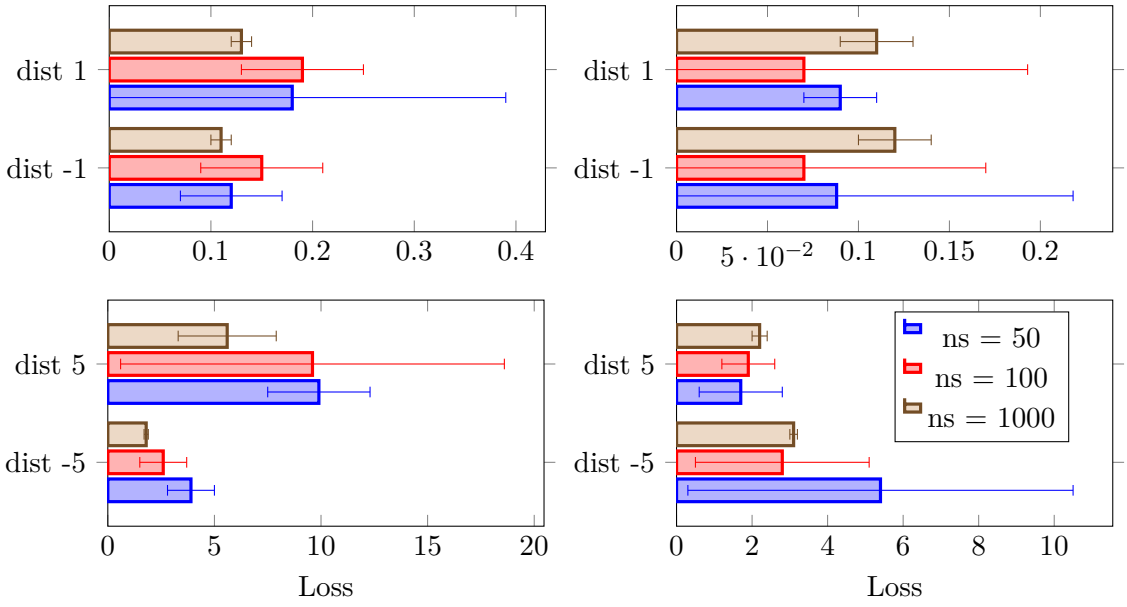
(b) With meareuement noise

Figure 6.7: The loss between optimizing the process and controlling it with the calculated H-matrix, for a ±1% and ±5% disturbance in the process Using ns = 50, 100 and 1000, for **Case A**, y=10 and the disturbance not included as a measurement.



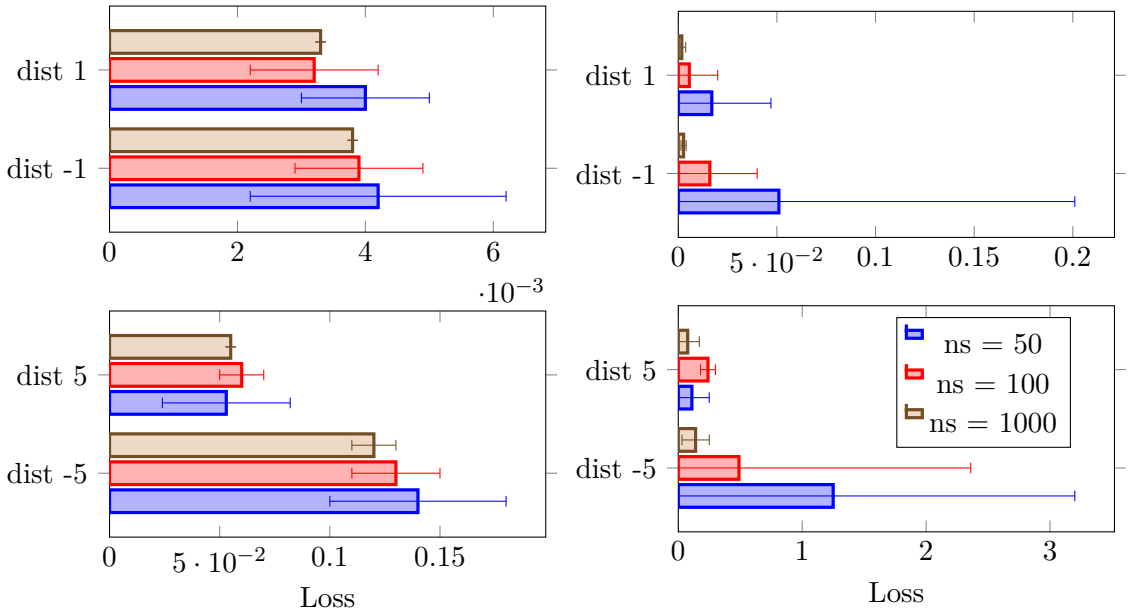
(a) Without meareuement noise **(b)** With meareuement noise

Figure 6.8: The loss between optimizing the process and controlling it with the calculated H-matrix, for a ±1% and ±5% disturbance in the process. Using 50, 100 and 1000, for **Case B**, y=10 and the disturbance included as a measurement.



(a) Without meareuement noise **(b)** With meareuement noise

Figure 6.9: The loss between optimizing the process and controlling it with the calculated H-matrix, for a ±1% and ±5% disturbance in the process. Using ns = 50, 100 and 1000, for **Case C** y=5 and the disturbance not includedas a measurement.



(a) Without meareuement noise

(b) With meareuement noise

Figure 6.10: The loss between optimizing the process and controlling it with the calculated H-matrix, for a ±1% and ±5% disturbance in the process. Using ns = 50, 100 and 1000, for **Case D** y=5 and the disturbance not included as a measurement.

The effect of using 5 and 10 measured variables

Reducing the number of measured variables from 10 to 5 variables improves the quality of H-matrix as a self-optimizing variable. This is clearly shown in the plot in Figure 6.11. In the cases with 5 measurements the loss values (grey and beige bars) are always significantly smaller, than the loss in the case with 10 measurements (red and blue bars). The difference between the two ($L_{y_n} - L_y$) is also given in Table 6.6 and using 5 measurements decreases the loss quite extensively.

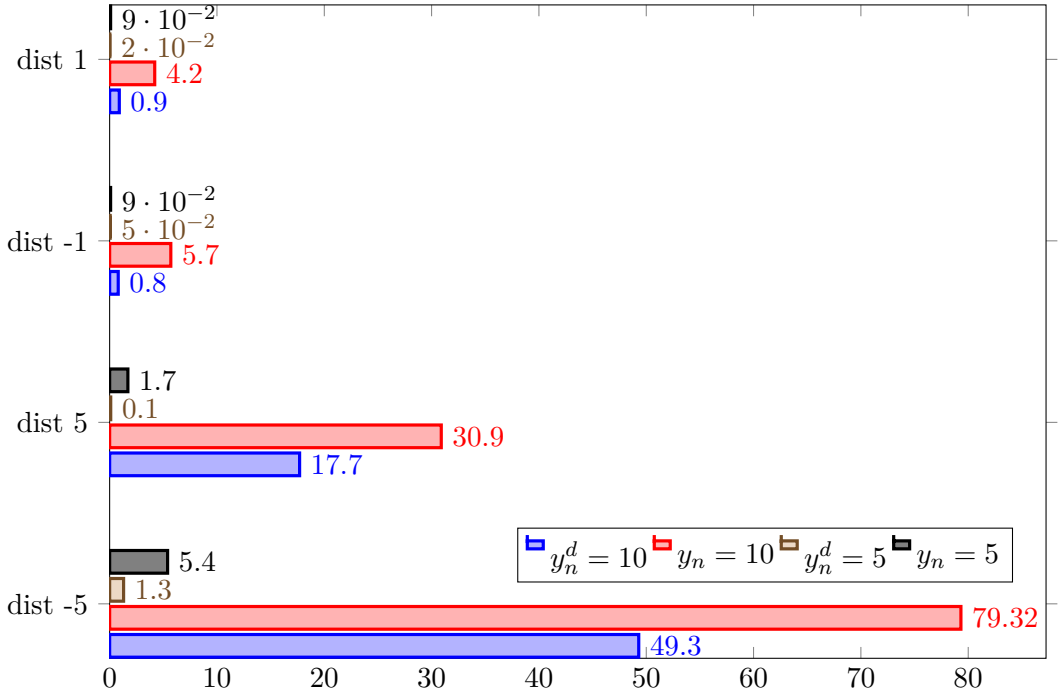
Table 6.6: The loss between optimizing the process and controlling it with the calculated H-matrix for four disturbances. The values compared using 10 or 5 measurement both (a) including and (b) not including the disturbance as a measured variable.

	Disturbance			
Case	1%	-1%	5%	-5%
Loss 10 y_n	4.2	5.7	30.9	79.3
Loss 5 y_n	0.09	0.088	1.7	5.4
Δ L	4.11	5.7	29.2	73.9

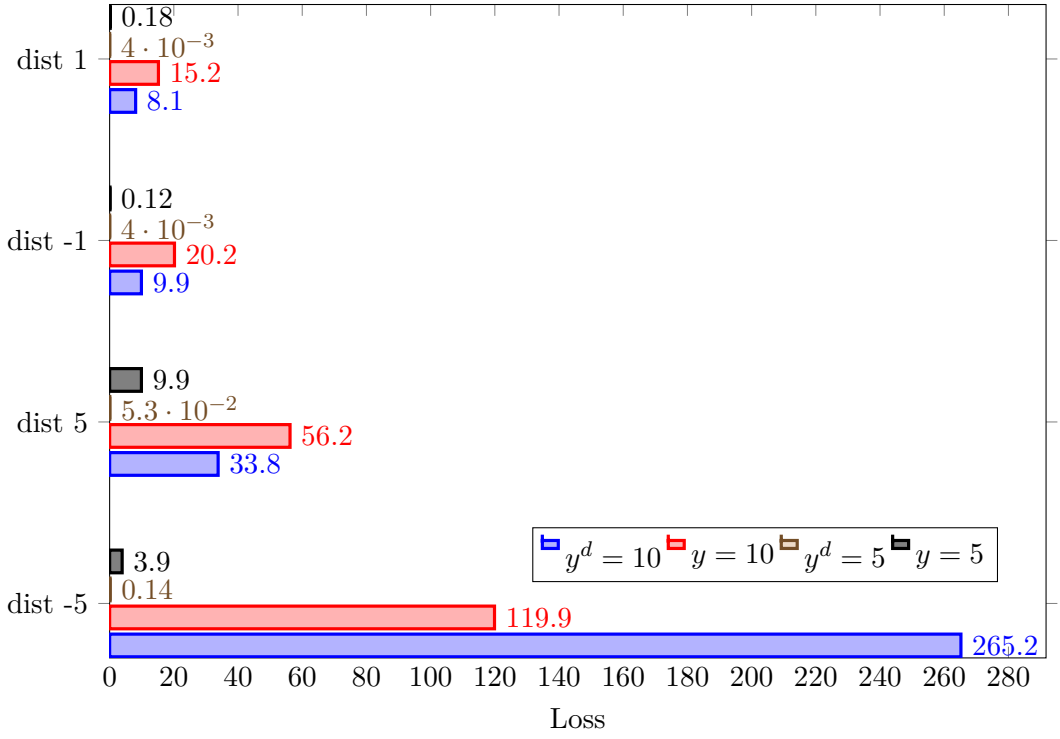
(a) Without measuring disturbance

	Disturbance			
Case	1%	-1%	5%	-5%
Loss 10 y_n^d	5.0	12.7	37.5	271.3
Loss 5 y_n^d	0.02	0.05	0.1	1.3
Δ L	4.98	12.65	37.4	269.7

(b) With measuring disturbance



a) With measuremetn noise



b) Without measurement noise

Figure 6.11: Comparing the effect of using 5 or 10 measured variables. The loss between optimizing the process and controlling it with the calculated H-matrix with ncomp=10, ns=50 for y=5 and y=10 including and not including the disturbance.

Including the disturbance as a measured variable

When the disturbance is included as a measured variable the control performance of the H-matrix is significantly improved. This can be seen if we compare the loss values in Figure 6.12 and 6.13, and comparing the red bar values with each other and the blue bar values with each other. From this it is clear that the values for Case A is higher compared to the loss values in Case A. The red bars represent using data samples with measurement noise, and the blue bars represent using data samples without measurement noise. The plot in Figure 6.12**a** and 6.13**a** show the loss when the disturbance is not included and the Figure6.12**b** and 6.13**b** shows the loss with the disturbance included as a measured variable.

For example, if we consider case A and case B including measurement noise (Table 6.7a). We see that the red bars in Figure 6.12**a**, which are for the case without the disturbance (Case A), are consistently larger than the red bars in Figure 6.12**b** where the disturbance is included (Case B). Likewise, we find the same trend for case A and case B for measurement without noise as well, marked with the blue bars in Figure 6.12**a** and 6.12**b**. The only exception is for the case without measurement noise when a -5% disturbance occur, marked in red in Table 6.7b.

In the case with 5 measurements (Table 6.7c and d) the result is always significantly better when the disturbance is included as a measured variable.

The same trend can be discovered in Figure 6.11, which shows the loss for all the four test cases, with noise (Figure 6.11a) and without measurement noise (Figure 6.11b). In this figure we see that the beige bar (y=5 incl. disturbance) is always smaller than the black bar (y=5 excl. disturbance). And likewise, the blue bar (y=10 incl. disturbance) is smaller than the red bar (y=10 excl. disturbance) except in the case without noise for the -5% disturbance.

6.3.4 Using data samples with measurement noise

When measurement noise is taken into account for the data used as a basis for the H-matrix calculation the loss will in general decrease. This can be seen by comparing the red and the blue bar in each of the plots in Figure 6.12 and 6.13. In most of the cases the red bar (data with measurement noise) is smaller than the blue bar (data without measurement noise). We found only two exceptions to this trend. They occurred for case D, (5 measured variables, including the disturbance) shown in Figure 6.13**b**. However, the loss for data not including noise is not particularly larger than for the loss for data including loss.

6.3.5 Other methods

Figure 6.12 and 6.13 also compare the data-based method to the exact local and null space method.

The smallest loss in all cases is achieved by using the H-matrix calculated by the null space method. For the test cases A and B, the cases with 10 measurements, the exact local method proved to give a better control structure than the data method. The loss when using the exact local method turned out to be at least two times smaller than the loss from the data method.

Table 6.7: The loss between reoptimizing the process and controlling it with the calculated H-matrix for four disturbances. When the process is inflicted by a ±1% and ±5% disturbance. For the cases where the measurement matrix both nclude, y_n^d, and not include, y_n, the disturbance as a measured variable. For 5 and 10 measured variables, using 1000 data samples with and without measurement noise.

	Disturbance			
Case	1%	-1%	5%	-5%
10 y_n	5.4	7.3	35.9	95.7
10 y_n^d	0.79	1.1	14.1	39.9
Difference	4.61	6.2	21.8	55.8

(a) With noise

	Disturbance			
Case	1%	-1%	5%	-5%
10 y	10.4	15.7	55.2	121.1
10 y^d	5.0	12.7	37.5	271.3
Difference	5.4	3.0	17.7	-150.2

(b) Without noise

	Disturbance			
Case	1%	-1%	5%	-5%
5 y_n	0.11	0.12	2.2	3.1
5 y_n^d	0.0018	0.0027	0.076	0.14
Difference	0.1082	0.1173	2.124	2.96

(c) With noise

	Disturbance			
Case	1%	-1%	5%	-5%
5 y	0.13	0.11	5.6	1.8
5 y^d	0.0033	0.0038	0.055	0.12
Difference	0.1267	0.1062	5.545	1.68

(d) Without noise

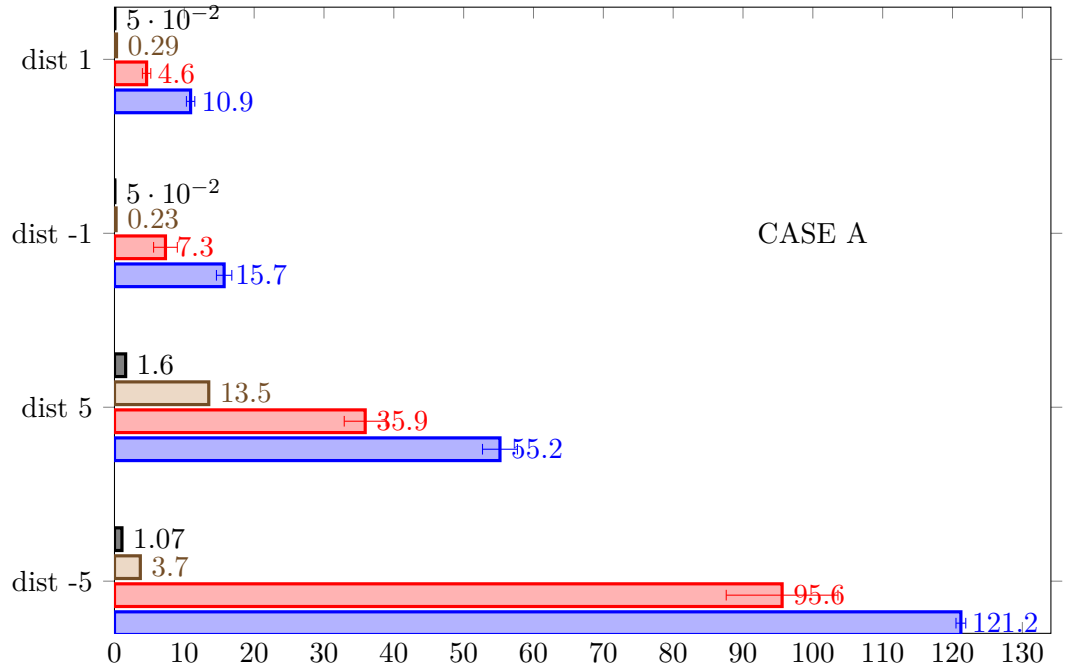
For case C and D on the other hand, the data method gave in general a better result than the exact local method.

6.3.6 Summary of all loss values

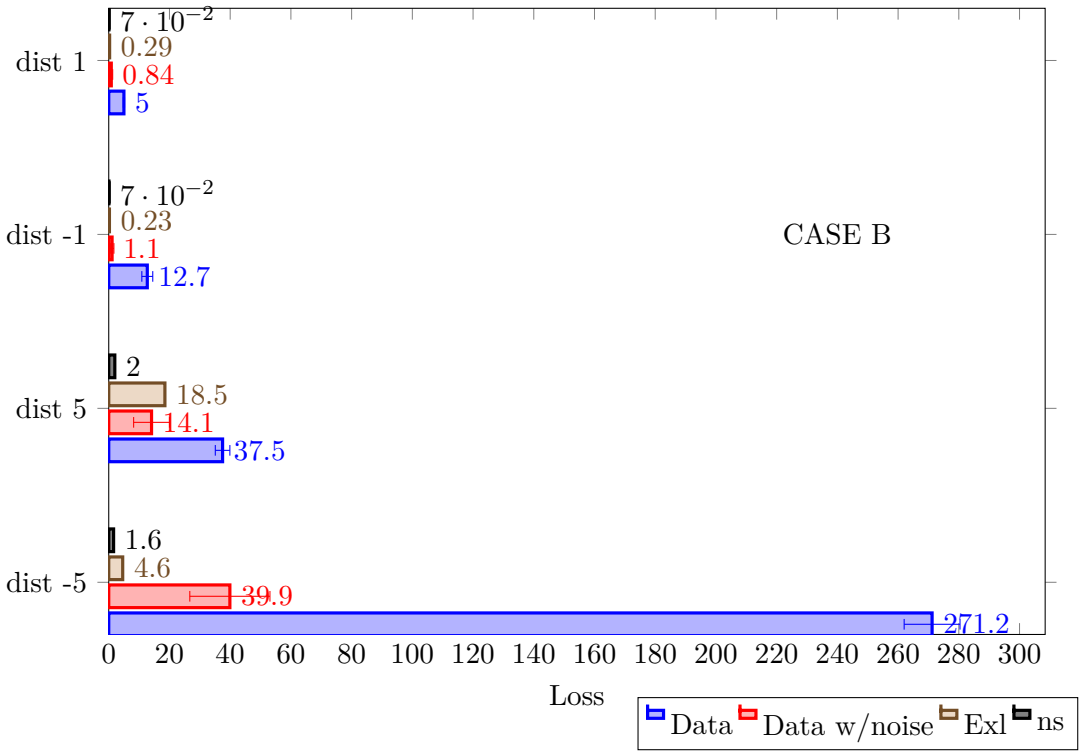
The average loss for all the test cases are summarized in Table 6.8. The numbers for all the plots presented in this chapter are taken for this table. The stars (*) behind some of the figures indicate how many times the control problem did not converge. Meaning that the H-matrix estimated from the PLS-regression resulted in an unfeasible control structure. This topic will be more thoroughly explained and discussed in the next section "*Discussion - CSTR-distillation test case*", as well as the other main findings from this case-study.

Table 6.8: The loss values for all test cases in this case study

	$ns = 50$				$ns = 100$				$ns = 1000$			
	$d = 1\%$	$d = -1\%$	$d = 5\%$	$d = -5\%$	$d = 1\%$	$d = -1\%$	$d = 5\%$	$d = -5\%$	$d = 1\%$	$d = -1\%$	$d = 5\%$	$d = -5\%$
NCOMP = 10												
10 measurements												
Without disturbance, $y = \begin{bmatrix} T_8 & T_{13} & T_{18} & T_{22} & L_T & V_B & \text{D} & \text{B} & \text{F} & z_F \end{bmatrix}$												
No noise, y	15.0	20.1	56.2**	119.9	10.7	19.9	58.0**	120.7	10.4	15.7	55.2	121.1
With noise, y_n	4.2	5.7	30.9***	79.3	4.0	6.8	43.1*	95.3	5.4	7.3	35.9	95.7
Exact local	0.29	0.23	13.5	3.7								
Null space	0.05	0.05	1.6	1.1								
With disturbance, $y = \begin{bmatrix} T_8 & T_{13} & T_{22} & L_T & V_B & \text{D} & \text{B} & \text{F} & z_F & F_0 \end{bmatrix}$												
No noise, y^d	8.1	9.9	33.8	265.2	4.9	19.1	36.4	279.7	5.0	12.7	37.5	271.3
With noise, y_n^d	0.91	0.84	17.7	49.3	0.94	1.8	10.0	53.4	0.79	1.1	14.1	39.9
Exact local	0.29	0.23	13.9	3.8								
Null space	0.07	0.07	2.0	1.6								
5 measurements												
Without disturbance, $y = \begin{bmatrix} T_8 & T_{22} & L_T & \text{B} & \text{F} \end{bmatrix}$												
No noise, y	0.18	0.12	9.9	3.9	0.19	0.15	9.6	2.6	0.13	0.11	5.6	1.8
With noise, y_n	0.09	0.088	1.7	5.4	0.07	0.068	1.9	2.8	0.11	0.12	2.2	3.1
Exact local	0.37	0.29	18.5	4.6								
Null space	0.0004	0.0003	0.013	0.006								
With disturbance, $y = \begin{bmatrix} T_8 & T_{22} & L_T & \text{B} & F_0 \end{bmatrix}$												
No noise, y^d	0.004	0.0042	0.053	0.14	0.0032	0.0039	0.06	0.13	0.0033	0.0038	0.055	0.12
With noise, y_n^d	0.02	0.05*	0.1	1.3	0.0056	0.016	0.24	0.49	0.0018	0.0027	0.076	0.14
Exact local	0.0053	0.0050	0.15	0.11								
Null space	0.0	0.0	0.0007	0.0002								
NCOMP = 5												
10 measurements, without the distutbance												
No noise, y	7.9**	7.1	37.3***	139.7								
With noise, y_n	15.7**	10.1	37.7**	122.6								
10 measurements, with the distutbance												
No noise, y^d	0.62	0.32	16.2	9.13								
With noise, y_n	0.26	2.4**	6.2*	18.5***								
5 measurements, without the distutbance												
No noise, y	0.66	0.63	37.2	15.6								
With noise, y_n	0.98	0.17	6.6*	7.8								
5 measurements, with the distutbance												
No noise, y^d	0.68	0.47	16.1	7.3								
With noise, y_n^d	0.82	0.61	25.4	10.5								

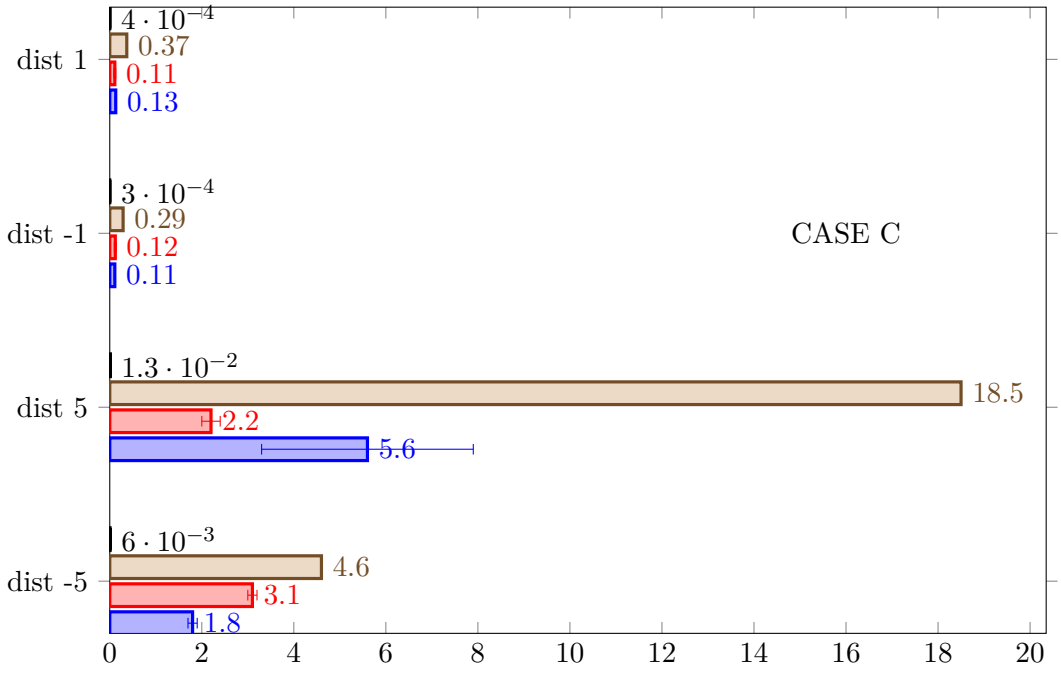


(a) CASE A: y= 10, Without the disturbance as a measured variable

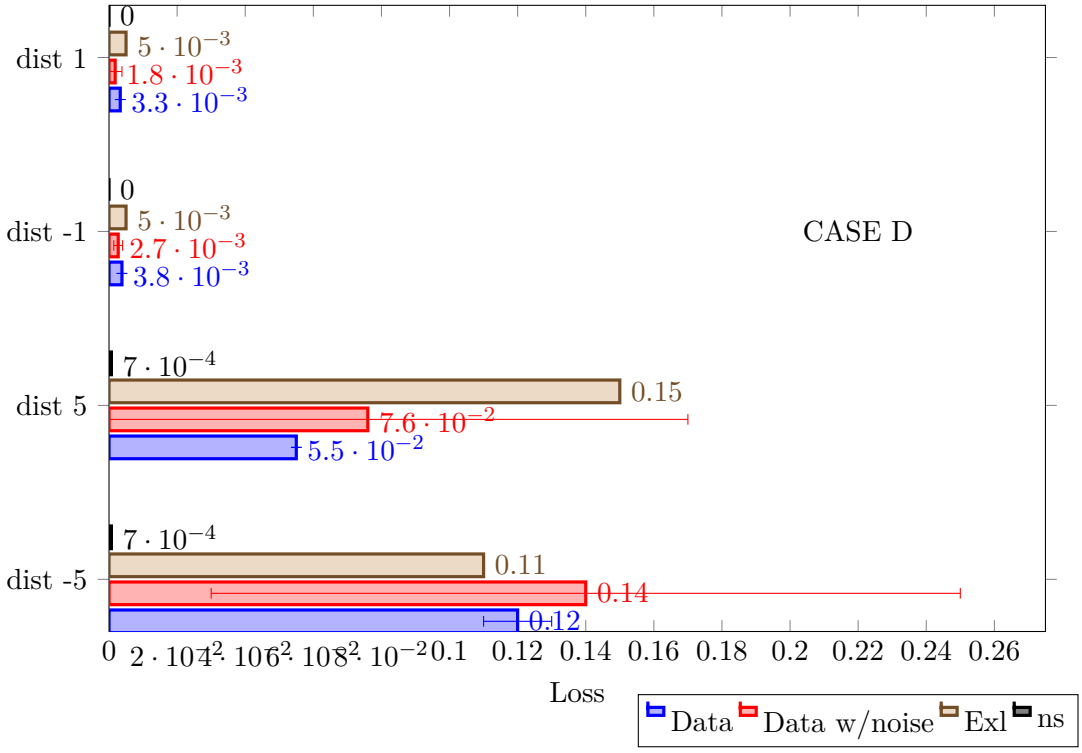


(b) CASE B: y= 10, Including the disturbance as a measured variable

Figure 6.12: The loss between optimizing the process and controlling it with the calculated H-matrix. The plot showes the loss when using the H-matrix found by the data-basedmethod with ncomp=10, ns=1000 and y=10 both with and without measurement noise. As well as the loss when using the H-matrix found by the exact local and null space method to control the process without any implementation error.



(a) CASE C: y= 5, Without the disturbance as a measured variable



(b) CASE D: y= 5, Including the disturbance as a measured variable

Figure 6.13: The loss between optimizing the process and controlling it with the calculated H-matrix. The plot showes the loss when using the H-matrix found by the data-basedmethod with ncomp=10, ns=1000 and y=5 both with and without measurement noise. As well as the loss when using the H-matrix found by the exact local and null space method to control the process without any implementation error.

6.4 Discussion - CSTR-distillation test case

6.4.1 Number of components

The number of components decides how many directions in the data that should be included in the modeling. And the clue is to find the ideal number to use in the modeling. We know that if we include too few number of components, some important directions or relations in the data will not be explained by the model. Including too many on the other hand will make the model too complex. Meaning that we risk also including noise in the modeling, and relations that are not real are used in the modeling.

By using the approach described in Section 4.3.2 we identified 10 as the ideal number of components (Figure 6.2 and 6.4) for all the cases studied in this test case. To begin preliminary research on the outcome when using less number of components than the ideal, we ran parallel test-cases. We used the same data set as a basis to calculate the H-matrix using both 10 and 5 number of components. Then we compared the resulting loss of using the H-matrix as a controlled variable. Since 10 was believed to be the ideal number of components, this should give the best H-matrix and therefore the lowest loss. As explained next, we were not able to come to a final conclusion on this topic.

For the case with 10 measured variables, y=10, it seems by Figure 6.5 that using 5 number of components gives in general the lowest loss. Despite of this, we recon that 10 number of components is still the best choice. One way to evaluate how well the H-matrix works is to look at the convergence rate of the control problem. In other words, is it possible to use the H-matrix to construct a control structure or does it result in an unfeasible solution? We found that the convergence rate is relatively low for the control problem when we use a H-matrix calculated with 5 number of components. This is probably because a unique input u cannot be found with these H-matrices. To be able to do so, (HG_y) must be invertible. This can be explained by looking at the control variable we use in this case, which can be described as:

$$c = Hy$$

Where:

$$y = G_y u + G_y^d d$$

Or in deviation variabels:

$$\Delta c = H\Delta y$$
$$\Delta y = G_y \Delta u + G_y^d \Delta d$$

Which gives the following expression to be controlled to zero:

$$\Delta c = HG_y \Delta u + HG_y^d \Delta d = 0$$

Solving this expression shows why (HG_y) must be invertible to find a unique u:

$$\Delta u = -(HG_y)^{-1} HG_Y^d \Delta d$$

If (HG_y) cannot be inverted a feasible control solution is not found and the control problem does not converge. I.e no unique u is found so we cannot control c to the desired set-point. We assume that the reason the problem did not converge is because using 5 number of components often returns a H-matrix which causes (HG_y) to be not invertible.

We also observed a high variance in the loss values when using 5 number of components. This is shown with an example in Table 6.9. The loss is, as explained in the previous section, the loss between optimizing the process for a disturbance versus controlling it with the estimated H-matrix. The losses presented in Table 6.9 are for the specific case with 50 data samples using 10 measured variables where the disturbance is included as a measurement. The loss is the difference between re-optimizing and controlling the process for a -1% disturbance. However, the variation seen in this specific example is typical for all the cases with ncomp=5. The variation in the loss values indicates that the solution is unstable. On the other hand, this large degree of variation was not observed to the same extent in the cases where 10 number of components were used, indicating that for theses cases a more stable solution was found.

In addition, the model accuracy decreased when 5 number of component were used. As seen in Table 6.4 the model is significantly more accurate in the cases where 10 number of components are used in the PLS-regression.

For these reasons we regard the results found using 10 number of components as more reliable than for the cases with 5 number of components.

For the case with 5 measured variables, y=5, using 10 number of components gave the best result for all the cases. An interesting observation made is that according to the bar plots in Figure 6.4, the ideal number of components in case C (not measuring the disturbance) is 8. However, a convergence test showed that the problem converged only 52 out of 100 trial runs with ncomp=8. Whilst the convergence rate was 100% for ncomp = 10. In the convergence test we generated 100 different data sets, and found 100 different H-matrices calculated with the same preconditions, so that only the data used as a basis changed. The problem did not converge, if we failed to control the process with the estimated H-matrix. The H-matrix then resulted in an unfeasible solution as explained above.

From these tests we see that the number of components used in the PLS-regression have an important impact both on the model accuracy and on the performance of the H-matrix found from the modeling. However, we are not able to draw any conclusive lines from these tests regarding what the effects really are. The approaches presented to decide the ideal number of components are found to be indicative, though not unambiguous. We believe that in order to fully be able to use this method this topic should be researched further.

Table 6.9: An example showing the variance in the loss calculated with 50 data samples and y_d=10. The loss is for a disturbance of -1% for the H-matrix calculated with 5 and 10 number of components.

ns=50 , y_d=10, disturbance = -1%		
	ncomp=5	ncomp = 10
L_{max}	649	1.51
L_{avg}	2.4	0.8
L_{min}	0.091	0.42

6.4.2 How many data samples should be used in the estimation

We are not able to see any clear advantage or disadvantage regarding using many or few data samples. However we observe a weak trend that using 1000 samples results in a lower loss.

As explained earlier the data sets were generated five times, and five different H-matrices were found and used in the control simulations. Each matrix resulted in a loss. We see that the cases where 1000 data samples were used, the variance between the maximum and minimum loss for the five simulation is somewhat smaller than in the cases with 100 and 50 data samples. This is seen by comparing the thinner lines (Figure 6.7-6.10) which indicate the minimum and maximum loss in the set with 5 loss calculations. This trend is however, not consistent enough to say anything conclusive.

A set of 5 runs are not enough to ensure statistical validity of the result. Because of the relatively large variance in the loss values at times, we recon more test runs are necessary to find a true average. Only then, can more valid conclusion be drawn. This will in general apply to all the results found in this test case. Unfortunately, due to limited computational power we were not able to run more parallel cases. However, the results found here are indicative, and we believe, worth more research.

6.4.3 The number of measured variables

In this test-case we studied using either 10 or 5 measured variables as a basis for the model estimation, the results were illustrated in Figure 6.11. We found that using 5 measured variables resulted in the lowest loss. However, the cost function model is more accurate when we use 10 measurements. At first it might seem strange that a less accurate model results in a better control performance. But these two trends can be explained by looking at the linear approximation of the quadratic cost function given in Equation 2.3 in the theory chapter.

$$J \approx J^* + \begin{bmatrix} J_u^* & J_d^* \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta d \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \Delta u^T & \Delta d^T \end{bmatrix} \begin{bmatrix} J_{uu}^* & J_{ud}^* \\ J_{du}^* & J_{dd}^* \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta d \end{bmatrix}$$

In the modeling of the cost function the whole beta matrix given from the PLS-regression is utilized. The components in the beta matrix are used to model both the linear and quadratic terms in the cost approximation. We see that 10 measurements results in a more accurate model than with 5 measurements, this could indicate that the linear part might be most important in terms of model accuracy. If this is true, and most of the cost function value is explained in the linear part of the cost function approximation, it seems like using 10 measurements gives a better model of the linear part than when 5 measurements are used. For 10 measurements, it could be that the second derivative term associated with the quadratic part of the cost function, is small and therefore less important for the cost estimation. Hence, a poor estimation of this matrix does not have a significant impact on the overall model accuracy. It will however, have a large impact on the quality of the H matrix.

This leads us to why the case with 5 measurements gives a better result in term of loss. We suspect that in this case the overall modeling is poorer than in the case with 10 measurements, but the second derivative matrix is more accurate estimated. This can explain why the loss when

using 5 measurement, are significantly lower than when using 10 measurements, despite the fact that the model is actually less accurate.

When estimating the H-matrix we use only parts of the estimated cost function, namely J_{yy}, the matrix in the quadratic term in the approximation. Thus, this is the important part when identifying the H-matrix. Since the performance of H as a self-optimizing control variable is better in the case for 5 measurements we believe that this part is more accurately modeled than in the case with 10 measurements. This is actually a quite reasonable outcome. For 5 measured variables the size of the J_{yy} is $5 \times 5 = 25$ elements, whereas for 10 measured variables the size of the J_{yy} is $10 \times 10 = 100$ elements. Hence, there are 4 times as many variables to be correctly estimated in the case with 10 measurements compared to the case with 5 measurements. Or in other words, 4 times as likely to make an error. This justifies why the H-matrix with 5 measurements performs better than in the case with 10 measurements.

In this case the minimum number of measured variables are $n_u + n_d = 2$. We chose to use 5 simply because it was half of the maximum number of variables available. It would however be interesting to run the same test done here with 2 measured variables. In order to study whether this would give a better or worse result compared to using 5 measured variables.

6.4.4 Including the disturbance as a measured variable

One of the first features highlighted about this method is that it does not need measurements of or even knowledge about process disturbances. However, in some cases we are able to measure the disturbance. For this reason, we also studied the cases where the disturbance is included in the measurement matrix. This gave a favorable outcome. Including the disturbance as a measurement reduces the loss significantly compared to using a measurement matrix without the disturbance included. However, this result is not unexpected. The disturbance is a direct cause for other parameters and the cost function to change. When it is included in the modeling, the regression works better because one of the direct causes for changes in the cost function is immediately available. Whilst in the case when the disturbance is not included the disturbance is only visible through the output measurements, Y, which is a product of the process gain times the inputs and disturbance matrix. The likelihood for making a modeling mistake is thereby higher.

6.4.5 Using data with and without measurement noise

In all real cases measurement noise will be present in data sampling. We already know that the null space and exact local method give their best result in cases without noise. The question we wanted to answer was how well the data-based method perform in cases with contra cases without measurement noise. After running parallel cases estimating the H-matrix using data with and without measurement noise we found that including noise actually gave the best in term of loss. This is seen from Figure 6.12 and 6.13.

This can be because the H-matrix found using data with measurement noise is more robust to small variations and numerical errors. In the cases with noise the model can handle small random changes better than the case without noise. When modeling without noise all the estimator sees are the exact pattern presented by the data. The model is therefore easier to estimate, and itself is more accurate. Most likely it is, however, accurate only for the data set used in the modeling. It will be more sensitive to changes in the data set. Adding measurement noise forces the estimator

(in this case the PLS-regression) to consider more random patterns in the modeling, making the model more robust to changes. We recon this can explain why the H-matrix estimated with measurement noise gives a lower loss when used in a control test-case.

There are a few exceptions, where the H matrix estimated by using data samples without measurement noise gives a better result. These are the cases with 5 measured variables, and the H-matrix is used in control for large disturbances of $\pm 5\%$.

The diversity in the trends advocate that more research is needed on the topic. It seems like measurement noise can improve the result in most cases. But we cannot say anything conclusive about when the measurement noise is an advantage or a disadvantage. Or if it is an advantage only when controlling small disturbances, as could be the case here.

6.4.6 The data-based method compared to the other methods

The null space method gave the best result in terms of loss in all the cases. This was expected since no implementation error is used in the control structure. The loss when using the null space method should therefore be zero for small disturbances (and in the cases run with $\pm 0.1\%$ disturbance the loss proved to be zero for the null space method). For the $\pm 1\%$ disturbance the loss is close to zero when using the null space method. It is possible it is not zero anymore at this point because the disturbance is large enough so that the linearization is not valid for the null space method anymore. Due to time issues, we did not run the control structure with implementation error. In future this should be done, in order to get a proper comparison with the data-based method to the null space method. We expect that in the case where the H-matrix is found using the null space method, implementation error would cause result from the null space method to perform poorer. Only then will it be truly interesting to compare the loss to the data-based method.

For our case the comparison to the exact local method is more interesting. In the cases with 10 measured variables (CASE A and B) the exact local method yields a lower loss compared with the data-based method. However, we found that in the cases with 5 measured variables (CASE C and D) the H-matrix calculated using the data-based method, for the most part, resulted in the lowest loss. In some cases the loss is actually significantly lower than for the exact local method.

Naturally, since this is only an average value of five simulations we cannot reach a final conclusion. However, the results found indicate that the data-based method can give equally and even better results compared to the exact local method. This trend, combined with the simple usability of the method (no need for a process model) gives good reasons to further develop and research the method.

Chapter 7

Final discussion and recommendation to future work

The approach investigated in this thesis was first developed and presented by Johannes Jäschke and Sigurd Skogestad in the article ´´*Using Process Data for Finding Self-optimizing Controlled Variables*" [7]. Based on this article, we have further tested ans investigated different aspects of the method. The idea described in the article uses plant data to estimate a quadratic cost function. It then uses the parameters found from the model estimation, to find the best way to combine measurement variables as a self-optimizing variable. The control variables are found as an estimate of the cost function gradient. Two of the main advantages is that this approach needs neither a process model nor disturbance measurements. Alternative methods are the exact local method and the null space method, among others.

7.1 Applicability

We found that in most cases the exact local or the null space method gave a better result than the data-based method. However, these are model based methods and their performance rely on a well functioning process model. Such a model can often be difficult to obtain and in some cases it will not be available at all. In these cases, methods like surface response methods[22] or extreme seeking [21] can be used. However, as pointed out by Jäsche and Skogestad [7] surface response methods rely on disturbance measurements which are often not available, and extreme seeking requires excitation of the process.

The data method is, along these lines, much simpler to use in practice. It does not require a process model, it can handle unforeseen disturbances that cannot be measured and does not require any extreme changes in the process operation to obtain necessary data. In order to calculate the combination matrix to find a good self-optimizing control structure it uses measurements from the process and the measurement gain matrix. The measurement gain matrix is easily found from a small step change in the inputs. Process measurements are normally available because process variables are already measured for supervisory reasons. Through the work done in this thesis we found the method to be easy to use. Once the MATLAB code (Appendix C) for running the data-method was well developed for one test case, it was rather easy to modify it and reuse the same code for the next test case. To start analyzing and applying the method to a new test

case, the degrees of freedom and number of disturbances must be known ($n_u + n_d$), along with the measurement gain. This is considered minor process information, compared to the extensive knowledge needed in order to use for example the exact local method.

However, the data based method rely on measurements of the cost function, the variable we wish to minimize (or maximize in cases with a profit function). For the cases where this is for example heating used in a heat exchanger or steam used in a distillation reboiler it is usually easily measurable. But there are cases where the cost function may not be that easily detected. In some cases the relationship between the process variables and the cost function may not be clear and therefore harder to model. This has not been studied in this thesis, but are interesting questions worth further investigation.

Moreover, to be able to use the data-method the measurements must lay close to the optimal value. This implies that in order for the data method to give a valid result, the process must be operated close to optimal operation when the measurement are sampled. The reason for this is that the method begins with Taylor approximating the cost function around the nominal optimal point. If the data used to estimate the cost function is not around this point, the model estimation will off course be poor. We see this as maybe the main downside to this method. In the test cases used in this thesis, it was easy to ensure that the data samples were around the optimal value. We could start by optimizing the process and then create the data around the nominal optimal point. In real cases this may not be so simple. In real process plants it can be difficult to know if the plant is optimally operated or not, and therefore we cannot know if the measurements from the plant can be used in the data-based method. How far from the optimum the data can be and still be applicable in the data-based method, is a topic of high interest in this context, but not addressed in this thesis. This question is left for future research.

As explained above the approach is based on a linear approximation of the cost function around the optimal point, which means that the solution is local. Consequentially, the approximation is valid only in this area. The result found can therefore not be extrapolated outside this range. Additionally, the validity area is limited of the active constraints. If the magnitude of a disturbance causes the active constraints to change, the preconditions for the calculation is also changed. The combination matrix estimated with the data method is valid only within a specific region where the active constraints are unchanged. Thus, the data method can only be used for disturbances of a certain size. Otherwise the control structure found, will perform poorly or even turn out to be unfeasible. For the test cases studied here the disturbances were never large enough to change the active constrain region.

7.2 Evaluation

The focus in this thesis has been to examine the data-base method by applying it to case studies, and change some of the preconditions and parameters used for the calculation of the H-matrix. How well one combination of measurements was, compared to another, was evaluated by using loss calculations. Either by using a mathematical expression for the loss as in the evaporator test case, or by comparing the cost function values when the process was re-optimized for a disturbance or controlled using the combination matrix. The difference between the two cost function values ($J_{opt} - J_{ctrl}$) was then the loss. Of the loss was small this indicated a good self-optimizing control structure.

The main topics addressed in this thesis has been:

- How well does the data-method work compared to model-based methods.
- How well does the method handle measurement noise in the data samples.
- How to identify the ideal number of components to use in the PLS-regresseion.
- How many data samples (ns) should be used in the calculation.
- If the disturbance can be measured, what impact does this have on the outcome from the method.
- How many measured variables (n_y) should be used in the calculation.

We have tried to answer these questions by using two case studies; an evaporator process and a CTSR-reactor and distillation column with recycle. The main difference between the case studies is how the loss is calculated.

Next we have summarized the main findings from our research followed by recommendations for future work.

7.2.1 Other methods

In most cases the exact local method was found to give a better result than the data-based method. But this is a well known modeled based method, and for well defined cases such as our test cases it is not surprising that it works well. However, there were some cases where the data-based method resulted in a lower loss. In the CSTR-distillation case study the data-based method proved to be better than the exact local method. It is interesting to observe that a method requiring so little information about the system at hand can actually find a better control structure than modeled based methods such as the exact local method. Based on these findings, we highly recommend further research on this approach to find self-optimizing variables.

7.2.2 Measurement noise

The question of how measurement noise is handled by the data method is also addressed in Jäscke and Skoestads article [7]. They concluded; "*The loss values for the simulation case with noise and the case without noise have been shown to be very similar*". The case studies completed in this thesis shows that using data samples with measurement noise actually results in a lower loss compared to the cases without measurement noise. Thus, we conclude that it is an advantage to use data with measurement noise. However, we found some exceptions where the loss was higher when measurement noise were included in the data. Thus, more research on this topic is needed. We still do not know how large the the measurement noise can be before the noise becomes a disadvantage instead of the advantage we found it to be. We speculate that the H-matrix is more robust when it is modeled using data with measurement noise, but little is known for sure why the result turned out to be better with measurement noise in the data samples.

7.2.3 Disturbance measurements

As stated earlier, one of the advantages with this method is that the disturbances do not need to be known or measured. Nevertheless, it is still interesting to know the effect of including the disturbances as measured variables for the cases where disturbances can be measured. The disturbance is the direct cause why the system changes,this makes the modeling of the cost function as a function of measurements more straight forward. And as expected the loss decreased significantly when the disturbance was included in the measurement matrix.

7.2.4 Number of components and number of samples

Jäscke and Skoestad wrap up their article by pointing out two topics that remained unanswered[7]. Namely how many number of components to use in the PLS-regression and how many data point or sample sets are needed in order to obtain a good model? In this thesis we have suggested different ways to identify the ideal number of components, and compared the outcome of using the "wrong" number of components. From the preliminary tests we have run on the matter we were not able draw any final conclusions. We observe that there is an effect both on the model accuracy and the result in terms of loss when the number of components is changed. We feel however that substantially more knowledge about the role the number of components-parameter has in the PLS-regression is needed to explain the effect further. As far as we can see from our test cases the ideal number of components seems case specific and depending on the data set used in as a basis for the calculation. Due to the importance we recon the number of components has on the outcome on the modeling, and the quality of H as a control parameter, we suggest this as a relevant topic for more research to develop the understanding of this method further.

An interesting finding from the evaporator case study, was that the number of data points used in the modeling had little impact on the loss. For the second case study discussed in Chapter 6 the difference between using many or few number of samples is minor. There might be a weak trend that using many (in this case 1000) data samples will all over lead to a lower loss compared to using few data samples (in this case 50 or 100). However, due to restricted time and computational limitations, we did not run enough cases to get a representative average.

7.2.5 Measured variables

The result from using many or few measured variables where different between the two test cases.

In the CSTR-distillation case study the loss was significantly reduced when we decreased the number of measured variables from 10 to 5. In this case study it would be interesting to first change the measured variables to a different combination of 5 variables, and second to reduce the number of variables even further to see if the effect continued or if the result got worse. Conversely, in the evaporator case study using 10 (not 5) measured variables gave the lowest loss. Also in this case it would be interesting to know if the result would change if we used a different set of measured variables.

This suggests that factors like, how correlated the measurements are, and which variables are included in the measurable matrix, are likely to influence the result. For large process plants where maybe thousands of variables are measured. Knowing how to choose the right variables to combine as a self-optimizing variable this is definitely an interesting question. It is therefore

necessary to gain more knowledge on what variables should be included in the model estimation, why and how many.

7.2.6 The difference between high model accuracy and good results in terms of loss

In both the evaporator and CSTR test cases we studied both model accuracy in terms of residuals between the estimated and measured cost function, and the loss values by using the H-matrix found from the modeling. When comparing the loss values and residuals we made an interesting observation. There seems to be no guarantee that a good model of the cost function, i.e a small residual, will result in a well performing H-matrix. For example, in the evaporator test case, the model accuracy increased with increasing number of components used in the PLS-regression. However, the calculated loss seems more or less unaffected by the increase in number of components. In addition, high loss values occurred for some number of components values, and these spikes was not observed in the residual indicating the model accuracy.

Another example is from the CSTR-distillation case study. We found no clear trend in the loss values when the number of data set (ns) used in the modeling was changed. However, the model accuracy, in terms of the residual $J_m - J_{est}$, was found to be more accurate when few data sets (in this case 50) was used, than when many data sets (in this case 1000) were used (Table 6.4). In the same case study we found that using 5 measured variables gave the lowest loss, yet the model accuracy was lower in these cases compared to using 10 measured variables.

The last example is the cases with and without measurement noise. Using data without measurement noise gave as explained earlier a more accurate, however less robust, model. And when using the H-matrix estimated on the basis of data with measurement noise, the result in terms of loss was better.

These examples indicate that a good model will not necessarily result in a well working H-matrix. This can be explained in the same way as we explained why 5 measurements gave a lower loss. When modeling the cost function, we use all the beta-matrix given from the PLS-regression to estimate:

$$J = \begin{bmatrix} 1 & Y_{aug}^{'} \end{bmatrix} \beta$$

Whilst for the H-matrix estimation we use only parts of the beta to estimate J_{yy}. We explain why a high model accuracy not necessarily results in a low loss with the following theory; in some cases it might be that the first elements in beta, the elements used to model the linear part of the cost function, are more accurately modeled. The last part of beta is used to estimate J_{yy} the second derivative matrix. It might be that these elements are very small or not very similar, and using one beta value instead of another in this second derivative matrix will not effect the model accuracy significantly. However, the accuracy of J_{yy} is important in the estimation of the H-matrix. This can be why a overall god estimation of the cost function still do not lead to a well working H-matrix.

7.2.7 Validity of the results

As mentioned throughout this discussion, the validity of these results are questionable due to the lack of a representative average. Some of the simulations run in the test cases turned out to be rather time consuming and the basis for the average calculation was therefore sat to five. Off course five test-runs are not representative enough to get a statistically valid result. However, we ran many different parallel cases changing only one parameter at the time. In some cases, like reducing the number of measurement from 10 to 5 in the CSTR-test case, or including the disturbance, the trends are considered as indicative and can lay a foundation for future research. As for questions, such as identifying the ideal number of components, some important issues have been highlighted, even if we are not able to conclude with anything.

7.2.8 Topics not discussed in this thesis

During the discussion presented here some topics in need of more research has been pointed out. As well as reasons why we were not able to make any final conclusions. Among these we consider gaining more understanding about the role the number of components play in the regression as especially important, as well as how to decide how many and which measured variables should be used. We suggest to continue the work started on the CSTR-distillation case study. The model in MATLAB for this process is working well, also together with the new codes for applying the data based method. Settings for the process itself, control structures and preconditions for the modeling can easily be changed. The main focus can therefore be to gain better understanding of the effect of chaining different parameters, rather than developing new codes for a process. We find this a good starting point for future research.

A topic not discussed in this thesis is the factors considering the data set itself. We suspect that changing the variation in the generated data will effect the result of using this method. Along these lines, it is interesting to gain more knowledge on how close to the nominal optimal point the data set must be. As well as how does the correlation between the different variables effect both the modeling and the performance of H as a control parameter. Although issues about the data sets are not studied in this thesis, we find this topic worth further investigation.

The results found in this thesis demonstrate the potential of the data-based method. The original idea for this thesis, to use a large process model of a biodiesel plant to test the data-based method is still interesting. We recommend to develop a way to run ChemCad through a programming language (such as Pyton) to be able to optimize and control the biodiesel plant. And basically conduct the same tests performed on the CSTR-distillation case study, with the biodiesel plant as a test case.

7.3 Conclusion

We believe the work presented should inspire to future research on this promising method to find self-optimizing variables. This initial research on the topic indicates the potential for the data-based method as a way to find self-optimizing variables. Little information about the process is needed in order to use this method. It can be used in most cases where we have access to empirical data of plant variables and the cost function, together with the measurement gain matrix. The method is thus a good alternative in cases where no process model is available. Though it was not found to be the general trend, there were cases where the data-based method worked better compared to the model based exact local method.

By the research presented here we have found some possible ways to improve the performance of the method, and the general understanding of the method is developed. Although we are not able to conclude anything definite from the case studies in this thesis, we have found indications that including the disturbance and using data samples with measurement noise can improve the result from the data-based method. Additionally, we have identified the importance of the number of measured variables used in the modeling. Likewise, we believe the number of components or directions in the data used in the PLS-regression is important for the outcome of the method. However, on this topic we were not able to draw a conclusion and this is a question left for further research.

In the end the conclusion is, more research is recommended and needed.

Nomenclature

Symbol	Description
c	Control variable
d	Process disturbance
F	Sensitivity matrix
G_d^y	Disturance gain matrix
G_d	Disturance gain matrix
G^y	Measurement gain matrix
g(x)	Inequality constriant
H	Self-optimizing combination (or selcetion) matrix
h(x)	Equality constraint
J	Cost function
J_{opt}	Optimized cost function
J_m	Measured cost function
J_{soc}	Cost function when using self optimizing variable
J_{mod}/J_{est}	Modeled / estimated cost function
J^*	Nominal value of cost function
J_u	Cost function gradient (first derivative)
J_{yy}	Estimated second derivative of the cost function
L	Loss
ns	Number of data sample sets
n_y	Number of measured variables
n_u	Number of process inputs (DoF)
n_d	Number of distrubances
u	Process input
ud	Input and disturbance matrix
W_d	Disturbance matrix
W_n	Measurement noise matrix
x	System state variable
y	Measured variables
Y	Measurement matrix
y^n	Measurement noise
β	Model parameter

Case spesific symbols:

Evaporator process	
F_1	Feed flow of raw materials
F_2	Product flow
F_3	Stream flow to condencer
F_{100}	Steam flow
F_{200}	Cooling water flow
T_1	Feed temperature
T_{200}	Cooling water temperature
X_1	Feed composition

Case spesific symbols:

CSTR distillation column process	
A	Feed component
B	Flow rate bottom product
CSTR	Continuous stirred-tank reactor
D	Flow rate recycle stream
F_0	Feed flow rate - the disturbance
M_B	Level in column
M_D	Level of condensate
M_R	Reactor level
P_c	Pressure in column
V_B	Re-boiler duty
VLV-q	Valve number q
T_t	Temperature at tray t
Tb_L	Boilingpoint light component
Tb_L	Boilingpoint heavy component
x	Composition
x_B	Bottom product composition
y	Measurements without measurement noise and not including the disturbance as a measurement
y^d	Measurements without measurement noise and including the disturbance as a measurement
y_n	Measurements with measurement noise and not including the disturbance as a measurement
y_n^d	Measurements with measurement noise and including the disturbance as a measurement
z_f	Feed composition

Subscript or superscript	
*	Nominal optimal values
T	Transposed matrix
av	Average loss
aug	Augmented matrix
centered	centered matrix
ctrl	Control
data	Data-bases method
est	Estimated value
exl	Exact local method
ns	Null space method
d	With disturbance as a measured variable
dd	Second derivative of cost function wrt. to disturbance
du	Second derivative of cost function wrt. to disturbance and process input
m	Measured values
mean	average value of many figures
mod	Modeled
max	maximum value
n	number of ...
n	With measurement noise
raw	Raw data samples
scaled	Scaled matrix
uu	Second derivative of cost function wrt. to process input
ud	Second derivative of cost function wrt. to process input and disturbance
wc	Worst case loss
w/n	With noise

Abbreviation	
ChemCad	Chemical process simulation software
dist	Disturbance size in %
DoF	Degrees of Freedom
exl	Exact local method
ncpmp	Number of components
PLS	Partial least square

Bibliography

[1] Sigurd Skogestad *Economic plantwide control* . Chapter 11 in John Wiley and Sons: Plantwide control, NTNU, Department of Chemical Engineering, Trondheim, Norway, 2012

[2] Sigurd Skogestad, Ivar J. Halvorsen, Truls Larsson and Marius S. Govatsmark *PLANTWIDE CONTROL: THE SEARCH FOR THE SELF-OPTIMIZING CONTROL STRUCTURE* . NTNU, Department of Chemical Engineering, Trondheim, Norway, 1999

[3] Sigurd Skogestad *SELF-OPTIMIZING CONTROL: A DISTILLATION CASE STUDY* . NTNU, Department of Chemical Engineering, Trondheim, Norway, 1999

[4] Johannes Jäschke and Sigurd Skogestad *Controlled variables from optimal operation data.* E.N. Pistikopoulos and A.C. Kokossis, editors, 21st European Symposium on Computer Aided Process Engineering, volume 29 of Computer Aided Chemical Engineering, pages 753 – 757. Elsevier 2011

[5] Sigurd Skogestad *Plantwide control: The search for the self-optimizing control structure.* NTNU, Department of Chemical Engineering, Trondheim, Norway, 1999

[6] Sigurd Skogestad, Ramprasad Yelchuru and Johannes Jäschke *Optimal Use of Measurements for Control, Optimization and Estimation using the Loss Method: Summary of Existing Results and Some New.* NTNU, Department of Chemical Engineering, Trondheim, Norway, 2013

[7] Johannes Jäschke and Sigurd Skogestad *Using Process Data for Fining Self-optimizing Controlled Variables.* NTNU, Department of Chemical Engineering, Trondheim, Norway, 2013

[8] Sigurd Skogestad *Self-optimizing control: the missing link between steady-state optimization and control.* NTNU, Department of Chemical Engineering, Trondheim, Norway, 2000

[9] Kim H. Esbensen *Multivariative Data Analsys - in practice.* 5th edition CAMO 2004

[10] Svante Wold, Michael Sjöström, Lennart Eriksson *PLS-regression: a basic tool of chemometrics.* Research group for Chemometrics, Institute of Chemistry, Umeå Univercity, Sweden, 2001

[11] Skogestad and Minasidis *Excercise 3: Control Structure design for generic chemical plant process* NTNU, Department of Chemical Engineering, Trondheim, Norway, 2013

[12] Larsson, Govatsmark, Skogestad and Yu, *Control Structure Selection for Reactor, Separator and Recycle Processes.* NTNU, Department of Chemical Engineering, Trondheim, Norway, 2003

[13] Vidar Alstad, Sigurd Skogestad, Eduardo S. Hori *Optimal measuremetns combinations as conrtlled variables.* NTNU, Department of Chemical Engineering, Trondheim, Norway, 2008

[14] Vidar Alstad and Sigurd Skogestad, *Null Space Method for Selecting Optimal Measurements Combinations as Controlled Variables*. NTNU, Department of Chemical Engineering, Trondheim, Norway, 2007

[15] Vinay Kariwala, Yi Cao and S. Janardhanan *Local Self-Optimizing control with Average Loss Minimization*. Division of chemical and biomolecular engineering, Nanyang Technological University, Singapore and School of Engineering, Cranfield University, Cranfield, Bedford, United Kingdom.

[16] Ivar Halvorsen, Sigurd Skogestad, John Morud and Vidar Alstad, *Optimal Selection of Controlled Variables*. NTNU, Department of Chemical Engineering, Trondheim, Norway, 2003

[17] Vidar Alstad, *Control structure selection for an evaporator example*. Available online: "http://urn.kb.se/resolve?urn=urn:nbn:no:ntnu:diva-2976". Part of thesis: Alstad, Vidar. Studies on selection of controlled variables. 2005.

[18] Y. Cao, *Self-Optimizing control structure selection via differentiation*. In European Control Conference (ECC 2003), pages 445-453.

[19] M. Govatsmark and S. Skogestad, *Control structure selection for an evaporation process*. European Symposium on Computer Aided Process Engineering, May 27-30, 2001, Kolding, Danmark. See C01-1, Volume 1, HS2001-6.

[20] R. Newell and P Lee *Applied process control - A case strudy*. Prentice Hall.

[21] Miroslav Krstic and Hsin-Hsiung Wang *Stability of extremum seeking feedback for general nonlinear dynamic systems*. Automatica 46(11): 595-601, 2000

[22] George E. P. Box and Norman R. Draper *Empirical Model-bulding and Response Surfaces*. John Wiley, New York, 1987

Appendix A

Test-case two: CSTR and distillation column with recycle

Additional information

A.1 System details

In this appendix the main important details of the CSTR-reactor distillation process are given (Figure A.1). The system equations are used to simulate the process, both to create data used to calculate the H-matrix, and to create a model in MATLAB to simulate the process with control and re-optimize it for disturbances.

For the modeling we assume that the reaction is an irreversible first order elementary reaction of: $\mathbf{A} \rightarrow \mathbf{B}$. We also assume that teh reactor temperature is so well controlled that the reaction rate constant (k_1) is constants. The product stream from the reactor consists of both A and B, which is separated in the column. The product is the bottom stream (B) and A is the distillate which is recycled back to the reactor. The relative volatility between the two is assumed to be 2.

A.1.1 The main equations

The main equations for the system is given in this paragraph. The mathematical model and steady-state equations are taken from an exercise given in a process control course **??**.

Steady state equations

Because of the law of mass conservation the overall mass balance for the system is:

$$F_0 = B$$

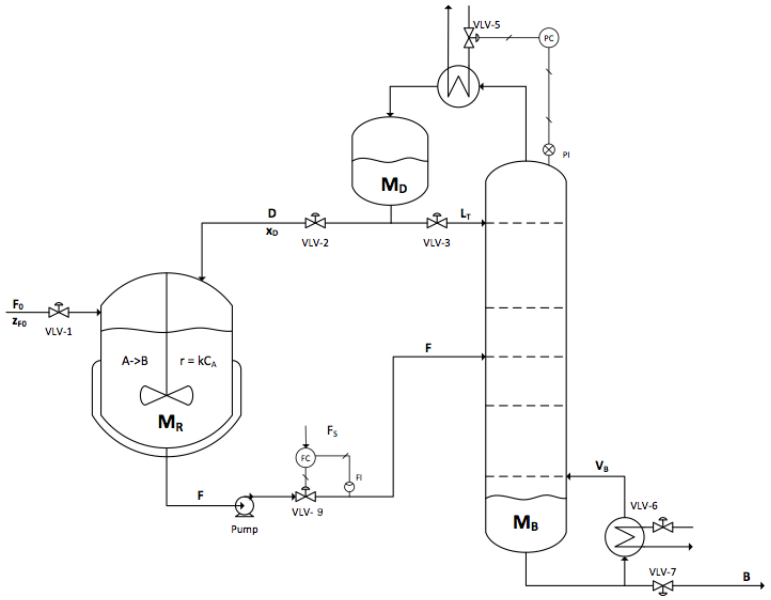


Figure A.1: Process plant with a CSTR-reactor and a 22 stage distillation column with recycle [11].

We can also construct the mass balance for each of the equipment. For the reactor:

$$F_0 + D = F$$
$$F_0 z_f + D x_d = F z + M_r k_1 z$$

For the columns: And more specific for the reactor:

$$F = D + B$$
$$F z = D x_D + B x_B$$

Mathematical (steady state))model

Assumptions for the column

- Constant pressure (by control).
- Constant relative volatility equal to 2.
- Constant molar flows
- Total condenser (all distillate recycled back to the reactor is in liquid phase).

Assumptions for the reactor

- The reactor is a CSTR reactor.

- The reaction has first order kinetic.
- The temperature is under perfect tight control and can therefor be considered constant, due to this the rate reaction constant does not change.

The **column** is modeled using the following equations:

Overall mass balance:

$$B = F + L - V$$
$$D = V - L$$

Component balance ofr tray i and for the feed tray:

$$Vy_{i-1} + Lx_{i+1} = Vy_i + Lx_i$$
$$Fz_F + Vy_{i-1} + Lx_{i+1} = Vy_iLx_i$$

Vapor-liquid equilibrium of component i, where a is the relative cilatility:

$$y_i = \frac{ax_i}{1 + (a-1)x_i}$$

The **reactor** is modeled using the following equations:

Overall mass balance:

$$F_0 + D = F$$

Component mass balance:

$$F_0 z_{F_0} + Dx_D = Fz_F + k_1 M_r$$

A.1.2 Nominal point of operation and optimal operation

To find the nominal operational point the process is optimized using *fmincon* in matlab. The nominal operational is therefor the optimal operation point when the process is run without any disturbance. The nominal operation values for all the system parameters are given in Table A.2.

An important value is the cost function, which is this case is the amount of steam used to heat the re-boiler. For each disturbance inflicted on the system the cost function was found by re-optimizing the system (Table A.1). The optimal value of the cost function changed, but it is important to note that the active constraints did not.

These values are compared to the cost function value when the process is controlled using a H-matrix as a combination matrix for self-optimizing variables instead of re-optimizing for the disturbance.

Table A.1: Optimal value of the cost function when the feed rate is changed

Disturbance in the feed	Cost function value [kmol/h]
0 %	1275.7
+1%	1301.3
-1%	1250.6
+5%	1408.1
-5%	1154.2

Table A.2: Nominal optimal operation values for the reactor distillation plant

	Symbol	Value	Units
Feel flow rate	F_0	7.6667	kmol/min
Reactor outflow	F	15.9667	kmol/min
Vapor boilup flow rate	V_B	21.2667	kmol/min
Reflux flow rate	L_T	12.9667	kmol/min
Distillate (reflux) flow rate	D	8.2833	kmol/min
Distillate composition	x_D	0.82	
Condensate molar holdup	M_D	212.5	kmol
Bottom product flow rate	B	7.6667	kmol/min
Bottom product composition	X_B	0.0105	
Reboiler molar holdup	M_B	147.5	kmol
Reactor composition	z_f	0.43	
Reactor holdup	M_R	2800	kmol

A.1.3 System matrices

For each of the measurement combinations there is a set of gain matrix, sensitivity, disturbance and measurement noise matrices.

The gain matrix is used to calculate H with the data based method. While the disturbance gain, sensitivity and noise matrix is used in the exact local method and in the null space method. The optimal sensitivity F is how sensitive the optimal measurements x_{opt} are with respects to the disturbances d. The measurement noise matrix is also used when the data used as a basis for calculating H. The matrices for all the four measurements cases are presented next.

10 measurements not including the disturbance in the measurements.

Gain matrix and Sensitivity matrix

$$G^y = \begin{bmatrix} 0.1504 \\ 0.3568 \\ -1.5075 \\ -1.3174 \\ -0.0006 \\ -0.8569 \\ -1.8563 \\ -0.0000 \\ -1.8563 \\ -0.0243 \end{bmatrix} \qquad F^y = \begin{bmatrix} 0.0529 \\ 0.1235 \\ -0.0321 \\ -0.1271 \\ -387.4932 \\ 8.9047 \\ 9.0066 \\ 1.0000 \\ 10.0066 \\ 0.1247 \end{bmatrix} \tag{A.5}$$

Measurement noise matrix

$$Wn^y = \begin{array}{c} \begin{matrix} T_8 & T_{13} & T_{18} & T_{22} & L_t & V_b & D & B & F & z_F \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.13 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.213 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.083 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.077 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.16 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.02 \end{bmatrix} \end{array} \tag{A.6}$$

10 measurements including the disturbance in the measurements.

Gain matrix and Sensitivity matrix

$$G^{y_d} = \begin{bmatrix} 0.1504 \\ 0.3568 \\ -1.3174 \\ -0.0006 \\ -0.8569 \\ -1.8563 \\ -0.0000 \\ -1.8563 \\ -0.0243 \\ 0 \end{bmatrix} \qquad F^{y_d} = \begin{bmatrix} 0.0529 \\ 0.1235 \\ -0.1271 \\ -387.4932 \\ 8.9047 \\ 9.0066 \\ 1.0000 \\ 10.0066 \\ 0.1247 \\ 1.0000 \end{bmatrix} \tag{A.7}$$

Measurement noise matrix

$$Wn^{y_d} = \begin{array}{c} \begin{matrix} T_8 & T_{13} & T_{22} & L_t & V_b & D & B & F & z_F & F_0 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.13 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.213 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.083 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.077 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.16 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.02 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.38 \end{bmatrix} \end{array} \tag{A.8}$$

5 measurements not including the disturbance in the measurements.

Gain matrix and Sensitivity matrix

$$G^y = \begin{bmatrix} 0.1504 \\ -1.3174 \\ -0.0006 \\ -0.0000 \\ -1.8563 \end{bmatrix} \qquad F^y = \begin{bmatrix} 0.0529 \\ -0.1271 \\ -387.4932 \\ 1.0000 \\ 10.0066 \end{bmatrix} \tag{A.9}$$

Measurement noise matrix

$$Wn^{y} = \begin{array}{c} \begin{matrix} T_8 & {}_{22} & L_t & B & F \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0.13 & 0 & 0 \\ 0 & 0 & 0 & 0.077 & 0 \\ 0 & 0 & 0 & 0 & 0.16 \end{bmatrix} \end{array} \tag{A.10}$$

5 measurements including the disturbance in the measurements.

Gain matrix and Sensitivity matrix

$$G^{y_d} = \begin{bmatrix} 0.1504 \\ -1.3174 \\ -0.0006 \\ -0.0000 \\ 0 \end{bmatrix} \qquad F^{y_d} = \begin{bmatrix} 0.0529 \\ -0.1271 \\ -387.4932 \\ 1.0000 \\ 1.0000 \end{bmatrix} \tag{A.11}$$

Measurement noise matrix

$$Wn^{y_d} \begin{array}{c} \begin{array}{ccccc} T_8 & {}_{22} & L_t & B & F_0 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0.13 & 0 & 0 \\ 0 & 0 & 0 & 0.077 & 0 \\ 0 & 0 & 0 & 0 & 0.38 \end{bmatrix} \end{array} \tag{A.12}$$

A.1.4 Converting from composition to temperature measurements

When the feed changes it effects the compositions at each tray in the column, however for small disturbances in the feed the change in composition will be very small and therefore hard to detect. For this reason, it is desirable to use a more sensitive measurements which will be easier to detect also for small changes in the feed. The convertion is done by using the formula given in Equation A.13.

$$T_x = T_b^L x + (1 - x) T_b^H \tag{A.13}$$

where T_b^L is the boiling temperature of the light component, in this case A, and T_b^H is the boiling temperature of the heavy component, in this case B. The boiling temperatures are set to be

Component A, T_b^L: 353 K

Component B, T_b^H: 373 K

A.1.5 Model validation

The testing of the modeling was done as described in Section 4.3.2 using the optimal number of components. The residual between the measured cost function and the estimated cost function by using beta was found for all 8 test cases for a data set of 50 samples. The average residual is given in Table **??** is calculated from calculating the average residual ten times. The plot of the residual for the first simulation for all the samples are presented in FigureA.2 for 10 measurements and Figure**??** for 5 measurements.

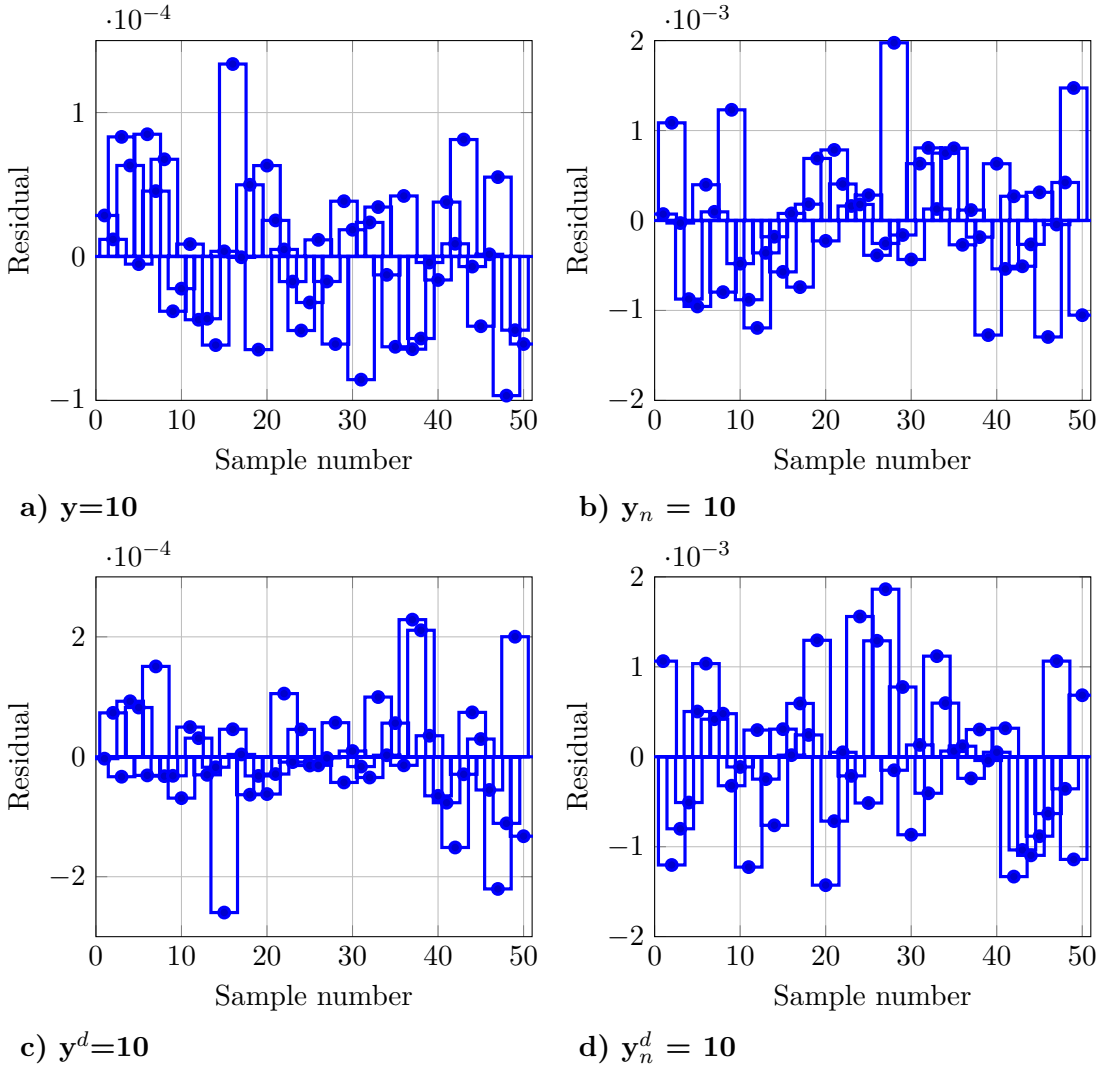


Figure A.2: For 10 measurements : $J_{measured} - J_{test}$.

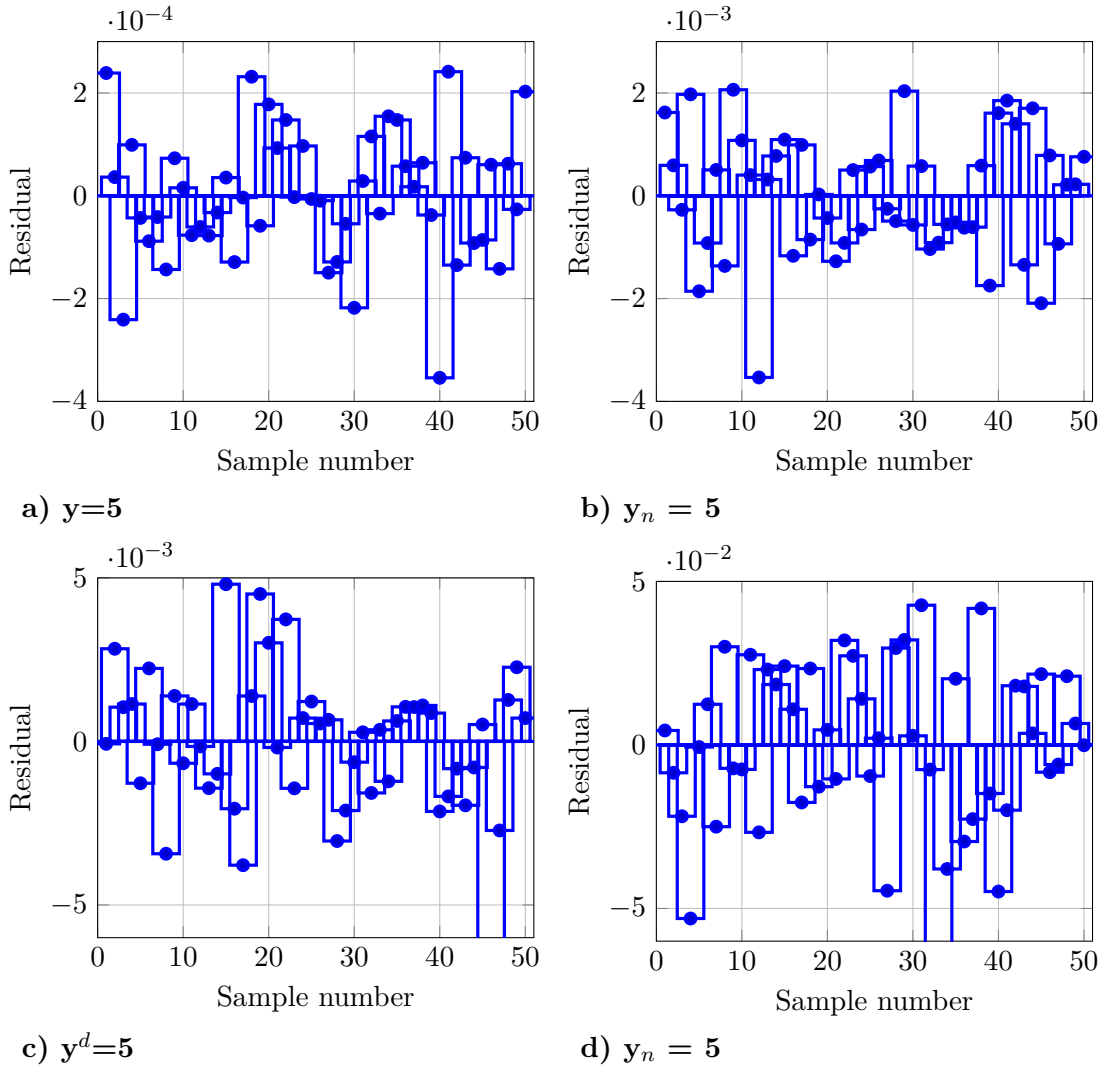


Figure A.3: For 5 measurements: $J_{measured} - J_{test}$.

Appendix B

The biodiesel plant

In the first 4-5 weeks of the project we were working with a a biodiesel model in ChemCad to generate data. This appendix contains the initial analysis of the biodiesel plant. It is included in this thesis as an appendix since it can be used in the future if someone decides to use it as a test case to study the data-bases method.

B.1 Creating data to test the data-based method

As explained earlier the scope of this thesis is to test the newly developed method to find the H-matrix by using historical plant data. In order to test the method to find the optimal H we need plant data to use in the method. To gather plant data we will use a model for the Esterfip-H process simulated in Chemcad. The model is developed by Marianne Øien in her master thesis from June 2013. A short summary of the most important features of the model will be given followed by a degree of freedom analysis and a discussion of possible disturbances. The reader is referred to Øiens thesis for more details and a more comprehensive discussion of the model.

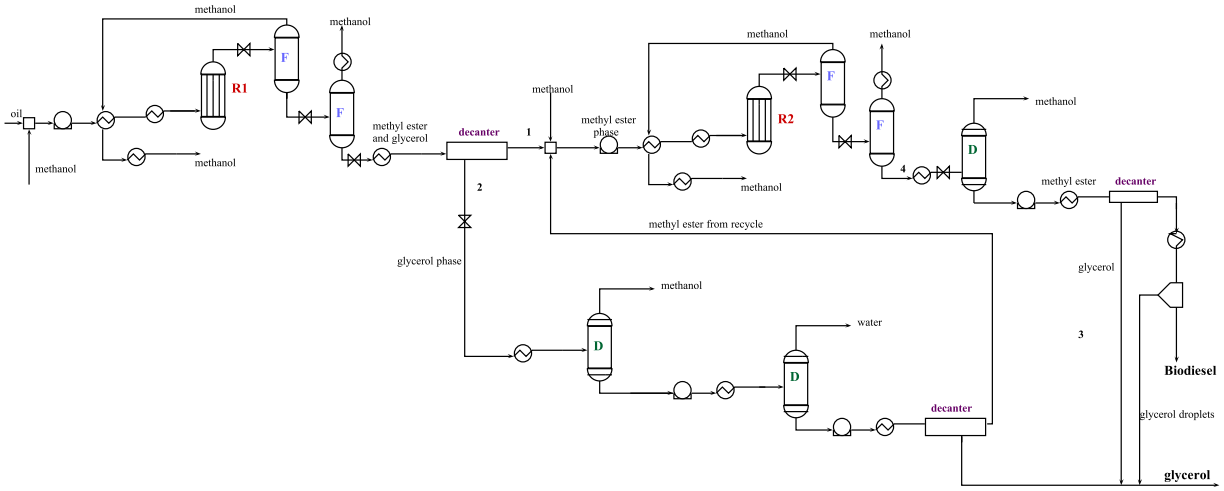


Figure B.1: Process flow sheet: The Esterfip-H process

The Esterfip-H process

The products from the Esterfip-H process are biodiesel and glycerol, the feed to the process consists of vegetable oil, in this case rapeseed oil, and methanol. The overall reaction is reacting triglyceride with methanol, giving the products glycerol and Methyl Ester. The process is a heterogeneous catalytic process, which means that the phase of the catalyst differs from the phase of the reactants. In this case the catalyst is solid, and consists of zinc and aluminum oxide. By using a heterogeneous catalyst problems related to formation of salt with the catalyst which leads to emulsion between the methyl ester and glycerol, is avoided. This makes the separation process after the reaction less problematic.

The process primarily involves two fixed-bed reactors (with catalyst), flash devices, distillation columns and shell-tube heat-exchangers. The overall process is presented in the flow sheet below.

Methanol is fed to the system so the reaction is in excess of methanol. The remaining methanol after the reaction is flashed off in two stages, before the two phases, methyl ester and glycerol, are separated in a decanter. The the separation is carried out based on the difference in density of the two phases. Methanol is added to the stream with mainly a methyl ester phase, this mix is then reacted in a second fixed bed reactor. The methanol is flashed off in the same matter as the previous step before (almost all) the rest of the methanol is removed by vacuum distillation. The glycerol is removed by a decanter, and any droplets left are removed by a coalescer, leaving a product stream with methyl ester (biodiesel) with a purity of minimum 96wt%.

Methanol is removed from the glycerol-phase stream by distillation and due to a close boiling temperature for glycerol and methanol, this is the most energy consuming step in the process. Water is the removed from the glycerol by distillation. One of the benefits with the Esterfip-H process compared to a conventional process is the high glycerol purity.

B.2 Cost function

As explained the first step in finding a good control structure is to define an objective function. The objective function usually has an economical point of view, and it can for example be minimizing energy consumption or maximizing production and profit. In this case it is desired to maximize the biodiesel and glycerol to production while save money on limiting the energy usage. However there are strict purity demands on the products than cannot be violated. Most beneficial would be if the energy consumption could be minimized while still satisfying the product specifications.

The income from the process are selling the biodiesel and the glycerol, the profit function can be expressed as:

$$\text{Profit} = P_{bd} \times m_{bd} + P_g \times m_g$$

Where P_{bd} is the price for biodiesel, m_{bd} is the production rate of biodiesel, P_g is the price for glycerol and m_g is the production rate of glycerol.

The expenses are buying raw materials, and steam and cooling water to heat up or cool down streams. In addition expenses for electricity must be taken into account. The expenses can be expressed as:

$$\text{Expenses} = P_{bd} \times m_{st} + P_w \times m_w + P_{el} \times E$$

Where the cost for electricity (P_{el}), steam prices (P_{st}) and the cost for cooling water (P_w) are separated as different expenses.

The cost function, J, is the profit minus the expenses:

$$J = P_{bd} \times m_{bd} + P_g \times m_g - P_{bd} \times m_{st} - P_w \times m_w - P_{el} \times E$$

The cost function is affected by the production rate, meaning the amount that is produced and the amount of steam, cooling water and raw material utilized, as well as the prices.

B.3 Degrees of Freedom

In order to optimize the process we need at least one degree of freedom, which can be manipulated to gain the best possible operation of the process under given conditions. In the degrees of freedom analysis we will go through all the main equipment in the process and which variables that are important associated with the equipment. In total there are six variables left as degrees of freedom, and they are all associated with the distillation columns.

The reactors, R1 and R2 The steam before the reactor is pre-heated, to give the desired temperature conditions for the reaction. Both the level and pressure must be controlled in the reactors, and since there are only two outlet streams form the reactor, there are no degrees of freedom to be adjusted for the reactors.

The flash tanks, F1, F2, F3 and F4 The inlet stream to the flash tanks are the product stream from the reactors. The stream contains methanol, methyl ester and glycerol. The level in the flash tanks are controlled by the outlet stream at the bottom, which mainly consists of methyl ester and glycerol. The methanol is flashed of as vapor, and the pressure is controlled by the vapor outlet stream. There are therefor no degrees of freedom to adjust for the flash tanks.

Decanter, DEC1, DEC2, DEC3 The decanter simply separate the methyl ester phase from the glycerol based on the difference in density. The level and pressure are controlled with the glycerol and methyl ester outlet streams respectively.

Distillation columns, D1, D2 and D3 For all the three columns the feed flow to the distillation columns are pre-heated by steam in a heat-exchanger. The pressure in the column is controlled by the condenser, the reflux level is controlled by the distillate streams and column level is controlled by the outlet streams at the bottom. For all the columns the bottom product is the most important and has a purity constraint. Each column has two degrees of freedom, which two depends on the control structure for the column. Initially, the reflux rate and the reboiler boil-up are the two variables that can be manipulated. This leaves the columns with a L/V-configurations.

B.4 Disturbances

In the Esterfip process there are several disturbances that a control system must be able to handle. We hope to find good control variables or combinations of variables that when kept constant still keeps the plant close to optimal operation.

The disturbances we will be looking into when designing a self-optimizing control structure for the plant are disturbances in the feed and prices. The feed rate and the temperature for the oil and methanol feed can vary, as well as the amount of methanol added to the plant. In addition, prices will also effect the optimal way of operating the plant. Steam is used to heat up streams before the reactors and distillation columns, and the steam prices can vary, effecting the optimal feed rate for example. The feed and product prices will very much effect the optimal way of running the plant. The optimal production rate depends on if the feed stock prices are high or low, and the same with the product prices.

Feed rate

If the feed composition changes the amount of energy required to achieve the desired purity of especially the bottom products will change. For example if the mass fraction of methanol increases, more energy is needed in the columns to separate the phases. The composition of the oil will also effect both the reactor conditions and the separation processes downstream.

The reflux ratio used in the columns depend on the ratio of oil or glycerol and methanol in the stream. If the ratio between methanol changes and oil (or glycerol), the optimal reflux ratio will also change.

If the feed rate increases the residence time in the reactor will go down. As a result the conversion will go down as well, leaving less product and more methanol in the outlet stream from the reactor. This will effect both the separation processes and the amount of recovered methanol.

Heating and cooling

Heating in the process is mostly carried out by counter current heat exchangers. The hot streams are either hot methanol from the flash tanks or steam. Streams or hot distillate from the columns are cooled down by cooling water. Both the steam and the cooling water are expenses, and the prices can vary. The cooling water is considered very cheap relatively to other expenses and chances in prices will be ignored. A change in the steam prices however, can effect the optimal way of running the process.

Feed and product prices

Product and feed prices can also vary with the demand in the market. There are generally two main modes of operation depending on the market conditions [1]. Mode one is ”buyer's market“ where the throughput is given. The process is then run with a trade-off between energy consumption and recovery of product. The second mode is when the product prices are relatively much higher than raw materials and energy prices. In this case it is optimal to increase the throughput as much as possible, and run the process at maximum production.

B.5 Constraints

The obvious constraints are the flow capacity in for pumps and pipes, as well as the level in all the equipment. All the levels in tanks, columns and decanters are controlled by one of the outlet streams such that there are no over-flooding of equipment.

As mentioned a certain residence time is required for the reactants in the reactors. The flow rate is therefor limited by a maximum flow rate, so that the reactants gets a minimum residence time in the reactors.

If the amount of methanol into the system increases, more heat is needed. There is however an upper limit on how much energy the reboiler can provide depending on the size. As a result there is a limit on how much methanol that can be added. For the model used here the upper limit of methanol present in the methyl ester phase is 0.6 wt%. There is a trade-off between the amount of methanol present in the flow; a higher concentration of methanol will give increased purity and drive the reaction in the desired direction, but also increase the cost because more energy is needed in the columns to separate the methanol from the glycerol or the methyl ester. The same principle goes for the condenser, since there will be a maximum amount of hot distillate that can be cooled to a desired temperature.

Back-off

The control of the distillation columns is crucial, because the purity specification on both biodiesel and glycerol are very strict. There is therefor necessary with a back-off on the set point for the product composition. The back-off is to ensure that the purity constraints are not violated.

B.6 Operational settings

The operational settings for the model are briefly described here, the sizes and initial conditions are taken from patents described in Øiens Master's thesis [**?**]. In addition the same assumptions made by her for her model will also be made here, in order to be able to use the already developed model directly.

Heating and cooling of streams When streams are heated by using steam the steam is cooled from 500 ^{o}C to 230 ^{o}C at 27 bar. The cooling water is heated from 6 ^{o}C to 140 ^{o}C at 4 bar. It is assumed that this is valid for all the heat exchangers. We will also assume that no phase transition occur.

The reactors The reactor conditions are important to achieve the desired conversion and to avoid undesired side-reactions. The reactor is a packed bed reactor with zinc aluminate catalyst ($ZnAl_2O_4$). According to the model used in this case the reactor volume is 60 m^3, the temperature is 483 K and the pressure is 62 bar. The transesterfication reaction is endothermic and a high temperature will increase the reaction rate [**?**], however a too high temperature will lead to degeneration of glycerol [**?**].

Phase separation and methanol removal The methyl ester and the glycerol are separated in decanters. The decanters are operated at atmospheric pressure and 50^{o}C. In order to get the best possible phase separation there is a minimum settling time to allow also the smallest droplets to move to the phase interface.

The reaction is carried out with excess of methanol in order to push the reaction in the desired direction, following Le Chateliers principle. After the first reactor the methanol is flashed off, which makes the separation of methyl ester and glycerol easier. If not removed, emulsions between glycerol and methyl ester can occur making it harder to separate the phases. After the second reactor the methanol is also flashed off before the rest of the methanol is removed by distillation. There are two sets of operation conditions of the flash tanks, either at 5 bar og 2.5 bar. At 5 bar the boiling temperature for methanol is significantly higher than for glycerol and methyl ester, therefore the vapor phase flashed of will contain 99.9 wt% methanol.

Due to the high purity specification for biodiesel the methanol is also removed by distillation. After the flash tank the rest of the methanol is removed from the methyl ester phase by vacuum distillation such that the final product contains less than 0.2wt% of methanol. Vacuum distillation is particularly well suited for separation compounds that are miscible with water [**?**]. The column used in the model is a 10 stage column with an initial reflux ratio of 2. The product stream is also run through a decanter and a coalescer to remove glycerol. The maximum energy

± 20% of the original energy performance, this results in a maximum feed load of 754 kmol/h with a maximum mass fraction of 0.6 wt% methanol in the feed.

The difference between the volatility of glycerol and methanol is relatively large, which makes the removal of methanol from the glycerol phase rather easy. Due to the difference in volatility a flash tank could be used to separate the phases, however because of the high purity specification vacuum distillation is utilized. This is to ensure and have better control over the amount of methanol in the glycerol product. The column is a 10 stage column and an initial reflux ratio of 2.

Water removal Water is induced to the system through the oil and methanol feed, or by leaks in the plant. The presence of water is damaging to the system and therefore undesired. Water leads to formation of soap, and therefor emulsion between the methyl ester and glycerol. Methyl ester in the glycerol phase is recycled to avoid loss of valuable product, and if water is not removed from the system it will accumulate. The water is removed by distillation before the methyl ester and glycerol is separated, and methyl ester recycled.

B.6.1 Summary of operation conditions

Table B.1: Equipment conditions

Equipment	Volume [m^3]	Pressure [bar]	Tepmerature [K]	other	
Reactors (R1-2)	60	62	483	Catalyst density:	1540 g/L
Decanter (DEC1-3)		1	50		
Flash tanks (F1-4)		5 (or 2.5)			
Distillation column D1				Stages:	10
Distillation column D2					
Distillation column D3				Stages:	10

Table B.2: List of variables in the system

Variable	Symbol	Unit
Feedflow oil	m	kg/h
Feedflow methanol	m	kg/h
R1 temperature	T	K
R1 pressure	P	bar
Methanol recirculation (R1) temperature out	T	K
F1 pressure	P	bar
F1 level	L	m
F2 pressure	P	bar
F2 level	L	m
Methanol out of F2 temperature	T	K
DEC 1 pressure	P	bar
DEC 1 level	L	m
Methanol addition	m	kg/h
Feedflow og methyl ester to R2	m	kg/h
Methanol recirculation (R2) temperature out	T	K
R2 temperature	T	K
R2 pressure	P	bar
F3 pressure	P	bar
F3 level	L	m
F4 pressure	P	bar
F4 level	L	m
Methanol out of F4 temperature	T	K
Vacuum distillation:		
BP: Biodiesel VP: Methanol		
D3 pressure	P	bar
D3 reflux level	L	m
D3 reflux rate	m	kg/s
D3 reboiler heat	Q	J/h
D3 level	L	m
D3 reflux composition	x	mol/L
D3 feed temperature	T	K
D3 bottom product comp	wt	
DEC3 pressure	P	bar
DEC3 level	L	m
DEC3 Biodiesel temperature	T	K
Vacuum distillation:		
BP: Glycerol VP: methanol		
D1 pressure	P	bar
D1 reflux level	L	m
D1 reflux rate	m	kg/s
D1 reboiler heat	Q	J/h
D1 level	L	m
D1 reflux composition	x	mol/L
D1 feed temperature	T	C
D1 bottom product comp	wt	
BP: Glycerol VP: water		
D2 pressure	P	bar
D2 reflux level	L	m
D2 reflux rate	m	kg/s
D2 reboiler heat	Q	J/h
D2 level	L	m
D2 reflux composition	x	mol/L
D2 feed temperature	T	K
DEC2 pressure	P	bar
DEC2 level	L	m

Appendix C

MATLAB codes

C.0.2 "Dummy" case-codes

There are three scripts needed to run the dummy case:

- dummymasterfile.m
- dummygeneratedata.m
- calcH.m

The dummygeneratedata.m must be run first, and the master file loads generated data, and then run the calcH.m file to find the H-matrix. This way we ensure that the same data set is used as a basis, unless we by purpose run the generate data script again to generate a new data set.

In the master file we also set which case we want to run, case A or B. The case must off course correspond to the same case that the data were generated for. Each time the dummygenerate-data.m file is run the name of the .mat-file saved with data must also be changed, and the correct file must be loaded in the dymmumasterfile.m file.

The calcH.m file is run from the dymmumasterfile.m script to model the cost function and find the H-matrix.

```
1  %% DUMMY CASE : dummymasterfile.m %%
2  %% 07.06.14 %%
3  %% Number of variables, inputs and disturbances
4  clear all
5  close all
6  clc
7
8  % Choose the case you want to run %
9  %CASE A
10 ny =3 ;       % measurments
11 ns =50;       % samples
12 nu =2 ;       % inputs
13 nd =1;        % disturbances
14 load dummyAncomp4.mat
```

```
% nr = 9 ;
% % % % % % % % % % % % % % % % % % % % % % % % %
% CASE B
ny =6 ;       % measurments
ns =50;       % samples
nu =4 ;       % inputs
nd =2;        % disturbances
%load dummyBncomp10.mat
nr = 27;

% If you want to run the dummy case for only one ncomp
% nr = 1
Jdiff=[];

for sy=nr
    ncomp=nr ;   % if nr = 1 set to one cnomp value
    run calcH
    Jdiff(:,sy)=J_diff;
end

J_average = sum(abs(Jdiff(:,1:nr)));
%
% figure(1)
% bar([1:ncomp+1],J_average)


% Save, plot and write the data from the simulation:
figure(2)
bar([1:j],resi)
figure(3)
bar([1:ns], J_diff)

%% Saving data for plots
ns=[1:ns] ;
Jdiff=[J_diff];
sumdiff = sum(abs(J_diff))/50
filname=['ncomp' num2str(ncomp) 'Case' num2str(ny) '.mat'];
file = filname;
save(file, 'Jdiff' , 'sumdiff' )

%data = [ns' J_diff];
%dlmwrite('dataJdiffA4ncomp.dat',data,'delimiter','\t')
```

```
%% DUMMY CASE : dummygeneratedata.m %%
%% 07.06.14 %%
% % % % % % % % % % % % % % % % % % % % % % % % %
%% Number of variables, inputs and disturbances
% Choose the case you want to run %
% %CASE A
ny =3 ;       % measurments
ns =50;       % samples
nu =2 ;       % inputs
```

```
10 nd =1;        % disturbances
11
12 % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
13
14 % Case B
15 % ny =6 ;       % measurments
16 % ns =5000;       % samples
17 % nu =4 ;       % inputs
18 % nd =2;        % disturbances
19 % % ncomp=10;
20
21
22 % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
23
24 %% Generating parameters randomly
25 Gy=randn(nu,ny)';
26 Gyd=randn(nd,ny)';
27
28 Juu=randn(nu,nu);
29 Jud=randn(nd,nu)';
30 Jdu=randn(nd,nu);
31 Jdd=randn(nd,nd);
32
33 Jsd=[Juu' Jud;Jdu Jdd];  %the second derivative values
34
35 Gp=[Gy Gyd];
36 Jcomb=[Juu' Jud];
37
38 H=Jcomb*pinv(Gp) ;  % pseudoinverse of Gp
39
40 %% PLS test
41 % Random inputs and disturbances:
42 allud=rand(ns,nu+nd);
43
44
45 %The measurement y=Gy*u + Gyd*d = Gp*[u d]'
46
47 %Each row is a variable (yi) and each column is a measurement of this
48 %variable (n_yi)
49
50 Ally=(Gp*allud');
51
52 %Cost function:
53
54 Jwhole=allud*Jsd*allud';
55
56 J=diag(Jwhole);
57
58 % Centering:
59 J=J-repmat(mean(J),ns,1);
60
61 % Scaling:
62 Jmax=max(abs(J));
63 Jscal=J/Jmax;
```

```
64
65 %%%%%
66 % FIX THE SCALING AND CENTERING: THIS IS THE BEST DUMMYCASEFILE!!
67 %%%%%
68
69
70 % Center the data (it isn't scaled)
71 % each element in a row minus the mean of the row
72 Ym=mean(Ally,2);
73 Ym=kron(Ym,ones(1,ns));
74 Y=Ally-Ym  ;
75
76
77 % Scaling the values - the data must be scaled!
78 % This is done by finding the maximum value of a variable among all ...
       the samples,
79 % and dividing the samples on the maximum value.
80 Ymax = max(abs(Ally'))' ;
81 Yscal=[];
82 for i=1:ny
83     Yscal(i,:) = Y(i,:)/Ymax(i,1);
84 end
85
86 scy = diag(Ymax) ;
87
88
89 %%
90 % Finding Yaug to be able to fit a quadratic cost function
91 Yadd=[] ;
92 l=1 ;
93 for i =1:size(Ally,1)
94     for j =i:size(Ally,1)
95         Yadd(l,:) = Yscal(i,:).*Yscal(j,:) ;
96         l=l+1 ;
97     end
98 end
99
100 Yaug=[Yscal; Yadd];
101
102
103
104 F=(-Gy*Juu*Jud)+Gyd;
105
106
107
108 save('dummyA', 'Yaug', 'Jscal', 'Gy','scy','H','Juu','F')
109 %save('gendatadummyB3', 'Yaug', 'Jscal', 'Gy','scy','H')
```

```
%% DUMMY CASE : calcH.m %%
%% 07.06.14 %%
%% Run the PLS-regression for the dummy case %%
% Run from dummymasterfile.m %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


[P,Q,T,U,beta,pctvar] = plsregress(Yaug',Jscal,ncomp);
% X-scores T
% X-loadings W
% Y-scores U
% Y loading C
beta=beta';


J_test = [ones(ns,1) Yaug']*beta';
J_diff = Jscal - J_test


% Figure: percentage variance explained in J:
% figure(1)
% plot(1:ncomp,cumsum(100*pctvar(2,:)),'-bo');
% xlabel('number of PLS components')
% ylabel('Precent Varance Explained in J')

a=0 ;
for i=1:ny
    for j=i:ny
        Jy(i,j)=beta(ny+2+a);
        a=a+1;
    end
end
Jy=Jy;
Jyy=Jy+Jy';


%% Checking if the H are the same
Hmat=[Gy';zeros(nd,ny)]*Jyy ; %/scy ;
Hmat=[Gy';zeros(nd,ny)]*Jyy/scy(1:ny,1:ny) ;
Hmat = Hmat(1:nu,:)
Hmat = Hmat(1:nu,1:nu)\Hmat
H = H(1:nu,1:nu)\H
%Haux =[Gy'; zeros(2,4)]*Jyy/scY(1:4,1:4);

res = H-Hmat


%% Comparing losses for different values of ncomp
% Generated the barplot to show the residual Jm-Jest
ay = Yaug  ;
Jm= Jscal';
Yaug = {}; % en array
```

```
53 J = {} ;
54 res=[]   ;
55 for j= 1:size(ay,1)-1
56     ncomp = j ;
57
58 n = size(ay,2) ;
59 Yaug{1,j} = [ay(:,2:ns)] ;
60 J{1,j} = [Jm(:,2:ns)];
61
62 [P,Q,T,U,beta,pctvar] = plsregress(Yaug{1,j}',J{1,j}',ncomp);
63  beta=beta' ;
64  Test=[ones(ns,1) ay']*beta';
65  diff(1,:) = Jm' - Test ;
66
67 for i = 1:n-1
68     Yaug{i+1,j} = [ay(:,1:i), ay(:,i+2:ns)];
69     J{i+1,j}    = [Jm(:,1:i), Jm(:,i+2:ns)];
70     [P,Q,T,U,beta,pctvar] = plsregress(Yaug{i+1,j}',J{i+1,j}',ncomp);
71     beta=beta' ;
72     Test=[ones(ns,1) ay']*beta';
73     res(i,:) = Jm' - Test ;
74 end
75 resi(j)  = norm(diag([diff ; res])) ;
76
77 end
```

C.0.3 Evaporator process

There are four scripts needed to run the evaporator test case:

- evapmasterfile.m
- evapgeneratedata.m
- evap5y.m
- evap10y.m

The evapgeneratedata.m should be run first. In the master file you choose if you want to run the case with 5 or 10 measurements. In the evap5y.m or the evap10y.m file you set if you want with or without noise and also load the generated data as evapadata.mat.

```
%% EVAPORATOR CASE STUDY : evapmasterfile.m %%
%% 07.06.14 %%
clear all
close all
clc
r=[];
data_loss = [];
for q=1:1
    ncomp=10;
    run evap5y.m
    %run evap10y.m
    %data_loss(q,:) = Loss_data ;
    %r(q,:)=sum(abs(Jscal-Test))/1000
    data_loss(q,:) = Loss_data ;
end
 av_loss_data=sum(data_loss)/q
% bar([1:ncomp],r(1:(ncomp),1))
% %
%  x=[1:(ncomp)]'
% % bar=data_loss(1:(ncomp),1)
% bar = r(1:(ncomp),1)
% data=[x bar]

%dlmwrite('lossallncomp_10yn.dat',data,'delimiter','\t')

%dlmwrite('evap_modelval_y10nn.dat',data,'delimiter','\t')
```

```
%% EVAPORATOR CASE STUDY : evapgeneratedata.m %%
%% 07.06.14 %%

clear all
close all
clc
%% Set how many data sample set you want, ns,
ns=1000;
nsv = ones(1,ns);
r=-1+2*rand(1,ns);      % random nr b/w -1 and 1
%% Generating data - The evaporator process:
%The measurement y=Gy*u + Gyd*d = Gp*[u d]'
% y = [P2   T2  T3  F2  F100  T201  F3  F5  F200   F1] (10 measurements)

%% Initial values
% Inputs:
F2000 = 208.0;        % Cooling water flowrate kg/min
F10    = 10.0;         % Feed flowrate kg/min

% Disturbances:
X10   = 5;             % Feed composition percent %
T10   = 40;            % Feed temperature C
T2000 = 25;            % Coolingwater inlet temperature C

% Initial values
ud0=[F2000;
     F10;
     X10;
     T10;
     T2000];



%% Initial values given in book
P20   = 50.5;         % Operating pressure                  kPa
T20   = 84.6;         % Product temperature                 C
T30   = 80.6;         % Vapor temperature                   C
F20   = 2;            % Product flowrate                    kg/min
F1000 = 9.3;          % Stream flowrate                     kg/min
T2010 = 46.1;         % Cooling water outlet temperature    C
F30   = 50.0;         % Circulating flowrate                kg/min
F50   = 8.0;          % Condencate flowrate                 kg/min
F2000 = 208.0;        % Cooling water flowrate              kg/min
F10    = 10.0;         % Feed flowrate                       kg/min

y0=[P20   ;
    T20   ;
    T30   ;
    F20   ;
    F1000 ;
    T2010 ;
    F30   ;
```

```
    F50   ;
    F2000 ;
    F10   ] ;

%% Generate random data
per=20  ;
perd=5 ;
% Inputs - in deviation variables
%mF200 =  (F2000-(0.1*F2000))+(2*(0.1*F2000)*rand(1,ns))-F2000 ;      ...
    % +- 10%
mF200 =  -0.1*F2000+0.2*F2000*rand(1,ns)   ;  % +- 10%

mF1   =  (F10-(0.1*F10))+(2*(0.1*F10).*rand(1,ns))-F10;              ...
    % +- 10%

% Disturbances - in deviation variables
mX1   = (X10-(0.05*X10))+(2*(0.05*X10)*rand(1,ns))-X10;              ...
    % +- 5%
mT1   = (T10-(0.2*T10))+(2*(0.2*T10)*rand(1,ns))-T10;                ...
    % +- 20%
mT200 = (T2000-(0.2*T2000))+(2*(0.2*T2000)*rand(1,ns))-T2000;        ...
    % +- 20%

% [int-(10% av int)] + 2*(10% av int)*rand(1,ns) - int

%y = [mP2 ; mT2 ; mT3 ; mF2 ; mF100 ; mT201 ; mF100 ; mF3 ; mF5 ; ...
    mF200 ; mF1];

% optimal J
Jnom = 600*F1000 + 0.6*F2000 + 1.009*(F20 + F30) + 0.2*F10 - 4800*F20;


save('evapdata','mF200', 'mF1', 'mX1' ,'mT1', 'mT200','Jnom','ns','y0')
```

```
%% EVAPORATOR CASE STUDY : evap5y.m %%
%% 07.06.14 %%

%% Generate data from case study matrices
% u = [F200 F1]
% d = [X1 T1 T200]
% Loading data file from generatedata.m
load evapdata.mat

%% Gain matrices
% Taken from the article
%y = [P2       T2    T3   F2    F100   T201    F3     F5    F200   F1]
Gy=[-0.0930 11.678;   % P2
     0.0000 0.1410;   % F2
    -0.0940 2.1700;   % T201
    -0.0320 6.5940;   % F3
     1.0000 0.0000];  % F200

```

```
Gyd=[-3.6260    0    1.9720;    % P2
      0.2670    0    0.0000;    % F2
     -0.6740    0    1.0000;    % T201
     -2.2530 -0.0660 0.6730;    % F3
      0         0       0  ];   % F200

Juu=[0.0060 -0.1330;
    -0.1330 16.7370];

Jud=[0.0230   0.0000 -0.0010;
    -158.373 -1.1610 1.4830];

Jdd = eye(3,3);
Jsd = [Juu  Jud;
       Jud' Jdd];

% The full gain matrix
Gp=[Gy Gyd];
Jcomb=[Juu' Jud];

% H-matrix according to the null space method
Hns=Jcomb*pinv(Gp);  % pseudoinverse of Gp

%% PLS test
ny=5;
nu=2;
nd=3;
%ncomp=5;

% Inputs and disturbances
u = [mF1 ;
     mF200];
d = [mX1 ;
     mT1 ;
     mT200];

ud=[mF1 ;
    mF200;
    mX1 ;
    mT1 ;
    mT200] ;

% Measurements:
% y = [P2       F2       T201    F3    F200] (5 measurements)
Wn = [1.285     0          0     0      0;
      0         0.027      0     0      0;
      0         0          1     0      0;
      0         0          0     0.494  0;
      0         0          0     0      4.355];


a=kron(diag(Wn),ones(1,ns));
noise = (a-(0.1*a))+(2*(0.1*a).*rand(ny,ns));
Ally = (Gp*ud) + noise;         % With measurement noise
```

```
%Ally = (Gp*ud) ;                % Without measurement noise


%% Center the data
% each element in a row minus the mean of the row
% Ym=mean(Ally,2)
% Ym=kron(Ym,ones(1,ns))
% Y=Ally-Ym                 % Ally with centered data

% Scaling the values
% The measurements
Ymax = max(abs(Ally'))' ;
Yscal=[];
for i=1:ny
    Yscal(i,:) = Ally(i,:)/Ymax(i,1);
end

scy = diag(Ymax) ;

% Finding Yaug:
Yadd=[] ;
l=1 ;
for i =1:size(Ally,1)
    for j =i:size(Ally,1)
        Yadd(l,:) = Yscal(i,:).*Yscal(j,:) ;
        l=l+1 ;
    end
end

Yaug=[Yscal; Yadd];

% The cost function
Jlin = ud'*Jsd*ud;

Jorg=diag(Jlin);
J=Jorg-repmat(mean(Jorg),ns,1); %centered

Jmax=max(abs(J));
Jscal=J/Jmax;


 %%

[P,Q,T,U,beta,pctvar] = plsregress(Yaug',Jscal,ncomp);
% X-scores T
% X-loadings W
% Y-scores U
% Y loading C
beta=beta' ;


Test=[ones(ns,1) Yaug']*beta';
%
% Making Jyy:
```

```
a=0 ;
for i=1:ny
    for j=i:ny
        Jy(i,j)=beta(ny+2+a);
        a=a+1;
    end
end
Jy=Jy;
Jyy=Jy+Jy';


%% Checking if the H are the same
Hmat=[Gy';zeros(nd,ny)]*Jyy ; %/scy ;
Hmat=[Gy';zeros(nd,ny)]*Jyy/scy;
Hmat = Hmat(1:nu,:)              ;   % This is the "first H-matrix we ...
    found before any scaling
Hmatsc = Hmat(1:nu,1:nu)\Hmat     ;  % Scaled H matrix
H = Hns(1:nu,1:nu)\Hns   ;               % H from Jcomp*Gp
%Haux =[Gy'; zeros(2,4)]*Jyy/scY(1:4,1:4);

res = H-Hmatsc ;

% figure(1)
% plot(1:ncomp,cumsum(100*pctvar(2,:)),'-bo');
% xlabel('number of PLS components')
% ylabel('Precent Varance Explained in J')


%% Exact local method
Wd = [0.25 0 0;
        0 8 0;
        0 0 5];

% y = [P2       F2       T201    F3    F200] (5 measurements)
Wn = [1.285     0          0     0      0;
      0         0.027      0     0      0;
      0         0          1     0      0;
      0         0          0     0.494  0;
      0         0          0     0      4.355];

% % Sensitivity matrix
F= Gyd-(Gy*Juu^(-1)*Jud) ;
Y = [F*Wd Wn];
Hexl=Gy'*(Y*Y')^(-1);


  Y = [F*Wd Wn];              % With measurement noise
 %Y = [F*Wd Wn*0];             % Without measurement noise

Mexl = Juu^(0.5) *(Hexl*Gy)^(-1) * (Hexl*Y);
Loss_exl = 0.5*norm(Mexl,'fro').^2

Mdata = Juu^(0.5) *(Hmat*Gy)^(-1) * (Hmat*Y);
Loss_data = 0.5*norm(Mdata,'fro').^2
```

```
180
181 Mns = Juu^(0.5) *(Hns*Gy)^(-1) * (Hns*Y);
182 Loss_ns = 0.5*norm(Mns,'fro').^2
183
184
185 Hns1 = null(F')';
186 Hns1 = Hns(1:2,:);
187 Mns1 = Juu^(0.5) *(Hns1*Gy)^(-1) * (Hns1*Y);
188 Loss_ns1 = 0.5*norm(Mns1,'fro').^2
189
190
191 Hmat
192 Hexl
193 Hns
```

```
1  %% EVAPORATOR CASE STUDY : evap10y.m %%
2  %% 07.06.14 %%
3  %Test case to calculate H, and comparing it to H found from PLS method
4  % clear all
5  % close all
6  % clc
7
8  %% Generate data Ffom case study matrices
9  % u = [F200 F1]
10 % d = [X1 T1 T200]
11 % Loading data file from generatedata.m
12 load evapdata.mat
13
14 %% Gain matrices
15 % Taken from the article
16 Gy=[-0.0930 11.678;   % P2
17     -0.0520 6.5590;   % T2
18     -0.0470 5.9210;   % T3
19      0.0000 0.1410;   % F2
20     -0.0010 1.1150;   % F100
21     -0.0940 2.1700;   % T201
22     -0.0320 6.5940;   % F3
23      0.0000 0.8590;   % F5
24      1.0000 0.0000;   % F200
25      0.0000 1.0000];  %F1
26
27 Gyd=[-3.6260    0    1.9720;    % P2
28      -2.0360    0    1.1080;    % T2
29      -1.8380    0    1.0000;    % T3
30      0.2670     0    0.0000;    % F2
31      -0.3170 -0.0180 0.0200;    % F100
32      -0.6740    0    1.0000;    % T201
33      -2.2530 -0.0660 0.6730;    % F3
34      -0.2670    0    0.0000;    % F5
35       0         0       0  ;    % F200
36       0         0       0  ];   % F1
37
```

```
38 Juu=[0.0060 -0.1330;
39     -0.1330 16.7370];
40
41 Jud=[0.0230   0.0000 -0.0010;
42     -158.373 -1.1610 1.4830];
43
44 Jdd = eye(3,3);
45 Jsd = [Juu  Jud;
46        Jud' Jdd];
47
48 % The full gain matrix
49 Gp=[Gy Gyd];
50 Jcomb=[Juu' Jud];
51
52 % H-matrix according to the null space method
53 Hns=Jcomb*pinv(Gp);  % pseudoinverse of Gp
54
55 %% PLS test
56 ny=10;
57 nu=2;
58 nd=3;
59 %ncomp=4;
60
61 % Inputs and disturbances
62 u = [mF1 ;
63      mF200];
64 d = [mX1 ;
65      mT1 ;
66      mT200];
67
68 ud=[mF1 ;
69     mF200;
70     mX1 ;
71     mT1 ;
72     mT200] ;
73
74 % Measurements:
75 % y = [P2       T2    T3   F2    F100   T201    F3     F5    F200   ...
       F1] (10 measurements)
76 Wn = [1.285     0     0     0       0      0     0      0      0      0;
77       0         1     0     0       0      0     0      0      0      0;
78       0         0     1     0       0      0     0      0      0      0;
79       0         0     0     0.027   0      0     0      0      0      0;
80       0         0     0     0       0.189  0     0      0      0      0;
81       0         0     0     0       0      1     0      0      0      0;
82       0         0     0     0       0      0     0.494  0      0      0;
83       0         0     0     0       0      0     0      0.163  0      0;
84       0         0     0     0       0      0     0      0      4.355  0;
85       0         0     0     0       0      0     0      0      0      ...
             0.189];
86 a=kron(diag(Wn),ones(1,ns));
87 noise = (a-(0.1*a))+(2*(0.1*a).*rand(ny,ns));
88 Ally = (Gp*ud) + noise;          % With measurement noise
89 %Ally = (Gp*ud) ;                % Without measurement noise
```

```
90
91
92  %% Center the data
93  % each element in a row minus the mean of the row
94  % Ym=mean(Ally,2)
95  % Ym=kron(Ym,ones(1,ns))
96  % Y=Ally-Ym                % Ally with centered data
97
98  % Scaling the values
99  % The measurements
100 Ymax = max(abs(Ally'))' ;
101 Yscal=[];
102 for i=1:ny
103     Yscal(i,:) = Ally(i,:)/Ymax(i,1);
104 end
105
106 scy = diag(Ymax) ;
107
108 % Finding Yaug:
109 Yadd=[] ;
110 l=1 ;
111 for i =1:size(Ally,1)
112     for j =i:size(Ally,1)
113         Yadd(l,:) = Yscal(i,:).*Yscal(j,:) ;
114         l=l+1 ;
115     end
116 end
117
118 Yaug=[Yscal; Yadd];
119
120 % The cost function
121 Jlin = ud'*Jsd*ud;
122
123 Jorg=diag(Jlin);
124 J=Jorg-repmat(mean(Jorg),ns,1); %centered
125
126 Jmax=max(abs(J));
127 Jscal=J/Jmax;
128
129
130  %%
131
132 [P,Q,T,U,beta,pctvar] = plsregress(Yaug',Jscal,ncomp);
133 % X-scores T
134 % X-loadings W
135 % Y-scores U
136 % Y loading C
137 beta=beta' ;
138
139
140 Test=[ones(ns,1) Yaug']*beta';
141 %
142 % Making Jyy:
143 a=0 ;
```

```
144 for i=1:ny
145     for j=i:ny
146         Jy(i,j)=beta(ny+2+a);
147         a=a+1;
148     end
149 end
150 Jy=Jy;
151 Jyy=Jy+Jy';
152
153
154 %% Checking if the H are the same
155 Hmat=[Gy';zeros(nd,ny)]*Jyy ; %/scy ;
156 Hmat=[Gy';zeros(nd,ny)]*Jyy/scy;
157 Hmat = Hmat(1:nu,:)              ;   % This is the "first H-matrix we ...
        found before any scaling
158 Hmatsc = Hmat(1:nu,1:nu)\Hmat     ;  % Scaled H matrix
159 H = Hns(1:nu,1:nu)\Hns   ;               % H from Jcomp*Gp
160 %Haux =[Gy'; zeros(2,4)]*Jyy/scY(1:4,1:4);
161
162 res = H-Hmatsc ;
163
164 % figure(1)
165 % plot(1:ncomp,cumsum(100*pctvar(2,:)),'-bo');
166 % xlabel('number of PLS components')
167 % ylabel('Precent Varance Explained in J')
168
169
170 %% Exact local method
171 Wd = [0.25 0 0;
172         0 8 0;
173         0 0 5];
174
175 Wn = [1.285     0     0     0      0      0     0      0      0      0;
176       0         1     0     0      0      0     0      0      0      0;
177       0         0     1     0      0      0     0      0      0      0;
178       0         0     0     0.027  0      0     0      0      0      0;
179       0         0     0     0      0.189  0     0      0      0      0;
180       0         0     0     0      0      1     0      0      0      0;
181       0         0     0     0      0      0     0.494  0      0      0;
182       0         0     0     0      0      0     0      0.163  0      0;
183       0         0     0     0      0      0     0      0      4.355  0;
184       0         0     0     0      0      0     0      0      0      ...
              0.189];
185
186 % % Sensitivity matrix
187 F= Gyd-(Gy*Juu^(-1)*Jud) ;
188 Y = [F*Wd Wn];
189 Hexl=Gy'*(Y*Y')^(-1);
190
191
192   Y = [F*Wd Wn];               % With measurement noise
193  %Y = [F*Wd Wn*0];             % Without measurement noise
194
195 Mexl = Juu^(0.5) *(Hexl*Gy)^(-1) * (Hexl*Y);
```

```
196 Loss_exl = 0.5*norm(Mexl,'fro').^2
197
198 Mdata = Juu^(0.5) *(Hmat*Gy)^(-1) * (Hmat*Y);
199 Loss_data = 0.5*norm(Mdata,'fro').^2
200
201 Mns = Juu^(0.5) *(Hns*Gy)^(-1) * (Hns*Y);
202 Loss_ns = 0.5*norm(Mns,'fro').^2
203
204
205 Hns1 = null(F')';
206 Hns1 = Hns(1:2,:);
207 Mns1 = Juu^(0.5) *(Hns1*Gy)^(-1) * (Hns1*Y);
208 Loss_ns1 = 0.5*norm(Mns1,'fro').^2
209
210 Hmat
211 Hexl
212 Hns
```

C.0.4 The CSTR-distillation process

This is the test case with the most MATLAB-files connected to the case. The MATLAB-files describing the process model were already developed, and given to the candidate by Vladimiros L. Minasidis. These files are:

- colamodSS.m
- CSTRSSmodel.m
- fun.m
- nlcon.m
- optScript.m

These files are necessary as a basis to run the CSTR test case. These are slightly manipulated compared to the original files when they are used to test the H-matrix as a control variable.

Second, there are some scrips that needs only to be run once. They create *.mat* files with data that other scrips need. These files are:

gainmatrix.m	→	gainmat.mat :	the measurement gain, G_y
desutbancegainmatrix.m	→	destgainmat.mat :	the distrubance gain, G_y^d
sensitivitymatrix.mat	→	sensFmatrix.mat :	the sensitivity matrix, F
optScript.m	→	xoptcstrdest.mat.mat :	the nominal optimal values

The generated *.mat* files and the files describing the model must all be in the same folder as the files used to run the PLS-regression and simulate the control of the process.

The files used to generate data, model the cost function and H, and to control and re-optimize the process are: cstrmasterfile.m, simulationtest2wdist.m, simulationtest2nodist.m, simulationtest2for5ywd.m, simulationtest2for5ynd.m, generatedata.m and
alldata_ dataH_ CSTRdest2.m. Running the master file activates and runs all the other files in turn.

File	description
cstrmasterfile.m	Set: • numer of data set, ns • numer of components, ncomp • case A, B C or D • Data with or without noise • disturbance size • Which H-matrix to use for control, data, exl or ns Runs: • simulationtest2wdist.m • simulationtest2nodist.m • simulationtest2for5ywd.m • simulationtest2for5ynd.m
simulationtest2xxxx.m	Simulates control and optimization of the process • loads hmatrix.mat • loads optimalval.mat Gives the result • model accuracy: Residual $J_M - J_{est}$ • loss calculation: $J_{opt} - J_{ctrl}$ Runs • generatedata.m
generatedata.m	Generates process data Runs: • alldata_ dataH_ CSTRdest2.m • loads x_ optcstrdest.mat • loads sensFmat.mat Saves: • optimalval.mat • cstrdest.mat
alldata_ dataH_ CSTRdest2.m	• Uses PLS-regression to model the cost function and find H-matrix with the data-based method. • Finds H using exact local method and null space method as well. • loads cstrdest.mat • loads gainmat.mat • loads destgainmat.mat Saves: • hmatrix.mat

```
%% CSTR-distillation process cstrmasterfile.m %%
% 07.06.14

clear all
clc
result =ones(1,2);
converge = [];

%Set number of samples  - ns
ns=50;

% Set number of components  - ncomp


% changes case in alldat_dataH-CSTRdest2.m  - dist AND #y
casenr = 1;
% 1 = y10  A
% 2 = yd10 B
% 3 = y5   C
% 4 = yd5  D
  if casenr ==1
        ncomp = 10;
        elseif casenr == 2
        ncomp = 10;
        elseif casenr == 3
        ncomp = 10;
        elseif casenr == 4
        ncomp = 10;
  end

% overwrite NCOMP
% ncomp =


% changes with or without measurement noise in ...
    alldat_dataH-CSTRdest2.m - n
msmtnoise = 1;
if  msmtnoise == 1
    case_n = 1  ; % No noise
elseif msmtnoise == 2
    case_n = 2  ; % With noise
end

% changes which H to be used in simulation simulationtest2xxxx.m
Method = 2;

% 1 = Hmat
% 2 = Hexl
% 3 = Hns

% Set the disturbance in the feed: ex: 0.1=10%, 0.01=1%
Fdist = 0.01;
```

```
52
53 for k=1:5
54     if Method ==2
55         k=1;
56     elseif Method ==3
57         k=1;
58     end
59 alt = casenr ;    % Changes simulation file
60     if alt ==1
61         run simulationtest2nodist.m
62         elseif alt == 2
63         run simulationtest2wdist.m
64         elseif alt == 3
65         run simulationtest2for5ynd.m
66         elseif alt == 4
67         run simulationtest2for5ywd.m
68     end
69
70 flag(:,k) = exitflag(:,1)  ;
71 result(k) = loss(:,n);              % loss form silumating
72 av_resi(k) = av_resi_modelval(:,n);  % residual b/w Jm and Jest
73 Htype;
74 modelval(k) = av_resi_modelval(:,n);
75 loss;
76 optcost = fvalopt*60;
77 fvalsoc;
78 end
79 result
80 flag
81 average=sum(result)/k
82 modval=sum(modelval)/k
83 %averageresi = sum(av_resi)/k
84 Htype
85
86
87 %run saveresults.m
```

```
1  %% CSTR-distillation process generatedata_optScriptx2.m %%
2  % 07.06.14
3  %%
4  % Script for optimization of reactor, separator and recycle process.
5  % The numerical description of the process is taken from Larsson et al ...
       (2003)
6  %
7  % clc
8  % clear all
9  global p;
10 global u;
11 %ns=50;
12 ny = 10 ;
13 T=@(x)80-x*20;
14 load x_opt_cstrdest.mat
```

```
15 load sensFmat.mat
16
17 %Boiling temperature of the light component
18 Tb_L=353;
19 %Boiling temperature of the heavy component
20 Tb_H=373;
21
22
23 % Column parameters
24 p.qF    = 1;
25 p.NT    = 22;
26 p.NF    = 13;
27 p.alpha = 2;
28 p.Vmax  = 1500/60;
29 % CSTR parameters
30 p.F0 = 460/60*1.0;
31 p.zF0 = 0.9;
32 p.k1 = 0.341/60;
33 p.F_int = 958/60 ;
34 % Flags
35 p.OPTI=0;
36 p.case_I=1;
37
38 % Initial values for disturbance and inputs:
39 F_int   = p.F0;
40 x_d_int = 0.8151 ;
41 L_int   = 12.97 ;
42
43 u.u1 = x_d_int ;
44 u.u2 = L_int ;
45
46 if p.case_I==1
47     % Constraints
48     lb=zeros(p.NT+8,1);
49     ub=[ones(p.NT,1); ones(8,1)*Inf];
50
51     % xB <= 0.0105
52     ub(1)=0.0105;
53     lb(1)=0.0105;
54     % Mr <= 2800;
55     ub(p.NT+7)=2800;
56     lb(p.NT+7)=2800;
57     % F0 = fixed;
58     lb(p.NT+8)=p.F0;
59     ub(p.NT+8)=p.F0;
60 else
61     % Constraints
62     lb=zeros(p.NT+8,1);
63     ub=[ones(p.NT,1); ones(8,1)*Inf];
64
65     % xB <= 0.0105
66     ub(1)=0.0105;
67     % Mr <= 2800;
68     ub(p.NT+7)=2800;
```

```
69      % V <= Vmax;
70      ub(p.NT+2)=p.Vmax;
71  end
72
73  % Initial value
74  x0= x_opt;
75
76  % Controlling active constriants:
77      % xB <= 0.0105
78      ub(1)=0.0105;
79      lb(1)=0.0105;
80      % Mr <= 2800;
81      ub(p.NT+7)=2800;
82      lb(p.NT+7)=2800;
83
84  for i=1:ns
85  % Change in the Feed - disturbance
86  F0dist = 0.1;
87  d_F0 = (F_int-(F0dist*F_int))+(2*(F0dist*F_int).*rand(1,ns))-F_int;
88  %d_F0 = d_F0(i) ;
89
90  % Change in input L
91  d_u2 = (L_int-(0.1*L_int))+(2*(0.1*L_int).*rand(1,ns))-L_int;
92  %d_u2 = d_u2(i) ;
93
94  p.F0 = F_int + d_F0(:,i);
95  u.u2 = L_int + d_u2(:,i);
96
97  % fmincon options
98  options = optimset('TolFun',10e-6,'TolCon',10e-6,'MaxFunEvals',1e4,...
99      'Display','none','Algorithm','active-set','Diagnostics','off'...
100     );
101
102 % Using boundry conditions to keep a veriable constant
103 % Feed:
104     lb(p.NT+8)   = p.F0;
105     ub(p.NT+8)   = p.F0;
106
107 % L
108    lb(p.NT+1)   = u.u2;
109    ub(p.NT+1)   = u.u2;
110
111
112 % fmincon
113 [x(:,i),fval(:,i),exitflag(:,i)]=fmincon(@fun,x0,[],[],[],[],lb,ub,@nlcon,options);
114 cost(:,i)= x(p.NT+2,i); %*60 ;
115
116 dF0(i,:) = p.F0 ;
117
118
119 Te= Tb_L*x(1:22,:)+(1-x(1:22,:))*Tb_H;
120 Topt= Tb_L*x_opt(1:22,:)+(1-x_opt(1:22,:))*Tb_H;
121
122 end
```

```
123
124 x_optT  = [Topt ; x_opt(23:30)];
125 out      = [Te(1:22,1:i) ; x(23:30,1:i)];
126
127
128 % In deviation variabels
129 dx = out(:,1:size(x,2))-kron(x_opt,ones(1,size(x,2)));
130 d_cost=cost-(x_opt(p.NT+2)); %*60);
131
132 exitflag ;
133 converge = 1./exitflag;
134
135 %% Picking the desired measurements for dx:
136 xval = [dx(8,:) ; dx(13,:) ; dx(18,:) ; dx(22,:)];
137 y    = [dx(8,:) ; dx(13,:) ; dx(18,:) ; dx(22,:) ; dx(23:28,:) ] ;
138 y5   = [dx(8,:) ; dx(22,:) ; dx(23,:) ; dx(26,:) ; dx(30,:)] ;
139 y5nd   = [dx(8,:) ; dx(22,:) ; dx(23,:) ; dx(26,:) ; dx(27,:)] ;
140 y9   = [dx(8,:) ; dx(13,:) ; dx(22,:) ; dx(23:28,:) ; dx(30,:) ] ;
141
142 y_opt = [x_opt(8,:) ; x_opt(13,:) ; x_opt(18,:) ; x_opt(22,:) ; ...
        x_opt(23:28,:) ] ;
143 y_optT = [x_optT(8,:) ; x_optT(13,:) ; x_optT(18,:) ; x_optT(22,:) ; ...
        x_optT(23:28,:) ] ;
144 y_optT9 = [x_optT(8,:) ; x_optT(13,:) ; x_optT(22,:) ; ...
        x_optT(23:28,:);  x_optT(30,:) ] ;
145 y_optT5 = [x_optT(8,:) ; x_optT(22,:) ;x_optT(23,:) ; x_optT(26,:) ; ...
        x_optT(30,:)] ;
146 y_optT5nd = [x_optT(8,:) ; x_optT(22,:) ;x_optT(23,:) ; x_optT(26,:) ; ...
        x_optT(27,:)] ;
147 %% Adding noise to the measurements
148 % Measurements:
149 %y = [8      13       18    22       Lt    Vb      D      B      F ...
            zf    ]
150 Wn = [1       0       0      0       0      0      0      0      0 ...
           0   ;
151        0       1       0      0       0      0      0      0      0 ...
                   0   ;
152        0       0       1      0       0      0      0      0      0 ...
                   0   ;
153        0       0       0      1       0      0      0      0      0 ...
                   0   ;
154        0       0       0      0       0.13   0      0      0      0 ...
                   0   ;
155        0       0       0      0       0      0.213  0      0      0 ...
                   0   ;
156        0       0       0      0       0      0      0.083  0      0 ...
                   0   ;
157        0       0       0      0       0      0      0      0.077  0 ...
                   0   ;
158        0       0       0      0       0      0      0      0      0.16 ...
               0   ;
159        0       0       0      0       0      0      0      0      0 ...
                   0.02];
160
```

```
%y9 = [8      13       22    Lt     Vb      D      B      F      zf    ...
    F0      ]
Wn9 = [1      0        0     0       0      0      0      0       0 ...
         0    ;
       0      1        0     0       0      0      0      0       0 ...
                0     ;
       0      0        1     0       0      0      0      0       0 ...
                0     ;
       0      0        0     0.13    0      0      0      0       0 ...
                0     ;
       0      0        0     0       0.213  0      0      0       0 ...
                0     ;
       0      0        0     0       0      0.083  0      0       0 ...
                0     ;
       0      0        0     0       0      0      0.077  0       0 ...
                0     ;
       0      0        0     0       0      0      0      0.16    0 ...
                0     ;
       0      0        0     0       0      0      0      0       0.02 ...
              0    ;
       0      0        0     0       0      0      0      0       0 ...
                0.38 ];

%y = [8      22   Lt     B      F0 ]
Wn5 = [1     0    0       0      0    ;
       0     1    0       0      0    ;
       0     0    0.13    0      0    ;
       0     0    0       0.077  0    ;
       0     0    0       0      0.38 ];

%y =    [8      22   Lt     B      F ]
Wn5nd = [1     0    0       0      0    ;
         0     1    0       0      0    ;
         0     0    0.13    0      0    ;
         0     0    0       0.077  0    ;
         0     0    0       0      0.16 ];

a=kron(diag(Wn),ones(1,ns));
n = (a-(0.1*a))+(2*(0.1*a).*rand(ny,ns));
yn = y+n;

a9=kron(diag(Wn9),ones(1,ns));
n9 = (a9-(0.1*a9))+(2*(0.1*a9).*rand(ny,ns));
yn9 = y9 + n9 ;

a5=kron(diag(Wn5),ones(1,ns));
n5 = (a5-(0.1*a5))+(2*(0.1*a5).*rand(5,ns));
yn5 = y5 + n5 ;

a5nd=kron(diag(Wn5nd),ones(1,ns));
n5nd = (a5nd-(0.1*a5nd))+(2*(0.1*a5nd).*rand(5,ns));
yn5nd = y5nd + n5nd ;

Wd = [0.1] ;
```

```
204
205 Yexl=[F*Wd Wn] ;
206 Yexl9=[F9*Wd Wn9] ;
207 Yexl5=[F5*Wd Wn5] ;
208 Yexl5nd=[F5nd*Wd Wn5nd] ;
209
210
211 save('optimalval','y_optT', 'y_optT9', 'y_optT5', 'y_optT5nd')
212
213 save('cstrdest','y','d_cost','ns','y_opt','Wn','yn','Wd','Yexl','Yexl9','Yexl5','Yexl5nd
214 'F5','F5nd','dF0','dx','y9','yn9','y5','yn5','y5nd', 'yn5nd')
215
216 % Results
217 if p.case_I==1
218     casename='case I: min operation cost(energy)\n';
219 else
220     casename='case II: max production rate\n';
221 end
222 %Print the results
223 % results_fmincon=sprintf(strcat(...
224 %     casename,...
225 %     'feed rate,             F0[kmol/h]      = %1$0.1f\n',...
226 %     'reactor effluent,      F[kmol/h]       = %2$0.1f\n',...
227 %     'vapor boilup,          V[kmol/h]       = %3$0.1f\n',...
228 %     'reflux,                L[kmol/h]       = %4$0.1f\n',...
229 %     'recycle (distilate),   D[kmol/h]       = %5$0.1f\n',...
230 %     'recycle composition,   xD[molA/mol]    = %6$0.4f\n',...
231 %     'bottom composition,    xB[molA/mol]    = %7$0.4f\n',...
232 %     'reactor composition,   zF[molA/mol]    = %8$0.4f\n',...
233 %     'reactor holdup,        Mr[kmol/h]      = %9$0.0f\n',...
234 %     'Column temperatures,   T_i[C]          = T_3   T_8   T_13  ...
    T_18\n ',...
235 %     '                                         %10$0.1f  %11$0.1f  ...
    %12$0.1f  %13$0.1f\n'...
236 %     ),x(p.NT+8)*60,x(p.NT+5)*60,x(p.NT+2)*60,x(p.NT+1)*60, ...
    x(p.NT+3)*60,...
237 %     x(p.NT), x(1), ...
    x(p.NT+6),x(p.NT+7),T(x(3)),T(x(8)),T(x(13)),T(x(18)))
238
239 run alldata_dataH_CSTRdest2.m
```

```
1  %% CSTR-distillation process alldata_dataH_CSTRdest2.m %%
2  % 07.06.14
3
4
5  %%  Step 1
6  % Generate data from case study matrices
7  % The data along with other parameters are calculated in other scrips
8  % The relevant parameters and data is loaded here
9  load cstrdest.mat       % Gives: ...
       'y','d_cost','ns','y_opt','Wn','yn','Wd','Y','F','dF0'
10 load gainmat.mat        % Gives Gy
```

```
load destgainmat.mat    % Gives Gyd


%% Step 2
% Here the number of measurements, inputs and disturbances are given. And
% The number of components to be used in the PLS—regnression. The ... number of
% Components are found in a different script.

ny = 10;
nd = 1; % F0
nu = 1; % L

%% Choosing the case
% nr. of y, incl. the disutbance or not, with or without noise

if casenr == 1
    disp('10 measurements without dist ')
% Not including the disturbance in the measurement: %
%y = [8     13    18    22      Lt     Vb     D      B     F      zf ] ... (10 measurements)
        ny=10 ;
        Gy=Gy;
        Yexl = Yexl ;
        F = F;
        if case_n == 1;
        Ally = y(:,1:size(y,2)) ; % No noise
        disp('no measurement noise')
        elseif case_n == 2
        Ally = yn(:,1:size(yn,2)); % With noise
          disp('with measurement noise')
        end

elseif casenr == 2
      disp('10 measurements with dist ')
% Including the disturbance in the measurements: %
%y = [8     13      22      Lt     Vb     D      B     F      zf     ... F0]  (10 measurements)
            ny=10 ;
            Gy=Gy9;
            Yexl = Yexl9 ;
            F = F9;
        if case_n == 1;
            Ally = y9(:,1:size(y9,2)) ; % No noise
              disp('no measurement noise')
        elseif case_n == 2
            Ally = yn9(:,1:size(yn9,2)); % With noise
               disp('with measurement noise')
        end


elseif casenr == 3
     disp('5 measurements without dist ')
% Using only 5 measurements, excl. disturbance F0 %
```

```
            ny=5 ;
            Gy=Gy5nd;
            Yexl = Yexl5nd ;
            F = F5nd;
        if case_n == 1;
            Ally = y5nd(:,1:size(y5nd,2)) ; % No noise
              disp('no measurement noise')
        elseif case_n == 2
            Ally = yn5nd(:,1:size(yn5nd,2)); % With noise
               disp('with measurement noise')
        end

elseif casenr == 4
     disp('5 measurements with dist ')
% Using only 5 measurements, incl. disturbance F0 %
            ny=5 ;
            Gy=Gy5;
            Yexl = Yexl5 ;
            F = F5;
        if case_n == 1;
            Ally = y5(:,1:size(y5,2)) ; % No noise
              disp('no measurement noise')
        elseif case_n == 2
            Ally = yn5(:,1:size(yn5,2)); % With noise
               disp('with measurement noise')
        end

end
%% Step 3
% Center the data
% If the data is fabricated using random disturbance and change in the
% inputs to "calculate" data, the data is already centered. And we can ...
    skip
% this step.
% Each element in a row minus the mean of the row
% Ym=mean(Ally,2)
% Ym=kron(Ym,ones(1,ns))
% Y=Ally-Ym                % Ally with centered data

% Scaling the values - the data must be scaled!
% This is done by finding the maximum value of a variable among all ...
    the samples,
% and dividing the samples on the maximum value.
Ymax = max(abs(Ally'))' ;
Yscal=[];
for i=1:ny
    Yscal(i,:) = Ally(i,:)/Ymax(i,1);
end

scy = diag(Ymax) ;

%% Step 4
% Finding Yaug to be able to fit a quadratic cost function
Yadd=[] ;
```

```
l=1 ;
for i =1:size(Ally,1)
    for j =i:size(Ally,1)
        Yadd(l,:) = Yscal(i,:).*Yscal(j,:) ;
        l=l+1 ;
    end
end

Yaug=[Yscal; Yadd];

%% Step 5
% Centering and scaling the cost function
Jorg = d_cost(1,1:size(d_cost,2))';
J=Jorg-repmat(mean(Jorg),ns,1); %centered

Jmax=max(abs(J));
Jscal=J/Jmax;


%% Step 6
% Running the PLS-regression

[P,Q,T,U,beta,pctvar] = plsregress(Yaug',Jscal,ncomp);
% X-scores T
% X-loadings W
% Y-scores U
% Y loading C
beta=beta' ;

% The modeled cost funtion
Test=[ones(ns,1) Yaug']*beta';

% Making Jyy:
a=0 ;
for i=1:ny
    for j=i:ny
        Jy(i,j)=beta(ny+2+a);
        a=a+1;
    end
end
Jy=Jy;
Jyy=Jy+Jy';


%% Step 7
% Calculating and scaling back the H matrix
Hmat=[Gy';zeros(nd,ny)]*Jyy ;
Hmat=[Gy';zeros(nd,ny)]*Jyy/scy;
Hmat = Hmat(1:nu,:)               ;   % This is the "first H-matrix we ...
    found before any scaling
Hmatsc = Hmat(1:nu,1:nu)\Hmat     ;  % Scaled H matrix
%H = Hns(1:nu,1:nu)\Hns   ;                % H from Jcomp*Gp
%Haux =[Gy'; zeros(2,4)]*Jyy/scY(1:4,1:4);

```

```
%res = H-Hmatsc ;

% figure(1)
% plot(1:ncomp,cumsum(100*pctvar(2,:)),'-bo');
% xlabel('number of PLS components')
% ylabel('Precent Varance Explained in J')


%% Exact local method and null space

Hexl=Gy'*(Yexl*Yexl')^(-1);
%disp('tror ikke Y*Y er interverbar')
Hns = null(F')';

J_test = [ones(ns,1) Yaug']*beta';
J_diff = Jscal - J_test;

%figure(1)
%bar([1:ns], J_diff)


save('hmatrix','Hmat','Hexl','Hns','beta', 'J_diff')
```

```
%% CSTR-distillation process simulationtest2wdist.m %%
% 07.06.14
%%
% Script for optimization of reactor, separator and recycle process.
% The numerical description of the process is taken from Larsson et al ...
    (2003)
run generatedata_optScriptx2.m
% clc
% clear all
global p;

T=@(x)80-x*20;
%Boiling temperature of the light component
Tb =353;
%Boiling temperature of the heavy component
Th=373;

% Column parameters
p.qF    = 1;
p.NT    = 22;
p.NF    = 13;
p.alpha = 2;
p.Vmax  = 1500/60;
% CSTR parameters
p.F0 = 460/60*1.0;
p.zF0 = 0.9;
p.k1 = 0.341/60;
% Flags
p.OPTI=0;
```

```
29 p.case_I=1;
30
31 if p.case_I==1
32     % Constraints
33     lb=zeros(p.NT+8,1);
34     ub=[ones(p.NT,1); ones(8,1)*Inf];
35
36     % xB <= 0.0105
37     ub(1)=0.0105;
38     % Mr <= 2800;
39     ub(p.NT+7)=2800;
40     % F0 = fixed;
41     lb(p.NT+8)=p.F0;
42     ub(p.NT+8)=p.F0;
43 else
44     % Constraints
45     lb=zeros(p.NT+8,1);
46     ub=[ones(p.NT,1); ones(8,1)*Inf];
47
48     % xB <= 0.0105
49     ub(1)=0.0105;
50     % Mr <= 2800;
51     ub(p.NT+7)=2800;
52     % V <= Vmax;
53     ub(p.NT+2)=p.Vmax;
54 end
55
56 % Initial value
57 x0= [ones(1,p.NT)*0.5 10 15 5 5 1.1 0.5 1000 400/60]';
58
59 % fmincon options
60 options = optimset('TolFun',10e-6,'TolCon',10e-6,'MaxFunEvals',1e4,...
61 'Display','none','Algorithm','sqp','Diagnostics','off'...
62 );
63 % fmincon
64 [x,fval,exitflag]=fmincon(@fun1,x0,[],[],[],[],lb,ub,@nlcon1,options);
65
66 % Results
67 if p.case_I==1
68     casename='case I: min operation cost(energy)\n';
69 else
70     casename='case II: max production rate\n';
71 end
72 % %Print the results
73 % results_fmincon=sprintf(strcat(...
74 %     casename,...
75 %     'feed rate,             F0[kmol/h]      = %1$0.1f\n',...
76 %     'reactor effluent,      F[kmol/h]       = %2$0.1f\n',...
77 %     'vapor boilup,          V[kmol/h]       = %3$0.1f\n',...
78 %     'reflux,                L[kmol/h]       = %4$0.1f\n',...
79 %     'recycle (distilate),   D[kmol/h]       = %5$0.1f\n',...
80 %     'recycle composition,   xD[molA/mol]    = %6$0.4f\n',...
81 %     'bottom composition,    xB[molA/mol]    = %7$0.4f\n',...
82 %     'reactor composition,   zF[molA/mol]    = %8$0.4f\n',...
```

```
%     'reactor holdup,        Mr[kmol/h]      = %9$0.0f\n',...
%     'Column temperatures,   T_i[C]          = T_3   T_8   T_13  ...
    T_18\n ',...
%     '                                         %10$0.1f  %11$0.1f  ...
    %12$0.1f  %13$0.1f\n'...
%     ),x(p.NT+8)*60,x(p.NT+5)*60,x(p.NT+2)*60,x(p.NT+1)*60, ...
    x(p.NT+3)*60,...
%     x(p.NT), x(1), ...
    x(p.NT+6),x(p.NT+7),T(x(3)),T(x(8)),T(x(13)),T(x(18)))

%% Controlling the process

y = [x(8) ; x(13) ; x(18) ; x(22) ; x(23:28) ] ;
F0_int = x(p.NT+8);

%Controlling only L -Brute force
% cs = x(p.NT+1) ;
% Aeq(1,23) = 1 ;
% Aeq(1,30) = 0 ;
% beq = cs ;

% Controlling x_D and L combinded soc
% cs = x(p.NT) - 0.0077*x(p.NT+1);
% Aeq(1,22) = 1 ;
% Aeq(1,23) = 0.0077 ;
% Aeq(1,30) = 0 ;
% beq = cs ;

%%
% Controlling with a full H-matrix :
 load hmatrix.mat     % loads Hmat, Hns and Hexl
 load optimalval.mat  % loads y_opt with temperatures



if Method == 1
    Htype = Hmat;
elseif Method == 2
    Htype = Hexl ;
else
    Htype = [Hns(1,:)+Hns(2,:)+Hns(3,:)] ;
end

 %% Data method - With disturbance
%Hmat is calculated with temperatur-data multiplied with optimal ...
    values of
%y with optimal temperatures
csT = Htype*y_optT9;

H1 = [Htype(1,1:3)] ; % Is to be transformed into composition
H2 = [Htype(1,4:10)]; % Is to be kept as it is
e=[1;1;1] ;
H1x = H1*(Tb-Th) ;   % Transforming the H-matrix
% Set point with composition
```

```
132 csx = csT-(H1*(e*Th)) ; % Transforming the set-point into composition
133
134 % The new H-matrix for compositon
135 Htype = [H1x H2];
136
137 % Using the self optimizing variable:
138 % Optimal setpoint
139
140 % Placing the H-elements at teh correct location
141 Aeq(1,8)  = Htype(1,1) ;
142 Aeq(1,13) = Htype(1,2) ;
143 Aeq(1,22) = Htype(1,3) ;
144
145 Aeq(1,23) = Htype(1,4) ;
146 Aeq(1,24) = Htype(1,5) ;
147 Aeq(1,25) = Htype(1,6) ;
148 Aeq(1,26) = Htype(1,7) ;
149 Aeq(1,27) = Htype(1,8) ;
150 Aeq(1,28) = Htype(1,9) ;
151 Aeq(1,29) = 0 ;
152 Aeq(1,30) = Htype(1,10);
153
154 beq = csx ;
155
156 %%
157
158 % ------------------------ % ------------------------ % ...
    ------------------------ %
159 n=1 ;
160
161 %for n=1:ns
162 % Change in the Feed - disturbance
163
164 d_F0 = (F0_int-(Fdist*F0_int))+(2*(Fdist*F0_int)) - F0_int;
165 % .*rand(1,ns)
166 p.F0 = F0_int + d_F0(:,n) ;
167
168
169 % Using boundry conditions to keep a variable constant
170 % Feed:
171     lb(p.NT+8)  = p.F0;
172     ub(p.NT+8)  = p.F0;
173
174 % fmincon - optimizing for the disturbance with 1 DOF
175 [xopt(:,n),fvalopt(:,n),exitflag(:,n)]=fmincon(@fun1,x0,[],[],[],[],lb,ub,@nlcon1,options);
176 costopt(:,n)= xopt(p.NT+2,n)*60 ;
177 exit_opt(:,n) = exitflag(:,n) ;
178
179
180 % fmincon - no DOF left, process controlled with the self-opt. var.
181 [xsoc(:,n),fvalsoc(:,n),exitflag(:,n)]=fmincon(@fun1,x0,[],[],Aeq,beq,lb,ub,@nlcon1,options);
182 costsoc(:,n)= xsoc(p.NT+2,n)*60 ;
183 Feed = p.F0 ;
184
```

```
185 exit_soc(:,n) = exitflag(:,n) ;
186
187 av_resi_modelval(:,n) = sum(abs(J_diff))/50
188
189 %
190 %end
191
192 exitflag = [exit_opt; exit_soc] ;
193 loss(:,n) = (fvalopt(:,n)-fvalsoc(:,n))*60 ;
194 fvalopt;
195 fvalsoc;
196 Htype;
197 if Method == 1
198      disp('Hmat')
199 elseif  Method == 2
200      disp('Hexl')
201     exlns = (fvalopt(:,n)-fvalsoc(:,n))*60  ;
202 elseif    Method == 3
203      disp('Hns')
204     lossns = (fvalopt(:,n)-fvalsoc(:,n))*60  ;
205 end
```

```
1  %% CSTR-distillation process simulationtest2for5ywd.m %%
2  % 07.06.14
3  %%
4  % Script for optimization of reactor, separator and recycle process.
5  % The numerical description of the process is taken from Larsson et al ...
       (2003)
6  run generatedata_optScriptx2.m
7  % clc
8  % clear all
9  global p;
10
11 T=@(x)80-x*20;
12 %Boiling temperature of the light component
13 Tb =353;
14 %Boiling temperature of the heavy component
15 Th=373;
16
17 % Column parameters
18 p.qF    = 1;
19 p.NT    = 22;
20 p.NF    = 13;
21 p.alpha = 2;
22 p.Vmax  = 1500/60;
23 % CSTR parameters
24 p.F0 = 460/60*1.0;
25 p.zF0 = 0.9;
26 p.k1 = 0.341/60;
27 % Flags
28 p.OPTI=0;
29 p.case_I=1;
```

```
30
31 if p.case_I==1
32     % Constraints
33     lb=zeros(p.NT+8,1);
34     ub=[ones(p.NT,1); ones(8,1)*Inf];
35
36     % xB <= 0.0105
37     ub(1)=0.0105;
38     % Mr <= 2800;
39     ub(p.NT+7)=2800;
40     % F0 = fixed;
41     lb(p.NT+8)=p.F0;
42     ub(p.NT+8)=p.F0;
43 else
44     % Constraints
45     lb=zeros(p.NT+8,1);
46     ub=[ones(p.NT,1); ones(8,1)*Inf];
47
48     % xB <= 0.0105
49     ub(1)=0.0105;
50     % Mr <= 2800;
51     ub(p.NT+7)=2800;
52     % V <= Vmax;
53     ub(p.NT+2)=p.Vmax;
54 end
55
56 % Initial value
57 x0= [ones(1,p.NT)*0.5 10 15 5 5 1.1 0.5 1000 400/60]';
58
59 % fmincon options
60 options = optimset('TolFun',10e-6,'TolCon',10e-6,'MaxFunEvals',1e4,...
61 'Display','none','Algorithm','sqp','Diagnostics','off'...
62 );
63 % fmincon
64 [x,fval,exitflag]=fmincon(@fun1,x0,[],[],[],[],lb,ub,@nlcon1,options);
65
66 % Results
67 if p.case_I==1
68     casename='case I: min operation cost(energy)\n';
69 else
70     casename='case II: max production rate\n';
71 end
72 %Print the results
73 % results_fmincon=sprintf(strcat(...
74 %     casename,...
75 %     'feed rate,             F0[kmol/h]      = %1$0.1f\n',...
76 %     'reactor effluent,      F[kmol/h]       = %2$0.1f\n',...
77 %     'vapor boilup,          V[kmol/h]       = %3$0.1f\n',...
78 %     'reflux,                L[kmol/h]       = %4$0.1f\n',...
79 %     'recycle (distilate),   D[kmol/h]       = %5$0.1f\n',...
80 %     'recycle composition,   xD[molA/mol]    = %6$0.4f\n',...
81 %     'bottom composition,    xB[molA/mol]    = %7$0.4f\n',...
82 %     'reactor composition,   zF[molA/mol]    = %8$0.4f\n',...
83 %     'reactor holdup,        Mr[kmol/h]      = %9$0.0f\n',...
```

```
84  %     'Column temperatures,   T_i[C]          = T_3   T_8   T_13  ...
        T_18\n ',...
85  %     '                                        %10$0.1f  %11$0.1f  ...
        %12$0.1f  %13$0.1f\n'...
86  %     ),x(p.NT+8)*60,x(p.NT+5)*60,x(p.NT+2)*60,x(p.NT+1)*60, ...
        x(p.NT+3)*60,...
87  %     x(p.NT), x(1), ...
        x(p.NT+6),x(p.NT+7),T(x(3)),T(x(8)),T(x(13)),T(x(18)))
88
89  F0_int = x(p.NT+8);
90  %% Controlling the process
91  %for k=1:1
92
93
94
95  %Controlling only L -Brute force
96  % cs = x(p.NT+1) ;
97  % Aeq(1,23) = 1 ;
98  % Aeq(1,30) = 0 ;
99  % beq = cs ;
100
101 % Controlling x_D and L combinded soc
102 % cs = x(p.NT) - 0.0077*x(p.NT+1);
103 % Aeq(1,22) = 1 ;
104 % Aeq(1,23) = 0.0077 ;
105 % Aeq(1,30) = 0 ;
106 % beq = cs ;
107
108 %%
109 % Controlling with a full H-matrix :
110
111 load hmatrix.mat     % loads Hmat, Hns and Hexl
112 load optimalval.mat  % loads y_opt with temperatures
113
114
115 if Method == 1
116     Htype = Hmat;
117 elseif Method == 2
118     Htype = Hexl ;
119 else
120     Htype = [Hns(1,:)+Hns(2,:)+Hns(3,:)] ;
121 end
122
123  %% Data method - Without disturbance
124 %Hmat is calculated with temperatur-data multiplied with optimal ...
        values of
125 %y with optimal temperatures
126 csT = Htype*y_optT;
127
128 H1 = [Htype(1,1:4)] ; % Is to be transformed into composition
129 H2 = [Htype(1,5:10)]; % Is to be kept as it is
130 e=[1;1;1;1] ;
131 H1x = H1*(Tb-Th) ;   % Transforming the H-matrix
132 % Set point with composition
```

```
133 csx = csT-(H1*(e*Th)) ; % Transforming the set-point into composition
134
135 % The new H-matrix for compositon
136 Htype = [H1x H2];
137
138 % Using the self optimizing variable:
139 % Optimal setpoint
140
141 % Placing the H-elements at teh correct location
142 Aeq(1,8)  = Htype(1,1) ;
143 Aeq(1,13) = Htype(1,2) ;
144 Aeq(1,18) = Htype(1,3) ;
145 Aeq(1,22) = Htype(1,4) ;
146
147 Aeq(1,23) = Htype(1,5) ;
148 Aeq(1,24) = Htype(1,6) ;
149 Aeq(1,25) = Htype(1,7) ;
150 Aeq(1,26) = Htype(1,8) ;
151 Aeq(1,27) = Htype(1,9) ;
152 Aeq(1,28) = Htype(1,10) ;
153 Aeq(1,29) = 0 ;
154 Aeq(1,30) = 0;
155
156 beq = csx ;
157
158 %%
159
160 % ------------------------ % ------------------------ % ...
      ------------------------ %
161 n=1 ;
162 %ns=1 ;
163 %for n=1:ns
164 % Change in the Feed - disturbance
165
166 d_F0 = (F0_int-(Fdist*F0_int))+(2*(Fdist*F0_int)) - F0_int;
167 % .*rand(1,ns)
168 p.F0 = F0_int + d_F0(:,n) ;
169
170
171 % Using boundry conditions to keep a variable constant
172 % Feed:
173     lb(p.NT+8)  = p.F0;
174     ub(p.NT+8)  = p.F0;
175
176 % fmincon - optimizing for the disturbance with 1 DOF
177 [xopt(:,n),fvalopt(:,n),exitflag(:,n)]=fmincon(@fun1,x0,[],[],[],[],lb,ub,@nlcon1,options);
178 costopt(:,n)= xopt(p.NT+2,n)*60 ;
179 exit_opt(:,n) = exitflag(:,n) ;
180
181
182 % fmincon - no DOF left, process controlled with the self-opt. var.
183 [xsoc(:,n),fvalsoc(:,n),exitflag(:,n)]=fmincon(@fun1,x0,[],[],Aeq,beq,lb,ub,@nlcon1,options);
184 costsoc(:,n)= xsoc(p.NT+2,n)*60 ;
185 Feed = p.F0 ;
```

```
186
187 exit_soc(:,n) = exitflag(:,n) ;
188
189
190 av_resi_modelval(:,n) = sum(abs(J_diff))/50
191 %
192 %end
193
194 exitflag = [exit_opt; exit_soc] ;
195 loss(:,n) = (fvalopt(:,n)-fvalsoc(:,n))*60 ;
196 fvalopt;
197 fvalsoc;
198 Htype;
199 if Method == 1
200      disp('Hmat')
201 elseif  Method == 2
202      disp('Hexl')
203     exlns = (fvalopt(:,n)-fvalsoc(:,n))*60
204 elseif    Method == 3
205      disp('Hns')
206     lossns = (fvalopt(:,n)-fvalsoc(:,n))*60
207 end
```

```
1  %% CSTR-distillation process simulationtest2for5synd.m %%
2  % 07.06.14
3  %%
4  % Script for optimization of reactor, separator and recycle process.
5  % The numerical description of the process is taken from Larsson et al ...
       (2003)
6  run generatedata_optScriptx2.m
7  % clc
8  % clear all
9  global p;
10
11 T=@(x)80-x*20;
12 %Boiling temperature of the light component
13 Tb =353;
14 %Boiling temperature of the heavy component
15 Th=373;
16
17 % Column parameters
18 p.qF    = 1;
19 p.NT    = 22;
20 p.NF    = 13;
21 p.alpha = 2;
22 p.Vmax  = 1500/60;
23 % CSTR parameters
24 p.F0 = 460/60*1.0;
25 p.zF0 = 0.9;
26 p.k1 = 0.341/60;
27 % Flags
28 p.OPTI=0;
```

```
29 p.case_I=1;
30
31 if p.case_I==1
32     % Constraints
33     lb=zeros(p.NT+8,1);
34     ub=[ones(p.NT,1); ones(8,1)*Inf];
35
36     % xB <= 0.0105
37     ub(1)=0.0105;
38     % Mr <= 2800;
39     ub(p.NT+7)=2800;
40     % F0 = fixed;
41     lb(p.NT+8)=p.F0;
42     ub(p.NT+8)=p.F0;
43 else
44     % Constraints
45     lb=zeros(p.NT+8,1);
46     ub=[ones(p.NT,1); ones(8,1)*Inf];
47
48     % xB <= 0.0105
49     ub(1)=0.0105;
50     % Mr <= 2800;
51     ub(p.NT+7)=2800;
52     % V <= Vmax;
53     ub(p.NT+2)=p.Vmax;
54 end
55
56 % Initial value
57 x0= [ones(1,p.NT)*0.5 10 15 5 5 1.1 0.5 1000 400/60]';
58
59 % fmincon options
60 options = optimset('TolFun',10e-6,'TolCon',10e-6,'MaxFunEvals',1e4,...
61 'Display','none','Algorithm','sqp','Diagnostics','off'...
62 );
63 % fmincon
64 [x,fval,exitflag]=fmincon(@fun1,x0,[],[],[],[],lb,ub,@nlcon1,options);
65
66 % Results
67 if p.case_I==1
68     casename='case I: min operation cost(energy)\n';
69 else
70     casename='case II: max production rate\n';
71 end
72 %Print the results
73 % results_fmincon=sprintf(strcat(...
74 %     casename,...
75 %     'feed rate,             F0[kmol/h]      = %1$0.1f\n',...
76 %     'reactor effluent,      F[kmol/h]       = %2$0.1f\n',...
77 %     'vapor boilup,          V[kmol/h]       = %3$0.1f\n',...
78 %     'reflux,                L[kmol/h]       = %4$0.1f\n',...
79 %     'recycle (distilate),   D[kmol/h]       = %5$0.1f\n',...
80 %     'recycle composition,   xD[molA/mol]    = %6$0.4f\n',...
81 %     'bottom composition,    xB[molA/mol]    = %7$0.4f\n',...
82 %     'reactor composition,   zF[molA/mol]    = %8$0.4f\n',...
```

```
%     'reactor holdup,        Mr[kmol/h]      = %9$0.0f\n',...
%     'Column temperatures,   T_i[C]          = T_3   T_8   T_13  ...
    T_18\n ',...
%     '                                         %10$0.1f  %11$0.1f  ...
    %12$0.1f  %13$0.1f\n'...
%     ),x(p.NT+8)*60,x(p.NT+5)*60,x(p.NT+2)*60,x(p.NT+1)*60, ...
    x(p.NT+3)*60,...
%     x(p.NT), x(1), ...
    x(p.NT+6),x(p.NT+7),T(x(3)),T(x(8)),T(x(13)),T(x(18)))

%% Controlling the process

y = [x(8) ; x(13) ; x(18) ; x(22) ; x(23:28) ] ;
F0_int = x(p.NT+8);

%Controlling only L -Brute force
% cs = x(p.NT+1) ;
% Aeq(1,23) = 1 ;
% Aeq(1,30) = 0 ;
% beq = cs ;

% Controlling x_D and L combinded soc
% cs = x(p.NT) - 0.0077*x(p.NT+1);
% Aeq(1,22) = 1 ;
% Aeq(1,23) = 0.0077 ;
% Aeq(1,30) = 0 ;
% beq = cs ;

%%
% Controlling with a full H-matrix :
 load hmatrix.mat     % loads Hmat, Hns and Hexl
 load optimalval.mat  % loads y_opt with temperatures



if Method == 1
    Htype = Hmat;
elseif Method == 2
    Htype = Hexl ;
else
    Htype = [Hns(1,:)+Hns(2,:)+Hns(3,:)] ;
end

 %% Data method
% Hmat is calculated with temperatur-data multiplied with optimal ...
    values of
% y with optimal temperatures
csT = Htype*y_optT5;

H1 = [Htype(1,1:2)] ; % Is to be transformed into composition
H2 = [Htype(1,3:5)]; % Is to be kept as it is
e=[1;1] ;
H1x = H1*(Tb-Th) ;   % Transforming the H-matrix
% Set point with composition
```

```
132 csx = csT-(H1*(e*Th)) ; % Transforming the set-point into composition
133
134 % The new H-matrix for compositon
135 Htype = [H1x H2];
136
137 % Using the self optimizing variable:
138 % Optimal setpoint
139
140 % Placing the H-elements at teh correct location
141 Aeq(1,8)  = Htype(1,1) ;
142 Aeq(1,22) = Htype(1,2) ;
143 Aeq(1,23) = Htype(1,3) ;
144 Aeq(1,26) = Htype(1,4) ;
145 Aeq(1,30) = Htype(1,5) ;
146
147 beq = csx ;
148
149 %%
150
151 % ------------------------ % ------------------------ % ...
      ------------------------ %
152 n=1 ;
153
154 %for n=1:ns
155 % Change in the Feed - disturbance
156 d_F0 = (F0_int-(Fdist*F0_int))+(2*(Fdist*F0_int)) - F0_int;
157 % .*rand(1,ns)
158 p.F0 = F0_int + d_F0(:,n) ;
159
160
161 % Using boundry conditions to keep a variable constant
162 % Feed:
163     lb(p.NT+8)  = p.F0;
164     ub(p.NT+8)  = p.F0;
165
166 % fmincon - optimizing for the disturbance with 1 DOF
167 [xopt(:,n),fvalopt(:,n),exitflag(:,n)]=fmincon(@fun1,x0,[],[],[],[],lb,ub,@nlcon1,options);
168 costopt(:,n)= xopt(p.NT+2,n)*60 ;
169 exit_opt(:,n) = exitflag(:,n) ;
170
171
172 % fmincon - no DOF left, process controlled with the self-opt. var.
173 [xsoc(:,n),fvalsoc(:,n),exitflag(:,n)]=fmincon(@fun1,x0,[],[],Aeq,beq,lb,ub,@nlcon1,options);
174 costsoc(:,n)= xsoc(p.NT+2,n)*60 ;
175 Feed = p.F0 ;
176
177 exit_soc(:,n) = exitflag(:,n) ;
178
179
180 av_resi_modelval(:,n) = sum(abs(J_diff))/50
181 %
182 %end
183
184 exitflag = [exit_opt; exit_soc] ;
```

```
loss(:,n) = (fvalopt(:,n)-fvalsoc(:,n))*60 ;
fvalopt;
fvalsoc;
Htype;
if Method == 1
     disp('Hmat')
elseif  Method == 2
     disp('Hexl')
    exlns = (fvalopt(:,n)-fvalsoc(:,n))*60  ;
elseif    Method == 3
     disp('Hns')
    lossns = (fvalopt(:,n)-fvalsoc(:,n))*60  ;
end
```

```
% Script for optimization of reactor, separator and recycle process.
% The numerical description of the process is taken from Larsson et al ...
    (2003)

run generatedata_optScriptx2.m
% clc
% clear all
global p;

T=@(x)80-x*20;
%Boiling temperature of the light component
Tb =353;
%Boiling temperature of the heavy component
Th=373;

% Column parameters
p.qF    = 1;
p.NT    = 22;
p.NF    = 13;
p.alpha = 2;
p.Vmax  = 1500/60;
% CSTR parameters
p.F0 = 460/60*1.0;
p.zF0 = 0.9;
p.k1 = 0.341/60;
% Flags
p.OPTI=0;
p.case_I=1;

if p.case_I==1
    % Constraints
    lb=zeros(p.NT+8,1);
    ub=[ones(p.NT,1); ones(8,1)*Inf];

    % xB <= 0.0105
    ub(1)=0.0105;
    % Mr <= 2800;
    ub(p.NT+7)=2800;
```

```
38      % F0 = fixed;
39      lb(p.NT+8)=p.F0;
40      ub(p.NT+8)=p.F0;
41  else
42      % Constraints
43      lb=zeros(p.NT+8,1);
44      ub=[ones(p.NT,1); ones(8,1)*Inf];
45
46      % xB <= 0.0105
47      ub(1)=0.0105;
48      % Mr <= 2800;
49      ub(p.NT+7)=2800;
50      % V <= Vmax;
51      ub(p.NT+2)=p.Vmax;
52  end
53
54  % Initial value
55  x0= [ones(1,p.NT)*0.5 10 15 5 5 1.1 0.5 1000 400/60]';
56
57  % fmincon options
58  options = optimset('TolFun',10e-6,'TolCon',10e-6,'MaxFunEvals',1e4,...
59  'Display','none','Algorithm','sqp','Diagnostics','off'...
60  );
61  % fmincon
62  [x,fval,exitflag]=fmincon(@fun1,x0,[],[],[],[],lb,ub,@nlcon1,options);
63
64  % Results
65  if p.case_I==1
66      casename='case I: min operation cost(energy)\n';
67  else
68      casename='case II: max production rate\n';
69  end
70  %Print the results
71  % results_fmincon=sprintf(strcat(...
72  %     casename,...
73  %     'feed rate,             F0[kmol/h]      = %1$0.1f\n',...
74  %     'reactor effluent,      F[kmol/h]       = %2$0.1f\n',...
75  %     'vapor boilup,          V[kmol/h]       = %3$0.1f\n',...
76  %     'reflux,                L[kmol/h]       = %4$0.1f\n',...
77  %     'recycle (distilate),   D[kmol/h]       = %5$0.1f\n',...
78  %     'recycle composition,   xD[molA/mol]    = %6$0.4f\n',...
79  %     'bottom composition,    xB[molA/mol]    = %7$0.4f\n',...
80  %     'reactor composition,   zF[molA/mol]    = %8$0.4f\n',...
81  %     'reactor holdup,        Mr[kmol/h]      = %9$0.0f\n',...
82  %     'Column temperatures,   T_i[C]          = T_3   T_8   T_13  ...
        T_18\n ',...
83  %     '                                         %10$0.1f  %11$0.1f  ...
        %12$0.1f  %13$0.1f\n'...
84  %     ),x(p.NT+8)*60,x(p.NT+5)*60,x(p.NT+2)*60,x(p.NT+1)*60, ...
        x(p.NT+3)*60,...
85  %     x(p.NT), x(1), ...
        x(p.NT+6),x(p.NT+7),T(x(3)),T(x(8)),T(x(13)),T(x(18)))
86
87  %% Controlling the process
```

```
88
89  y = [x(8) ; x(13) ; x(18) ; x(22) ; x(23:28) ] ;
90  F0_int = x(p.NT+8);
91
92  %Controlling only L -Brute force
93  % cs = x(p.NT+1) ;
94  % Aeq(1,23) = 1 ;
95  % Aeq(1,30) = 0 ;
96  % beq = cs ;
97
98  % Controlling x_D and L combinded soc
99  % cs = x(p.NT) - 0.0077*x(p.NT+1);
100 % Aeq(1,22) = 1 ;
101 % Aeq(1,23) = 0.0077 ;
102 % Aeq(1,30) = 0 ;
103 % beq = cs ;
104
105 %%
106 % Controlling with a full H-matrix :
107  load hmatrix.mat     % loads Hmat, Hns and Hexl
108  load optimalval.mat  % loads y_opt with temperatures
109
110
111
112 if Method == 1
113     Htype = Hmat;
114 elseif Method == 2
115     Htype = Hexl ;
116 else
117     Htype = [Hns(1,:)+Hns(2,:)+Hns(3,:)] ;
118 end
119
120  %% Data method
121 % Hmat is calculated with temperatur-data multiplied with optimal ...
       values of
122 % y with optimal temperatures
123 csT = Htype*y_optT5nd;
124
125 H1 = [Htype(1,1:2)] ; % Is to be transformed into composition
126 H2 = [Htype(1,3:5)]; % Is to be kept as it is
127 e=[1;1] ;
128 H1x = H1*(Tb-Th) ;   % Transforming the H-matrix
129 % Set point with composition
130 csx = csT-(H1*(e*Th)) ; % Transforming the set-point into composition
131
132 % The new H-matrix for compositon
133 Htype = [H1x H2];
134
135 % Using the self optimizing variable:
136 % Optimal setpoint
137
138 % Placing the H-elements at teh correct location
139 Aeq(1,8)  = Htype(1,1) ;
140 Aeq(1,22) = Htype(1,2) ;
```

```
141 Aeq(1,23) = Htype(1,3) ;
142 Aeq(1,26) = Htype(1,4) ;
143 Aeq(1,27) = Htype(1,5) ;
144 Aeq(1,30) = 0 ;
145
146 beq = csx ;
147
148 %%
149
150 % ———————————————— % ———————————————— % ... ———————————————— %
151 n=1 ;
152
153 %for n=1:ns
154 % Change in the Feed – disturbance
155 %
156 d_F0 = (F0_int-(Fdist*F0_int))+(2*(Fdist*F0_int)) - F0_int;
157 % .*rand(1,ns)
158 p.F0 = F0_int + d_F0(:,n) ;
159
160
161 % Using boundry conditions to keep a variable constant
162 % Feed:
163     lb(p.NT+8)  = p.F0;
164     ub(p.NT+8)  = p.F0;
165
166 % fmincon – optimizing for the disturbance with 1 DOF
167 [xopt(:,n),fvalopt(:,n),exitflag(:,n)]=fmincon(@fun1,x0,[],[],[],[],lb,ub,@nlcon1,options);
168 costopt(:,n)= xopt(p.NT+2,n)*60 ;
169 exit_opt(:,n) = exitflag(:,n) ;
170
171
172 % fmincon – no DOF left, process controlled with the self-opt. var.
173 [xsoc(:,n),fvalsoc(:,n),exitflag(:,n)]=fmincon(@fun1,x0,[],[],Aeq,beq,lb,ub,@nlcon1,options);
174 costsoc(:,n)= xsoc(p.NT+2,n)*60 ;
175 Feed = p.F0 ;
176
177 exit_soc(:,n) = exitflag(:,n) ;
178
179 av_resi_modelval(:,n) = sum(abs(J_diff))/50
180 %
181 %end
182 exitflag = [exit_opt; exit_soc] ;
183 loss(:,n) = (fvalopt(:,n)-fvalsoc(:,n))*60 ;
184 fvalopt;
185 fvalsoc;
186 Htype;
187 if Method == 1
188      disp('Hmat')
189 elseif  Method == 2
190      disp('Hexl')
191     exlns = (fvalopt(:,n)-fvalsoc(:,n))*60  ;
192 elseif    Method == 3
193      disp('Hns')
```

```
194       lossns = (fvalopt(:,n)-fvalsoc(:,n))*60  ;
195   end
```