Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Chemical Engineering

Hege Landbø

Control of a Continuous Flow Microreactor:

Comparison of Decentralized PI-control, PI-control With Decoupling, and MPC

Master's thesis in Chemical Engineering
Supervisor: Johannes Jäschke
Co-supervisor: Marcin Dudek
June 2021

NTNU
Norwegian University of Science and Technology

Hege Landbø

Control of a Continuous Flow Microreactor:

Comparison of Decentralized PI-control, PI-control With Decoupling, and MPC

Master's thesis in Chemical Engineering
Supervisor: Johannes Jäschke
Co-supervisor: Marcin Dudek
June 2021

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Chemical Engineering

NTNU
Norwegian University of
Science and Technology

Preface

This master thesis concludes my 5-year Master of Science degree at the Department of Chemical Engineering, with specialization in Process Systems Engineering, at the Norwegian University of Science and Technology, NTNU. The experimental work performed for this thesis, during the spring of 2021, was done at the Ugelstad Laboratory under the supervision of Marcin Dudek (Co-supervisor).

I would like to thank my supervisor, Associate professor Johannes Jäschke for his help, guidance and support on both my specialization project and master thesis. I appreciate him giving me room to try new things, and having faith in me succeeding in my attempts. I would also like to thank my Co-supervisor Marcin Dudek, who has set aside so much of his time to share his knowledge and help me in a completely new field of research. His enthusiasm for the research, and guidance in the experimental work has been of great importance, and is much appreciated. I also wish to thank doctoral student Zawadi Ntengua Mdoe, for his help with the model predictive controller. I have learned a lot from working interchangeably with the two groups, Process systems engineering and the Ugelstad laboratory, and i believe that many students will benefit from such a cooperation in the future.

Declaration of Compliance

I, Hege Landbø, hereby declare that this is an independent work according to the exam regulations of the Norwegian University of Science and Technology (NTNU).
Signature:

Place and Date: Trondheim - Gløshaugen, June 2021

Abstract

Microreaction technology has become an increasingly popular field within academic and industrial research. The small dimensions in which processes are performed, introduces many advantages and possibilities. Along with the increased number of applications of microreactors, the need for appropriate control systems to strictly regulate the process parameters, have grown. As microreactors are often complex multivariable systems, a controller will often face challenges related to strong interactions, and the need for fast and accurate control. This thesis aims to implement three different control structures for the control of concentration and flow of a continuous flow microreactor. The three control structures are decentralized PI-control, decoupled PI-control and model predictive control (MPC). This is done for the purpose of comparing their performance, and evaluating which of the three is best suited for this type of system.

The microreactor system studied in this work consisted of two syringe pumps, one with dye and one with water, connected to micro-tubes. The tubes were then connected in a mixing tee to one outflow. A spectrophotometer was used to measure the concentration of the outflow, while the total flow was set to be the sum of the two input flows, which were assumed to be perfectly regulated by the pumps. The three controllers were compared by performing steps in the setpoint of both the total outflow and the concentration, and then evaluating the performance in terms of setpoint tracking of each controller.

Both the PI-controller with decoupling, and the MPC, outperformed the decentralized PI-controller with lower integral of absolute error (IAE) and mean absolute error (MAE), and lower settling times. The decoupled PI-controller was able to reduce the interactions in the system, and was relatively easy to implement. The MPC was a bit more tedious to tune, but was also more or less unaffected by the interactions. For this system, the decoupled PI-controller was found to be the best fit, as it was more easily tuned and implemented, and gave good results in terms of setpoint tracking. However, it should be noted that for more complex systems, the decoupling equations may become more complicated, and one should therefore not discard MPC in further evaluations of control structures for microreactor systems.

Sammendrag

Mikroreaktor-teknologi har sett en sterk økning i popularitet innenfor akademisk og industriell forskning de siste tiårene. De små dimensjonene introduserer mange fordeler og muligheter. Parallelt med økningen i antall applikasjoner av mikroreaktorer, har også behovet for passende regulerings-systemer for streng kontroll av prosess-parametre økt. Ettersom mikroreaktorer ofte er komplekse multivariable systemer, vil regulatorer ofte møte på utfordringer relatert til stekre interaksjoner mellom prosess-variablene, og behovet for rask of nøyaktig kontroll av parametrene. Denne oppgaven har som mål å implementere tre ulike regulatorer for reguleringen av konsentrasjon og strømningen av væske i en mikroreaktor. De tre regulator-strukturene er PI-regulator, PI-regulator med dekobling og modell-basert prediktiv regulering (MPC). Dette blir gjort med det formål å sammenlikne deres ytelse, og evaluere hvilken av de tre strukturene som er best egnet for denne typen mikroreaktor.

Mikroreaktoren aktuell for denne oppgaven bestod av to sprøytepumper, en som inneholdt fargeløsning og en med vann, tilkoblet mikrorør. Rørene ble igjen sammenkoblet i et T-formet kontaktpunkt til ett felles utgangsrør. Et spektrofotometer ble brukt til å måle konsentrasjonen i utstrømningen, og den totale strømningen ble satt til å være summen av strømningen av fargeløsning og vann inn i systemet. Disse ble antatt å være perfekt regulert av pumpene selv. De tre regulatorene ble sammenliknet ved å gjøre stegvise endringer i setpunktet til både den totale strømningen og konsentrasjonen, og evaluere ytelsen relatert til hvordan hver regulator følger setpunktet.

Både PI-regulatoren med dekobling og MPC'en, overgikk PI-regulatoren med lavere integral av absolutt error (IAE) og gjennomsnittlig absolutt error (MAE), i tillegg til å bruke mindre til å nå "steady state". Den dekoblede PI-regulatoren klarte å redusere interaksjonene i systemet, og var relativt enkel å implementere. MPC'en var litt mer tidkrevende å tune, men var også mer eller mindre upåvirket av interaksjonene. For dette systemet ble det konkludert med at PI-regulatoren med dekobling var best egnet ettersom den var enklere å tune implementere enn MPC, og viste gode resultater i form av å følge det bestemte setpunktet. Det burde allikevel nevnes at for mer komplekse systemer kan likningene for dekobling bli kompliserte, og man bør derfor ikke forkaste MPC i de videre evalueringene av kontrollstrukturer for mikroreaktorer.

Table of Contents

List of Tables

List of Figures

Abbreviations

CV	Controlled variable
FOPTD	First Order Plus Time Delay
IAE	Integral of Absolute Error
IPA	Isopropanol
LTCC	Low Temperature Cofired Ceramics
MAE	Mean Absolute Error
MIMO	Multiple-Input-Multiple-Output
MPC	Model Predictive Control
MV	Manipulated variable
PCR	Polymerase Chain Reaction
PFA	Perfluoroalkoxy Alkane Polymers
PID	Proportional-Integral-Derivative
QP	Quadratic Program
RGA	Relative Gain Array
SIMC	Skogestad Internal Model Control

CHAPTER 1

Introduction

During the past two decades, microreaction technology has become increasingly popular within academic research, and have also generated interest within the process industry. Providing the advantages of micro-structures, such as enhanced heat and mass transfer, microreactors enables a wider operating window, such that processes in a microreactor can be operated outside the operating window of a conventional reactor. It also enables tighter control of the process, giving higher yields and selectivity [1]. The increased attention around micro reaction technology, have boosted the need for optimal control structures for such systems. For most reactions and synthesis, a precise manipulation of a solution or the reaction reagents, is crucial for achieving high yields of pure product components. For microreactors this is especially important as the process parameters change at a much higher rate compared to large scale processes, which makes tight control of the process highly important. It also makes control more achievable due to the many detection techniques enabling strict process monitoring [2, 3].

Many researchers have over the years developed different types of controllers for microreactors, and the most frequently mentioned is the well known Proportional-Integral-Derivative (PID)-controller. PID-controllers exists in many various forms and delivers robust control and acceptable performance for a large range of applications. In addition to being economically beneficial, the theory and adaption is widely known within the industry. Even though many complex control strategies have been developed for microreactor systems, PID controllers are the most commonly implemented for practical use [4]. However, its been mentioned that there are some challenges regarding the applications of standard PID-controllers, due to the complicated and coupled structures that often occur in microreactors. The tuning of such controllers could in certain cases become a tedious process, and it may be relatively complicated for end-users [3].

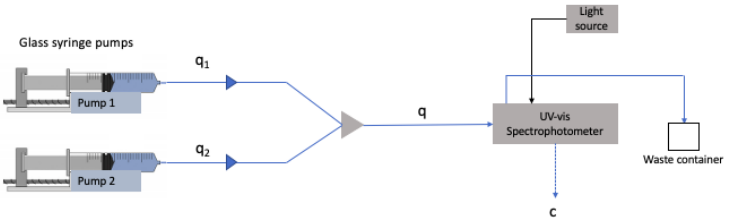


Figure 1.1: Schematic model of the current process, with two input flows and one mixed output flow. The concentration is measured by the spectrophotometer.

The use of a technique known as decoupling for a PID-controller, is a method for eliminating interactions between the process variables, and thus possibly solving some of the challenges with coupled systems. However, depending on the decoupling method used, the control tuning process could become as complicated, if not more complicated, as for standard PID-control. The decoupling method will also increase in complexity with the number of variables in the system. This method has, to the best of the author's knowledge, not been tested for micro reaction systems.

Model predictive controllers (MPC) is an advanced control technique that has been proven applicable to multivariable nonlinear processes, and is able to handle process constraints. They are often used for complicated systems with multiple inputs and outputs, but is not as well documented for microreactors as PID-controllers. The use of an MPC may, to some degree, be a more intuitive process for the user, but the development requires an accurate model and lot more knowledge about the system being controlled [5].

In this thesis the three different controllers mentioned above, are implemented and tested on a simple microreactor system containing two input flows and one output flow, as shown in Figure 1.1. The controllers control the concentration, c, and fluid flow, q, of the outflow of the system. To measure the concentration of the outflow, a spectrophotometer is connected to the system, measuring the absorbance of the fluid.

The system is tightly coupled, which means that changing one of the input flows will have a strong effect on both the concentration and the total flow. This makes control of these variables challenging. This work aims to implement the three mentioned controllers to see which performs best in terms of tracking a given setpoint, ease of implementation, and which are more capable to overcome the challenges induced by the interactions occurring in the coupled system. The experimental work done in this thesis can be seen as an introductory experiment for the further development of a control system for this microreactor.

The thesis is structured in the following way;

- **Chapter 2** contains background theory on the main topics: microreactors and spectroscopic detection methods, control systems for microreactors, PID-control and

decoupling, and MPC.

- The experimental methodologies is presented in **Chapter 3**, giving an overview of the software and lab equipment used, and the experimental procedure.
- **Chapter 4** gives a description of the preparational work done prior to testing the controllers. Here, the method for measuring concentration with the spectrophotometer and the model development, is presented.
- In **Chapter 5**, the method, results and discussion is presented for each of the three experiments run with decentralized PI-control, PI-control with decoupling and MPC.
- The last chapter, **Chapter 6**, consists of an overall discussion and comparison of the results, conclusion and recommendations for future work.

CHAPTER 2

Background

The first section of this chapter gives a general overview on microreactors and the spectroscopic detection techniques used with these systems. Specifically, spectrophotometry is described, as this is the spectroscopic analysis used for experiments in this thesis. Lastly, some of the most relevant work published on control systems for microreactors, is briefly presented.

Section 2.2 of this chapter covers some main subjects within control theory that will be relevant for the work done in this thesis. These subjects include basic theory on PID controllers, a method for model-based tuning, and multiloop control and decoupling for multiple-input-multiple-output (MIMO) systems.

Section 2.3 presents the basic concept of model predictive control, how it works and guidelines for tuning the control parameters.

2.1 Microreactors

Microreactors are miniaturized chemical reaction systems that, in general, are 3-dimensional structures with inner dimensions under a millimeter in size, running continuous flow operations [6]. Research on chemical reactions in micro structured systems began evolving in the late 1980's and the early 1990's, and the first micro heat-exchanger was built in Germany in 1989. During the past three decades, micro reaction technology has gained increased interest within academic and research areas, and the field is growing rapidly due to the many advantages of their miniature dimensions [2, 6].

Microreactors come in many different structures, in which the details depend on the end use of the reactor. Two main types of structures are chip-type microreactors and mi-

crocapillary reactors, also called microchannels. The material used for these reactors also depends on the end use, in which chemical compatibility with the components being used, and the required pressure and temperature for the reactions are among the factors that have to be taken into account [2].

The common uses of microreactors are screening in microanalytical chemistry, biological analysis of cells and proteins, and reaction kinetics and mechanisms studies [7]. A main benefit of microstructured reactors is the high surface-area-to-volume ratio. This gives high heat exchanging efficiency that can allow for very fast heating and cooling, making these reactors beneficial for chemical reactions with highly exothermic or endothermic reactions, and processes where isothermal conditions should be maintained. This makes it possible to investigate reactions with conditions that are e.g. within explosion limits of the reactants, which is not possible on a larger scale [1]. In addition to high heat exchanging efficiency, diffusion times are significantly reduced due to the small dimensions of the reactor, resulting in effective and fast mixing processes [6]. Overall, the improved mass- and heat-transfer rate provides the possibility for tighter control of the process [1]. One is then able to avoid typical problems occurring in reactors of bigger dimensions, such as large temperature and concentration gradients. These problems often result in locally overheated areas, production of side-products and lower production yields [8]. The use of microreactors have also opened up for possibilities of developing complex reactions due to the reduced amount of materials needed to optimize the conditions [7].

In recent years, microreaction technology have also gained increased interest outside the academic field, for instance within the pharmaceutical and chemical industry [7]. In addition to the advantages mentioned above, two main driving forces for the interest within the industry are often mentioned. One is the ability to closely investigate and control chemical processes and therefore possibly perform safe and reliable scale-up to pilot and production scale. The scale-up is done with by combined method of increasing dimensions and replicating optimized miniature units [9]. The other driving force is the possibility for optimization of process plants already in operation [6].

2.1.1 Spectroscopic detection methods for microreactors

Online process monitoring is an important part of microreaction technology for the ability of online observation of chemical and physical properties within the reactor. For instance, it provides possibilities for accurate control, acquiring kinetic information and discovering unrevealed species present in a reaction [1].

Several detection techniques for process monitoring have been developed for integration with microreactors. A large amount of these detection methods involve some kind of spectroscopic analysis. Spectroscopy is generally defined as the study of absorption and emission of electromagnetic radiation by matter [10]. All substances absorb light or other radiation at specific wavelengths, depending on the difference in energy of excited electrons in the molecule [11]. This gives rise to characteristic peaks in various spectra for different substances, which can be used to identify concentration and composition of the

substance [1].

The spectroscopic analysis often used with microreactors are fluorescence, ultraviolet-visible (UV-vis), infrared (IR), Raman, and nuclear magnetic resonance (NMR) spectroscopy [1]. These sub-disciplines within spectroscopy are distinguished by using different type of radiation energy, as insinuated by their names. In this work, UV-vis spectroscopy is utilized, and this is described next in more detail.

UV-vis spectroscopy is considered the most widely used spectrophotometric technique for the analysis of a variety of compounds [11]. A UV-vis spectrophotometer can be utilized to continuously measure the concentration of a flow, by measuring the absorbance of UV-vis light in a substance. The measurable UV wavelength is in the range from approximately 180-400 nm, while the visible component goes up to around 800 nm [12]. In the spectrophotometer a light beam is directed through the sample, and the intensity of the light reaching the detector on the other side is measured. This intensity, I, is directly related to the absorbance, A, of the sample through Beer Lamberts law, [13]

$$A = log_{10}\left(\frac{I_0}{I}\right) = \epsilon cl \tag{2.1}$$

Where I_0 is the reference intensity, i.e. the intensity of the light beam when sent through a pure solvent. Solvents are usually transparent, and ideally should not absorb light in the UV-vis range. ϵ is the molar absorption coefficient or molar absorptivity, l is the optical path length, and c is the concentration of a solute. Concentration can then be calculated by rearranging this equation, however, this requires knowing the values of ϵ and l. The law states that there is a linear relationship between the absorbance and the concentration for diluted solutions, which is the basis for a second method for determining the concentration. With this other method, one utilizes the linear equation describing the relationship between a set of measured absorbance values for a substance, and the corresponding known concentrations [13].

Deviations from the law can occur in certain circumstances, in which the relationship between the concentration and the absorbance can become non-linear and give wrongful calculations. There are mostly three categories of deviations, which are important to keep in mind when applying the linear equation for concentration measurements. One is a limitation in the law itself, where too high concentrations of the solution causes interactions between the solute molecules or with the solvent resulting in different behaviour. One should therefore make sure to use sufficiently diluted solutions. Another category of deviations are chemical deviations, which are caused by the solvent e.g. changing resonance when pH changes. The third category involves instrumental limitations [14]. Beer Lambert‘s law is followed when the radiation is of a monochromatic source, and thus is only strictly linear if the molar absorptivity of a molecule at two different wavelengths are the same. As the source of radiation in a spectrophotometer is commonly polychromatic (has multiple optical frequencies), it needs to be separated into all its wavelengths by grating units or filters to create a monochromatic beam. This may cause deviations, and is a factor

that falls into the category of instrumental limitations. For this reason it is common to measure absorbance at a wavelength where the curve is at a peak. This is a point on the absorbance vs. wavelength curve where there is minimal change in absorbance per unit change in wavelength [14].

2.1.2 Control systems for microreactors

Online process monitoring of microfluidic systems has, as mentioned, opened up for the possibility for accurate control of various conditions within a microreactor. This is important for the ability to study complex reactions in need for strictly controlled conditions. One of the biggest challenges has therefore been to develop and implement appropriate control structures for these systems [4]. Several papers on the development of different types of controllers for microreactors, have been published over the years during the growth of micro reaction technology. A majority of these papers have focused on the theoretical approach and practical implementation of PID-controllers, in which some of the most relevant works are referenced below.

Joon Lee et al.(2004) proposed a PI-controller for control of a thermal microsystem for polymerase chain reaction (PCR) [15], and Besser et al.(2006) demonstrated the control of a miniature reactor performing a catalytic steam reforming reaction of a methanol-water mixture, by implementation of a standard PID-control structure [4]. Quiram et al.(2007) employed a PID-controller for automation of gas-phase catalyzed reactions, however, this paper focused more on the design of the process than the actual implementation of the controller [16]. Dinca et al.(2009) introduced a detailed description of the design of a PID-controller for temperature control of a PCR microreactor [17]. In this paper, they concluded that a simple PID algorithm performed better for the PCR microreactor, than more elaborated techniques such as Fuzzy-PID or predictive control.

Gómez de Pedro et al. (2012) presented in their study a low temperature cofired ceramics (LTCC) based microfluidic system for high temperature reactions [18]. By integrating the microfluidics and a miniaturized thermally controlled platform, they were able to perform continuous size-controlled synthesis of cadmium selene (CdSe) quantum dots. They developed a micro-controller that included a digital PID-controller with input signal from an analog-to-digital converter. One important point made in this paper is that many other reaction parameters such as stirring ratio, mass transference and fluctuations of temperature and concentration, are difficult to control in batch reactors. However, with microreactors combined with microfluidic devices, the parameters can be automatically controlled with computer assisted systems. Heo et al.(2016) developed a new Tuning-free controller for flow control in a microfluidic network, which was then compared to a conventional PID-controller [3]. As microfluidic systems are coupled multiple-input-multiple-output (MIMO) systems, they argue that a standard PID-controller cannot easily be implemented if the system is complex, and that its tuning process is tedious and difficult to conduct for end-users.

Model predictive control is especially interesting in the microreaction field, as the systems are often complex MIMO systems, as mentioned above. Despite this, and the fact that control of microreactors have been an important part of the research on micro reaction technology the last two decades, there have been few reports on the use of MPC for these systems. Some of the relevant papers found are mentioned below.

Bleris et al.(2006) provides in their paper an application of real-time model predictive control of a flow stream in a microfluidic system [19]. They focus on the real-time control of microflows in Systems-on-a-chip (SoC), and mention the technological challenges regarding the computational requirements of an MPC, and the fast dynamics of microscale processes. However, they are able to implement the controller with sample rates as low as 0.023 sec. A similar article was published the same year, by the same authors, proposing an MPC for temperature control in a wafer cross-section geometry, and temperature control in a non-isothermal fluid flow in a microdevice [20]. Maddala and Rengaswamy (2012) demonstrated how an MPC with appropriate objective function settings could be used to control the spatial and temporal dynamics of droplets in a microfluidic system [21]. They emphasize the importance of such control, e.g. for droplets to arrive at different locations at exact times, pass a certain point in a specific sequence or to produce chemical or digital signals. They used their previously proposed control strategy with MPC, to generate digital signals at the exit of a microfluidic loop [22]. An issue that is mentioned also in this paper, is the challenge of computational complexity when performed online.

2.2 PID-control

Consider a process with a given number of process inputs, u, and process outputs, y. Controlling this process would mean to adjust the manipulated variables (MV's) of the system, so that the controlled variables (CV's) are behaving in a desired way, resistant to any disturbances that might occur in the process. Usually the MV's are the process inputs, and the CV's are the measured process outputs. Figure 2.1 shows a block representation of a conventional feedback control loop, also called closed loop control.

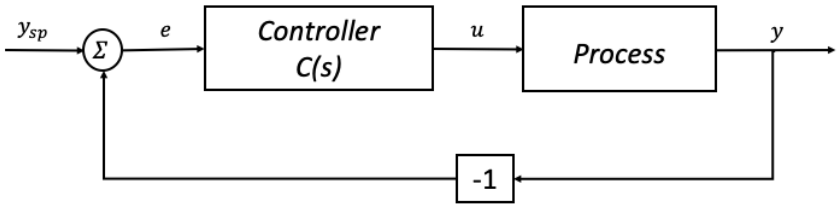


Figure 2.1: A block representation of a single negative feedback loop, representing closed loop control.

The process output, y, is the controlled variable that is being kept at its desired setpoint y_{sp}. The difference between the measured output and the setpoint value, is the control error, $e = y_{sp} - y$. This error is the input to the controller, which changes the input variable, u, to the process so that this error is reduced. The negative sign indicates that this is a negative feedback loop, and the MV, u, will move to counteract the deviation in the CV, y [23]. The controller in the figure, denoted by $C(s)$, is the the Laplace domain transfer function of a compensator that determines the manipulated variable, u. A PID-controller is used as a compensator in PID-control systems [24].

2.2.1 Properties of PID-controllers

PID-control is the main tool for feedback control, and the first controller of this kind came on the market in 1939. They have up until this day been the most widely used controllers in the process industry [24]. PID-controllers consists of three elements with three different functions, namely proportional (P), integral (I) and derivative (D) functions, as shown in the time domain representation in Equation 2.2.

$$C(t) = K_P e(t) + K_I \int_0^t e(t)dt + K_D \frac{d}{dt} e(t) \tag{2.2}$$

K_P, K_I and K_D are the coefficients for the proportional, integral and derivative terms, respectively. The three elements stands for different controller actions, and can be described as followed:

- **Proportional** element - Proportional to the current error, e, at time t.

- **Integral** element - Proportional to the integral of the error, $\int e$, up to the time t, i.e. it takes into account the accumulated error.
- **Derivative** element - Proportional to the derivative of the error, $\frac{d}{dt}e$, at time t. This can be seen as a prediction of future error.

This means that the PID-controller takes into account the past, present and predicted future error when determining the process input, u [24]. With proportional action only, there will always be some bias term keeping the the steady state error at non-zero, resulting in an offset from the desired setpoint value. With integral action, even the smallest error will over time result in a change in the control signal, which always leads to a zero steady state error. When adding the derivative term to the controller function, control performance may be improved. The derivative term provides an extrapolation of the error by the tangent of the curve, and thus a prediction of the process output is made [23].

In the ideal form, the Laplace transfer function of the controller, $C(s)$, can be defined as,

$$C(s) = \frac{u(s)}{e(s)} = K_c(1 + \frac{1}{\tau_I s} + \tau_D s) \tag{2.3}$$

In which the controller output, $u(s)$, is the sum of the three terms multiplied by the transfer function of the error, $e(s)$. The parameters K_c, τ_I and τ_D are the tunable PID-parameters that can be tuned through specified methods, trial and error or both, for optimal control. Tuning with trial and error alone is however tedious work, and it is not easy to find good values without a well motivated systematic procedure [25]. The mentioned PID parameters are referred to as the proportional gain, integral time and derivative time, respectively [26].

Within the PID-controller family are P-controllers, PI-controllers, PD-controllers and PID-controllers [26]. For this thesis, the main focus will be kept on PI-controllers which include the proportional and integral elements, but have neglected the derivative element. This because the process discussed in this theses has first order dynamics, which will be described below, and derivative action is recommended primarily for processes with dominant second order dynamics. The derivative term also has disadvantages such as increased input usage and noise sensitivity [25].

For control, it is convenient to express all variables as deviation variables, with the nominal value as reference;

- $\Delta u = u - u_{nom}$
- $\Delta y = y - y_{nom}$
- $\Delta y_{sp} = y_{sp} - y_{sp,nom}$

To simplify notation, the Δ denoting deviation variables will be omitted in the rest of this thesis.

2.2.2 Model-based tuning of PI-controller

When tuning the controller parameters with a defined model-based tuning method, one needs a mathematical model that describes the process and its characteristics. One common method for identifying the process characteristics and the resulting best fit frequency domain model, is to perform a unit step to the input variable, $u(t)$, and examine the step response of the output variable, $y(t)$ [24]. Figure 2.2 show examples of dynamic step responses of three different systems [27].

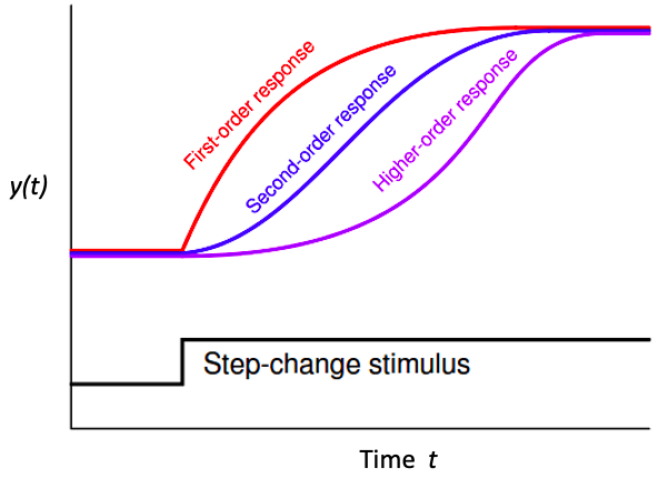


Figure 2.2: Example of three dynamic step responses to a unit step in the input. The red curve shows a typical form of a first order response, the blue curve shows a second order response, and the purple curve shows an example of a response of higher order.

By examining the response curve, one should be able to decide the model that will give the best fit, and a general rule is that more parameters describing the system gives a more accurate, but also more complicated model. Equation 2.4 show the open loop transfer function $G(s)$ representing a first order plus time delay model (FOPTD-model).

$$G(s)_{1^{st}order} = \frac{k}{(\tau_1 s + 1)} e^{-\theta s} \tag{2.4}$$

This transfer function then describes the open loop dynamics between $y(s)$ and $u(s)$, $y(s) = G(s)u(s)$. As this is the model that will be approximated from the experiments completed for this thesis, higher order systems, as well as other special cases of process dynamics, will not be discussed in further details. The parameters of $G(s)$ can then be approximated graphically from the step response curve, as shown in Figure 2.3, where $k = \frac{\Delta y}{\Delta u}$.

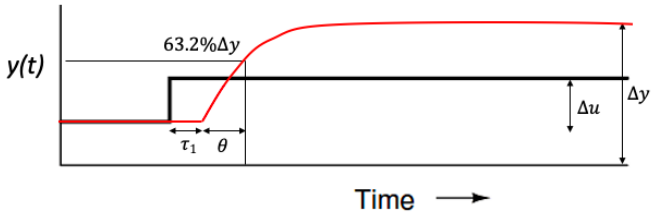


Figure 2.3: Graphical method for approximation of a first order plus time delay model. The parameters of the model, $G(s)$, are shown in the figure where k is given by $k = \frac{\Delta y}{\Delta u}$.

In Equation 2.4, k represents the steady state gain, i.e. the change in the output induced by the corresponding input. As seen in Figure 2.3, τ_1 is the process time constant also known as residence time, and it indicates the time it takes for the output to reach 63.2% of the steady state response output, i.e. the speed of the response. θ is the process time delay, and indicates the time it takes from the input change until the system starts responding [5].

SIMC-method for tuning PI-controller

When the model is acquired, a tuning method can be chosen for PID-parameter tuning. Skogestad(2003) developed a method based on the Internal model control (IMC) method by Rivera et al.(1986), which is defined as the SIMC-method (Simple-IMC-method) [25, 28]. The method is based on the desired closed loop response being defined as a smooth first order response with time delay,

$$T(s) = \frac{y}{y_{sp}} = \frac{1}{\tau_c s + 1} e^{-\theta s} \tag{2.5}$$

where τ_c is the closed loop time constant, and θ is the effective delay. The method is proven to work well for a wide range of processes, and for both setpoint tracking and disturbance rejection. The method provides a single tuning rule for first or second order plus time delay models, and requires only one tunable parameter [25]. The derivation of the tuning rules can be found in Appendix D, and the resulting tuning parameter rules for a FOPTD-model is as follows,

$$K_c = \frac{1}{k} \frac{\tau_1}{(\tau_c + \theta)} \tag{2.6}$$

$$\tau_I = min(\tau_1, 4(\tau_c + \theta)) \tag{2.7}$$

From this method, the only parameter that needs tuning is τ_c. To get a positive and nonzero controller gain, K_c, τ_c needs to be in the range (θ,∞). Skogestad mentions in his paper that there should be a trade off between fast response and good disturbance rejection, and having a robust and stable controller with small input variations. The first is obtained by small values of τ_c, while the latter is obtained with larger values. A good trade off would in many cases be to have $\tau_c = \theta$, which provides tight control with fast response and good robustness. In cases where there is very short or no delay at all, the choice of τ_c could be significantly more challenging. In his paper, Skogestad (2014) suggests several ways for choosing the value of τ_c in this case. However, it is mentioned that the easiest

way is to simply to choose τ_c based on physical insight, by deciding how fast one wants the system to respond.

2.2.3 Multiloop control: Decentralized PI-control

When there exists multiple inputs and multiple outputs to and from the process, it is defined as a MIMO-system. A control system with several single loop controllers is then defined as a multiloop control system, also called decentralized control. This gives several options for optimal control configuration. In a MIMO system, the process model, $G(s)$, consists of transfer functions from each input to each of the outputs, resulting in a matrix form process model as shown in equation 2.8.

$$\begin{bmatrix} y_1(s) \\ y_2(s) \\ ... \\ y_n(s) \end{bmatrix} = \begin{bmatrix} g_{1,1}(s) & g_{1,2}(s) & ... & g_{1,m} \\ g_{2,1}(s) & g_{2,2}(s) & ... & ... \\ ... & ... & ... & ... \\ g_{n,1}(s) & ... & ... & g_{n,m} \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \\ ... \\ u_m(s) \end{bmatrix} \tag{2.8}$$

In the case where one has non-zero off-diagonal elements in the model matrix, $G(s)$, it means that one or more of the outputs depend on more than one input. The process is thus an interacting coupled process, and the choice of optimal control configuration becomes less obvious.

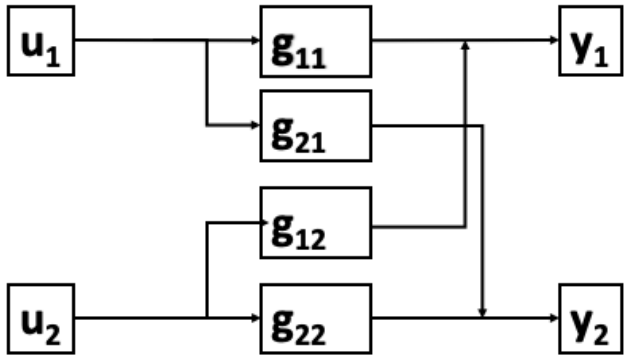


Figure 2.4: A block diagram of a two-way interacting process, in which both of the outputs, y_1 and y_2, depend on both of the inputs, u_1 and u_2. The dynamic relationship between the inputs and outputs are describes by the process transfer functions, $g_{1,1}, g_{1,2}, g_{2,1}$ and $g_{2,2}$.

Consider a two-way interacting process with two inputs and two outputs, as seen in Figure 2.4. $G(s)$ then becomes,

$$G(s) = \begin{bmatrix} g_{1,1} & g_{1,2} \\ g_{2,1} & g_{2,2} \end{bmatrix} \tag{2.9}$$

where all elements are non-zero. Controlling this process with a decentralized controller, means to have two single loop feedback controllers, as shown in Figure 2.1. As there are $n = 2$ controlled variables, and $n = 2$ manipulated variables, one has 2 different ways to control the process. This raises the question of what the optimal input-output pairing is, and thus which multiloop configuration to choose.

A commonly used method for deciding the optimal pairing of a MIMO-process, is to perform a steady state relative-gain-array (RGA) analysis, a method developed by Bristol(1966) [29]. The RGA gives a measure of the interactions in a system, and an indication of the most effective pairing of the variables [30]. For a 2x2 system, the RGA can be computed as,

$$\Lambda(G(s)) = \begin{bmatrix} \lambda_{1,1} & \lambda_{1,2} \\ \lambda_{2,1} & \lambda_{2,2} \end{bmatrix} = \begin{bmatrix} \lambda_{1,1} & 1-\lambda_{1,1} \\ 1-\lambda_{1,1} & \lambda_{1,1} \end{bmatrix} ; \lambda_{1,1} = \frac{1}{1 - \frac{g_{1,2}g_{2,1}}{g_{1,1}g_{2,2}}} \tag{2.10}$$

which gives the following steady state RGA-matrix;

$$\Lambda(G(0)) = \begin{bmatrix} \lambda_{1,1} & 1-\lambda_{1,1} \\ 1-\lambda_{1,1} & \lambda_{1,1} \end{bmatrix} ; \lambda_{1,1} = \frac{1}{1 - \frac{K_{1,2}K_{2,1}}{K_{1,1}K_{2,2}}} \tag{2.11}$$

with $K_{i,j}$ being the steady state process gain [5].

Consider the model shown in Figure 2.4, and assume one wants to use u_1 to control y_1. The relative gain, λ_{11}, is then the ratio between the *open loop gain* between u_1 and y_1, $g_{1,1}(s)$, and the *closed loop gain* showing the effect of u_1 on y_1, $\hat{g}_{1,1}(s)$. For the open loop gain, u_2 is kept constant, and for the closed loop gain, the other loop is closed and y_2 is kept constant (perfectly controlled) [30].

Having $\lambda_{1,1}$ equal to 1, indicates that the gain, $g_{1,1}$, is unaffected by closing the other loop. This results in a first pairing rule;

- *Pair on RGA-elements close to 1* [30].

To avoid instability with integral action in the loop, one should also follow a second pairing rule;

- *Avoid pairing in negative steady state RGA-elements* [30].

Say the optimal pairing was found, and it was decided to use u_1 to control y_1, and u_2 to control y_2. The decentralized diagonal PID-controller is then shown as,

$$C(s) = \begin{bmatrix} C_1(s) & 0 \\ 0 & C_2(s) \end{bmatrix} \tag{2.12}$$

where the controllers will base their input updates only on the error input signal related to their respective CV. Control loop interactions will then be introduced in the presence of a hidden third feedback loop, consisting of the two controllers and the two transfer functions

$g_{1,2}$ and $g_{2,1}$, seen in Figure 2.4. These interactions can cause closed loop instability and difficulties in tuning the controllers [5]. The controllers will "fight" each other, possibly causing an oscillating effect on the outputs.

Tuning a multiloop control system, as the one shown in Figure 2.4, is not as straightforward as for single loop control, and there exists several different approaches. One of the methods is called sequential tuning, and involves choosing an input-output pair in which its controller is tuned and the loop is closed. Then, the other controller is tuned for the second pair, depending on the open loop response of the second loop, while keeping the first loop closed. The performance of the control system is highly dependant on the order in which the loops are tuned and closed. Usually, one chooses to first tune the fastest loop in the system, as this loop is assumed to be less affected by the interactions with a slower loop than the other way around [31, 32].

2.2.4 Multivariable control: Decoupled PI-control

A way of reducing the control loop interactions is to introduce a multivariable control scheme such as *decoupling control*. By adding decouplers to the described 2x2 decentralized control system, one is able to mathematically decouple the system equations by having the decouplers compensate for the unwanted interactions. In this way, ideally, a change in the setpoint $y_{1,sp}$ would only affect output y_1, and equivalently a change in setpoint $y_{2,sp}$ would only affect output y_2.

Inverted decoupling is a decoupling technique where one loop's process input is viewed as a disturbance to the other loop's controller output, and a feed-forward approach is used to compensate for this disturbance [33]. A two-way inverted decoupling structure is shown in Figure 2.5. This method, first introduced by Shinsky(1988), introduces advantages compared to conventional decoupling methods [34]. One such advantage is that the apparent process seen by each controller is the same with the decpouplers implemented, as without. This avoids the need for re-tuning of the current controllers [33]. Other decoupling methods will not be reviewed in this thesis, however, an overview can be found in the book by Seborg et al. [5].

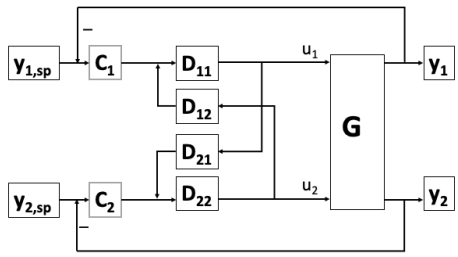


Figure 2.5: Block diagram of a decentralized control scheme with two feedback loops, with inverted decoupling implemented for elimination of interactions.

For this method, the decoupling matrix, $D(s)$, is designed such that $G(s)D(s) = L(s)$, where $L(s)$ is a diagonal transfer matrix representing the new and easier to control plant,

$$L(s) = \begin{bmatrix} g_{1,1} & 0 \\ 0 & g_{2,2} \end{bmatrix} \tag{2.13}$$

i.e. the diagonal of the original process itself [35]. $D(s)$ can be computed by $D(s) = G(s)^{-1}L(s)$, which gives the following equation;

$$D(s) = \begin{bmatrix} d_{1,1} & d_{1,2} \\ d_{2,1} & d_{2,2} \end{bmatrix} = \frac{1}{g_{1,1}g_{2,2} - g_{1,2}g_{2,1}} \begin{bmatrix} g_{1,1}g_{2,2} & -g_{1,2}g_{2,2} \\ -g_{2,1}g_{1,1} & g_{1,1}g_{2,2} \end{bmatrix} \tag{2.14}$$

which leads to the expressions for the input variables, u_1 and u_2, shown as;

$$u_1 = C_1(s)d_{1,1} - C_2(s)d_{1,2} \tag{2.15}$$

$$u_2 = -C_1(s)d_{2,1} + C_2(s)d_{2,2} \tag{2.16}$$

Rearranging 2.15 to find expression for $C_1(s)$ and 2.16 to find expression for $C_2(s)$, and then substituting expression for $C_1(s)$ into equation 2.16 and expression for $C_2(s)$ into equation 2.15, gives the following decoupled system [35];

$$u_1 = C_1(s) - u_2\frac{g_{1,2}}{g_{1,1}} \tag{2.17}$$

$$u_2 = C_2(s) - u_1\frac{g_{2,1}}{g_{2,2}} \tag{2.18}$$

Consequently, the decoupling matrix $D(s)$ can be shown as,

$$D(s) = \begin{bmatrix} 1 & -\frac{g_{1,2}}{g_{1,1}} \\ -\frac{g_{2,1}}{g_{2,2}} & 1 \end{bmatrix} \tag{2.19}$$

Which, when implemented in the control structure, will compensate for the unwanted interactions in the control loops. This method is a case of dynamic decoupling, in which $L(s)$ is a diagonal matrix at all frequencies [30].

In some cases, this may lead to decoupling elements that are not realizable, e.g. if an element relies on future values of its input to determine its output. Mathematically this implies that the element includes a delay term, $e^{-\theta s}$, where $\theta > 0$. However, inverted decoupling can always be reconfigured to achieve realizable decoupling elements, but this may be at the expense of the stability of the decoupler. One may need to resort to other decoupling methods such as steady state decoupling. Steady state decoupling is when $L(0)$ is diagonal, and is obtained by selecting a constant $D(0) = G(0)^{-1}$.

2.3 Model predictive control

A different multivariable control strategy is model predictive control (MPC). MPC is an advanced control technique that is used for control of complicated multivariable problems with inequality constraints on the inputs and/or outputs. The main idea of MPC is to predict the future behaviour of a system, and use a predefined dynamic model of a process, together with process measurements, to solve an online optimization problem. The solution to the optimization is the optimal control actions that drives predicted future output values to their setpoint. The first MPC related technology was developed in the 1970s, and was referred to as dynamic matrix control (DMC). Since then, the techniques have evolved and MPC has become a considerably more attractive technology within industrial practices. In fact, over 4500 applications of MPC were reported worldwide by the end of 1999, and the use has increased since then [5].

The main objectives of MPC is to control the output variables to their setpoints, or to keep them within a specified range, while simultaneously satisfying the set of input and output constraints. This is done in a matter that also prevents large and unnecessary changes in the input variables. The method requires an accurate dynamic model, e.g. a linear empirical model, of the process. It is crucial that this model is a good representation of the actual process, as an inaccurate model will result in inaccurate predictions of the output variables [5]. Advantages of MPC compared to PID controllers, are first and foremost its ability to take future predictions into account when optimizing the current control action. It is then possible to detect early warnings of potential problems. The MPC is also able to consider the process operational constraints when solving the control equations, and it provides system decoupling by capturing the interactions between input and output variables [5].

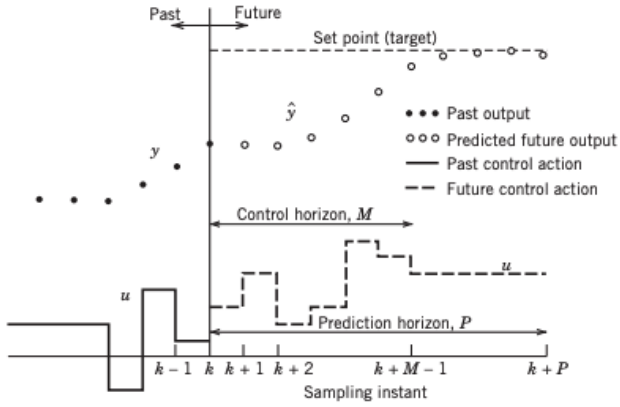


Figure 2.6: The basic concept of MPC for a system with one input and one output. For each sample time k, the controller estimates the current state, and predicts P number of future output values, $\hat{y}$, with M corresponding optimal control actions, u. The current optimal input value, u_k is injected back into the plant.

Figure 2.6 show the basic concept of control of a system with one input and one output [5]. MPC gets the current measurement of the output variables, y, at instant k from the process, and estimates the current state of the plant at this instant. From solving the optimization problem with the current state as the starting point, a predicted output trajectory, $\hat{y}$, is produced with the optimal current and future input values, u. The current optimal input value, u_k is the updated value that is injected back into the process plant.

M denotes the *control horizon*, which is the number of calculated future input values, including the current input. P is the prediction horizon, determining the number of future output predictions, including the current output. P and M are among the tuning parameters of an MPC.

A common choice of a model for an MPC is an empirical linear model. This can be a step response model (time domain), transfer function model or a state space model. State space models are the most commonly used models in research and industry due to their common framework for nonlinear and linear control problems [5].

The control calculations are performed by dynamic optimization to find the input values that move the predicted outputs toward the setpoints in an optimal way. The optimization problem of a linear MPC can be formulated as a quadratic program (QP), involving a quadratic cost function and linear constraints. The constraints defined for an MPC can be hard or soft constraints. Hard constraints can not be violated at any time. However, constraint violations may be unavoidable, and hard constraints will then result in infeasible QP solution. For instance, this may be the case for output variables when large disturbances occur. This can be solved by introducing them as soft constraints, which can be violated, but the violation is penalized and added to the cost function [5]. A typical optimization problem for an MPC can be defined as,

$$\begin{aligned}
\min_{\Delta u_k} J = &\sum_{k=1}^{P}(y_{k+1} - y_{sp,k+1})^T Q(y_{k+1} - y_{sp,k+1}) + \sum_{k=1}^{P-1} \Delta u_k^T R \Delta u_k + \rho \sum_{k=1}^{P} \epsilon_k^2 \\
&\text{Subject to} \\
&x_{k+1} = Ax_k + Bu_k \\
&y_{k+1} = Cx_{k+1} \\
&x_0 = \hat{x} \\
&y_{i,min} - \epsilon_k \leq y_{i,k+1} \leq y_{i,max} + \epsilon_k \\
&u_{j,min} - \epsilon_k \leq u_{j,k} \leq u_{j,max} + \epsilon_k \\
&\Delta u_{j,min} - \epsilon_k \leq \Delta u_{j,k} \leq \Delta u_{j,max} + \epsilon_k
\end{aligned} \tag{2.20}$$

where,

- $\Delta u = u_k - u_{k-1}$
- Q - weight matrix, determines the the importance of each output variable.

- R - weight matrix, determines the the importance of each control move variable.
- ρ - penalty weight, determining the penalty term relative to the other cost function terms.
- ϵ - slack variable, quantifying the worst-case constraint violation.
- x_k - state at sample time k.
- $y_{i,min}, y_{i,max}$ - lower and upper bounds of i^{th} output variable.
- $u_{i,min}, u_{i,max}$ - lower and upper bounds of i^{th} input variable.
- $\Delta u_{j,min}, \Delta u_{j,max}$ - lower and upper bounds of j^{th} control move variable.

The optimization problem uses a state space representation of the linearized prediction model. The initial state, x_0, is estimated typically by using a Kalman filter [36].

2.3.1 Guidelines for tuning control parameters

The tuning parameters of an MPC are the prediction horizon, P, the control horizon, M, the two weighting matrices, Q and R, and lastly the penalty weight, ρ, and the slack variable ϵ. From the book by Seborg(2017), the optimal choice of tuning parameter values can summarized as, [5]

- Select M such that $N/3 < M < N/2$, where N is the "model horizon". N should be chosen so that $N\Delta t = t_s$, where t_s is the settling time for the open loop response.
- Select P such that $P = M + N$, so that the full effect of the last MV move is taken into account.
- The values of Q and R are selected based on how one wants to prioritize the control and regularization of the different output and input variables, respectively.
- Choose higher ϵ values when large violations are allowed, and large ρ value to decrease violations. Trial and error approach are necessary to find optimal value.

There are several other approaches to tune an MPC. For instance, one systematic approach for tuning the MPC parameters for MIMO systems is presented by Shah and Engell (2011) [37]. However, the approach presented above, will be used in this work.

CHAPTER 3

Experimental overview

This chapter presents an overview of the equipment and chemicals used in the experiments performed in this thesis, as well as the main procedures for preparing, running and cleaning the experimental equipment.

3.1 Chemicals

One 500 mL stock solution of dye in deionized water, later referred to as MQ water (purified by using an ion exchange cartridge), was prepared and divided into five 100mL bottles that lasted through the total period of experimental work. The solution was a 10000 ppm malachite green solution [38], diluted to a 100ppm solution. As the malachite green solution undergoes degradation reactions when exposed to light over longer periods of time, the bottles containing the solution were made out of amber-stained glass and were covered with tin foil when not in use [39].

3.2 Lab setup

Figure 3.1 show the flow sheet of the microreactor system including the controller, with data from the spectrophotometer being sent to the computer which determines the pump commands.

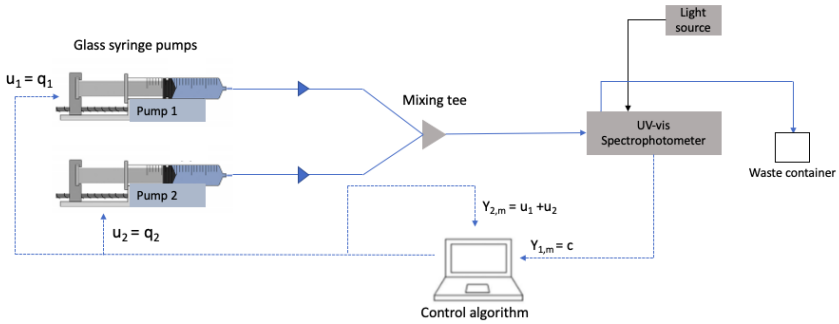


Figure 3.1: Process flow sheet of the experimental system.

The microreactor used for the experiments done in this work consists of two Mid Pressure neMESYS 1000N syringe pumps from Cetoni [40], with tubes in the micro-scale connected to the outlet of each pump. Both syringes are of glass material, and the microtubes are made of Perfluoroalkoxy alkane polymers (PFA). The tubes were connected in a Y-shaped mixing tee where the content was mixed before sent through a Qmix Lambda spectrophotometer, equipped with a UV detector allowing measurement between 190 and 650 nm, also from Cetoni [40]. A pulsed xenon light source (PX-2, Ocean Insight) was connected to the spectrophotometer via an optic fiber cable. The light source was controlled via an arbitrary waveform generator (RSDG 805, RS PRO).

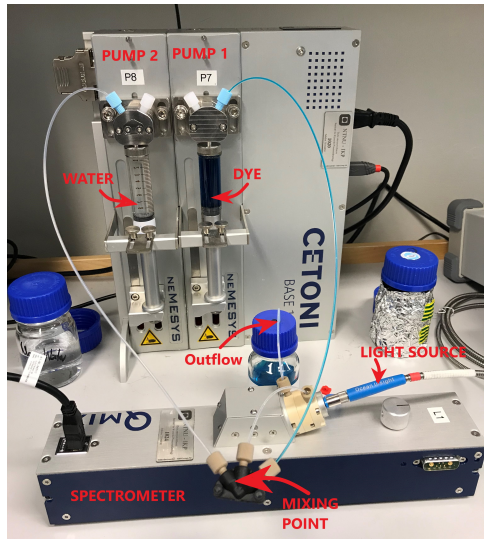


Figure 3.2: Experimental setup in the lab. Pump 1 contains the dye solution, pump 2 contains water, and the two flows are mixed at the mixing point, shown in the picture. The mixture is sent through the UV/vis-spectrometer, where the absorbance of light from the light source is measured.

Figure 3.2 show the lab setup of the microreactor. Pump 1 can be seen to the right in Figure 3.2, and contains the solution of dye. Pump 2, to the left in the same figure, contains pure MQ water. The pumps are mechanical and are connected to a computer. The mixture runs through the spectrophotometer, which sends light through the solution at 20Hz and with an integration time of 500ms. The integration time is defined as the time before data is sent to the analog to digital converter (A/D converter). This results in a data transfer speed of 2 spectra per second (1/integration time), with 10 cycles per integration time. The A/D converter sends real-time data to the computer which processes the data during experiments, as shown in Figure 3.1.

3.2.1 Software setup for lab equipment

For managing the pump system, Cetoni's Qmix SDK development package with integration to Python interface was used, in which the pumps were run by Python commands [41]. The Ocean Direct API for Python provided the integration with the spectrophotometer, and the pumps and spectrometer could thus be managed simultaneously in the same Python script [42].

For the MPC and PID-control shceme, Matlab and Simulink were used to represent the controllers and computing the input updates. A TCP/IP socket was used for sending information back and fourth between Matlab and Python, over the local network [43]. This made it possible to get experimental measurements from the spectrophotometer through Python, calculate new input values based on the measurements in Matlab, and finally updating the pump commands with the new input values in Python.

Python was run with the development environment Spyder, and version specification of all relevant programs and packages are given in Table 3.1.

Table 3.1: Specified versions of programs and packages used.

Spyder	4.0.1
Python	3.6
Matlab	R2020a
Simulink	R2020a
OceanDirect	1.12
QmixSDK	20200902

The CPU specifications for the computer used for implementation of the controllers: Intel(R) Core(TM) i5-8400 CPU with 8GB RAM.

3.3 Experimental Procedure

The syringes were mounted to the pumps in their initial position, and then filled with their respective liquids. The tubes were cleaned both before and after each experiment by run-

ning clean water through the system for several minutes. This was done to avoid any dye being left inside the spectrophotometer, contaminating the measurements. At one point the tube through the spectrophotometer was cleaned with pure isopropanol (IPA) for removal of any contaminants accumulated over time. It was also important to make sure that there were no air bubbles within the system, as this would cause large errors in the measurement values.

Before starting any experimental tests, the reference intensity of MQ water was taken for the absorbance and concentration calculations. To make sure the reference measurements were consistent and accurate throughout all the experiments, background tests were performed before and after each experiment. Figure A.1 in Appendix A shows a plot of the intensities versus wavelengths for all performed background tests. This result shows that the reference values used for the calculation of the concentrations were all in the same range, which shows consistency throughout the experiments. Some further comments regarding these results are given in Appendix A.

There were some deviations from the expected concentration values when calculating the steady state concentrations during experiments. These deviations would vary from day to day, and were assumed to be caused by contamination in the spectrophotometer, despite cleaning the tubes with water, or instrumental limitations as briefly described in Section 2.1.1. Another reason for the varying deviations could be the small differences in background measurements, as explained in Appendix A. To account for this deviation, a number of reference measurements of the concentration were made before each experiment. The concentration values obtained throughout the experiments were then shifted with the difference between the mean of the reference measurements and the expected steady state value, before they were sent to the control-model.

The expected values of the concentration of dye in the outflow, were calculated from the steady state component balance of malachite green solution, and gave the following equation for the steady state concentration;

$$C = \frac{q_1 C_0}{q} \tag{3.1}$$

Where C is the concentration of dye in the outflow, C_0 is the initial concentration of the Malachite green solution, q_1 is the flow of dye solution, and q is the total flow.

The flow was not measured during experiments due to the difficulties of integrating the flowmeter in Python, and as the regulation of the concentration was the main priority of this work. However, the flow rate of the syringe pumps was measured once prior to the experiments to make sure that the values provided to the pumps were in fact the same as the flow measured by the flowmeter. Instead of using a flowmeter during experiments, it was assumed that the pumps were regulated perfectly and reacted instantaneously to Python commands, and that there were no accumulation of volume in the system, i.e. $q = q_1 + q_2$.

The different experiments were run for varying amounts of time, with a period of 10 minutes for the longest experiment. As both syringes had a volume of 10 mL, they

could run for approximately 50 minutes with a flow of 200 $\mu L/min$. After each day the equipment was used, the syringes were disconnected and cleaned with pure IPA and MQ water.

CHAPTER 4

Preparational work

This chapter provides an overview of the preparational work done prior to starting the experiments with the controllers. Section 4.1 presents the method for developing a calibration curve for the online concentration measurements. The approach and result of developing an approximate model of the process is described in Section 4.2.

4.1 Calibration curve for concentration measurement

As mentioned in Section 2.1.1, the concentration of a component in a fluid can be found by a pre-calculated calibration curve, given by the linear relationship of Beer Lamberts law. A calibration curve was therefore made prior to any experiments for calculation of unknown concentrations. This was done by running flow with different known concentrations through the system, measuring the intensity and calculating the absorbance corresponding to each concentration.

Figure 4.1 shows the absorption spectra of the Malachite green solution mixed with water to 25 different concentrations between 2 ppm and 50 ppm, for wavelengths between 187 nm and 667 nm. For each wavelength, the R^2-value was calculated from linear regression between all concentrations, to determine which wavelengths gave the most accurate linear fit. The R^2-value was calculated by Equation 4.1, giving a value between 0 and 1 where highest value indicates best linear fit.

$$R^2 = \frac{SS_{regression}}{SStotal} \tag{4.1}$$

Here, $SS_{regression}$ stands for the sum of squares due to regression, while SS_{total} is the total sum of squares. It was also decided to use one of the two peaks in the spectra, both due to this giving minimal deviation from actual value, as explained in Section 2.1.1,

and the fact that these peaks are characteristic for molecules containing aromatic groups, such as the Malachite green dye.

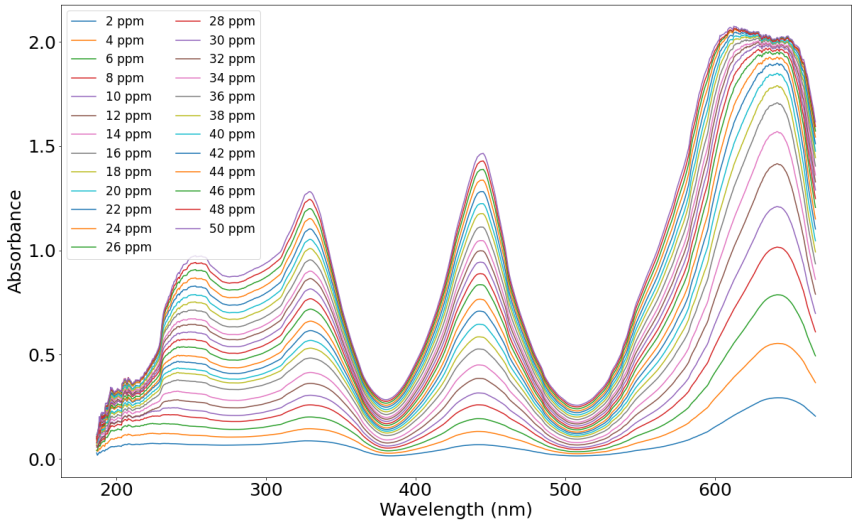


Figure 4.1: The absorption spectra for 50 sample solutions with known concentrations between 2 ppm and 50 ppm. The bottom blue line show the 2 ppm absorption spectra, while the top purple line show the 50 ppm absorption spectra.

Figure 4.2 show the resulting linear regression between the 25 concentration points for the chosen wavelength of 446 nm, which could then be used as a calibration curve. The curve was tested against other known concentrations to see if the curve in fact gave an accurate estimate of the true concentration. Figure 4.3 show the result of this test, and the majority of the points lies within the 95% confidence limits. Tests were done for three different total flows, namely 100, 200 and 400 $\mu L/min$, while the calibration curve was made with 200 $\mu L/min$ total flow. A calibration curve was also made and tested for the wavelength of 328 nm, which gave similar results. However, due to promising results from testing and a high R^2-value, the regression line for the 446 nm wavelength was chosen as the calibration curve for further concentration calculations. The regression line is given by Equation 4.2.

$$A = 0.0295C + 0.0341 \tag{4.2}$$

The test results for the three different total flows, as shown in Figure 4.3, demonstrates that the flow rate has some influence on the calculation of the concentrations. Higher flow rates seem to give more accurate measurements, which could be explained by the better precision of the pumps at the higher ranges of flow rates. Conversely, at lower flow rates, even small deviations in any of the settings of the pump could lead to larger errors in the final concentration. From Figure 4.3, it can be seen that the points deviate slightly more from the calibration curve with higher concentrations. This is in accordance with the de-

viation in Beer-Lambert's law, in which the law only states a linear relationship for low concentrations, as explained in Section 2.1.1.

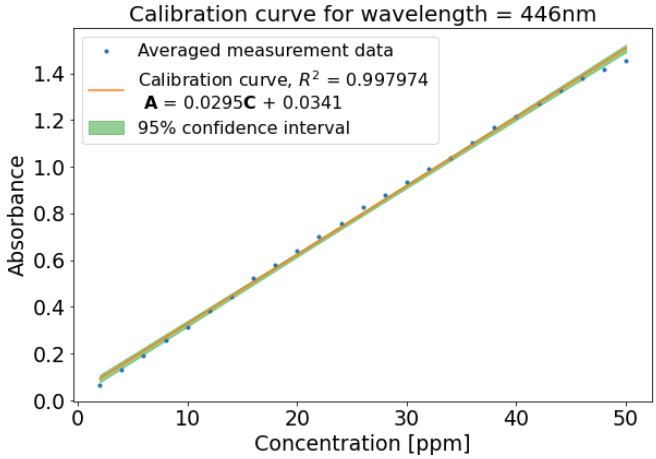


Figure 4.2: Linear relationship between the concentration of the dye solution and the measured absorbance. The orange line show the linear regression for the 25 concentrations and the corresponding absorbance values for the wavelength of 446 nm. The blue dots represent the averaged measurement data at this wavelength, and the 95% confidence interval is shown in green. The resulting linear equation used for concentration calculations was $A = 0.0295C + 0.0341$.

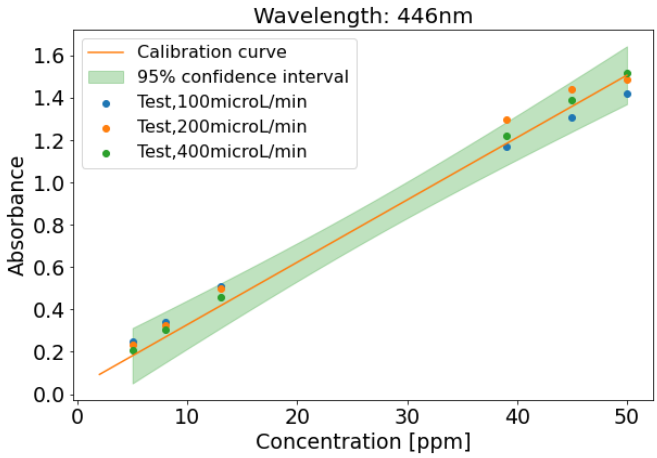


Figure 4.3: The result of testing the concentration measurement of three different flow rates, 100, 200 and 400 $\mu L/min$.

Overall, these results shows good promise in terms of sufficient concentration mea-

surements for further experimental work. Although higher flowrates seemingly give more accurate measurements, the operational total flow should not be too high due to the finite pumping volumes of the syringes. This so that experiments can be done over a sufficient time period. Therefore, a total flow of 200 $\mu L/min$ was chosen as the initial operational flow for all further experiments.

4.2 Development of the estimated process model

For the development of a control structure, a well defined approximation of the process model is necessary for optimal controller tuning in PID-control, and for the optimization problem in MPC. As described in Section 2.2.2, one way to approximate a dynamic model of a process is to analyze the output response to a unit step in the input, and then find the parameters of the transfer function model graphically. The obtained model is then a so called step response model.

The process studied in this work can be defined as a 2x2 MIMO system, as the one illustrated in Figure 2.4. To estimate a dynamic model for this case, a step response of both outputs to a change in each of the inputs was done. From this, a model was approximated graphically of the form FOPTD in the Laplace domain.

Three different steady state conditions were tested to see if there were any significant differences for the three operating conditions. The steady state values of the two inputs and two outputs are listed in Table 4.1. For each steady state condition, a 20% step in each of the two inputs was made, with the other input kept constant. While the experimental results of Case 3 can be seen in Figure 4.4, results from Case 1 and 2 can be seen in Figures C.1 and C.2 in Appendix C.

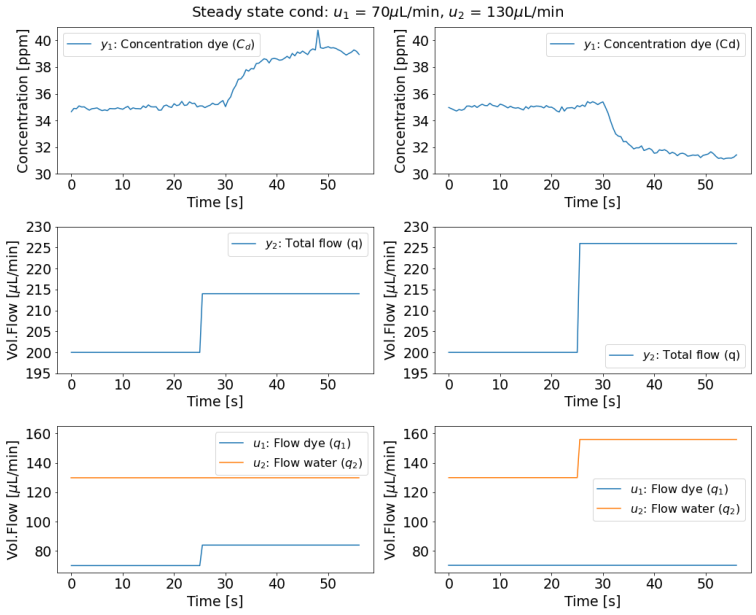


Figure 4.4: Step response with operating conditions of Case 3. A 20% step in the input values u_1 and u_2 can be seen in the bottom left and right plots, respectively, and the corresponding responses of each of the output variables, y_1 and y_2, can be seen in the top four plots.

Table 4.1: Steady state nominal operating conditions for Case 1, 2 and 3. The values show the initial steady state values of the two inputs and the two outputs of the system.

	$q_1(u_1)[\frac{\mu L}{min}]$	$q_2(u_2)[\frac{\mu L}{min}]$	$C_d(y_1)[ppm]$	$q(y_2)[\frac{\mu L}{min}]$
Case 1	30	170	15	200
Case 2	50	150	25	200
Case 3	70	130	35	200

As this is a 2x2 system with two-way interactions, as described in section 2.2.3, four transfer functions were needed to describe the dynamics;

- $g_{1,1}$ - Transfer function from u_1 to y_1
- $g_{1,2}$ - Transfer function from u_2 to y_1
- $g_{2,1}$ - Transfer function from u_1 to y_2
- $g_{2,2}$ - Transfer function from u_2 to y_2

The system could thus be defined in the following matrix form,

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} g_{1,1} & g_{1,2} \\ g_{2,1} & g_{2,2} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{4.3}$$

Where the response of y_1 can be approximated as a first order response, and the response of y_2 is simply a pure gain response equal to the input step value. This results in transfer functions $g_{1,1}$ and $g_{1,2}$ of the form FOPTD;

$$g = \frac{k}{(\tau_1 s + 1)} e^{-\theta s} \tag{4.4}$$

The graphical method for finding the process parameters k, τ_1 and θ was explained in Section 2.2.2, and is shown for the current model by the example in Figure 4.5 and Equation 4.5. The example show the step response of Case 3, shown in Figure 4.4. This gives the parameters for the open loop transfer function from u_1 to y_1, g_{11}, as $y_1 = g_{11}u_1$ in this case.

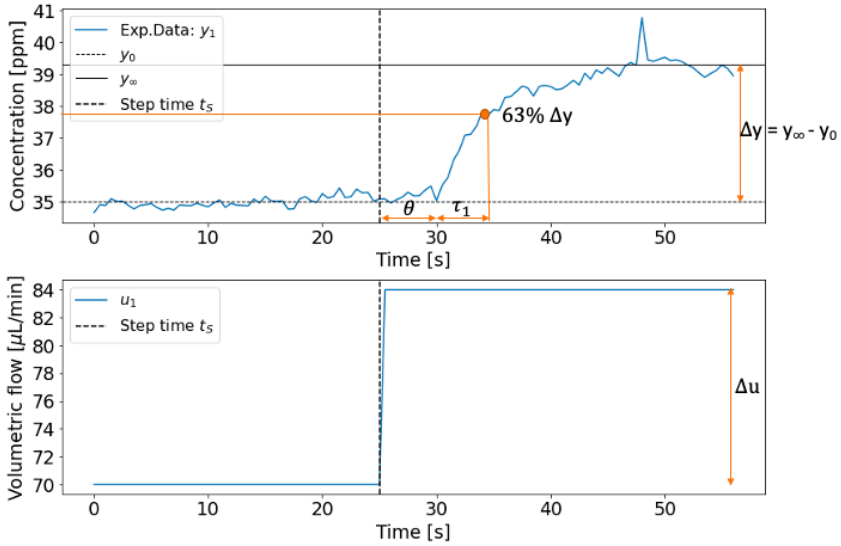


Figure 4.5: Graphical method for finding the parameters of the first order plus time delay transfer function model, $g_{1,1}$, for Case 3.

$$
\begin{aligned}
k &= \frac{\Delta y}{\Delta u} \\
\tau_1 &= t(63\%\Delta y) \text{ - } t(\text{initial response of } y) \\
\theta &= t(\text{initial response of } y) \text{ - } t(\text{step time})
\end{aligned}
\tag{4.5}
$$

As seen in Figure 4.4 and Figures C.1 and C.2 in Appendix C, the step response of the total flow, $y2$, is equal to the step in each of the inputs, $u1$ and $u2$. This follows from the assumptions stated in Section 3.3, and that the total flow is not actually measured, but simply taken as the sum of the two inputs. A change of flow in each of the inputs, can therefore be seen immediately in the output flow, with no effective delay. This results in transfer functions $g_{2,1}$ and $g_{2,2}$ being equal to 1 for all cases.

Table 4.2 show the resulting transfer functions for all input-output pairs from the graphical method performed on all three steady state cases.

The approximated model's were simulated with Simulink in Matlab, with the same input step performed for all cases. The Simulink model for this step response can be seen in Figure F.1 in Appendix F. To solve the problem of algebraic loops that occurred in Matlab when implementing the pure gain transfer functions $g_{2,1}$ and $g_{2,2}$, they were implemented as first order transfer functions with a very small time constant of $\tau_1 = 0.001$. The model responses were compared to the process responses shown above, and the results can be seen in Figure 4.6 below, and Figures C.3 and C.4 in Appendix C.

Table 4.2: Resulting transfer functions for Case 1, 2 and 3, from a step response test using a graphical method to find parameter values.

	$g_{1,1}$	$g_{1,2}$	$g_{2,1}$	$g_{2,2}$
Case 1	$\frac{0.438}{(8s+1)}e^{-5s}$	$\frac{-0.0616}{(4s+1)}e^{-5s}$	1	1
Case 2	$\frac{0.339}{(5.7s+1)}e^{-5s}$	$\frac{-0.113}{(5.5s+1)}e^{-5s}$	1	1
Case 3	$\frac{0.307}{(4.66s+1)}e^{-5s}$	$\frac{-0.142}{(3.7s+1)}e^{-5s}$	1	1

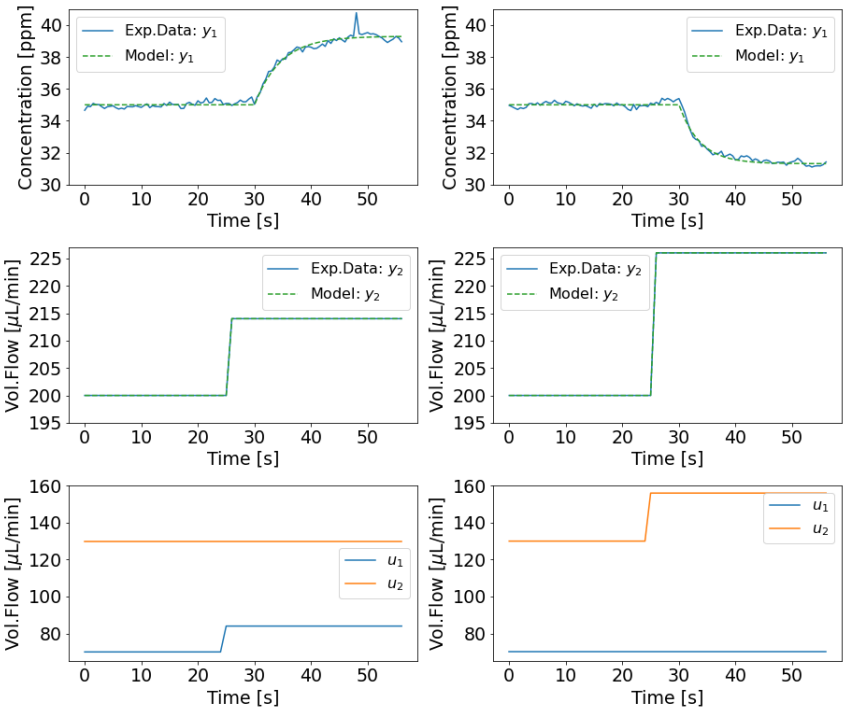


Figure 4.6: Experimental and model data of step response for Case 3. The plot show a 20% step in the input values u_1 and u_2 to the bottom left and right, respectively, and the corresponding responses of each of the output variables, y_1 and y_2. The simulated model response is shown as the green stippled line.

The transfer functions in the three different cases had the same delay time, but different gains and time constants. This shows that a controller tuned by a model developed in this way, should possibly be limited to a predefined operational range. The reason for the different dynamics in each case can be explained by the increase in total flowrate. A

20% increase in u_1 for Case 3 gives a higher total flowrate than for Case 1, and increased flow gives a faster response of the system. However, the model transfer functions show quite similar parameters, so the effect of changing operating conditions such as setpoint changes, might not affect the quality of the controller to a very large extent.

To maintain the linear relationship of the calibration curve, as explained in Section 2.1.1, as well as giving room for increase and decrease of the concentration setpoint, it was decided to run all experiments with the steady state nominal concentration value of 35 ppm, i.e. Case 3. Since the steady state concentrations through all experiments then would be in the range (25 ppm, 45 ppm), a more general transfer function based on the parameters of case 2 and 3 was defined as,

$$G(s) = \begin{bmatrix} \frac{0.35}{5s+1}e^{-5s} & \frac{-0.1}{5s+1}e^{-5s} \\ 1 & 1 \end{bmatrix} \tag{4.6}$$

Where both the parameters τ_1 and θ are equal for g_{11} and g_{12}. This was done to simplify the equations, and generalize for a larger concentration range. The model response of the general model was compared to the experimental response of Case 2 and 3, and the results are shown in Figures C.5 and C.6 in Appendix C. These figures show that the model worked reasonably well for both cases, although slightly better for Case 2, and could be seen as a good generalization of the model.

CHAPTER 5

Method, results and preliminary discussion for control structures

This chapter presents the methods for developing and tuning the three different controllers evaluated in this thesis. These are a decentralized PI-controller, a PI-controller with decoupling, and a model predictive controller. The description of the development of each controller is followed by the results and discussions on their performance and implementation.

The nominal steady state values of the process variables given in table 5.1, were applied in the experimental testing of all controllers presented in this chapter.

Table 5.1: Nominal steady state values of input and outputs of the system, used for all experiments.

Process variable	Nominal steady state value
y_1	35
y_2	200
u_1	170
u_2	130

The Matlab and Python codes for all three controllers are given in Appendix G, and show the code for running the experiments with the lab setup, as well as the model testing.

5.1 PI-controller for concentration and flow control

In the following section, the development of a decentralized PI-controller consisting of two single loop controllers, is described. The results and discussion of the model simulations are given in Section 5.1.2, and the experimental results and discussion are presented in Section 5.1.3.

5.1.1 Development of a decentralized PI-controller

The generalized open loop transfer function model, developed from the step response tests in Section 4.2, was used to tune the PI-controller. The model is shown as,

$$G(s) = \begin{bmatrix} g_{1,1} & g_{1,2} \\ g_{2,1} & g_{2,2} \end{bmatrix} = \begin{bmatrix} \frac{0.35}{5s+1} & \frac{-0.1}{5s+1} \\ 1 & 1 \end{bmatrix} \tag{5.1}$$

Since only one input is used to control each of the outputs, a steady state relative-gain-array (RGA)-analysis was performed to find the optimal input-output pairing, as described in Section 2.2.3. For this 2x2 system, the steady state RGA could be computed from Equation 2.11 as,

$$\Lambda(G(0)) = \begin{bmatrix} 0.78 & 0.22 \\ 0.22 & 0.78 \end{bmatrix} \; ; \lambda_{11} = 0.78 \tag{5.2}$$

From the rules presented in Section 2.2.3, the best input-output pairing seemed to be controlling y_1 with u_1 and y_2 with u_2. For tuning this multiloop control system, the sequential tuning method was chosen, as briefly described in Section 2.2.3. Controller $C_1(s)$, controlling y_1, was therefore tuned based on the open loop transfer function from u_1 to y_1, $g_{1,1}$. After closing this loop, an open loop step response for y_2 was performed by doing a step in u_2 with the first loop still closed, as shown in Figure F.2 in Appendix F. This gave a different transfer function than $g_{2,2}$, thus including the dynamics coming from the interactions. The resulting transfer function is,

$$g_{2_{nd}loop} = \frac{1.284}{5s+1} \tag{5.3}$$

which is found graphically from the step response shown in figure C.7 in Appendix C. The model response of the resulting transfer function, Equation 5.3, is shown in the same plot. As can be seen in this figure, the time constant, τ_1, is actually equal to 0, according to this method. However, there is clearly some settling time before the system reaches steady state, and the response was therefore approximated as a first order system without time delay, with $\tau_1 = 5$. The choice of which loop to close first in the sequential tuning approach for this controller, is purely based on the lack of dynamics in $g_{2,2}$. Therefore, the loop containing $C_1(s)$ was chosen as the first loop in order to obtain a PI-controller tuned with the SIMC-rules.

The resulting tuning parameters for the two controllers, found by the SIMC rules presented in Section 2.2.2, are given in Table 5.2. The parameters for controller 1 is based on

the choice of tight control, with $\tau_c = \theta$. The parameters of controller 2 are selected based on different values of τ_c, to see the effect of having a relatively slow or fast second controller. The effects of the different values were tested during experiments, and are shown in the following section.

Table 5.2: Tuning parameters of PI-controllers $C_1(s)$ and $C_2(s)$. Three values of τ_c were tested for controller $C_2(s)$, and the resulting parameter tunings are shown as $C_2^1(s)$, $C_2^2(s)$ and $C_2^3(s)$.

	$C_1(s)$	$C_2^1(s)$	$C_2^2(s)$	$C_2^3(s)$
K_c	1.4285[μL/min ppm]	0.3894 [-]	0.5563 [-]	0.7788 [-]
τ_I $[s]$	5	5	5	5
τ_c $[s]$	5	10	5	2

The Simulink model for PI-control of the total flow and the concentration is shown in Figure F.3 in Appendix F. The model was tested by performing steps in the setpoint value of each of the outputs, $y_{1,sp}$ and $y_{2,sp}$, while keeping the other output setpoint constant. The first step was made by a 20% increase from the nominal steady state value at time t = 50 sec, followed by a 2x20% decrease in the nominal value at time t = 250 sec. A last step in the setpoint was made by a 20% increase, back to the nominal value, at time t= 450 sec. The sample time was 1 sec, and the model runtime was 600 sec. The experimental testing was performed in the same way, and the results are showed in the next section.

5.1.2 Simulation results: Decentralized PI-control

The following two figures, Figures 5.1 and 5.2, show the model simulation result of tracking the setpoints of y_1 and y_2 when changing the setpoints, $y_{1,sp}$ and $y_{2,sp}$, respectively.

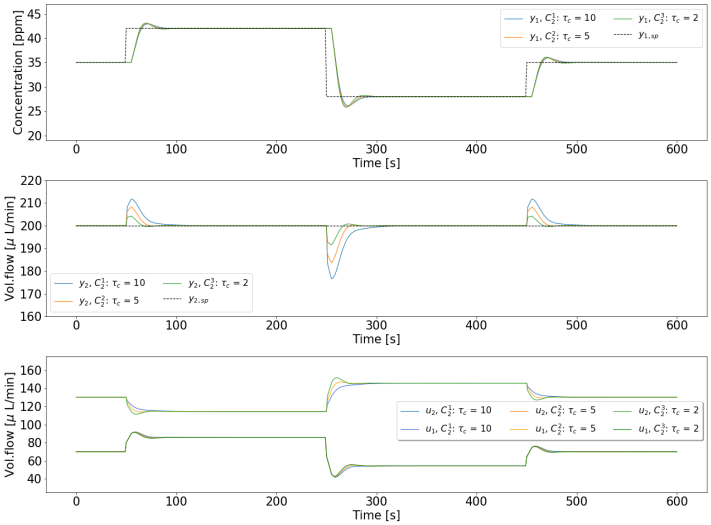


Figure 5.1: Model simulation of controlling the concentration and flowrate of the outflow of the system, when changing the concentration setpoint $y_{1,sp}$. The bottom graph show the control moves of the two MV's, u_1 and u_2. The results of the different tuning values of τ_c for controller $C_2(s)$ are shown in different colours, blue, orange and green.

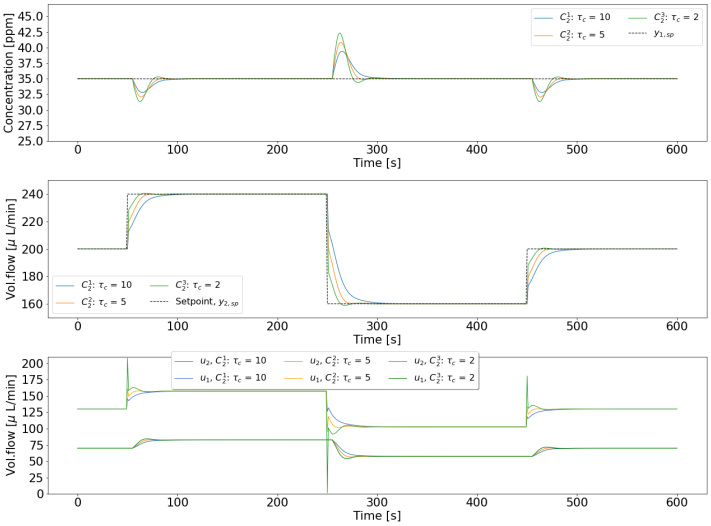


Figure 5.2: Model simulation of controlling the concentration and flowrate of the outflow of the system, when changing the total flowrate setpoint $y_{2,sp}$. The bottom graph show the control moves of the two MV's, u_1 and u_2. The results of the different tuning values of τ_c for controller $C_2(s)$ are shown in different colours, blue, orange and green.

The simulation results, shown in Figure 5.1 and 5.2, show a reasonably good performance of both controllers in terms of setpoint tracking. Here, it would seem like both controllers benefit from having a lower value of τ_c for $C_2(s)$, as this results in lower settling time for the response to the setpoint change, and faster correction of the deviation caused by this change in the other output variable.

The common order in which to tune the loops in sequential tuning, is to close the fastest loop first. To determine which of the undesigned loops are required to be fast, a common procedure is to perform a bandwidth estimation to find the required bandwidth of the individual loops [31]. However, as $g_{2,2}$ is equal to one, due to the assumption of perfectly regulated input flows, it was decided to close loop one first. This because it simplified the process of tuning $C_1(s)$ using the SIMC-rules. Shiu et al. (1998) recommends in their paper to repeat the tuning for two iterations for a two-loop subsystem [32]. In this case only one iteration of sequential tuning was performed, as the model simulation showed sufficient results in terms of stable control of the process, as seen in both Figure 5.1 and 5.2.

5.1.3 Experimental results: Decentralized PI-control

Figure 5.3 and Figure 5.4 show the result of testing the controllers experimentally on the actual process, with the same setpoint tracking as for the model simulation.

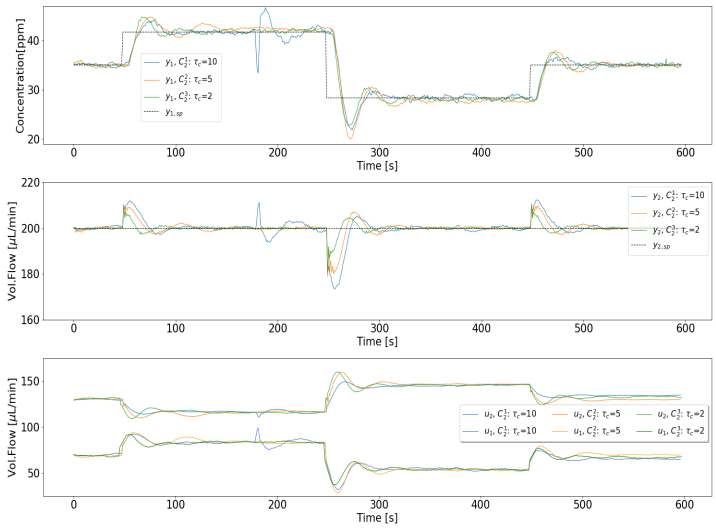


Figure 5.3: Experimental result of controlling the concentration and flowrate of the outflow of the system, when changing the concentration setpoint $y_{1,sp}$. The bottom graph show the control moves of the two MV's, u_1 and u_2. The results of the different tuning values of τ_c for controller $C_2(s)$ are shown in different colours, blue, orange and green.

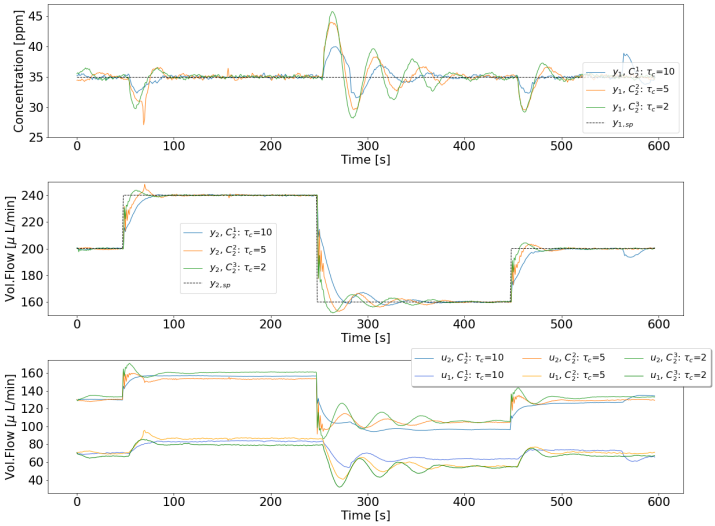


Figure 5.4: Experimental result of controlling the concentration and flowrate of the outflow of the system, when changing the total flowrate setpoint $y_{2,sp}$. The bottom graph show the control moves of the two MV's, u_1 and u_2. The results of the different tuning values of τ_c for controller $C_2(s)$ are shown in different colours, blue, orange and green.

The controlled responses of the process to the setpoint changes, shown in Figures 5.3 and 5.4, are very similar to the model simulations, shown in the previous section. Apart from some disturbances, and the increased oscillations in the two outputs for the second step of $y_{2,sp}$, in Figure 5.4, the responses show the same trend. This could then be an indication that the model captures a large part of the dynamics in the system. However, the experimental result showed oscillations, especially when tracking the setpoint change of y_2, as shown in Figure 5.4. This is a clear indication of interactions in the system. It is then reasonable to assume that the input flows are in fact not perfectly regulated, and that the system incorporates dynamics that are not accounted for in the model. One way of decreasing the interactions could be to tune the second controller based on the actual process response instead of the model response, as done in this case. Then one would perhaps have seen the need for several iterations of the sequential tuning, which could have improved the controller performance.

In the results shown in this and the previous section, three different values of τ_c for controller $C_2(s)$ were tested. It is clear for both the simulation result and the result of controlling the actual process, that lower values of τ_c gives a faster response when controlling the flow, and that this affects the control of the concentration. When changing the setpoint value of y_1, seen in Figure 5.3, it seems that having $\tau_c = 2$ provides tighter and better control of both outputs. However, when changing the setpoint of y_2, as in Figure 5.4, the value of $\tau_c = 2$ results in an increase of the oscillations and longer settling time. This can especially be seen for the concentration output. As mentioned, the sequential tuning method benefits from closing the faster loop first, as this is less affected by the interactions

with the slower loop. This might be the reason for the increased oscillations, as the first loop no longer is the fastest. To avoid the conflict between the two controllers and reduce the oscillations, it would therefore seem reasonable to have a slower flow controller and thus a higher τ_c.

5.2 PI-controller with two-way decoupling for concentration and flow control

This section presents the development and implementation of a decoupler-block to the control structure presented in Section 5.1.1, in order to eliminate the interactions and thereby improve the controller performance. The model results and discussion are presented in Section 5.2.2, followed by the experimental results and discussion in Section 5.2.3.

5.2.1 Development of a PI-controller with two-way decoupling

Two decoupler blocks were implemented in the PI-control structure presented in section 5.1.1, as shown in figure F.4 in Appendix F. Following the procedure presented in section 2.2.4, the resulting decpoupling matrix was found for the current control system;

$$D(s) = \begin{bmatrix} 1 & -\frac{g_{1,2}}{g_{1,1}} \\ -\frac{g_{2,1}}{g_{2,2}} & 1 \end{bmatrix} = \begin{bmatrix} 1 & -\frac{\frac{-0.1}{5s+1}}{\frac{0.35}{5s+1}} \\ -\frac{1}{1} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.2857 \\ -1 & 1 \end{bmatrix} \tag{5.4}$$

By using the method of inverted decoupling, as explained in Section 2.2.4, the original tuning of the controllers presented in Section 5.1.1 could be retained.

The new decoupled system was tested by the same procedure as for the original control structure. For both $y_{1,sp}$ and $y_{2,sp}$, three step changes were made, while keeping the other setpoint constant, to see the closed loop responses in y_1 and y_2. For both cases, the first step was done at time $t = 50$ sec with an increase of 20% from nominal value. The second step was done at time $t = 250$ sec with a decrease of 2x20%, and the third step at time $t = 450$ sec, with an increase of 20% back to nominal steady state value. The sample time was 1 sec, with a total runtime of 600 sec. The results of these tests are shown in the following section.

5.2.2 Simulation results: Decoupled PI-control

Figures 5.5 and 5.6 show the result of the model simulation of controlling y_1 and y_2 to their setpoints, while changing the concentration setpoint, $y_{1,sp}$, and the total flow setpoint, $y_{2,sp}$, respectively.

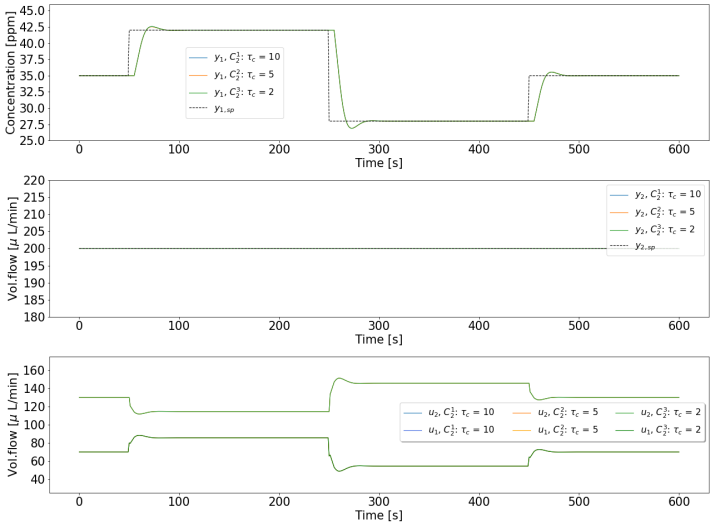


Figure 5.5: Simulation of the controlled model when changing the concentration setpoint, $y_{1,sp}$. The setpoint for the total flow, $y_{2,sp}$, is kept constant. The bottom graph show the control moves of the two MV's, u_1 and u_2. The results of the different tuning values of τ_c for controller $C_2(s)$ are shown in different colours, blue, orange and green. In this plot, all values of τ_c gave the same result.

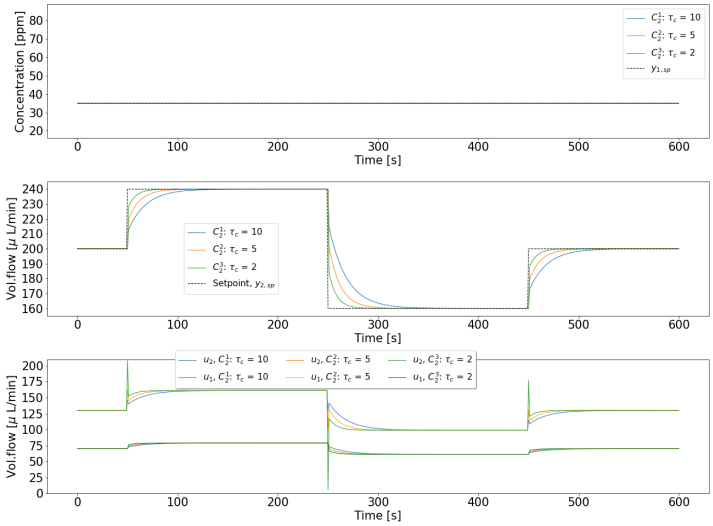


Figure 5.6: Simulation of the controlled model when changing the setpoint for the total flowrate, $y_{2,sp}$. The concentration setpoint, $y_{1,sp}$, is kept constant. The bottom graph show the control moves of the two MV's, u_1 and u_2. The results of the different tuning values of τ_c for controller $C_2(s)$ are shown in different colours, blue, orange and green.

The results from the model simulations of the PI-controller with a decoupling scheme, shown in Figure 5.5 and 5.6, showed that both the flow and and the concentration is controlled nearly perfectly, with all interactions eliminated. This also makes the first controller, $C_1(s)$, unaffected by the different values of τ_c for $C_2(s)$. Overall, the model results of the PI-controller with decoupling show a large improvement from the decentralized PI-controller presented in Section 5.1. The interactions are eliminated, and the controllers do not oppose each other.

5.2.3 Experimental results: Decoupled PI-control

Figures 5.7 and 5.8 show the experimental result of controlling the actual process, with the same setpoint changes as for the model, shown in the two previous figures.

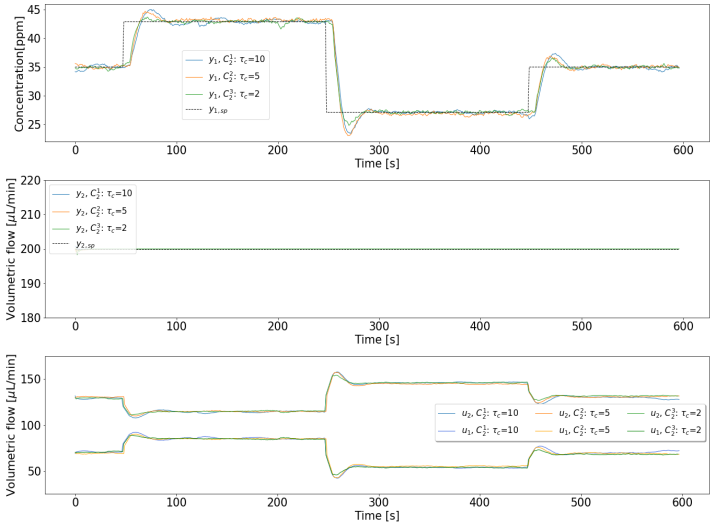


Figure 5.7: Simulation of the controlled model when changing the concentration setpoint, $y_{1,sp}$. The setpoint for the total flow, $y_{2,sp}$, is kept constant. The bottom graph show the control moves of the two MV's, u_1 and u_2. The results of the different tuning values of τ_c for controller $C_2(s)$ are shown in different colours, blue, orange and green.

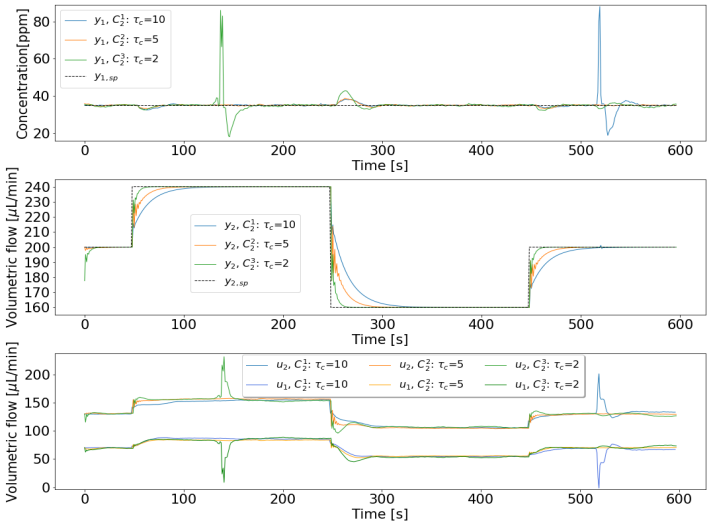


Figure 5.8: Simulation of the controlled model when changing the setpoint for the total flowrate, $y_{2,sp}$. The concentration setpoint, $y_{1,sp}$, is kept constant. The bottom graph show the control moves of the two MV's, u_1 and u_2. The results of the different tuning values of τ_c for controller $C_2(s)$ are shown in different colours, blue, orange and green.

In the experimental results, in Figure 5.7 and 5.8, one can see that the first controller, $C_1(s)$, is unaffected by the different values of τ_c for $C_2(s)$, which coincides with the observation made for the simulation results. Comparing the model and the experimental result of changing the concentration setpoint, shown in Figure 5.5 and 5.7, one can see that the plots are almost identical. The fact that the total flow seems completely unaffected by the change in concentration, can be explained by the assumption that $q = q_1 + q_2$. The decoupling matrix, presented in Equation 5.4, shows that the contribution added to the process input, u_2, is $-u_1$. The second controller will thus decrease the input flow of water by the exact amount that u_1 is increased by. As the total flow is not measured, but simply the sum of the two flows where the changes are made without any delay, this will result in a perfect flow of 200 $\mu L/min$ at all times. On the contrary, the concentration is measured online during the experiments, possibly introducing dynamics not captured by the model.

The model result of changing $y_{2,sp}$, Figure 5.6, show that a very sudden and large decrease in both inputs are made to ensure the decrease in total flow as well as keeping the concentration constant. When comparing this to the experimental result, shown in Figure 5.8, one can instead see some oscillating behaviour in the inputs. This may be due to the concentration output not acting precisely the way the model had predicted, and the contribution to input u_1 from the decoupling structure is not making up for all interactions with the second loop. These sudden changes in the flows can then be seen in the total flow output, as this is directly connected to the two input flows.

In Figure 5.8, one can see sudden jumps in the concentration at around $t = 130$ sec

and $t = 520$ sec. These are disturbances in the measurement equipment, and not the actual concentration at this time. As described in Section 3.3, air bubbles inside the tubes may get caught inside the spectrophotometer, resulting in unreasonable measurement values. Contamination of some sort may also result in wrongful measurements, as described in Appendix B. Its clear that the controller tries to counteract this by changing the flows, but since this measurement error is just temporary, it results in an overreaction and the concentration ends up below the setpoint.

The experimental results show very similar plots as the simulation results, shown in the previous sub-section. This indicates that the decoupling scheme has in fact improved performance significantly, compared to the decentralized PI-controller. The interactions are not completely eliminated, but are significantly reduced and the controllers do not oppose each other to the same extent.

5.3 Model predictive control for concentration and flow control

MPC is, as mentioned, a multivariable control technique that is often used for complicated MIMO control problems. This section will therefore present the method for developing and tuning a linear MPC for controlling the concentration and the total flow of the process evaluated in this thesis. The model results are presented and discussed in Section 5.3.2, and the experimental results and discussion are given in Section 5.3.3.

5.3.1 Development of a model predictive controller

The model predictive controller was designed and implemented with the *Model Predictive Control Toolbox* in Matlab, and with the corresponding MPC block in Simulink [44]. The toolbox transforms the current linear transfer function model into a state space representation, and uses a simple Kalman filter for state estimation at each sample time. It solves an optimization problem equivalent to the one given in Equation 2.20. The Simulink model can be seen in Figure F.5 in Appendix F.

The main challenge with implementing the MPC, was utilizing the ”black box” structure of the MPC toolbox. This resulted a time consuming tuning process, which did not always give intuitively understandable results. Additionally, when not being able to see how the optimization problem was constructed in detail, one was not able to fully understand the impact or the meaning of the different penalization variables, as they were denoted differently in Matlab. This might have resulted in sub-optimal problem definition and parameter tuning. However, this tool is a good starting point for making an MPC for a relatively simple problem and study the main features and properties of the controller. Therefore, this tool was seen as sufficient enough for making an MPC that could be compared with the other two controllers presented.

The following hard constraints were set for this control problem;

$$-150 \leq \Delta u_1 \leq 150 \tag{5.5}$$

$$-150 \leq \Delta u_2 \leq 150 \tag{5.6}$$

As disturbances in the measurements are likely to occur, the constraints on the input control moves were set to avoid any excessive input usage as a response to this. For the same reason, no constraint were put on the output variables, as this could cause infeasible solutions of the optimization problem, as mentioned in Section 2.3. The code running the two mechanical pumps included a safety measure to avoid any negative input flows, to ensure that the fluids were not contaminated. Therefore, it was not seen necessary to include any additional constraints on the MV's. The effect of adjusting the different tuning variables, presented in Section 2.3, is demonstrated and discussed in Appendix E. Based on the results from these model simulations and a further trial and error approach, the tuning settings for the MPC were set as given in Table 5.3;

Table 5.3: Table showing the tuning parameters of the model predictive controller.

Tuning parameter	Tuning parameter value
P	40
M	10
Q	$\begin{bmatrix} 50 & 0 \\ 0 & 10 \end{bmatrix}$
R	$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$

As no soft constraints were defined for this QP problem, the values of the tuning parameters ϵ and ρ were not defined.

For the experimental tests, the same setpoint step changes were made as for the PID controllers presented in Section 5.1 and 5.2. Steps in each of the output setpoints, $y_{1,sp}$ and $y_{2,sp}$, were made while keeping the other constant. The steps were made at times $t = 50$ sec, $t = 250$ sec and $t = 450$ sec, with an increase of 20%, a decrease of 2x20% and then an increase of 20% of nominal steady state values, respectively.

5.3.2 Simulation results: MPC

Figure 5.9 show the model result of implementing the controller with the optimal tuning parameters given in table 5.3.

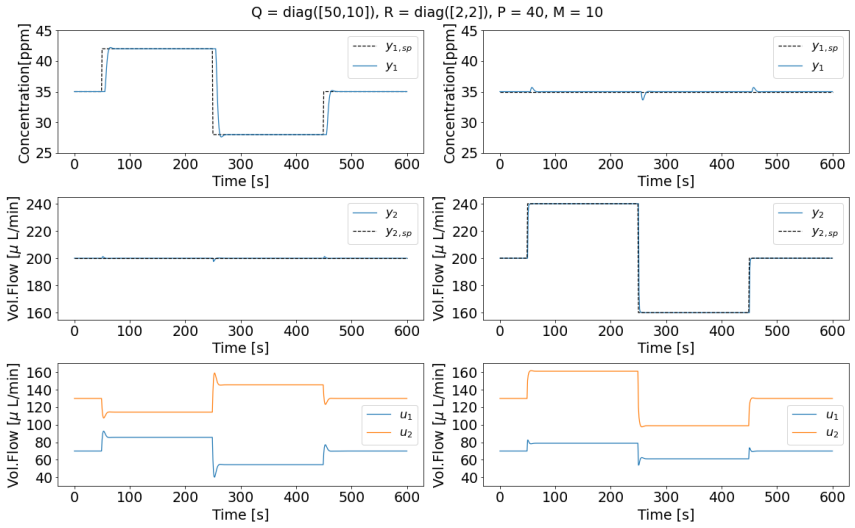


Figure 5.9: Model simulation of controlling the concentration and total flow of the system, with the MPC. The plots to the left show the setpoint tracking of y_1, while the right plots show the setpoint tracking of y_2. The values of the MPC tuning parameters are shown in the top of the figure.

Figure 5.9 show the model result of tracking the setpoints of y_1 and y_2, when doing a step in both $y_{1,sp}$ and $y_{2,sp}$. It shows good results in terms of setpoint tracking and rejecting the interactions in the system, without making any unnecessarily large changes in the input variables. The settling times are also significantly reduced compared to both PI-controllers. The MPC tuning parameters were chosen based on the analysis done in Appendix E, and with a trial and error approach. In the setpoint tracking weighting matrix, Q, the concentration output is weighted 5 times more heavily than the total flow output, as control of this variable was seen as top priority. However, the total flow seems to be just as tightly controlled as the concentration, and reducing values of the weighting elements might be possible without lowering the performance.

5.3.3 Experimental results: MPC

Figure 5.10 show the experimental result of implementing the controller with the actual process.

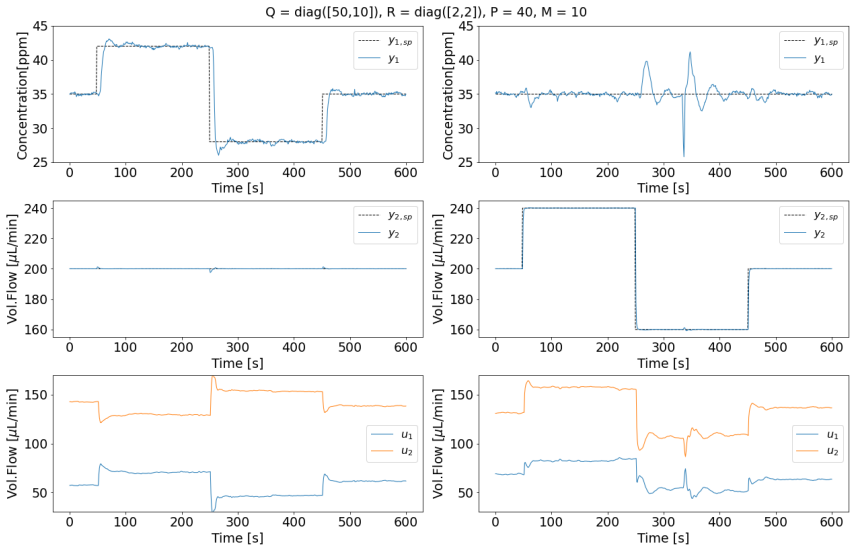


Figure 5.10: Experimental result of controlling the concentration and total flow of the system, with the MPC. The plots to the left show the setpoint tracking of y_1, while the right plots show the setpoint tracking of y_2. The values of the MPC tuning parameters are shown in the top of the figure.

When comparing the experimental result, seen in Figure 5.10, with the model result in Figure 5.9, it seems like the step changes in $y_{1,sp}$ give very similar results as the model. However, a larger overshoot than for the model result, and some small oscillatory behaviour can be seen for the concentration output. The control moves are also slightly more aggressive in the second step due to this. As for the other two controllers, this is most likely due to the model inaccuracy. For the setpoint steps in $y_{2,sp}$, the concentration measurements were quite noisy. This makes it difficult to interpret what are the disturbances and what are the interactions from the second loop. However, the penalization of the control moves seem to keep the input movements to a minimum, even with the large disturbances.

Due to the lack of flow measurements, it is not surprising that the total flow output in Figure 5.10 is near identical to the model response in Figure 5.9, with very small deviations from the setpoint. The concentration, on the other hand, deviates noticeably, but mostly due to the unplanned measurement disturbances. If one would disregard the concentration measurements between $t = 300 - 450$ sec, in the top right plot of Figure 5.10, one could conclude that the MPC showed good performance in terms of low settling times, and setpoint tracking with rapid and relatively smooth responses.

CHAPTER 6

Discussion, conclusions and future work

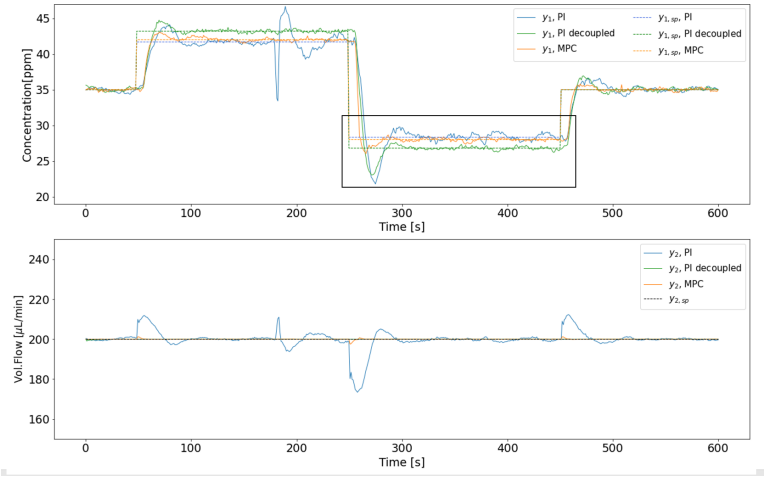
This chapter presents a summarizing discussion of the results presented in Chapter 5, as well as a conclusion and recommendations for further work.

6.1 Discussion

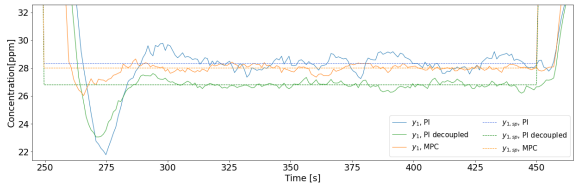
The three controllers presented in the previous chapter showed promising results in terms of setpoint tracking of the two output variables. Some challenges were common for the implementation of all control structures. For instance, the effect of dynamics unaccounted for in the model could be detected in the experimental results as increased oscillatory behaviour compared to the model results. This effect would be increased by the lack of flow measurement, giving an incorrect picture of the real process. Heo et al.(2016) states that when using mechanical pumps in microreactors, such as the syringe pumps used in this work, fluctuating flows due to deformation of channel walls and the motion of the electric motor is unavoidable [3]. These are elements neglected in the developed model and the following control structures, where the flows are assumed perfectly regulated and constant throughout the system. The disturbances in the concentration measurements occurred at seemingly random instances throughout experiments with all three controllers. This resulted in unnecessary control reactions, which became a challenge for the overall evaluations of controller performance. These disturbances were due to wrongful measurements at only a few sample times during the experiment, and were not a measure of the actual concentration being driven from the setpoint. Therefore, these disturbances could not be used for evaluating the controllers perfomance in terms of disturbance rejection.

Figure 6.1 show the previously presented experimental results of the three controllers compared. Here, the value of τ_c for both PI-controllers were based on which value gave best performance in terms of oscillations and error measurements. For the decentralized

PI-controller the result with the parameter value $\tau_c = 10$ is shown, and the result with $\tau_c = 2$ is shown for the decoupled PI-controller. As mentioned in Section 3.3, the nominal steady state value of the concentration deviated from the actual value, and the deviation was different for each experiment. The setpoints were calculated from these nominal values, which then resulted in slightly different steps in $y_{1,sp}$ as can be seen in Figure 6.1b. This was not the intention, as the setpoint changes were supposed to be calculated from the actual nominal value, but this was not discovered until after the experiments were completed. However, as the difference in setpoints are quite small (<2 ppm), it is assumed that this did not affect the controllers differently and that a comparison could be made regardless of this.



(a)



(b)

Figure 6.1: Figure (a): Result of controlling the concentration, y_1, and the total flow, y_2, of the process with the three different controllers. Here, the step changes are done for the setpoint of y_1. For the PI-controller, $C_2(s)$ was tuned with the value og $\tau_c = 10$, while the PI-controller with decoupling was tuned with the value of $\tau_c = 2$. Figure (b): Framed section of Figure (a), showing the second step in the concentration setpoint, $y_{1,sp}$, in which the three controllers operates with three different setpoints. This was due to the 20% and 2x20% step being calculated from the nominal steady state value, before they were shifted to the actual value of 35 ppm.

Table 6.1: Percentage overshoot for all three controllers for the setpoint tracking of y_1. The Overshoot is computed by the following formula: Overshoot = Amplitude/$\Delta y_{1,sp}$

	Step 1	Step 2	Step 3
PI-control [%]	36.4	50	25.7
PI-control, decoupled [%]	15.7	23	23.5
MPC [%]	14.3	12.9	8.6

From Figure 6.1, one can see that the MPC and the decoupled PI-controller performed better at keeping the total flow constant. They also had a lower percentage overshoot in all three steps, compared to the PI-controller. The percentage overshoots are given in Table 6.1, which shows that the MPC in fact had the lowest overshoot of all. The settling time of the MPC was $t_s = 47$ sec for the second step, while for the PI-controller with and without decoupling was $t_s = 55$ sec and $t_s = 70$ sec, respectively. This trend could be seen for the other steps as well, showing that the MPC was able to reach steady state more rapidly than the other two controllers. The decentralized PI-controller had a value of $\tau_c = 10$, while the decoupled PI-controller had a value of $\tau_c = 2$, which can explain the longer settling time for the decentralized controller, as larger closed loop time constant results in slower response. In terms of oscillations, the decentralized PI-controller showed more oscillatory behaviour than the other two controllers due to the increased interactions between the two control-loops. However, with $\tau_c = 10$ these oscillations were minimized.

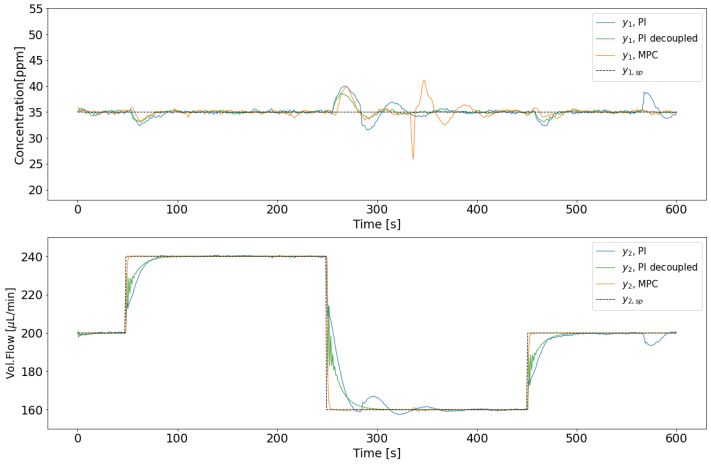


Figure 6.2: Result of controlling the concentration, y_1, and the total flow, y_2, of the process with the three different controllers. Here, the step changes are done for the setpoint of y_2. For the PI-controller, $C_2(s)$ was tuned with the value og $\tau_c = 10$, while the PI-controller with decoupling was tuned with the value of $\tau_c = 2$.

Figure 6.2 shows the response of all controllers for the steps done in $y_{2,sp}$. In this plot, both the PI-controller with decoupling and the MPC seems to perform well in terms of keeping the concentration constant. However, all three controllers are affected by the

change in total flow, resulting in some oscillatory behaviour. The settling time for the second step in $y_{2,sp}$ of the decentralized PI-controller, decoupled PI-controller and the MPC, was $t_s = 114$ sec, $t_s = 40$ sec and $t_s = 4$ sec, respectively. This shows a significantly shorter settling time for the MPC. In addition, no oscillations can be seen in the total flow, y_2, with the MPC.

Figure 6.3 show the plotted absolute deviations for the results in Figure 6.1a and 6.2. The deviations resulting from the disturbances give quite large outcomes on the absolute error, as can be seen in the top right plot for the MPC, for time $t = 330 - 420$ sec. To measure the performance of the three controllers, the mean absolute error (MAE) and the integral of the absolute error (IAE) was calculated. Due to the concentration measurement disturbances resulting in deviations in concentration and total flow, these data points were removed for most accurate measure of the controller performance. The values corresponding to Figure 6.1a are given in Table 6.2, while the values corresponding to Figure 6.2 are given in Table 6.3.

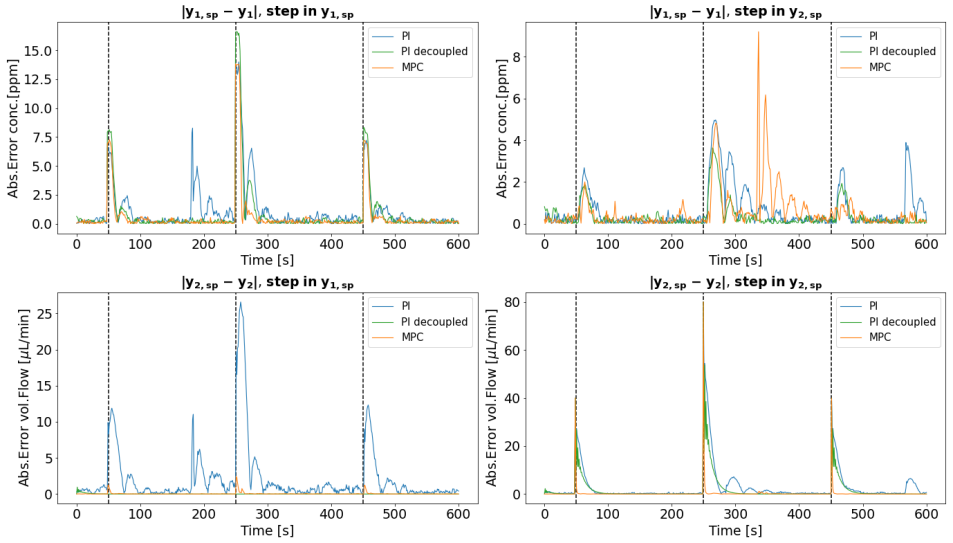


Figure 6.3: Absolute deviations of the output values from their desired setpoints, for the three controllers. The top two plots show the deviations of y_1 for the change in $y_{1,sp}$ and $y_{2,sp}$, respectively. The two bottom plots show the deviations of y_2 for the change in $y_{1,sp}$ and $y_{2,sp}$, respectively. The times where the setpoint changes are indicated by the vertical lines.

Table 6.2: Mean absolute error (MAE) and the integral of the absolute error (IAE), between the output value and the setpoint for the three controllers. This is from the result of changing the value of $y_{1,sp}$, as shown in Figure 6.1a

	PI-control	PI-control, decoupled	MPC
MAE(y_1) [ppm]	1.07	0.87	0.72
MAE(y_2) [$\mu L/min$]	2.05	0.01	0.06
IAE(y_1) [ppm]	683.36	520.02	468.48
IAE(y_2) [$\mu L/min$]	1273.84	6.16	37.69

Table 6.3: Mean absolute error (MAE) and the integral of the absolute error (IAE), between the output value and the setpoint for the three controllers. This is from the result of changing the value of $y_{2,sp}$, as shown in Figure6.2

	PI-control	PI-control, decoupled	MPC
MAE(y_1) [ppm]	0.60	0.35	0.40
MAE(y_2) [$\mu L/min$]	3.02	1.75	0.37
IAE(y_1) [ppm]	387.88	209.45	358.39
IAE(y_2) [$\mu L/min$]	1299.38	1041.82	239.12

From Table 6.2 one can see that the MPC and the PI-controller with decoupling have quite similar MAE-values, both lower than the decentralized PI-controller. The decoupling PI-controller is able to reduce interactions to an extent that gives an IAE-value of 6.156 $\mu L/min$, showing better performance than both of the other controllers. In Table 6.3, all three controllers show similar performance in keeping the concentration constant, with the PI-decoupling shceme showing slightly lower MAE- and IAE-values. In tracking the total flow setpoint, $y_{2,sp}$, the MPC showed values of IAE = 239.1239 $\mu L/min$ and MAE = 0.370 $\mu L/min$. This was significantly lower than the other two controllers.

Average runtime for one control-loop for all three controllers was 0.5 sec. This means that a sample-rate lower than this would not be possible with this control algorithm and hardware. With the fast dynamics of microreactor systems, one might need to have very small sampling-rates in order to maintain control of e.g. strict reaction conditions. In that case, a faster algorithm would have to be obtained. In this case, a TCP/IP-connection was used, which could have increased computation time, and could be avoided with implementing the control structure in Python. Optimizations performed by model predictive controllers are known to get computationally expensive with large and complex problems. It has therefore mostly been applied to process industry with slow dynamics [19]. This control problem is fairly simple, and the MPC was not more computationally expensive than the other controllers. However, this should be kept in mind when developing an MPC for more complicated microreactors.

6.2 Conclusions

The three controllers presented in this thesis were successfully implemented and gave relatively good results in terms of controlling the concentration and total flow to desired setpoint values, and compared to the model results. Based on the comparative results presented in this chapter, both the MPC and the decoupled PI-controller out-performed the regular PI-controller in all areas. These two controllers had less oscillations, shorter settling time, lower percentage overshoot for setpoint changes, and lower MAE- and IAE-values.

Although the MPC showed an overall lower settling time and lower overshoot for the concentration setpoint tracking than the decoupled PI-controller, they both showed similar results in terms of error measurements. In addition, one should take the ease of implementation and tuning into account. Both of these controllers were relatively simple to implement on this system. However, without a systematic approach for tuning of the MPC parameters, and with the MPC toolbox in matlab, the implementation of this controller was more time-consuming, as mentioned Section 5.3.1. It also raises the question whether current tuning choices for the MPC gave the most optimal results. With more complex problems, the implementation of a decoupling structure could also become more complicated and in worst case result in an unstable decoupling structure.

From this, a conclusion can be made that for this particular system, even though the MPC showed very good results in terms of performance, the proposed decoupling structure can be seen as the best fit in this case. This due to the fact that it showed good performance results, and it was very easily implemented and tuned. However, one should definitely consider the use of MPC if introducing more complexity to this system, and if the decoupling structure becomes very complicated.

6.3 Recommendations for future work

As the MPC in general is known for being able to handle complex multivariable systems, a controller of this type would perhaps be more suitable for complicated microrector systems, and should be investigated further. To determine if a controller is good or not, one should evaluate its performance and robustness. In further work with any of these controller, the robustness should be tested in terms of giving reasonable results for a wider range of process conditions. The performance should also be measured in term of disturbance rejection, by introducing (planned) disturbances of e.g. a third input flow. As mentioned previously, fluctuating flows when using mechanical pumps is unavoidable. Therefore, to properly include the dynamics of the liquid flow in the model, a flowmeter or another type of flow measuring equipment could be included in the system. This would improve the control of both the concentration and the total flow, but it will also complicate the model and thus the control structure.

Bibliography

[1] Jun Yue, Jaap C. Schouten, and T. Alexander Nijhuis, "Integration of microreactors with spectroscopic detection for online reaction monitoring and catalyst characterization", *Industrial and Engineering Chemistry Research*, vol. 51, no. 45, pp. 14583–14609, 2012.

[2] Paul Watts and Charlotte Wiles, "Recent advances in synthetic micro reaction technology", *Chemical Communications*, , no. 5, pp. 443–467, 2007.

[3] Young Jin Heo, Junsu Kang, Min Jun Kim, and Wan Kyun Chung, "Tuning-free controller to accurately regulate flow rates in a microfluidic network", *Scientific Reports*, vol. 6, no. March, pp. 1–12, 2016.

[4] W. C. Shin and R. S. Besser, "Toward autonomous control of microreactor system for steam reforming of methanol", *Journal of Power Sources*, vol. 164, no. 1, pp. 328–335, 2007.

[5] Thomas F.; Mellichamp Duncan A.; Doyle III Francis j. Seborg, Dale E.; Edgar, *Process Dynamics and Control fourth edition*, vol. 53, John Wiley & Sons, Inc, 4 edition, 2019.

[6] Klaus Jähnisch, Volker Hessel, Holger Löwe, and Manfred Baerns, *Chemistry in Microstructured Reactors*, vol. 43, 2004.

[7] Xingjun Yao, Yan Zhang, Lingyun Du, Junhai Liu, and Jianfeng Yao, "Review of the applications of microreactors", *Renewable and Sustainable Energy Reviews*, vol. 47, pp. 519–539, 2015.

[8] H. Löwe and W. Ehrfeld, "State-of-the-art in microreaction technology: concepts, manufacturing and applications", *Electrochimica Acta*, vol. 44, no. 21, pp. 3679–3689, 1999.

[9] Klavs F. Jensen, "Flow Chemistry - Microreaction Technology Comes of Age", *AIChE Journal*, vol. 63, no. 3, pp. 858–869, 2017.

[10] ATA Scientific Intstruments, "Spectrometry and Spectroscopy: What's the Difference?", 2020.

[11] Muhammad Sajid Hamid Akash and Kanwal Rehman, "Essentials of pharmaceutical analysis", chapter 3, pp. 29–56. Springer Nature Singapore Pte Ltd., 2020.

[12] Navy Environmental Health Center, "UV Radiation GUide", *Navy environmental health center*, , no. April, 1992.

[13] Dr Jenny A Koenig, "3.B.4 Using spectrophotometry to determine the concentration of a substance in a mixture. Learning Objectives *", pp. 3–5.

[14] Akul Mehta, "Ultreviolet-Visible (UV-Vis) Spectroscopy - Limitations and Deviations of Beer-Lambert law", 2013.

[15] Yong Joon Lee, Su Whan Sung, Dae Sung Yoon, Geunbae Lim, and Sunwon Park, "MODELING AND CONTROL OF THERMAL MICROSYSTEMS", 2000.

[16] D. J. Quiram, K. F. Jensen, Martin A. Schmidt, P. L. Mills, J. F. Ryley, M. D. Wetzel, and D. J. Kraus, "Integrated Microreactor System for Gas-Phase Catalytic Reactions. 3. Microreactor System Design and System Automation", *Industrial & Engineering Chemistry Research*, vol. 46, no. 25, pp. 8319–8335, 2007.

[17] Milhai P. Dinca, Marin Gheorghe, and Paul Galvin, "Design of a PID controller for a PCR micro reactor", *IEEE Transactions on Education*, vol. 52, no. 1, pp. 116–125, 2009.

[18] Sara Gómez De Pedro, Cynthia S. Martínez-Cisneros, Mar Puyol, and Julián Alonso-Chamarro, "Microreactor with integrated temperature control for the synthesis of CdSe nanocrystals", *Lab on a Chip*, vol. 12, no. 11, pp. 1979–1986, 2012.

[19] Leonidas G. Bleris, Jesus G. Garcia, Mark G. Arnold, and Mayuresh V. Kothare, "Model predictive hydrodynamic regulation of microflows", *Journal of Micromechanics and Microengineering*, vol. 16, no. 9, pp. 1792–1799, 2006.

[20] Leonidas G. Bleris, Jesus Garcia, Mayuresh V. Kothare, and Mark G. Arnold, "Towards embedded model predictive control for System-on-a-Chip applications", *Journal of Process Control*, vol. 16, no. 3, pp. 255–264, 2006.

[21] Jeevan Maddala and Raghunathan Rengaswamy, "Droplet digital signal generation in microfluidic networks using model predictive control", *Journal of Process Control*, vol. 23, no. 2, pp. 132–139, 2013.

[22] Jeevan Maddala, Babji Srinivasan, Swastika S. Bithi, Siva A. Vanapalli, and Raghunathan Rengaswamy, "Design of a model-based feedback controller for active sorting and synchronization of droplets in a microfluidic loop", *AIChE Journal*, vol. 58, no. 7, pp. 2120–2130, 2011.

[23] Karl J Astrom and Tore Hagglund, *PID Controllers : Theory , Design and Tuning*, International Society for Measurement and Control, 2 edition, 1995.

[24] M Araki, "Control system, robotics and automation", vol. II, pp. 694–696, 2017.

[25] S.Skogestad and C.Grimholt, "PID Control in the Third Millenium - Lessons Learned and New Approaches", *Advances in Industrial Control*, p. 600, 2012.

[26] Liuping Wang, *Pid Control System Design and Automatic Tuning Using Matlab/Simulink (Wiley - IEEE)*, John Wiley and Sons Ltd, 2020.

[27] S Bharadwaj Reddy, "Dead time".

[28] Daniel E. Rivera, Manfred Morari, and Sigurd Skogestad, "Internal Model Control. 4. PID Controller Design", *Ind. Eng. Chem. Process Des. Dev.*, vol. 25, no. 1, pp. 252–265, 1986.

[29] Edgar H. Bristol, "On a new measure of interaction for multivariable process control", *IEEE Transactions on Automatic Control*, vol. 11, no. 1, pp. 133–134, 1966.

[30] Sigurd Skogestad and Ian Postlethwaite, *Multivariable Feedback Control: Analysis and Design*, Wiley-Interscience, 2 edition, 2005.

[31] Morten Hovd and Sigurd Skogestad, "Sequential design of decentralized controllers", *Automatica*, vol. 30, no. 10, pp. 1601–1607, 1994.

[32] Shing Jia Shiu and Shyh Hong Hwang, "Sequential Design Method for Multivariable Decoupling and Multiloop PID Controllers", *Industrial and Engineering Chemistry Research*, vol. 37, no. 1, pp. 107–119, 1998.

[33] Harold L. Wade, "Inverted decoupling: A neglected technique", *ISA Transactions*, vol. 36, no. 1, pp. 3–10, 1997.

[34] F G Shinskey, *Process-control Systems Application, Design and Adjustment*, McGraw Hill Book Company, 1988.

[35] E. Gagnon, A. Pomerleau, and A. Desbiens, "Simplified, ideal or inverted decoupling?", *ISA Transactions*, vol. 37, no. 4, pp. 265–276, 1998.

[36] MathWorks, "Optimization problem", 2021.

[37] Gaurang Shah and Sebastian Engell, "Tuning MPC for desired closed-loop performance for SISO systems", *18th Mediterranean Conference on Control and Automation, MED'10 - Conference Proceedings*, pp. 628–633, 2010.

[38] U.A, "Safety Data Sheet . Safety Data Sheet", *Material Safety Data Sheet*, vol. 4(2), no. 1907, pp. 8–10, 2012.

[39] Leonidas Perez-Estrada, Ana Agüera, M Hernando, S Malato, and Amadeo Fernández-Alba, "Photodegradation of Malachite Green under Natural Sunlight Irradiation: Kinetic and Toxicity of the Transformation Products", *Chemosphere*, vol. 70, pp. 2068–2075, 2008.

[40] Cetoni GmbH, “Cetoni -Datasheet”.

[41] Cetoni GmbH, “Qmix SDK Documentation”.

[42] Ocean Insight, “Ocean Direct”.

[43] Nathan Jennings, “Socket Programming in Python (Guide)”, 2018.

[44] MathWorks, “Model Predictive Control Toolbox”, 2021.

[45] Sigurd Skogestad, “Simple analytic rules for model reduction and PID controller tuning”, *Journal of Process Control*, vol. 13, no. 4, pp. 291–309, 2003.

APPENDIX A

Background measurements

Figure A.1 show all the background tests that have been performed before and after a series of experiments. They are labeled with the date in which the background tests were done. All reference measurements show the same trend, and seem to lie within an acceptable range. The largest deviations are seen around a wavelength of 300 nm and 550 nm. The wavelength that was used for the calculation of concentration during experiments, was 446 nm, and the deviation around this wavelength is relatively small. This indicates that there has been consistency throughout the experiments, and that the measured concentration would not have been affected too much from these variations.

As an example, consider a solution with an intensity measurement of 500. At wavelength 446 nm, the highest reference measurement according to figure A.1, records an intensity of approximately 1800, and the lowest is approximately 1400. This results in two different concentration calculations on the two different dates, respectively,

$$\begin{aligned} A_{I=1800} &= log\frac{1800}{500} = 0.55 \\ C_{I=1800} &= \frac{A_{I=1800}}{m} - \frac{b}{m} = 17.48ppm \end{aligned} \tag{A.1}$$

$$\begin{aligned} A_{I=1400} &= log\frac{1400}{500} = 0.44 \\ C_{I=1400} &= \frac{A_{I=1400}}{m} - \frac{b}{m} = 13.76ppm \end{aligned} \tag{A.2}$$

Where $m = 0.0295$ and $b = 0.0341$. The variations in the reference measurements obviously have some impact on the calculated concentrations, however, the maximum de-

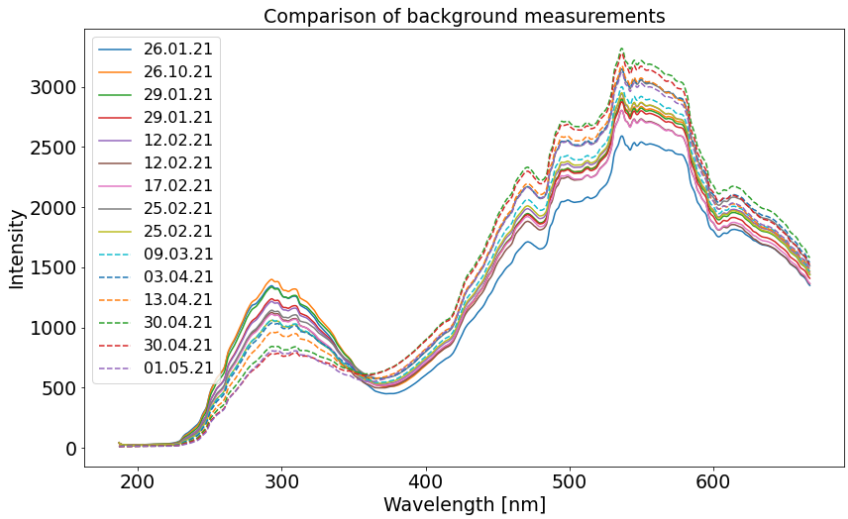


Figure A.1: Background measurements, also called reference measurements, taken of pure water flow. The measurements were taken before any experiments were started, and after all experiments had ended, on that day.

viation is only around 4 ppm at this wavelength. By shifting the concentration for each experiment, as explained in Appendix B, this deviation could to some extent be avoided.

APPENDIX B

Measurement variations

As mentioned in Section 3.3, the measurements were shifted with the deviation of the mean of the reference values taken before the start of the experiment, and the expected concentration value. Figure B.1 show two such reference measurements, taken on the same day.

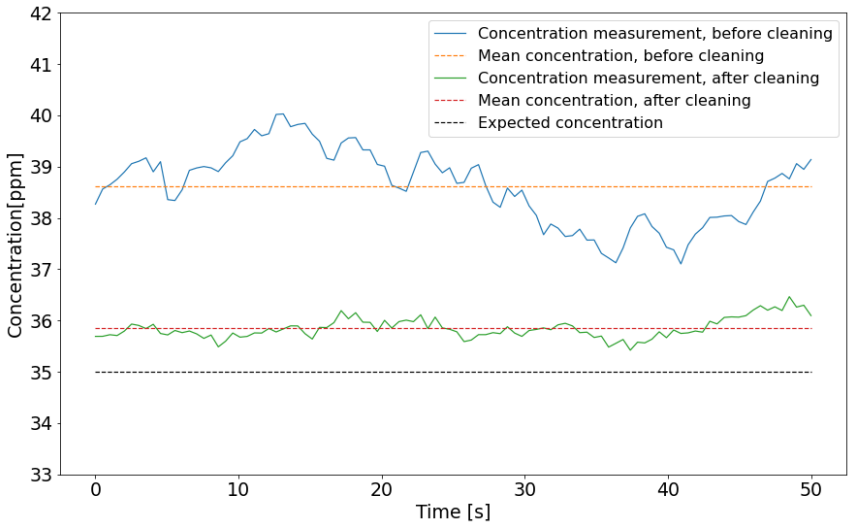


Figure B.1: The deviation of the reference measurement of the concentration before and after cleaning with IPA, and the actual concentration.

The first measurements, shown in blue, vary to a relatively large extent from the mean value. This caused problems with controlling the concentration, as the controller would adjust the inputs to compensate for the deviations from setpoint, and therefore adjust to wrong flow settings. These measurement movements were also sudden and random, which resulted in relatively large and unwanted jumps in the flow inputs. It was therefore decided to wash the equipment more thoroughly with pure isopropanol, and the result after the cleaning is shown in green in figure B.1. This curve is clearly at a steady state, with a much smaller deviation from the actual value of 35 ppm. This substantiates the assumption made in section 3.3, that the wrongful measurements are in fact due to contamination of the spectrophotometer, and that the expected value should be treated as the actual value.

The MPC controller that was tested in this case, before and after cleaning, showed a much improved result with the second experiment. The result of controlling the total flow and the concentration with the MPC during a setpoint step test, is shown in Figure B.2.

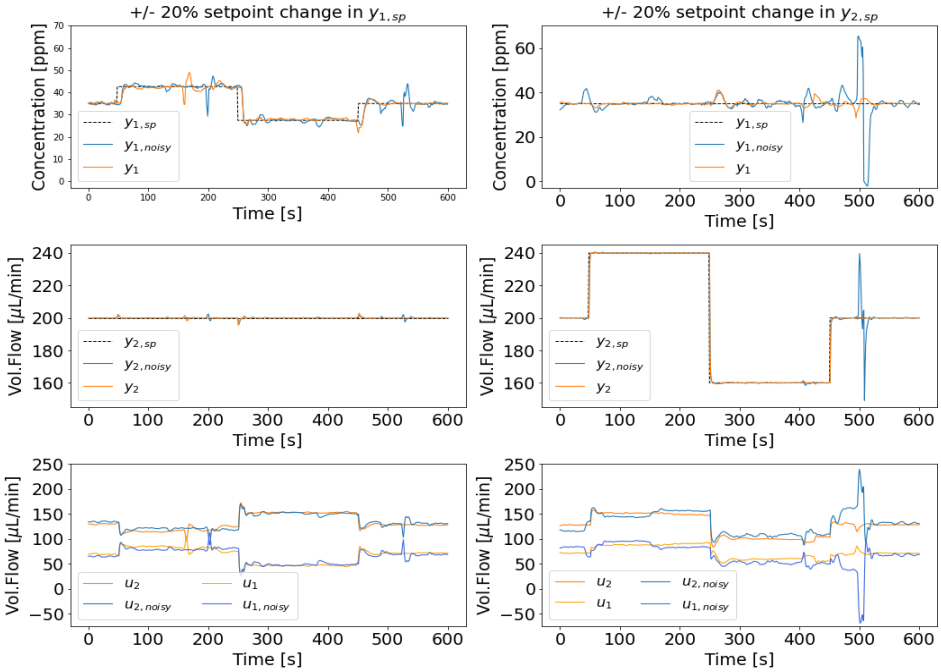


Figure B.2: Example of two experiments with the MPC run before and after cleaning with IPA. The blue graph shows the experiment before cleaning, and shows more noisy measurements than the orange graph, resulting in a system that is more difficult to control.

The blue graph shows the experiment done before cleaning the equipment, while the orange graph shows the experiment after. Although there are some disturbances in the concentration measurements after the cleaning, there is a significant improvement from the blue graph. The controller does not need to adjust the inputs as often, and with as much input usage, as is seen by the blue curves in the two bottom graphs.

This raises the question on whether the equipment should be washed with isopropanol prior to every experiment that is performed in the future.

APPENDIX C

Additional step responses

The figures in this section are additional results from the work with developing a process model for the system, which is described in Section 4.2. Figure C.7 show the response curve used to tune the second controller for the decentralized PI-control structure, presented in Section 5.1.1.

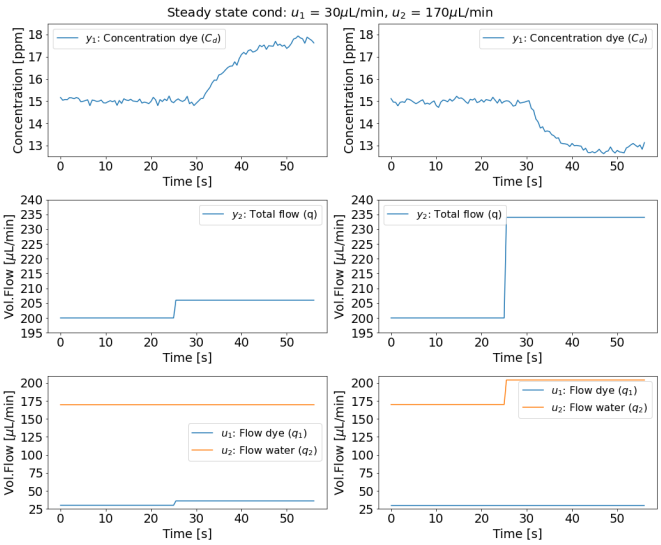


Figure C.1: Step response test for Case 1, with steady state conditions as given in the top of the figure. The left and right plots show the result of performing a 20% step change in the input variable u_1 and u_2, respectively. The input changes are shown in the bottom two plots, while the corresponding responses of y_1 and y_2 can be seen in the top four plots.

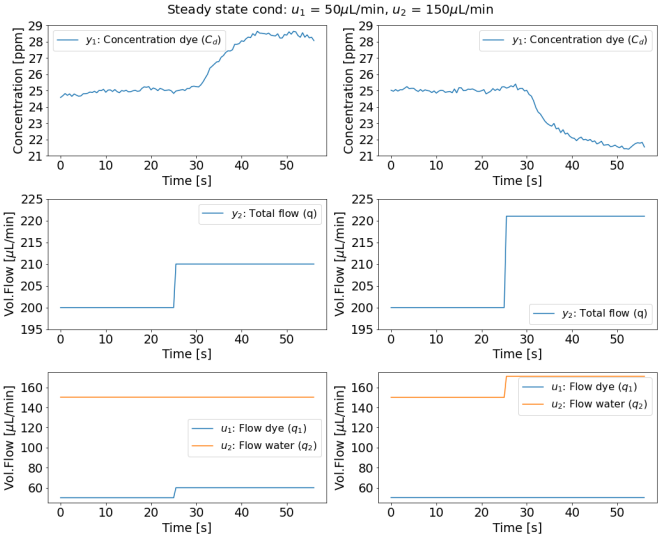


Figure C.2: Step response test for Case 2, with steady state conditions as given in the top of the figure. The left and right plots show the result of performing a 20% step change in the input variable u_1 and u_2, respectively. The input changes are shown in the bottom two plots, while the corresponding responses of y_1 and y_2 can be seen in the top four plots.

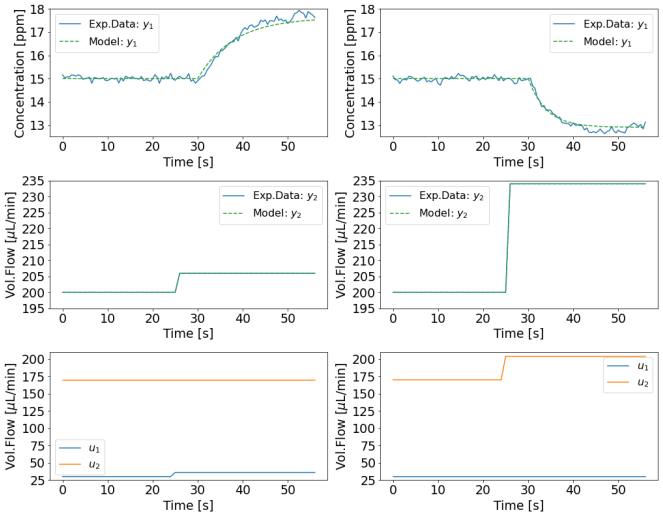


Figure C.3: Experimental and model data of step response for Case 1. The plot show a 20% step in the input values u_1 and u_2 to the bottom left and right, respectively, with the corresponding responses of each of the output variables, y_1 and y_2, in the top four plots. The simulated model response is shown as the green stippled line.

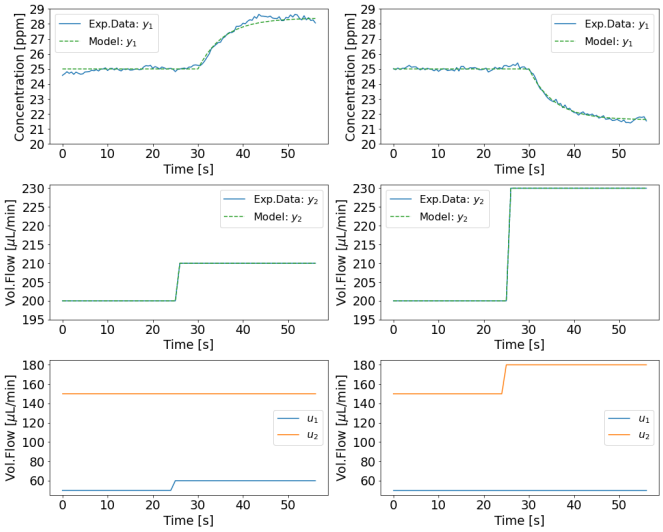


Figure C.4: Experimental and model data of step response for Case 2. The plot show a 20% step in the input values u_1 and u_2 to the bottom left and right, respectively, with the corresponding responses of each of the output variables, y_1 and y_2, in the top four plots. The simulated model response is shown as the green stippled line.

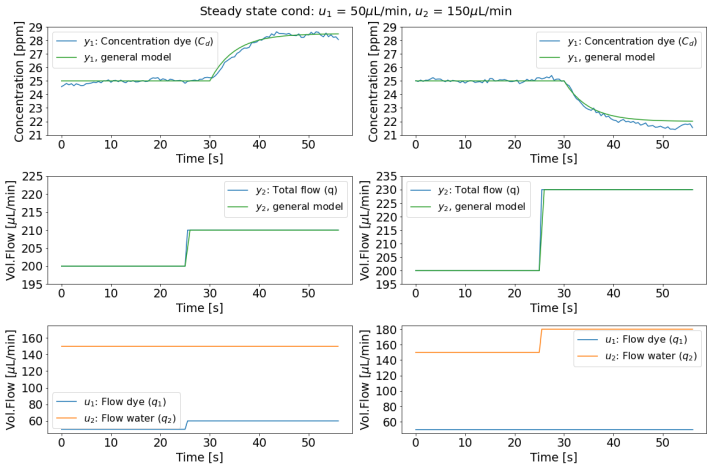


Figure C.5: Experimental step response for Case 2 compared with the model response of the general model given in equation 4.6. The plot show a 20% step in the input values u_1 and u_2 to the bottom left and right, respectively, with the corresponding responses of each of the output variables, y_1 and y_2, in the top four plots. The simulated model response is shown as the green stippled line.

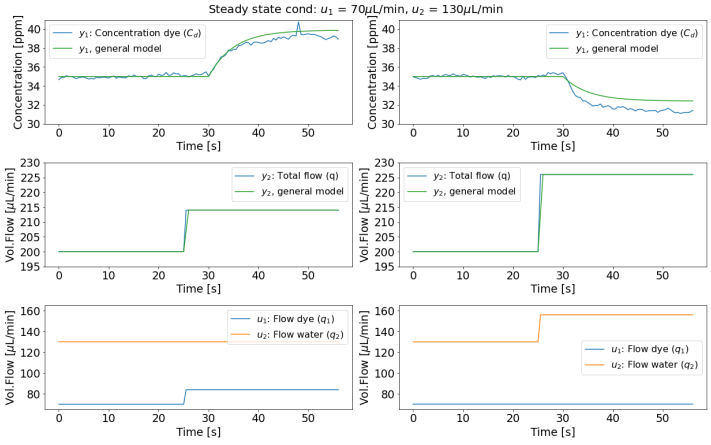


Figure C.6: Experimental step response for Case 3 compared with the model response of the general model given in equation 4.6. The plot show a 20% step in the input values u_1 and u_2 to the bottom left and right, respectively, with the corresponding responses of each of the output variables, y_1 and y_2, in the top four plots. The simulated model response is shown as the green stippled line.

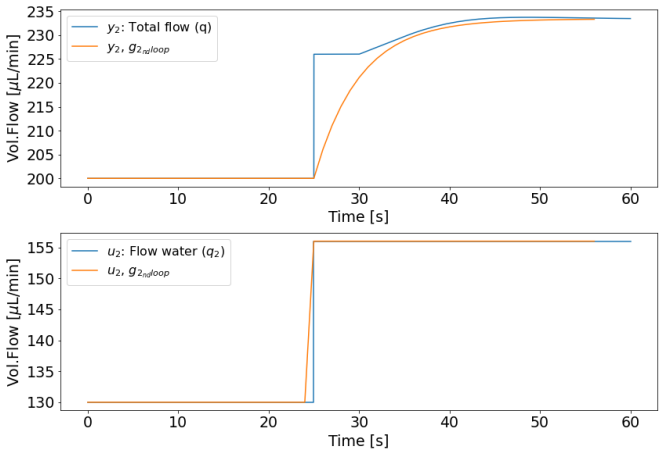


Figure C.7: Step response of y_2 when performing a step in input u_2, with loop 1 closed. With loop 1 closed, the output y_1 is controlled by controller $C_1(s)$. The orange graph shows the resulting model step response of the graphically approximated transfer function.

APPENDIX D

Derivation of the SIMC tuning rules

Deriving the SIMC rules, as done in the paper by Skogestad(2012) [25], starts by defining the closed loop setpoint response as,

$$\frac{y}{y_{sp}} = \frac{G(s)C(s)}{G(s)C(s)+1} \tag{D.1}$$

where the output measurement is assumed to be perfect and without noise. Rearranging this equation gives the corresponding controller function,

$$C(s) = \frac{1}{G(s)} \frac{1}{\frac{1}{y/y_{sp}} - 1} \tag{D.2}$$

The desired closed loop response can be defined as a smooth first order response with time delay as follows,

$$(\frac{y}{y_{sp}})_{desired} = \frac{1}{\tau_c s + 1} e^{\theta s} \tag{D.3}$$

By substituting this desired response, in addition to the FOPTD-model in equation 2.4, into equation D.2, and rearranging the expression, one obtains the following controller function,

$$C(s) = \frac{(\tau_1 s + 1)}{k} \frac{1}{\tau_c s + 1 - e^{\theta s}} \tag{D.4}$$

The delay can be approximated by a first-order Taylor expansion, $e^{\theta s} = 1 - \theta s$, resulting in PI-controller in series form;

$$C(s) = \frac{(\tau_1 s + 1)}{k} \frac{1}{(\tau_c + \theta) s} \tag{D.5}$$

Series form of the PID controller is used for this derivation due to the PID rules being simpler than with the ideal form. With this, the PI parameters can be found based on the IMC-rules in the paper by Rivera et.al.(1986), with the modifications done by Skogestad(2003) with the SIMC method; [28, 45]

$$K_c = \frac{1}{k}\frac{\tau_1}{(\tau_c + \theta)} \qquad \text{(D.6)} \qquad \tau_I = min(\tau_1, 4(\tau_c + \theta)) \qquad \text{(D.7)}$$

As this derivation was based on the FOPTD-model from equation 2.4, one gets a PI-controller with no derivative action, and $\tau_D = 0$. PID-control is primarily recommended only when the process shows signs of dominant second order dynamics [45]. If a second order model had been considered, the SIMC-rules would provide the following additional parameter value;

$$\tau_D = \tau_2 \qquad \text{(D.8)}$$

Where τ_2 comes from the expression of a second order plus time delay model shown as,

$$G(s)_{2^{nd}order} = \frac{k}{(\tau_1 s + 1)(\tau_2 s + 1)} e^{-\theta s} \qquad \text{(D.9)}$$

Some special cases of processes will require different SIMC PID-settings, which, as mentioned earlier, is not explored in this thesis. However, a table showing these settings can be found in the paper by Skogestad(2003) [45].

APPENDIX E

MPC model simulations: Effect of changing tuning variables

The method for tuning the model predictive controller was based on a trial and error approach, with some simple guidelines given in Section 2.3.1. To demonstrate how different values of the tuning variables P, M, Q and R affect the output, they were all changed in turn, while keeping the others constant. The settling time, t_s, of the open loop response of the system was approximately 30 sec, and the sampling time was 1 sec. This gave the value of $N = 30$. The initial values of P and M were then decided according to the guidelines, in which M would be in the range (10,15) and P in the range (40,45). The initial values were decided as $M = 10$ and $P = 40$.

Figure E.1 show the effect of changing the tuning parameter Q. This parameter is chosen based on which output one chooses to prioritize in regards to tracking its setpoint. On the left, the weight corresponding to the concentration output, y_1, has the larger value. One can then see that the controller provides fast and tight control of the concentration, while reacting slower for the control of the total flow. This effect is reversed in the plots on the right in the figure.

In Figure E.2, the weighting matrix R is changed, while Q is kept constant in a matter that prioritizes the control of the concentration. R is the weighting matrix for the control moves, Δu, and a higher weight will penalize large input movements of the corresponding input variable. The effect of changing this tuning parameter is not obvious in the figure. However, one can see that with a larger penalization of Δu_2, the controller uses longer time to correct for the changes in y_2. This is a consequence of the controller making smaller control actions in each step, not being able to correct the deviation instantly.

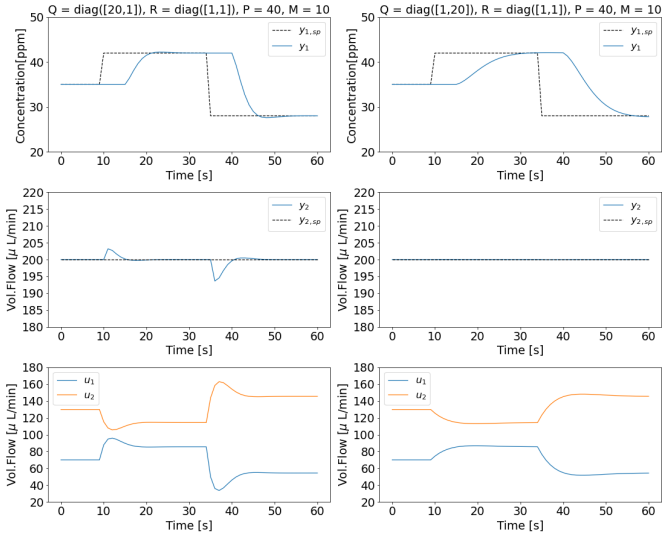


Figure E.1: Control of concentration and total flow with the MPC, where the values of tuning parameter Q is changed. The left plots show the effect of prioritizing control of the concentration, while the plots on the right show the effect of prioritizing control of the flow.

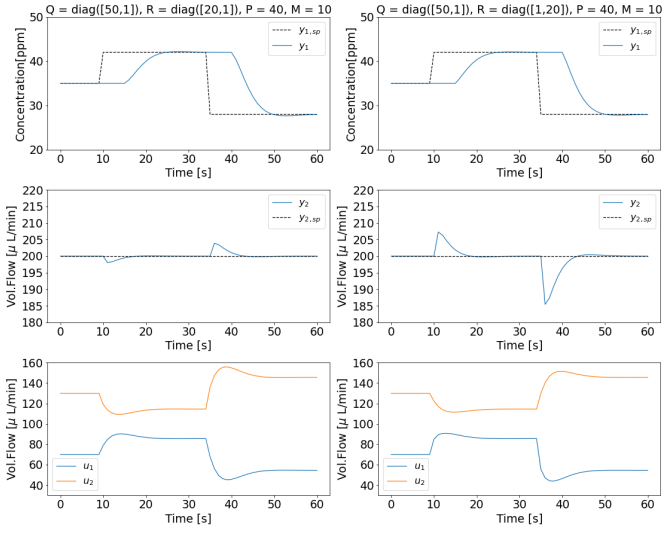


Figure E.2: Control of concentration and total flow with the MPC, where the values of tuning parameter R is changed. The left plots show the effect of larger penalization of input usage for u_1, while the plots on the right show the effect of larger penalization input usage for u_2.

For the two Figures E.3 and E.4, no change was seen when changing the values of M from 10 to 40, or the value of P from 40 to 90. Increasing the value of the model horizon, M, is said to make the controller more aggressive, while increasing the value of the prediction horizon, P, makes it less aggressive. As the effect is not visible in the model simulation, the two initial values were assumed to be sufficient.

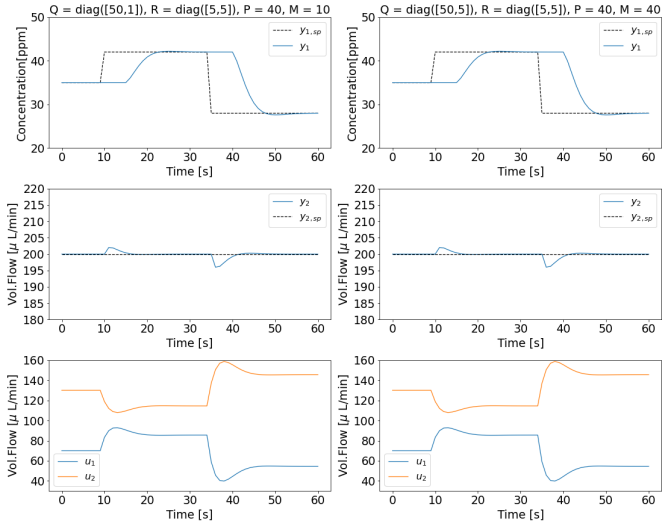


Figure E.3: Control of concentration and total flow with the MPC, where the control horizon M is changed. The left plots show a lower value of M, while the plots on the right show a higher value. However, there seemed to be no visible effect of changing this variable.

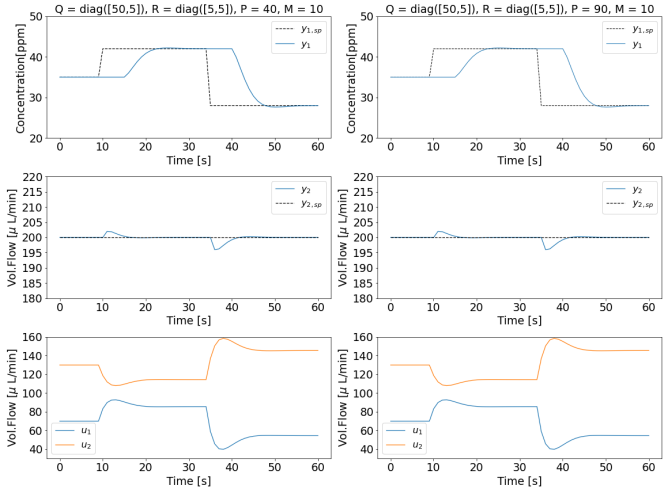


Figure E.4: Control of concentration and total flow with the MPC, where the prediction horizon P is changed. The left plots show a lower value of P, while the plots on the right show a higher value. However, there seemed to be no visible effect of changing this variable.

APPENDIX F

Simulink models

All simulink models used to simulate the system, are shown in the figures presented in this appendix.

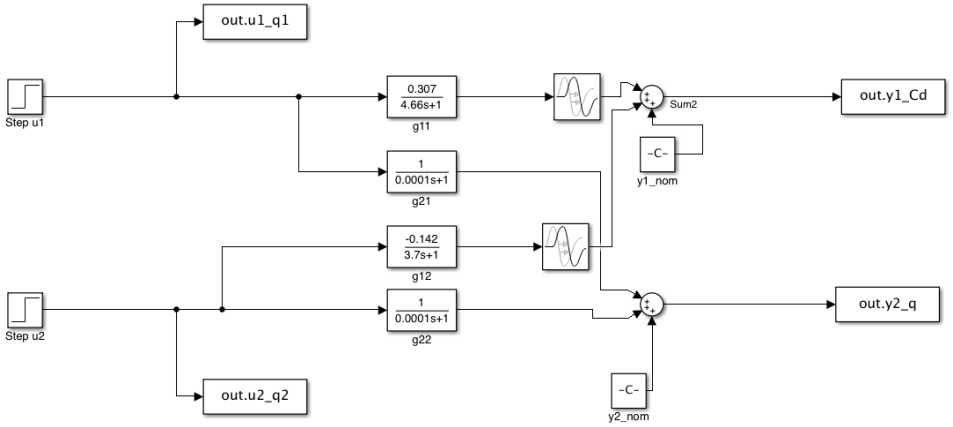


Figure F.1: Simulink model used for performing the open loop step response tests with the approximated transfer function models.

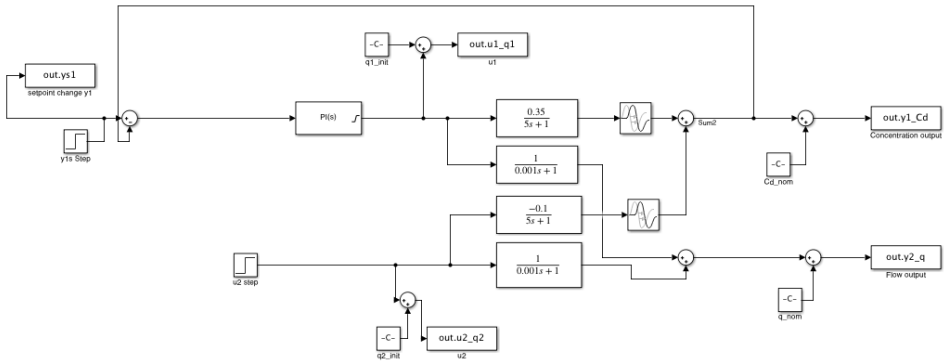


Figure F.2: Simulink model used to tune the second loop of the decentralized PI-controller. The first loop containing controller $C_1(s)$ is closed, while the second loop is open. A 20% step in input variable u_2 is done.

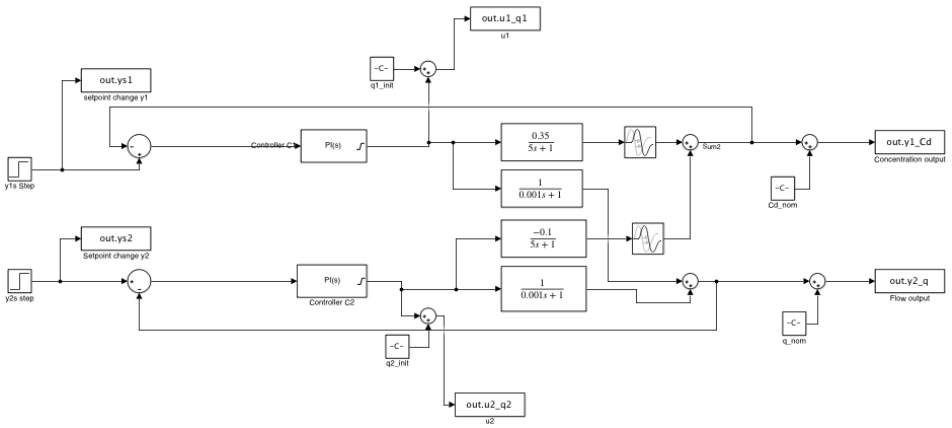


Figure F.3: Simulink model for model simulation of decentralized PI-control. The two controllers, $C_1(s)$ and $C_2(s)$, were tuned with the sequential approach.

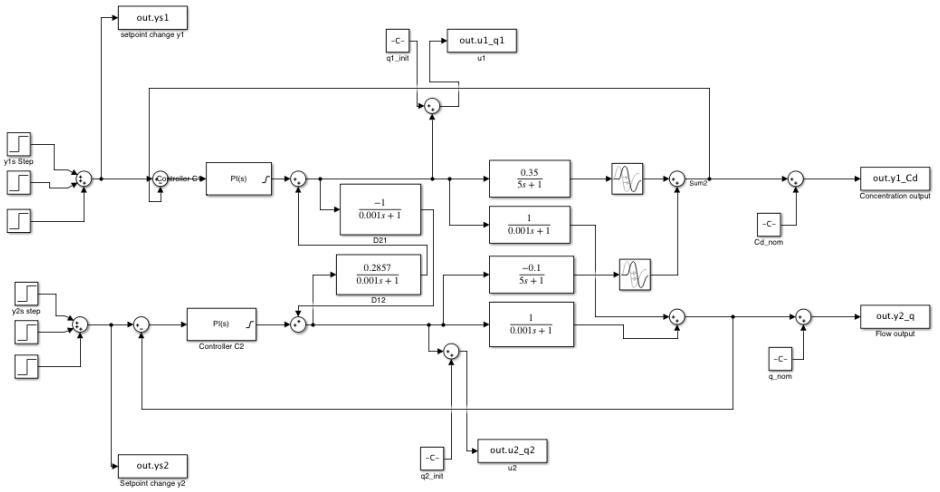


Figure F.4: Simulink model for model simulation of decentralized PI-control with decoupling. The two decoupling blocks $D_{1,2}$ and $D_{2,1}$ are approximated as first order transfer functions with a small time constant, to avoid problems of algebraic loops.

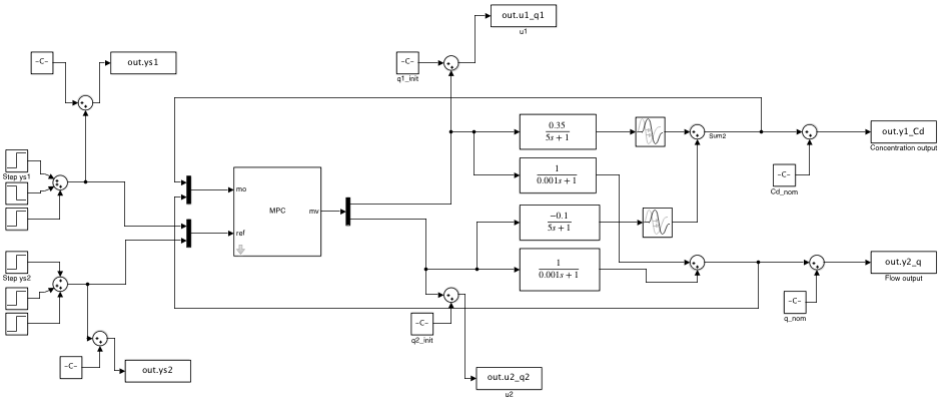


Figure F.5: Simulink model for model simulation of model predictive control. The MPC block performs state estimation and optimization, before injecting the optimal control move to the process.

APPENDIX G

Code: Matlab and Python

G.1 Matlab code for model simulation with controllers

```
%% Code for control of model with PI-control and
   PI-control with decoupling
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Setting controller tuning parameters

%Controller 1
tauC = 5;
Kc = 1/0.35 * 5/(5+tauC);
tauI = 5;
P1 = Kc;
I1 = Kc/tauI;

%Controller 2
tauC_2 = 2;
Kc_2 = 1/1.284*5/tauC_2;
tauI_2 = 5;
P2 = Kc_2;
I2 = Kc_2/tauI_2;


%% Simulate model
% Simulation time
simtime = 600;
```

```
% Step times
step_time = 25;
step_time_2 = 250;
step_time_3 = 450;

% Nominal steady state values
q1_nom = 70;
q2_nom = 130;
q_nom = q1_nom+q2_nom;
Cd_in = 100;
Cd_nom = q1_nom/q_nom*Cd_in;

% Assign setpoint step size for each step time
ys_step_1 = 0.2*Cd_nom;
ys2_step_1 = 0.2*q_nom*0;
ys_step_2 = -2*0.2*Cd_nom;
ys2_step_2 = -2*0.2*q_nom*0;
ys_step_3 = 0.2*Cd_nom;
ys2_step_3 = 0.2*q_nom*0;

% Assign input step size for first step (for step response test)
u1_step_1 = 0.2*q1_nom*0;
u2_step_1 = 0.2*q2_nom*0;

%Simulate relevant model
sim("controlModel");
%sim("ControlModelDecoupled");
%sim("controlModel_tuning");
%sim("ControlModelTuning_decoupled");
```

```
%% Code for simulating control of model with MPC
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% System model: Linear transfer function model

g11 = tf( 0.35, [5 1], 'IOdelay', 5.0,'TimeUnit','seconds');
g12 = tf( -0.1, [5 1], 'IOdelay', 5.0,'TimeUnit','seconds');
g21 = tf( 1, [0.001 1], 'IOdelay', 0,'TimeUnit','seconds');
g22 = tf( 1, [0.001 1], 'IOdelay', 0,'TimeUnit','seconds');


sys = [g11,g12;g21,g22];

sys.InputName = {'Flow dye', 'Flow water'};
sys.OutputName = {'Concentration dye', 'Total flow'};

%% Tuning parameters for controller

P = P; %Prediction horizon
M = M; %Control horizon

Q = {[Q_y1,0;0,Q_y2]}; %Setpoint tracking weights
R = {[R_Du1,0;0,R_Du2]}; %MV rate weights

%% Construct MPC
dt = dt; %Sample time [s]
MPC = mpc(sys,dt,P,M);

%Include hard constraints on MV rates
delUmax = 150;

MPC.MV(1).RateMin = -delUmax;
MPC.MV(2).RateMin = -delUmax;
MPC.MV(1).RateMax = delUmax;
MPC.MV(2).RateMax = delUmax;

%Assign weights
MPC.Weights.OutputVariables = Q;
MPC.Weights.ManipulatedVariablesRate = R;


%% Simulate model
%Simulation time
simTime = 600;

%Step times
```

```
step_time = 50;
step_time_2 = 250;
step_time_3 = 450;

%Nominal steady state values
q1_nom = 70;
q2_nom = 130;
q_nom = q1_nom+q2_nom;
Cd_in = 100;
Cd_nom = q1_nom/q_nom*Cd_in;

%Assign step size for each step time
ys_step_1 = 0.2*Cd_nom*0;
ys2_step_1 = 0.2*q_nom;
ys_step_2 = -2*0.2*Cd_nom*0;
ys2_step_2 = -2*0.2*q_nom;
ys_step_3 = 0.2*Cd_nom*0;
ys2_step_3 = 0.2*q_nom;

%Simulate
sim('MPCsim');
```

G.2 Matlab and Python code for controlling actual plant

```
%% Matlab code for PI-control of actual plant with TCP/IP
   connection to Python

        This code receives measurements from Python,
        and updates input values through one step of
        simulink simulation with the PI-controllers,
        with and without decoupling.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Setting PI-controller tuning parameters

%Controller 1
tauC = 5;
Kc = 1/0.35 * 5/(5+tauC);
tauI = 5;
P1 = Kc;
I1 = Kc/tauI;

%Controller 2
tauC_2 = 10;
Kc_2 = 1/1.284*5/tauC_2;
tauI_2 = 5;
P2 = Kc_2;
I2 = Kc_2/tauI_2;

%% Start and pause simulation, waiting for next input
open_system('ControlLabExp')
set_param(gcs,'SimulationCommand','start','SimulationCommand','pause');

%% Open server, wait for client to connect
s = tcpip('0.0.0.0', 9998,  'NetworkRole', 'server');
fopen(s);

%% Main loop

all_data = [];
count=0;

while count<600
    while(1) % Loop until Python sends readable data
        nBytes = get(s,'BytesAvailable');
        if nBytes>0
            break;
        end
    end
```

```
    command = fread(s,nBytes); % Read binary as str

    data=str2num(char(command')); % Transform str to numerical

    all_data = [all_data;data]; % Store history data
    if isempty(data)
        data=[0,0,0,0];
    end

    % Measured data from plant:
    y1_m = data(1);
    y2_m = data(2);
    y1_sp = data(3);
    y2_sp = data(4);

    % Set paramters in simulink model using the measured data
      received from python:
    set_param('ControlLabExp/y1_m','Value',num2str(y1_m))
    set_param('ControlLabExp/y2_m','Value',num2str(y2_m))
    set_param('ControlLabExp/y1_sp','Value',num2str(y1_sp))
    set_param('ControlLabExp/y2_sp','Value',num2str(y2_sp))

    % Only for decoupling: Assign values to decoupling blocks:
    set_param('tcp_server/K2','Gain',num2str(data2))
    set_param('tcp_server/K3','Gain',num2str(data3))

    % Run the simulink model for one step:
    set_param(gcs, 'SimulationCommand', 'step');

    % Get updated input variables:
    u1=out.input1.data(end,:);
    u2=out.input2.data(end,:);
    u = [u1,u2];

    % Send new input data to python
    fwrite(s, jsonencode(u)); % encode data using json
    count=count+1;
end
fclose(s);
set_param(gcs,'SimulationCommand','stop');
```

```
%% Matlab code for controlling actual plant with MPC, with
   TCP/IP connection to Python

        This code receives measurements from Python,
        and updates input values through one step of
        simulink simulation with the MPC.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Define system model: Linear transfer function model

g11 = tf( 0.35, [5 1], 'IOdelay', 5.0,'TimeUnit','seconds');
g12 = tf( -0.1, [5 1], 'IOdelay', 5.0,'TimeUnit','seconds');
g21 = tf( 1, [0.001 1], 'IOdelay', 0,'TimeUnit','seconds');
g22 = tf( 1, [0.001 1], 'IOdelay', 0,'TimeUnit','seconds');


sys = [g11,g12;g21,g22];

sys.InputName = {'Flow dye', 'Flow water'};
sys.OutputName = {'Concentration dye', 'Total flow'};

%% MPC tuning parameters

P = P; %Prediction horizon
M = M; %Control horizon
Q = {[Q_y1,0;0,Q_y2]}; % Setpoint tracking weights
R = {[R_Du1,0;0,R_Du2]}; % MV rate weights

 %% Construct MPC
dt = 1; %Sample time [s]

MPC_lab = mpc(sys,dt,P,M);

%Include hard constraints on MV rates
delUmax = 150;
MPC_lab.MV(1).RateMin = -delUmax;
MPC_lab.MV(2).RateMin = -delUmax;
MPC_lab.MV(1).RateMax = delUmax;
MPC_lab.MV(2).RateMax = delUmax;

%Assign weights
MPC_lab.Weights.OutputVariables = Q;
MPC_lab.Weights.ManipulatedVariablesRate = R;
MPC_lab.Weights.ECR = ECRweight;

%% Start and pause simulation, waiting for next input
```

```
open_system('MPCsim_lab')
sim_time = 600;
set_param(gcs,'SimulationCommand','start','SimulationCommand','pause');

%% Open server, wait for client to connect
s = tcpip('0.0.0.0', 9998,  'NetworkRole', 'server');
fopen(s);

%% Main loop

all_data = [];
count=0;

while count<sim_time
    while(1) % Loop until Python sends readable data
        nBytes = get(s,'BytesAvailable');
        if nBytes>0
            break;
        end
    end
    command = fread(s,nBytes); % Read binary as str

    data=str2num(char(command')); % Tranform str to numerical

    all_data = [all_data;data]; % Store history data
    if isempty(data)
        data=[0,0,0,0];
    end

    % Measured data from plant:
    y1_m = data(1);
    y2_m = data(2);
    y1_sp = data(3);
    y2_sp = data(4);

    % Set paramters in simulink model using the measured data
      received from python:
    set_param('MPCsim_lab/y1_m','Value',num2str(y1_m))
    set_param('MPCsim_lab/y2_m','Value',num2str(y2_m))
    set_param('MPCsim_lab/y1_sp','Value',num2str(y1_sp))
    set_param('MPCsim_lab/y2_sp','Value',num2str(y2_sp))

    % Run the simulink model for one step:
    set_param(gcs, 'SimulationCommand', 'step');
```

```
    % Get updated input variables:
    u1=out.input1.signals.values(end,:);
    u2=out.input2.signals.values(end,:);
    u = [u1,u2];

    % Send new input data to Python:
    fwrite(s, jsonencode(u)); % encode data using json
    count=count+1;
end
fclose(s);
set_param(gcs,'SimulationCommand','stop');
```

```
# ================================================================
# Python code for running pumps, with TCP/IP connection to Matlab
# ================================================================


"""
_____________________________________________________________________
GETTING SETUP READY:
    Starting pumps + bus
    Starting spectrometer
    Setting parameters
    Take reference and dark measurements
    Take reference measurements for nominal concentration
_____________________________________________________________________
"""
#_____________IMPORT PACKAGES_____________________________________

import socket
import time
import json
import numpy as np
import sys
import pandas as pd
import scipy.io
import os
import sys
import time
import AutoTestingFlowConc

sys.path.append("C:\\Program Files\\Ocean Insight\\OceanDirect SDK\\appl
from OceanDirectAPI import OceanDirectAPI, spectrometer

#__________INITIATE PUMPS_____________________________________________

os.chdir("C:/Users/Lab/.spyder-py3")
pump = AutoTestingFlowConc.runningPumps()

pump.s1_openConfig()
pump.s2_busStart()
pump.s3_deviceLookUp()

#___________INITIATE SPECTROMETER_____________________________________

od = OceanDirectAPI()
```

```
ids = od.get_device_ids()
device = od.open_device((ids[0]))

#____________SET SYRINGE PARAMETERS_____________________________________

diam_inner = 14.5673
piston_stroke = 60

pump.chooseDevice(0)
pump.s4_setSyringeConfig(diam_inner,piston_stroke)
pump.s5_setUnits()
pump.s6_enablePump()

diam_inner = 14.5673 #23.0329
piston_stroke = 60

pump.chooseDevice(1)
pump.s4_setSyringeConfig(diam_inner,piston_stroke)
pump.s5_setUnits()
pump.s6_enablePump()


#__________TAKE REFERENCE MEASUREMENT FOR CALCULATIONS_________________

import numpy as np
pump.chooseDevice(1)
pump.generate_flow_without_time(200)
time.sleep(10)
device.set_integration_time(500000)

ref = device.take_reference()
ref = np.array(ref)

#__________TAKE DARK REFERENCE FOR CALCULATIONS_____________________

dark = device.take_dark()
dark = np.array(dark)

import pandas as pd
import matplotlib.pyplot as plt
wavelength = device.get_wavelengths()
ref = ref - dark
ref_avg = pd.DataFrame(ref).rolling(110,min_periods = 1).mean()
ref_avg = ref_avg.to_numpy().reshape(1024,)
```

```
#___REFERENCE MEASUREMENTS FOR FINDING NOMINAL CONCENTRATION______

absorbance_wl = []
time_period_step = 100 #30 seconds
index_wl = 565 #Wavelength = 446.136nm

for sec in range(time_period_step):
    intensity_values = list(device.get_formatted_spectrum())
    intensity_values = np.array(intensity_values) - dark
    intensity_values = pd.DataFrame(intensity_values).rolling(110,min_pe
    intensity_values = intensity_values.to_numpy().reshape(1024,)
    absorbance_values = -np.log10(intensity_values/ref_avg)
    absorbance_wl.append(absorbance_values[index_wl])

m = 0.029456
b = 0.034128

absorbance_wl = np.array(absorbance_wl)
concentration = (absorbance_wl-b)/m
concentration_mean = np.mean(concentration)
conc_dev = concentration-35


y1_nom = concentration_mean


#_________Establishing connection and running main loop__________
"""
_____________________________________________________________________
FUNCTION FOR ESTABLISHING CONNECTION WITH MATLAB AND RUNNING
EXPERIMENT LOOP
        Establish client
        Define nominal steady state values for Simulink
        Open connection by sending steady state values to matlab
        Run main loop
_____________________________________________________________________
"""
#_________Defining step times and experiment runtime_____________

STEP_TIME = 100
STEP_TIME_2 = 250
STEP_TIME_3 = 450
RUN_TIME = 600

#_____________________________FUNCTION_____________________________
```

```
def tcp_sim():
    # Establish client:
        Assign IP address and port number.
        Should be as same as server side (in MATLAB)

    sever_port = ("localhost", 9998)

    try:
        # Create an AF_INET, STREAM socket (TCP):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    except:
        print('Failed to create a socket. ')
        sys.exit()
    print('Socket Created :)')

    sock.connect(sever_port)
    print("start a client")
    time.sleep(0.01)

    #================SS values for Simulink vars================
         Setpoints are deviation variables, therefore 0 is
         steady state

    y1_sp = 0
    y2_sp = 0
    u1_nom = 70
    u2_nom = 130
    y2_nom = u1_nom + u2_nom
    y1_nom = y1_nom

    #=====================SEND DATA TO MATLAB====================
    # Send data to Matlab:

    y1_m = 0
    y2_m = 0
    exp_data = np.array([y1_m,y2_m,y1_sp,y2_sp], float)

    # transform data into string format
    s = str(exp_data)

    # using utf8 to encoding data
    s_1 = bytes(s,encoding='utf8')
```

```
    # send the encoded data to the server
    sock.send(s_1)

    #=========================MAIN LOOP============================

    count = 0

    while count<RUN_TIME:

        time_start_loop = time.time()

        # Receive updated inputs from Matlab:
             Receive the maximum of 1k bytes at one time
             Save data in a buffer

        buf = sock.recv(1000)

        # use json to decoding the data:
        buf_1 = json.loads(buf)

        control_output1 = buf_1[0] + u1_nom
        control_output2 = buf_1[1] + u2_nom

        # Make sure no negative flows:
        if control_output1<0:
            control_output1=0
        if control_output2<0:
            control_output2=0

        #=================UPDATE FLOWS:================================
        pump.chooseDevice(0)
        pump.generate_flow(control_output1)
        pump.chooseDevice(1)
        pump.generate_flow(control_output2)


        #=============GET CONCENTRATION MEASUREMENT:===================
        intensity_values = list(device.get_formatted_spectrum())
        time.sleep(0.01)
        intensity_values = np.array(intensity_values) - dark
        intensity_values = pd.DataFrame(intensity_values).rolling(110,mi
        intensity_values = intensity_values.to_numpy().reshape(1024,)
        absorbance_values = np.log10(ref_avg/intensity_values)
        absorbance_wl = absorbance_values[index_wl]
        m = 0.029456
```

```
        b = 0.034128
        concentration = (absorbance_wl-b)/m

        #================UPDATE SETPOINT PARAMETERS:==================
        # Update if reached a step in setpoint

        y2_m = (control_output1 + control_output2) - y2_nom
        y1_m = concentration - y1_nom

        if count >= STEP_TIME and count<STEP_TIME_2:
            y2_sp = 0.2*y2_nom
            y1_sp = 0
        elif count >= STEP_TIME_2 and count<STEP_TIME_3:
            y2_sp = -0.2*y2_nom
            y1_sp = 0
        elif count >= STEP_TIME_3 :
            y1_sp = 0
            y2_sp = 0
        else:
            y1_sp = 0
            y2_sp = 0

        #================SEND BACK DATA TO MATLAB:====================
        exp_data = np.array([y1_m, y2_m, y1_sp,y2_sp],float)

        # transform data into string format
        s = str(exp_data)

        # using utf8 to encoding data
        s_1 = bytes(s,encoding='utf8')

        # send the encoded data to the server
        sock.send(s_1)

        #================WAIT UNTIL ONE SECOND PAST:==================
        # To ensure sample time of 1 second

        count = count + 1
        time_end_loop = time.time()
        time_dev = time_end_loop-time_start_loop
        wait = 1-time_dev
        if wait>0:
            time.sleep(wait)

    # Close socket object after communication is finished:
```

```
    sock.close()
    pump.stopPumping(stop_all=True)


if __name__ == '__main__':
    tcp_sim()
```

NTNU

Norwegian University of
Science and Technology