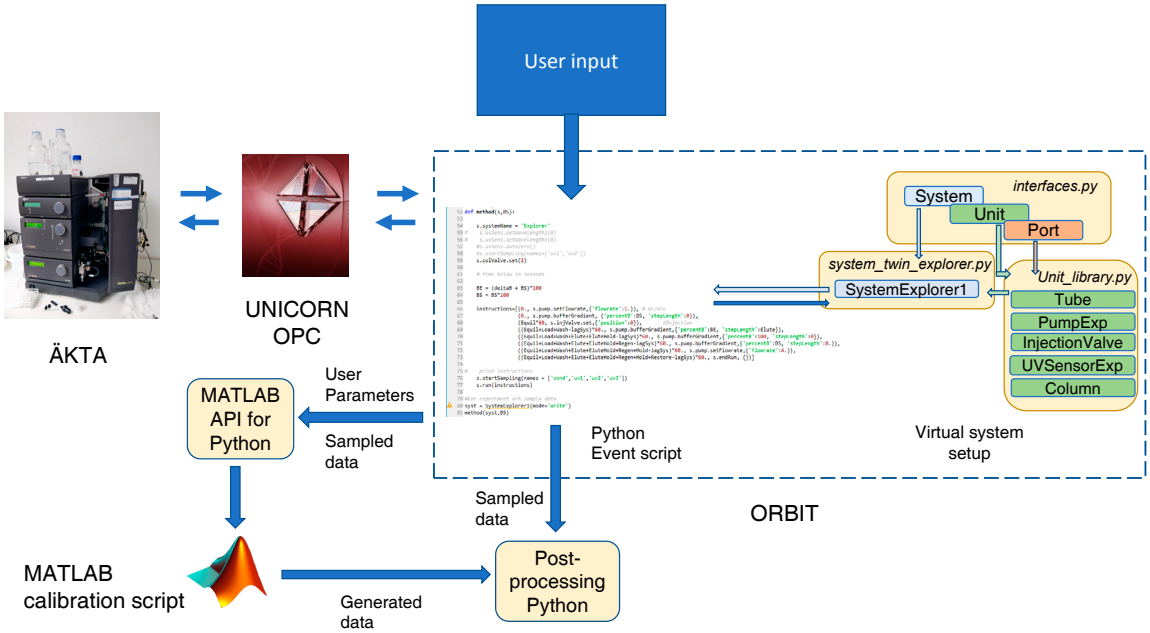
Integration of Modelling Environments to an ÄKTA System



Linus Gustafsson & Daniel Nilsson

Department of Chemical Engineering
Master Thesis 2018

LUND UNIVERSITY

Integration of modelling environments to an ÄKTA system

by

Linus Gustafsson and Daniel Nilsson

Department of Chemical Engineering
Lund University

June 2018

Supervisor: **Associate senior lecturer Niklas Andersson**
Examiner: **Professor Bernt Nilsson**

Postal address
P.O. Box 124
SE-221 00 Lund, Sweden
Web address
www.chemeng.lth.se

Visiting address
Getingevägen 60

Telephone
+46 46-222 82 85
+46 46-222 00 00
Telefax
+46 46-222 45 26

Preface

We would like to thank Professor Bernt Nilsson for the opportunity to work on this particular subject as it was our goal to be able to couple programming and chemistry in our master's thesis. We would also like to thank Associate senior lecturer Niklas "The Fire Extinguisher" Andersson, Simon Tallvod and Dennis Bogren for guiding us through laboratory chromatography which was new territory for us and with several pitfalls where we needed direction. Special thanks are in order to Simon Tallvod for setting up a first application of the Matlab API for Python which we could build upon. Another special thanks also go out to our roommates Ameer Shareef, Viktor Åberg and Dennis Bogren for providing a comfortable environment during the workdays.

This master's thesis has been a collaboration between Linus Gustafsson and Daniel Nilsson carried out from January to June, 2018 at the Department of Chemical Engineering at the Faculty of Engineering, Lund University, Sweden.

Abstract

Automated information flow is an important way of optimizing the time used for analyzing processes. Other added benefits are that in a well-made automated system, the probability of human error diminishes because less liability is put on the person to manually transport and utilize the data. This notion of automation can be applied to the field of chromatography.

This thesis investigates the possibility of integrating modelling softwares (Matlab and JModelica.org) into the control software of a chromatography system for an automated information flow. This integration can be utilized for automated calibration and optimization of the process which will save a lot of time and work with data handling.

The software integration has proved successful and data can be passed from the control software to the modelling softwares automatically. The main control center is Python which controls the events and extracts data from the chromatography system as well as calling the modelling software with defined input parameters. Matlab has been utilized to make an Experiment – Calibration – Optimization (ECO) scheme and JModelica.org has been used to simulate the chromatography process. The ECO scheme has not been fully automated, which is the next step to take where multiple experiments are conducted and the best result is chosen to proceed in the ECO scheme making it possible to automate.

Sammanfattning

Automerat informationsflöde är ett viktigt sätt att optimera tiden som används för att analysera processer. Andra fördelar med väl gjorda automerade system är att chansen för mänskliga fel minskar på grund av att det leder till mindre ansvar för användaren att manuellt överföra och använda datan. Idén av automation kan även appliceras på kromatografi.

Denna avhandlingen undersöker möjligheten att integrera mjukvara för modellering (Matlab och JModelica.org) i mjukvaran som kontrollerar ett kromatografisystem för att automatisera flödet av information. Denna integreringen can användas för automatisk kalibrering och optimering vilket reducerar tiden och arbetsbördan som är kopplad till datahantering.

Integreringen av mjukvarorna har givit positiva resulted och datan kan skickas från mjukvaran för kontroll till modelleringsmjukvaran med automatik. Mjukvaran som används för control är Python som kontrollerar event och extraherar data från kromatografisystemet och kallar på modelleringsmjukvaran med definierade parametrar. Matlab har används för att göra en Experiment – Kalibrering – Optimering (ECO) plan och JModelica.org har använts för att simulera kromatografimodellen. ECO planen har ej fullt automatiserats vilket är nästa steg i utvecklingen för autmation. För att möjliggöra automatiseringen måste ett flertal experiment göras i experimentfasen där det mest troliga experimentet väljs för att fortsätta i ECO planen.

Table of Contents

1 Introduction

Chromatography is a commonly used separation technique in the biopharmaceutical industry for both the production and analysis of proteins. In the chromatography process mixtures of proteins is introduced to a mobile phase which flows through a column packed with stationary phase. The proteins in the mobile phase has different affinity for the stationary phase and will be either trapped or slowed down with a degree that depends on the affinity. In this thesis, ion exchange chromatography has been utilized where the proteins affinity depends on the ion interactions between the protein and the stationary phase [1].

The ÄKTA system is coupled hardware that regulates a column chromatography system and interacts with the associated software Unicorn, which is used to control and visualize the system [2]. There is a supplement to Python called Open OPC that communicates with Unicorn so that the ÄKTA-system can be controlled in the Python environment.

In Python an object oriented package called Orbit has been created to be a virtual copy of Unicorn that makes it possible for Unicorn to run multiple operations simultaneously [3]. If the Matlab API for Python is installed, the Matlab engine can be called upon in Python for simulations and optimizations of generated data from the ÄKTA-system or data simulated by Matlab itself.

There is also an object oriented program called JModelica.org developed in Lund, that defines different objects with help of equations. JModelica.org uses the Modelica syntax to create these objects, but the object models can be used from a Python environment. From Python, JModelica.org can be called for simulations using Functional Mock-up Interface [4].

The main purpose of this thesis is to establish an environment in Python that enables simultaneous runs of ÄKTA-system, MATLAB and JModelica.org where the simulated models give reliable outputs that represent reality. To validate the use of the connections, an Experiment-Calibration-Optimization scheme will be set up using the Matlab API for Python and experimental data obtained from the ÄKTA system. JModelica.org will also be tested by simulating the chromatography model using calibrated data.

2 Theory

The theory for this report describes general chromatography principles, how the process is modelled and necessary information about calibration and optimization that has been utilized.

2.1 Chromatography

Chromatography comprises an important group of methods for separation of closely related components in complex mixtures. The components are dissolved in a mobile phase which can be liquid phase, gas phase or supercritical fluid. The mobile phase is then transported through a column or a solid surface containing a non-miscible stationary phase. The components in the mobile phase are then separated depending on the affinity to the stationary phase [5].

2.1.1 Ion-exchange chromatography

Ion exchange chromatography is a chromatography method that involves exchange of charged sample component ions with species on stationary phase containing counter ions. This results in binding of sample components to the stationary phase by electrostatic attraction. To separate components with different retention time a gradient with competing counter ions can be applied resulting in separation based on the affinity of components to the stationary phase. This method of separation is widely used in life sciences as a protein purification procedure [6].

2.1.2 Chromatography cycle

The chromatographic cycle can be described by 4 different steps that can be summarized as Loading, Washing, Elution and regeneration. The different steps of a chromatographic separation with arbitrary time frames can be seen in figure 2.1.

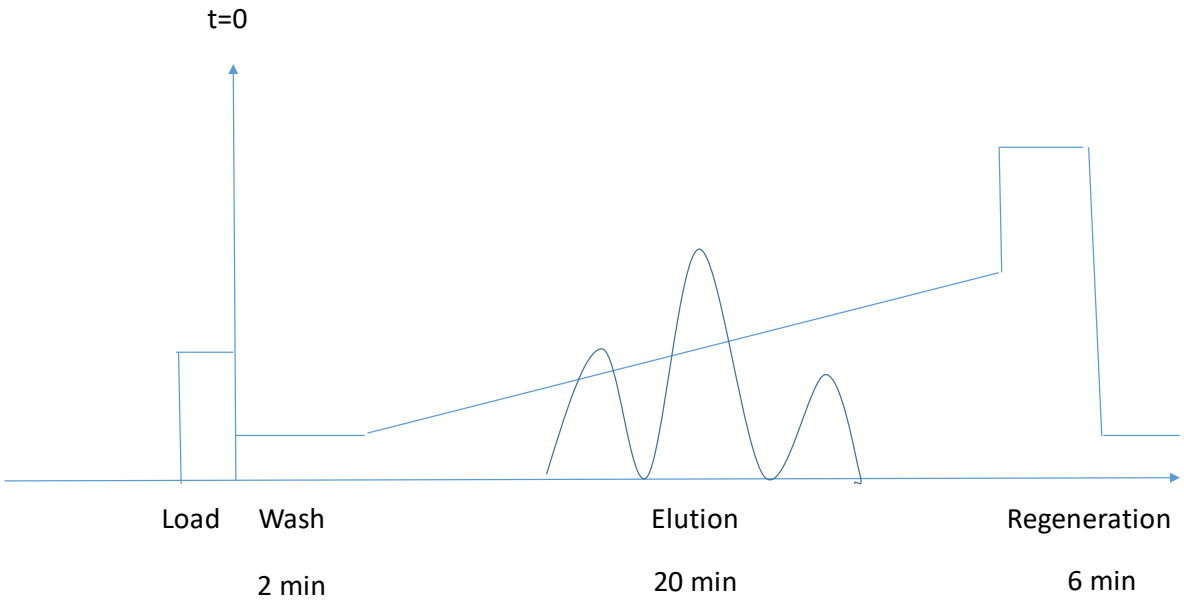


Figure 2.1. Typical chromatographic cycle with loading, washing, elution and regeneration.

In the loading step the feed containing protein mixture is loaded into the column, this is where some of the proteins adsorb.

In the washing step the column is washed with a mobile phase containing a low salt concentration buffer. Here unabsorbed proteins other components with low affinity to stationary phase are washed out.

In the elution step, higher concentration mobile phase is pumped through the column causing the proteins to desorb. Depending on the affinity to the stationary the components will be eluted at different times. The gradient is made mixing two buffers one with high salt concentration and one with low salt concentration.

In the regeneration step pure high concentration mobile phase (the buffer containing high salt concentration) is pumped though the column, this is done to desorb the rest of the proteins which did not desorb during the elution [7].

2.1.3 Chromatography Model

In order to simulate the chromatography process, a model must be brought forth. The general rate model describes how the concentration changes with time by means of convective flow, dispersion, diffusion into the particles and adsorption to particles [8]. This model is commonly used to describe chromatography events.

2.1.3.1 Column Model

A partial differential equation (PDE) can be set up for the change in concentration over time in the column (see Eq. (1)). It accounts for dispersion and convection in the mobile phase as well as adsorption on the stationary phase surface [8].

$$\frac{\delta c_i}{\delta t} = D_{ax}\frac{\delta^2 c_i}{\delta z^2} - \frac{F}{A*e}\frac{\delta c_i}{\delta z} - \frac{1-e_c}{e}\frac{\delta q_i}{\delta t} \quad (1)$$

D_{ax} is the dispersion coefficient and is estimated by using the Peclét number, N_{Pe} (see Eq. (2)):

$$D_{ax} = \frac{vL}{N_{Pe}} \quad (2)$$

c_i is the spatially and temporally distributed concentration of component i (g/L), x is the axial coordinate in the column (m), F is the flow (m^3/min) in the column, A is the cross section area of the column (m^2), e is the porosity of the packed column (m^3 mobile phase/m^3 column), L is the length of the packed column (m), N_{Pe} is the Peclét number (advective/diffusive transport rate), e_c is the column porosity outside of the packing (m^3 mobile phase/m^3 column outside packing) and q_i is the concentration of adsorbed component i on the packed bed (g/L).

The porosity of the column can be calculated from the column porosity and the particle porosity, e_p (m^3 mobile phase/m^3 stationary phase) (see Eq. (3)):

$$e = e_c + (1 - e_c)e_p \quad (3)$$

In this thesis, the column porosity and particles porosity were given to us by the department of chemical engineering, Lund University.

The concentration for salt in the column is set up in a similar way to the concentration of a component, but without adsorption present (see Eq. (4)):

$$\frac{\delta c_s}{\delta t} = D_{ax}\frac{\delta^2 c_s}{\delta z^2} - \frac{F}{A*e}\frac{\delta c_s}{\delta z^2} \tag{4}$$

where c_s is the spatially and time distributed salt concentration.

Two boundary conditions have been set up for the column model. At the inlet a Dirichlet condition has been set up, ignoring the concentration change due to dispersion at the inlet (see Eq. (5)):

$$\begin{cases} c(t) = c_b, at\ z = 0 \\ \frac{\delta c_i}{\delta z} = 0\ at\ z = 0 \end{cases} \tag{5}$$

where $c_b$is an estimated inlet concentration of the component (g/L).

The boundary at the outlet is described by a von Neumann condition (see Eq. (6)):

$$\frac{\delta c_i}{\delta z} = 0\ at\ z = L \tag{6}$$

The column model above has been implemented in Matlab, however in JModelica.org the dispersion has been ignored resulting in (see (Eq. (7)):

$$\frac{\delta c_i}{\delta t} = -\frac{F}{A*e} * \frac{\delta c_i}{\delta z} - \frac{1-e_c}{e} * \frac{\delta q_i}{\delta t} \tag{7}$$

2.1.3.2 Adsorption Model

To describe the adsorption of a component onto the column surface the competitive Langmuir adsorption model is utilized. The kinetic Langmuir model (see Eq. (8)) accounts for the adsorption and desorption of a component depending on the salt concentration present. The adsorption of salt onto the column in neglected and thus $\frac{\delta q_{salt}}{\delta t}$ = 0 [8][9].

$$\frac{\delta q_i}{\delta t} = k_{kin,i}(H_{0,i}{c_s}^{-\beta_i}c_i(1 - \sum(\frac{q_j}{q_{max}})) - q_i) \tag{8}$$

$k_{kin,i}$ describes binding kinetics of component i onto the stationary phase, $H_{0,i}$ describes the equilibrium between adsorption and desorption for component i, β_i describes ion-exchange characteristics between the salt and component i, q_j is the amount component j adsorbed onto the stationary phase and $q_{max,j}$ is the adsorption capacity of the column. $1 - \sum(\frac{q_j}{q_{max}})$ accounts for competitive adsorption between the components [9].

2.1.3.3 Discretization

To solve the column model, the PDE model is transformed to a system of Ordinary Differential Equations (ODEs). The column is divided into discrete grid points using the finite volume method and each grid point is represented by an ODE [10]. In Matlab, the second order derivative is approximated using a 3-point central approximation (see Eq. (9)) and the first order derivative is approximated using 2-point backward approximation (see Eq. (10)):

$$\frac{\delta^2 c}{\delta x^2} = \frac{c_{n+1} - 2c_n + c_{n-1}}{h^2} \tag{9}$$

$$\frac{\delta c}{\delta x} = \frac{c_n - c_{n-1}}{h} \tag{10}$$

where h is the distance between each discretization grid point (m).

The JModelica.org model has used a simpler approach to discretization than the above mentioned, where the first order derivatives have been approximated by a series of packed tanks similar to equation 10.

2.2 Calibration

To model the concentration profiles in the column, unknown parameters must be estimated by calibrating the parameters using experiments. But, to calibrate the model, experimental and simulation data need to be comparable and the estimation of the parameters should be evaluated.

2.2.1 Calibration of extinction coefficient

There are two alternatives when it comes to comparing component concentrations between experiments and simulation. The first is converting the experimental UV absorbance signal to mass concentration and comparing the experiment concentration to the mass concentration from the simulation. The other alternative is to do the reverse and convert the mass concentration in the simulation to UV absorbance and compare it to the UV absorbance from the experiment. In this thesis the first alternative has been chosen and the experimental UV absorbance is converted to mass concentration using Lambert-Beer's law (see Eq. (11)). To use this method Lambert-Beer's law need to be valid [11].

$$A_{280,\mathrm{i}} = e_{280,\mathrm{i}} b c_i \tag{11}$$

$A_{280,\mathrm{i}}$ is the 280 nm UV absorbance measured for component i (AU), $e_{280,\mathrm{i}}$ is the mass extinction coefficient corresponding to 280 nm UV absorbance for component i (L/(g*cm)), b is the length of the measurement cell (cm) and c_i is the concentration of component i (g/L).

2.2.2 Parameter calibration using lsqcurvefit

When making the kinetic Langmuir model for the chromatography column, important parameters which are unknown must be considered and depending on how much is known about the system, the model can be viewed in different ways. There are three ways a model for parameter estimation can be defined [12]. The Black-box model, the White-box model and the Grey-box model. The Black-box model refers to a model that is built largely from measured data using model parameters and/or structure estimation techniques. On the other hand, the White-box predefined structure and parameters, and there is no need of tuning. Grey-box model is a model with predefined structure with needing of parameter tuning [12].

2.2.3 Evaluation of the quality of estimate

When parameters are estimated the question that arises is if the quality of the estimates are good. There are different ways to assess the fitting, they are based on either statistical properties of the estimate or on the statistical properties of the residuals [12].

To evaluate the estimations, data is generated with the estimated parameters using the chromatography model and compared to the real data graphically. If the fit is visually viable, the standard error is evaluated and used to get a confidence interval of the fitted data [12]. The standard error (see Eq. (20)) and confidence interval for estimated parameter is calculated (see Eq. (21)) and for the covariance of the error used to calculate the standard error see figure 2.2.

Assume
-White measurement error
-correct model structure

$$y = y^{\text{true}} + \varepsilon$$

$$y^{\text{true}} = y^{M} = Xp$$

Then parameter Estimation is normally distributed

$$E[p^M] = p \; and \; Var[p^M] = \sigma^2 (X^T X)^{-1}$$

Estimated covariance of the error is based on the sum of square error , Q

$$\sigma^2 = \frac{Q}{N - P}$$

N= number of data points and P = number of parameters

Figure 2.2. Covariance of the error calculated based on sum of square error, Q.

lsqcurvefit generates the sum of square error Q which is used to calculate estimated covariance of the error, standard error and confidence interval for calibrated parameters.

$$s_{pi} = \sqrt{\sigma^2 (X^T X)_{ii}^{-1}} \tag{20}$$

$$p_i \pm s_{pi} T_{inv}\left(\left(1 - \frac{a}{2}\right), N - P\right) \tag{21}$$

2.3 Optimization

Optimization is done on the process to make each process cycle as efficient as possible with regards to time, production, purity or other quantities estimation the efficiency of the process.

2.3.1 Pooling

The whole purpose of chromatography is to extract product from the column which is richer in desired components than the composition of the feed solution. This is also known as pooling, and the extracted product is called the pool [7].

One of the most important objectives of chromatographic separation is the purity of the pool, so a strict constraint is that the purity reaches a specified goal [8]. The purity itself is not a sensitive objective, since less solution can be pooled to accommodate for the purity constraint, which is why other objectives are needed to measure the effectiveness of the process. This is where the yield or the productivity of the specific component can be used as objectives if the purity constraints are met [13].

2.3.2 Productivity and Yield

The yield, Y, in question is the percentage of theoretical yield which is defined as the amount of component of interest that was pooled during the experiment, m_i, in relation to the amount of the component that was injected to the system (the maximum amount that can be pooled), m_0 (see Eq. (23)) [8]:

$$Y = \frac{m_i}{m_{i,0}} \tag{23}$$

The productivity, P, is defined as the amount of the component that has been pooled, m_i, per volume of the pool, V, and the time for the chromatography cycle, t_{cycle} (see Eq. (24)) [8]:

$$P = \frac{m_i}{V * t_{cycle}} \tag{24}$$

2.3.3 Multi component optimization

Optimization of a chromatographic process is done by maximizing a vital quantity related to the pooling of a desired component [8]. To maximize this quantity, a minimization method is used to minimize an objective function, which in turn maximizes the specified quantity.

The objective function that will be used in this thesis is a weighted sum of productivity (g/(L*min)) and yield (see Eq. (22)).

$$Q = -\left(w \frac{P}{P_{max}} + (1 - w)Y\right) \tag{22}$$

w is the weighting factor between 0 and 1 which correlates how important the respective objective is to the user. The productivity is divided with its maximum achievable value, P_{max}, to scale it to a range of 0 to 1 for it to be comparable with the yield.

3 Materials and Methods

The materials and methods section of this thesis will cover the chromatography system, the different softwares involved, which chemicals are used, how calibration was done and developed into an auto calibration script and how the ECO scheme is applied.

3.1 ÄKTA system

The ÄKTA chromatography system is a GE Healthcare Life Sciences product for the purification of proteins and the ÄKTAexplorer model used in this thesis can be seen in figure 3.1 [2].

Figure 3.1. Photo of the ÄKTAexplorer system used in this thesis

On top of the system stands two large flasks of salt buffer that are pumped into the system. The leftmost flask is buffer A comprised of a low concentration salt solution. The right flask is buffer B which is a salt solution with higher concentration. Buffer A is used as the bulk fluid through the system while buffer B is used as an eluent to desorb solutes from the stationary phase. The buffers are connected to one piston pump each, pump A and pump B, to pump them through the system. These pumps are set to a total flowrate in the system and a ratio splitting the pump load between the two pumps (e.g. 70% pump A and 30% pump B of 1 mL/min total flowrate). After the pumps, the flows join again and enter a T-cross and a flow restrictor that maintains even flow through the system. When the flow has been steadied it enters an injection valve (V1) with 7 ports and the configuration of ÄKTA valves can be seen in figure 3.2.

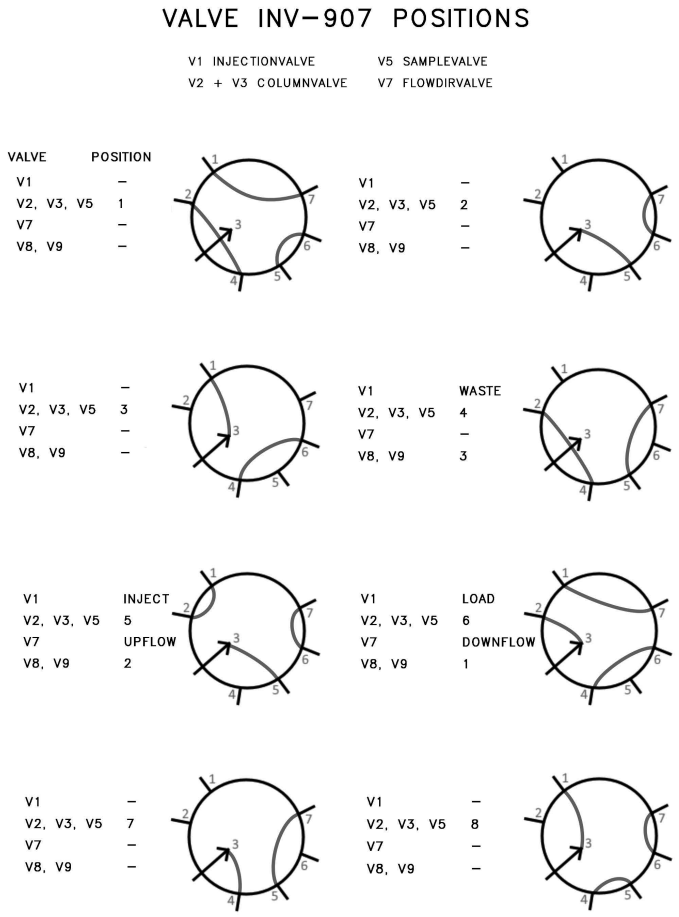


Figure 3.2. Different ÄKTA valves and their configurations

The valve is fitted with a 0.5 mL injection loop where analyte can be injected into the buffer stream for analysis and separation. The injection loop is overfilled with analyte when injected and the overflow ends up in the waste so that the specific volume of analyte is known. The buffer with analyte then enters a column valve (V2) which chooses if the upcoming column should be bypassed or not. The output from the column will enter another column valve (V3) which decides whether to send the solution to the analysis instruments or directly to the waste. The chain of analysis instruments starts of with UV monitoring cell to measure the UV absorbance of the solution to quantify the analytes during the cycle. The range of UV wavelengths possible is 190-700 nm, but this thesis has chosen to use 280 nm UV absorbance [14]. After the UV cell comes the conductometer used for the detection of dissolved ions in the solution. The conductivity measured is used to visualize the salt gradient that is present for elution of the analytes. The analyzed solution enters a final sample valve which decides if the solution should be sampled, recycled, used onto another column (if present) or go to the waste. A complete flowsheet of this ÄKTA system setup can be seen in figure 3.3.

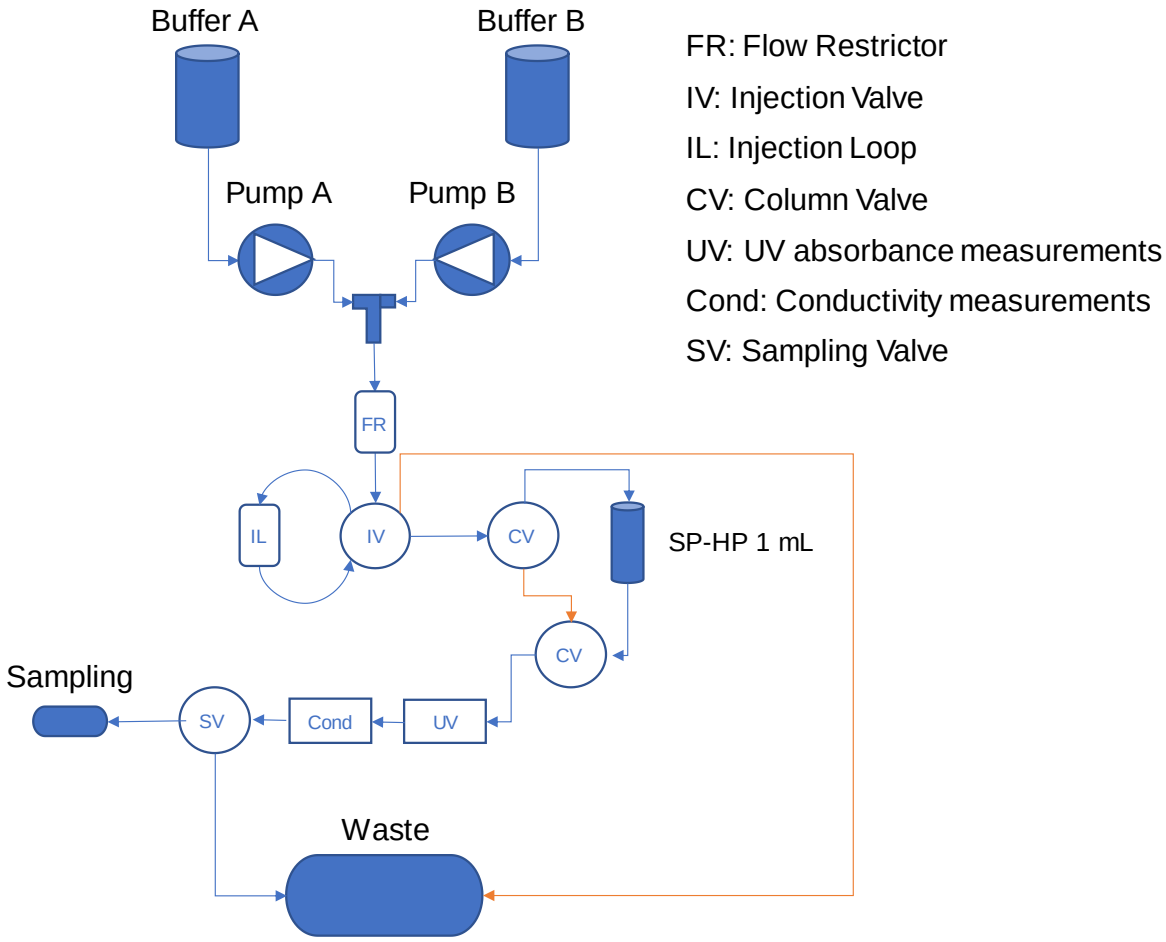


Figure 3.3. A flowsheet of the current ÄKTA system configuration. The orange lines are alternative pathways in the system.

3.2 Unicorn

To visualize and control the ÄKTA chromatography system, an accompanying software known as Unicorn is utilized. The program is built around 4 modules: system control, method editor, evaluation and administration. In this thesis, mainly the system control has been utilized. In the system control interface pumps, valves, measurement instruments etc. can be regulated, a real-time chromatogram for UV signals, conductivity and buffer ratio set points can be seen, a schematic of the ÄKTA system is visible and an action history window can also be viewed [15]. The real-time chromatogram which is displayed in the system control is heavily utilized in this thesis to catch disturbances and to early evaluate the data generated.

3.3 Orbit control script and MATLAB API for Python

The chromatography ÄKTA system is controlled by the software Unicorn where valves, pumps, measurement instruments etc. can be regulated. The Unicorn software has limitations and can be bypassed by using what is known as the Orbit controller. The Orbit controller can be used by importing the packaged orbit system, that represents the specific ÄKTA system that is going to be used, into Python. The accompanying virtual system from the Orbit package regulating pumps, valves and instruments can be used to specify the chromatography cycle in an event script in Python.

The data from the ÄKTA system can be sampled from the machine and used to generate plots, calibrate parameters or optimize the cycle. In the Python script, the Matlab engine package can also be imported and used to enable the Matlab API for Python and send information between the two programs. Matlab is called in the script to simulate the salt and component concentrations in the chromatography cycle and can also use the experimental data to calibrate Langmuir adsorption reaction parameters. In general, the script consists of 4 parts: parameters, ÄKTA system method, Matlab call and post-processing. A schematic of the information flow can be seen in figure 3.4.

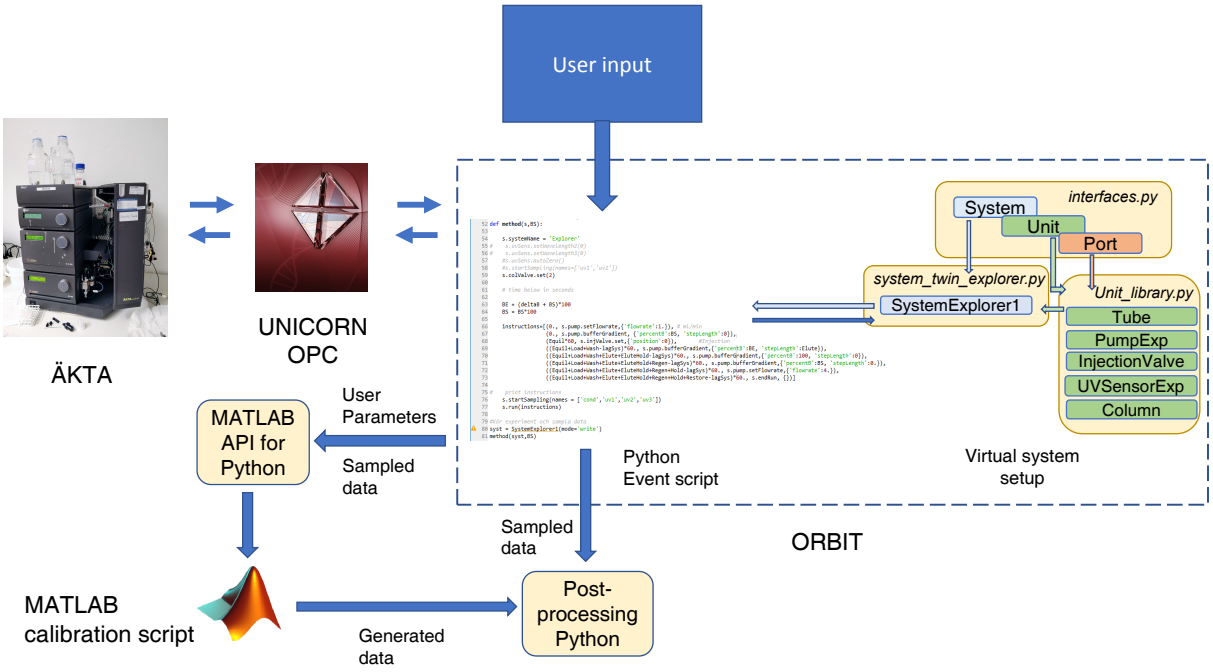


Figure 3.4: An overview of the exchange of information in the control system

The schematic above, possibly with slight variations, is the main method how the results from the chromatography experiments are generated. The idea is to define parameters and generate or load in data from the Python environment which can be sent via the API to Matlab for simulation, calibration or optimization.

3.4 JModelica.org

JModelica.org is an open source Modelica-based platform developed for industrial simulation and optimization. The program was developed at the Department of Automatic Control, Lund University and is currently maintained and further developed by Modelon AB with academia collaboration [16]. Modelica is an object oriented, equation based language used to create models for use by JModelica.org. Along with the Functional Mock-up Interface (FMI) standard and the Modelica Standard Library, JModelica.org can simulate the Modelica model in a Python environment. Different Modelica objects that are created can be linked together in JModelica.org which can be a good way to build a complete process. If optimization or parameter calibration is desired, the Optimica extension can be utilized by specifying an optimization problem which Optimica translates and solves with numerical algorithms [4].

In this thesis, only simulation of the chromatography cycle on the column using FMI and Modelica Standard Library has been used. Since there was no previous experience of Modelica programming going into this work, OpenModelica video tutorials were utilized to learn the basics of Modelica [17]. OpenModelica is another open source Modelica-based platform, but there is an advantage for beginners using this program. The program has a built-in Interactive Development Environment (IDE) where you type in and simulate your Modelica code, whereas in JModelica.org, a separate .mo file must be created and translated with FMI in Python and simulated from there. There is nothing directly disadvantageous with the path JModelica.org takes, however it can be hard to grasp at first without even knowing if the Modelica code has been implemented correctly. The Modelica model developed in OpenModelica can also be used as a .mo file and implemented in JModelica.org.

3.5 Chromatography phases and Protein analytes

The mobile phase in the experimental chromatography setup consists of buffer A, buffer B and varying analytes depending on the analyte is of interest. The eluent of the mobile phase, i.e. buffer A and buffer B, both contain 0.05 M sodium phosphate, which is a buffering system that keeps the buffer pH at 7. However, buffer B has a higher total salt concentration of 0.5 M where sodium chloride has been added to make up the 0.5 M concentration. Buffer B is used to interact with the ion-exchange adsorption of the analytes in the column and buffer A is used as a bulk fluid for transport and dissolution.

The column utilizes ion-exchange and is packed with SP sepharose, a strong high performance cation exchange resin, also known as a HiTrap SP-HP column [18].

The analytes that will be studied in this thesis is ribonuclease A, cytochrome C and lysozyme. When an analyte solution consists of all the analytes above, it can be referred to as an RCL solution. Ribonuclease A from bovine pancreas is an endoribonuclease capable of splitting RNA into smaller pieces and is used to remove RNA from preparations of plasmid DNA [19][20]. Cytochrome C from equine heart is a mitochondrial protein for electron transport and is used as an indicator of cell apoptosis [21] which is a mechanism for cell self-termination [22]. Lysozyme from chicken egg white is a glycoside hydrolase which catalyzes the hydrolysis of bacteria cell walls. Lysozyme is used for bacteriolysis, preparation of protoplasts, as a flavor enhancer in food and drinks, sample preparation when isolating nucleic acids, anti-inflammatory on the mucous membrane, tissue healing, protection against bleeding and more within pharmacology [23].

According to Sellberg et al., 2017 the elution order of the RCL solution is first ribonuclease A, cytochrome C and last lysozyme [9]. Ribonuclease A has much lower affinity to the stationary phase than the other analytes and will be eluted early in the salt gradient and is not part of the separation problem. Cytochrome C and lysozyme are more similar in affinity which is the main reason why modelling and optimization is implemented for the separation.

3.6 Calibration of extinction coefficient

To convert the absorbance to concentration for comparison, the extinction coefficient must be calibrated for the specific component. This is done by loading a known concentration of the component onto the column and eluting the component with a salt gradient. The extinction factor is calibrated for a specific buffer system, since small differences in buffer type, ionic strength affects absorptivity. For this reason, a self buffering ion solution for Buffer A and Buffer B was used that stabilized at around pH 7. Buffer A consists of 50 mM sodium phosphate

with distilled water and Buffer B consists of Buffer A with natrium chloride added to a concentration of 500 mM [9][11].

By comparing the integrated absorption with the injected quantity an extinction factor could be calculated at given system conditions (see Eq. (12-15)) [11].

$$\int c_i(t)dt \Rightarrow m_i \tag{12}$$

$$c_i V_{loop} = m_i \tag{13}$$

$$\int A_{280,i}(t)\ dt = e_{280,i} b \int c_i(t)dt => \int c_i(t)dt = \frac{\int A_{280,i}(t)\ dt}{e_{280,i} b} \tag{14}$$

V_{loop} is the volume of the injection loop (m^3) and m_i is the mass of component i [g].

Since concentration and volume of injected component is known it is possible to get extinction factor by putting equation 13 equal to equation 14 (see Eq. (15)).

$$c_i V_{loop} = \frac{\int A_{280,i}(t)\ dt}{e_{280,i} b} \Rightarrow e_{280,i} = \frac{\int A_{280,i}(t)\ dt}{c_i V_{loop} b} \tag{15}$$

The limitations of using this method to get the extinction factor is that the injected concentration and volume must be precise.

3.7 Parameter calibration using lsqcurvefit

The model used in this thesis falls under these category Grey-box model with predefined structure and need of tuning of parameters. The values of the model parameters that are predefined can be seen in appendix A. Two parameters were chosen for calibrating the model, since the model depends on Langmuir kinetics the chosen parameters were k_{kin} that denotes a parameter describing binding kinetics [9], and H_0 which is a factor based on q_{max}, the adsorption rate constant, k_{ads}, and the desorption rate constant, k_{des}. Reaction rate depends strongly on these parameters and are crucial for good model representation [24].

$$k_{kin} = k_{des,0} c_s^{\beta} \tag{16}$$

$$H = H_0 c_s^{-\beta} = \frac{k_{ads} q_{max}}{k_{des,0}} c_s^{-\beta} \tag{17}$$

The function used for parameter estimation was lsqcurvefit, a function in MATLAB that is a nonlinear least-squares solver and uses trust-region-reflective algorithm by default which is a subspace trust-region method and is based on the interior-reflective Newton method [25].

$$min \left|\left| \mathrm{F(x, xdata)} - \mathrm{ydata} \right|\right| = \min \sum (F(x, xdata_i) - ydata_i)^2 \tag{18}$$

The first step used in our calibration script was to make sure that the calibration script generated trustworthy data. This test was done by generating data with known parameter values, then lsqcurvefit was used on the model with initial parameter guesses close to the values used to generate data. After confirming that the calibrations script and its algorithms worked, and it

generated almost identical values to the ones given to generate data, the next step was to calibrate with true data from the ÄKTA-system. The order in which the procedure of calibrating and validating the model was done can be seen in figure 3.5 [26].

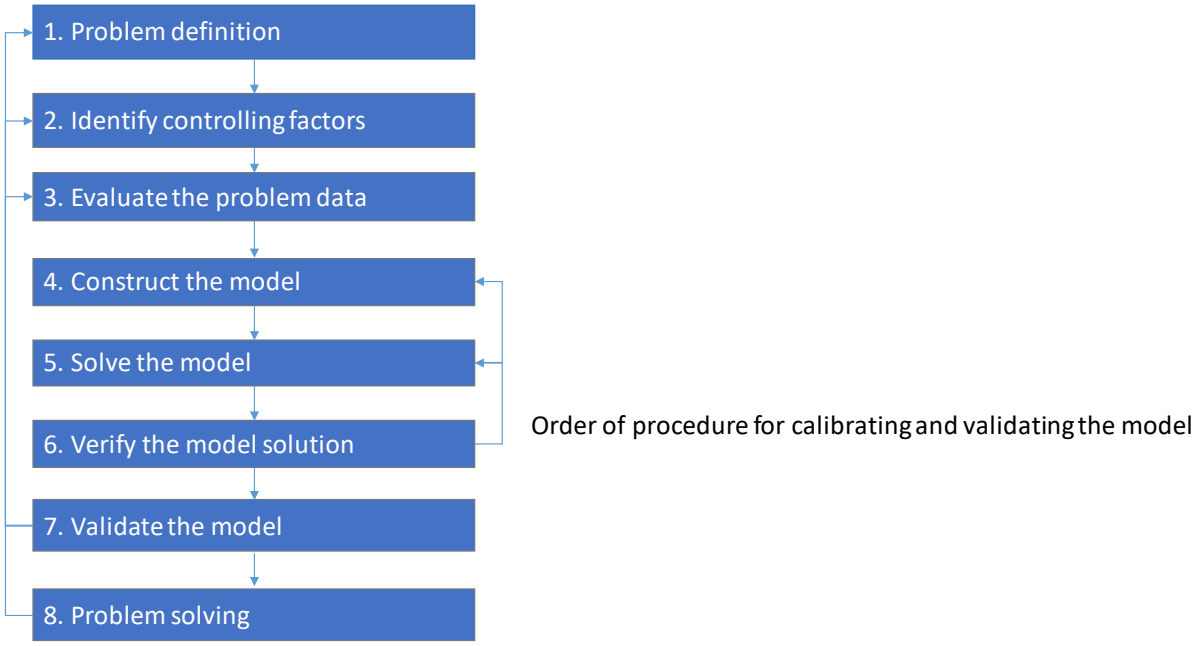


Figure 3.5. Order of procedure for calibrating and validating model.

3.7.1 Data analysis and preprocessing

For good parameter estimation unnecessary data and outliers were removed. The data generated by ÄKTA-system or any real process system are generally of varying quality. This might be due to some malfunction in the measurement device or unexpectedly large disturbances [12]. The advantage of using the ÄKTA-system is that the data that is going to be generated in Orbit can be seen beforehand in Unicorn which makes it easy to remove unnecessary data by choosing data points where the product has been eluted. This selection of data is done by first doing an experiment and observing at which time the component absorbance is measured, i.e. when component is eluted. Once the time interval in which the gaussian absorbance top of a component is observed, data points prior and after the time interval are put to zero.

The generated UV data from the ÄKTA-system is converted into concentration by Lambert-Beer's law and then sent to MATLAB. The generated conductivity data is also converted to concentrations by utilizing the Total Dissolved Solids (TDS) factor. TDS represents the weight of cations, anions and undissociated components in a litre of water and can be used to relate conductivity to the amount of dissolved ions. For water without significant amounts of non measurable content or poorly dissociated ions, the TDS factor is often in the range of 0.55-0.7 g*cm/L*mS [27]. The TDS factor has been set to 0.65 in this thesis for conductivity calculations (see Eq. (19)):

$$c_s = \kappa * \frac{TDS}{M_{salt}} \tag{19}$$

κ is the conductivity data measured from the experiment (mS/cm), TDS is the TDS factor with a value of 0.65 (g*cm/L*mS) and M_{salt} is the molar weight of the measured salt (g/mole). The

molar weight used for the calculations accounts for both the sodium phosphate and the sodium chloride.

3.7.2 Model Parameter calibration

Once the data is chosen, lsqcurvefit is used on the model to calibrate the unknown parameters. lsqcurvefit uses the kinetic Langmuir model and tries to fit the data in a least-square sense, using given guesses as the initial starting point [25]. The fitting is done against the data in the last grid point since it represents the concentration profile of the component when it has gone through the whole chromatography column. With some knowledge of the parameters, constraints can be set up. In lsqcurvefit these constraints are called upper and lower boundary and limit the range in which lsqcurvefit can estimate the parameters. The boundaries are in place to ease the workload for lsqcurvefit and to generate more precise parameters [25].

3.7.3 Model parameter validation

After each calibration of adsorption parameters to a single component experiment, the fit was evaluated using the standard error and corresponding 95 % confidence interval as described in the theory.

To further validate the model parameters, a validation experiment was run with the same conditions as the calibration experiments. A cocktail of 0.5 g/L of ribonuclease A, 0.5 g/L cytochrome C and 0.5 g/L lysozyme was injected onto to column and eluted for the experiment. The calibrated parameters were used to generate a model response which was compared to the validation experiment data.

If the model parameters are fitted well, the simulated data will correspond well with the validation data, in this thesis the validation step was done visually by using estimated parameters on the model comparing to experimental data graphically [26].

3.8 Auto calibration script

Putting everything together it was possible to make an auto calibration script, the same arguments that were used in the instruction to Unicorn are used as arguments to the Matlab script that calibrates parameters. There is some data that was predetermined and put in as constant values in the Matlab script. There is lag time in the ÄKTA system due to transport of buffers and substrate throughout the system that is not simulated in the model. This lag time was calculated and subtracted from the execution of each event in the script i.e. loading, wash and elution. This was an easy way to bypass the problem of having to model the whole system to get a lag time. The instructions are given to the ÄKTA system one lag time before the instruction should appear in the response. The lag time was calculated by making a chromatographic run bypassing the column and subtracting the input time in Unicorn to the response time.

3.8.1 The Auto Calibration Script limitations

- The time where the substrate is eluted must be known to calibrate the extinction factor and to eliminate non-essential data.
- If the data is in bad condition, the calibration will not provide a good fit. Examples of bad data is peak tailing or disturbances in integration range.
- The constant β parameter must be chosen well since it will be constant.
- The kinetic parameters k_{kin} and H_0 need to be relatively well known to be able to make good guesses and make good predefined lower and upper boundaries in lsqcurvefit.

- The auto calibration script will iterate the extinction factor based on the mass injected to the column, thus the volume and concentration of the injection must be very precise or else the model will not yield reliable data.

3.8.2 The Auto Calibration process

The Auto Calibration starts with Unicorn performing instructions in the ÄKTA system making a true chromatography run. The script makes sure to sample the time, conductivity and UV for 280 nm. Once this is done the script stores the time, UV absorption and conductivity in a dictionary with different keys. Once the data is stored the UV and conductivity is converted to concentrations. The conductivity is for the salt gradient and shows the buffering course. UV is used for concentrations of substrate and is calculated by first iterating to get the extinction factor and then converting absorption to concentration with Beer-Lambert's law.

Since there is a relationship between the absorption and the concentration by the Beer-Lambert's law, exists extinction factor can be calculated by iteration (see Eq. (15)). Once the extinction factor is known, the absorption (UV) data is converted to concentration data.

With the concentration and time data from the experiment it is now possible to calibrate with lsqcurvefit. With API it is possible to use Python and Matlab simultaneously, in the Matlab script we are using a dispersion model to simulate the chromatography course. The unknown kinetic parameters k_{kin} and H_0 are calibrated by lsqcurvefit in the Matlab script. The concentration and time data are sent into Matlab as arguments, as well as the experimental conditions and parameters for the lsqcurvefit.

lsqcurvefit will loop through the kinetic Langmuir model and try to change the parameters until the data points are fitted well to the simulated data. lsqcurvefit will then output calibrated parameters, the squared norm of the residuals Q and Jacobians used for standard error calculation. This data is stored in a dictionary and recovered in the Python script, where an assessment condition is put up. With Q, the standard error is calculated and the assessment condition set for the calibrated parameters is for the standard error to be small.

The condition is that if the standard error is above 0.05 for k_{kin} and 10^{-5} for H_0 a message is generated telling the user that a bad standard error for fitting was reached, try to change boundaries or generate better experimental data. Now the user will see a plot from Matlab showing how the simulation looks with the current calibrated parameters against the experimental data. If the experimental data does not look like it can be fitted the user can choose to redo the experiment. If the experimental data looks good, without too much disturbances in the integration range, but the standard error is too high, the user can type into the Python shell a new guess and boundaries of type float. Once this is done the loop will redo the Matlab calibration. This can be done until a good fitting is reached where the standard error condition is met.

When the condition is met, the script now uses calibrated data and simulates the process again, the simulation data is then used for post processing where the simulated data can be compared to the real data. The output from script is then the standard error, mass of injected substrate, mass of simulated substrate, the calibrated parameters k_{kin} and H_0, and the 95% confidence interval for k_{kin} and H_0.

One kinetic parameter that is not calibrated is β, which is given to the model prior to auto calibration. This is due to the hard nature of calibrating three parameters simultaneously and that the β parameters do not vary in the same degree as the chosen parameters. The β used for the

calibrations made in this thesis was taken from Sellberg et al., 2017. A schematic of the information flow in the auto calibration can be seen in figure 3.6.

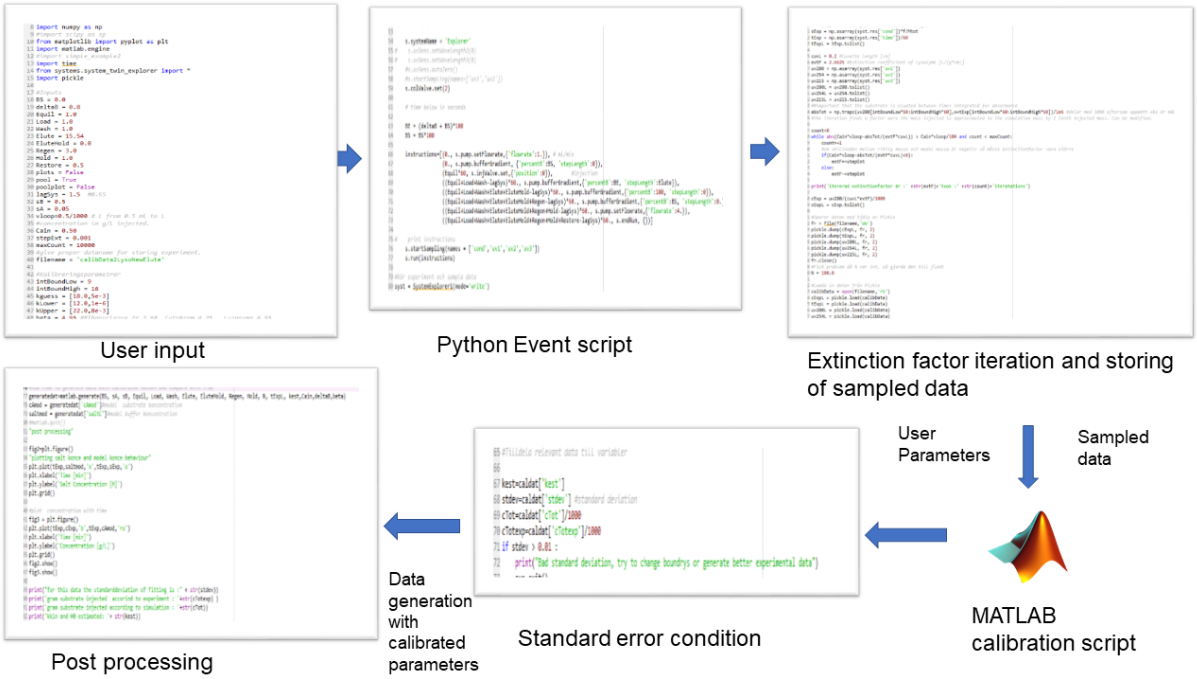


Figure 3.6. An overview of the Auto calibration script

3.9 Optimization

To start off the optimization part, the elution time of a single component run is found to get a specific retention time of the component. This result can be used to get a rough understanding of how elution time affects the retention time of the specified component which can be utilized for multi component elution [28].

The last part of the optimization consists of multi component optimization of elution time using yield and productivity as objectives.

3.9.1 Optimization on retention time

To optimize the retention time, a retention time from a given run needs to be calculated. In chromatography, the chromatographic peaks are gaussian which is due to that the substrates have followed independent paths through the stationary phase [5]. Therefore, the time in which the molecules of the same kind takes to pass through the column varies. However, all the molecules make up for a cohesive zone, with a mean in which the dispersion can be calculated [5]. In order to calculate the retention time for the gaussian top, a method was used called getpeakmax which gives an interpolated max value of the peak. The retention time is a mean of the time it takes for substrates to pass column and is measured at the peak maximum.

To achieve a specified retention time for a component, an algorithm has been created to change the elution time of a model simulation until the components retention time of the simulation is within an error margin of the specified retention time. A bisection search method was created for this purpose as in figure 3.7 which is provided an upper and a lower limit of elution time where the answer can be found. The mean of these limits is sent to a model simulation and

generates a retention time. The difference between the specified retention time and the simulated is calculated and if the simulated retention time is bigger than the specified, then the simulated elution time is set as the upper limit to be used for the mean in the next simulation. Vice versa if the simulated retention time is smaller than the specified retention time. The error criteria is then tested on the difference and if it is not met, than another mean is taken of the new elution time limits and the loop continues. A maximum iterations stopping criteria has also been implemented so that the loop cannot become infinite.

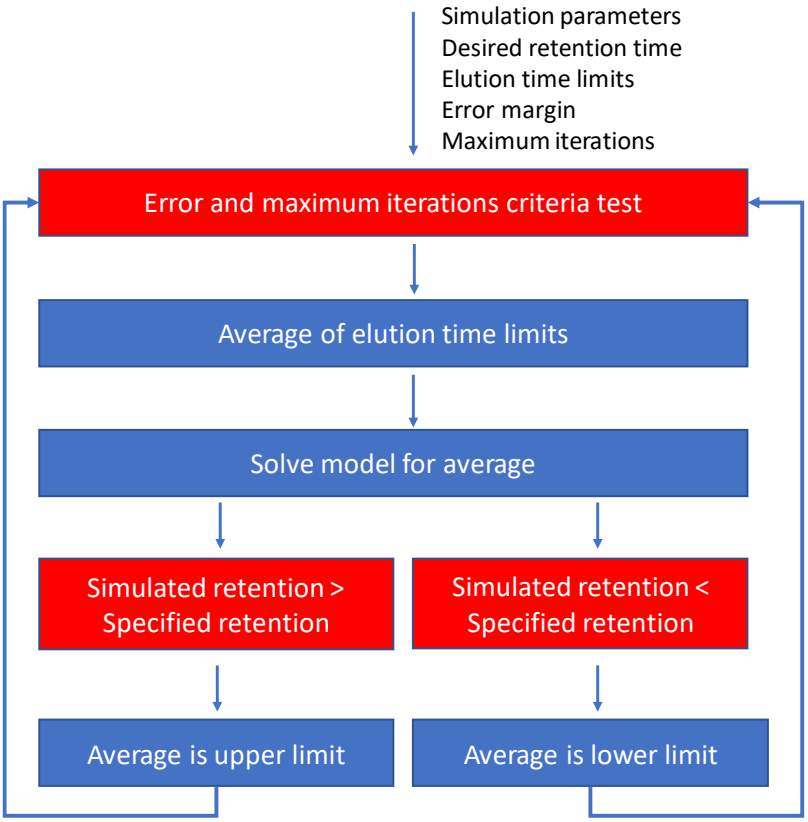


Figure 3.7. A schematic of the bisection search method used to find the elution time for a specified retention time of a component

A simulation of expected elution time to achieve a specific retention time of 10 min for cytochrome C has been performed. The adsorption parameters used in this simulation were calibrated using 15.54 min elution time.

3.9.2 Getpeakmax algorithm

getpeakmax is a script that contains a sequence of instructions, an algorithm that interpolates max value of peak and gives the time where this happens. The max value at peak give the retention time of the substrate.

First it starts by finding the max value of the concentration data using built in function max, in this case the maximum concentration. Then it finds the indices where this value is found.

In the next step it finds the first and last indices where the concentration data is bigger than half the maximum concentration using the built in function find. After those values are found it interpolates between the indices above half the maximum concentration and two indices before/after to find the exact value of half the maximum concentration. The algorithm then creates

a spline interpolation in the half peak height interval using spline function and reverses it to be able to use a minimization algorithm fminbnd to find the exact maximum peak value.

3.9.3 Multiple Component Optimization

When optimizing an objective of a multi component system it is important that the model gives a good representation of the real chromatography process, which means that the system must be calibrated well, or the optimization may not yield any trustworthy results. This means that for every optimization in this thesis, the adsorption parameters must be calibrated for all components with the same system conditions as the optimization. This calibration is done by eluting a single component with the same buffer concentrations and gradient as the optimization script. The calibration procedure is the same as described above in Auto Calibration script.

When the model has been supplied with the calibrated parameters, objectives describing the effectiveness of the pooling can be optimized. To find the objectives for the components in a multi component system, a simplex pooling method in MATLAB is used. The simplex pooling method calculates objectives, such as purity, yield and productivity of the components in the pool, and other data, such as pooling times, concentration in pool and loaded amount, and uses the MATLAB function fminsearch to minimize its internal objective function which will maximize the objective. fminsearch uses the Nelder-Mead simplex method which creates a simplex around the initial guess and modifies the simplex repeatedly until a stopping criteria is met [29].

The chromatography model will be solved and sent to the simplex pooling method which generates an optimal pool that satisfies the purity constraints. This will be done in a loop using an external fminsearch which tries to minimize a defined objective function. The objective function used in this thesis is the weighted sum of productivity and yield described in the theory.

The purity constraints that are specified in the simplex pooling method have been set so that a purity of at least 95 % cytochrome C is found in the pool, with a maximum of 1 % lysozyme. The simplex pooling method will not yield a result if the purity constraints cannot be met, so there will be a minimum elution time that will be a lower limit of the initial guess that the external fminsearch can use. Guessing a higher elution time is safer in fminsearch to avoid local minimums.

The optimization procedure of the weighted yield and productivity objective can be seen in figure 3.8 below where adsorption parameters are calibrated from experimental data and used for optimization. Once an optimal elution time has been brought forth it is then tested in experiments to see if the calibrated parameters will differ. This procedure of experiment, calibration, optimization and validation is referred to as an ECO (Experiment, Calibration, Optimization) cycle in this thesis. As a rule of thumb, the parameters will be significantly different if the elution times used for the two datasets are very different from one another. This is due to the change in system conditions that the elution time dictates.

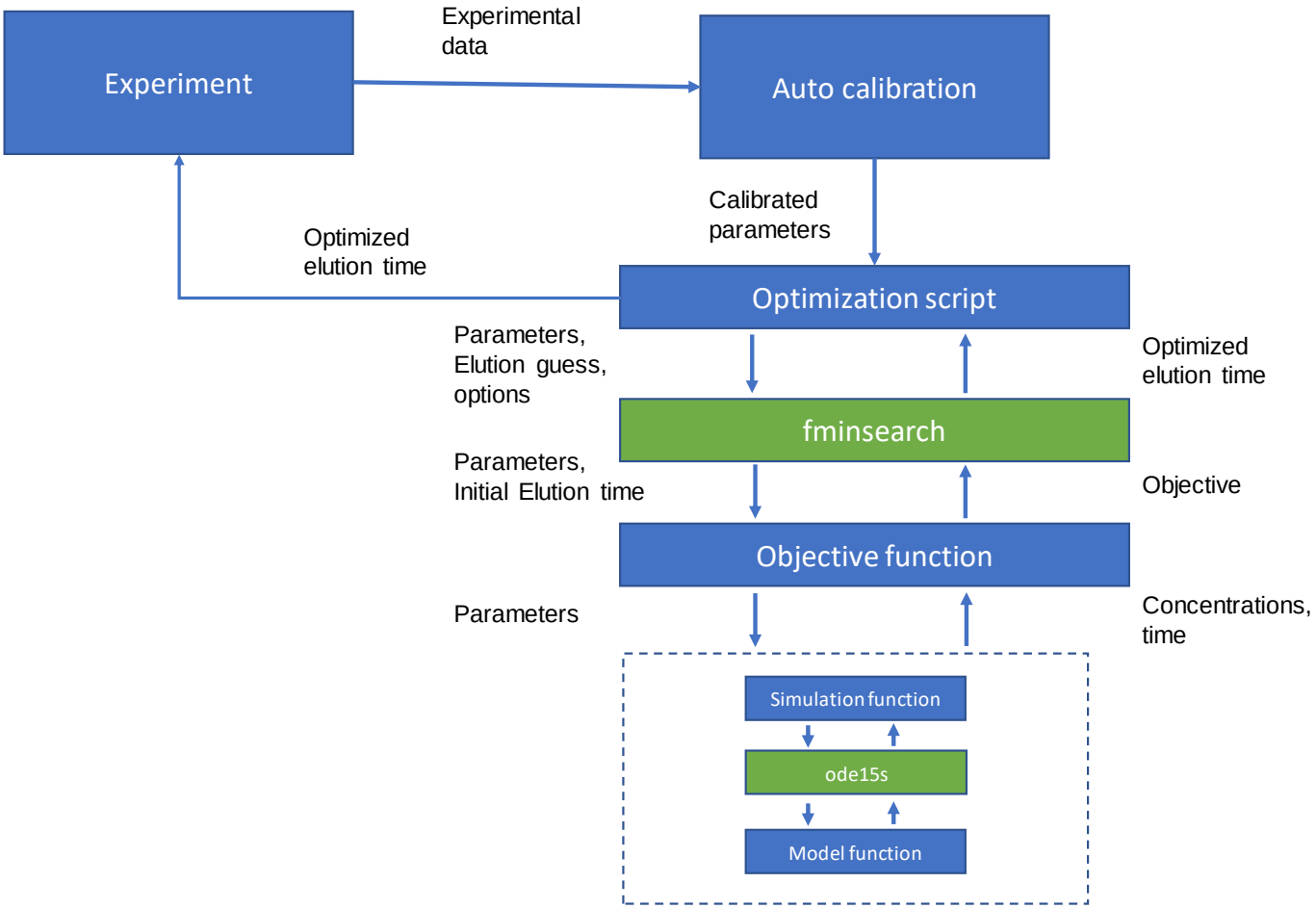


Figure 3.8. An ECO scheme using yield and productivity as the objective to find an optimal elution time.

Optimization using different objective weightings are performed in this thesis. The adsorption parameters for these optimizations have been calibrated for maximum productivity (using a weighting of 1.0) in a previous loop of the ECO scheme. The result of optimizing with these different weightings can be used to determine an optimal objective weighting for the system. This optimal point can show itself as a Pareto optimum, where the yield and the productivity are as high as they can be without sacrificing too much of one another [30].

3.10 JModelica.org simulations

As mentioned before, JModelica.org was used to simulate the chromatography model using predetermined and calibrated parameters. The first thing to start with is to make a model file (.mo) containing the model of the chromatography cycle. This model file is written in Modelica syntax and can either be coded in a text editor such as WordPad or NotePad++, or it can be written in the OpenModelica IDE. The model file created consists of 3 different sections: the assignment section where parameters are assigned values and variables are declared for use, the initial equation section where the initial values of the variables are given to solve the system and the equation section where time events and model equations are defined. An example of the structure in the OpenModelica IDE can be seen in figure 3.9 below where a bouncing ball is simulated. The code for the chromatography model is too long to be included in a good picture, so another example was chosen.

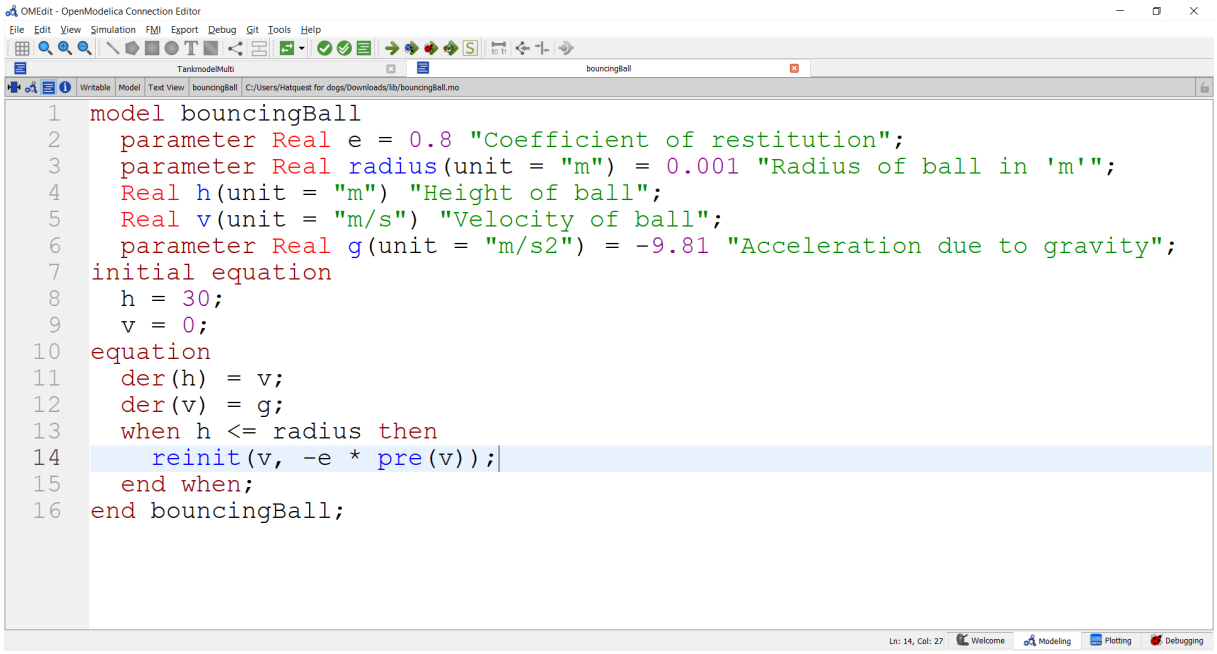


Figure 3.9. An example of the structure of a Modelica model file.

When the model file is functioning, which can be tested by simulating it in OpenModelica, it is time for the model file to be compiled into a Functional Mock-up Unit (FMU) using the Functional Mock-up Interface (FMI) included with JModelica.org. A Python script can be created for this purpose where the compile_fmu and load_fmu package is imported from the JModelica.org folder and used on the model file. When the FMU of the model file is loaded, the FMU has built-in functions for simulation which can take input parameters to specify the settings of the simulation such as simulation time and solver options.

From the simulation, the data can either be extracted for use outside of JModelica.org such as plotting in Python or further processing in Matlab, or the internal plot GUI can be used to visualize the simulation. A schematic of the information flow using JModelica.org can be seen in figure 3.10.

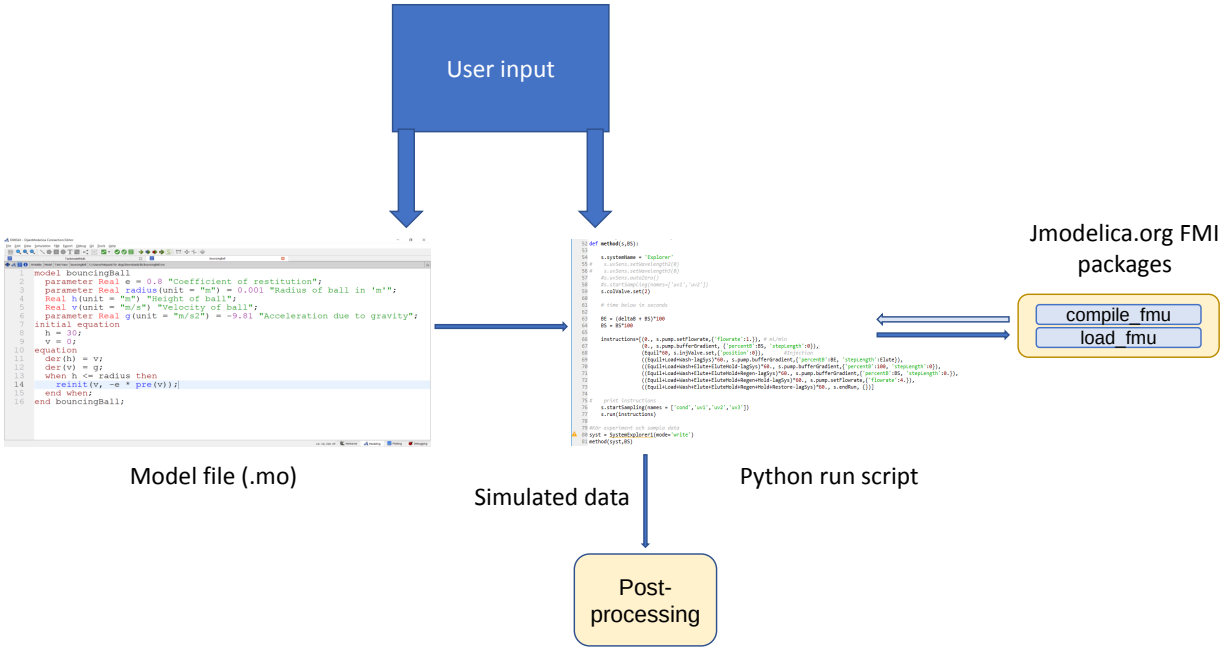


Figure 3.10. A schematic of the information flow using JModelica.org.

A JModelica.org chromatography simulation has been performed in this thesis which is compared to a similar Matlab simulation to test the different software and models against one another. Due to the dispersion being left out of the JModelica.org model there should be some difference between the models, but mostly the simulations are done to show that JModelica.org can be used for this type of simulations

4 Results

Worth mentioning for the results is that an initial loop of the ECO scheme has been done using an elution time of 8 minutes and the optimization, using a weight factor of 0.9, yielded the optimal elution time 15.54 minutes which is used as a basis for the results. So the results given below is for a second loop in the ECO scheme.

4.1 Autocalibration results

4.1.1 Component adsorption parameter calibrations

Auto calibration on kinetic parameters was done using kinetic Langmuir model. The auto calibration was done on lysozyme, cytochrome C and ribonuclease A individually with a concentration of 0.5 g/L to get the kinetic data for k_{kin} and H_0. The injection volume was 0.5 ml and elution time used for calibrated was 15.54 minutes. The elution time was chosen after the optimization part. Experimental data are compared to simulated data using calibrated parameters and can be seen for cytochrome C in figure 4.1.

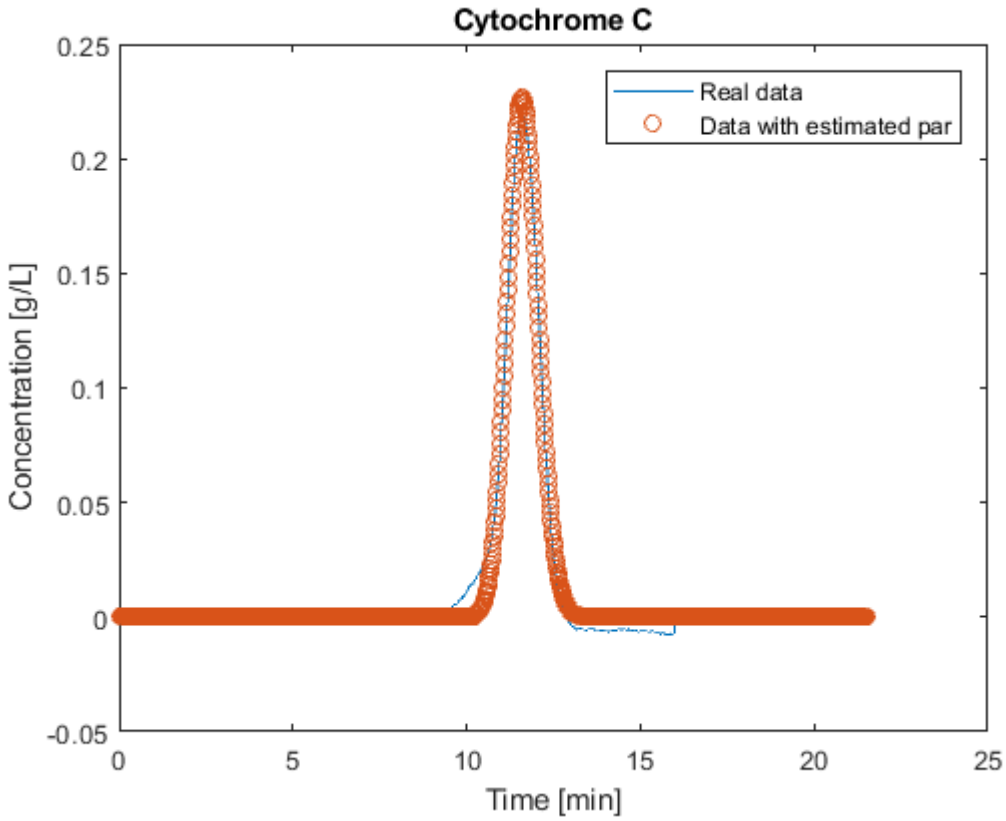


Figure 4.1. The figure shows experimental data converted to concentration and simulated data with calibrated parameters from Auto calibration script. The component is cytochrome C with injected concentration of 0.5 g/L and using kinetic Langmuir model.

Experimental data are compared to simulated data using calibrated parameters and can be seen for lysozyme in figure 4.2.

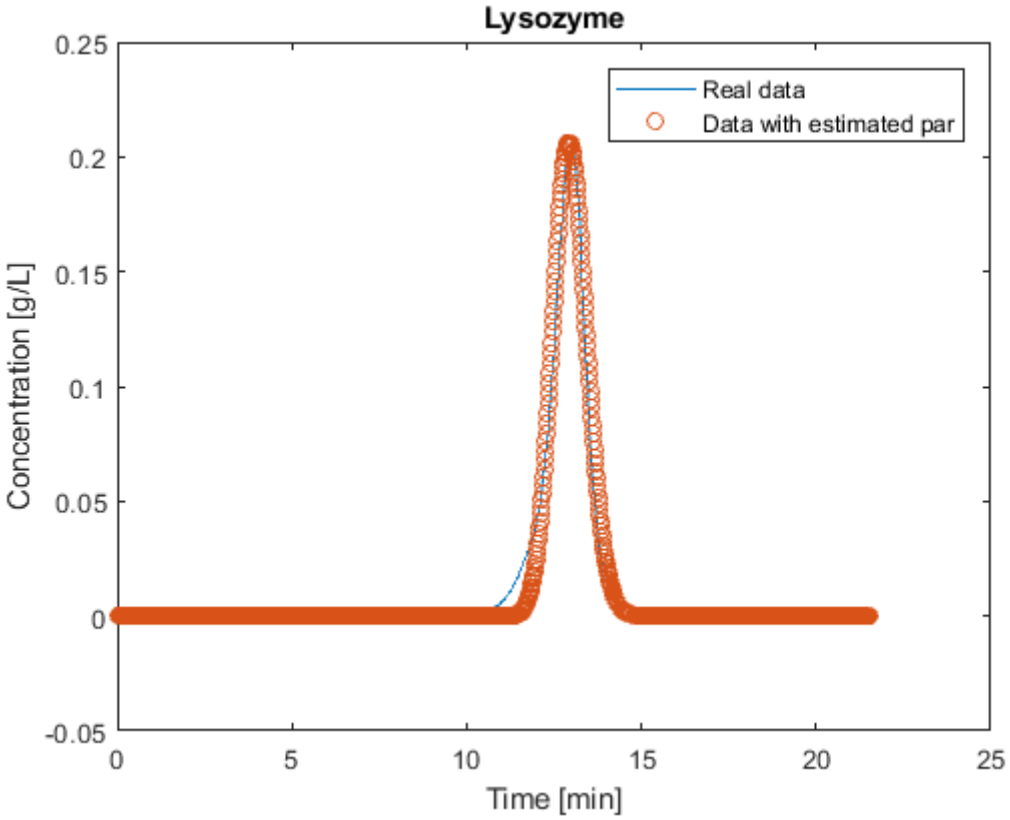


Figure 4.2. The figure shows experimental data converted to concentration and simulated data with calibrated parameters from Auto calibration script. The component is lysozyme with injected concentration of 0.5 g/L and using kinetic Langmuir model.

Experimental data are compared to simulated data using calibrated parameters and can be seen for ribonuclease A in figure 4.3.

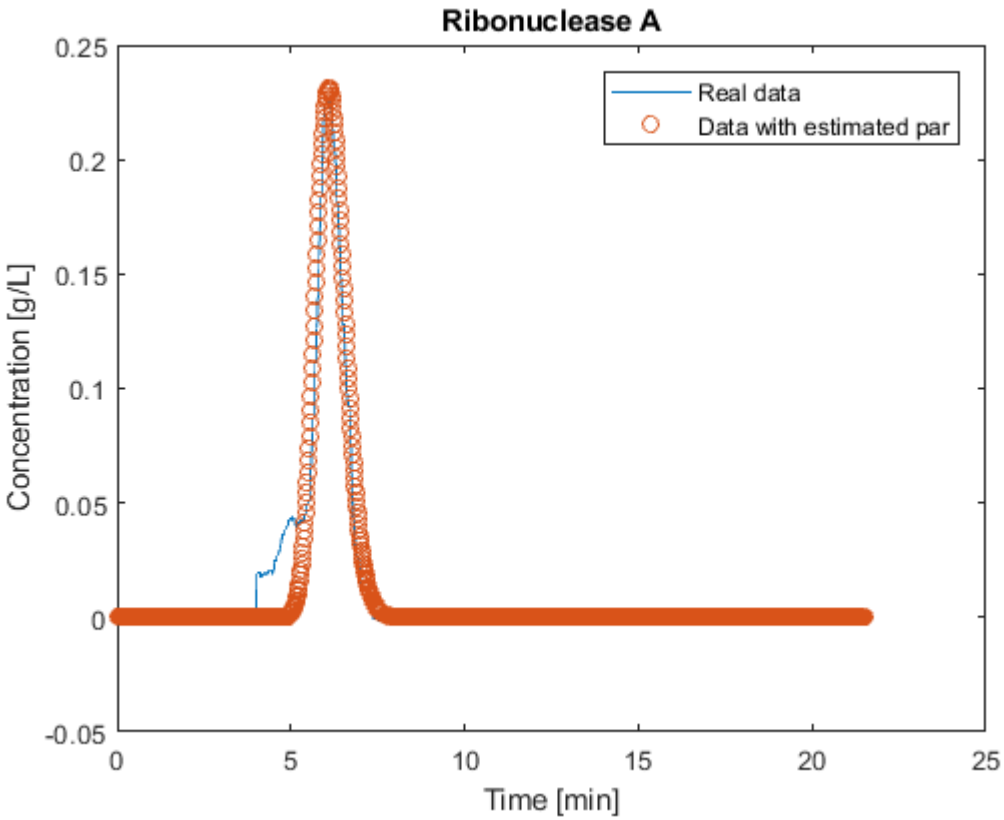


Figure 4.3. The figure shows experimental data converted to concentration and simulated data with calibrated parameters from Auto calibration script. The component is ribonuclease A with injected concentration of 0.5 g/L and using kinetic Langmuir model.

4.1.2 Multi component model validation

A multicomponent kinetic Langmuir model was simulated with calibrated parameters for each component and a validation experiment with 0.5 g/L of each component was performed. The calibrated kinetic parameters used were based on the second loop in the ECO scheme. The concentrations of the model were then converted to absorbance units mAU and compared to the validation experiment data. The comparison between experimental absorbance and simulated absorbance can be seen in figure 4.4.

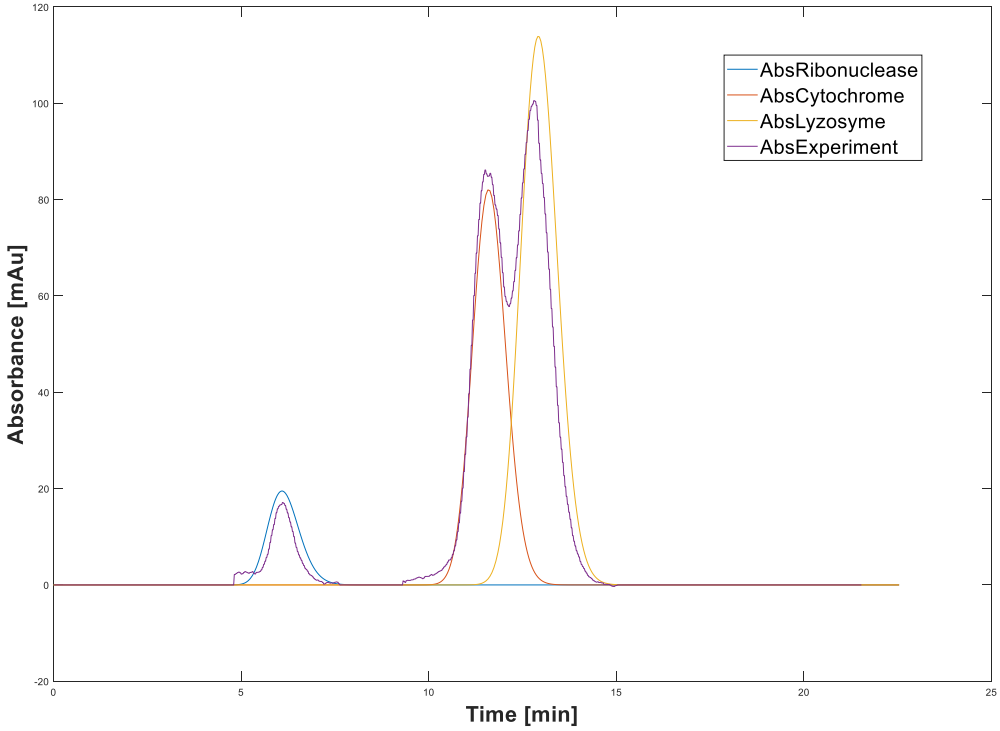


Figure 4.4. The figure shows experimental data for multicomponent and simulated data converted to absorbance with calibrated parameters from Auto calibration script. The multicomponent solution contained 0.5 g/L lysozyme, 0.5 g/L cytochrome C and 0.5 g/L ribonuclease A

To show the effects of the buffering system on the calibrated parameters, the calibrated kinetic parameters from the initial ECO loop based on 8 minutes elution time were used to model the cycle at 15.54 minutes elution time. The resulting cycle was compared to the validation experiment at 15.54 minutes. The comparison can be seen in figure 4.5 and the calibrated adsorption parameters for elution time 8 minutes can be seen in appendix E.

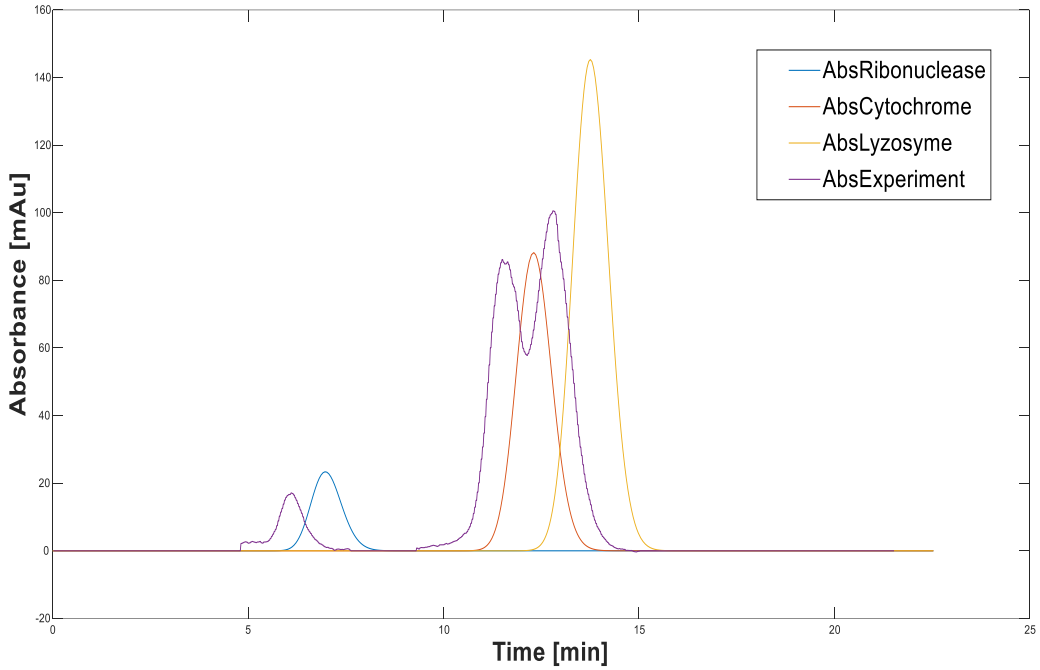


Figure 4.5. The figure shows experimental data for multicomponent and simulated data with 15.54 minutes elution time converted to absorbance with calibrated parameters from the initial ECO scheme loop. Calibrated kinetic parameters are based on 8 minutes elution time. The multicomponent solution contained 0.5 g/L lysozyme, 0.5 g/L cytochrome C and 0.5 g/L ribonuclease A.

4.1.3 Chromatography cycle buffering course

The buffering course consisting of a loading phase, elution phase and regeneration phase of the experiment can be seen in 4.6.

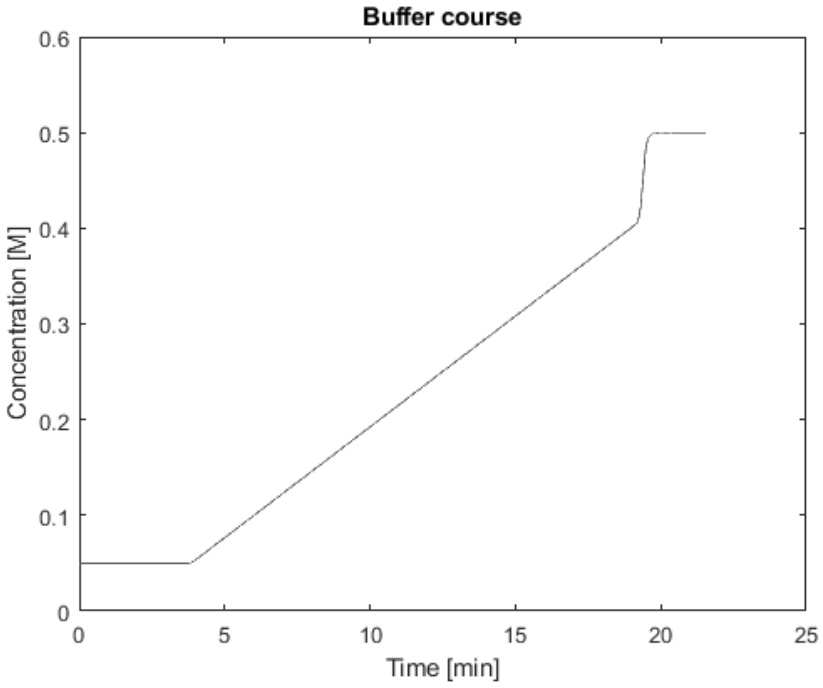


Figure 4.6. The figure shows buffer course of the simulation. It contains a loading phase, elution phase and regeneration phase as used in the experiments and calibration.

The extinction factor and calibrated kinetic parameters for lysozyme, cytochrome C and ribonuclease A can be seen in table 4.1.

Table 4.1. The table shows calibrated kinetic parameters for lysozyme, cytochrome C and ribonuclease A. Elution time used in calibration, concentration of components and injected volume are also presented.

Compound	**Elution time [min]**	k_{kin} **[1/min]**	H_0 **[M$^{\beta}$]**	**Extinction factor [g/L*cm]**	**Concentration [g/L]**	**Injection volume [ml]**
Ribonuclease A	15.54	8.04	3.81E-04	0.4065	0.5	0.5
Cytochrome C	15.54	19.11	5.56E-03	1.81	0.5	0.5
Lysozyme	15.54	13.81	3.69E-03	2.76	0.5	0.5

The Standard error, 95 % confidence interval for the fitted parameters and integration range can be seen in table 4.2. The integration range is the time interval in which the peak profile was integrated to obtain the extinction coefficient and it is also the range of peak data that is used as experimental data in lsqcurvefit.

Table 4.2. The table shows integration range used for calibration, standard errors of estimated kinetic parameters and the 95% confidence interval for the kinetic parameter estimates.

Compound	**Integration range [min]**	**Standard error** k_{kin} **[1/min]**	**Standard error** H_0 **[M$^{\beta}$]**	**95% confidence interval** k_{kin} **[1/min]**	**95% confidence Interval** H_0 **[M$^{\beta}$]**
Ribonuclease A	[4,8]	4.39E-03	1.35E-06	[8.03 8.06]	[3.78E-04 3.84E-04]
Cytochrome C	[9,16]	4.04E-02	5.23E-06	[19.04 19.19]	[5.55E-03 5.56E-03]
Lysozyme	[9,16]	3.38E-02	5.44E-06	[13.74 13.87]	[3.68E-03 3.71E-03]

4.2 Optimization

4.2.1 Retention time

The simulation performed in figure 4.7 was to find the elution time that gives a retention time for cytochrome C of 10 minutes within an error margin of 10 seconds. The bisection search method was initiated with a lower elution time limit at 5 minutes and an upper limit at 90 minutes. The method found that to obtain a retention time of 10.08 minutes, an elution time of 11.64 minutes is needed.

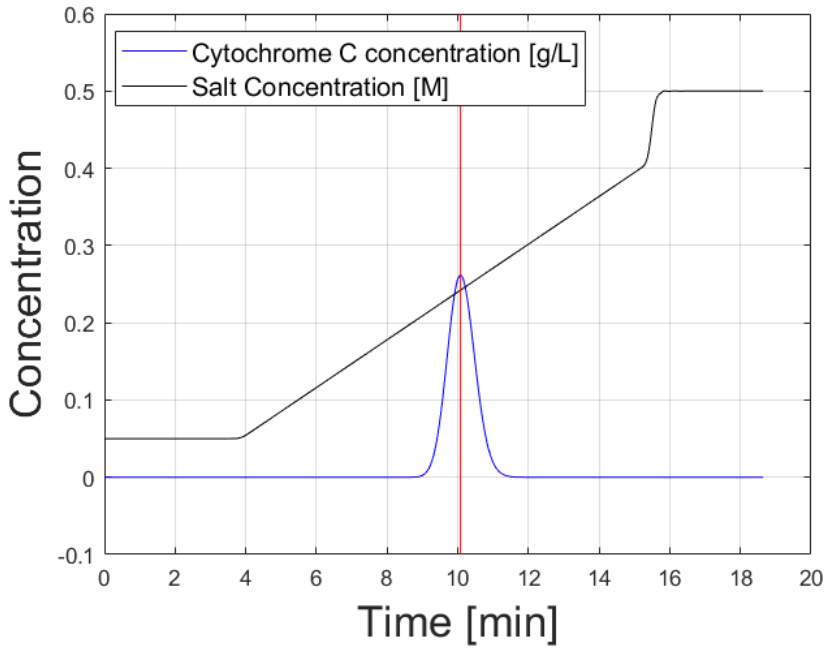


Figure 4.7. A simulation to obtain an elution time to give cytochrome C a 10 minutes retention time.

4.2.2 Yield and productivity

An optimization simulation of cytochrome C pooling effectiveness has been performed using a weighted sum of productivity and yield as the minimization objective (see Eq. (obj)) and the elution time as decision variable. The weight factor has been changed from 1.0 to 0.0 with a step length of 0.1 and the yield and productivity from this run has been plotted to the left in figure 4.8. To the right in figure 4.8, the yield and the elution time has been plotted from the optimization simulation. The last point where the yield is close to 1 has been neglected in the elution time plot, due to its unproportional elution time value.

When the weight factor is 1, only productivity is used as the objective and when the factor is 0 only yield is used as the objective. The tabulated result of productivity, yield and optimal elution time at each weighting factor can be found in appendix B.

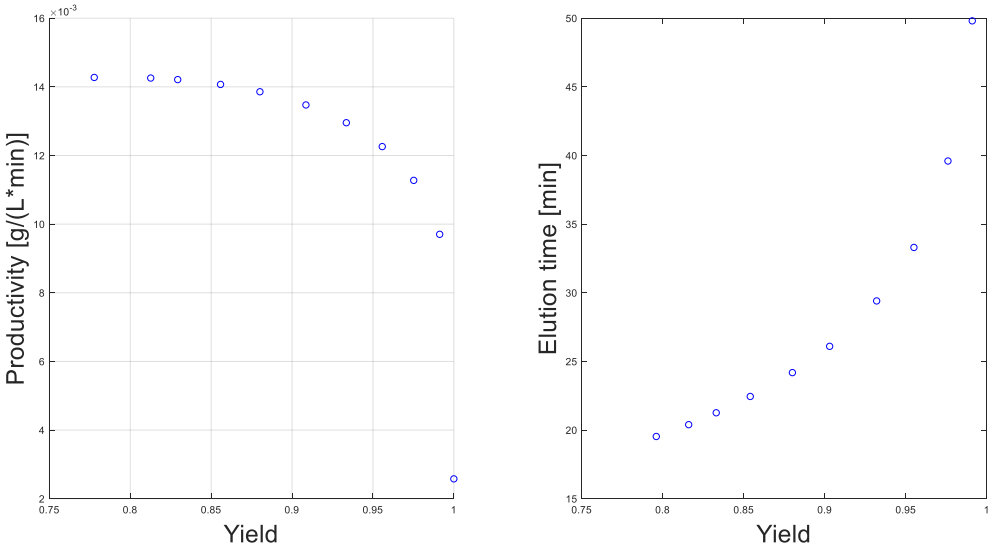


Figure 4.8: Optimized productivity and yield (left) and optimized elution time and yield (right) of pooled cytochrome C using different weighting.

Weightings of 0.5 and 0.9 were used to illustrate the pooling and optimization that takes place at each of the weighting factors in figure 4.8. These can be seen in figures 4.9 and 4.10 with a pooling plot generated by the simplex pooling method at the top of the figure and the simulated model run at the bottom. The pooling and optimization data of the simulations can be found in appendix C. The simplex pooling method yields a pooling plot with unit axes due to the internal algorithms of simplex pooling which utilizes scaled values.

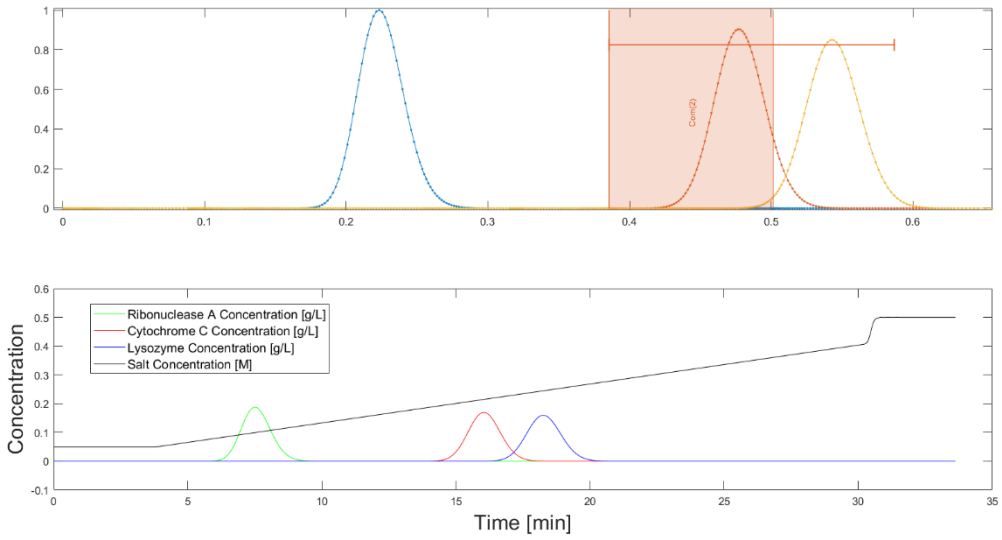


Figure 4.9: Optimization of weighted sum of productivity and yield using 0.5 as the weighting factor.

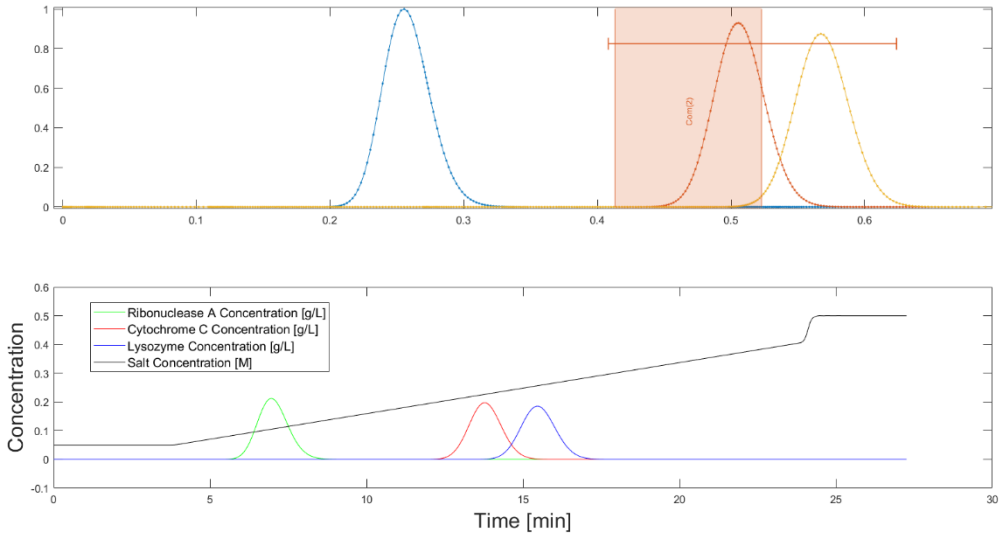


Figure 4.10: Optimization of weighted sum of productivity and yield using 0.9 as the weighting factor.

4.3 JModelica.org model

Multi component models of Matlab and JModelica.org have been run using the same system parameters and plotted in figure 4.11. The top left figure is the JModelica.org simulation of ribonuclease A, cytochrome C, lysozyme and salt concentration and the top right is the same simulation using Matlab. In the bottom, both of the results have been plotted in the same graph for comparison. JModelica.org run statistics from this simulation can be seen in appendix D.

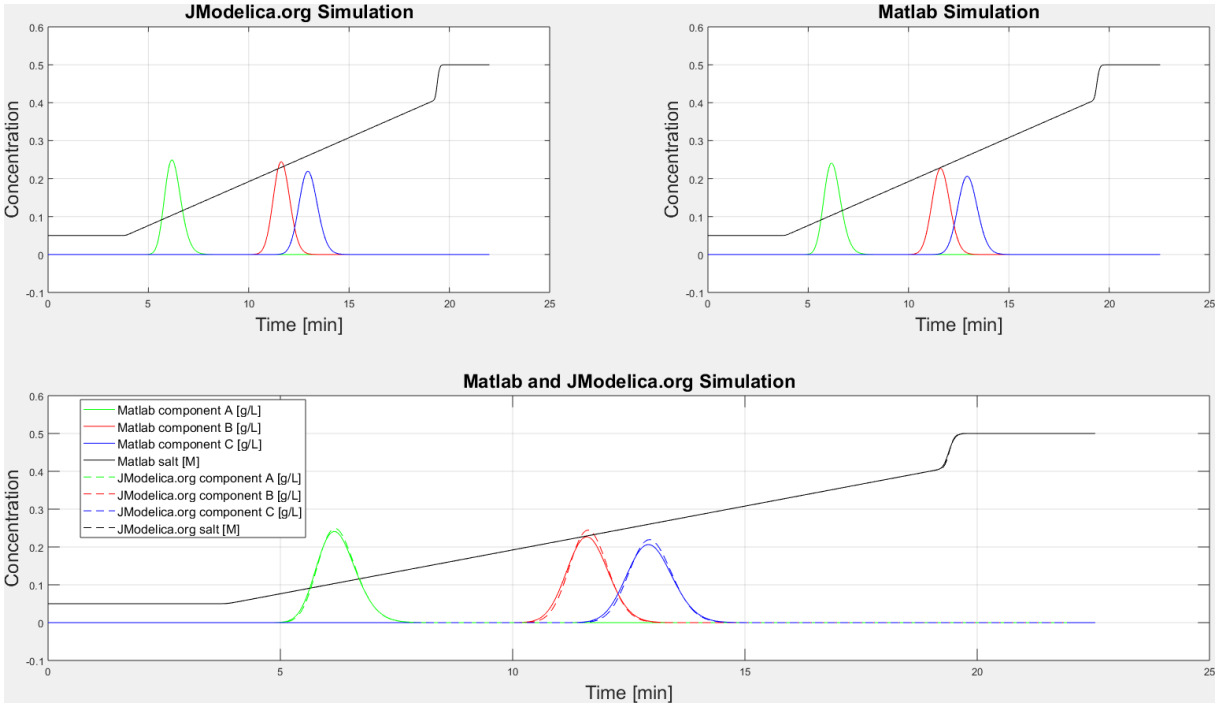


Figure 4.11. JModelica.org versus Matlab multi component chromatography simulation.

5 Discussion

5.1 API utilization

The Matlab API for Python has been extensively used in this thesis to generate all the data in the results. The chromatography model is written in Matlab, which is given adsorption parameters from the Python environment via the API in order to simulate the chromatography cycle.

The adsorption parameters have been calibrated using data sampled from chromatography runs on the ÄKTA system which is sent to Matlab along with calibration parameters from the Python environment via the API. The Matlab function `lsqcurvefit` is the calibration method which returns the calibrated parameters which is valid for the conditions which were present when the experimental data was sampled.

The calibrated parameters are then sent from the Python environment to Matlab via the API for optimization along with optimization parameters. The Matlab method `fminsearch` is utilized to find the optimal elution time which maximizes percentage yield and productivity of cytochrome C depending on the weighting of the objective function. This completes a cycle of the ECO scheme and new adsorption parameters must be calibrated at the optimized elution time.

The adsorption parameters generated via the API are even utilized to simulate the JModelica.org model.

The real strength of the API is that the access of data can be utilized to automate certain processes such as calibration and optimization, as has been shown in this thesis. Transferring data between the programs can be done manually, but the efficiency of the work is reduced since the data is evaluated first after the data transfer is done. The API can access parameters and data defined in the Python environment and directly send it to Matlab using various data types and without the need for external storage.

Another benefit of using the API is that Matlab is extensively utilized in education here at the chemical engineering institution at Lunds Tekniska Högskola which would make these ÄKTA systems more compatible with the existing education.

5.2 Parameter calibration

When choosing a model to represent the chromatographic separation it is important to weigh the pros and cons of using an advanced model or a simple model. More advanced models require more experimental work due to additional parameters that need to be calibrated. A too simple model will not describe the process with sufficiently high accuracy [7].

5.2.1 Kinetic Langmuir model

In this thesis, parameter calibration was performed on a reaction dispersive model to describe the chromatographic column separation. Choosing this model for smaller resin particles, it is not necessary to model the concentration profile in the particles as it will be relatively constant. For particles to be considered small the residence time in the bulk liquid is much longer than the residence time in the particle liquid. This occurs with slow flow rates and fast diffusion rates. In the Langmuir adsorption model, the external mass transfer resistance, the diffusion within particles and the sorption rate can be lumped into one parameter known as the kinetic rate of adsorption.

The advantage of using this model is that it is fast and requires less computational power, it has also been used in other studies to simulate ion-exchange chromatography (model).

5.2.2 Auto Calibration

The Auto Calibration was performed using Python and Matlab with the Matlab API for Python. The advantage of using Matlab for calibration is that we could use the reactive dispersive model that contains convection, dispersion and adsorption which had been developed prior to the thesis.

The parameters calibrated on were k_{kin} that describes binding kinetics of component onto stationary phase and H_0 describing equilibrium between adsorption and desorption of component. The reason for choosing these parameters was that they have a big impact on the separation of the components in the chromatographic process. When using the auto calibration script, we sometimes got a bad fitting which was observed in the plot of experimental data vs simulated data. Knowing how k_{kin} and H_0 affects the gaussian concentration profile we could deduce what parameters needed to be modified. k_{kin} will affect the amplitude and width of the gaussian concentration profile while H_0 will alter the retention time. By changing the initial guess, lower boundaries and upper boundaries that are used in lsqcurvefit it was possible to get better fitted data.

The auto calibration has a condition where the standard error must be below a limit for the calibrated parameters to be accepted, a smaller standard error means a better fit. This condition can be altered by the user depending on how exact the calibrated values need to be.

As can be seen from in results figure 4.1 the calibration for cytochrome C was successful, the calibrated parameters used in model gave a good fitting when compared to experimental data. The standard error for k_{kin} was $4.04*10^{-2}$ compared to the estimated k_{kin} of 19.11 and the standard error for H_0 was $5.23*10^{-6}$ compared to estimated H_0 of $5.56*10^{-3}$. The standard error for the best fit parameter tells you the average error of regression related to the parameter. That was the reason for using the standard error as a condition for evaluating the fit of calibration. The confidence interval tells you the interval within which the calibrated parameter with 95 % certainty contains a true value for a specific calibration. For k_{kin} the confidence interval was [19.04 19.19] and for H_0 [$5.55*10^{-3}$ $5.56*10^{-3}$].

In figure 4.2 the calibration for lysozyme was also successful, the calibrated parameters used in model gave a good fitting when compared to experimental values. The standard error for k_{kin} was $3.38*10^{-2}$ compared to the estimated k_{kin} of 13.81 and the standard error for H_0 was $5.44*10^{-6}$ compared to the estimated H_0 of $3.69*10^{-3}$ Again, the standard errors are low which indicates that a good fit was reached. The 95% confidence interval for k_{kin} was [13.74 13.87] and for H_0 it was [$3.68*10^{-3}$ $3.71*10^{-3}$].

And finally, the calibration of ribonuclease A gave a good fit, however in the case of ribonuclease A we had to cut out some of the data that represented the actual compound. This was done since ribonuclease A contained a double peak, meaning it's measured as two components. Since one of the absorbance tops was much smaller than the other, we could put the integration range where we disregarded the smaller top. This made it possible to get a good fit for the bigger top as seen in figure 4.3. The standard error for k_{kin} was $4.39*10^{-3}$ compared to estimated k_{kin} of 8.04 and the standard error for H_0 was $1.35*10^{-6}$ compared to estimated H_0 of $3.81*10^{-4}$.

Ribonuclease A is easily separated from cytochrome C and lysozyme, which made that component less crucial for the optimization this can be seen in the multi component figure 4.4 where ribonuclease is eluted much faster.

After the calibration for each component was done we used the kinetic Langmuir model for three components and inserted the calibrated data achieved during the auto calibration runs.

Since the calibrated parameters are dependent on the buffering course and the experimental conditions, we had to use the same buffering course and experimental conditions for the multi component run. We made a solution containing 0.5 g/L lysozyme, 0.5 g/L cytochrome C and 0.5 g/L ribonuclease A and made a chromatographic run storing the data in a file. That data was sent into the Multi component model in MATLAB with MATLAB API for Python. The multi-component model gave concentration profiles for the components which was then converted to absorbance with use of extinction factors retrieved in the auto calibration script. Figure 4.4 show experimental data and simulated data as absorbance mAU vs time minutes for the multi-component solution. The simulated retention time for the components was similar to the experimental retention time, however the amplitude of the absorbance varied. The amplitude is dependent on the mass of the substrate injected since the amount affects the absorbance. The solutions used in the calibration probably contained a different concentration than the solution in the experiment. This can be due to human error when making the solutions, instrumental errors and how the solution were stored since water evaporates making the protein solutions more concentrated.

Figure 4.5 emphasizes the need for recalibration at new buffering conditions where the components of the model calibrated at 8 minutes elution time are eluted earlier than the experimental validation. The jump from 8 minutes elution time to 15.54 minutes is rather large which could contribute to the significant error and so if the elution time changes less between the ECO loops it is possible that the calibrated parameters will be acceptable.

5.3 Optimization

5.3.1 Retention time

The retention time simulation, which can be seen in figure 4.7, can confirm a relatively strong affinity to the stationary phase for cytochrome C if this simulation is related to simulations of the other substances ribonuclease A and lysozyme. This phenomenon can be seen in figure 4.4 where cytochrome C is eluted just before the lysozyme.

One thing to note with this retention time simulation is that the adsorption parameters used for cytochrome C were taken from simulations with 15.54 minutes of elution time. To get more accurate results, the cytochrome C adsorption parameters should be calibrated at the same elution conditions as the simulation. This could result in the need of an ECO scheme previously described in “Multi Component Optimization”.

5.3.2 Yield and productivity

From figure 4.8, the trend of the weighting of the objectives in the fminsearch method and the range of the objectives can be seen. Since the trend of the objectives is not linear, it can be used to determine a reasonable yield or productivity that does not sacrifice too much of the other objective. This can be seen as an optimal point with consideration of the model and the methods limitations and is known as the Pareto optimality.

From the elution time and yield plot in figure 4.8, it can be stated that an increased elution time increases the yield of cytochrome C. This increase in yield can be attributed to a slower increasing salt gradient, meaning more time for the substance currently being desorbed to come into equilibrium with its surrounding salt buffer. The increase of desorption time at each salt concentration thereby gives increased resolution of the adjacent peaks and a higher separation efficiency. However, the productivity decreases due to the increased amount of time that the separation must undergo to reach the yield. The nonlinear way in which these compare is what creates the necessity of optimization which is a balance between the need for larger production and the utilization of raw materials.

The figures 4.9 and 4.10 are shown to see an implementation of the optimization method and to see how the background of the weighting simulations looks like. When 0.5 is used as the weighting factor, the objectives are valued equally when optimizing. As can be seen in figure 4.8, the point representing the objectives at a weighting factor of 0.5 has not dropped any significant productivity yet has increased the yield from around 79 % (at weighting factor 1.0) to 90 %. With a weighting factor at 0.9, the productivity is around its peak value which is not greatly different from the value at weighting factor 0.5, but the yield is around 82 %.

But let us say that the raw materials are cheap and readily available, then operating at a higher weighting factor is not a problem since more of the same purity product is being sampled every minute than at a lower weighting factor. However, if the raw materials are more expensive or in shorter supply, then the material loss due to a shorter elution time is not worthwhile and the system would be operated at a lower weighting factor. In a lab with production demands, most likely there would be a tradeoff between the objectives and a weighting factor could be decided from current market prices and stocks.

5.4 JModelica.org model

As can be seen in figure 4.11, the simulations of JModelica.org and Matlab are almost equal with the minor difference attributed to the inclusion of dispersion in Matlab which gives slight deviation from the gaussian peaks simulated by JModelica.org. Since the difference is negligible between the two simulations, it can be concluded that the dispersion that was modelled in Matlab does not have a big impact in the model.

Due to the lack of previous knowledge about JModelica.org, the ability to couple Modelica objects and make complete processes has not been utilized. This ability can be very effective when making a complete process where each process component is represented by an individual model file, but it will also increase the complexity of the model.

The purpose of this simulation was to see if JModelica.org could be called upon from the Python environment to simulate the chromatography system which has been confirmed.

6 Conclusion

The real strength of the Matlab API for Python is the ability to automate the access of data which can simplify the data transfer in an ECO scheme. This data transfer can be done manually by sending data files between the programs, however this transfer is not very efficient. With the API, data can be sent directly from the Python environment to Matlab in various data types and does not need to be saved for external storage. The data access reduces the processing time of each chromatography run and the experiment can be evaluated quicker and with less human errors. The idea is to have all the simulation parameters defined in the Python environment that can be sent to Matlab, which can be changed at any time. This creates a hierarchy of programs where Python controls the underlying ÄKTA system and Matlab.

By using internal algorithms in Python it is possible to automate the calibration, the extracted data from the experiments are used directly. With the auto calibration script, it is possible to do the experiments and then extract the data, that data is used directly in MATLAB models with Matlab API for Python. The results of the auto calibrations show that the calibrations yield good parameter estimations. However, the results of calibration are dependent on both the generated experimental data and on the accuracy of the user inputs.

JModelica.org has at least the same capacity for simulation as Matlab but has not been further utilized. JModelica.org is an open-source software and is still in development which poses a challenge to a less experienced programmer, however the engine for computations is still prime and should be further investigated.

7 Future work

To further improve and utilize the work of this thesis the main topics are to diversify the applications of the API and to improve upon the code structures.

The ECO scheme can be automated further by making a coupling between the calibration and the optimization methods. The main issues with the coupling is that when experimental data is generated, the fit of the calibration may vary with the success of the experiments. There is a tendency for disturbances in the UV signal to interfere with the peak signal. For this to work smoothly, the need for a good validation method for the parameter estimations should be implemented. Another way to make the automation possible is to make multiple experiments in the experiment phase and choose the most accurate one to represent the system.

To expand upon the complexity of the problem more parameters could be chosen for calibration, such as β, more decision variables could be added in the calibration for more flexibility, different optimization methods could be chosen, such as fminunc and fmincon and multiobjective optimization could be expanded upon, for example by using multiobjective optimization methods.

Furthermore, the functions for calibration and optimization could be have assertions and other defensive programming added into them for sustainable programming.

JModelica.org was not utilized to its full potential in this thesis and was only used for simulation. There is reason to believe that JModelica.org has greater capacity for process simulation than Matlab does and should not be neglected. The ability to couple different Modelica objects together can be a powerful simulation tool. Calibration and optimization is also available in the Optimica extension which can be further investigated.

As a last important note, the use of the automation in other applications such as model calibration, automating experiments using optimized values, utilization of other simulation programs (JModelica.org for example) or using different ÄKTA system setups (recycling, multiple columns).

8 References

[1] H. Schmidt-Traub, *Preparative chromatography of fine chemical and pharmaceutical agents*. Weinheim: Wiley-VCH, 2006.

[2] "ÄKTAexplorer - GE Healthcare Life Sciences", *GE Healthcare Life Sciences*, 2018. [Online]. Available: https://www.gelifesciences.com/en/us/shop/chromatography/chromatography-systems/aktaexplorer-p-05772#related-documents. [Accessed: 19-Apr-2018].

[3] N. Andersson, A. Löfgren, M. Olofsson, A. Sellberg, B. Nilsson and P. Tiainen, "Design and control of integrated chromatography column sequences", *Biotechnology Progress*, vol. 33, no. 4, pp. 923-930, 2017.

[4] J. Åkesson, K. Årzén, M. Gäfvert, T. Bergdahl and H. Tummescheit, "Modeling and optimization with Optimica and JModelica.org—Languages and tools for solving large-scale dynamic optimization problems", *Computers & Chemical Engineering*, vol. 34, no. 11, pp. 1737-1749, 2010.

[5] D. Harris, *Quantatitve chemical analysis*. New York: W. H. Freeman, 2010, pp. 537-558.

[6] D. Sheehan and S. O'Sullivan, "Ion Exchange Chromatography", *Encyclopedia of Life Sciences*, 2001.

[7] B. Nilsson, "Introduction to Preparative Chromotography", Lund University, Sweden, Department of chemical Engineering, 2017.

[8] M. Degerman, *Design of robust preparative chromatography*. Lund: Lund University, 2009.

[9] A. Sellberg, A. Holmqvist, F. Magnusson, C. Andersson and B. Nilsson, "Discretized multilevel elution trajectory: A proof-of-concept demonstration", *Journal of Chromatography A*, vol. 1481, pp. 73-81, 2017.

[10] B. Nilsson, "Modelling and Simulation III - Space Dependent Systems Part A", Lund University, Sweden, Department of Chemical Engineering, 2017.

[11] *Extinction Coefficients*. 2013. [ebook] USA, Rockford: Thermo Fisher Inc. Available at: https://tools.thermofisher.com/content/sfs/brochures/TR0006-Extinction-coefficients.pdf. [Accessed:03-May-2018].

[12] K. Hangos and I. Cameron, *Process modelling and model analysis*. San Diego: Academic Press, 2001.

[13] B. Nilsson, "Optimisation – Formulation and Methods", Lund University, Sweden, Department of chemical Engineering, 2017.

[14] *ÄKTA system Training Guide*. (n.d.). [ebook] Cleaveland, Ohio: Case Western Reserve University. Available at: https://physiology.case.edu/media/eq_manuals/eq_manual_amersham_akta_explorer_100_fplc_user_guide.pdf [Accessed 19 Apr. 2018].

[15] *System control software UNICORN™ 7 software*. 2016. [ebook] Sweden, Uppsala: GE Healthcare Bio-Sciences AB. Available at: https://cdn.gelifesciences.com/dmm3bwsv3/AssetStream.aspx?mediaformatid=10061&destinationid=10016&assetid=17378 [Accessed 19 Apr. 2018].

[16] "Jmodelica.org", 2010. [Online]. Available: http://www.jmodelica.org/page/28. [Accessed: 23- Apr- 2018].

[17] "Spoken Tutorial Project, IIT Bombay", *Spoken-tutorial.org, 2017*. [Online]. Available: http://spoken-tutorial.org/tutorial-search/?search_foss=OpenModelica&search_language=English. [Accessed: 23- Apr- 2018].

[18] "HiTrap SP HP cation exchange chromatography column - GE Healthcare Life Sciences", *GE Healthcare Life Sciences*, 2018. [Online]. Available: https://www.gelifesciences.com/en/ae/shop/chromatography/prepacked-columns/ion-exchange/hitrap-sp-hp-cation-exchange-chromatography-column-p-00794?current=17115101. [Accessed: 19- Apr- 2018].

[19] "Ribonuclease A from bovine pancreas R6513", *Sigma-Aldrich*, 2018. [Online]. Available: https://www.sigmaaldrich.com/catalog/product/sigma/r6513?lang=en&region=SE. [Accessed: 26- Apr- 2018].

[20]"PDB101: Molecule of the Month: Ribonuclease A", *RCSB: PDB-101*, 2008. [Online]. Available: http://pdb101.rcsb.org/motm/105. [Accessed: 26- Apr- 2018].

[21] "Cytochrome c equine C6749", *Sigma-Aldrich*, 2018. [Online]. Available: https://www.sigmaaldrich.com/catalog/product/sigma/c6749?lang=en&region=SE. [Accessed: 01- May- 2018].

[22] D. Green, *Means to an end*. Cold Spring Harbor, N.Y.: Cold Spring Harbor Laboratory Press, 2011.

[23] "Lysozyme 10837059001", *Sigma-Aldrich*, 2018. [Online]. Available: https://www.sigmaaldrich.com/catalog/product/roche/10837059001?lang=en&region=SE. [Accessed: 01- May- 2018].

[24] B. Nilsson, "Modelling and Simulation III - Space Dependent Systems Part B", Lund University, Sweden, Department of chemical Engineering, 2017 (r)https://se.mathworks.com/help/optim/ug/lsqcurvefit.html

[25] The MathWorks, inc., "Solve nonlinear curve-fitting(data-fitting) problems in least-squares sense – MATLAB lsqcurvefit – MathWorks Nordic, " The MathWorks, Inc., [Online]. Available: https://se.mathworks.com/help/optim/ug/lsqcurvefit.html#buuhcjo_seealso. [Accessed:03-May-2018].

[26] B. Nilsson, "Model Calibration and Model Validation", Lund University, Sweden, Department of chemical Engineering, 2017.

[27] *Conductivity Theory and Practice*. Villeurbanne: Radiometer Analytical SAS, 2018.

[28] S. Moldoveanu and V. David, *Essentials in modern HPLC separations*. Waltham, MA: Elsevier, 2013, pp. 53-83.

[29] The MathWorks, inc., “Find minimum of unconstrained multivariable function using derivative-free method – MATLAB fminsearch – MathWorks Nordic, “ The MathWorks, Inc., [Online]. Available: https://se.mathworks.com/help/Matlab/ref/fminsearch.html [Accessed 23 Apr. 2018].

[30] "Pareto Front", *Cenaero.be*, 2018. [Online]. Available: http://www.cenaero.be/Page.asp?docid=27103&. [Accessed: 23- May- 2018].

9 Appendices

9.1 Appendix A: Simulation and model parameters

Table 9.1. List of modelling and simulation parameters

Parameter	Description	Value
e_c	Column porosity	0.319
e_p	Particle porosity	0.856
N_{Pe}	Peclét Number	0.500
D_p	Particle diameter [microm]	34.0
D_c	Column diameter [mm]	7.00
L	Column length [mm]	25.0
F	Flow rate [mL/min]	1.0
q_{max}	Maximum capacity of column [g/L]	2000
β_{ribo}	Ion-exchange interaction parameter/characteristic charge for ribonuclease A	3.68
β_{cyto}	Ion-exchange interaction parameter/characteristic charge for cytochrome C	4.25
β_{lyso}	Ion-exchange interaction parameter/characteristic charge for lysozyme	4.95
N	Amount of grid points for discretization	100

9.2 Appendix B: Optimization weighting results

Table 9.2. Table of optimized elution time for maximizing productivity and yield depending on how the objectives are weighted.

Weighting	Productivity [g/(L*min)]	Yield	Optimal Elution time [min]
1.0	0.0143	0.796	19.533
0.9	0.0143	0.816	20.394
0.8	0.0142	0.833	21.260
0.7	0.0141	0.854	22.448
0.6	0.0139	0.880	24.186
0.5	0.0136	0.903	26.100
0.4	0.0130	0.932	29.408
0.3	0.0123	0.955	33.304
0.2	0.0112	0.976	39.599
0.1	0.0098	0.991	49.809
0.0	0.0020	0.999	466.481

9.3 Appendix C: Pooling and Optimization results

Table 9.3. Pooling and optimization data from optimization experiments using weighting of 0.5 and 0.9. The amount in pool, loaded amounts, productivity, purity and yield are for the components in the order ribonuclease A, cytochrome C and lysozyme. The cut times is described as the start time of the pooling and the end time of the pooling.

Weighting factor	**0.5**	**0.9**
Optimization		
Productivity [g/(L*min)]	0.0135	0.0143
Yield	0.909	0.813
Optimal Elution time [min]	26.627	20.260
Pooling		
Amount in Pool [mg]	[0, 0.227, 0.0023]	[0, 0.2032, 0.0021]
Loaded amounts [mg]	[0.250, 0.250, 0.250]	[0.250, 0.250, 0.250]
Cut Times [min]	[12.963, 16.860]	[11.265, 14.251]
Pool Concentrations [g/L]	[0, 0.0583, 5.888e-4]	[0, 0.0680, 6.8729e-4]
Cycle Time [min]	16.860	14.251
Productivity [g/(L*min)]	[0, 0.0135, 1.361e-4]	[0, 0.0143, 1.440e-3]
Purity	[0, 0.990, 0.0100]	[0, 0.990, 0.0100]
Yield	[0, 0.909, 0.0092]	[0, 0.813, 0.0082]
Iterations	153	67

Function Counts	307	135
Algorithm	Nelder-Mead simplex direct search	Nelder-Mead simplex direct search
Termination Criteria	1.000e-4	1.000e-4
Convergence Criteria	1.000e-4	1.000e-4

9.4 Appendix D: JModelica.org run statistics

Table 9.4. Jmodelica.org run statistics from the simulated chromatography model in figure 3.10.

Number of steps	726
Number of functions evaluations	999
Number of Jacobians evaluations	17
Number of function eval. due to Jacobian eval.	11900
Number of error test failures	16
Number of nonlinear iterations	979
Number of nonlinear convergence failures	0
Number of time events	4
Solver options	
Solver	CVode
Linear multistep method	BDF
Nonlinear solver	Newton
Linear solver type	DENSE
Maximal order	5
Tolerances (absolute)	1e-6
Tolerances (relative)	0.0001

9.5 Appendix E: Calibration results at elution time 8 minutes

Table 9.5. The table shows calibrated kinetic parameters for lysozyme, cytochrome C and ribonuclease A at elution time 8 minutes. Elution time used in calibration, concentration of components and injected volume are also presented.

Compound	Elution time [min]	k_{kin} [1/min]	H_0 [M$^{\beta}$]	Extinction factor [g/L*cm]	Concentration [g/L]	Injection volume [ml]
Ribonuclease A	8.00	12.20	8.18E-04	0.450	0.5	0.5
Cytochrome C	8.00	21.40	5.56E-03	1.81	0.5	0.5
Lysozyme	8.00	13.81	3.69E-03	2.76	0.5	0.5

9.7 Appendix F: Populärvetenskaplig sammanfattning

Automation: Optimering av din tid

Tänk dig att du har fått en deadline på ditt nya projekt. Ditt arbete för projektet beror helt på din praktiska noggrannhet och tidsoptimering, samtidigt så ska du lämna in massvis med pappersarbete. Du har en tuff tid framför dig ifall du ska klara detta på egen hand inom tidsramen. Här kommer automatisering in i bilden. Det noggranna och tidskrävande arbetet som du behöver göra kan du hålla i ordning och göra smidigare ifall din process automatiseras. Detta ger dig mer tid över att utfärda pappersarbetet på ett bra sätt.

Om vi återgår till vetenskapens värld så kan man applicera liknande tänkebanor kring uppreningen av läkemedel. För att kunna förutspå hur en uppreningsprocess ska bete sig behöver processen modelleras i en dator. Informationsflödet mellan datorn och processen, även utbytet av information mellan mjukvaror i datorn, kan göras manuellt eller automatiserat. Automatiseras informationsflödet så kan hela informationsutbytet ske i ett steg till skillnad från vid manuellt utbyte där flera steg är ett måste. Denna automatisering kommer ej utan sina smärtor, men när den väl är implementerad så sparar den massvis med börda och tid för kommande arbete.

I ett examensarbete från Lunds Universitet har Matlab och Python, två väldigt väl använda mjukvaror, kopplats samman genom en Application Programming Interface (API). Python hanterar det huvudsakliga informationsflödet där experimentella resultat importeras från uppreningsprocessen för att sedan användas tillsammans med Matlab för att få en bild av hur processen bör bete sig. I Matlab har en modell av uppreningsprocessen skapats och kräver endast att man nämner vilka betingelser som processen sker vid för att få en simulering av processen.

Ett alternativt modelleringsprogram har även testats i examensarbetet vid namn JModelica.org som först utvecklades vid Lunds Universitet och som sedan tagits över av Modelon AB. Skillnaden mellan Matlab och JModelica.org är att JModelica.org är skapad specifikt för industriell simulering och optimering medan Matlab har en mer allmän anfallsvinkel.

Dessa båda modelleringprogrammen har alltså en koppling till informationsflödet som sker från uppreningsprocessen och kan användas för att förutspå hur väl den modellerade processen kommer att fungera. På grund av det integrerade informationsflödet kan all simulering göras i ett steg och sparar mycket tid som annars går till spillo vid informationshantering. Förhoppningen är att kunna automatisera andra systems informationsflöde med denna teknik för att optimera personers tidsanvändning och för att minska den mänskliga faktorn vid experiment.

Texten ovan är en populärvetenskpalig sammanfattning av examensarbetet *Integration of modelling environments to an ÄKTA system* av Linus Gustafsson och Daniel Nilsson.