Automated Simulation and Parameter Estimation of Chromatography Processes

Johan Olsson

Supervisor: Associate Senior Lecturer Niklas Andersson
Examiner: Professor Bernt Nilsson

LUND
UNIVERSITY

Department of Chemical Engineering

Department of Chemical Engineering
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Media-Tryck.
Lund 2019

Abstract

Preparative chromatography is an important separation technique in downstream processing for several industries. It has the ability to separate two or more component from each other to achieve high purity. Most components that are separated with preparative chromatography are high value components and it is therefore necessary to use small amounts in the development of the process too lower costs. The ability to model these systems and simulate them before running experiments could lower these costs and also make the process more efficient.

At the Department of Chemical Engineering at Lund University, liquid chromatography systems (ÄKTA from GE) are controlled with a in-house developed software called *Orbit*. In this thesis, a simulation software was integrated into the software so that an experienced user could simulate methods with the same syntax as before. The software was able to simulate the whole flow path with tubes, columns and other units. The user was then able to follow different components throughout the system and predict their behavior. For the column, a simplified mass steric action model was implemented so that the adsorption of components could be described.

A parameter estimation software was also implemented into the software package. This gave the user the option to estimate model parameters for a component using only two data sets. The software designed so that *Orbit* could conduct the experiments on the ÄKTA machines and then from that data, it could estimate model parameters. In this study, data sets was created using the simulation software and known model parameters. Using these data sets, the software was able to estimate the model parameters in different cases with great precision and robustness. Even if the user made a bad guess, the software was still able to find precise parameters.

Sammanfattning

Kromatografi är en viktig separationsteknik vid upprening av produkter inom många processindustrier. Metoden används för att separera två eller fler komponenter från varandra för att utvinna produkter med högt utbyte och hög renhet. Komponenterna som utvinns vid kromatografi är oftast väldigt dyra att producera och därmed så är det väldigt viktigt att att använda små mängder i utvecklingsfasen för att dra ner kostnaderna av slutprodukten. Möjligheten att modellera och simulera hur dessa komponenter beter sig i systemen innan man utför experiment hade kunna sänka kostnaderna i utvecklingsfasen, men även effektivisera arbetet.

Vid institutionen för kemiteknik vid Lunds Universitet används en utvecklad mjukvara för att styra vätskekromatografisystem (ÄKTA systemen från GE). I denna studie så har denna mjukvara integrerats med en simuleringsmjukvara som tar informationen som skickas till systemen och simulerar dem på förhand. Mjukvaran kan simulera hela systemet med tuber, kolonner och andra enheter som tillhör maskinerna. En användare kan följa komponenterna genom systemet, undersöka hur de rör sig genom tuber och enheter, men även förutspå resultat innan de genomförs experimentellt. För kolonnen så implementerades en adsoptionsmodell för att modellera hur komponenterna adsorberas och eluerar.

I mjukvaran så implementerades även en modellparameterskattning. Den gav användaren valet att skatta modellparametrar utifrån enbart två stycken experimentella dataset av en komponent. Även denna mjukvara var integrerad med resterande vilket ledde till att experiment utförda på ÄKTA systemen kunde direkt användas i parameterskattningen. I denna studien så skapades experimentella data med hjälp av simuleringsmjukvaran och kända modellparametrar från tidigare studier. Utifrån den datan så kunde mjukvaran skatta modellparametrar med bra precision och robusthet. Även om användaren gjorde en dålig gissning på modellparametrarna så skattade mjukvaran parametrarna mycket väl.

Acknowledgments

I want to start of by thanking Bernt Nilsson for giving me the opportunity to preform my master thesis in his research group. He has provided me with guidance and knowledge throughout the whole process and giving me the tools to succeed while challenging me in my work.

This work could not have been done without the help from my supervisor Niklas Andersson. Thanks to his guidance, our discussions and his expertise this thesis was made possible. I am very grateful for his help and his ability to always find time to help.

I would also like to thank Anton Löfgren, Mikael Yamanee-Nolin and Simon Tallvod for always being open to discuss my problems and sharing their knowledge.

At last I would like to thank the whole Department of Chemical Engineering for providing a wonderful work environment.

Contents

Nomenclature

Latin Letters

A	Flow area	m^2
c	Mobile phase concentration	$\frac{mol}{m^3}$
D	Dispersion coefficient	$\frac{m^2}{s}$
F	Volumetric flow	$\frac{m^3}{s}$
GP_{column}	Number of grid points for a column	–
GP_{tube}	Number of grid points for a tube	–
h	Distance between two gridpoints	m
K_{eq}	Equilibrium constant	$\frac{mol}{m^3}$
k_{kin}	Kinetic adsorption coefficient	$\frac{m^3}{mols}$
N	Dispersion	$\frac{mol}{m^2 s}$
$N_{columns}$	Number of columns	–
N_{comp}	Number of components	–
N_{items}	Number of simulated items	–
N_{obs}	Number of observed values	–
N_{tubes}	Number of tubes	–
N_x	State variables	–
q	Stationary phase concentration	$\frac{mol}{m^3}$
Q_{dist}	Objective function for the distance between peaks	–
Q_{res}	Objective function for the squared residuals	–
t	Time	s
t_{ret}	Retention time	s
u_{int}	Interstitial flow velocity	$\frac{m}{s}$
v	Flow velocity	$\frac{m}{s}$

w	Weight for objective function	w

Greek Letters

Δ	Difference between two points	–
ν	Characteristic charge	–
ε	Total fluid fraction of the column	–
ε_c	Column void fraction	–
ε_p	Packing porosity	–

Overall Subscipts and Superscripts

ax	Axial direction
$data$	Experimental data
i	Component
in	Inlet
$init$	Initial
max	Maximum
n	Grid point index
s	Salt
sim	Simulated data
z	Point in the axial length of the item

1

Introduction

Chromatography is an important separation technique in downstream processing for several industries. It has the ability to separate two or more components from each other to achieve high purity. In liquid chromatography, the liquid is pumped through a system and separation occurs where the liquid is introduced to a new phase. In this thesis, the second phase is a solid phase called *stationary phase* in the form of a packed column. It is in the column that the components adsorb on the stationary phase, and when the composition of the liquid changes, the components desorb. The change in liquid composition in this thesis is an increase in salt concentration that weakens the interaction between components and stationary phase. When the goal of the chromatography is to purify components from each other it is called *preparative chromatography* and this will be the focus of this thesis.

In many industries, high cost components is produced and purified using preparative chromatography. This requires a long experimental phase to gather data of the system and build a good method of separation. Since the component is expensive to produce, this leads to high research costs if a lot of component is needed for the downstream process research. If this time and number of experiments could be decreased, the cost of developing a process for these components could be greatly reduced.

There are many proven models for the adsorption of components on a packed column. Using these models, simulation of components with known kinetics can greatly reduce the experimental phase and therefore the cost of production. Instead of spending time on conducting a great number of experiments and try to understand the complex system, a small number datasets could be used to estimate the model parameters.

At the Department of Chemical Engineering at Lund University, a software for controlling a chromatography processes has been developed. Users can specify their own run methods and the software can make decisions based on the information given e.g. controlling pH and choosing the cutoff points to achieve the highest purity. The integration of a simulation and parameter estimation software would grant users the ability to both simulate their run methods before starting an experiment, but also automate parameter estimation for unknown components.

1.1 Aim

This project has two aims. The first is to develop a software that can simulate any type of instructions that are given to the chromatography system and show the results of the whole procedure. This will give the user the ability to predict the outcome of the experiments and change parameters before starting the experiment. The study also covers the implementation of the ability to conduct a robust parameter estimation of a adsorption column model. Given very little information, the software should be able to find the parameters for any component.

The second aim is to automate this procedure so that the user only sets up a few experiments in the code and starts them all with a press of a button. Then the idea is that the

software will handle the rest. All this is to be integrated in the already used software *Orbit* at the Department of Chemical Engineering at Lund University.

2

Background

In this chapter, the background and theory behind the software will be presented.

2.1 High Pressure Liquid Chromatography (HPLC)

The separation in chromatography is based on differences in equilibrium constants for the components of a mixture placed in the diphasic system (a mobile and a stationary phase). In an HPLC system, the liquid (also called the mobile phase) is pumped through a column containing the stationary phase. The liquid is pumped through the stationary phase and depending on the physical and chemical properties of the components and the specific column, separation occurs [1].

The stationary phase can be of many types, the important thing is that the equilibrium area where adsorption and desorption can occur is large. A large area provides more sites where the adsorption and desorption can occur which increases the separation in the column. One type is to have an absorbent with a relatively large surface area and very accessible pore channels (i.e a resin). Another common type is to have coated walls in the column where the phase interaction can occur. There are a number of different types of columns that can be used in a HPLC system. Each utilizes different chemical and physical properties to separate the components from each other. For protein separation, ion-exchange column is probably the most common. It has a stationary phase that is either negatively or positively charged and binds components depending on their charge [2]. Another type is the size-exclusion column that separates components based on physical size. In this project, an ion-exchange column will be used.

There are two main areas of use for a chromatography system: Preparative or analytical. In preparative, the goal is to purify a substance from a mixture and store it for further use. Unlike in analytical where the information gathered from the separation is more important to understand the substance [1].

This project will focus on both parts: The analytical part is used to understand how the components interacts with the system and is needed for the parameter estimation. When an understanding of how the components interacts, the focus will be on how to use this in preparative methods.

2.2 System

The system that will be simulated is an ÄKTA Pure chromatography machine made by GE Healthcare. It is a High Pressure Liquid Chromatography machine with plenty of flexibility. The system is designed so that the user can set up the system with different modules depending on the purpose of the experiment. There are different types of modules e.g. UV sensor, pH sensor, pumps and different kinds of valves. The different kind of valves are used for

separate purposes such as sending the fluids to the column or having an inlet for the injection pump [3]. A picture of the machine can be seen in Figure 2.1.

Figure 2.1 The ÄKTA Pure chromatography system.

The modules are connected to each other with tubes which comes in different flow area sizes and are used widely over the system. The key component of the system is the choice of column where the separation of components occur.

2.3 Orbit

The software that ships with the machine is called *Unicorn*. In *Unicorn*, the user can setup chromatography runs with different settings on buffer gradient, flow paths and flow velocity. The user can also view the measurements made by the sensors on-line while commencing an experiment.

Unfortunately *Unicorn* has several limitations such as no real time control, complex scripting and simultaneous run of multiple connected ÄKTA systems. These things are all necessary when creating complex HPLC processes that is needed for more complex problems. This was the reason why the program *Orbit* was created. All these demands are met in *Orbit* through OPC (software interface standard for Windows to communicate with industrial hardware).

As a user, one can specify different phases e.g. a wash of the system, loading the column, elution and regeneration phases. Once specified, *Orbit* will create a "method" out of this information and send it to the chromatography machine. While all these phases are executed by the ÄKTA Pure, other calculations and control systems runs in the background of the software. This allows great flexibility for the user and the possibility to make complex processes with ease. [4] An overview of orbits role can be seen in Figure 2.2.

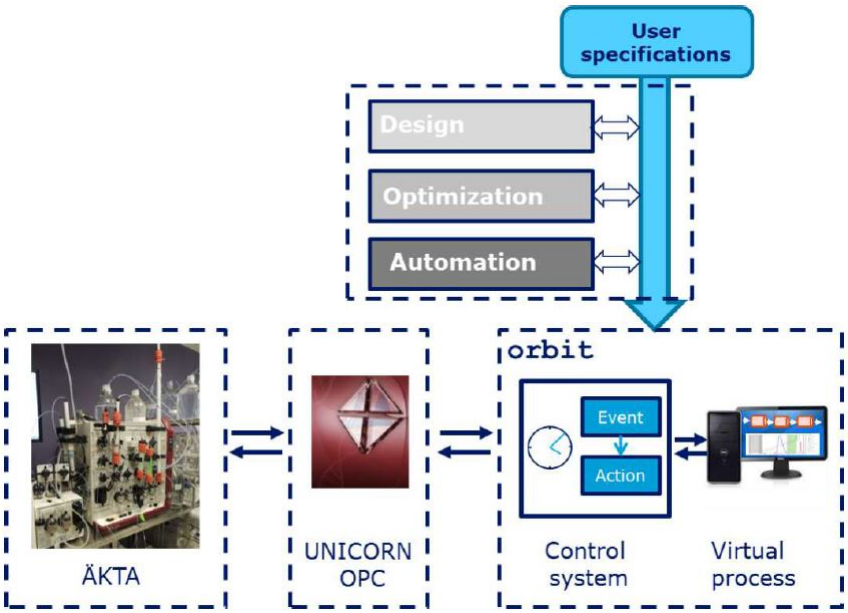

Figure 2.2 An overview over the Orbit controller program developed in Python. [4]

2.4 Simulation of Chromatography Processes

There has been several other studies on simulation of chromatography processes over the years [5]–[9]. In these studies several different models of adsorption is discussed and calibrated depending on process conditions. In a study, Karlsson et al. [5] simulated a ion-exchange step for a antibody purification step using Langmuir kinetics with mobile phase modulators (MPM). In this model, there is a certain amount of available ligands on the stationary phase which the proteins compete against each other to bind into. The salt is considered as inert and the concentration of salt affects the retention of proteins. Karlsson et al. also states that in 2001, downstream processing of antibodies contributed to 80 % of the total production cost. To lower these costs, development of cheap methods for designing and optimizing purification steps is seen as a high priority. In this development, simulations is a great way of reducing the number of experiments thus reducing cost.

In a study by Jakobsson et al. [6], a model-based approach to an ion-exchange chromatography step is introduced. It shows the possibility to make robust calibration within a confidence interval for two components. By showing that this type of calibrations is possible, single-component experiments can be avoided which lowers the developments costs. The adsorption model in this study was the steric mass action model (SMA). The key role of the SMA model is that the electro-neutrality must be conserved in the interaction between proteins and solid phase in the form of an equilibrium reaction. Both salt and protein compete for the available binding sites in the column and a binding of a protein also shields a number of ligands. Jakobsson et al. also presents the possibility of a linear relation between pH and the equilibrium model parameter in a narrow pH range. This gives robustness to the model since pH variations is one of the most difficult process parameter to control.

In Al-Kaisy's master thesis [7], four different adsorptions models were investigated and calibrated against experimental data. Besides the already mentioned MPM and SMA models, the self-association (SAS) model and the generalized Langmuir (GL) model was investigated. Since the focus of this thesis is not on the model, the two later mentioned will not be explained further. Al-Kaisy showed that all these models could be calibrated into a good fit on experimental data when the column was loaded with a small amount of proteins. For the experiments with a high load of proteins in the column, all models had a hard time fitting

the data. When validating the model parameters against a three component experiment with a low protein load, all models showed an acceptable fit.

In Persson's master thesis [8] he investigated the usage of a two-column recirculation process to achieve better process performance than traditional batch chromatography. In the study, he calibrated an SMA model from experiments to three proteins: Lysozyme, cytochrome c and ribonuclease A. With the help of these models, he performed an optimization based on yield, purity and productivity. The ion-exchange column used in Persson's work is still used at the Department of Chemical Engineering at Lund University, which makes this study an excellent reference for similar experiments.

In Sellberg's doctoral dissertation [9], the difficulties of parameter estimation was discussed. It was stated that parameter estimation in mathematical modeling of chromatographic processes is the most difficult part. Therefore, several types of experimental designs are presented as an aid when performing an estimation. In pulse experiments, characterization of the chromatographic equipment is achieved (e.g. column void volume and porosity of the stationary phase) by pumping a small sample of non-binding component through the column. For estimating parameters in the adsorption model, linear range experiments are used. By loading the column with a small amount of the studied component and eluting at different gradients, the different behaviors for the adsorption model in the linear range of the adsorption isotherm can be observed. Sellberg also presents suggestions on numerical methods for non-linear parameter estimation and selection of initial guess.

3

Method

In this chapter the methods of this thesis is presented. First, the mathematical model is introduced followed by the simulation, parameter estimation and experimental methods.

3.1 Mathematical Model

As described in section 2.2, the system is built up with tubes, columns and other units. From now on, all these will be referred to as *items* in the system. In the simulations, only the tubes and the columns will be discretized and simulated. All other items such as sensors or valves will only work as perfectly mixed tanks with an inlet and an outlet. A sketch over the a tube system is shown in Figure 3.1

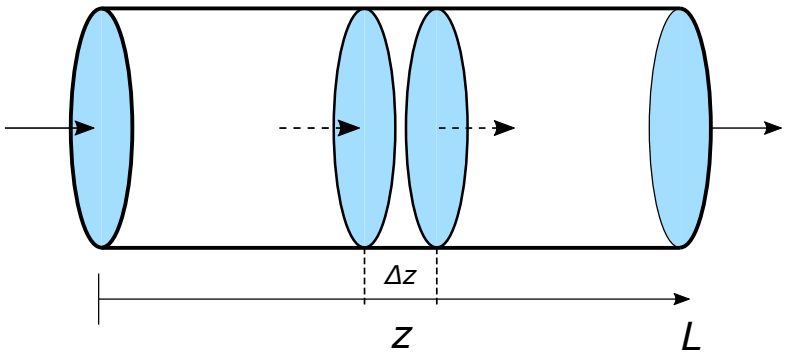

Figure 3.1 A sketch over a tube

where z is the position at a specific point on the tube, Δz is the small step in the length of the tube and L is the total length of the tube. Since each item can be seen as a tube where a fluid flows through, a component mass balance can be set up for a tube[10]. While deriving this model, a few assumptions are made:

- The liquid has constant density throughout the whole tube.
- The system is under isothermal conditions.
- The convection and dispersion is only in the axial direction.
- The concentration within the particles of the column is assumed to be uniform.

With this information, the component mass balance can be derived as following:

$$Rate\ of\ \textbf{Accumulation} = Rate\ of\ Flow\ \textbf{In} - Rate\ of\ Flow\ \textbf{Out}$$

$$A\Delta z\frac{d(\bar{c}_{i,z})}{dt} = (F_z c_{i,z} - AN_z) - (F_{z+\Delta z} c_{i,z+\Delta z} - AN_{z+\Delta z}) \tag{3.1}$$

where A is the cross sectional area of the tube, Δz is the small step in the direction of the tube, $\bar{c}$ is the mean concentration in the mobile phase, t is time, F is the volumetric flow rate, c is the concentration in the mobile phase and N is the dispersion. The subscript i denotes component and the subscript z denotes at what point along the tube. If Δz goes towards zero:

$$\lim_{\Delta z \to 0} = \frac{F_z c_{i,z} - F_{z+\Delta z} c_{i,z+\Delta z}}{A\Delta z} = -v\frac{dc_i}{dz}$$

$$\lim_{\Delta z \to 0} = \frac{AN_z - AN_{z+\Delta z}}{A\Delta z} = -\frac{dN}{dz}$$

where v is the flow velocity and D_{ax} is the dispersion coefficient in the axial direction. Since $N = -D_{ax}\frac{dc}{dz}$

$$-\frac{dN}{dz} = D_{ax}\frac{d^2 c_i}{dz^2}$$

with this information a lumped model can describe the component mass balance. The first term on the right side of the equal sign to represent the convection and the second term is to represent the axial dispersion[10].

$$\frac{\partial c_i}{\partial t} = -v\frac{\partial c_i}{\partial z} + D_{ax}\frac{\partial^2 c_i}{\partial z^2} \tag{3.2}$$

3.1.1 Dispersion Coefficient

The dispersion coefficient can be expressed as

$$D_{ax} = \frac{vL}{\mathrm{Pe}} \tag{3.3}$$

where Pe is the Peclet number which indicates the ratio of convective to dispersive transport. If the magnitude of $Pe \to \infty$, dispersion is negligible and a plug flow behavior is obtained. But if $Pe \to 0$, the system is approaching a well mixed situation [10]. In this thesis, the D_{ax} for the software is set to $1.0 \cdot 10^{-8}$ for both the columns and tubes.

3.1.2 Column Adsorption

Due to the porous packing and the adsorption properties in the ion-exchange column, the lumped model for the column looks a bit different. First, due to the porous packing, the flow velocity needs to be substituted into the interstitial flow velocity as following:

$$u_{int} = \frac{v}{\varepsilon}$$

where u_{int} is the interstitial flow velocity and ε is the total fluid fraction of the column which can be calculated from Equation 3.4:

$$\varepsilon = \varepsilon_c + (1 - \varepsilon_c)\varepsilon_p \tag{3.4}$$

where ε_c is the column void fraction and ε_p is the packing porosity. For this thesis, these parameters will be be set to the values used in Persson's master thesis [8]. These values are found in Table 3.1.

Table 3.1 Values for column void and packing porosity[8].

Parameter	Value
ε_c	0.319
ε_p	0.856

Secondly, an adsorption term is also added to describe the separation. There are several type of adsorption models mentioned in section 2.4, and in Persson's master thesis, model parameters for three proteins is established. Due to this, the simplified SMA model is chosen so that the model parameters from his master thesis can be used as a reference in this work. The model is simplified in the sense that the salt is regarded as an inert component that does not adsorb. The model is described in Equation 3.5:

$$\frac{\partial q_i}{\partial t} = k_{kin,i}\left(K_{eq,i}\cdot c_i\left(1-\sum_{j=1}^{N_{comp}}\frac{q_i}{q_{max,j}}\right)^{\nu_i} - q_i\cdot c_s^{\nu_i}\right) \tag{3.5}$$

where q is the concentration in the stationary phase, k_{kin} is the kinetic adsorption coefficient, K_{eq} is the equilibrium constant, N_{comp} is the total number of components and ν is the characteristic charge. The subscript s denotes that it is salt. With these additions, the lumped model for the adsorption column is defined as in Equation 3.6 [11]:

$$\frac{\partial c_i}{\partial t} = -\nu\frac{\partial c_i}{\partial z} + D_{ax}\frac{\partial^2 c_i}{\partial z^2} - \frac{1-\varepsilon_c}{\varepsilon}\frac{\partial q_i}{\partial t} \tag{3.6}$$

3.1.3 Initial and Boundary Conditions

To simulate all the items in the system, boundary and initial conditions needs to be set up for each item. At each inlet of the items, a Dirichlet condition is set as seen in Equation 3.7.

$$c_i\big|_{z=0} = c_{in,i}(t) \tag{3.7}$$

where the subscript *in* indicates the inlets concentration for the tube. For each outlet of the items, a von Neumann condition is set as seen i Equation 3.8.

$$\left.\frac{\partial c_i}{\partial z}\right|_{z=L} = 0 \tag{3.8}$$

At the beginning of each simulation, the initial conditions is set so that the system is filled with a solvent with a low salt concentration. The initial condition can be seen in Equation 3.9.

$$c_{init}(t=0,z) \tag{3.9}$$

where the subscript *init* indicates that it is the initial concentration.

3.2 Method of Simulation

In this section, the method of the simulation will be discussed. This software is based on preexisting software written by Niklas Andersson at the Department of Chemical Engineering at Lund University. The software has been further developed throughout this thesis.

3.2.1 Creating a Mesh

With the mathematical model for tubes and columns stated in section 3.1, the system can be simulated. Each item is discretized into a mesh with a number of grid points where the first grid point is the inlet and the last is the outlet. The last grid point is also connected to the next item's first grid point, creating a long chain of connected items. Depending on what type of item, different number of grid points were used. In Table 3.2, the number of grid points used are stated.

Table 3.2 The number of grid points used in the simulations

Item	Number of Grid Points
Tube	10
Column	50

With increasing the number of grid points, the accuracy of the simulation increases to the expense of an increasing number of calculations. When the number of calculations is increased, it results in an increase in total calculation time for the simulation. This gives a trade-off between accuracy and total simulation time. A low number of grid points also increases the numerical dispersion in the system meaning that this effect will be more present than the actual dispersion in the system.

The number of grid points is chosen based on trials by Oliver Persson in his master thesis at the Department of Chemical Engineering at Lund University [8]. He found that a higher number than 50 grid points for the column does not give higher accuracy because the numerical dispersion is low enough. So for this system, the number of grid points in the tubes are a low, but since the accuracy in the tubes are not as important as in column, this is regarded as feasible for this study. The same goes for the number of grid points in the column which can also be regarded as low, but increasing this would create a very big system with high calculation times. Therefore 50 grid points are kept for the column.

3.2.2 Method of Lines

To discretize in space, the method of lines (MOL) is used to turn the partial differential equations (PDEs) into larger systems of ordinary differential equations (ODEs) [12]. To approximate the derivatives at each element, the finite volume method is utilized [13]. For the first-order derivatives, a two-point backward difference is used and for the second-order derivatives a three-point central difference is used to make the estimation. This can be seen in Equation 3.10 respectively 3.11.

$$\frac{\partial c_n}{\partial z} = \frac{c_n - c_{n-1}}{h} \tag{3.10}$$

$$\frac{\partial^2 c_n}{\partial z^2} = \frac{c_{n+1} - 2c_n + c_{n-1}}{h^2} \tag{3.11}$$

where h is distance between two grid points. The subscript n declares the grid point index for the item.

3.2.3 The Simulation Procedure

Thanks to the integration of the simulation software into *Orbit*, an experienced user of *Orbit* will not have to change any syntax in the method creation. When setting up phases for the desired simulation experiment, there is no difference compared to setting up a regular experiment. The user does have to specify what components that will be present in the system and

what types of buffers the system is connected to. The user also has the option to turn on and off the simulation software depending on the desired experiment. When the specified method is sent to the simulation software, the following procedure takes place:

1. All the ports (inlet and outlets of items) are collected, simulation parameters (e.g. number of grid points and dispersion coefficient) are set, initial states are saved and active pumps are defined.
2. The software checks so that in all instructions, there is a buffer connected to the system. If there is not one set, a new buffer is created and connected. All the components of the system is also stored.
3. An empty vector for the total solution of all items is created. After the vector is created, the initial values for every point is gathered and set.
4. A for-loop over all instructions in the method is started:
    a) The flowpath for the current instruction is found. The inlet and outlets of the system is set and flow direction for all items. An empty vector is created for the items that will be simulated during the current instruction. The vector is then filled with the solution from the previous simulated instruction.
    b) The current instruction is simulated
    c) The results from the simulation is added to the total solution vector.
    d) Return to the top and continue with the new instruction.

Note that this description of the simulation procedure is a simplification. The steps mentioned above are the fundamental steps that one need to understand. Many more steps do occur in the procedure, but are not necessary for the general understanding.

3.2.4 Solving ODE:s in Python

Since *Orbit* is written in the programming language *Python*, the simulation software also uses *Python* due to the simplicity while integrating both the systems.

To solve a system with ODE:s, the function *solve_ivp* in the library of *scipy.integrate* is very useful. The function numerically integrates a system of ODE:s given an initial value. A user can specify what type of integration method to be used. For this thesis, backward differentiation formula (BDF) was chosen based on simulation experience at the Department of Chemical Engineering at Lund University. The method is an implicit multi-step method with a variable order based on the derivative approximation [14].

3.2.4.1 The Usage of the Jacobian Sparsity Matrix

One way to speed up the numerical calculations of the ODE solver is to pass on a Jacobian sparsity matrix. The idea is that the solver gets a matrix which tells the solver which state variables that will affect the partial derivatives. This way the solver does not have to calculate the Jacobian at every given point, instead it can just follow the given matrix.

The Jacobian gives the relationship between the existing state variables in the system and their partial derivative. This gives a (n,n) matrix with the partial derivatives representing the rows and the state variables being the columns. The script has the following procedure [15]:

1. Creates an empty matrix with the size of the Jacobian
2. Calculates the initial partial derivatives of the system with some randomized initial values.

3. Then one state variable at the time is changed. New partial derivatives is calculated and the ones that changed at each time is noted.

4. In the end a matrix filled with zeros and ones is formed. The zeros represent that the state variable has no effect on the partial derivative and vice versa.

An illustration over how a Jacobian sparsity matrix can look like if represented as above is shown in Figure 3.2. The blue boxes represents where the state variables have a direct impact on the partial derivatives.

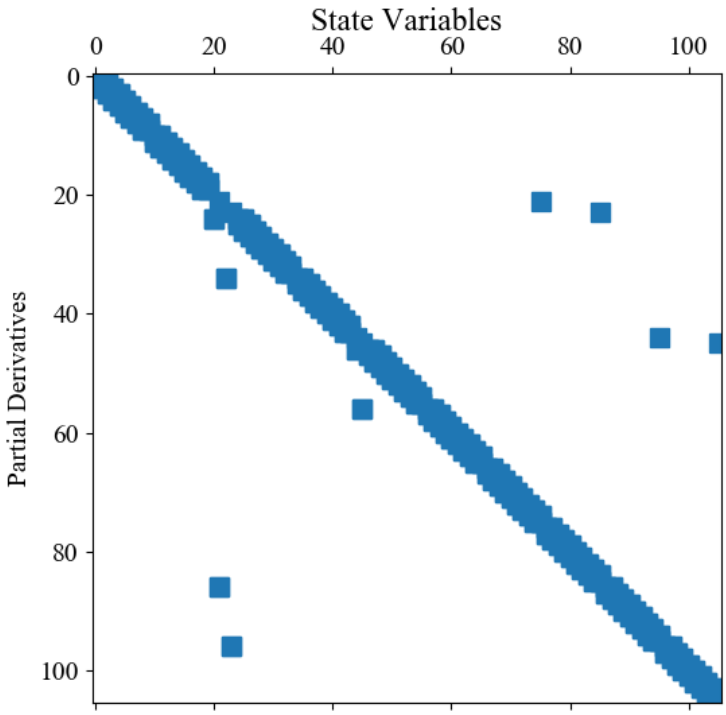

Figure 3.2 An illustration over a Jacobian sparsity matrix for a system. The blue boxes represent where the state variables have a impact on the partial derivative.

3.3 Method of Parameter Estimation

With the results from the simulations, a parameter estimation of the components is possible if some real data of the same run-method is given. The parameter estimation can be seen as a "calibration" of the model curve to the actual data. For this thesis, the focus will be on estimating k_{kin}, K_{eq} and ν of the protein components. This is done in steps by minimizing two different objective functions.

3.3.1 Objective Functions

The first objective function aims to minimize the distance between the experimental and the simulated peak. A weight is also introduced so that the user can specify whether the concentration or the retention time is more important. The objective function can be seen i Equation 3.12

$$Q_{dist} = \sqrt{\left(w \cdot (t_{ret,data} - t_{ret,sim})\right)^2 + \left((1-w) \cdot (c_{ret,data} - c_{ret,sim})\right)^2} \tag{3.12}$$

where Q_{dist} is the objective function for distance between peaks, w is the weight on the retention time of the objective function and t_{ret} is the retention time. The subscript *data* denotes experimental data and *sim* denotes simulated data. t_{ret} and c_{ret} is found by describing the profile with splines and using the bisection method. This is necessary to remove discrete behavior. If the goal of the objective function is to minimize the retention time only, the weight should be set to 1, thus the concentration difference will be zero. An illustration of the variables can be found in Figure 3.3.

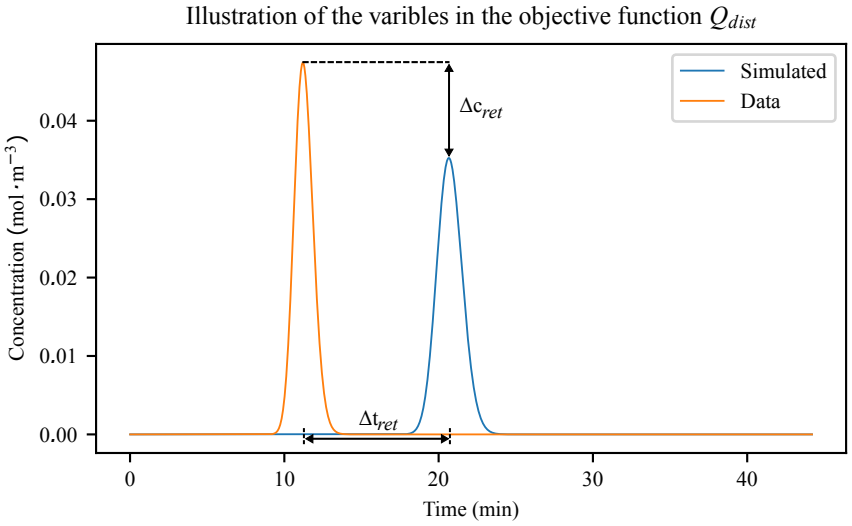

Figure 3.3 An illustration of the variables in the distance measurements for Q_{dist}.

The second objective function aims to minimize the sum of the squared residuals between the experimental data and the simulated result. The function can be seen in Equation 3.13.

$$Q_{res} = \sum_{j=1}^{N_{obs}} \left(c_{j,data} - c_{j,sim}\right)^2 \tag{3.13}$$

where Q_{res} is the objective function for the squared residuals and N_{obs} is the number of observed values.

3.3.2 Minimize Methods

There are different numerical methods for minimizing the objective function seen in Equation 3.12. In the library *scipy.optimize*, the function *minimize* gives the user over 10 different methods for minimization. As a user, you specify the objective function which is to be optimized and what parameters that the method can adjust to achieve better results. In this case, the objective function can be found in Equation 3.12 and the parameters to be adjusted are k_{kin}, K_{eq} and ν. For this thesis, three different methods for minimization are used. Their description from the function's documentation is listed below [16]:

- **COBYLA** - Based on linear approximations to the objective function, this is a constrained Optimization by Linear approximation method.
- **SLSQP** - Uses sequential least squares programming to minimize a function of several variables with any combinations of bounds, quality and inequality constraints.

- **Nelder Mead** - Uses the simplex method. It is a very robust method, but slow compared to the other two methods if the numerical computation of the derivative can be trusted.

For the objective function found in Equation 3.13 the library *scipy.optimize* has a function called *curve_fit* which creates the objective function itself as long as the user provides the data, simulated results and adjustable parameters [17]. This function uses the Levenberg-Marquardt algorithm to solve the minimization of the sum of the squares. In this thesis, the method will not be further explained. For further robustness, the *least_squares* function in the *scipy.optimize* was also used where a Trust Region Reflective algorithm (trf) with a 3-point finite difference scheme for numerical estimation of the Jacobian proved to be very effective. This method was more robust than Levenberg-Marquardt, but the number of calls to the simulation function is twice as high thus increasing computational time.

3.3.3 The Parameter Estimation Procedure

The goal is to build a robust software that can take few sets of data and make a parameter estimation for a simplified SMA model. Throughout the minimization of the objective functions, the user is asked to do an initial guess. This puts the user in a tricky situation since one does not always have knowledge of the component that is under investigation. This means that the parameter estimation procedure needs to be designed so that with a bad initial guess, robust methods will reach the solution in any case. There are many numerical difficulties to handle to reach the solution. Each of the three parameters are strongly coupled to each other and shows different sensitivity in different areas e.g. k_{kin} shows different sensitivity depending on the values of K_{eq} and ν. Thus, the parameters also show more direct effects on retention time and peak width:

- A higher k_{kin} gives a smaller peak width and vice versa.
- A higher K_{eq} increases retention time and vice versa.
- ν is strongly coupled to both retention time and peak width. But its effects can be exposed by changing salt concentration since these two are strongly coupled.

A general description of the procedure is shown below. In all minimizations, both the data set for the 8 min linear gradient and the 20 min linear gradient was included. Note that this does not include all safety methods (methods created in case a minimization method crashes or reaches a bad result), but presents an insight to the software.

1. Adjust K_{eq} until the peak elute within the elution phases. If K_{eq} is too low, the component will not be adsorbed by the column and therefore be pumped right through the column. But if K_{eq} is too big, the adsorption term will be too strong thus binding all the component and nothing will elute. In either of these cases, the minimize methods have no chance of finding the minimum of Equation 3.12. This is due to a too big or too low K_{eq}, there is no sensitivity meaning that increase or decrease of K_{eq} does not affect the objective.

2. K_{eq} is estimated by minimizing Equation 3.12 so that the retention time of the data and the model is identical.

$$\begin{aligned} \min_{K_{eq}} \quad & Q_{dist} \\ \text{s.t.} \quad & \text{Equation } 3.5 - 3.9 \\ & w = 1 \end{aligned} \tag{3.14}$$

3. k_{kin} is adjusted so that it shows sensitivity. This is done by calculating the residuals between the data and the simulated model for a number of different k_{kin} values. The newly selected k_{kin} is based on where the derivative of the residuals is the biggest.

4. All parameters are estimated by minimizing Equation 3.13 with the function *curve_fit* presented in section 3.3.2.

$$\begin{aligned} \min_{k_{kin},K_{eq},\nu} \quad & Q_{res} \\ \text{s.t.} \quad & \text{Equation } 3.5 - 3.9 \end{aligned} \tag{3.15}$$

5. k_{kin} is estimated by minimizing 3.12 so that the top of the simulated peak gets as close as possible to the data peak. This step is done to increase the sensitivity of k_{kin}.

$$\begin{aligned} \min_{k_{kin}} \quad & Q_{dist} \\ \text{s.t.} \quad & \text{Equation } 3.5 - 3.9 \\ & w = 0.7 \end{aligned} \tag{3.16}$$

6. All three parameters are under estimation while minimizing Equation 3.13 using the *least_squares* function with the trf-method. This follows the same minimization objective as in Equation 3.15.

3.4 Experimental Methods

In this section, the method for the different experimental cases is presented.

3.4.1 Experimental System

It is important to remember that the model simulations and parameter estimations for this system is only applicable for this system. The model is very sensitive, e.g. if the pH would change or a different column would be used in the experiments, the model would no longer be applicable [6].

The column used in this thesis is the *HiTrap SP HP 1 ml* from *GE Healthcare* which has a particle diameter of 24-44 μm, bed diameter of 7 mm and a height of 25 mm [18]. The solutions used in the experiments are stated in Table 3.3.

Table 3.3 Solutions used in the simulations.

Solution	NaCl	Protein
Buffer A	0.5 mM	-
Buffer B	0.5 M	-
Protein solution	-	1 g/L

For simulating experimental data of proteins, the model parameters from Persson's master thesis is used [8]. These parameters can be seen in Table 3.4. Note that the parameter q_{max} was not investigated in this thesis and therefore assumed to be according to Persson's findings.

Table 3.4 SMA model parameters for ribonuclease A, cytochrome c and lyzozyme [8].

Parameter	Ribonuclease A	Cytochrome c	Lysozyme	Unit
k_{kin}	$1.0 \cdot 10^{-2}$	$1.0 \cdot 10^{-4}$	$1.0 \cdot 10^{-6}$	$m^3\ mol^{-1}\ s^{-1}$
K_{eq}	$6.8 \cdot 10^{7}$	$5.0 \cdot 10^{10}$	$4.0 \cdot 10^{12}$	$mol\ m^{-3}$
ν	3.3	4.1	4.7	-
q_{max}	51	51	51	$mol\ m^{-3}$

In Figure 3.4, an illustration on the simulated ÄKTA Pure system is shown and in Table 3.5 a description of the items in the figure is shown. In the beginning of the simulations the whole system is filled with Buffer A except for the super loop that is filled with the protein solution.

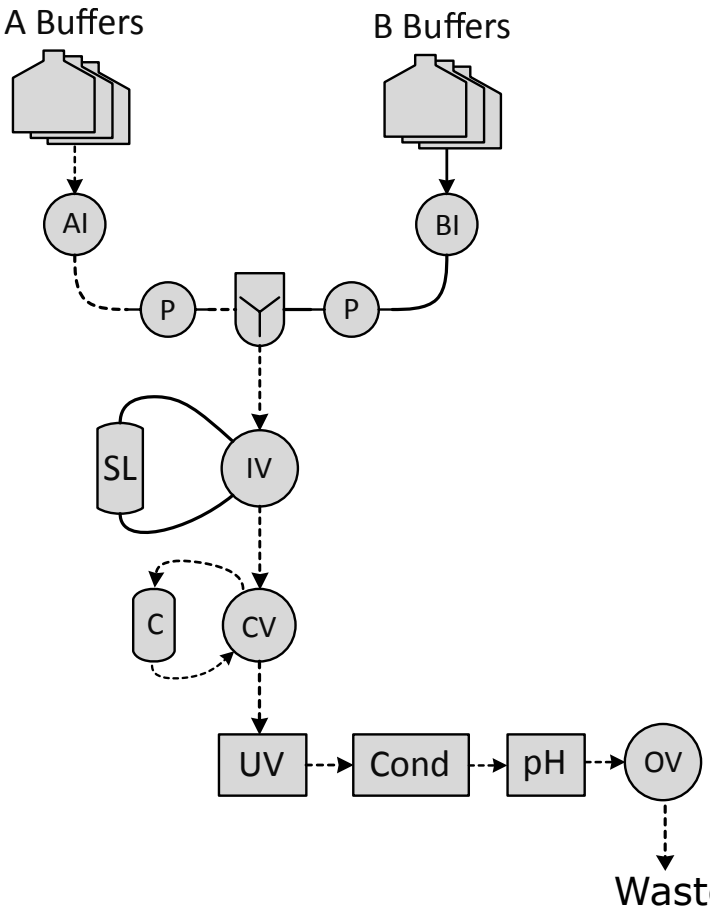

Figure 3.4 An illustration how the units and tubes are connected in the simulated ÄKTA Pure system. The dotted lines are the main flow path, where as the solid line is the optional flow paths.

Table 3.5 Explanation of abbreviations from Figure 3.4

Item	Explanation
AI	Inlet valve A
BI	Inlet valve B
P	Pump
IV	Injection valve
SL	Super-loop
CV	Column valve
C	Column
UV	UV sensor
Cond	Conductivity sensor
pH	pH sensor
OV	Outlet valve

With this simulated flow path, the system consists of:

- 12 tubes
- 1 column
- 4 items (mixer, UV-sensor, conductivity sensor and pH-sensor acting as perfectly mixed tanks)

This result in a system where the number of state variables can be described by the number of components and the number of grid points (see Table 3.2). This is shown in Equation

$$N_x = N_{comp} \cdot (N_{tubes} * GP_{tubes} + 2 \cdot (N_{columns} \cdot GP_{columns}) + N_{items}) \tag{3.17}$$

where N_x is the number of state variables, N_{tubes} is the number of tubes, GP_{tubes} is the number of grid points for tubes, ($N_{columns}$ is the number of columns, $GP_{columns}$ is the number of grid points for columns and N_{items} is the number of items in the system. The amount of state variables for the columns is doubled since both concentration and adsorption is modeled in the columns and for the items there is no grid points. For the system in this thesis with 2 components, this results in a total of 448 state variables.

3.4.2 Simulation Case 1 - Simulation of 3 components

Before this thesis, the simulation software was based on a component mass model for a tube system without the adsorption term. Throughout this thesis, an adsorption term was added to simulate ion-exchange column behavior in the ÄKTA Pure system. To showcase the functionality of the simulation software, the three component system described in section 3.4.1 were simulated in a linear gradient system. In Table 3.6, the experimental run method is described.

3.4.3 Integrated Parameter Estimation

Note that all the data for these cases was based on previous experimental parameters from Persson's master thesis [8]. Using parameters from his work, experimental data was simulated using the simulation software. This means that no practical experimental data was obtained from an GE Äkta Pure system.

In section 3.3.3 the general parameter estimation procedure in the software is presented. To showcase the result of every step, a guess was provided to the software and after every

Table 3.6 An overview over the phases in the simulation linear gradient experiments

Phase name	Buffer	Time (min)
Flush	100% Buffer A	1
Inject	Injection of protein	1
Wash	100% Buffer A	1
Gradient elution	From 20% to 100% Buffer B	8
Final Elution	100% Buffer B	5
Regenerate	100% Buffer A	5

step in the procedure, the estimated parameters were saved. The software was provided of experimental data representing lysozym. The guess for the parameter estimation procedure investigation is shown below:

$$k_{kin} = 1.0 \cdot 10^{-6}\ m^3\ mol^{-1}\ s^{-1}$$

$$K_{eq} = 5.0 \cdot 10^{14}\ mol\ m^{-3}$$

$$\nu = 3.5$$

To test the robustness of the parameter estimation software, three different cases were investigated with a different protein in each case. Two experimental data sets were created for each case using a linear gradient. The first dataset had a gradient of 8 min and the second had a gradient of 20 min. An overview of the whole method for the simulation of the datasets and cases can be seen in Table 3.7.

Table 3.7 An overview over the phases in the parameter estimation experiments

Phase name	Buffer	Time (min)
Flush	100% Buffer A	1
Inject	Injection of protein	1
Wash	100% Buffer A	1
Gradient elution	From 20% to 100% Buffer B	8 or 20
Final Elution	100% Buffer B	15
Regenerate	100% Buffer A	6

The investigated cases were very similar. For each case, three different initial guesses were made. Two of the guesses were based on the protein that was not under investigation e.g. if the experimental dataset was made with ribonuclease A, two of the initial guesses were the parameters of cythochrome c and lyzozyme. The last guess was based on the parameters that the software will use if the user does not provide a guess. In total there were four different guesses made in this thesis: three based on the actual experimental parameters and one default by the software. These were referred to as:

- Initial Guess Ribonuclease A: (Initial Guess R)
- Initial Guess Cythochrome C: (Initial Guess C)
- Initial Guess Lysozym: (Initial Guess L)
- Initial Guess Default: (Initial Guess D)

The three cases and their guesses are presented in Table 3.8, Table 3.9 and Table 3.10 which are defined as case 2-R, case 2-C and case 2-L.

Table 3.8 Initial guesses for the parameters of ribonuclease A in case 2-R.

Parameter	Initial Guess D	Initial Guess C	Initial Guess L	Unit
k_{kin}	$1.0 \cdot 10^{-10}$	$1.0 \cdot 10^{-4}$	$1.0 \cdot 10^{-6}$	$m^3\ mol^{-1}\ s^{-1}$
K_{eq}	$2.0 \cdot 10^{9}$	$5.0 \cdot 10^{10}$	$4.0 \cdot 10^{12}$	$mol\ m^{-3}$
ν	4.2	4.1	4.7	-

Table 3.9 Initial guesses for the parameters of cythochrome c in case 2-C.

Parameter	Initial Guess R	Initial Guess D	Initial Guess L	Unit
k_{kin}	$1.0 \cdot 10^{-2}$	$1.0 \cdot 10^{-10}$	$1.0 \cdot 10^{-6}$	$m^3\ mol^{-1}\ s^{-1}$
K_{eq}	$6.8 \cdot 10^{7}$	$2.0 \cdot 10^{9}$	$4.0 \cdot 10^{12}$	$mol\ m^{-3}$
ν	3.3	4.2	4.7	-

Table 3.10 Initial guesses for the parameters of lysozyme in case 2-L.

Parameter	Initial Guess R	Initial Guess C	Initial Guess D	Unit
k_{kin}	$1.0 \cdot 10^{-2}$	$1.0 \cdot 10^{-4}$	$1.0 \cdot 10^{-10}$	$m^3\ mol^{-1}\ s^{-1}$
K_{eq}	$6.8 \cdot 10^{7}$	$5.0 \cdot 10^{10}$	$2.0 \cdot 10^{9}$	$mol\ m^{-3}$
ν	3.3	4.1	4.2	-

For each of the cases the experimental parameters compared to the initial guess and the estimation will be presented. In some of the cases, the estimation is so good that a plot of the experimental simulation and estimated simulation would not show a big difference. Therefore the residual sum of squares (RSS) for between the simulated data and simulation with the parameter estimation is also provided to give the reader a sense of how close the parameter estimation is. The number of calls to the simulator for the different parameter estimations (SimCalls) will also be provided.

4

Results & Discussion

In this chapter, the results from the experimental methods and its cases is shown together with a discussion of the model behavior and the software implementation.

4.1 Simulation Case 1 - Simulation of 3 components

Since the whole system was simulated by the software, every item with its own grid points has its own solution. This means that for every item (e.g. a tube or a valve), the concentration of components and salt can be shown over time. To show this, a few items are plotted over time below in Figure 4.1, Figure 4.2, Figure 4.3 and Figure 4.4.

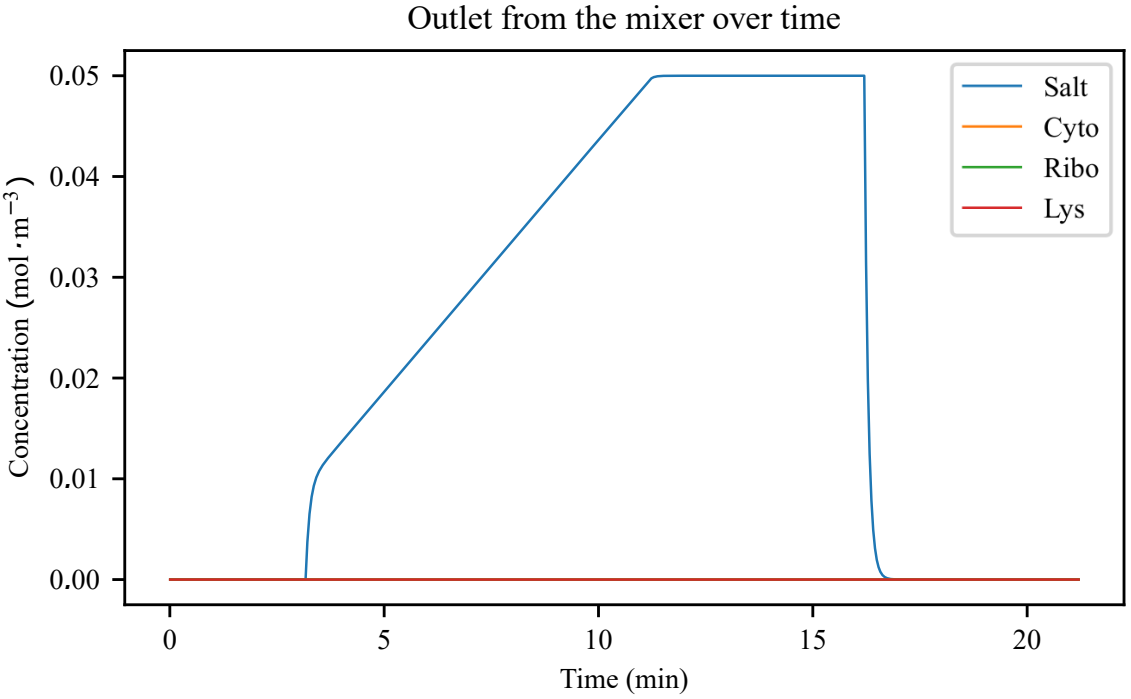

Figure 4.1 Concentration of the components at the outlet of the mixer over time. Note that the salt concentration is scaled by 10^{-4} to illustrate the elution sequence in the same figure.

From the figures, one can follow the components through the whole system. At the mixer, the only component that is visible is the salt. This is because the protein components never pass the mixer since they are injected at a later stage of the system. In Figure 4.2, the protein components passes as pulses right before they enter the column. The protein solution contained 1 g/L of each protein, but since all proteins have different molar masses, the concentration of them differ.

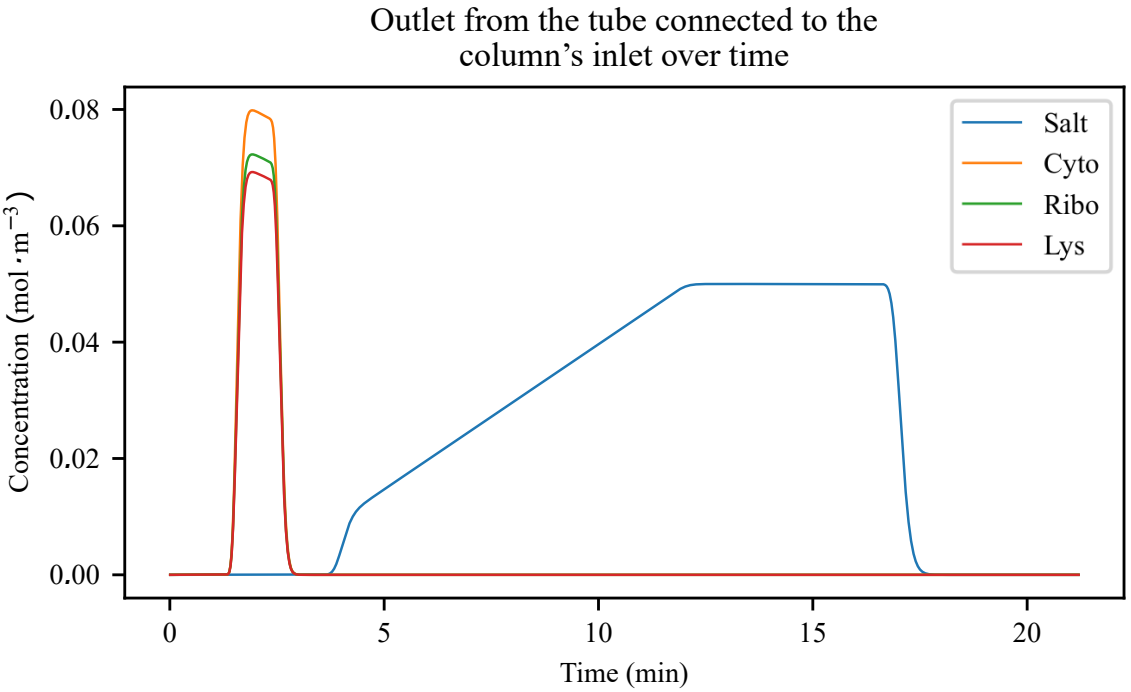

Figure 4.2 Concentration of the components at the outlet of the tube that is connected to the column over time. Note that the salt concentration is scaled by 10^{-4} to illustrate the elution sequence in the same figure

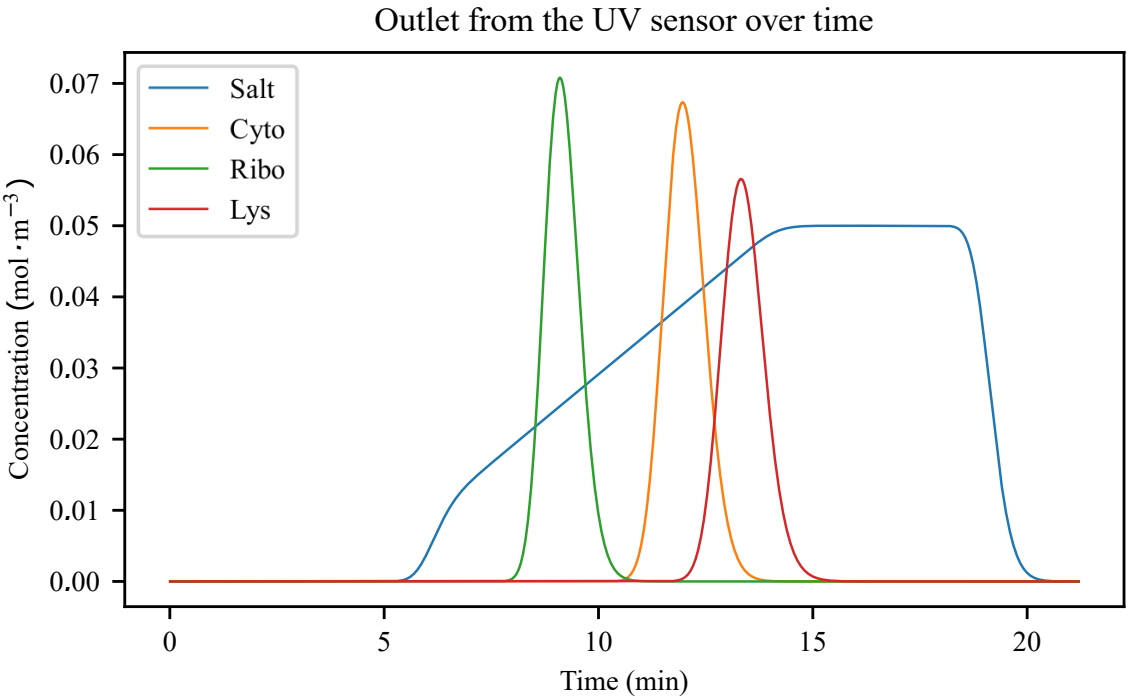

Figure 4.3 Concentration of the components at the outlet of the UV sensor over time. Note that the salt concentration is scaled by 10^{-4} to illustrate the elution sequence in the same figure

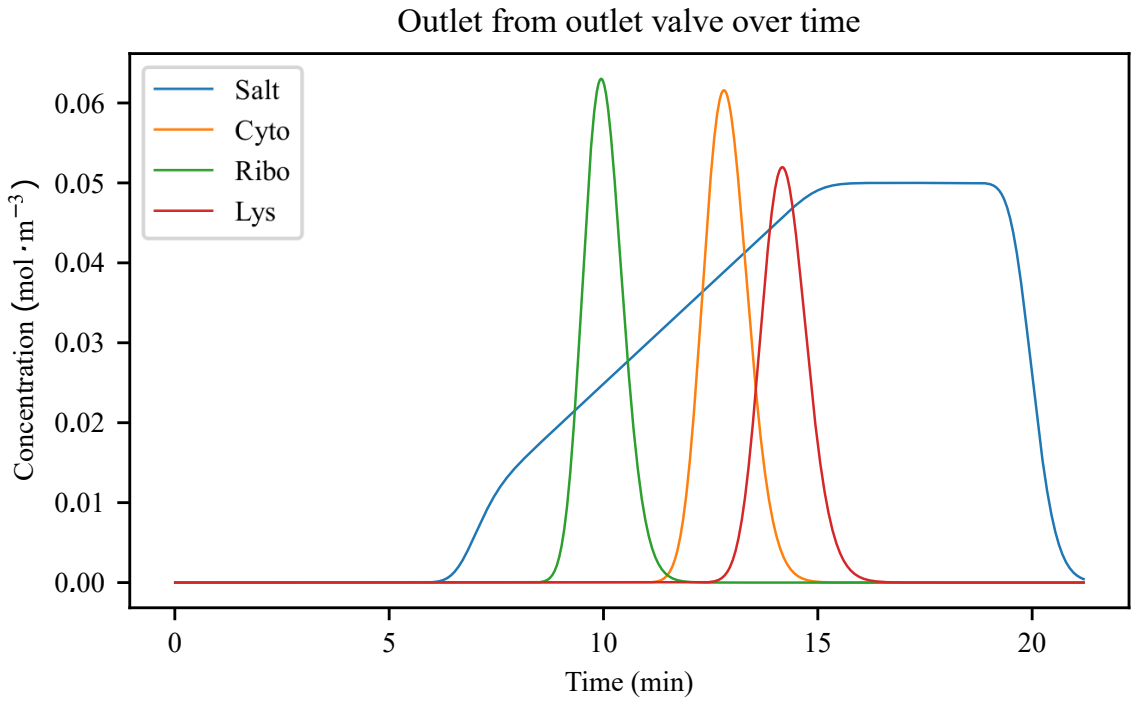

Figure 4.4 Concentration of the components at the outlet of the outlet valve over time. Note that the salt concentration is scaled by 10^{-4} to illustrate the elution sequence in the same figure

At the outlet of the UV sensor, the separation done by the column can be seen since the protein components are now shaped as peaks instead of the pulse that was seen before the column. If comparing the results from the UV sensor (Figure 4.3) and the outlet valve (4.4) it is clear that the peaks are passing at different times. This shows that there is a distance for the solution to travel before reaching the next unit, in this case tubes. Therefore, the peaks will reach each items at different times. This can also be seen on the salt curve on each figure. For the mixer, the salt concentration returns to 0 around 17.5 min and at the outlet valve the salt concentration returns to zero at 21.5 min.

The effect of the dispersion term can also be seen when comparing the salt concentration of the mixer and the outlet valve. At the mixer, buffer A and buffer B meet for the first time and is instantly mixed. Since there is no modeling done for describing the mixing, this results in the mixer acting as a perfectly mixed tank. This kind of mixing is probably not the case in reality, its effect is not in the scope of the study and was therefore not further evaluated. Looking at the salt concentration of the outlet valve, the curve for the salt is much smoother. This is because of the dispersion of the salt through out the whole system. An an increase of smoothness around the salt curve can also be seen by looking at the figures in the order of the flow path. The dispersion can also be seen on the protein components if comparing the peaks at the UV sensor and the outlet valve. At the UV sensor, ribonuclease A and cythochrome c barely overlaps. But at the outlet valve, these two components show a slightly larger overlap. This shows that the dispersion term is active for the whole system, not only the salt.

Keep in mind that due to the numerical dispersion, from the low number of grid points, this effect overlaps the dispersion term. To simulate the dispersion more precisely, the number of grid points should be chosen to a higher number, and the dispersion coefficient should be adjusted to fit the simulated system. For instance, the dispersion coefficient is much higher in a column than in a tube due to the packing in the column.

4.1.1 The Effect of the Jacobian Sparsity Matrix

To prove that the Jacobian sparsity matrix decreases the calculation time as described in section 3.2.4.1, the time for a simulation to complete was measured with and without the Jacobian sparsity matrix. A mean was calculated over 10 simulations for each of the cases. This was measured for the simulation in Case 1 explained in section 3.4.2.

When the Jacobian sparsity matrix was given to the ode-solver, the simulation time was 35.4 s while without 94.6 s. This shows that the Jacobian sparsity matrix makes the ode-solver 2.5 times faster in this simulation (Reduced the computational time by approximate 60 %). From these results, the Jacobian sparsity matrix has a great impact on computational time. If the matrix would not be given to the ode-solver, computational time would greatly increase. This would result in a problem in the parameter estimation software where the minimization methods calls the simulation function over 800 times resulting in very large computational times.

4.2 Parameter Estimation Procedure

To illustrate the sensitivity of k_{kin}, the derivative of objective function Q_{dist} in respect to k_{kin} is shown in Figure 4.5.

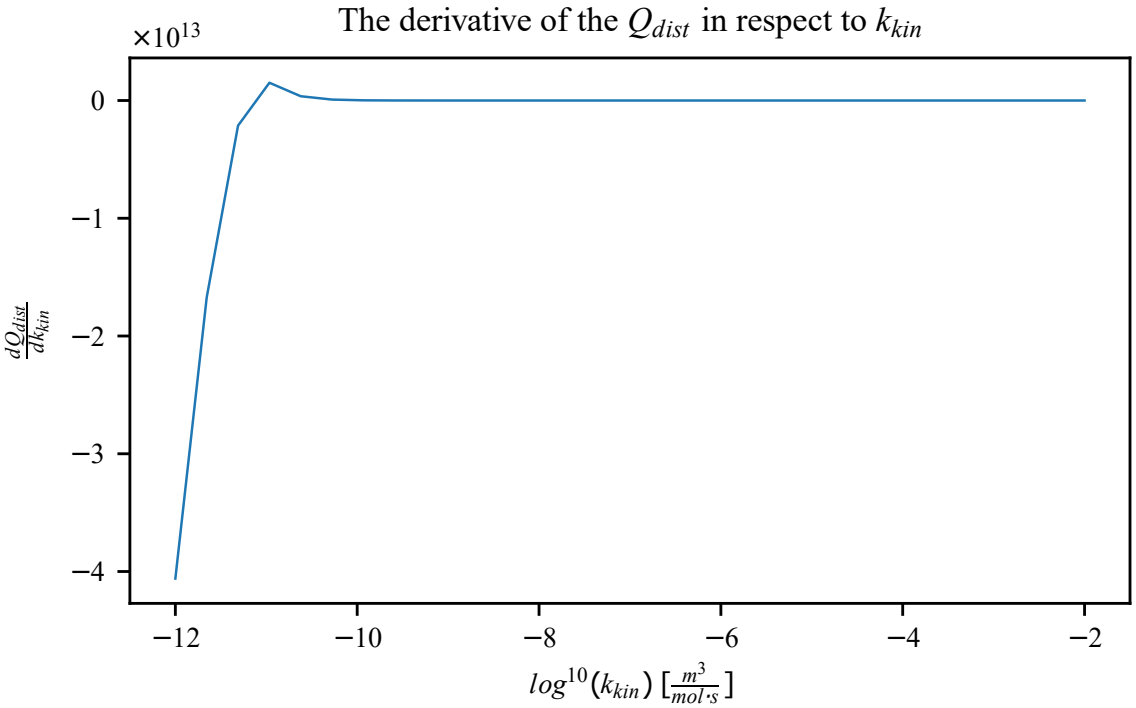

Figure 4.5 The derivative of Q_{dist} with respect to k_{kin}

From the figure, the sensitivity of k_{kin} can be observed. With very low k_{kin}, the peak will not elute and therefore the peak is outside of the elution window (the elution window is the time span where buffer B is introduced to the system). When increasing k_{kin} slightly the peak enters the elution window and a smaller value on Q_{dist} is achieved and therefore the derivative is very high for these values since the distance step is very high. When k_{kin} increases more, the derivative decreases rapidly reaching a very low value in comparison to the high sensitivity case. This lower derivative resulted in a lower sensitivity of the parameter which leads to less room of estimation possibilities for the *curve_fit* function.

To showcase the different steps taken by the parameter estimation software, an illustration of the steps taken can be seen for both a 8 min linear gradient and a 20 min linear gradient in Figure 4.6 and Figure 4.7 respectively. In Table 4.1, the estimated parameters for each step is shown.

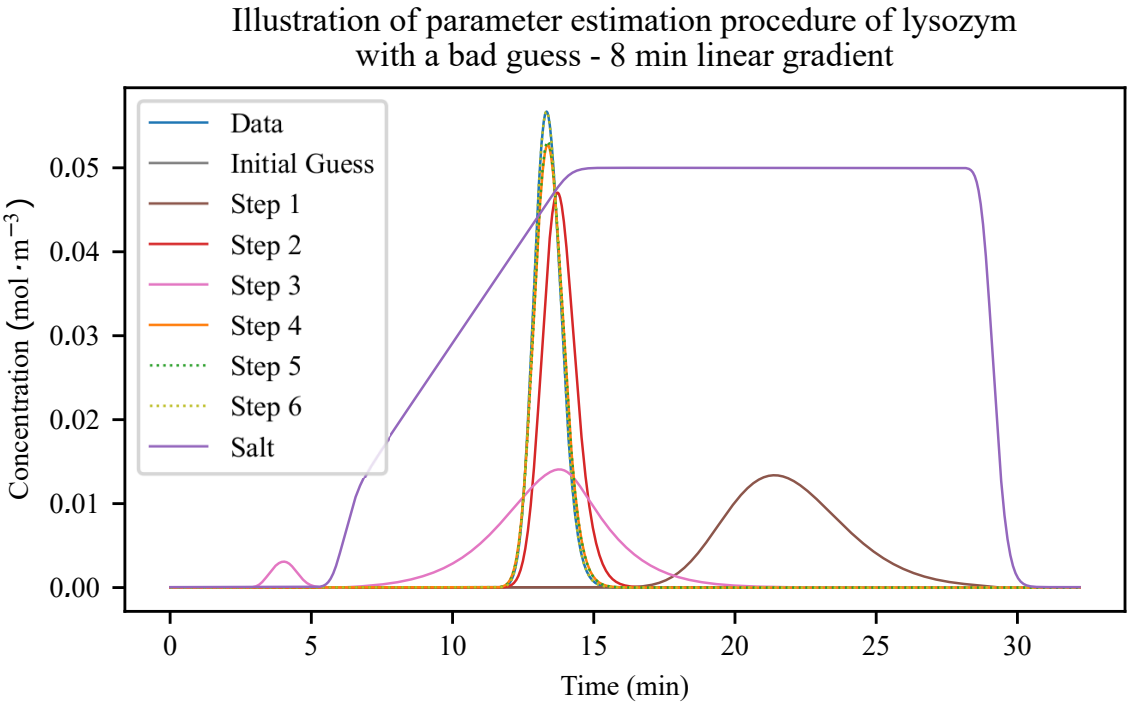

Figure 4.6 An illustration of the steps taken in the parameter estimation procedure in the software for a 8 min linear gradient. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

Table 4.1 Parameter estimation procedure result for one component.

Step	k_{kin}	K_{eq}	ν
Data	$1.00 \cdot 10^{-6}$	$4.00 \cdot 10^{12}$	4.70
Initial Guess	$1.00 \cdot 10^{-6}$	$5.00 \cdot 10^{14}$	3.50
Step 1	$1.00 \cdot 10^{-6}$	$2.10 \cdot 10^{10}$	3.50
Step 2	$1.00 \cdot 10^{-6}$	$3.92 \cdot 10^{9}$	3.50
Step 3	$1.00 \cdot 10^{-11}$	$3.92 \cdot 10^{9}$	3.50
Step 4	$1.21 \cdot 10^{-10}$	$1.73 \cdot 10^{11}$	4.17
Step 5	$3.99 \cdot 10^{-9}$	$1.73 \cdot 10^{11}$	4.17
Step 6	$4.52 \cdot 10^{-9}$	$4.00 \cdot 10^{12}$	4.70

At a first glance, the peak for the initial guess can not be found in either of the figures. But by looking at the size of K_{eq} for the guess, one can assume that the binding of the component was too strong and therefore no component leaves the column for the initial guess. In step 1, the peak was visible in both figures and shows that the software was able to bring the peak into the elution window of the simulation. This was done by lowering the size of K_{eq} so that the peak was visible during the elution phase. In this case, the software recognized that there was no peak visible from the initial guess meaning that the adsorption term is too strong. If

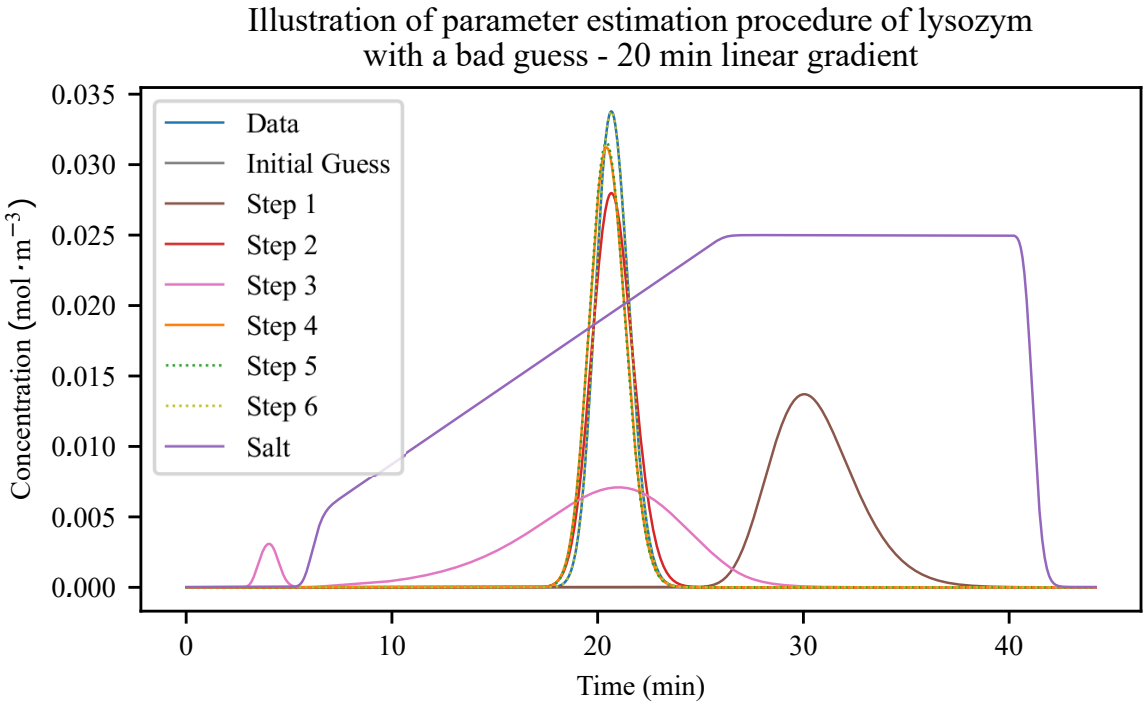

Figure 4.7 An illustration of the steps taken in the parameter estimation procedure in the software for a 20 min linear gradient. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

the adsorption was too low, the whole peak would instead elute before any salt was added into the column and the software would instead increase K_{eq} to put the peak in the elution window.
After step 2, K_{eq} was estimated so that the retention times of the data and the simulated case was the same. In both figures, the retention times were very similar between the data and the "Step 2". However the peak height was not very similar and the shape of the peak was a bit off.

In step 3, the peak looks a bit different. Here k_{kin} was set to a much lower value where it shows sensitivity. The result of the decreased k_{kin} contributed to a much lower peak height than the data and a peak that was much broader. But this step was still implemented since it was shown in experiments that without setting the k_{kin} to a value where it shows high sensitivity, the *curve_fit* was not able to make a good estimation. And since this function is much faster than the trf-method, this k_{kin} adjustment step is necessary to increase the speed of the software. A small peak before the elution phase starts can also be seen in the figures. The explanation for this was that since the adsorption term was very low due to the low k_{kin} value, very small amounts of component was adsorbed in every grid point and therefore a small amount managed to go through the whole column without being adsorbed.

Step 4 utilized the *curve_fit* function and managed to get closer to the real parameters but both the peak height and retention time was still not fitted to the data. In this case, step 5 did not seem to make any big difference since the peak for this step directly aligned with the peak for step 4. Step 5 was added to the procedure to make sure that the value k_{kin} was at its best before using the trf-method to decrease computational time, but shows to be inefficient in some cases such as this. Finally, step 6 reached the experimental data parameters with the trf-method. This method is also the most time consuming method with more than 60 % of the simulation calls depending on what case.

This procedure was not the one with the lowest computational time and may have not

been the most efficient, but it was the most robust procedure. Since the aim of the thesis was to create a robust parameter estimation software, it was chosen to fill that requirement.

4.3 Parameter Estimation Robustness - Case 2

For the three cases in the test of the software's robustness, the estimated parameters for each guess is provided together with RSS for both the linear gradients. In a separate table, number of simulations calls for each guess is shown together with the mean for the case. To show how good the concentration curve fits with the estimated parameters, a plot for the 20 min linear gradient is shown together with the experimental data set. Since the estimated parameters are very close to the experimental parameters, only one concentration plot is shown for each case. The rest of the 20 min linear gradient plots can be found in Appendix A and the 8 min linear gradient plots can be found in Appendix B. In these plots, the concentration is measured in the outlet of the UV-sensor of the system. Also note that units of the y-axis in the plots does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

Since all cases showed very similar results a general discussion of the results is presented after the results from all the cases.

4.3.1 Case 2-R

The estimated parameters are shown in Table 4.2 and in Table 4.3 the number of simulation calls are shown. The concentration plot for initial guess L is found in 4.8.

Table 4.2 Parameter estimation results with RSS from case 2-R.

Estimated Parameters	k_{kin}	K_{eq}	ν	RSS (8 min)	RSS (20 min)
Experimental	$1.00 \cdot 10^{-2}$	$6.80 \cdot 10^{7}$	3.30	-	-
Initial Guess D	$1.87 \cdot 10^{-4}$	$6.80 \cdot 10^{7}$	3.30	$4.58 \cdot 10^{-11}$	$2.96 \cdot 10^{-11}$
Initial Guess C	$3.70 \cdot 10^{-4}$	$6.80 \cdot 10^{7}$	3.30	$2.11 \cdot 10^{-11}$	$1.10 \cdot 10^{-11}$
Initial Guess L	$2.97 \cdot 10^{-4}$	$6.80 \cdot 10^{7}$	3.30	$3.41 \cdot 10^{-11}$	$1.50 \cdot 10^{-11}$

Table 4.3 Number of simulation calls from case 2-R.

Initial Guess	**SimCalls**	**Mean SimCalls**
Initial Guess D	1150	
Initial Guess C	1114	1159
Initial Guess L	1214	

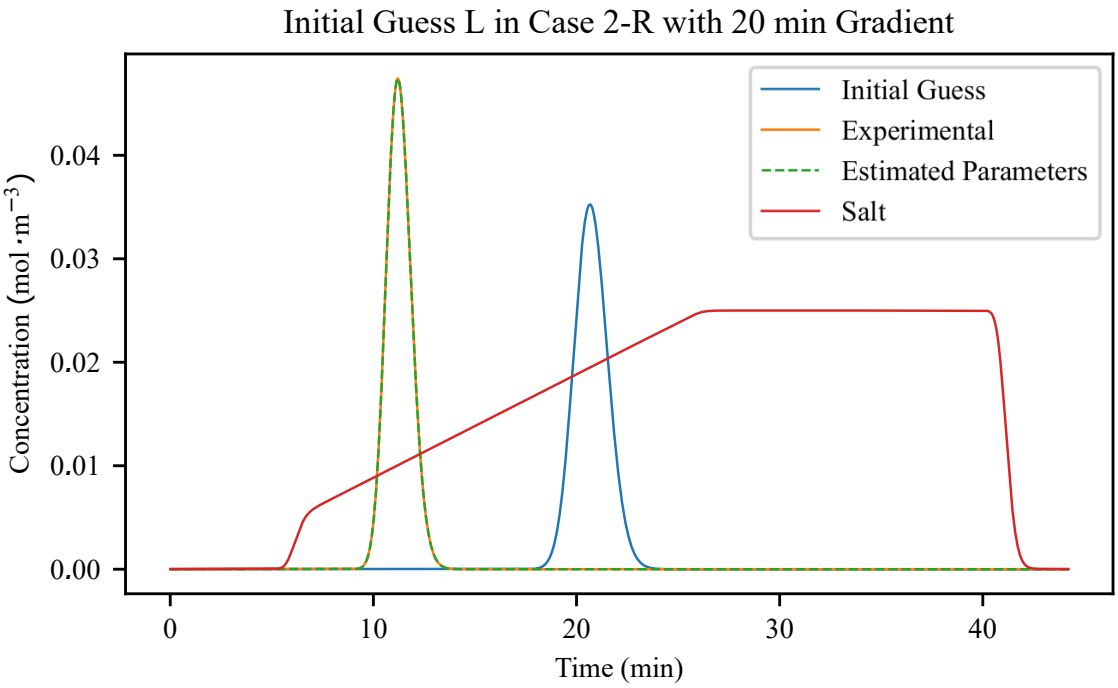

Figure 4.8 Results of case 2-R from initial guess L with a salt gradient of 20 min. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

4.3.2 Case 2-C

The estimated parameters are shown in Table 4.4 and in Table 4.5 the number of simulation calls are shown. The concentration plot for initial guess D is found in 4.9.

Table 4.4 Parameter estimation results with RSS from case 2-C.

Estimated Parameters	k_{kin}	K_{eq}	ν	RSS (8 min)	RSS (20 min)
Experimental	$1.00 \cdot 10^{-4}$	$5.00 \cdot 10^{10}$	4.10	-	-
Initial Guess R	$1.56 \cdot 10^{-7}$	$5.00 \cdot 10^{10}$	4.10	$1.10 \cdot 10^{-10}$	$1.35 \cdot 10^{-10}$
Initial Guess D	$6.88 \cdot 10^{-9}$	$5.02 \cdot 10^{10}$	4.10	$4.79 \cdot 10^{-8}$	$7.25 \cdot 10^{-8}$
Initial Guess L	$7.05 \cdot 10^{-9}$	$5.02 \cdot 10^{10}$	4.10	$4.50 \cdot 10^{-8}$	$6.96 \cdot 10^{-8}$

Table 4.5 Number of simulation calls from case 2-C.

Initial Guess	**SimCalls**	**Mean SimCalls**
Initial Guess R	1110	
Initial Guess D	866	954
Initial Guess L	886	

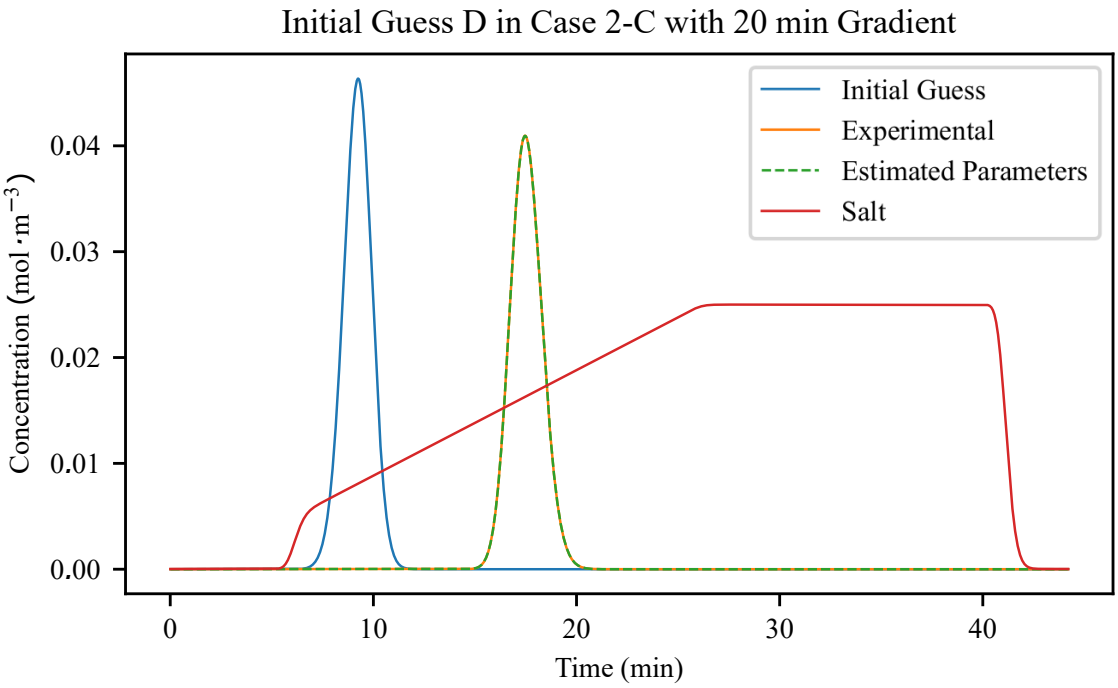

Figure 4.9 Results of case 2-C from initial guess D with a salt gradient of 20 min. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

4.3.3 Case 2-L

The estimated parameters are shown in Table 4.6 and in Table 4.7 the number of simulation calls are shown. The concentration plot for initial guess R is found in 4.10.

Table 4.6 Parameter estimation results with RSS from case 2-L.

Estimated Parameters	k_{kin}	K_{eq}	ν	RSS (8 min)	RSS (20 min)
Experimental	$1.00 \cdot 10^{-6}$	$4.00 \cdot 10^{12}$	4.70	-	-
Initial Guess R	$8.62 \cdot 10^{-9}$	$4.00 \cdot 10^{12}$	4.70	$1.48 \cdot 10^{-11}$	$6.56 \cdot 10^{-12}$
Initial Guess C	$2.12 \cdot 10^{-9}$	$4.00 \cdot 10^{12}$	4.70	$8.30 \cdot 10^{-11}$	$8.31 \cdot 10^{-11}$
Initial Guess D	$8.76 \cdot 10^{-10}$	$4.00 \cdot 10^{12}$	4.70	$4.95 \cdot 10^{-10}$	$4.81 \cdot 10^{-10}$

Table 4.7 Number of simulation calls from case 2-L.

Initial Guess	**SimCalls**	**Mean SimCalls**
Initial Guess R	1540	
Initial Guess C	1230	1294
Initial Guess D	1112	

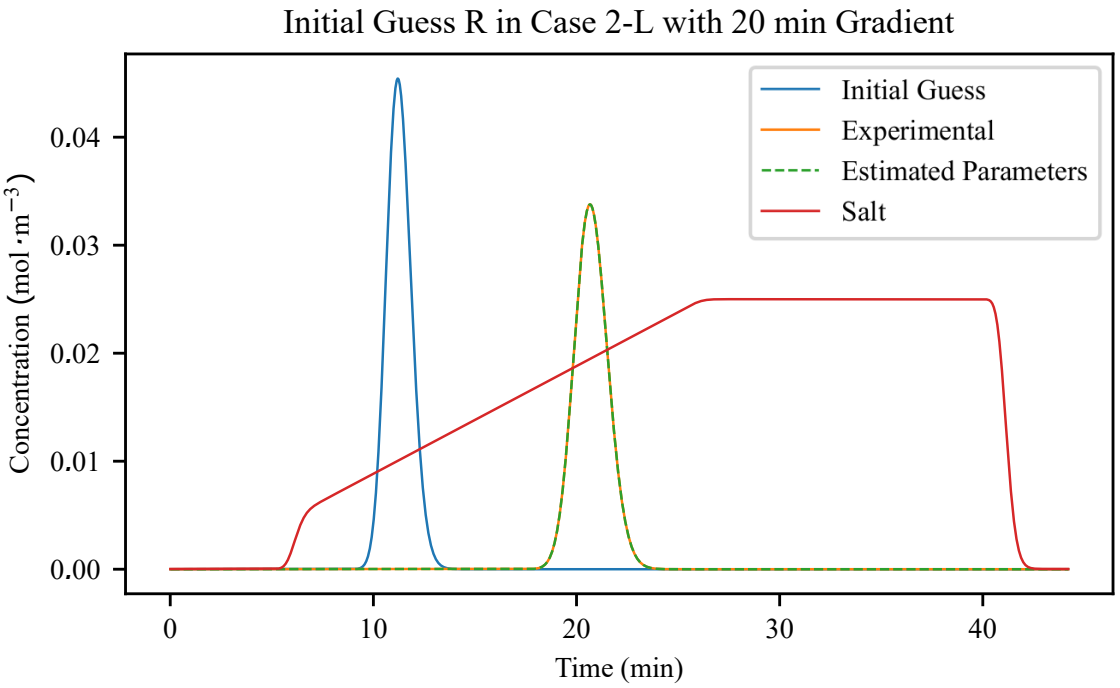

Figure 4.10 Results of case 2-L from initial guess R with a salt gradient of 20 min. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

4.3.4 Summation on the Parameter Estimation

From the results in the tables and figures, the estimated parameters were very similar to the experimental parameters except for the k_{kin} that was smaller for all initial guesses. Looking at the concentration plots, the curves fits each other very well on all initial guesses with no distinguishable visual difference. This shows that the size of k_{kin} does not have a big impact on the result of the model. It also shows that the software is able to find a k_{kin} where it gives a good parameter estimation result without reaching the experimental parameters. The amount of SimCalls for the initial guesses ranges from 866-1540 and the ones that took the most calls were the ones where the guess was far away from the correct parameters.

The RSS-values for each case were very small. This can also be seen visually in the concentration plots where the curve for the experimental parameters and the estimated parameters were close to identical. The reason that the RSS-values are not zero in the cases could be the small differences that the ode-solver gives in each iteration of solving the same system. These differences are very small, but noticeable in the RSS-value. In case 2-C, the RSS-values are slightly higher for initial guess D and L. This is due to the estimated K_{eq} parameter that differs slightly from the experimental parameter. This gives a higher RSS-value but there was still no visual difference in the concentration plots meaning that the difference does not impact the overall result.

The goal of the parameter estimation was to fit the experimental data as good as possible. This implies that the peaks from the experimental parameters and the estimated parameters should overlap completely. For this study, this showed to be achievable since the experimental data was created from the same simulation software that was used in the parameter estimation software as well. The software also showed good robustness since the same procedure could find the correct parameters for all the initial guesses in all cases. In all cases, the experimental parameters were found and the software was able to estimate a very good model. This means that the created software can approach the correct parameters from a wide range of initial guesses and that the software shows robustness in its calculations.

The cost of this robustness was the number of SimCalls. As mentioned in section 3.3.2, the trf-method makes twice as many SimCalls than the Levenberg-Marquardt method. This is due to the 3-point finite difference scheme for numerical estimation of the Jacobian that requires twice as many calculations. Although this trf-method gives robustness, it adds calculation time in the form of more SimCalls. But when using only the Levenberg-Marquardt method, the software did not reach correct parameters in all cases and had a hard time estimating the results of the steps taken in the minimization. Therefore the trf-method became necessary to achieve the robustness demand in the software. This did however take at least 850 SimCalls for each guess, and in most of the cases even more. Depending on the processor of the computer running the software, the computational time differed. In this study, guesses took between 2-4 hours to complete. This was however only the time for the parameter estimation software to be completed. If a user would want to do the whole procedure, meaning that the experiments on the ÄKTA Pure was to be conducted too, the total time of the software would increase. Adding the total experimental time, the total time of the parameter estimation method would be around 3.5-5.5 hours. This might be regarded as a long time, but since only two experiments needs to be conducted, the amount of protein that is used can be lowered compared to other experimental methods. This computational time could of course be reduced with a better processor or using a cluster of computers.

The whole procedure can be regarded as automated as soon as the user starts the procedure. The regular workflow is that experiments are conducted on the ÄKTA Pure and from the results, the user uses that data to estimate parameter in a separate script/program. With this automation, the efficiency of a parameter estimation can be increased leading to less data

analysis since the software does that by itself.

This "perfect parameter estimation" might not be necessary for the the estimated model parameters to be useful. In some cases, the user is only interested of e.g. the retention time and not the peak height. This is often the case when trying to separate two or more components where the user wants to decide at what time the flow path should change so the components can end up at different end points. So even if the software does not show a perfect fit, some information can still be useful. But in those cases, validation of the estimated model should be conducted to make sure that the model shows good results in a wide range. Otherwise the model might only be applicable in certain cases and therefore not suitable for use in a wide range of determination of process design.

It is important to remember that the results from the parameter estimation software must be validated against one or more separate linear gradient experiments. Otherwise the result can never be used as a valid model since it has not shown to be accurate for more than two cases of linear gradients. For this thesis a validation was unnecessary since the experimental parameters were known and the estimated resulted in the same values. But if the software were to be used in a lab, validation is necessary to validate the model for further use. There are also other problems that arise when conducting real experiments of the ÄKTA machines. From the experiments, the data of the peaks comes in the form of a UV-signal. This signal can be transformed into concentrations by Lambert's-law (if a calibration curve of the sensor has been made). If there is any errors in this calibration, the concentrations from the data and the ones achieved in the simulations will not be the same. From real experimental data, other differences also arises such as signal saturation, different dispersion etc. All these differences needs to be accounted for when using the software in real experiments.

5

Conclusions

An integration of a simulation software to the preexisting and widely used ÄKTA machine software *Orbit* was successfully implemented. The software was able to simulate a complete ÄKTA system with tubes, columns and other units. The system also implemented a simplified steric mass action model that described the adsorption of components in the columns. With the software, the user has the ability to follow components through the whole system and predict results before conducting experiments on the system. The computational time was reduced 60 % by providing a Jacobian sparsity matrix to the ode-solver.

A parameter estimation software was also implemented to the software. With only two simulated data sets, the software was able to estimate precise SMA model parameters for a single component. The software's main focus was robustness, meaning that it could reach a good solution even with a bad guess. If a user did not provide a guess, the software sets one for the user. In all the investigated cases, the software was able to find the correct model parameters. The cost of this robustness was a great amount of calls to the simulation function. The number of simulation calls varied around 866-1540 and resulted in computational times of 2-4 hours. With this implementation the experiment and model estimation was automated with the same syntax as an experienced *Orbit* user is familiar with.

6

Future Work

In this thesis, all the experimental data is based on simulations of known model parameters. To really evaluate the parameter estimation software and its robustness, it should be tested against real experimental data from an ÄKTA Pure system.

There is also room for improvement of the software. In this thesis, only one component have been under investigation. The next step would be to add the possibility of making parameter estimations for multiple components. When trying to implement multiple components, other problems arise. If the experiments are conducted on an ÄKTA Pure machine, the only experimental data for the components would be a single UV-signal of one particular wavelength or multiple UV-signals of particular wavelengths. This means that the risk of components showing up on different wavelengths of the UV-signals are high and it is therefore hard to know which peak is which component. This means that the software needs to get experimental data where the different component peaks can be separated from a single or multiple signals and then make the parameter estimation. It would also be of interest to make parameter estimations of the q_{max}. For this type of estimation, overload experiments are usually conducted. In an overload experiment, the column is loaded with a high amount of high concentration component solution and then the column is eluted.

As with all software, the goal is to lower the computational time. In this thesis, all parameter estimations took over 2 hours to complete. This was only for a single component, meaning that if more components were to be introduced, the computational time would increase additionally. The focus of this thesis have been to create a robust software that can handle a wide variety of cases, but if development of this software would be continued, focus should also be on computational time.

Bibliography

[1] G. Guiochon and O. Trapp, *Basic principles of chromatography*, in, *Ullmann's Encyclopedia of Industiral Chemistry*. Jan. 15, 2012.

[2] M. Kastner, *Protein Liquid Chromatography*, ser. Journal of Chromatography Library. Elsevier Science, 2000.

[3] GE. (Aug. 22, 2018). Äkta pure, [Online]. Available: `https://www.gelifesciences.com/en/bg/shop/chromatography/chromatography-systems/akta-pure-p-05844`.

[4] N. Andersson, "The orbit controller," Department of Chemical Engineering at Lund University, Documentation, May 7, 2018.

[5] D. Karlsson, N. Jakobsson, A. Axelsson, and B. Nilsson, "Model-based optimization of a preparative ion-exchange step for antibody purification," *Journal of Chromatography A*, **1055**:no. 1, pp. 29 –39.

[6] N. Jakobsson, M. Degerman, E. Stenborg, and B. Nilsson, "Model based robustness analysis of an ion-exchange chromatography step," *Journal of Chromatography A*, **1138**:no. 1, pp. 109 –119.

[7] S. Al-Kaisy, *Model calibration and optimization of a protein purification process*, Master Thesis, Department of Chemical Engineering at Lund University, 2015.

[8] O. Persson, *Increasing yield in chromatography using batch-to-batch recirculation*, Master Thesis, Department of Chemical Engineering at Lund University, 2016.

[9] A. Sellberg, *Open-loop optimal control of chromatographic separation processes*, eng, PhD thesis, Lund University, May 2018.

[10] K. Hangos and I. Cameron, *Process Modelling and Model Analysis*. 2001.

[11] M.Michael, A. Epping, and A. Jupke, "Modeling and determination of model parameters," in H. Schmidt-Traub (Ed.), *Preparative Chromatography: of Fine Chemicals and Pharmaceutical Agents*, Apr. 12, 2005.

[12] M. E. Davis, *Numerical Methods and Modeling for Chemical Engineers*. 1984.

[13] R. J. LeVeque, *Finite Volume Methods for Hyperbolic Problems*. 2002.

[14] The SciPy Community. (Nov. 12, 2018). Scipy.integrate.solve_ivp, [Online]. Available: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html#scipy.integrate.solve_ivp`.

[15] N. Andersson, "Jvm_jpattern," Department of Chemical Engineering at Lund University, MATLAB Script, May 7, 2018.

[16] The SciPy Community. (Nov. 13, 2018). Scipy.optimize.minimize, [Online]. Available: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html#scipy.optimize.minimize`.

[17] The SciPy Community. (Nov. 13, 2018). Scipy.optimize.curve_fit, [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html#scipy.optimize.curve_fit.

[18] GE Healtcare. (Nov. 12, 2018). Hitrap sp hp cation exchange chromatography column, [Online]. Available: https://www.gelifesciences.com/en/se/shop/chromatography/prepacked-columns/ion-exchange/hitrap-sp-hp-cation-exchange-chromatography-column-p-00794.

A

Parameter Estimations Plots - 20 min Gradient

Here are the concentration plots from the 20 min gradients in the parameter estimation described in section 3.4.3.

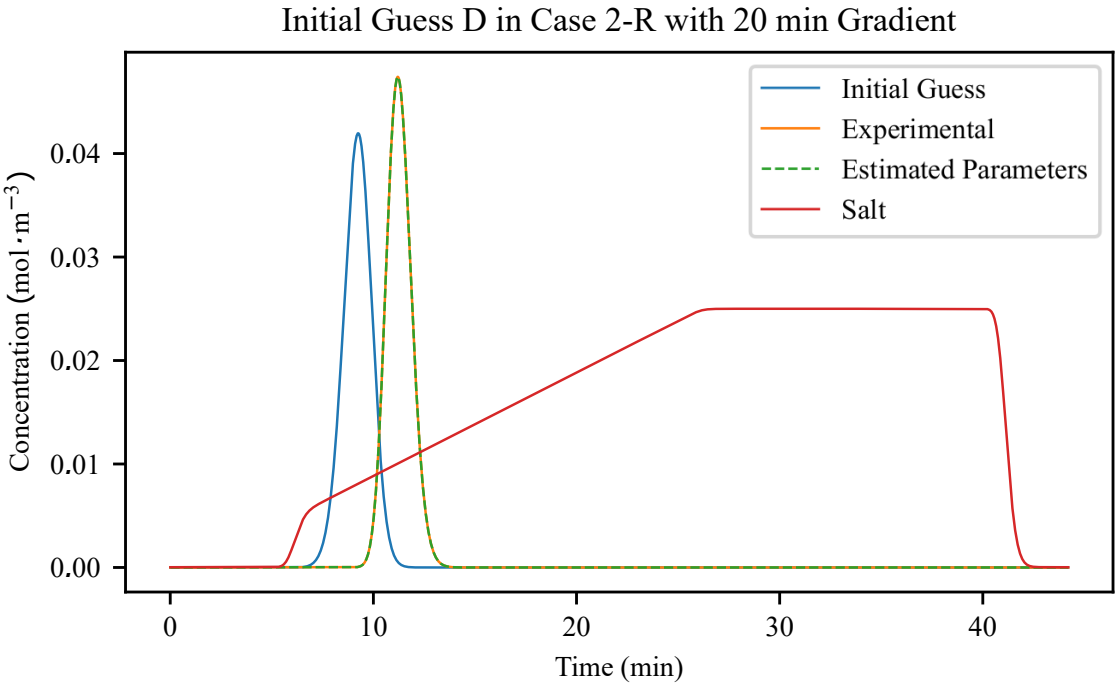

Figure A.1 Results of case 2-R from initial guess D with a salt gradient of 20 min. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

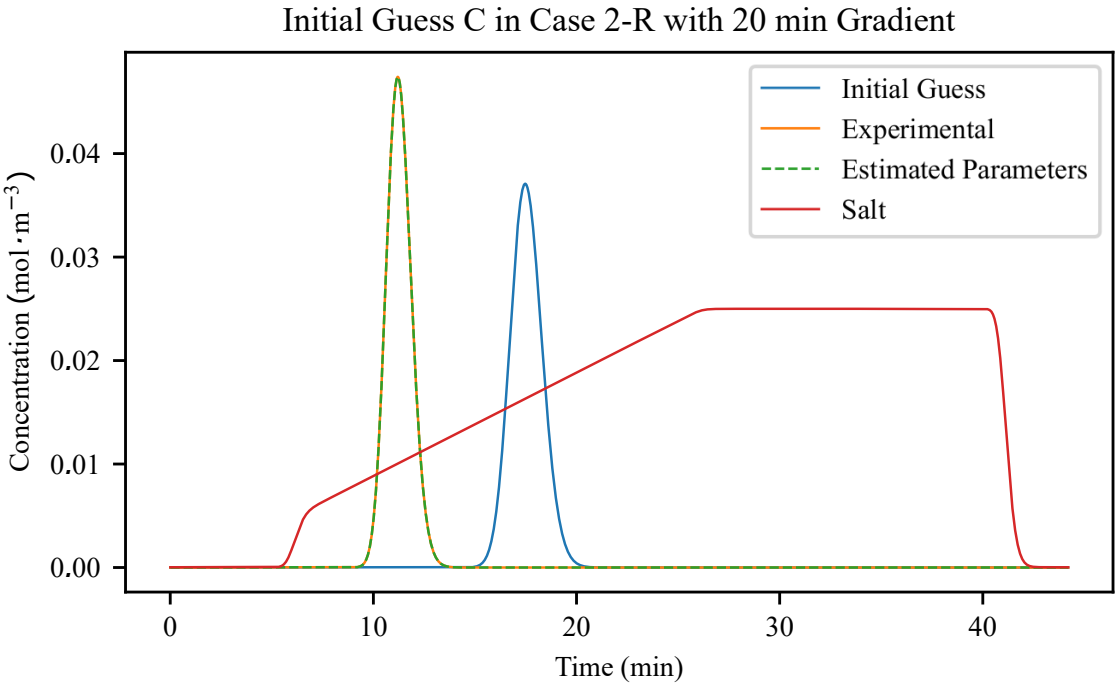

Figure A.2 Results of case 2-R from initial guess C with a salt gradient of 20 min. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

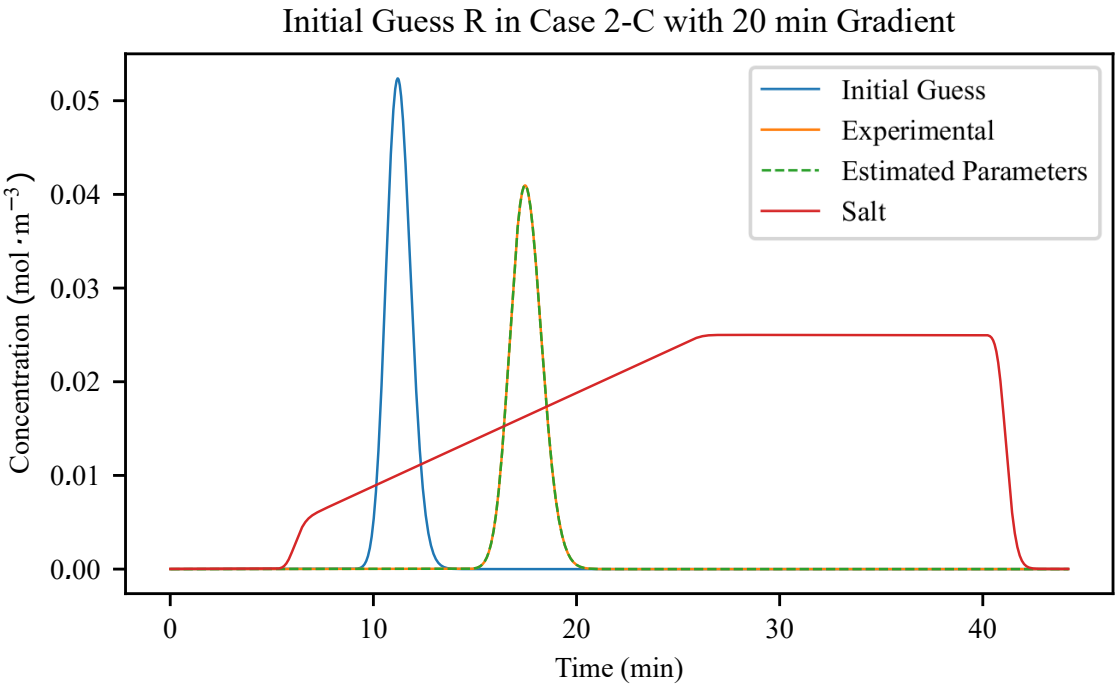

Figure A.3 Results of case 2-C from initial guess R with a salt gradient of 20 min. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

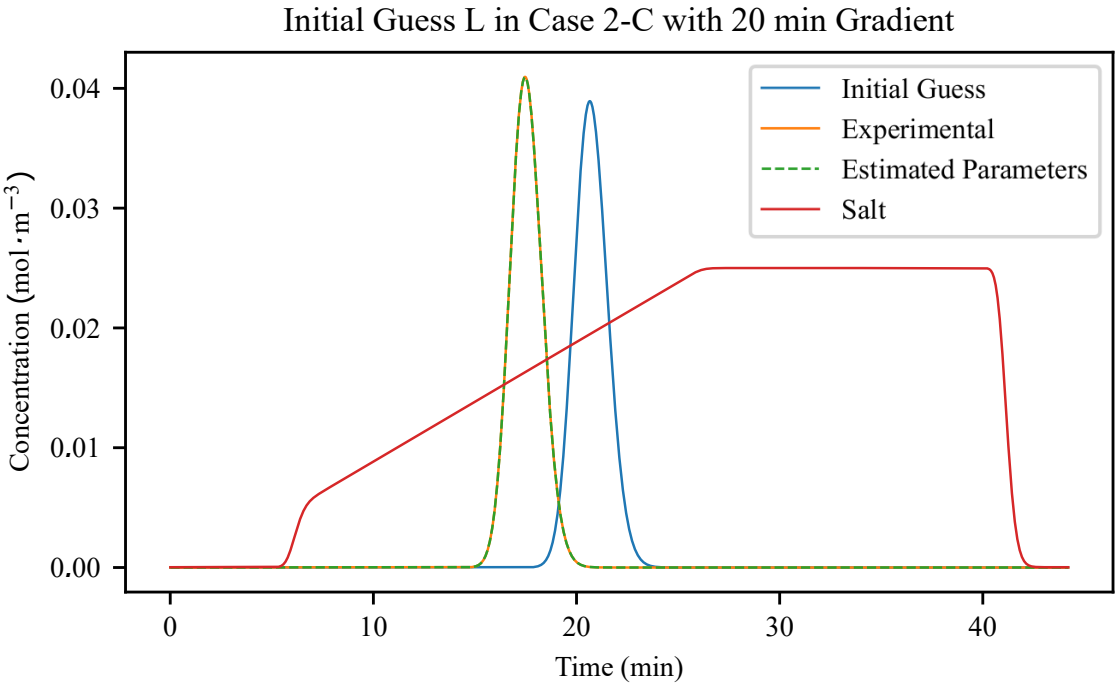

Figure A.4 Results of case 2-C from initial guess L with a salt gradient of 20 min. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

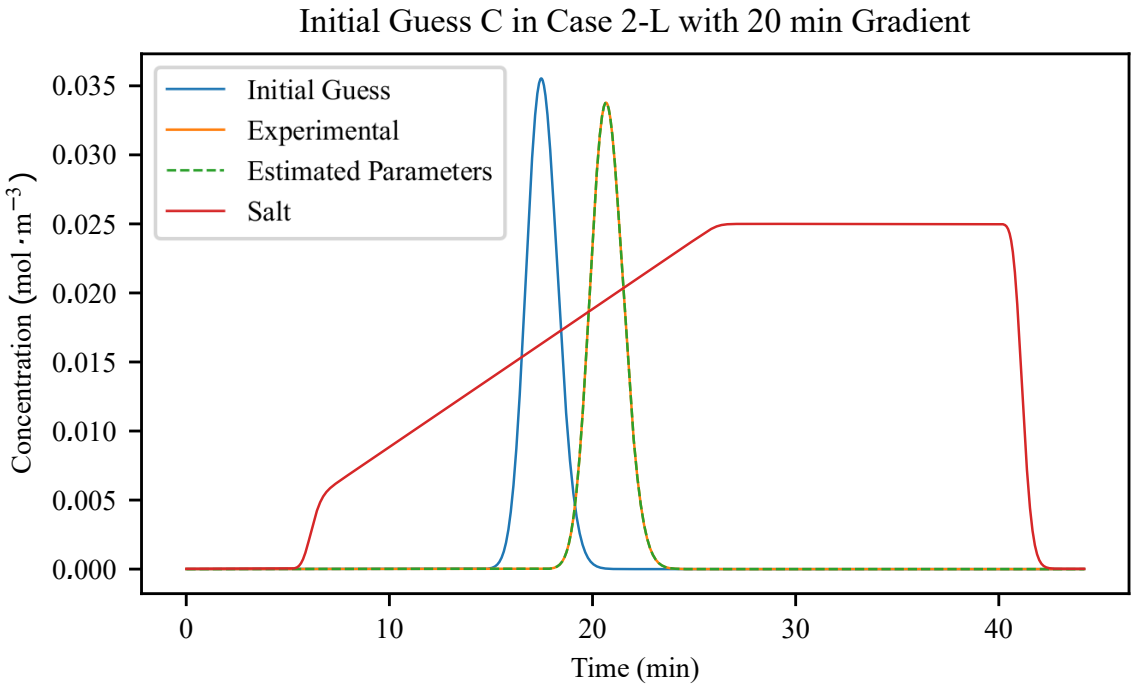

Figure A.5 Results of case 2-L from initial guess C with a salt gradient of 20 min. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

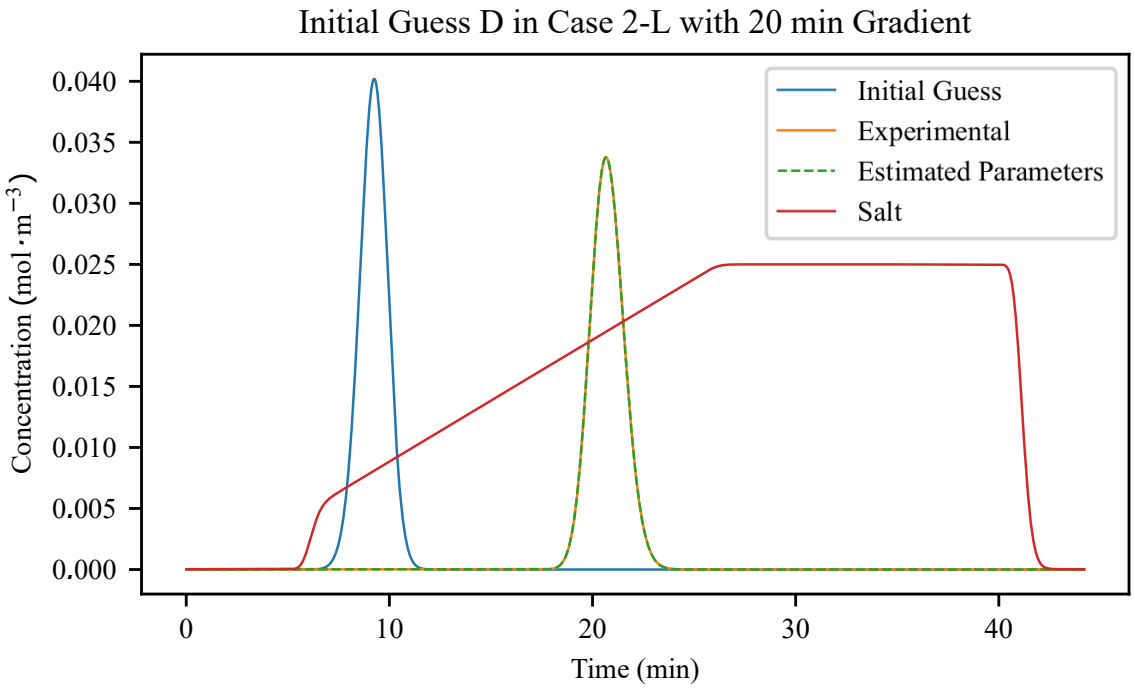

Figure A.6 Results of case 2-L from initial guess D with a salt gradient of 20 min. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

B

Parameter Estimations Plots - 8 min Gradient

Here are the concentration plots from the 8 min gradients in the parameter estimation described in section 3.4.3.

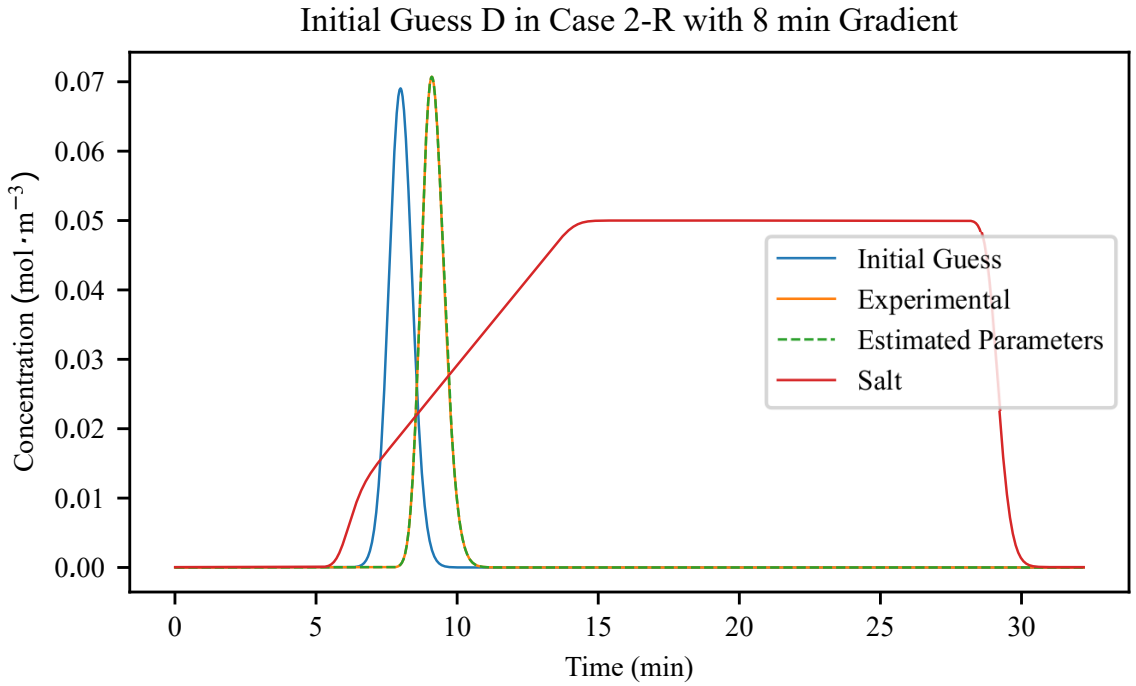

Figure B.1 Results of case 2-R from initial guess D with a salt gradient of 8 min. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

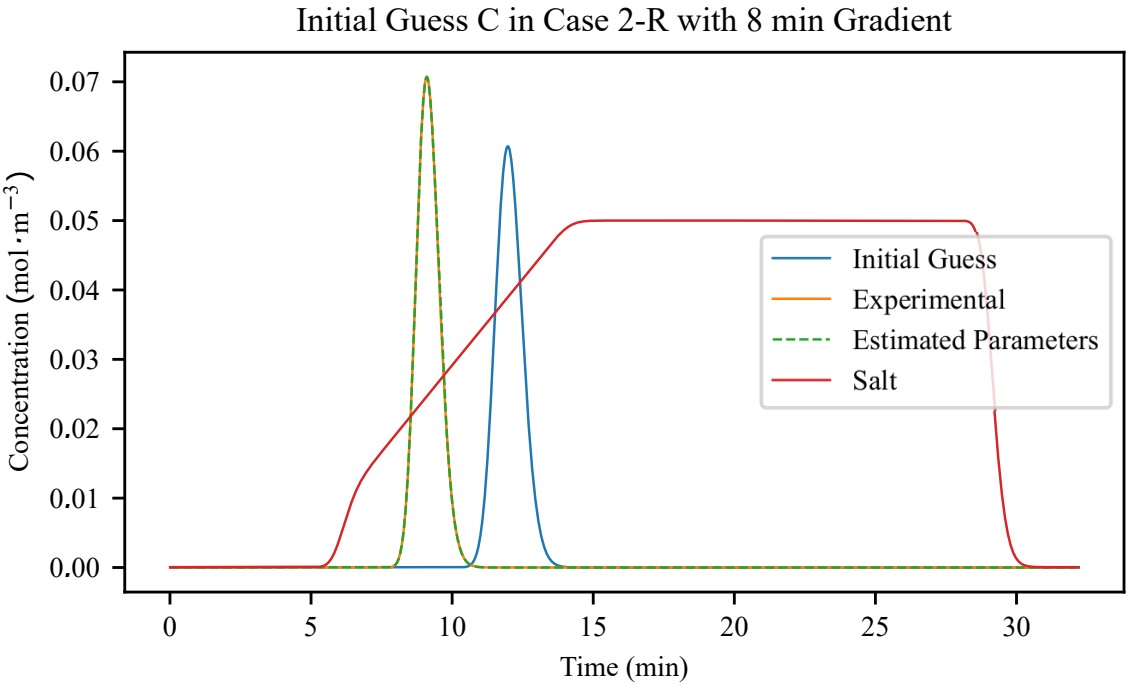

Figure B.2 Results of case 2-R from initial guess C with a salt gradient of 8 min. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

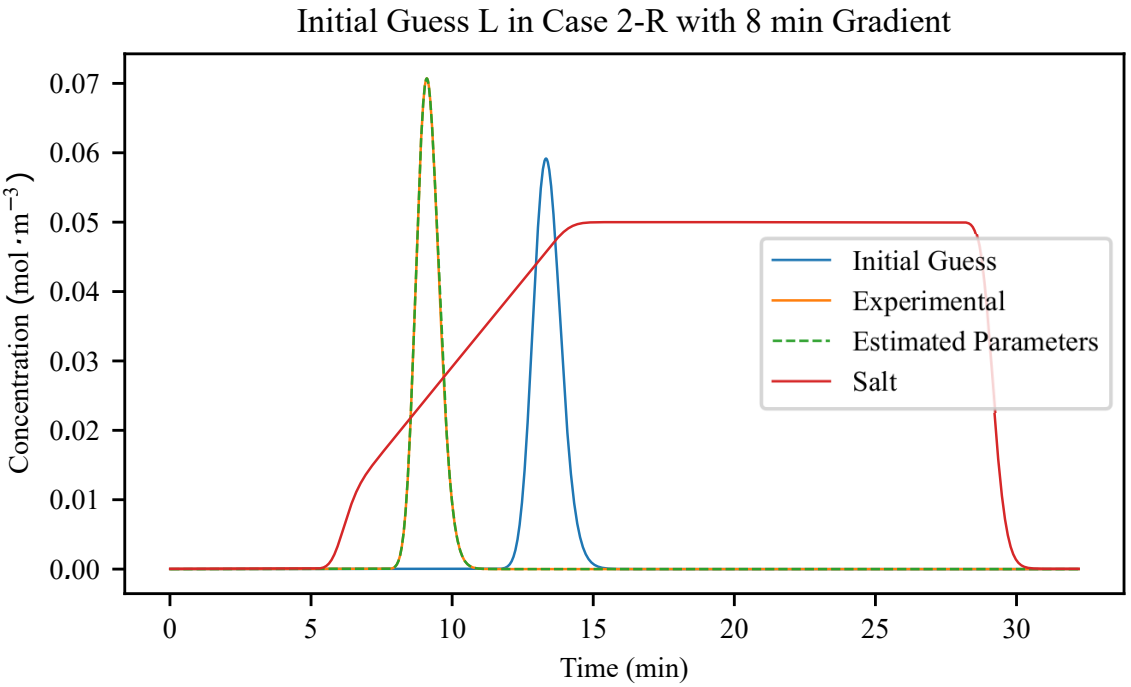

Figure B.3 Results of case 2-R from initial guess L with a salt gradient of 8 min. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

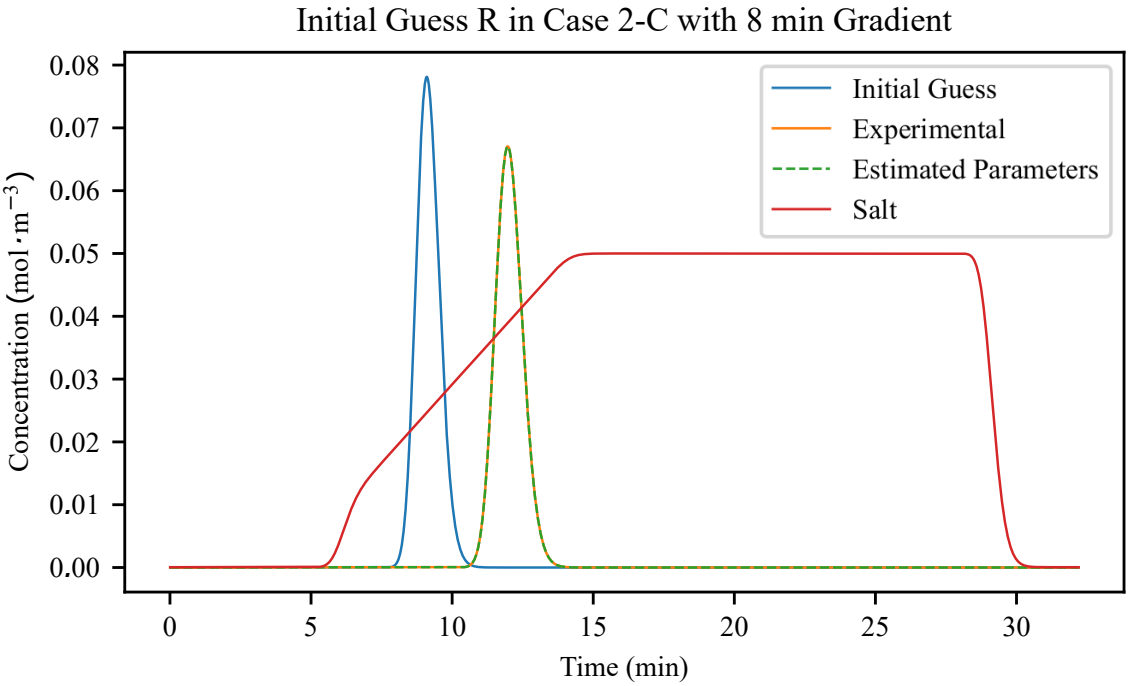

Figure B.4 Results of case 2-C from initial guess R with a salt gradient of 8 min. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

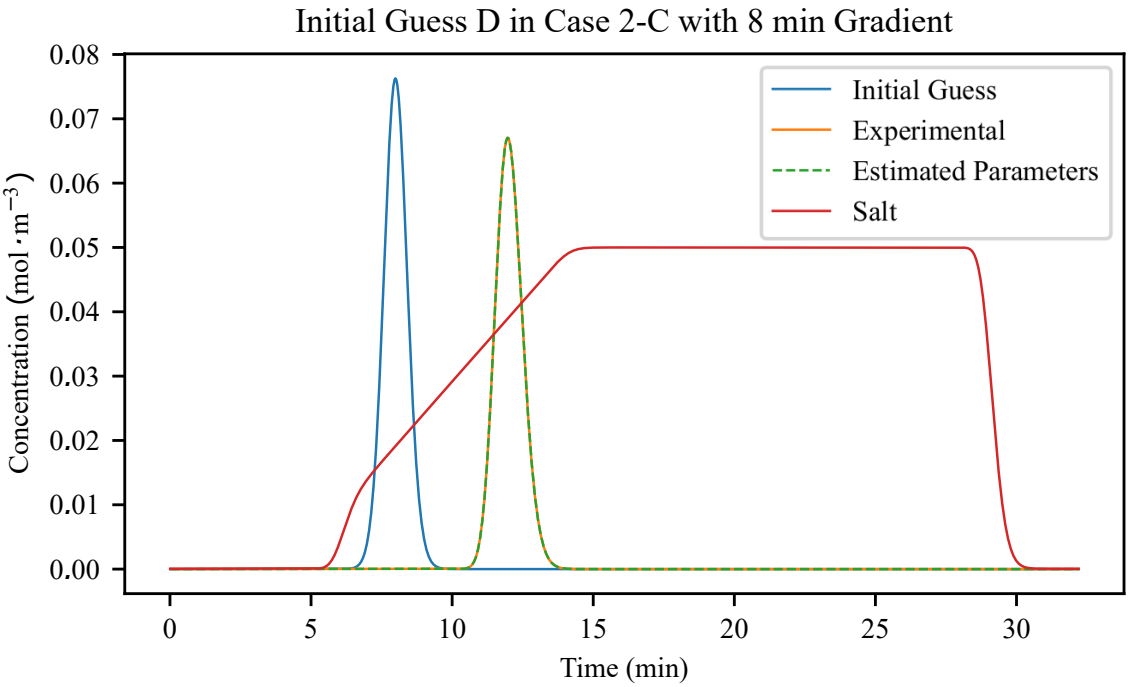

Figure B.5 Results of case 2-C from initial guess D with a salt gradient of 8 min. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

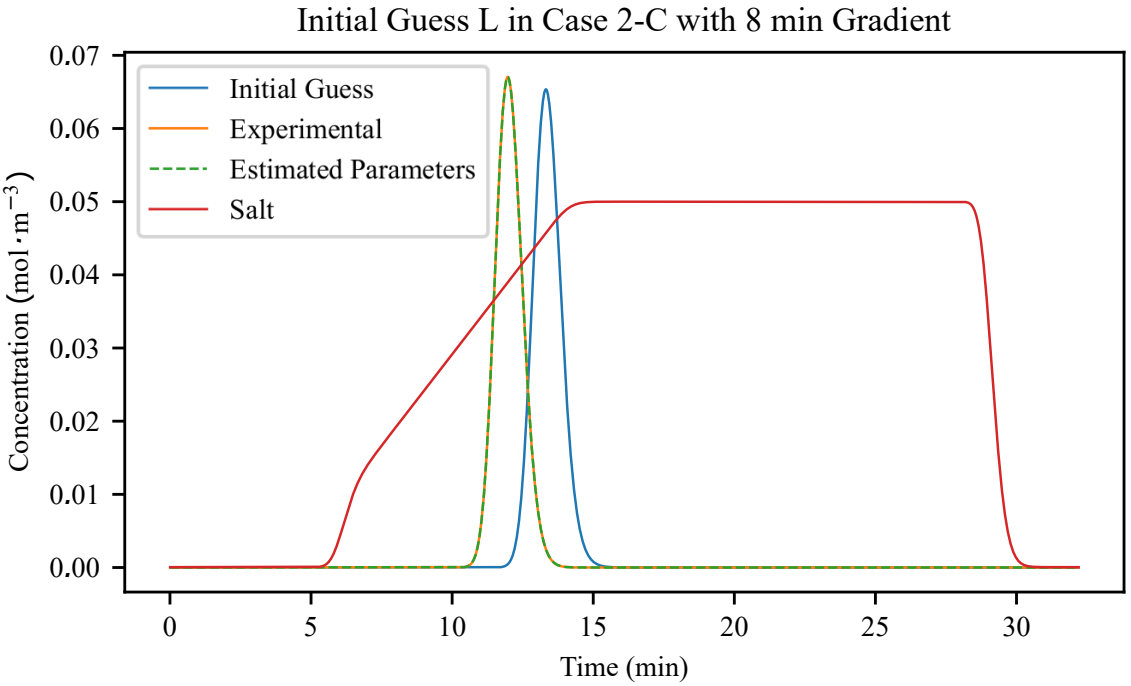

Figure B.6 Results of case 2-C from initial guess L with a salt gradient of 8 min. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

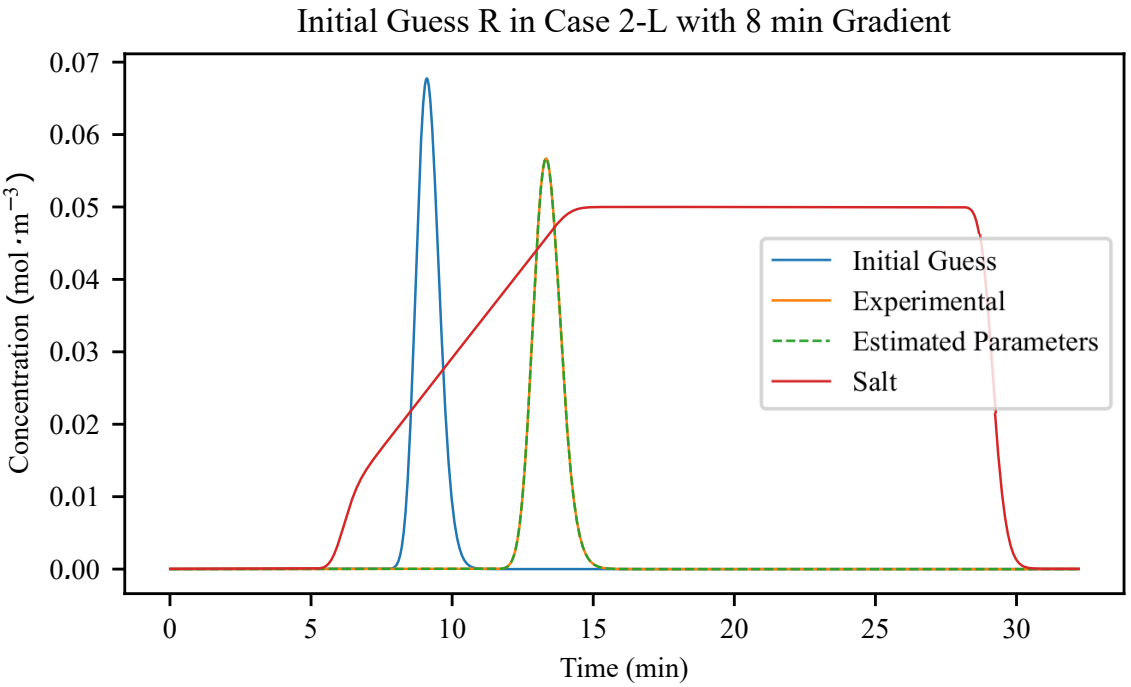

Figure B.7 Results of case 2-L from initial guess R with a salt gradient of 8 min. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

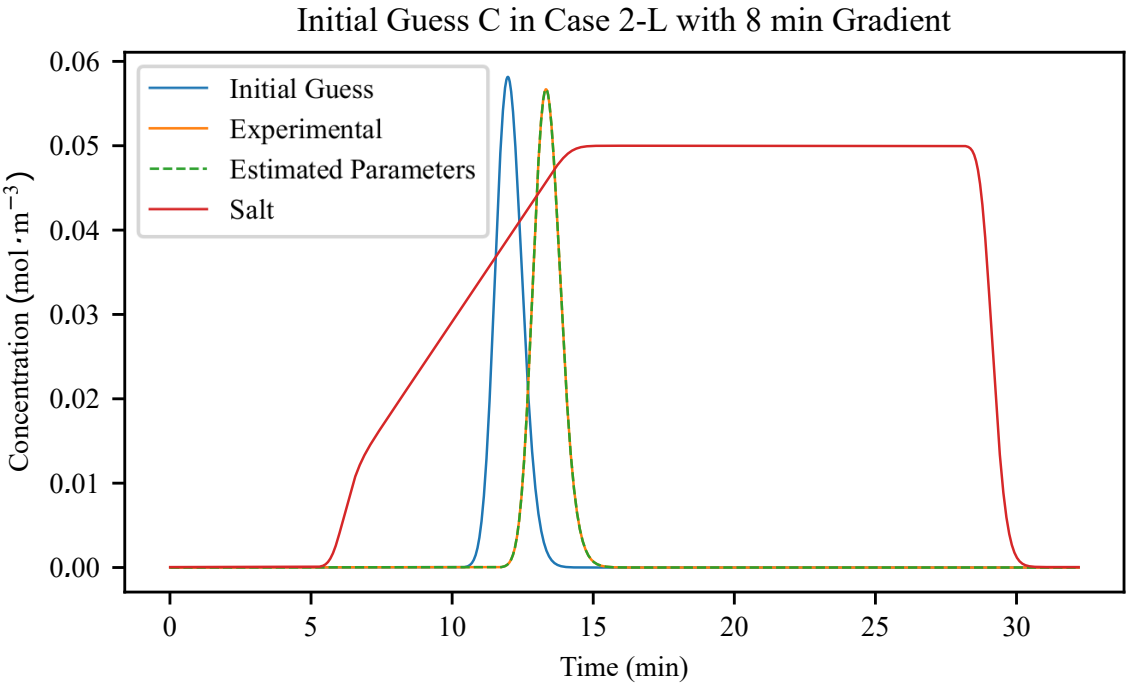

Figure B.8 Results of case 2-L from initial guess C with a salt gradient of 8 min. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

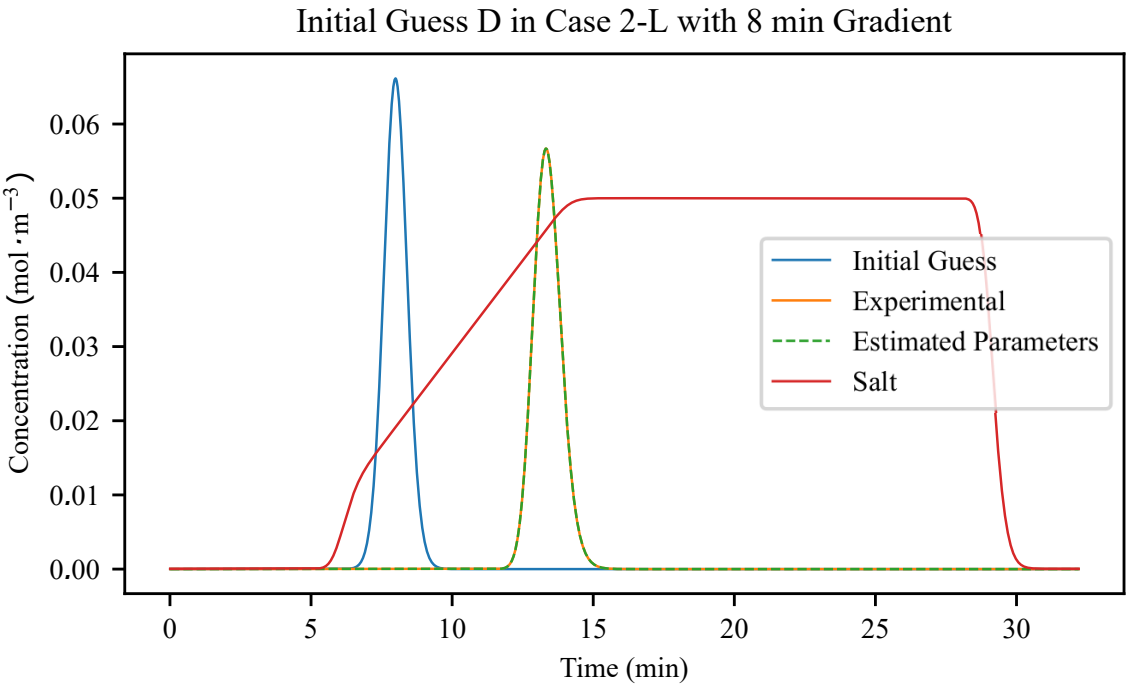

Figure B.9 Results of case 2-L from initial guess D with a salt gradient of 8 min. Note that units on the y-axis does not correspond to the salt concentration, i.e. the salt concentration is shown as an illustration of the elution sequence.

C

Syntax Explanation for Orbit

In this appendix, a short summary of the syntax of the simulation and parameter estimation syntax is presented. For the simulation of the instructions, a class called `orbitSim` is used. When estimating parameters the class `orbitCalibrate` is used.

In the method function of every orbit-script, the user needs to specify the buffer and number of components as in Listing C.1. First the different components are defined. If some parameters are unknown, these values can be set to 1. Then the buffers are defined by its content and then the system is filled with buffers as the initial set up. Finally the buffers are connected to a inlet of the inlet valve.

```
def method(S,gradientTime):
    ''' ********* Define buffers and components ********** '''
    # --- Define the components in the system
    c1 = Component(name = 'Salt',
                   S=S,
                   Mw = 18.01,
                   K_uv=1) # to match the conductivity
    c2 = Component(name = 'Ribo',
                   S=S,
                   Mw = 13.7e3,
                   K_uv = 1080e5,
                   k_kin = 1e-2,
                   beta = 3.3,
                   H_0 = 6.8e7,
                   q_max = 51)
    c2 = Component(name = 'Lys',
                   S=S,
                   Mw = 14.3e3,
                   K_uv = 10000,
                   k_kin = 1e-6,
                   beta = 4.7,
                   H_0 = 4e12,
                   q_max = 51)

    # --- Define the buffers in the system
    b1 = Buffer(conc = {'Salt':0.0005e3, 'Ribo':0})
    b2 = Buffer(conc = {'Salt':0.5e3}) # mol/m3
    b3 = Buffer(conc = {'Salt':0.0005e3, 'Ribo':1/13.7e3*1e3,
                        'Lys':1/14.3e3*1e3}) #mol/m3

    # --- Fill system with buffer
    b1.fillSystem(S)
    b2.fill(S.pump.p['OutB'].tube)
    b2.fill(S.inlValveB.p['Out'].tube)
    b2.fill(S.inlValveB)

    b3.fill(S.supLoop)
```

```
38      b3.fill(S.supLoop.p['Out'].tube)
39
40      # ---- Connect buffers to inlets
41      b1.connect(S.inlValveA.p['5'])
42      b2.connect(S.inlValveB.p['1'])
```

Listing C.1 Defining buffers and components in the method fuction for the process

In the run options in the method to Orbit, the user can specify if a simulation or a calibration should be preformed. An example of this can be seen in Listing C.2. In this example, the method is set for a calibration and not only a simulation. As soon as the dictionary `simulation` is not empty, an object of `orbitSim` is created in the `process`-object created in the method function. If `'simulate'` is set to True, the instructions sent to orbit will be simulated directly.

```
P = Process(name='Gradient_Opti',
            phases=phases,
            S=S)

options = {'mode':'test',
           'timeFactor':30.,
           'sampleSignals':['uv1'],
           'sampleTime':1.0,
           'loops':[],
           'logData':False,
           'logRun':False,
           'simulation':{'calibrate':True,
                         'simulate': False}}
```

Listing C.2 Options for the process

When a `process`-object is created, a `orbitSim`-object is saved within. An example how a calibration is preformed is presented in Listing C.3 and an explanation of the code is presented below:

- A `orbitCalibrate`-object is created. This object contains all minimization functions.
- A `system`-object is created based on the system file that defines all the units on the ÄKTA system including the tubes etc. This `system`-object (S1 and S2) is passed on to the `method` function that creates a `process`-object (P1 and P2).
- The `system`-object, the process instructions, the `orbitSim`-object and experimental concentration and time data is added to the `orbitCalibrate`-object. This is needed for the calibration to separate which simulation instructions that are coupled to which experimental data.
- A guess (x0) is created as a list. Then the parameter estimation is called by the function `calibrate` in the `orbitCalibration`-object. The function returns the estimated parameters

```
1       # Creates the orbitCalibrate object
2       orbitCali = orbitCalibrate()
3
4       # Simulates the second session and adds the data to the calibrator
5       S2 = SystemPure(mode='test')
6       P2,resTime2,resCond2 = method(S2,20)
```

```
7     orbitCali.addRunData(S2,P2.I,P2.orbsim,data2,tsol2)
8
9     # Simulates the first session and adds the data to the calibrator
10    S1 = SystemPure(mode='test')
11    P1,resTime1,resCond1 = method(S1,8)
12    orbitCali.addRunData(S1,P1.I,P1.orbsim,data1,tsol1)
13
14    # kkin, beta, H_0
15    x0 = [1e-4, 4.1, 5e10]
16
17    # Parameter estimation
18    calibrated_x,n = orbitCali.calibrate(x0)
```

Listing C.3 An example how a calibration of a component is set up